MikroTik

RouterOS入门到精通 v6.7.1e

#include "MikroTik CRS3系列交换机配置手册"

#include "MikroTik RouterOSv7教程..."

- 学习RouterOS必备中文教程
- 功能与原理讲解
- RB、CCR和CRS相关硬件介绍
- 多线接入与策略路由
- 防火墙与QoS流控
- PPPoE与Hotspot认证
- VPN隧道应用

编：余松

www.irouteros.com

前 言

我第一次使用 RouterOS 产品是 2003 年底，版本是 2.7.14，正式开始编写 RouterOS 教程是从 2006 年，当然要感谢曾经的成都网大科技，提供了这样的平台！！随后，第一次参加 MikroTik 的 MUM 是 2006 年 9 月在新加坡，当时培训讲师还是 MikroTik 的 CTO Arnis。2010 年在广州 MUM 拿到了 MikroTik TRAINER 培训师认证，编号 TR0092，在 2013 年过期，由于工作变动，未在续考，但仍然把 RouterOS 当作一种爱好！

RouterOS 不管从功能，还是性能方面已经超过了许多中端路由器，随着 RouterOS 在国内被越来越多的人接受，从最开始的网吧多线路与流控和小区宽带，到后来的 VPN 应用和企业网络管理，还是当前 RouterOS 的 WLAN 无线应用，都在不断冲击整个网络行业！在 2005 年后出现了几款类似的软件路由系统，虽然从个别方面比起 RouterOS 优越，但整体上仍然难以超越。

从 2003 年开始系统接触和使用 RouterOS，多年的学习和工作中积累了大量的实际经验，在这期间也得到广大 RouterOS 爱好者的支持和帮助，也是大家相互学习的过程。本套课程通过各方面的数据整理后编辑而成，从最基础的安装、登陆配置、多线路策略和流量控制、到诸如 WLAN 配置、VPN 实现、脚本编写等高级应用进行深入的讲解，适合于所有使用或学习 RouterOS 的朋友。

本人还编写了另外三套教程：《RouterOS 脚本编写和 WLAN 无线教程》、《CRS3 系列交换机配置手册》和 MikroTik 配套的监控软件《The Dude 中文实用教程》，这些教程和手册可以到我的网站下载。

从 RouterOS 6.0 版本开始，MikroTik 对 RouterOS 做大的改动，包括 Queue 进行了大的改动，新增了 FastPath 功能和 Fasttrack 功能，用于提升转发性能。新增的基于 Tilera 产品 CCR 系列，还有 ARM 平台的引入进一步提升了 MikroTik 硬件平台的性能。又新增了二层交换的 CRS 系列产品，当前的 CRS3 系列产品也更加完善。

RouterOS v7 beta 版本也在 2020 年发布，Linux 内核升级到了 5.6，新增了许多功能，当前改动最大的是路由，从路由引擎上做了相当多的改进，策略路由表创建不再兼容 v6 版本的配置。还新增了两款商业应用软件 ZeroTier 和 Docker。

学习 RouterOS 还是希望大家具备一定的路由交换的基础，这样看更容易理解，网络的路由交换是任何网络工程师入门的基础，因此我在教程里开始加入了一些基础知识，希望缺乏基础网络知识的朋友能了解下，但这些基础知识还不够，需要大家自己去学习相关的路由交换知识，比较完善的教学资料如 CCNA 或 HCNA 等。

版本: **v6.7.1e**
适用: **RouterOS 6.x**
作者: 余松
作品登记号 川作登字-2020-A-00161540
Web: **www.irouteros.com**
E-mail: **athlon_sds@163.com**
如有更新，恕不通知！
该手册由本人多年整理编写而成，请勿非法篡改或其他商业用途！

目 录

前 言 **1**
目 录 **2**
应用说明 **18**
基础知识 **23**
OSI 七层参考模型 23
二层 数据链路层 24
三层 网络层 25
四层 传输层 27
RouterOS 基本 FAQ 28
第一章 RouterOS 基本操作 **30**
1.1 RouterOS 安装介绍 30
NetInstall 安装 30
DVD/CD 光盘安装 49
1.2 RouterOS 登录方式 53
方式 1 Console 连接 53
方式 2 Winbox 55
方式 3 webfig 57
方式 4 显示器+键盘 58
方法 5 MAC 层访问 (Telnet 与 Winbox) 59
MAC WinBox Server 59
MAC 登陆列表 60
MAC telnet 访问客户端 60
1.3 Winbox 操作 61
Winbox 界面概述 62
菜单栏 67
快速查询 68
过滤筛选 69
自定义显示列表 71
隐藏选项 71
详细模式 71
拖&放 72
流量监测 73
项目复制与静态绑定 74
Winbox IPv6 连接 76
Winbox 目录转移设置 77
1.4 Webfig 操作 77
连接到 RouterOS 78
操作界面介绍 80
列表项目配置 81
Files 文件操作 84

1.5 CLI（command Line interface）命令行操作 85
路径和帮助命令 85
基本操作命令 88
Tab 快速输入 92
快速设置 Setup 92
1.6 安全模式 94
1.7 RouterOS 简单网络配置事例 95
第一步：网络接口配置 96
第二步：添加 IP 地址 96
第三步：添加默认网关 97
第四步，NAT 地址转换 98
第五步，DNS 配置 99
端口映射 100
主机带宽控制 101
ADSL/PPPoE 拨号配置 102
1.8 基本网络故障排查命令与方法 103
第二章 System 系统管理 108
2.1 RouterOS 账号管理 108
Group（账号分组权限） 108
User（RouterOS 管理账号） 111
Active 查看在线活跃账号 111
Remote AAA（远程 AAA） 112
创建账号 113
2.2 RouterOS 备份与复位管理 114
系统备份与恢复 114
导出指令（Export） 116
导入指令（Improt） 118
系统复位 119
2.3 系统重启与关机 119
2.4 RouterOS 主机名 120
2.5 Resource（资源管理） 121
CPU 122
IRQ 配置管理 123
USB 端口信息 125
PCI 信息 125
RouterOS x86 平台多 CPU 设置 126
2.6 Watchdog 监测 127
2.7 RouterOS 功能包（Packages） 128
查看功能包 130
功能包操作事例 130
2.8 升级和降级 RouterOS 131
RouterOS 升级包区别 131
手动升级 133
在线升级 136
版本降级 137
2.9 SNTP client 138
2.10 telnet 和 ssh 客户端 139
telnet 客户端 139

SSH 客户端 139
2.11 IP Service（应用服务端口） 140
service 服务配置 141
常用服务清单 142
2.12 Note 笔记 144
2.13 Log 日志管理 145
Log（日志） 145
Logging 日志管理 146
Log 管理执行方式 148
使用 The Dude 管理器记录系统日志 149
2.14 IP Accounting 访问日志记录 152
IP 访问记录 153
IP 访问快照 153
Web 获取 IP 访问信息 154
2.15 Supout.rif 技术支持档 155
2.16 SNMP 157
启用 SNMP 157
Community 团体名 158
SNMP 配置实例 159
SNMP write 161
2.17 File 目录 162
2.18 Neighbor Discovery（邻居探测） 162
Discovery 配置 163
第三章 MikroTik RB/CCR/CRS 介绍 166
3.1 RouterBOARD 的发展 166
3.2 RouterBOARD 产品命名与标识 169
RouterBOARD 基本命名规则 169
集成无线网卡命名 170
无线网卡接口类型命名 170
外壳类型命名 170
Cloud Core Router 命名 171
Cloud Router Switch 命名 171
3.3 RouterBOARD Throughput（吞吐量） 171
3.4 RouterBOARD 的型号与分析 178
3.5 RouterBOARD reset 复位键 179
早期的 RB 设备 179
当前 RB 设备 181
RB Reset 复位键的功能说明 182
3.6 Reset 和 Mode 按键 182
3.7 升级 RB/CCR/CRS 固件 184
3.8 RouterBOARD Settings 186
3.9 RouterBOARD Switch 介绍 188
RB 采用的交换芯片 188
3.10 RouterBOARD Switch 配置 190
基本原理 190
Switch All Ports 功能 191
Host Table(MAC 列表) 191
Port（端口设置） 192

Vlan Table(VLAN 列表) 193
Rule Table（策略列表） 194
Port isolation（端口隔离） 195
CPU Flow Control（CPU 流量控制） 196
Switch 的 VLAN 配置（Trunk 与 access） 197
Switch 的 VLAN 配置（Trunk 与 Hybrid） 199
3.11 RouterBOARD 端口镜像 201
端口镜像配置 201
Rule 指定镜像内容 202
3.12 CRS 交换机介绍 205
术语和解释 205
端口交换（Port Switching） 206
CRS 交换配置 209
CRS Bonding 硬件加速 212
CRS Port isolation（端口隔离） 213
3.13 RouterBOARD LCD 触摸屏 214
LCD 触摸屏校准 215
LCD 截屏 215
LCD PIN 密码 215
LCD 流量图 216
3.14 Cloud 217
DDNS（动态域名解析） 218
Update time（时间更新） 219
Backup（云备份） 221
3.15 Flashfig 222
Flashfig 配置实例 222
第四章 Interface（接口配置） 229
4.1 Interface 基本操作 229
4.2 流量监控 230
4.3 以太网接口（Ethernet） 231
接口状态 233
线路故障探测 235
4.4 Lists 236
List（列表） 236
Members（成员） 237
创建静态列表事例 237
4.4 3G 接口扩展设置 238
联通 3G 拨号测试 239
电信 CDMA 上网卡配置 241
4.5 LTE 客户端配置 244
Scanner（扫描器） 245
参考配置 245
Huawei E3372h-153 电信 4G 测试 246
Passthrough（透传） 248
4.6 GPS 249
状态监控 249
USB GPS 接收器 250
ltAP mini 的 GPS 配置 252

GPS 位置上传 253
4.7 PoE-Out（PoE 供电） 254
PoE 端口设置 254
如何工作 255
Safety（安全） 256
PoE Out 线序 256
升级 PoE Out 固件 257
PoE Out 事例 257
PoE 固件 v 1.x 和 v 2.0 区别 258
第五章 IP 与 ARP 配置 259
5.1 IP 地址 259
5.2 ARP 地址解析协议 260
5.3 ARP 模式 260
Enabled 261
Disabled 261
Reply-only 261
Proxy-arp 262
Local Proxy Arp 265
5.4 ARP 绑定 265
ARP 双向绑定事例 267
ARP Timeout 设定 267
第六章 Route（路由） 268
6.1 RIB 和 FIB 268
举例 269
6.2 RIB 路由信息库（生成路由表） 271
默认路由（Default route） 271
直连路由（Connected routes） 272
等价多路径路由(ECMP) 272
网络接口作为网关 273
路由距离（Distance） 273
6.3 FIB（转发信息库） 273
Routing table lookup（路由表查询） 274
6.4 RouterOS 路由配置 275
静态路由 275
ECMP 等价多路径路由 276
策略路由 277
6.5 ISP 目标地址路由 277
6.6 默认网关断线处理 279
6.7 RouterOS 策略路由介绍 280
6.8 RouterOS 源地址策略路由 283
源地址策略路由实例 283
策略路由导致内网段互访失效 287
6.9 奇偶数源地址策略路由标记 289
6.10 光纤和 ADSL 静态路由 293
6.11 HTTP 端口的策略路由 295
6.12 PPTP VPN 借线路由操作 298
配置 PPTP-Server 298
配置 PPTP-Client 301

路由配置 301
6.13 RouterOS PCC 负载均衡 303
PCC 原理 304
PCC 的负载均衡事例 306
六线的 PCC 配置 316
PCC 多线比例权重路由 317
PCC 在 IDC 应用 319
6.14 单网卡多次拨号 PPPoE 配置 323
Bridge VLAN 多线路拨号 323
单网卡 VRRP 多拨配置 325
6.15 多线接入的路由问题 330
PCC 返程路由配置 330
返程路由原理 331
返程路由的三种场景 332
策略路由导致外网 nat 端口映射内网访问失效 336
6.16 PPPoE 走下行，商务专线走上行 337
第七章 DHCP 操作配置 341
7.1 DHCP-Client 设置 341
Lease 租约脚本实例 343
7.2 DHCP-Server 设置 343
DHCP Lease 租约 346
7.3 DHCP Options 348
Classless static Route 348
Option-set 349
7.3 Alerts 警报 350
7.4 DHCP-Relay 配置 350
7.5 Vendor Classes（厂商 ID 分类） 353
7.6 Rate limit（流量控制） 355
第八章 DNS 357
8.1 DNS 配置 357
8.2 Cache 监测 358
8.3 静态 DNS 域名解析 359
8.4 清除 DNS cache（缓存） 359
8.5 DoH（DNS over HTTPS） 360
实例 360
8.6 DNS 劫持 361
8.7 DNS forward 转发 362
8.8 RouterOS 搭建私有 DDNS 服务 365
特别说明： 366
DDNS RouterOS 配置 367
节点 RouterOS 372
客户端 RouterOS 374
8.9 DNS cache 捕获域名 IP 地址 375
第九章 Firewall filter（防火墙过滤） 380
9.1 Firewall filter 380
属性描述 381
Chains 过滤链表 385
防火墙过滤方式 387

防火墙过滤流程 387
自定义链表 388
规则条件执行 388
Connection states（连接状态） 389
9.2 防火墙 filter 事例 390
基本配置事例 390
Input 链表 391
Forward 链表 394
Jump 规则的使用 395
基于协议的自定义链表 398
ICMP 类型：代码值 399
源 IP 地址与目标 IP 地址 400
文本过滤 402
过滤指定域名 402
9.3 Address-list 403
使用 address-list 创建黑白名单事例 405
v6.36 允许 address-list 添加域名 407
9.4 Interface list 事例 408
9.5 内网多 IP 段访问控制 409
两个 IP 网络段单向访问控制 411
9.6 RouterOS L7 协议 411
L7 代码控制某聊天软件登录 414
9.7 使用 wireshark 分析网页视频 414
9.8 DMZ 配置事例 417
9.9 RouterOS ip settings 418
9.10 DoS 防御 420
诊断 SYN 攻击 421
9.11 保护路由器 421
第十章 nat（网络地址翻译） 423
10.1 nat 介绍 423
10.2 src-nat 429
Same nat 429
PCC src-nat 负载均衡 430
10.3 dst-nat 431
端口映射配置 432
dst-nat 数据转移 432
dst-nat 数据重定向 432
DNS 重定向 432
1:1 nat 实例（netmap） 432
非对称端口映射 433
正确配置 nat 规则接口参数 433
10.4 Stats 434
10.5 Connection 连接状态 435
10.6 Tracking 连接跟踪 436
10.7 高性能 nat 实践 438
10.8 Raw 440
第十一章 RouterOS Fastpath 442
11.1 FastPath 支持列表 443

11.2 CCR 系列的 Fastpath 445
11.3 IPv4 FastTrack 447
第十二章 Queue（带宽控制） 450
12.1 流控原理 450
12.2 Interface Queue（接口队列） 452
12.3 Queue Type（队列类型） 453
PFIFO/BFIFO 454
SFQ 454
PCQ 455
RED 455
Bursts 456
12.4 Burst 突发值 457
Burst 原理 457
Burst 事例 457
12.5 Simple Queue 简单队列 464
Simple queue 应用举例（仅适用于 v5 和之前版本） 465
Simple Queue 接口流控配置（适用于 v6） 467
Simple Queue 配置父级流控无效问题 468
simple queue 的 CPU 负载问题 469
12.6 RouterOS v6.0 Queue 变动 469
V6.0 前 469
V6.0 后 471
V6.0 配置变动 472
V6.0 Queue 事例 473
环境 1：Queue tree 设置 2M，simple queue 设置 6M 474
环境 2：Queue tree 设置 6M，simple queue 设置 2M 475
12.7 HTB 等级令牌桶介绍 475
双重限制 476
优先级 477
事例 1：普通事例 477
事例 2：max-limit 事例 478
事例 3：inner 队列 limit-at 478
事例 4：leaf 队列的 Limit-at 479
12.8 v6.0 Simple Queue 等级优先 481
12.9 Queue tree 队列树 484
简单的 Queue Tree 实例 485
Queue tree v6 实例 487
Queue tree HTB 配置实例 489
12.10 PCQ 介绍 492
pcq-classifier 分类器 493
PCQ 动态流量分配实例 497
12.11 HTB 与 PCQ 流量控制 500
12.12 网吧的 PCQ 与 HTB 503
HTB 游戏优先 504
12.13 Connection Rate 流量控制 505
Connection-rate HTB 事例 508
12.14 L7 流控 523
12.15 PPP Profile QoS 526

第十三章 Mangle 分类标记 530
13.1 Mangle 介绍 530
13.2 Mangle 应用 531
Peer-to-Peer 传输标记 531
Mangle 限制 2 级代理 532
13.3 Nth 的 mangle 应用 532
Nth 原理介绍 532
Passthrough 对 Nth 的控制 533
Nth 在负载均衡的应用 535
Nth 在端口映射的应用 537
第十四章 Bridge 网桥（二层） 540
14.1 Bridge 配置 540
Bridge setting 542
Bridge Port 543
Bridge monitor 544
Bridge host 545
14.2 Bridge 防火墙 545
Bridge Filter 548
Bridge nat 549
14.3 Bridge 实现二层端口隔离 550
14.4 创建二层透明流控过滤 552
14.5 通过 Bridge Filter 控制 MAC 地址 553
过滤源和目标 MAC 地址 554
过滤指定厂商的 MAC 地址 557
14.6 Bridge nat 修改 VRRP 接口 MAC 559
14.7 Bridge VLAN（access/trunk） 565
三层接口（管理口）配置 568
ingress-filtering 和 frame-type 571
14.8 Bridge HW-Offloading(v6.41) 572
L2 Hardware-offloading 介绍 572
L3 Hardware Offloading 介绍 575
14.9 Bridge VLAN 过滤(v6.41) 579
VLAN 实例 1 (不同 VLAN 配置 Access 接口) 581
VLAN 实例 2 (配置 Trunk 和 Access 接口) 582
VLAN 实例 3 (配置 Trunk 和 Hybrid 接口) 583
VLAN 实例 4(配置 VLAN 三层接口) 584
VLAN 实例 5 (配置 VLAN 三层管理 IP) 586
14.10 不同 MikroTik 硬件的二层 VLAN 配置 587
CRS3xx 系列 587
CRS1xx/CRS2xx 系列 588
其他集成交换芯片设备 588
其他没有集成交换芯片的设备 589
14.11 IGMP Snooping(v6.41) 589
14.12 Fast Forward 590
14.13 (R)STP 592
STP 和 RSTP 594
选举过程 595
实例 596

14.14 QinQ 598
标签堆叠（Tag stacking） 598
三层标签堆叠实例 599
802.1ad 实例 601
14.15 Bridge 下二层错误配置 602
单交换芯片的多 Bridges 配置 602
错误的 VLAN interface 603
interface vlan 加入 bridge 604
第十五章 Interface VLAN（三层） 606
15.1 802.1Q 协议 607
15.2 简单 VLAN 事例 608
15.3 Trunk 连接 609
15.4 三层 QinQ 配置 611
第十六章 Bonding 612
16.1 Bonding 基本操作 612
16.2 Bonding 的七种模式 613
1、balance-rr（Round-robin policy） 613
2、 active-backup 613
3、balance-xor 613
4、broadcast 613
5、802.3ad（IEEE 802.3ad 动态链接聚合） 614
6，balance-tlb（Adaptive transmit load balancing） 614
7， balance-alb（ Adaptive load balancing） 614
16.3 Link monitoring 连接监测 614
ARP 监控 614
MII 监控 615
16.3 RouterOS 与交换机做 Bond 配置 616
方法一、balance-rr 616
方法二、balance-alb 617
16.4 官方 EoIP 隧道的 Bonding 618
16.5 RouterOS 与 CentOS 7 配置 bond 620
RouterOS 配置： 621
CentOS 7 服务器配置 621
第十七章 VRRP (虚拟路由冗余协议) 624
17.1 协议介绍 624
Virtual Router (VR) 625
虚拟 MAC 地址（Virtual MAC address） 625
Master（主路由器） 626
Backup（备份路由器） 626
Virtual Address 626
IPv4 ARP 626
IPv6 ND 626
17.2 VRRP 状态器 627
初始状态 627
Backup 状态 627
Master 状态 628
17.3 简单配置 VRRP 628
操作路径： /interface vrrp 628

IPv4 628
IPv6 628
17.4 VRRP 配置实例 630
基本配置实例 630
负载共享(Load sharing) 631
VRRP 无抢占模式（no Preemption） 633
第十八章 RouterOS 数据流(Packet Flow) 634
18.1 IP 数据流流程图 634
二层网络 634
三层网络 634
18.2 RouterOS 6.0 对 Queue 调整后的数据流 636
18.3 功能模块与结构 637
判断处理 637
18.4 功能处理流程 638
Bridge 设置 use-ip-firewall=yes 638
路由 – 从 Ethernet 到 Ethernet 接口 638
路由 - 从一个桥接口道不同的桥接口 639
IPsec 加密 639
IPsec 解密 639
第十九章 RADIUS 641
19.1 RADIUS 客户端 641
RADIUS 连接终止 644
19.2 RouterOS 连接 RADIUS 备份 644
RADIUS 备份模式 644
第二十章 HotSpot 热点认证网关 646
20.1 HotSpot 介绍 646
20.2 HotSpot Setup 向导配置 648
20.3 HotSpot 接口设置 650
20.4 HotSpot profile 策略 651
20.5 Walled Garden 内院 654
IP 方式 Walled Garden 656
20.6 Hotspot binding 658
20.7 Hotspot host 列表 659
20.8 HotSpot Users 管理 661
20.9 Hotspot active 在线管理 665
20.10 Hotspot 配置事例 666
20.11 Hotspot 即插即用功能 676
20.12 HotSpot 防火墙部分 678
20.13 HotSpot HTTPS 认证登录 681
Self-signed certificates（自签证书） 681
证书信任问题 686
Hotspot HTTPs 登录 686
SSL 证书密钥长度影响 CPU 负载 687
第二十一章 PPPoE 688
21.1 PPPoE Client 设置 689
监测 PPPoE 客户端 690
21.2 ADSL 拨号上网事例 691
21.3 Scanner（搜索器） 692

21.4 PPPoE Server 设置 692
MRRU 介绍 693
21.5 基于 802.11g 无线网络的 PPPoE 服务 694
21.6 Winbox 配置 PPPoE 服务 697
21.7 基于 VLAN 的 PPPoE 认证 701
平台总结 702
QinQ 添加脚本 705
21.8 高并发 PPPoE 认证服务集群 706
网络规划参数： 707
配置出口 nat 路由器 707
配置 PPPoE 认证 1 708
配置 PPPoE 认证 2 709
华为 5700 交换机配置 710
PPPoE 认证负载均衡的冗余配置 711
第二十二章 PPTP 713
22.1 PPTP 客户端 714
22.2 PPTP 服务器 715
22.3 PPTP 应用事例 716
企业与家庭办公网络 PPTP 隧道配置事例 716
通过 PPTP 隧道连接终端客户 721
Windows 的 PPTP 设置 724
第二十三章 PPTP 与 L2TP 服务 725
23.1 同时建立 PPTP 和 L2TP 服务器 725
L2TP 的 Windows 注册表修改 728
23.2 VPN 的几种应用方式 729
23.3 BCP 协议 731
第二十四章 Open VPN 738
24.1 OVPN Client 738
24.2 OVPN Server 739
OVPN 服务器配置 739
24.3 使用 certificate 创建 OVPN 证书 740
24.4 OVPN 配置 745
24.5 OVPN bridge 模式 752
第二十五章 SSTP 介绍 757
25.1 端到服务器连接 760
25.2 点对点的 SSTP 762
25.3 基于 SSTP 隧道的远程端口映射 763
第二十六章 IPSec 配置 766
26.1 IPsec 基本术语： 766
Peer 对等体： 766
SA 安全联盟 766
协商方式 766
IPSec 封装模式 767
IPSec 使用的认证算法和加密算法 767
通信保护协议： 767
26.2 点对点 IPSec（IKEv1）隧道模式 768
Office 1 配置 769
Office2 配置 769

绕过 NAT 和 Fasttrack 规则 770
26.3 基于 DDNS 建立 IPSec（IKEv2）隧道 771
站点 1 (server)配置 771
站点 2 (client)配置 772
26.4 Windows L2TP/IPsec 连接 774
RouterOS 配置 774
Windows 配置 778
第二十七章 EoIP 隧道 782
27.1 EoIP 介绍 782
27.2 EoIP 应用实例 783
第二十八章 GRE 和 IPIP 隧道协议 789
28.1 GRE 隧道 789
配置事例 790
28.2 IPIP 隧道 791
配置事例 792
第二十九章 OSPF 794
29.1 OSPF 介绍 794
OSPF 术语 794
OSPF 路由通信 796
邻居探测 797
数据库同步 798
路由表计算 798
OSPF 路由类型 799
29.2 基本的 OSPF 配置 800
网络配置 801
配置 router-id 801
Area 配置 802
重分布默认路由 803
29.3 多 area 配置 804
29.4 基于 PPPoE 的 OSPF 事例 805
第三十章 BGP 818
30.1 BGP 介绍 818
自治系统 AS（Autonomous System） 818
BGP 分类 818
BGP Instance 实例 819
BGP Peers 819
路由重分布 820
路由过滤 820
RouterOS 的 BGP 静态路由 822
BGP Advertisements 823
BGP 负载均衡 823
30.2 BGP 事例 1 824
配置 BGP Peer 825
network 宣告和路由过滤 825
路径选择 825
负载均衡设置 826
30.3 BGP 事例 2 827
第三十一章 Proxy 代理 836

31.1 Proxy 基本介绍 836
31.2 Proxy 配置 837
1、启用常规 proxy 服务 837
2、启用透明 proxy 服务 838
31.3 启用缓存. 838
RAM proxy 缓存 839
Store proxy 缓存 839
31.4 access 访问列表 841
proxy 访问控制实例 842
31.5 Direct 直接访问列表 844
31.6 cache 缓存管理 845
31.7 连接列表 845
31.8 Web proxy 应用事例 846
31.9 重定向 URL 请求 848
第三十二章 CHR 与虚拟化 851
32.1 CHR(Cloud Hosted Router) 851
系统要求 852
安装 CHR 852
32.2 CHR 许可 853
等级许可说明 853
32.3 购买许可 854
从免费许可升级到 P1 或更高版本 855
32.4 CHR 安装到 vmware esxi 平台 858
32.5 在 VPS 上安装 CHR 869
查看 VPS 配置 869
使用 dd 命令安装 RouterOS 870
32.6 KVM 873
基本要求 873
创建 KVM RouterOS 虚拟机 874
创建接口 874
连接虚拟机 875
配置虚拟网络实例 876
删除 KVM 虚拟机 878
32.6 MetaRouter 介绍 878
MetaRouter 事例 880
第三十三章节 IPv6 887
33.1 地址格式 888
33.2 Prefix（前缀） 889
地址类型 889
33.3 Interface Identifier（接口标识） 890
EUI-64 890
33.4 IPv6 地址分配方式 891
手动静态地址分配 891
33.5 IPv6 ND 892
StateLess Address AutoConfiguration（无状态地址自动配置） 892
Neighbors（邻居状态） 893
IPv6 无状态地址自动配置实例 894
33.6 DHCPv6-Server 897

组播地址 898
DUID 898
身份联盟（IA） 898
DHCPv6 基本配置 899
Bindings 900
Rate limiting（限速） 900
33.7 DHCPv6-Client 901
快速设置 901
属性 902
DHCPv6 Server 与 Client 配置实例 903
33.8 Route（IPv6 路由） 904
33.9 IPv6 防火墙 906
IPv6 路由器本地防火墙策略 906
IPv6 用户侧防火墙策略 907
33.10 IPv6 应用实例 908
家宽获取并分配 IPv6 地址 908
实现 PPPoE 服务器 IPv4 和 IPv6 双栈网络配置 912
33.11 IPv6 Setting 915
第三十四章 RouterOS Stores/disk 功能 **916**
34.1 RouterOS 使用 U 盘扩展存储 916
34.2 存储 log 日志信息 917
34.3 Web-Proxy 使用 U 盘存储 920
34.4 Store/disk 的其他应用 921
第三十五章 UPNP **923**
35.1 UPNP 接口 923
35.2 配置事例 924
第三十六章 RouterOS 工具 **926**
36.1 Netwatch 监控 926
36.2 绘图显示（Graphing） 928
健康情况 929
接口流量图 929
流控图显示 929
资源图表 930
36.3 Bandwidth-test 带宽测试 932
36.4 Torch (实时通信监听) 934
36.5 E-mail 发送工具 938
36.6 Fetch 文件拷贝 939
发送信息到远端主机 940
返回结果赋值给一个变量 941
SFTP 941
36.7 Packet Sniffer 941
36.8 SMB 948
36.9 Profiler 工具 954
多 CPU 平台查看 954
36.10 Traffic Generator（流量发生器） 956
基于 Winbox 的简单使用方法 956
36.11 Speed test 961
事例 961

第三十七章 SOCKS 代理 963
37.1 SOCKS Server 963
Access List（访问列表） 963
Active Connections（活动连接） 963
User（账号） 964
37.2 应用实例 964
FTP 服务通过 SOCKS 代理服务器 964
Firefox 如何连接 socket5 代理 966
第三十八章 User Manager v4 970
38.1 User Manage 基本配置 970
创建 Customer 账号 971
操作接口 972
语言设置 974
38.2 对接 RouterOS 配置事例 975
38.3 PPPoE 认证的事例操作 979
38.4 Hotspot 认证 User Manager 连接 995
38.5 用户自助服务 996
38.6 User Manager 数据导出与导入 999
第三十九章 Scheduler（计划任务） 1000
39.1 计划任务介绍 1000
39.2 计划任务事例 1000
Changelog 1004
参考文献： 1010

应用说明

主要特征

我第一次接触 RouterOS 在 2003 年，当时版本是 v2.7.14，现在 RouterOS 已经发展到了 v7.x 版本，主流版本是 v6，使用 Linux 3.3.5 的内核，最新的 v7 版本使用 Linux 5.6.3 内核。

RouterOS 将每个版本分为多个等级，以区分功能的不同和限制。等级越高功能越多，限制越少，Level6 是最高等级，Level0 为 24 小时试用版本，以下是 RouterOS 的等级功能与简介（关于虚拟机平台的 CHR 等级是按照接口速率分级，具体可以参考 32 章节）：

等级/功能	LeveL 0	LeveL 1	LeveL 3	LeveL 4	LeveL 5	LeveL 6
价格	无许可	官网注册申请	不单独销售	$45	$95	$250
无线 AP	24 小时限制	无	无	支持	支持	支持
无线客户端和 Bridge	24 小时限制	无	支持	支持	支持	支持
RIP, OSPF 和 BGP	24 小时限制	无	支持	支持	支持	支持
EoIP 隧道	24 小时限制	1	无限制	无限制	无限制	无限制
PPPoE 在线用户	24 小时限制	1	200	200	500	无限制
PPTP 在线用户	24 小时限制	1	200	200	500	无限制
L2TP 在线用户	24 小时限制	1	200	200	500	无限制
OVPN 在线用户	24 小时限制	1	200	200	无限制	无限制
VLAN 接口	24 小时限制	1	无限制	无限制	无限制	无限制
HotSpot 在线用户	24 小时限制	1	1	200	500	无限制
RADIUS 客户端	24 小时限制	无	支持	支持	支持	支持
Queues 流控规则	24 小时限制	1	无限制	无限制	无限制	无限制
Web proxy	24 小时限制	无	支持	支持	支持	支持
User Manager 在线用户	24 小时限制	1	10	20	50	无限制
KVM 主机限制	none	1	无限制	无限制	无限制	无限制

x86 平台

- AMD、Intel、VIA 和其他兼容的 x86 平台
- SMP – RouterOS 3.0 后相容多核心 CPU 和多 CPU（RouterOS v6.x 对多核心处理做更好的优化）；
- 内存：最小 32MB，RouterOS v2.9 版本支持 1G 内存，RouterOS v3.0 后支持 2G 内存；
- 存储：IDE 或 SATA 接口硬盘，CF 存储卡、USB 或 DOM 闪存盘，最小至少需要 32MB 空间，建议系统硬盘不大于 80G，否则安装格式化要等待很长时间，所以尽量选择小容量的 SSD 或 CF 卡等 Flash 存储，不支持 SAS 和 SCIS 接口，不支持阵列卡；
- RouterOS 5.0 采用 Linux 2.6 内核，并开始支持的扩展槽 PCI、PCI-e、PCI-X，
- RouterOS 6.0 采用 Linux 3.3.5 以上的内核，并解除对 16 个 CPU 核心限制。
- RouterOS 7.0 采用 Linux 5.6.3 内核，并支持 UEFI 引导
- 2016 年推出 CHR，基于 x86 虚拟化平台

MIPS 平台

- **MIPS** – 4kc MIPS　RouterBOARD 500 (532, 512 和 511)与 RouterBOARD 100 (133、133c、150、192)

- **Atheros** – 24kc MIPS RouterBOARD 400(411/411A/411AH、433/433AH/433UAH、450/450G、493/493AH)
- **Atheros** – 24kc MIPS RouterBOARD 700(711、711A、750/750G、750UP、751 系列，SXT、Groove、OmniTik)
- **Atheros** –74kc MIPS RouterBOARD2011 系列
- **QCA** – 24kc MIPS hAP 系列
- **QCA** –74kc MIPS RouterBOARD900 系列(911、951、912)
- **MTK7621 –** 1004kc MIPS RB750Gr3，RB-M11，RB-M22

PPC 平台

- RouterBOARD1000、RouterBOARD1100、RouterBOARD800、RouterBOARD600、RouterBOARD333
- RouterBOARD1100AH/AHX2, RouterBOARD1200

Tilera 平台

- 系统支持 Tile-GX - CCR1009、CCR1016、CCR1036 和 CCR1072(Tile-GX 系列是专门针对高端服务器市场，例如大型数据中心和云计算等网络应用产品。Tile-GX 为 64 位 RISC 架构处理器，Tilera Tile-GX 采用台湾台联电 40nm 晶圆，处理器功耗从 15w~48w)
- 基于除了 CCR1016、1036 和 1072 平台的 RouterOS 是 64bit Linux，其他都是 32bit 系统

ARM 平台

- RouterBOARD3011 是第一款基于高通 ARM 平台的 IPQ-8064 处理器
- RB1100×4 和 RB4011iGS Alpine AL21400 处理器
- hAP ac2 采用高通的 IPQ-4018 处理器
- CCR2004 系列采用 Annapurna 公司的 AL32400（ARM 64bit）
- RB5009

安装

- Netinstall: 网络安装，基于 PXE 或 EhterBoot 启用的网卡的网络安装
- Netinstall: 利用安装在 windows 中的 Netinstall 软件向 U 盘写入安装档，实现 U 盘安装
- CD 镜像档安装

配置

- MAC 地址访问与配置
- WinBox - 基于窗口化配置的 GUI 图形工具
- Web 接口配置工具 webfig（webbox 从 5.0 被 webfig 替代）
- 强大的命令行配置接口，集成脚本编辑功能，可通过本地终端、console 管理、telnet 和 ssh 访问配置
- API - 能创建你自己的配置和监测应用程序

备份与恢复

- 支持二进制配置备份文件 ，并支持对备份文件加密
- 通过 Exprot 和 import 可生成读写的文本格式

Firewall

- 连接状态过滤功能

- 源和目标 NAT
- NAT 协议支持(h323, pptp, quake3, sip, ftp, irc, tftp)
- 内部连接状态，路由和数据报标记
- 防火墙过滤，对 IP 地址和地址范围，端口和端口范围，IP 协议，DSCP 等其他功能
- 支持地址列表创建
- 专业的 Layer7 匹配器
- IPv6 支持
- PCC - 每次连接分类器，用于负载均衡配置
- Nth - 第 N 次连接数据标记，用于连接排序和负载均衡
- 支持 tls 过滤
- …

路由

- 静态路由
- 虚拟路由转发（Virtual Routing and Forwarding - VRF）
- 路由策略
- 接口路由
- ECMP 路由策略
- IPv4 动态路由协议: RIP v1/v2, OSPFv2, BGP v4
- IPv6 动态路由协议: RIPng, OSPFv3, BGP
- 双向转发检测 (BFD)
- Openflow

MPLS

- 支持 IPv4 静态标记绑定
- IPv4 的标记分布协议
- RSVP 传输工程隧道
- VPLS MP-BGP 自动探测和信号发送
- MP-BGP 基于 MPLS IP VPN

VPN

- Ipsec - 隧道和传输模式, 证书或者 PSK, AH 和 ESP 安全协议，RB1000 支持硬件加密
- 点对点隧道(OpenVPN、PPTP、PPPoE、L2TP、SSTP)
- 高级 PPP 功能(MLPPP、BCP)
- 简单隧道(IPIP、EoIP)
- 支持 6to4 隧道(IPv6 基于 IPv4 网络)
- VLAN - IEEE802.1q 虚拟局域网，支持 Q-in-Q 模式
- 基于 MPLS 的 VPN

Wireless

- IEEE802.11a/b/g/n/ac 无线 AP 和客户端
- Nstreme 和 Nv2 私有协议
- 无线分布式系统(WDS)
- 虚拟 AP
- 安全支持 WEP, WPA, WPA2

- 访问控制列表
- 无线客户漫游
- 支持 WMM
- HWMP+ 无线 Mesh 协议
- MME 无线路由协议
- 支持 3G 和 LTE（4G）无线模块

DHCP

- 支持每个接口的 DHCP 服务
- DHCP 客户端和中继
- 静态和动态 DHCP 租约
- 支持 RADIUS
- 自定义 DHCP 选项

Hotspot

- 即插即用网络访问功能
- 通过 web 对本地网络客户验证
- 用户帐户记录
- 支持 RADIUS 验证与计费

QoS

- 令牌桶(HTB)QoS 体系，定义优先级
- 简单快速的 QoS 工具(Simple queues)
- 动态客户端速率均衡(PCQ)
- Queues 在 6.0 后做了较大调整，比对优化的性能，处理方式也较之前版本有所变动。

Proxy

- HTTP 缓存代理服务器
- 透明 HTTP 代理
- 支持 SOCKS 协议支持
- 支持缓存到一个指定的驱动器
- 支持父级代理
- 访问控制列表
- 缓存列表

FastPath

- 基于 RouterBOARD 硬件
- 允许 IPv4 数据报转发不在经过 Linux 内核处理，直接由芯片处理，进一步提升转发速度

工具

- Ping, traceroute
- 带宽测试 Bandwidth test, ping flood
- 数据报 sniffer 工具, torch 工具

- Telnet, ssh
- E-mail 和 SMS 短信发送工具
- 自动脚本执行工具
- 文件 Fetch 工具

其他功能

- 桥接 Bridging – 生成树协议(STP, RSTP), 桥接防火墙与 MAC nat 功能
- DDNS 工具
- NTP 客户端/服务器和 GPS 系统
- VRRP 虚拟冗余路由协议
- SNMP
- M3P – MikroTik 数据报封装协定
- MNDP – MikroTik 邻居探测协议，支持 CDP（思科探测协议）
- 支持 RADIUS 验证与计费
- TFTP 服务器
- 支持 Synchronous 接口(仅支持 Farsync 卡)
- Asynchronous – 串口 PPP 拨号 dial-in/dial-out
- 支持 ISDN
- 支持 SMB 的文件共享功能

操作配置

RouterOS 提供了强大的命令配置接口。你可以通过简易的 Windows 远程图形软件 WinBox 管理路由器。同样也提供了网页配置的 Webfig：

- Winbox 与 webfig 完全一至的配置界面
- 提供实时的配置和监控界面
- 支持多个连接访问
- 为管理员提供用户分组策略配置
- 恢复和撤销历史记录，undo/redo 操作
- 安全模式操作
- Scripts 能事先安排运行时间和执行内容，脚本支持所有的命令操作。

路由器可用通过下面的接口进行管理：

- **本地 teminal console** - PS/2 或 USB 键盘和 VGA 显示适配器进行控制
- **Serial console** – 任何 (默认使用 COM1) RS232 异步串口，串口默认设置为 9600bit/s, 8 data bits, 1 stop bit, no parity, hardware (RTS/CTS) flow control。
- **Telnet** - telnet 服务预设运行在 TCP 端口 23
- **SSH** - SSH (安全 shell) 服务预设运行在 TCP 端口 22
- **MAC Telnet** - MikroTik MAC Telnet 协议是预设启用在所有的类似以太网卡的接口上。
- **Winbox** - Winbox 是 RouterOS 的一个 Windows 远程图形管理软件，使用 TCP 端口 8291（3.0rc13 版本后支持修改 winbox 的端口），同样也可用通过 MAC 地址连接。
- **Webfig** - 通过 HTTP 登录到路由的一种 web 接口管理，管理接口类似 Winbox，无需安装任何客户端。

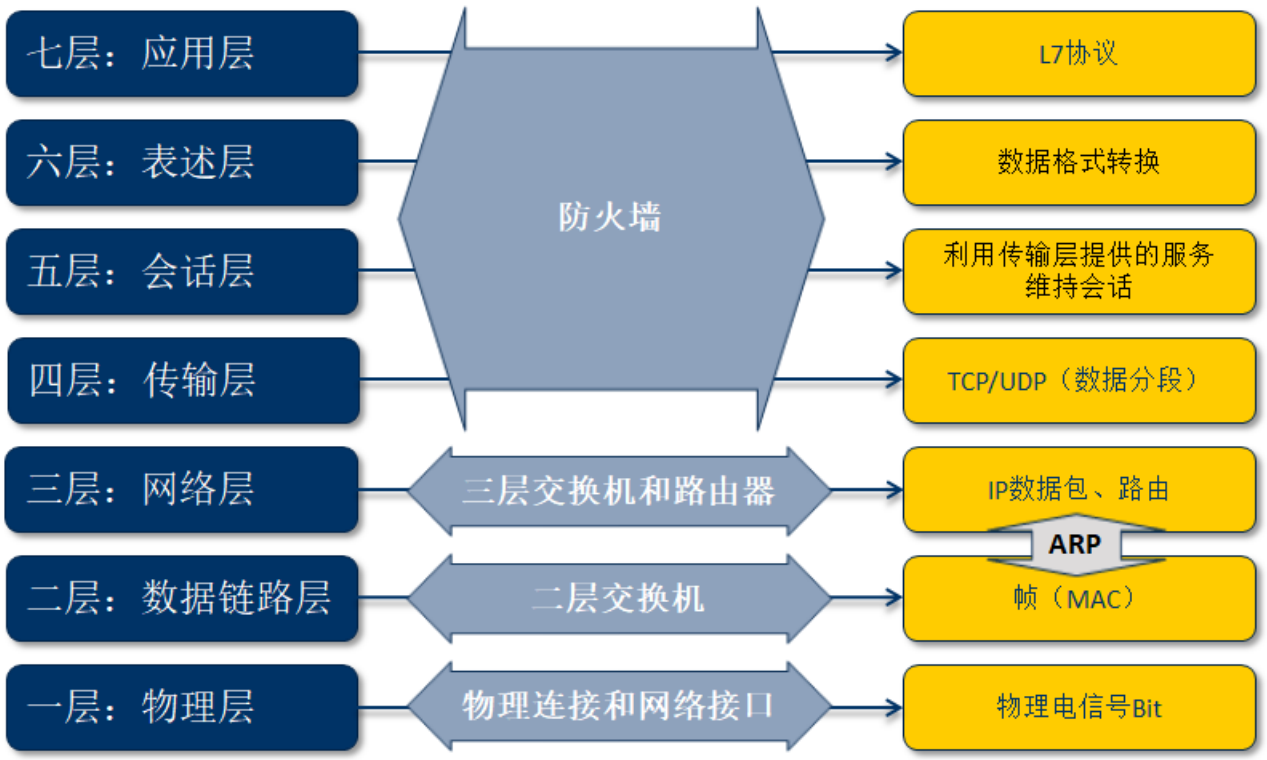
基础知识

在学习 RouterOS 前，我建议大家需要了解一些必备的网络知识，特别是路由交换，可以去查看一些相关的网络书籍和数据，例如 CCNA 和 HCNA 这些入门网络工程师的数据。

- **熟悉 OSI 七层参考模型**
- **需要了解基本的 TCP/IP 知识**
- **路由器的工作原理**

OSI 七层参考模型

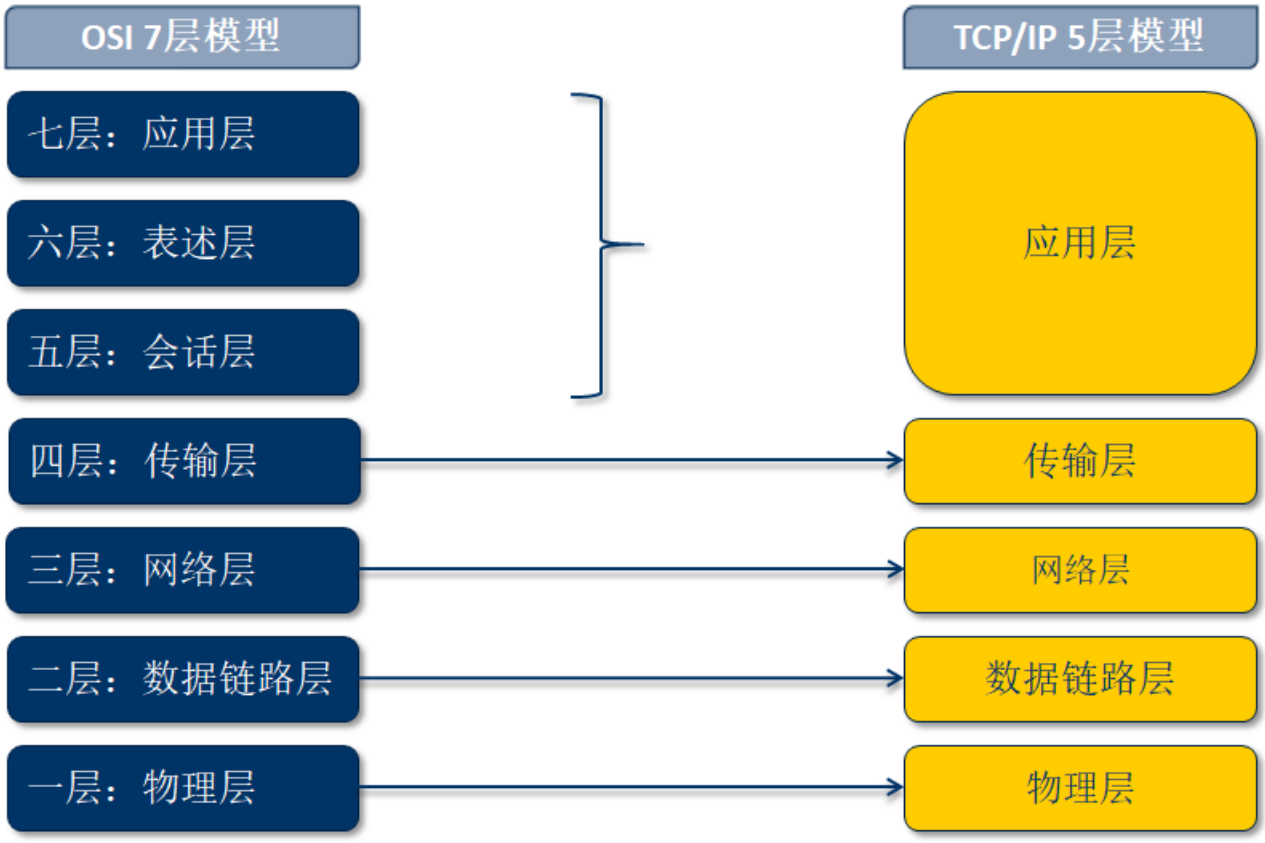
OSI 七层参考模型是任何学习网络的人都必须了解的模型，如下图一个七层模型的结构和对应设备和协议：



OSI 七层模型是有 7 层组成，分别是物理层、数据链路层、网络层、传输层、会话层、表述层和应用层组成，前三层都有对应的设备。(相关视频介绍)

- 运行在物理层的有 Hub 和中继器（早期网络应用，现在已经淘汰），以 bit 流，即“011010101”,物理层传输最基本的数据；
- 运行在数据链路层的有二层交换机和网桥设备，通过 MAC 地址通信将 bit 流封装为帧进行传输；
- 运行在网络层的有路由器和三层交换机，通过 IP 地址通信；
- 传输层主要是通过 TCP 和 UDP 等传输协议将我们的 IP 数据报发送到指定目的地；
- 会话层主要对 TCP 和 UDP 进行建立会话的维护
- 应用层，主要是 http、DNS、FTP、P2P 等应用协议等；

TCP/IP 则划分为五层，但和七层协议是对应的，只是后面三层合并为应用层



在早期网络教材 TCP/IP 协议分层是五层，随着网络的发展提出了 TCP/IP 四层模型，将物理层和数据链路层合并，不管是哪一种都是基于七层模型分类，四层模型个人感觉是让程序员更好的理解网络分层：五层模式更适合网络工程师，网络工程师口中的二层（交换机），三层（路由器）都是基于七层模型或者 TCP/IP 的五层模型，如果 TCP/IP 四层模型，那么交换机变一层，路由器变二层了。

二层 数据链路层

二层交换机属数据链路层设备，可以识别数据报中的 MAC 地址信息，根据 MAC 地址进行转发，并将这些 MAC 地址与对应的端口记录在自己内部的一个地址表中。

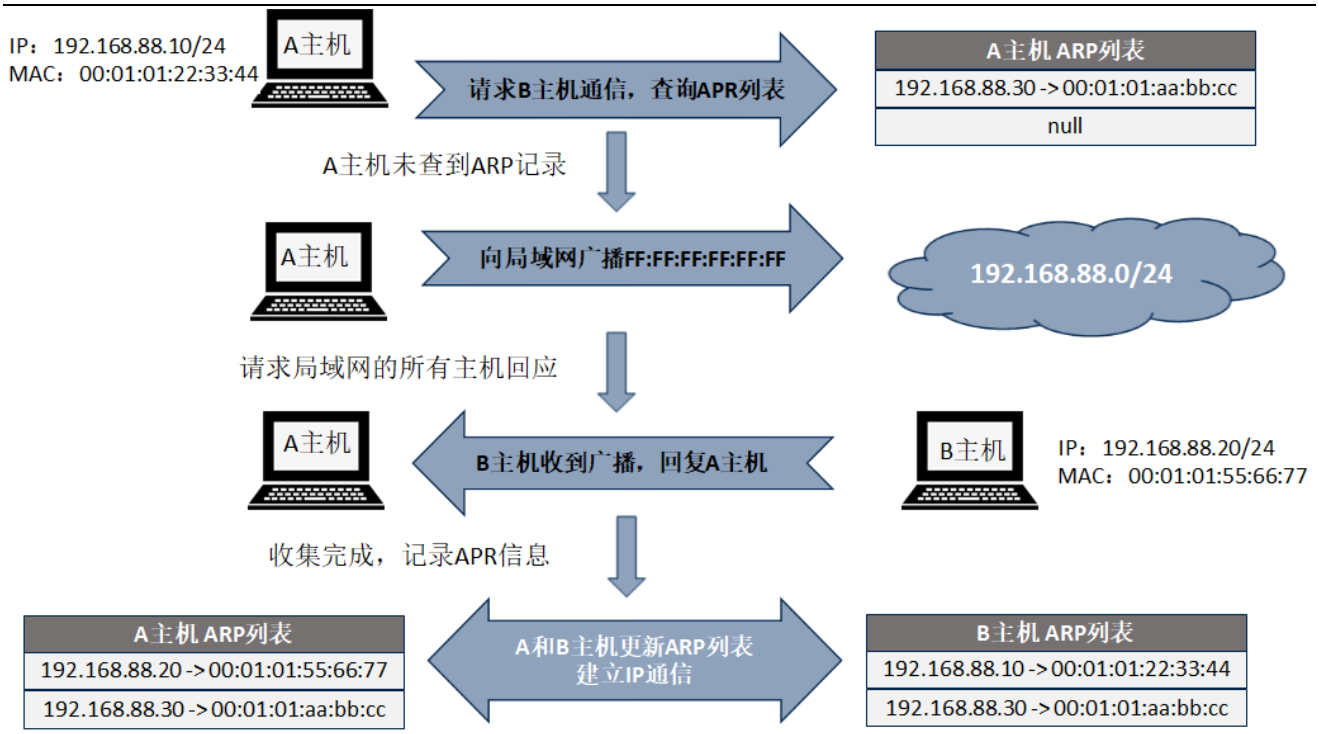
交换机的工作原理是学习、存储和转发，从各个网口上学习 MAC 地址，并存储到交换机的列表中，如果有数据发送到交换机，则查找列表中的目标 mac 转发数据

网络层下面是数据链路层，为了它们可以互通，需要“粘合”协议。ARP（地址解析协议）用于把网络层(3 层)地址映射到数据链路层(2 层)地址，RARP(反向地址解析协议)则反之。

虽然 ARP 的定义与网络层协议无关，但它通常用于解析 IP 地址；最常见的数据链路层是以太网。因此下面的 ARP 和 RARP 的例子基于 IP 和以太网，但要注意这些概念对其他协议也是一样的。

ARP 地址解析协议

网络层地址是由网络管理员定义的抽象映射，它不去关心下层是哪种数据链路层协议。然而，网络接口只能根据二层物理地址（MAC）来互相通信，二层 MAC 地址和三层 IP 地址通过 ARP 协议得到相互需要的地址信息。每台主机都有自己的 ARP 列表，建立 MAC 与 IP 的对应关系，下面是一个 ARP 的工作原理图：

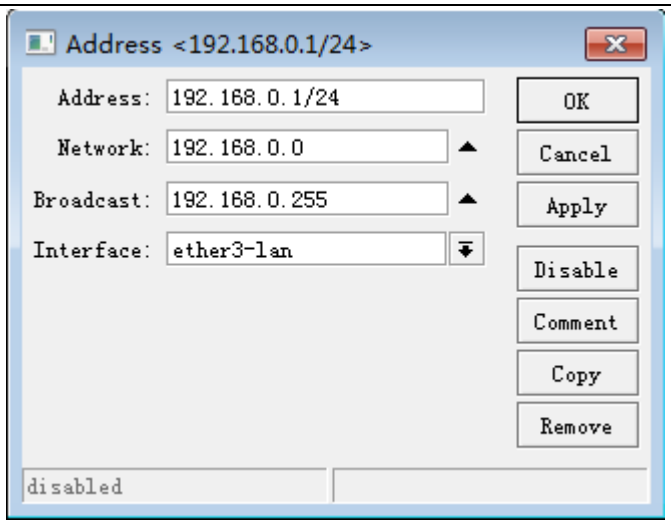


三层 网络层

- IP 地址组成是由主机地址和子网掩码组成
- 主机 IP 地址规定了主机在网络中的具体 IP，即名称
- 子网掩码规则了这个 IP 段在网络中的范围
- IP 地址是 192.168.0.1，子网掩码 255.255.255.0
- 即代表了 IP 地址是 192.168.0.1，局域网中可以通信的范围是 192.168.0.1-192.168.0.254

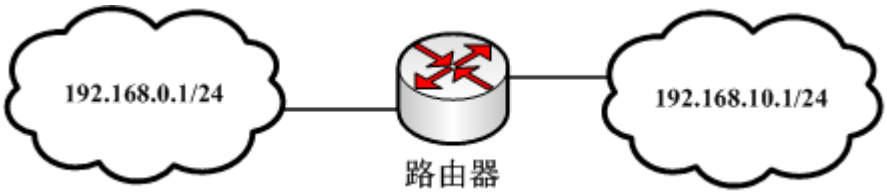
IP 地址

- 255.255.255.0 如用二级制表示每个 255 都是 2 的 8 次方（0-255）256
- 我们可以用多少位代替十进制表示为 24（3 个 255，3*8=24）如右图的 winbox 添加 IP 地址
- Address：IP 地址/子网掩码
- Network：网络地址 192.168.0.0
- Broadcast：广播地址 192.168.0.255
- Interface：将这个 IP 设置到那一个网卡上
- 可以理解为 Network 是起始地址，Broadcast 为结束地址，但这个两个地址被保留，只能是中间范围的 IP 地址可以被使用

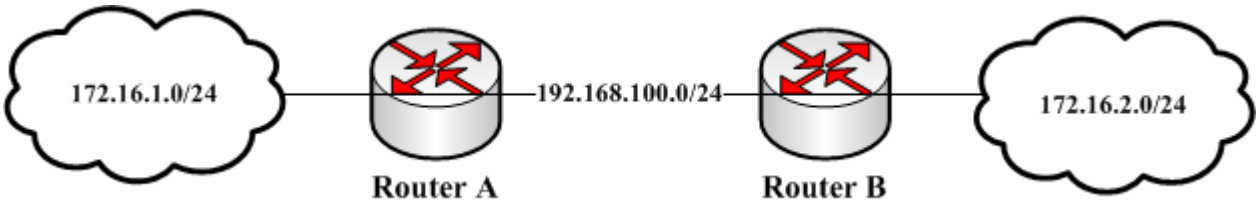


路由器

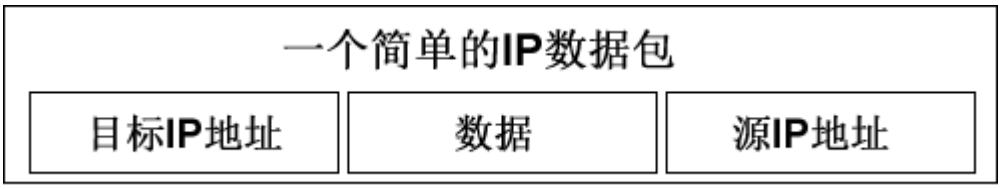
相同子网 IP 地址段连接可以使用交换机，但如果两个或两个以上不同 IP 段，如 192.168.0.1/24 和 192.168.10.1/24 之间是不能通过二层网络转发通信的，需通过三层的路由器连接



路由器：连接两个或两个以上不同 IP 网段的设备



路由表：是路由器根据 IP 数据报的源和目标地址进行路径选择的依据。IP 数据的传输类似于接力，每个路由器都是一个接点，根据 IP 数据报的目的地将数据通过一个接一个的路由器放送到指定的目的地。



路由表

如果一个主机有多个网络接口，当向一个特定的 IP 地址发送分组时，它怎样决定使用哪个接口呢？答案就在路由表中。来看下面的例子：

目的	子网掩码	网关	标志	接口
201.66.37.0	255.255.255.0	201.66.37.74	A	eth0
201.66.39.0	255.255.255.0	201.66.39.21	A	eth1

主机将所有目的地为网络 201.66.37.0 内主机(201.66.37.1-201.66.37.254)的数据通过接口 eth0(IP 地址为 201.66.37.74)发送，所有目的地为网络 201.66.39.0 内主机的数据通过接口 eth1(IP 地址为 201.66.39.21)发送。标志 A 表示该路由状态为“active”（即启动状态）。对于直接连接的网络，一些软件并不象上例中一样给出接口的 IP 地址，而只列出接口。

此例只涉及了直接连接的主机，那么目的主机在远程网络中如何呢？如果你通过 IP 地址为 201.66.37.254 的网关连接到网络 73.0.0.0，那么你可以在路由表中增加这样一项：

目的	子网	网关	标志	接口
73.0.0.0	255.0.0.0	201.66.37.254	AS	eth0

此项告诉主机所有目的地为网络 73.0.0.0 内主机的分组通过 201.66.37.254 路由过去。标志 S(static)表示此项通过静态指定把分组导向外部网关。类似的，也可以定义通过网关到达特定主机的路由，也标志为 S：

目的	子网	网关	标志	接口
91.32.74.21	255.255.255.255	201.66.37.254	AS	eth0

重迭路由

假设在路由表中有下列重迭项：

目的	子网	网关	标志	接口
1.2.3.4	255.255.255.255	201.66.37.253	AS	eth0
1.2.3.0	255.255.255.0	201.66.37.254	AS	eth0
1.2.0.0	255.255.0.0	201.66.37.253	AS	eth1
default	0.0.0.0	201.66.39.254	AS	eth1

之所以说这些路由重迭是因为这四个路由都含有地址 1.2.3.4，如果向 1.2.3.4 发送数据，会选择哪条路由呢？在这种情况下，会选择第一条路由，通过网关 201.66.37.253。原则是选择具有最长(最精确)的子网掩码。类似的，发往 1.2.3.5 的数据选择第二条路由。

注意：这条原则只适用于间接路由(通过网关)。把两个接口定义在同一子网在很多软件实现上是非法的。例如下面的设置通常是非法的（不过有些软件将尝试在两个接口进行负载平衡）：

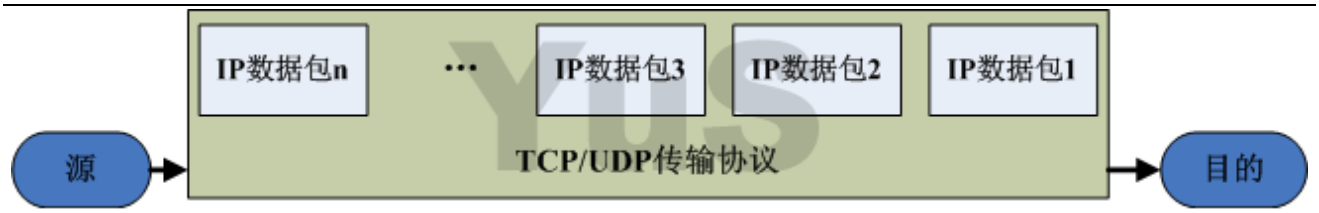
接口	IP 地址	子网掩码
eth0	201.66.37.1	255.255.255.0
eth1	201.66.37.2	255.255.255.0

对于重迭路由的策略是十分有用的，它允许缺省路由作为目的为 0.0.0.0、子网掩码为 0.0.0.0 的路由进行工作，而不需要作为路由软件的一个特殊情况来实现。

四层 传输层

传输层顾名思义就是，将数据传输到需要达到的目的地，通常传输协议采用 TCP 和 UDP 协议，TCP 是装载着 IP 数据报由源发向目标。

- TCP 是一个可靠的协议，有差错检测确认数据完整的到达目的地，如 HTTP 和 FTP 协议等。
- UDP 提供端到端的数据报/无连接服务，UDP 不能保证数据报的接收顺序同发送顺序相同，甚至不能保证他们是否全部到达，UDP 开销小，效率高。如 DNS、SNMP 等



RouterOS 基本 FAQ

1、什么是 RouterOS

- 一种基于 Linux 内核的独立路由操作系统，不需要依附在其他的操作系统安装；
- 支持大部分主流的网络协议，具备操作系统的特性；
- 脚本的编辑功能，实现系统智能化运行；
- 兼容 x86 平台的硬件，如 Intel 和 AMD 等，开发了基于 MIPS、PowerPC、ARM 和 Tile 等平台的 RouterBOARD、CRS 和 CCR 系列，还开发了基于虚拟化的 CHR 平台。

2、MikroTik RouterOS 能做什么？

MikroTik RouterOS 是路由操作系统，是基于 Linux 核心开发，兼容 x86 PC 的路由软件，将普通 PC 变为高性能路由器，现在已移植到 MikroTik RouterBOARD 等硬件平台运行。RouterOS 开发和应用上不断的更新和发展，软件经历了多次更新和改进，使其功能在不断增强和完善。特别在 Wlan 无线、认证、策略路由、带宽控制和防火墙过滤等功能上有着非常强大的功能，其极高的性价比，受到许多网络人士的青睐。

3、在我购买 MikroTik RouterOS 许可前是否可以测试该软件？

是的，你可以从 MikroTik 的网站下载 x86 安装文件，并可以通过安装档建立自己的 MikroTik x86 路由器，路由器在未注册前，所有的功能可以测试 24 小时，有足够的时间测试，因为时间是根据系统运行时间记录的，如果你每天测试 8 个小时，一共可以测试 3 天。

4、运行速度如何？

使用普通 PC 的 RouterOS 处理能力相对于多数路由器都要快，随着 x86 构架的 PC 不断发展 CPU 处理速度和性能也在不断提升。MikroTik 官方陆续已经推出各种基于 MIPS，ARM 和 Tile GX 平台的硬件路由器，由于这些平台的硬件优化更加深入，某些方面已经超过了高性能的 x86 处理器。

5、MikroTik RouterOS 是否支持多 CPU 处理？

在 RouterOS 3.0 版本后支持多 CPU 运行，3.0-4.0 版本支持 8 核心处理器，早期的版本不支持多 CPU。5.0 版本后支持 16 核心处理器，特别是在 6.0 版本对多核 CPU 做了更好的优化，解除了 CPU 数量限制。

6、MikroTik RouterOS 是否支持 SATA、USB 和 SCSI 安装

RouterOS 3.0 版本支持 SATA 和 USB 安装，早先版本不支持，SCSI 不支持，v6.39 以后支持 NVMe SSD 驱动。

7、软件路由相对于 Cisco 路由器如何？

这个问题可能现在有点过时，不过对于早期 RouterOS 能实现专业路由器大部分的功能，成本却是他的很小一部分，更灵活处理和方便升级，相对简易的管理与维护，但随着 MikroTik 产品的不断发展，现在已经成为专业无线、路由、交换设备的生产厂商。

8、MikroTik RouterOS 支持那些硬件平台，有什么区分？

RouterOS 现在基于多种平台，但已经逐渐从 PC 的 x86 构架向硬件平台的 RouterBOARD、CRS、CCR 等过度，这些硬件平台包括 MIPS、PPC、Tilera 和 ARM 等构架的处理器，软件和硬件的相互优化能有效的发挥 RouterOS 性能。

9、MikroTik RouterOS 是否需要其他 OS（操作系统）支持？

不需要任何操作系统的支持，MikroTik RouterOS 是独立的操作系统。OS 是基于 Linux 内核，并且非常稳定。你的硬件驱动将会被 RouterOS 自动识别（需 RouterOS 所支持的驱动），安装在 PRIMARY MASTER HDD（即 IDE1 主盘）、Flash 硬盘或 USB 硬盘，外接硬盘只为 Web 缓存和 User Manager 数据存储。

10、安装后路由器的安全如何？

访问路由器需要通过用户名和密码，通过添加用户或分配用户组管理用户登录路由器，远程访问可以通过用户的 IP 地址限制。防火墙过滤能很好的保护你的路由器和你的网络。

11、MikroTik RouterOS script（脚本）有什么作用？

脚本是 RouterOS 路由操作系统的重要组成部分，即 RouterOS 内部语言，可以通过脚本编写将不同应用功能联系在一起，完成指定的工作，实现路由器运行的智能化。

第一章 RouterOS 基本操作

1.1 RouterOS 安装介绍

1、使用 netinstall 网络安装程序，主要用于 RouterBOARD、CRS 或 CCR 平台，也支持 PXE 的 x86 平台。

2、通过 ISO 镜像文件刻录到光盘引导安装（用于 x86 系统）：即支持 AMD、Intel、VIA 和其他 x86 系统，硬盘支持 IDE、SATA 硬盘接口（不支持 SAS 和 RAID 卡），ISO 镜像小于 30M，所以建议安装到容量较小的 SSD 或 FLASH 硬盘上，如果没有特殊需求，建议系统硬盘不超过 32G。

3、使用 U 盘安装，仅支持 X86 平台，RouterOS 限 v3.0 以上版本（U 盘作为系统盘使用）。

4、CHR（Cloud Host Router）基于虚拟化平台安装，请参见第三十二章节

NetInstall 安装

Netinstall 是网络安装工具，这种方式应用于 RouterBOARD、CRS 和 CCR 的 RouterOS 系统安装和复位,也支持 PXE 网卡引导安装 RouterOS 到 PC。Netinstall 安装工具需要从官方网站下载：
https://mikrotik.com/download

1、安装 RouterBOARD

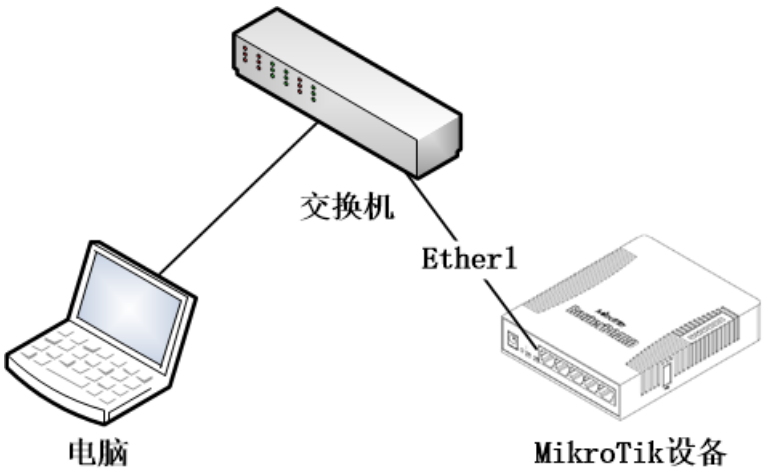
这个事例将介绍如何一步一步在一个 RouterBOARD 上重新安装 RouterOS 软件，同样在你丢失了 RouterBOARD 登录密码后，也可以通过该方法复位 RouterOS。

Netinstall 早期需要通过 console 口连接进入 RouterBOARD 的“BIOS”修改引导方式为 ethernet（以太网引导），即通过使用 netinstall 工具进行网络安装，现在 RouterBOARD 提供了新的方式可以在没有 console 口的情况安装，一种是 winbox 设置引导，一种是长按固件复位按钮引导（具体 RouterBOARD 的复位操作参见 3.5 章节）。

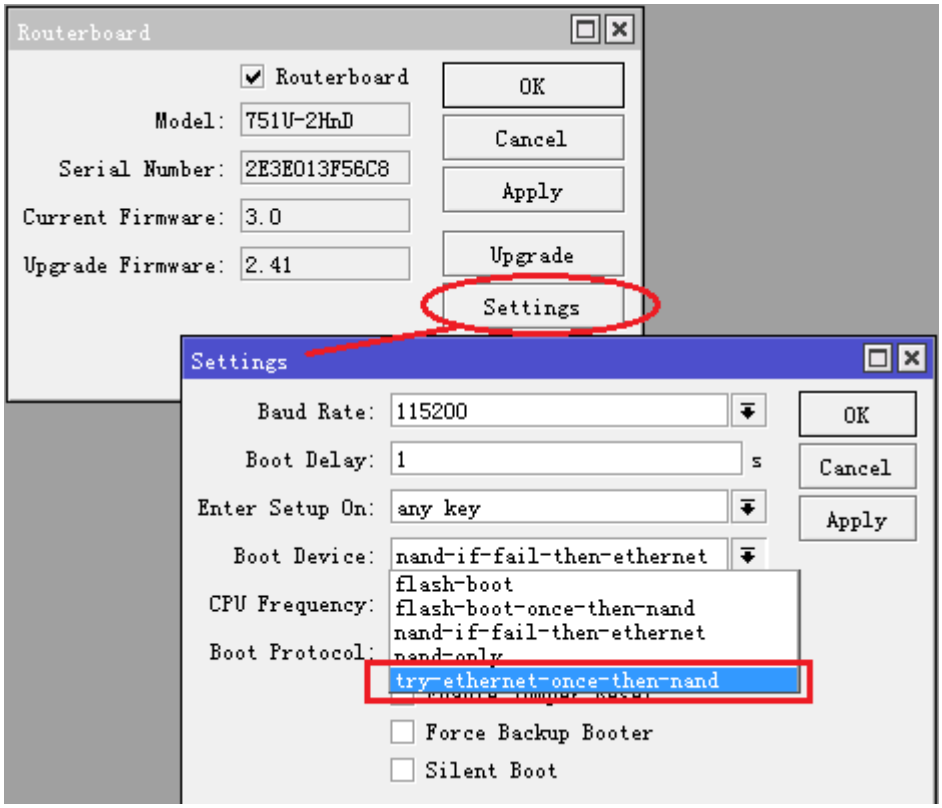
首先需准备 netinstall 软件和 RouterOS 安装包，可以从 https://www.mikrotik.com/download 下载，在下载页面选择你需要的 RouterOS 对应硬件型号和软件版本，我这里演示用的是 RB751U，所以选择的是 mispbe，并且在这个版本中有多个选择，如果用于 netinstall，选择 Main package。

	6.42.9 (Long-term)	6.43.2 (Stable)	6.44beta20 (Testing)
MIPSBE	CRS1xx, CRS2xx, DISC, hAP, hAP ac, hAP ac lite, LDF, LHG, mANTBox, mAP, NetBox, NetMetal, PowerBox, QRT, RB9xx, cAP, hEX Lite, RB4xx, wAP, BaseBox, DynaDish, RB2011, SXT, OmniTik, Groove, Metal, Sextant, RB7xx		
Main package			
Extra packages			
SMIPS	hAP mini, hAP lite		
Main package			
Extra packages			
TILE	CCR		
Main package			
Extra packages			
The Dude server			

准备好工具和软件后，现在需要计算机和 RB 设备通过网线连接，将网线接在 MikroTik 设备的 Ether1 上，官方资料是电脑直连 ether1 口，但我强力建议中间过一台交换机，因为 MikroTik 设备启动时会自检导致电脑网卡中断后，重新自适应导致 boot 引导协商失败。



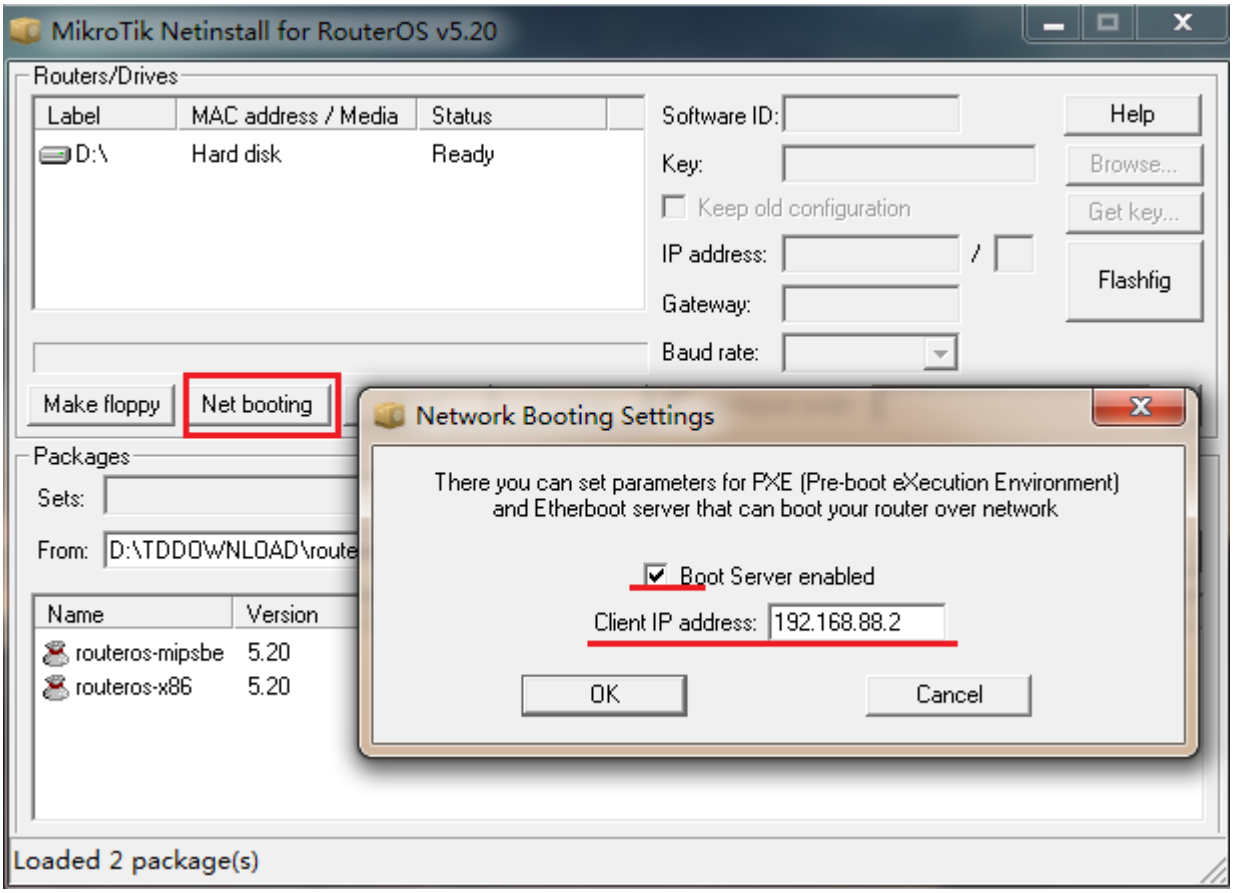
然后，可以选择通过 reset 按钮，长按 15 秒引导 netinstall，如果还能进入系统，也可以通过 winbox 选择 system->routerboard->settings 修改重启后的引导方式。如下图：



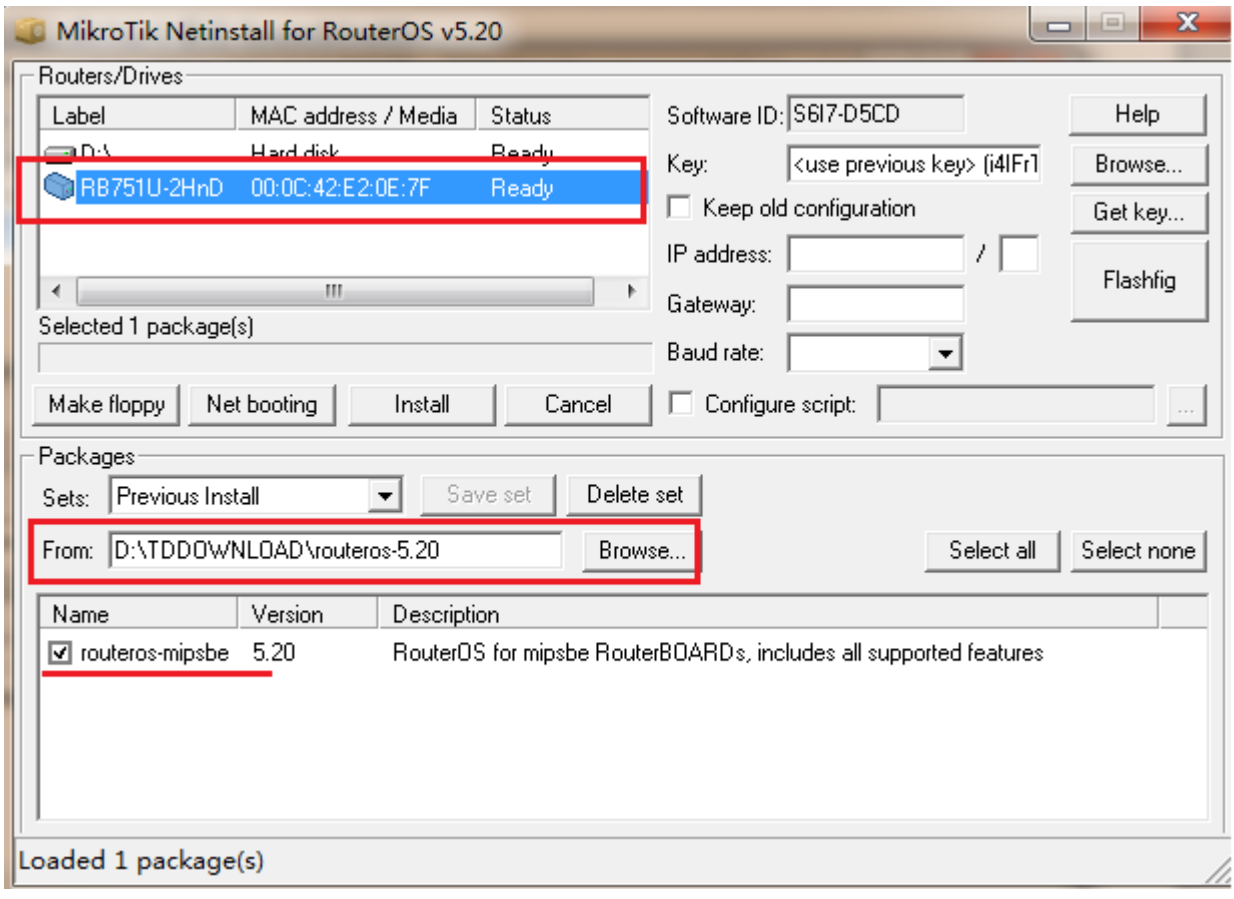
这里我们选择 try-ethernet-once-then-nand，即通过网络引导一次，如果失败则使用 nand 引导。如果无法进入设备配置，长按 RouterBOARD 的复位键（RB 复位键说明）。

以上配置完成后，启动 netinstall，注意如果你是 windows vista 或 7、8、10 系统，第一次启动这个软件，需要允许它网络连接，建议关闭掉其他在运行的软件，避免影响 netinstall 软件安装。

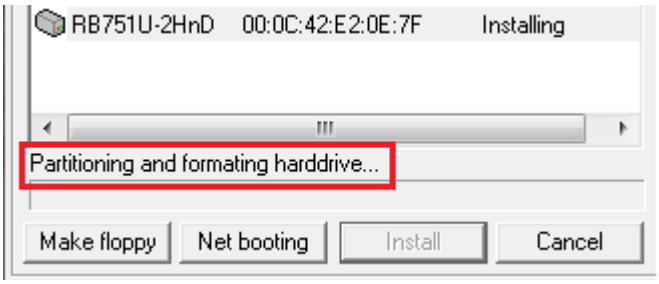
启动软件后，首先配置 Net booting，选择 Boot Server Enabled，设置网络连接 ip 地址，这个地址一定要和你计算机的网卡在同一网络段中，如我的计算机是 192.168.88.3/24，配置给 Netinstall 分配给 RB 的地址是 192.168.88.2



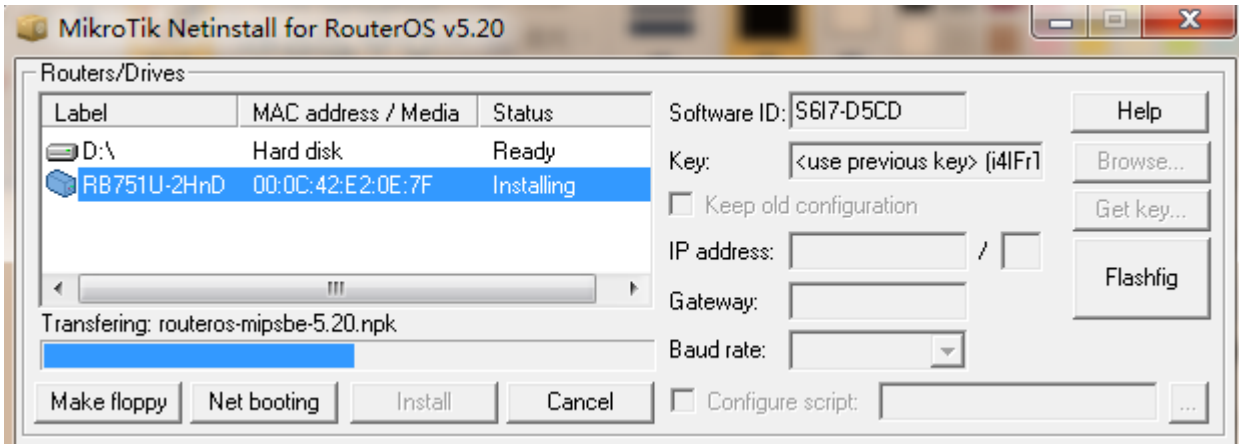
设置好以上参数，就可以通过命令重启 RouterBOARD，重启后 RouterBOARD 会自动搜索以太网引导服务器，找到后会在 netinstall 中显示设备型号和 mac 地址，显示 Ready 状态，你指定好安装包路径后，会自动选择设备的安装包，如下图



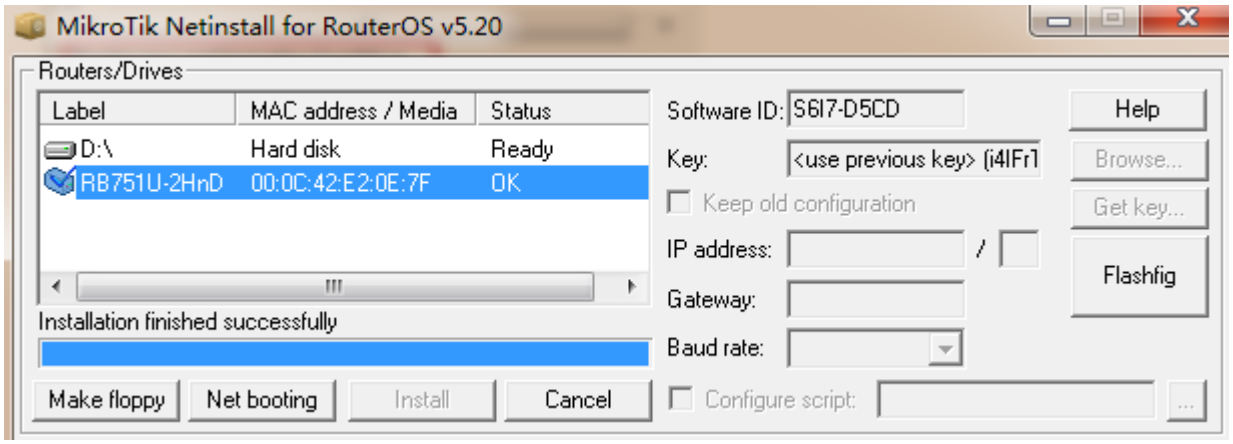
之后，我们点击 install 按钮，首先 netinstall 会对 RouterBOARD 的存储进行分区和格式化



之后传输安装包，进行安装：



安装完成后显示 installation finished successfully



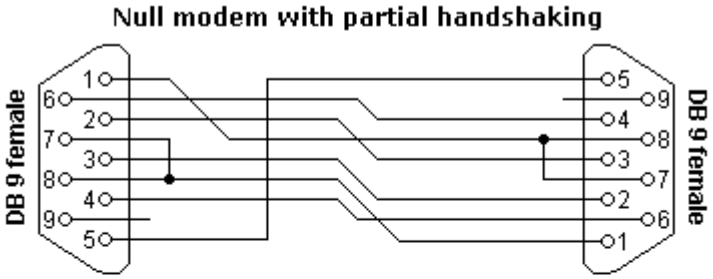
退出 netinstall，RouterBOARD 设备重启，这样系统安装/复位工作就完成。

2、RouterBOARD Console 的操作

以上操作是介绍了 RouterBOARD 的复位和 netinstall 安装等，下面随便介绍下 RouterBOARD 通过 console 口进入路由器进行 Netinstall 的操作，这样让大家了解如何进入 RouterBOARD 的 BIOS 操作。

串口连接线(Console 连接线)

部分 RouterBOARD 带有 Console 接口，MikroTik 对串口有特殊的定义，包括 PC 的 com 接口，下面是串口连接线的线序：

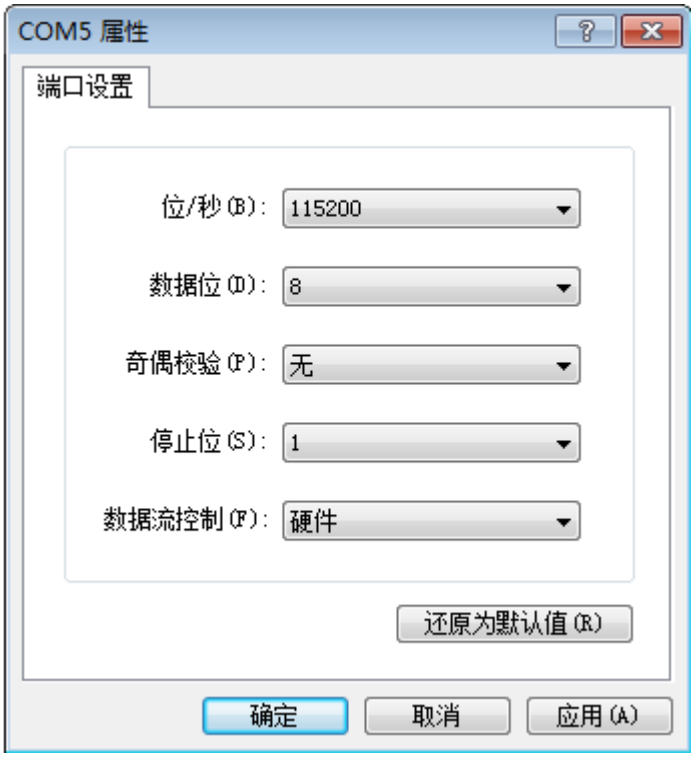


Connector 1	Connector 2	Function
1	7 + 8	RTS_2 → CTS_2 + CD_1
2	3	Rx ← Tx
3	2	Tx → Rx
4	6	DTR → DSR
5	5	Signal ground
6	4	DSR ← DTR
7 + 8	1	RTS_1 → CTS_1 + CD_2

在路由器启动完成后，会发出“嘀嘀”的两声（部分 RouterBOARD 带有蜂鸣器），之后在显示屏上，出现登录的提示，如果在终端显示中，没有提示任何信息，需要检查一下网线或是串口线是否连接好。

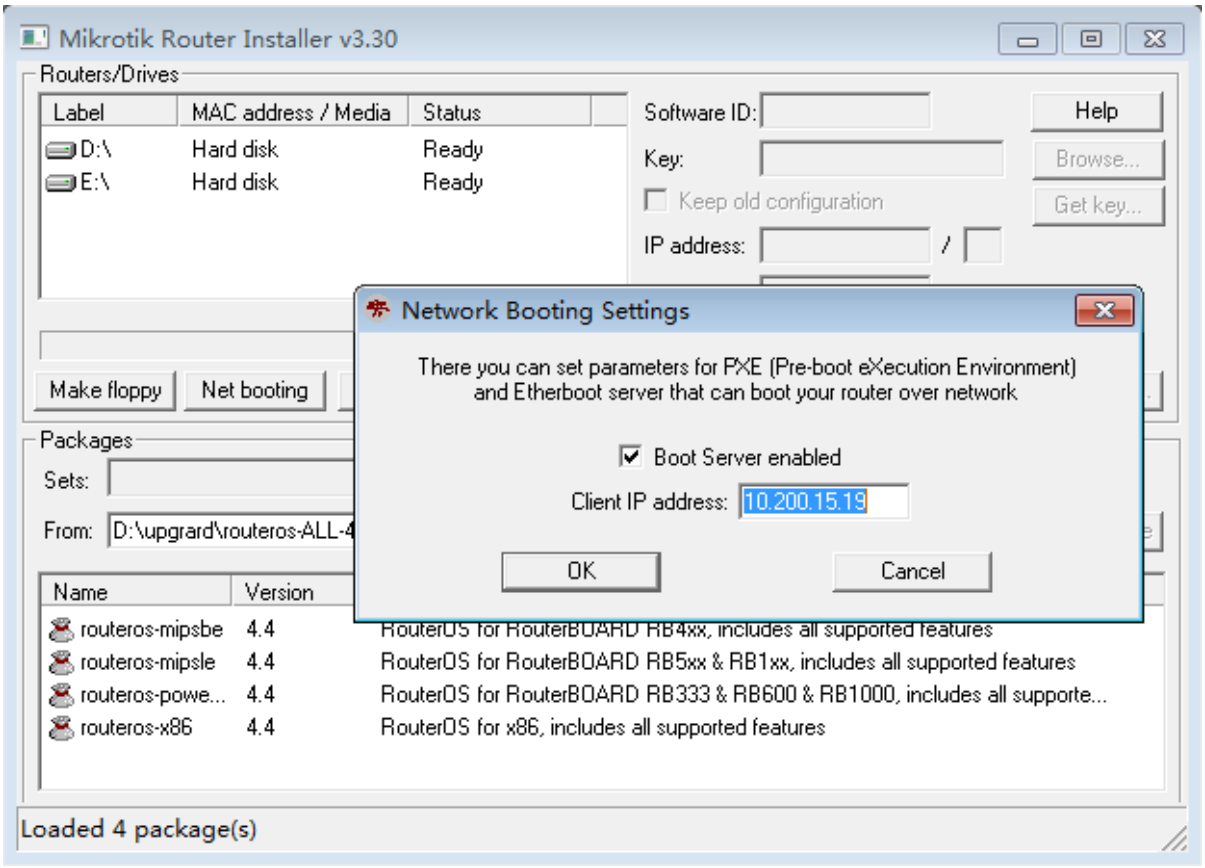
串口控制（管理端）功能允许通过一个串行接口访问路由器的串口终端控台一个特殊的串行接口线通过工作站或者便携式计算机的串口（COM）连接到路由器的串口。在 windows 计算机上常用的串口连接程序是超级终端（HyperTerminal）。

当通过 Console 接口连接时，需准备 Console 线和相关软件，RouterBOARD 串口每秒速率为 115200，x86 平台串口连接每秒速率为 9600，其他参数为系统默认值。如果你的 vista 或者 WIN 7 操作系统，可以直接从 windows xp 里将超级终端程序拷贝到 vista 和 win7 系统上，文件分别是 hypertrm.dll 和 hypertrm.exe。也可以下载 SecureCRT 软件进行操作，下面是 windows 自带的超级终端配置：



1. 启动 Netinstall 程序，打开 Net Booting 按钮，在启用 Boot Server 功能，由 Netinstall 做网络安装引导服务端。如果安装 Netinstall 本机的 IP 地址是 **10.200.15.18/24**，那需要给 Boot Server 客户端的分配一个 IP 地址段，用于临时分配给 RouterBoard 的 IP 地址，本事例的地址为 **10.200.15.19**）。

注： 网线连接的是 RouterBoard 的 ether1 网卡接口，不然无法获取引导信息。



2. 设置 RouterBoard 从以太网卡引导，首先进入 RouterBoard BIOS (重起 RouterBOARD 后，在超级终端下出现提示时 press any key… 后按任意键进入 BIOS 设置):

```
RouterBoard 450G

CPU frequency: 680 MHz
  Memory size: 256 MB

Press any key within 2 seconds to enter setup

RouterBOOT-2.20
What do you want to configure?
   d - boot delay
   k - boot key
   s - serial console
   o - boot device
   u - cpu mode
   f - cpu frequency
   r - reset booter configuration
   e - format nand
   g - upgrade firmware
   i - board info
   p - boot protocol
   t - do memory testing
   x - exit setup
your choice:
```

进入 BIOS 后你可以看到可用命令的清单，设置引导设备，选择“boot device”，按“o”键可以进入

```
your choice: o - boot device

Select boot device:
   e - boot over Ethernet
 * n - boot from NAND, if fail then Ethernet
   1 - boot Ethernet once, then NAND
   o - boot from NAND only
   b - boot chosen device
your choice:
```

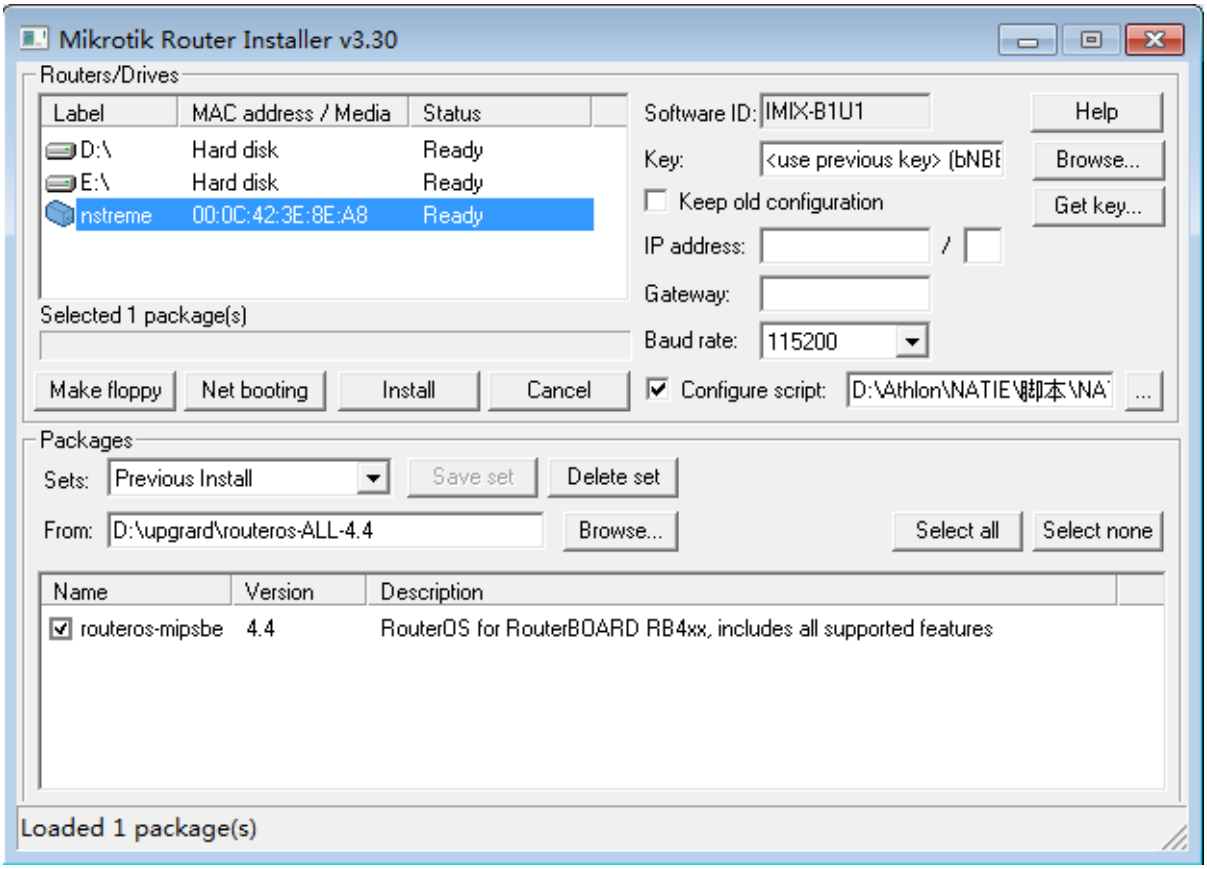
按“e”键，是选择从以太网卡引导 RouterBoard：

```
Select boot device:
   e - boot over Ethernet
 * n - boot from NAND, if fail then Ethernet
   1 - boot Ethernet once, then NAND
   o - boot from NAND only
   b - boot chosen device
your choice: e - boot over Ethernet
```

当选择完成后，返回 RouterBoard BIOS 首页，选择“x“，退出 BIOS。路由器将会重启。

3. 在启动时，RouterBoard 将试着从以太网卡上去寻找引导信息。如果成功，运行 Netinstall 的 Windows 工作站，将会分配给 RouterBoard 一个 IP 地址。在上面过程完成后，RouterBoard 将等待安装信息。

   在 Windows 上，将会出现一个新的设备列表，显示当前连接的 RouterBoard 设备。

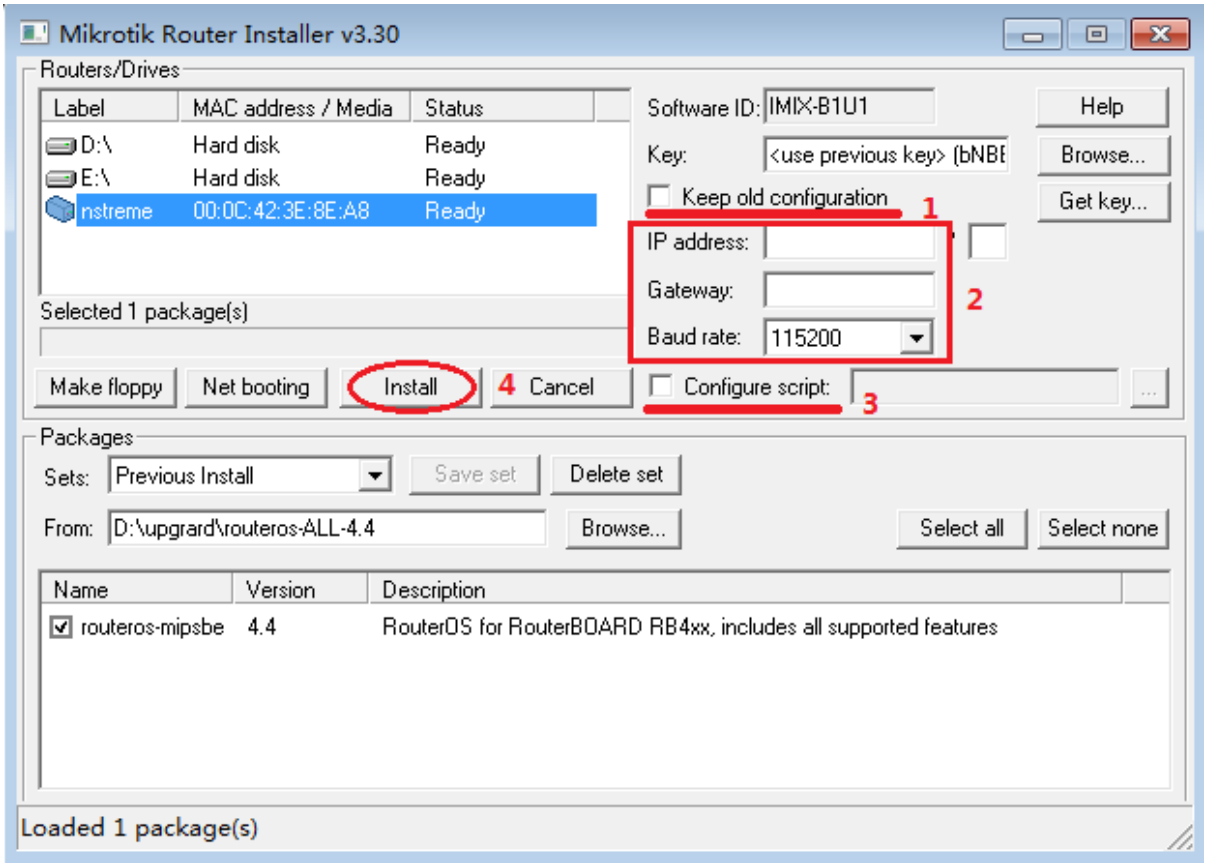


这时 Netinstall 会自动识别 RouterBOARD 的型号，并在指定的目录下查找对应的功能包，Netinstall 自动为 RB450G 查找到了合适的 RB4xx 的安装包，你也可以手动指定安装功能包路径。

在超级终端显示如下等待安装信息

```
Welcome to MikroTik Router Software remote installation
Press Ctrl-Alt-Delete to abort

mac-address: 00:0C:42:3E:8E:A8
mac-address: 00:0C:42:3E:8E:A9
mac-address: 00:0C:42:3E:8E:AA
mac-address: 00:0C:42:3E:8E:AB
mac-address: 00:0C:42:3E:8E:AC

software-id: IMIX-B1U1 key:
bNBBSe/onQwGhhk/RW1XBfWTVeOnnja/UsnbuTgcDVckt7fl5zf0Iobz03GWXjCr6vUQ34XSfB9pdGmX
czOmEA==

Waiting for installation server...
```

根据自己的需要配置的默认置参数：



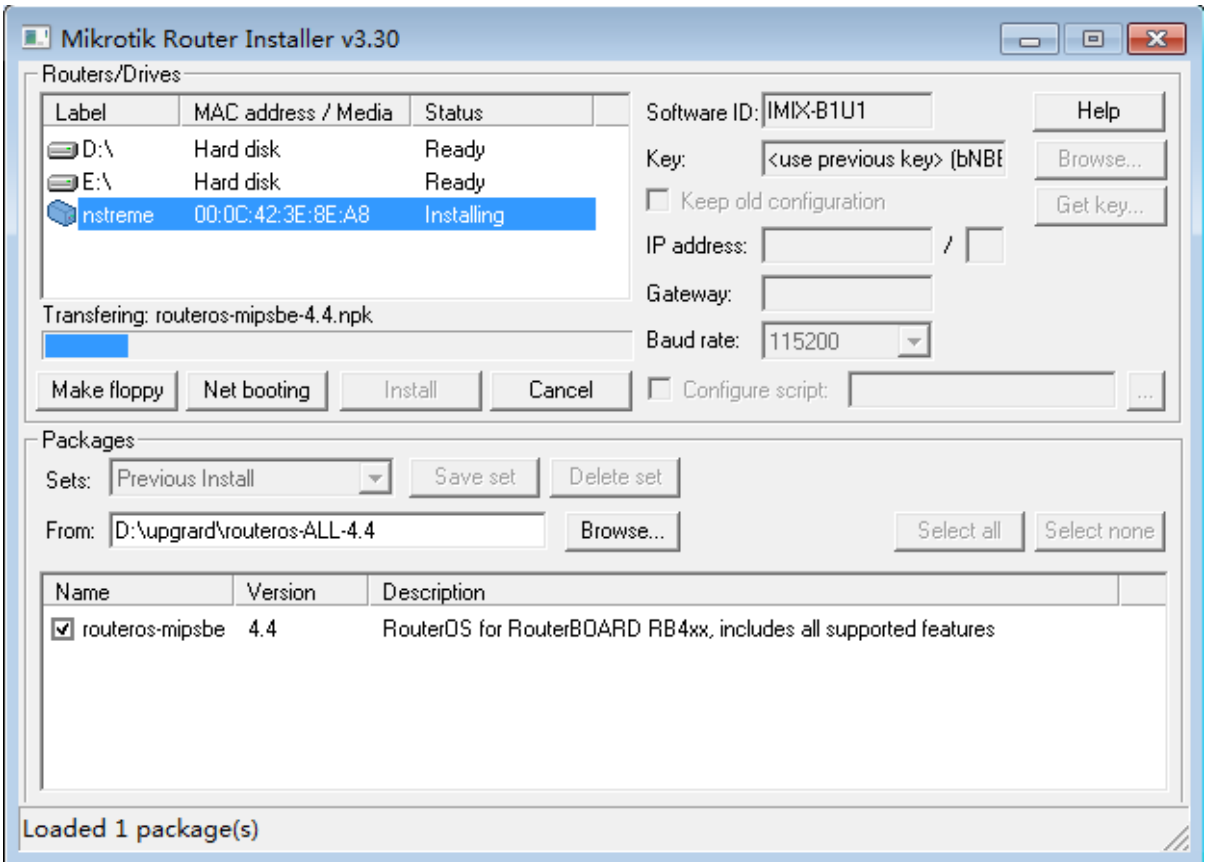
1、Keep old configuration　保留原来的配置不变
2、配置 ip 地址和网关，并设置传输速率采用 115200
3、配置 RouterBOARD 的脚本
4、开始安装

安装过程在超级终端显示的安装进度

```
Welcome to MikroTik Router Software remote installation
Press Ctrl-Alt-Delete to abort

mac-address: 00:0C:42:3E:8E:A8
mac-address: 00:0C:42:3E:8E:A9
mac-address: 00:0C:42:3E:8E:AA
mac-address: 00:0C:42:3E:8E:AB
mac-address: 00:0C:42:3E:8E:AC

software-id: IMIX-B1U1 key:
bNBBSe/onQwGhhk/RW1XBfWTVeOnnja/UsnbuTgcDVckt7fl5zf0Iobz03GWXjCr6vUQ34XSfB9pdGmX
czOmEA==

Waiting for installation server...
Found server at 00:1E:EC:B0:B2:17

Formatting disk.....
```

```
installing routeros-mipsbe-4.4 [############                                ]
```

Netinstall 显示的安装情况



4. 当安装工作完成，在安装程序中 Reboot 键或在超级终端里敲击“回车”，路由器将重启。

这里需要注意：记住设置完后回到 RouterBoard BIOS 中设置为 boot from NAND only（仅从 RouterBoard 的闪存引导）。 这样完成后，就能正常启动 RouterOS。

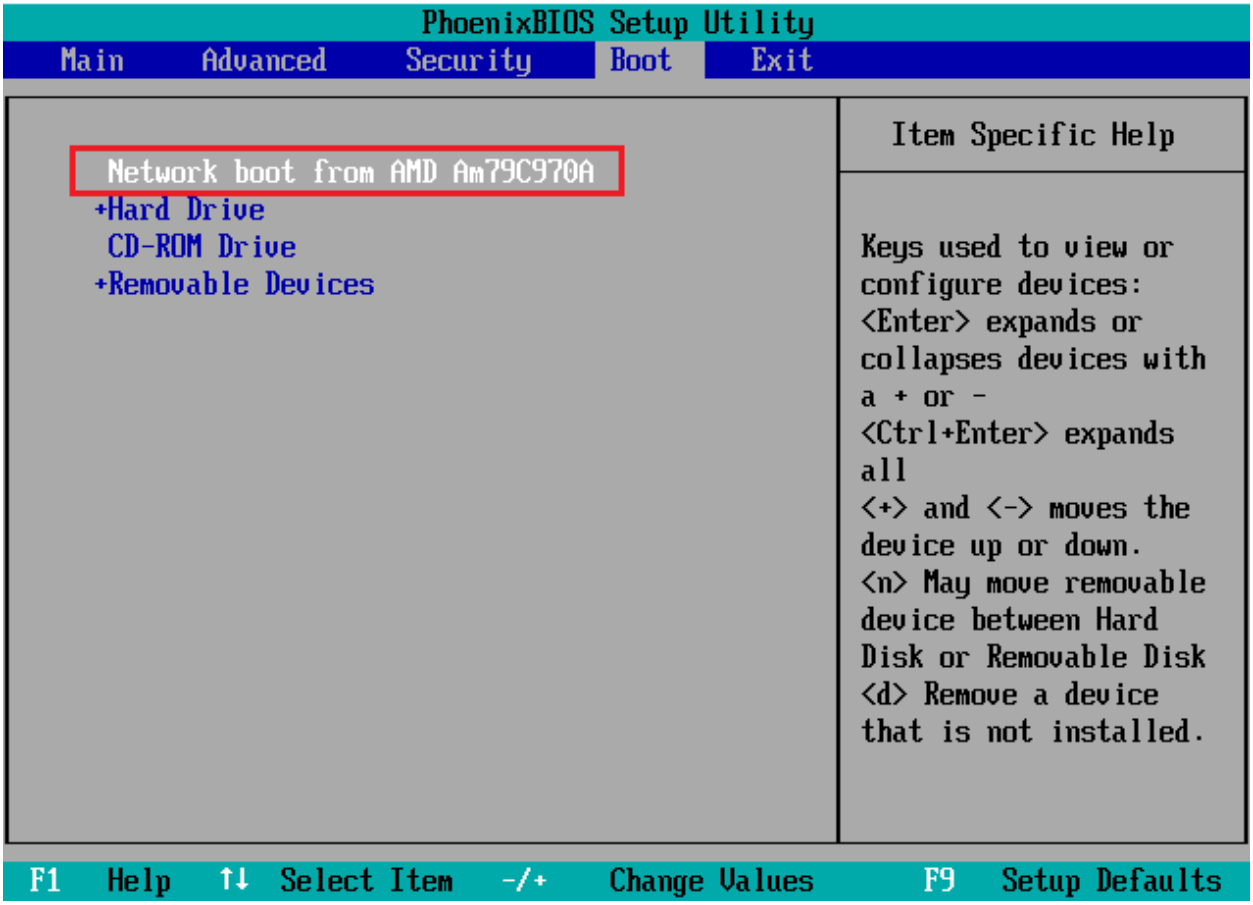
```
Select boot device:
 * e - boot over Ethernet
   n - boot from NAND, if fail then Ethernet
   1 - boot Ethernet once, then NAND
   o - boot from NAND only
   b - boot chosen device
your choice: n - boot from NAND, if fail then Ethernet
```

在路由器启动完成后，会发出“嘀嘀”的两声，之后在显示屏上，出现登录的提示，如果在终端显示中，没有提示任何信息，表示安装正常。

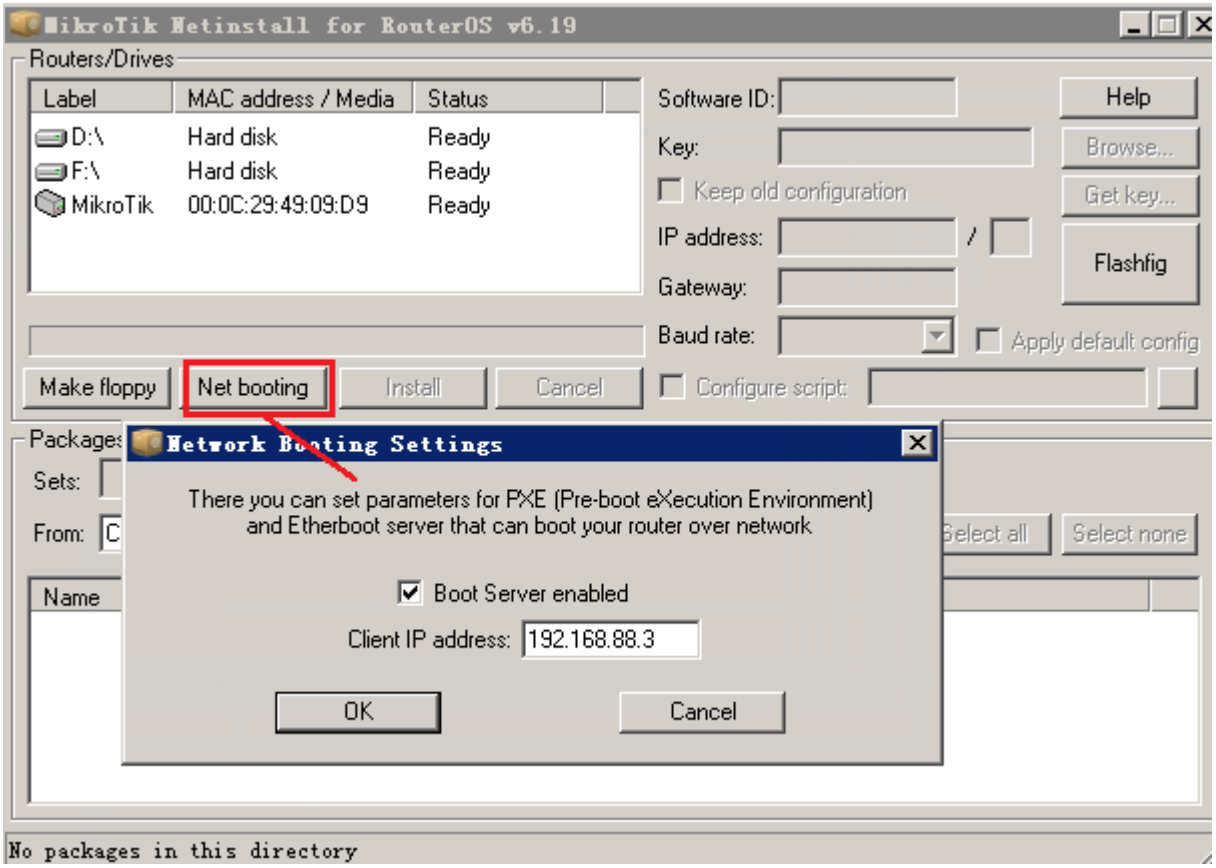
3、PXE 安装 RouterOS 到 x86

在 x86 平台除了能通过光盘安装外，还能通过 PXE 网络引导安装，首先需要网卡支持网络引导，大多数网卡都支持网络引导，安装方式类似于 RouterBAORD 的 netinstall 安装，注意 x86 平台对硬盘连接有要求，特别是服务器硬盘不能连接阵列卡，需要与板载 SATA 接口直连，下面介绍安装方法：

1、进入 x86 主板的 BIOS 设置网卡引导（不同 x86 主板 BIOS 接口不同），以下的 BIOS 界面仅供参考。



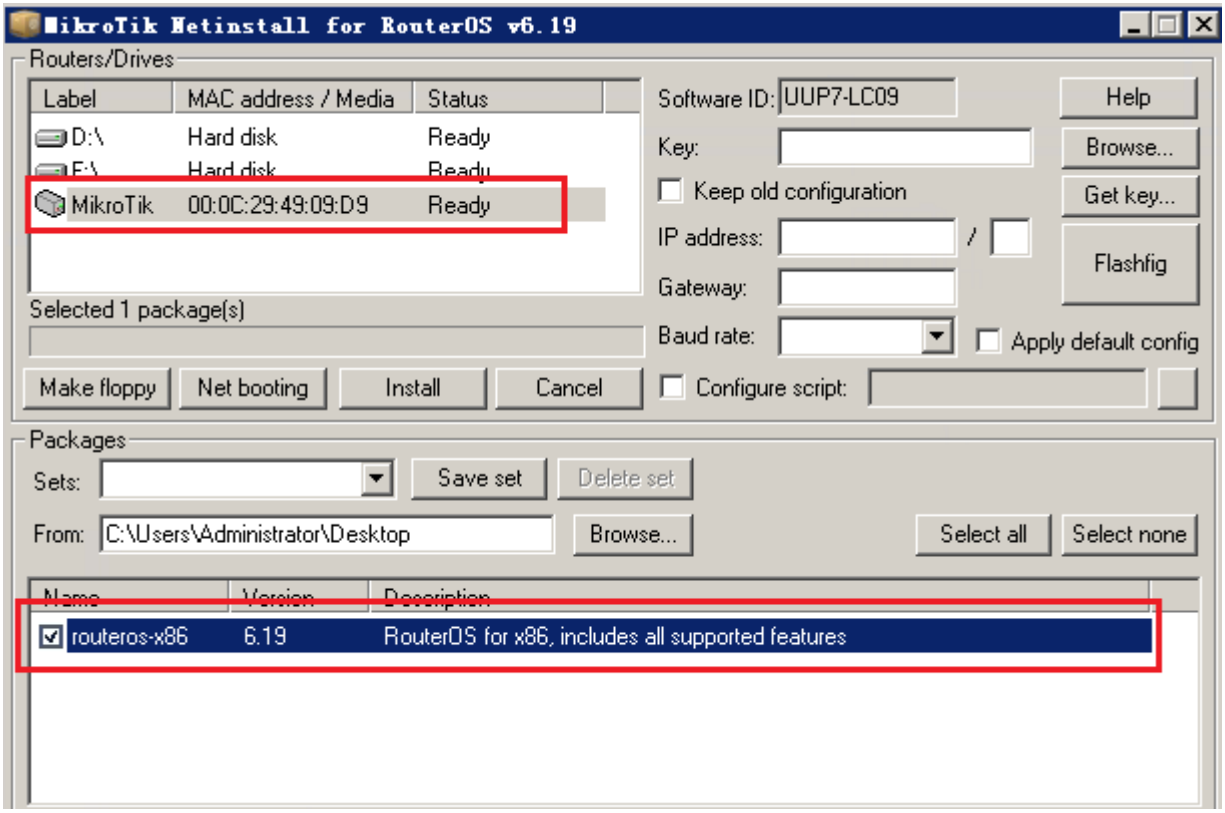
将安装 windows 的电脑与安装 RouterOS 的 x86 主机的网卡连接，并在 windows 系统上准备 netinstall 软件，打开后将 netbooting 设置与本地 ip 地址同一段，windows 计算机的 ip 是 192.168.88.2，分配给被安装主机 ip 是 192.168.88.3。准备好 x86 的安装包 routeros-x86-6.19.npk（可以从 download.mikrotik.com 下载，注意下载包是 Main Package），下载后放在与 netinstall 相同目录。



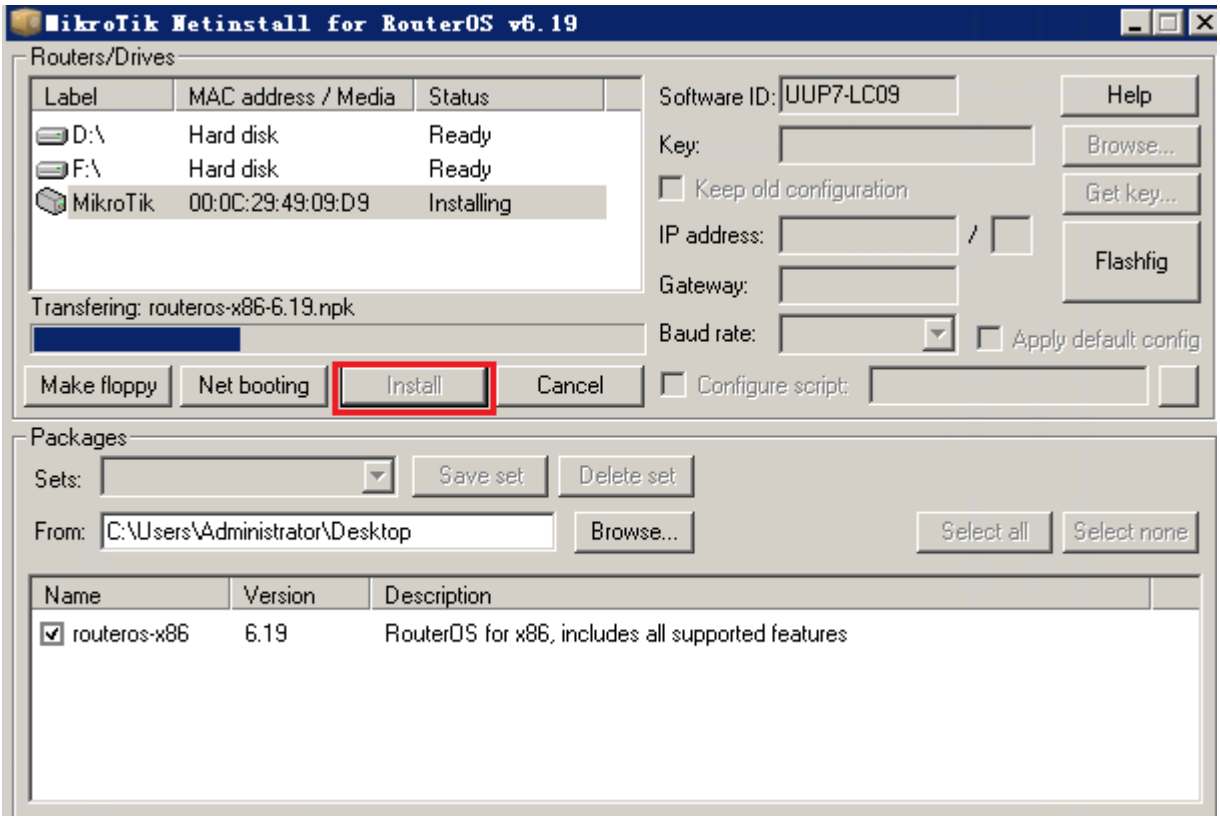
BIOS 和 netinstall 都设置好后，启动主机从网卡的 PXE 引导，如果引导成功进入下面的接口：

```
Welcome to MikroTik Router Software remote installation
Press Ctrl-Alt-Delete to abort

mac-address: 00:0C:29:49:09:D9

Waiting for drivers...
Retrieving drivers...
Loading drivers

Looking for harddrives...

Found harddrive as IDE Primary master (disk C)
disk label: MikroTik
software-id: UUP7-LC09

Waiting for installation server...
```

在 netinstall 出现安装主机的 MAC 地址，并显示 status 为 Ready，等待安装指令，安装包会自动在相同目录下搜索，找到匹配的安装包后，会显示如下：



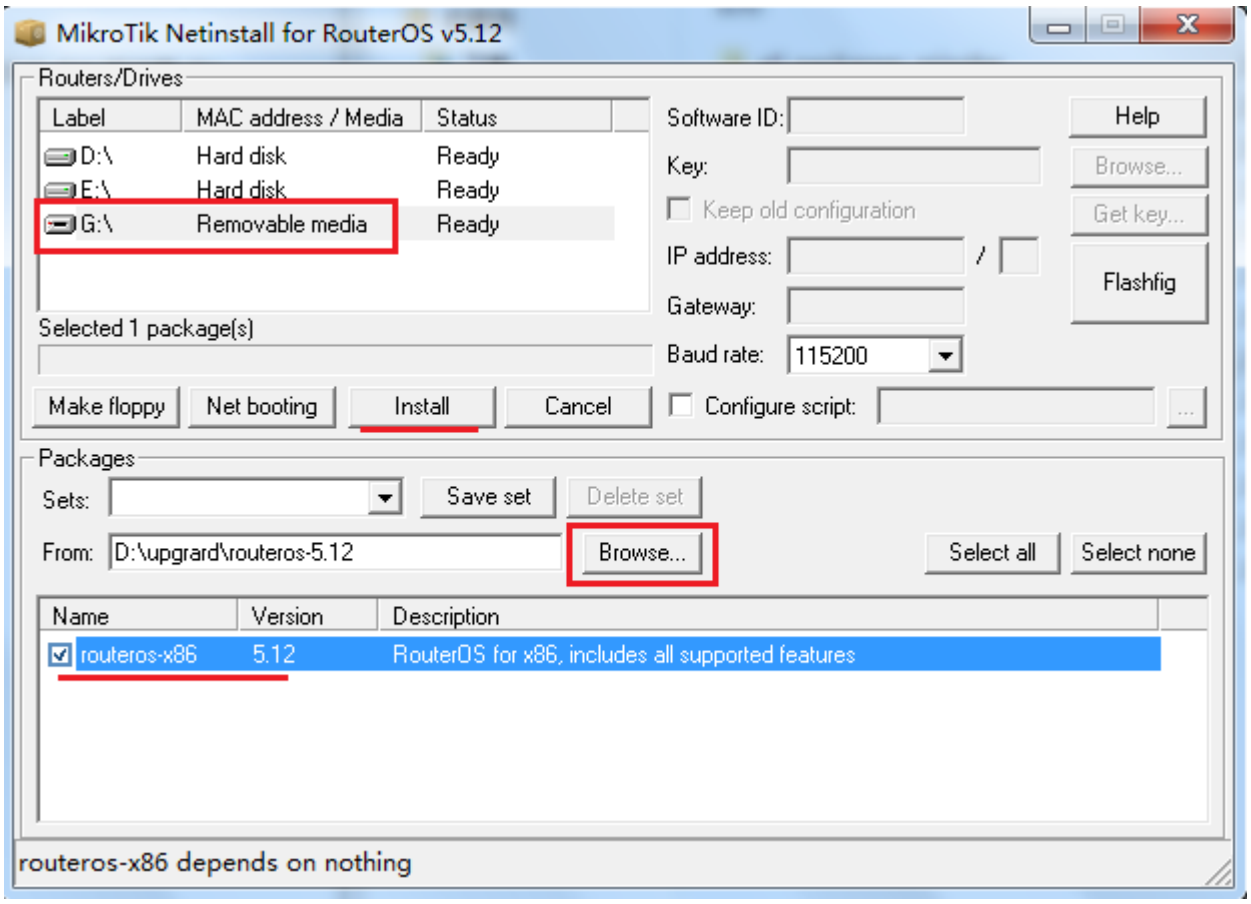
点击 install 安装，之后自动安装 RouterOS 到 x86 设备上



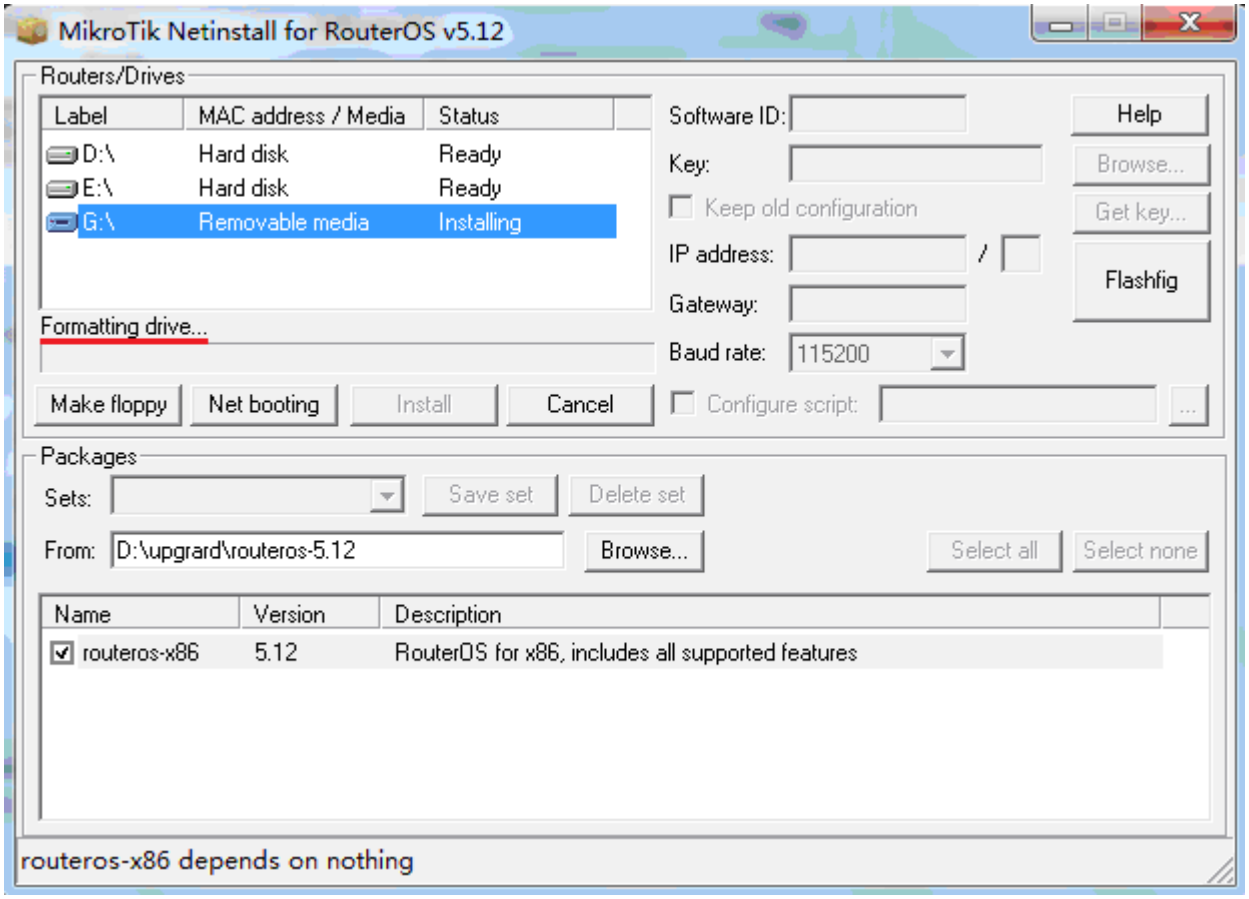
4、U 盘安装

U 盘安装，是将 RouterOS 系统写入到 U 盘中，整个 RouterOS 系统在 U 盘运行，即 U 盘作为系统盘使用，而并非 U 盘去安装 RouterOS 到 x86 平台的系统盘中。

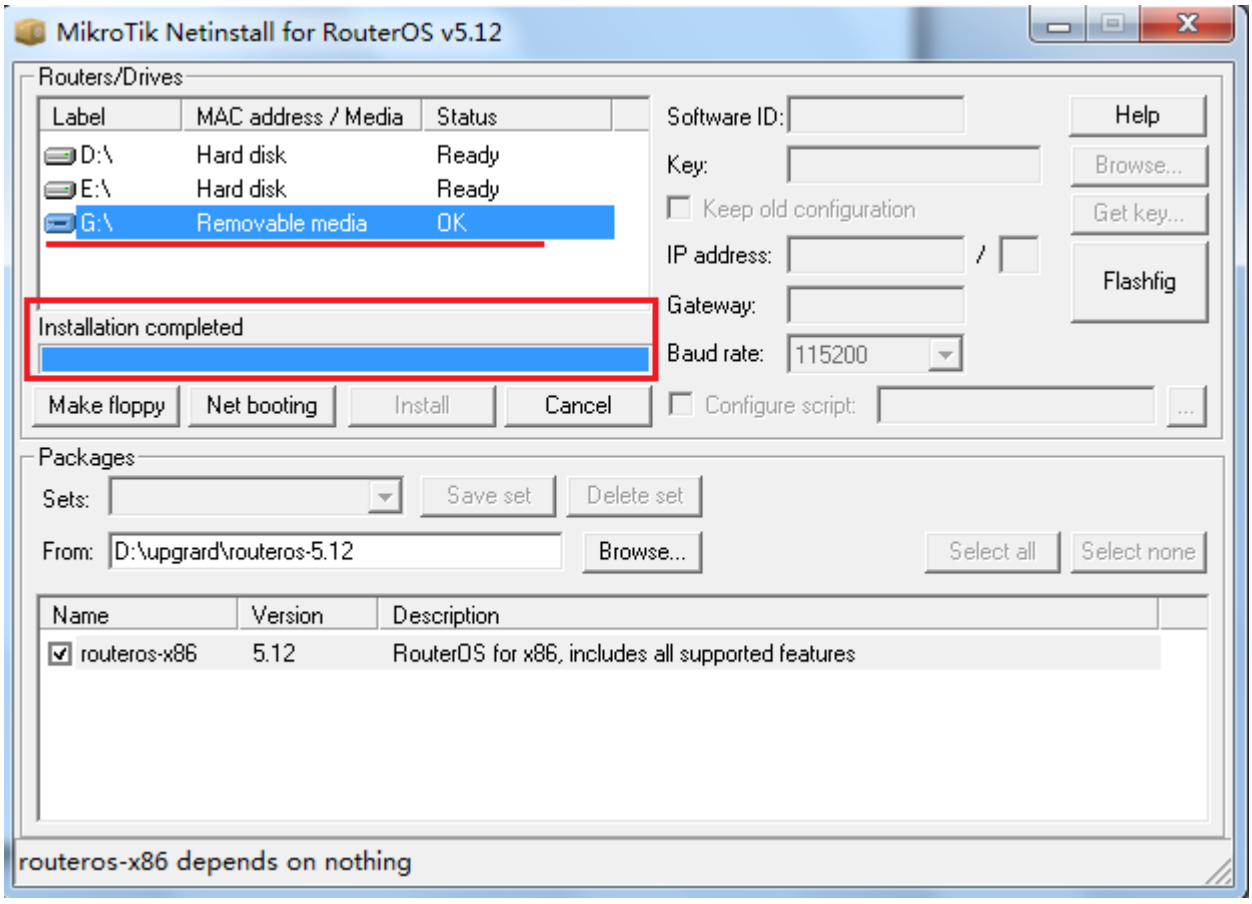
1、首先准备好 netinstall 软件， U 盘安装需要使用 RouterOS v3.0，将 U 盘插入一台 Windows 计算机的 USB 接口后，选择 routeros-x86 安装包启动 netinstall 软件，并找到 Removable media 设备，即移动存储的 U 盘。选择 Browse 选择安装包的路径，找到路径后，软件会自动筛选相应的功能包，U 盘使用的是 routeros-x86 功能包，然后点击 install 安装系统。



点击 install 安装后，会出现 formatting driver 的提示，表示正在格式化 U 盘：



3、之后会 copy 文件到 U 盘上，最后显示 Installation completed 的提示，即安装完成：



最后取出 U 盘，将 U 盘插入到 PC 上，并设置计算机的 BIOS 通过 USB 引导启动，启动后可以看到系统正在安装。

5、自定义 RouterOS 默认启动预配置脚本

基于 RouterBOARD、CCR 和 CRS 的 RouterOS 硬件设备，允许用户自定义复位脚本，即复位恢复出厂设置不使用官方配置，使用自定义的配置。

官方默认配置脚本，可以通过/system default-configuration print 查看：

```
[admin@MikroTik] > /system default-configuration print
            script: :global ssid;
                    #| WISP Bridge:
                    #|   * wireless and LAN interfaces are bridged;
                    #| wlan1 Configuration:
                    #|       mode:            ap-bridge;
                    #|       band:            2ghz-b/g/n;
                    #|       tx-chains:       0;1;
                    #|       rx-chains:       0;1;
                    #|       ht-extension:    20/40mhz-Ce;
                    #| LAN Configuration:
                    #|       DHCP Client: enabled on bridge (LAN port);

                    :log info Starting_defconf_script_;
                    #-----------------------------------------------------------------------------
                    # Apply configuration.
                    # these commands are executed after installation or configuration reset
                    #-----------------------------------------------------------------------------
                    :if ($action = "apply") do={
                    # wait for interfaces
                    :local count 0;
                    :while ([/interface ethernet find] = "") do={
                    :if ($count = 30) do={
                    :log warning "DefConf: Unable to find ethernet interfaces";
                    /quit;
                    }
                    :delay 1s; :set count ($count +1);
                    };

                      :local count 0;
                      :while ([/interface wireless print count-only] < 1) do={
                        :set count ($count +1);
                        :if ($count = 40) do={
                          :log warning "DefConf: Unable to find wireless interface(s)";
                          /ip address add address=192.168.88.1/24 interface=ether1
comment="defconf";
                          /quit
                        }
```

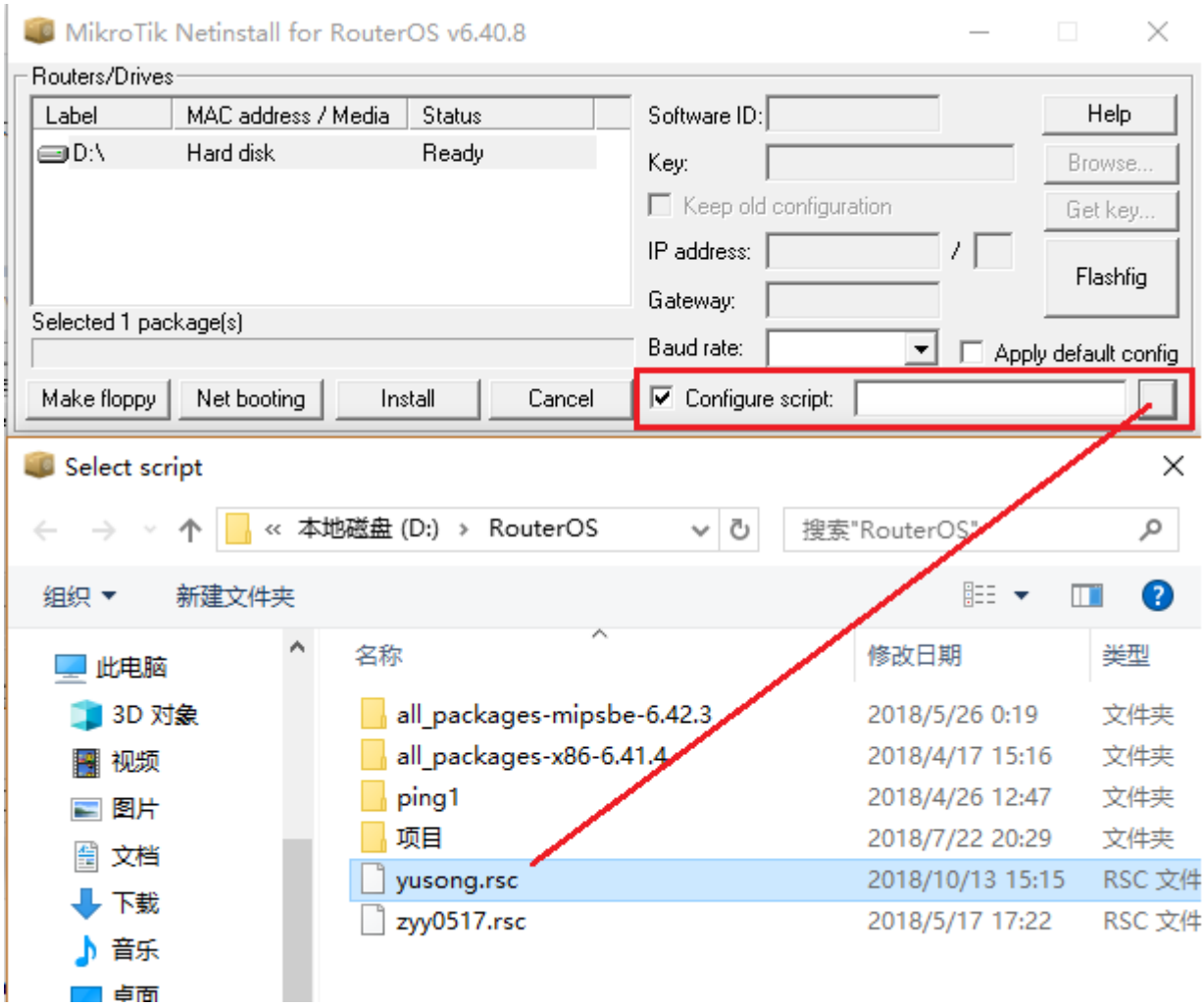
```
                        :delay 1s;
                      };
                      /interface wireless {
                        set wlan1 mode=ap-bridge band=2ghz-b/g/n disabled=no
wireless-protocol=802.11 \
                            distance=indoors
之后省略…
```

你可以参考官方的脚本，导出进行修改，也可以自己定义配置，按照 RouterOS 脚本要求编辑，写入自己的期望复位后的预配置，特别是对 OEM 商非常有用，下面是一个简单的配置脚本，通过文本编辑如下：

```
:delay 5s
/ip address add address=192.168.88.1/24 interface=ether1 comment="default configuration"
/system identity set name=YuSong_Router
/ip service
set telnet disabled=yes
set ftp disabled=yes
set api disabled=yes
set api-ssl disabled=yes
 /tool graphing interface add interface=all store-on-disk=yes
 /tool graphing resource add store-on-disk=yes

/ip cloud
set ddns-enabled=yes

/system clock
set time-zone-name=Asia/Shanghai
/system logging
set 0 action=disk
set 1 action=disk
set 2 action=disk

/ip firewall connection tracking set tcp-established-timeout=5h
```

编辑完成后，修改后缀名为.rsc

通过 netinstall 重新安装 MikroTik 硬件设备的 RouterOS 系统，进入 netinstall 后，按照之前的方法正确引导 MikroTik 连接 netintall，连接成功后，选择 configure script，指定配置文件的路径，然后选择安装包，之后再点击 install 安装



安装完成后，默认配置就会写入到 MikroTik 硬件中，每次复位后都是你指定的配置。

DVD/CD 光盘安装

DVD/CD 安装是基于 PC 的 x86 硬件最常见的安装方式，通过 DVD/CD 将 MikroTik RouterOS 安装到基于 x86 平台的 PC 硬件上

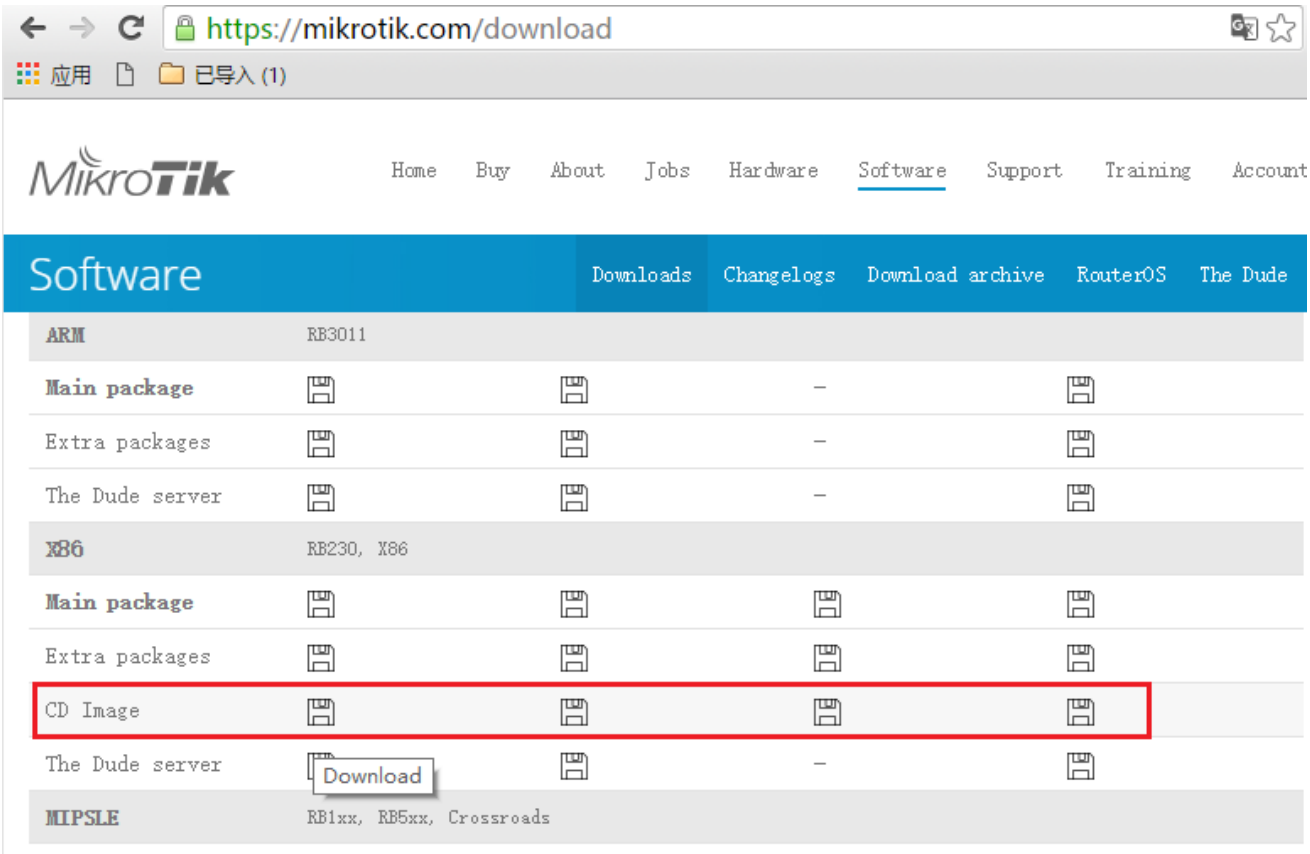
注意：PC 平台的硬盘接口支持 IDE 接口和 SATA 接口，不支持 RAID 卡

CD 安装要求：

- 基于 x86 平台的 PC 硬件
- CD-ROM
- MikroTik RouterOS ISO 镜像文件
- 安装 CD 通过刻录软件刻录

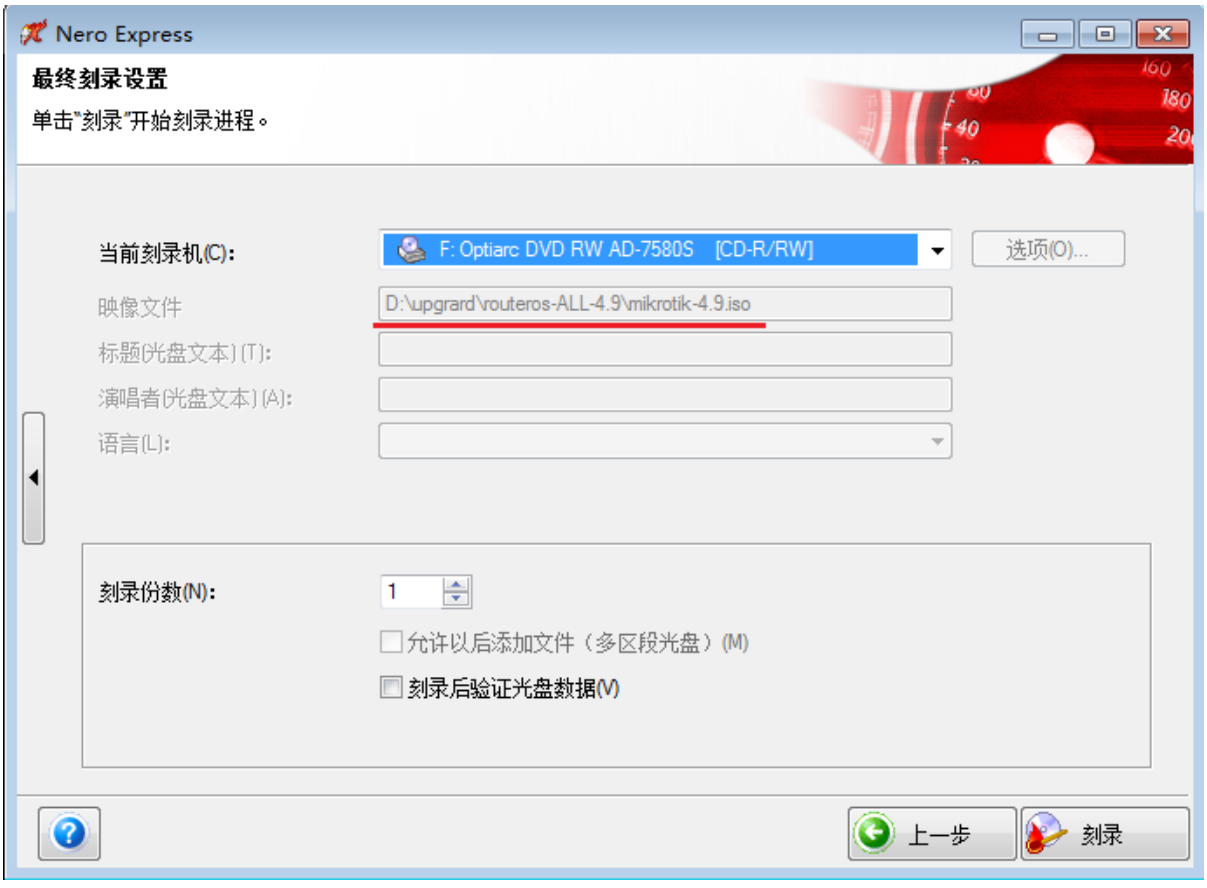
准备 MikroTik RouterOS 安装

1. 下载 MikroTik 的镜像文件[下载页面](),

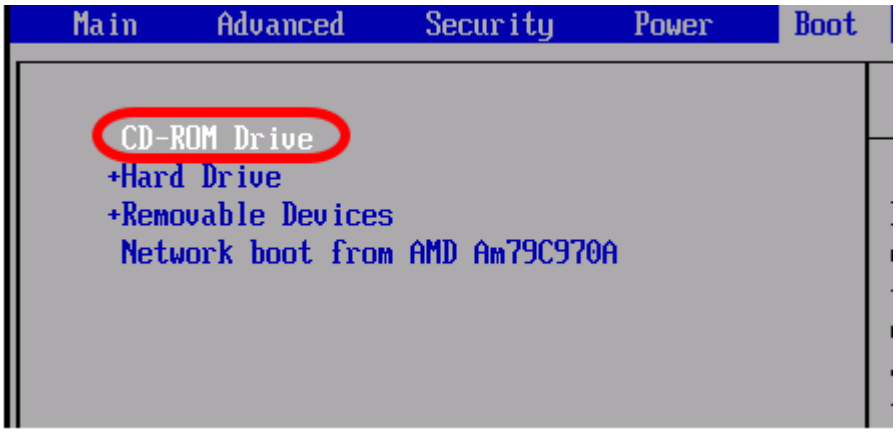


选择 x86 下的 CD Image，根据自己需要选择不同的版本，如 bugfix 版本，稳定版和开发版本等。

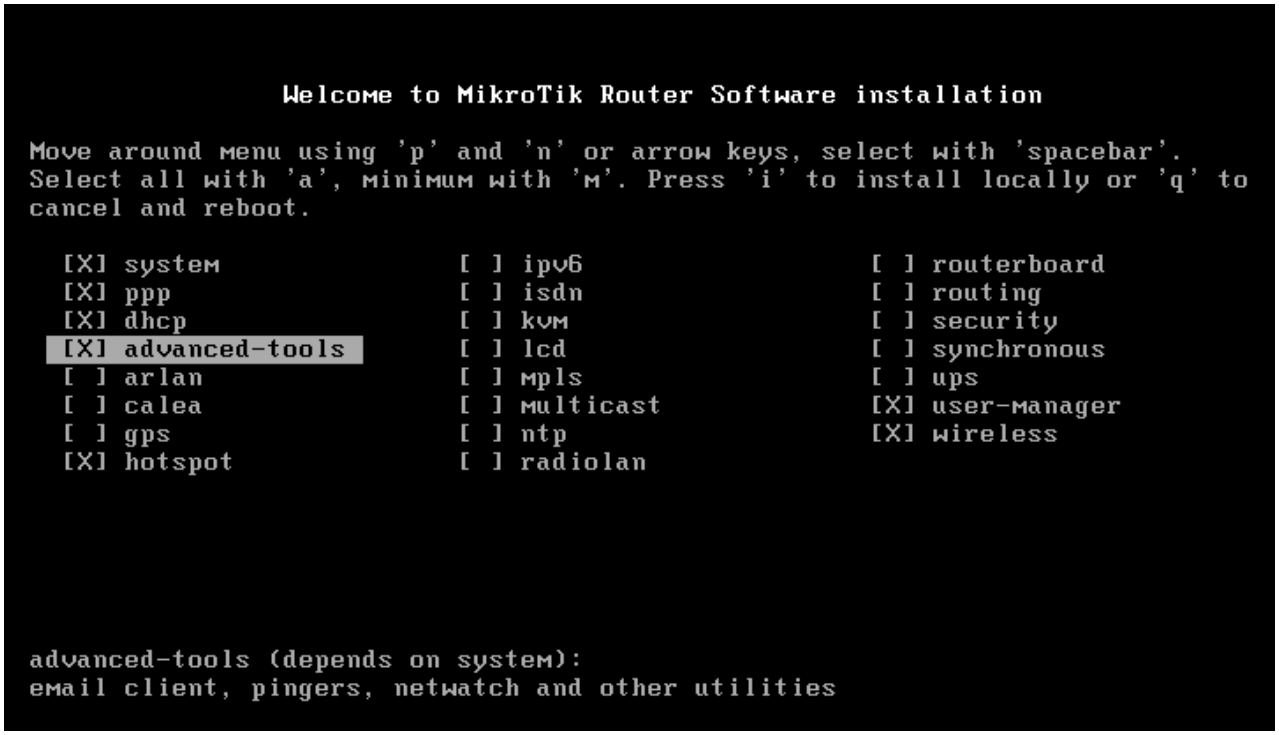
2.把 ISO 镜像下载到硬盘后，通过刻录光驱和刻录程序，将镜像刻录为 CD/DVD



3. 刻录好 CD/DVD 后，将准备好的 PC 装好光驱，并设置 BIOS 为 CD-ROM 引导，然后放入 CD 进行安装：



4. PC 将从 RouterOS CD/DVD 光盘引导启动，安装并选择相应的功能包



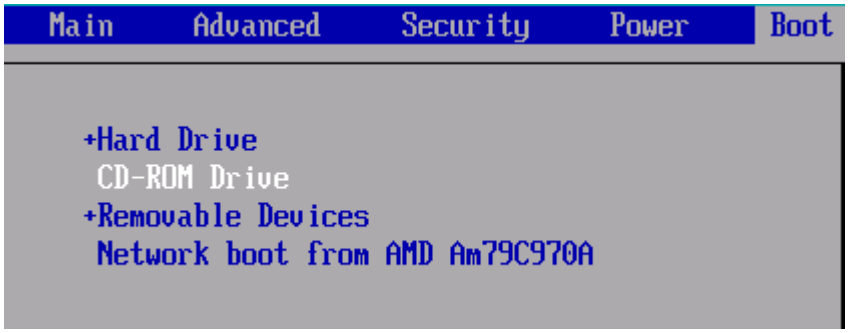
5. 选择你需要安装的功能包，使用“空格”选择功能包，“a”可以选择所有功能包，或者“m”选择最小安装，按“i”开始安装 RouterOS，如果你之前在同一台 PC 上安装过 RouterOS，你可以复位和保存设置，提示如下“**Do you want to keep old configuration?**”如果选择“**n**”复位设置，选择“**y**”保存之前配置

```
cancel and reboot.

  [X] system              [ ] ipv6                [ ] routerboard
  [X] ppp                 [ ] isdn                [ ] routing
  [X] dhcp                [ ] kvm                 [ ] security
  [X] advanced-tools      [ ] lcd                 [ ] synchronous
  [ ] arlan               [ ] mpls                [ ] ups
  [ ] calea               [ ] multicast           [X] user-manager
  [ ] gps                 [ ] ntp                 [X] wireless
  [X] hotspot             [ ] radiolan

advanced-tools (depends on system):
email client, pingers, netwatch and other utilities

Do you want to keep old configuration?  [y/n]:n

Warning: all data on the disk will be erased!

Continue? [y/n]:_
```

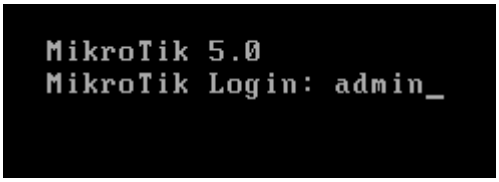
6. 安装完成后，将提示你重启

```
advanced-tools (depends on system):
email client, pingers, netwatch and other utilities


Do you want to keep old configuration?  [y/n]:n

Warning: all data on the disk will be erased!

Continue? [y/n]:y

Creating partition........
Formatting disk...

installed system-5.0
installed wireless-5.0
installed user-manager-5.0
installed hotspot-5.0
installed advanced-tools-5.0
installed dhcp-5.0
installed ppp-5.0
Checking disk integrity...

Software installed.
Press ENTER to reboot
_
```

7. MikroTik RouterOS 安装成功后，不要忘记将 CD-ROM 引导修改为硬盘引导



8. 启动后 RouterOS 提示登录信息，登录预设帐号是 admin，没有密码



10. 第一次安装的 RouterOS 需要注册许可，否则只能使用 24 小时，在下面我们可以看到 **software-id** ，你通过安装后的得到的 software-id 与销商联系购买许可。

1.2 RouterOS 登录方式

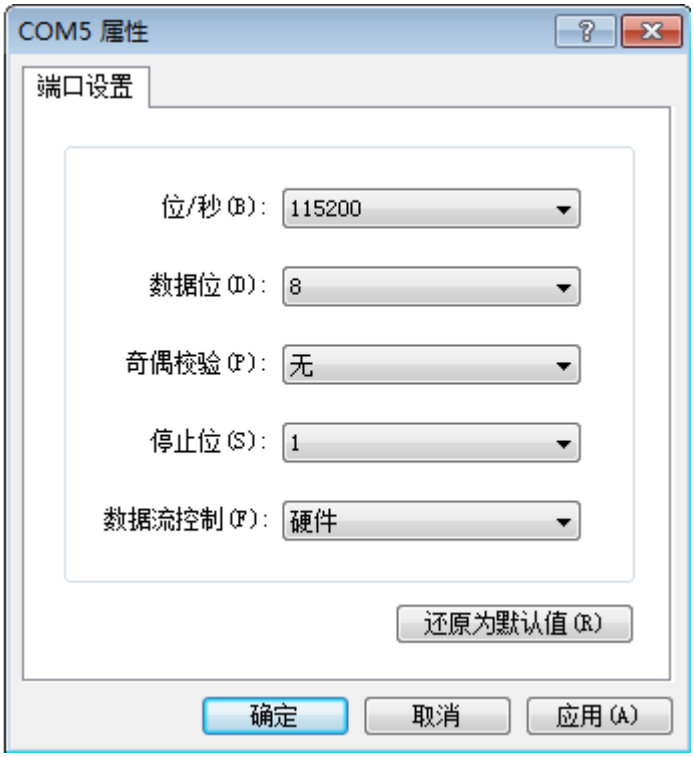
在安装完成 RouterOS 后，准备第一次登陆 RouterOS，RouterOS 初始账号是“**admin**”，密码是“**空**”下面介绍如何通过哪些方式可以连接到 RouterOS。

方式 1 Console 连接

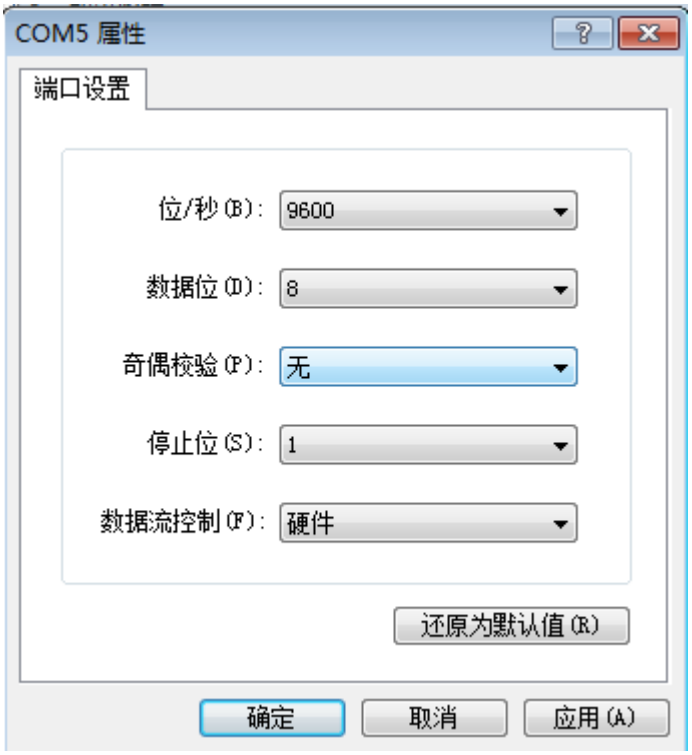
如果你的设备是 RouterBOARD，没有显示器接口连接功能，你必须找到一条 Console 线（普通的 Console 线），或者采用方式 2。

X86 平台通过标准的 DB9 模式串口线连接到路由器，x86 平台串口连接的默认设置为每秒速率：9600 bits/s (**RouterBOARD 系列串口是 115200 bits/s**),使用终端仿真程序（如在 windows 中的超级终端或 SecureCRT，UNIX/Linux 的 minicom）连接到路由器。超级终端的具体参数设置如下：

将 Console 线的一端插到路由设备的 Console 口上，另一端插到 PC 上（运行 windows 或者 linux 操作系统），如果你 PC 没有 Console 口，你可以使用一个 USB-Serial 适配器（USB 转串口适配器），然后运行一个终端程序 HyperTerminal 或者 SecureCRT（windowsXP 以前系统都自带有超级终端，Vista 和 win7 可以将 windowsXP 的超级终端文件拷贝直接使用，文件 hypertrm.dll 和 hypertrm.exe）。RouterBOARD 连接参数如下：



基于 PC 的 RouterOS 连接参数：



通过以上配置你可以连接到 x86 平台的 RouterOS。

串口控制线配置

基于 PC 的 RouterOS 的 DB9 串口线序排列如下：

Router Side (DB9f)	Signal	Direction	Side (DB9f)
1, 6	CD, DSR	IN	4
2	RxD	IN	3
3	TxD	OUT	2
4	DTR	OUT	1, 6
5	GND	-	5
7	RTS	OUT	8
8	CTS	IN	7

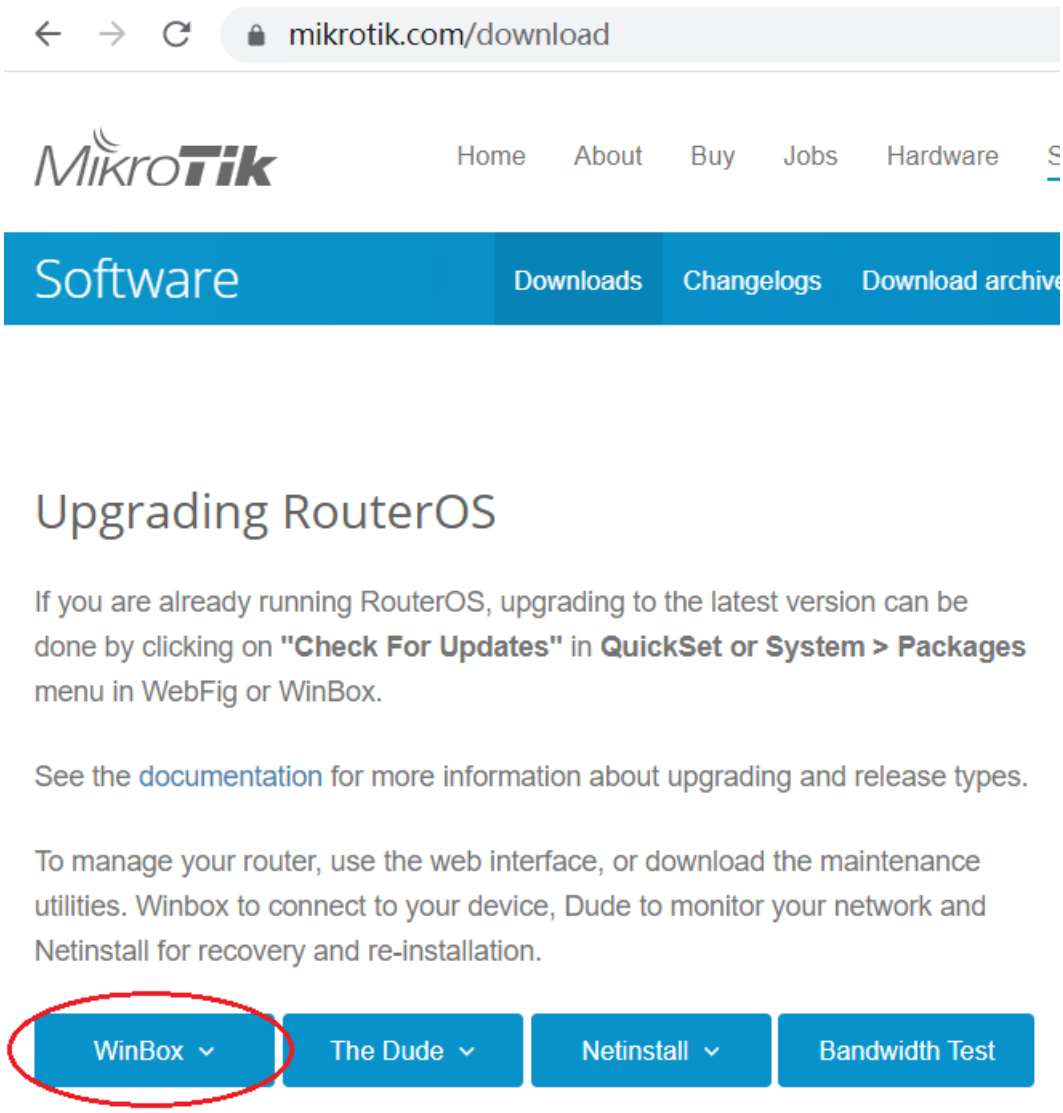
基于 RouterBOARD 系列的串口线序如下：

DB9f	功能	DB9f	DB25f
1+4+6	CD+DTR+DSR	1+4+6	6+8+20
2	RxD	3	2
3	xD	2	3
5	GND	5	7
7+8	RTS+CTS	7+8	4+5

注： 早期的 MikroTik RouterOS 需要定义以上的串口线序，才可以正常通信，不过 5.0 后的 x86 和 RB 系列用标准串口线序也可以进行通信。

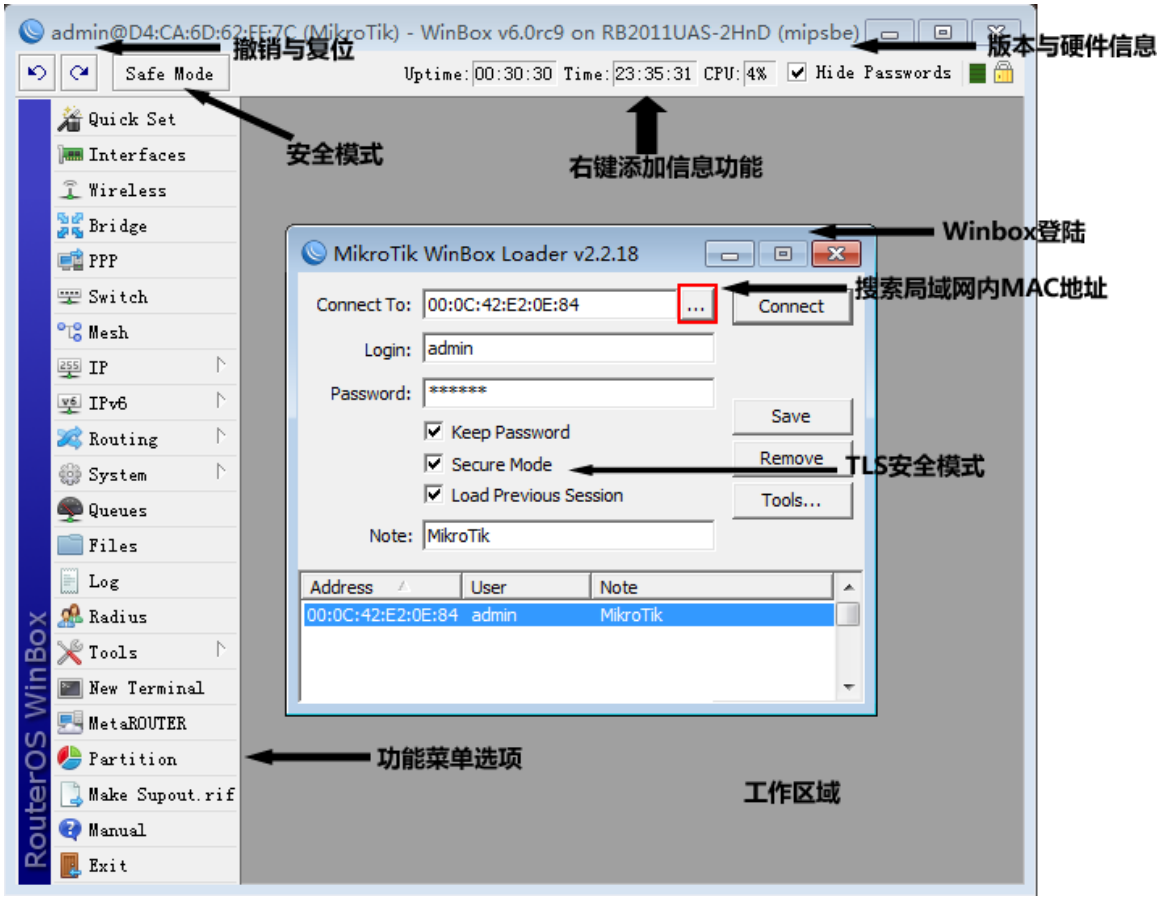
方式 2 Winbox

首先获取 winbox 软件，可以访问官方网站，https://mikrotik.com/download



下载 winbox 应用程序后无需安装，直接可以使用，在 windows 打开 winbox 后，确保你的计算机与路由器已通过网线连接，或者他们两个连接在同一局域网内的交换机上。

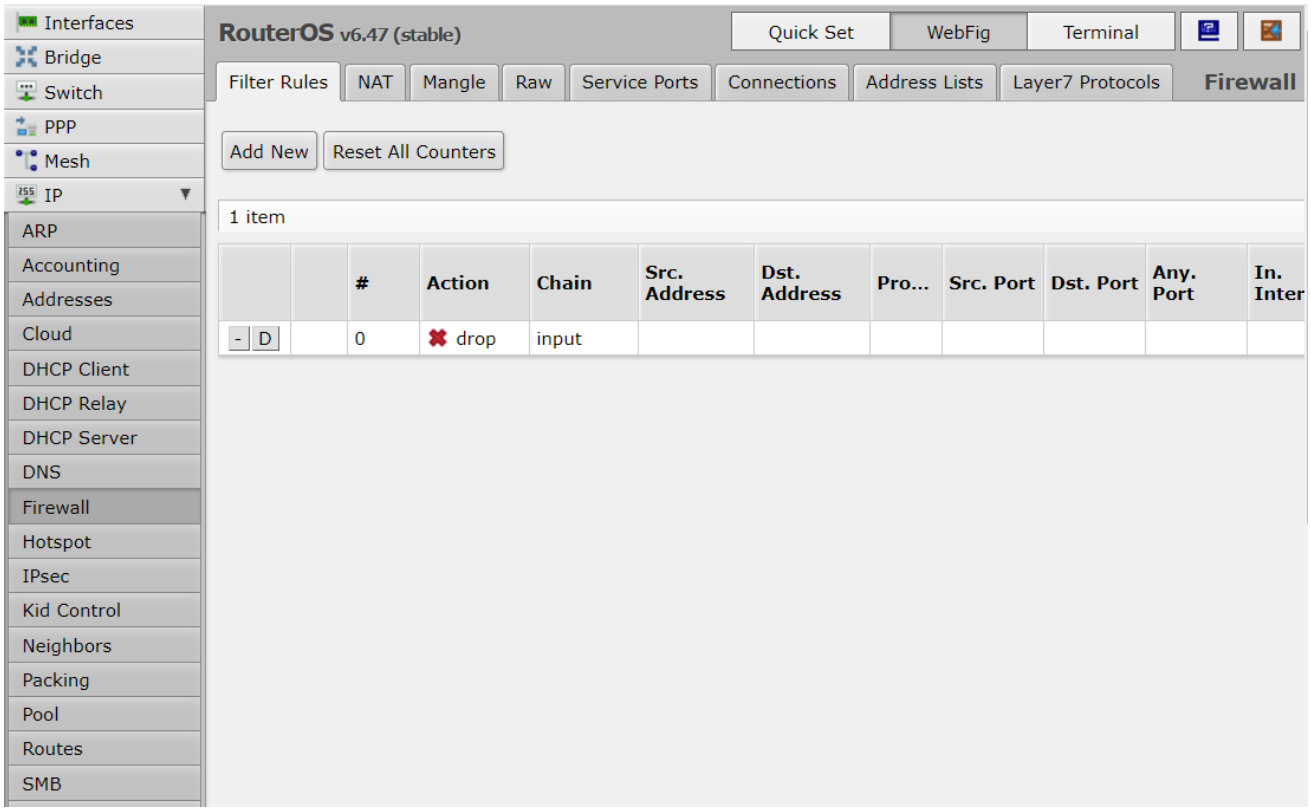
运行 winbox v2 版本，点击 按钮，winbox 会寻找你的路由器 MAC 地址，如果找到将会显示在 winbox 列表框内，选择连接并登陆，你可以设置一下初始化参数，但最好配置一个 IP 地址到对应接口上，通过 IP 连接到 RouterOS，因为通过 MAC 连接设备不是 100%的可靠



注意：winbox v3 版本界面有改动，具体请参考 1.3 章节 Winbox 操作

方式 3 webfig

从 RouterOS v5 版本开始提供完善的 web 配置界面，官方命名为 webfig，通过打开路由器的 web 接口进行配置与当前主流的路由器配置方式一样，配置界面与 winbox 类似



具体配置请参见 1.4 webfig 操作

方式 4 显示器+键盘

如果是基于 PC 的 RouterOS 安装，简单的方式就是通过显示器+键盘进行配置(注意：RouterBOARD 产品不支持，仅采用方法 1 和方法 2) ，启动后可以在显示屏上看到如下登陆提示：

```
MikroTik v6.0
Login:
```

输入登陆名 **admin** 回车后，密码为空，你可以看到如下信息：

```
  MMM          MMM        KKK                                TTTTTTTTTTT       KKK
  MMMM        MMMM        KKK                                TTTTTTTTTTT       KKK
  MMM MMMM MMM   III   KKK   KKK   RRRRRR      OOOOOO       TTT      III   KKK   KKK
  MMM   MM   MMM    III   KKKKK      RRR   RRR   OOO   OOO      TTT      III   KKKKK
  MMM          MMM   III   KKK KKK    RRRRRR     OOO   OOO      TTT      III   KKK KKK
  MMM          MMM   III   KKK   KKK   RRR   RRR    OOOOOO       TTT      III   KKK   KKK

  MikroTik RouterOS 6.0 (c) 1999-2013        http://www.mikrotik.com/


Terminal ansi detected, using single line input mode
[admin@MikroTik] >
```

这样你可以配置路由器了，RouterOS 提供了 setup 命令进行向导配置

方法 5 MAC 层访问 (Telnet 与 Winbox)

RouterOS 支持 MAC 地址连接，在同一局域网内即使没有给 RouterOS 设置 IP 地址，也可以通过 MAC 地址访问管理 RouterOS 设备. 通过 MAC 地址仅在局域网内的 MikroTik RouterOS 路由器之间连接登陆。

操作路径: ***/tool mac-server***

属性描述

interface (name | all; 默认: **all**) –连接 MAC 服务器客户端的接口名
all – 所有接口

注：这是基于网络接口的菜单列表选项，你可以选择添加那些网口可以支持 MAC 地址访问。当 disabled=yes 状态的意思是不允许在接口列表中添加的接口通过 mac 地址进行访问. all interfaces 默认设置为允许任何接口进行 mac 地址远程访问。

使只有 **ether1** 接口能通过 mac 远程访问服务器：

```
[admin@MikroTik] tool mac-server> print
Flags: X - disabled
 #    INTERFACE
 0    all
[admin@MikroTik] tool mac-server> remove 0
[admin@MikroTik] tool mac-server> add interface=ether1 disabled=no
[admin@MikroTik] tool mac-server> print
Flags: X - disabled
 #    INTERFACE
 0    ether1
[admin@MikroTik] tool mac-server>
```

MAC WinBox Server

用于管理该网络接口是否支持 winbox 的 mac 地址登陆。

操作路径: ***/tool mac-server mac-winbox***

属性描述

interface (name | all; 默认: **all**) – 允许使用 mac 地址的协议连接的接口名
all – 所有接口

注：这是基于网络接口的菜单列表选项，你可以选择添加那些网口可以支持基于 winbox 的 MAC 地址访问。Disabled (disabled=yes) 意思是在这些接口中是不允许使用 mac 地址连接的接口.

仅启用 **ether1** 接口的 MAC 服务器

```
[admin@MikroTik] tool mac-server mac-winbox> print
Flags: X - disabled
 #    INTERFACE
 0    all
[admin@MikroTik] tool mac-server mac-winbox> remove 0
[admin@MikroTik] tool mac-server mac-winbox> add interface=ether1 disabled=no
[admin@MikroTik] tool mac-server mac-winbox> print
Flags: X - disabled
 #    INTERFACE
 0    ether1
[admin@MikroTik] tool mac-server mac-winbox>
```

MAC 登陆列表

操作路径: **/tool mac-server sessions**

属性描述

interface (*只读: name*) –连接客户端的接口
src-address (*只读: MAC address*) – 客户 mac 地址（源地址）
uptime (*只读: 时间*) –客户端连接到服务器上的时间

查看 mac 地址连接访问:

```
[admin@MikroTik] tool mac-server sessions> print
 # INTERFACE SRC-ADDRESS         UPTIME
 0 wlan1     00:0B:6B:31:08:22 00:03:01
[admin@MikroTik] tool mac-server sessions>
```

MAC telnet 访问客户端

操作路径: ***/tool mac-telnet***

(MAC address) – 兼容设备的 mac 地址

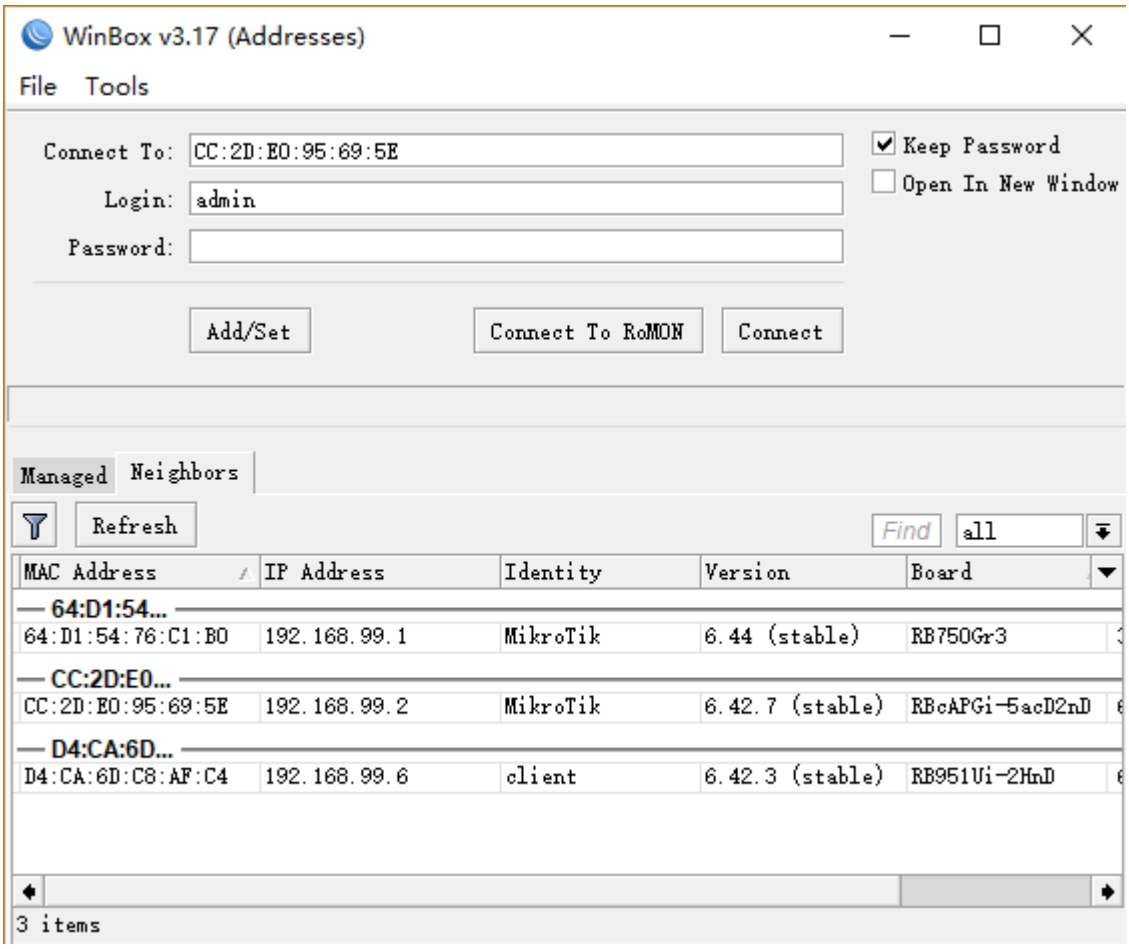
通过 MAC 地址登陆同一局域网的 RouterOS:

```
[admin@MikroTik] > /tool mac-telnet 00:02:6F:06:59:42
Login: admin
Password:
Trying 00:02:6F:06:59:42...
Connected to 00:02:6F:06:59:42

  MMM        MMM          KKK                                TTTTTTTTTTT       KKK
  MMMM       MMMM         KKK                                TTTTTTTTTTT       KKK
  MMM MMMM MMM   III   KKK   KKK   RRRRRR      OOOOOO       TTT      III    KKK   KKK
  MMM   MM   MMM    III   KKKKK      RRR   RRR   OOO   OOO      TTT      III    KKKKK
  MMM         MMM    III   KKK KKK    RRRRRR     OOO   OOO      TTT      III    KKK KKK
  MMM        MMM    III   KKK   KKK   RRR   RRR    OOOOOO       TTT      III    KKK   KKK
```

```
  MikroTik RouterOS 6.0 (c) 1999-2013       http://www.mikrotik.com/

Terminal linux detected, using multiline input mode
[admin@MikroTik] >
```

1.3 Winbox 操作

MikroTik RouterOS 能通过多种方式远程配置，包括 **Telnet**, **SSH**, **WinBox** 和 **Webfig**。在这里介绍怎样使用 **Winbox**，Winbox 是 MikroTik RouterOS 的远程图形管理接口（GUI），也是当前 RouterOS 主要的配置工具，下面是 winbox v3 的界面（可以登录路由器的 web 界面下载，也可以到官方页面下载：https://mikrotik.com/download）：



注：在上图的左上角，可以看到 winbox 版本为 3.17，非 RouterOS 版本号

Winbox 支持 IP 地址、域名和 MAC 地址登陆管理路由器。MAC-telnet 功能，即使用 MAC 地址登陆管理路由器，在之前的 **MAC Winbox Server** 介绍过如何在 RouterOS 开关该功能。MAC-telnet 是在 RouterOS 路由器，在没有配置 IP 地址或者设置了 IP 防火墙参数后无法连接到 RouterOS 情况下使用，通过路由器网卡 MAC 地址登录的方式。MAC-telnet 仅能使用在来自同一个广播域中（因此在网络中不能有路由的存在），且路由器的网卡应该被启用。注:在 Winbox 中嵌入了通过 MAC 地址连接路由器的功能，并内置了探测工具。 这样在管理员忘记或复位了路由器后，同样可以通过 MAC 登陆到 RouterOS 上，进行图形接口操作。

Winbox 控制台使用 TCP 8291 端口，在登陆到路由器后可以通过 Winbox 控制台操作 MikroTik 路由器的配置并执行与本地控制面板同样的任务。（winbox 视频介绍）

注意：Winbox v2 的版本不再此介绍，以下内容全部基于 winbox v3 版本。如果你还在使用 Winbox v2 版本，只能连接管理 RouterOS v5 和以前的版本，RouterOS v6 早期版本还能连接，但存在一些问题，后期 RouterOS v6 版本已经无法连接使用。而且网上出现的 Winbox 中文版非官方版本，是第三方修改不建议使用。同时 Winbox v3 后期版本，无法兼容 RouterOS v5 以下版本。

警告：RouterOS v7 版本只能使用 winbox 3.21 以上版本登录，旧版本 winbox 不支持 RouterOS v7

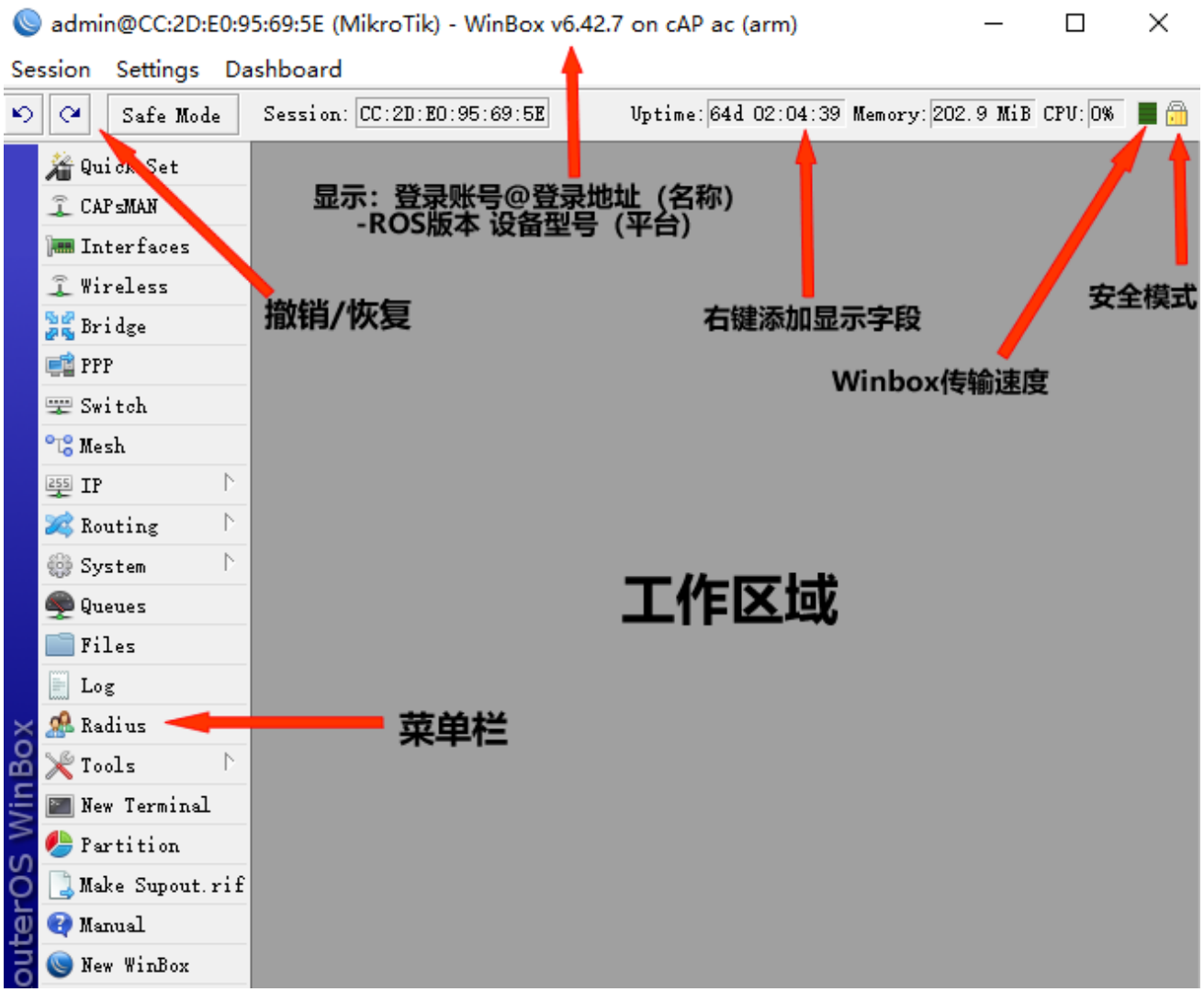
MikroTik 的 winbox 最新版本是 v3.x 主要集中在安全和使用功能上的修改，新的 winbox 3 如下特点：

- 增加了组功能，可对各类 RouterOS 设备分组管理，增大了设备保存清单条目
- 在邻居发现功能中能对 IP 地址、MAC、RouterOS 版本分类
- 能过滤搜索，具备邻居（Neighbors）发现功能
- Winbox 能自动升级新版本
- 新增了当连接丢失，提供重新连接功能，并能保存连接会话
- 增加了设备管理密码验证功能

Winbox 界面概述

Winbox 界面设计的非常直观友好窗口，包括以下：

- 工具栏在顶部用户可以添加一些系统状态，如 CPU、内存使用和工作时间，新的版本中加入了当前日期和时间。
- 菜单栏在左侧，所有功能列表菜单和子菜单，这个列表根据安装功能包不同而增减变化，例如 IPv6 功能包在 RouterOS 系统没有添加，此时 IPv6 菜单和他的子菜单将不会被显示在左侧栏。
- 工作区域，显示所有工作的窗口



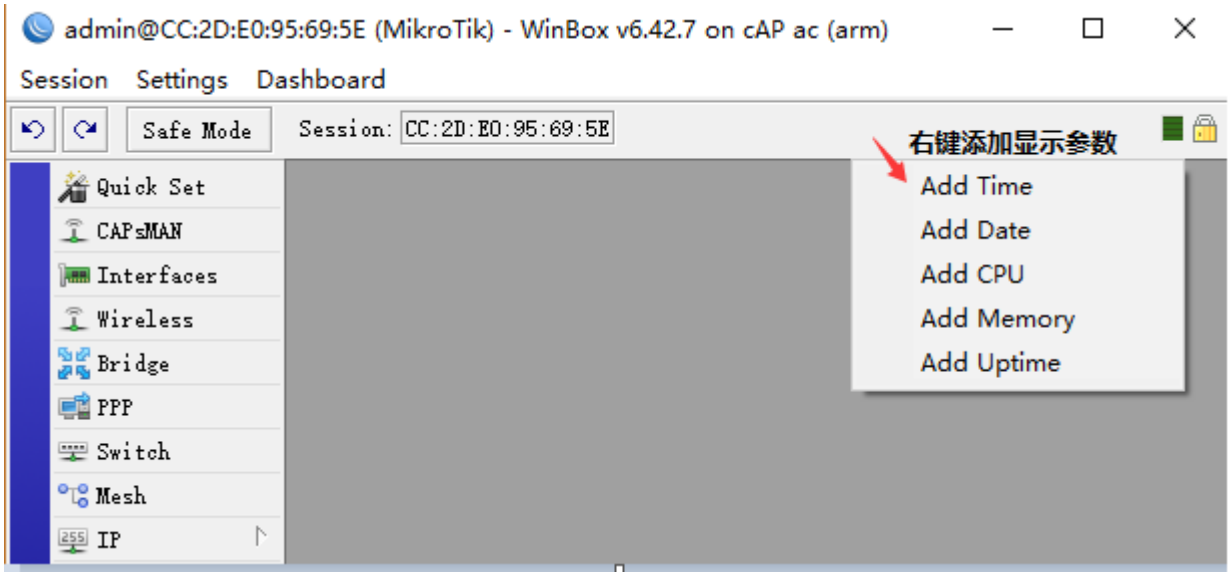
标题栏显示路由器身份信息，显示格式如下：

```
[登录账号]@[路由器的 IP 或者 MAC] ( [RouterID] ) - Winbox [RouterOS 版本] on [RB 型号] ([平台])
```

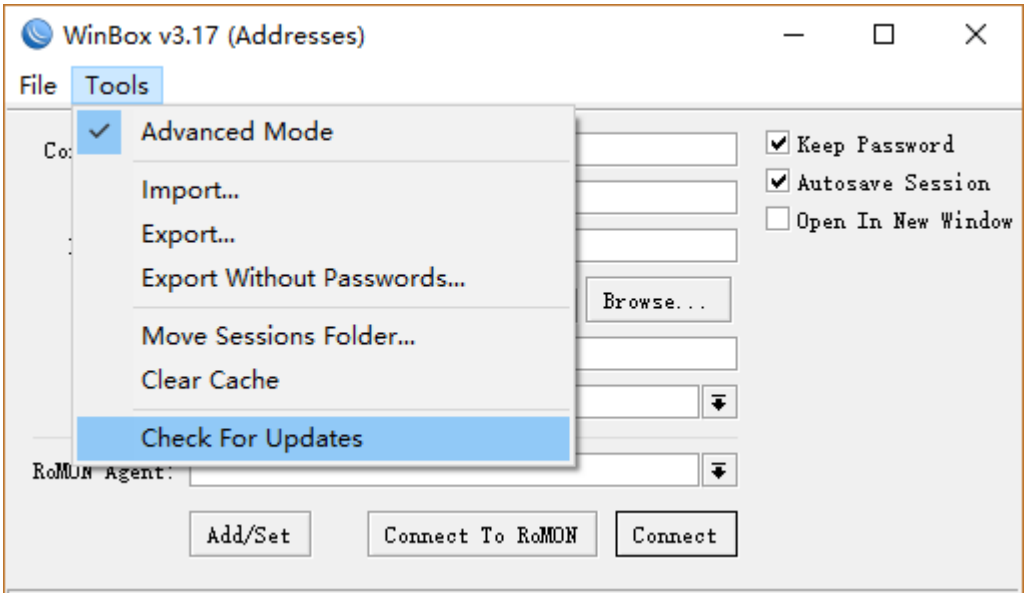
从截图上我们能看到通过 MAC 地址，使用账号 **admin** 登陆，路由器的身份 ID 是 **MikroTik**，当前安装 RouterOS 版本是 **v6.42.7**（非 winbox 版本），RouterBOARD 型号 **cAP ac**，平台是 **ARM**

在左边工具栏有 **undo** 和 **redo** 按钮，快速撤销和恢复操作，特别管理员操作失误，可以通过撤销和恢复进行还原。

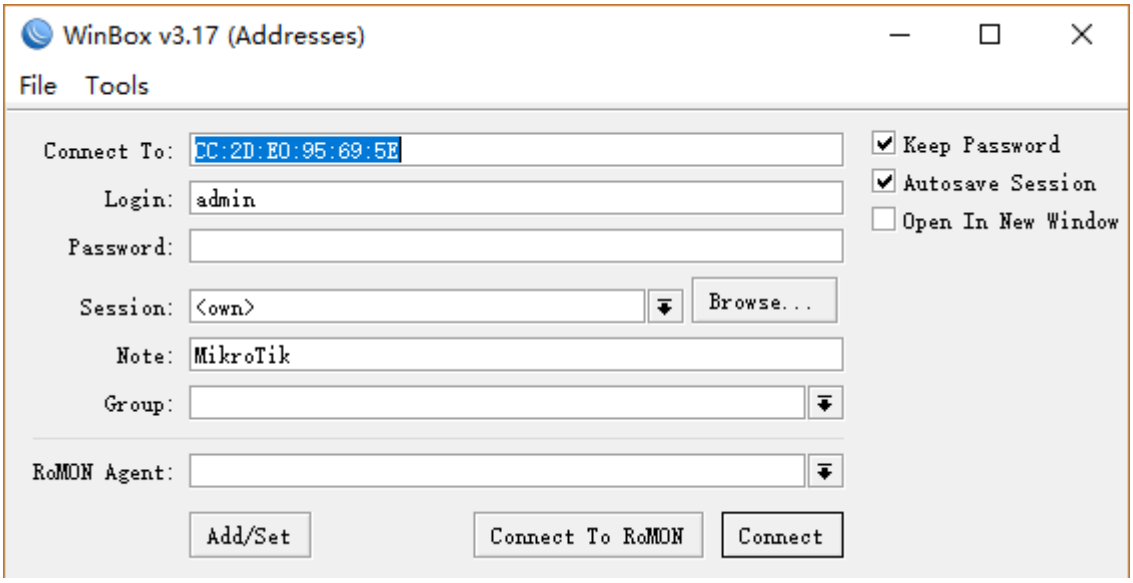
右键显示时间、日期、CPU 使用率、内存使用情况和已工作时长信息



打开 Tools 菜单，可以找到 Check For Updates 选项，可以通过连接 MikroTik 服务器，检查版本更新情况：



Advanced Mode 勾选框是高级模式和简要模式的界面切换，下面截图是高级模式界面：



具体说明下 Winbox 界面的各个按钮和选项

Winbox 窗口按钮描述

- **简要模式下:**

-- 按钮/复选框

- **Connect** – 连接按钮
- **Connect To RoMON** – 连接 RoMON 代理按钮
- **Add/set** – 保存/编辑路由器管理列表按钮
- **Open In New Window** – 复选框，用于连接时是否新建一个 winbox 窗口

-- 字段

- **Connect To:** - 连接路由器的 IP 或 MAC 地址
- **Login** – 登录设备的的账号名称
- **Password** -登录设备的的密码
- **Keep Password** – 是否保存密码到管理列表复选框

- 高级模式:

-- 按钮/复选框

- **Browse** – 浏览指定会话保存的文件目录
- **Secure mode** – 复选框如果选择，Winbox 将使用 DH-1984 作为密钥交换，并使用 RC4-drop3072 加密会话安全
- **Autosave session** -为建立连接的设备自动保存会话

-- 字段:

- **Session** – 保存会话的路由器
- **Note** – 为保存到管理列表的设备添加备注
- **Group** – 为保存到管理列表的设备指定分组
- **RoMON Agent** – 从设备列表中选择 RoMON 代理

Winbox 菜单项的描述

-- File

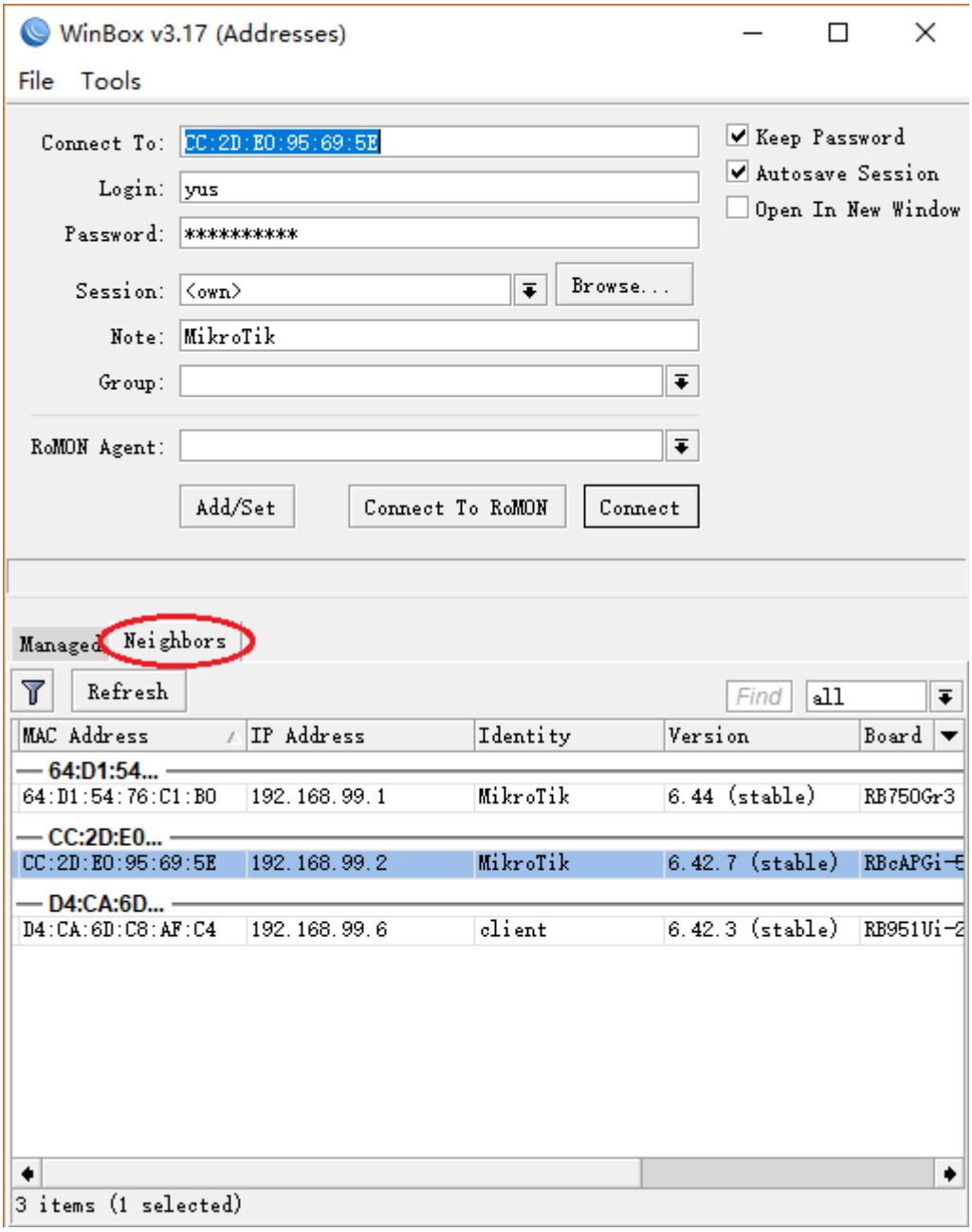
- **New** – 创建一个新的设备管理列表
- **Open** – 打开一个设备管理列表文件
- **Save As** – 保存当前设备管理列表到指定文件
- **Exit** – 退出 Winbox 程序

-- Tools

- **Advanced Mode** – 启用或禁用高级模式
- **Import** – 导入保存的会话文件
- **Export** – 导出并保存会话文件
- **Move Session Folder** – 修改会话文件保存路径
- **Clear cache** – 清除 winbox 缓存
- **Check For Updates** – 检查 winbox 更新

警告: 默认情况下，设备管理列表不加密。如果要加密它，需要设置主密码（set master password）

在 winbox v3 版本将老版本的 按钮键取消，代替这个功能的是 Neighbors 邻居发现列表，当点击开 Neighbors 列表后，会自动搜索当前局域网内的 MikroTik 设备，如下图：



菜单栏

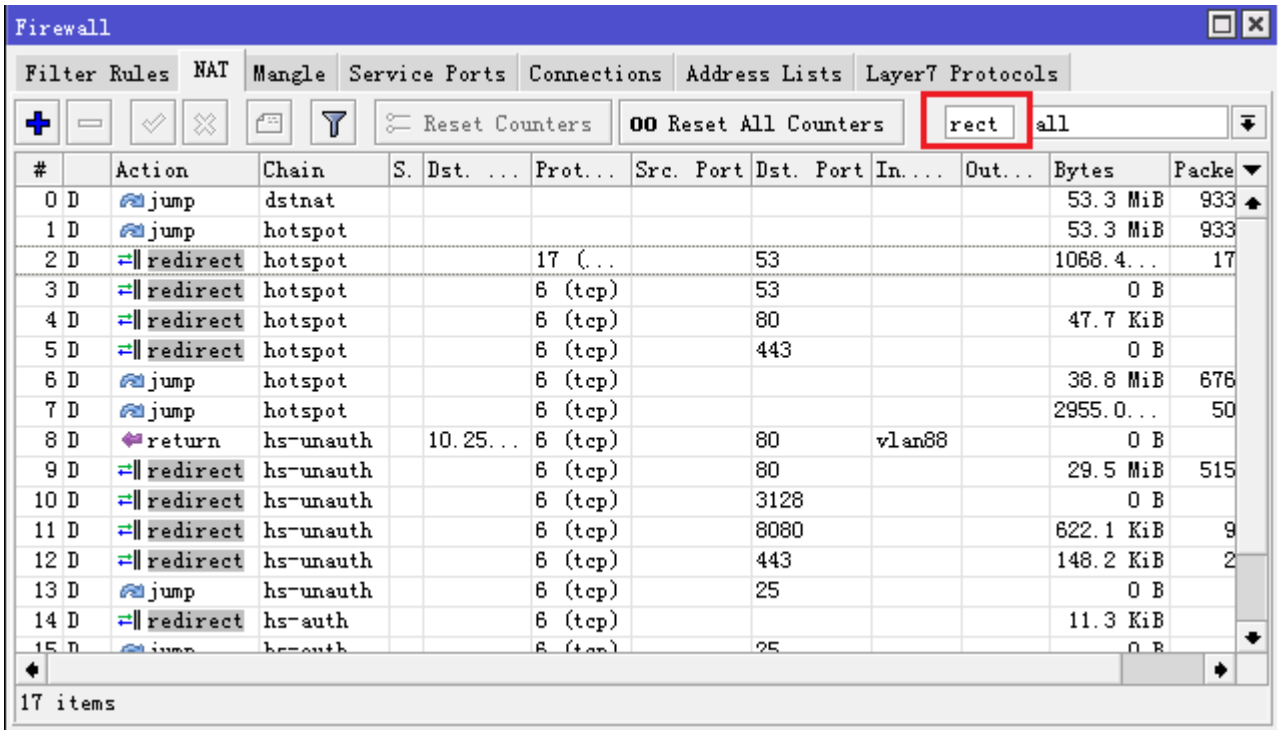
在进入 Winbox 后，左侧为管理菜单栏，包含了主要的功能菜单：

进入每个功能菜单后，在子窗口有自己的工具栏，大多窗口都有相同的工具栏按钮：

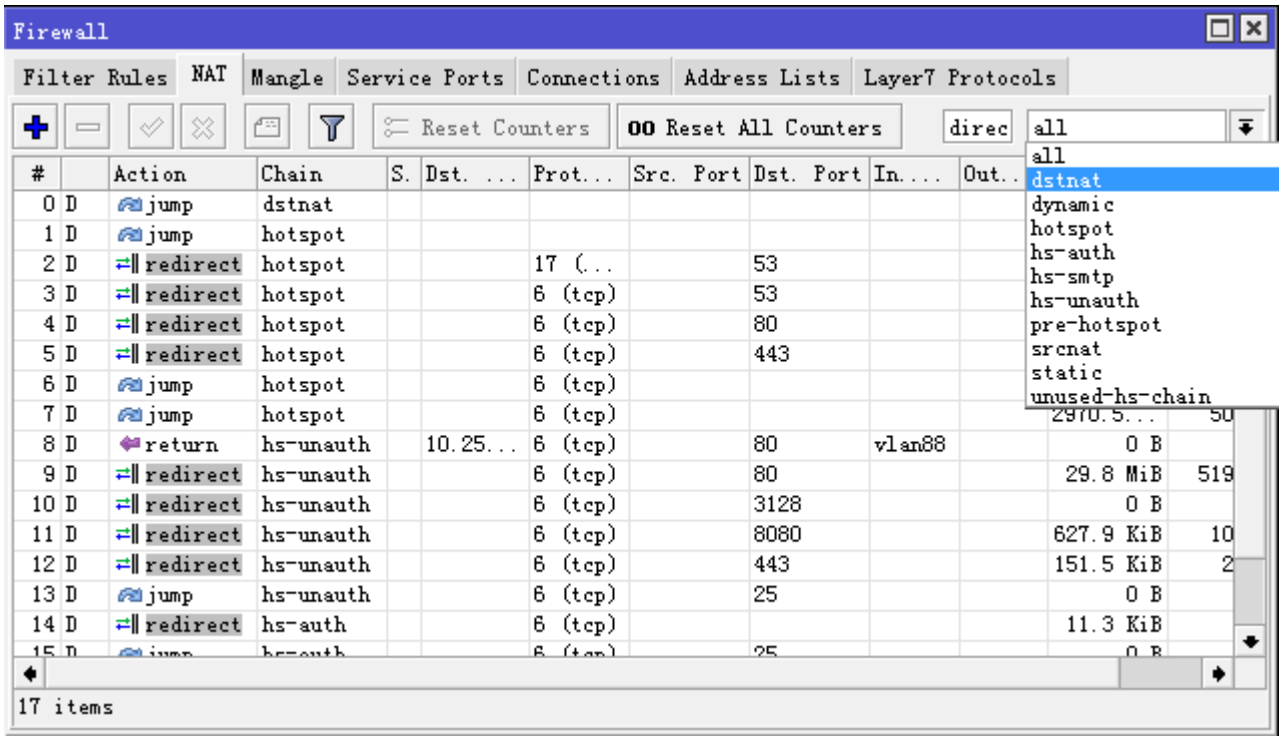
图示	功能		图示	功能
	添加一条项目			定义或编辑一个注释
	删除一条存在项目			过滤查询
	启用一条项目			撤销操作
	禁用一条项目			恢复操作

快速查询

几乎所有的窗口有快速查询字段输入框，在子窗口的右侧顶部，任何文本字段输入后在当前窗口下查询相关的信息，如下面的截图输入“**redirect**”查询的结果



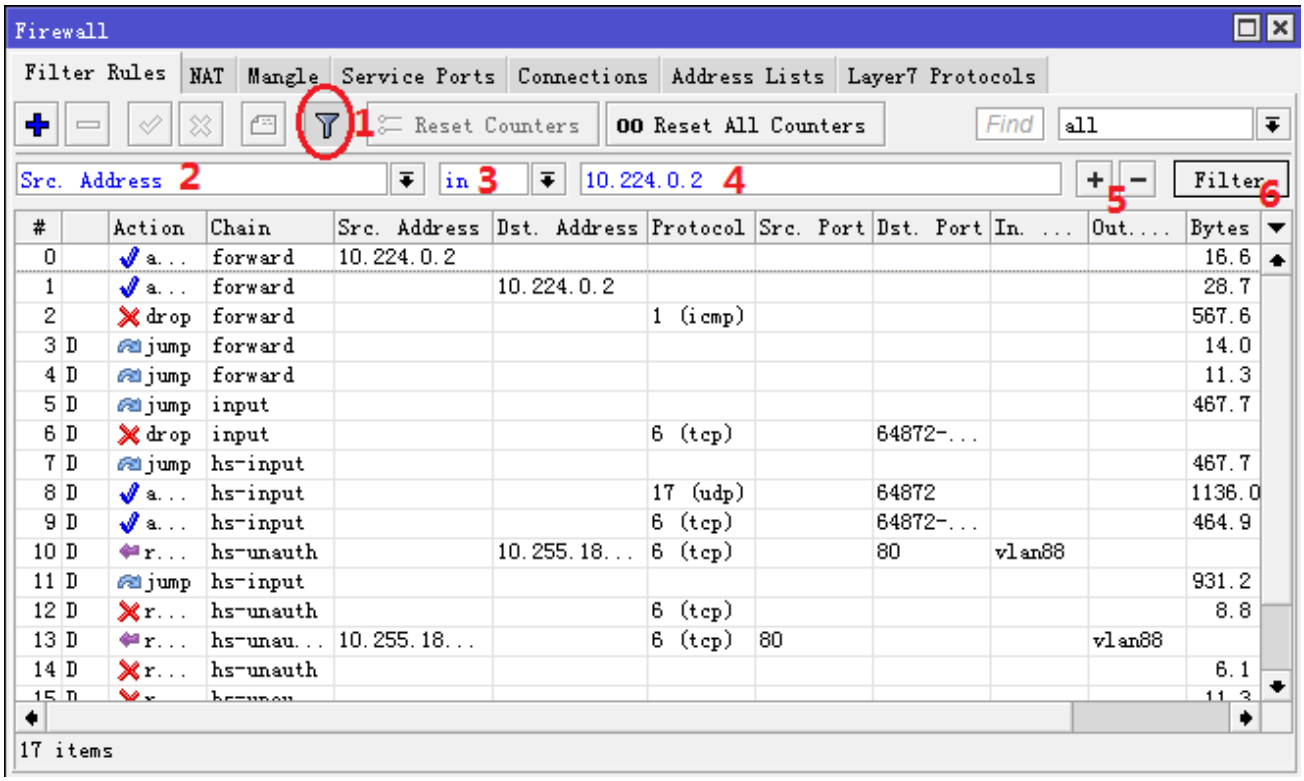
注：上面是在 nat 中的快速查询框，注意的右侧由于个下拉列表，这个是可以调出配置的路由表，例如： dstnat 被选中，这时仅 dstnat 的列表中查询



类似的下拉列表在 firewall filter 和 ip route 中也有

过滤筛选

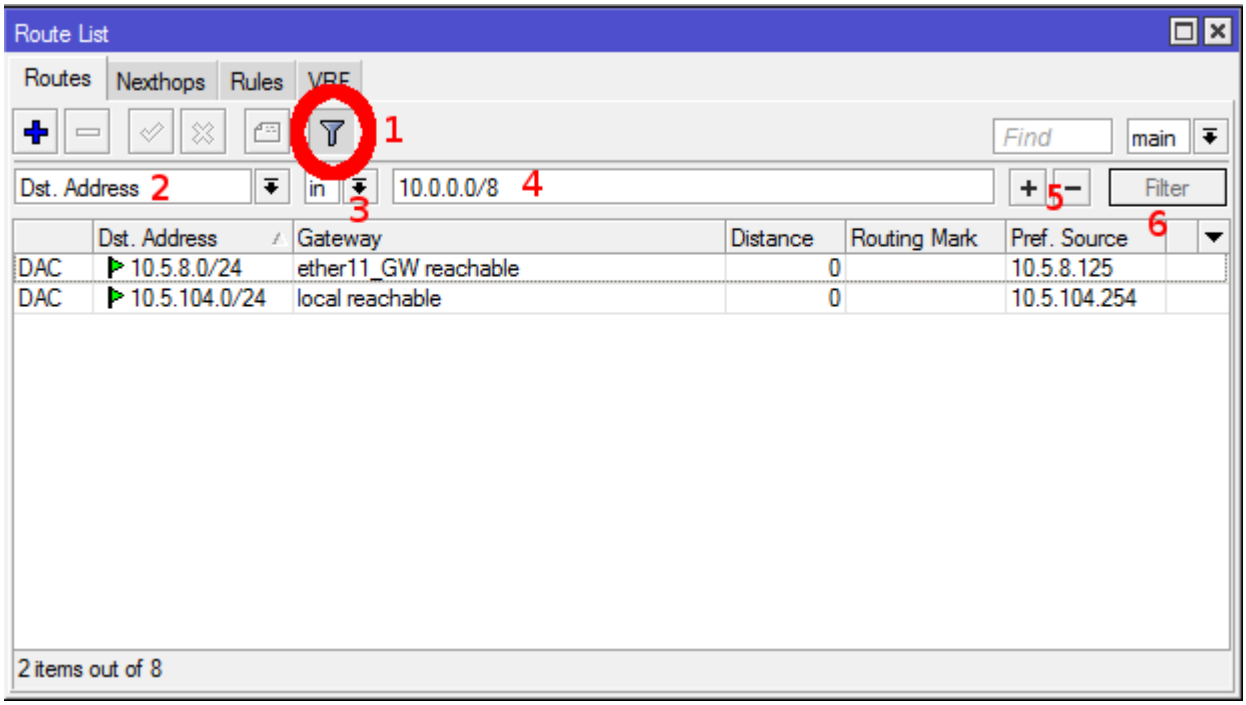
大部分窗口都有 **Filter** 按钮。当点击这个按钮，会显示出多个过滤查询选项，如下显示



上图在 firewall filter 中查找 src.address 为 10.224.0.2 的规则

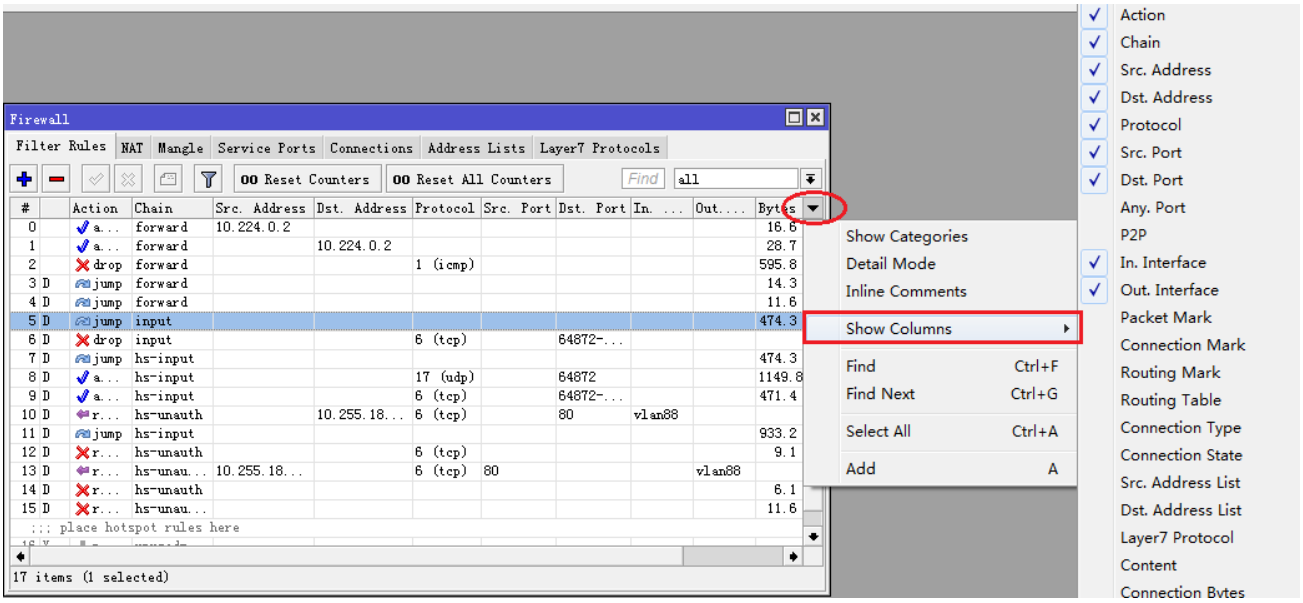
1. 按 Filter 按钮
2. 从第一下拉列表选择 **Src.Address**
3. 通过下列框选择 **in** 格式，in 的意思是将过滤是否目标地址值在指定的网络范围内。
4. 输入我们要比较的网络参数（在我们这个事例里，输入 10.224.0.2）
5. 这个按钮是添加或者删除其他的过滤
6. 按 **Filter** 按钮应用我们的过滤设置

下面是 IP Route 路由表中过滤目标地址，包含在 10.0.0.0/8 段的路由规则



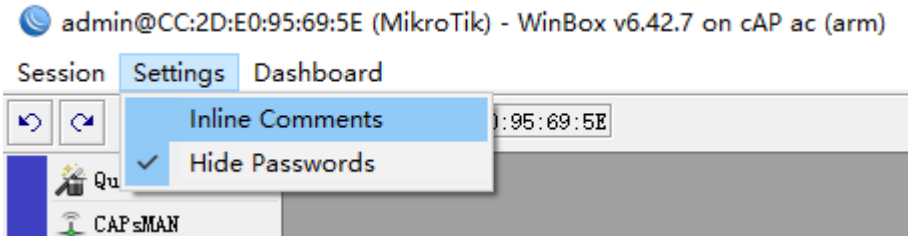
自定义显示列表

默认的 winbox 显示大多的使用参数，然而有时需要查看其他参数，例如 Queue Simple 里 tx 和 rx 速率，Firewall Filter 或 nat 信息等，Winbox 允许自定义显示每个窗口的信息。例如显示 Queue Simple 里的 tx 和 rx 速率



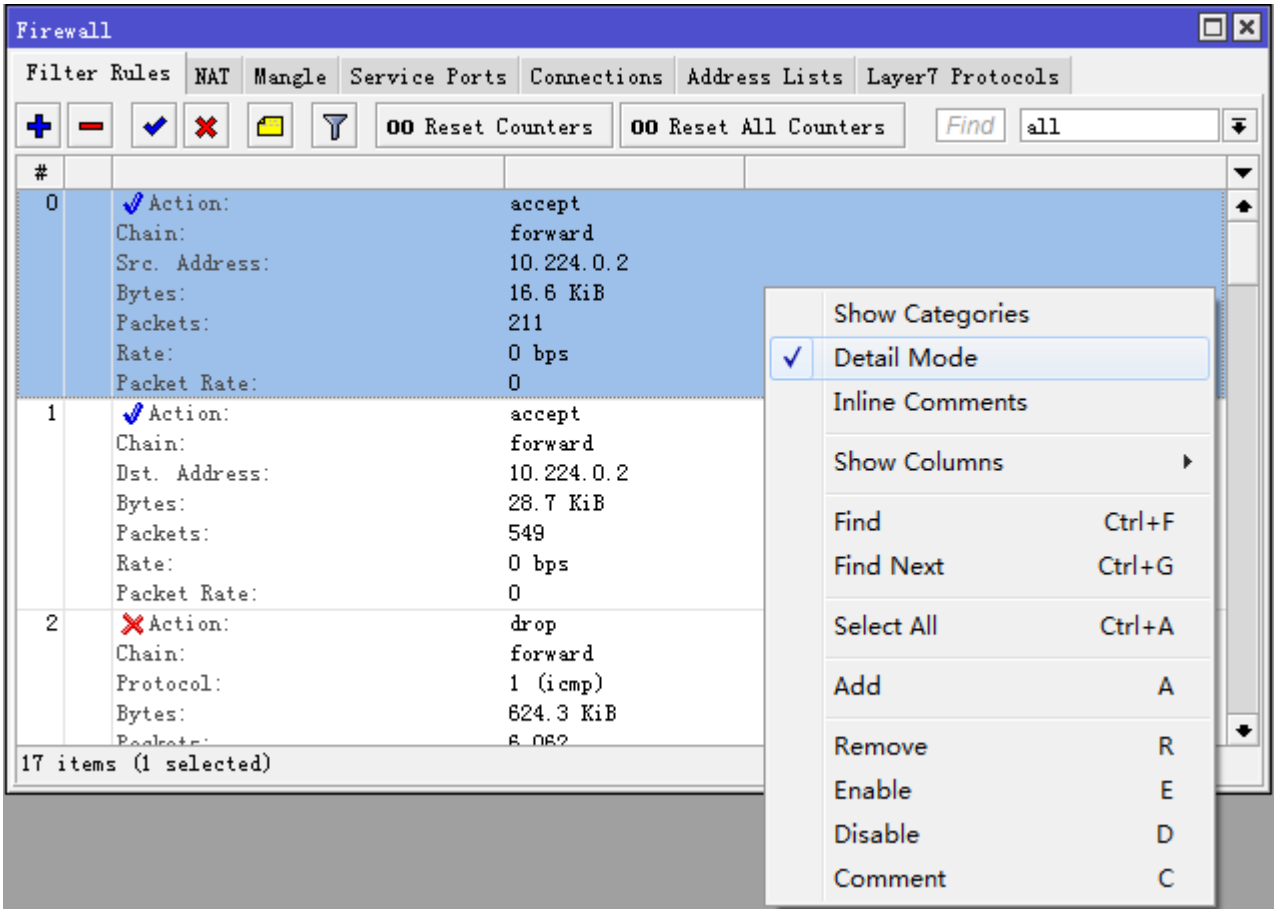
隐藏选项

在 winbox 打开后，顶部菜单有 settings 选项，里面有 Inline Comments 和 Hide Passwords，Inline Comments 是隐藏注释，Hide Passwords 是隐藏 hotspot users 和 PPP secrets 下账号的密码。



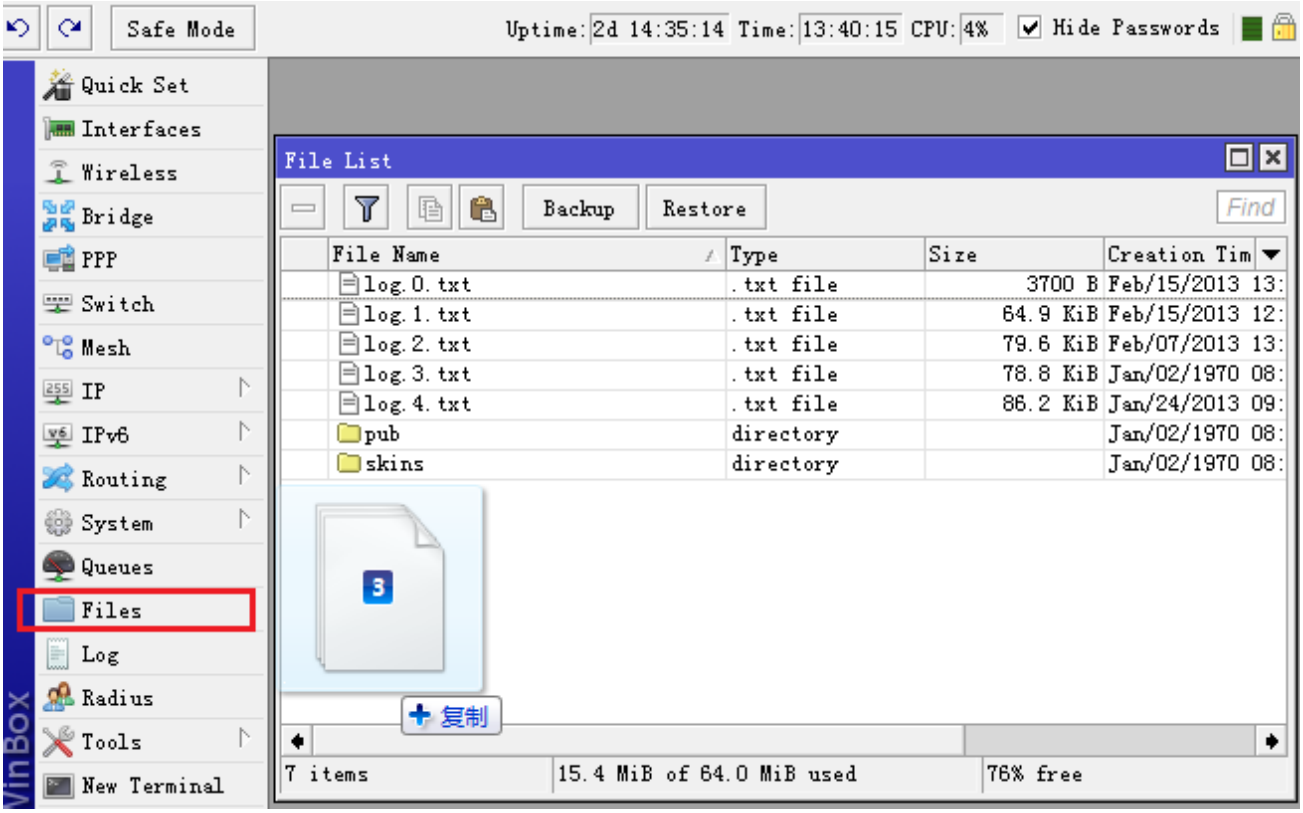
详细模式

在 winbox 中可以启用详细模式，在这个模式下所有的参数都被显示在窗口的栏目中，启用详细模式右键鼠标，点击列表项目中的 **Detail mode**



拖&放

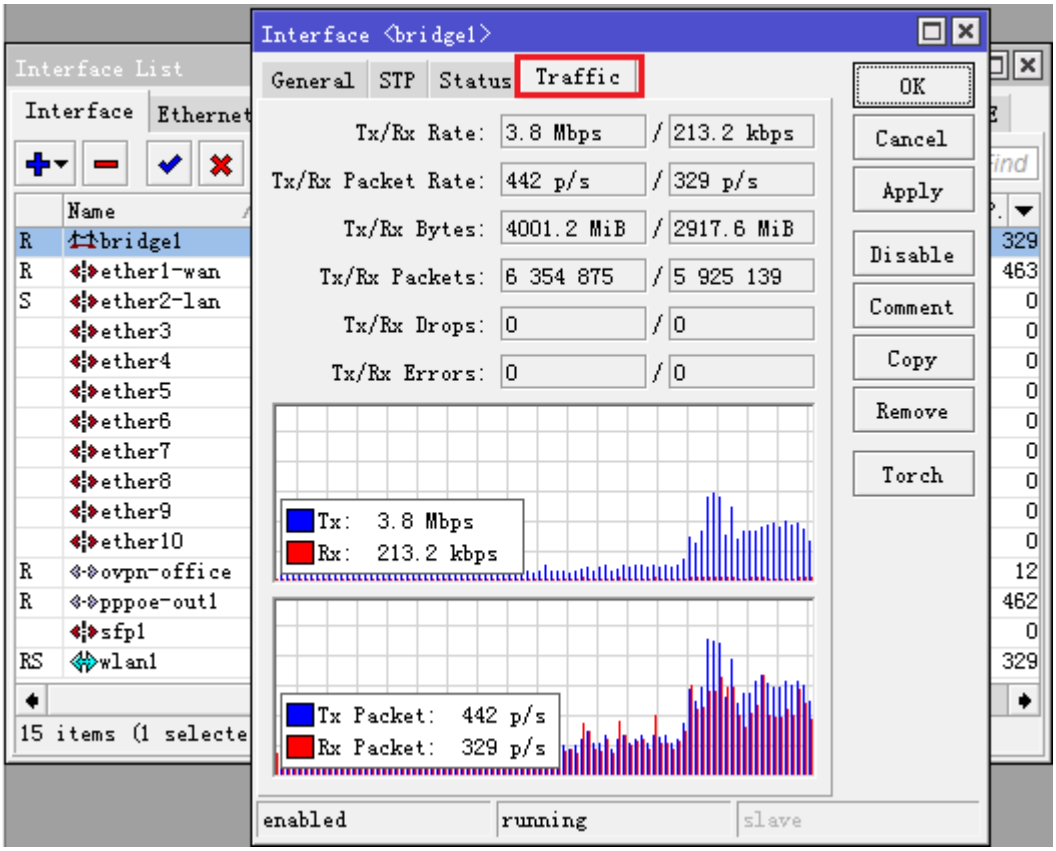
Winbox 可以上传和下载文件，我们可以使用 winbox 拖&放功能实现，我们可以从 windows 计算机上将文件拖&放到 File 列表下，在 5.0 后可以随意拖放到 winbox 中的任意位置，即可上传文件。也可以从 file 列表下拖出文件。



注：拖&放功能不能在 Linux 下的 winbox 实现，这个不是 winbox 的问题，而是 wine 不支持拖&放功能。

流量监测

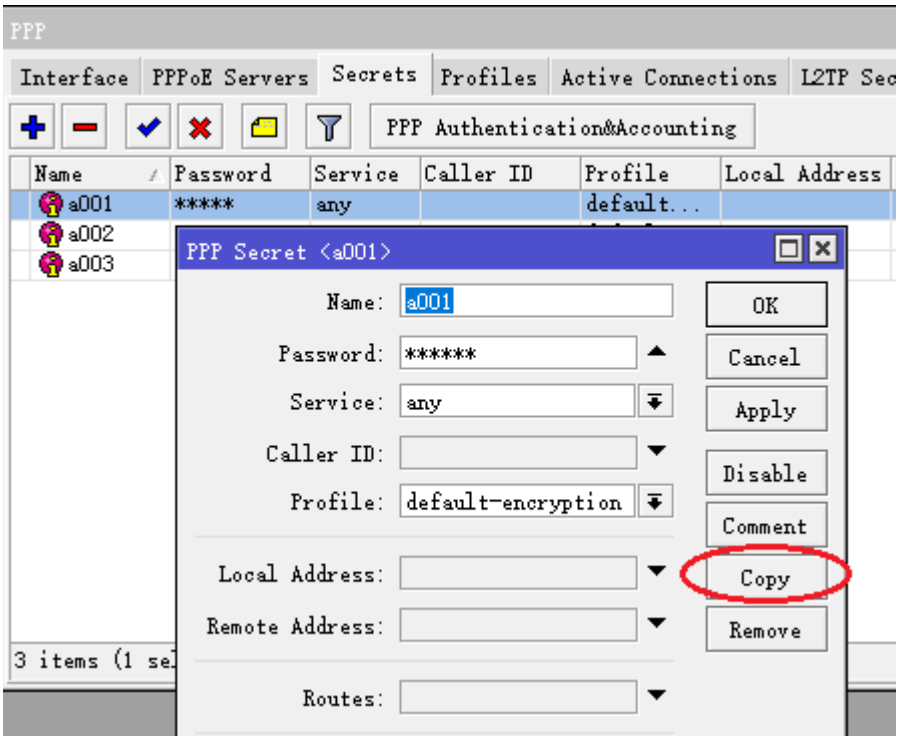
Winbox 在各种接口、防火墙规则和流控规则的窗口选项中，都有一个 Traffic 菜单，提供实时流量的查看和统计，实时流量仅在打开时可以看到，窗口关闭后不再记录。



项目复制与静态绑定

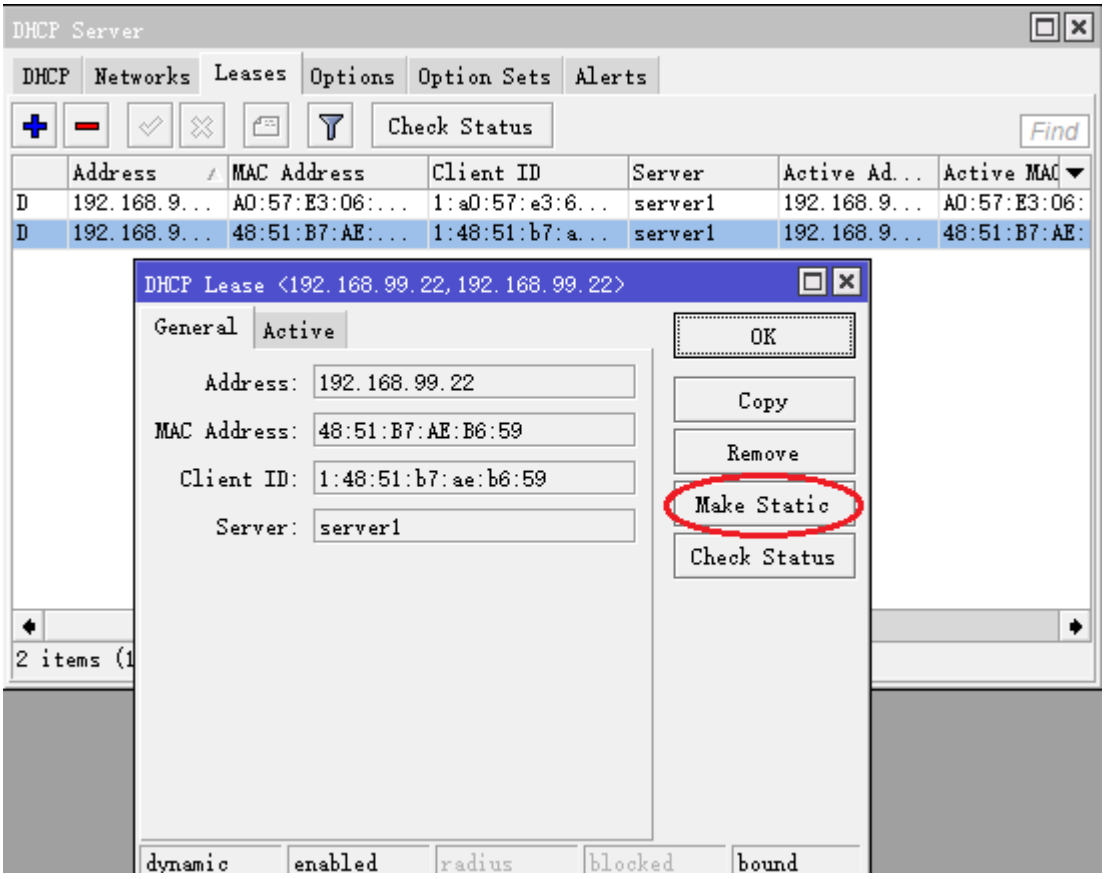
项目复制

简化了同类型规则的配置操作，可以将同类型的规则复制一份，稍作修改即可使用，例如我们在创建 PPP 下，创建账号时，可以从其他账号复制后，修改账号和密码就可以使用。我们打开 a001 的账号，选择 copy，会弹出新的窗口。



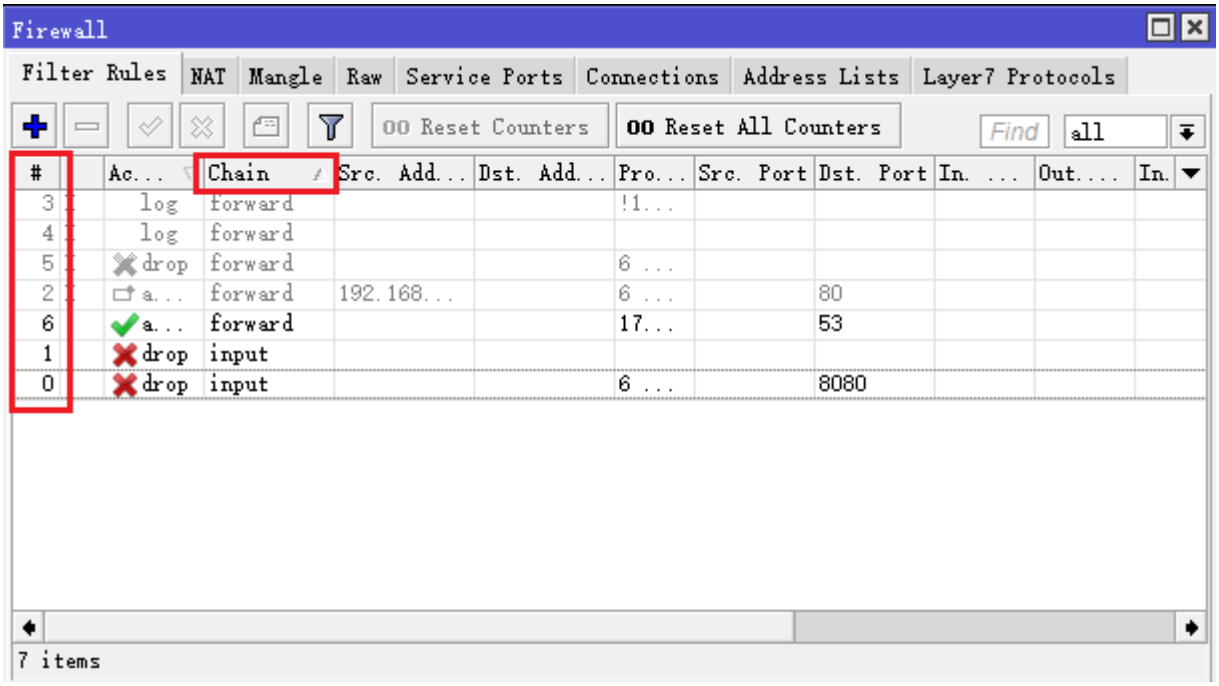
静态绑定

静态绑定是将动态规则变为静态，如在 DHCP Leases 和 IP ARP 下点击一个条目，在右侧按钮会有 Make Static 选项，如下面是 DHCP Server 的 Leases 租约配置，我们可以把一个主机获取的 IP 地址做静态绑定。

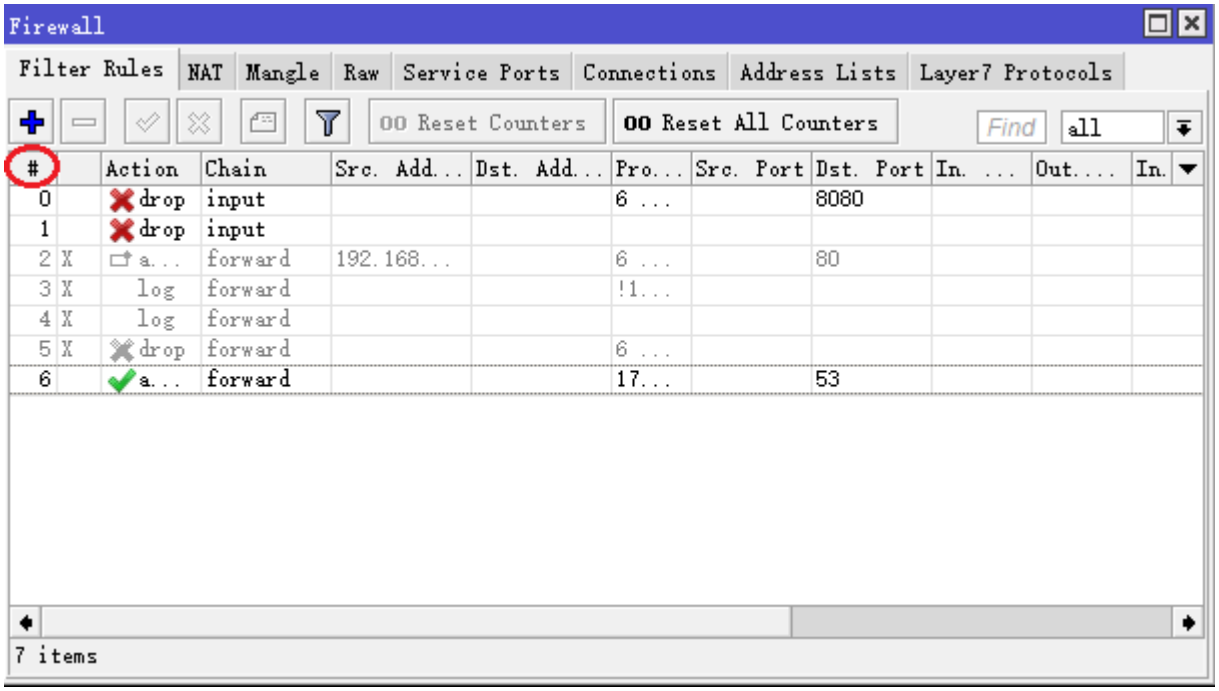


序列重排

在 Firewall 和 Queue 有规则顺序的菜单下配置，我们如果对某一类型属性进行了排序，将打乱正常的序列（#号下的数字顺序），如我们对 chain 属性进行了排序，这时正常序列被打乱。

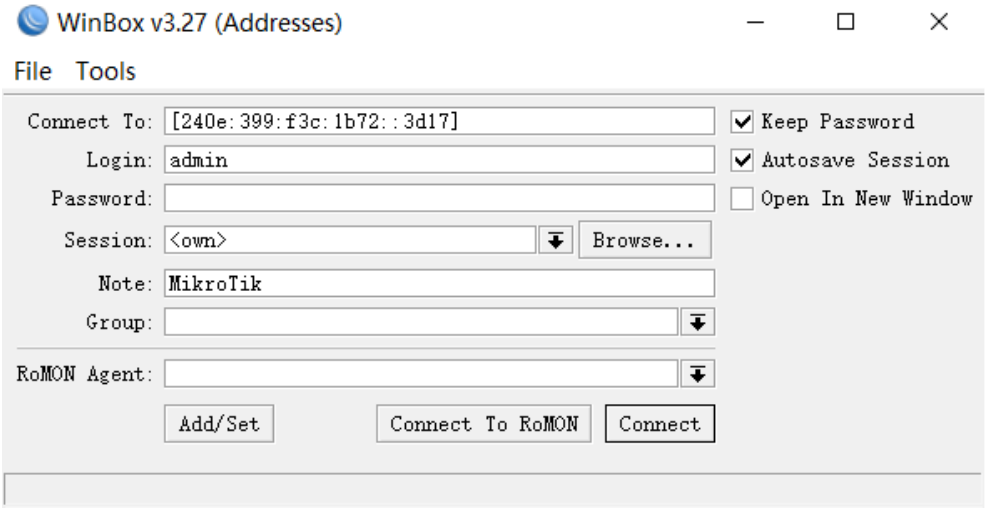


通过点击＃号，恢复正常序列排序



Winbox IPv6 连接

在使用 Winbox v3 版本支持 IPv6 地址连接，但需要在 IPv6 地址加上方括号，如[240e:399:f3c:1b72::3d17]



Winbox 目录转移设置

在 windows Vista/7 winbox 设置存储在以下目录：

%USERPROFILE%\AppData\Roaming\Mikrotik\Winbox\winbox.cfg

可以拷贝文件到其他 windows 系统上可以保留操作信息

故障分析

- **我能在 Linux 上运行 Winbox?**

  能，使用 Wine 图形接口，可以运行 Winbox 并连接到 RouterOS。

- **我不能打开 Winbox 控制台**

  检查路由器上**/ip service print** 的 winbox 服务端口和地址是否正确，确定地址是你能连接到的指定网络，确定端口为你指定的端口。如果你的服务端口和访问地址被修改，你可以通过下面的命令设置回默认值 **/ip service set winbox port=8291 address=0.0.0.0/0** 。

1.4 Webfig 操作

从 RouterOS v6 版本开始，x86 平台 ether1 接口默认 IP 地址是 192.168.88.1，而 RB/CCR/CRS 等硬件设备，如果是单网卡的 ether1 默认是 192.168.88.1，如果是多网卡的 192.168.88.1 会配置到 ether2-etherX 接口上，通过在浏览器中输入 http://RouterIP 可以访问到 RouterOS 的 web 页面

通过 http 网页管理 RouterOS 有两种方式 webbox 和 webfig，但在 5.0 版本后随着 webfig 功能的稳定和完善，webbox 被取消。这里将不再介绍 webbox 的操作。

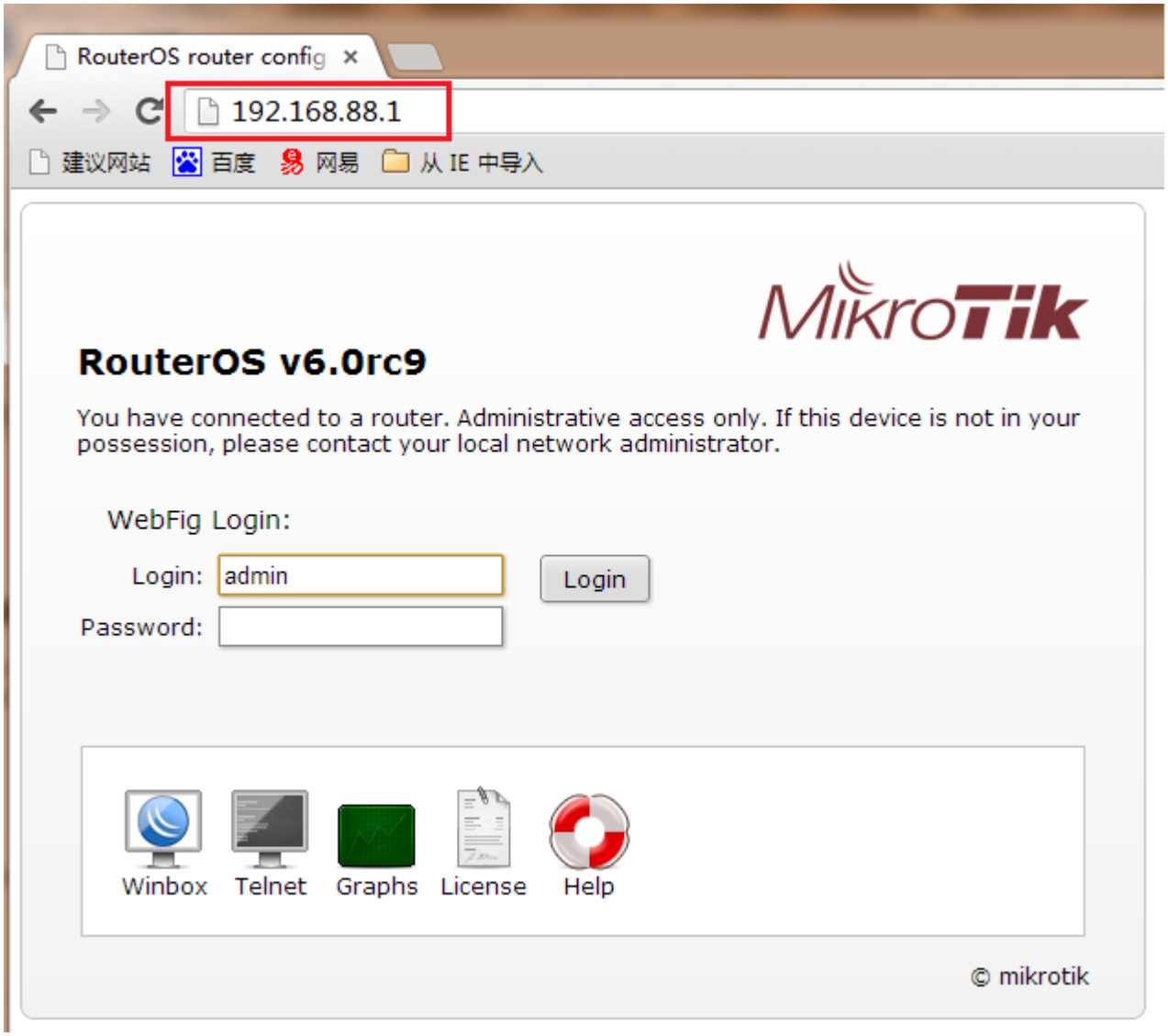
WebFig 是 RouterOS 基于 Web 浏览器的配置工具。能直接管理路由器，不需要增加任何管理软件，通过 Web 浏览器配置，Webfig 是一个独立的平台，可以通过移动设备访问，被用于路由器的直接管理配置。

WebFig 被设计为 winbox 的复制版本，都有相同的布局，webfig 能管理大部分的 RouterOS 功能，但 webfig 只能通过三层网络及 IP 访问，而 winbox 则能使用二层的 MAC 访问，也能通过 IP 访问，管理手段较多，

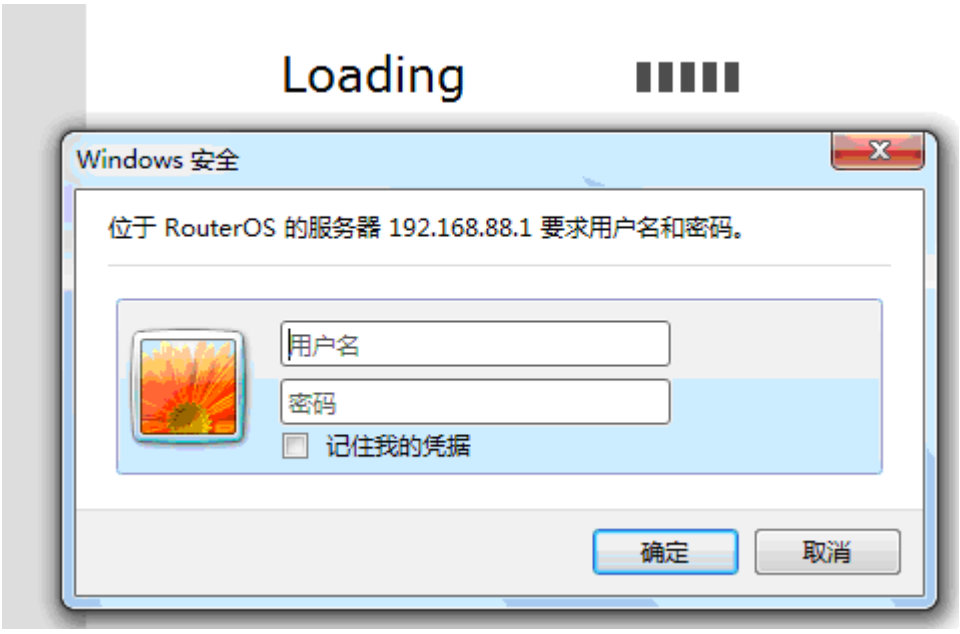
注：因为 webfig 是基于 IP 层访问，如果你需要修改连接路由器的 IP 地址，请谨慎操作；当配置二层的 bridge 或者 ip firewall filter 的 input 链表需要考虑到是否影响 IP 访问

连接到 RouterOS

WebFig 能通过在浏览器输入 RouterOS 的 IP 地址，访问到主页并选择 webfig 的图示启动，如下图。



在点击 webfig 的图示后，会出现登录提示框，要求你输入登录路由器的用户名和密码



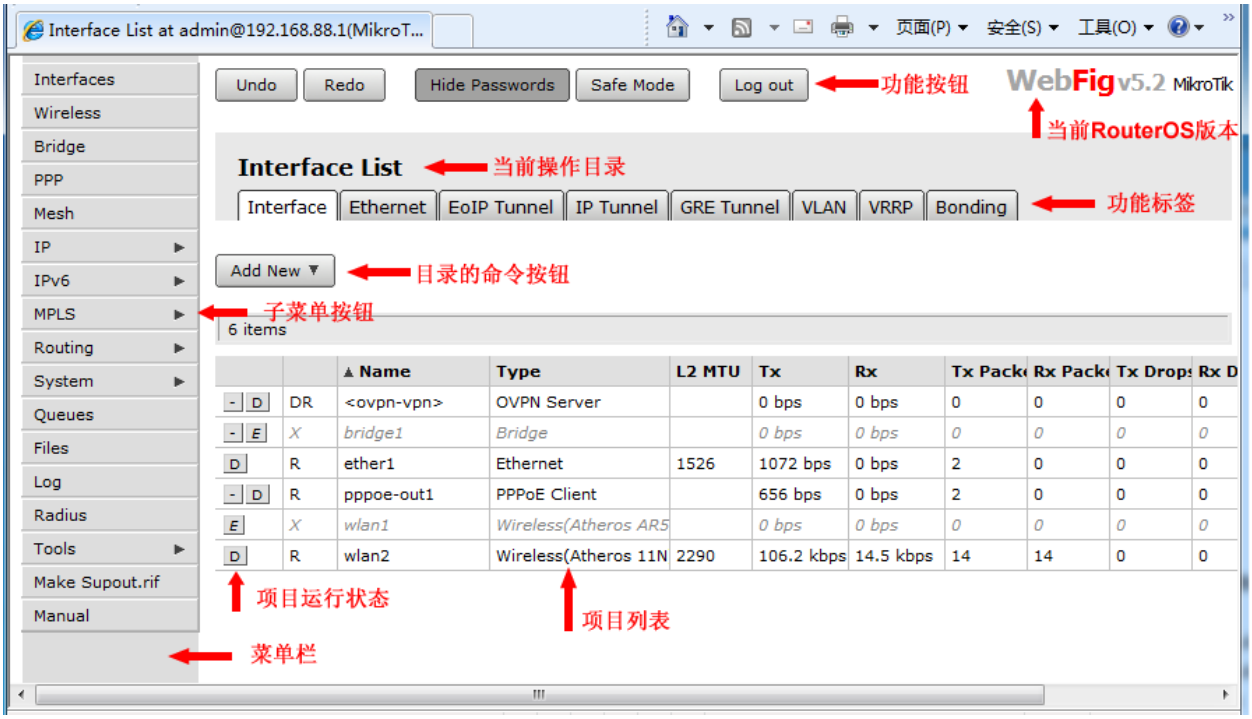
输入正确的用户名和密码即可登录到 RouterOS 的 webfig 配置

IPv6 连接

RouterOS 的 http 服务现在能监听 IPv6 的地址，能通过 IPv6 浏览，在你的浏览器里输入 IPv6 地址需要一个中括号，例如 **[2001:db8:1::4]**. 如果是要求连接到本地地址，不要忘记在，指定网卡名称或者网卡的 ID，例如 **[fe80::9f94:9396%ether1]**.

操作界面介绍

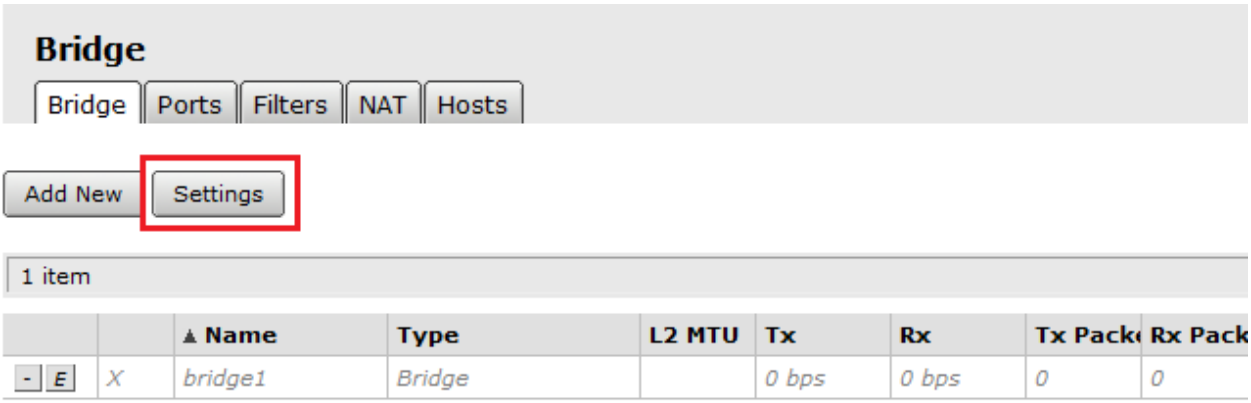
WebFig 接口被设计为近似于 Winbox，几乎同样的布局，菜单栏在左侧，undo 和 redo 在顶部，工作区域在中间部分



菜单栏在左侧如同 winbox 格式布局，小箭头代表该菜单下包含子菜单，当点击箭头子菜单会显示在出来

在页面的顶端你可以看到 undo/redo 两个按钮，同样也有 hide password 隐藏密码和 safe mode 安全模式的按钮，在右侧有一个 log out 注销退出的按钮。顶部的右侧可以看到 RouterOS 的版本或者是 RouterBOARD 的信息

工作区域，我们看到当前的操作目录和各种功能卷标，下面点可以看到一个 Add New 添加新项目的按钮，在 Bridge 页面可以看到多一项 setting 按钮

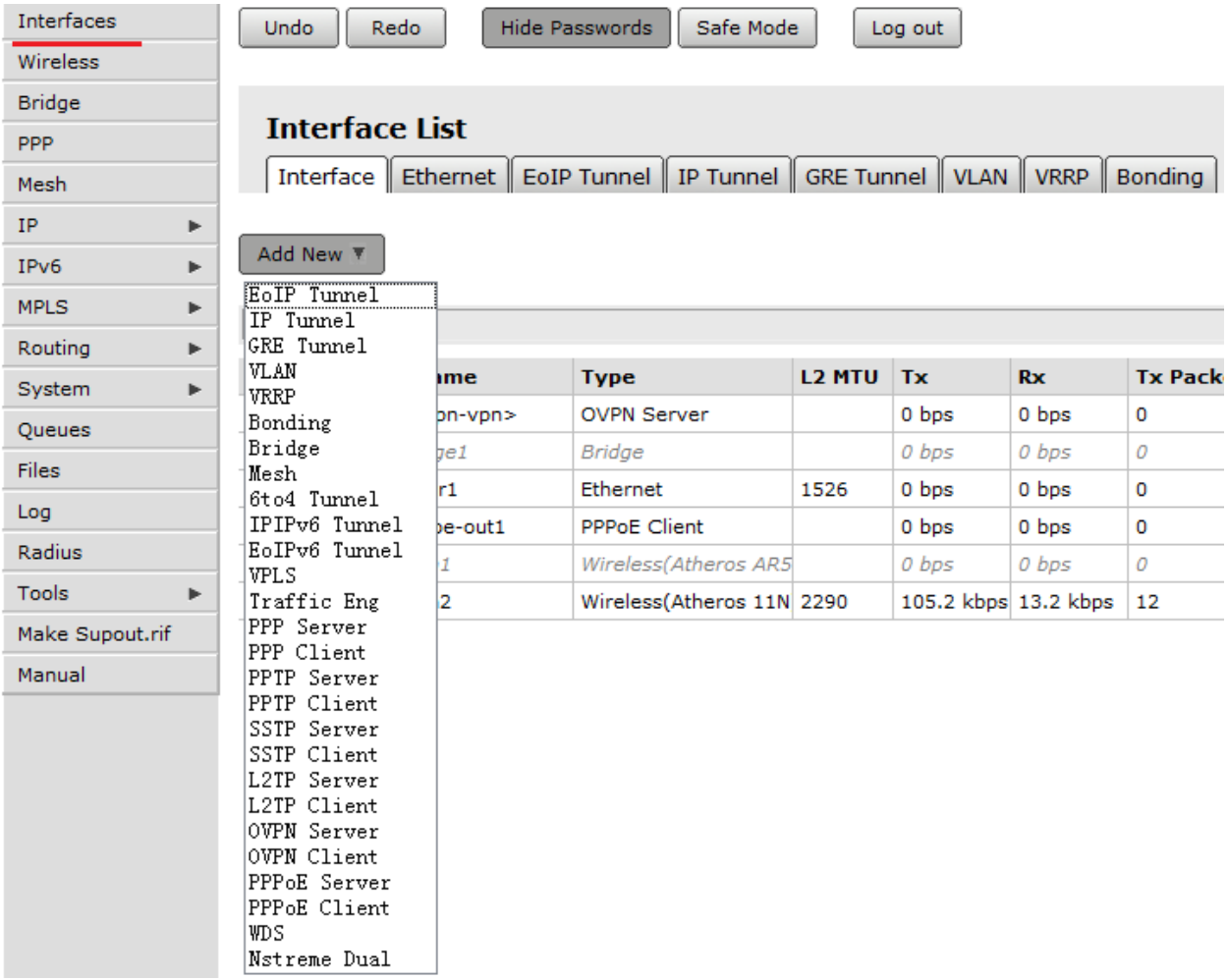


在下面的项目列表里，第一栏我们可以看到相应的字符按钮，他们的代表内容如下：

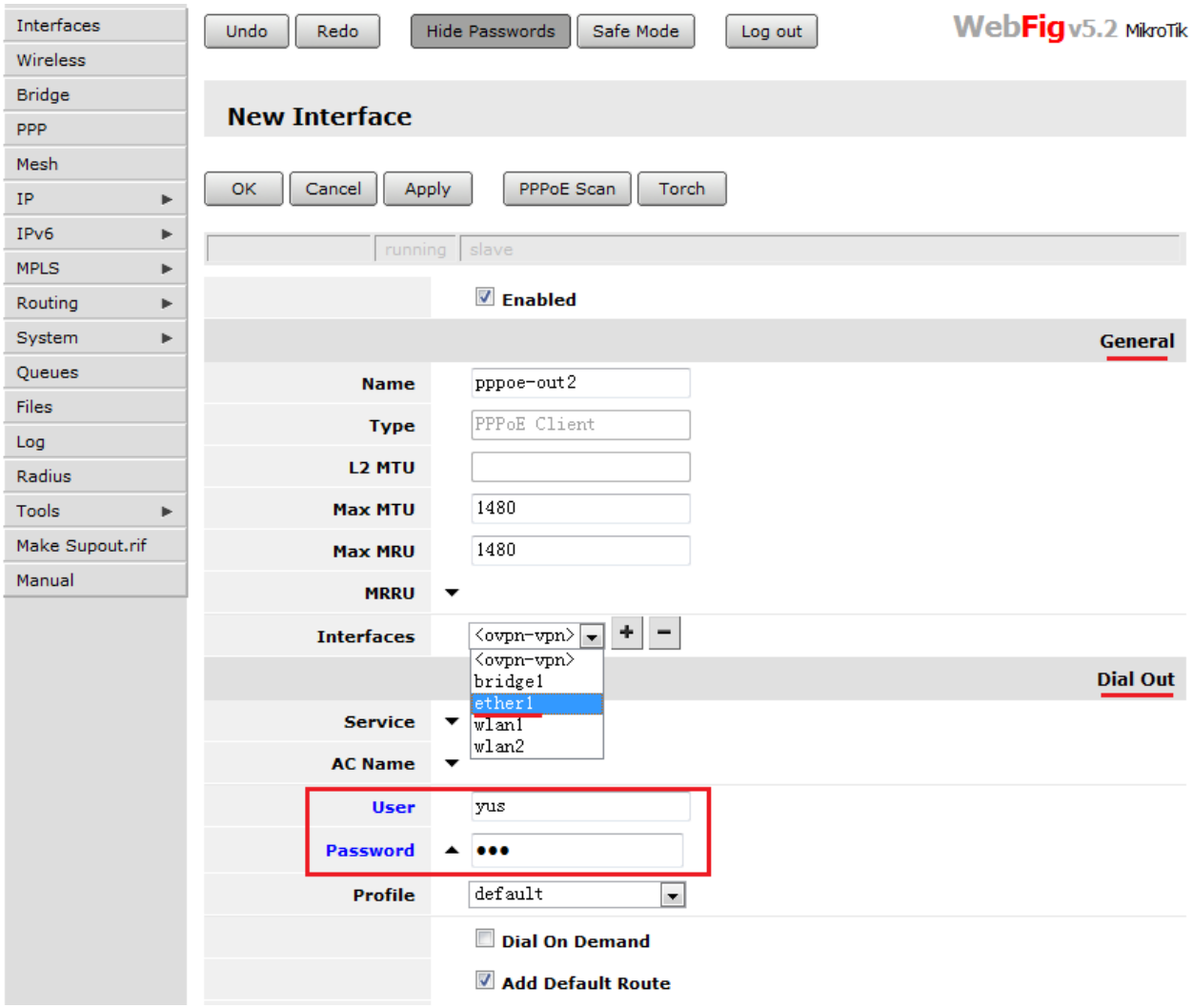
- E - 当前项目启用状态
- D - 当前项目禁用状态
- - - 删除当前项目

列表项目配置

每点击一个菜单，都会打开一个新的页面，并显示出当前菜单目录下的相应功能和参数，例如：在 interface 菜单下，并且选择 Add New 新建一个接口,这个操作如同 winbox 里的加号按钮：



现在我们在 interface 里增加一个 pppoe-client 拨号接口，这里我们可以看到 pppoe-client 拨号的配置，参数和 winbox 的相同，只是位置有所变化，比如 General 和 Dial out



这里的按钮如下：

- *Ok* – 应用修改参数并退出
- *Cancel* – 退出不做任何修改;
- *Apply* – 应用修改参数并停留在该页面
- *Remove* – 删除当前项目

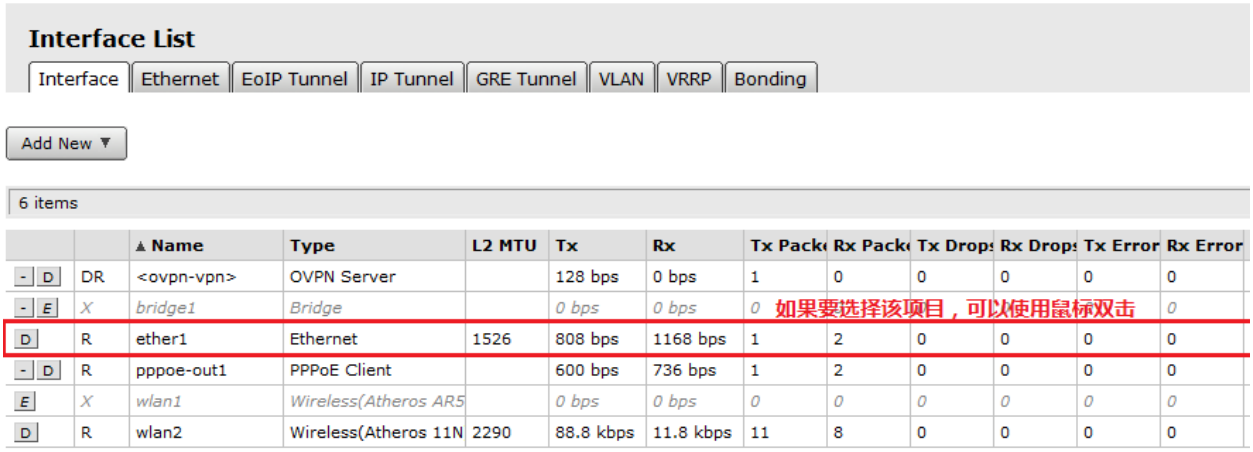
我们可以看到 pppoe-client 里后面两项 PPPoE Scan 和 Torch 按钮

- *PPPoE Scan* – 搜索当前 interface 下的 PPPoE 服务器
- *Torch* – 监测该接口下的网络流量;

当然 pppoe-client 里状态栏如同 winbox 当前的接口状态，显示了当前接口的状态，如 pppoe-out1 表示已经连接，并在运行，而灰色字体的 slave 表示并不是 switch 模式的从属网卡

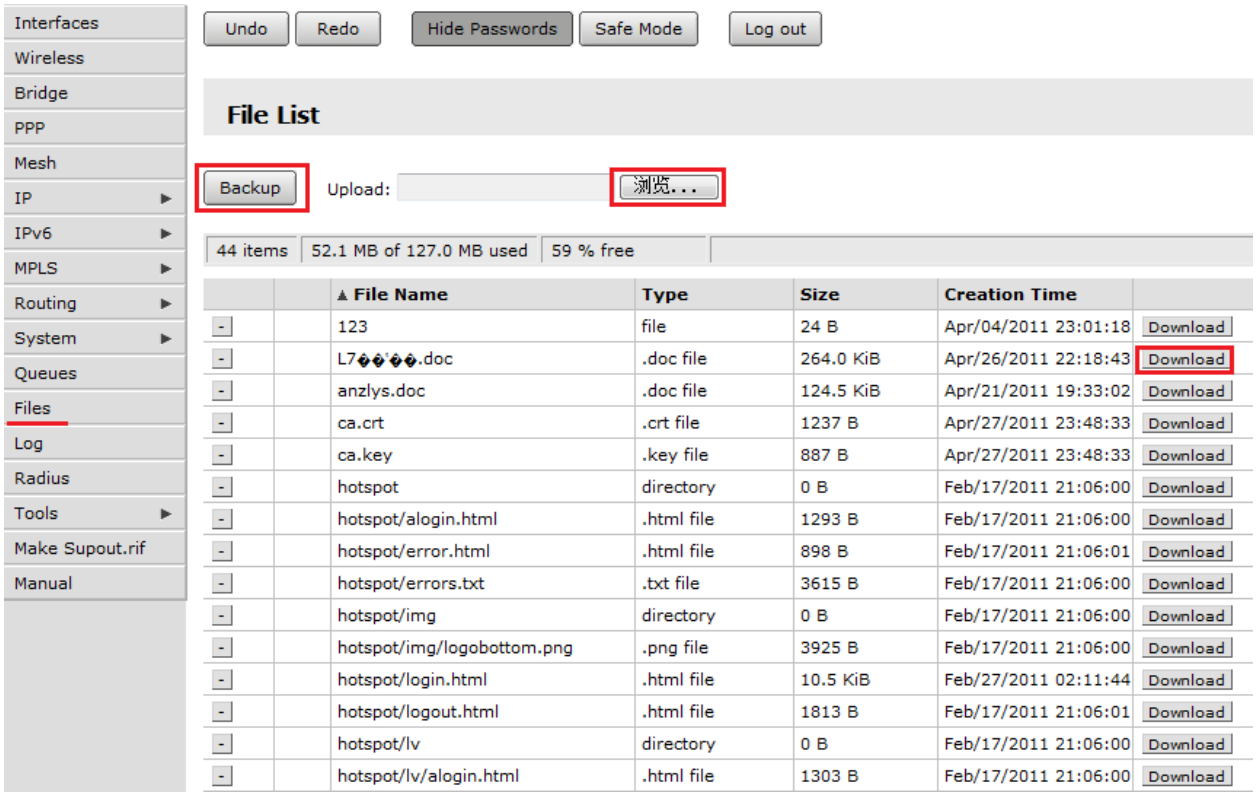
下面的图显示了 ether1 的状态，link ok 表示以太网卡网线已经接入，并且有信号。

当我们需要修改和编辑 interface 清单中的一个网卡时，我们可以通过双击鼠标选择



Files 文件操作

Webfig 允许你备份 RouterOS 配置，并且能下载和上传相应的档，如下图，我们可以选择 Backup 备份路由器的配置，在右边点的 Upload 可以选择需要上传的档，在列表中有每个档后面都有一个 Download 按钮，提供下载的功能



1.5 CLI（command Line interface）命令行操作

Console 终端控制台被用于 MikroTik 路由器的配置和管理终端，或者用于 serial port、telnet、SSH、winbox 里的终端或者直接使用显示屏加键盘操作等，Console 同样也可以用于脚本编写。通过 CLI 可以在 Console 设置和管理 RouterOS。

注：通常大部分管理员，会通过 Winbox 进行操作，CLI 命令行相对较少，但如果需要使用到 Script 脚本编写，学习 RouterOS 的 CLI 命令行操作是必须的。

路径和帮助命令

下面列表给出了 RouterOS 的基本路径操作和命令执行操作

命令	指令
..	移动到上一级目录
/	移动到根目录
/command	执行根目录中的指令
command [Enter]	执行一条指令
[?]或[F1]	显示该目录中的所有指令列表
command [?]或[F1]	显示指令的说明和变量列表
command argument [?]	显示指令的变量说明
[Tab]	使指令/字段完整，如果输入内容含糊不清，第二次键入 [Tab]会给出存在的选项

在任何操作目录使用‘？’都可用获取在当前目录中的命令信息，如下是在根目录下使用“？”。

```
[admin@MikroTik] > ?

beep – 发声脚本命令
certificate – 证书管理
console – 控制平台
delay – 延迟脚本命令
do – 执行命令
driver – 驱动管理
environment – 当前变量值列表
error – 定义错误值
execute – 在控制台下运行指定脚本
file – 路由器本地文件管理
find – 查询指定项目的值
for – for 循环脚本命令
foreach – foreach 查找脚本命令
global – 定义全局变量
if – if 判断脚本命令
import – 导入.rsc 脚本配置文件
interface – 接口配置和管理
ip – IPv4，包括 TCP/IP 协议等相关选项
ipv6 – IPv6 选项
led – LED 指示灯控制
len – len 长度脚本命令
local – 定义局部变量
log – 系统日志
nothing – 无任何操作或返回空信息
parse – 解析字符串,并返回控制面板命令
password – 修改当前管理账号密码
pick – 返回字符串或者数组值脚本命令
ping – ping 工具
port – 串口接口管理
ppp – 点对点协议
put – 打印出指定值
queue – 带宽控制管理
quit – 退出控制台
RADIUS – RADIUS 客户端设置
redo – 恢复命令
resolve – 使用本地 dns 解析域名，并返回域名解析值
return – 返回函数值
set – 修改指定项目属性
setup – 系统基本参数的向导设置
snmp -- SNMP settings
special-login – 指定登录用户
store – 存储管理
system – 系统信息和配置管理
terminal – 终端平台命令操作接口
time – 返回命令执行的时间
toarray – 转换指定内容为数组
```

```
tobool – 转换指定内容为布尔型
toid – 转换指定内容为内部编码值
toip – 转换指定内容为 IP 地址
toip6 --转换指定内容为 IPv6 地址
tonum --转换指定内容为整型值
tool – 各种诊断工具
tostr --转换指定内容为字符串
totime --转换指定内容为时间
typeof – 返回值类型
undo – 撤销命令
user – 管理员账号管理
while – while 脚本命令
export – 导出当前目录下的配置或导出为配置脚本
```

进入 IP 目录，使用“？”查看当前目录下菜单

```
 [admin@MikroTik] ip> ?

 .. – 返回上级目录
service/ -- IP 服务
socks/ -- SOCKS 4 代理
arp/ -- ARP 项目管理
upnp/ -- UPNP 管理
dns/ -- DNS 设置
address/ -- 地址管理
accounting/ -- 传输记录
proxy/ --
vrrp/ -- 虚拟路由冗余协议
pool/ -- IP 地址池
packing/ -- 数据包封装设置
neighbor/ -- 邻居
route/ -- 路由管理
firewall/ -- 防火墙管理
dhcp-client/ -- DHCP 客户端设置
dhcp-relay/ -- DHCP 中继设置
dhcp-server/ -- DHCP 服务设置
hotspot/ -- HotSpot 管理
ipsec/ -- IP 安全设置
proxy/ -- HTTP 代理
export --

[admin@MikroTik] ip>
```

在下面的例子中，你可用通过输入目录名称移动到不同的目录下。

```
[admin@MikroTik] >                  # 当前位置在根目录
[admin@MikroTik] > driver           # 输入'driver'进入到驱动管理目录中
[admin@MikroTik] driver> /          # 输入'/'从任何目录中回到根目录
```

```
[admin@MikroTik] > interface          # 输入'interface'进入接口管理目录中
[admin@MikroTik] interface> /ip       # 输入'/ip'从任何目录进入 IP 管理目录
[admin@MikroTik] ip>                  # 返回到 ip 目录
```

一个指令或一个变量参数不需要完整的输入，如果是含糊不清的指令或变量参数需要完整的输入。如输入 **interface** 时，你只要输入 **in** 或 **int**，需要显示完整的指令可以使用**[Tab]**键

通过指令的组合，可以在当前的目录执行在不同目录操作，如：

```
[admin@MikroTik] ip route> print                    查看路由表
[admin@MikroTik] ip route> .. address print         查看 IP 地址列表
[admin@MikroTik] ip route> /ip address print        查看 IP 地址列表
```

你可以使用" / "和" .. "在非当前目录下的命令操作，例如 ping 命令只能在根目录下执行：

```
[admin@MikroTik] ip route> /ping 10.0.0.1
10.0.0.1 ping timeout
2 packets transmitted, 0 packets received, 100% packet loss
[admin@MikroTik] ip firewall nat> .. service-port print
Flags: X - disabled, I - invalid
 #    NAME                                                    PORTS
 0    ftp                                                      21
 1    tftp                                                     69
 2    irc                                                      6667
 3    h323
 4    sip
 5    pptp
[admin@MikroTik] ip firewall nat>
```

基本操作命令

RouterOS 的对各个参数的添加，修改、删除、查找和移动下面的基本命令适用于所有操作目录下，

- **add** – 该命令作为规则添加，与 set 命令有相同处，除了不用指定规则条目编号。通常在有相关序列的规则中，会添加到该项目规则的最后，例如 simple queue 和 ip firewall filter。在使用 add 命令添加规则时，某些只能必须指定，某些则可以默认，例如添加网关时候，必须指定 gateway 属性，其他参数可以默认。
  - 通用参数
    - *copy-from* – 拷贝存在的规则条目，创建新规则时可以借用已经存在的规则条目属性，复制到新的规则。如果被拷贝项有规则名称，通常你必须为新的拷贝规则设置一个规则名称。
    - *place-before* – 指定一个新规则在已存在规则序列之前，因此无需在添加规则后使用 move 命令移动规则
    - *disabled* – 通过该命令控制新规则的启用和禁用状态
    - *comment* – 设置新规则的备注属性描述
- **find** – 查找命令，用于相关属性查找，find 命令调用参数与 set 相同，额外加上一些参数如 disabled 或 active，他们返回属性是 yes 或 no，这些作为标记属性根据不同目录下的情况确定，可以通过 print 命令输出顶部标识确认他们，find 命令查找到的内容，返回的是所有条目的内部编号值，而非 print 所看到的条目编号。
- **move** – 修改条目在列表中的顺序

  - 参数
    - 第一个参数指定被移动条目编号 first argument specifies the item(-s) being moved.
    - 第二个参数指定移动到那个条目编号之前
- **print** – 查看并显示指定功能菜单下的参数，类似与 cisco 的 show 和华为的 display。例如：**/system clock print** ：显示系统日期和时间；**/ip route print** ：显示所有路由条目信息。
  - 通用参数
    - *from* – 从列表中仅显示指定条目
    - *where* – 仅显示给定条件的条目，该语法类似与 find 命令查找指定属性
    - *brief* – 强制显示命令用于偏平化输出
    - *detail* – 详细显示属性参数形式
    - *count-only* – 显示列表下的条目数量
    - *file* – 显示列表下的内容，并指定文件名称，以文本形式存入路由器的 Files 目录下
    - *interval* – 指定连续显示间隔周期，间隔时间单位为秒
    - *oid* – 查看当前菜单下那些属性具有 SNMP OID 值
    - *without-paging* – 不分页显示，命令执行后，连续显示参数直到结束
- **remove** – 从一个功能菜单下，通过 print 命令显示列表，并从中删除指定条目
- **set** – 允许修改指定条目的属性参数。有部分参数需要清除而非修改，通过相反的 **unset** 命令。

下面是一个简单的 CLI 操作命令，例如你可以通过**/ip route print** 命令查看路由表：

```
[admin@MikroTik] > ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        G GATEWAY          DIS   INTERFACE
 0 A S  0.0.0.0/0                                r 10.0.3.1          1      bridge1
 1 ADC  1.0.1.0/24          1.0.1.1                                  0      bridge1
 2 ADC  1.0.2.0/24          1.0.2.1                                  0      ether3
 3 ADC  10.0.3.0/24         10.0.3.144                               0      bridge1
 4 ADC  10.10.10.0/24       10.10.10.1                               0      wlan1
[admin@MikroTik] >
```

IP 地址配置，可以进入 ip address 菜单进行

```
[admin@MikroTik] > ip address
[admin@MikroTik] /ip address>
[admin@MikroTik] /ip address>/ip address add address 10.0.0.1/24 interface ether1
```

我们也可以不用进入 ip address 目录，可以直接在根目录下进行配置，

```
[admin@MikroTik] >/ip address add address 10.0.0.1/24 interface ether1
```

在配置 IP 地址中，配置'address'和'netmask'参数时，可以将 IP 地址和子网掩码通过“/”连接一起定义，也可以将子网掩码单独定义，这两种方式是相同的，例如下面的两个输入是等价的：

```
[admin@MikroTik] >/ip address add address 10.0.0.1 netmask 255.255.255.0 interface ether1
```

在配置 IP 地址和路由前，如果你有即插即用网卡安装到路由器中，请检查/interface 中的接口列表，多数情况下设备驱动会自动安装，并且相关的接口信息会显示在**/interface print** 列表中，例如：

```
[admin@MikroTik] interface> print
Flags: X - disabled, D - dynamic, R - running
 #     NAME                          TYPE            RX-RATE    TX-RATE    MTU
 0   R ether1                       ether           0          0          1500
 1   R ether2                       ether           0          0          1500
 2 X   wavelan1                     wavelan         0          0          1500
 3 X   prism1                       wlan            0          0          1500
[admin@MikroTik] interface>
```

当某设备的网卡被禁用，你可以通过**/interface enable name** 命令启用网卡，例如：

```
[admin@MikroTik] interface> print
Flags: X - disabled, D - dynamic, R - running
 #     NAME                          TYPE            RX-RATE    TX-RATE    MTU
 0 X   ether1                       ether           0          0          1500
 1 X   ether2                       ether           0          0          1500
[admin@MikroTik] interface> enable 0
[admin@MikroTik] interface> enable ether2
[admin@MikroTik] interface> print
Flags: X - disabled, D - dynamic, R - running
 #     NAME                          TYPE            RX-RATE    TX-RATE    MTU
 0   R ether1                       ether           0          0          1500
 1   R ether2                       ether           0          0          1500
[admin@MikroTik] interface>
```

禁用命令是 disable，禁用 ether2 网卡

```
[admin@MikroTik] interface> disable ether2
[admin@MikroTik] interface> print
Flags: X - disabled, D - dynamic, R - running
 #     NAME                          TYPE            RX-RATE    TX-RATE    MTU
 0   R ether1                       ether           0          0          1500
 1   X ether2                       ether           0          0          1500
[admin@MikroTik] interface>
```

接口的名称能通过**/interface set** 指令来改变其参数：

```
[admin@MikroTik] interface> set ether1 name=Local; set ether2 name=Public
[admin@MikroTik] interface> print
Flags: X - disabled, D - dynamic, R - running
 #     NAME                          TYPE            RX-RATE    TX-RATE    MTU
 0   R Local                        ether           0          0          1500
 1   R Public                       ether           0          0          1500
[admin@MikroTik] interface>
```

通过 **add** 命令添加规则，如添加 IP 地址操作：

```
[admin@Office] /ip address> prin
Flags: X - disabled, I - invalid, D - dynamic
 #    ADDRESS                NETWORK             BROADCAST         INTERFACE
 0    10.200.15.1/24        10.200.15.0         10.200.15.255     lan
 1 D 222.212.60.227/32   222.212.48.1        0.0.0.0           ADSL
[admin@Office] /ip address> add address=192.168.10.1/24 interface=lan
[admin@Office] /ip address> prin
Flags: X - disabled, I - invalid, D - dynamic
 #    ADDRESS              NETWORK                BROADCAST         INTERFACE
 0    10.200.15.1/24      10.200.15.0          10.200.15.255       lan
 1 D 222.212.60.227/32   222.212.48.1        0.0.0.0             ADSL
 2    192.168.10.1/24     192.168.10.0         192.168.10.255     lan
[admin@Office] /ip address>
```

通过 **remove** 命令删除不需要的规则

```
[admin@Office] /ip firewall filter> prin
Flags: X - disabled, I - invalid, D - dynamic

 0 X chain=forward action=drop layer7-protocol=OICQ

 1 X chain=forward action=drop dst-address-list=OICQ

 2 X chain=forward action=log log-prefix=""
[admin@Office] /ip firewall filter> remove 2
[admin@Office] /ip firewall filter> prin
Flags: X - disabled, I - invalid, D - dynamic
 0 X chain=forward action=drop layer7-protocol=OICQ

 1 X chain=forward action=drop dst-address-list=OICQ
[admin@Office] /ip firewall filter>
```

通过 **move** 移动规则的前后顺序，例如将 2 规则移动到 0，即第三条规则优先到第一规则执行：

```
[admin@MikroTik] /ip firewall mangle> print
Flags: X - disabled, I - invalid, D - dynamic
 0    chain=output action=mark-routing new-routing-mark=route1 passthrough=yes
      connection-mark=pcc1

 1    chain=output action=mark-routing new-routing-mark=route2 passthrough=yes
      connection-mark=pcc2

 2    chain=output action=mark-routing new-routing-mark=route3 passthrough=yes
      connection-mark=pcc3
[admin@MikroTik] /ip firewall mangle> move 2 0
```

Tab 快速输入

在 console 控制台有两种方式说明快速而简单的输入命令，通过[Tab]键完成和通过缩写输入命令。这样的操作类似于 UNIX 的 shell，如果你在一个半截单词后按[Tab]键，Console 控制台试着找到当前关联的命令。如果仅有一个命令匹配，将自动显示完全，如下：

```
/inte[Tab]_ 变为 /interface _
```

如果有多个匹配命令参数

```
/interface set e[Tab]_  变为  /interface set ether_
```

如果你仅输入了共有部分的命令，按了一下 tab 键没有效果，当你连续按下第二次 tab 键后，将显示所有可能的参数和命令

```
[admin@MikroTik] > interface set e[Tab]_
[admin@MikroTik] > interface set ether[Tab]_
[admin@MikroTik] > interface set ether[Tab]_
ether1 ether5
[admin@MikroTik] > interface set ether_
```

[Tab]键能被用于提示任何关联内容，例如在控制台下的命令、参数,对应的参数仅有几个可能的值。

另外一种方式是，输入少数的命令字符，即缩写命令和参数。你可以输入命令开始的字符，当然输入的命令字符，要与指定的命令相对应，例如 ping 命令 10.0.0.1，连续 3 次，大小 100byte

```
[admin@MikroTik] > pi 10.1 c 3 si 100
```

等同于：

```
[admin@MikroTik] > ping 10.0.0.1 count 3 size 100
```

也可以不用从一个命令的起始字符输入，可以输入当前菜单下命令的关键词符，console 会自动查找到对应的命令和参数，例如以下操作：

```
[admin@MikroTik] > interface x[TAB]_
[admin@MikroTik] > interface export _
[admin@MikroTik] > interface mt[TAB]_
[admin@MikroTik] > interface monitor-traffic _
```

快速设置 Setup

当初始化路由器时，通过使用**/setup** 指令设置下列配置内容：

- 重新设置路由器配置
- 加载接口驱动
- 配置 IP 地址和网关

- 设置 DHCP 客户端
- 设置 DHCP 服务端
- 设置 pppoe 客户端
- 设置 pptp 客户端

使用 Setup 指令，在路由器上配置 IP 地址，执行**/setup** 指令行：

```
[admin@MikroTik] > setup
  Setup uses Safe Mode. It means that all changes that are made during setup
are reverted in case of error, or if Ctrl-C is used to abort setup. To keep
changes exit setup using the 'x' key.
[Safe Mode taken]
  Choose options by pressing one of the letters in the left column, before
dash. Pressing 'x' will exit current menu, pressing Enter key will select the
entry that is marked by an '*'. You can abort setup at any time by pressing
Ctrl-C.
Entries marked by '+' are already configured.
Entries marked by '-' cannot be used yet.
Entries marked by 'X' cannot be used without installing additional packages.
    r - reset all router configuration
 + l - load interface driver
 * a - configure ip address and gateway
    d - setup dhcp client
    s - setup dhcp server
    p - setup pppoe client
    t - setup pptp client
    x - exit menu
your choice [press Enter to configure ip address and gateway]: a
```

配置 IP 地址和网关，输入 **a** 或[Enter]

```
 * a - add ip address
 - g - setup default gateway
    x - exit menu
your choice [press Enter to add ip address]: a
```

选择 **a** 添加一个 IP 地址，首先，设置程序将要询问你选择那一个接口添加 IP 地址，如果设置程序没有指定出，合适的接口，可以通过键入[Tab]两次，查看可选的接口。 在接口选择后，分配 IP 地址和子网掩码：

```
your choice: a
enable interface:
ether1   ether2   wlan1
enable interface: ether1
ip address/netmask: 10.1.0.66/24
#Enabling interface
/interface enable ether1
#Adding IP address
/ip address add address=10.1.0.66/24 interface=ether1 comment="added by setup"
```

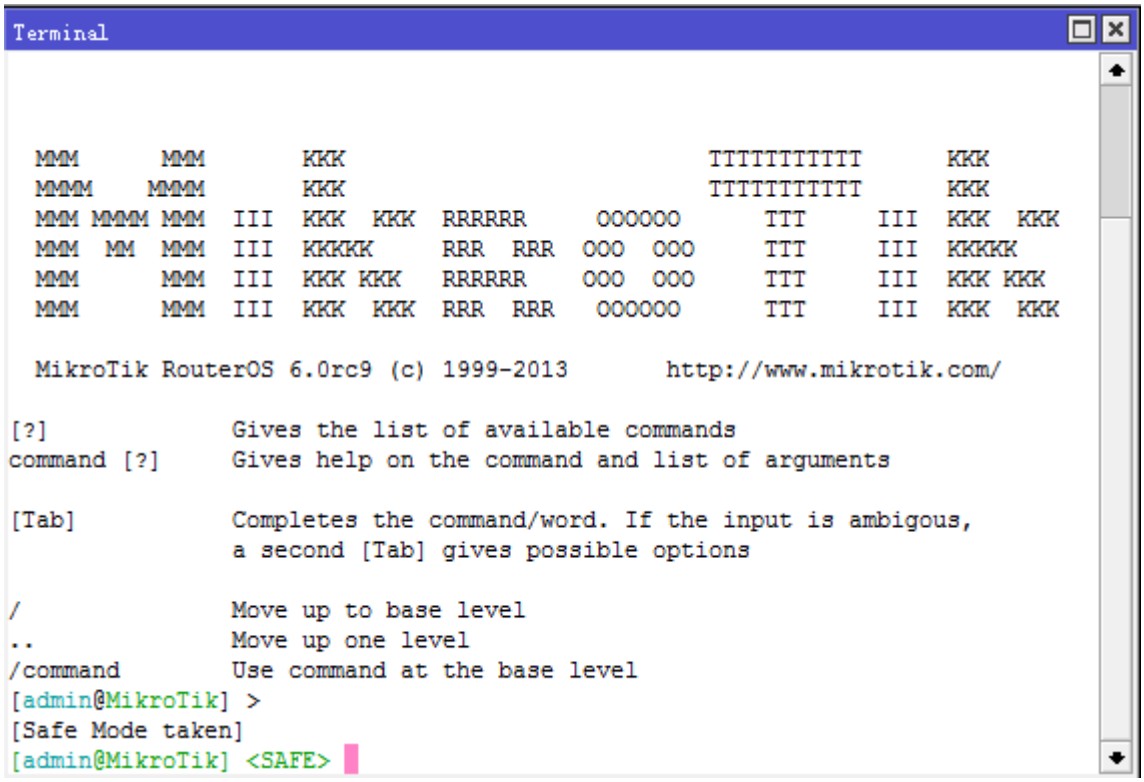
```
 + a - add ip address
 * g - setup default gateway
   x - exit menu
your choice: x
```

1.6 安全模式

通常情况下我们是通过网络连接到 RouterOS，并操作修改配置，然而错误的配置，会造成路由器不能访问（除了本地终端控制），但这个时候已经不能使用 undo 撤销来还原操作，在此时已经与路由器断开连接。为了将这一的风险降低到最低，我们可以使用 Safe 模式。

在命令行 Safe 模式通过组合键**[Ctrl]+[X]**开启，退出 safe 模式，再一次按**[Ctrl]+[X]**。

```
[admin@MikroTik] ip route>[Ctrl]+[X]
[Safe Mode taken]

[admin@MikroTik] ip route<SAFE>
```



当在命令行开启 Safe 模式，**Safe Mode taken** 消息提示在屏幕上，所有配置修改都会被执行，虽然路由器在 Safe 模式下，但如果 Safe 模式连接异常中断，Safe 模式下的配置将自动解除，你可以看到所有的配置在历史记录里都会有一个 F 标记，如下，我们添加了一个默认网关，然后进入/system history 查看，能看到 route added 前有一个 F 标记：

```
[admin@MikroTik] ip route>
[Safe Mode taken]

[admin@MikroTik] ip route<SAFE> add gateway=1.1.1.1
```

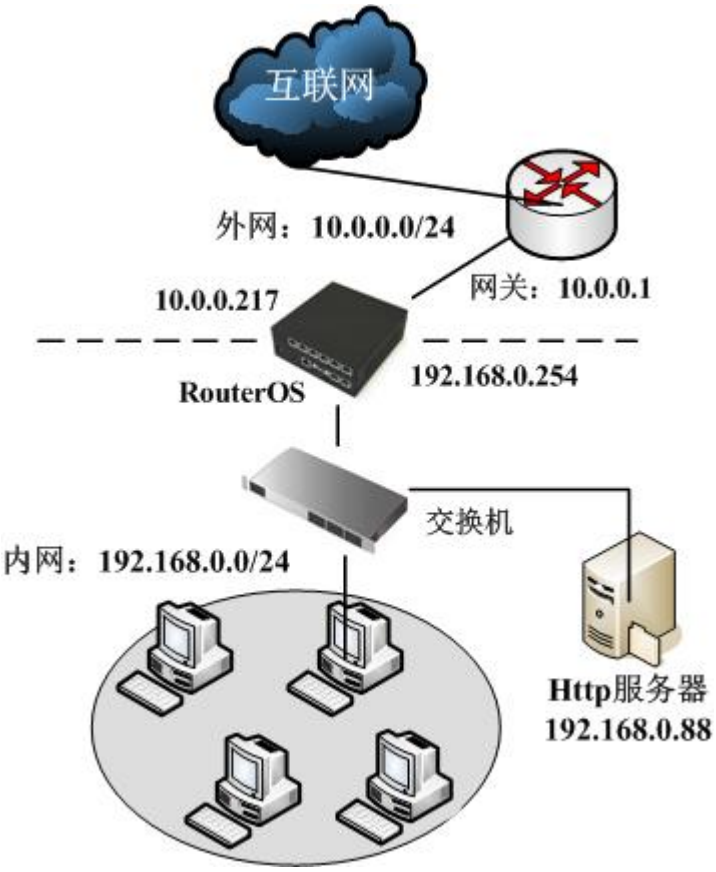
```
[admin@MikroTik] ip route<SAFE> /system history print
Flags: U - undoable, R - redoable, F - floating-undo
  ACTION                                   BY                POLICY
F route added                              admin             write
```

现在，如果 telnet 或者 winbox 连接中断，超时 9 分钟后，所有在 Safe 模式下的配置修改将会被自动解除。退出 Safe 模式可以使用 **[Ctrl]+[D]**

如果在 Safe 模式下做了太多修改，没有足够的空间存放历史记录（历史记录可以存储 100 条执行命令），这时连接将退出 Safe 模式，不会修改并自动解除，因此，最好在 Safe 模式下做一些小范围的修改

1.7 RouterOS 简单网络配置事例

通过下面一个简单的网络拓扑作为配置实例，根据该网络需要通过 RouterOS 完成配置（推荐 hAP ac2 单线接入上网配置视频）：



在当前的实例中我们涉及到两个网络，即接入 ISP 网络的 WAN 外网和 LAN 内网：

- LAN 内网地址段：192.168.0.0/24。路由器的地址在这个网络中为 192.168.0.254
- 接入 ISP 的外网 WAN 为 10.0.0.0/24。路由器的地址是在网络中为 10.0.0.217
- 外网 DNS 为 61.139.2.69，202.98.68.96

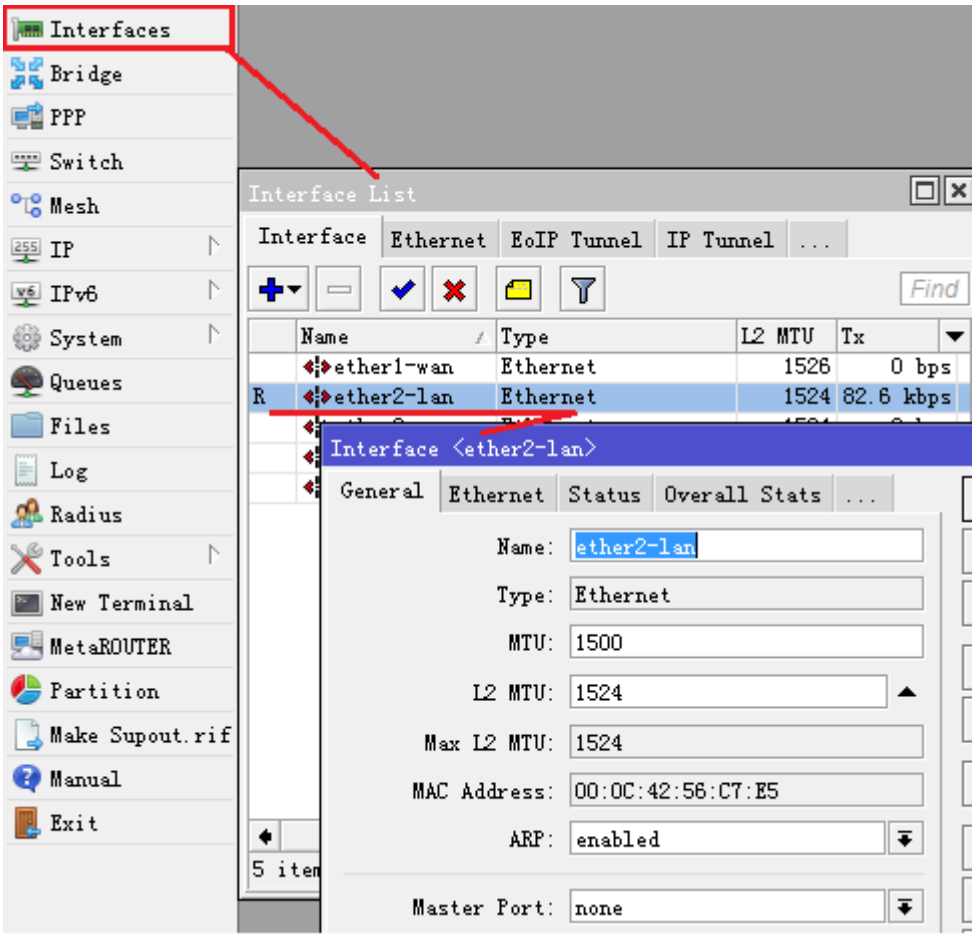
配置 RouterOS 实现内网访问 ISP 网络，我们的步骤一共分为五步

- 首先：启动设备后，检查 interface 接口网口连接是否正常，并定义网口名称
- 第二：配置对应网口的 IP 地址
- 第三：配置默认网关路由

- 第四：配置 nat 地址转换规则
- 第五：配置 DNS 服务器

第一步：网络接口配置

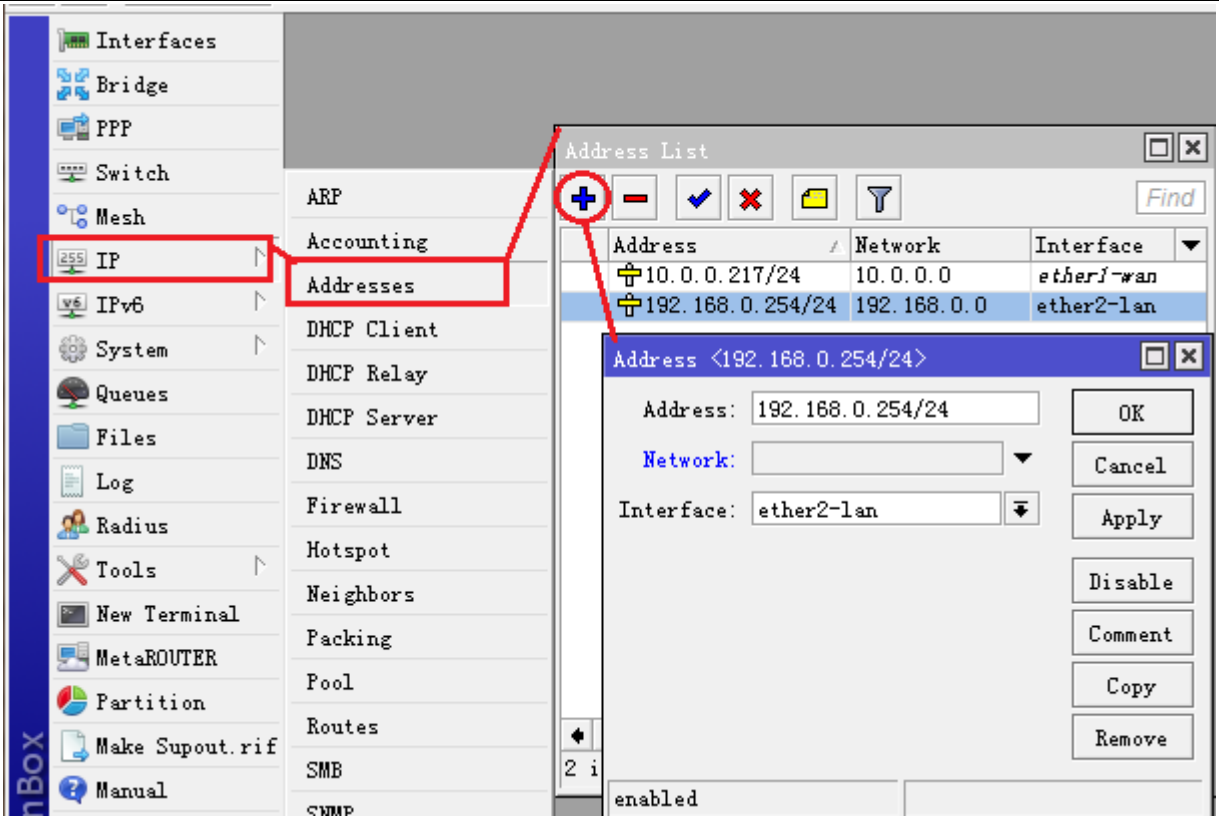
在/interfaces 列表中修改 ether2 为 ether2-lan 定义为内网接口，如图：



同样将 ether2 修改为 ether2-lan，指定内网接口，同理将 ether1 也设置为 ether1-wan，以此区分接口。

第二步：添加 IP 地址

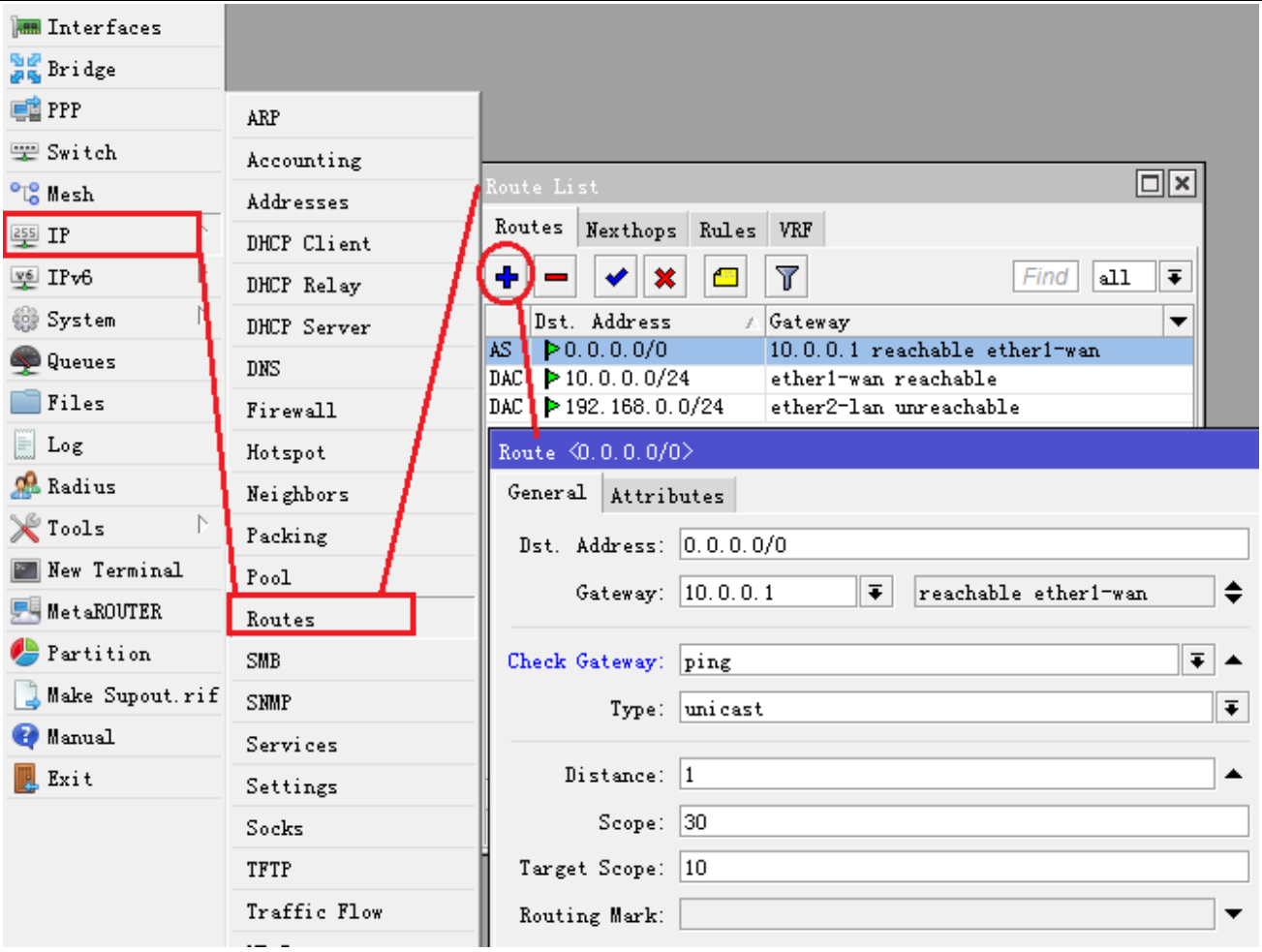
在/ip address 中添加 IP 地址和选择网卡接口，添加内网和外围的 IP 地址如图：



同理在 ip address 菜单下，添加 ether1-wan 口的 IP 地址。

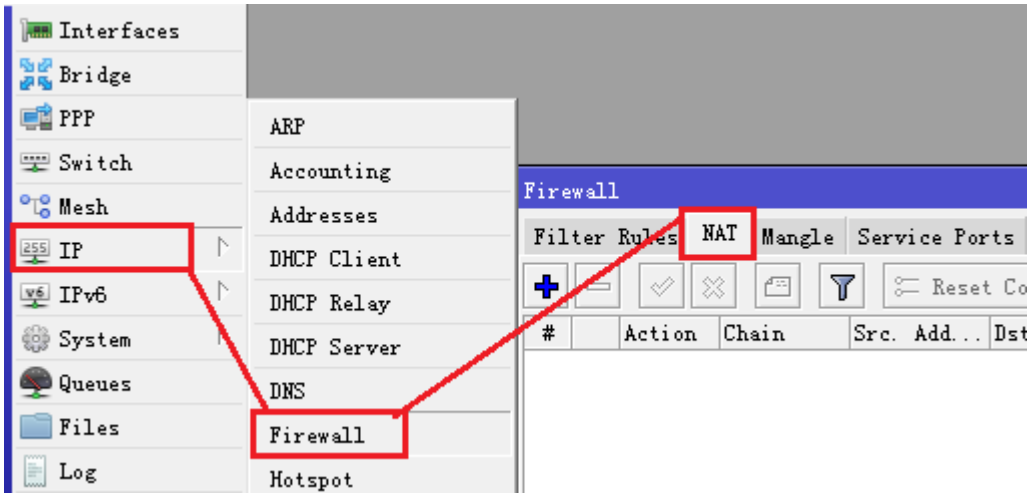
第三步：添加默认网关

在/ip routes 里添加默认网关 10.0.0.1，开启 check-gateway=ping（网关 ping 监测）如图：

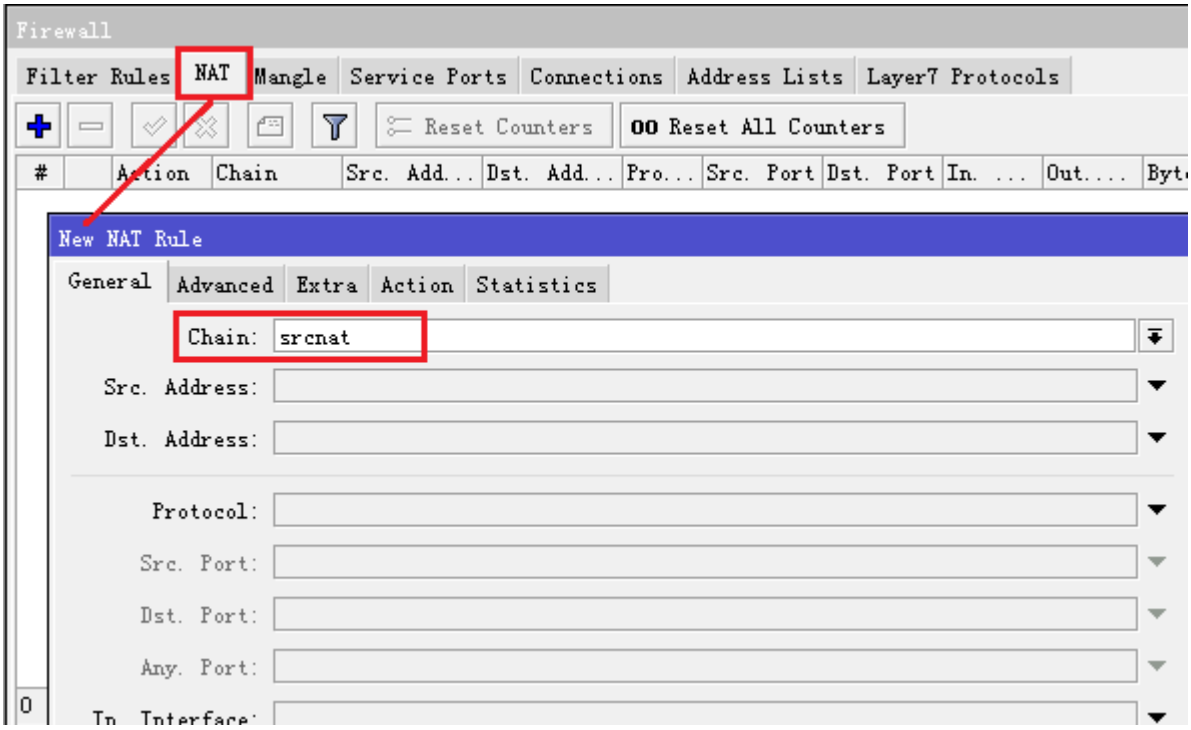


第四步，NAT 地址转换

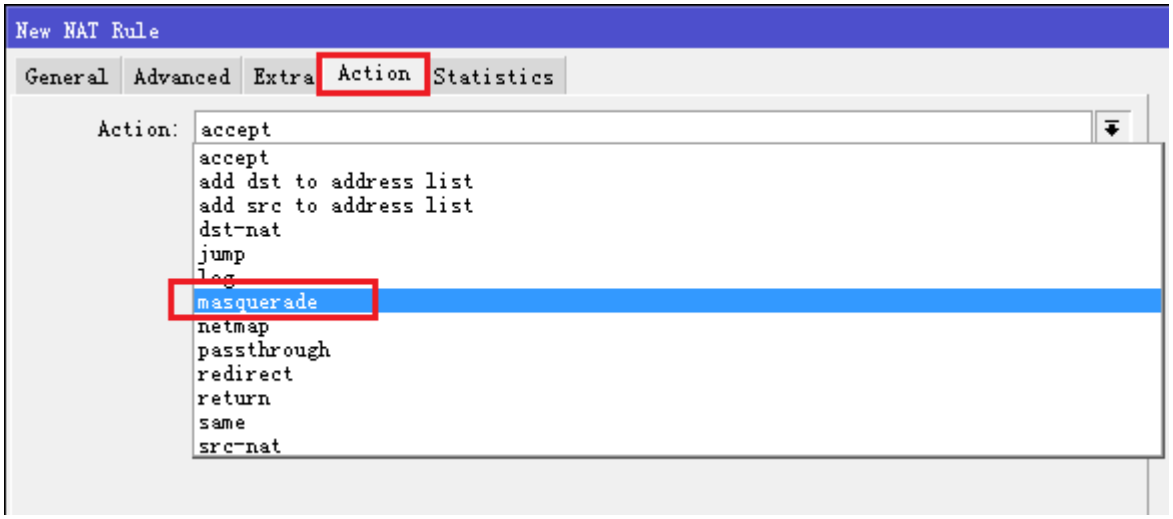
配置 nat 转换，用于隐藏局域网 IP 地址，能通过 WAN 口访问外网，在/ip firewall nat 下点击“+”添加伪装规则：



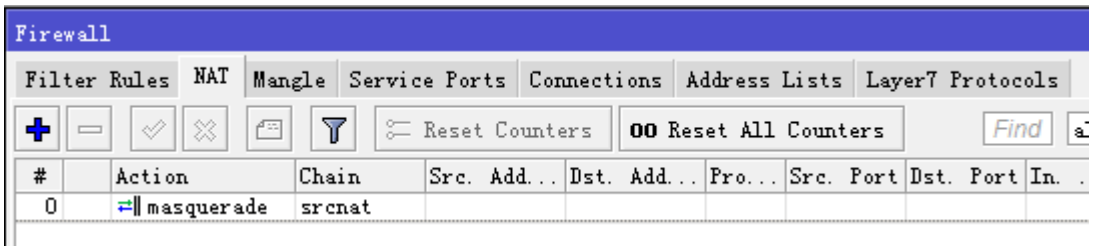
在 NAT 里添加新的规则，在 chain 里选择 srcnat 链表：



再选择 action 里的 action=masquerade 规则：



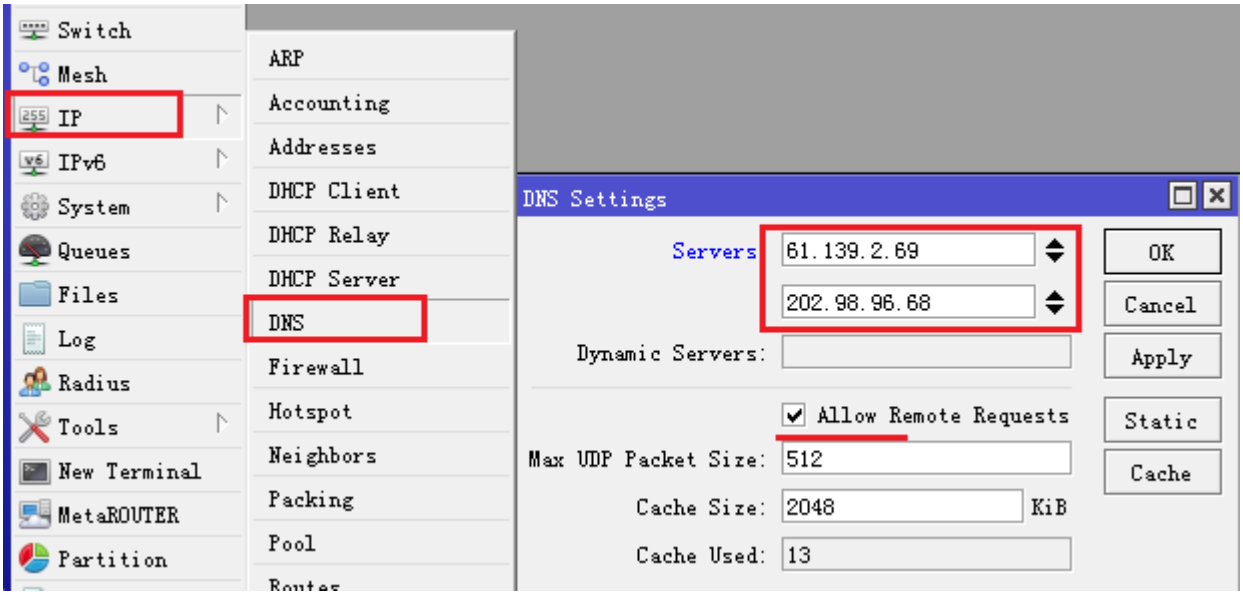
添加完成后：



对于初学者，这样的 srcnat 配置是可以正常上网，但不太推荐，具体原因请看：[正确配置 nat 规则接口参数]

第五步，DNS 配置

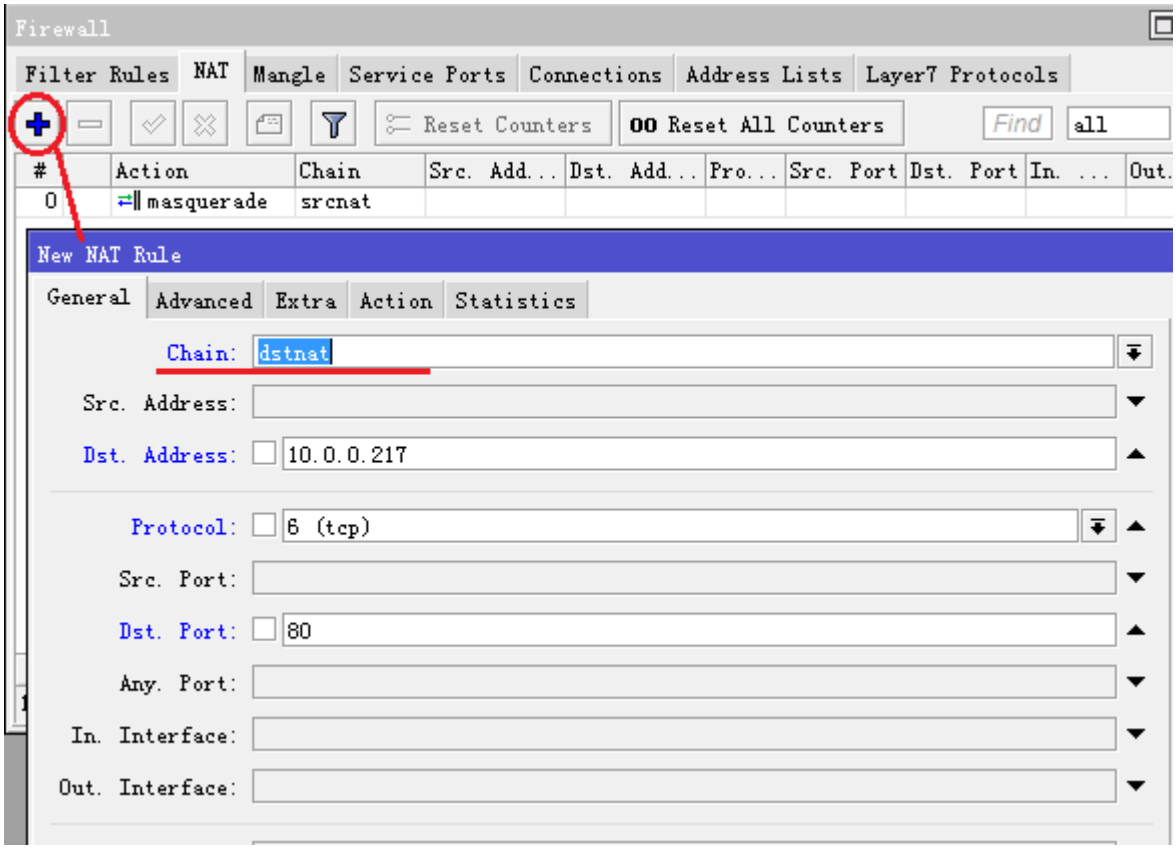
在/ip dns 的 settings 中添加多个 DNS 服务器地址，根据需要启用 DNS 缓存（allow remote requests），并能通过 Cache size 修改 DNS 缓存大小：



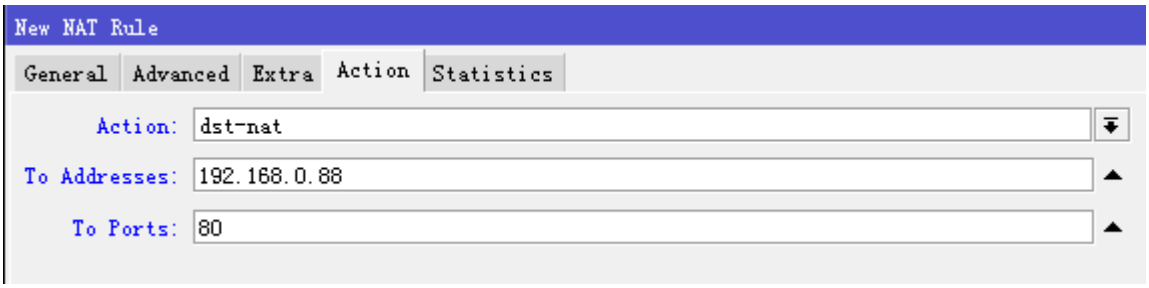
到此，上述的单在线网事例就已经配置完成！

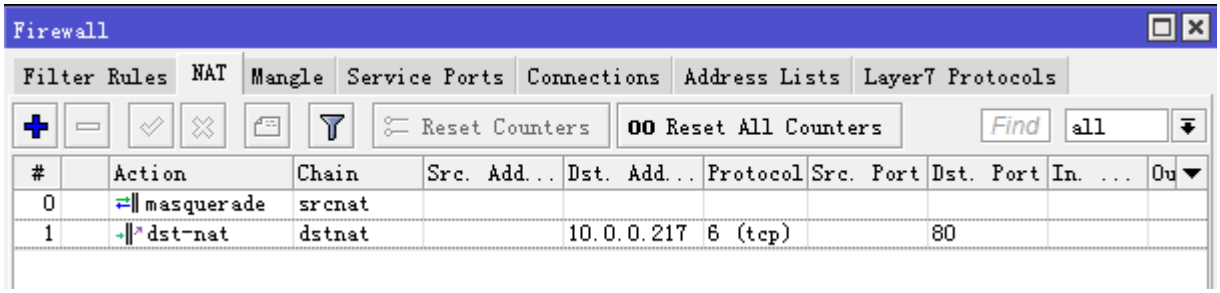
端口映射

根据以上网络拓扑，需要将内网的 http 服务器发布到外网，内网的 http 服务器 IP 地址 192.168.0.88，这里需要做端口映射规则，进入 ip firewall nat 里，选择 chain=dstnat，我们的外网 IP 地址是 10.0.0.217 配置到 dst-address，dst-port 为 tcp 协议 80 端口，如下图



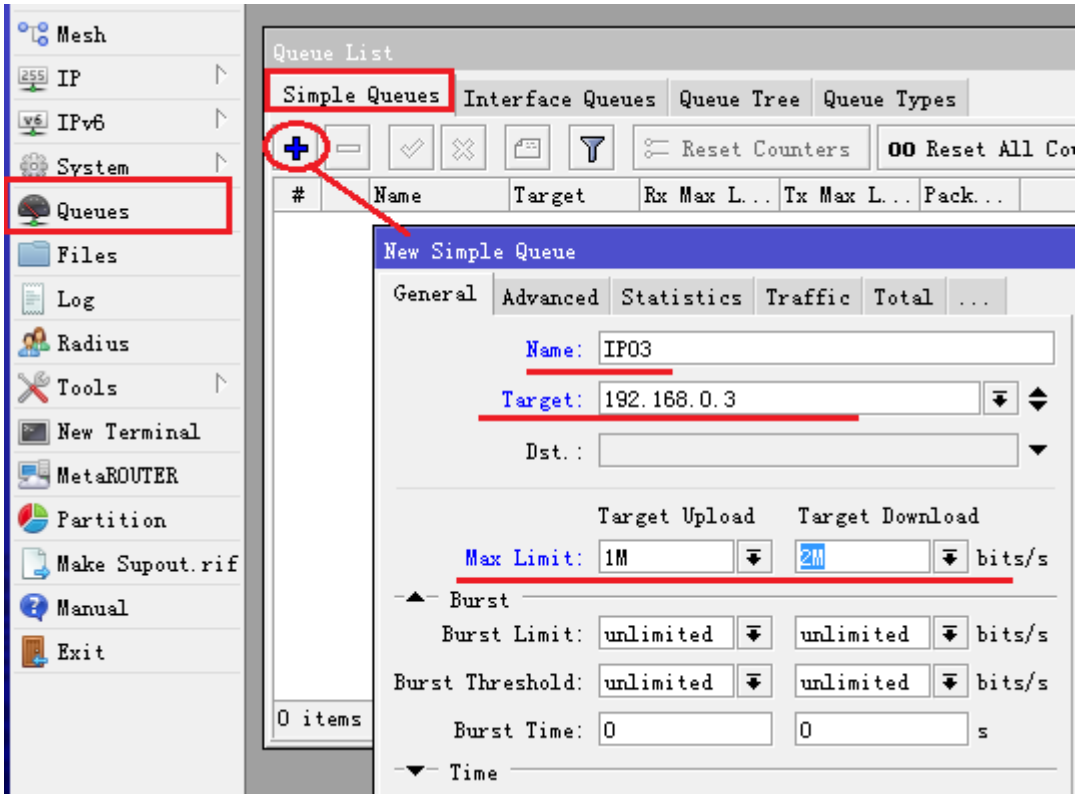
在 action 选择 dst-nat 操作，to-address 设置内网 http 服务器 IP 地址，和端口 80

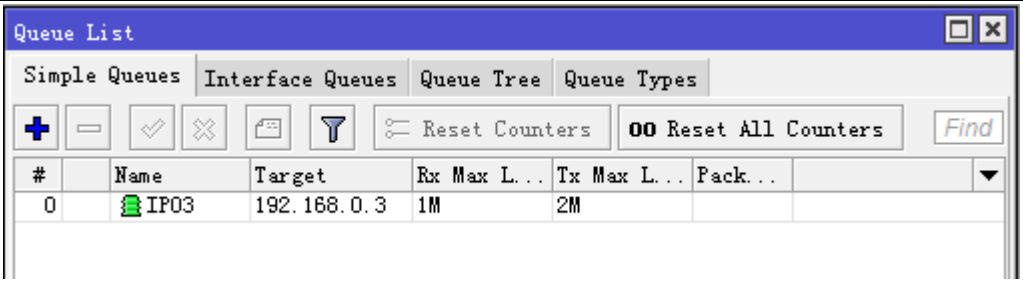




主机带宽控制

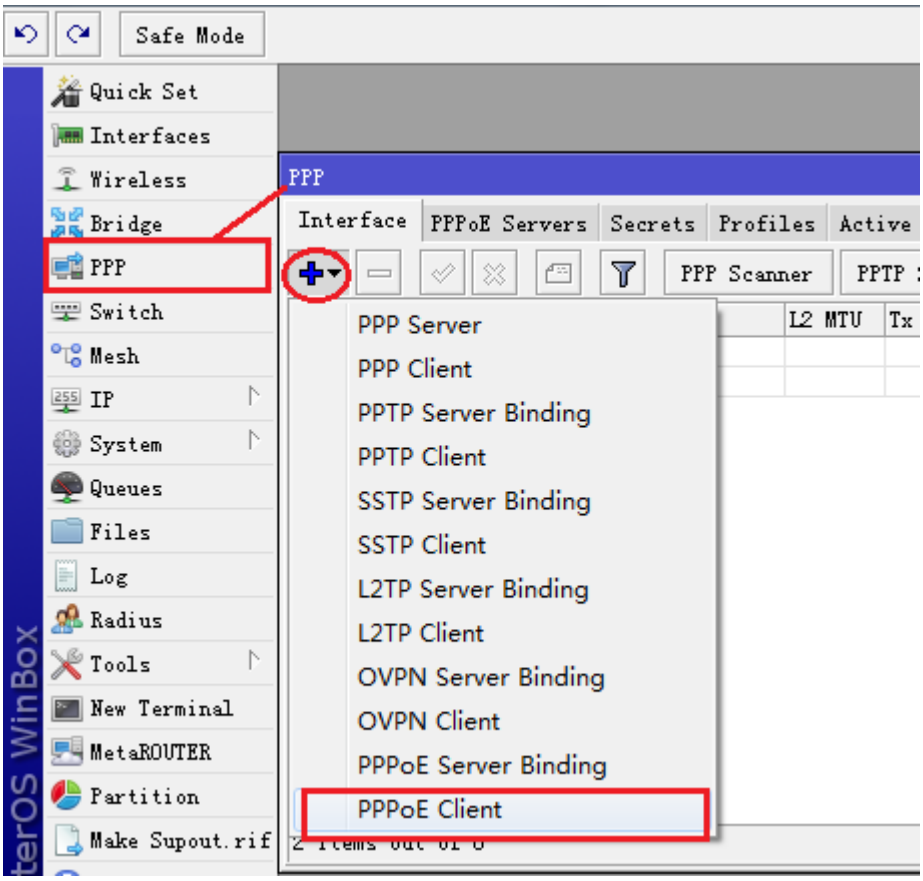
进入 Queue 添加带宽控制规则，选择 simple queue，添加主机 IP 是 192.168.0.3，并取名为 IP03，设置带宽为上行(upload)1m，下行(download)2m



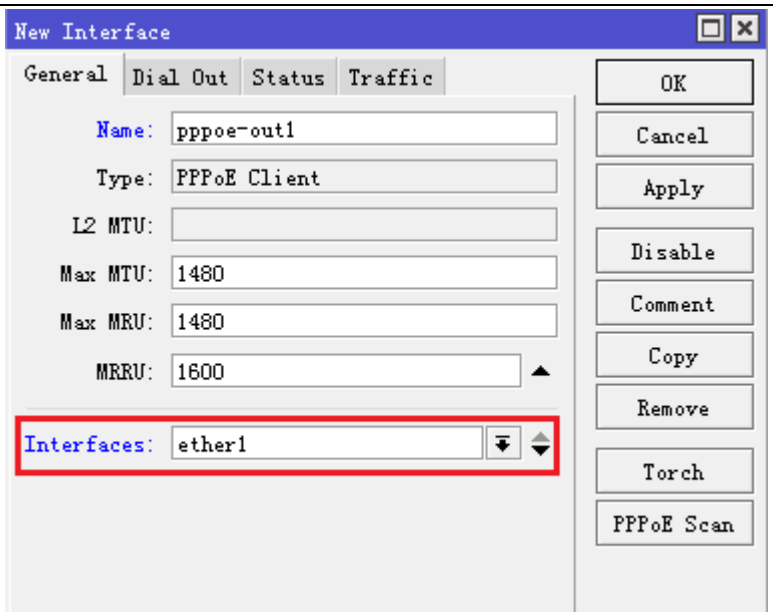


ADSL/光猫的 PPPoE 拨号配置

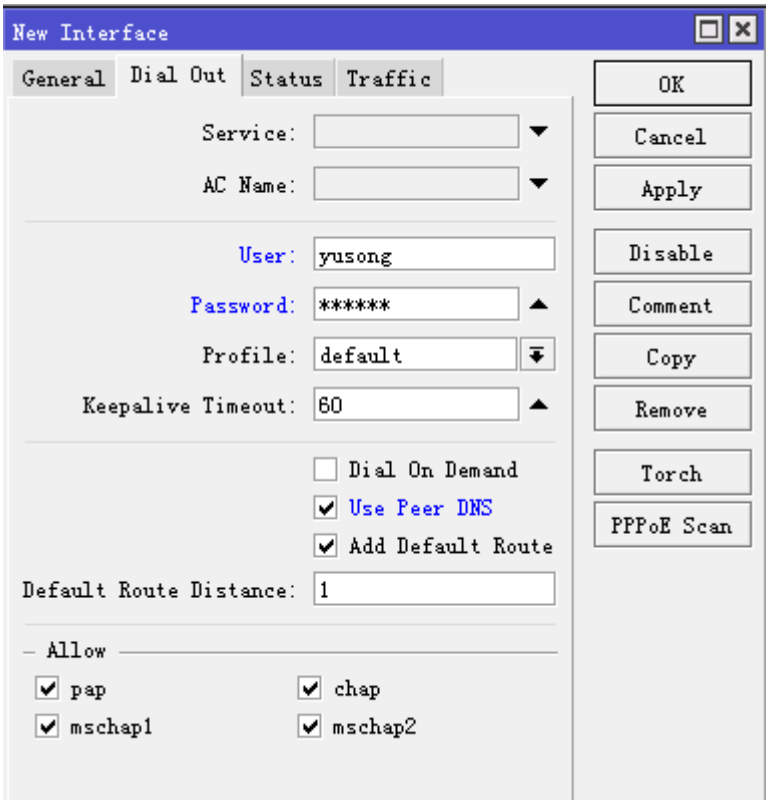
在家庭宽带接入时，如 ADSL/光猫的桥接模式拨号，都需要建立 PPPoE-client 的拨号接口，需进入 ppp 里点加号创建一个 PPPoE-client：



PPPoE-client 创建后取名 pppoe-out1，选择拨号的 interface 物理接口为 ether1



点击 Dial-out 里配置拨号参数，用户名 user 为 yusong，密码 password 为 123456，将 use-peer-dns 和 add-default-route 选择上

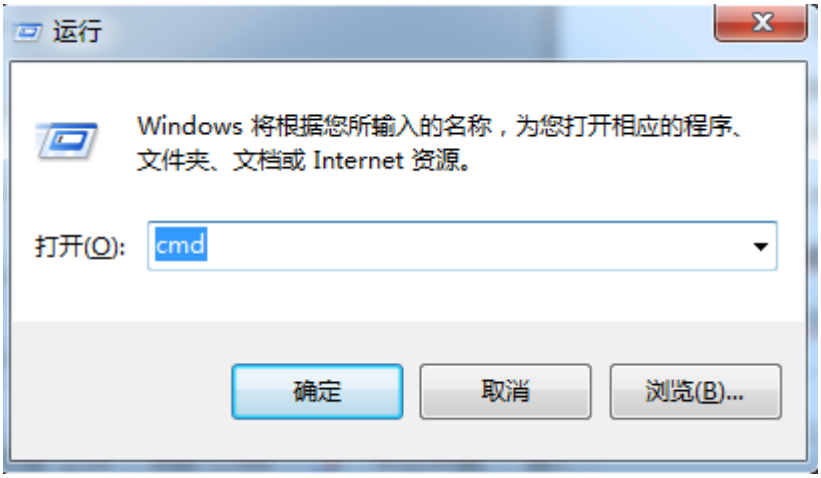


网线连接后 pppoe-out1 会自动拨号，连接成功后 pppoe-out1 会显示 R，代表拨号成功，可以进入/ip address 下查看自动获取的 ip 地址，网关也会自动添加到/ip route 中，会自动配置 DNS 到/ip dns 下

1.8 基本网络故障排查命令与方法

通常在实际的网络生产环境会遇到各种各样的问题，造成网络异常、中断和不稳定，而我们需要及时处理和排除故障，因此需要运用到很多网络排除的方法和工具。我们当前的互联网都是基于标准的 ISO 七层模型，因此在网络判断方面各个操作系统都基本相同，下面主要以 windows 系统为主，也是大家主要使用的系统，当然这些基本的故障排查工具和命令，也基本通用于 RouterOS 或 Linux 等系统，只是格式有所不同，原理是一模一样。

例如 windows，在进行网络排除时，我们一般会进入 windows 命令终端，通过开始菜单，选择“运行”，输入 cmd，回车，进入 windows 的命令终端。开始菜单没有运行选项，则可以通过 win+R 组合键开启“运行”



下面我们举例以下常用的 windows 命令：

Ping 命令

ping 命令是最常用，也是最基本的网络检测命令，管理员通过该命令去检测对端主机或设备的 IP 网络是否可达，以及测试到底主机或设备的延迟和掉包率。Ping 使用 Internet Control Message Protocol (ICMP)协议，Ping 发送 echo 请求包到目标主机，并等待该主机的 ICMP 响应。 Ping 操作后会统计最小、平均和最大的时间延迟，发送 ICMP 包的数量，接收数量和丢包情况等。

Windows 系统：

```
C:\>ping 10.255.255.4
 Pinging 10.255.255.4 with 32 bytes of data:
Reply from 10.255.255.4: bytes=32 time=1ms TTL=61
Reply from 10.255.255.4: bytes=32 time<1ms TTL=61
Reply from 10.255.255.4: bytes=32 time<1ms TTL=61
Reply from 10.255.255.4: bytes=32 time<1ms TTL=61
Ping statistics for 10.255.255.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0%
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

可以通过 ping 10.255.255.255.4 –t 连续 ping 主机，通过 Ctrl+C 停止 ping

MikroTik 系统：

```
[admin@MikroTik] > ping 10.255.255.4
10.255.255.4 64 byte ping: ttl=62 time=2 ms
10.255.255.4 64 byte ping: ttl=62 time=8 ms
10.255.255.4 64 byte ping: ttl=62 time=1 ms
10.255.255.4 64 byte ping: ttl=62 time=10 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 1/5.2/10 ms
```

通过 Ctrl+C 停止 ping

ARP 命令

APR 查询系统中缓存本地局域网的 ARP 表，ARP 表用来维护本地局域网 IP 地址与 MAC 地址的对应关系，在 windows 里也可以通 arp 命令绑定 IP 和 MAC。如果本地局域网某个主机禁止了 ping，可使用 arp -a 命令查看对方主机的 mac，确认是否在线；还可以通过 arp 来判断局域网是否内存在 arp 欺骗。

```
C:\>arp -a

接口: 192.168.70.48 --- 0x1a
  Internet 地址         物理地址              类型
  192.168.70.1          d4-ca-6d-fa-4b-43     动态
  192.168.70.59         e8-b1-fc-4c-e6-63     动态
  192.168.70.255        ff-ff-ff-ff-ff-ff     静态
```

ipconfig 命令

ipconfig 用于查看本地主机网络的 TCP/IP 配置，在命令行输入”ipconfig”

```
C:\>ipconfig

Windows IP 配置

无线局域网适配器 无线网络连接:
   连接特定的 DNS 后缀 . . . . . . . :
   IPv4 地址 . . . . . . . . . . . . : 192.168.70.48
   子网掩码 . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . : 192.168.70.1
```

ipconfig 还有一些附加命令，我们可以通过？获取附加参数的帮助提升 "ipconfig /?" 或 “ipconfig -?”。我们可以使用 ipconfig /all 查看当前本地网络适配器的所有配置信息，除了能看到 IP 地址、子网掩码和网关外，还能看到当前 PC 主机名，网卡 MAC 地址等信息。

```
C:\>ipconfig /all

Windows IP 配置

  主机名 . . . . . . . . . . . . . : Yu
  主 DNS 后缀 . . . . . . . . . . . :
  节点类型 . . . . . . . . . . . . : 混合
  IP 路由已启用 . . . . . . . . . . : 否
  WINS 代理已启用 . . . . . . . . . : 否

无线局域网适配器 无线网络连接:

   连接特定的 DNS 后缀 . . . . . . . :
   描述. . . . . . . . . . . . . . . : Intel(R) Wireless-N 7260
   物理地址. . . . . . . . . . . . . : 48-51-B7-AE-D4-51
   DHCP 已启用 . . . . . . . . . . . : 是
```

```
自动配置已启用. . . . . : 是
IPv4 地址 . . . . . : 192.168.70.48(首选)
子网掩码 . . . . . : 255.255.255.0
获得租约的时间 . . . . . : 2016 年 2 月 15 日 8:18:27
租约过期的时间 . . . . . : 2016 年 2 月 15 日 11:48:26
默认网关. . . . . : 192.168.70.1
DHCP 服务器 . . . . . : 192.168.70.1
DNS 服务器 . . . . . : 192.168.70.1
TCPIP 上的 NetBIOS . . . . . : 已启用
```

Netstat 命令

netstat 是监听并显示当前主机的 TCP 连接和端口、以太网接口统计信息、IP 路由表、统计 IP、ICMP、TCP 和 UDP 协议。同样 netstat 也有相应的附加命令参数通过"netstat –?"查看。该命令常用于本地端口查看，例如使用 netstat –n，可以查看到当前 TCP 连接状态。

Nslookup 命令

该命令是最主要的 DNS 解析测试和问题判断工具，例如你想知道 www.routeros.com 网站的当前所有 IP 地址，你可以通过“nslookup www.routeros.com”查看

```
C:\>nslookup www.routeros.com
服务器:  UnKnown
Address:  192.168.1.1

非权威应答:
名称:    www.routeros.com
Addresses:  2a02:610:7501:1000::2
          159.148.147.196
```

这里可以看到 www.routeros.com 域名包含了两个 IP 地址，一个 ipv4 和一个 ipv6

Traceroute 命令

traceroute 用于查看 IP 包到达目的主机通过路由设备列表，即通常说的路由跟踪，该工具在大型企业和 ISP 是非常实用的，能分析和判断路由是否正确，路由是否按照管理者的要求正确到达。windows 系统下采用 tracert ，而类 Unix 系统包括 traceroute 或 tracepath 等、

Traceroute 运行在 TTL 和 ICMP “Time Exceeded” 信息基础上，TTL 是限制 IP 数据报生存周期，避免在互联网中产生永久环路。IP 数据报每经过一个路由设备会自动递减 1，直到 TTL 值为 0，路由器将会丢弃收到的 TTL=0 的 IP 数据报并向 IP 包的发送者发送 ICMP time exceeded 消息

通过该命令可以看到数据报是如何通过网络，并查看到可能存在的问题，如延迟增高，中断和环路等情况，判断存在那一个节点故障。

Windows 下使用路由跟踪命令 tracert:

```
C:\>tracert 10.255.255.2
Tracing route to 10.255.255.2 over a maximum of 30 hops
  1    <1 ms    <1 ms    <1 ms  10.13.13.1
  2     1 ms     1 ms     1 ms  10.255.255.2
```

```
Trace complete.
```

在类 Unix 系统下使用路由跟踪，这里 Traceroute 与 tracepath 时相同的，只是 tracepath 不要求超级用户权限：

```
andris@andris-desktop:~$ tracepath 10.255.255.6
 1:  andris-desktop.local (192.168.10.4)                    0.123ms pmtu 1500
 1:  192.168.10.1 (192.168.10.1)                             0.542ms
 1:  192.168.10.1 (192.168.10.1)                             0.557ms
 2:  192.168.1.2 (192.168.1.2)                               1.213ms
 3:  no reply
 4:  10.255.255.6 (10.255.255.6)                             2.301ms reached
     Resume: pmtu 1500 hops 4 back 61
```

在 MikroTik 系统下路由跟踪:

```
[admin@MikroTik] > tool traceroute 10.255.255.1
     ADDRESS                                    STATUS
   1       10.0.1.17 2ms 1ms 1ms
   2    10.255.255.1 5ms 1ms 1ms
[admin@MikroTik] >
```

Log 日志

在各种网络设备都会有日志记录的功能，网络设备的系统会监控并记录主要的设备运行情况和问题调试情况，在 RouterOS 在 log 中可以查看到。该内容可以参考 2.13 章节

Torch (实时通信监听)

实时通信监听被称为 torch 它是用于监视正在运行的一个接口的网络通信流量情况，你可以监听通过该接口的协议、网络端口、源和目的地址，并可以通过分类监听。该功能有助于管理对接口网络情况的判断和指定 IP、协议和端口的分析。请参考 37.4 章节

Profiler 进程资源

Profiler 工具是查看 RouterOS 中各种进程占用 CPU 的情况，有助于鉴别进程占用 CPU 资源情况。参考 37.9 章节

第二章 System 系统管理

2.1 RouterOS 账号管理

对一台网络设备管理是非常重要，管理好坏直接关系到网络的稳定，特别是管理员、操作员和普通访问员的权限设置是非常重要，否则会影响网络的安全性，因此也是最基本的网络维护工作。

操作路径：/user （Winbox 路径是在 system->users 菜单下）

初始化的 RouterOS 设备管理账号默认是 admin，密码为空，admin 权限是最高的在分组里是“full”

在命令行中查看管理账号

```
[admin@MikroTik] /user> print
Flags: X - disabled
 #    NAME                GROUP               ADDRESS             LAST-LOGGED-IN
 0    ;;; system default user
      admin               full                                    jan/31/1970 22:57:15
```

修改 admin 的密码有两种方式，一种是直接进入 user 菜单下修改 admin 的密码，另一方式是当前账号下使用 password 修改密码，如下面修改 admin 的密码为 123456。

```
[admin@MikroTik] > user
[admin@MikroTik] /user> print
Flags: X - disabled
 #    NAME                GROUP               ADDRESS             LAST-LOGGE
 0    ;;; system default user
      admin               full                                    may/02/201
[admin@MikroTik] /user> set admin password=123456
```

或者在根目录下使用 password 修改当前账号的密码：

```
[admin@MikroTik] > password
Change password of the currently logged user.

confirm-new-password --
new-password --
old-password --

[admin@MikroTik] > password
old-password: ***
new-password: ******
confirm-new-password: ******
[admin@MikroTik] >
```

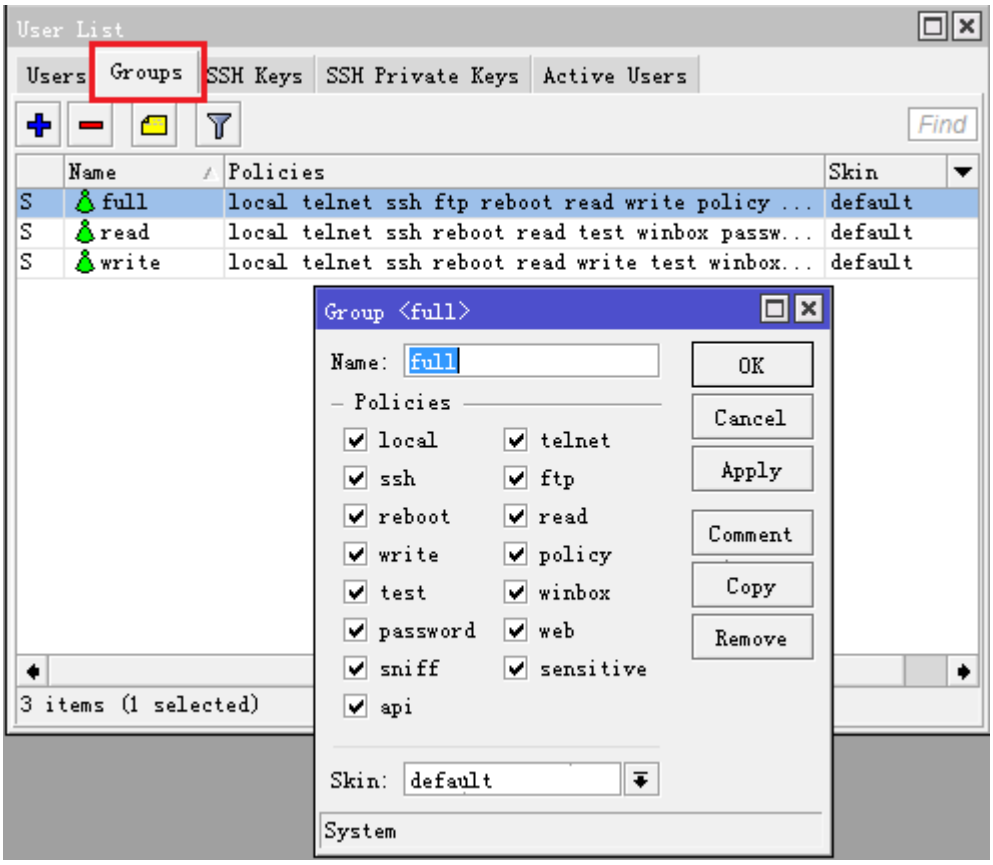
Group（账号分组权限）

RouterOS 建立了账号的权限，根据需要管理员可以分配不同权限和策略的账号，在 RouterOS 中默认分配了三种权限 Full、write 和 read。

属性

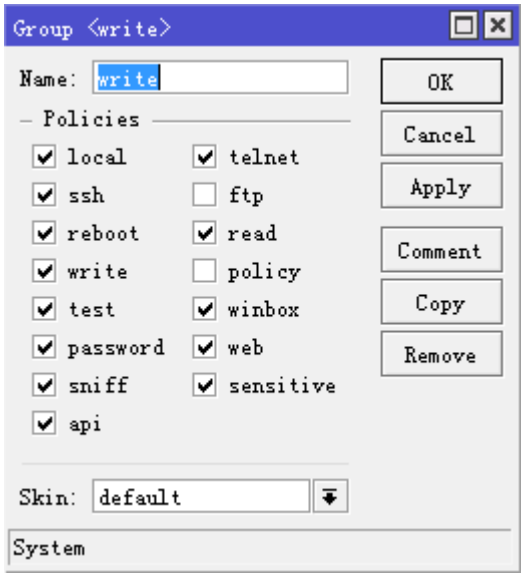
属性	描述
name (*字符*; 默认:) **policy** (*local* | *telnet* | *ssh* | *ftp* | *reboot* | *read* | *write* | *policy* | *test* | *web* | *sniff* | *api* | *winbox* | *password* | *sensitive*; 默认:)	账号分组名称 登录策略: • local – 允许登录访问本地控制台管理 • telnet – 允许通过远程 telnet 登录管理 • ssh – 允许通过远程的 SSH 登录管理 • web – 允许通过远程的 web 页面登录管理 • winbox – 允许通过远程 Winbox 登录管理 • password – 允许该账号修改自身密码 • api – 允许访问 RouterOS API 接口 • dude – 允许登录 Dude 服务 配置策略: • ftp – 允许远程登录 FTP，并传输文件到 RouterOS，该策略的账号能同时进行文件的读写和删除操作。不受路由器"read/write"配置策略限制。 • reboot – 允许重启 RouterOS • read – 允许查看 RouterOS 配置，但不能修改配置，但对 FTP 策略账号无效 • write – 允许修改 RouterOS 的配置，但不能对 RouterOS 账号管理操作，该策略并不允许查看 RouterOS 配置，因此必须同时确保 read 策略被选择 • policy – 允许对 RouterOS 账号管理，通常被用于与 write 策略使用。也能查看全局变量和创建其他管理账号（需要 test 策略） • test – 该策略允许运行 ping, traceroute, bandwidth-test, wireless scan, snooper 和其他测试命令。 • sensitive – 允许查看 RouterOS 敏感信息，具体请看敏感信息介绍 • sniff – 该策略允许使用 sniffer 攻击抓包 • romon – romon 管理

进入 winbox 的 system—user 菜单下的 groups：

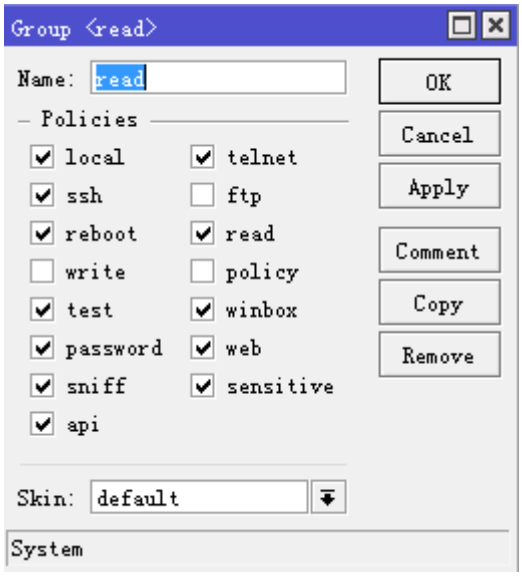


在每个组里我们可以定义他们的权限策略，包括是否有权使用 local、telnet、ssh、ftp、reboot、read、write、policy、winbox、web…

三种权限的区别，当然 full 权限是最高的具备所有的操作权限，而 write 是普通管理，没有 ftp 和 policy 权限，ftp 无法上传和下载文件，policy 权限及无法修改任何账号的密码包括当前的账号。



Read 权限除了不能上传文件和修改密码外，不能写入任何的配置，即只能登陆设备查看



User（RouterOS 管理账号）

RouterOS 账号管理存储了管理人员的相关信息，包括：名称、密码、访问 IP 地址和权限分组

属性

属性	描述
address (*IP/子网| IPv6 prefix*; 默认:)	允许登录 RouterOS 的主机 IP 或网络段
group (*字符*; 默认:)	该账号称归属于那个分组权限（Group）
name (*字符*; 默认:)	账号名称，设置规则第一个字符必须是字母或数字，但后面可以包含"*", "_", "." 和 "@"符号.
password (*字符*; 默认:)	账号密码，如果密码为空白，及代表密码为空，密码设置遵照 Unix 字符标准可以包含"*" 和"_" 符号。

注意：RouterOS 默认会预先设置一个最高权限的 admin 账号，初始化的密码为空

```
[admin@MikroTik] user> print
Flags: X - disabled
  #    NAME                                      GROUP ADDRESS
  0    ;;; system default user
       admin                                     full  0.0.0.0/0

[admin@MikroTik] user>
```

在 RouterOS 至少会有一个最高权限的账号，如果当前 RouterOS 只有一个最高权限的管理账号，是无法将其删除的。

Active 查看在线活跃账号

操作路径: /user active

使用/user active print 命令可以查看当前在线活跃的账号的状态

只读属性

属性	描述
address (*IP/IPv6 地址*)	账号登录主机的 IP/IPv6 地址，如果是 0.0.0.0 及代表本地登录
group (*字符*)	账号属于那个分组权限
name (*字符*)	账号名称
radius (*true* | *false*)	是否验证通过 RADIUS 服务器
via (*console* | *telnet* | *ssh* |*winbox* | *api* | *web*)	账号登录模式
when (*时间*)	登录时间和日期

通过命令查看当前在线活跃账号：

```
[admin@dzeltenais_burkaans] /user active> print detail
Flags: R - radius
 0    when=dec/08/2010 16:19:24 name="admin" address=10.5.8.52 via=winbox

 2    when=dec/09/2010 09:23:04 name="admin" address=10.5.101.38 via=telnet

 3    when=dec/09/2010 09:34:27 name="admin" address=fe80::21a:4dff:fe5d:8e56 via=api
```

Remote AAA（远程 AAA）

操作路径: /user aaa

启用 RouterOS 账号登录的 AAA，账号登录验证使用 RADIUS 服务器，账号登录会优先本地账号查询，如果不存在，会向 RADIUS 服务器请求。

属性

属性	描述
accounting (*yes* | *no*; 默认: **yes**)	启用 RADIUS 记账
exclude-groups (分组名称; 默认:)	排除分组，即包含不允许使用的分组，对于一个通过 RADIUS 验证的账号，如果 RADIUS 服务器提供的分组指定为该列表，default-group 属性将被替换。 目的是用于当一个账号（没有策略修改权限）擅自修改为其他 RADIUS 服务器，并通过 admin 权限登录。
default-group (*字符*; 默认: **read**)	账号分组默认使用 RADIUS 服务器的用户验证
interim-update (*时间*; 默认: **0s**)	更新间隔时间

use-radius (*yes* |*no*; 默认: **no**)	启用 RADIUS 验证

注意：当你启用 RADIUS，并使用 winbox 登录，在 RADIUS 服务器上你需要启用 CHAP 加密

SSH Keys

操作路径: /user ssh-keys

在该菜单允许导入公有密钥用于 SSH 验证

警告：当账号添加了 SSH-keys 后，该账号无法使用 SSH 密码登录

属性

属性	描述
user (*字符*; 默认:)	将 SSH key 分配给那个账号名称

当通/user ssh-keys 命令导入 ssh key，将会有提示两个参数：

- **public-key-file** – 在 RouterOS root 目录下包含 key 的文件名
- **user** – 将 key 分配给的账号名称

Private keys（私有密钥）

操作路径: /user ssh-keys private

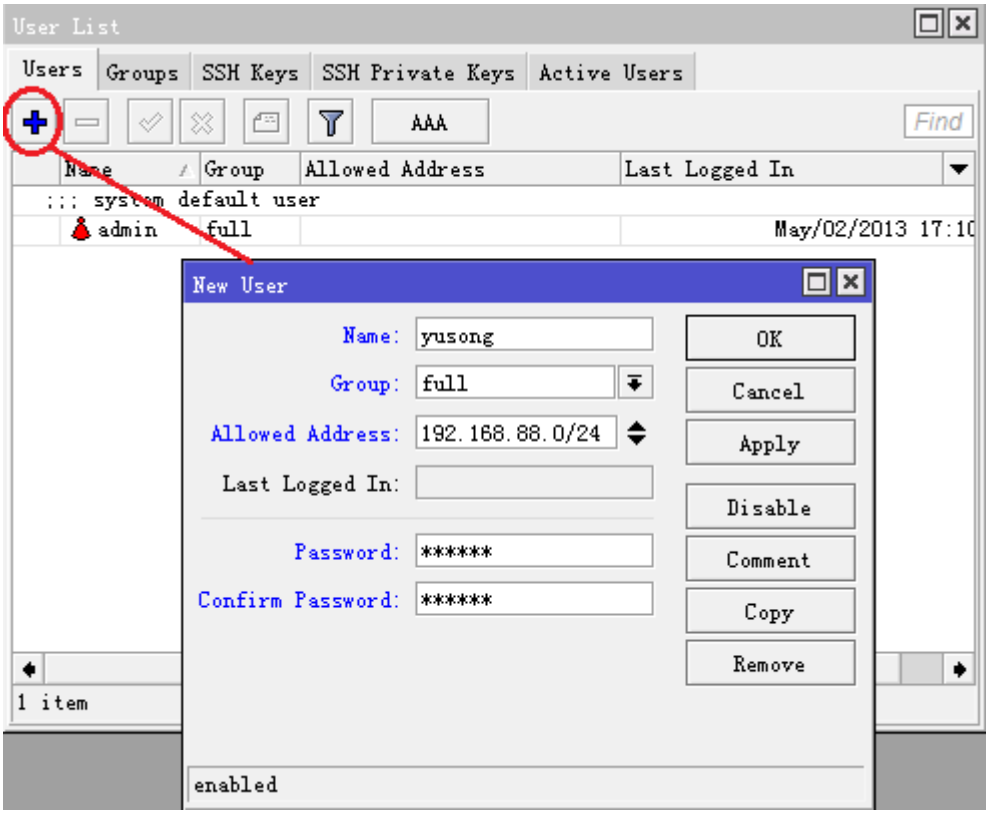
该目录下被用于导入和列出被导入的 Private keys（私有密钥），Private keys 被用于验证使用证书远程登录的方式。

从/user ssh-keys private 目录下导入 ssh keys，会有以下三个提示：

- **private-key-file** -在 RouterOS root 目录下包含私有密钥 key 的文件名
- **public-key-file** - 在 RouterOS root 目录下包含公有密钥 key 的文件名
- **user** - 将 key 分配给的账号名称

创建账号

下面是新建一个 yusong 的账号，Group 为 full，allowed-address=192.168.88.0/24 即只允许从 192.168.88.0/24 的网段登陆访问 RouterOS，其他地址段将会被拒绝。



2.2 RouterOS 备份与复位管理

RouterOS 配置是及时生效，并自动回写到 flash 中，因此配置完成后，不需要像华为交换机一样，需要手动保存配置。

Backup Save 命令：备份配置通常使用 backup 下的 save 命令将系统备份为二进制文件，v6 提供了对文件加密的选项，备份后存放在 File 目录下，可使用 FTP 访问或在 winbox 中的 file 列表中下载备份文件，并可以通过 load 命令恢复备份文件（winbox 是 restore 按钮）。

export 导出命令：（类似华为 display configuration），导出全局或指定目录下的配置，可指定生成文本文件（后缀为.rsc 能用于配置编辑），同样保存在 file 目录下，能使用 FTP 或在 winbox 中的 file 中下载文件，导入配置可以直接复制到命令行，也可以通过 import 命令导入文本文件。

reset-configuration 系统复位命令将所有的配置信息从 RouterOS 中全部删除掉，在做此操作前，最好先将路由器的配置备份一次。

系统备份与恢复

操作路径：***/system backup***

Save 指令是保存当前配置到一个备份文件中，显示文件在**/file** 目录中。如果需要恢复指定的备份文件，可通过**/system backup** 中的 **load** 指令加载配置，还原当前备份文件的配置。从 RouterOS v6.13 开始可以对备份文件做加密，增加了 don't-encrypt 和 password 命令

命令描述:

load name=[filename] – 加载备份文件的配置

save name=[filename] – 保存当前的配置到文件中
dont-encrypt – 告诉系统不使用任何加密，并生成可查看的编辑文本（此方式不安全）
password – 当 password 没有设定，在恢复时要求输入当前管理员的密码，当 password 被设定，密码输入则替换为当前设置密码。

加密介绍

从 RouterOS v6.13 的备份文件默认要求使用加密，如果当前 RouterOS 管理员的管理密码配置，或 backup 的“password”参数被设置，恢复时要求输入密码。如果当前管理员没有设置管理密码（即 admin 没有设置密码），这时备份文件将不会被加密。

例如：将当前的配置保存到文件 **test**：

```
[admin@MikroTik] system backup> save name=test
Saving system configuration
Configuration backup saved
[admin@MikroTik] system backup>
```

在路由器中查看保存的文件：

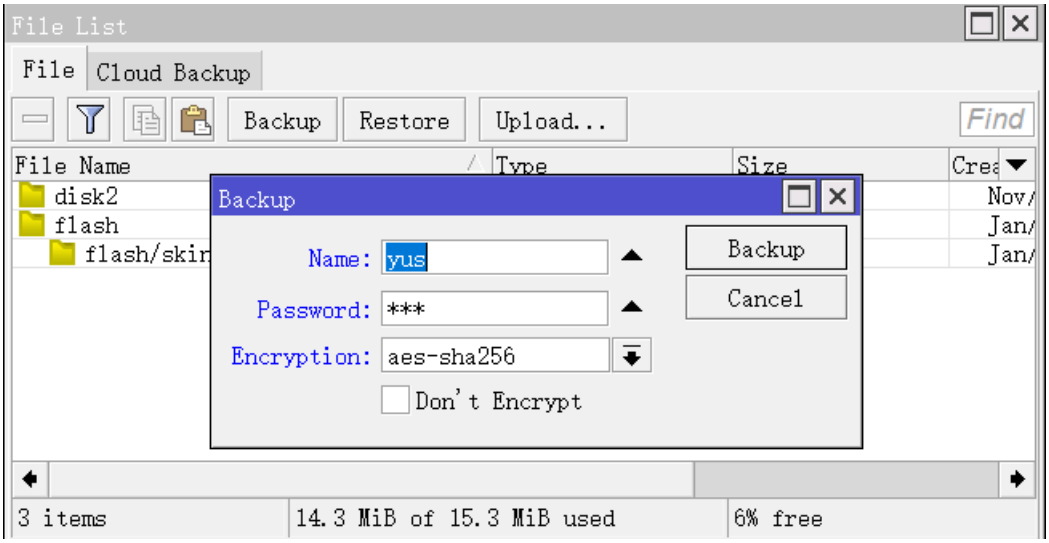
```
[admin@MikroTik] > file print
  # NAME                          TYPE        SIZE       CREATION-TIME
  0 test.backup                   backup      12567      aug/12/2002 21:07:50
[admin@MikroTik] >
```

导入备份文件 **test**:

```
[admin@MikroTik] system backup> load name=test
Restore and reboot? [y/N]: y
...
```

注：为了保证恢复备份不会失败，请将备份的文件恢复到相同型号的设备和版本，特别 RouterOS 版本跨度较大的情况下，配置路径和参数可能发生改变！

Winbox 下配置直接在 files 菜单下，通过 backup 和 restore 操作，备份要求输入文件名和加密密码



RouterOS 云备份

从 RouterOS v6.44beta 版本开始提供云备份功能，具体细节如下：

- 免费为 MikroTik 路由器用户提供
- 仅支持 aes-sha265 的备份文件加密
- 如果你将下载密钥保存，也可以从另外一台路由器下载备份文件
- 一条命令生成、加密和上传备份文件
- 一条命令下载和应用备份
- 连接 IP cloud 服务器，全程采用加密方式，因此你的备份文件通过安全加密隧道传输
- 每台路由支持 1 个云备份文件

如果你只是下载备份文件，仍然需要在/system backup load 里执行文件恢复， 配置云备份基本操作如下：

首先要确保 RouterOS 能连接网络，并正确配置 DNS 服务器，通过下面命令生成密钥

```
[admin@MikroTik] /system backup cloud> print
— retrieving key
— downloading
```

然后执行云备份操作命令

```
[admin@MikroTik] /system backup cloud> upload-file action=create-and-upload password=yus
status: finished
```

查看备份情况：

```
[admin@MikroTik] /system backup cloud> print
0 name=”cloud-20180927-074829” size=9.8KiB ros-version=”6.44beta9” date=sep/27/2018 07:48:31
status=”ok” secret-download-key=”pXNxUwPjU5s3P1ejlhUjoJV”
[admin@MikroTik] /system backup cloud>
```

注意: 如果 The Dude 和 user-manager 安装在当前系统中，backup 工具不会对他们做备份，因此这两个功能备份需要单独操作。

导出指令（Export）

指令名称：**export**

Export 指令用于导出脚本配置，这个命令可以在任何目录下执行。**export** 同样也可以通过 **file** 属性指定保存的.rsc 文件名，可用 FTP 或者进入 winbox 下载。Export 导出的是明文，且是标准的 RouterOS 配置脚本，所以可以直接进行编辑。

指令描述:

hid-sensitive – 隐藏敏感参数，如密码等信息。
file=[filename] – 保存的文件名。

例如：

```
[admin@MikroTik] > ip address print
Flags: X - disabled, I - invalid, D - dynamic
 #    ADDRESS             NETWORK          BROADCAST        INTERFACE
 0    10.1.0.172/24       10.1.0.0         10.1.0.255       bridge1
 1    10.5.1.1/24         10.5.1.0         10.5.1.255       ether1
[admin@MikroTik] >
```

导出一个脚本文件，该文件包含了 IP 地址配置参数：

```
[admin@MikroTik] ip address> export file=address
[admin@MikroTik] ip address>
```

在路由器中查看汇出的文件：

```
[admin@MikroTik] > file print
 # NAME                              TYPE          SIZE        CREATION-TIME
0  address.rsc                       script        315         dec/23/2003 13:21:48
 [admin@MikroTik] >
```

Compact Export

从 v5.12 新增 compact export 功能，允许仅导出配置的相关参数，而非所有 RouterOS 默认配置，例如下面的导出配置：

```
[admin@MikroTik] > export compact
# jan/02/2012 01:57:23 by RouterOS 5.12
#
/ip pool
add name=dhcp_pool1 ranges=10.1.0.2-10.1.0.254
/ip dhcp-server
add address-pool=dhcp_pool1 disabled=no interface=ether3 name=dhcp1
/ip address
add address=10.1.0.1/24 interface=ether3
/ip dhcp-client
add disabled=no interface=ether1
/ip dhcp-server network
add address=10.1.0.0/24 gateway=10.1.0.1
/ip dns
set allow-remote-requests=yes max-udp-packet-size=512 servers=10.5.8.1
/ip firewall nat
add action=masquerade chain=srcnat out-interface=ether1
/ip smb shares
set [ find default=yes ] directory=/pub
/system ntp client
set primary-ntp=10.1.1.1 secondary-ntp=10.1.1.2
/system routerboard settings
```

```
set cpu-frequency=266MHz
/tool bandwidth-server
set authenticate=no
[admin@MikroTik] >
```

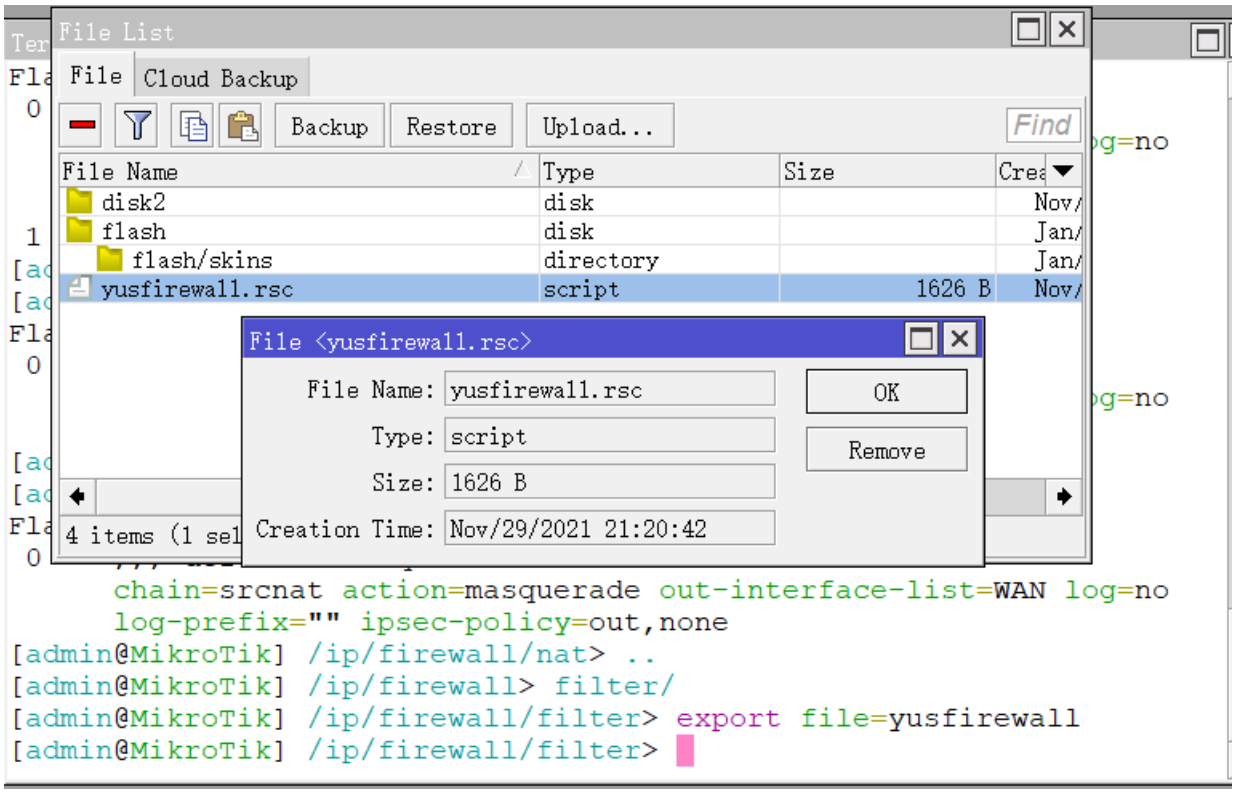
注意：从 v6rc1 版本后 ”export compact"被默认执行，如果要使用旧的导出格式，使用 export verbose

```
/export verbose file=myConfig
```

Compact export 作为紧凑导出配置，省略了 RouterOS 默认配置，这种默认配置是无法被删除，通过”*”表示，如下面 OSPF 事例中默认配置

```
[admin@SXT-ST] /routing ospf instance> print
Flags: X - disabled, * - default
 0   * name="default" router-id=0.0.0.0 distribute-default=never
       redistribute-connected=as-type-1 redistribute-static=no
       redistribute-rip=no redistribute-bgp=no redistribute-other-ospf=no
       metric-default=1 metric-connected=20 metric-static=20 metric-rip=20
       metric-bgp=auto metric-other-ospf=auto in-filter=ospf-in
       out-filter=ospf-out
```

Winbox 的 file 目录下，可以查看生成的.rsc 的文件：



导入指令（Improt）

操作路径：***/import***

import 命令只能在根目录下使用**/import “filename”**指令导入指定配置。这种方式适用于部分配置或者功能。

指令描述：

file=[filename] – 导入的路由器配置文件

例如使用下面的指令操作导入配置文件：

```
[admin@MikroTik] > import address.rsc
Opening script file address.rsc
Script file loaded successfully
[admin@MikroTik] >
```

系统复位

操作路径：***/system> reset-configuration***

这个指令将会清除掉路由器的所有配置，包括登陆的账号和密码（恢复为“admin“和空密码），在 reset 指令执行后路由器将会重启。复位后 RouterOS 会使用出厂默认配置，可以通过命令查看默认配置（/system default-configuration print）。对于 MikroTik 硬件设备，复位后会使用默认配置，不同型号 MikroTik 的出厂默认配置可以参考：各种 MikroTik 设备的初始化默认配置介绍。

在命令行服务器配置，例如：

```
[admin@Office] /system> reset-configuration
Dangerous! Reset anyway? [y/N]: y
```

复位时你在命令后也可以带相关参数，如下：

```
[admin@MikroTik] /system> reset-configuration
keep-users   no-defaults   run-after-reset   skip-backup
```

复位时可带 keep-users 指令，描述如下：

Keep-users – 复位不删除管理用户账号
no-defaults - 不加载任何默认配置，清空所有
skip-backup - 当 yes 被指定，自动备份将不会再复位前创建
run-after-reset - 在 run after reset 指定后 export 档后，复位会导入指定的 RouterOS 脚本配置文件。

注意：如果 MikroTik RB、CRS 和 CCR 硬件设备使用 netinstall 安装，并且指定了一个初始的脚本配置写入 ROM 中，则 reset 复位命令执行在清除配置，实际是载入的该脚本。要阻止这样的情况，必须重新使用 netinstall 安装路由器，配置文件指定为空（请参见自定义 RouterOS 默认启动脚本）。

2.3 系统重启与关机

操作路径：***/system reboot***

注：当重启时系统会自动搜索是否有新版本的安装包，如果发现相应安装包系统将升级和安装功能。

重启命令将发送信息给运行中的处理器，并停止和卸除系统文件，重启路由器。

```
[admin@MikroTik] > system reboot
```

```
Reboot, yes? [y/N]: y
system will reboot shortly
[admin@MikroTik] >
```

操作路径：***/system shutdown***

在路由器电源关闭前，应停止路由系统的运行，重启命令将发送信息给运行中的处理器，并停止和卸除系统文件，关闭路由器。

在一些系统需要大概 10 秒（如果没有升级操作，通常最少需要 5 秒）才能安全关闭电源。

```
[admin@MikroTik] > system shutdown
Shutdown, yes? [y/N]: y
system will shutdown promptly
[admin@MikroTik] >
```

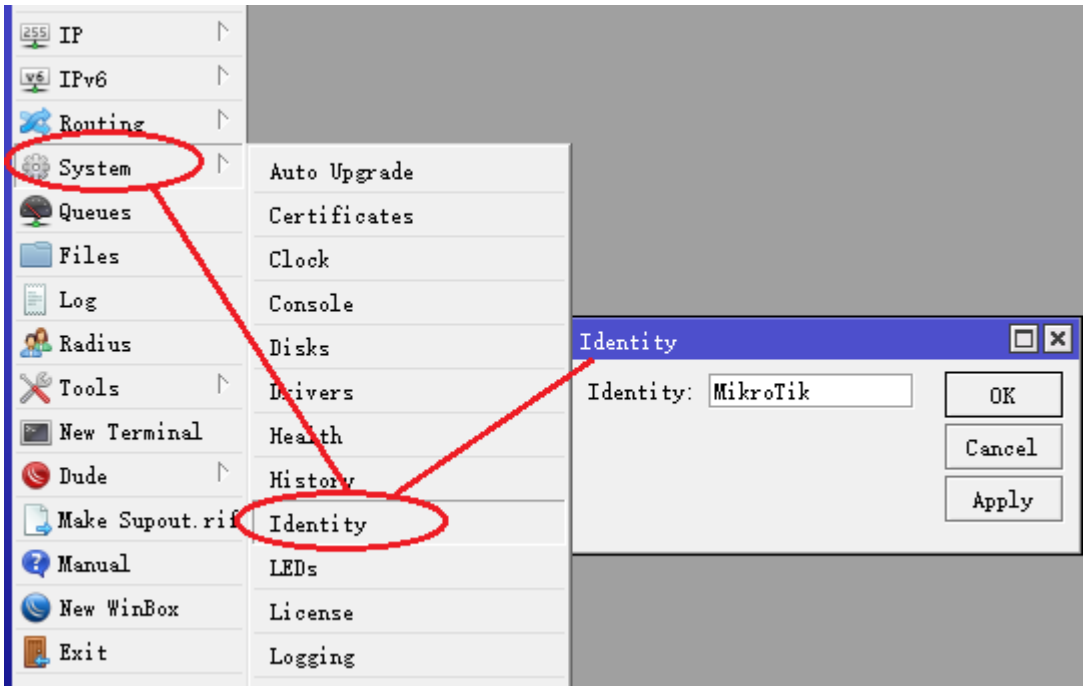
2.4 RouterOS 主机名

操作路径：***/system identity***

通过命令可以查看路由器主机名，这个主机身份名也会被初始化到 DHCP 客户端的“主机名（host name）”或者 Wlan 的 SSID 名，下面是查看路由器主机身份名：

```
[admin@MikroTik] > system identity print
    name: "MikroTik"
[admin@MikroTik] >
```

Winbox 中查看：



设置路由器身份名：

```
[admin@MikroTik] > system identity set name=MyRouterOS
[admin@MyRouterOS] >
```

2.5 Resource（资源管理）

操作路径：**/system resource**

查看系统资源可以了解 RouterOS 的运行情况

属性	描述
architecture-name (*字符*)	CPU 平台构架，包括 powerpc, x86, mipsbe 、mipsle 、tilegx 和 ARM、 .
bad-blocks (百分比)	显示当前 NAND 坏区块的百分比
board-name (*string*)	RouterBOARD 型号名称
cpu (*字符*)	CPU 型号.
cpu-count (*整型*)	系统识别的 CPU 核心数，Intel HT 被识别为独立的 CPU 核心
cpu-frequency (*string*)	当前 CPU 频率
cpu-load (*百分比*)	CPU 使用的百分比，显示所有 CPU 的平均值，要查看每个 CPU 使用率可以查看 CPU 子菜单
free-hdd-space (*字符*)	系统磁盘或 NAND 存储剩余空间
free-memory (*字符*)	未使用的内存
platform (*字符*)	平台名称，通常显示 "MikroTik"
total-hdd-space (*字符*)	磁盘或 NAND 存储的总大小
total-memory (*字符*)	系统内存总大小
uptime (*时间*)	系统从开机到当前的运行时间
version (*字符*)	安装 RouterOS 版本序列号
write-sect-since-reboot (*整型*)	最近一次路由器重启，存储扇区写入数量
write-sect-total (*整型*)	存储器的总扇区写入数量.

通过 monitor 命令实时的 CPU 占用率、内存和硬盘等使用情况。

查看基本的系统资源情况：

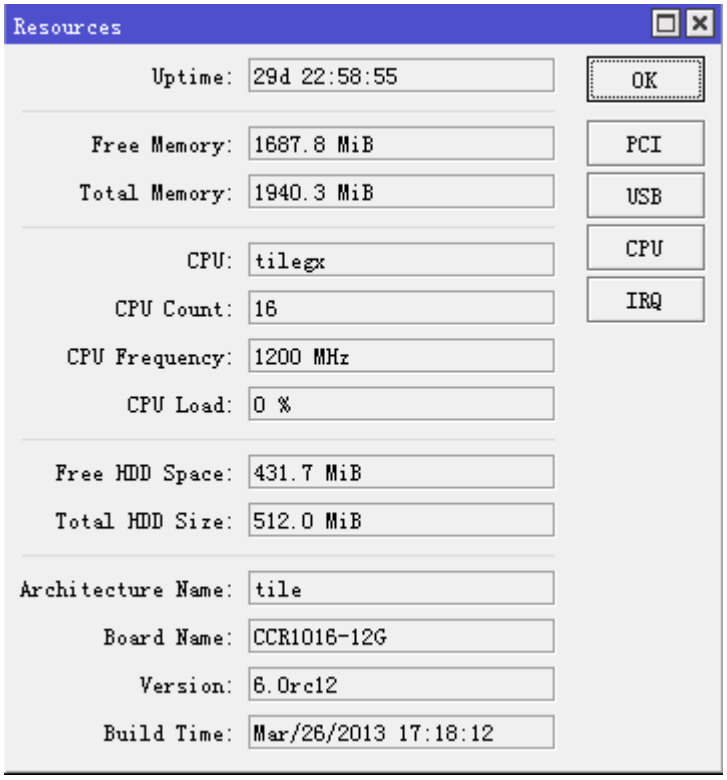
```
[admin@MikroTik] /system resource> print
                   uptime: 4w1d22h57m21s
                  version: 6.0rc12
               build-time: Mar/26/2013 17:18:12
             free-memory: 1685.4MiB
            total-memory: 1940.2MiB
                      cpu: tilegx
                cpu-count: 16
            cpu-frequency: 1200MHz
                 cpu-load: 0%
           free-hdd-space: 431.7MiB
```

```
    total-hdd-space: 512.0MiB
  architecture-name: tile
         board-name: CCR1016-12G
           platform: MikroTik
```

实时查看系统 CPU 和空闲内存使用情况：

```
[admin@MikroTik] /system resource> monitor
              cpu-used: 0%
 cpu-used-per-cpu: 0%,0%,0%,0%,0%,0%,0%,1%,0%,0%,1%,0%,0%,0%,0%,0%
         free-memory: 1728704KiB
```

使用 winbox 查看：



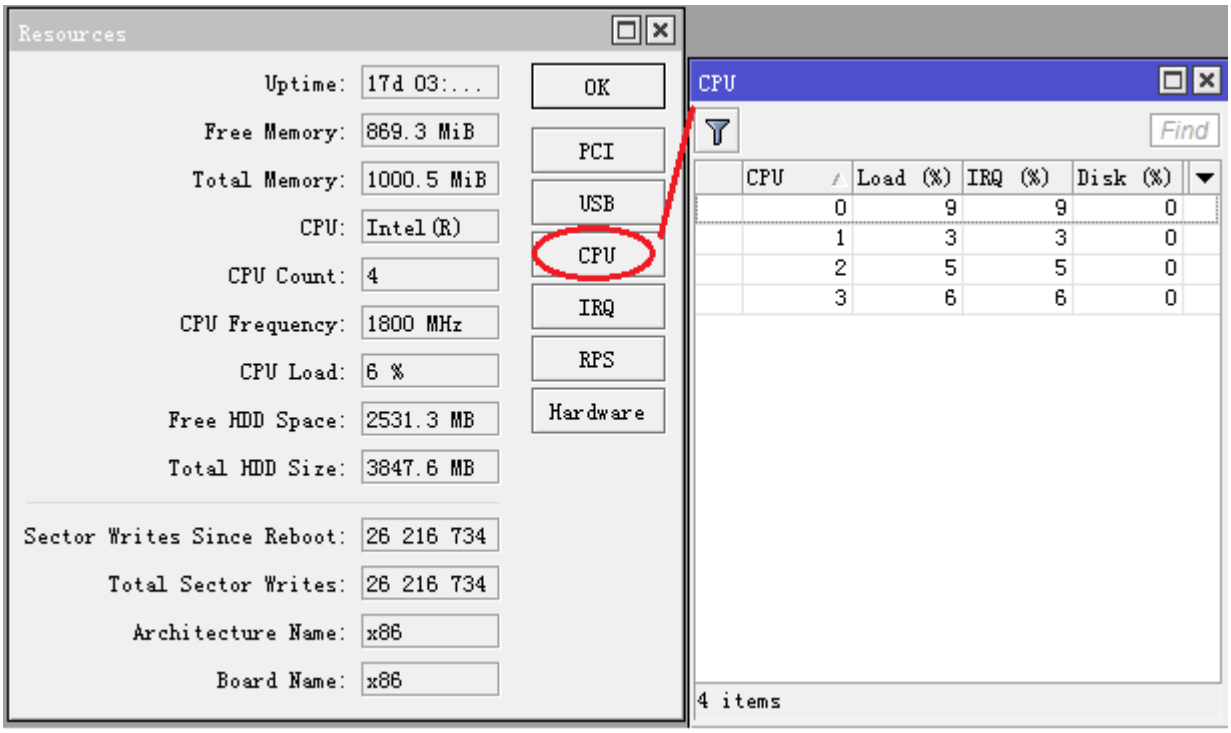
CPU

操作路径: /system resource cpu

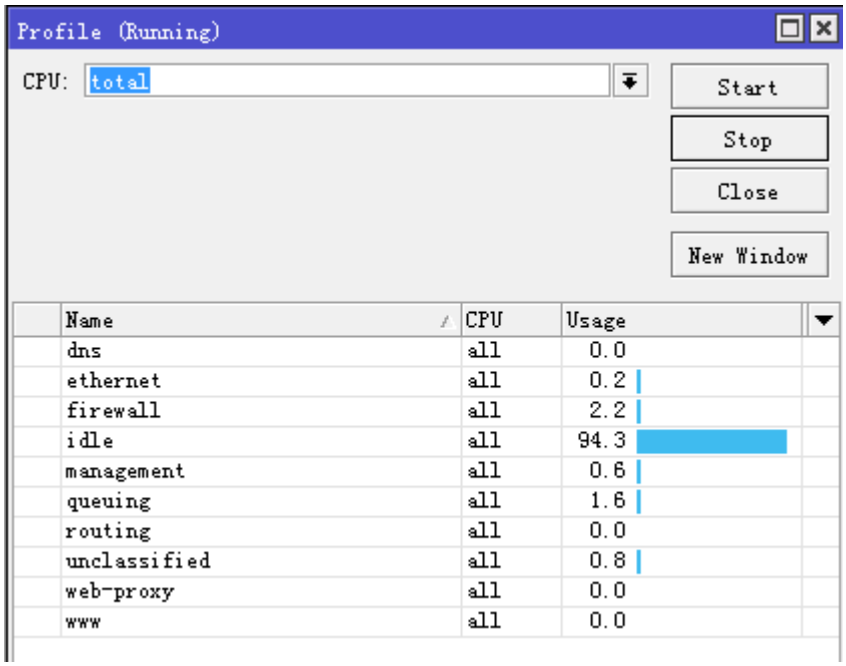
在 RouterOS v5 版本后增加了 CPU 菜单，该菜单显示每个 CPU 使用率，也包括 IRQ 和 Disk 使用率

```
[admin@MikroTik] /system resource cpu> print
CPU LOAD          IRQ          DISK
0    5%           0%           0%
[admin@MikroTik] /system resource cpu>
```

在 Winbox 中查看每个 CPU 的占用情况：



在 tool 里还增加了每个功能的 CPU 占用情况，进入 tool profile 下可以查看 RouterOS 各个功能 CPU 使用情况，和 windows 资源管理器类似



IRQ 配置管理

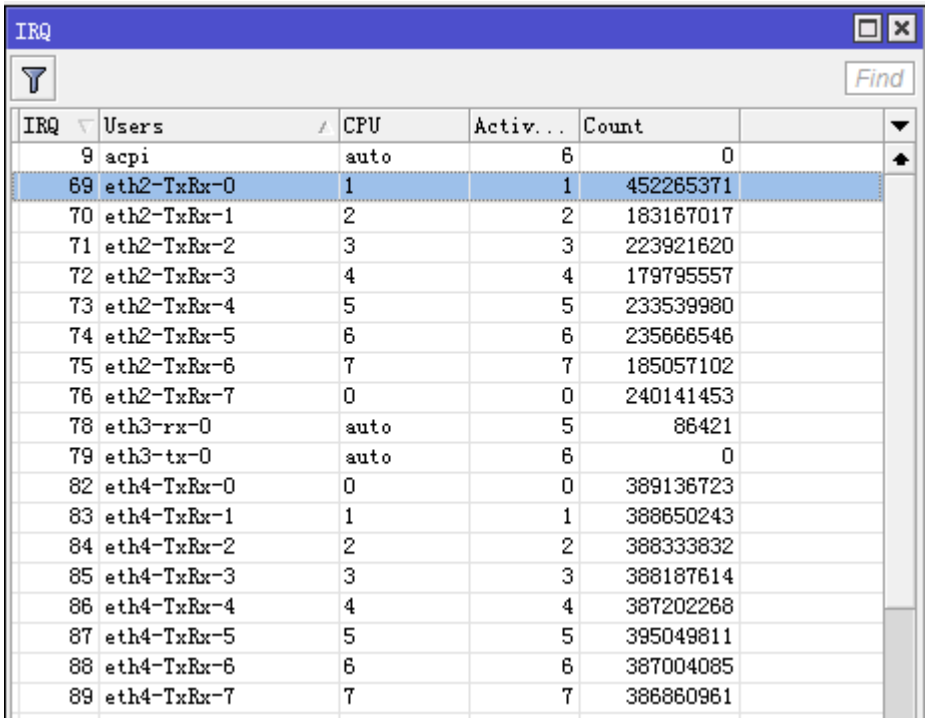
命令路径: ***/system resource irq print***

IRQ “中断”简单的说就是，每个硬设备（如：硬盘、网卡、USB 设备等）都需要和 CPU 通信，让 CPU 响应这些硬设备的请求，以便 CPU 及时知道发生了什么事情，这样 CPU 可能就会放下手中的事情去处理应急事件，硬设备主动打扰 CPU 的现象就可称为硬件中断，就像你正在工作的时候收到短信一样，你就去打开手机查看，这样的情况可以称为中断。

中断是一种较好的 CPU 和硬件沟通的方式，还有一种方式叫做轮询（polling），就是让 CPU 定时对硬件状态进行查询然后做相应处理，就好像你每隔 5 分钟去检查一下邮箱看看有没有人联系你一样，这种方式是不是很浪费你（CPU）的时间？所以中断是硬件主动的方式，比轮询（CPU 主动）更有效一些。

硬件中断发生频繁，是件很消耗 CPU 资源的事情，在多核 CPU 条件下如果有办法把大量硬件中断分配给不同的 CPU 或其他核心上处理，这样系统会得到更好的负载平衡性能。现在的服务器上动不动就是多 CPU 或多核、多网卡、多硬盘，如果能让网卡中断独占 1 个 CPU 或核心，磁盘 IO 中断独占 1 个 CPU 的话，将会大大减轻单一 CPU 的负担、提高整体处理效率。

RoueterOS 从 v5.0 版本后增加了 IRQ 中断配置属性，这种方式在 Linux 上称为“中断亲和”，在多 CPU 系统下，可以通过调整各个硬件的 CPU 中断，提升系统 CPU 在高负载下的性能。

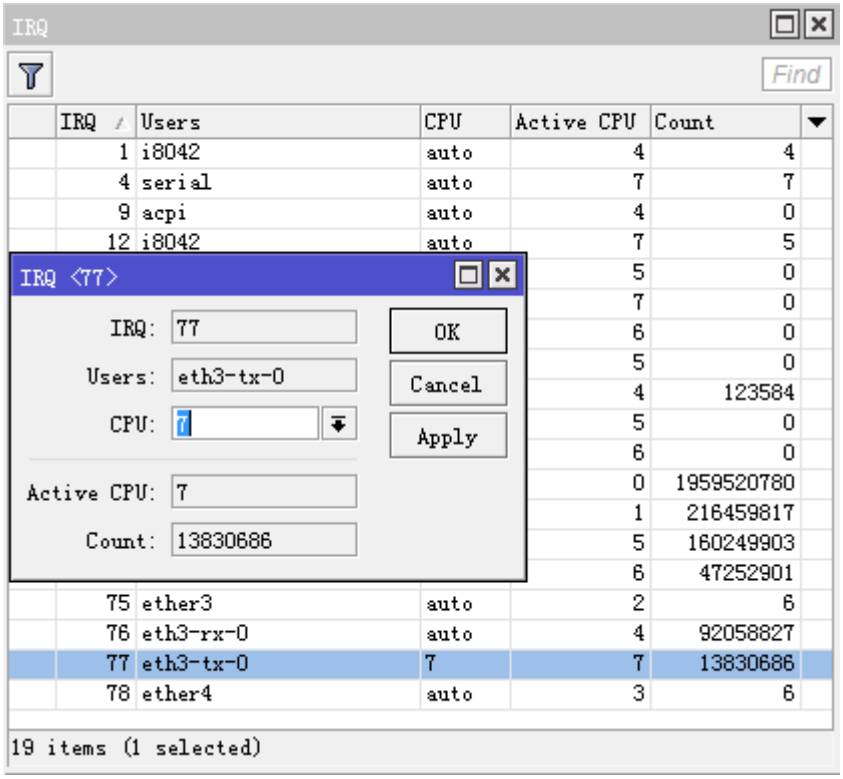


IRQ	Users	CPU	Activ...	Count
9	acpi	auto	6	0
69	eth2-TxRx-0	1	1	452265371
70	eth2-TxRx-1	2	2	183167017
71	eth2-TxRx-2	3	3	223921620
72	eth2-TxRx-3	4	4	179795557
73	eth2-TxRx-4	5	5	233539980
74	eth2-TxRx-5	6	6	235666546
75	eth2-TxRx-6	7	7	185057102
76	eth2-TxRx-7	0	0	240141453
78	eth3-rx-0	auto	5	86421
79	eth3-tx-0	auto	6	0
82	eth4-TxRx-0	0	0	389136723
83	eth4-TxRx-1	1	1	388650243
84	eth4-TxRx-2	2	2	388333832
85	eth4-TxRx-3	3	3	388187614
86	eth4-TxRx-4	4	4	387202268
87	eth4-TxRx-5	5	5	395049811
88	eth4-TxRx-6	6	6	387004085
89	eth4-TxRx-7	7	7	386860961

上图是我们 eth3 只支持 2 个中断请求，而 eth2 和 eth4 分别支持 8 组中断请求，也就是可以更灵活均衡的分配给其他 CPU 处理，该硬件是至强双 CPU，每个 CPU 为 4 核心，共有 8 核心处理器。

关于网卡的中断请求，要看网卡的芯片版本，比如代号为 Kawela 的 Intel 82576 芯片在千兆网卡里面属于功能最强大的，它支持 PCIe 2.0 x4 接口，支持 MSI-X 中断方式，提供了 16 个 TX 和 RX 队列，82574 是支持 2 个 TX 和 RX 队列，82575 支持 8 个 TX 和 RX 队列。网上常见的如 Intel 82580 4 口网卡，能被 RouterOS v6 正常识别，也提供了每个网口 8 个中断。

RouterOS 会自动分配 IRQ 给对应的硬设备，当然 IRQ 配置肯定是在多 CPU 的前提下才有效，单核单 CPU 是没有意义的。如果你的系统负载不高的情况下可以不用理会 IRQ，在高负载下时，IRQ 调整将有助于 CPU 处理的均衡，如下图我们可以手动设置中断请求到指定的 CPU：



USB 端口信息

操作路径: ***/system resource usb print***

显示所有路由器可用的 USB 端口。

device (*只读: 文本*) – 设备编号
name (*只读: 文本*) – USB 端口名称
speed (*只读: 整型*) – 该端口工作的带宽速度
vendor (*只读: 文本*) – USB 设备销售商名称

显示所有可用 USB 端口：

```
[admin@MikroTik] system resource usb> print
 # DEVICE VENDOR                    NAME                    SPEED
 0 1:1                              USB OHCI Root Hub       12 Mbps
[admin@MikroTik] system resource usb>
```

PCI 信息

操作路径: ***/system resource pci print***
category (*只读: 文本*) – 设备类型
device (*只读: 文本*) – 设备编号
device-id (*只读: 整型*) – 十六进制设备 ID
irq (*只读: 整型*) – 该设备使用的 IRQ 编号
memory (*只读: 整型*) – 该设备使用的内存长度
name (*只读: 文本*) – 设备名称
vendor (*只读: 文本*) – 设备销售商名称
vendor-id (*只读: 整型*) – 设备十六进制销售商

查看 PCI 情况：

```
[admin@MikroTik] system resource pci> print
 # DEVICE    VENDOR                        NAME                          IRQ
 0 00:13.0   Compaq                        ZFMicro Chipset USB (rev... 12
 1 00:12.5   National Semi                 SC1100 XBus (rev: 0)
 2 00:12.4   National Semi                 SC1100 Video (rev: 1)
 3 00:12.3   National Semi                 SCx200 Audio (rev: 0)
 4 00:12.2   National Semi                 SCx200 IDE (rev: 1)
 5 00:12.1   National Semi                 SC1100 SMI (rev: 0)
 6 00:12.0   National Semi                 SC1100 Bridge (rev: 0)
 7 00:0e.0   Atheros Communications        AR5212 (rev: 1)               10
 8 00:0d.1   Texas Instruments            PCI1250 PC card Cardbus ... 11
 9 00:0d.0   Texas Instruments            PCI1250 PC card Cardbus ... 11
10 00:0c.0   National Semi                DP83815 (MacPhyter) Ethe... 10
11 00:0b.0   National Semi                DP83815 (MacPhyter) Ethe... 9
12 00:00.0   Cyrix Corporation             PCI Master (rev: 0)
[admin@MikroTik] system resource pci>
```

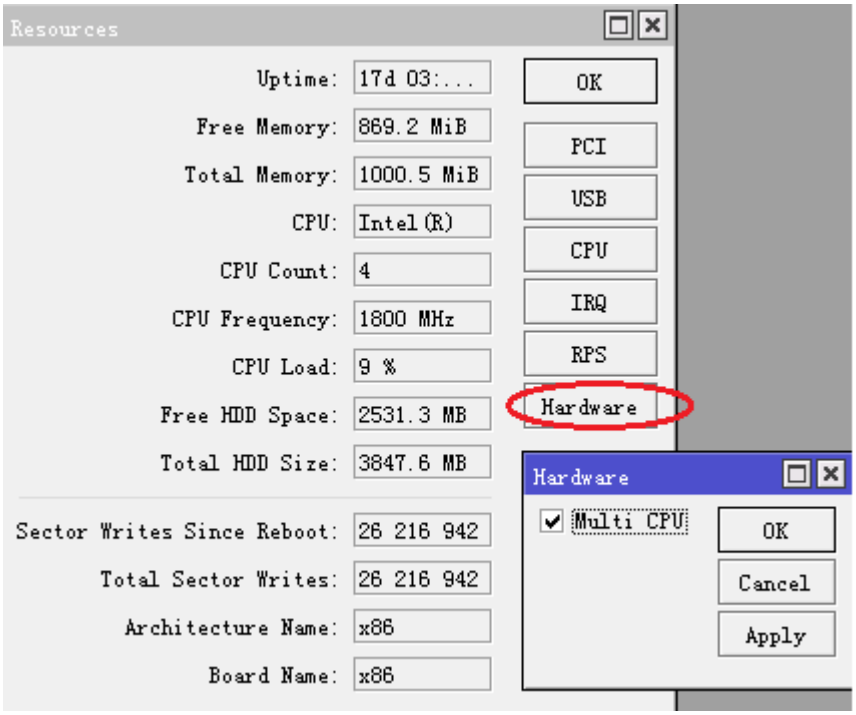
RouterOS x86 平台多 CPU 设置

操作路径：***/system hardware***

如果你使用的是多 CPU 或多核心 CPU，可以通过 hardware 功能开关多 CPU 的支持，这个功能仅在 x86 的系统下配置，也可以根据自己的运行情况改变多 CPU 的支持，通过下面的命令启用多 CPU 功能：

```
[admin@MikroTik] > system hardware
[admin@MikroTik] /system hardware>
..  /  :  edit  export  get  print  set
[admin@MikroTik] /system hardware> set multi-cpu=yes ;
[admin@MikroTik] /system hardware> prin
  multi-cpu: yes
[admin@MikroTik] /system hardware>
```

设置完成后，重启路由即可。



2.6 Watchdog 监测

Watchdog 监测系统运行情况，一旦系统软件故障或停止响应，将会自动重启，由此来避免死机和停止工作。

功能包需求: ***system***
等级需求: *Level1*
操作路径: ***/system watchdog***

通过 watchdog 可以监控一个 IP 地址没有响应或者系统被死锁，一旦发生这样的情况将发出重启指令。软件定时器是用来提供上一次的记录，但是在特殊的情况下(由硬件故障引起的) 它能锁定自己. 对于 RouterBOARD 的硬件监测设备来说它能在任何异常情况下重启。

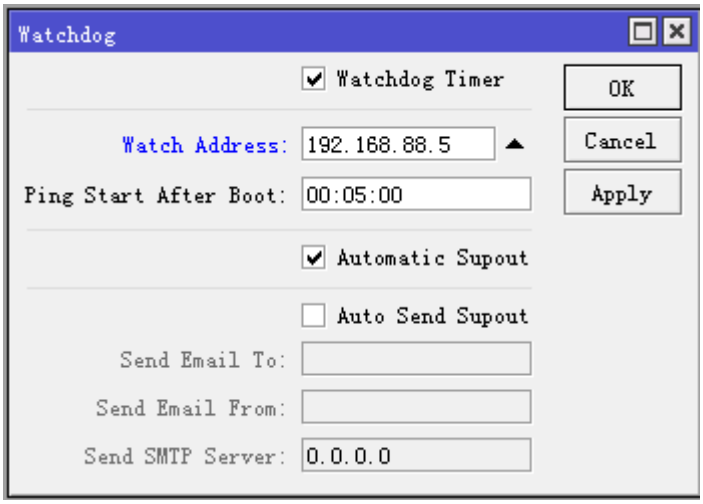
属性描述

auto-send-supout (yes | no; 默认: **no**) – 技术支持档将通过邮件发送到指定邮箱。
automatic-supout (yes | no; 默认: **yes**) –当软件错误发生时，将是自动生成，如果新的技术支持档产生将命名为"autosupout.rif" ，而之前的文件，会重命名为"autosupout.old.rif"。
no-ping-delay (时间; 默认: **5m**) – 在重启以后多久去测试和 ping **watch-address**. 默认设置是如果 **watch-address** 被设置为不可达，这时路由器将在 6 分钟的时候重启.
send-email-from (文本; 默认: **""**) – 发送邮件的来源地址，确定**/tool e-mail** 功能开启。
send-email-to (*文本*; 默认: **""**) – 接收技术支持档的邮件地址。
send-smtp-server (*文本*; 默认: **""**) – SMTP 服务地址，如果没有设置可以通过操作路径**/tool e-mail** 开启功能。
watch-address (*IP 地址*; 默认: **none**) – 如果设置这功能了的话，一旦 6 个连续的 ping 包没有响应，系统会重启。
none – 不启用
watchdog-timer (yes | no; 默认: **no**) – 是否启用 watchdog 功能。

下面是一个系统崩溃的邮件发送配置，一旦系统崩溃，自动产生的 supout.rif 技术支持档，并自动通过 192.0.2.1 发送到 **support@example.com**：

```
[admin@MikroTik] system watchdog> set auto-send-supout=yes \
\... send-to-email=support@example.com send-smtp-server=192.0.2.1
[admin@MikroTik] system watchdog> print
        watch-address: none
      watchdog-timer: yes
       no-ping-delay: 5m
   automatic-supout: yes
   auto-send-supout: yes
   send-smtp-server: 192.0.2.1
       send-email-to: support@example.com
[admin@MikroTik] system watchdog>
```

如我们通过 ping 监控 IP 地址 192.168.88.5，当在 5 分钟后没有响应，路由器会自动重启。



2.7 RouterOS 功能包（Packages）

RouterOS 提供了各种功能包的安装和管理，功能包可以从 http://www.mikrotik.com/download.html 页面下载，提供了 http 方式下载，管理员可以根据需要安装功能包，对于 RouterOS 而言正确的安装使用功能包有助于系统维护和减小系统开销。

RouterOS 安装和管理时每个功能包的组成：

功能包	包含功能
advanced-tools (mipsle, mipsbe, ppc, x86, tile)	包含各种工具 ping、netwatch、ip-scan、sms tool 和 wake-on-LAN
calea (mipsle, mipsbe, ppc, x86, tile)	数据收集功能，指定适用于在美国标准的 "Communications Assistance for Law Enforcement Act"
dhcp (mipsle, mipsbe, ppc, x86, tile)	动态主机控制协议客户端和服务器
gps (mipsle, mipsbe, ppc, x86, tile)	支持全球定位系统设备
hotspot (mipsle, mipsbe, ppc, x86, tile)	HotSpot 热点认证系统

ipv6 (mipsle, mipsbe, ppc, x86, tile)	支持 IPv6
mpls (mipsle, mipsbe, ppc, x86, tile)	多协议卷标交换（Multi Protocol Labels Switching）
multicast (mipsle, mipsbe, ppc, x86, tile)	组播协议支持;
	IGMP（Internet Group Managing Protocol）- 代理 Proxy
ntp (mipsle, mipsbe, ppc, x86, tile)	网络对时协议客户端和服务端
Openflow(mipsle, mipsbe, ppc, x86, tile)	Openflow 协议，当前 RouterOS 支持 Openflow v1.0.0
ppp (mipsle, mipsbe, ppc, x86, tile)	PPP、PPTP、L2TP、PPPoE, ISDN PPP 客户端和服务器
routerboard (mipsle, mipsbe, ppc, x86, tile)	访问和管理 RouterBOOT 固件，仅支持 RouterBOARD 硬件
routing (mipsle, mipsbe, ppc, x86, tile)	动态路由协议如 RIP, BGP, OSPF 和路由管理如 BFD 和路由过滤
security (mipsle, mipsbe, ppc, x86, tile)	IPSEC、SSH 和 winbox 加密连接
system (mipsle, mipsbe, ppc, x86, tile)	路由器基本功能，如静态路由、ip 地址、sNTP、telnet、API、queue、firewall、web-proxy、DNS 缓存、TFTP、IP 地址池、SNMP、sniffer、e-mail 工具、graphing、Bandwidth 测试、torch、EoIP、IPIP、桥接、VLAN、VRRP，在 RouterBOARD 平台也包含 MetaROUTER 虚拟器
ups (mipsle, mipsbe, ppc, x86, tile)	支持 APC ups
user-manager (mipsle, mipsbe, ppc, x86, tile)	MikroTik User Manager 类 RADIUS 系统
wireless (mipsle, mipsbe, ppc, x86, tile)	Wireless 接口支持，802.11abgn
isdn (x86)	支持 ISDN
lcd (x86)	支持 LCD 显示面板
radiolan (x86)	支持 RadioLan 网卡
synchronous (x86)	支持 FarSync
xen (discontinued x86)	XEN 虚拟机（在 4.0 后已经取消）
kvm (x86)	KVM 虚拟机
routeros-mipsle (mipsle)	mipsle 组合包(RB100 系列和 RB500 系列) 包含 system、hotspot, wireless、ppp、security、mpls、advanced-tools、dhcp、 routerboard、ipv6 和 routing)
routeros-mipsbe (mipsbe)	mipsbe 组合包(RB400 系列、700 系列、RB900 系列和 RB2011 系列)包含 system、hotspot, wireless、ppp、security、mpls、advanced-tools、dhcp、routerboard、ipv6 和 routing)
routeros-powerpc (ppc)	PowerPC 组合包(RB333、RB600/A、RB800 和 RB1000 系列) 包含 system、hotspot, wireless、ppp、security、mpls、advanced-tools、dhcp、routerboard、ipv6 和 routing)
routeros-x86 (x86)	x86 组合包(Intel/AMD PC, RB230) 包含 system、hotspot, wireless、ppp、security、mpls、advanced-tools、dhcp、 routerboard、ipv6 和

	routing)
routeros-tile (tile)	Tilera 组合包(CCR1016 和 CCR1036 系列) 包含 system、hotspot, wireless、ppp、security、mpls、advanced-tools、dhcp、 routerboard、ipv6 和 routing)

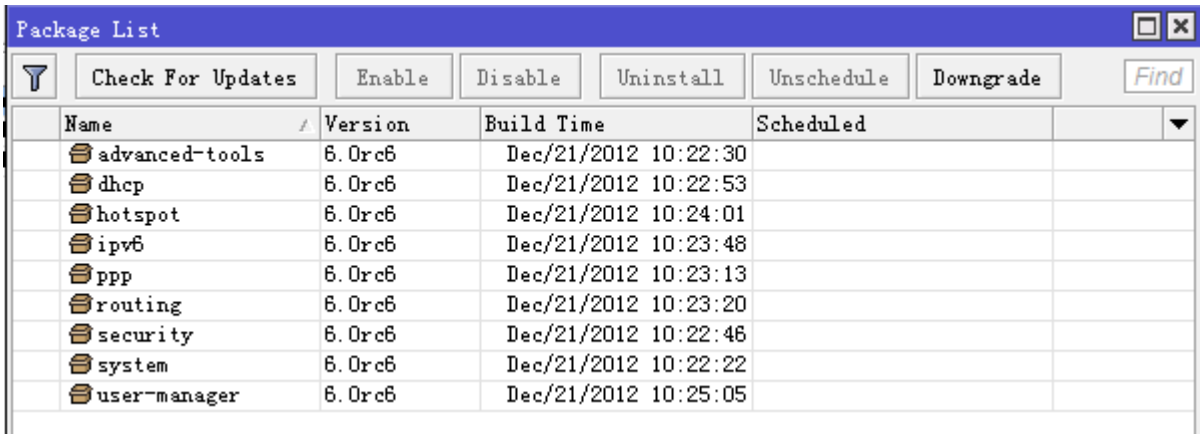
查看功能包

操作路径： **/system package**

在目录下执行命令生效仅在路由器重启后，即选择了对某一功能包进行操作后，必须正常重启路由器，执行的命令才能生效。

命令	属性
disable	计划下一次重启后禁用功能包，该功能提供的所有功能将无法获得
downgrade	降级 RouterOS 版本，会提示重启，在重启过程中将检查是否有低版本的 RouterOS 功能包上传到路由器，并试着降级 RouterOS
print	输出功能包信息，如版本、功能包状态和计划状态
enable	计划下一次重启后启用功能包
uninstall	计划下一次重启后从路由器删除功能包，
unschedule	为功能包取消计划任务

在 winbox 下查看功能包信息，进入 system packetage：



功能包操作事例

显示出可获得的功能包

```
[admin@MikroTik] > /system package print
Flags: X - disabled
 #    NAME                      VERSION                   SCHEDULED
 0 X ipv6                      6.0rc6
 1    system                    6.0rc6
```

```
2 X mpls                  6.0rc6
3 X hotspot               6.0rc6
4   routing               6.0rc6
5   wireless              6.0rc6
6 X dhcp                  6.0rc6
7   routerboard           6.0rc6
8   routeros-mipsle       6.0rc6
9   security              6.0rc6
10 X ppp                  6.0rc6
11  advanced-tools        6.0rc6
```

删除功能包，并重启

```
[admin@MikroTik] > /system package uninstall ppp;
[admin@MikroTik] >/system reboot;
Reboot, yes? [y/N]:
```

禁用功能包，并重启

```
[admin@MikroTik] > /system package disable hotspot;
[admin@MikroTik] >/system reboot;
Reboot, yes? [y/N]:
```

降级 RouterOS 版本，确定已将低版本的 RouterOS 安装包上传到路由器。

```
[admin@MikroTik] > /system package downgrade;
[admin@MikroTik] >/system reboot;
Reboot, yes? [y/N]:
```

取消删除和禁用的命令

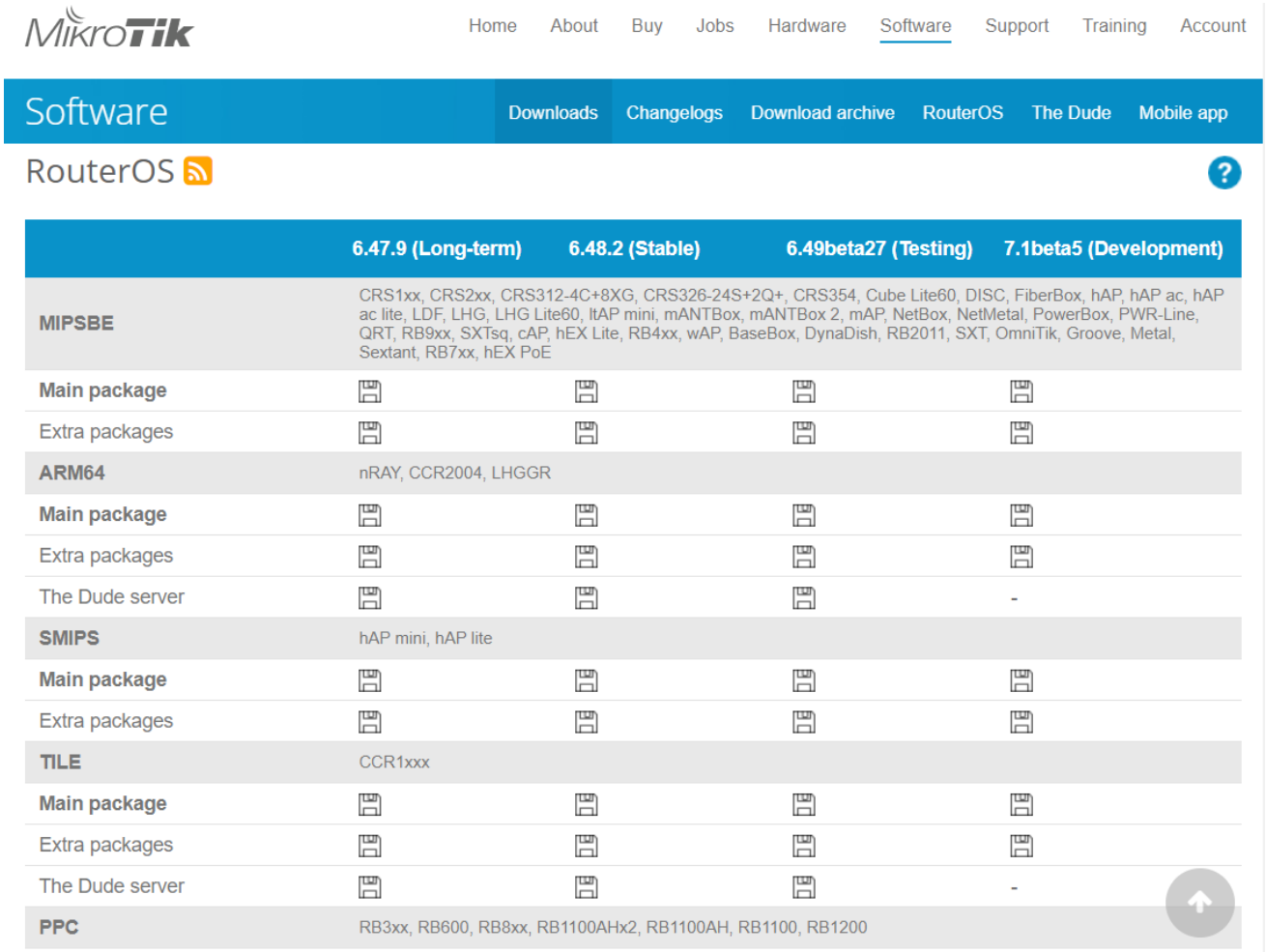
```
[admin@MikroTik] > /system package unschedule ipv6
```

2.8 升级和降级 RouterOS

在升级或降级操作前，我们需要了解各个 RouterOS 硬件版本区别，因为不同硬件版本对应了不同的升级档，升级或降级的功能包建议从 www.mikrotik.com/download 网站下载。

RouterOS 升级包区别

早期 MikroTik 官方每次发行新版本提供了所有硬件合计的 BT 下载，如“Routeros-ALL-6.0rc6”里面有多个硬件版本，每个文件对应不同的硬件做升级和降级设置，但 2015 年下半年官方已经取消了 BT 下载，提供不同平台的主升级包和扩展升级包下载，如下截图：



	6.47.9 (Long-term)	6.48.2 (Stable)	6.49beta27 (Testing)	7.1beta5 (Development)
MIPSBE	CRS1xx, CRS2xx, CRS312-4C+8XG, CRS326-24S+2Q+, CRS354, Cube Lite60, DISC, FiberBox, hAP, hAP ac, hAP ac lite, LDF, LHG, LHG Lite60, ltAP mini, mANTBox, mANTBox 2, mAP, NetBox, NetMetal, PowerBox, PWR-Line, QRT, RB9xx, SXTsq, cAP, hEX Lite, RB4xx, wAP, BaseBox, DynaDish, RB2011, SXT, OmniTik, Groove, Metal, Sextant, RB7xx, hEX PoE			
Main package				
Extra packages				
ARM64	nRAY, CCR2004, LHGGR			
Main package				
Extra packages				
The Dude server				-
SMIPS	hAP mini, hAP lite			
Main package				
Extra packages				
TILE	CCR1xxx			
Main package				
Extra packages				
The Dude server				-
PPC	RB3xx, RB600, RB8xx, RB1100AHx2, RB1100AH, RB1100, RB1200			

Main package 和 Extra packetages，两种功能个包都可以用于升级 RouterOS：

- **Main package**：集成主要功能的单 npk 包，能用于 netinstall 的安装。
- **Extra packages**：为 zip 压缩文件，将功能拆分为多个 npk 功能包，里面包含基本的 system 系统包和各种功能扩展包，根据功能需要选择上传升级，不能用于 netinsatll 的安装。

在版本上官方也提供了多种选择， v6 一共有三个版本可以选择，包括 Long-term、Stable 和 Testing，如上图显示 v6.47.9 为对之前一个版本的 bug 修正，v6.48.2 为当前发行稳定版本，v6.49beta27 为下一个测试候选版本。也就是用户可以选择三种不同的版本，beta 开发候选版本一般会有新功能加入，喜欢尝鲜的朋友可以更新测试，而 long-term 类似于老版本修补的稳定版。

下面是 Extra packetages 功能名称对应的各种 RouterOS 硬件型号，以下信息仅供参考，具体请参考官方信息 www.mikrotik.com/download

- **all_packages_mipsbe** - 对应所有 Atheros 芯片的 RB400、700、900、2011 系列产品和 hAP ac、RBSXT、OmniTIK、Groov 等采用 MIPS-BE
- **all_packages_mipsle** - 对应 RB100 系列和 RB500 系列（RB133、RB133c、RB150、RB192、RB532）MIPS 4Kc 芯片
- **all_packages_ppc -** 对应 RB300、RB600、RB800 和 RB1000 系列（RB333、RB600、RB800、RB1000、RB1100/AH/AHx2 和 RB1200）PowerPC 芯片
- **all_packages_x86** - 对应所有 x86 构架的 PC 设备（AMD、Intel、VIA 和其他 x86 PC）
- **all_packages_tile -** 对应基于 tilera-gx 构架的 CCR1016 和 CCR1036 系列

- **all_packages-smips** - 对应 hAP lite/mini 设备
- **all_packages-arm** – 对应 ARM 芯片，如 hAP ac2、RB3011、RB4011 和 RB1100x4 等设备
- **all_packages-arm64** – 对应 ARM64 位芯片，如 CCR2004、LHGGR 等

其他文件：

mikrotik-x.x.iso 光盘镜像文件，用于 x86 平台安装。

如果是早期 2.9 版本的 BT 档区分如下：

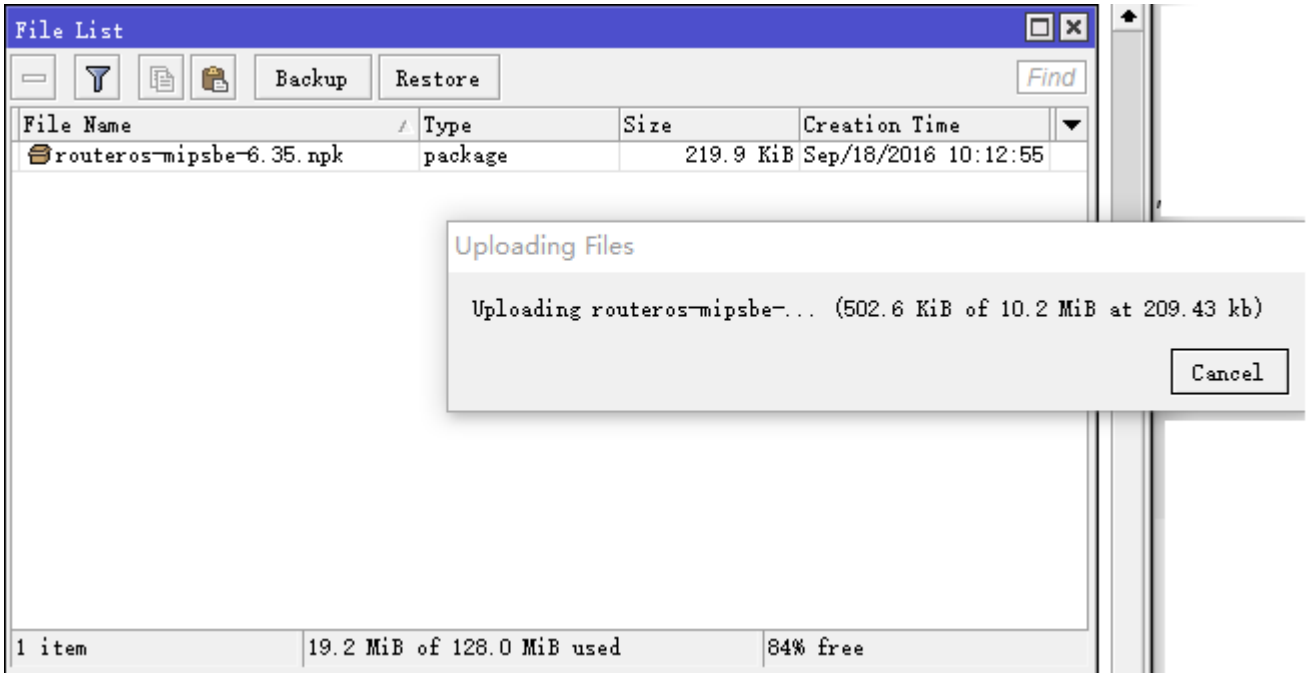
- **all_packages_ns** 对应 RB100 系列和 RB500 系列（RB133、RB133c、RB150、RB192、RB532）MIPS 4Kc 芯片
- **all_packages_x86** 对应所有 x86 构架的 PC 设备（AMD、Intel、VIA 和其他 x86 PC）

手动升级

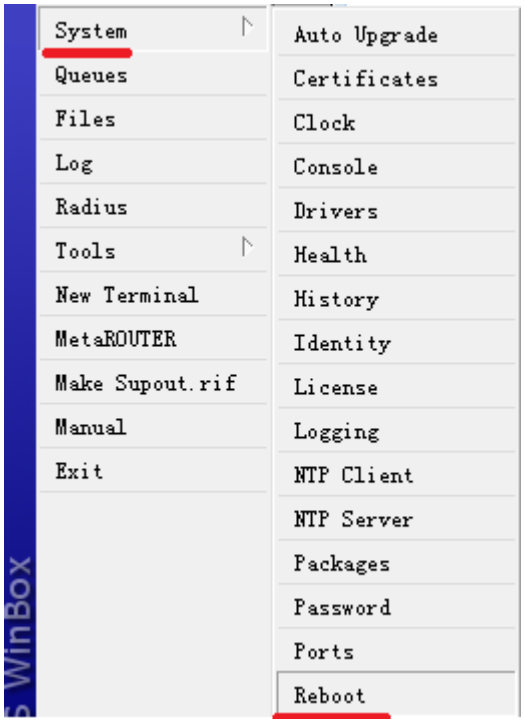
通过功能包升级，之前我们提到官网有两种功能包，一种是 Main package，另一种是 Extra packages，Main 是一个包含所有功能的整体 npk 文件，而 Extra packages 的 zip 档解压后，包含各种功能包，根据需要上传对应的功能扩展包。在升级操作很简单，只需要上传到 File 根目录下，然后通过 reboot 命令重启路由器可以完成升级操作(当然重启升级过程中不能断电)。

Main package 升级

例如：routeros-mipsbe-6.35.npk 的文件名，就是一个包含完成功能的 npk 文件（注意 routeros-mipsbe-xxx，是对应的 RB400、700、900、2011 系列产品和 RBSXT、OmniTIK、Groov 等），升级时 RouterOS 会自动判断当前你安装在 RouterOS 上有那些功能，会匹配 Main 升级包的对应功能，原有的功能不会丢失，也不会新增其他功能。下面是在 windows 计算机上，使用 winbox 通过拖放操作将 routeros-mipsbe-6.35.npk，上传到 RouterOS 的 File 根目录下，也可以在 ftp 服务开启的情况下，通过“FTP：//路由器 IP 地址”上传:



上传完成后，通 system reboot 命令正常重启 RouterOS，



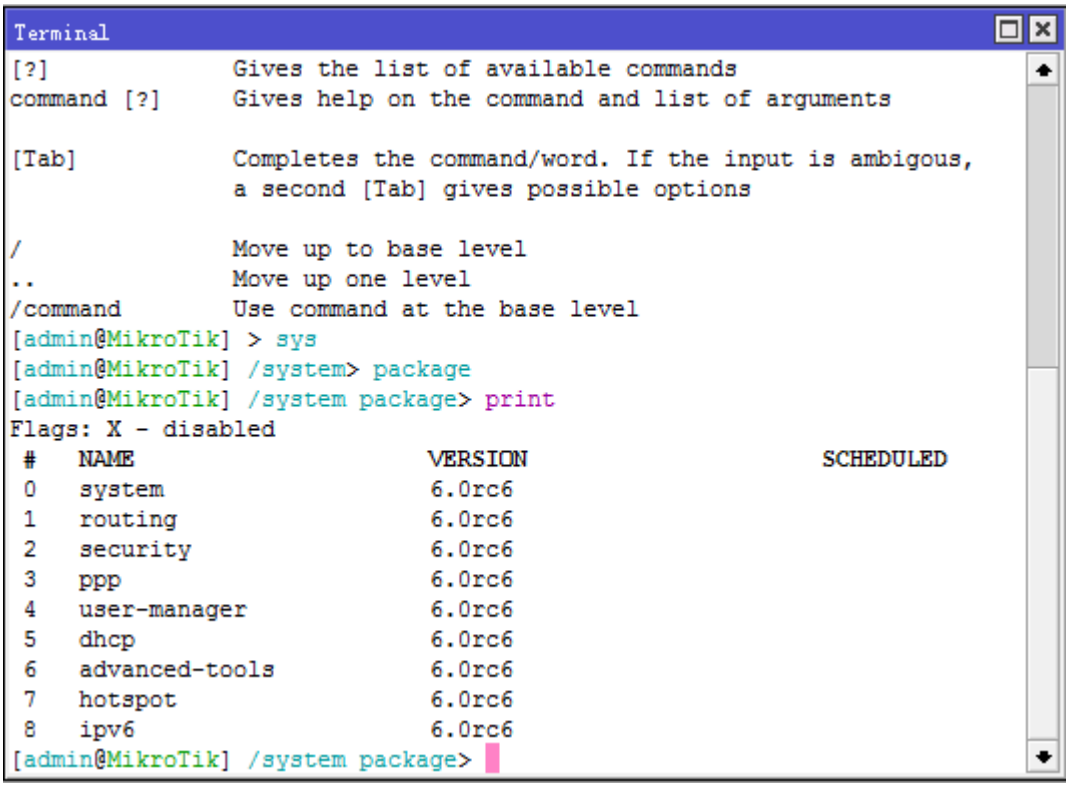
Extra packages 升级

使用 Extra packages 下载的功能包文件名如：all_packages-mipsbe-6.36.zip，解压后我们可以看到如下列表的档

电脑 › 新加卷 (D:) › RouterOS › all_packages-mipsbe-6.36

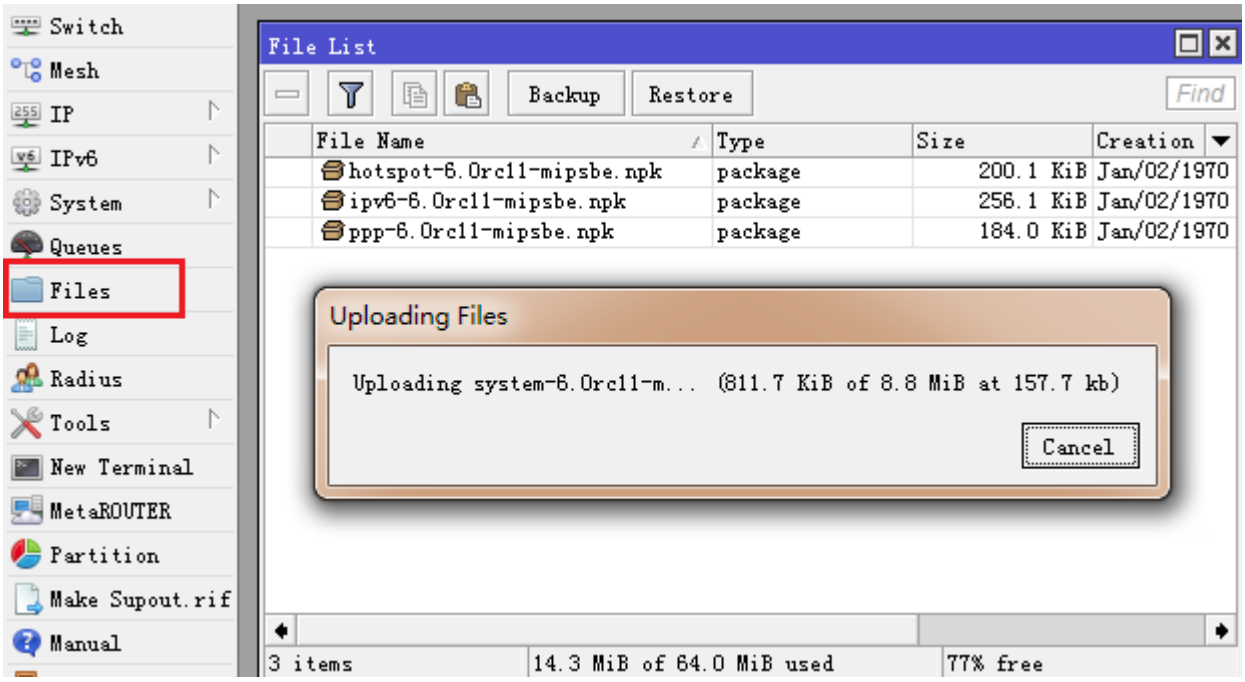
名称	修改日期	类型	大小
advanced-tools-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	101 KB
calea-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	21 KB
dhcp-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	181 KB
gps-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	21 KB
hotspot-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	189 KB
ipv6-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	237 KB
lcd-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	61 KB
lte-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	1,945 KB
mpls-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	101 KB
multicast-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	77 KB
ntp-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	265 KB
openflow-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	81 KB
ppp-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	309 KB
routing-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	129 KB
security-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	333 KB
system-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	7,010 KB
ups-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	69 KB
user-manager-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	853 KB
wireless-cm2-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	1,161 KB
wireless-rep-6.36-mipsbe.npk	2016/7/21 10:34	NPK 文件	1,193 KB

根据你使用 RouterOS 的情况不同，选择上传的升级包文件（注：system-x.x.x.npk 的升级包是必须要，否则无法升级）。如何来确定你当前使用的功能包，可以通过在 system package>的目录中查询对照如下图：

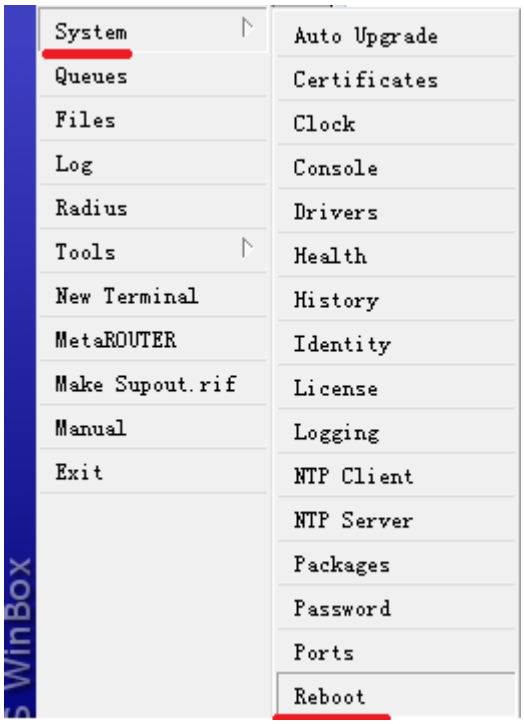


建议根据自己的需求安装或升级功能包(需要无线选择 wireless，需要 PPPoE 认证选择 PPP 等等)，如果你过多的安装一些不用的功能会影响路由器的性能。根据你在 system package 中的功能包选择，并选择对应的功能包进行升级，记住 **system** 功能包是必须选择安装。

选择好对应的 RouterOS 功能包后，通过“FTP：//路由器 IP 地址”上传功能包，或者直接打开 Winbox 的 Files 目录，通过拖放的方式将升级包上传到 RouterOS 根目录下：



功能包上传完成后，同样通过 System reboot 命令正常重启路由器，并升级版本。



RouterOS 在重启时，同时也在执行功能包的安装，重启后根据路由器性能不同会花费十几秒到 1 分钟的升级时间，如果是 PC 或者通过串口连接的 RB 设备可以在显示屏上看到安装进度条。重启完路由器后回看到路由器已经升级为新的版本。

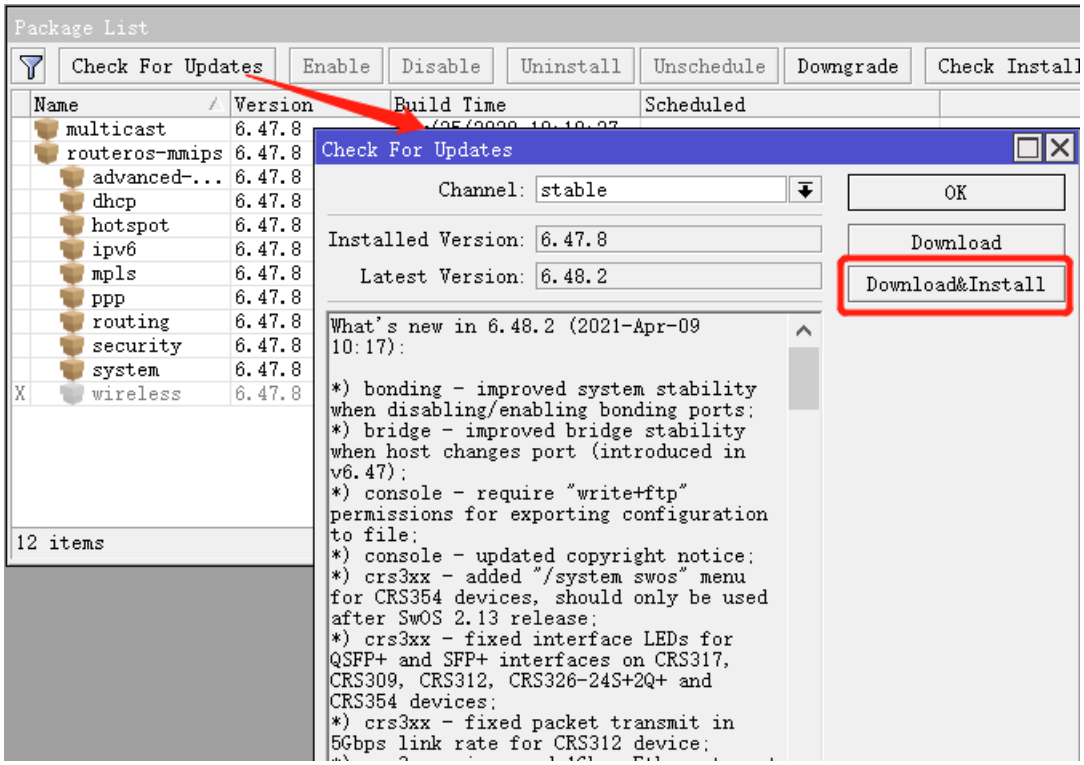
在线升级

RouterOS 6.x 具备在线升级的选项，只要你的 RouterOS 配置网络后能正常连接网络（DNS 配置正确），可以直接从官方下载并升级 RouterOS 最新版本，当前 RouterOS 的版本划分为开发版本、稳定版本、测试版本和 bug 修复版本：。

- **long term** – bug 修正版本，即对前一个版本的 RouterOS 做 bug 修复
- **statble** – 当前版本，即当前发行的相对稳定版本
- **testing** – 当前版本的一个测试版本。
- **development** – 正在开发的全新版本，主要提供测试。

因此在你选择版本升级时，一定要考虑使用哪一个版本，并看清楚官方的 changelog 信息，是否有你需要修复 bug 的地方或者你感兴趣的功能等。

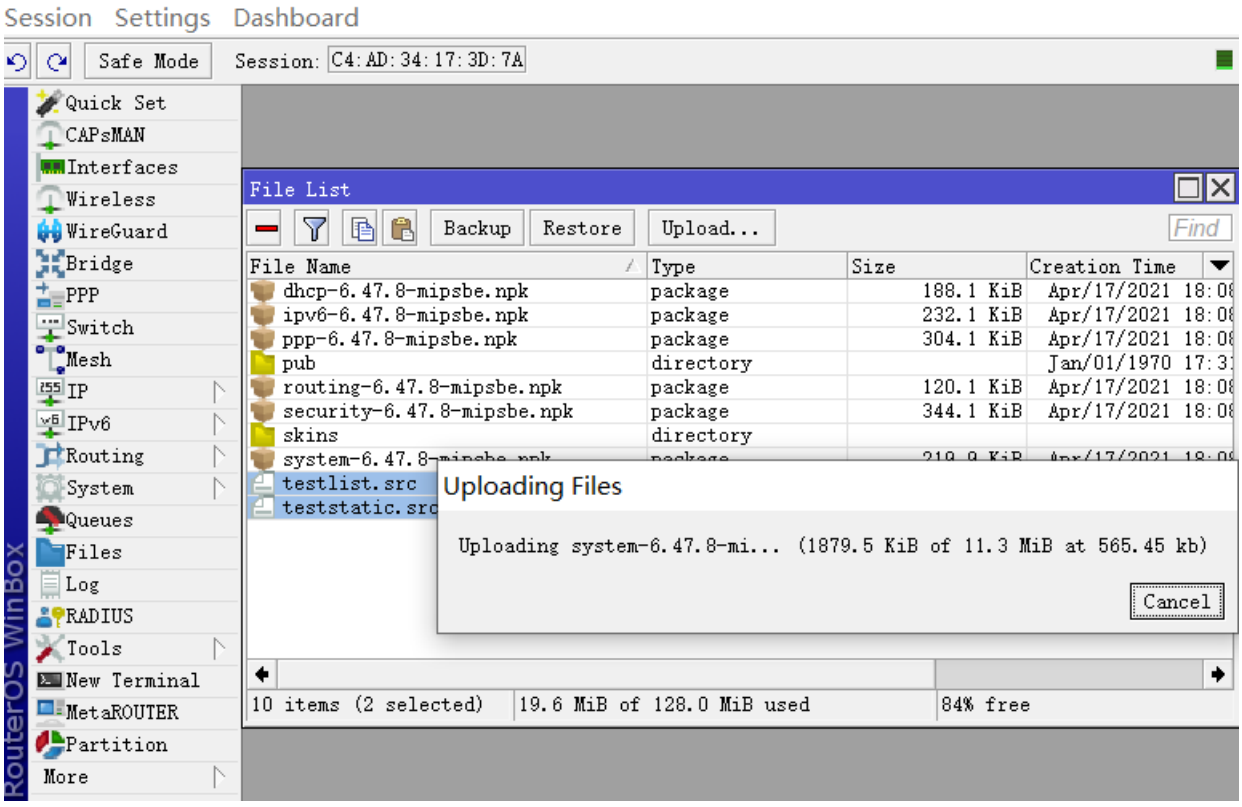
RouterOS 进入 system -> package 目录下选择 check for updates 进行在线升级，当你选择好你需要升级的版本，并点击 Download&Install，RouterOS 会自动连接远程的 MikroTik 升级服务器，注意：确保 RouterOS 已经联网，同时确认 RouterOS 的 DNS 配置正确。



版本降级

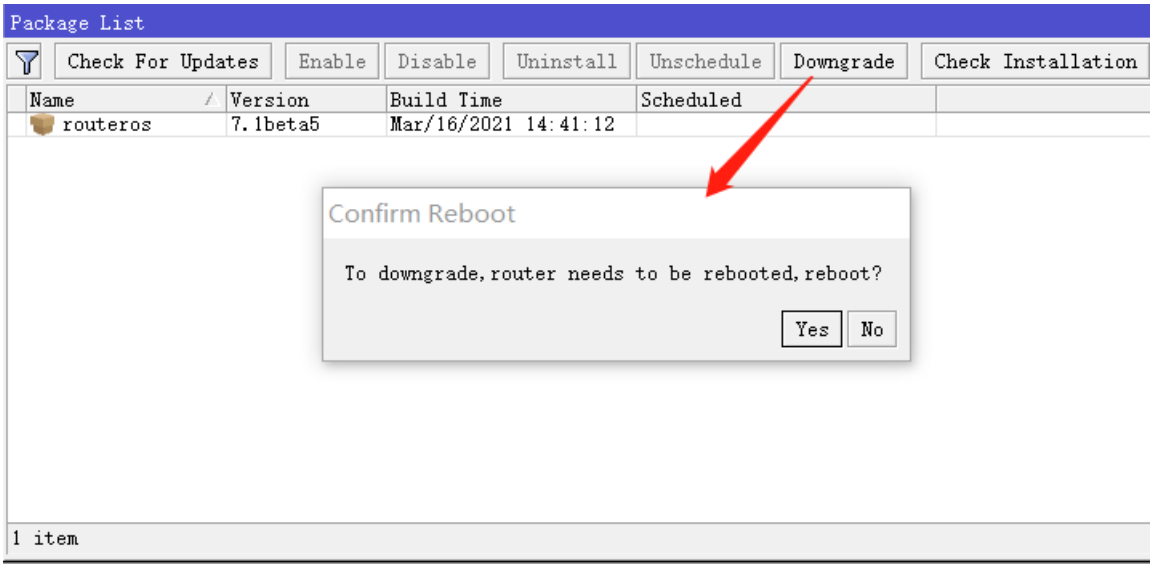
将 RouterOS 高版本降级到低版本，需要要将低版本的功能包上传到 RouterOS 的 Files 文件目录中，上传到 Files 目录通常支持两种方式：

1、通过 Winbox 将低版本的功能包拖到 Files 的**根**目录下：



2、在 IP service 中确认 ftp 服务开启，然后通过 ftp 上传低版本功能包文件到 File 目录下

上传功能包文件完成后，进入 System->package 下选择 Downgrade，并重启 RouterOS



注意：当前大多数 RB 设备的存储只有 16Mb，没有足够的大的空间上传降级的功能包，如果要降级，需要考虑选择 netinstall 重新安装。

2.9 SNTP client

SNTP 是简单网络时间协议(Simple Network Time Protocol)，SNTP client 是 NTP 客户端，该功能集成在默认的系统中（system package）。NTP 服务器和 NTP 客户端集成在 ntp 功能包中（ntp package），需要选择安装。

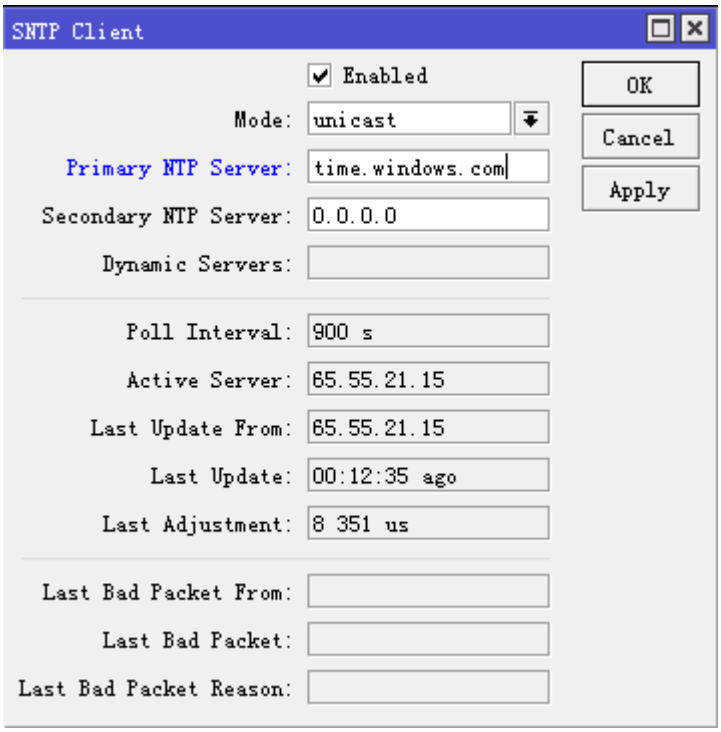
操作路径 **/system ntp client**

由于 system 功能默认安装了 sntp 客户端，当 ntp 功能包安装并启用后，SNTP 客户端将自动禁用，默认启用 ntp 功能包选项

- **enabled** (*yes* 或 *no*; 默认：*no*)：是否启用
- **mode** (*broadcast* 或 *unicast*; 默认: *broadcast*) ： 在 broadcast 模式中，客户端不会发送任何请求，只监听来至 NTP 服务器的广播信息。在 unicast 模式下，客户端定期发送请求到当前选择的服务器，并等待从服务器的回复信息。
- **primary-ntp**, **secondary-ntp** (IP 地址) ：NTP 服务器的 IP 地址。这个属性仅作 mode=unicast 模式下生效。当值设置为 0.0.0.0 视为空，如果两个参数都为空，SNTP 客户端将自动禁用。如果两个参数非空状态，SNTP 将会在两个服务地址中选择发送请求。

对于 RouterOS 设置 ntp 客户端来说是非常有必要的，特别是 RouterBOARD 设备，RouterBOARD 不像 x86 设备，主板都带有 BIOS 电池，能保持时间和信息，由于 RouterBOARD 设备不含电池，即断电后 BIOS 时间将无法保持，只能被清空复位，所以通过 ntp 客户端能在开机后通过网络连接到时间服务器，并同步当前时间。

如果你没有 NTP 服务器，或不知道 NTP 服务器的地址，可以设置 windows 自带的 NTP 服务器“time.windows.com”，使用域名填写请保证 dns 能正常解析。



注：RouterOS 在 6.16 后，新增了 RouterBOARD 时间保存功能，以前我们设置好 RouterBOARD 时间后（如不开 sntp 客户端），一旦重启，时间恢复到 1970 年，现在时间会在重启后保存和做时钟调整，并在重启完成后做时间初始化。这样保证了大家使用 RouterBOARD 在没有网络情况下不用担心设置完时间后，重启无法保存的问题。但需注意，重启保存时间仅是当前重启时间，如果长时间不开机，时间越长误差越大。当做复位和重装 RouterOS 后，时间同样会恢复到 1970 年。

注：MikroTik 硬件设备和 CHR 提供了 IP Cloud 功能，也提供时间同步，即 MikroTik 官方提供的 ntp 对时，无需通过 ntp 的第三方平台。

2.10 telnet 和 ssh 客户端

RouterOS 在 system 下提供了 telnet 和 SSH 客户端工具，便于能对网络中的其他设备进行远程登录管理。

telnet 客户端

RouterOS 提供 telnet 客户端远程登录到对端设备，操作路径在/system telnet，连接 ip 地址支持 IPv4 和 IPv6

```
/system telnet 192.168.88.1
/system telnet 2001:db8:add:1337::beef
```

telnet 客户端能指定端口登录对端设备

```
/system telnet 192.168.88.1 port=2323
```

SSH 客户端

RouterOS 提供 SSH 客户端，支持 SSHv2 登录到其他 SSH 服务器，注意：SSH 客户端需要 security 功能包支持，SSH 连接到远程主机，连接 ip 地址支持 IPv4 和 IPv6

```
/system ssh 192.168.88.1
/system ssh 2001:db8:add:1337::beef
```

连接到远程主机也可以在后面加上 user 参数，写入远程主机登录用户名

```
/system ssh 192.168.88.1 user=root
/system ssh 2001:db8:add:1337::beef user=root
```

当路由器有多个 ip 时，可以指定登录远程主机 SSH 的源 ip 地址，通过 src-address 指定源 ip，如下面事例将源地址设置为 192.168.89.2 去登录 192.168.88.1 的 SSH 主机

```
/system ssh 192.168.88.1 src-address=192.168.89.2
/system ssh 2001:db8:add:1337::beef src-address=2001:db8:bad:1000::2
```

当对端 SSH 设备为了保证安全修改了 SSH 登录端口，也可以通过 port 设置登录端口

```
/system ssh 192.168.88.1 src-address=192.168.89.2 port=222
```

执行远程命令

RouterOS SSH 提供了将命令远程发送的功能，通过""在登录方式后，加入对端 RouterOS 执行的命令

```
/system ssh 192.168.88.1 "/ip address print"
/system ssh 192.168.88.1 command="/ip address print"
/system ssh 2001:db8:add:1337::beef "/ip address print"
/system ssh 2001:db8:add:1337::beef command="/ip address print"
```

注意：如果服务器不支持 pseudo-tty（ssh-T 或 ssh 主机命令），例如 RouterOS 的 SSH 服务，将不能通过 SSH 发送多行命令，例如"/ip address \n add address=1.1.1.1/24"这样多行命令，是不能发生到 RouterOS。

2.11 IP Service（应用服务端口）

MikroTik RouterOS 提供常用的网络服务，如常用的 telnet、SSH、WWW、FTP 和 winbox 等，用于管理 RouterOS，默认情况下这些端口大部分处于启用状态，出于安全考虑这些接口允许禁用或修改端口和限制访问 IP 地址。

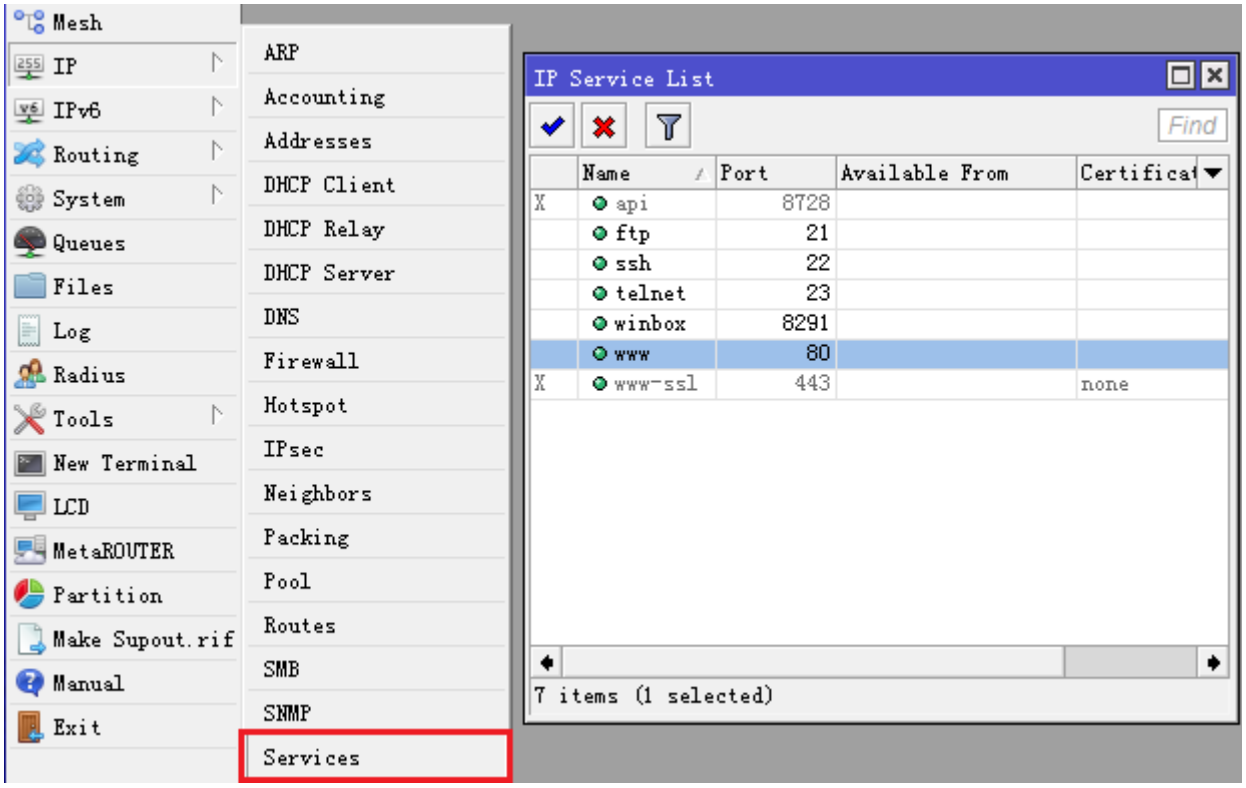
操作路径: ***/ip service***

服务端口如下：

服务协议	描述
telnet	Telnet 服务
ftp	FTP 服务

www	Webfig http 服务
ssh	SSH 服务
www-ssl	Webfig https 服务
api	API 接口服务
winbox	Winbox 服务接口，也可用于 Tik-App 手机 APP 应用和 Dude 服务
api-ssl	SSL 服务的 API 接口

以上只允许修改现有的服务端口，无法新增其他服务，进入 winbox 情况：



配置属性描述

属性	描述
address (*IP 地址 / 掩码 | IPv6/0..128*; 默认:)	设置允许访问的 IPv4/IPv6 地址
certificate (*name*; 默认: **none**)	服务使用的证书的名称。仅适用于依赖证书的服务的 www-ssl 和 api-ssl
name (*name*; 默认: **none**)	协议服务名称
port (*integer: 1..65535*; 默认:)	协议服务监听端口号

service 服务配置

修改 WWW 服务要求只能从 10.10.10.0/24 段的 8081 端口访问，首先查看当前 service 接口情况：

```
[admin@MikroTik] /ip service> print
Flags: X - disabled, I - invalid
 #    NAME       PORT ADDRESS                                 CERTIFICATE
 0    telnet       23
 1    ftp          21
 2    www            80
 3    ssh           22
 4 X www-ssl     443                                          none
 5 X api        8728
 6    winbox     8291
[admin@MikroTik] /ip service>
```

设置 www 端口为 8081，允许 10.10.10.0/24 访问，其他地址均被拒绝

```
[admin@MikroTik] /ip service> set www port=8081 address=10.10.10.0/24
```

查看设置结果：

```
[admin@MikroTik] /ip service> print
Flags: X - disabled, I - invalid
 #    NAME      PORT    ADDRESS                                 CERTIFICATE
 0    telnet      23
 1    ftp         21
 2    www       8081    10.10.10.0/24
 3    ssh         22
 4 X www-ssl    443                                          none
 5 X api       8728
 6    winbox    8291
[admin@MikroTik] /ip service>
```

指定 API 服务允许访问的 IPv4 和 IPv6 地址

```
[admin@dzeltenais_burkaans] /ip service> set api address=10.5.101.0/24,2001:db8:fade::/64
[admin@dzeltenais_burkaans] /ip service> print
Flags: X - disabled, I - invalid
 #    NAME      PORT   ADDRESS                                 CERTIFICATE
 0    telnet    23
 1    ftp       21
 2    www         80
 3    ssh       22
 4 X www-ssl   443                                           none
 5    api       8728   10.5.101.0/24
                       2001:db8:fade::/64
 6    winbox    8291
```

常用服务清单

以下是常用的协议和端口的清单，有助于对网络协议和端口使用的了解：

端口/协议	描述
20/tcp	文件传输协议 FTP [数据连接]
21/tcp	文件传输协议 FTP [控制连接]
22/tcp	安全命令行解释 SSH 远程登录协议 (仅与安全封装一起)
23/tcp	远程通信网络协议
53/tcp	域名服务器 DNS
53/udp	域名服务器 DNS
67/udp	自举协议 或 DHCP 服务器 (仅与 dhcp 功能包一起)
68/udp	自举协议 或 DHCP 客户 (仅与 dhcp 功能包一起)
80/tcp	万维网（WWW）HTTP
123/udp	网络时间协议 NTP (仅与 ntp 功能包一起)
161/udp	简单网络管理协议 SNMP (仅与 snmp 功能包一起)
443/tcp	安全接口层 SSL 加密 HTTP(仅与 hotspot 功能包一起)
500/udp	Internet Key Exchange IKE protocol (仅与 ipsec 功能包一起)
520/udp	选路信息协议 RIP (仅与路由功能包一起)
521/udp	选路信息协议 RIP (仅与 routing 功能包一起)
179/tcp	边界网关协议 BGP (仅与 routing 功能包一起)
1080/tcp	SOCKS 代理协议
1701/udp	Layer 2 Tunnel Protocol L2TP (仅与 ppp 功能包一起)
1718/udp	H.323 Gatekeeper Discovery (仅与 telephony 功能包一起)
1719/tcp	H.323 Gatekeeper RAS (仅与 telephony 功能包一起)
1720/tcp	H.323 呼叫安装 (仅与 telephony 功能包一起 e)
1723/tcp	点对点隧道协议 PPTP (仅与 ppp 功能包一起)
1731/tcp	H.323 音频呼叫控制 (仅与 telephony 功能包一起)
1900/udp	通用即插即用 uPnP
2828/tcp	通用即插即用 uPnP
2000/tcp	带宽测试服务器 bandwidth-test
3986/tcp	Winbox 代理
3987/tcp	安全 winbox SSL 代理 (仅与安全功能包一起)
5678/udp	MikroTik Neighbor Discovery Protocol

8080/tcp	HTTP 网络协议（仅与 web proxy 功能包一起）
8291/tcp	Winbox
20561/udp	MAC winbox

2.12 Note 笔记

系统笔记功能（system note）让你设置任意文本和信息，当每次登录 terminal 会显示文本信息。通过文本可以发布警告信息给不同的管理员，也可以提示详细的路由器操作配置。可以通过 FTP 上传文本文件命名为 sys-not.txt，或通过 edit 在/system note 编辑。

操作路径: /system note

note (字符; 默认:) 显示的笔记内容.
show-at-login (*yes* | *no*; 默认: **yes**) 是否在每次登录显示笔记内容

事例

下面通过 note 功能编辑多行的文本，可以使用 ASCII 码编辑

```
[admin@MikroTik] /system note> edit note
```

编辑内容：

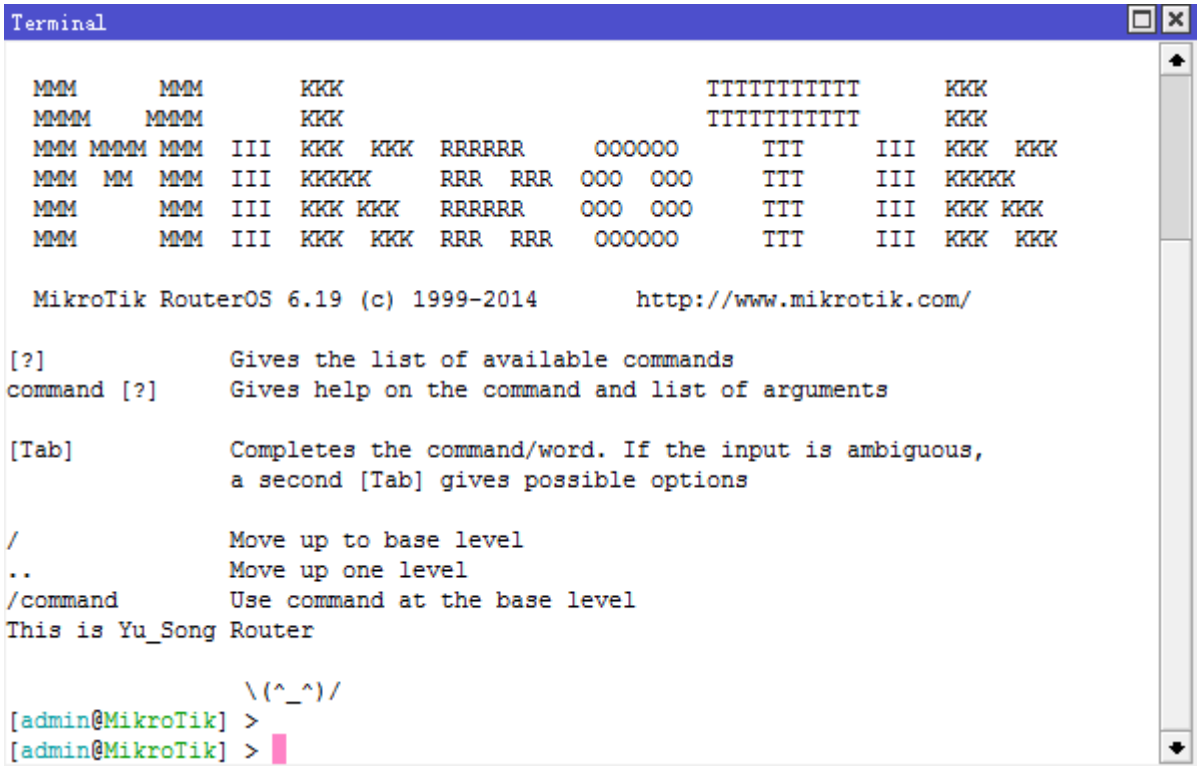
```
This is Yu_Song's Router

                    \(^_^)/




C-c quit C-o save&quit C-u undo C-k cut line C-y paste F5
```

在文本编辑下方有退出（Ctrl+C）、保存（Ctrl+O）、撤销（Ctrl+U）、剪切（Ctrl+K）、粘贴（Ctrl+Y）等功能

设置在 terminal 登录时显示

```
[admin@MikroTik] /system note> set show-at-login=yes
```

设置完成后，打开 terminal 显示如下：



2.13 Log 日志管理

任何网络设备都具备系统日志记录的功能，系统会监控并记录主要的设备运行和调试情况，RouterOS 会记录下不同的系统事件和状态信息。日志记录默认保存在内存中，当系统重启后会自动清除，日志可选择存储到硬盘或者利用记录 syslog 的软件，将日志存储到远程的服务器。

友情提示：日志查询是溯源故障的基本大法，因此作为网络工程师，学会查看日志是最基本的职业素养。

Log（日志）

RouterOS 中的日志有不同的分组或项目，日志信息来至于系统各个功能的运行状态。系统默认日志存储到内存中，在/log 下可以查看，当 RouterOS 系统重启或者断电后日志会丢失，如果要保存日志可以在 system logging 里配置日志记录类型到本地磁盘。

操作路径：***/log***

属性描述

message (*只读: 文本*) – 信息文本
time (*只读: 文本*) – 事件的日期和时间
topics (*只读: 文本*) – 项目信息的从属

所有日志信息默认都存放到路由器本地内存中，查看路径/log，每条日志包含发生的日期时间、标题和日志内容，如下进入 log 菜单后，使用 print 命令查看：

```
[admin@MikroTik] /log> print
jan/02/1970 02:00:09 system,info router rebooted
```

```
sep/15 09:54:33 system,info,account user admin logged in from 10.1.101.212 via winbox
sep/15 12:33:18 system,info item added by admin
sep/15 12:34:26 system,info mangle rule added by admin
sep/15 12:34:29 system,info mangle rule moved by admin
sep/15 12:35:34 system,info mangle rule changed by admin
sep/15 12:42:14 system,info,account user admin logged in from 10.1.101.212 via telnet
sep/15 12:42:55 system,info,account user admin logged out from 10.1.101.212 via telnet
01:01:58 firewall,info input: in:ether1 out:(none), src-mac 00:21:29:6d:82:07, proto UDP,
10.1.101.1:520->10.1.101.255:520, len 452
```

如果日志内容是在当天发生的，那么只会显示时间，未显示日期。在上面的例子中，可以看到第二个消息是在当年 9 月 15 日发生(年份没有添加)，最后一个消息是今天发生的，所以只显示时间。

例如，下面的命令，通 print 查看标题是 info 的所有日志消息，并探测新日志，直到按下 Ctrl+C 结束

```
[admin@MikroTik] /log > print follow where topics~".info"
12:52:24 script,info hello from script
-- Ctrl-C to quit.
```

如果 print 是在 follow 模式下，你可以在键盘上按“空格”插入分隔符:

```
[admin@MikroTik] /log > print follow where topics~".info"
12:52:24 script,info hello from script

 = = =   = = =   = = =     = = =   = = =   = = =     = = =   = = =   = = =

-- Ctrl-C to quit.
```

Logging 日志管理

logging 是用于管理各项 RouterOS 的系统运行和 debug 记录日志，即 logging 是用于管理在/log 显示和存储日志内容信息的方式，日志可以用于系统分析和故障排查。

操作路径: ***/system logging***

属性描述

属性	描述
action (*name*; 默认: **memory**)	指定系统日志 action 方式，或由用户在 action 菜单下自定义执行日志方式
prefix (*string*; 默认:)	在日志信息添加前缀说明
topics (*account, bfd, caps, ddns, dns, error, gsm, info, iscsi, l2tp, manager, ntp, packet, pppoe, radvd, rip, script, smb, sstp, system, timer, vrrp, web-proxy, async, bgp, certificate, debug, dot1x,*	记录属于指定标题分类的所有信息。 ”!字符可用在标题分类之前，以排除属于这个标题的信息。例如，想记录 NTP 调试信息而不需要太多细节: /system logging add topics=ntp,debug,!packet

dude, event, hotspot, interface, isdn, ldp, mme, ospf, pim, pptp, raw, route, sertcp, snmp, state, telephony, upnp, warning, wireless, backup, calc, critical, dhcp, e-mail, firewall, igmp-proxy, ipsec, kvm, lte, mpls, ovpn, ppp, radius, read, rsvp, simulator, ssh, store, tftp, ups, watchdog, write; 默认: **info**)	

默认情况下日志是被存储在内存中，我们进入/system logging 查看，默认的进行日志类型 action 都是 memory，即存储在内容，当路由器重启后会自动清除。

```
[admin@MikroTik] /system logging> print
Flags: X - disabled, I - invalid, * - default
 #     TOPICS                                  ACTION                                  PREFIX
 0   * info                                    memory
 1   * error                                   memory
 2   * warning                                 memory
 3   * critical                                echo
[admin@MikroTik] /system logging>
```

相关属性：

Topic	描述
critical	标记为关键的日志项，在每次登录时都被显示到到控制台
debug	Debug 日志项
error	错误日志信息
info	事件信息日志
packet	显示接收/发送数据包内容的日志项
raw	显示接收/发送 raw 内容的日志项
warning	警告日志信息.

为保证日志内被存储下来，并汇出用于分析，可设置日志存在到本地磁盘中，我们将 info 日志类型设置 action=disk，即存储在本地磁盘，配置如下：

```
[admin@MikroTik] /system logging> set 0 action=disk
[admin@MikroTik] /system logging> print
Flags: X - disabled, I - invalid, * - default
 #     TOPICS                                  ACTION                                  PREFIX
 0   * info                                    disk
 1   * error                                   memory
 2   * warning                                 memory
 3   * critical                                echo
[admin@MikroTik] /system logging>
```

我们可以在 RouterOS /files 目录下找到 log.0.txt 文件，该档可以通过 winbox 或者 ftp 下载

```
[admin@MikroTik] /system logging> /file print
 # NAME                                TYPE                                SIZE    CREATION-TIME
 0 skins                               directory                                   aug/12/2013 22:30:24
 1 user-manager1                       user-manager store                          aug/13/2013 04:42:47
 2 disk1                               disk                                        jan/24/2016 16:39:44
 3 dude                                dude store                                  feb/02/2016 04:45:41
 4 log.0.txt                           .txt file                           118     feb/18/2016 10:26:29
[admin@MikroTik] /system logging>
```

默认日志记录中，只有 info、error、warnimg、critical 等 4 种日志类型，我们可以通过手动添加方式，将 firewall 日志记录到 log 中，下面是在 logging 中添加 firewall 产生的日志信息，并存储到系统内存中。

```
[admin@MikroTik] system logging> add topics=firewall action=memory
[admin@MikroTik] system logging> print
 Flags: X - disabled, I - invalid
 #    TOPICS                                                ACTION PREFIX
 0    info                                                  memory
 1    error                                                 memory
 2    warning                                                memory
 3    critical                                              echo
 4    firewall                                               memory
[admin@MikroTik] system logging>
```

Log 管理执行方式

Log 管理执行方式，是设置执行存储时分割日志文件大小，文件长度或远程发送等功能参数。

操作: **/system logging action**

属性描述

disk-lines (整型; 默认: **100**) – 在日志文件存储到硬盘的记录数量(仅在 action 设置为 **disk**)
disk-stop-on-full (yes | no; 默认: **no**) – 是否在 disk-lines 数量达到后停止存储日志信息
email-to (*名称*) – 发送到指定的 email 地址 (仅在 action 设置为 **email**)
memory-lines (*整型*; 默认: **100**) – 在本地缓存记录的数量(仅在 action 设置为 **memory**)
memory-stop-on-full (yes | no; 默认: **no**) -是否在 memory-lines 数量达到后停止存储日志信息
name (*名称*) – 一个 action 操作的名称
remember (yes | no; 默认: **yes**) – 是否保存日志信息， 其中尚未显示在控制面板的 (仅在 action 设置为 **echo**)
remote (*IP address*:*port* ; 默认: **0.0.0.0:514**) – 远程日志服务器的 IP 地址和 UDP 端口(仅在 action 设置为 **remote**)
target (disk | echo | email | memory | remote; 默认: **memory**) – 记录存储设备或目标
- **disk** – 日志记录到硬盘
- **echo** – 日志显示在控制面板屏幕上
- **email** – 日志通过 email 发送

memory – 日志被存储到本地内存
remote – 日志发送到远程服务主机

注：你不能删除或重命名默认 action 规则

添加一个新的 action 取名为 long，将日志记录到本地内存，在内存中的记录为 1000 条，这样在**/log** 中会显示 1000 条记录，用于查看很多的信息：

```
[admin@MikroTik] system logging action> add name=long \
\... target=memory memory-lines=50 memory-stop-on-full=yes
[admin@MikroTik] system logging action> print
Flags: * - default
 #    NAME                                TARGET REMOTE
 0 * memory                               memory
 1 * disk                                 disk
 2 * echo                                 echo
 3 * remote                               remote 0.0.0.0:514
 4    long                                memory
[admin@MikroTik] system logging action>
```

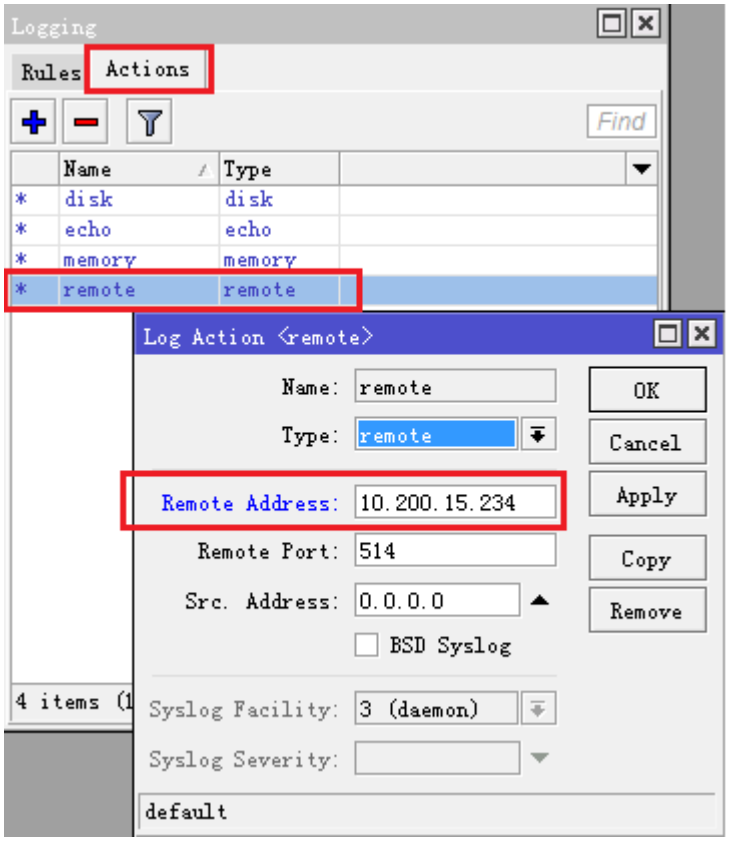
通过 ip firewall filter 记录所有访问 80 端口，并在 log 中添加前缀“80port”的相关信息：

```
[admin@MikroTik] /ip firewall filter>   add chain=forward protocol=tcp dst-port=80
 action=log log-prefix=80port
[admin@MikroTik] /ip firewall filter> print
Flags: X - disabled, I - invalid, D - dynamic

 0    chain=forward action=log protocol=tcp dst-port=80 log-prefix="80port"
```

使用 The Dude 管理器记录系统日志

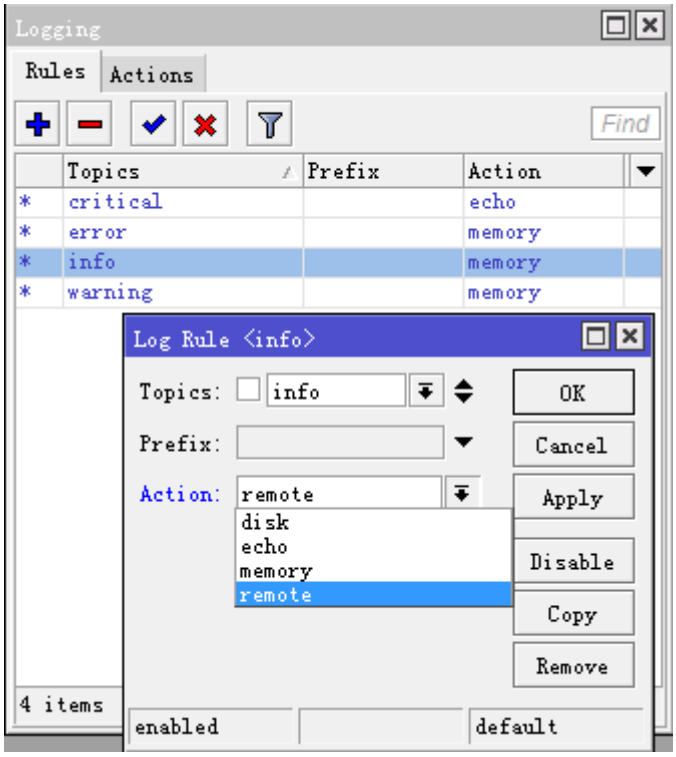
在 MikroTik 提供的 syslog 软件，用于记录 RouterOS 的系统日志信息，但这个软件只能记录 1000 条，不能做长时间记录和定期存储。在新版本的 The Dude 网络管理软件中增加了系统日志记录和下载的功能，这个我们可以通过 The Dude 管理器对我们需要的 RouterOS 日志信息进行记录和管理。

这里我们使用的是 The Dude 3.0beta8 的版本，首先我们需要进入 RouterOS 的 system logging 配置系统日志的远程记录参数：

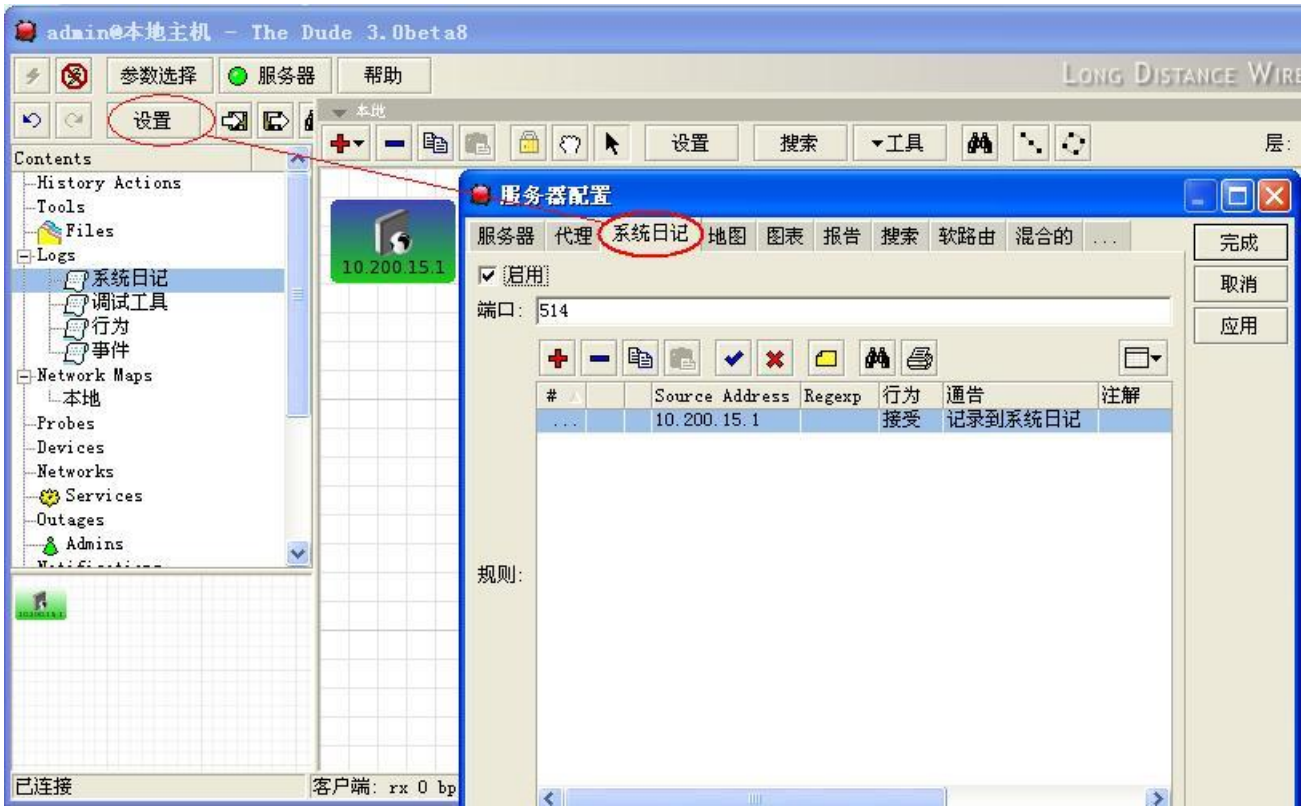


进入 system logging 后配置 actions 里的 remote 参数，将 Remote Address 配置为指定的系统信息接受的 The Dude 服务器 IP 地址。

然后将需要记录的日志信息，设置为 remote：

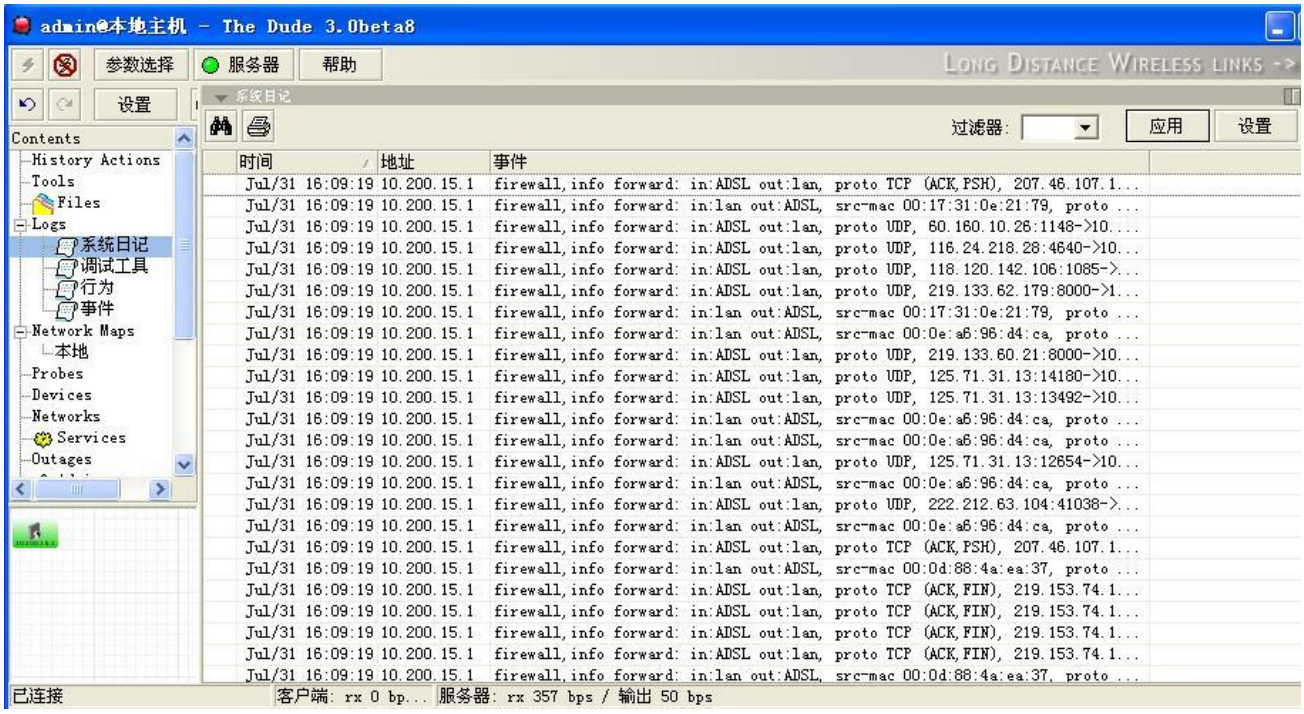


下面是在 10.200.15.234 的 The Dude 网络管理器上配置系统日志信息，进入“设置”，选择“系统日记”，并配置相应的端口和 IP 地址。



配置 The Dude 系统日志记录参数：

配置完成后，我们在 The Dude 管理器的 log 下系统日记看到接受到 IP 地址为 10.200.15.1 的 RouterOS 的日志：



在 The Dude 的 log 记录中，有一个“设置”选项，可以配置日志记录的存储参数：

这里我们设置日志记录存储的文件名字、产生新的日志文件时间间隔等参数。

通过 The Dude 网络管理软件，可以方便的记录每台 RouterOS 的日志信息和情况，同时达到监控的目的。关于 The Dude 网络监控软件的功能介绍可以参考 **The Dude 中文实用教程**

2.14 IP Accounting 访问日志记录

IP 访问日志记录，用于内向外或者外向内发送的所有连接（包括源地址、目标地址、数据报和字节）都会被记录下来。同时当启用 Hotspot 和 PPP 认证时，账号也随之被记录到相应的连接中。在 RouterOS 中主要应用于记录内网与外网之间的访问日志，以便对网络中的所有数据作记录进行检查和分析，或者出于安全考虑为以后非法连接提供依据。

规格

功能包要求：***system***
认证等级：*Level1*

操作路径: ***/user***, ***/ppp***, ***/ip accounting***, ***/RADIUS***
硬件使用: 传输记录需要根据记录内容大小增加内存

IP 访问记录

操作路径: ***/ip accounting***

当每个包通过路由器时，匹配 IP 数据报源和目的地址会成对的在访问列表中并且这个对的流量会增加。PPP，PPTP，PPPoE，ISDN，以及 HotSpot 客户的流量也可以在每个用户的基础上计算。数据报的数量和字节的数量都会被计算。如果没有与之前的 IP 或用户对匹配，那么新的记录将被添加到里表中。

属性描述

enabled (yes | no; 默认: **no**) - 是否启用了本地 IP 访问记录日志
account-local-traffic (yes | no; 默认: **no**) - 是否计算来自/到达路由器的流量访问
threshold (*整型*; 默认: **256**) - 在管理列表中的 IP 对的最大数量（最大值为 8192）

当临界值限制达到时，没有新的 IP 对将被添加到管理列表中。在管理列表中没有计算的每个包都将被添加到 **uncounted** 计数器。启用 IP 访问管理：

```
[admin@MikroTik] ip accounting> set enabled=yes
[admin@MikroTik] ip accounting> print
                       enabled: yes
       account-local-traffic: no
                   threshold: 256
[admin@MikroTik] ip accounting>
```

IP 访问快照

操作路径: ***/ip accounting snapshot***

当数据收集的快照做好后，管理列表会被清空并且新的 IP 对与流量数据会被添加近来。

属性描述

bytes (*只读: 整型*) - 字节总数，以条目匹配
dst-address (*只读: IP address*) - 目的 IP 地址
dst-user (*只读: 文本*) – 接受者的名称(如果可应用)
packets (*只读: 整型*) - 包的总数，以这个条目匹配
src-address (*只读: IP address*) - 源 IP 地址
src-user (*只读: 文本*) – 发送者的名称（如果可用）

注：仅当用户通过一个 PPP 隧道连接到路由器或被 HotSpot 认证时才显示用户名。在获取快照之前，列表是空的。

取一个新 IP 访问的快照：

```
[admin@MikroTik] ip accounting snapshot> take
[admin@MikroTik] ip accounting snapshot> print
```

```
 # SRC-ADDRESS      DST-ADDRESS       PACKETS     BYTES       SRC-USER     DST-USER
 0 192.168.0.2      159.148.172.197 474           19130
 1 192.168.0.2      10.0.0.4              3         120
 2 192.168.0.2      192.150.20.254       32         3142
 3 192.150.20.254   192.168.0.2          26         2857
 4 10.0.0.4         192.168.0.2           2          117
 5 159.148.147.196  192.168.0.2           2          136
 6 192.168.0.2      159.148.147.196       1          40
 7 159.148.172.197  192.168.0.2         835          1192962
[admin@MikroTik] ip accounting snapshot>
```

Web 获取 IP 访问信息

操作路径: ***/ip accounting web-access***

web 页面报告似的使用标准的 Unix/Linux wget 工具收集流量数据并存储到文件或者使用 MikroTik 的日志下载软件。如果 web 报告启用且 web 页面被查看，那么当连接起始为 web 页面时，**snapshot** 将被生成。**Snapshot** 将在 web 页面上显示。被有 wget 工具 http 连接使用的 TCP 协议保证任何一点的流数据都不会丢失。**Snapshot** 图像将在来自 wget 的连接被初始化时生成。Web 浏览器或 wget 可以连接到 URL：**http://routerIP/accounting/ip.cgi**

属性描述

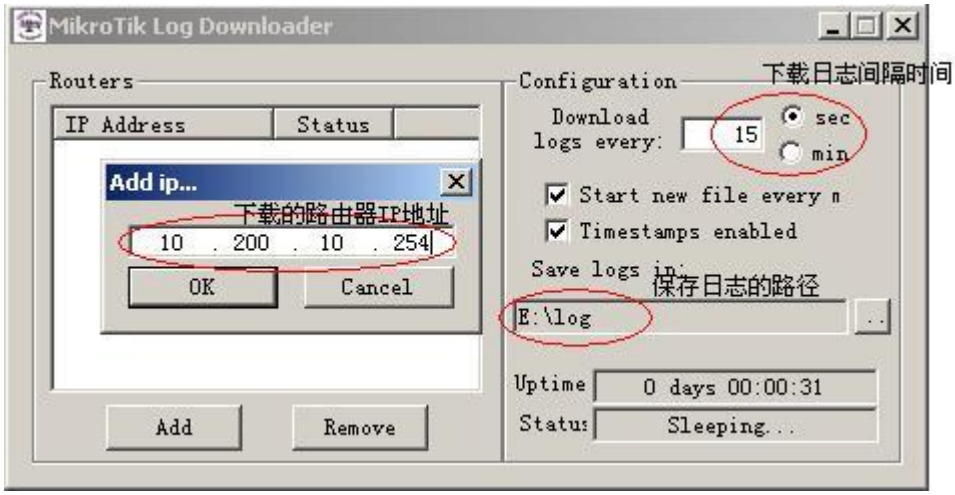
accessible-via-web (yes | no; 默认: **no**) – 是否 snapshot 通过 web 可用
address (*IP 地址*/子网掩码; 默认: **0.0.0.0**) -允许存取 snapshot 的 IP 地址范围

仅启用来自 **192.168.10.10** 主机对流量日志的 web 访问：

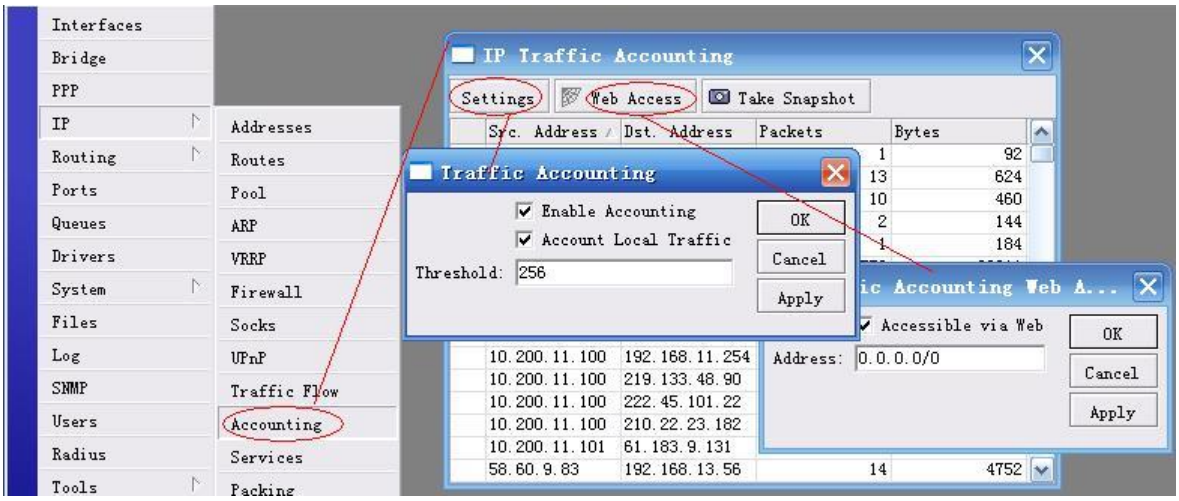
```
[admin@MikroTik] ip accounting web-access> set accessible-via-web=yes \
\... address=192.168.10.10/32
[admin@MikroTik] ip accounting web-access> print
      accessible-via-web: yes
                 address: 192.168.10.10/32
[admin@MikroTik] ip accounting web-access>
```

下面是通过 Log Downloader 和 winbox 操作的事例

首先打开 Log Downloader 程序，如图所示，添加需要记录 RouterOS 的日志的 IP 地址，并配置相应的参数：



然后在 RotuerOS 打开，并启用日志的远程记录，在 ip accounting 中设置：



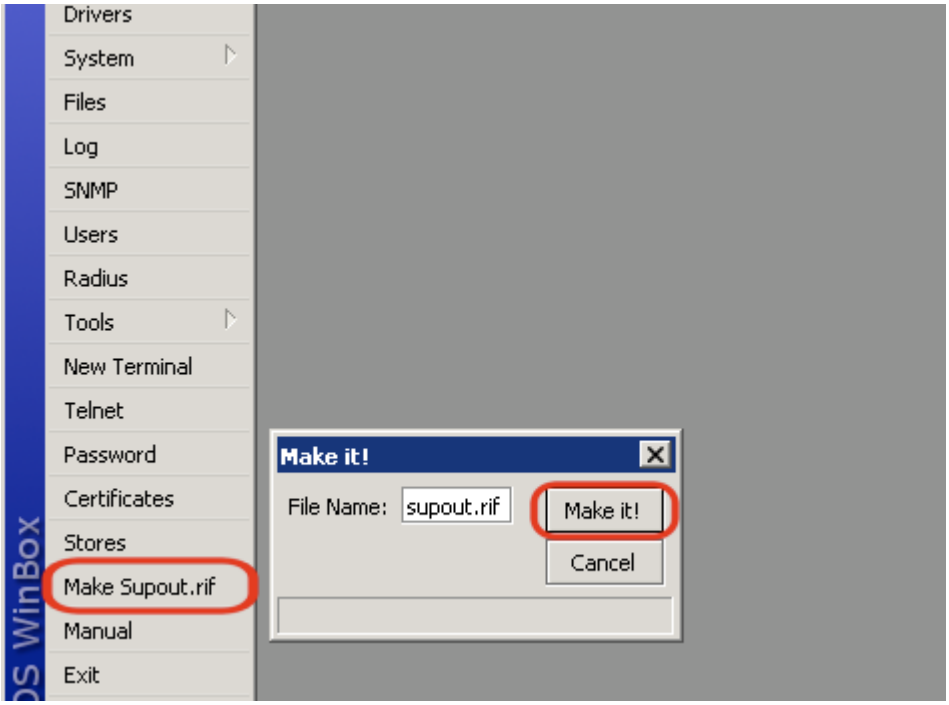
注：该软件只能通过 RouterOS 的 web 端口记录，即 web 端口默认必须是 80，最近在 bbs.routerclub.com 的论坛上有人发一个自己开发的日志记录软件解决了其他端口记录的问题。

2.15 Supout.rif 技术支持档

技术支持档被用于调试 RouterOS 和快速解决技术问题，通过 Make supout.rif 功能，所有的 MikroTik RouterOS 的信息都被存储在这个技术支持档中（默认 supout.rif），文件生成后都会存储在路由器的 files 目录下，可以通过 FTP 或者 winbox 下载。这个文件不会包含路由器密码，但也最好不要向他人透露文件信息，直接发送给 MikroTik 技术支持（support@mikrotik.com）

生成 Support 文件

生成技术支持档（support Output file）通过点击 **Make Supout.rif**,



执行命令后，不要阻止档生成过程，请等到 supout 窗口消失，之后在 files 目录下找到文件

Console 操作

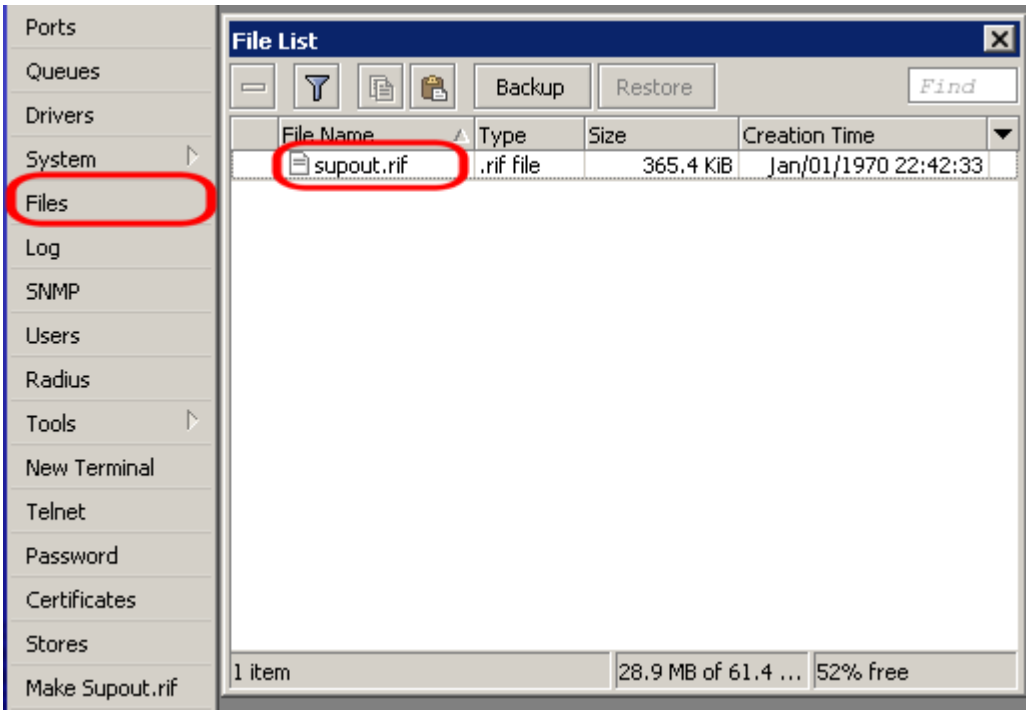
生成 suppout.rif 在 console 下运行命令

```
[admin@MikroTik] /system> sup-output
creating supout.rif file, might take a while
............................................./nova/lib/logmaker/2115.ps.lom: 1: ps: not found
done
[admin@MikroTik] /system>
```

命令执行后，请等到 console 提示 **done**.

下载技术支持档

技术支持档能通过 FTP 下载，注意检查防火墙没有阻止 FTP 连接，且 RouterOS 本地检查 ip service 中 FTP 服务没有被禁用

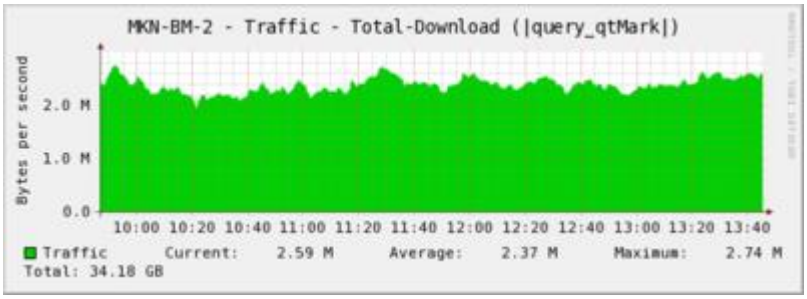


找到文件后，通过 e-mail 的附件，并说明问题发送给 MikroTik 技术支持(support@mikrotik.com)

2.16 SNMP

SNMP(Simple Network Management Protocol)简单网络管理协议是为管理 IP 网络设备的 internet 标准协议。SNMP 可用于流量图管理工具的各种数据，例如 Cacti、MRTG 或 The Dude，SNMP 写入支持需获取指定的 OID，SNMP 所支持的 SNMP OID 包括 v1、v2 或 v3。

如下图通过 cacti 记录网络流量，现在网上能找到 MikroTik 的 Cacti 范本，可直接调用。



注意：从 6.18 的 SNMP 开始实现 OID 黑名单功能 ，超时默认为 30 秒，当达到 600 秒被列入黑名单

关于 Cacti 搭建和通过 snmp 采集 RouterOS 系统数据可查看以下文章：

树莓派（RaspberryPi 3/4）基于 CentOS 搭建 Cacti1.2.16
Cacti1.2.16 安装向导设置
Cacti 1.2.16 添加 MikroTik RB 设备状态和流量图

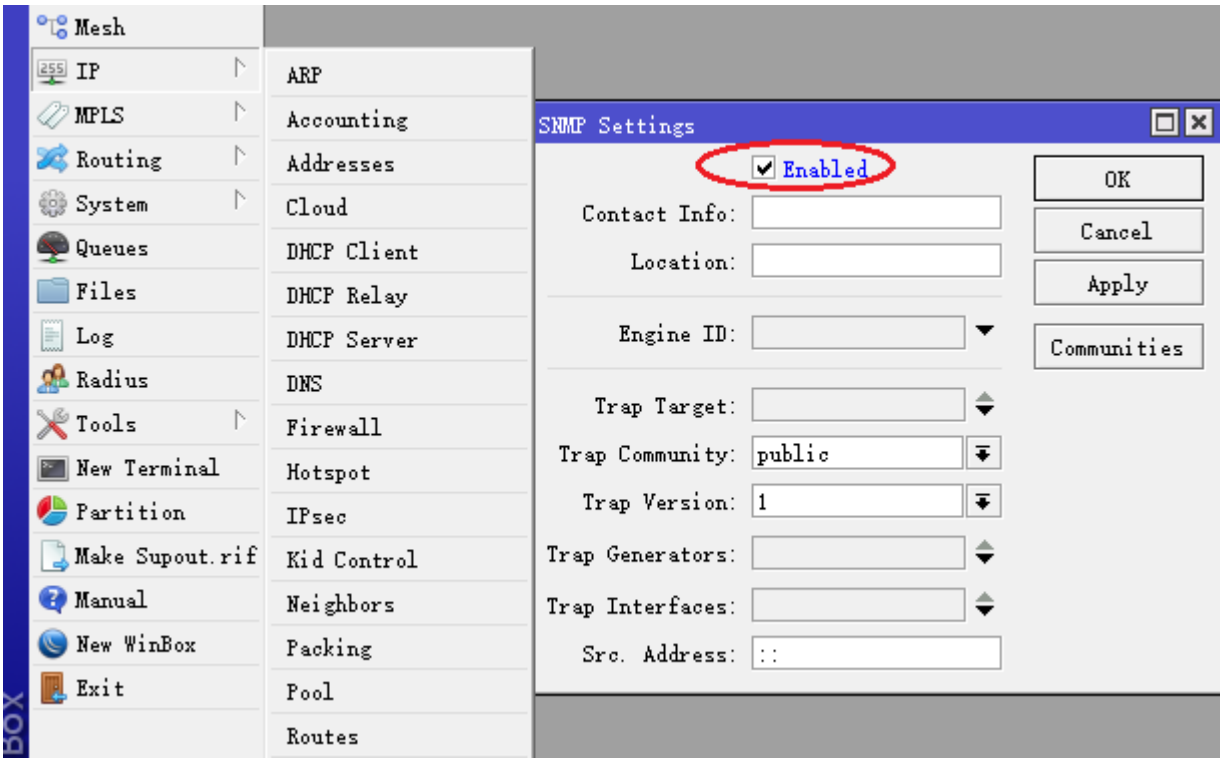
启用 SNMP

操作路径: /snmp

启用 RouterOS SNMP，进入 snmp 目录，通过 print 查看当前 snmp 配置，通过 set 命令启用

```
[admin@MikroTik] /snmp> print
        enabled: no
        contact:
        location:
        engine-id:
        trap-community: (unknown)
        trap-version: 1
[admin@MikroTik] /snmp>   set enabled yes
```

Winbox 操作如下：



在以上参数可以设定基本的 SNMP 管理信息，包括联系、团体名和版本号等，所有 SNMP 数据将根据团体名（community）获取，该设置需要进入 community 目录配置

Community 团体名

操作路径: /snmp community

该菜单下允许设置访问 SNMP 数据的团体名，虽然在 v1 和 v2c 有一定的安全设置，但仅明文的团体名字元和限制访问 IP 地址，从 SNMP v3 开始，一些安全选项被引入，通过 MD5/SHA1 授权（用户+密码），DES 加密（从 6.16，并加入 AES）

```
[admin@MikroTik] /snmp community> print value-list
                          name: public
```

```
                address: 0.0.0.0/0
               security: none
            read-access: yes
           write-access: no
 authentication-protocol: MD5
     encryption-protocol: DES
 authentication-password: *****
     encryption-password: *****
```

警告:默认设置仅配置了 public 的团体名，没有其他的安全设置，因此当启用 snmp 后，请配置需要的安全策略

参数说明：

属性	描述
address (*IP/IPv6 地址*; 默认: **0.0.0.0/0**)	允许连接的 SNMP 服务器 IP 地址
authentication-password (字符串; 默认: **""**)	验证 SNMP 服务器连接密码（仅用于 SNMPv3）
authentication-protocol (*MD5* | *SHA1*; 默认: **MD5**)	选择用于验证的协议(仅用于 SNMPv3)
encryption-password (*字符串*; 默认: **""**)	用于加密的密码(仅用于 SNMPv3)
encryption-protocol (*DES* | *AES*; 默认:**DES**)	用于团体名加密协议的加密类型(仅用于 SNMPv3)，AES 从 v6.16 开始支持
name (*字符串*; 默认:)	团体名
read-access (*yes* | *no*; 默认: **yes**)	是否为该团体规则开启只读访问
security (*authorized* | *none* | *private*; 默认:**none**)	
write-access (*yes* | *no*; Default: **no**)	是否为该团体规则开启写操作

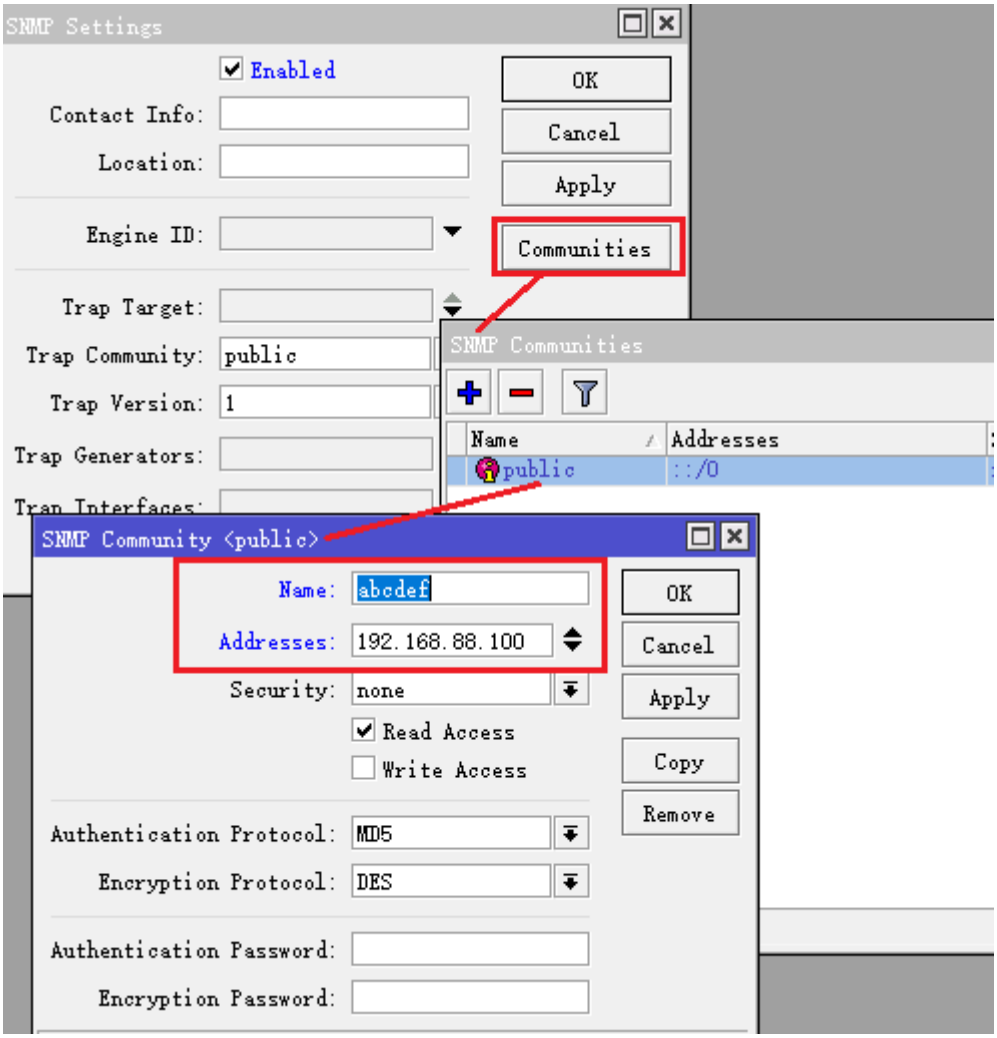
SNMP 配置实例

例如，有一台 Cacti 服务器 192.168.88.100，SNMP 团体名为 abcdef，采用 SNMPv2c 只读连接

首先我们需要配置团体名规则，首先进入 snmp communit，添加 Cacti 主机 IP，仅允许 192.168.88.100 访问 snmp，设置团体名 abcdef

```
[admin@MikroTik] /snmp community
[admin@MikroTik] /snmp community> add addresses=192.168.88.100/32 name=abcdef
[admin@MikroTik] /snmp community> print
Flags: * - default
 #     NAME      ADDRESSES                                SECURITY     READ-ACCESS
 0 * public     0.0.0.0/0                                   none          yes
 1     abcdef   192.168.88.100/32                           none          yes
[admin@MikroTik] /snmp community>
```

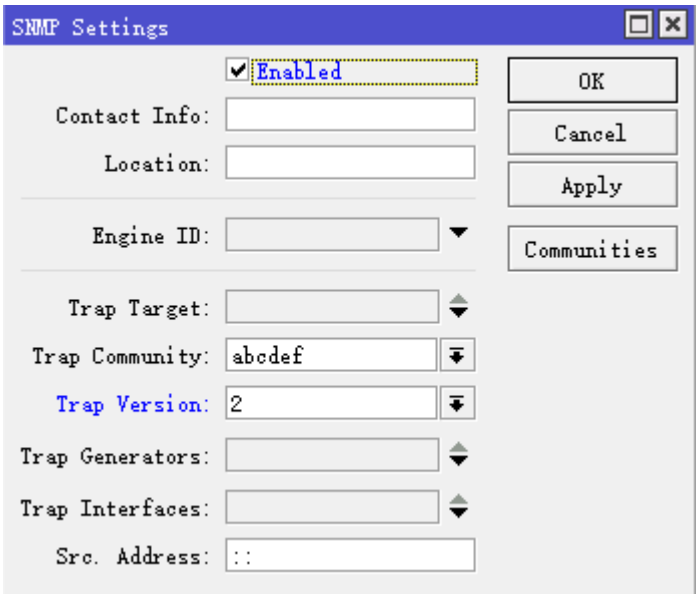
Winbox 配置



将 abcdef 团体名规则应用到 snmp 配置，并启用 snmp 服务

```
[admin@MikroTik] /snmp community>..
[admin@MikroTik] /snmp> set enabled=yes trap-community=abcdef trap-version=2
[admin@MikroTik] /snmp> print
               enabled: yes
               contact:
              location:
             engine-id:
          trap-target:
     trap-community: abcdef
        trap-version: 2
    trap-generators:
[admin@MikroTik] /snmp>
```

Winbox 配置



SNMP write

从 RouterOS v3 开始，SNMP 写入支持部分属性，SNMP 写入允许修改路由配置。考虑到安全，只有在 write-access 启用后，才能修改路由器或路由的 SNMP 相关参数。

SNMP 的 write-accesss 选项从 v3.14 可以设置，如下面对指定的 community 规则启用 write-access

```
/snmp community set <number> write-access=yes
```

修改路由器 ID

通过 SNMP 设置命令修改路由器的系统 ID，命令如下：

```
snmpset -c public -v 1 192.168.0.0 1.3.6.1.2.1.1.5.0   s New_Identity
```

- snmpset – SNMP SET 请求被用于修改一个网络信息
- public – 路由器的团体名；
- 192.168.0.0 – 路由器的 IP 地址；
- 1.3.6.1.2.1.1.5.0 – 对应 RouterOS 路由器 ID 的 SNMP 值；

SNMPset 命令等同于下面的 RouterOS 命令

```
/system identity set identity=New_Identity
```

重启

通过 SNMP set 命令重启路由器，设置重启 SNMP 参数值，设置重启赋值为 1

```
snmpset -c public -v 1 192.168.0.0 1.3.6.1.4.1.14988.1.1.7.1.0 s 1
```

- 1.3.6.1.4.1.14988.1.1.7.1.0, SNMP 重启路由器值;
- s 1, snmpset 命令赋值

snmpset 命令等同于以下 RouterOS 命令

```
/system reboot
```

2.17 File 目录

File 菜单下显示了路由器用户存储空间的所有文件内容。 能查看和编辑文件内容，并能删除文件。无法直接在 file 菜单中创建新文件，要创建文件，通过脚本方式完成。

操作路径: /file

当 RouterOS 的 ".npk"安装包上传到 file 目录下，将显示安装包信息，上传日期和时间

查看 File 下的内容：

```
[admin@dzeltenais_burkaans] /file> print
 # NAME                    TYPE          SIZE              CREATION-TIME
 0 autosupout.rif          .rif file     357368            oct/05/2010 09:47:01
 1 sample.txt              .txt file     230               oct/11/2010 12:14:43
[admin@dzeltenais_burkaans] /file> set 1 contents=Hello

[admin@dzeltenais_burkaans] /file> print detail where name~"sample"
 1 name="sample.txt" type=".txt file" size=5 creation-time=oct/11/2010 12:15:38
contents=Hello
[admin@dzeltenais_burkaans] /file>
```

查看安装包信息：

```
[admin@493G] /file> print detail
 0 name="multicast-5.0rc2-mipsbe.npk" type="package" size=245643
creation-time=jan/05/1970 21:44:25      package-name="multicast"
package-version="5.0rc2" package-build-time=oct/11/2010 06:34:02
    package-architecture="mips"
```

警告: 设备的文件列表中有一个名为"flash"的目录，保存的文件必须存储在其中，因为它之外的所有内容都保存在 RAM 中，在重新启动时将丢失。但不包括 .npk 安装包文件，npk 包将在系统丢弃 RAM 内容之前完成升级或降级过程。

注意: 对于具有 NAND 闪存的多核设备(如 CCR 系列路由器，RB4011iGS)， RouterOS 使用回写，它将文件先缓存到 RAM 内存中，而不是直接写入到闪存。写入可能被延迟 40 秒，这有助于减少 CPU 消耗，从而获得更好的网络处理性能。然而，当设备突然断电时，这可能会导致出现空文件或零长度的文件，因为文件没有完全保存在闪存上。

2.18 Neighbor Discovery（邻居探测）

Neighbor Discovery protocols 邻居发现协议，可以在二层广播域中找到兼容 MikroTik 邻居发现协议(MNDP)、思科发现协议(CDP)或链路层发现协议(LLDP)的设备，同时也会将本设备信息发送的二层网络中。

操作路径: /ip neighbor

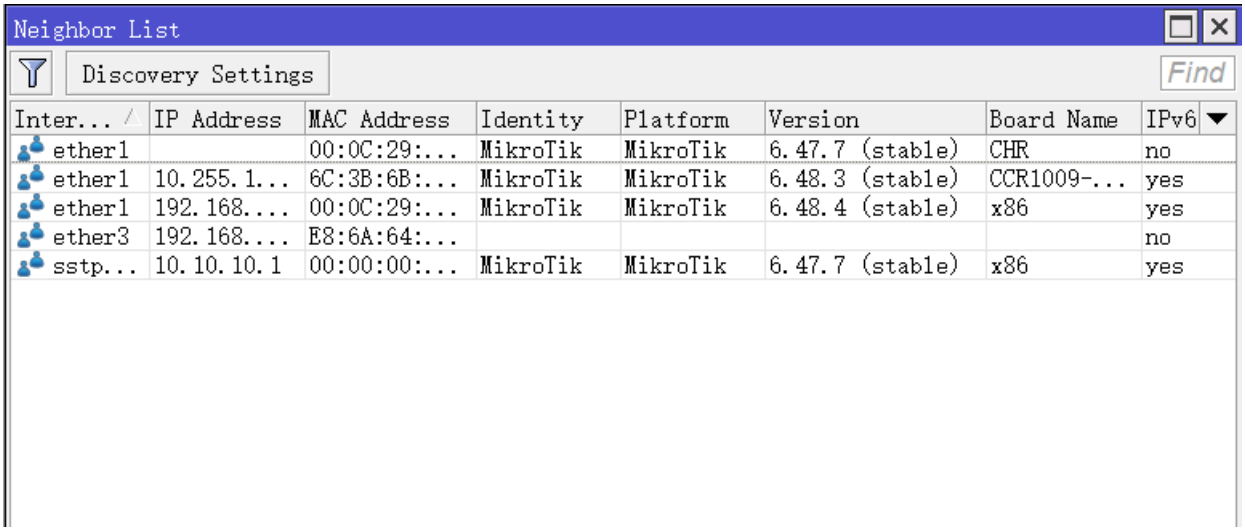
在 ip neighbor 表中，会显示已发现在二层广播域的所有邻居，这些信息包括连接到哪个网络接口，它的 IP/MAC 地址和其他相关参数，该列表是只读，显示如下：

```
[admin@MikroTik] /ip neighbor print
 # INTERFACE ADDRESS       MAC-ADDRESS        IDENTITY   VERSION    BOARD
 0 ether13   192.168.33.2    00:0C:42:00:38:9F  MikroTik   5.99       RB1100AHx2
 1 ether11   1.1.1.4         00:0C:42:40:94:25  test-host  5.8        RB1000
 2 Local     10.0.11.203     00:02:B9:3E:AD:E0  c2611-r1   Cisco I...
 3 Local     10.0.11.47      00:0C:42:84:25:BA  11.47-750  5.7         RB750
 4 Local     10.0.11.254     00:0C:42:70:04:83   tsys-sw1  5.8        RB750G
 5 Local     10.0.11.202     00:17:5A:90:66:08   c7200     Cisco I...
```

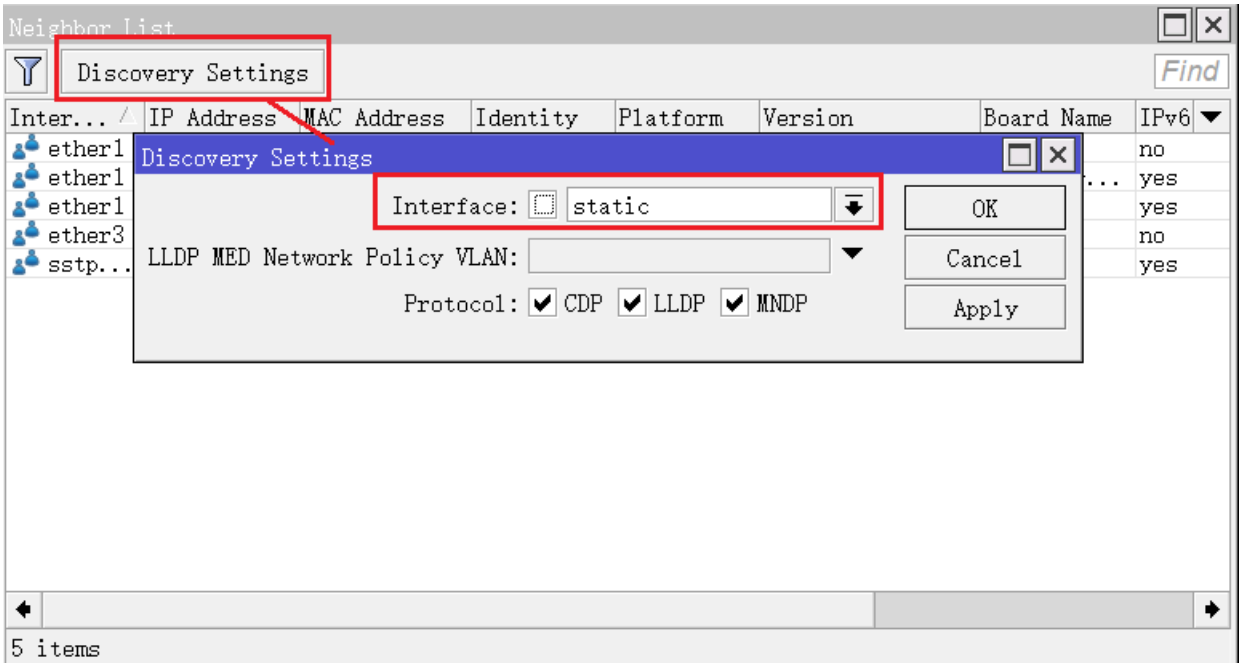
从 RouterOS v6.45 开始，每个接口的邻居发现数量被限制在“总内存 容量(mb)*16”，以避免内存耗尽。

Discovery 配置

使用 interface list 来控制指定的接口是否参与邻居发现。如果该接口在发现接口列表中，则会向外广播系统的基本信息，并处理接收到的二层网络广播的发现报文。将某个接口从接口列表中删除后，将禁止在该接口上邻居探测，也禁止在该接口向外广播。



在默认情况下，RouterOS 会允许所有静态接口开启邻居发现协议



注意：邻居发现协议，当选择的接口启用后，也会向外广播本设备信息，为安全起见路由器的对外接口（WAN）需关闭邻居发现协议，避免被外部网络发现。在一些大型企业或校园的内部网络中，也尽可能的配置自己的邻居发现接口策略。部分 RB 设备(hAP ac 系列、RB3011、RB4011 和 hEX 等)默认配置，指定了 ether1 作为 WAN 口，因此 WAN 接口被排查在邻居发现协议外。

操作路径：/ip neighbor discovery-settings

属性	描述
discover-interface-list (*string*; 默认: **static**)	指定 interface list 成员，允许运行邻居发现协议
lldp-med-net-policy-vlan (*integer 0..4094*; 默认: **disabled**)	为 LLDP-MED 网络策略 TLV 发布的 VLAN ID。允许为支持 LLDP-MED 的设备(如 VoIP 电话)分配 VLAN ID。该 TLV 只会添加到发现 LLDP-MED 设备的接口上。其他 TLV 值是预定义的，不能修改： Application Type - Voice VLAN Type - Tagged L2 Priority - 0 DSCP Priority – 0
protocol (*cdp* | *lldp* | *mndp*; 默认: **cdp,lldp,mndp**)	已使用的发现协议列表

从 RouterOS v6.44 开始，邻居发现协议工作在单个接口上。当发现接口列表中包含一个主接口(如 bonding 接口或 Bridge 接口)时，它的所有从属接口都会被自动添加到邻居发现协议。但可能要求从属接口加入到邻居发现协议中，可以单独指定从属接口的配置。

下面的实例是一个 bond1 接口，包含了两个从属接口 ether5 和 ether6，我们可以指定仅 ether5 参与到邻居发现协议中

```
/interface bonding
add name=bond1 slaves=ether5,ether6
```

在 interface list 创建一个 only-ether5 列表

```
/interface list
add name=only-ether5
```

添加成员是 ether5

```
/interface list member
add interface=ether5 list=only-ether5
```

设置邻居发现列表是 only-ether5

```
/ip neighbor discovery-settings
set discover-interface-list=only-ether5
```

现在，邻居列表显示了接收到邻居发现协议的主接口和实际从属接口。

```
[admin@R2] > ip neighbor print
 # INTERFACE ADDRESS       MAC-ADDRESS       IDENTITY    VERSION     BOARD
 0 ether5     192.168.88.1    CC:2D:E0:11:22:33    R1          6.45.4    CCR1036-8G-2S+
   bond1
```

第三章 MikroTik RB/CCR/CRS 介绍

RouterBOARD、CCR（Cloud core Router）和 CRS（Cloud Router Switch）是 MikroTik 硬件产品，基于 MIPS、PPC、Tilera 和 ARM 等平台开发的硬件，采用的是基于 Linux 内核的 RouterOS 系统，即将 RouterOS 变成了硬件。就如 Cisco 的路由器使用的是 IOS，基于 Unix 开发；Juniper 基于 FreeBSD 开发的 JunOS 系统。MikroTik 的路由器则使用的是 RouterOS，都是由硬件和操作系统软件组成，可以理解为他们之间仅仅是硬件和软件的不同，所以 RouterBOARD 是硬件路由器，而这样的路由器针对的是低功耗、高稳定性的无线网络和中小型有线网络。后期 MikroTik 又开发了两款新品一个是基于 Tilera 构架的 Core Cloud Router（CCR 系列）和仍然基于 MIPS 构架的 Cloud Router Switch（CRS 系列），CCR 基于高端路由，CRS 基于路由交换，在这里仍然把两款产品归到 RouterBOARD 中介绍。

如果我们换一种思路，我们把 PC 也当作一种路由器硬件平台，把 RouterOS 安装到 PC 上那是硬件路由器，只是 PC 对我们的日常生活来说太普遍了，一时我们不能理解和接受，就叫 RouterOS 为软路由，其实市面上很多路由器使用的是嵌入式平台，如 ARM、MIPS 或者 PowerPC 等，他们只是换了一个硬件平台，运行的软件其实也是 Linux 嵌入系统或者 FreeBSD 等等这些软件和 RouterOS 本质上没有什么区别，要是他们的软件也开放出来，基于 PC 平台，大家都可以叫他软路由。

RouterBOARD 为低功耗的路由器，一般无扩展卡时（无线网卡、USB 扩展和其他设备连接）功耗大多在 4-5w，PowerPC 处理器略高 5-12w 左右，具备 MiniPCI 或者 MiniPCI-e 扩展槽的设备，在扩展无线网卡后，根据扩张的数量和功率大小不同，功耗也有所变化。

我们可以把 RouterBOARD 分了 3 类：

1、 无线型 RouterBOARD：侧重于无线网络的 RouterBOARD 设备，如 RB411、RB711、RB911、STX、Groov 等
2、 有线型 RouterBOARD：侧重于有线网络的 RouterBOARD 设备，如 RB450、RB750、RB850、RB1100、CCR 系列和 CRS 系列等
3、 混合型 RouterBOARD：以上两者兼有，如 RB433、RB493、RB2011、RB751、RB951 和 hAP 等

注：在之后的内容里，我将 RouterBOARD 缩写为 RB，

提示：CRS3xx 系列交换机介绍单独整理了一份教程，可以下载 **MikroTik CRS3xx 系列操作配置手册**

3.1 RouterBOARD 的发展

早期的 RB 是 RB230，这款是 x86 的平台在 2002 年推出，虽然是低功耗平台，但基于 x86 平台，一些硬件集成方面欠缺，非 SoC 平台，真正的 SoC 平台是从 2006 年的 RB112 开始，包括 RB100、RB300、RB400、RB500、RB600、RB700、RB800、RB1000、RB1100、RB1200，但许多型号已经被停产和淘汰。

2006 年

RB112、RB150、RB153、RB532、RB502 推出，接着推出了 RB133、RB133c、RB532rc5 和 RB192 等这些产品是 RB 第一代产品、RB100 和 RB500 系列为后期的 RB 产品奠定了基础，这些产品都是基于 MIPS 4kc 的处理器

2007 年

分别推出了 RB333 和 RB600 两款基于 PowerPC 平台，性能有大幅提升，等成本相对较高，只能算过渡产品

2008-2009 年

推出了 RB400 系列、包括 RB411、RB433、RB450、RB493 等系列产品，到现在仍然是 RB 产品线的主流产品，RB1000 也在 08 年推出

2010 年

RouterOS4.0 支持 11n 协议后，RB 进入 11n 时代，并推出了 RB700 系列，RB711 具备 11n 的 5G 传输，针对低端市场的有线产品推出了 RB750 系列

2011 年

RB711 演变的 RBSXT 的 5G11n 室外设备、400 系列高性能版本 RB435G、2.4G 大功率 11n 的 RB711-2Hn、具备 USB 和 PoE 的 RB750UP、集成 2.4G 11n 的 RB751 和扩展 USB 的 RB751U 和 RB751G，以及 RB1100、RB1100AH 和双核的 RB1100AH×2，还增加了一款万兆网卡的 RB1200，和支持 SFP 光模块的 RB 设备，各种整合的无线设备如 SXT、Groove 等。

2012 年

在 2012 年是 RouterBOARD 产品更新最大的一年，CCR 系列路由出现，让 Mikrotik 产品向高端更进一步，RouterOS6.0 的逐步完善，在性能方面不断优化，如 Queue 的不断改进，加入了 Fastpath 功能，进一步提高转发量。

2013 年至今

在 CCR 系列基础上继续发展了多种型号，并且开发了 CRS 系列的路由交换设备，从 CRS3xx 系列开始全面支持 Hw-offloading 二层硬件下发交换芯片处理（Bridge 和 Switch 配置），早期 RB700 系列开始逐步淘汰由 RB900 系列接替，单板的 OEM 产品开始减少，整机一体化的产品开始增加，并在 2014 年下半年推出 802.11ac 产品，2017 年推出了 802.11ad 产品。

RouterBOARD 早期产一览表：

型号	基本信息	以太网口	MiniPCI	集成 WLAN
RB100 系列				
RB112	MIPS 4kc 175Mhz，16MB RAM	1×100M	2	无
RB133c	MIPS 4kc 175Mhz，16MB RAM	1×100M	1	无
RB133	MIPS 4kc 175Mhz，32MB RAM	3×100M	3	无
RB150	MIPS 4kc 175Mhz，32MB RAM	5×100M	无	无
RB153	MIPS 4kc 175Mhz，32MB RAM	5×100M	3	无
RB192	MIPS 4kc 175Mhz，32MB RAM	9×100M	2	无
RB500 系列				
RB502	MIPS 4kc 266Mhz，32MB RAM	1×100M	1	无
RB532	MIPS 4kc 266Mhz，32MB RAM	3×100M	2	无
RB532rc5	MIPS 4kc 399Mhz，64MB RAM	3×100M	2	无
RB300 系列				

RB333	PowerPC 333MHz, 64MB DDR RAM	3×100M	3	无
RBCRD 验证型				
RB/CRD	MIPS 4kc 184Mhz，32MB RAM	3×100M	无	802.11bg
RB400 系列				
RB411	Atheros 300Mhz , 32MB RAM (CPE)	1×100M	1	无
RB411R	Atheros 300Mhz , 32MB RAM (CPE)	1×100M	无	802.11bg
RB411A	Atheros 300Mhz , 64MB RAM	1×100M	1	无
RB411AR	Atheros 300Mhz , 64MB RAM	1×100M	1	802.11bg
RB411U	Atheros 300Mhz , 64MB RAM	1×100M	1+1pci-e	无
RB411AH	Atheros 680MHz （超频 800MHz）	1×100M	1	无
RB411UAHR	Atheros 680MHz （超频 800MHz）, 64MB RAM,1 USB	1×100M	1+1pci-e	802.11bg
RB433	Atheros 300Mhz , 64MB RAM	3×100M	3	无
RB433AH	Atheros 680MHz （超频 800MHz）, 128MB RAM	3×100M	3	无
RB433UAH	Atheros 680MHz , 128MB RAM,2 USB	3×100M	3	无
RB435G	Atheros 680MHz , 128MB RAM,2 USB	3×1G	5	无
RB493AH	Atheros 680Mhz , 64MB RAM	9×100M	3	无
RB493G	Atheros 680Mhz , 256MB RAM.1USB	9×1G	3	无
RB450	Atheros 300Mhz , 32MB RAM	5×100M	无	无
RB450G	Atheros 680Mhz（超频 800MHz）, 256MB RAM	5×1G	无	无
RB600 系列				
RB600	PowerPC 400MHz（超频 533MHz）, 64MB DDR RAM	3×1G	4	无
RB600A	PowerPC 400MHz（超频 533MHz）, 128MB DDR RAM	3×1G	4	无
RB700 系列				
RB711	Atheros 400MHz , 32MB RAM(CPE)	1×100M	无	802.11an
RB711A	Atheros 400MHz , 64MB RAM	1×100M	无	802.11an
RB711-2Hn	Atheros 400MHz , 32MB RAM(CPE), 1 USB	1×100M	无	802.11bgn
RB750	Atheros 300Mhz CPU, 32MB RAM	5×100M	无	无
RB750G	Atheros 680Mhz CPU, 32MB RAM	5×1G	无	无
RB750UP	Atheros 300Mhz CPU, 32MB RAM, 1 USB ,	5×100M	无	无
RB751	Atheros 300Mhz CPU, 32MB RAM,	5×100M	无	802.11bgn
RB751U	Atheros 300Mhz CPU, 32MB RAM, 1 USB	5×100M	无	802.11bgn
RB751G	Atheros 680Mhz CPU, 32MB RAM, 1 USB	5×1G	无	802.11bgn
RBSXT	Atheros 400MHz , 32MB RAM(CPE), 1 USB	1×100M	无	802.11an
RB800 系列				
RB800	PowerPC 800MHz ， 256M DDR RAM,1 CF	3×1G	4+1pci-e	无
RB900 系列				
RB911 Lit2/5	AR9344 600MHz，64MB RAM	1×100M	无	802.11bgn/an
RB912UAG	AR9342 600MHz，64MB RAM	1×1G	1×pci-e	802.11bgn/an
RB951-2n	Atheros 300MHz，32MB RAM	5×100M	无	802.11bgn
RB951G-2HnD	AR9344 600MHz，128MB RAM，1 USB	5×1G	无	802.11bgn
RB941-2nD	hAP lite QCA9531-BL3A-R 650 MHz，16MB RAM	4×100M	无	802.11bgn
RB952Ui-5ac2nD	hAP AC lite QCA9531 650MHz，64MB RAM	5×100M	无	802.11bgn/ac
RB1000 系列				
RB1000	PowerPC 1.3GHz ， 512M DDR RAM	4×1G	无	无
RB1100	PowerPC 800MHz ， 512M DDR RAM	13×1G	无	无
RB1100AH	PowerPC 1066MHz ， 2G DDR RAM	13×1G	无	无

RB1100AH×2	PowerPC 双核 ，2G DDR RAM	13×1G	无	无
RB1200	PowerPC 1066MHz ，2G DDR RAM	10×1G	无	无
RB2011 系列				
RB2011LS-IN	AR9344 600MHz，64MB RAM	5×1G，5×100M	无	802.11bgn
RB2011UAS-IN	AR9344 600MHz，64MB RAM，SFP×1	5×1G，5×100M	无	802.11bgn
RB2011UAS-2HnD-IN	AR9344 600MHz，128MB RAM，SFP×1，1 USB	5×1G，5×100M	无	802.11bgn
RB3011UiAS-RM	ARM IPQ-8064 2×1.4GHz, 1GB RAM	10 × 1G,1 × SFP	无	无
CCR 系列				
CCR1009	Tile GX 9×1.2GHz, 1GB	8×1G	无	无
CCR1016	Tile GX 16×1.2GHz, 2GB	12×1G	无	无
CCR1036	Tile GX 36×1.2GHz, 4GB/8GB,	12×1G	无	无
CCR1072	Tile GX 72×1GHz, 16GB,	8×10G	无	无
CRS 系列				
CRS125	AR9344 600MHz，128MB RAM，SFP×1，1 USB	24×1G	无	802.11bgn
CRS226	400MHz，64MB RAM，SFP+×1，1 USB	24×1G	无	无

注：由于 MikroTik 产品一直在更新，表中的很多产品已经停产，因此该表仅供参考，具体型号和参数请连接 www.routerboard.com

3.2 RouterBOARD 产品命名与标识

RouterBOARD 产品型号较多，各种字母和数字标识繁琐，下面介绍下产品命名的特点

RouterBOARD 基本编号适合大多型号，除 RB600，RB800 和 RB1000 系列型号外，他们的区别如下：

- RB1XX，即 RB100 系列
- RB133，即是 100 系列，有 3 个以太网口，3 个 MiniPCI 无线扩展接口
- RB493，即 400 系列，有 9 个以太网口，3 个 MiniPCI 扩展

RouterBOARD 基本命名规则

<主板名称> <主板特征> - <集成无线网卡> <无线网卡特征> - <连接接口类型> - <外壳类型>

后缀代表含义：

- AH，A 代表高内存，H 代表高性能（高 CPU）
- e，代表有 PCI-e 接口扩张
- G，代表高性能和千兆网口
- U，代表 USB 扩展
- R，代表集成无线模块
- P，PoE
- L，Low 低成本产品
- S，SFP 光扩展接口

- x（N），代表 CPU 内核数量，如 x2，x16，x32

集成无线网卡命名

如果集成无线网卡设备，命名格式将是如下：

<频段><发射功率><协议><信道数量>

频段

- 5 – 支持 5Ghz
- 2 – 支持 2.4Ghz
- 52 – 支持 5Ghz 和 2.4Ghz

发射功率

- 无命名 - "普通发射功率" - <23dBm at 6Mbps 802.11a; <24dBm at 6Mbps 802.11g
- H - "高发射功率" - 23-24dBm at 6Mbps 802.11a; 24-27dBm at 6Mbps 802.11g
- HP - "较高发射功率" - 25-26dBm 6Mbps 802.11a; 28-29dBm at 6Mbps 802.11g
- SHP - "超高发射功率" - 27+dBm at 6Mbps 802.11a; 30+dBm at 6Mbps 802.11g

无线协议

- 无命名 – 仅支持 802.11a/b/g
- n – 支持 802.11n
- ac – 支持 802.11ac

无线通道数量

- 无命名 – 单无线通道
- D – 双无线通道
- T – 三无线通道

无线网卡接口类型命名

- 无命名 – 扩展槽根据类型或型号决定
- MMCX – MMCX 接口
- u.FL - u.FL 接口

外壳类型命名

- 无命名 – 一个产品的主型号默认使用外壳
- BU - board unit (无外壳) – 特殊需求情况下，为仅主板需求的客户提供
- RM – 机柜外壳型
- IN – 室内外壳型
- OUT – 室外防水型
- SA – 集成扇形天线型
- HG – 高增益天线外壳
- EM – 内存扩展型

例如：RB912UAG-5HPnD

- RB (RouterBOARD)
- 912 – 9 系列主板，1 个以太网接口，两个无线网卡接口（集成和 MiniPCIe 接口）
- UAG – USB 接口，大内存和千兆以太网接口
- 5HPnD – 内置 5GHz 较高发射功率网卡，支持双通道的 802.11n 协议

Cloud Core Router 命名

Cloud Core Router (缩写 CCR)

<4 组产品数字编号>-<端口数量>-<外壳类型>

4 组数字编号

- 第一个数字代表产品系列
- 第二个数字保留所用
- 第三和第四代表设备上的 CPU 核心数量

端口数量

- -<n>G – 千兆以太网端口数量
- -<n>S – SFP 端口数量
- -<n>S+ - SFP+端口数量

Cloud Router Switch 命名

Cloud Router Switch (缩写 CRS)命令

<3 组产品数字编号>-<端口数量>-<集成无线网卡>-<外壳类型>

- 3 组数字编号

- 第一个数字代表产品系列
- 第二和第三代表所有有线接口数量(Ethernet, SFP, SFP+)

- 端口数量

- -<n>G – 千兆以太网端口数量
- -<n>S – SFP 端口数量
- -<n>S+ - SFP+端口数量

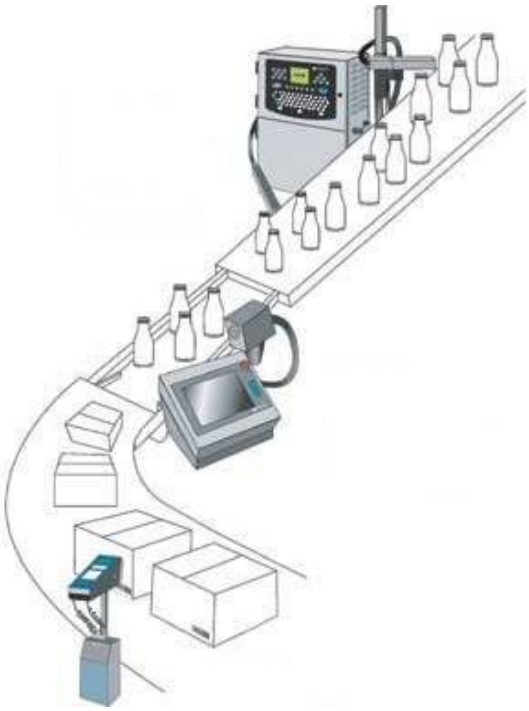
更多具体的 RouterBOARD 信息请访问 www.routerboard.com 的网站

3.3 RouterBOARD Throughput（吞吐量）

路由器包转发性能 Throughput，一般是指网络数据从一个以太网口到另一以太网口的转发性能，这代表了交换机或路由器的性能。MikroTik 官方从 1518Byte、512Byte 和 64Byte 给出了详细的吞吐量测试数据，在这方面

MikroTik 路由器和交换机数据是非常真实的，比很其他一些厂商数据含糊不清良心许多！这些数据包括桥接和路由的性能转发测试，但不含 nat，因为 nat 转发和会话数量有很大关系。

这章节，带大家如何看一台路由器性能，通常数据包处理都要经过 CPU（路由器或三层交换机支持由交换芯片处理，这里不做讨论），如当前 RB、CCR 和 CRS 等设备采用的是 MIPS、PowerPC、Tile 和 ARM 等构架的 CPU，基于 x86 平台的就是大家熟知的 Intel 和 AMD。CPU 的处理能力直接决定数据包转发性能，处理数据包就像流水线一样，对每个包进行拆包、重新封装和转发。



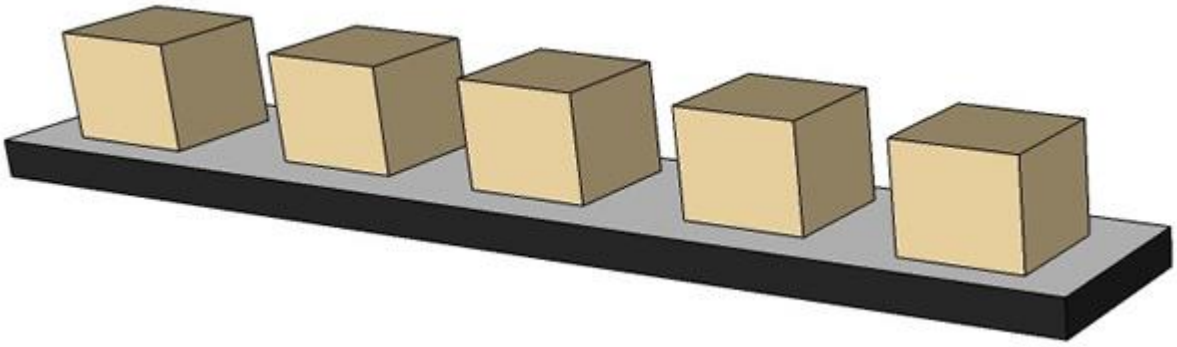
数据包的大小决定了处理速度和吞吐量，例如我们在上网时，发送聊天信息、网页浏览、播放视频和下载数据，封装的数据包大小是不一样的。假如一个数据包封装了网页图片是 512Byte，CPU 每秒能处理 10000 个，并不能做到比 512Byte 小一半的 256 Byte 包每秒就能处理 20000 个，对于 CPU 处理数据包有点类似计件式的工作，当然实际情况不是这样，只是做一个类比。下图是一台 CCR2004-16G-2S+的吞吐量数据

CCR2004-16G-2S+		AL32400 All port test (7.1beta6)					
Mode	Configuration	1518 byte		512 byte		64 byte	
		kpps	Mbps	kpps	Mbps	kpps	Mbps
Bridging	none (fast path)	2591.6	31472	5648.4	23136	9697.9	5275.7
Bridging	25 bridge filter rules	2249.5	27318.5	2451.7	10042.1	2424.2	1318.8
Routing	none (fast path)	2532.4	30753.8	5239.6	21461.2	8664.2	4713.3
Routing	25 simple queues	1174.6	14264.6	1217.6	4987.2	1316.8	716.3
Routing	25 ip filter rules	866	10517.1	860.7	3525.5	842.1	458.1

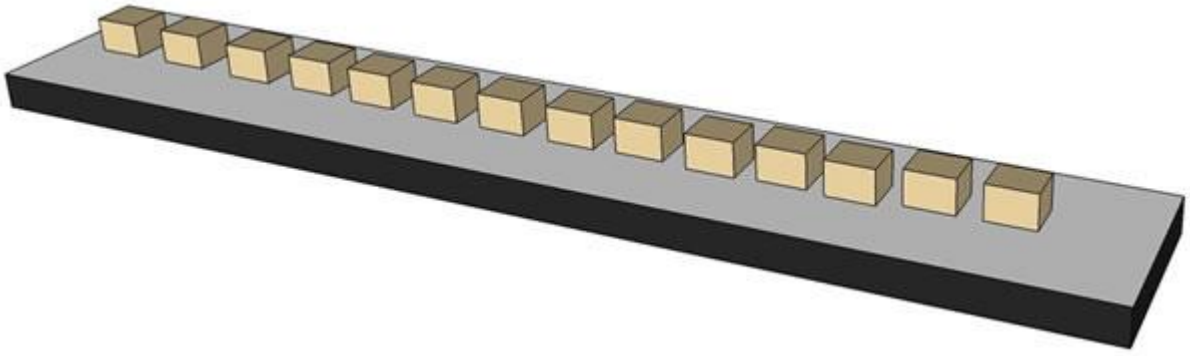
一共分了 3 组数据 1518Byte、512Byte 和 64Byte，每组数据又分为 Mbps 和 Kpps，Mbps 大家比较好理解是每秒多少 Mbit，Kpps（packet per second）则是，每秒处理多少 K 的数据包。测试数据包含了 bridge 桥接转发和路由转发两种，bridge 转发下增加了配置 25 条桥接过滤规则转发，routing 路由转发也增加了 25 条流控规则和 25 条防火墙过滤规则的转发性能，这样的吞吐量数据更接近用户实际使用的环境。

我们以第一项 Bridging none（fast path）为例：每个数据包封装 1518Byte，CCR2004 能处理 31.47Gbps，每秒能转发 2591.6k 的数据包，如果将数据封装大小调整为 64Byte，每秒能处理 5.27Gbps，每秒转发 9697.9k 的数据包，数据包封装越小，转发的数量就越多，对 CPU 性能要求越高。

举一个例子，假设一个在流水线检验员，检查每个产品的包装是否合格，第一天，产品包装是大箱子，里面有 8 个小包装，但他只需检查大包装是否合格，无需检查里面的小包装，他一分钟能检查 20 个大箱包装，出货数量也多。



第二天，流水线不再用大包装了，都是小包装，那请问他的每分钟的检查数量是否会提高 8 倍？估计提升 2 倍都很不错了。



这和我们的路由器处理大包和小包的道理是一样，CPU 处理大数据包很轻松，而处理小数据包就很吃力，大包装的数据多，一次就可以转发很多数据，而小包数据少，数量又多，很考验 CPU 的性能。我们衡量路由器 CPU 性能一个重要指标是看 64byte 的小包转发能力，三种封装有不通的衡量：

- 64byte 包吞吐量反应了 CPU 的性能
- 512byte 包反应了 CPU 和内存整体综合性能
- 1518byte 包吞吐量反应了内存性能

吞吐量的标准单位是 pps（packet per second），在 MikroTik 给出的 Mbps 和 Kpps 数据是可以相互转换的，例如 1518Byte 的 31472Mbps 约等于 2591.6kpps*1000*1518*8

MikroTik 所有吞吐量测试是通过以下方式完成：

- 通过路由器转发测试路由转发
- 精简 RouterBOARD 设置，仅安装 system 功能包
- 通过 Agilent N2X 设备测试

图 1 是早期 RouterBOARD 在 64byte 数据报，使用 Bridge 桥接模式下的对比图

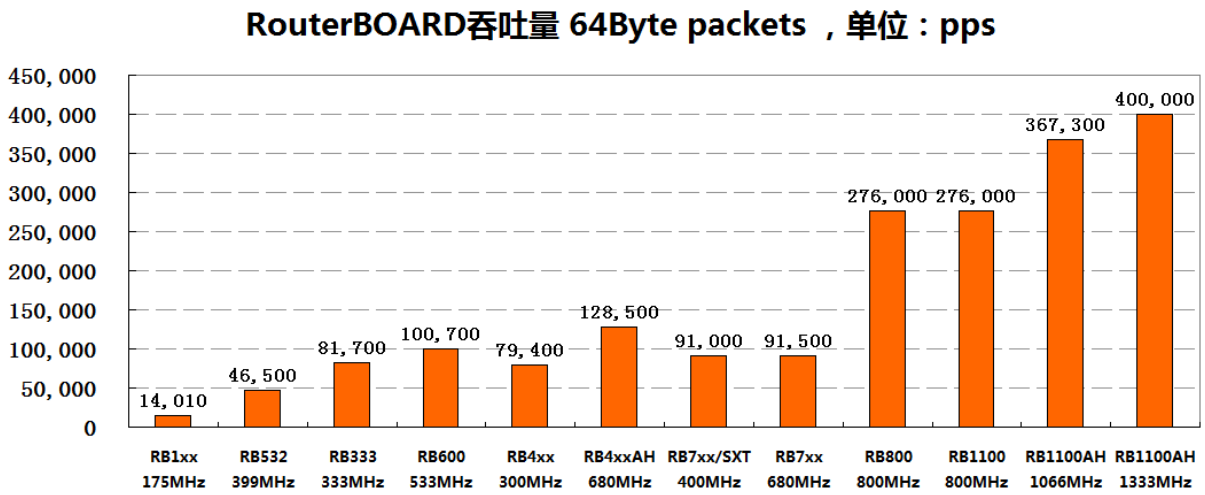


图 1

下面图 2 则是近年来，RB、CRS 和 CCR 等产品的对比，从图 1 的 RB4xxAH（680MHz）做对比几代产品的差别，注意单位是 kpps，每秒转发包

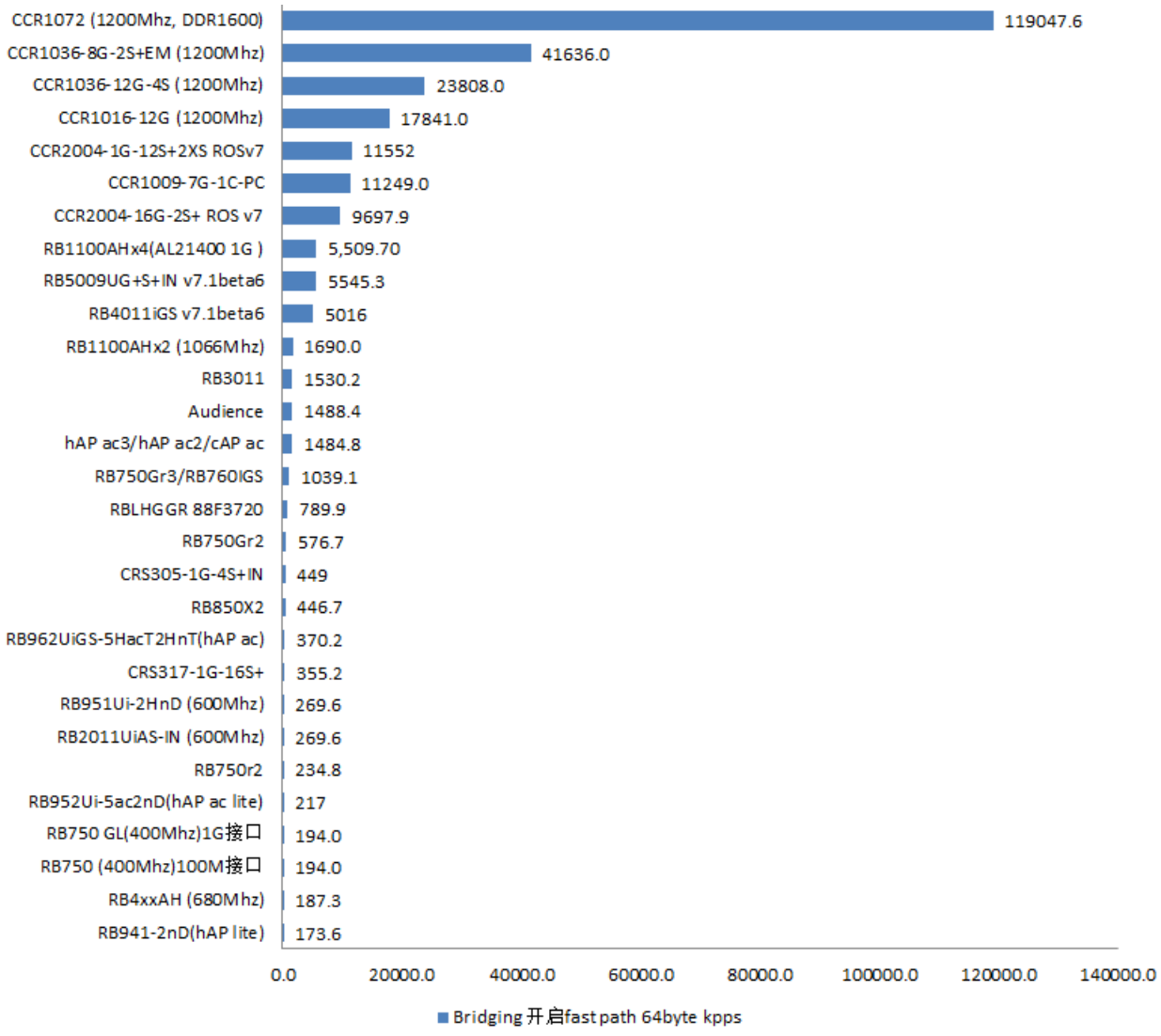


图 2

图 2 的数据可以看到 CCR 系列的吞吐量性能和 RB 系列有明显的差距。

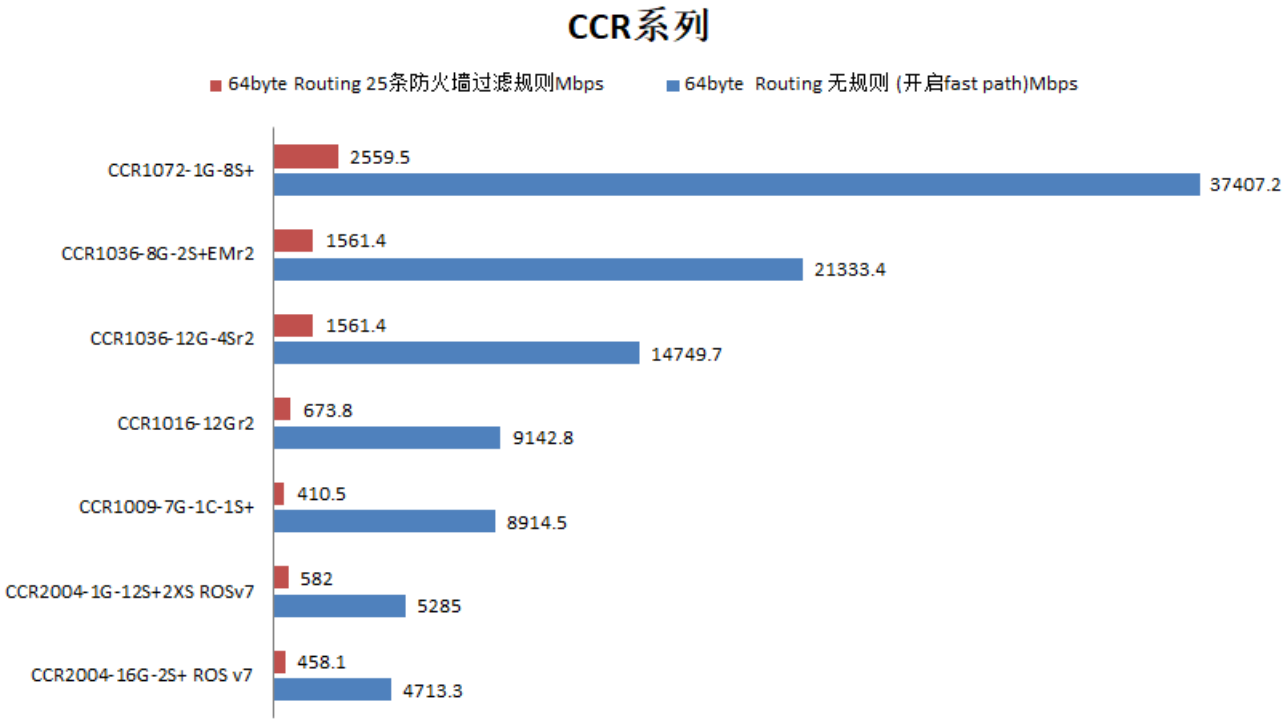
以上基于 64byte，且是纯桥接转发性能，没有做任何策略的情况下。接下来看看路由转发性能，同时不能只从一组参数入手，把添加策略后的情况纳入看又是如何。

在官方提供的数据里，有 birdging 和 routing 测试，即桥接和路由测试，如下面的表格。

RouterOSv6 64byte 吞吐量（kpps）	RB850X2	RB750Gr2	RB750Gr3
CPU	PPC P1023NSN5CFB 533MHz	QCA9556 720MHz	MT7621A 880MHz
Bridging 开启 fast path	446.7	576.7	1039.1
Bridging 25 条 bridge 过滤规则	203.2	117.8	174.3
Routing 默认开启 fast path	336.9	454.8	1035
Routing 25 条防火墙过滤规则	120.3	68.7	93.8

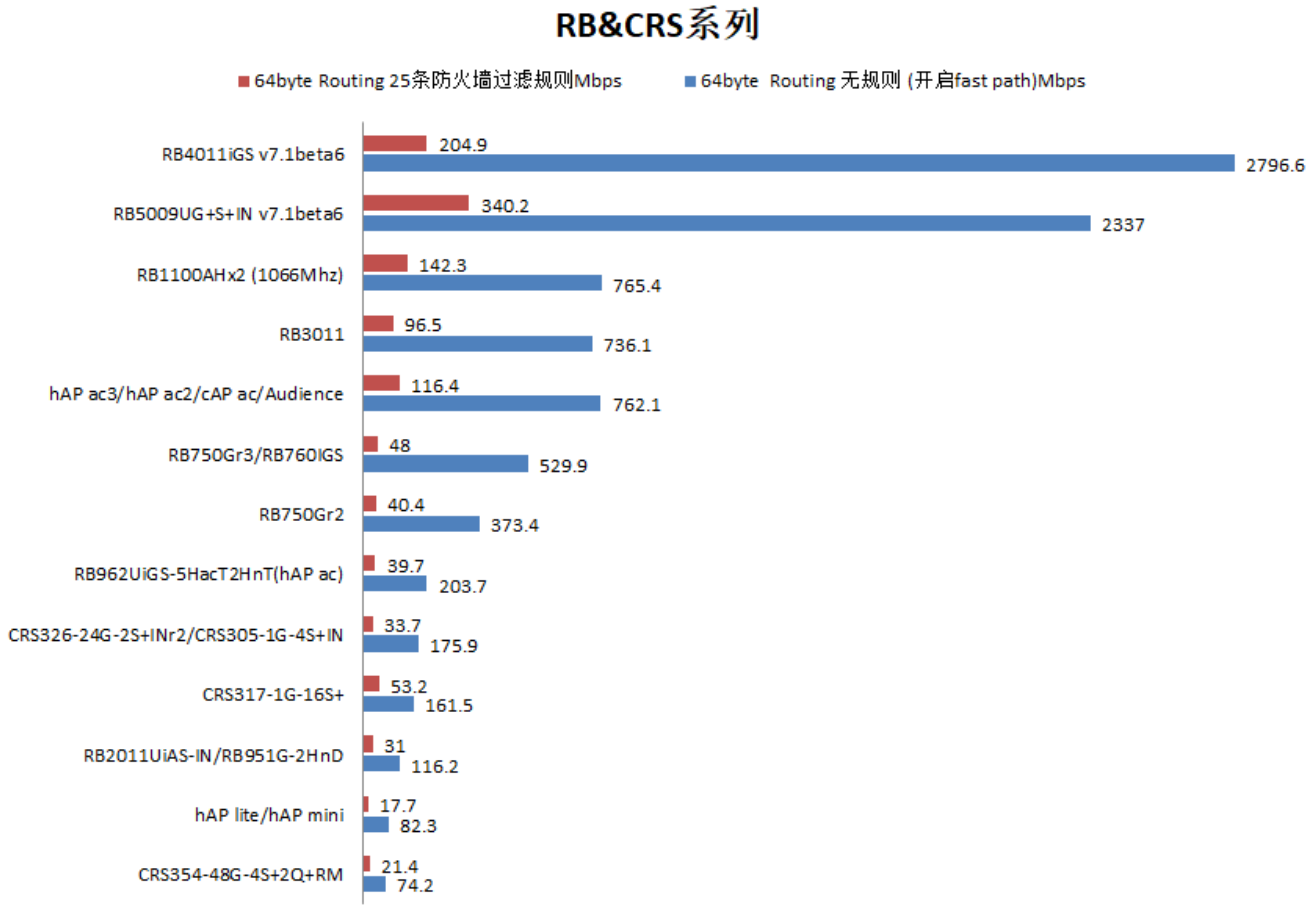
从这里一对比，处理频率最高的是 MT7621A，达到 880MHz，纯桥接和路由转发性能很猛，但进入实际运行环境性能还不如频率只有 533MHz PowerPC 处理器的 RB850x2，在桥接和路由模式下配置规则性能丢失分别是 83%和 91%，都高于其他两款产品。大家注意！MT7621A 是 4 核，频率也高！配置规则后在转发数据方面，没有比基于 QCA9556 720MHz 的 RB750Gr2 高多少。

下面图是，64byte 的路由模式下，选择路由模式的数据更能体现路由器作为三层 IP 路由转发性能，桥接其实很多 RB 设备将二层数据下发给交换芯片处理的能力，这个模式下有两种：无规则转发和添加 25 条防火墙规则转发，数据是 Mbps，也就是在这个模式下实际的带宽转发流量。



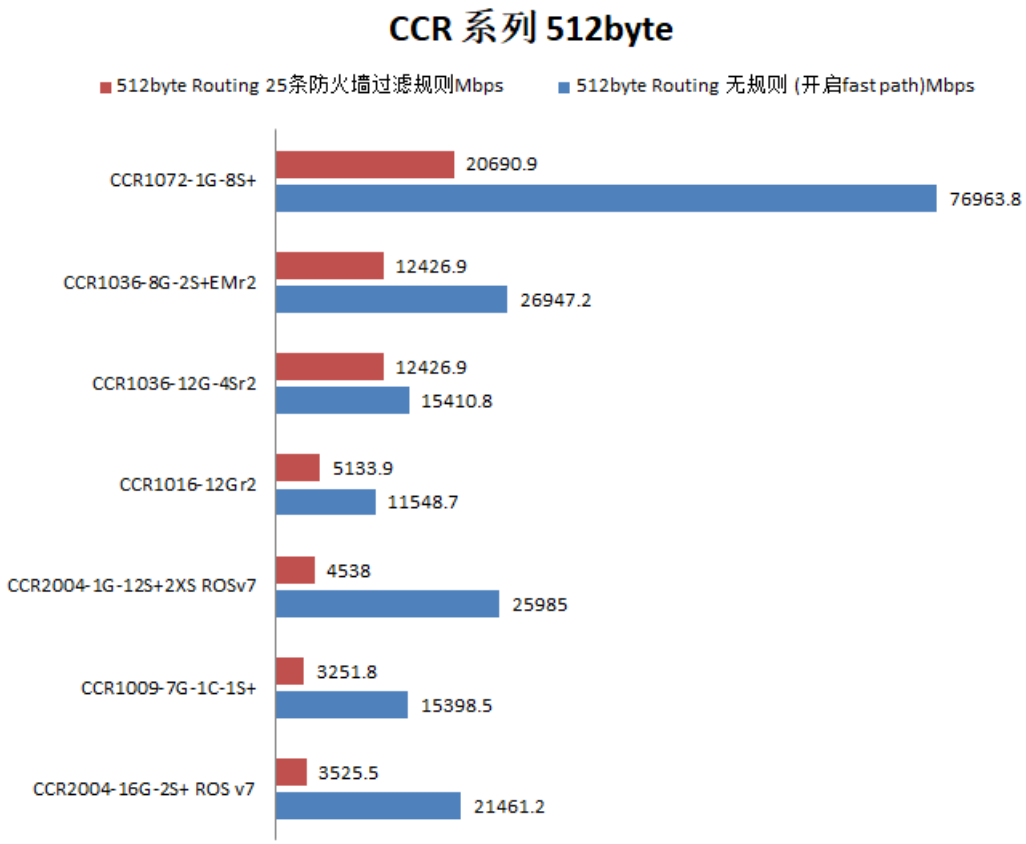
CCR2004-1G-12S+2XS 虽然无规则转发性能不如 CCR1009-7G-1C-1S+，但在添加 25 条防火墙规则后的吞吐量性能，反超 CCR1009-7G-1C-1S+。

同样我们在看看 RB 和 CRS 系列：



RB4011 在路由无规则的 64byte 转发性能上看打败了 RB5009，加入 25 条防火墙规则后，RB5009 反超了。

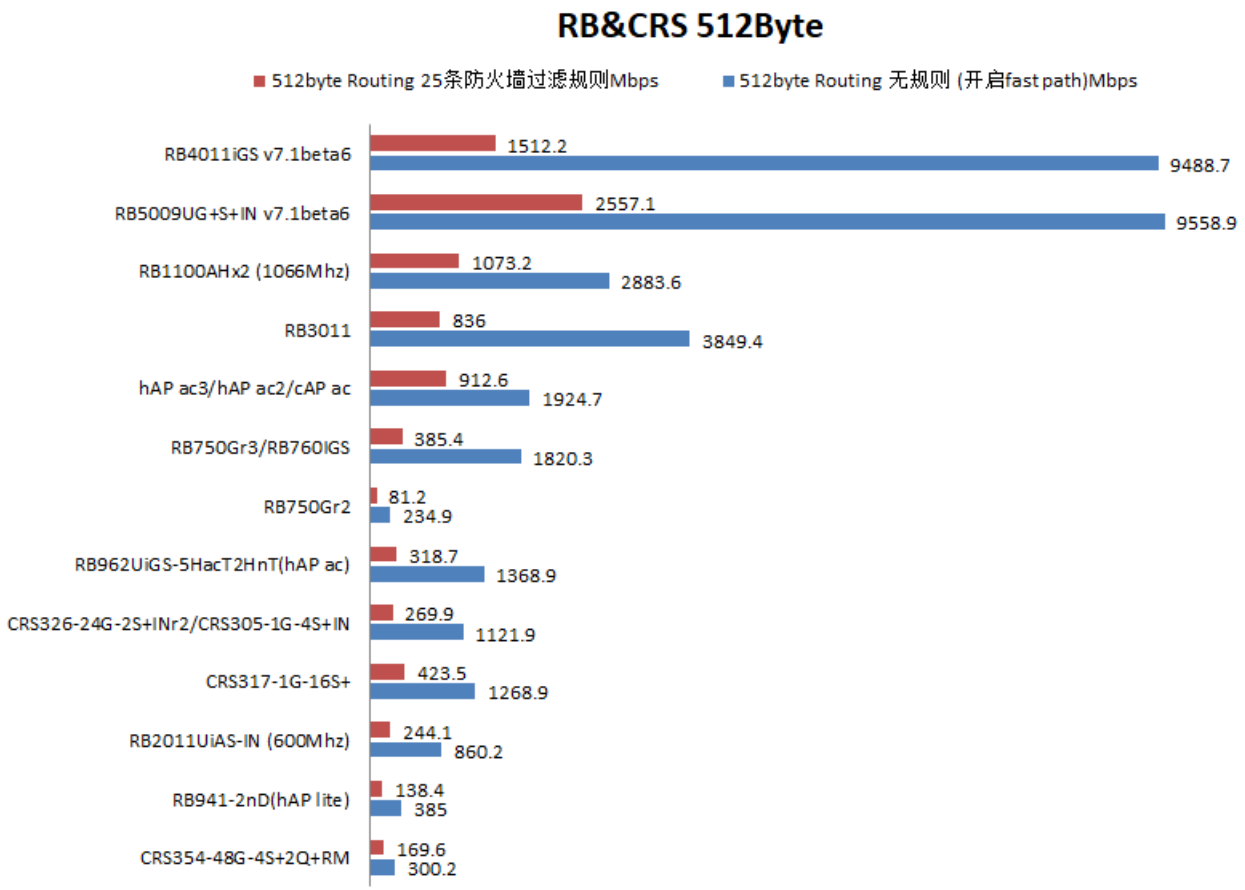
为了更了解设备性能，再来看看 512byte 综合性能的测试



你敢信这个 512byte 的综合性能，CCR2004-1G-12S+2XS 无规则下干翻了 CCR1036-12-4Sr2，直逼 CCR1036-8G-2S+EMr2，有 25 条防火墙规则下差距还是很明显，但和 CCR1016-12Gr2 差距不大了，CCR1xxx 系列是 Tile 64bit 构架处理器，CCR2004 是基于 ARM 64bit 的 4 核心处理器，也就单核性能已经超过了 CCR1xxx 系列。规则越多对 Tile 构架的 CCR1xxx 性能影响更大，因此在高负载下 CCR2004 的单核性能更加明显，CCR1xxx 系列完全需要 Linux 内核优化众核心的调度，来解决负载问题。

我个人认为提升 64byte 包转发性能主要在于 CPU 的时钟频率，以及多核心 CPU 的工作数量，但单核性能又决定了对数据包拆包、分析和过滤等性能，因此在 CCR1xxx 系列在多核心数量体现了小包转发的优势，但受到单核心性能影响，在防火墙过滤规则创建后性能下降很厉害的原因。

又来看看 RB 和 CRS 系列



RB5009 在 512byte 无规则情况下，略胜 RB4011，在 25 条防火墙规则几乎是碾压，说明 RB5009 在综合性能上比 RB4011 更胜一筹。RB5009 使用的是 Cortex A72 处理器，而 RB4011 是 Cortex A15，两者 1.4G 频率相同的情况下，包转发性能差不多，但更先进的 Cortex A72 在处理防火墙规则时性能表现几乎是碾压式的。

hAP ac 的综合性能也接近 RB750Gr3，CRS317 作为一款 CRS 交换机，在 CRS 系列里 CPU 性能也是最好的，仅次于 hAP ac。hAP ac2 对 BR750Gr3 在无规则情况下两者差距不大，一旦创建防火墙规则后，hAP ac2 是 RB750Gr3 性能的 2.3 倍。

对于普通家庭用户，除了 RouterOS 自带的基本防火墙规则，一般不会创建更多规则，因此选择 RB750Gr3 和 hAP ac2 已经足够了，少量功能要求和汇聚的用户 hAP ac2 也够用，300-400 元的 RouterOS 设备家用性价非常不错了。

对于多线汇聚和更多功能要求的发烧友，1-2G 左右选择 RB4011 或者 RB5009 吧，带宽更大的就是 CCR 系列了，如果 CCR 性能还是不够，那还是上 x86 服务器至强系列（但兼容性不敢保证），如果选择虚拟平台的 CHR 性能释放要欠缺点！

以上数据来至 MikroTik 官方。

3.4 RouterBOARD 的型号与分析

RB411 系列与 RB433 系列

- RB411 与 RB433 主要区别是无线 miniPCi 的接口和以太网口数量不同，根据安装网卡数量的不同使用场合也有区别，RB411 是无线型而 RB433 是混合型。
- 在无线方面 RB411 与 RB433 都可以作为城市的无线覆盖产品，433 系列主要用于中继与多点覆盖和多点传输，411 系列主要用于普通无线覆盖，点对点，与点对多点的低成本解决方案
- 有线方面 RB433 系列可以实现双线接入

RB411AR 与 RB711

- 都是集成无线网卡，RB411AR 更侧重于 WiFi 覆盖，RB711 和 RB711A 侧重于 5G 骨干无线传输
- RB711/A 是集成 5G-a/n 的无线网卡， 23dBm 仅支持 5G 传输，不支持 2.4G， 支持协议是 802.11a 和 802.11n，不支持 MiniPCI 扩展
- RB411R/AR 是集成 2.4G 无线网卡，仅支持 2.4G 传输，不支持 5G，支持协议是 802.11bg
- RB411R 没有 MiniPCI 扩展功能，RB411AR 支持 1 个 MiniPCI 槽扩展
- 新的 RB711-2Hn 更多考虑到 11n 的 WiFi 覆盖

RB450 系列与 RB750 系列

- RB450 和 RB450G 是较早的 5 口路由器，没有无线功能，CPU 分别是 300MHz 和 680MHz，主板与外壳分开销售，也主要针对 OEM。
- RB750 和 RB750G 其实是 RB450 和 RB450G 的简化版本，RB750 系列销售整套的产品(含塑料外壳和电源)，产品能直接面对终端客户（虽然价格和性能有很大优势，但在国内这样的外壳包装就比较掉价）
- RB450 的 CPU 是 AR7130 300MHz 其实不如 RB750 的 AR7240 的 400MHz，但 MikroTik 为了不让 RB450 在性能和价格上尴尬境地，刻意将将后期的 RB750 的 CPU 降频到 300MHz。
- RB450G 整体性能上要强于 RB750G，不管从内存还是交换芯片，都好于 RB750G，他们的 CPU 都相同，所以性能上 RB450G 强于 RB750G，但性价比上 RB750G 更有优势
- RB750 和 RB450 完全能满足 50 台计算机的网络环境，RB450G 表现就比较好，如果不建议 CPU 较高，可以支持 180 台计算机的网络，RB750G 就相对低点 150 台
- 从发展来看 RB750 系列被 MikroTik 重点发展，后期会增加 RB751 系列，增加 USB 和集成 11n 的无线网卡

注： RB400 支持 switch 功能，即通过 IC 控制二层数据转发，不需要经过 CPU 处理（RB100 系列有这样的功能，但不被完全支持），也具备数据镜像的设置

RB1100 系列

RB1100 是 13 个千兆以太网口的，非常满足多线路接入的网络，一次可以接入 12 条外线，这样节省了交换机的费用，同样如果你只有一条外线可以把 12 个以太网口设置成类似交换机的功能，直接替代了主交换机的功能，RB1100AH 和 RB1100AH×2 也是同样的 13 个口，只是性能更强。

RB1200 的 CPU 和 RB1100AH 相同，只是采用了 10 个万兆网卡的解决方案，看似网卡吞吐量大了，但设备性能没有任何提升

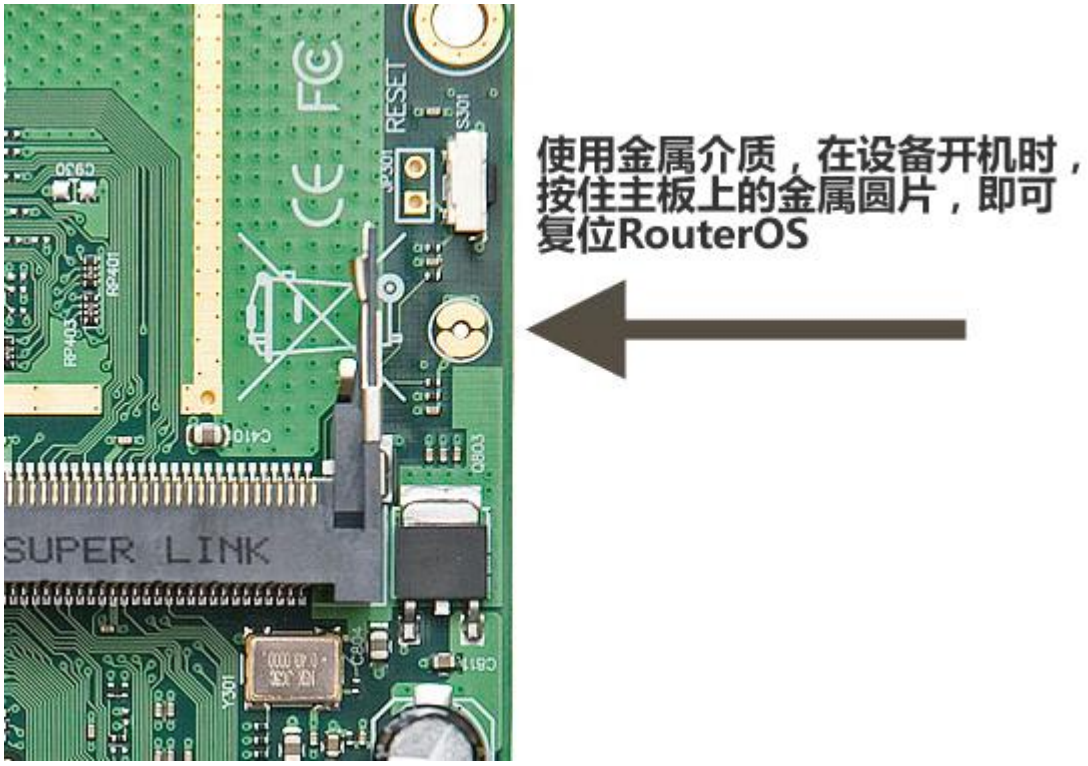
注：RB1000 产品是 MikroTik 第一款 1.3G 处理器的高性能路由器，但也是最失败的，因为返修率太高，曾经出现 10 台里有 4-5 台无法启动，但 MikroTik 一直否认，这就是为什么后来被 800MHz 处理器的 RB1100 替代，RB1100 系列（RB1100、RB1100AH 和 RB1100AH×2）作为 13 口的多线路由器，针对有线网络设计，特别是网吧和小区宽带，但到 2014 年 CCR 成为主流后，只保留了 RB1100AH×2 一款产品。

3.5 RouterBOARD reset 复位键

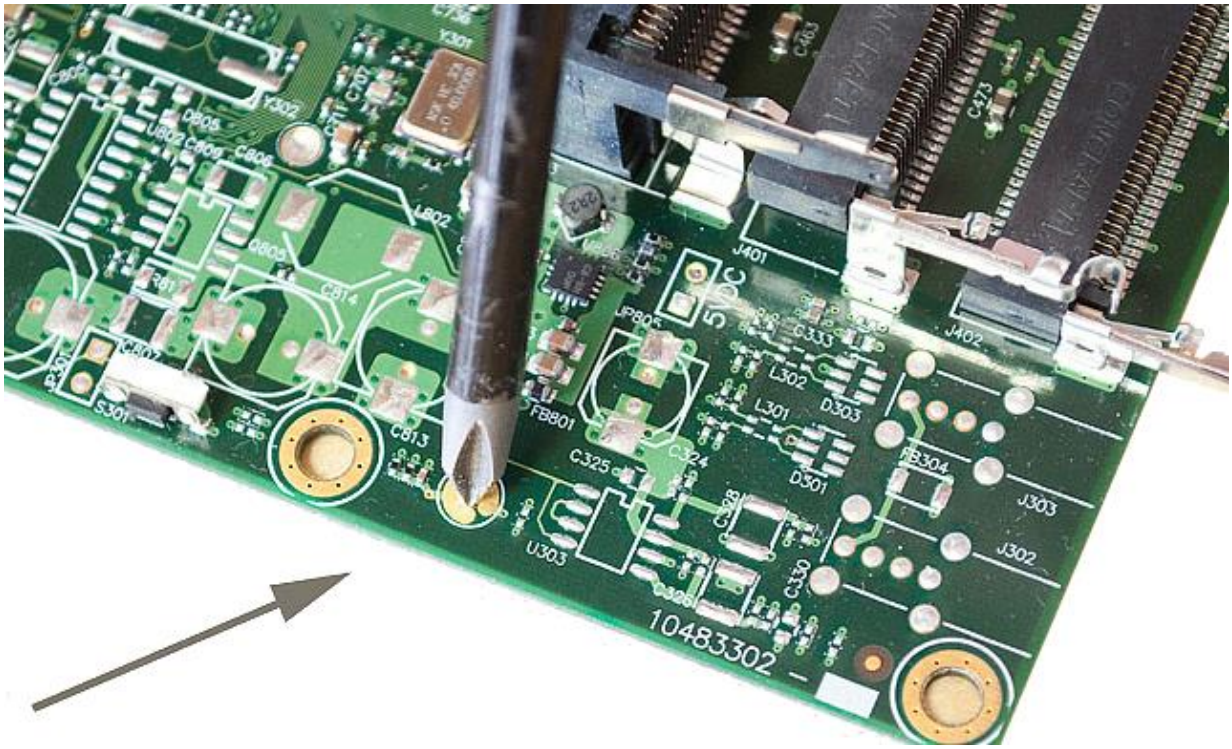
RouterBOARD 是硬件化的 RouterOS，当 RouterBOARD 的 RouterOS 密码丢失或配置错误后，可以通过 reset 键复位或者 Netinstall 重装 RouterOS 系统，reset 复位键对于大多网络工程师来说不陌生，启动时候按住则可以复位，早期和当前 RB 设备的复位有区别。

早期的 RB 设备

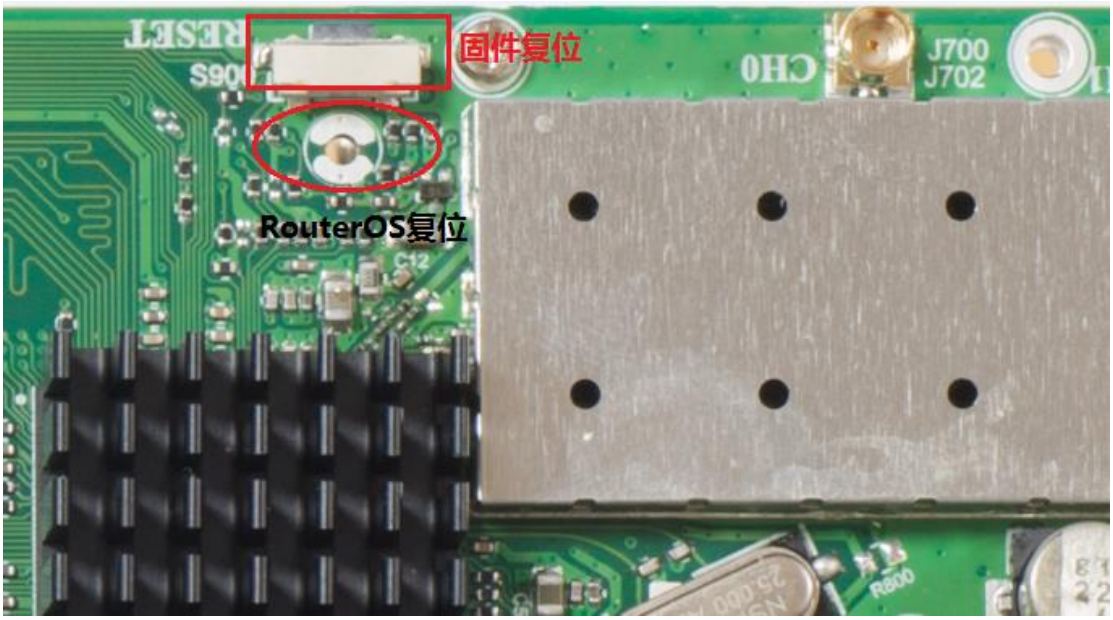
如下图是 RB411 主板的金属复位片。**注意：**旁边白色的按钮，则是固件复位按钮，而非 RouterOS 的复位按钮。



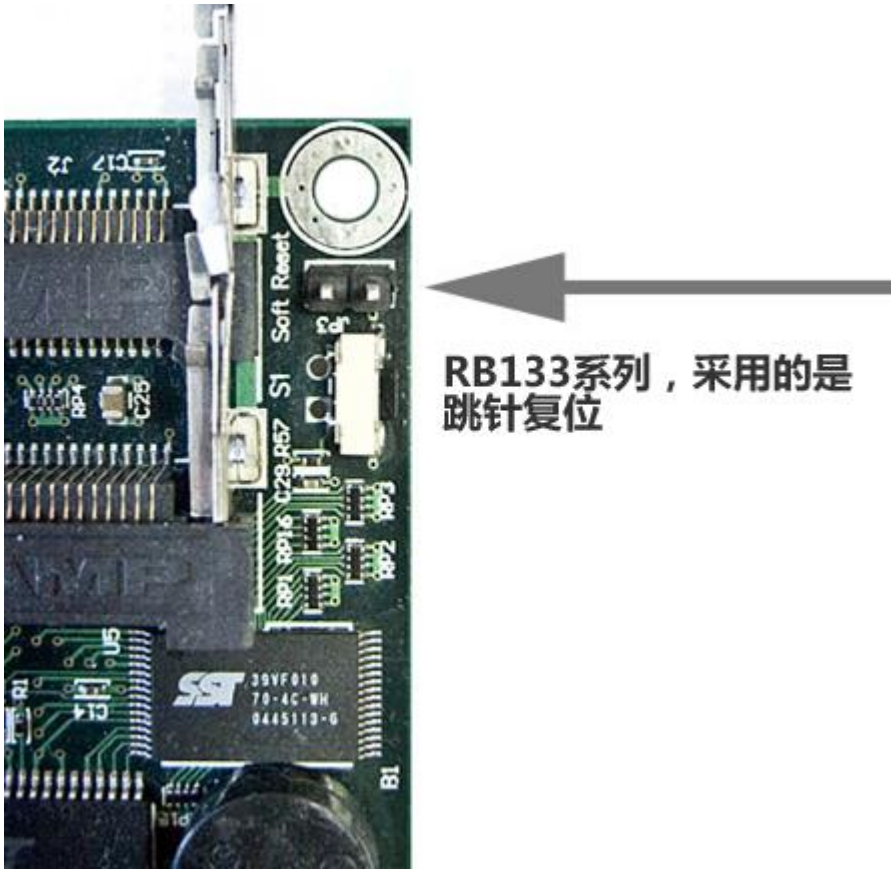
下图是通过一个十字螺丝刀复位的操作，你也可以用其他金属工具将两个铜片短路。



下图是早期 RB2011USA-2HnD 的复位按钮，靠外侧的是固件复位按钮（启动时长按固件复位按钮，可以通过 ethernet 方式引导），后面圆形的是 RouterOS 复位接口（在启动前短路这个复位接口，直到启动完成，RouterOS 软件就会复位）。



下图是 RB133 复位，我们使用的是跳针，即需要跳线帽进行 RouterOS 配置的复位：



注意，当你复位完成后，不要忘了把跳线帽移除，否则会出现重启后，再次复位配置。

当然遇到忘记 RouterOS 登陆密码，那就需要使用另外一种方式，一个是之前提到到圆形复位铜片，但这个对于有些安装外壳的 RouterBOARD 不太方便，所以需要另一种方式，即在 RouterBOARD 启动时按照固件复位按钮，采用 netinstall 引导安装

当前 RB 设备

当前的 RB 设备，在外壳周围都提供了 reset 按钮：

如上图，在 RB 设备启动时，根据按 reset 键的时间长短有不同的功能，功能包括，固件复位，RouterOS 系统复位，CAPs 模式，Netinstall 安装启动等。

RB Reset 复位键的功能说明

- **载入备份 RouterBOOT 固件**
  启动后，按住复位键，在通电 3 秒后，会重新载入备 Boot 固件。这种情况通常在固件升级失败后，重新启动载入备份固件。（**3 秒**)

- **复位 RouterOS 配置**
  通电后，如果你持续按住复位键，直到 LED 灯闪烁，放开复位键后，RouterOS 配置将会复位(**5 秒**)

- **启用 CAPs 模式**
  让该设备连接到网络中的 CAPsMAN，持续按住复位键大于 5 秒，LED 灯会常亮，放开后会转入 CAPs 模式 (**10 秒**)

- **启动 RouterBOARD Netinstall 模式（网络安装）**
  通电后持续按住复位键，直到 LED 灯常亮关闭，放开复位键，此时 RouterBOARD 会寻找网络中的 Netinstall 服务。你同样可以根据 windows 中的 Netinstall 程序查找到，并显示 RouterBOARD 信息再放开复位键。(**15 秒**)

注：操作视频链接

3.6 Reset 和 Mode 按键

MikroTik 硬件设备都有 reset 按钮，部分设备还带有一个 Mode 按钮，他们作用如下：

- Reset 按钮除了能在路由器启动时复位、设置 CAP 和启动网络安装 RouterOS 外，还能在路由器运行时定义执行脚本。
- Mode 按钮在部分 RouterBOARD 设备支持，此按钮提供了对 RouterOS 的脚本支持，也就是当按下 mode 按钮，RouterOS 会执行指定的脚本。

当前支持 mode 按钮的设备列表：

- RBcAP-2nD (cAP)
- RBcAPGi-5acD2nD (cAP ac)
- RBwsAP5Hac2nD (wsAP ac lite)
- RB750Gr3 (hEX)
- RB760iGS (hEX S)
- RB912R-2nD (LtAP mini, LtAP mini LTE/4G kit)
- RBD52G-5HacD2HnD (hAP ac^2)
- RBLHGR (LHG LTE/4G kit)
- RBSXTR (SXT LTE/4G kit)
- CRS328-4C-20S-4S+RM
- CRS328-24P-4S+RM
- CCR1016-12G r2
- CCR1016-12S-1S+ r2
- CCR1036-12G-4S r2
- CCR1036-8G-2S+ r2
- RBD53G-5HacD2HnD (Chateau)
- RBD53GR-5HacD2HnD (hAP ac^3)

按钮功能属性

属性	描述
enabled (*no* | *yes*; 默认: **no**)	启用或禁用 reset 或 mode 按钮
hold-time (*time interval Min..Max*; 默认:)	按钮被按下的一段时间范围，mode 按钮时间范围可以调整（Min..Max）： Min -- 0s..1m (时间), Max -- 0s..1m (时间) ，用于设置最小时间和最大时间范围，该参数获得仅从 RouterOS 6.47 开始
on-event (*字符*; 默认:)	Reset 或 Mode 按钮被按下后，执行的脚本。脚本可以在此项定义，也可以调用 "/system scripts" 下的定义好的脚本名称

Mode 按钮实例：

按下 Mode 按钮后，执行一个 info 日志信息脚本，在 log 下显示 mode button is pressed

```
/system script add name=test-script source={:log info message=("mode button pressed ");}
/system routerboard mode-button set on-event=test-script enabled=yes
```

修改 mode 按钮的 hold-time 时间为 3~5 秒，即按压 mode 按钮，到 3 秒开始到第 5 秒结束的时间范围执行脚本，配置如下：

```
/system script add name=test-script2 source={:log info message=("test2");}
/system routerboard mode-button set on-event=test-script2 hold-time=3..5 enabled=yes
```

Reset 按钮工作方式也相同，在 /system routerboard reset-button 定义，实例如下：

```
/system script add name=test-reset-button source={:log info message=("reset button pressed");}
/system routerboard mode-button set on-event=test-reset-button hold-time=0..10 enabled=yes
```

注意：从 RouterOS 6.47 reset 按钮也支持 hold-time 参数，用于修改按钮时间间隔范围，reset 按钮执行脚本只能是在路由器已经启动完成下操作。(相关视频操作)

3.7 升级 RB/CCR/CRS 固件

RB/CCR/CRS 系列硬件产品 BIOS 固件会不定期更新，除了对 RouterOS 系统升级外，同样 MikroTik 会对启动引导和硬件兼容性进行修正和更新。

早期 RouterBOARD 固件后缀名为.fwf，每个系列的 RouterBOARD 所使用的固件都不相同，如下面的列表：

RB 型号	固件前缀
CCR1016，CCR1036	-
RB1000，RB1100，RB1100AH，RB1100AHx2，RB1200	mpc8548
RB2011 系列	ar9344
RB900 系列	ar9344
RB800	mpc8343
RB600	mpc8343
RB333	mpc8323
RB400 系列(411/A/AH、433/AH、433AH、450/G、493/AH)：	ar7100
RB700 系列（750、750G、711、751）包括 SXT，Groove	ar7100
RB532	rc32434
RB100 系列（112、133/C、150、192）	adm5120

每隔一段时间，RouterBOARD 的固件都会更新一次，所以通过在 RouterOSv5 版本都需要单独下载固件升级，升级固件只能在命令行操作，首先我们需要查看 RouterBOARD 当前的固件情况如下图：

```
[admin@MikroTik] > system routerboard
[admin@MikroTik] /system routerboard> print
          routerboard: yes
                model: 751U-2HnD
        serial-number: 2E3E013F56C8
    current-firmware: 2.36
    upgrade-firmware: 2.38
[admin@MikroTik] /system routerboard>
```

如上面看到的，current-firmware 是当前的固件信息：2.36，而我们最新的估计是 2.38 所以我们要通过上传固件到 RouterOS 的 file 目录中（通过拖放的方式放到 winbox 中的 Files 列表根目录下）

上传完后，然后我们通过 upgrade 命令升级：

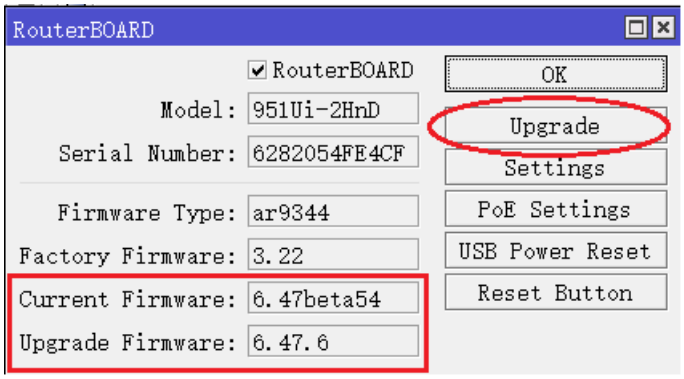
```
[admin@Office] /system routerboard> upgrade
Do you really want to upgrade firmware? [y/n]
y
```

```
firmware upgraded successfully, please reboot for changes to take effect!
[admin@Office] /system routerboard>
```

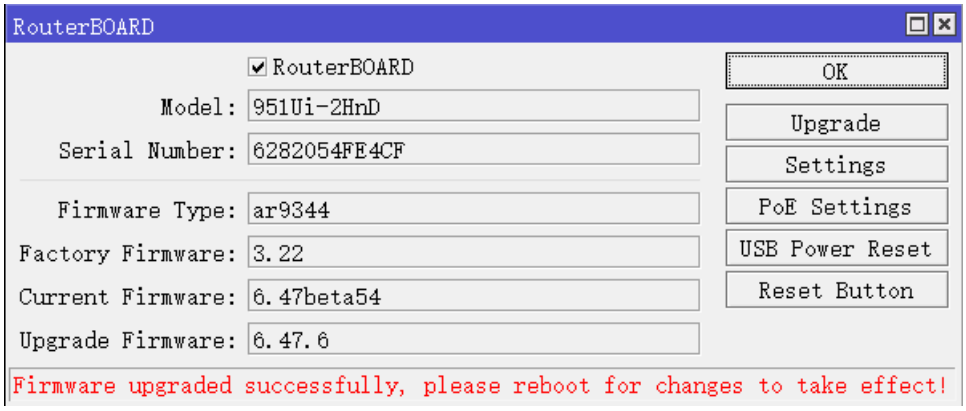
按照提示升级固件，升级后要求重启设备，才能完成更新。

但从 RouterOS v6 后期版本，固件跟随 RouterOS 系统整合在一起，升级 RouterOS 系统后，可以查看到固件版本信息和 RouterOS 版本号类似，不再需要单独的下载固件文件，全部都集成到了 RouterOS 系统，跟随 RouterOS 系统一起

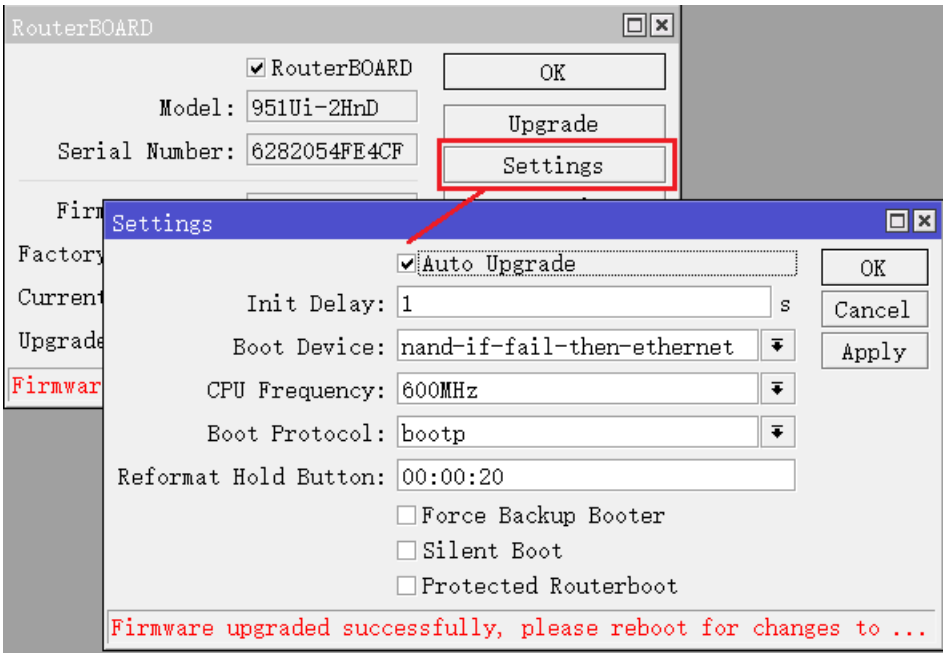
如下图，Winbox 进入 system RouterBOARD 菜单，可以看到当前固件是 6.47beta54，可以升级固件是 6.47.6，点击 Upgrade 进行升级：



点击后，会出现以下提示，要去重启路由器生效：



如果在对 RouterOS 系统升级，希望自动更新固件，可以在 routerboard 菜单点击 settings，勾选 Auto Upgrade，每次升级 RouterOS 系统重启完成后，都自动升级固件，但还是需要再次重启路由器：



3.8 RouterBOARD Settings

操作路径: /system RouterBOARD settings

在 settings 菜单下，可以查看并设置当前 RotuerBOARD 固件参数，例如 RS232 连接速率，引导设备，跳线设置等

```
[admin@MikroTik] /system routerboard settings> print
              baud-rate: 115200
             boot-delay: 2s
        enter-setup-on: any-key
            boot-device: nand-if-fail-then-ethernet
          cpu-frequency: 600MHz
          boot-protocol: bootp
  enable-jumper-reset: yes
  force-backup-booter: no
            silent-boot: no
```

属性描述

属性	描述
baud-rate (*整型*; 默认: **115200**)	如果 RB 设备有 RS232 接口，选择主板上的 RS232 接口的速率
boot-delay (*时间*; 默认: **1s**)	引导时多长时间对键盘相应
boot-device (*nand-if-fail-then-ethernet ...*; 默认:**nand-if-fail-then-ethernet**)	选择 RouterBOOT 加载操作系统的引导方式

	• flash-boot - • flash-boot-once-then-nand - • nand-if-fail-then-ethernet - • nand-only - • try-ethernet-once-then-nand -
boot-protocol (*bootp* |*dhcp ...*; 默认: **bootp**)	Boot protocol to use: • bootp – RouterOS 引导的默认选项 • dhpc – 用于 OpenWRT 和其他可用的 OS
cpu-frequency (根据型号; 默认: **根据设备信号**)	这个选项允许修改设备的 CPU 频率，频率修改根据设备型号，即非所有设备可以修改频率，根据此选项信息提示。
cpu-mode (*power-save* | *regular*; 默认: **power-save**)	是否选择 CPU 的休眠模式在 HTL 指令状态下，暂停指令 HLT(Enter Halt State Instruction)，在等待中断信号时，该指令使 CPU 处于暂停工作状态。在 CPU 的空闲周期状态下，多数 OS 会使用 HLT 指令，当 CPU 进入休眠模式，会进入低功耗状态，但在较低的温度环境下建议使用正常模式（regular）
enable-jumper-reset (*yes* | *no*; 默认: **yes**)	是否禁用跳线复位，禁用可用于忽略意外操作导致复位跳线
enter-setup-on (*any-key* | *delete-key*; 默认: **any-key**)	选择那一个键进入 BIOS 配置模式在引导提示周期内。注意一些串口终端程序，可能不支持使用 delete 键进入设置，有些支持 Backspace 键。
force-backup-booter (*yes* | *no*; 默认: **no**)	如果使用备份 RouterBOOT，仅用于主引导器未知的崩溃，且无法修复。因此你无需在启动设备按重置 reset 键。可选择该设置做每次加载 • yes – 备份引导器将被使用 • no – 主引导器使用
memory-frequency (根据型号; 默认: 根据型号)	该选项可根据具体设备型号，修改内存设备的内存频率，在 winbox 中可根据此选项信息提示查看设置参数
silent-boot (*yes* | *no*; 默认: **no**)	这个选项禁止在串口终端输出信息，且蜂鸣器在引导过程中不发声。该功能用于有一些温度监控设备或 modem 连接到串口 • yes – 串口终端无输出，且蜂鸣器在引导过程中不发声(不会禁用 RouterOS 的 beep 脚本命令) • no – 正常启动和输出串口终端信息

3.9 RouterBOARD Switch 介绍

该节主要介绍的是 RouterBOARD 的 RouterOS 交换芯片功能(该功能从 v4.0 版本开始出现)，不同型号的 RouterBOARD 使用不同的交换芯片，他们大多（Other 是其他 RouterBOARD 产品）支持基本的“Port Switching”端口交换功能，即实现以太网交换机 。

当前 RouterBOARD 使用了几种类型的交换芯片，这些芯片有不同的功能。他们大多都具备基本的 Port Switching 功能（端口交换功能），但他们之间还有些区别：

表一

功能	QCA8337	Atheros8327	Atheros8316	Atheros8227	Atheros7240
端口交换	yes	yes	yes	yes	yes
端口镜像	yes	yes	yes	yes	yes
TX 流控	yes	yes	yes	yes	yes
RX 流控	yes	yes	no	no	no
主机 MAC 列表	2048 条	2048 条	2048 条	1024 条	2048 条
Vlan 列表	4096 条	4096 条	4096 条	4096 条	16 条
策略数量	92 条	92 条	32 条	no	no

表二

功能	ICPlus175D	MT7621	RTL8367	98PX1012	Other
端口交换	yes	yes	yes	no	yes
端口镜像	yes	yes	yes	no	no
TX 流控	no	no	no	no	no
RX 流控	no	no	no	no	no
主机 MAC 列表	2048 条 2	2048 条	2048 条	no	no
Vlan 列表	no	no	no	no	no
策略数量	no	no	no	no	no

RB 采用的交换芯片

RouterBoard	交换芯片型号
RB4011iGS+	RTL8367 (ether1-ether5); RTL8367 (ether6-ether10);
RB1100AHx4	RTL8367 (ether1-ether5); RTL8367 (ether6-ether10); RTL8367 (ether11-ether13)
RB750Gr3 (hEX), RB760iGS (hEX S)	MT7621 (ether1-ether5)
RBM33G	MT7621 (ether1-ether3)
RB3011 series	QCA8337 (ether1-ether5); QCA8337 (ether6-ether10)
RB OmniTik ac series	QCA8337 (ether1-ether5)

RBwsAP-5Hac2nD (wsAP ac lite)	Atheros8227 (ether1-ether3)
RB941-2nD (hAP lite)	Atheros8227 (ether1-ether4)
RB951Ui-2nD (hAP); RB952Ui-5ac2nD (hAP ac lite); RB750r2 (hEX lite); RB750UPr2 (hEX PoE lite); RB750P-PBr2 (PowerBox); RB750P r2; RBOmniTikU-5HnDr2 (OmniTIK 5); RBOmniTikUPA-5HnDr2 (OmniTIK 5 PoE)	Atheros8227 (ether1-ether5)
RB750Gr2 (hEX); RB962UiGS-5HacT2HnT (hAP ac); RB960PGS (hEX PoE); RB960PGS-PB (PowerBox Pro)	QCA8337 (ether1-ether5)
RB953GS	Atheros8327 (ether1-ether3+sfp1)
RB850Gx2	Atheros8327 (ether1-ether5) ether1 可选项
RB2011 series	Atheros8327 (ether1-ether5+sfp1); Atheros8227 (ether6-ether10)
RB750GL; RB751G-2HnD; RB951G-2HnD; RBD52G-5HacD2HnD (hAP ac^2), RBD53iG-5HacD2HnD (hAP ac^3), RBD53GR-5HacD2HnD&R11e-LTE6 (hAP ac^3 LTE6 kit), RBD53G-5HacD2HnD-TC&EG12-EA (Chateau LTE12)	Atheros8327 (ether1-ether5)
RBcAPGi-5acD2nD (cAP ac), RBwAPGR-5HacD2HnD (wAP R ac and wAP ac LTE series), RBwAPG-5HacD2HnD (wAP ac), RBD25G-5HPacQD2HPnD (Audience), RBD25GR-5HPacQD2HPnD&R11e-LTE6 (Audience LTE6 kit),	Atheros8327 (ether1-ether2)
RBD22UGS-5HPacD2HnD (mANTBox 52 15s)	Atheros8327 (ether1-sfp1)
RB1100AH	Atheros8327 (ether1-ether5); Atheros8327 (ether6-ether10)
RB1100AHx2	Atheros8327 (ether1-ether5); Atheros8327 (ether6-ether10)
CCR1009-8G-1S-1S+; CCR1009-8G-1S	Atheros8327 (ether1-ether4)
RB493G	Atheros8316 (ether1+ether6-ether9); Atheros8316 (ether2-ether5)
RB435G	Atheros8316 (ether1-ether3) ether1 可选项
RB450G	Atheros8316 (ether1-ether5) ether1 可选项
RB450Gx4	Atheros8327 (ether1-ether5)
RB433GL	Atheros8327 (ether1-ether3)
RB750G	Atheros8316 (ether1-ether5)

RB1200	Atheros8316 (ether1-ether5)
RB1100	Atheros8316 (ether1-ether5); Atheros8316 (ether6-ether10)
DISC Lite5	Atheros8227 (ether1)
RBmAP2nD	Atheros8227 (ether1-ether2)
RBmAP2n	Atheros7240 (ether1-ether2)
RB750	Atheros7240 (ether2-ether5)
RB750UP	Atheros7240 (ether2-ether5)
RB751U-2HnD	Atheros7240 (ether2-ether5)
RB951-2n	Atheros7240 (ether2-ether5)
RB951Ui-2HnD	Atheros8227 (ether1-ether5)
RB433 series	ICPlus175D (ether2-ether3); 旧型号采用 ICPlus175C
RB450	ICPlus175D (ether2-ether5); 旧型号采用 ICPlus175C
RB493 series	ICPlus178C (ether2-ether9)
RB816	ICPlus178C (ether1-ether16)

在路径/interface ethernet switch，使用 print 命令查看设备中的交换芯片型号：

```
[admin@MikroTik] > /interface ethernet switch print
Flags: I - invalid
 #   NAME      TYPE          MIRROR-SOURCE     MIRROR-TARGET     SWITCH-ALL-PORTS
 0   switch1   Atheros-8327   none              none
 1   switch2   Atheros-8227   none              none
```

根据交换芯片类型，获得各自的配置功能和参数

3.10 RouterBOARD Switch 配置

随着 RouterOS 4.0 发布后，RouterBOARD 系列路由产品开始支持基于交换芯片的硬件处理，如 RB450、RB750、RB433、RB493 和 RB1100 等路由器，在 v6.41 版本加入二层交换机的 master 配置迁移到 bridge port 下。

警告：从 6.41 后，在 RB 和 CRS 设备的 Switch 和 Bridge 同时负责二层交换配置，Bridge 又重复添加了 VLAN 配置，对于 RB 和 CRS1xx/2xx 系列，还不能完全实现二层交换芯片处理，必须在 Switch 菜单配置 VLAN，如果 VLAN 配置在 Bridge 会消耗 CPU 资源。当前只有 CRS3xx 系列能实现 Bridge 的 VLAN 配置下发交换芯片处理，Switch 菜单中没有 VLAN 配置选项。所以不同产品的二层 VLAN 配置存在差异。对于 CRS3xx 系列交换机可以参考单独的手册《CRS3 系列交换机配置手册》

基本原理

在 v6.41 前的 RB 和 CRS 设备，实现二层端口的线速转发，要求把所需以太网端口指定到同一个“master port”，在同一个 master port 下的端口组成一个二层交换，由 master 端口负责与交换芯片和 RouterOS 系

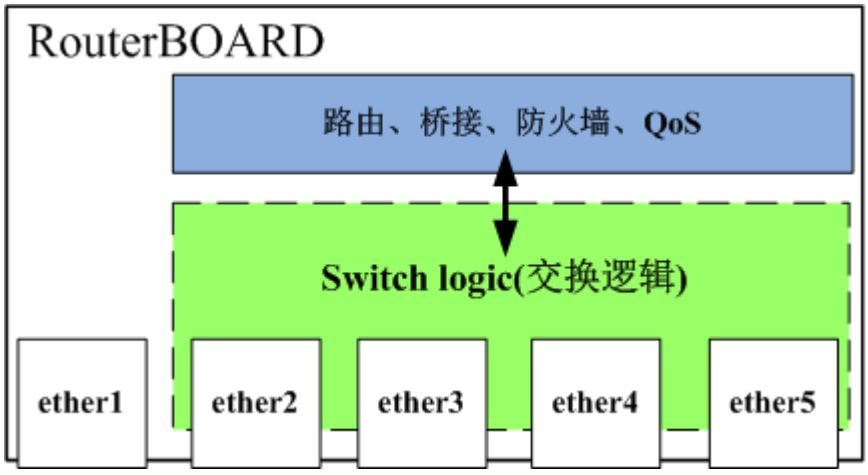
统建立数据通道，v6.41 版本开始 master port 被取消（为防止对新版本 RouterOS 操作内容理解混乱，master port 内容从《RouterOS 入门到精通》v6.6 开始删除），取而代之的是在/bridge port 配置交换端口，请参考 14.7 章节。

Switch All Ports 功能

Ether1 端口在 RB450G/RB435G/RB850Gx2 有一个功能，允许将 ether1 删除或添加到默认的交换罗辑组中。进入/interface ethernet switch，配置命令 set switch1 switch-all-ports=no

属性	描述
switch-all-ports (no | *yes*; 默认: **yes**)	修改 ether1 的交换端口分组，此参数仅对 RB450G/RB435G/RB850Gx2 设备有效。 yes - ether1 属于交换端口分组，并支持 Atheros8316/Atheros8327 所有其他高级功能，包括数据统计（/interface ethernet print stats） no - ether1 不属于交换端口分组，使其成为独立的以太网端口，即一个独立的三层接口，但取消了该端口与其他端口的二层交换功能。

"yes" 即 ether1 是交换逻辑的一部分，并支持交换逻辑组，"no" 即 ether1 不属于交换逻辑的一部分，成为一个独立的以太网端口



Host Table(MAC 列表)

主机列表是交换芯片记录到对应端口的 mac 映射表。该列表包含 2 类条目：动态和静态。动态条目是自动添加，即被称为学习过程。当交换芯片从端口接收到一个数据帧，会自动添加数据帧的源 MAC 地址到列表中，创建一个动态的条目，假如这个 MAC 是 X，那么当其他交换端口上的 A 主机需要和 MAC X 通信，会进入主机列表查询，当发现 MAC X 在主机列表中，主机 A 将会查询到 MAC X 对应的端口，交换芯片会将主机 A 的数据转发到 MAC X 所在的端口，整个过程就是交换机工作的基本原理是学习，存储和转发。每条动态条目超时时间为 5 分钟。

完成整个这个工作前提是所有端口都必须在交换组里，因此在主机列表中没有看到动态的 MAC 地址，请确认以太网端口加入到了/bridge port 下，当动态 MAC 地址存在于列表中，可以将动态 MAC 修改为静态 ，当添加静态条目后，可以获得更多的功能。

属性	描述
copy-to-cpu (*no* | *yes*; 默认: **no**)	是否根据匹配帧的目标 MAC 地址拷贝到 CPU 端口(对于 CRS3xx 系列交换机来说，匹配的目的地址或源地址)
drop (*no* | *yes*; 默认: **no**)	是否丢弃在某个端口接收到帧匹配的源 MAC 地址(CRS3xx 系列交换机包括目的地址或源地址匹配)
mac-address (*MAC;* 默认: **00:00:00:00:00:00**)	主机的 MAC 地址
mirror (*no* | *yes*; 默认: **no**)	是否根据帧的目标 MAC 地址镜像数据到指定的端口(CRS3xx 系列交换机包括目的地址或源地址匹配)
ports (*name*; 默认: **none**)	选择网络端口名称，静态 MAC 地址可以映射到多个网络端口，包括交换机的 CPU 端口
redirect-to-cpu (*no* | *yes*; 默认: **no**)	是否根据帧的目标 MAC 地址重定向 Switch CPU 接口(CRS3xx 系列交换机包括目的地址或源地址匹配)
share-vlan-learned (*no* | *yes*; 默认: **no**)	静态 MAC 地址查找是与 SVL (shared-VLAN-learning)使用，还是与 IVL (independent-VLAN-learning)结合使用。SVL 模式用于不支持 IVL 二层网络，或关闭 IVL (independent-learning=no)的 VLAN 列表。
switch (*name*; 默认: **none**)	指定 MAC 地址将被分配到的 switch
vlan-id (*integer: 0..4095*; 默认:)	静态添加的 MAC 地址的 VLAN ID

交换芯片的 MAC 地址容量是有限的，具体大小参见 RB 介绍表。一旦主机表完全满了，会使用不同的方法来应对，例如，交换机会删除旧的 MAC 条目，为新的 MAC 地址腾出空间(在 QCA-8337 和 Atheros-8327 使用)，另一种则是忽略新的 MAC 地址，只在超时后删除 MAC 条目(Atheros8316, Atheros8227, Atheros-7240, ICPlus175D 和 Realtek-RTL8367 使用)，最后一个选项是前两个选项的组合——只允许更新一定数量的条目，并保持其他 MAC 地址部分完整，直到超时(MediaTek-MT7621 使用)。

警告：对于 Atheros8316, Atheros8227 和 Atheros-7240 交换芯片，当创建 bridge 后，加入指定端口到交换分组上， 并且至少有一个 hardware offloaded 端口处于活动状态时，switch-cpu 端口将始终参与主机 MAC 学习。这样的情况可能导致 switch-cpu 端口从非 hardware offloaded 端口学习到 MAC 地址。当单个 Bridge 包含 hardware offloaded 和非 hardware offloaded 接口时，可能会导致丢包。此外，当在同一交换组上使用重复的 MAC 地址时，无论主机是否位于不同的逻辑网络上，都可能出现丢包。因此对于上述芯片的交换机端口上，要么使用 hardware offloaded，要么禁用 hardware offloaded。

Port（端口设置）

此菜单下的属性用于配置支持 VLAN 过滤的交换芯片。这些属性仅对支持 VLAN 列表的交换芯片可用（CRS3xx 系列全部由 bridge vlan 代替，RB4011 和 RB1100AHx4 在 v7.1rc1 后全部由 bridge vlan 代替）。

属性	描述
vlan-mode (*check* | *disabled* | *fallback* | *secure*; 默认: **disabled**)	根据 VLAN 表修改 VLAN 查找机制，用于 ingress 流量，对应模式如下： • **disabled** – 忽略 VLAN 表，视作数据包没有做 vlan 标记。 • **fallback** – 默认模式，处理数据报时，VLAN 标签将不会出现在 VLAN 表中，视作数据包没有 VLAN 卷标。数据报的 vlan 标记会出现在 vlan 表,但接入的端口不匹配任何 vlan 表条目，也不会丢弃。 • **check** – 丢弃那些没有在 VLAN 列表的 vlan 标签包。VLAN 标签包出现在 vlan 列表,但进入端口无法匹配任何 VLAN 标签，进入条目将不会被丢弃。 • **secure** -丢弃那些没有在 VLAN 列表的 vlan 标签包。VLAN 标签包出现在 vlan 列表,但进入端口无法匹配任何 VLAN 标签，进入条目将会被丢弃。
vlan-header (*add-if-missing* | *always-strip* | *leave-as-is*; 默认: **leave-as-is**)	设置对 egress 流量的操作. • **leave-as-is –** 数据报在出端口保持不变 • **always-strip –** 如果 VLAN 包头在数据报出现，会被删除掉，即取消标签（access） • **add-if-missing –** 如果 VLAN 包头没有出现，会添加到数据报中，即打上标签（trunk）
default-vlan-id (*auto* | *integer: 0..4095*; 默认: **auto**)	添加指定 VLAN ID

QCA8337 和 **Atheros8327** 交换芯片上，默认 vlan-header=leave-as-is。交换芯片将通过使用 default-vlan-id 属性来确定哪些端口是 access 端口，default-vlan-id 只能在 access/hybrid 端口上使用。

Vlan Table(VLAN 列表)

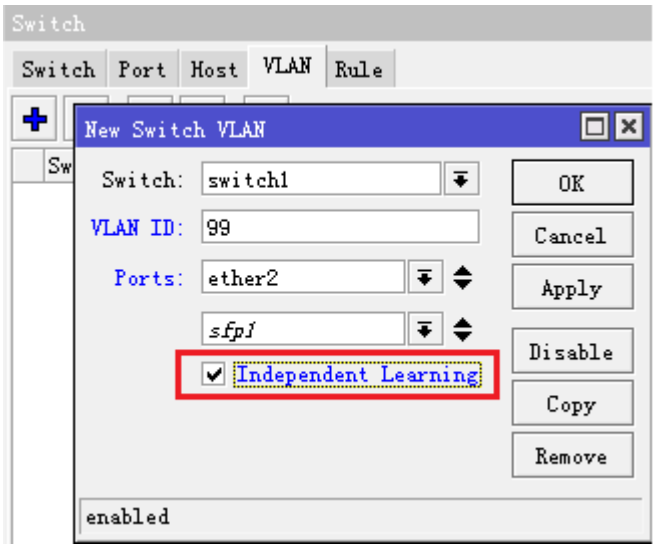
Vlan 清单为带有 802.1q 的数据报指定转发规则。这些规则优先级高于交换组配置的'master-port'属性。列表包含了相应的 vlan 标记 id 到交换组的一个或多个端口上，因此带有 vlan 标记的包，会根据相应的设置被调度到指定到交换组的一个或多个端口，即通过精确的逻辑控制带有 vlan 标记的包通过 vlan 模式进行端口间的调度。

修改每个交换组端口的 vlan 参数通过 **/interface ethernet switch port**

属性	描述
disabled (*no* | *yes*; 默认: **no**)	是否启用 switch VLAN 规则

属性	描述
independent-learning (no | *yes*; 默认: **yes**)	选择共享 vlan 学习(SVL)，还是独立 vlan 学习(IVL)。
ports (*name*; 默认: **none**)	对应 VLAN 的接口成员列表，该设置接受逗号分隔，例如 ports=ether1,ether2.
switch (*name*; 默认: **none**)	VLAN 规则要用于的 switch 名称。
vlan-id (*integer: 0..4095*; 默认:)	配置 vlan id

基于 Vlan 卷标 id 的主机 mac 转发会被动态或手动记录添加到 host 列表中。QCA8337 和 AR8327 交换芯片支持 IVL（Independent VLAN learning 独立 VLAN 学习），即允许同一主机 MAC 地址出现在多个 VLAN 中，在 VLAN 列表中可以选择打开



这里我们补充一个 VLAN 定义：

- **IVL:** Independent Vlan Learning 独立 Vlan 学习
- **SVL:** Shared Vlan Learning 共享式 Vlan 学习

即交换机内 MAC 表的两种方式，由 IEEE 802.1Q 定义。简单的理解，IVL 就是每个 Vlan 有一个 MAC 端口映射表，相同 MAC 可以出现在多个表里面；而 SVL 是在交换机内建一张大表，映射关系是 Mac-Vlan-端口，而一个 MAC 在表中只出现一次，只属于一个 Vlan。在实际网络生产环境中，支持 IVL 非常重要，因为可能一个主机要求同时被分配到多个 VLAN 中，如果不支持 IVL，同一主机 MAC 就无法在多个 VLAN 出现，在主机列表中只能出现在一个 VLAN 上。

Vlan 包头配置选项可以设置 VLAN 卷标模式在出端口(配置在 /interface ethernet switch port)，从 RouterOS V6 开始，该选项可以支持 QCA8337, AR8316, AR8327, AR8227 和 AR7240 交换芯片，并包含以下参数：

Rule Table（策略列表）

Rule 策略表，允许全线速处理二层、三层和四层的各种过滤、转发和 VLAN 标记，每条规则包含一个条件部分和执行部分，策略列表的策略类似于/ip firewall filter 一样的顺序执行。

IPv4 和 IPv6 指定条件时不能同时出现在相同规则中，由于策略表受到交换芯片硬件性能的限制，规则数量根据条件的不同，如 MAC 层、IPv4、IPv6 和四层传输等受到限制。例如 Atheros8316 交换芯片支持 8-32 条可用规则，而 Atheros8327/QCA8337 交换芯片支持 24-96 条，你可以通过/interface Ethernet switch rule print 命令查看最后规则是否出现”invalid”提示，表示规则不能加入交换芯片中，Rules 数量请参阅 3.9 章节

属性	描述
copy-to-cpu (*no* | *yes*; 默认: **no**)	是否发送数据到 switch-cpu 端口
mirror (*no* | *yes*; 默认: **no**)	是否发送一份拷贝数据到镜像 mirror-target 的端口
new-dst-ports (*name*; 默认: **none**)	修改指定的目的端口，允许多个端口，包括交换机的 CPU 端口。选择后，如果是空设置，代表丢弃数据包。当不使用此参数时，将接受数据
new-vlan-id (*integer: 0..4095*)	将 VLAN ID 更改为指定的值或添加一个新的 VLAN 标签 (此属性仅应用于 Atheros8316 交换芯片)
new-vlan-priority (*integer: 0..7*)	更改 VLAN 优先级字段
rate (*integer: 0..4294967295*)	匹配入口流量限制(每秒位数)，只能应用于前 32 个规则(此属性仅应用于 Atheros8327/QCA8337 交换芯片)
redirect-to-cpu (*no* | *yes*; 默认: **no**)	修改匹配数据目的端口到 switch-cpu 端口

同一规则中不能同时包含 IPv4 和 IPv6 的特定条件。由于 rule 策略表完全是在交换芯片中处理，所以规则的数量是有限的。根据在规则中使用的条件(MAC 层、IP 层、IPv6、传输层)的数量，Atheros8316 的规则数量可能从 8 到 32，Atheros8327/QCA8337 的规则数量可能从 24 到 96。你总是可以在修改你的规则集后执行/interface Ethernet switch rule print 命令，在列表末尾有“ 'invalid' ”的规则，这意味超过了交换芯片的容量。

Port isolation（端口隔离）

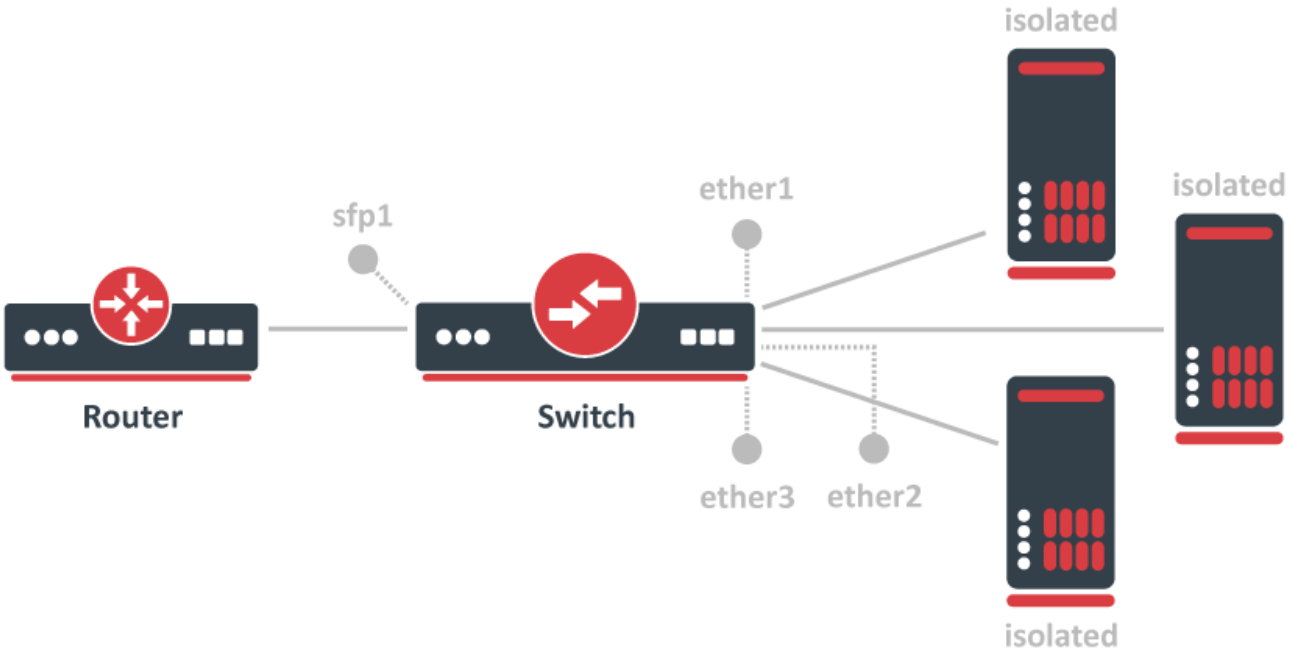
Port isolation 端口隔离提供了，对二层网络划分隔离的功能，通过同一交换机端口隔离来实现。从 RouterOS v6.43 开始，所有交换机芯片上都支持端口隔离功能。

属性	描述
forwarding-override (*interface*; 默认:)	指定进入当前接口的 ingress 流量转发到指定接口。可以使用“，”分隔多个接口。即允许流量转发到该端口，通常作为上行接口。

(R/M)STP 将只能在 PVLAN 设置中正常工作，(R/M)STP 不能在有多个隔离的交换组设置中正常工作，因为交换组无法正常接收到 BPDU，因此无法检测网络环路。

forwarding-override 仅对进入流量有影响。没有指定 forwarding-override 的交换机端口能够通过所有交换机端口发送数据包。

在某些情况下，需要将所有流量转发到上行端口，而所有其他端口彼此隔离。这种设置被称为私有 VLAN 配置，交换机将把所有以太网帧直接转发到上行端口，允许过滤不需要的包，并限制交换机端口之间其他端口访问。



首先创建 bridge1，并将指定以太网端口加入到 bridge1：

```
/interface bridge
add name=bridge1
/interface bridge port
add interface=sfp1 bridge=bridge1 hw=yes
add interface=ether1 bridge=bridge1 hw=yes
add interface=ether2 bridge=bridge1 hw=yes
add interface=ether3 bridge=bridge1 hw=yes
```

默认情况下，Bridge 的 rstp 协议是启用的。对于部分 RB 设备不支持 Hardware Offloading，可以导致无法支持硬件处理，请参阅 Bridge HW-Offloading 功能，根据型号的不同，选择关闭或启用 rstp 协议。

隔离交换机 ether1-ether3 端口，sfp1 上行口除外：

```
/interface ethernet switch port-isolation
set ether1 forwarding-override=sfp1
set ether2 forwarding-override=sfp1
set ether3 forwarding-override=sfp1
```

CPU Flow Control（CPU 流量控制）

所有的交换芯片都有一个特殊的端口，叫做 switchX-cpu，这是一个交换芯片通往 CPU 的接口，它的意思是将流量从交换芯片转发到 CPU，即无法被交换芯片处理的流量。默认情况下，交换机芯片会确保这个特殊的

CPU 通道不拥塞，当容量超过时，会发送暂停帧来确保端口不饱和，这个特性被称为 CPU 流量控制。如果没有这个特性，对于路由或管理流量至关重要的数据可能会被丢弃。

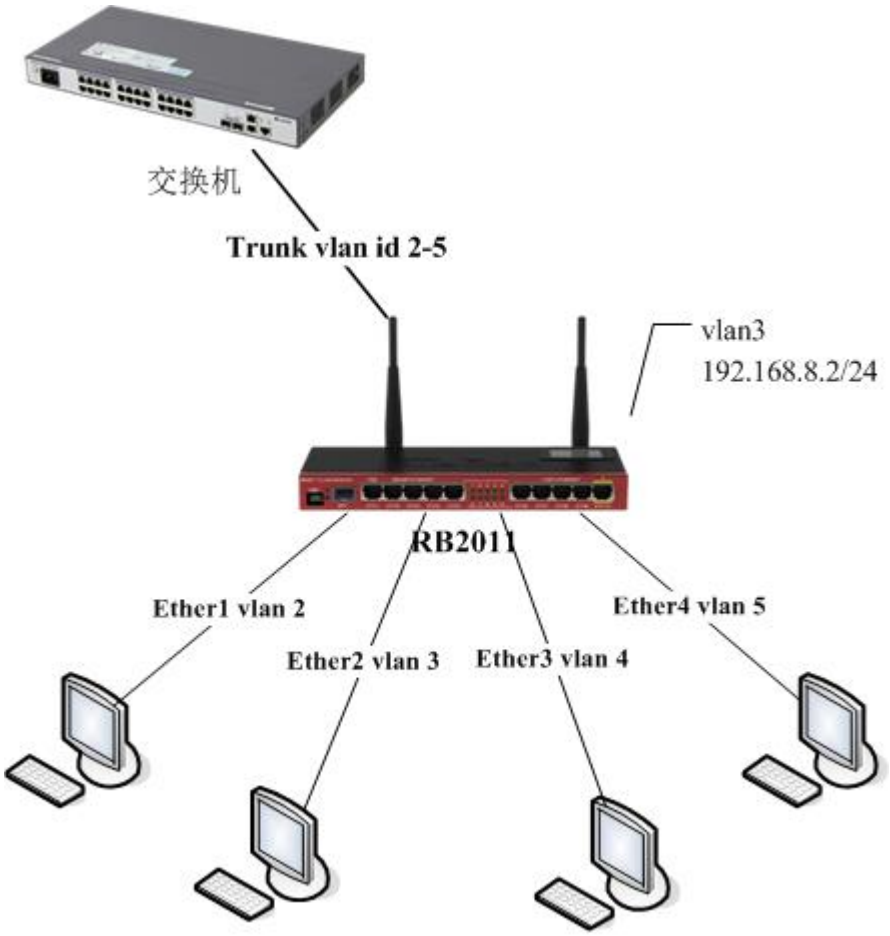
从 RouterOS v6.43 开始，在 Atheros8227, QCA8337, Atheros8327, Atheros7240 和 Atheros8316 设备上可以禁用 CPU Flow Control 功能，其他交换芯片默认开启此功能，不能更改。使用如下命令关闭 CPU 流量控制：

```
/interface ethernet switch set switch1 cpu-flow-control=no
```

Switch 的 VLAN 配置（Trunk 与 access）

RouterBOARD 设备采用 Atheros 交换芯片支持 802.1Q 协议加速，此功能在 RouterOS v6 开始支持，包括 **QCA8337, Atheros8316, Atheros8327, Atheros8227** 和 **Atheros7240**，基于 Atheros 交换芯片是将二层协议下发给交换芯片处理配置是在 switch 功能下。

下面我们以 RB2011iLS-IN 路由器为例，如何在 RouterOS 里基于交换芯片配置 VLAN 的 trunk 和 access，在这个实例中，sfp1 接口作为 trunk 口，上联一台管理交换机，ether1-ether4 作为 access 口



首先，在/interface bridge 创建 Bridge1 桥接端口，并将指定端口加入到 bridge1 中：

```
/interface bridge
add name=bridge1 protocol-mode=none
/interface bridge port
add bridge=bridge1 interface=ether1 hw=yes
```

```
add bridge=bridge1 interface=ether2 hw=yes
add bridge=bridge1 interface=ether3 hw=yes
add bridge=bridge1 interface=ether4 hw=yes
add bridge=bridge1 interface=sfp1 hw=yes
```

默认情况下，Bridge 的 rstp 协议是启用的。对于部分 RB 设备不支持 Hardware Offloading，可以导致无法支持硬件处理，请参阅 Bridge HW-Offloading 功能，根据型号的不同，选择启用或关闭 rstp 协议。

在 switch port 下配置端口 vlan

```
/interface ethernet switch port
set sfp1 vlan-header=add-if-missing vlan-mode=secure
set ether1 default-vlan-id=2 vlan-header=always-strip vlan-mode=secure
set ether2 default-vlan-id=3 vlan-header=always-strip vlan-mode=secure
set ether3 default-vlan-id=4 vlan-header=always-strip vlan-mode=secure
set ether4 default-vlan-id=5 vlan-header=always-strip vlan-mode=secure
```

- "vlan-mode=secure"将接口加入 VLAN 列表，
- "vlan-header=always-strip" 用于 access 口，即去掉数据帧中的 VLAN 包头。
- "vlan-header=add-if-missing" 对应 trunk 口，即添加 VLAN 包头到为标记帧中。
- "default-vlan-id" 为进入到 access 口的 VLAN，将 sfp1 配置为 trunk 接口（打上卷标），其他 ether1-ether4 为 access 口（去标签）

添加 VLAN 表的规则，即指定包含 VLAN ID 的数据帧在那些接口通信，并启用 IVL

```
/interface ethernet switch vlan
add ports=sfp1,ether1 switch=switch1 vlan-id=2 independent-learning=yes
add ports=sfp1,ether2 switch=switch1 vlan-id=3 independent-learning=yes
add ports=sfp1,ether3 switch=switch1 vlan-id=4 independent-learning=yes
add ports=sfp1,ether4 switch=switch1 vlan-id=5 independent-learning=yes
```

配置 VLAN 管理 IP

这里我们将介绍如何配置 VLAN 的三层管理 IP 地址，管理 IP 地址将通过 trunk 口到达路由器，并指定 VLAN ID 3 为管理接口

配置交换芯片与 CPU 连接，在 RB2011 有两组交换芯片，switch1 和 switch2，switch1 管理 sfp1，ether1-ether5，switch2 管理 ether6-ether10。因此这里需要注意，我们配置的是 sfp1 和 ether1-ether4，所以选择的是 switch1_cpu，设置 "vlan-header=leave-as-is" 原因是管理 IP 数据到 CPU 保持原有的标记（tag）模式。

```
/interface ethernet switch port
set switch1_cpu vlan-mode=secure vlan-header=leave-as-is
```

修改 VLAN 列表条目，允许 VLAN ID 3 的数据能在 sfp1，ether2 和 switch1_cpu，首先我们查看 VLAN 列表配置

```
[admin@MikroTik-J] /interface ethernet switch vlan> print
Flags: X - disabled, I - invalid
```

```
#    SWITCH                                VLAN-ID     PORTS
0    switch1                                 2         sfp1
                                                       ether1
1    switch1                                 3         sfp1
                                                       ether2
2    switch1                                 4         sfp1
                                                       ether3
3    switch1                                 5         sfp1
                                                       ether4
```

找到 VLAN id 3 的对应序号为#1，将 VLAN id 3 的配置加入 switch1-cpu，注意这里用 set 命令是设置，而非把 switch1-cpu 添加入规则，因此我们的命令是所有的接口都要设置一次 ports=sfp1,ether2,switch1-cpu 如下，当然这 winbox 下可以直接通过图像接口添加。

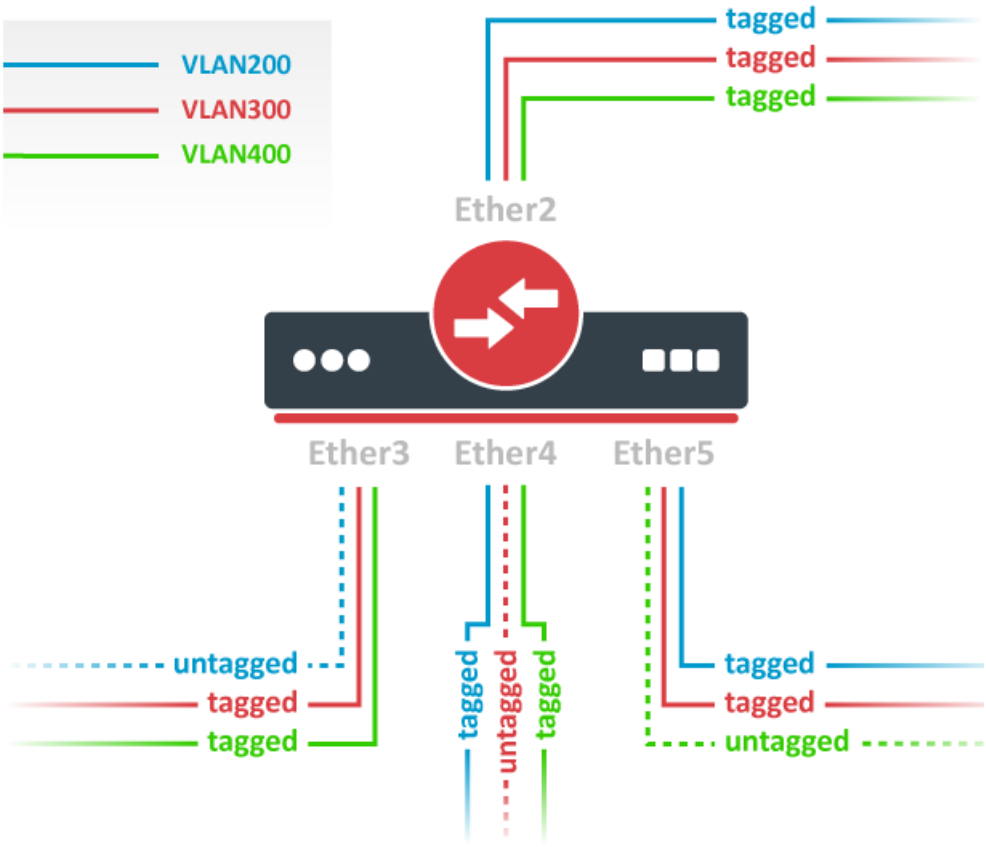
```
[admin@MikroTik] /interface ethernet switch vlan> set 1 ports=sfp1,ether2,switch1-cpu
[admin@MikroTik-J] /interface ethernet switch vlan> print
Flags: X - disabled, I - invalid
 #    SWITCH                                VLAN-ID     PORTS
0    switch1                                 2         sfp1
                                                       ether1
1    switch1                                 3         sfp1
                                                       ether2
                                                       switch1-cpu
2    switch1                                 4         sfp1
                                                       ether3
3    switch1                                 5         sfp1
                                                       ether4
```

以上 VLAN 交换芯片菜单配置完成后，进入 interface vlan 菜单，添加三层接口的 VLAN，这里需注意 master-port 从 switch-cpu 接收所有的数据，因此三层 VLAN 接口配置必须是 master-port，即 master-port 是 sfp1，配置管理 IP 地址 192.168.8.2/24 到 vlan3 接口上。

```
/interface vlan
add name=vlan3 vlan-id=3 interface=sfp1
/ip address
add address=192.168.8.2/24 interface=vlan3 network=192.168.88.0
```

Switch 的 VLAN 配置（Trunk 与 Hybrid）

VLAN 混杂端口（Hybrid），允许转发 tagged 和 untagged 流量，但支持此配置的 RB 设备，仅采用 **QCA8337 和 Atheros8327** 的交换芯片。在下列配置 ether2 为 trunk 端口，ether3-ether5 为 hybrid 端口



首先，在/interface bridge 创建 Bridge1 桥接端口，并将指定端口加入到 bridge1 中：

```
/interface bridge
add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether2 hw=yes
add bridge=bridge1 interface=ether3 hw=yes
add bridge=bridge1 interface=ether4 hw=yes
add bridge=bridge1 interface=ether5 hw=yes
```

默认情况下，Bridge 的 rstp 协议是启用的。对于部分 RB 设备不支持 Hardware Offloading，可以导致无法支持硬件处理，请参阅 Bridge HW-Offloading 功能，根据型号的不同，选择启用或关闭 rstp 协议。

这里的添加 VLAN 规则和之前实例不同，允许端口通过指定的 VLAN id 帧。

```
/interface ethernet switch vlan
add ports=ether2,ether3,ether4,ether5 switch=switch1 vlan-id=200
add ports=ether2,ether3,ether4,ether5 switch=switch1 vlan-id=300
add ports=ether2,ether3,ether4,ether5 switch=switch1 vlan-id=400
```

在交换机端口菜单中设置所有端口的 vlan-mode，并在规划的混合端口上设置 default-vlan-id:

```
/interface ethernet switch port
set ether2 vlan-mode=secure vlan-header=leave-as-is
set ether3 vlan-mode=secure vlan-header=leave-as-is default-vlan-id=200
set ether4 vlan-mode=secure vlan-header=leave-as-is default-vlan-id=300
set ether5 vlan-mode=secure vlan-header=leave-as-is default-vlan-id=400
```

在 QCA8337 和 Atheros8327 芯片中，使用 vlan-mode=secure 时，忽略 switch port 的 vlan-header 选项。VLAN 列表处理所有出口 tagged/untagged，并在所有端口上作为 VLAN-header=leave-as-is 工作。这意味着进入和离开都有标签，只有 default-vlan-id 帧在出口端口是 untagged 不带标签的。

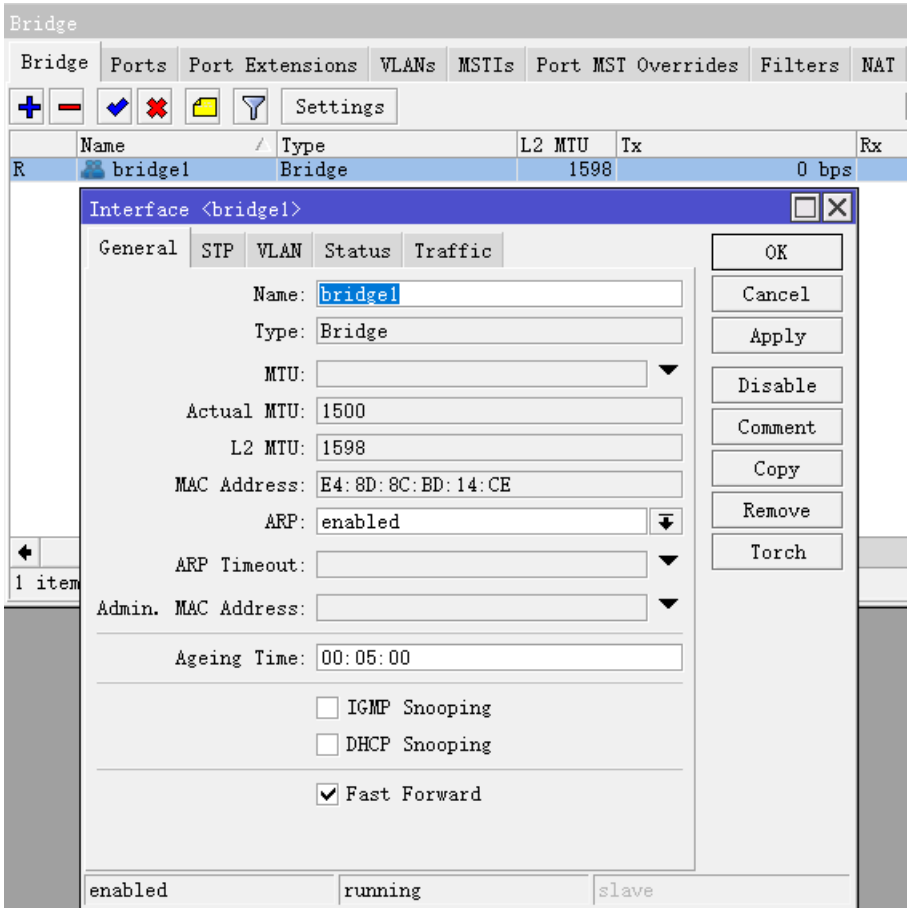
3.11 RouterBOARD 端口镜像

v3.26 后开始支持在 switch 下配置端口镜像，让交换芯片“sniff”在一个端口上所有的传输流量（mirror-source），并拷贝一份数据发送到另一个端口（mirror-target）。

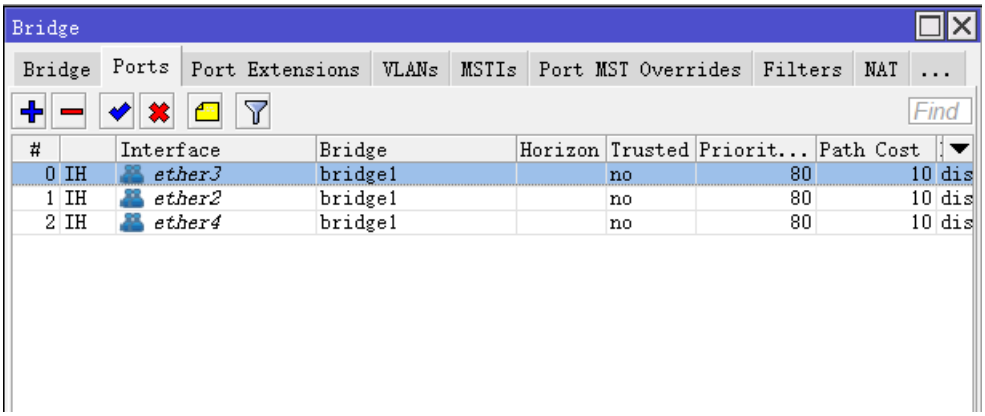
这个功能可以轻松的配置一台网络监测设备接收所有指定端口的传输流量。注意：mirror-source 和 mirror-target 端口必须属于同一交换配置下。（查看那个端口属于哪个交换配置在“/interface Ethernet switch port”目录下），同样 mirror-target 能设定特殊的“CPU”值，意思是“sniffed”数据报被发送至 CPU 端口。

端口镜像配置

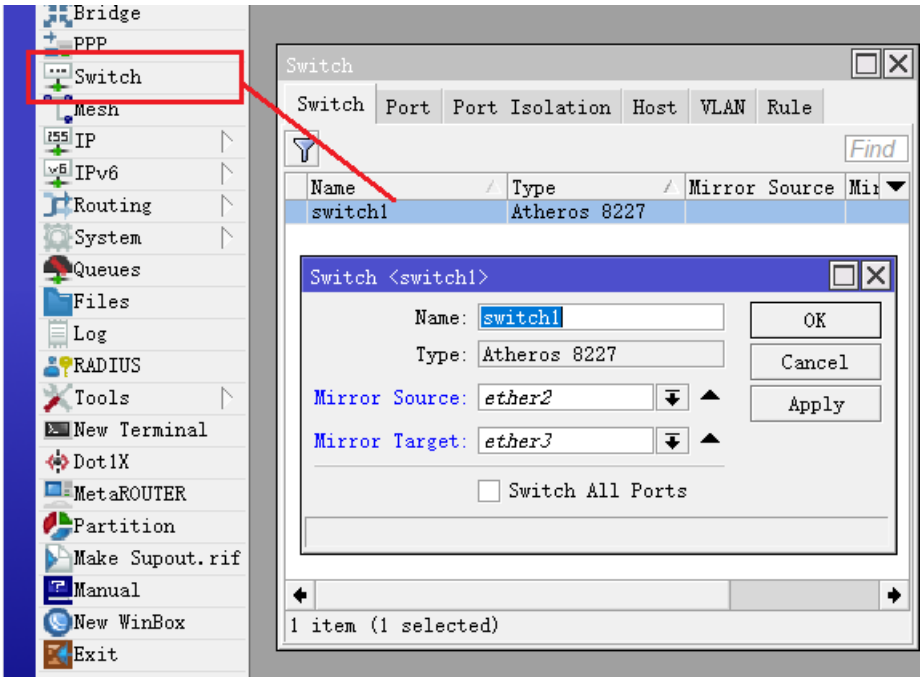
首先，进入 bridge 功能，并创建一个 bridge1 接口，



将 ether2，ether3 和 ether3 加入一个 bridge，



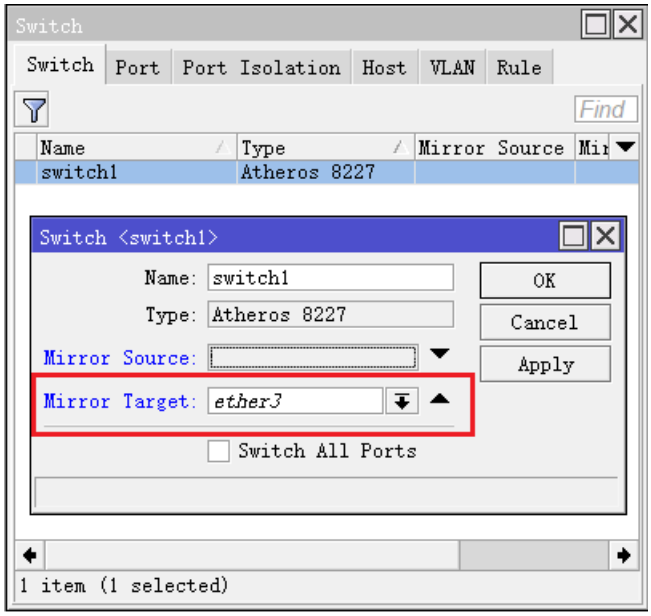
需将 ether2 的数据拷贝并发送到 ether3，即镜像 ether2 到 ether3，进入 switch 选项，打开 switch1，



mirror source 为镜像源端口 ether2，mirror target 为镜像目标端口 ether3

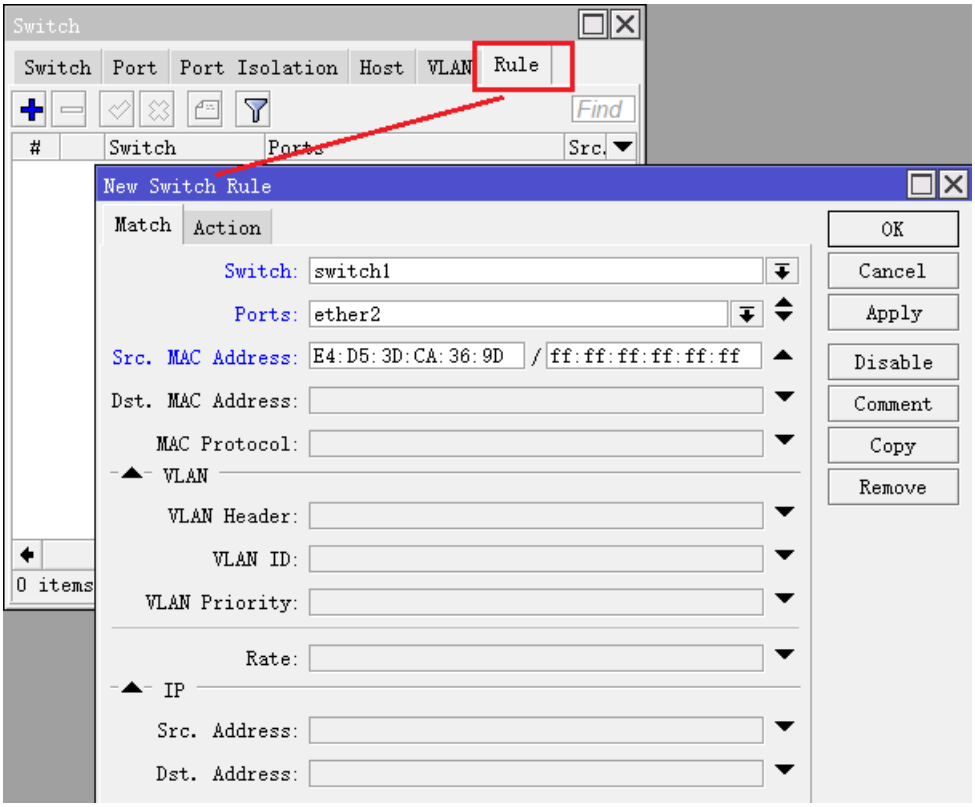
Rule 指定镜像内容

在 switch rule 下，可以定义需要镜像的内容到指定端口，在定义之前，在 switch 下仅设置 Mirror Target 接口为 ether3，源接口不再指定：



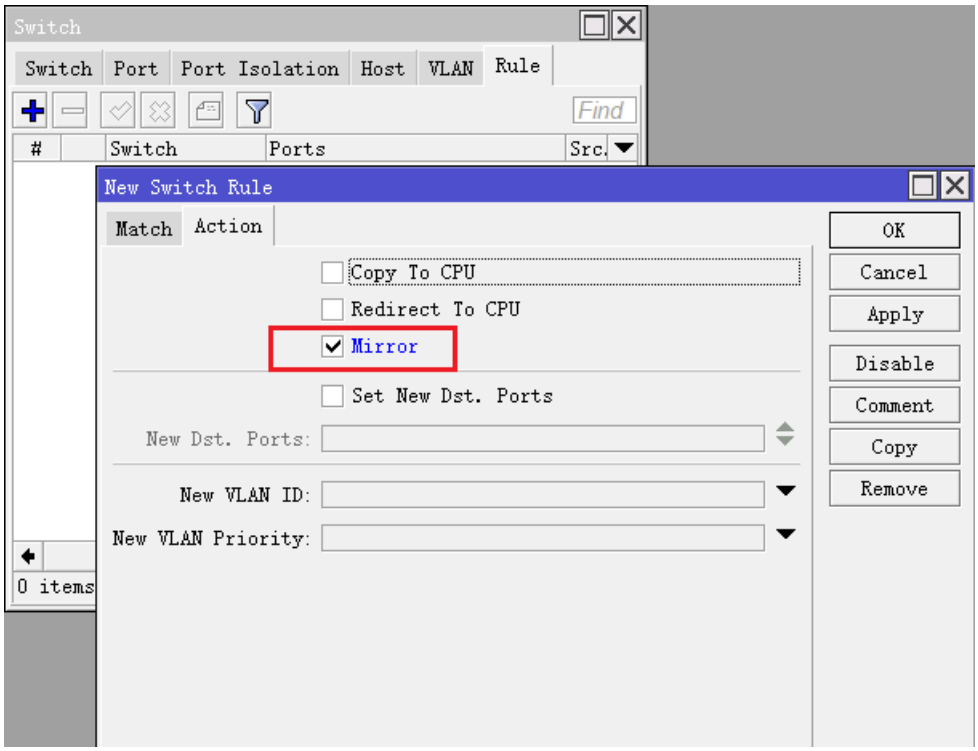
1、镜像 MAC

在 switch 菜单下，我们可以看到 rule 规则菜单，选择后点击添加



在规则中，将 ether1 接口上源 MAC 地址为 E4:D5:3D:CA:36:9D

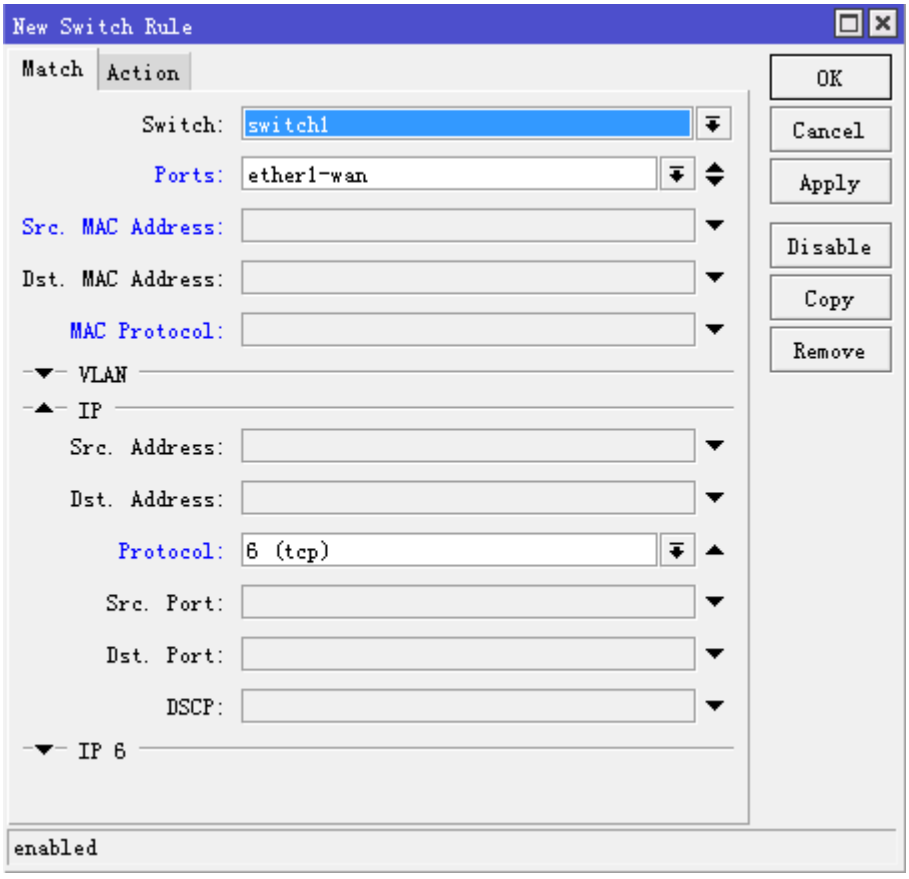
选择 action 菜单下的 Mirror，即镜像到之前我们定义的 Mirror Target 接口 ether3 上



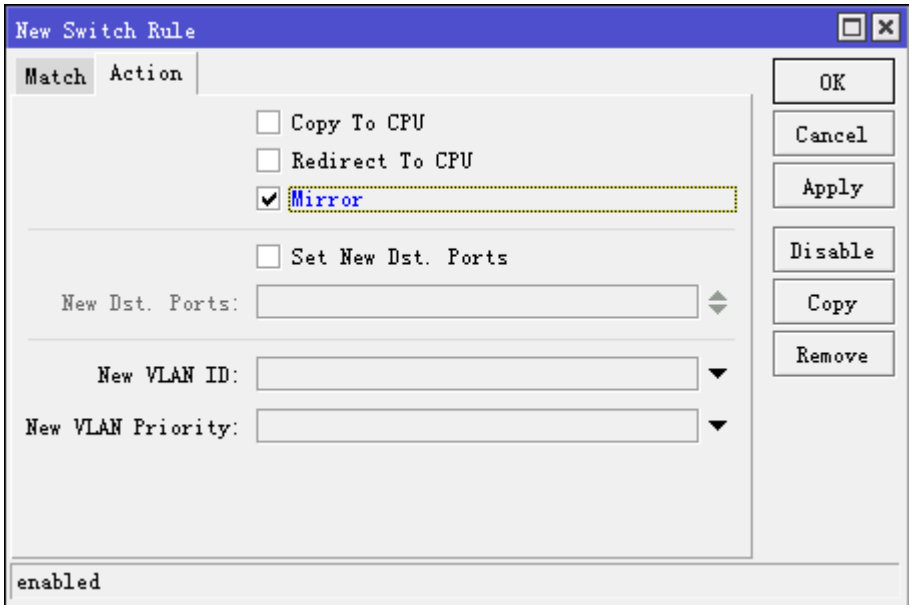
这样源 MAC 为 E4:D5:3D:CA:36:9D 的所有数据被镜像到 ether3 接口，剩下的就是你通过网络设备接收分析数据

2、指定 TCP 协议镜像

镜像 TCP 协议到 ether3 上，之前的配置一样，直接添加新的规则，配置如下



执行 action 为 Mirror：



3.12 CRS 交换机介绍

CRS（Cloud Router Switch）系列是 MikroTik 开放的交换机，CRS 交换机设计为二层交换网络，当前有 CRS1xx/2xx/3xx 系列产品，下面是一些包括 VLAN、二层网络管理。

提示：但 CRS3xx 系列交换机在 VLAN 配置方面很大区别于 CRS1xx/2xx，关于 CRS3xx 系列交换机可以参考单独的手册《CRS3 系列交换机配置手册》

术语和解释

- CVID –Customer VLAN id：用户使用内层的 VLAN tag
- SVID - Service VLAN id：服务商使用外层的 VLAN tag
- IVL - Independent VLAN learning – 独立 vlan 学习，交换机在学习 MAC 地址并建立 MAC 地址表的过程中同时附加 VLAN ID，同一个 MAC 地址可以出现在不同的 VLAN 中，这样的方式也可以理解为每个 VLAN 都有自己独立的 MAC 地址表
- SVL – Shared VLAN Learning - 共享 VLAN 学习，交换机在学习 MAC 地址并建立 MAC 地址表的过程中并不附加 VLAN ID，或者说它的 MAC 地址表是为所有 VLAN 共享使用
- TPID - Tag Protocol Identifier 标签协议标识，VLAN Tag 中的一个字段，IEEE 802.1q 协议规定该字段的取值为 0x8100
- PCP – Priority Code Point（优先级代码）：PCP 字段(3Bit)定义了 8 种传输类型，在 802.1p 中定义了 PCP
- DEI - Drop Eligible Indicator 对帧进行队列管理，当 DEI 为 true 帧会优先丢弃，DEI 为 false 帧会与正常报文一起通过
- DSCP - Differentiated services Code Point 差分服务代码点
- Drop precedence – 丢弃优先级，交换机在接收报文的时候就会给报文分配丢弃级别。交换机在对报文进行处理时可以修改报文的丢弃级别，取值为 0、1 或 2。

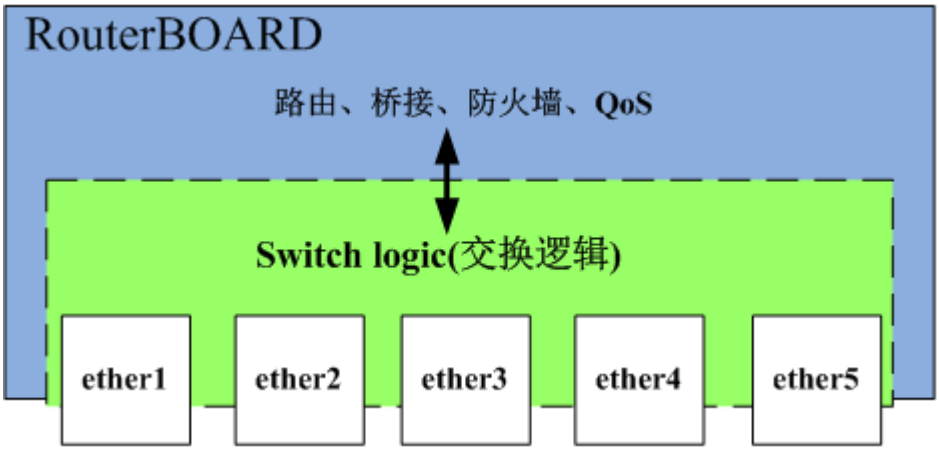
802.1Q VLAN Tag 的 PCP 字段(3Bit)定义了以下 8 种 Traffic types。

Traffic Types	字母缩写	优先级	协议举例	业务特征
Network Control	NC	7	BGP, PIM, SNMP	网络维护与管理报文的可靠传输，要求低丢包率
InternetWork Control	IC	6	STP, OSPF, RIP	大型网络中区分于普通流量的网络协议控制报文
Voice	VO	5	SIP, MGCP	适用于语音业务，一般要求时延小于 10 ms
Video	VI	4	RTP	适用于视频业务，一般要求时延小于 100 ms
Critical Applications	CA	3	NFS, SMB, RPC	适用于要求确保最小带宽的业务
Excellent Effort	EE	2	SQL	用于一般的信息组织向最重要的客户发送信息
Best Effort	BE	0(default)	HTTP, IM, X11	缺省业务类型，只要求"尽力而为"的服务质量
Background	BK	1	FTP, SMTP	适用于不影响用户或关键应用的批量传输业务

端口交换（Port Switching）

类似于其他 RouterBOARD，CRS 各个以太网端口数据交换允许在相同交换组中进行线速转发，就像一台以太网交换机。这个功能设置同样在 interface ethernet 菜单下的"master-port" ，所有在 master-port 知道的接口将独立于 RouterOS，直接在交换芯片处理

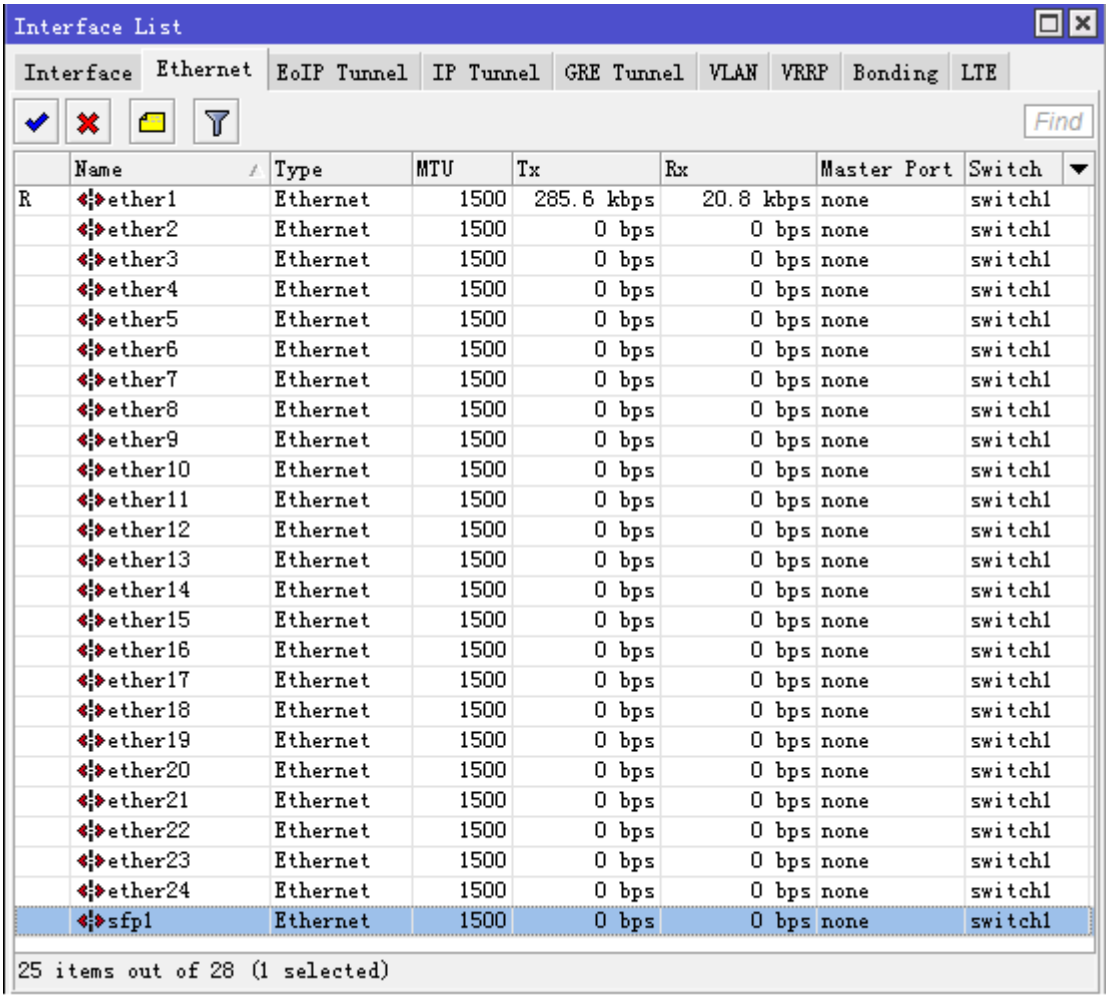
下面是 RouterBOARD 5 口交换芯片 Here is a general diagram of RouterBoard with a five port switch chip:



这里能看到，当一个端口接收到一个数据报，首先会传递给交换逻辑处理。交换逻辑决定数据报去哪一个端口。传递数据报是向 RouterOS，同样也可以被称为传递给交换芯片（cpu port）。即交换转发数据报给 cpu port，数据报开始被 RouterOS 某个接口处理进入的数据。当数据报不必通过交换逻辑发送到 cpu port 处理，也就不占用任何的 CPU 时钟周期，这样所有帧转发实现线速转发。

CRS 系列交换机支持多 Master-port 配置，且没有端口选择限制

例如，CRS125 交换机有 24 个以太网机口和 1 个 SFP 接口，默认情况下 Master-port 没有配置：



通常其他 RouterBOARD 只能配置一个交换组，CRS 允许配置多个交换组，下面是 3 个交换组配置情况

Interface List

Interface | Ethernet | EoIP Tunnel | IP Tunnel | GRE Tunnel | VLAN | VRRP | Bonding | LTE

Find

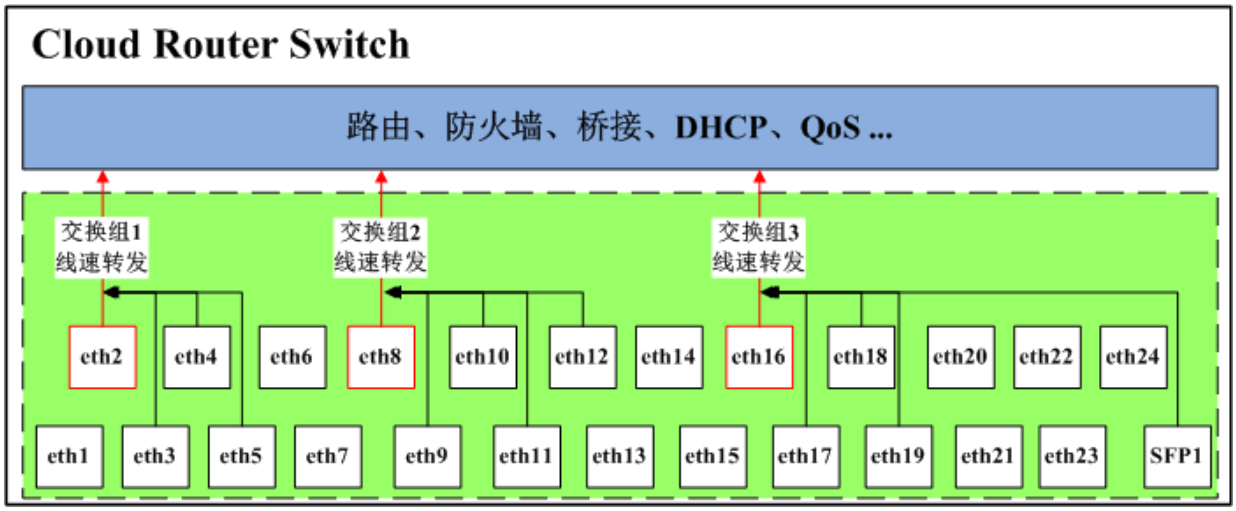
	Name	Type	MTU	Tx	Rx	Master Port	Switch
R	ether1	Ethernet	1500	268.1 kbps	14.2 kbps	none	switch1
	ether2	Ethernet	1500	0 bps	0 bps	none	switch1
S	ether3	Ethernet	1500	0 bps	0 bps	ether2	switch1
S	ether4	Ethernet	1500	0 bps	0 bps	ether2	switch1
S	ether5	Ethernet	1500	0 bps	0 bps	ether2	switch1
	ether6	Ethernet	1500	0 bps	0 bps	none	switch1
	ether7	Ethernet	1500	0 bps	0 bps	none	switch1
	ether8	Ethernet	1500	0 bps	0 bps	none	switch1
S	ether9	Ethernet	1500	0 bps	0 bps	ether8	switch1
S	ether10	Ethernet	1500	0 bps	0 bps	ether8	switch1
S	ether11	Ethernet	1500	0 bps	0 bps	ether8	switch1
S	ether12	Ethernet	1500	0 bps	0 bps	ether8	switch1
	ether13	Ethernet	1500	0 bps	0 bps	none	switch1
	ether14	Ethernet	1500	0 bps	0 bps	none	switch1
	ether15	Ethernet	1500	0 bps	0 bps	none	switch1
	ether16	Ethernet	1500	0 bps	0 bps	none	switch1
S	ether17	Ethernet	1500	0 bps	0 bps	ether16	switch1
S	ether18	Ethernet	1500	0 bps	0 bps	ether16	switch1
S	ether19	Ethernet	1500	0 bps	0 bps	ether16	switch1
	ether20	Ethernet	1500	0 bps	0 bps	none	switch1
	ether21	Ethernet	1500	0 bps	0 bps	none	switch1
	ether22	Ethernet	1500	0 bps	0 bps	none	switch1
	ether23	Ethernet	1500	0 bps	0 bps	none	switch1
	ether24	Ethernet	1500	0 bps	0 bps	none	switch1
S	sfp1	Ethernet	1500	0 bps	0 bps	ether16	switch1

25 items out of 28 (1 selected)

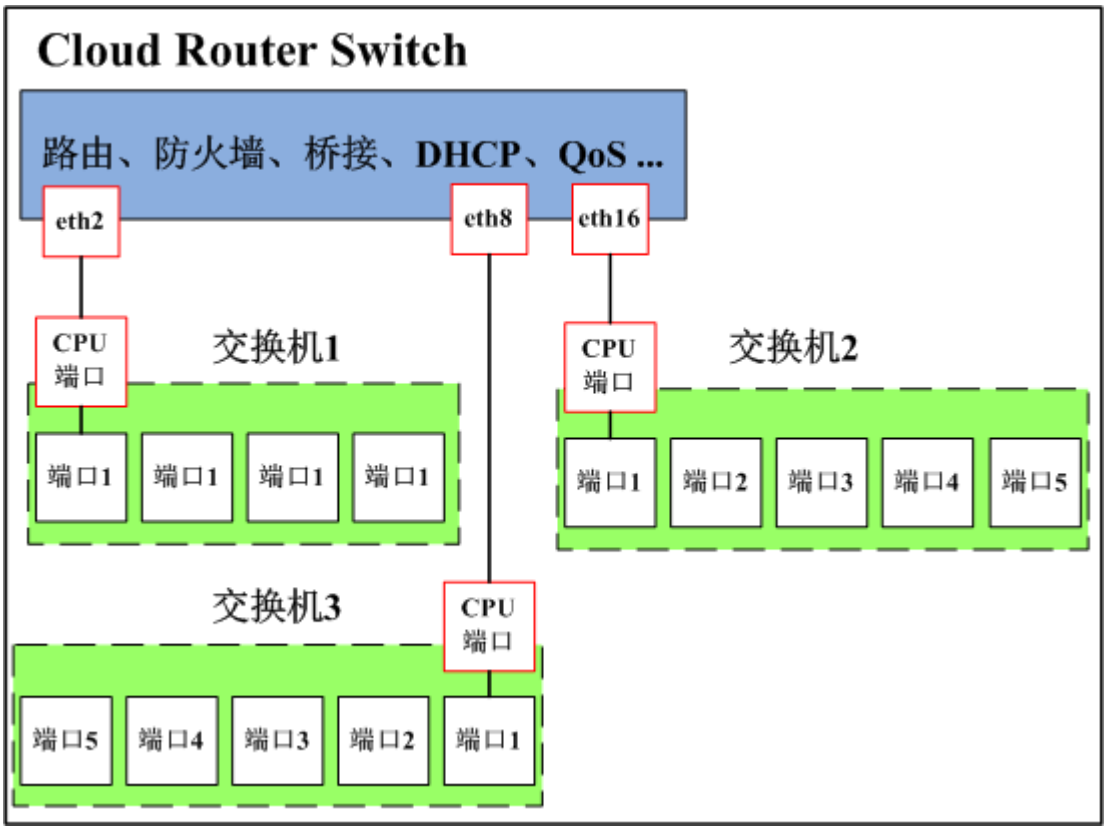
- 交换组 1：ether2-ether5
- 交换组 2：ether8-ether12
- 交换组 3：ether16-ether19，sfp1

这样就划分了三组交换，剩下的接口仍然作为路由口使用，ether2 作为交换组 1 的 Master-port，同样 ether8 和 ether16 分别作为其他 2 个组的 Master-port

之前我们讲到当一个接口被定义为 Master-port，就直接与 CPU 相连，现在我们定义该 3 个 Master-port，即他们都同时连接到了 CPU



实质上，这个配置将 CRS 划分为了 3 个交换机，通过 3 个 Master-port 直接连接交换机 CPU：

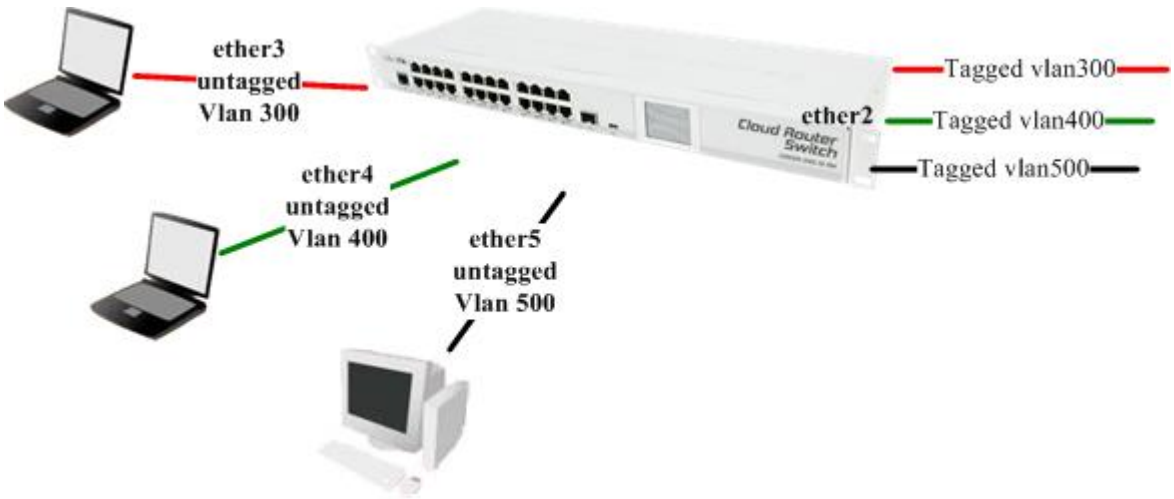


CRS 交换配置

Cloud Router Switch 基本交换配置，关于 CRS 路由交换功能的配置介绍都是基于 RouterOS v6.12，所以在参考此文档时，请将 CRS 路由交换设备升级到 v6.12 或以上

VLAN

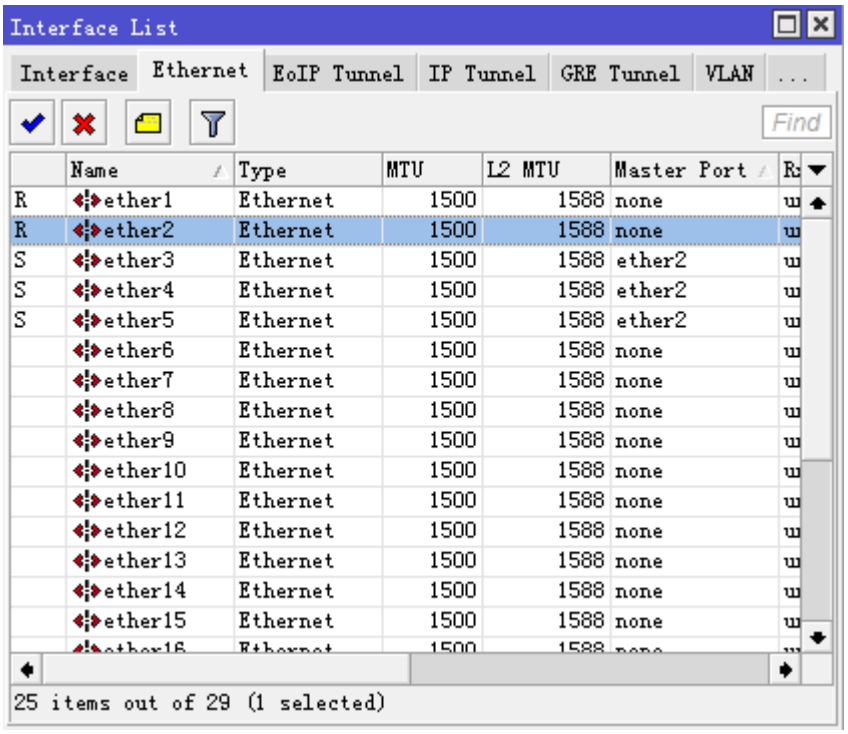
基于端口 VLAN



如上图，是一个简单的交换机 VLAN 实例，ether2 为 trunk 口（或打标签口 tagged），ether3、4、5 分别是 access 口（去标签口 untagged）

步骤 1、命令操作如下：

创建交换端口组，和其他 RouterBOARD 交换口配置一样，将 ether3、4、5 的 master-port 设置为 ether2



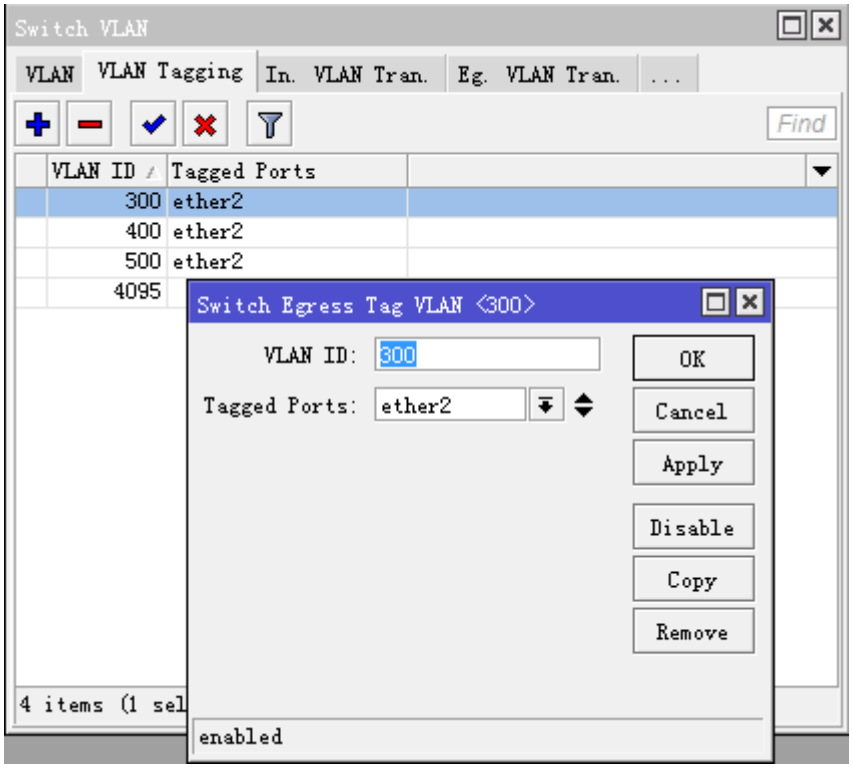
注意：以下配置是在 v6.41 以前，之后通过 bridge 创建交换端口，具体参见 14.7 和 14.8 章节

```
/interface ethernet
set ether3 master-port=ether2
set ether4 master-port=ether2
set ether5 master-port=ether2
```

步骤 2、将 ether2 创建 vlan trunk 口，进入 switch egress-vlan 把 VLAN 200、VLAN 300 和 VLAN 400 打标记到 ether2 口，

```
/interface ethernet switch egress-vlan-tag
add tagged-ports=ether2 vlan-id=300
add tagged-ports=ether2 vlan-id=400
add tagged-ports=ether2 vlan-id=500
```

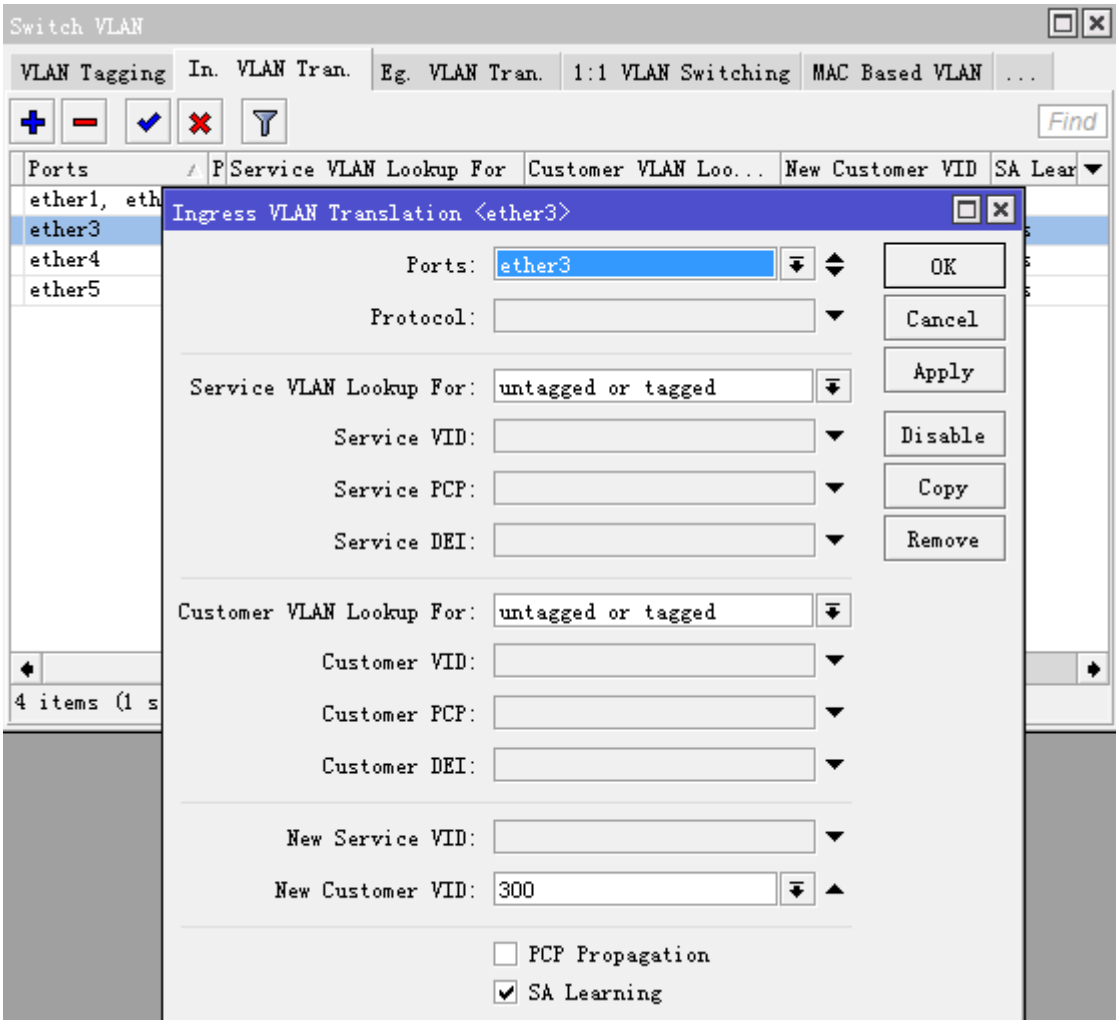
Winbox 中操作路径名有所不通，是在 switch vlan 中的 vlan tagging 下



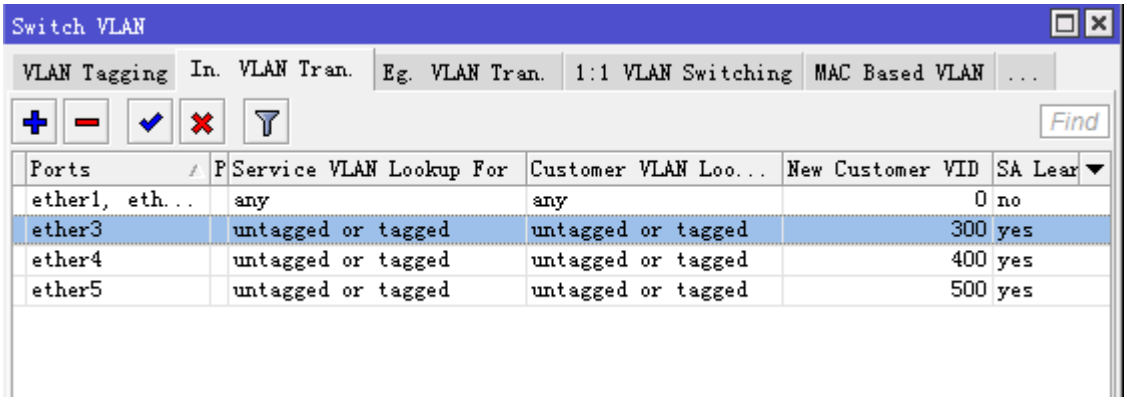
步骤 3、下面是添加 ether3、4、5 为 acess 口（取消卷标），将对应 vlan 配置到相应端口

```
/interface ethernet switch ingress-vlan-translation
add ports=ether3 new-customer-vid=300 sa-learning=yes
add ports=ether4 new-customer-vid=400 sa-learning=yes
add ports=ether5 new-customer-vid=500 sa-learning=yes
```

winbox 中设置 ether3 的 vlan 为 300：



Winbox 中配置完成后：



特别注意：当你在使用 winbox 或三层网络连接管理配置 CRS 时，且连接口配置为是 access 口，对应的 vlan tagging 必须配添加，否则导致该 access 口无法通过 winbox 或三层网络连接访问（vlan1 除外）。需谨慎操作，所以下面第 2 步首先配置 vlan tagging（trunk 口配置）。

CRS Bonding 硬件加速

从RouterOS v6.42所有的CRS3xx系列的交换功能支持Bonding接口的硬件加速,但仅支持802.3ad 和balance-xor 两种模式，其他 Bonding 模式只能使用 CPU 处理。其他相关的 Bonding 设置可以参考第十六章节。

提示：CRS3xx 系列交换机介绍单独整理了一份教程，可以下载 **MikroTik CRS3xx 系列操作配置手册**

下面事例创建一个硬件加速的 bonding 接口，当 802.3ad 模式选择，将支持 LACP (Link Aggregation Control Protocol)

```
/interface bonding
add mode=802.3ad name=bond1 slaves=ether1,ether2
```

Bonding 接口可以与其他接口一起添加到 bridge 中：

```
/interface bridge
add name=bridge
/interface bridge port
add bridge=bridge interface=bond1 hw=yes
add bridge=bridge interface=ether3 hw=yes
add bridge=bridge interface=ether4 hw=yes
```

注意：不允许将已经添加到 bonding 的接口配置到 bridge port 下，RouterOS 会提示无法添加接口，bonding 和 bridge 的接口配置下网络端口无法重复添加

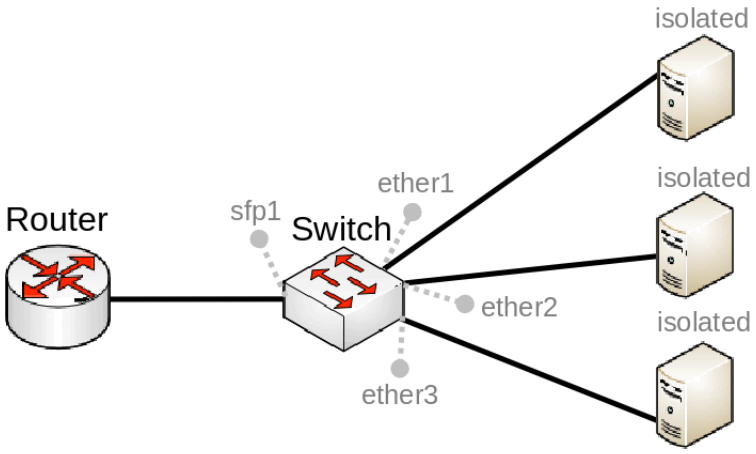
Bonding 接口配置完成后，确保硬件加速开启，在命令行下查看是 HW 为“yes”，Winbox 下查看 bridge port 下端口前缀为“H”

```
/interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic, H - hw-offload
 #      INTERFACE                  BRIDGE                    HW
 0    H bond1                    bridge                   yes
 1    H ether3                   bridge                   yes
 2    H ether4                   bridge                   yes
```

注意：集成的交换芯片会使用 Layer2+Layer3+Layer4 的传输数据做为哈希算法策略,如果手动改变该哈希策略，bonding 将失效。

CRS Port isolation（端口隔离）

从 RouterOS v6.43 可以配置端口隔离，即交换组下端口只能与 uplink 接口通讯，其他端口直接被隔离，Switch 下会将所有以太网帧转发到 uplink 端口,即转发到上联的路由器，如下图 sfp1 作为 uplink 端口连接路由器，ether1、ether2 和 ether3 配置端口隔离



配置交换端口的隔离，首先将指定的以太网端口配置交换组中

```
/interface bridge
add name=bridge1 protocol-mode=none
/interface bridge port
add interface=sfp1 bridge=bridge1 hw=yes
add interface=ether1 bridge=bridge1 hw=yes
add interface=ether2 bridge=bridge1 hw=yes
add interface=ether3 bridge=bridge1 hw=yes
```

进入 switch port-isolation 下配置所有端口转发到 sfp1：

```
/interface ethernet switch port-isolation
set ether1 forwarding-override=sfp1
set ether2 forwarding-override=sfp1
set ether3 forwarding-override=sfp1
```

注意： 在同一个交换芯片下，允许配置多个 uplink 端口，指定多个 uplink 端口在命令行下通过逗号隔离

3.13 RouterBOARD LCD 触摸屏

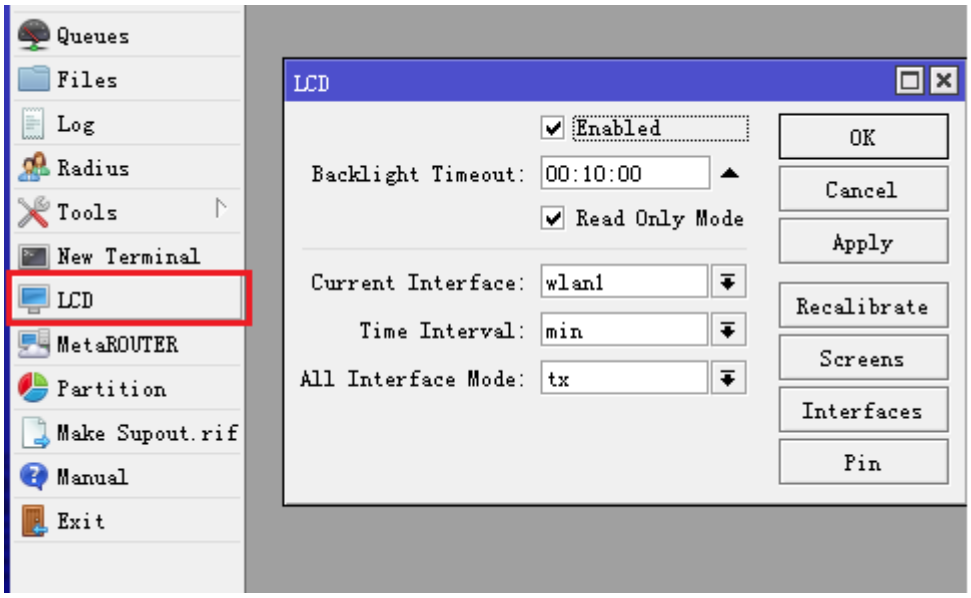
RouterBOARD 2011U 系列和 CCR 系列室内机型都集成了一块 LCD 触摸屏，方便快速查看设备状态和简单配置情况。触摸屏需要按压屏幕表面才能回馈操作。因此较轻或者快速短暂按压可能不能有回馈（这触摸屏看来也是最便宜的那种，毕竟不是专业做手机的），官方建议如果用手操作比较吃力的话，建议用户笔一类的介质进行操作，囧！

操作路径: /lcd

LCD 触摸屏能进入/lcd 菜单下进行配置

属性	描述
enabled (yes | no; 默认: yes)	触摸屏背光开过
backlight-timeout (时间周期: 5m..2h | never; 默认: 30m)	自动关闭 LCD 触摸屏时间

read-only-mode (yes | no; 默认: yes)	启用或禁用只读模式
current-interface (字符; 默认:空)	网络接口上哪一个状态将首先显示
time-interval (min | hour | daily | weekly; 默认: min)	在指定时间周期里当前接口的统计状态显示在屏幕上
all-interface-mode (rx | tx; 默认: tx)	显示所有接口的统计状态



LCD 触摸屏校准

在第一次使用 LCD 触摸屏前，需要做一次校准。在第一次成功校准后，信息会存储到路由器，如果没有做校准，触摸屏叫自动校准。

在校准和重新校准的过程中，你必须在屏幕上触摸 4 次（屏幕上会自动出现 **X**，跟着显示点击），前三次触摸用于计算校准变量，第四次触摸被用于测试是否校准成功。如果校准没有成功，校准变量将不会保存，并再次触摸 4 此。最后会显示一个校准结果。

LCD 截屏

LCD 菜单下支持 LCD 截屏功能，能截取当前 LCD 屏幕的图片，截屏后能创建一个 BMP 格式的图片文件，截取文件会放在 file 目录下，如果截屏没有设置文件名将不会被保存。如下面截屏操作：

```
[admin@MikroTik]  /lcd take-screenshot file-name=screen-1
Screenshot taken
[admin@MikroTik] >
```

LCD PIN 密码

操作路径: /lcd pin

PIN 密码是用于 LCD 操作的安全保护，即当 Read-Only 模式为启用情况下，我们可以通过触摸屏命令路由器重启或添加 IP 等操作，当你通过触摸屏执行一条指令时，会要求你输入 PIN 密码。默认的 PIN 密码是 1234

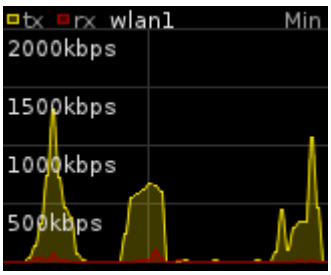
如下面图，我们可以通过触摸屏重启和复位配置，当做这些操作时我们就需要输入 PIN 密码，已确认管理能有权利执行。

熟悉	描述
pin-number (数字; 默认: 1234)	PIN 密码
hide-pin-number (yes | no; 默认: no)	是否将 LCD 触摸屏输入的密码隐藏

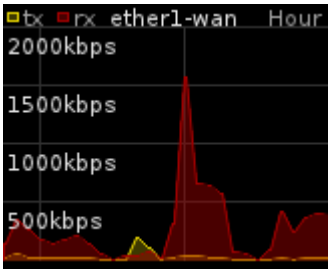
LCD 流量图

可以查看单网卡的 RX 和 TX 流量图，流量从屏幕的右方向左方更新。在触摸屏的右上角，显示时间间隔，下面可以设置时间间隔的长度

- **Min (分钟)** – 以分钟显示流量，单位为秒。垂直分割线用于分割前 30 秒时间，垂直分割线时间区域表示为从右到左前 30 秒，后 24 秒

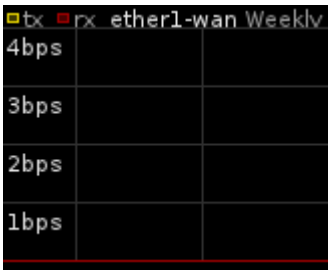


- **Hour** – 以小时显示流量。单位为每 5 分钟。垂直分割线为 1 小时，垂直分割线时间区域分三段，表示为从右到左为前 60 钟，中间 60 分钟，后 15 分钟



- **Daily –** 以天为显示，单位为小时，垂直分割线为 1 天。垂直分割线时间区域分三段，表示为从右到左为前 12 天，中间 12 天，后 3 天

- **Weekly –** 以周显示流量。单位为天。垂直分割线为 1 周，垂直分割线时间区域分三段，表示为从右到左为前 7 周，中间 7 周，后 4 周



3.14 Cloud

从 RouterOS 6.14 开始 MikroTik 为自家的网络设备提供云服务功能，包括动态域名解析服务、网络对时和云备份（x86 平台不支持）。

操作路径: /ip cloud

从 RouterOS 6.27 开始 cloud 操作设置有所调整，即更加体现 DDNS 服务和时间同步功能的特点，6.27 变动如下：

*) cloud – 添加时间探测功能"/system clock time-zone-autodetect";
*) cloud – 重命名"/ip cloud enabled"，修改为 "/ip cloud ddns-enabled";
*) cloud – 将"/ip cloud update-time" 从"/ip cloud ddns-enabled"中独立出来，执行单独命令。
*) cloud – 当设置"/ip cloud ddns-enabled=no" 会发送信息给服务器，将禁用该 RouterBOARD 的 DDNS 服务名;
*) cloud – "/ip cloud force-update" 命令将同时工作于关闭 DDNS 服务，即 "/ip cloud ddns-enabled = no"，等同于用户想要禁用 DDNS 服务，如果要再次开启 DDNS 服务，cloud 服务器会重新分配 DDNS 服务名;

从 v6.44 后，Cloud 提供三种服务：

- ddns ：提供 IPv4 和 IPv6 公网地址的 DNS 域名解析
- update time：时间同步(当 NTP 无法工作时，实现时间同步，能精确到秒)
- backup：云备份，v6.44 后新增配置备份功能

注意：IP/Cloud 服务在 Cloud Hosted Router (CHR)不能直接开启，需要申请许可

DDNS（动态域名解析）

DDNS 是一种动态 DNS 服务，能定期更新 IPv4 对应 A 记录和 Ipv6 对应的 AAAA 记录。当运营商分配的是动态公网 IP 地址，IP 地址会定期变化，但你总是需要一个可以用来远程连接设备的地址时，DDNS 服务是非常有用的。在 MikroTik 设备可以 cloud 服务协助你完成设备的远程访问。

下面是 DDNS 的工作流程：

- 路由器每 1 分钟检查一次网关 IP 地址是否变更
- 路由器会用 15 秒等待 cloud 服务器回应请求
- DDNS 记录 TTL 值时间为 60 秒
- 使用 UDP/15252 端口，发送加密数据包到 **cloud.mikrotik.com** 和 **cloud2.mikrotik.com**

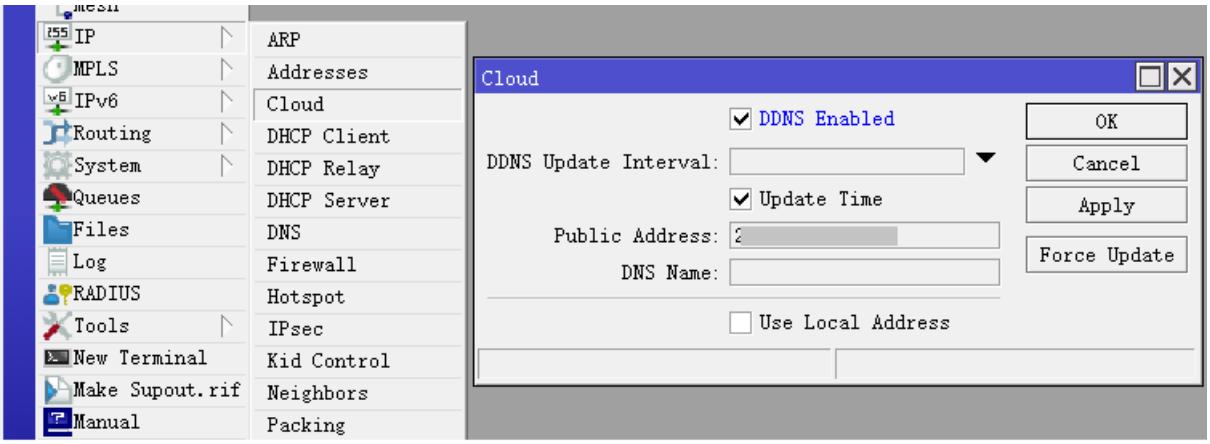
从 RouterOSv6.43 开始，**cloud2.mikrotik.com** 支持 IPv6 连接，当你使用 IPv6 地址时，DNS AAAA 记录会创建并记录你设备的 IPv6 公网地址。如果你的设备仅使用 IPv4 地址连接 cloud2.mikrotik.com，那么仅会做 A 记录，记录你的 IPv4 地址

启用 DDNS 服务，命令行配置如下，v6.27 后使用 **set ddns-enabled=yes**：

```
[admin@MikroTik] /ip cloud> set ddns-enabled=yes
[admin@MikroTik] /ip cloud> print
      ddns-enabled: yes
  ddns-update-interval: none
           update-time: yes
        public-address: 159.148.147.211
              dns-name: 123456789abc.sn.mynetname.net
                status: updated
[admin@MikroTik] /ip cloud>
```

当服务启用，设备自动生成的 DNS 域名将永久的保存到 MikroTik 云服务器，并将当前设备最新的公网 IP 地址，和域名匹配记录到 MikroTik 云服务器

Winbox 中启用 DDNS 服务



强制更新 DNS 操作：

```
[admin@MikroTik] > /ip cloud force-update
```

注意：在实际连接到设备过程中，首先通过域名解析后返回的 IP 地址连接到设备，请确保 WAN 口访问没有受到限制。

警告：如果路由有多个 IP 地址或网关，会导致 DDNS 的 IP 地址更新出现问题，如果要确保正确的公网 IP 地址更新，请解析 **cloud.mikrotik.com** 和 **cloud2.mikrotik.com** 域名的主机 IP，指定到路由器的默认路由。

如果你是在运营商 nat 下的路由器，cloud 会提示 DDNS 请求 IP 与路由器本地 IP 地址不相同，将无法工作，确保获取的 IP 是公网地址。IP cloud 对于在国内运营商的 PPPoE 拨号获取动态公网 IP 的用户是非常方便，无需在申请第三方 DDNS 服务，。

Update time（时间更新）

设备上正确的时间非常重要，错误的时间会导致系统日志问题，设备的隧道和 HTTPS 连接故障，更新系统的时间，通常 NTP 或 SNTP，对于 NTP 服务器，首先你需要指定服务器的 IP 地址或域名，并在 RouterOS 系统进行配置。通常情况下，配置 NTP/SNTP 只是为了在设备上设置正确的时间，为了简单起见，MikroTik 提供了云的时间更新服务。具体操作如下：

- 近似的时间（精度秒，取决于 UDP 数据包延迟)
- 更新条件是在重新启动后和每次 DDNS 更新周期进行时间更新(当路由器的公网 IP 地址更改时或在使用 force-update 命令之后)
- 通过加密数据包发送到 **cloud.mikrotik.com** 和 **cloud2.mikrotik.com** 的 UDP/15252 端口
- Time-zone 时区，取决设备的公网 IP 在 MikroTik 商业数据库的地址归属

在设备上启用时间更新：

```
[admin@MikroTik] > /ip cloud set update-time=yes
```

启用自动时区探测：

```
[admin@MikroTik] > /system clock set time-zone-autodetect=yes
```

注意：如果 **/ip cloud update-time** 设置为 auto，且 NTP 或 SNTP 客户端服务没有启用，设备的时间更新将通过 MikroTik 云服务器

属性	描述
ddns-enabled (*yes* | *no*; 默认: **no**)	如果设置为 yes，那么设备将向 MikroTik 的云服务器发送一条加密信息。然后，服务器将解密该信息，并验证发送方是否是为可信的 MikroTik 设备。如果一切正常，那么 MikroTik 的云服务器将为该设备创建一个 DDNS 记录并向该设备发送。每分钟，路由器上的 cloud 服务都会检查发送到 MikroTik 的云服务器的 WAN IP 地址是否匹配，如果 IP 地址发生变化，则会向云服务器发送加密更新。
ddns-update-interval (*time, 最小值 60 秒*; 默认: **none**)	如果设置为 none，它将继续在内部检查 IP 地址更新，并根据需要连接到 IP 云服务器更新。如果使用的 IP 地址不在路由器本身上（如路由器没用公网 IP），不能作为路由器内部的一个值来检查。
update-time (*yes* | *no*; 默认: **yes**)	如果设置为 yes，将通过云服务器更新路由器时间，前提是 NTP 和 SNTP 客户端为启用。如果 update-time 设置为 yes，即使在 ddns-enabled 设置为 no 的情况下，时钟也会被更新
public-address (*read-only: IP 地址*)	显示发送到云服务器的 IPv4 地址，仅当完成于云服务器更新后，才会显示 IPv4 地址
public-address-ivp6 (*只读: IP 地址*)	显示发送到云服务器的 IPv6 地址，仅当完成于云服务器更新后，才会显示 IPv6 地址
warning (*只读: 字符*)	显示警告消息，如果设备发送的 UDP 包头存在多次不同 IP 地址到 MikroTik 的云服务器。例如:“DDNS 服务器接收到来自 IP 123.123.123.123 的请求，但是你的本地 IP 是 192.168.88.23;DDNS 服务可能无法工作"
dns-name (*只读: 域名*)	显示分配给设备的 DNS 名称。名称由 12 个字符的序列号组成，后面加上.sn.mynetname.net。只有在成功完成至少一个 ddns 请求之后，该字段才可见。
status (*只读: 字符*)	下面是描述当前 DNS -service 状态信息： • **updating...** • **updated** • **Error: no Internet connection** • **Error: request timed out** • **Error: REJECTED. Contact MikroTik support** • **Error: internal error** - （不应该发生的。可能的原因是路由器内存耗尽）

在 cloud 中有可以设置使用本地地址，操作路径: /ip cloud advanced

属性	描述
use-local-address (*yes* | *no*; 默认: **no**)	默认请求下，域名解析通过从 UDP 包头获取公网 IP 地址，用于记录设备的 IP 地址，如果你希望发送本地或内部私网 IP 地址，可以选择该属性为 yes

Backup（云备份）

从 v6.44 开始，设备配置备份可以通过 MikroTik 云服务器完成，备份服务允许将你的加密备份文件上传到 MikroTik 服务器。下载备份并应用文件只需要确保能连接到 MikroTik 云服务器即可。下面是备份操作相关细节：

- 每个设备提供一个免费文件备份
- 最大备份空间 15MB
- 通过 UDP/15252 和 TCP/15252 发送备份文件到 **cloud2.mikrotik.com**

创建一个新的备份，并上传到 MikroTik 云服务器：

```
[admin@MikroTik] > /system backup cloud upload-file action=create-and-upload
password=test123!!!
[admin@MikroTik] > /system backup cloud print
 0 name="cloud-20180921-162649" size=13.2KiB ros-version="6.44beta9" date=sep/21/2018
16:26:49 status="ok" secret-download-key="AbCdEfGhIjKlM1234567890"
```

注意： create-and-upload 命令将创建一个新的备份文件，使用 AES 加密备份文件并上传。对于 upload 操作命令，密码属性没有作用，因为 upload 操作命令上传的只是已经创建的系统备份文件

从服务器下载备份文件，会保存到设备的内存 RAM 中（并非保存在 file 文件系统下）：

```
[admin@MikroTik] > /system backup cloud download-file action=download number=0
### OR
[admin@MikroTik] > /system backup cloud download-file action=download
secret-download-key=AbCdEfGhIjKlM1234567890
```

警告： secret-download-key 是唯一的识别码，可用于将加密的备份下载到其他设备。通过使用 secret-download-key 从任何位置和任何设备下载加密的备份，所以需要保管好个识别码。下载的备份仍然使用 AES 加密，不过请确保使用的是强密码。

删除上传的备份文件：

```
/system backup cloud remove-file number=0
```

上传一份现有的备份文件：

```
[admin@MikroTik] > /system backup save encryption=aes-sha256 name=old_backup
password=test123!!!
[admin@MikroTik] > /system backup cloud upload-file action=upload
src-file=old_backup.backup
[admin@MikroTik] > /system backup cloud print
```

```
 0 name="cloud-20180921-164044" size=13.2KiB ros-version="6.44beta9" date=sep/21/2018
16:40:44 status="ok" secret-download-key="AbCdEfGhIjKlM1234567890"
```

注意：确保备份是使用 AES 加密的，否则云服务器将拒绝备份上传。因为每个设备只有一个免费文件槽，所以在上传新备份之前，会删除现有的备份

警告：当导入备份时，所有的 MAC 地址都被设置为备份时设备正在使用的 MAC 地址。这样可以降低替换设备失败的几率，但如果要把给备份同时导入多个设备，这会在多个设备上设置相同的 MAC 地址，导致出现 MAC 冲突，导致网络故障。如果导入到非备份设备，可以通过在命令行输入 **reset-mac-address** 命令复位接口原来的 MAC 地址

3.15 Flashfig

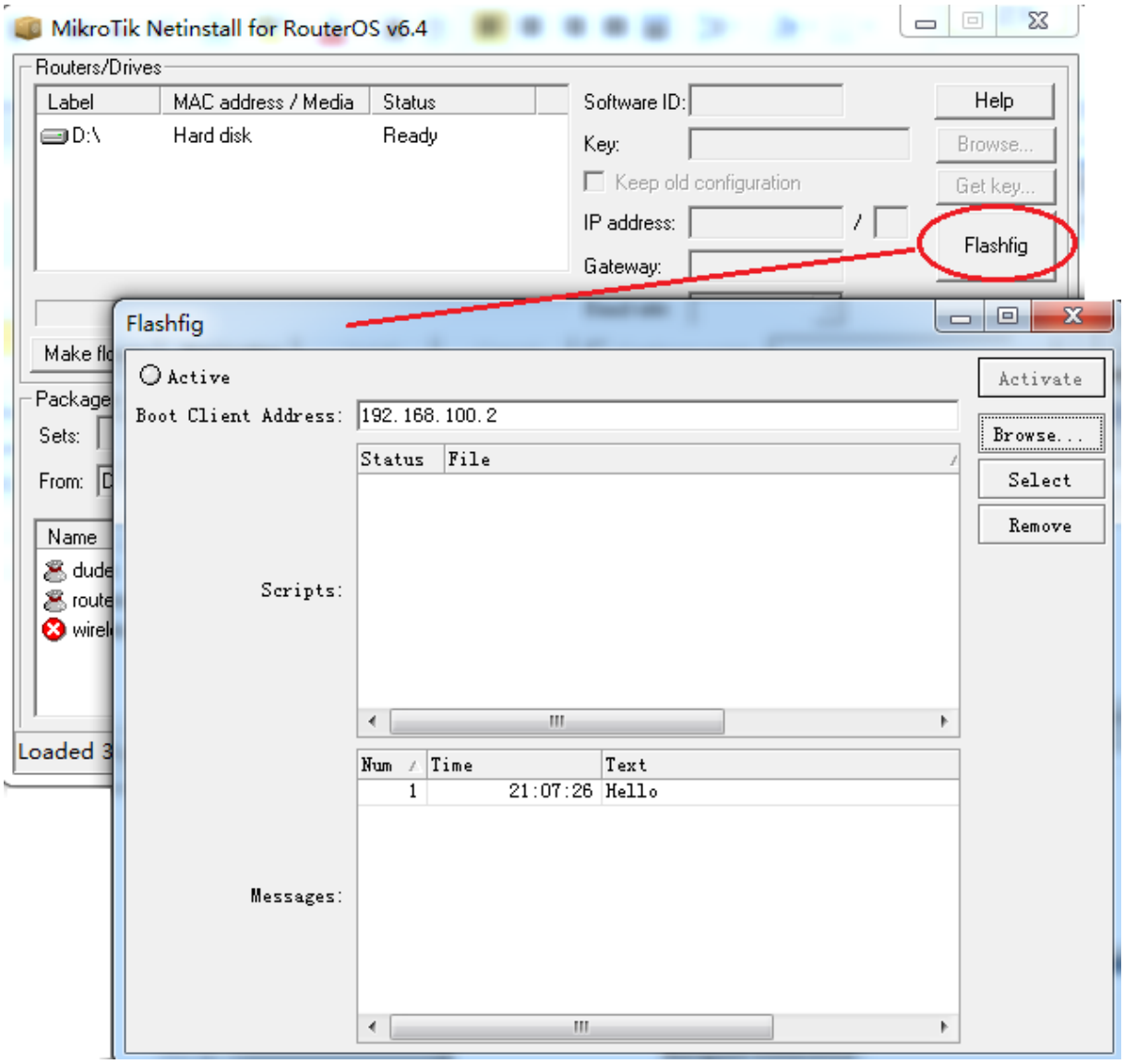
Flashfig 是一个应用于大量 RouterBOARD 的初始配置程序，主要被用于 MikroTik 分销商，ISP 或需要短时间对大量 MikroTik 路由进行配置的公司。

Flashfig 应用于 MikroTik RouterBOARD 的 RouterOS 配置刷新，在 3 秒内完成，并且可以批量刷新 RouterBOARD 的配置，完成这个操作需要将 RouterBOARD 通电和连接网络，通过网络连接到运行 Flashfig 应用的 windows 计算机。Flashfig 应用程序集成在 Netinstall 软件中。

Flashfig 支持所有的 RouterBOARD，运行 Flashfig 软件的计算机和 RouterBOARD 都必须在同一广播域，即二层以太网络连接。

Flashfig 配置实例

以下的实例将对如何使用 Flashfig 进行介绍，Flashfig 主要基于 MikroTik RouterBOARD 产品。Flashfig 应用程序集成在 Netinstall 中



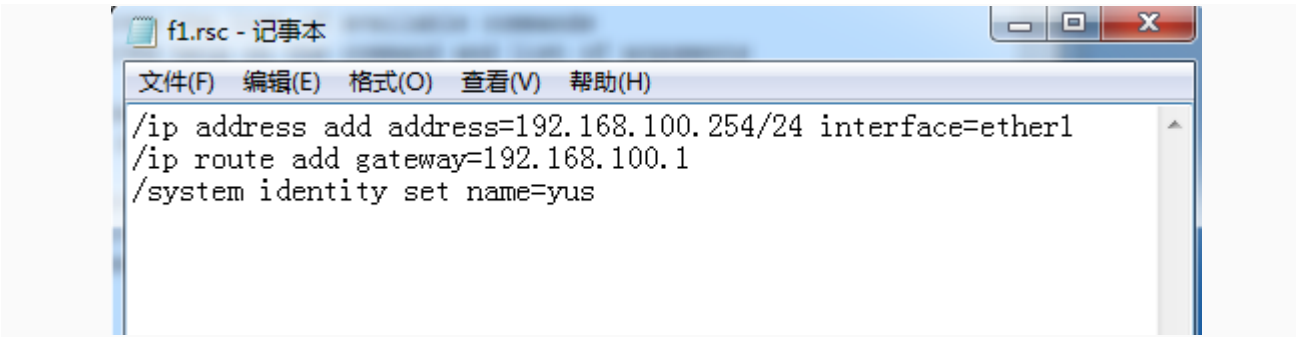
配置前准备，Windows 计算机必须具备以下条件 ：

- 以太网接口
- .rsc 的 RouterOS 配置脚本文件，类似 export/import 文件;
- 下载最新的 Netinstall 应用软件，从 www.mikrotik.com 上下载

RouterBOARD 配置，RouterBOARD 首选引导为 Flashfig

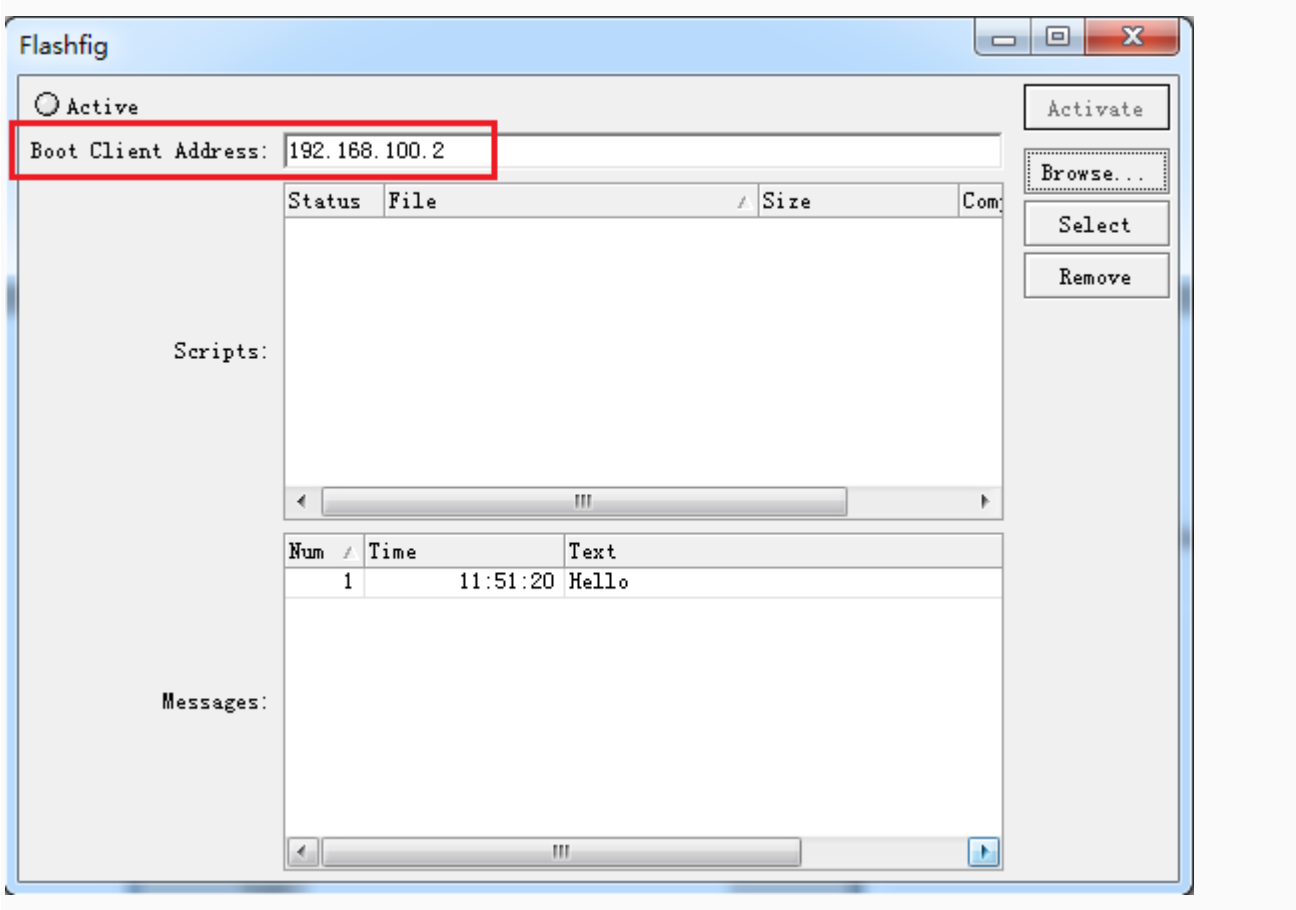
Netinstall 配置

- 打开 Netinstall，并运行 Flashfig;
- 准备.rsc 文件， .rsc 文件是 RouterOS 脚本文件格式，能直接应用到 RouterOS 的 CLI 命令，可以通过 txt 的文本创建脚本文件。

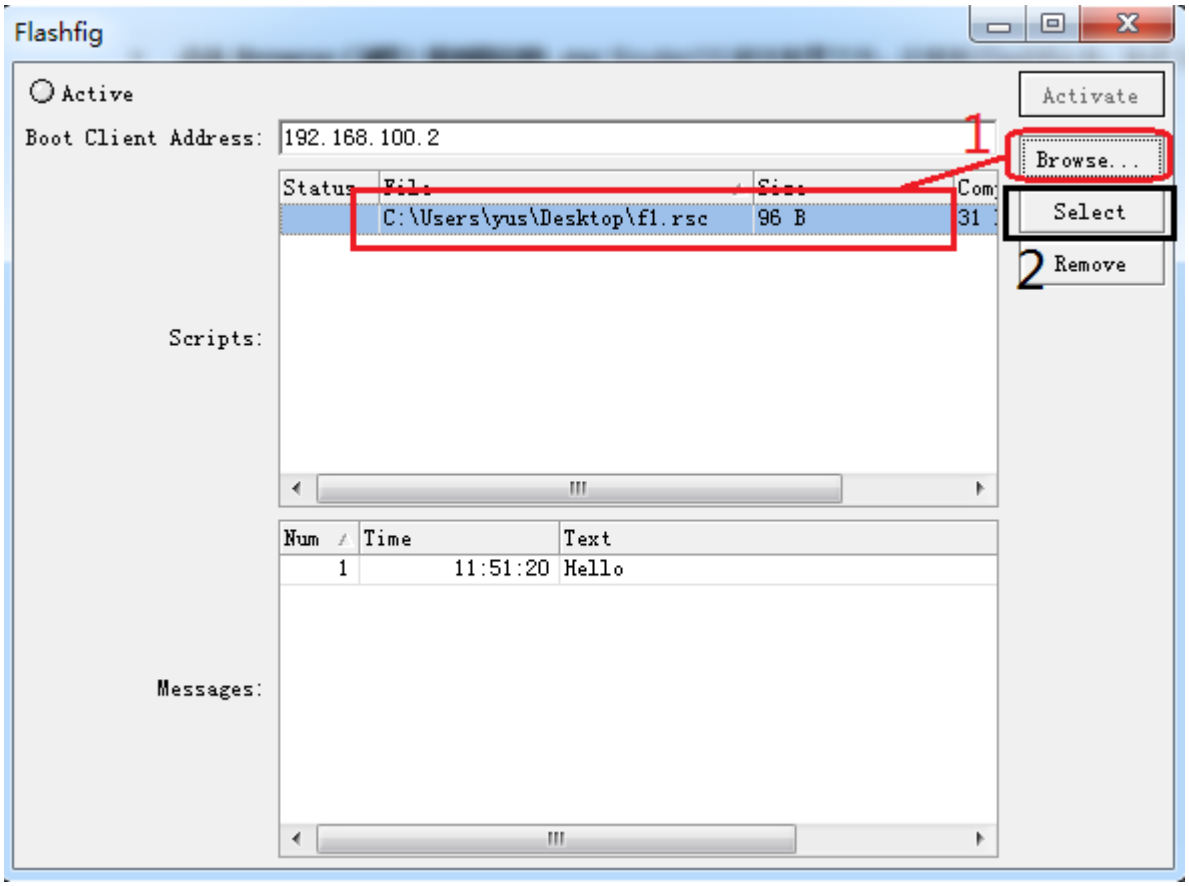


上面是编辑的 f1.rsc 脚本，编辑好后保存。

分别引导客户端的 IP 地址，这个 IP 地址必须和 windows 计算机在相同子网中，这里我们设置 windows 计算机 IP 为 192.168.100.99，分配给 RouterBOARD 引导地址为 192.168.100.2

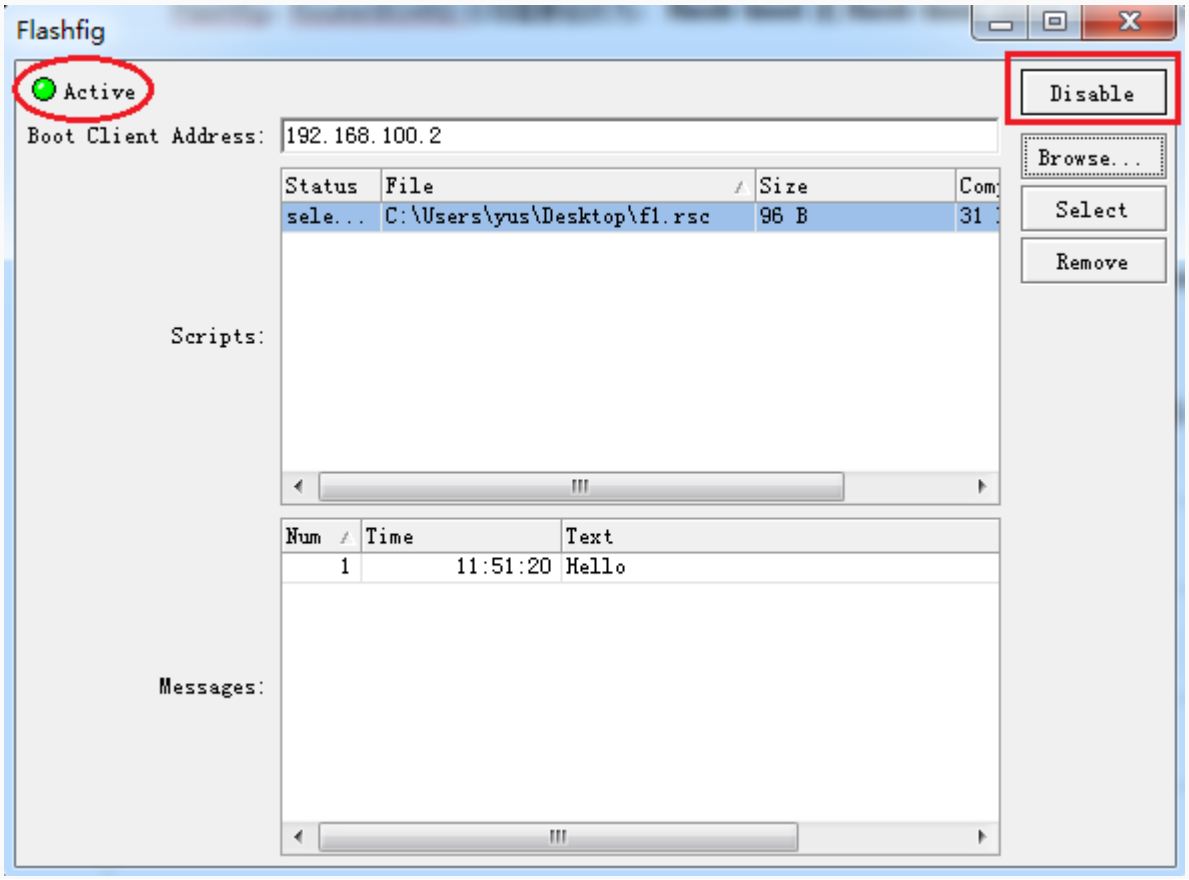


点击 Browse（浏览）找到指定的 .rsc RouterOS 脚本配置文件，应用到 Flashfig 中，并点 Select 选择急脚本文件指定生效。



注意：选择脚本文件不能使用中文路径。

点击 Activate 启动 Flashfig 服务，任何支持 Flahsfig 的 RouterBOARD，启动后通过网络都会连接到 Flashfig，RouterBOARD 引导需要修改为：flash-boot 或 flash-boot-once-then-nand。



RouterBOARD 配置

所有 RouterBOARD 在出厂时，就被设置为 Flashfig 模式引导，即在 RouterBOARD 出厂为无配置状态。

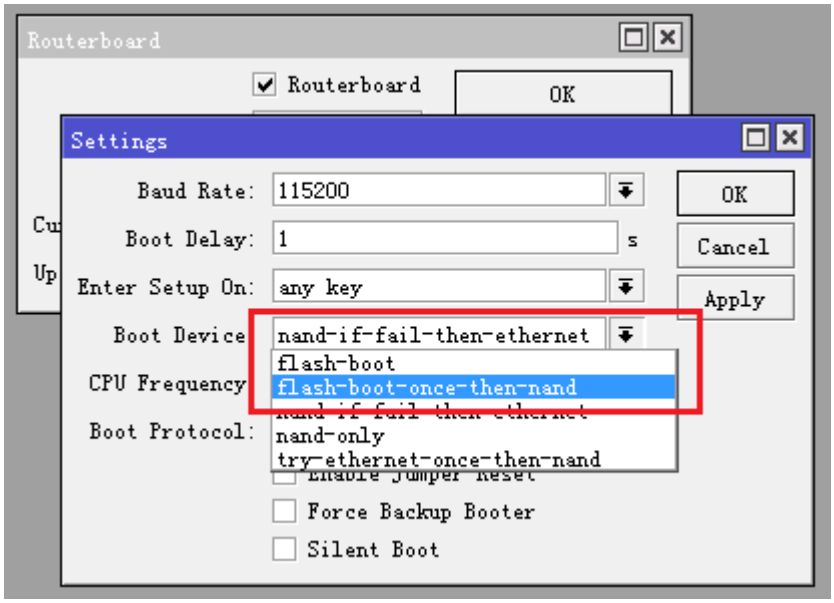
如果你的 RouterBOARD 的 Flashfig 没有启动，可以通过 Winbox 或 Console 连接修改引导，CLI 配置如下

```
/system routerboard settings set boot-device=flash-boot
```

也可以选择：

```
/system routerboard settigs set boot-device=flash-boot-once-then-nand
```

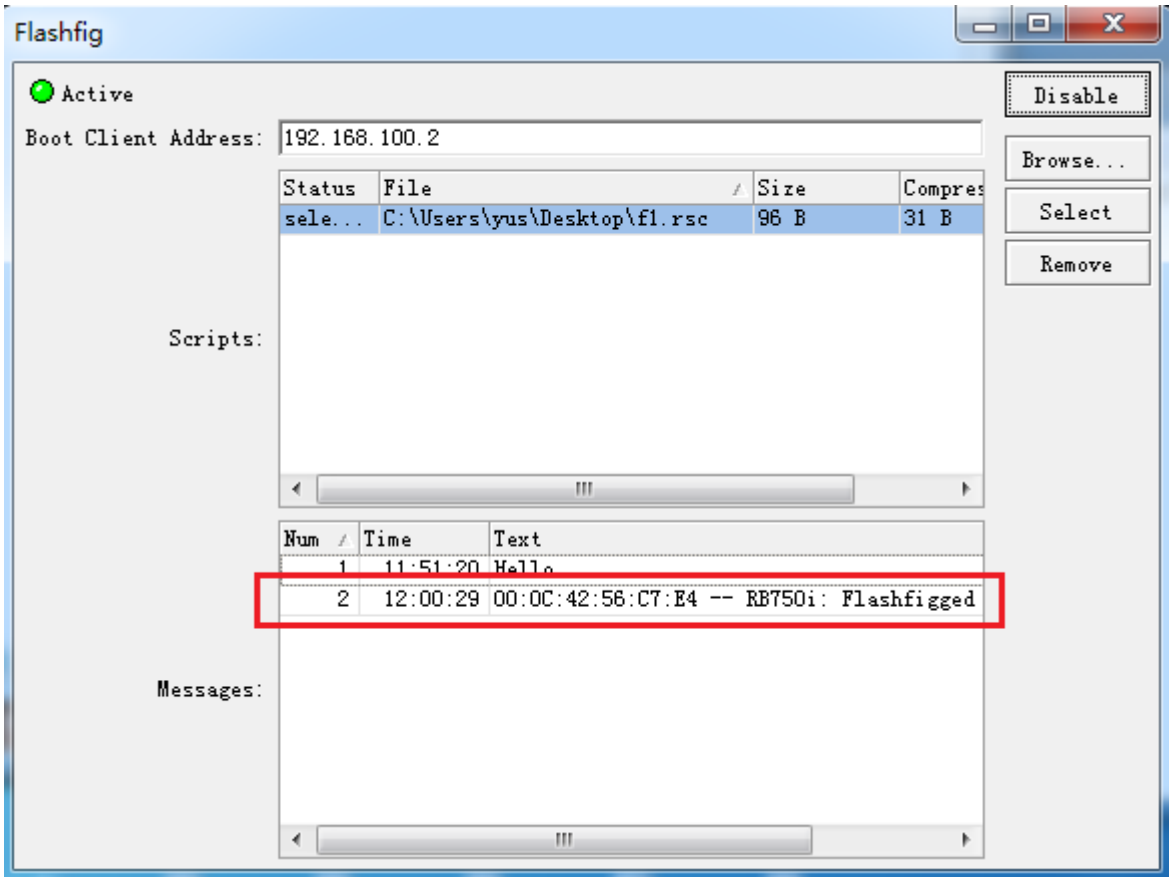
在 Winbox 下进入/system roeuterboard 配置：



- flash-boot：只使用 Flashfig 引导
- flash-boot-once-then-nand：Flashfig 引导一次，如果失败就选择 NAND 的系统 Flash 引导，建议选择此项，便于失败后重新操作。

以上配置准备好后，RouterBOARD 已经准备好连接 Flashfig 服务

再次确认 RouterBOARD 和 Flashfig 计算机在相同的二层广播域，且 IP 地址在同一子网，RouterBOARD 重启或通电启动，检查 Flashfig 程序，RouterBOARD 是否连接

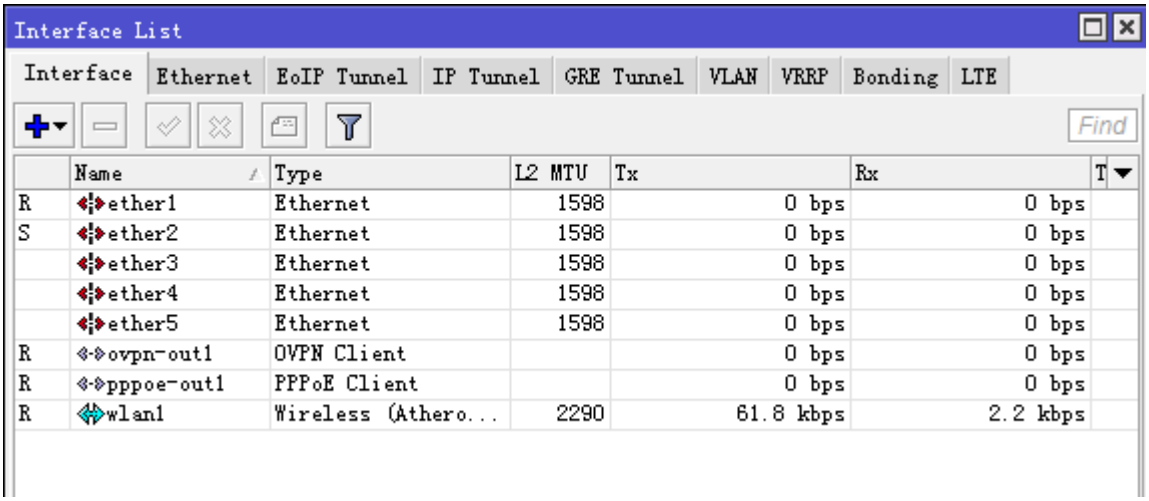


在 Messages 中提示，”RBxxx Flashfigged”，且 RouterBOARD 发出声音或 LED 闪烁信号，就可以将 RouterBOARD 断电重启。

第四章 Interface（接口配置）

在 interface 中包括硬件接口网卡配置与虚拟接口的网卡配置，物理接口：Ethernet、wireless、ISDN 等，虚拟接口：PPP、PPPoE、PPTP、L2TP、OVPN、SSTP、EoIP、IPIP、Gre 和 Bonding 等等。

MikroTik RouterOS 支持各种网络适配器，同样也支持各种隧道协议和 VLAN 的虚拟接口。 这些接口可以在 interface 查看，也可以进行相应的配置。



	Name	Type	L2 MTU	Tx	Rx	T
R	ether1	Ethernet	1598	0 bps	0 bps	
S	ether2	Ethernet	1598	0 bps	0 bps	
	ether3	Ethernet	1598	0 bps	0 bps	
	ether4	Ethernet	1598	0 bps	0 bps	
	ether5	Ethernet	1598	0 bps	0 bps	
R	ovpn-out1	OVPN Client		0 bps	0 bps	
R	pppoe-out1	PPPoE Client		0 bps	0 bps	
R	wlan1	Wireless (Athero...	2290	61.8 kbps	2.2 kbps	

4.1 Interface 基本操作

操作路径：***/interface***

属性描述

name (*文本*) – 接口名称
status – 显示接口状态
type (只读*:* arlan | bridge | cyclades | eoip | ethernet | farsync | ipip | isdn-client | isdn-server | l2tp-client | l2tp-server | moxa-c101 | moxa-c502 | mtsync | pc | ppp-client | ppp-server | pppoe-client | pppoe-server | pptp-client | pptp-server | pvc | radiolan | sbe | vlan | wavelan | wireless | xpeed) – 接口类型
mtu (*整型*) – 接口最大传输单位(bytes)

查看下面的接口列表：

```
[admin@MikroTik] /interface> print
Flags: D - dynamic, X - disabled, R - running, S - slave
 #      NAME                                TYPE          MTU L2MTU   MAX-L2MTU
 0  R   ether1                              ether        1500  1598        2028
 1    S ether2                              ether        1500  1598        2028
 2      ether3                              ether        1500  1598        2028
 3      ether4                              ether        1500  1598        2028
 4      ether5                              ether        1500  1598        2028
 5  R   wlan1                               wlan         1500  2290
```

```
 6  R  ovpn-out1                          ovpn-out     1500
 7  R  pppoe-out1                         pppoe-out    1480
```

进入/interface bridge 桥接配置，添加一个桥：

```
[admin@MikroTik] /interface bridge> add
[admin@MikroTik] /interface bridge> prin
Flags: X - disabled, R - running
 0  R name="bridge1" mtu=1500 arp=enabled mac-address=00:00:00:00:00:00
      protocol-mode=none priority=0x8000 auto-mac=yes
      admin-mac=00:00:00:00:00:00 max-message-age=20s forward-delay=15s
      transmit-hold-count=6 ageing-time=5m
[admin@MikroTik] /interface bridge>
```

RouterOS 3.22 后支持新的命令

```
/interface print stats
```

该命令将显示包括 packets, bytes, drops 和 errors 信息。

所有网卡支持这个功能的都会被显示，一些网卡不支持 Erro 和 Drop (RB4XX 除了 RB450G ether 2-5), 所以这些设备不能显示这些参数的统计

```
[admin@MikroTik] /interface> print stats
Flags: D - dynamic, X - disabled, R - running, S - slave
 #      NAME                BYTES                    PACKETS            DROPS      ERRORS
 0  R   ether1       733426639/412942115       2753779/2460189
 1  X   wlan1              0/0                      0/0                0/0        0/0
 2  R   pppoe-out1   657305779/347536100       2683439/2442508      0/0        0/0
 3  R   wlan2         307621318/865149222       2430478/2652910      0/0        0/0
 4  X   bridge1            0/0                      0/0                0/0        0/0
 5 DR   <ovpn-vpn>      622503/37458              748/571              0/0        0/0
```

注意：RouterBOARD 和 x86 平台在端口连接状态显示有所区别，RouterBOARD 设备如果以太网接口有连接会显示前缀“R”，而 x86 平台不管是否有连接前缀都为“R”

4.2 流量监控

指令名称：***/interface monitor-traffic***

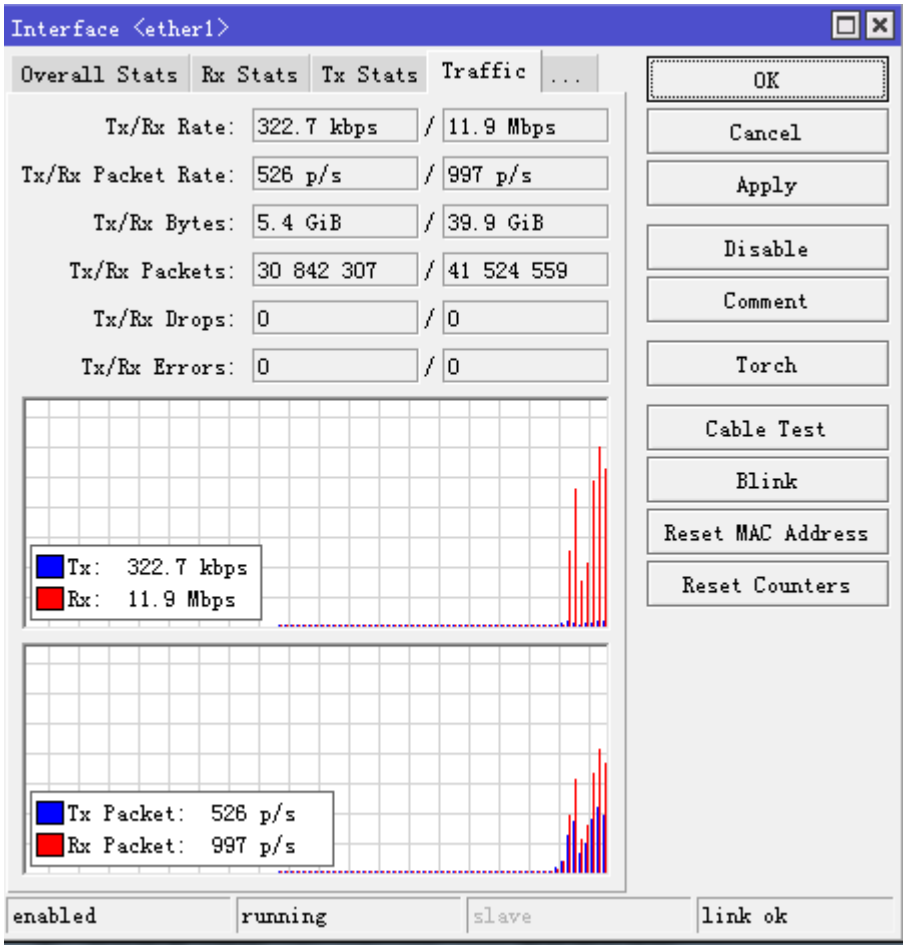
注：可以监控网络接口的数据流量，并且能同时监视多个网络接口的流量情况，在执行监控命令是，参数只能是网卡名称，不能使用网卡的编号。

例如：在命令行下的多网卡流量监视：

```
[admin@MikroTik] interface> monitor-traffic ether1,wlan1
    received-packets-per-second: 1          0
       received-bits-per-second: 475bps     0bps
```

```
          sent-packets-per-second: 1          1
             sent-bits-per-second: 2.43kbps 198bps
-- [Q quit|D dump|C-z pause]
```

在 winbox 中可通过图像查看流量情况：



4.3 以太网接口（Ethernet）

MikroTik RouterOS 支持各种类型的以太网接口，范围从 10Mbps 到 10Gbps 以太网线，1Gbps 和 10Gbps 的 SFP/SFP+接口和 40Gbps 的 QSFP 接口。部分 RouterBoard 设备配备了组合（combo）接口，该接口同时包含两种接口类型(1Gbps 电口以太网和 SFP 接口)。

功能包：***system***
等级：*Level1*
操作路径：***/interface ethernet***

属性描述

属性	描述

arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; 默认: **enabled**)	地址解析协议模式： • disabled – 禁用接口的 ARP 功能 • enabled – 启用接口的 ARP 功能 • proxy-arp – 接口将开启 ARP 代理功能 • reply-only – 接口仅对 ARP 列表（/ip arp）的静态条目做响应，动态的 ARP 将不做响应。即对 IP 地址与 MAC 做静态绑定。
auto-negotiation (*yes* | *no*; 默认: **yes**)	当启用，接口将自动协商链路的最大连接速率 • **1:** Auto-negotiation 在另一端不应被禁用，否则以太网接口将无法工作。 • **2:** Gigabit 链路无法在 auto-negotiation 禁用状态下工作
bandwidth (*整型* / *整型* ; 默认: **unlimited/unlimited**)	设置接口的 rx/tx 带宽速率，单位 kbps。TX 速率支持所有的 Atheros 交换芯片（switch-chip）。 RX 速率仅支持 AR8327/QCA8337 交换芯片。v6.41 后移除
cable-setting (*默认* | *short* | *standard*; 默认: **default**)	修改网线长度设置（仅适用于 NS DP83815/6 网卡)
comment (*字符*; 默认:)	注释描述
disable-running-check (*yes* | *no*; 默认: **yes**)	禁用网卡运行（running）检查，如果这个值被设置为“no”，路由器会自动检测网卡是否与网络中的设备连接。默认值是“yes”，因为较老的网卡不支持该属性(仅适用于 x86)
full-duplex (*yes* | *no*; 默认: **yes**)	是否开启全双工，即定义双向数据传输
l2mtu (*integer [0..65536]*; 默认:)	二层 MTU 值
mac-address (*MAC*; 默认:)	接口的 MAC 地址
master-port (*name*; 默认: **none**)	设置接口为二层接口，从属于那个网卡下的交换组。v6.41 后移除，改为 bridge 配置
mdix-enable (*yes* | *no*; 默认: **yes**)	是否在该接口上开启 MDI/X 自动线序纠错功能。
mtu (*integer [0..65536]*; 默认: **1500**)	三层 MTU 值
name (*字符*; 默认:)	网卡名称
speed (*10Mbps* | *10Gbps* | *100Mbps* | *1Gbps*; 默认:)	设置接口的传输速率，默认情况下会自动识别网卡的最大传输速率

从 3.17 开始 RouterOS 支持 Intel 10Gbit PCI-E 以太网卡，后期对 10Gbit 网卡逐步完善。

进入 interface ethernet 查看以太网卡参数：

```
[admin@MikroTik] /interface ethernet> print detail
Flags: X - disabled, R - running, S - slave
 0 R  name="ether1" mtu=1500 l2mtu=1526 mac-address=00:0C:42:37:58:66
      arp=enabled auto-negotiation=yes full-duplex=yes speed=100Mbps

 1    name="ether2" mtu=1500 l2mtu=1522 mac-address=00:0C:42:37:58:67
      arp=enabled auto-negotiation=yes full-duplex=yes speed=100Mbps
      master-port=none bandwidth=unlimited/unlimited switch=switch1

 2    name="ether3" mtu=1500 l2mtu=1522 mac-address=00:0C:42:37:58:68
      arp=enabled auto-negotiation=yes full-duplex=yes speed=100Mbps
      master-port=none bandwidth=unlimited/unlimited switch=switch1
```

```
3     name="ether4" mtu=1500 l2mtu=1522 mac-address=00:0C:42:37:58:69
      arp=enabled auto-negotiation=yes full-duplex=yes speed=100Mbps
      master-port=none bandwidth=unlimited/unlimited switch=switch1

4     name="ether5" mtu=1500 l2mtu=1522 mac-address=00:0C:42:37:58:6A
      arp=enabled auto-negotiation=yes full-duplex=yes speed=100Mbps
      master-port=none bandwidth=unlimited/unlimited switch=switch1
[admin@MikroTik] /interface ethernet>
```

对于 RouterBOARD 设备有 switch 功能的参数有别于 x86，会有 mater-port 和 bandwidth 属性，请参考 RouterBOARD 章节，在 v6.41 后，bridge 调整不再有 mater-port 和 bandwidth 属性

接口状态

属性描述

属性	描述
advertising (*10M-full* | *10M-half* | *100M-full* | *100M-half* | *1000M-full* | *1000M-half* | *2500M-full* | *5000M-full* | *10000M-full*)	宣告以太网接口协商速度和双工模式，仅在启用 `auto-negotiation` 时适用
auto-negotiation (*disabled* | *done* | *failed* | *incomplete*)	当前自动协商状态： • `disabled` – 协商关闭 • `done` – 协商完成 • `failed` – 协商失败 • `incomplete` – 协商未完成
default-cable-settings (*short* | *standard*)	默认线缆长度设置(仅适用于支持 NS DP83815/6 的网卡) • `short` – 支持短线缆 • `standard` – 支持标准线缆
full-duplex (*yes* | *no*)	是否开启全双工
link-partner-advertising (*10M-full* | *10M-half* | *100M-full* | *100M-half* | *1000M-full* | *1000M-half* | *2500M-full* | *5000M-full* | *10000M-full*)	对端以太网接口协商速度和双工模式，仅在启用 `auto-negotiation` 时适用
rate (*10Mbps* | *100Mbps* | *1Gbps* | *2.5Gbps* | *5Gbps* | *10Gbps* | *40Gbps* |)	连接的实际数据速率.
status (*link-ok* | *no-link* | *unknown*)	当前接口的连接状态 • `link-ok` – 网卡连接到网络 • `no-link` – 网卡未能连接到网络 • `unknown` -连接不被识别(如果卡没有报告连接状态)

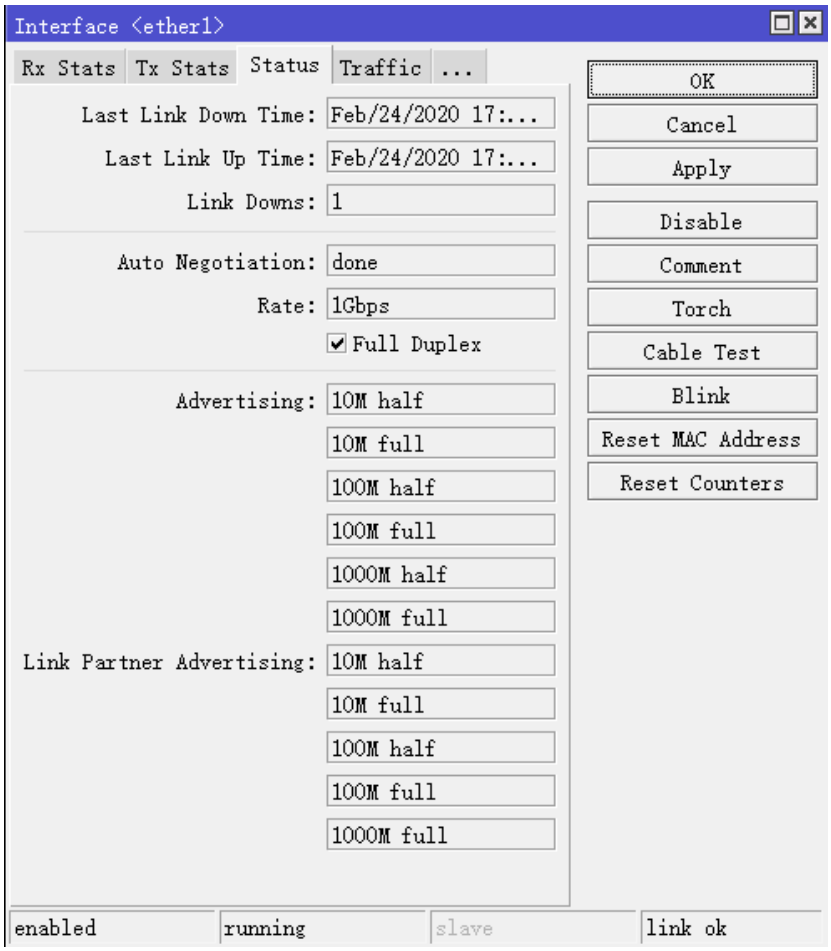
tx-flow-control (*yes* | *no*)	是否使用 TX 流量控制
rx-flow-control (*yes* | *no*)	是否使用 RX 流量控制
combo-state (*copper* | *sfp*)	combo 接口当前的 combo 模式：电口或光口
sfp-module-present (*yes* | *no*)	光模块是否在槽中
sfp-rx-lose (*yes* | *no*)	光模块接收信号是否丢失
sfp-tx-fault (*yes* | *no*)	光模块发送是否处于故障状态
sfp-type (*SFP-or-SFP+* | *DWDM-SFP* | *QSFP+*)	光模块类型
sfp-connector-type (*SC* | *LC* | *optical-pigtail* | *copper-pigtail* | *multifiber-parallel-optic-1x12* | *no-separable-connector* | *RJ45*)	光模块连接器类型
sfp-vendor-name (*string*)	模块制造商
sfp-vendor-serial (*string*)	模块 SN 号
sfp-manufacturing-date (*date*)	模块生成日期
sfp-wavelength (*nm*)	模块发射信号波长
sfp-temperature (*C*)	模块温度（摄氏）
sfp-supply-voltage (*V*)	模块工作电压
sfp-tx-bias-current (*mA*)	模块 tx 电流
sfp-tx-power (*dBm*)	当前模块发射功率
sfp-rx-power (*dBm*)	当前模块接收功率
sfp-connector-type (SC | LC | optical-pigtail | copper-pigtail | multifiber-parallel-optic-1x12 | no-separable-connector | RJ45)	模块通信接口连接类型
sfp-link-length-9um (m)	单模 9/125um 光纤模块支持通信长度
sfp-link-length-50um (m)	多模 50/125um 光纤模块支持通信长度(OM2)
sfp-link-length-62um (m)	多模 62.5/125um 光纤模块支持通信长度(OM1)
sfp-link-length-copper (m)	光转电模块的支持的通信长度

通过 Monitor 命令可以查看 ether1 网卡的连接状态，link-ok 表示线缆连接正常，rate（速率）为 1Gbps：

```
[admin@MikroTik] > /interface ethernet monitor ether1
                      name: ether1
                    status: link-ok
```

```
          auto-negotiation: done
                      rate: 1Gbps
                 full-duplex: yes
              tx-flow-control: no
              rx-flow-control: no
                 advertising: 10M-half,10M-full,100M-half,100M-full,1000M-half,1000M-full
link-partner-advertising: 10M-half,10M-full,100M-half,100M-full,1000M-full
```

winbox 的接口状态显示：



修改以太网卡 mac 地址：

```
[admin@MikroTik] interface ethernet>set 0 mac-address=00:0C:42:03:11:0A
```

线路故障探测

RouterOS v6rc4 版本后，能探测网线连接故障，探测网线大概的故障距离，如网线未连接或某个位置断开。RouterOS 将有如下通知：

- 网线中的哪一组线缆损坏
- 故障距离
- 线缆是否损坏、短路或开路

注意：这个功能仅支持 RouterBOARD 和 CCR 设备，如 SXT-G、SXT Lite,、RB711G、RB2011、RB750 和 CCR 系列和其他采用相同交换芯片的设备。

下面是 CCR 线路测试：

```
[admin@CCR] > interface ethernet cable-test ether2
           name: ether2
         status: no-link
  cable-pairs: open:4,open:4,open:4,open:4
```

在上面的事例中，线缆没有短路，但以太网线的四条线缆显示“open”，距离为 4 米，通过判断线缆在距离路由器 4 米中断。

4.4 Lists

在 RouterOS v6.36 后，新增了接口列表，可以通过 list 定义接口管理列表，并能配置到各个功能下，例如 Neighbor、discovery、Firewall、Bridge 和 MAC server 等

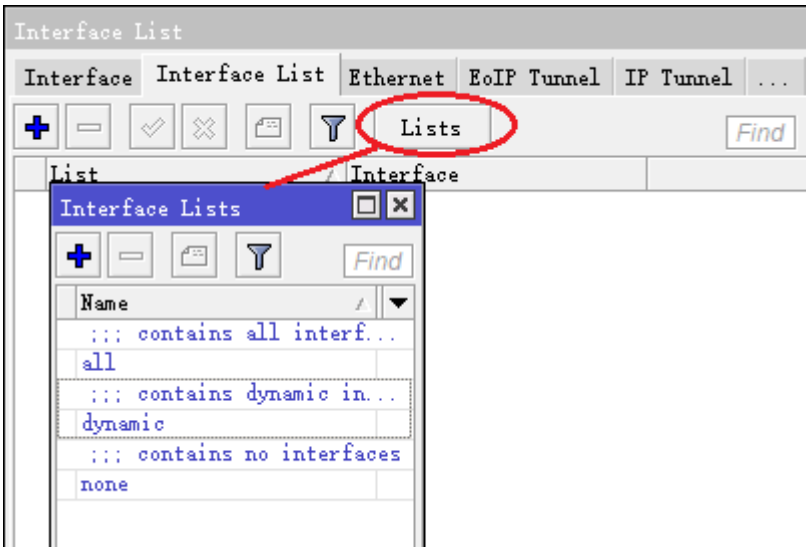
操作路径： /interface list

List（列表）

在该菜单下包含了路由器当前的所有接口，有三种默认类型用于接口列表类型创建：

- all – 包含所有接口
- none – 不包含任何接口
- dynamic – 包含动态接口

在 winbox 中可以看到三个默认类型：



描述	属性
name (*字符*)	接口列表名称

include (*字符*)	定义接口列表包含的接口成员，在命令行可以通过逗号添加多个列表
exclude (*字符*)	定义接口列表排除的接口成员，在命令行可以通过逗号添加多个列表

在创建成员时，添加规则按照以下流程：

1. include 成员首先加入接口列表
2. exclude 成员其次排除出列表
3. 最后添加静态接口配置成员，静态接口列表非系统默认，即手动创建列表

Members（成员）

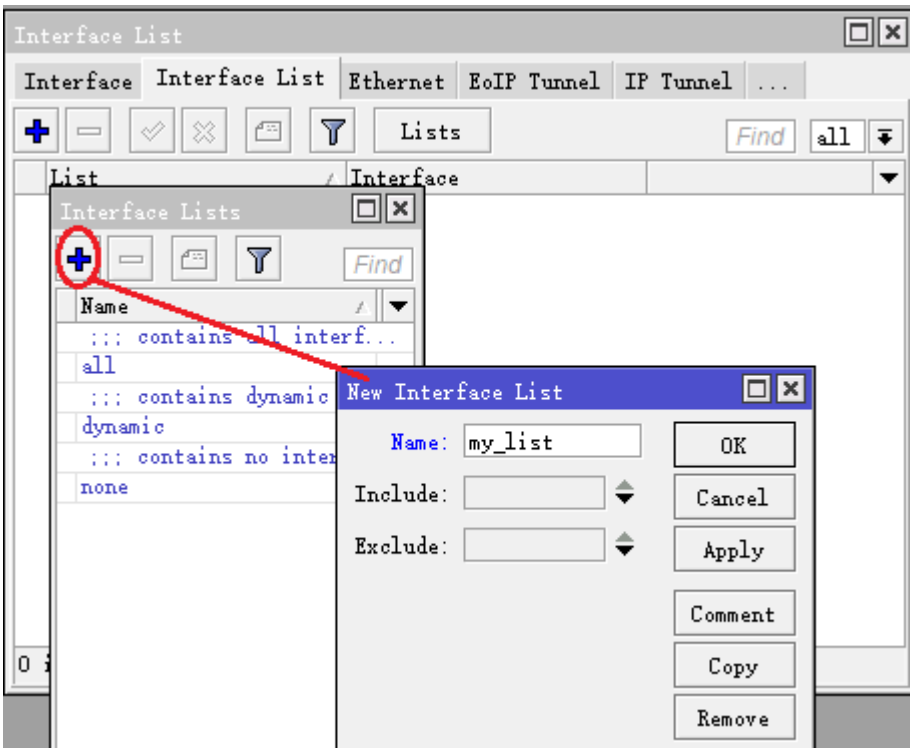
操作路径: /interface list member

在该子菜单下主要为手动创建列表的静态接口配置添加接口成员，在该菜单下不能添加动态接口，只能通过 list 菜单下的 include 和 exclude 设置

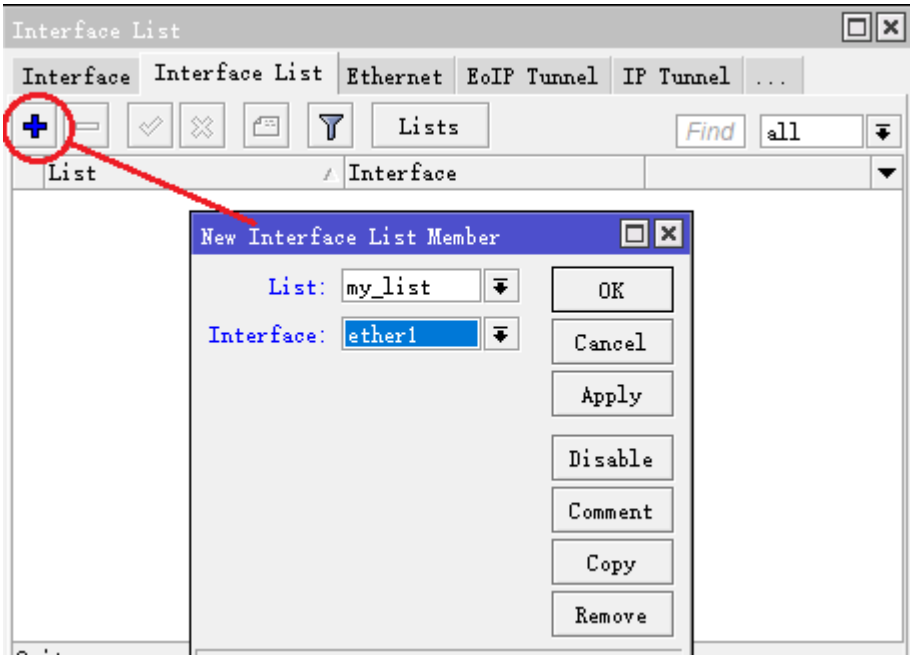
属性	描述
interface (*字符*)	接口名称
list (*字符*)	接口列表名称

创建静态列表事例

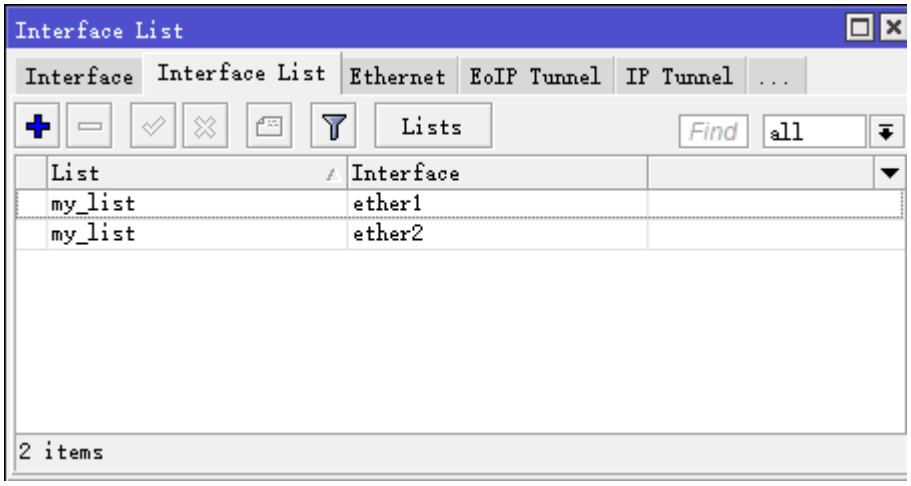
在 winbox 中，点击加号，创建 my_list 的静态接口列表



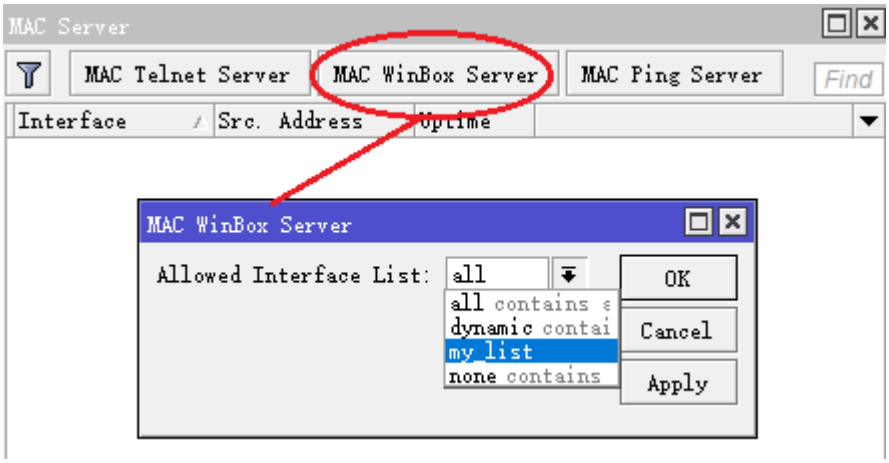
在后在 interface list 菜单下，添加需要归入 my_list 列表名单的接口：



将 ether1 和 ether2 加入 my_list 列表中：



这里把 my_list 应用到 MAC server 中，限制 winbox mac 地址登录接口，进入 tool mac-server 菜单配置：



4.4 3G 接口扩展设置

RouterOS 可以支持 3G 网络接入，这里介绍联通 WCDMA 和电信 CDMA 接入的 3G 网络连接，首先我们要准备一个电信或联通的 3G 上网卡，一般选择 USB 接口，将 USB 接口的 3G 卡插入 PC 的 USB， MikroTik RouterBOARD 支持 USB 的设备包括 RB433UAH、RB453G、RB411U、RB411UAHR、RB750U、RB751、RB2011、CCR1016 等，支持 RouterOS V5.0. 同样能工作在之前的 2.9 和 3.0 版本，只是配置有点变化

首先 USB 上网卡能被 RouterOS 识别，你可以通过在系统资源（system resource）里的 USB 查看是否找到 USB 网卡驱动

```
[admin@rb411u] > /system resource usb print
 # DEVICE VENDOR                    NAME                    SPEED
 0 2:1                              RB400 EHCI              480 Mbps
 1 1:1                              RB400 OHCI              12 Mbps
 2 1:3      Option N.V.             Globetrotter HSDPA... 12 Mbps
[admin@rb411u] >
```

确定 USB 端口在 Port 列表下找到：

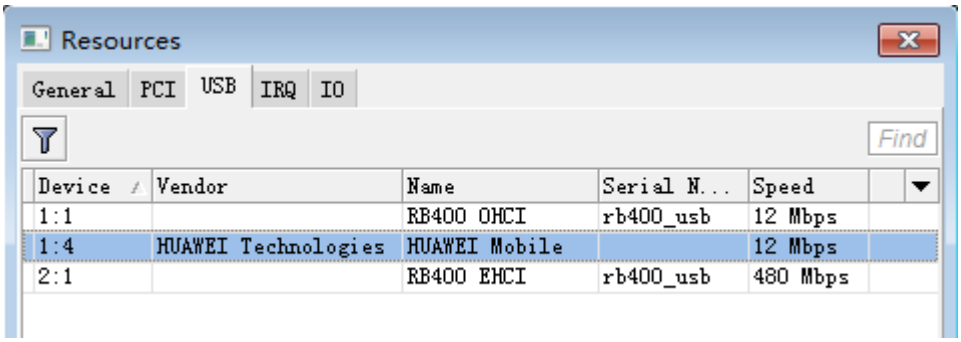
```
[admin@rb411u] > /port print
Flags: I - inactive
 #    NAME                    CHANNELS    USED-BY                 BAUD-RATE
 0    serial0                 1           Serial Console          auto
 1    usb2                    3                                   9600
[admin@rb411u] >
```

RouterOS V3.23 之前网卡信息被全部显示在一个 port 上，从 v3.23 后一个 port 对于一个网卡信息，频道有 0，1，2 等代码

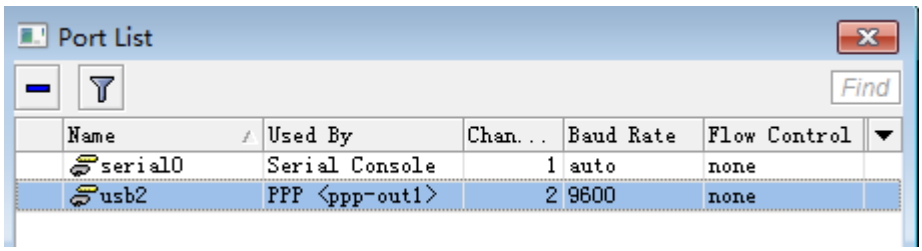
联通 3G 拨号测试

无线网卡型号：华为 3G USB 网卡 E220，基于联通的 WCDMA，测试主板：RB411U

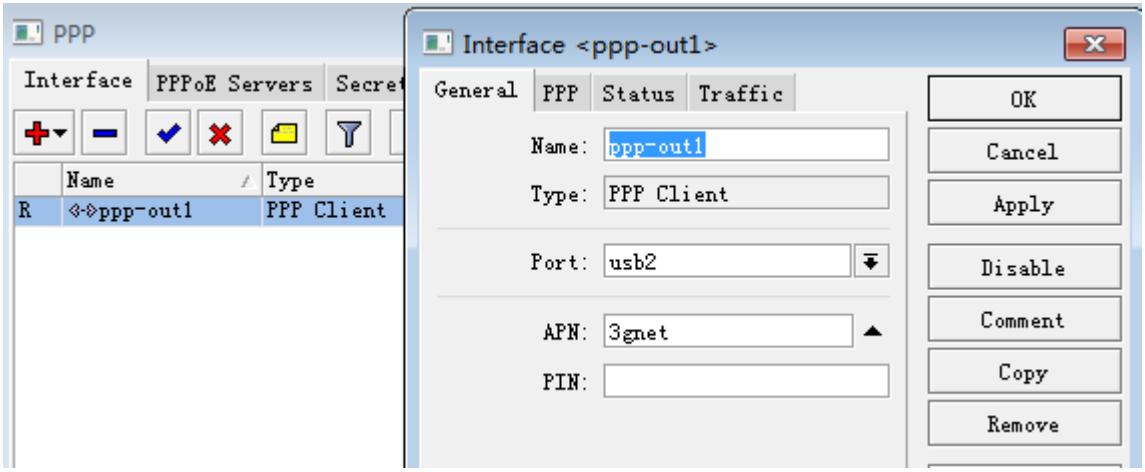
将华为的 USB 网卡，插入 RB411U 的 USB 接口，在 system resource 里可以找到网卡信息，如下显示的 USB 信息



然后我们进入 system port 里可以查看到一个拨号接口 usb2，参数预设即可，不用修改



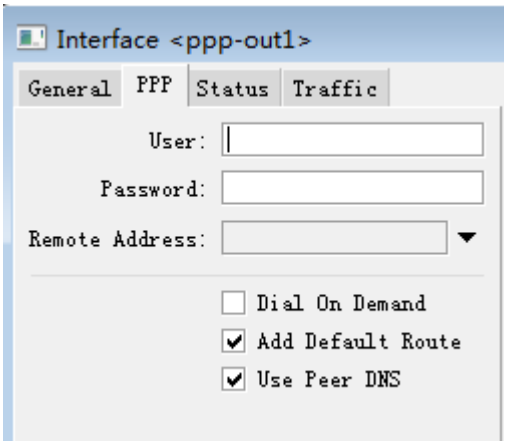
进入 ppp 目录下，添加一个 ppp-client，并设置 port 为 usb2，APN 设置为“3gnet”，这样 PPP-client 可拨号了



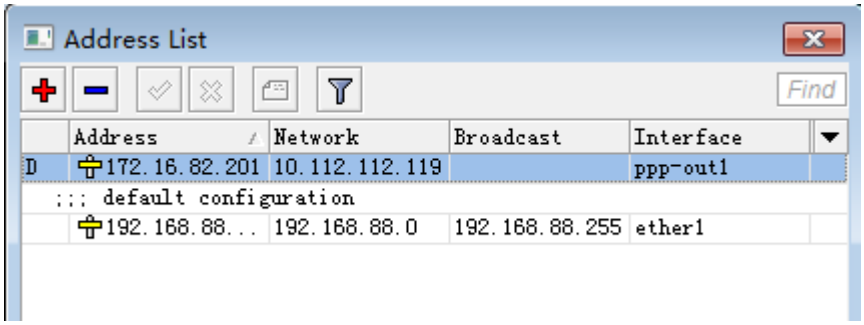
这里我们可以通过日志 log，查看拨号情况，显示如下：

```
Jan/02/1970 00:0...  async ppp info  ppp-out1: initializing modem...
Jan/02/1970 00:0...  async ppp info  ppp-out1: dialing out...
Jan/02/1970 00:0...  async ppp info  ppp-out1: authenticated
Jan/02/1970 00:0...  async ppp info  ppp-out1: could not determine remote address, using
                                     10.112.112.119
Jan/02/1970 00:0...  async ppp info  ppp-out1: connected
```

注： 我们要把 PPP-client 里的 Dial_On_Demand 关闭掉，这里没有账号密码



下面是获取到的 IP 地址

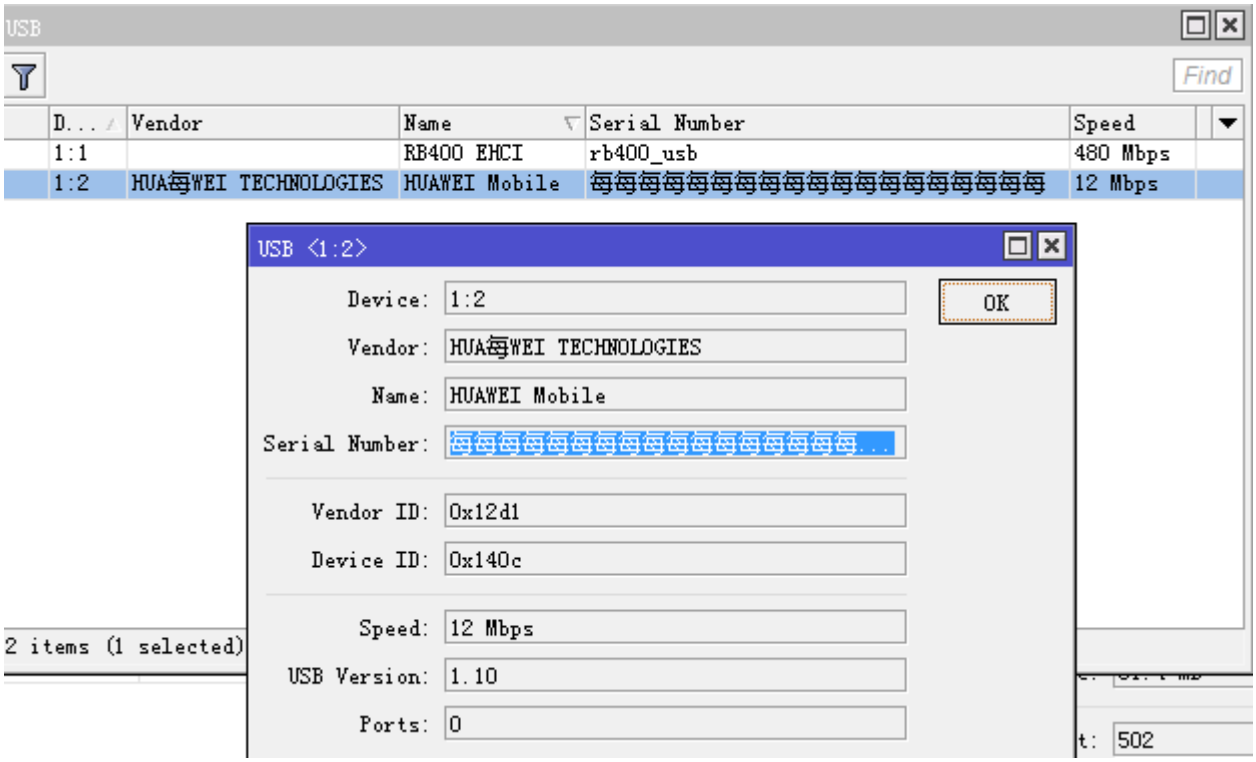


测试迅雷下载可以到 50-150KB 的范围波动，可能室内的原因，如果在室外或者信号好的地方带宽会相对稳定

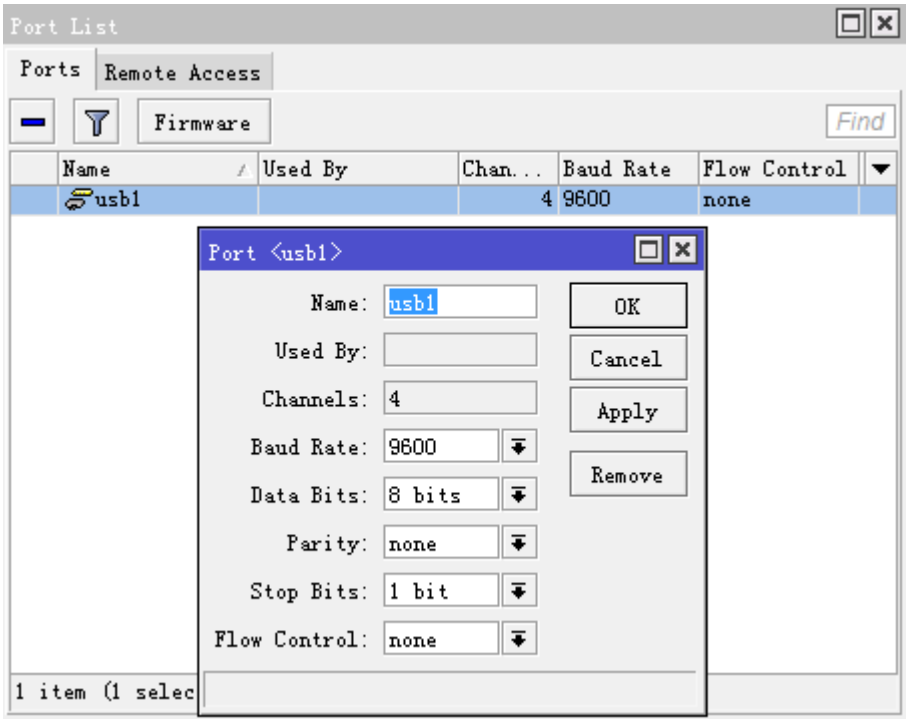
电信 CDMA 上网卡配置

配置电信 CDMA 3G 上网卡采用的是华为 E177 USB 接口，测试平台在 RB751-2HnD、x86 硬件和 CCR1016 上都可以识别，只是 USB 列表中显示会有乱码。

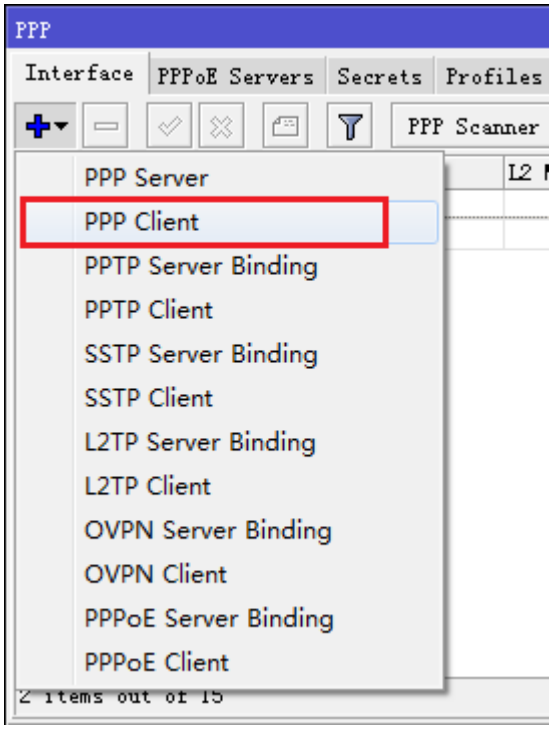
当将上 USB 网卡插到设备的 USB 接口后，我们可以在 system resource 中的 USB 菜单下找到设备，如下图，打开 USB 后看到，只不过我测试的这款华为 E177 在 USB 列表中显示错误，但并没有影响使用



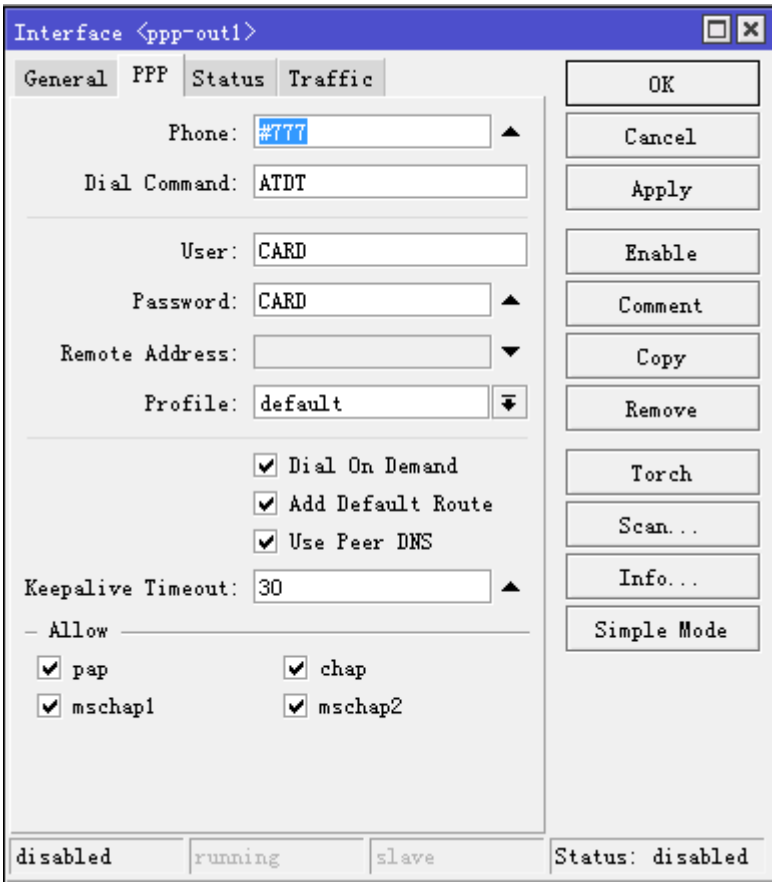
之后我们可以在 system port 中被识别为 usb1（这个根据设备接口情况具体识别），参数默认就可以，如下图



剩下的就是建立 ppp 拨号，进入 interface，点加号添加 PPP-client，



设置 ppp-out1 拨号，注意以下配置参数需要点击对话框的右侧“advanced mode”显示



配置仅需要设置 phone：#777，user 和 password 都设置为“CARD”，其他参数预设（Add Default Route 和 Use Peer DNS 需要选择上，自动获取网关和 DNS）。这样就可以拨号连接，当连接成功后 ppp-out1 会显示 R，切在 ip address 中获取 ip 地址。

4.5 LTE 客户端配置

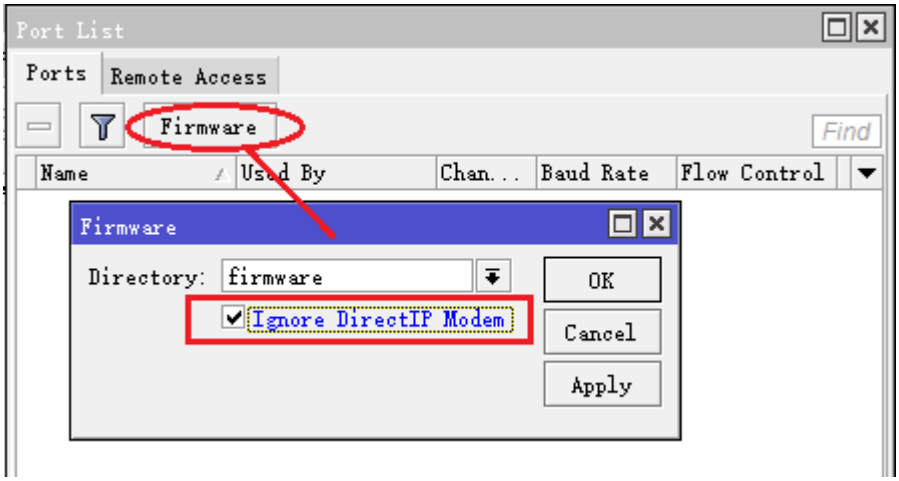
操作路径: /interface lte

RouterOS 支持当前的 LTE 4G 网络，国内三大运营商都能支持，主要根据 RouterOS 支持的 4G 模块，关于 RouterOS 对 4G 模块支持型号，部分可以参考官方资料：

https://wiki.mikrotik.com/wiki/Manual:Peripherals

当前支持 Direct-IP 模式类型的 4G 上网卡（QMI 在将来会支持），如果要启用 PPP 接口代替 LTE 接口，需要修改 Direct-IP 模式，需通过***/port firmware direct-ip-mode=no*** 命令操作，并重启 RouterOS，注意：使用 PPP 仿真模式，将无法得到本机 LTE 网卡最大传输速率。

在 Winbox 进入 system port 菜单



参数介绍

属性	描述
add-default-route (*yes* | *no*; 默认: **no**)	是否为 LTE 接口添加默认路由
authentication (*pap* | *chap*; 默认:)	选择用户验证的协议
apn (*string*; Default: "")	服务商的接入点名称（Access Point Name）
band (; 默认: "")	
comment (*字符*; 默认: "")	项目注释
default-route-distance (*整型*; 默认: **1**)	从 v6.2 开始，当 add-default-route 选择 yes，可以设置自动添加路由距离值
disabled (*yes* | *no*; 默认: **yes**)	是否禁用该 LTE 接口，CLI 下默认是 yes
mac-address (*MAC*; 默认: "")	当前 LTE 网卡的 MAC 地址
modem-init (*字符*; 默认: "")	Modem 初始化字符串

mtu (*整型*; 默认: **1500**)	最大传输单元，LTE 网卡能发送最大数据包长度
name (*字符*; 默认: **""**)	网卡名称描述
network-mode (*3g* | *auto* | *gsm* | *lte*; 默认: **auto**)	选择或强制 LTE 网卡工作方式
password (*string*; Default: **""**)	用户接入验证的密码
pin (*string*; Default: **""**)	SIM 卡的 PIN 代码
user (*string*; Default: **""**)	用户验证名称.

Scanner（扫描器）

可以通过***/interface lte scan*** 扫描 LTE 网卡，扫描有如下只读属性：

属性	描述
duration (*整型*)	持续扫描时间
freeze-frame-interval ()	数据显示时间间隔
number (*整型*)	接口编号

可通过 /interface lte info 命令，可查看 LTE 网卡获取相应参数

参考配置

设置正确的参数到 LTE 网卡（由网络运营商提供）：

```
/interface lte
set [ find ] apn=phoneprovider.net authentication=chap name=lte1 network-mode=auto password=web
user=web
```

添加 DHCP 客户端配置到 LTE 网卡：

```
/ip dhcp-client
add default-route-distance=1 disabled=no interface=lte1
```

配置对 RouterOS 下的内网做 nat 转换访问互联网：

```
/ip firewall nat
add action=masquerade chain=srcnat out-interface=lte1
```

当以上配置添加成功后，可以通过 info 命令查看 LTE 网卡参数（参数获取要求 LTE 硬件支持）：

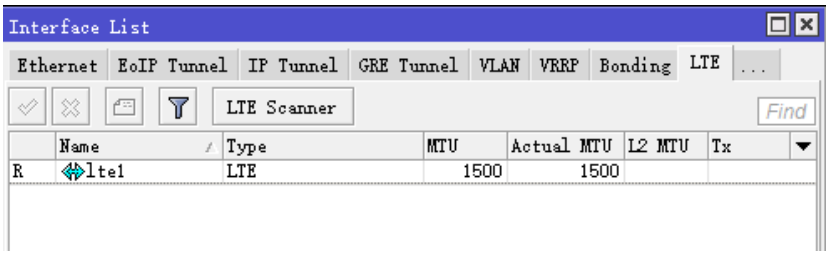
```
[admin@MikroTik] > /interface lte info lte1 once
status: call in progress
pin-status: no password required
```

```
functionality: full
manufacturer: Huawei Technologies Co., Ltd.
model: ME909u-521
revision: 12.631.07.01.00
current-operator: vodafone ES
current-cellid: 44436007
access-technology: Evolved 3G (LTE)
signal-strengh: -79 dBm
frame-error-rate: n/a
earfcn: n/a
imei: 860461024123456
imsi: 234012555034981
uicc: n/a
rssi: -79dBm
rsrp: -109dBm
rsrq: -13dB
sinr: -1dB
[admin@MikroTik] >
```

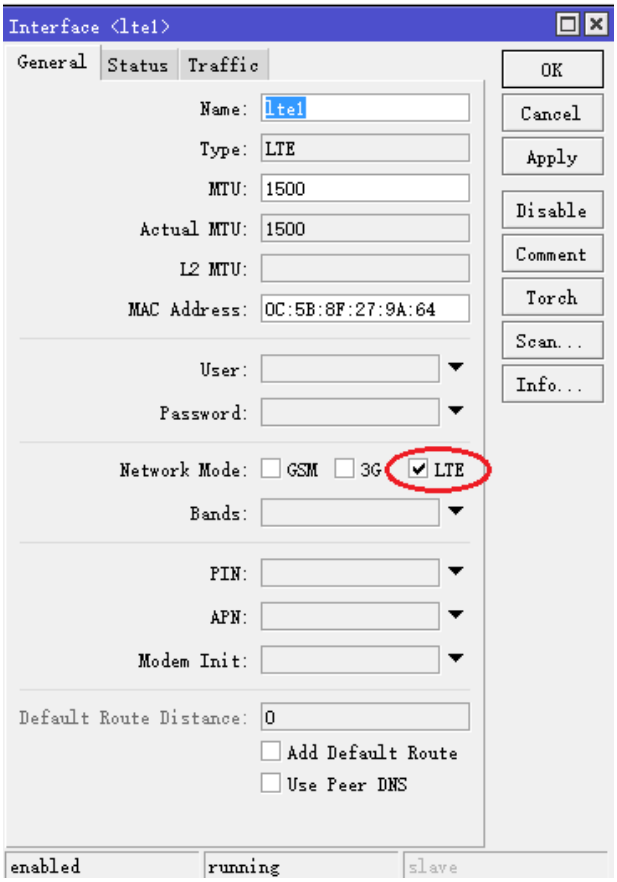
Huawei E3372h-153 电信 4G 测试

Huawei E3372h-153 是 USB 接口，支持电信 4G 或联通 3G/4G，测试平台 RB951Ui-2HnD，能正常识别，因此在其他 RB 或者 x86 平台的 RouterOS 都是可以支持的。

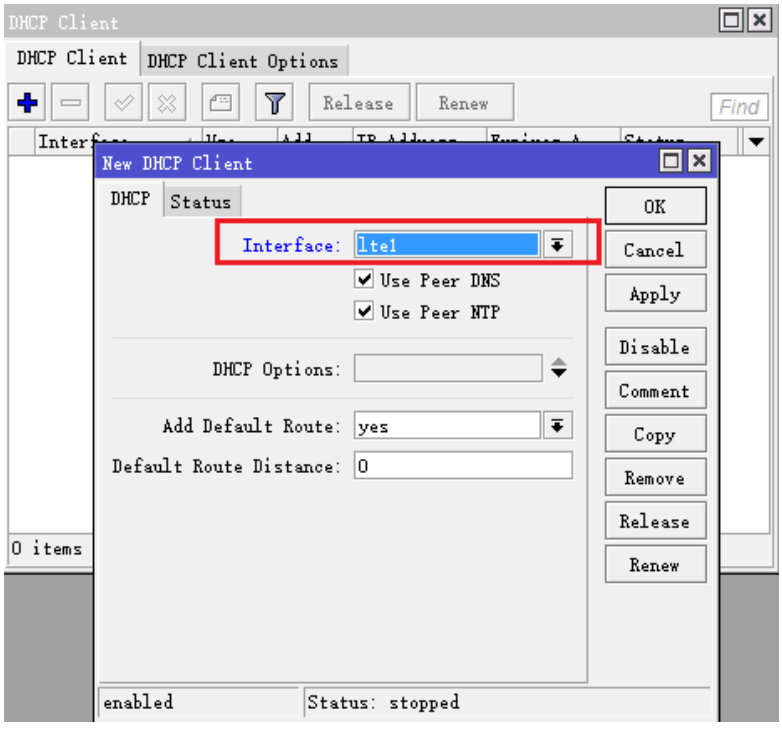
首先将电信 SIM 卡插入到 E3372h-153，然后将 E3372h-153 安装到 RB951Ui-2HnD 的 USB 接口，打开 Winbox，进入 interface-> LTE 菜单， RouterOS 自动识别到 E3372h-153，并显示 lte1，因此不需要做任何设置自动接入电信网络，在 lte1 状态显示为“R”



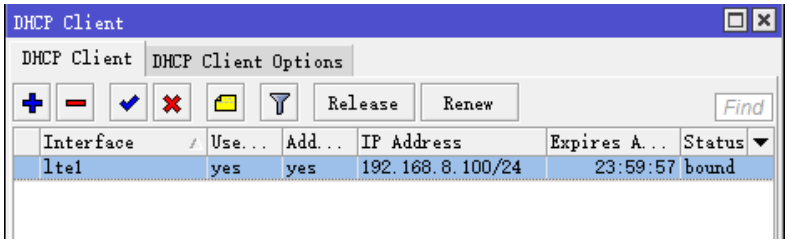
双击打开 lte1，接入的是 FDD 4G 网络，因此选择 network-mode 为 LTE



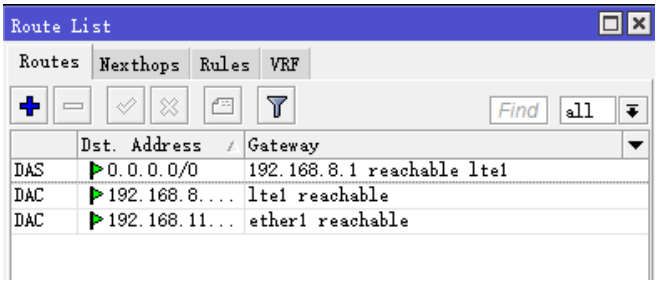
配置完成后，进入 ip->DHCP-client 创建 DHCP 客户端，选择上 use peer dns 和 add defaulte route=yes，操作如下：



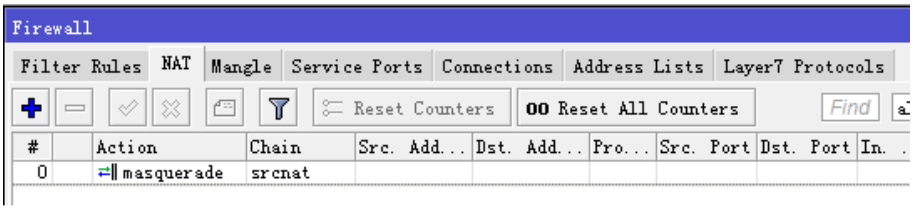
添加完成后，lte1 接口自动获取到 IP 地址，状态如下：



进入 ip route 查看路由，默认路由 192.168.8.1，已经自动添加



在确保内网 IP 地址配置完成后，进入 ip firewall nat 创建 masquerade 的 nat 规则，确保内网 IP 地址上网



剩下内网口配置省略，根据网络情况操作

下面是 RouterOS 下面的 windows 电脑测试 dns，说明已经正确接入 4G 网络

```
C:\Users\yus>ping 61.139.2.69

正在 Ping 61.139.2.69 具有 32 字节的数据:
来自 61.139.2.69 的回复: 字节=32 时间=61ms TTL=54
来自 61.139.2.69 的回复: 字节=32 时间=26ms TTL=54
来自 61.139.2.69 的回复: 字节=32 时间=36ms TTL=54

61.139.2.69 的 Ping 统计信息:
    数据包: 已发送 = 3，已接收 = 3，丢失 = 0 (0% 丢失)，
往返行程的估计时间(以毫秒为单位):
最短 = 26ms，最长 = 61ms，平均 = 41ms
```

测试版本是 RouterOS v6.38，Huawei E3372h-153 无法通过 info 命令返回相关参数，可能是 RouterOS 软件驱动问题。

Passthrough（透传）

从 RouterOSv6.41 开始，RouterOS 将支持 LTE 适配器 Passthrough 功能（需要 LTE 适配器支持），即 IP 地址直接应用路由器下面的客户端设备。在这样的配置下路由器将不会从 LTE 适配器获取 IP 地址， LTE 适配器

Passthrough 功能如果适配器支持，能透传 IPv4 和 IPv6 地址。 部分适配器支持多 APN，你可以透传不同的 APN 到指定的路由器接口。

Passthrough 仅能工作一个主机上，路由器会自动探测第一次接收到的数据包 MAC 地址，并作为 Passthrough 主机，如果多个主机在 Passthrough 接口下，可以通过指定 MAC 地址给 Passthrough 锁定主机。

事例：配置 ether1 为 Passthrough 接口：

```
[admin@MikroTik] > /interface lte set lte1 apn=apn1/ether1
```

配置 Passthrough 到 ether1 的主机 MAC 地址 00:0C:42:03:06:AB

```
[admin@MikroTik] > /interface lte set lte1 apn=apn1/ether1/00:0C:42:03:06:AB
```

配置多 APN 到 ether1 和 ether2：

```
[admin@MikroTik] > /interface lte set lte1 apn=apn1/ether1,apn2/ether2
```

4.6 GPS

RouterOS 通过扩展 GPS 接收器，支持 GPS 定位功能，需要单独安装 gps 软件包，安装完成后，可以在 system 菜单下找到 gps 选项

操作路径：/system gps

属性	描述
enabled (*yes* | *no*; 默认: **no**)	是否启用 GPS
port (*字符*; Default:)	对应 GPS 接收器的端口名称（USB/Serial）
channel (*整型 [0..4294967295]*; 默认: **0**)	设备使用的端口通道
init-channel (*整型[0..4294967295]*; 默认:)	频道初始化字符串
init-string (*字符*; 默认:)	GPS 初始化 AT 字符串
set-system-time (*yes* | *no*; 默认: **no**)	是否设置路由器的日期和时间从 GPS 接收

状态监控

操作路径: /system gps monitor

该命令可以查看 GPS 接收器的数据信息，包括经纬度坐标、速度和当前连接卫星数量等

属性描述

属性	描述

Altitude（*integer*）	海拔高度
date-and-time (*date*)	接收到 GPS 的日期和时间
latitude (*none* | *string*)	维度，度分秒格式
longitude (*none* | *string*)	经度，度分秒格式
speed (*none* | *string*)	GPS 测量当前移动速度
bearing (*none* | *string*)	GPS 所指向的罗盘方向
valid (*yes* | *no*)	
satellites (*integer*)	接收到的卫星数量

RouterOS 支持 GPS 软件功能包，支持包括 GPS、中国北斗、欧洲伽利略和俄罗斯格林纳斯，RB 设备集成 GPS 接收器的设备包括 LtAP、LtAP mini、KNOT 等设备，不集成 GPS 接收器的 RB 和 x86 设备可以使用 USB 扩展 GPS 模块。

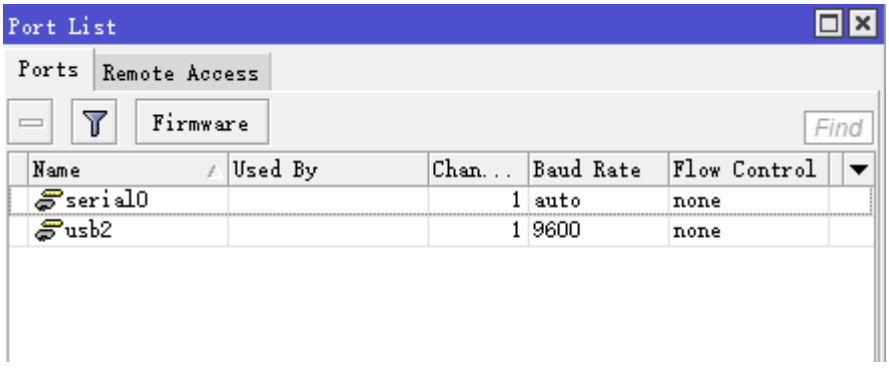
USB GPS 接收器

该实例介绍如何给未集成 GPS 接收器的 RouterOS 设备添加 USB 接口的 GPS 接收器。

首先需要将 USB GPS 接收器与 RouterOS 硬件的 USB 接口连接，连接完成后，在/port 菜单下查看是否识别到，查看端口使用状态：

```
[admin@MikroTik] /port> print
Flags: I - inactive
 #    NAME                CHANNELS USED-BY                BAUD-RATE
 0      serial0                  1                             auto
 1      usb2                     1                             9600
```

Winbox 打开路径是在 system port 下面



在 GPS 接收器配置前，需要先禁用掉 console 下的 usb2 端口，避免被 console 占用，

```
[admin@MikroTik] > /system console print
Flags: X - disabled, U - used, F - free
 #    PORT                                                    TERM
```

```
 0 U serial0                                                                vt102
  1 U usb2                                                                 vt102
[admin@MikroTik] > /system console disable 1
```

调整 USB 的 GPS 接收器设置，下面是 usb2 端口的 GPS 设置配置，注意部分 GPS 接收器有所不同，下面提供两种参考配置，以正常接收到 GPS 信号为准。

配置 1：波特率为 4800，奇偶效验：odd

```
[admin@MikroTik] /port> set 0 baud-rate=4800 parity=odd
[admin@MikroTik] /port> print detail
Flags: I - inactive
 0    name="usb2" used-by="" channels=1 baud-rate=4800 data-bits=8 parity=odd
stop-bits=1 flow-control=none
```

配置 2：波特率为 4800，不设置奇偶效验

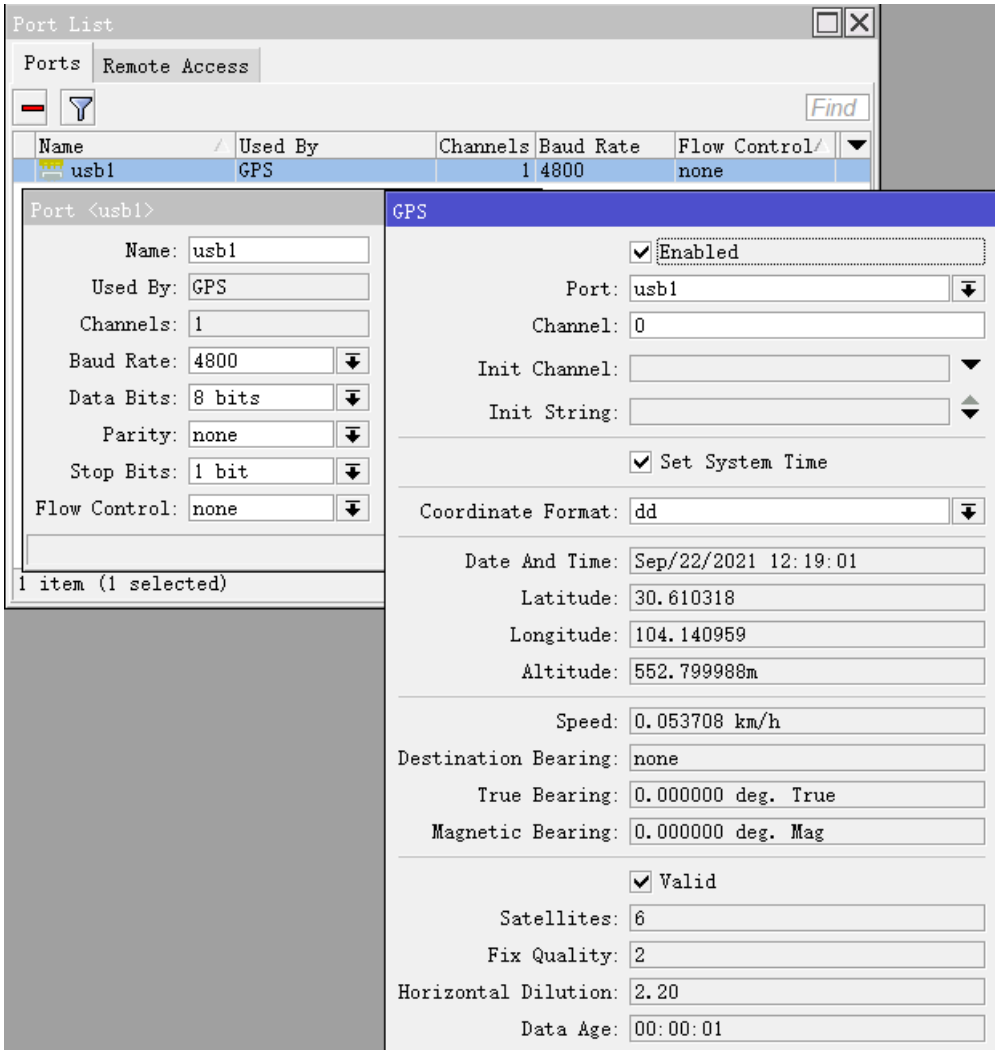
```
[admin@MikroTik] /port> set 0 baud-rate=4800
[admin@MikroTik] /port> print detail
Flags: I - inactive
 0    name="usb2" used-by="" channels=1 baud-rate=4800 data-bits=8 parity= none
stop-bits=1 flow-control=none
```

在配置完 port 菜单下的 usb2 后，启用 GPS

```
[admin@MikroTik] /system gps> set enable=yes port=usb2
[admin@MikroTik] /system gps> print
           enabled: yes
              port: usb1
           channel: 0
      init-channel: 0
       init-string:
  set-system-time: no
```

注意：如果要修改 port 下 usb2 的参数，请先禁用 GPS 功能（/system gps set enable=no）

Winbox 配置情况：



使用命令查看 GPS 接收状态

```
[admin@MikroTik] /system gps> monitor
  date-and-time: jul/02/2015 14:36:08
               latitude: N 56 57' 32.262''
              longitude: E 24 9' 7.262''
               altitude: 36.700001m
                  speed: 0.000000 km/h
  destination-bearing: none
          true-bearing: 0.000000 deg. True
     magnetic-bearing: 0.000000 deg. Mag
                  valid: yes
             satellites: 7
```

ltAP mini 的 GPS 配置

ltAP mini 的 GPS 模块是集成的，因此配置与 USB 外接 GPS 接收器有所区别，第一步操作仍然先禁用掉 serial0 端口。

```
[admin@MikroTik] > /system console print
Flags: X - disabled, U - used, F - free
```

```
#    PORT                                                              TERM
0 U serial0                                                           vt102

[admin@MikroTik] > /system console disable 0
```

第二步，port 下的配置有所不同，即配置默认就可以，无需其他设置

```
[admin@MikroTik] /port> print detail
Flags: I - inactive
 0    name="serial0" used-by="" device="" channels=1 baud-rate=auto data-bits=8
parity=none   stop-bits=1 flow-control=none
```

启用 GPS

```
[admin@MikroTik] /system gps> set enable=yes port=serial0
[admin@MikroTik] /system gps> print
            enabled: yes
               port: serial0
            channel: 0
       init-channel: 0
        init-string:
  set-system-time: no
```

ltAP mini 内置了 GPS，仅提供了一个 ufl 外接天线接口，因此无法有效接收到 GPS 信号，需要外接天线，当你安装好外接天线后，在 RouterOS 的 GPS 菜单下选择 gps-antenna-select=external

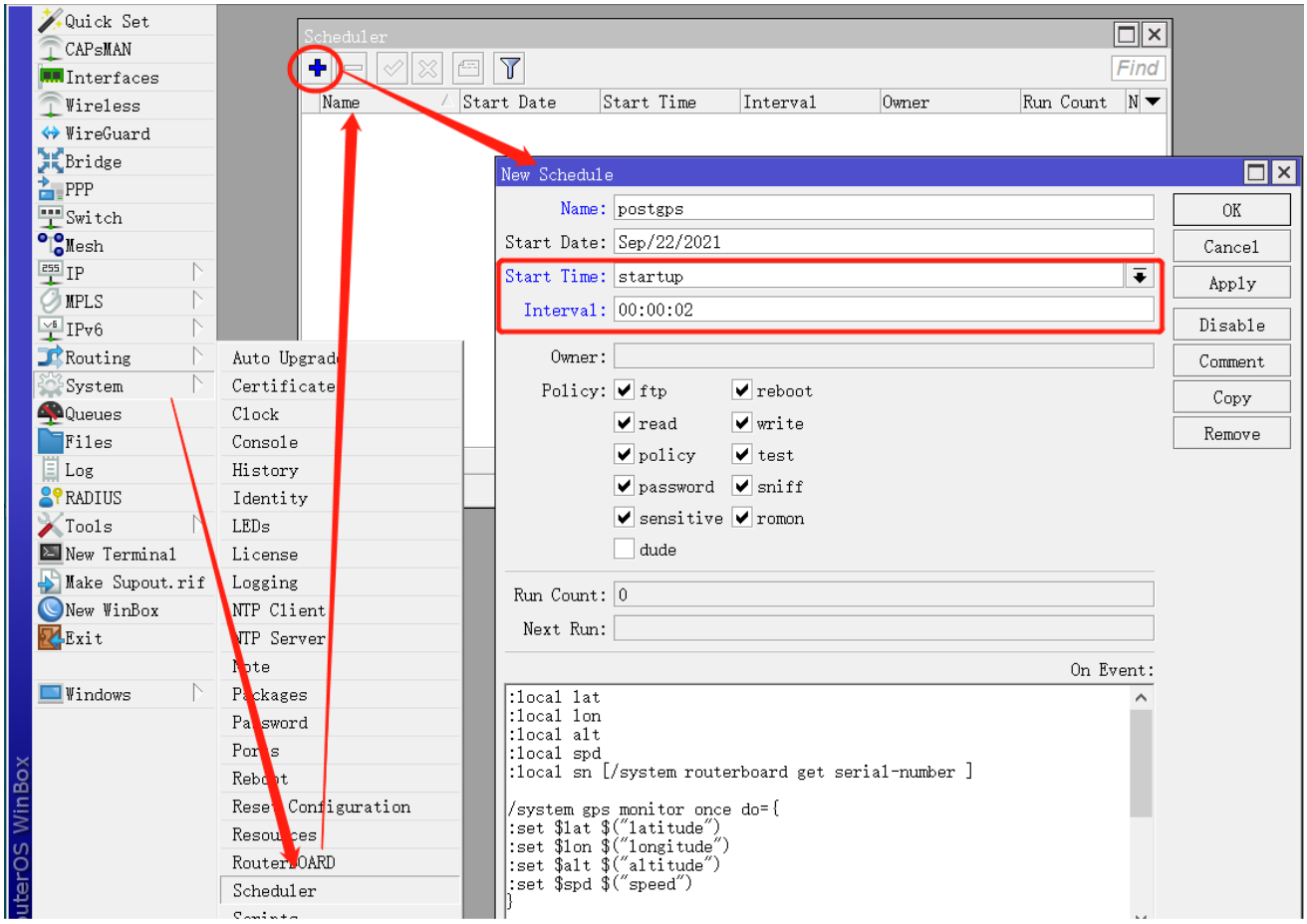
```
[admin@MikroTik] > /system gps set gps-antenna-select=external
```

GPS 位置上传

MikroTik 提供 GPS 在线跟踪方案，通过脚本提交 GPS 定位信息，可以通过脚本向 http 页面提交 GPS 信息，通过计划任务每间隔 2-3 秒执行一次脚本，即定期发送一次位置信息，提交脚本的时间间隔可根据实际情况在计划任务中调整。

```
{
:local lat
:local lon
/system gps monitor once do={
:set $lat $("latitude")
:set $lon $("longitude")
}
tool fetch mode=http url="http://YOURSERVER.com/index.php" port=80 http-method=post
http-data=("{\"lat\":\"" . $lat . "\",\"lon\":\"" . $lon . "\"}")
http-content-type="application/json"
:put ("{\"lat\":\"" . $lat . "\",\"lon\":\"" . $lon . "\"}")
}
```

Schedule 计划任务配置：



车载或户外移动上传 GPS 信息，可以通过连接的 WiFi 或者 3G/4G 网络上传到服务器。

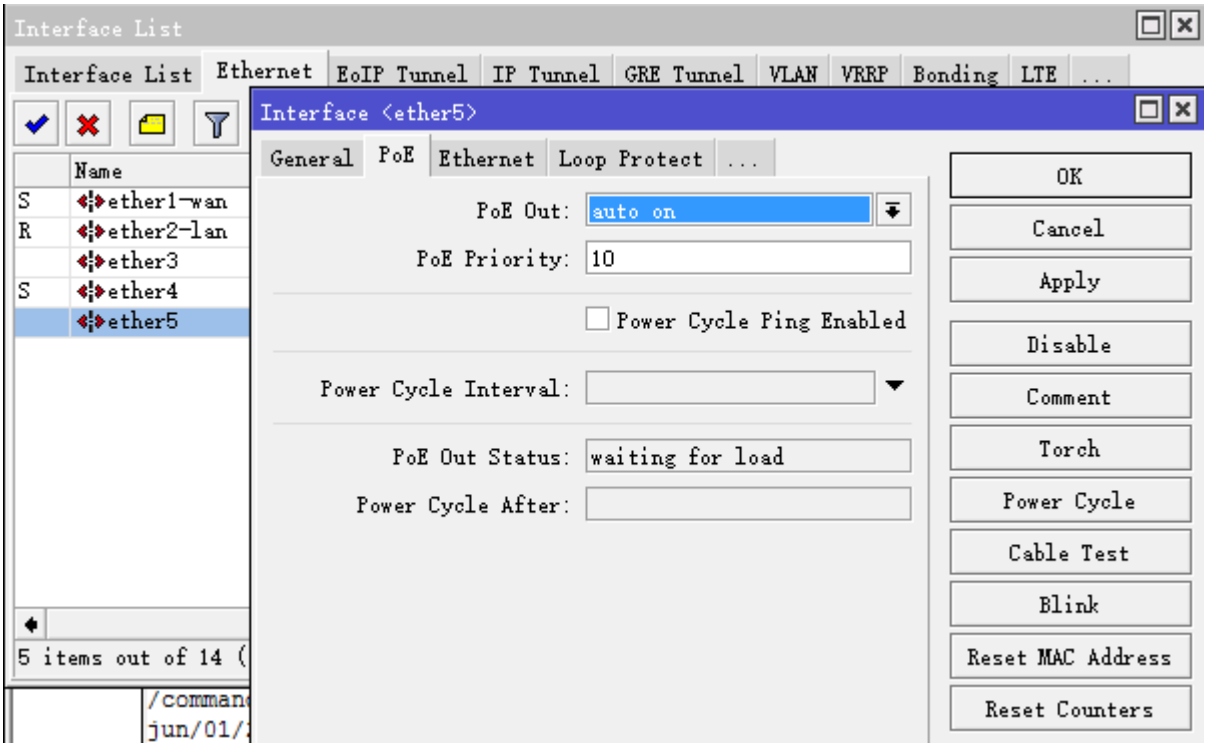
4.7 PoE-Out（PoE 供电）

该章节介绍 POE-out 功能如何在支持 POE-out 的 RouterBOARD 上使用，例如 RB750UP 、RB951Ui-2HnD 和 OmniTik UPA-5HnD. 这些路由器提供了 PoE 电源输出。在将来这个功能可以在更多设备上获得，例如 SwOS，且会使用新的 PoE-out 控制器固件

注意： 当升级最新的 PoE-out 控制器固件版本，将不能在降级到之前的版本

PoE 端口设置

PoE Out 在命令行下进入***/interface ethernet poe*** 目录下配置，winbox 则直接进入网络接口的 PoE 菜单配置，v6.10 前的配置将不在多讲。



从 RouterOS 6.10 开始，提供了以下的配置参数：

属性	描述
poe-out (默认: *auto-on*)	• ***auto-on*** – 主板会自动探测当前端口情况，根据接入线缆是否在范围内，匹配后启用 PoE 输出情况，电源接通电阻范围应在 3kΩ 至 26.5kΩ。 • ***forced-on*** – 取消自动探测，即 PoE 供电一直输出 • ***off*** – 关闭该端口的自动探测和电源输出
poe-priority (*0 .. 99, 默认: 10*)	针对多 PoE 端口的优先级设置。最高优先级为 0，最低为 99，如果有两个或以上端口有相同优先级，越小的端口编号将获得最高优先级，例如 ether2 和 ether3 具备相同优先级，当电流超过最大值，ether3 将会被关闭。每 6 秒 PoE 端口将探测是否能为关闭的端口提供电源输出。
monitor	显示指定端口 PoE-out 状态

RouterOS v6.10 前设置，以及 SwOS 的 PoE out 配置

老的 firmware 支持以下配置功能：

- **auto-on** – 主板会自动探测当前端口情况，根据接入线缆是否在范围内，匹配后启用 PoE 输出情况，电源接通电阻范围应在 3kΩ 至 26.5kΩ。
- **forced-on** – 取消自动探测，即 PoE 供电一直输出
- **off** – 关闭该端口的自动探测和电源输出

如何工作

auto-on 模式

选择 auto-on，PoE 输出端口的操作顺序如下：

- 用低电压测试连接端口的电阻，如果探测到电阻在 3kΩ ～26.5kΩ 范围，供电会自动打开；
- 当没有检查到过载或短路发生，PoE 端口将持续供电
- 当网线被拔出，PoE 端口回到探测状态，并保持关闭状态，直到该端口 PoE 状态被探测到

forced-on 模式

选择 forced-on ，PoE 输出端口的操作顺序如下

- 检测电阻范围在 0Ω ～∞Ω，即使网线被拔出供电也会输出
- 当没有检查到过载或短路发生，PoE 端口将持续供电
- 当网线被拔出，该端口仍然会供电

off 模式

选择 off ，PoE 输出端口将关闭探测和供电，仅作为普通以太网端口使用

Safety（安全）

PoE 输出有以下安全特点：

端口检查机制

auto-on 模式下被认为是相对安全的，首先将检查电阻范围在 3kΩ～26.5kΩ 后，才会启用 PoE 输出

过载保护

当 PoE 输出启用，该端口检测到电流过载，未避免硬件损坏，PoE 输出将会关闭，新的固件能在过载保护发生后 120 秒，重新开启 PoE 输出，如果外部连接设备恢复正常将继续供电。

提示: PoE-Out 固件版本 2.0，允许 1A 电流输出，所有端口最大支持 2.2A 电流

短路探测

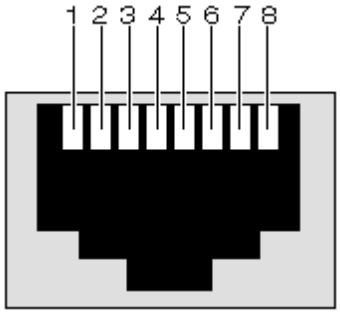
当 PoE 输出端口检查到短路发生，为确保不会产生设备损坏，正在供电的所有 PoE 端口将关闭

PoE Out 线序

默认的 PoE 输出正负极在 MikroTik 设备的 100MBps 接口采用标准协议

标准线序

PoE 输出在 RouterBOARD 设备使用标准为：蓝色组为正极，棕色组为负极



Mikrotik PoE 标准：(4,5pin +) (7,8pin -)

相反线序

有一些设备使用不同的正负极，这些设备与标准相反，蓝色组为负极，棕色组为正极

线序处理

如果你的设备采用的是与标准相反的正负极，并使用 MikroTik 设备供电（如 RB750UP 或 OmniTik UPA）你必须交换网线线序

网线一端采用标准的 RJ45 T568B 线序（橙白，橙，绿白，蓝，蓝白，绿，棕白，棕），另外一端将交换蓝色组和棕色组线序（橙白，橙，绿白，棕，棕白，绿，蓝白，蓝）

因此根据设备 PoE 手册配置网线线序，MikroTik 设备支持标准的 PoE 线序。

升级 PoE Out 固件

PoE-Out 固件在新的 RouterOS v6 版本不在提供独立的固件升级，v6 版本不在有 PoE 固件升级命令，RouterOS 升级降级将会自动完成 PoE 固件配置。

注意：RouterOS v5.24 以前版本有固件升级的命令操作，大致流程如下：

- 升级 RouterOS 到最新版本，v5 版本范围
- 检查当前固件版本使用命令 /interface ethernet poe settings print
- 使用升级命令/interface ethernet poe settings upgrade

升级命令执行后出现以下提示

```
Do you really want to upgrade PoE firmware? [y/n]
```

重启路由器，固件替换生效

```
Please reboot to finish PoE firmware upgrade.
```

以上为 v5.20 的操作

PoE Out 事例

PoE-Out 固件从 2.0 版本开始运行设置端口优先级，支持接口的 PoE 输出设备而言，在电流过大情况下优先保护级别高的设备供电，优先级为 0 代表优先级最高，数字越大越低，当电流过大后，优先级低的端口将会被关闭掉

配置优先级

下面的事例是如何配置 RouterBOARD 的 PoE 优先级：

```
/interface ethernet poe set ether2 poe-priority=10
/interface ethernet poe set ether3 poe-priority=11
/interface ethernet poe set ether4 poe-priority=12
/interface ethernet poe set ether5 poe-priority=13
```

以上配置 ether2 优先级最高，最低为 ether5，但供电过载被监测到，ether5 供电将会被关闭，每 6 秒重新检测接口是否过载，如果仍过载将继续关闭，如果关闭 ether5 接口供电仍然超过限制，ether4 供电也将会关闭。

相同优先级

当所有端口 PoE 优先级相同，低端口编号的优先级最高，因此在下面的事例中，如果出现过载，ether5 会被先关闭

```
/interface ethernet poe set ether2 poe-priority=10
/interface ethernet poe set ether3 poe-priority=10
/interface ethernet poe set ether4 poe-priority=10
/interface ethernet poe set ether5 poe-priority=10
```

监控 poe-out 状态

所有 PoE-out 状态的监控，可以通过命令/interface ethernet poe monitor <interface> 完成，如下：

```
[admin@MikroTik] > interface ethernet poe monitor [find]
                  name: ether2 ether3 ether4 ether5
   poe-out-voltage: 23.2V   23.2V   23.2V
   poe-out-current: 224mA   116mA   64mA
     poe-out-power: 5.1W    2.6W    1.4W
```

这里是 RB750UP 为 3 个设备的供电状态显示

PoE 固件 v 1.x 和 v 2.0 区别

功能	version 1.x	version 2.0
端口最大电流	500mA	1A
设备输出总电流	2A	2.2A
poe 优先级	不支持	支持
状态监控	不支持	支持
PoE 长距离网线模式	不支持	支持
检查 PoE 固件版本	较新 RouterOS	支持

第五章 IP 与 ARP 配置

这里将介绍 IP 地址配置和地址解析协议 ARP，基于 TCP/IP 协议使用 IP 地址连接其它网络设备，并借助于地址解析协议（ARP）与同一子网的三层设备通信。

功能规格

需要功能包：***system***
需要等级：*Level1*
操作路径：***/ip address***, ***/ip arp***

5.1 IP 地址

操作路径：***/ip address***

IP 地址是指互联网协议地址（Internet Protocol Address）, IP 地址是 IP 协议提供的一种统一的地址格式，它为互联网上的每一个网络和每一台主机分配一个逻辑地址。

IP 协议就是使用这个地址在主机之间传递信息，这是 Internet 能够运行的基础。IP 地址的长度为 32 位，分为 4 段，每段 8 位，用十进制数字表示，每段数字范围为 0～255，段与段之间用句点隔开。一个完整的 IP 地址选址需要子网掩码配置，子网掩码区分了不同网段的 IP 地址。

RouterOS 能在一个接口上添加多个 IP 地址，如果当桥接模式在两个物理接口间被配置，该物理接口上添加 IP 地址并不是必须的（从 RouterOS 的 2.8 版本起），即使在物理接口上配置 IP 地址，在桥接模式状态下，IP 地址将属于桥接口。通过/ip address print detail 查看地址归属的接口。

MikroTik RouterOS 有下面的地址类型：

- **Static** – 管理员手动分配 IP 地址到指定接口
- **Dynamic** – DHCP，ppp，ppptp 或 pppoe 等协议连接后，自动分配的 IP

属性描述

address (*IP 地址*) – 主机的 IP 地址，组成 X.X.X.X/子网掩码
broadcast (*IP 地址*; 默认: **255.255.255.255**) – 广播 IP 地址，通过默认 IP 地址和子网掩码自动计算出的
disabled (yes | no; 默认: **no**) – 指定那一个 IP 地址禁用或启用
interface (*名称*) – 接口名称
actual-interface (只读: *名称*) – 仅适用于逻辑或虚拟接口，像桥（bridges）或隧道（tunnels）
netmask (*IP 地址*; 默认: **0.0.0.0**) – 指明网络地址，属于一个 IP 地址的一部份。
network (*IP 地址*; 默认: **0.0.0.0**) – IP 地址网段。点对点连接时，网段到远程地址结束。

注：不能在同一台路由器上设置相同子网段的不同 IP 地址，例如：10.0.0.1/24 地址分配到 ether1 接口上，并且 10.0.0.132/24 地址分配到 ether2 接口上，这样是非法的，这两个地址属于同一个网段 10.0.0.0/24。因为路由器的解释是连接两个或两个以上不同网段的设备。但允许在一个接口上配置相同子网段的多个 IP 地址，即 10.0.0.1/24 和 10.0.0.132/24 可以同时配置到 ether1 接口上，或者启用 proxy-arp。

例如：添加 IP 地址 10.10.10.1/24 到 ether2 接口上

```
[admin@MikroTik] ip address> add address=10.10.10.1/24 interface=ether2
[admin@MikroTik] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #    ADDRESS            NETWORK         BROADCAST       INTERFACE
  0    2.2.2.1/24         2.2.2.0         2.2.2.255       ether2
  1    10.5.7.244/24      10.5.7.0        10.5.7.255      ether1
  2    10.10.10.1/24      10.10.10.0      10.10.10.255    ether2

[admin@MikroTik] ip address>
```

在 5.0 后 IP 地址显示参数中，已将 Broadcast 参数显示去掉。

5.2 ARP 地址解析协议

操作路径：***/ip arp***

每个主机通过 IP 地址通信，但局域网的设备，通过 MAC 地址进行数据交换，三层通信建立在二层的基础上，地址解析协议 ARP 用于 OSI 第三层与第二层的 IP 和 MAC 连接。每台路由器都会有一个 ARP 清单，同一子网通信通常是建立为动态 ARP 列表，但为增强网络稳定性和安全性，可建立静态 ARP 列表，如防御 ARP 病毒。

属性描述

address (*IP 地址*) – 对应的 IP 地址
interface (名称) – 被分配 IP 地址的接口名称
mac-address (*MAC 地址*; 默认: **00:00:00:00:00:00**) – 相应的 MAC 地址

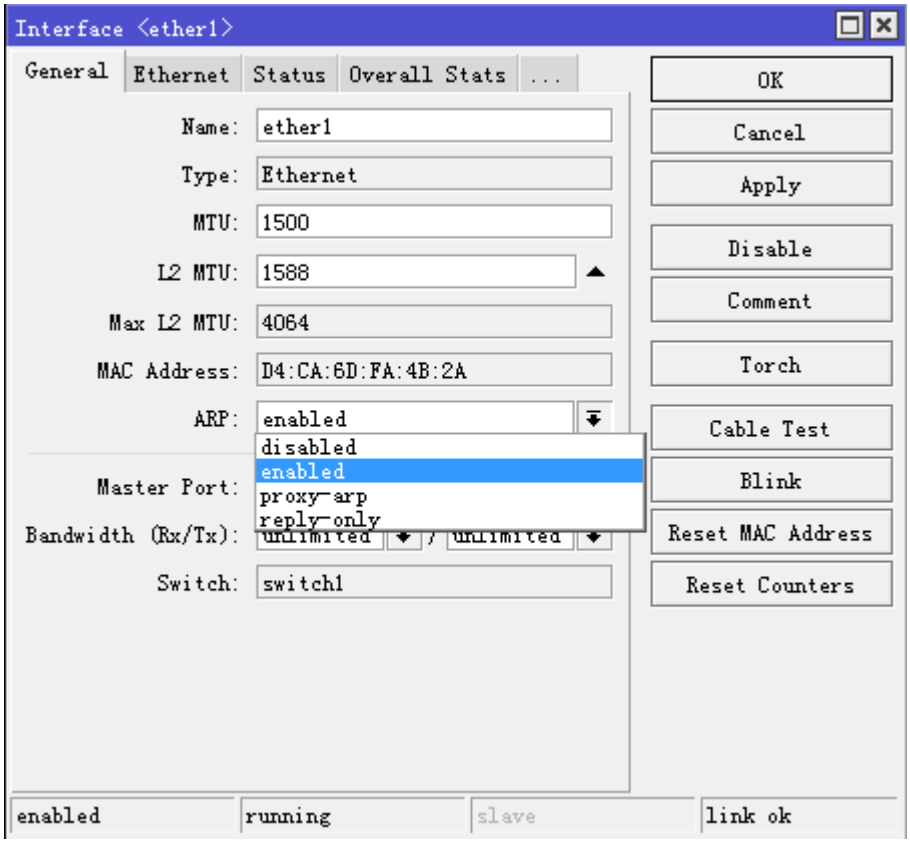
注：RouterOS 最大的 ARP 的条目数为 8192，所以在同一广播域内主机数量被限制在 8192 台。

进入 ip arp 列表查看当前路由器接口上获取的 ARP 列表信息

```
[admin@MikroTik] /ip arp> print
Flags: X - disabled, I - invalid, H - DHCP, D - dynamic, P - published
 #    ADDRESS          MAC-ADDRESS       INTERFACE
 0 D 192.168.10.37    08:63:61:CD:1F:AB    VAP
 1 D 192.31.3.14      0C:54:A5:56:A9:0D    ether1
 2 D 192.168.70.78    F0:CB:A1:A3:D8:41    wlan1
 3 D 192.31.3.1       00:1C:54:23:58:44    ether1
 4 D 192.168.10.42    0C:1D:AF:CD:D7:FA    VAP
 5 D 192.168.70.79    60:D8:19:CD:FE:60    wlan1
```

5.3 ARP 模式

RouterOS 在各个网络接口上提供了 ARP 模式的选择，用于提供在不同网络环境下的应用。如下图可以看到 ether1 的 ARP 选项：



Enabled

所有接口默认工作模式都为 enabled，在该模式下 ARP 协议会自运行，当遇到新的 ARP 请求会动态添加到 ARP 列表中

Disabled

如果 ARP 功能在接口上被关闭，使用 **arp=disabled**，来至客户端的 ARP 请求将不被路由器响应，因此必须添加静态的 ARP 条目。首先需要在/ip arp 清单添加对端 windows 主机的 IP 和 MAC 对应关系，然后在 windows 主机通过 **arp** 命令将路由器的 IP 和 MAC 地址必须添加到 windows 计算机中：

```
C:\> arp -s 10.5.8.254   00-aa-00-62-c6-09
```

注意：如果 arp 设置为 disabled，该以太网网卡将关闭 arp 协议，即该网卡三层网络会停止工作，如果没有配置静态 ARP，IP 网络将无法通信。

Reply-only

如果在接口上的 arp 属性设置为 reply-only，这时路由器只应答来至静态 ARP 的请求，即对/ip arp 列表中的静态 ARP 条目进行匹配。Reply-only 与 disabled 不同的是不要求主机绑定路由器的 ARP 信息。

例如：添加静态的 IP 与 ARP 条目

```
[admin@MikroTik] ip arp> add address=10.10.10.10 interface=ether2 mac-address=06:21:00:56:00:12
```

```
[admin@MikroTik] ip arp> print
Flags: X - disabled, I - invalid, H - DHCP, D - dynamic
  #    ADDRESS            MAC-ADDRESS         INTERFACE
  0 D 2.2.2.2            00:30:4F:1B:B3:D9     ether2
  1 D 10.5.7.242         00:A0:24:9D:52:A4     ether1
  2   10.10.10.10        06:21:00:56:00:12     ether2
[admin@MikroTik] ip arp>
```

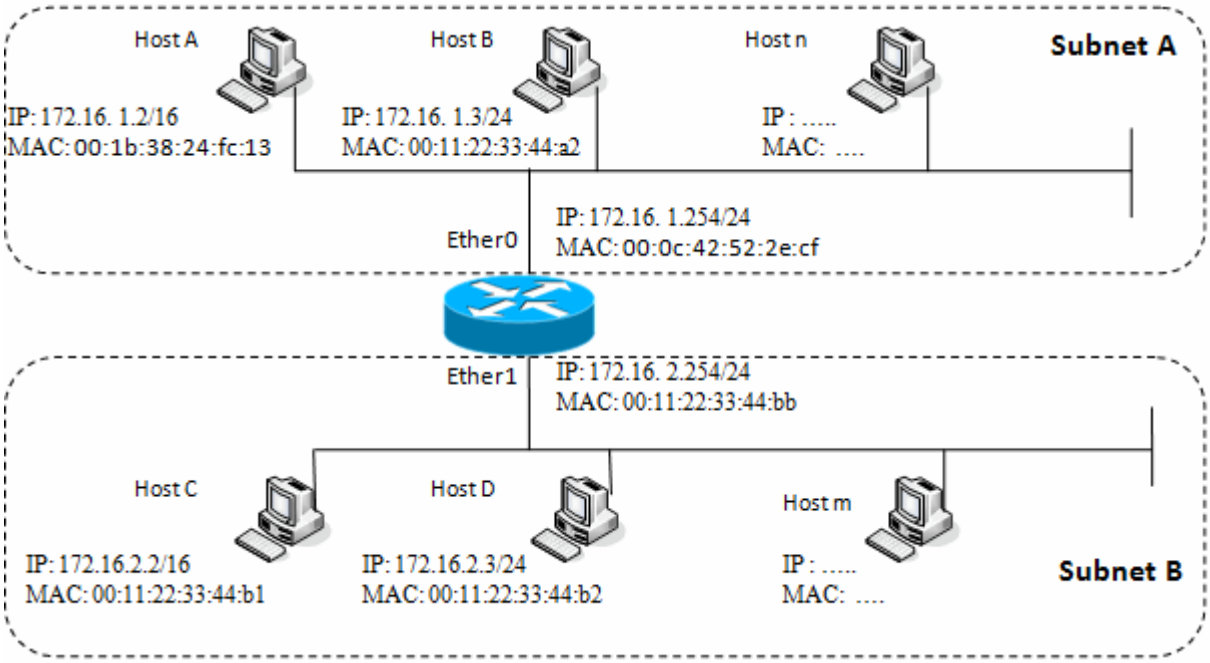
如果在一个接口上使用静态 ARP 记录会使网络更安全，即我们常说的路由器 ARP 绑定，除了配置静态 ARP 条目，还必须将该接口上的 arp 设置为 'reply-only'，相关操作在下面的**/interface** 目录中，如果非静态的 ARP 条目，将无法与路由器进行通信：

```
[admin@MikroTik] ip arp> /interface ethernet set ether2 arp=reply-only
[admin@MikroTik] ip arp> print
Flags: X - disabled, I - invalid, H - DHCP, D - dynamic
  #    ADDRESS            MAC-ADDRESS         INTERFACE
  0 D 10.5.7.242         00:A0:24:9D:52:A4 ether1
  1   10.10.10.10        06:21:00:56:00:12 ether2

[admin@MikroTik] ip arp>
```

Proxy-arp

当一台路由器设置 Prxoy-arp（ARP 代理），将实现 ARP 透明代理的功能，即直接连接不同接口的两个相同子网，例如 ARP 请求是从一个网络的主机发往另一个网络上的主机，那么连接这两个网络的路由器，通过替换 ARP 请求的 MAC 地址响应该请求，这个过程称作 ARP 代理。这样可以欺骗发起端的 ARP 请求，使它误以为路由器就是目的主机，而事实上目的主机是在路由器的**“另一端”**，这个功能类似于“nat 地址转换”功能。

通常可以用于两个通过 PPTP 或 L2TP 隧道连接，且两端 LAN 网络使用的是相同子网段的情况下，即实现跨路由的 ARP 代理请求，我们可以通过下面的一个实例分析和讲解 ARP 代理是如何工作的。



看看上面的网络结构图，其实在路由的 ether0 和 ether1 上都配置的是/24 的子网段，但由于 Host A 错误的配置了/16 的子网掩码，因此无法和对端的 Host D 通信，这里在不改变主机配置情况下实现两台主机的通信，就需要 ARP 代理实现。

我们来分析一下，Host A（172.16.1.2）在子网段 A 想发送数据报到 Host D（172.16.2.3）的子网段 B。Host A 子网掩码/16（意味着 Host A 相信所有 172.16.0.0/16 都是直连，即在同一子网段）。 首先 Host A 相信与 Host D 在同一子网段，便发出了一个 ARP 请求弄清 Host D 的 MAC 地址，在这个实例 Host A 查找的目标 IP 地址其实不在一个子网段，则 Host A 会广播 ARP 请求在子网段 A 里，同时会向默认网关发送数据报

该信息的数据报通过分析软件看到如下内容：

No.	Time	Source	Destination	Protocol	Info
12	5.133205	00:1b:38:24:fc:13	ff:ff:ff:ff:ff:ff	ARP	Who has 173.16.2.3? Tell 173.16.1.2

数据报详细内容

```
Ethernet II, Src: (00:1b:38:24:fc:13), Dst: (ff:ff:ff:ff:ff:ff)
    Destination: Broadcast (ff:ff:ff:ff:ff:ff)
    Source: (00:1b:38:24:fc:13)
    Type: ARP (0x0806)
Address Resolution Protocol (request)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (0x0001)
    [Is gratuitous: False]
    Sender MAC address: 00:1b:38:24:fc:13
    Sender IP address: 173.16.1.2
    Target MAC address: 00:00:00:00:00:00
    Target IP address: 173.16.2.3
```

通过这个 ARP 请求，Host A（172.16.1.2）询问 Host D（172.16.2.3）。ARP 请求数据报将 Host A 的源 mac 地址和目标广播地址(FF:FF:FF:FF:FF:FF)被封装在以太网帧内。二层广播意味着将向在相同二层广播域内的所有主机发送，包括路由器的 ether0，但不会到达 Host D，由于路由器工作在三层网络不会转发二层广播。

由于路由器知道目标地址 Host D(172.16.2.3)在另一端的子网内 ，通过 ARP 代理将自己的 MAC 替换，并发送给 Host A

No.	Time	Source	Destination	Protocol	Info
13	5.133378	00:0c:42:52:2e:cf	00:1b:38:24:fc:13	ARP	172.16.2.3 is at 00:0c:42:52:2e:cf

数据报详细内容

```
Ethernet II, Src: 00:0c:42:52:2e:cf, Dst: 00:1b:38:24:fc:13
```

```
    Destination: 00:1b:38:24:fc:13
    Source: 00:0c:42:52:2e:cf
    Type: ARP (0x0806)
Address Resolution Protocol (reply)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (0x0002)
    [Is gratuitous: False]
    Sender MAC address: 00:0c:42:52:2e:cf
    Sender IP address: 172.16.1.254
    Target MAC address: 00:1b:38:24:fc:13
    Target IP address: 172.16.1.2
```

即 Proxy ARP 代替了 Host D 向 Host A 发送回应数据报，在响应时将路由器自己的 MAC 地址设置为源 MAC 地址，目标 MAC 为 Host A。类似告诉 Host A：“把到 Host D 的数据发给我，我知道如何到达。”

当 Host A 接收到 ARP 回应后，ARP 列表将更新如下

```
C:\Users\And>arp -a

Interface: 173.16.2.1 --- 0x8
  Internet Address      Physical Address      Type
  173.16.1.254          00-0c-42-52-2e-cf     dynamic
  173.16.2.3            00-0c-42-52-2e-cf     dynamic
  173.16.2.2            00-0c-42-52-2e-cf     dynamic
```

当 MAC 列表更新后，Host A 将所有 Host D (172.16.2.3)的数据报都直接发送到路由 (00:0c:42:52:2e:cf) ，再由路由器转发到 Host D

接口上配置 ARP 代理的命名为 arp=proxy-arp，如在下面是路由器的以太网接口设置：

```
[admin@MikroTik] ip arp> /interface ethernet print
Flags: X - disabled, R - running, S – slave
#      NAME           MTU       MAC-ADDRESS         ARP
0   R ether1          1500   00:30:4F:0B:7B:C1     enabled
1   R ether2          1500   00:30:4F:06:62:12     enabled
[admin@MikroTik] /interface ethernet
[admin@MikroTik] /interface ethernet> set 1 arp=proxy-arp
 [admin@MikroTik] /interface ethernet> print

 Flags: X - disabled, R - running
   #      NAME                   MTU    MAC-ADDRESS          ARP
   0   R ether1                1500   00:30:4F:0B:7B:C1    enabled
   1   R ether2                1500   00:30:4F:06:62:12    proxy-arp
 [admin@MikroTik] interface ethernet>
```

Local Proxy Arp

从 v6.38 开始，如果接口的 arp 参数设置为 local-proxy-arp ，此时路由器对该接口执行代理 ARP。也就是说，在同一个接口响应全网主机的流量，这样在同一个接口将会处理全局域网的发送和接收流量。

在一个正常的局域网中，默认的行为是让两个网络主机直接相互通信，而不涉及路由器。启用 local-proxy-arp 功能后，路由器将用自己的接口 MAC 地址，去响应全局域网的 ARP，而不是两台主机之间直接通信

例如，当主机 A (192.168.88.2/24) 请求主机 B 的 MAC (192.168.88.3/24)，路由器会主动回应自己的 MAC 地址，路由器将负责在主机 A(192.168.88.2)和主机 B （192.168.88.3）之间转发流量。主机 A 和主机 B 上的所有 arp 缓存项都会引用路由器的 MAC 地址。在这种情况下，路由器正在对整个子网 192.168.88.0/24 执行 local-proxy-arp。

下面实例，RouterOS 的 local-proxy-arp 可以配置在一个网桥上，其中有 DHCP 服务器和隔离的网桥端口，来自同一子网的主机只能通过 RouterOS bridge1 接口的 IP 进行三层转发，配置如下：

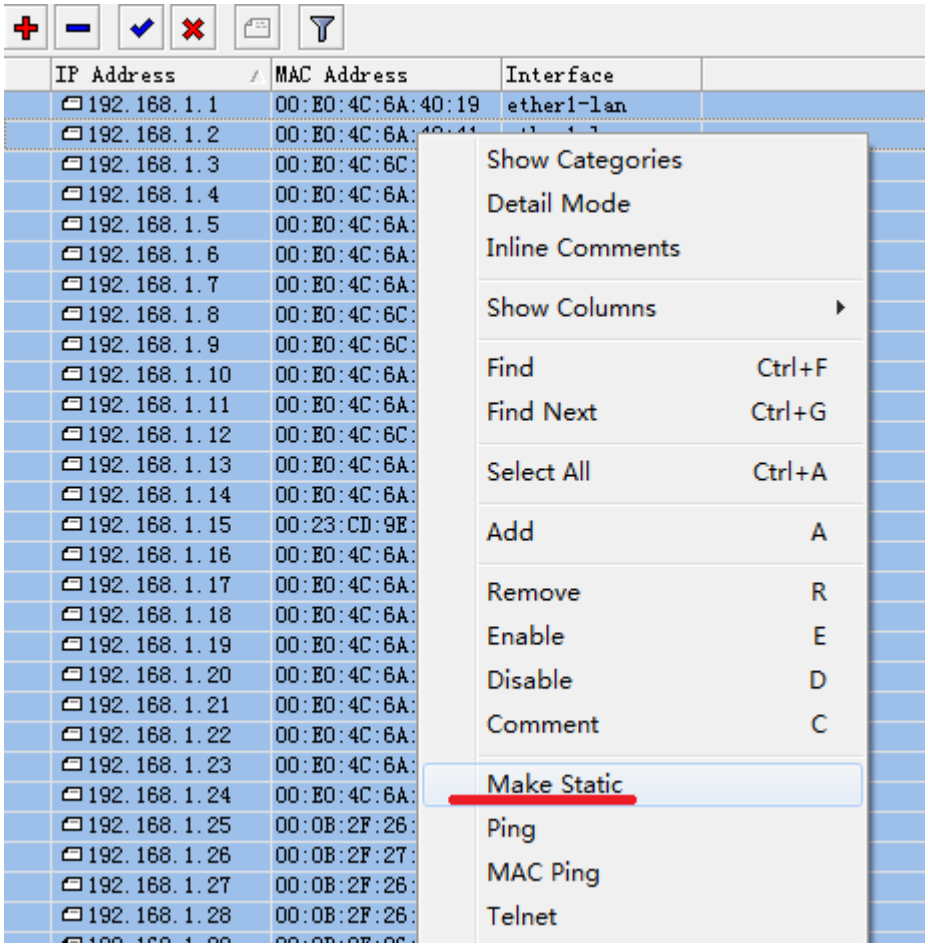
```
/interface bridge
add arp=local-proxy-arp name=bridge1
/interface bridge port
add bridge=bridge1 horizon=1 interface=ether2
add bridge=bridge1 horizon=1 interface=ether3
add bridge=bridge1 horizon=1 interface=ether4
```

这种有点类似局域网 ARP 欺骗。

5.4 ARP 绑定

每个路由器都一个 ARP 列表，记录 ARP 信息，由 IP 地址和相符合的 MAC 地址构成，ARP 提供了动态的 IP 与 MAC 地址对应关系，在 ARP 列表中自动产生，建立动态的 ARP 关系，路由器通过 ARP 列表的记录来响应各个主机的 IP 数据通讯。RouterOS 动态 ARP 列表是 60s 更新一次，为了避免用户私自修改 MAC 或出现的 ARP 病毒，以及用户擅自修改 IP 地址，我们可通过静态的 ARP 记录，要求路由器只对添加了静态的 ARP 关系的主机做响应。如通过下面的设置：

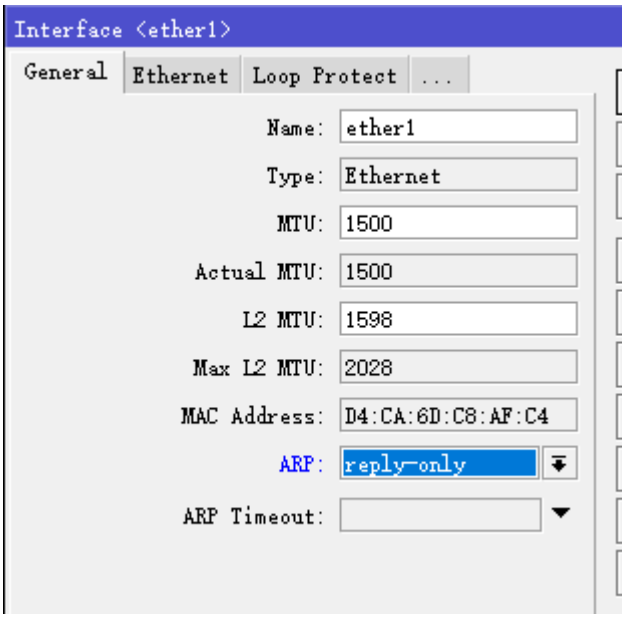
第一步：在 WinBox 中进入 ip arp 菜单，将动态 ARP 设置为静态的 ARP 条目，选中需要设置静态的条目，右键点出下拉菜单，选择 Make Static。



或者通过命令操作手动设置主机和 MAC 的 ARP 关系：

```
[admin@MikroTik] ip arp> add address=192.168.1.248 interface=ether1-lan
mac-address=00:21:00:56:00:12
```

第二步：设置 ether1 的 arp 选项，设置 arp=reply-only，仅响应静态 ARP 的请求：



命令操作如下：

```
[admin@MikroTik] > interface ethernet set ether1 arp=reply-only
```

ARP 双向绑定事例

首先将所有/ip arp 列表中的所有 LAN 口的 ARP 信息变为静态的，我们可以通过脚本做批处理的修改。注意，可能通过脚本命令不一定能将所有的内网的 ARP 参数修改完，可能需要手动添加。

```
:foreach i in [/ip arp find dynamic=yes interface=LAN] do={
  /ip arp add copy-from=$i}
```

然后设置 LAN 的网卡 ARP=disabled，如果 ARP 功能在接口上被关闭，即关闭了 ARP 协议，无法学习到 IP 和 MAC 相关信息，来至客户端的 ARP 请求将不被路由器响应，因此必须手动添加静态的 ARP 才行。 例如，通过 arp 命令将路由器的 IP 和 MAC 地址必须添加到 windows 工作站中：

```
[admin@MikroTik] ip arp> /interface ethernet set LAN arp=disabled
```

现在路由器已经绑定了内网主机的所有 IP 地址后，现在需要对 Windows 计算机做对路由器绑定的设置

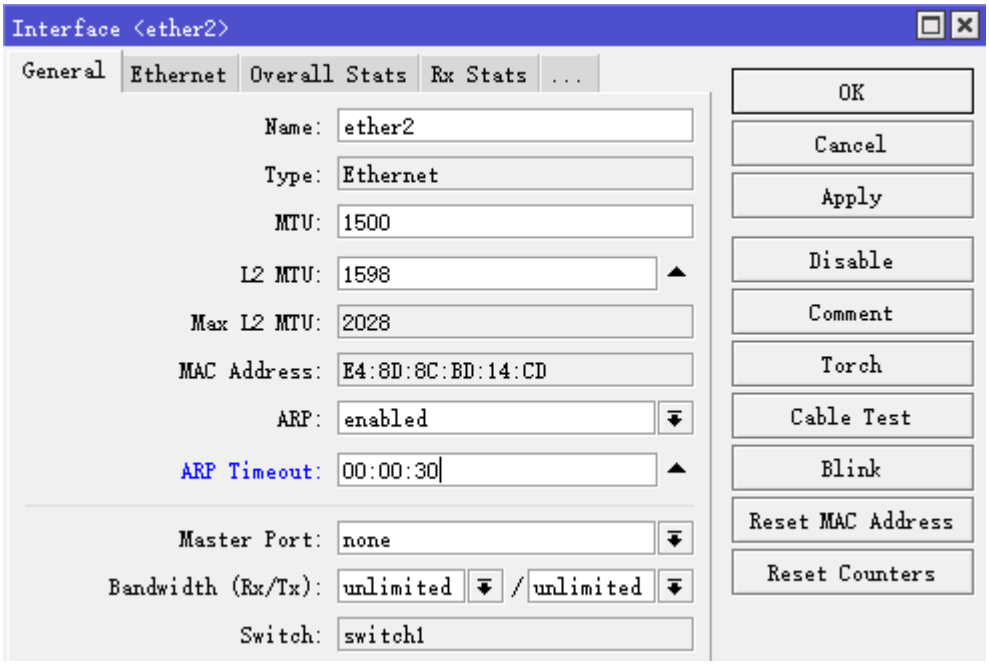
```
C:\> arp -s 10.5.8.254   00-aa-00-62-c6-09
```

如果命令操作麻烦，也可以编辑 windows 的批处理脚本（.dat）进行操作。

ARP Timeout 设定

v6.36 版本后新增了 arp-timeout 属性，允许修改默认的 ARP 刷新时间，默认是 60 秒，我们根据自己需要设定超时时间，重新刷新 arp，在一定范围内有助于对 arp 病毒的防御。

通过 winbox 打开后，可以看到 ARP Timeout 设置属性：



第六章 Route（路由）

下面的内容介绍了 RouterOS 的路由管理，针对目标、源地址和策略路由等，在各种网络环境具体使用，至于哪一种路由方式，需要根据用户自己的网络情况来选择，这章主要以静态路由和策略路由为主，关于动态路由在后面一章介绍。

需要功能包：***system***
软件等级：*Level1*
操作路径：***/ip route***, ***/ip route rules***

RouterOS 支持各种路由规则和策略，在实际网络环境中可选择适合自己网络的路由规则，如 Tel 和 Un 多线接入、目标和源地址策略路由、基于端口的策略路由和多线路绑定的负载均衡、已经负载均衡演变的权重路由，下面是 RouterOS 支持的几种方式的路由：

- **支持源 IP 地址的策略路由**
- **支持目标 IP 地址策略路由**
- **支持网页等 TCP 或 UDP 端口策略路由**
- **支持 ECMP、Nth 和 PCC 负载均衡**
- **支持 IP 地址列表的策略路由**
- **各种策略都可以组合使用**

路由器是连接多个子网，保障路由信息连续的设备，对于 RouterOS 来说需要通过 RIB 和 FIB 功能保持路由正常工作。

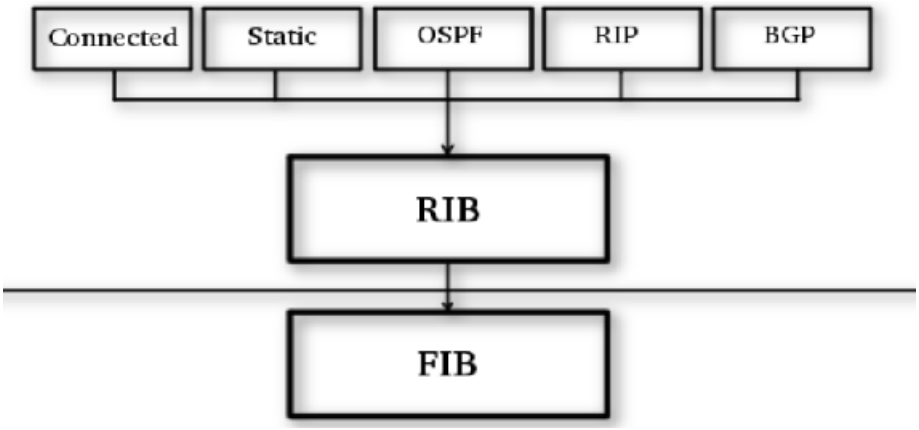
6.1 RIB 和 FIB

RIB 是由各种路由协议（如 RIP、OSPF、IS-IS、BGP 和静态路由等）是管理路由信息库，包含路由表信息以及路由相关联的信息(路由迭代信息、路由共享信息以及路由扩展信息)。RIB 通过对路由表选出最佳路由后，把筛选的路由下发到 FIB，FIB 来指导转发

FIB 接收到 RIB 下发的路由表后，对每条转发项都指明了要到达某子网或主机的报文应通过路由器的哪个物理接口发送，FIB 可以直接下发给硬件处理，如果出现的路由表信息不 FIB 中，再由 RIB 生成，并重新更新 FIB 表。

- **RIB(Routing Information Base)** 路由信息库又可以称为路由表，是一个存储在路由器或者联网设备中的路由文件表或类数据库。路由表存储着指向特定网络地址的路径。在 RouterOS 包含路由分组信息，即通过 routing-mark 标记创建独立的路由表。如果没有 routing-mark 标记，则保存在 main 路由表中。
- **FIB(Forwarding Information Base)**转发信息库，是决定目标交换的查找表，FIB 的条目与 RIB 路由表条目之间有一一对应的关系，即 FIB 是 RIB 路由表中包含的路由信息的一个镜像。当网络拓扑或路由发生变化时，RIB 路由表被更新，FIB 的内容随之发生变化

RIB 和 FIB 的关系，RIB 根据路由协议生成路由表，并下发给 FIB，建立缓存，并执行路由转发，如下图

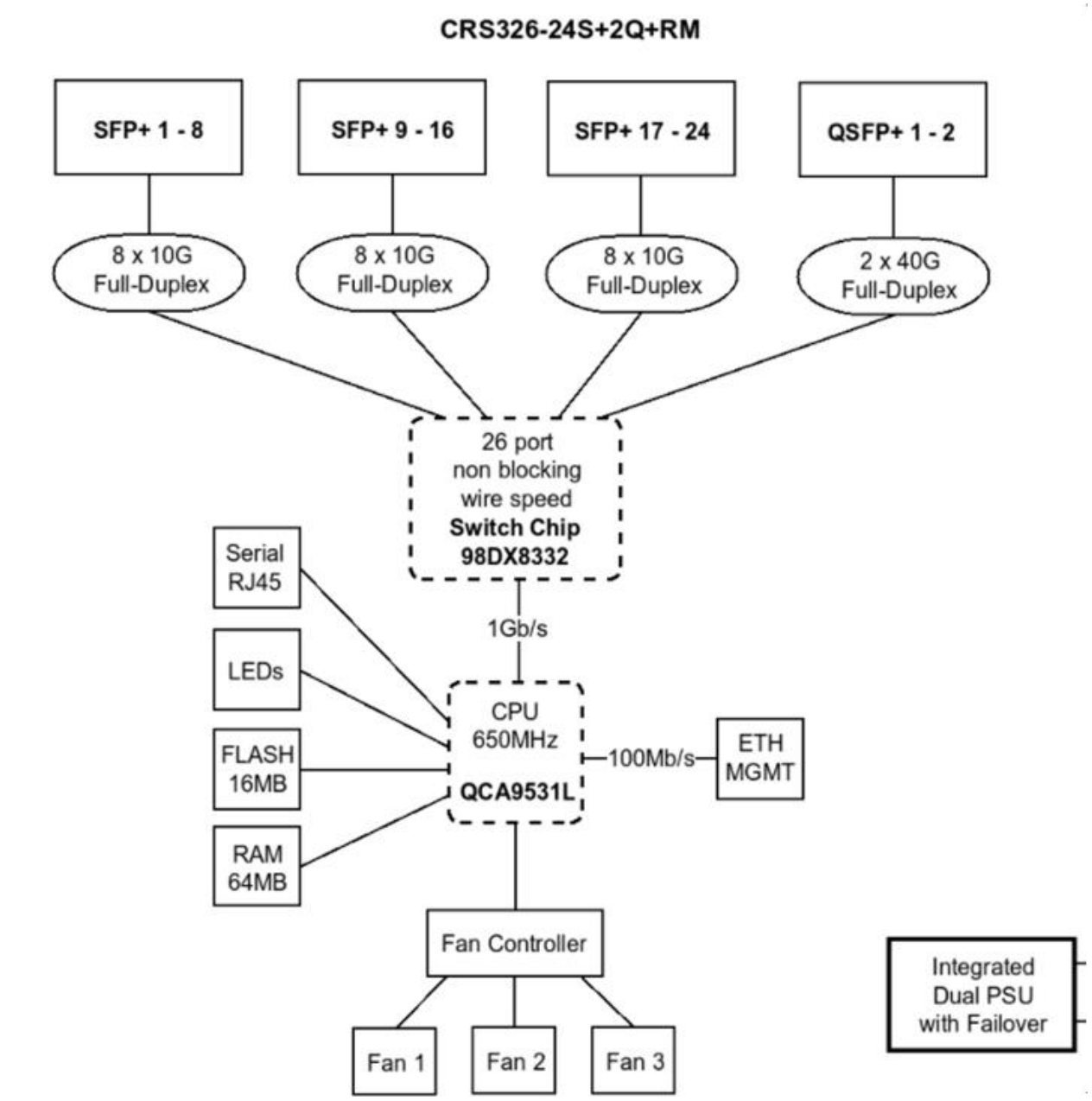


说的通俗点，RIB 就像公司的领导层，制定了公司的章程制度、项目方案和工作计划，而 FIB 就像项目经理，拿到公司制定的方案后，按照工作岗位分配执行下去，每个人各司其职。RIB 是不参与整个实际过程的，如果遇到不在计划中的某个问题，就上报 RIB，重新讨论该问题，制定解决方案，方案主体不会改变，仅修补后再交给 FIB 继续执行。

这种 RIB 加 FIB 的结构，使得决策层面 RIB 和转发层面 FIB 分离，决策和转发分离使路由器在开发和扩展方面更加有连续性。典型的三层交换机来说，CPU 和交换芯片是独立分开的，CPU 负责 RIB 的决策，然后生成 FIB，而 FIB 下发到交换芯片处理。FIB 表中的条目数也是影响路由器性能的重要因素，通常来讲 FIB 条目越多，查找花费的时间越长，但基于 ASIC 开发的交换芯片来说，随着技术的不断升级和功能的扩展，转发性能也不断提升，完全能实现路由查询转发达到线速。

举例

下面是 CRS326-24S+2Q+RM 的设计框架图：



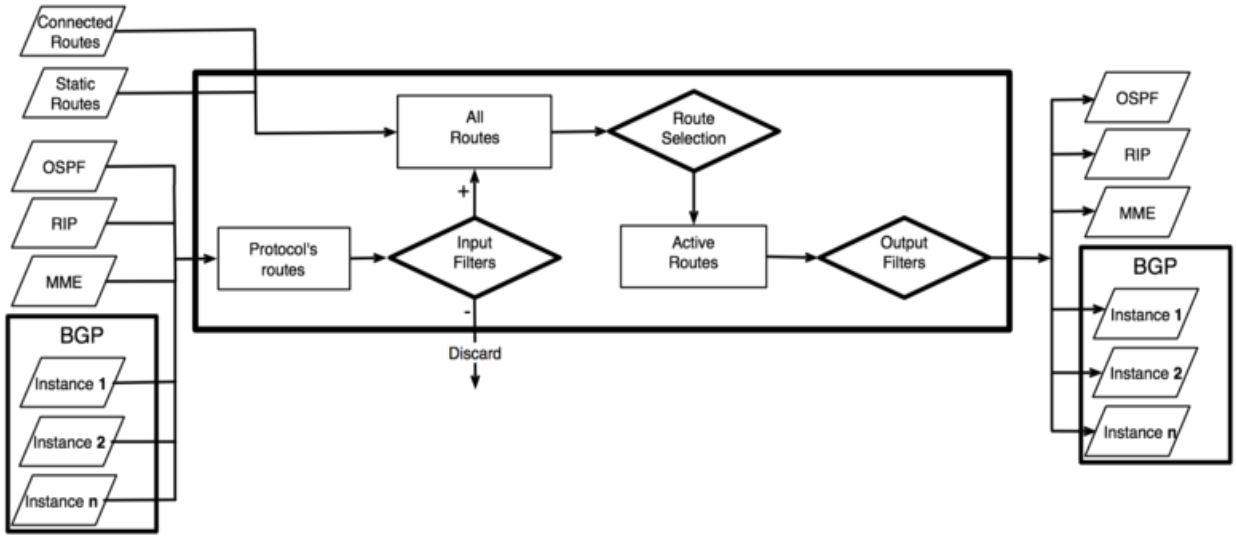
CRS326-24S+2Q+RM 是 24 个 10G+2 个 40G 端口产品，在 RouterOS v7 版本可以支持二层和三层线速转发，从上图框架图可以看到，该型交换机的 CPU 为 QCA9531L 连接 Switch Chip 98DX8332 的交换芯片，但他们之间总线只有 1Gb/s，如果数据要交给 CPU 处理，只能跑 1Gb/s，且采用的 CPU 是当前 MikroTik 在售路由器中性能垫底的一款，显然不合理。

其实处理二层和三层转发仅需要生成相应的策略下发给 98DX8332 的交换芯片，二层转发生成 FBD 表（MAC+VLAN 和 PORT 的对应关系），三层生成 FIB 表。数据转发处理都给交换芯片完成，因此 CPU 负责日常 RouterOS 的系统运行，规则生成等任务，决策和转发功能分离是软件和硬件结合最好的解决方案。

这样设计路由交换产品时，可以选择一个弱鸡 CPU（能有效处理 RIB），搭配一个高性能交换芯片，也能达到三层路由线速转发的性能。只要大家遵循一个标准通信协议，CPU 和交换芯片可以任意搭配。

对于系统构建 FIB 需要一定的时间，它需要先接收路由信息包，建立路由表，选路，建立转发表并把转发表的适当的子集推送到线卡上，减少路由器 CPU 到线卡的压力。对于特别大的 FIB 整个过程会需要大约几分钟，所以了解路由器 FIB 最大条目数十分重要，但厂商一般不会在产品指标说明中提供 FIB 的大小限制。

6.2 RIB 路由信息库（生成路由表）



RIB (Routing Information Base) 包含完整的路由信息，包括管理者配置的静态路由和策略路由。通过从连接的网络中，通过相关路由协议学习到路由信息 RIB，并用于过滤路由信息，计算每条目标信息的最佳路径，建立和更新 **FIB 转发信息库**，在不同路由协议间分配路由。

RIB 是一个数据库，它列出了网络目的地及其网关的条目(指定路径的下一个设备的 IP 地址或下一跳)。路由表中的一个这样的条目叫做 **route**。当一个 IP 数据包从一个网段传递到另一个网段时，就被称为"**hop 一跳**"。

通常路由转发通过目标 IP 地址决定，每条路由都有 **dst-address** 属性，此属性决定了这条路由的转发方向。如果当多条路由指向同一目标网络，而取决这个目标地址路径选择则由子网掩码大小确定，即以精确子网掩码为准。这样的操作查找给定目标 IP 地址，选择具体路径的过程被称为路由表查询（**routing table lookup**）。

如果路由表包含多条相同 **dst-address** 的目标 IP 地址路由，则其中只有一条可用于转发 IP 数据。此路由信息被创建到 **FIB** 中并标记为 **active**（活动状态）。

当转发决策使用其他附加参数，如源 IP 地址匹配，这样就是我们所说的**策略路由（policy routing）**。策略路由通过策略路由规则表执行，将基于不同路由表选择目标地址、源地址、源接口以及路由标记（routing-mark 在 firewall mangle 中标记数据报）。

默认情况下路由都会首先到预设的 main 路由表查询，如果创建了 routing-mark 路由标记，将分配到具体的新路由表中。因此对于策略路由而已是新建了一个路由表进行查询，不再匹配默认的 main 主路由表。

默认路由（Default route）

即目标路由（dst-address）为 0.0.0.0/0 (IPv4)和::/0 (IPv6)，到任何目标地址的路由，这类路由被称为默认路由。如果路由表中包含一条有效的默认路由，当路由表中其他路由无法找到目的地时，此时该路由表中查询路由将选择该默认路由，作为唯一的路径选择。通常家庭路由器，路由表只又直连路由和一个 ISP 运营商的默认路由，以转发所有出站的流量到 ISP 网关，如果下面 10.155.125.1 就是路由器的默认路由

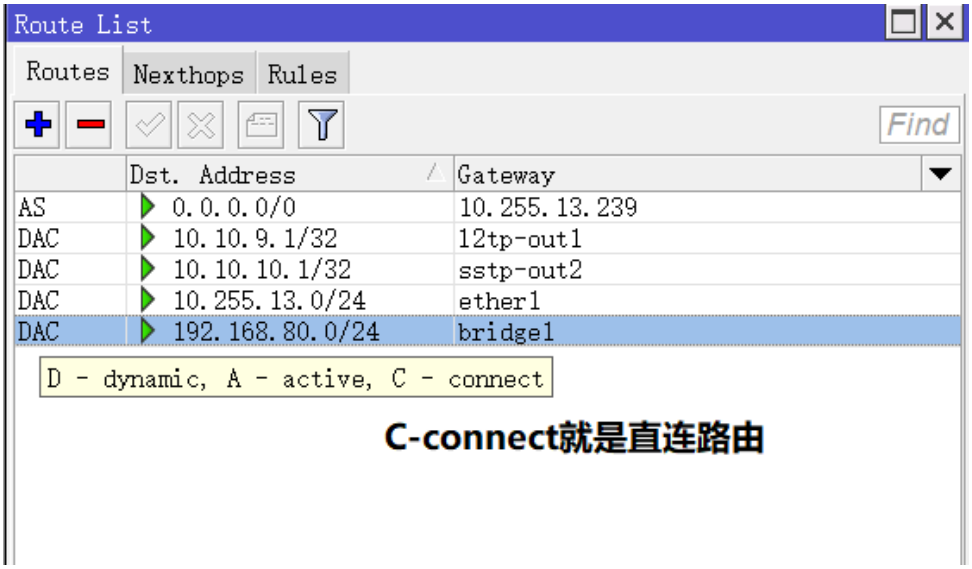
```
[admin@MikroTik] /ip route> print
```

```
Flags: D - dynamic; X - disabled, I - inactive, A - active; C - connect, S - static,
r - rip, b - bgp, o - ospf, d - dhcp, v - vpn
Columns: DST-ADDRESS, GATEWAY, Distance
#      DST-ADDRESS        GATEWAY         D
   DAd 0.0.0.0/0          10.155.125.1    1
   DAC 10.155.125.0/24    ether12         0
   DAC 192.168.1.0/24     vlan2           0
```

直连路由（Connected routes）

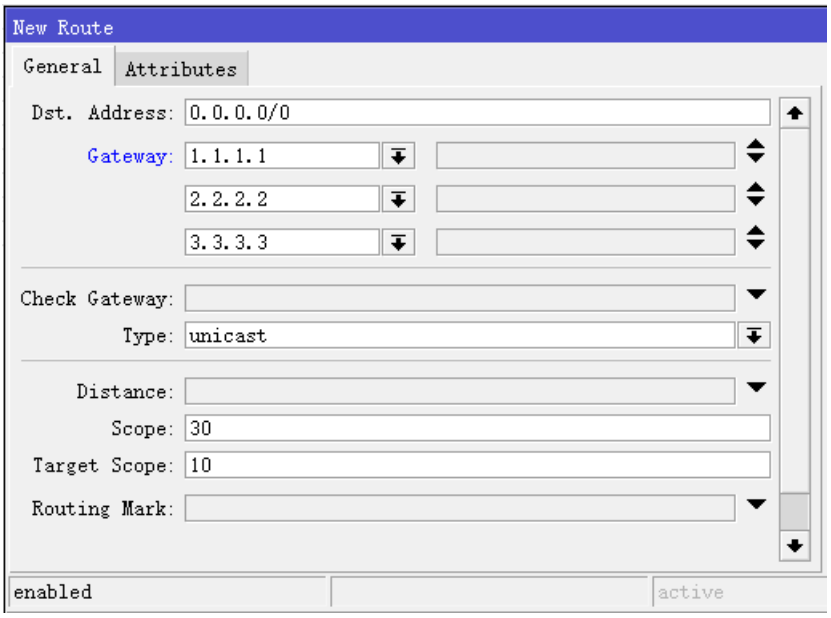
直连路由代表主机可以直接到达的网络(相同的二层广播域)，直连路由也可以理解，在路由器任何接口创建的 IP 网络(在*/ip address* 或*/ipv6 address* 中指定)，这些路由是自动创建的。RIB 会跟踪直连路由状态，但不会去修改，当直连路由接口处于 down 状态，直连路由会自动失效。而与之相反的是非直连路由，即不在同一 IP 子网，通过路由协议从其他路由器学到的路由，分为静态路由和动态路由。

如我们配置 192.168.80.1/24，那 192.168.80.0～255 这个网络地址范围内都称为直连路由，因为他们都属于路由器本地接口上同一子网段，如下图：



等价多路径路由(ECMP)

之前提到相同 dst-address 的路由只能有一条被转发，但通过配置，使用 ECMP (Equal cost multi-path)路由，ECMP 是在多条不同链路网关到达同一目的地址的路由协议，ECMP 最大的特点是实现了等值情况下，多路径负载均衡和链路备份的目的，在静态路由和 OSPF 中基本上都支持 ECMP 功能。如下图，在 ip route 中添加新的路由规则，gateway 允许设置多个网关



ECMP 能做到转发决策被缓存，转发数据报有相同的源地址、目标地址、路由标记和 ToS 信息被发送到同一网关。这意味着一个连接将使用在每个方向只有一个连接，因此 ECMP 路由被用于每次连接的负载均衡（但注意 ECMP 路由不能应用于 nat 下的负载均衡如 PCC 或 Nth 负载均衡）。

网络接口作为网关

网络接口（interface）在部分情况下可以作为网关替代 IP 地址，网络接口作为网关以下情况说明：

- 不同于直连路由，路由的配置网络接口作为下一跳，不能进行下一跳路由查询（nexthop lookup）。
- 可以分配多个网络接口到 gateway 作为网关，同样可以创建 ECMP 路由，但不能用于直连路由的网关

网络接口作为网关，通常用于点对点协议，而不能用于直连路由的网关做下一跳查询。

路由距离（Distance）

Cisco 文档描述了"administrative distance"，即管理距离，当路由器学习到一个相同目标路由有多种路由协议时，管理距离会把优先权给路由距离值低的路由协议。

RouterOS 内部的默认路由距离值：

协议	distance
直连路由	0
静态路由	1
eBGP	20
OSPF	110
RIP	120
MME	130
iBGP	200

6.3 FIB（转发信息库）

FIB (Forwarding Information Base) 包含 IP 数据包转发所需信息的镜像，即由 RIB 生成路由规则下发给 FIB 执行：

- 所有活动状态的路由（active routes）
- 策略路由规则

默认情况下(当没有使用 routing-mark 路由标记时)，所有活动路由都在 main 表中，对于 main 表会有一个“隐藏”规则(“catch all”规则)将 main 表用于所有目标路由的查找。

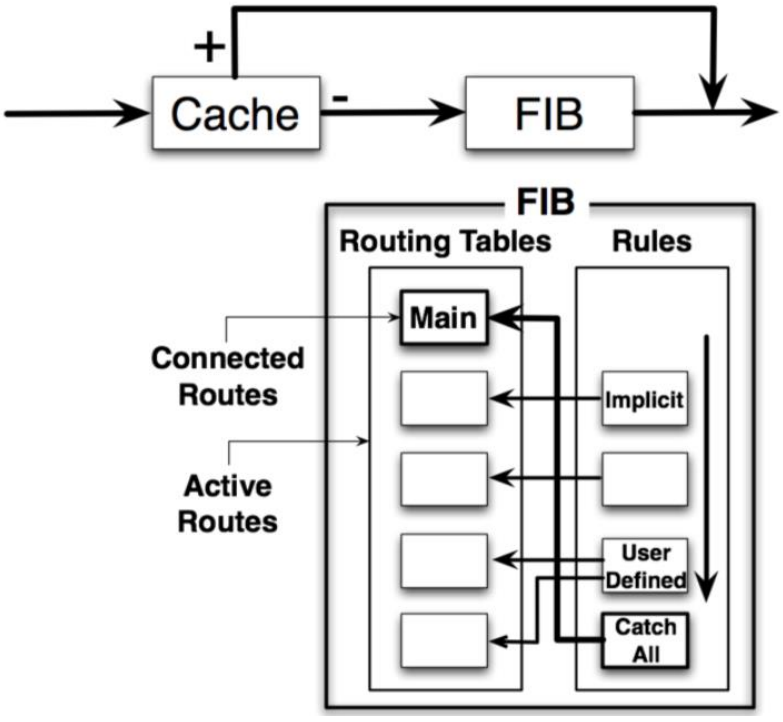
Routing table lookup（路由表查询）

FIB 使用以下信息确定 IP 包的目标路由：

- source address（源地址）
- destination address（目标地址）
- source interface（源接口）
- routing mark（路由标记）
- ToS (RouterOS 在策略路由规则中不使用它，但它是路由缓存查找关键字的一部分)

路由决策的结果被保存在路由缓存(Cache)中，这样提高路由转发性能。当具有相同源地址、目的地址、源接口、routing-mark 和 ToS 进行路由查询时，一旦命中缓存，将会快速转发。Cache 也能用于 ECMP 的负载平衡，当查找路由条目值属于同一连接且相同目标时，也能命中路由缓存。如果 IP 包没有命中路由缓存，则通过创建新的路由决策

从下面的流程图可以看到,当路由查询命中到缓存将直接转发，如果没有命中将建立新路由规则表，并加入路由缓存。



FIB 维护路由表中包含转发信息的镜象，当网络中路由或拓朴结构发生了变化时，IP 路由表就被更新，而这些变化也反映在 FIB 中。RouterOS 使用的是 Linux 内核，Linux 内核有一种 Route Table 的数据结构，它是 FIB 的一份 cache。在 Linux 系统中路由一般采取的手段是，先到路由缓存中查找，如果能查找到，那就直接将对应的一项取出作为路由的规则；如果查不到，那么就到 FIB 中根据规则增加一项路由表，在路由缓存中将新增的路由表添加进去。

FIB 强调的是作为转发的路由表，RIB 是用来做路由管理的表。通常有了动态路由协议的参与才能理解，RIP、OSPF、BGP、ISIS 都是动态路由协议，它们学习到的路由首先要通告给 RIB 表。RIB 表把所有路由协议学习到的路由汇总到一起，经过优选，把优选结果的路由加入到 FIB 表，供转发使用。

6.4 RouterOS 路由配置

RouterOS 路由配置有手动和自动添加两种方式：

- **自动添加路由** 是当在一个网卡上添加了 IP，会自动创建一个条路由（如 PPPoE-Client、PPTP-Client 和 DHCP-Client 等自动添加网关）。
- **静态路由** 是用户自定义将 IP 数据报指定到默认网关的路由，这需要手动指定预设的网关。

当然在使用 PPPoE-Client、PPTP-Client 和 DHCP-Client 自动添加路由外，只能手动添加默认的路由规则，即默认路由，除非使用了动态路由协议（RIP 或 OSPF）

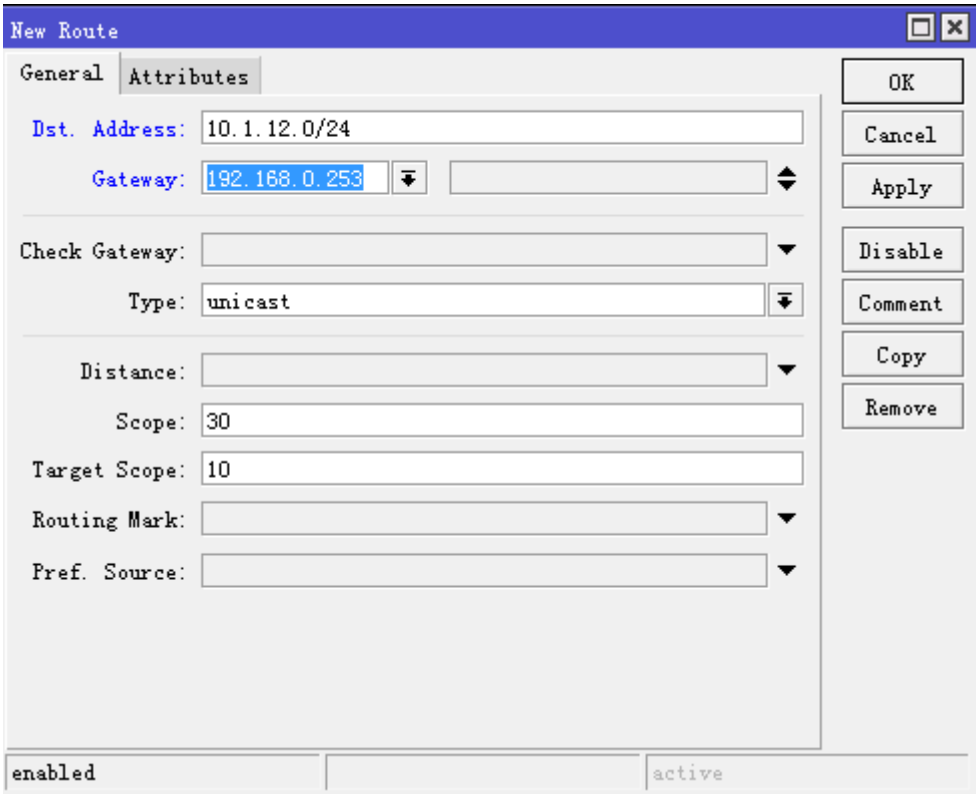
静态路由

操作路径: ***/ip route***

在一个路由器两个 IP 段中，添加内网静态路由到网络 10.1.12.0/24 和默认网关出口路由为 0.0.0.0/0，在配置 RouterOS 默认路由时，命令行添加默认的 gateway：

```
[admin@MikroTik] ip route> add dst-address=10.1.12.0/24 gateway=192.168.0.253
[admin@MikroTik] ip route> add gateway=10.5.8.1
[admin@MikroTik] ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf
 #     DST-ADDRESS       G GATEWAY        DISTANCE INTERFACE
 0 A S 10.1.12.0/24       r 192.168.0.253          Local
 1 ADC 10.5.8.0/24                                Public
 2 ADC 192.168.0.0/24                             Local
 3 A S 0.0.0.0/0         r 10.5.8.1                Public
[admin@MikroTik] ip route>
```

下面是使用 Winbox 进入 ip route 配置静态路由，指定 10.1.12.0/24 走网关 192.168.0.253：



ECMP 等价多路径路由

ECMP-"Equal-Cost Multi-Path Routing"即等价多路径路由是三层路由常见的协议，在 Cisco 和华为等路由器中的三层路由配置是很常见的，是最基本的静态路由负载均衡，但不能用于 nat 网络下的路由负载均衡。

Equal-Cost Multi-Path Routing（等价多路径路由）均衡策略是每 10 分钟后，会重新计算均衡线路，非 nat 网络的三层路由目标和源地址是没有隐藏的，目标和源 IP 是相互能识别的，虽然线路改变了，但目标和源地址是为改变的，连接不会受到影响。但 nat 网络下私网 IP 到出口 IP 是做了隐藏的，ECMP 重新计算线路路径，目标 IP 虽然不变，但多线路出口的 IP 改变了，源出口 IP 改变后，目标 IP 会认为是非法连接，导致目标 IP 无响应出现频繁掉线情况，所以当基于 nat 的负载均衡时"Equal-Cost Multi-Path Routing"是不能选择的，在 nat 网络的负载均衡请选择 PCC 策略。

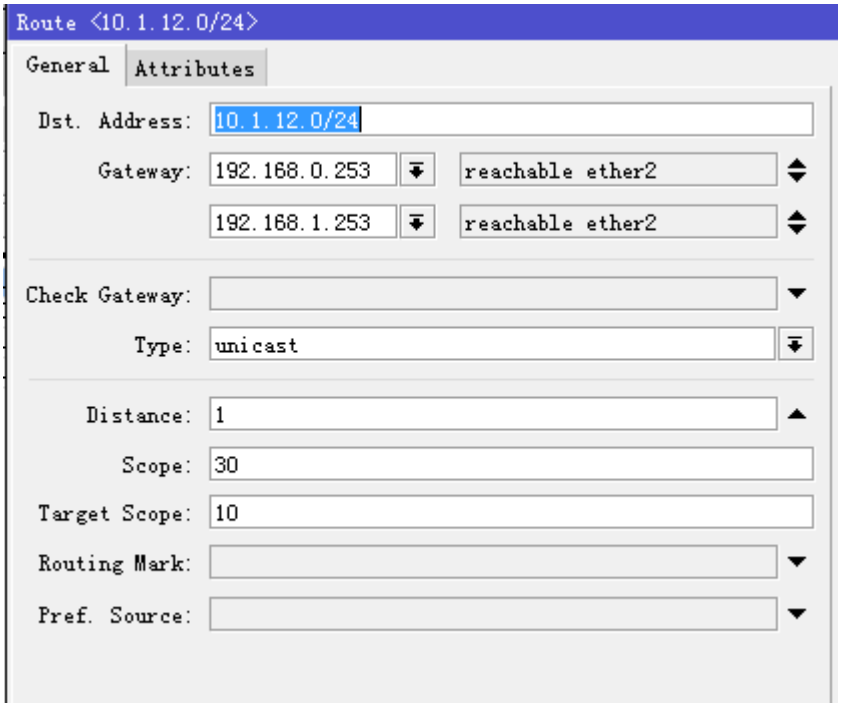
- Equal-Cost Multi-Path Routing 的操作，通过在 ip route 添加多网关的静态路由（格式如：**gateway=x.x.x.x,y.y.y.y**）路由协议会建立动态的多路路由。

假设路由网络中有两条网关到 10.1.12.0/24，一个网关是 192.168.0.253，一个是 192.168.1.253，我们配置 ECMP 负载均衡规则：

```
[admin@MikroTik] ip route> add dst-address=10.1.12.0/24 gateway=192.168.0.253,192.168.1.253
[admin@MikroTik] ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 0 A S  0.0.0.0/0                          10.5.8.1             1
 1 A S  10.1.12.0/24                       192.168.0.253        1
                                           192.168.1.253
```

```
 2 ADC  10.5.8.0/24         10.5.8.2          ether1                0
 3 ADC  192.168.0.0/24      192.168.0.2       ether2                0
 4 ADC  192.168.1.0/24      192.168.1.2       ether3                0
[admin@MikroTik] ip route>
```

使用 winbox 配置：



策略路由

策略路由是通过创建两个或两个以上的路由表来实现选择不同的路由选择，涉及到 ip route、/ip firewall manlge、/ip firewall address-list 等多个配置，策略路由主要实现多条线路的路由控制，如指定源地址策略路由，目标地址策略路由、端口策略路由、地址列表的策略路由，也可以通过多种方式组合使用：

- 通过 **mangle** 标记期望的数据（源地址、目标地址和端口），设置一个 **routing-mark**
- 在 **ip route** 或 **ip route rules** 中配置目标和各种路由协议
- 使用 **address-list** 定义地址列表，并进行 routing-mark 标记

对于 RouterOS 的 nat 环境下配置负载均衡策略，也属于策略路由，对于 ECMP 不考虑此范畴（因为没有创建新的路由表）。最早出现的 nat 环境下的负载均衡是 **Nth**，**Nth** 采用第 N 次连接的负载均衡，这样基本实现了不掉线的真正负载均衡，但存在一个弊端就是要求对 IP 验证的网站会多次要求验证（如银行网银和论坛验证等），这样需要通过策略指定一些 IP 或者端口走固定的线路。由于 Nth 的特殊性，将在单独一章进行讲解。

趋于完善的负载均衡策略是 **PCC（Per connection classified）**每次连接分类的负载均衡，这样的策略对每次的连接进行分类保持连续性的负载均衡，弥补了 Nth 的不足。而且我们可以通过 PCC 更好的实现多线路的权重路由，即按照比例分配到不同网关的流量。

6.5 ISP 目标地址路由

ISP 最常见的是 Tel 和 Un 两家运营商的双线方式，通过路由指定分别让内网主机访问走 Tel 和 Un 线路。该双线中，我们需要选择一条线路为主线，即默认路由出口，比如 Tel 为主线，缺省网关设置为 Tel 网关地址；Un 线路需要通过导入路由表（Tel 和 Un 的路由表脚本到 bbs.routerclub.com 的网站查找），ISP 目标地址路由既可以做静态路由，也可以做策略路由方式较多，下面是策略路由的做法。

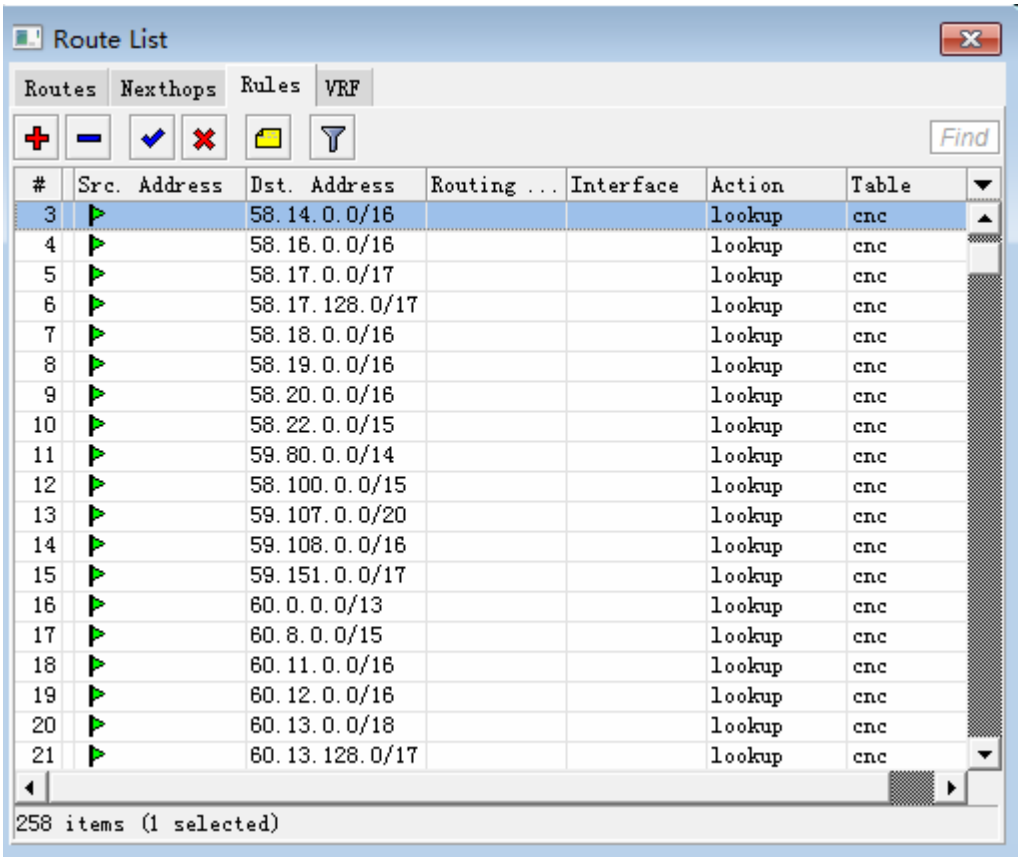
上传 Tel 或 Un 路由脚本后，在根目录下使用 import 命令导入：

```
[admin@MikroTik] > import cnc1.rsc
```

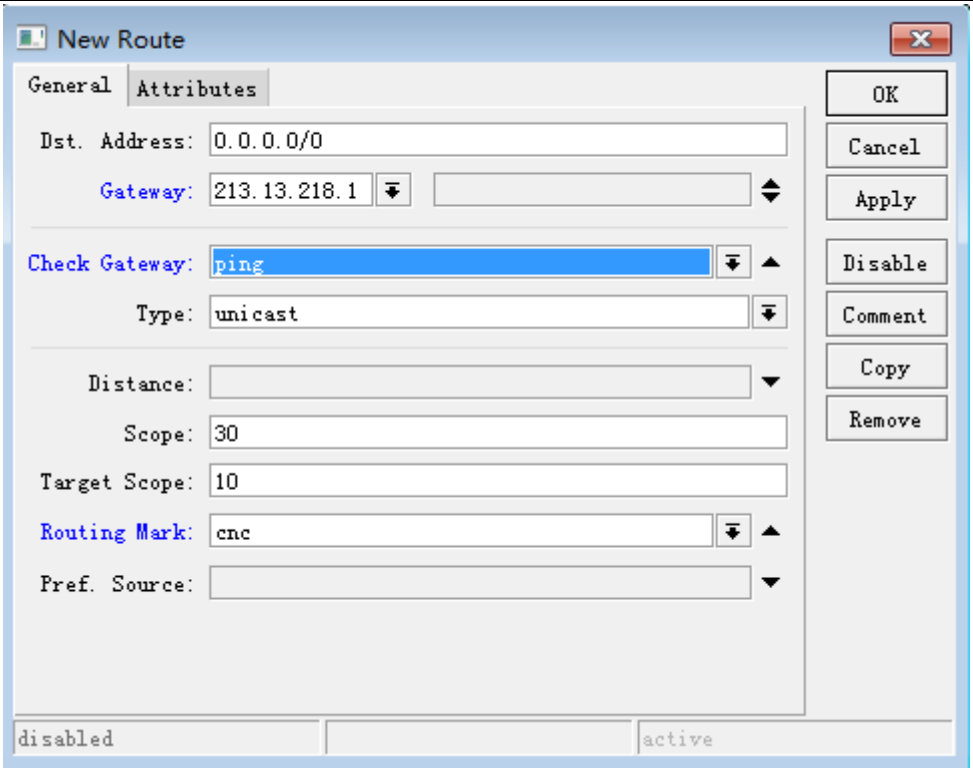
设置路由规则时命令如下：

```
/ip route add gateway="对应网关地址" check-gateway=ping routing-mark=telecom 或者 cnc
```

如导入的 Un 线路标记为 cnc，我们可以在 ip route rules 里找到：



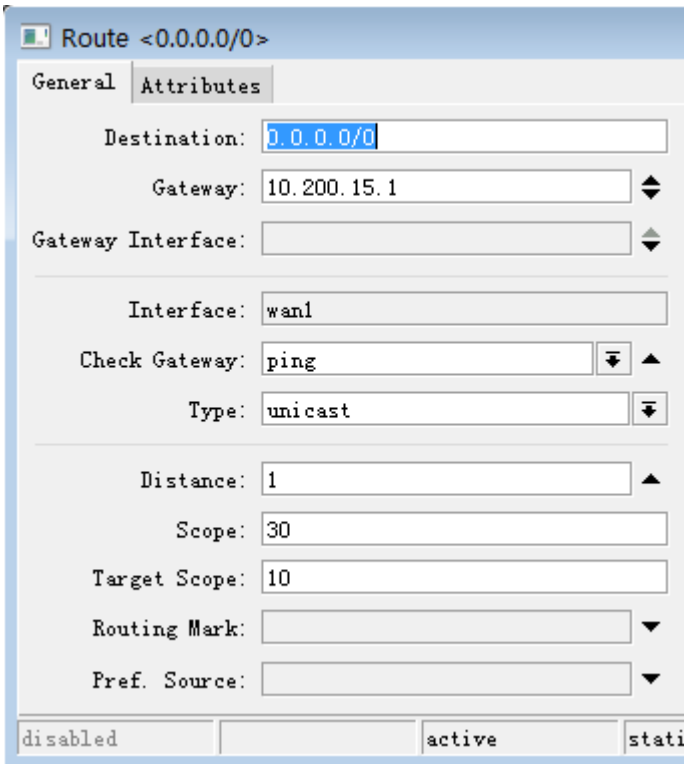
之后我们在 ip route 中 routing-mark 选择对应的 cnc 路由表



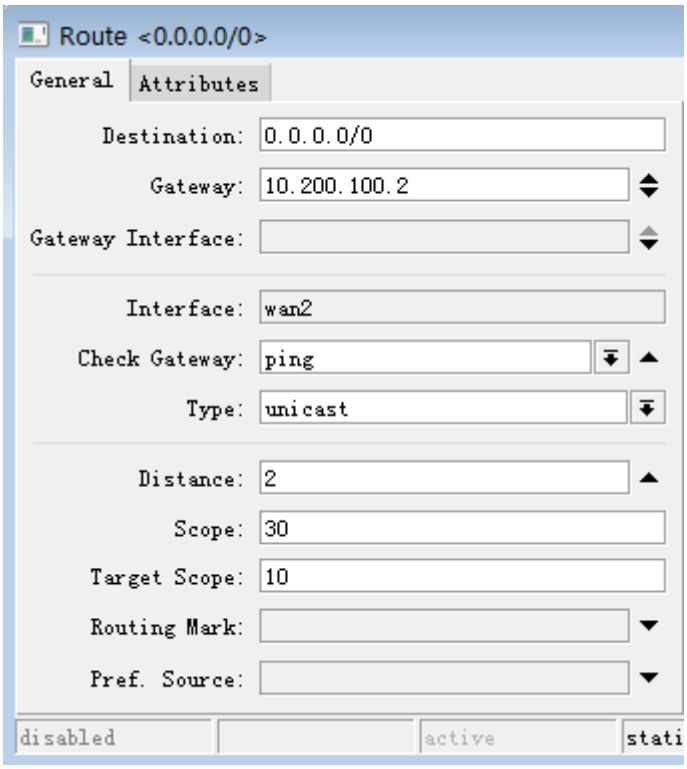
6.6 默认网关断线处理

在多线路情况下，我们可以通过配置备份线路，避免默认网关异常中断，可以用其余的线路进行备份，即配置默认网关和备份网关，我们通过定义 **distance（路由距离）**对多个网关进行备份，根据 distance 来判断 1 为最优先，2 其次，依次类推。

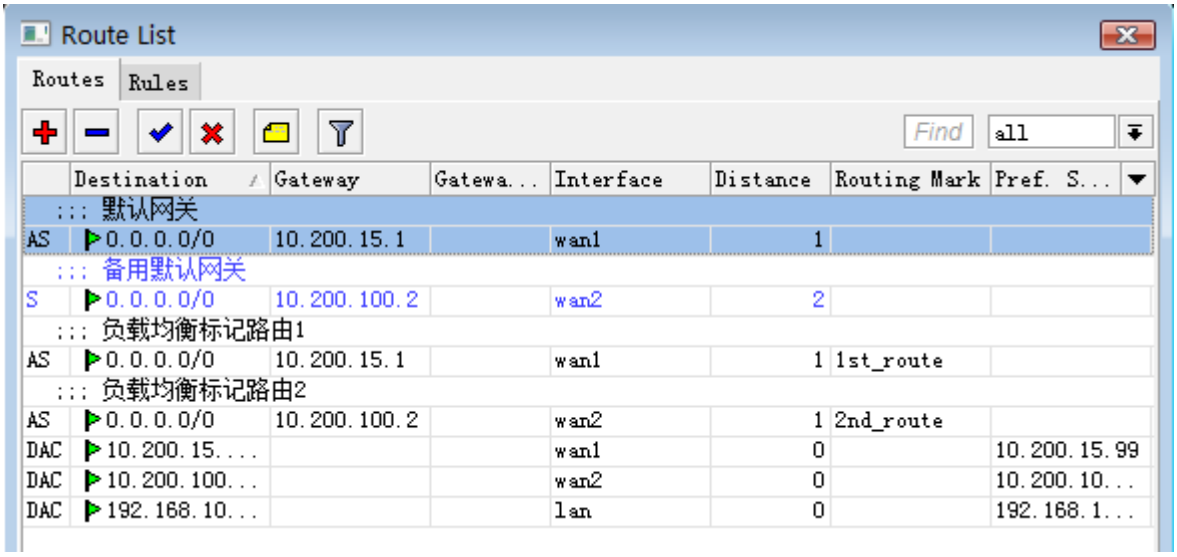
如下图：默认网关的 **distance** 设置为 1，并设置 check-gateway=ping，通过 ping 监测网关状态：



备份网关的 **distance** 设置为 2，并设置 check-gateway=ping，通过 ping 监测网关状态：



配置完成后的路由标如下图：



6.7 RouterOS 策略路由介绍

策略路由对于基 Linux 系统开发的 RouterOS 来说，是维护两个以上的路由表，将需要指定的路由规则放入新建的路由表，完成策略路由规则的配置，因此通过创建多个路由表，来灵活调用路由规则。创建路由表三个菜单下配置，策略路由可以让我们有选择性的调度 IP 数据报到不同的网关出口。/ip firewall mangle mark-routing

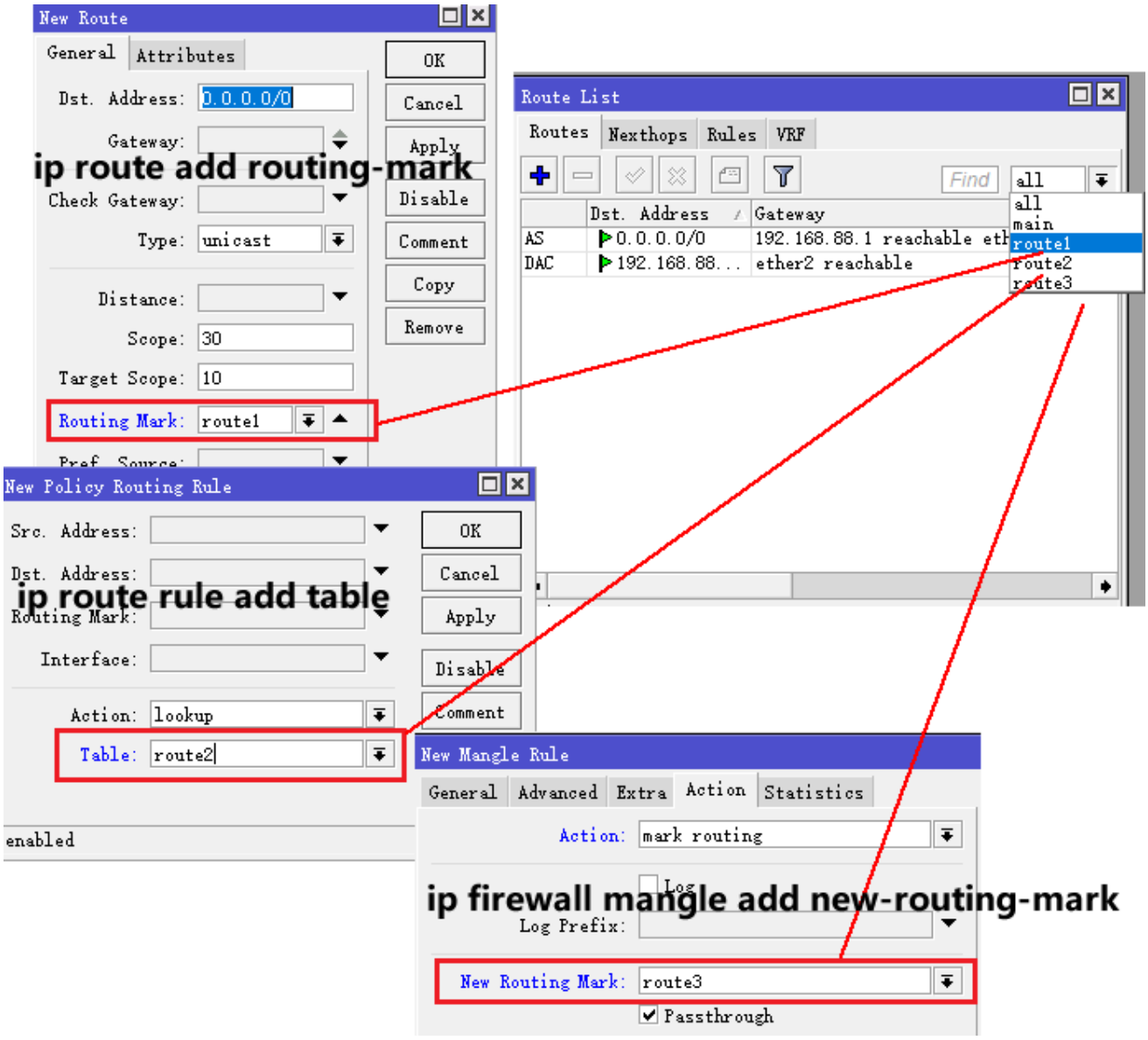
- /ip route routing-mark

- /ip route rule

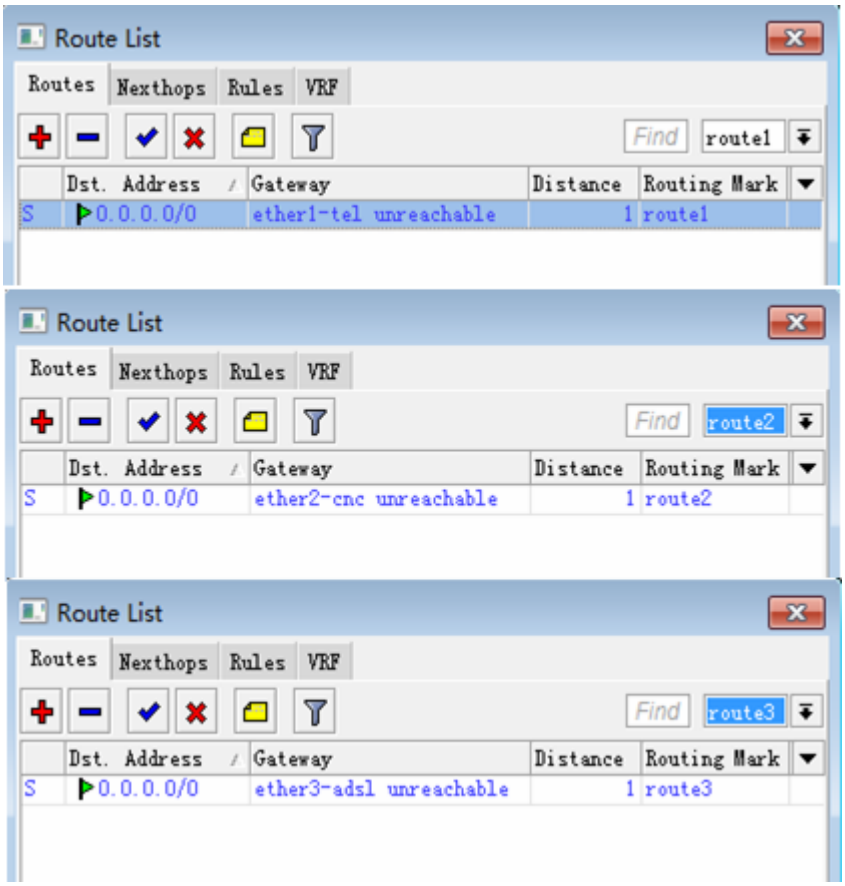
他们创建路由表之间关系是平等的：

mark-routing = routing-mark = table

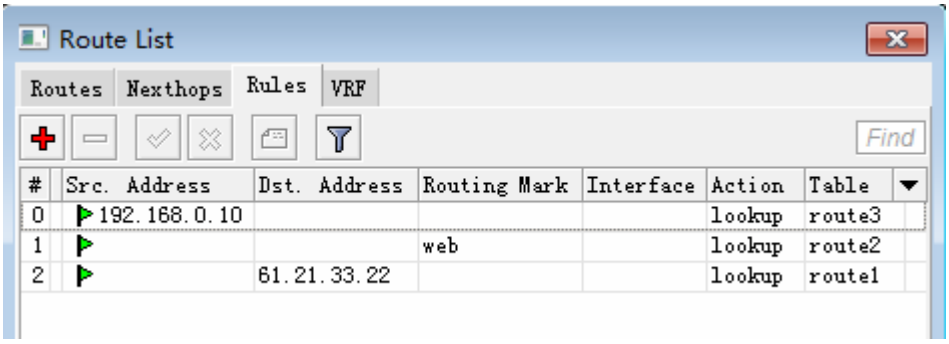
通过下面的操作创建三个策略路由表，分别在 ip firewall mangle、ip route routing-mark 和 ip route rule 中的建立的路由表，可以在 ip route 的右侧列表找到对应的路由表。对于源地址策略路由，端口策略路由，PCC 负载均衡等都会涉及到添加新路由表。



根据这些定义好的路由表 route1、route2 和 route3，给他们设置各自的路由和网关，即每个表有自己的路由

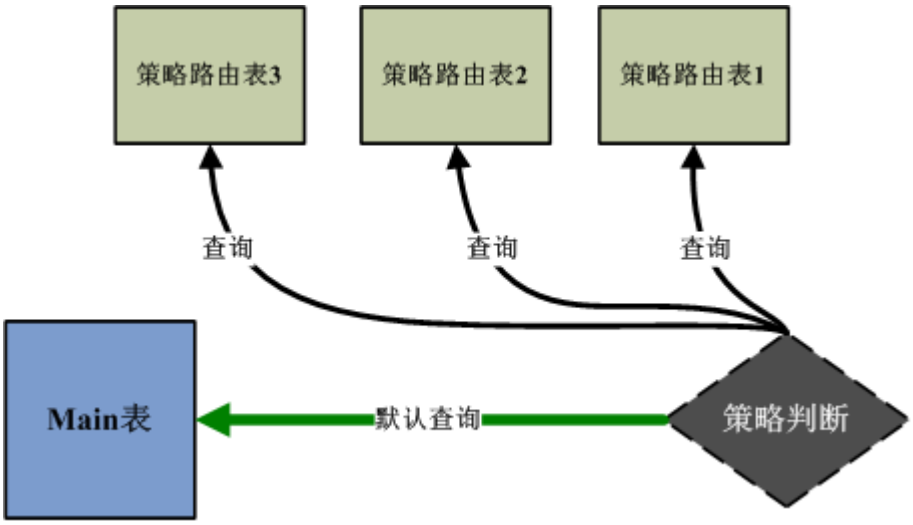


我们可以通过 ip route rules 来分配源地址、目标地址和端口的策略路由，并通过 table 选择各自的路由表，如下图：



注意：在 ip route rule 中的规则是从上往下的执行，最上规则优先执行。

当他们被定义后都会在 ip route 中新建路由表，如图：



当有多个路由表，RouterOS 会首先处理创建的策略路由表，最后剩下的 IP 数据到 Main 表查询，直连路由查询只能在 Main 表完成。

注意：最新 RouterOS 版本的限制，routing make 最大条目数不能超过 251 条。

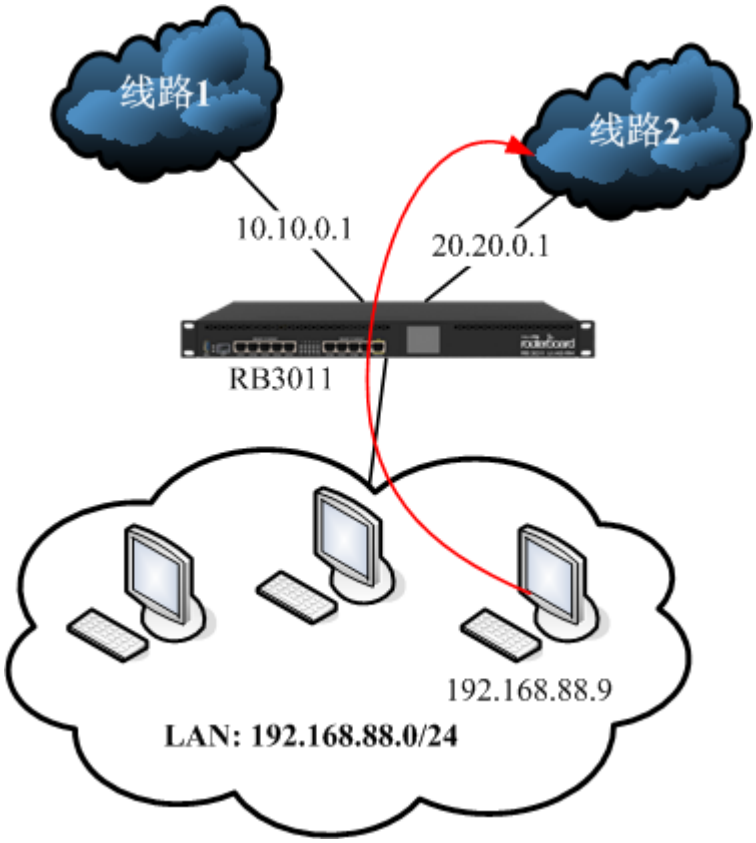
6.8 RouterOS 源地址策略路由

该章节，主要讨论关于源地址策略路由的方法和可能遇到情况，之前已经讲过，RouterOS 策略路由是维护两个或两个以上的路由表，因此创建源地址策略路由也是要新建路由表，在 RouterOS 里配置源地址策略路由方法有两种：

- 基于 ip route rule 直接配置源地址策略，配置简单，可选参数少
- 基于 ip firewall mangle 标记路由策略，有更多的参数可以选择，配置更加灵活

源地址策略路由实例

两种方法都会新建一张路由表，操作方法各有不同，通过下面的简单事例讲解下：



根据上图的一个双线网络结构，配置主机 192.168.88.9 走线路 2，其余主机走线路 1

接口情况如下：

```
Ether1：10.10.0.2/24
Ether2：20.20.0.2/24
Ether3：192.168.88.1/24
```

首先配置 ip 地址

```
[admin@MikroTik] >/ip address
[admin@MikroTik] /ip address> add address=10.10.0.2/24 interface=ether1
[admin@MikroTik] /ip address> add address=20.20.0.2/24 interface=ether2
[admin@MikroTik] /ip address> add address=192.168.88.1/24 interface=ether3
```

添加默认路由，使用 10.10.0.1 作为路由的默认网关

```
[admin@MikroTik] /ip address> /ip route
[admin@MikroTik] /ip route>add gateway=10.10.0.1
```

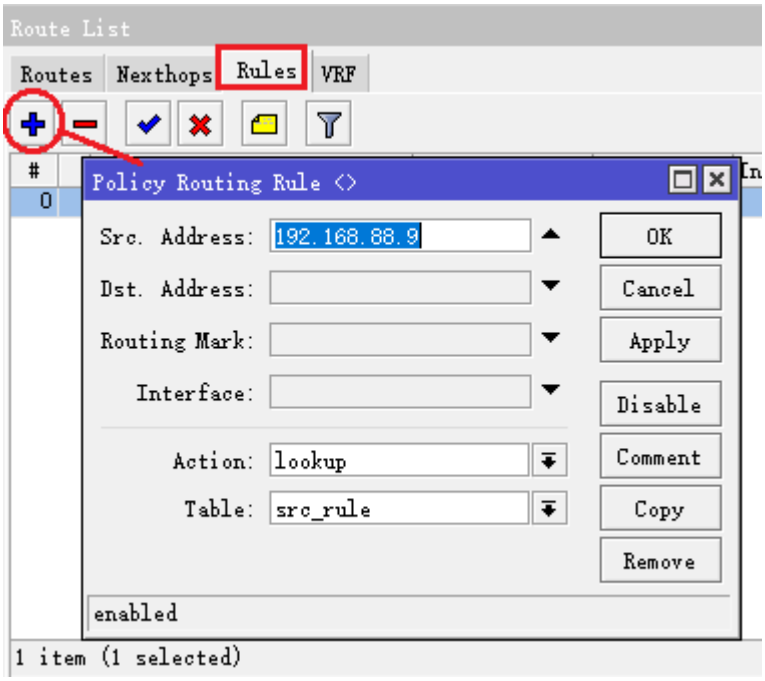
添加 nat 规则

```
[admin@MikroTik] /ip route > /ip firewall nat
[admin@MikroTik] /ip firewall nat > add chain=srcnat action=masquerade
```

方法一：

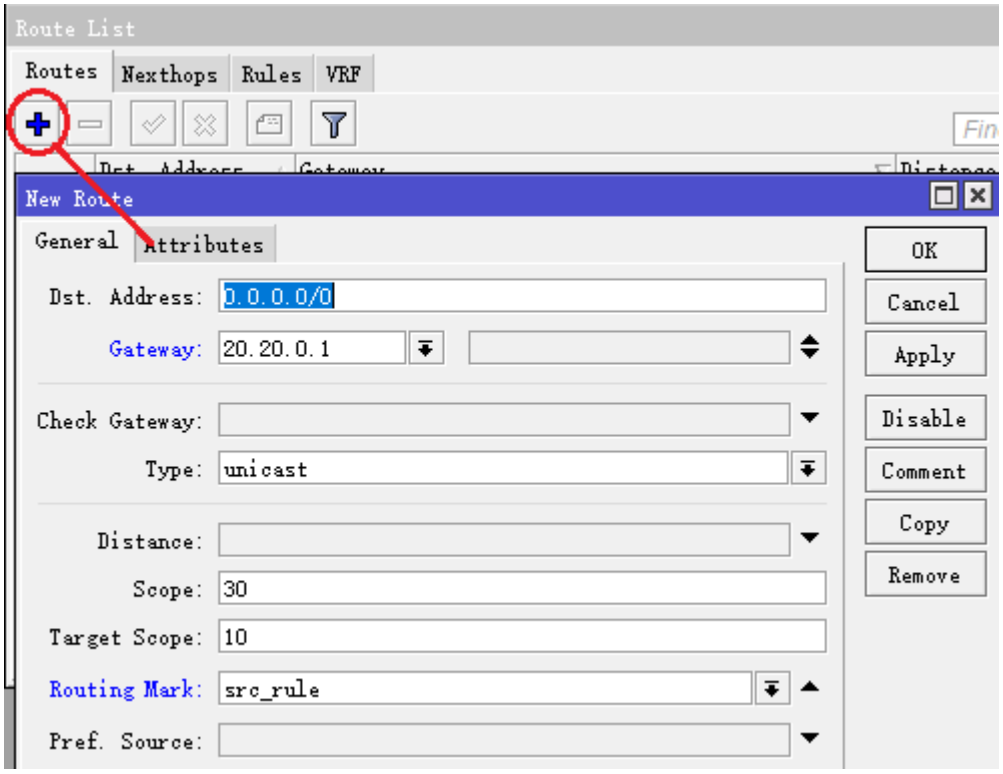
配置 192.168.88.9 源地址策略路由，首先选择 ip route rule 的方法配置

进入 ip route rule，新建一个规则，并设置 src-address=192.168.88.9，table=src_rule



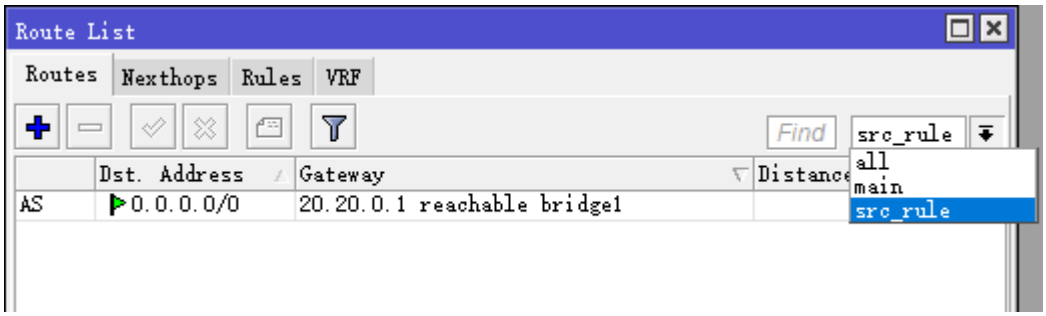
这里新建一个 src_rule 路由表，将源地址为 192.168.88.9 放入进行路由查询

进入 ip route，添加规则，选择 routing-mark=src_rule，即 src_rule 表的网关 20.20.0.1



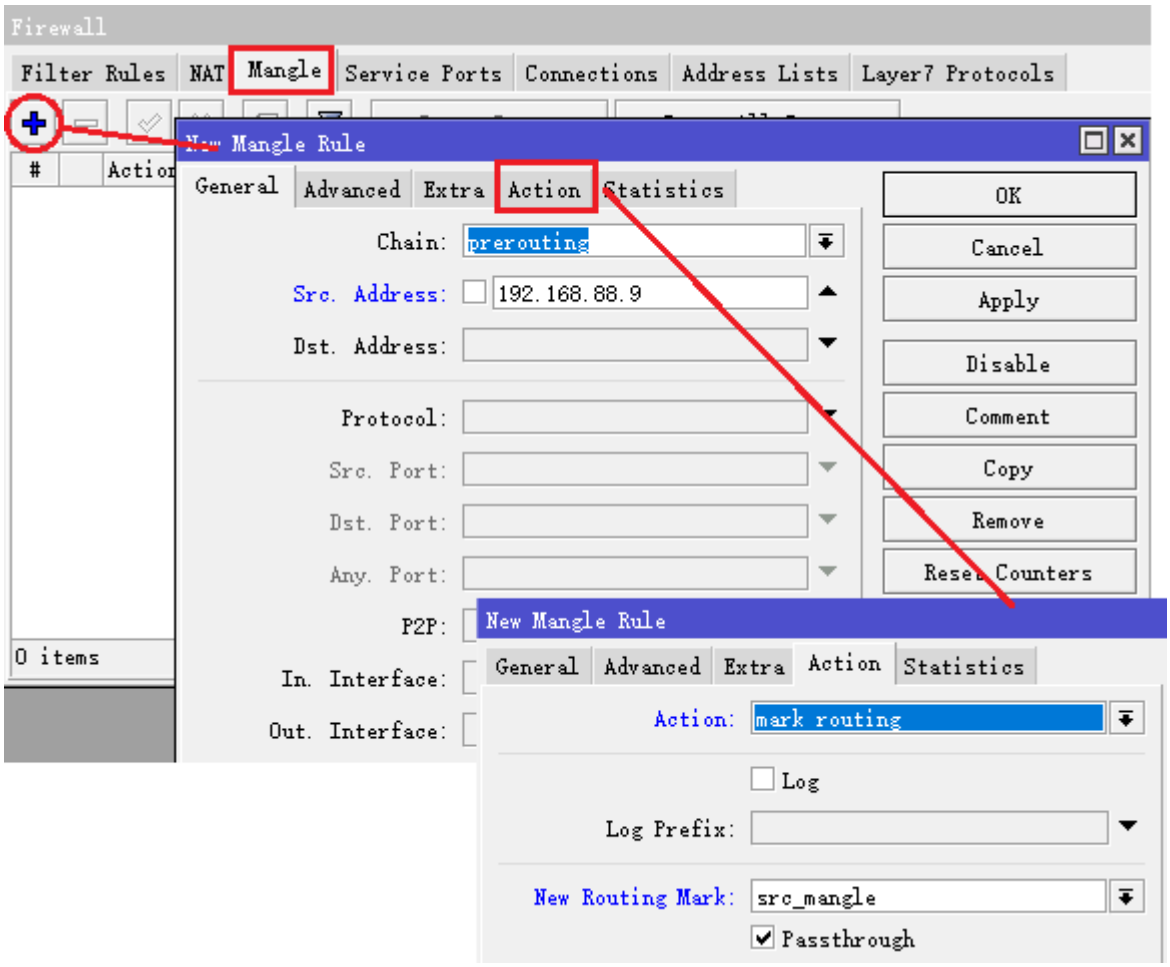
主机 192.168.88.9 的源地址策略路由配置完成。

在 ip route 右上角的下拉菜单，可以选择添加的 src_rule 路由表，以及 main 主路由表

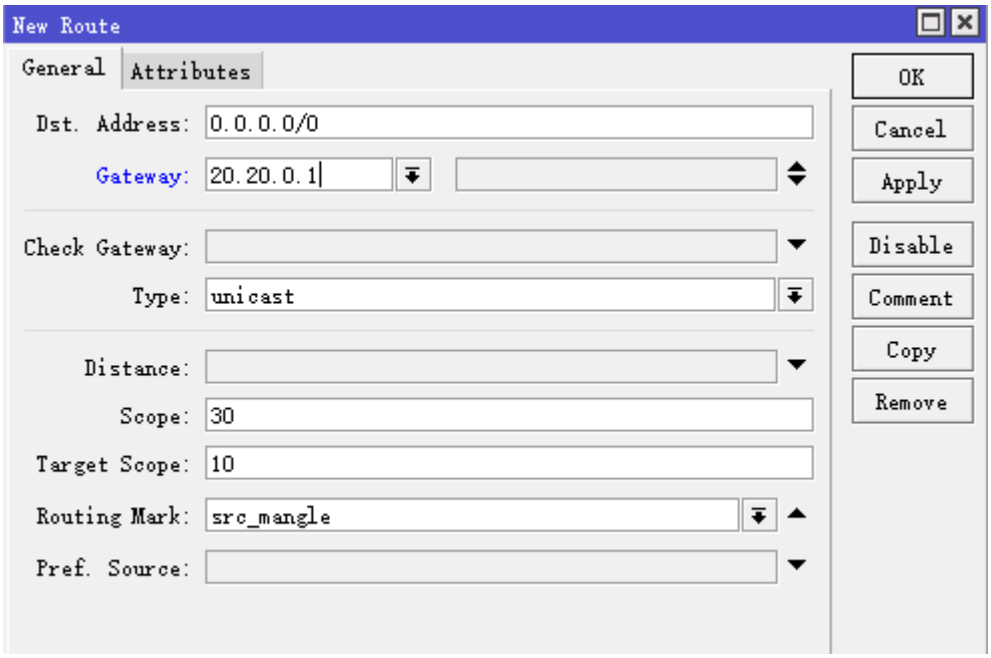


方法二：

接下来看看 mangle 标记的方法，进入 ip firewall mangle 标记源地址 192.168.88.9 的策略，选择链表 prerouting



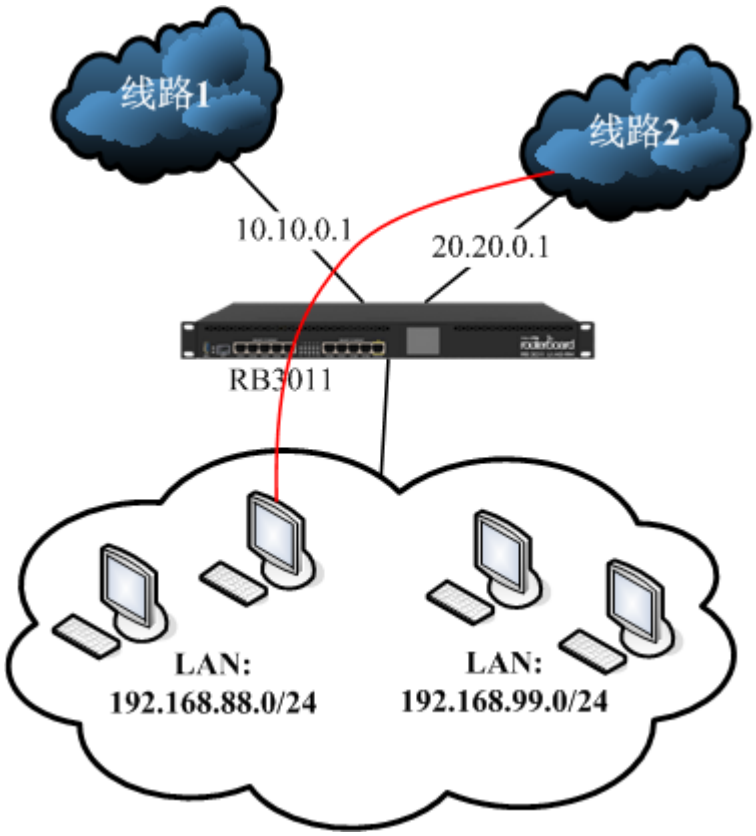
进入 ip route 为 src_mangel 路由表添加网关



以上两种方式实现的结果都是一样的，但可以看到 mangle 添加路由标记时，可选择参数明显比 rule 多很多。

策略路由导致内网段互访失效

在上面的实例中，在路由器上对局域网只配置了一个子网段，因此不存在跨网段的问题，现在我们在路由器上增加一个 192.168.99.0/24 的网段，让 192.168.88.0/24 走线路 2，同时又能访问 192.168.99.0/24 的局域网

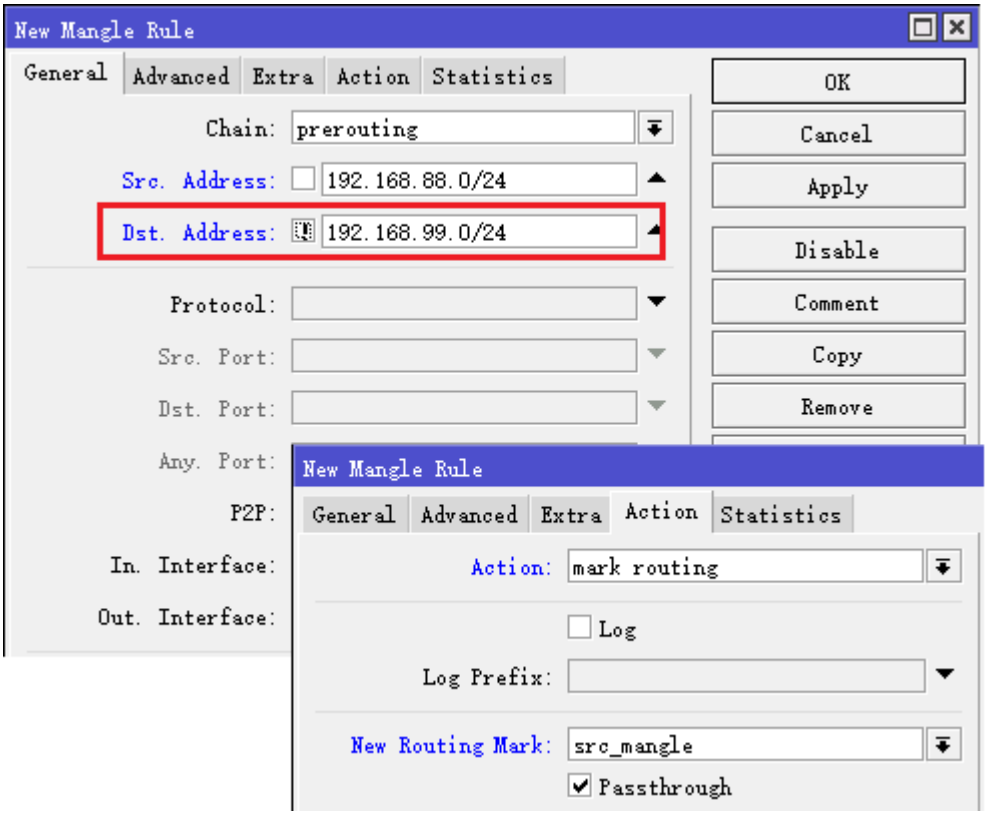


在 ether4 新增一个 192.168.99.1/24 的 ip 地址

```
[admin@MikroTik] /ip address> add address=192.168.99.1/24 interface=ether4
```

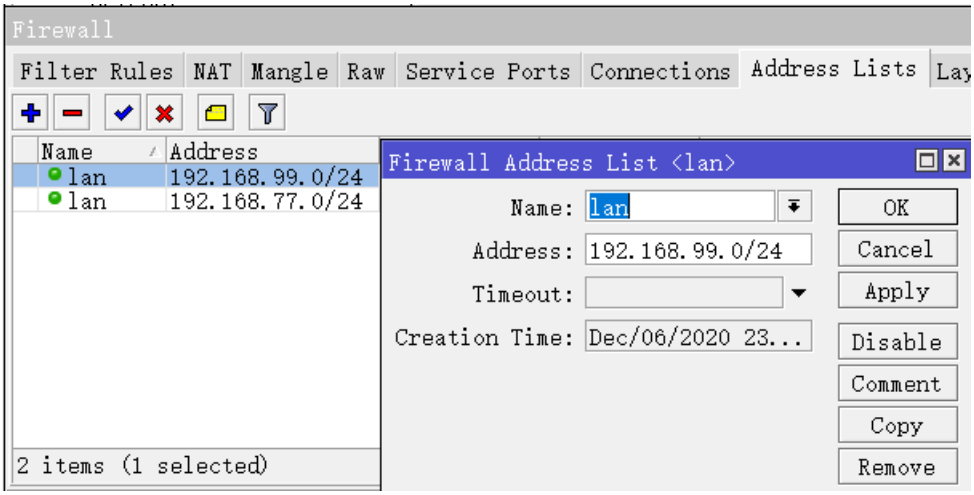
现在假设使用 ip route rule 配置 192.168.88.0/24 源地址走线路 2 出去，会导致 192.168.88.0/24 能通过线路 2 的网关 20.20.0.1 出去上网，但无法通过路由跨网段访问 192.168.99.0/24 的局域网，因为 RouterOS 已经将 192.168.88.0/24 的地址段路由重定向到 20.20.0.1 的网关上，不在查询 main 主路由表。

因此这里我们需要使用 mangle 规则的路由标记，来排除访问 192.168.99.0/24 的路由，除了 192.168.99.0/24 都去 src_mangle 路由表查询，而 192.168.99.0/24 去 main 表查询路由。

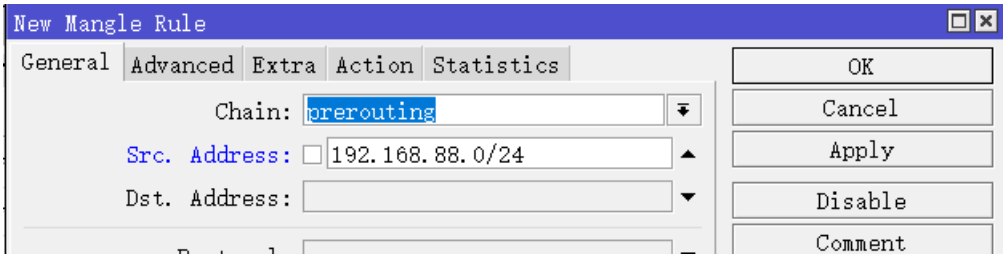


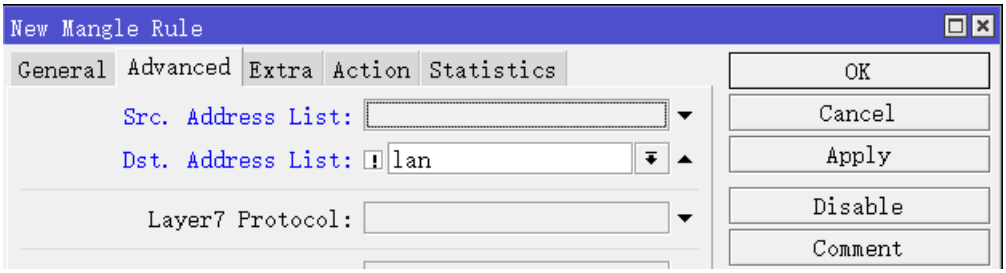
如上图，我们在 dst-address 设置了到目标 192.168.99.0/24，并在前方勾选 "!"（代表非），排除 192.168.99.0/24，通过 mangle 参数就可排除策略路由导致访问内网失效的问题。

如果有多个内网 ip 地址段，可使用/ip firewall address-list，在 address-list 定义一个 lan 的地址列表：



然后在 mangle 规则里调用 lan 地址列表，src-address=192.168.88.0/24，同时并排除 dst-address-list 为 lan 地址列表，

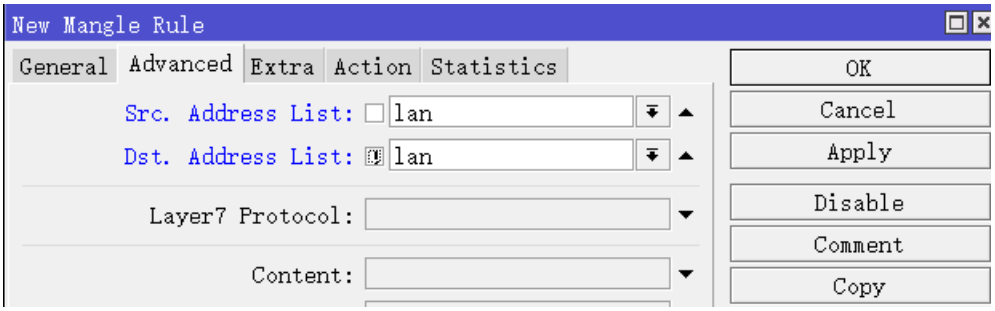




这样排除了目标路由是内网 IP 段的标记，也可用于排除外网某个主机 IP，要求去 main 表查询路由，而不标记到策略路由表查询。

因此基于 mangle 标记的策略路由，可以调用很多参数，适应各种网络情况，这也是 RouterOS 在处理策略路由方面非常灵活的一个表现。

注：该实例，和 PCC 导致内网多 IP 段互通失败问题是同理，也适用于 PCC 的解决方案。PCC 要求内网地址段都做负载均衡，在 mangle 规则里调用 lan 地址列表，src-address-list 为 lan 地址，同时并排除 dst-address-list 为 lan 地址列表：



6.9 奇偶数源地址策略路由标记

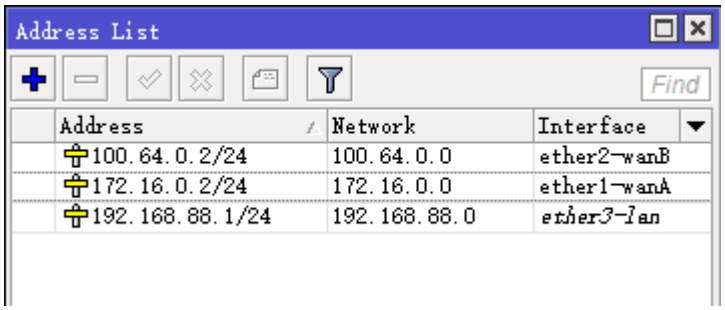
加入一个办公网络，在双线使用相同带宽，相同运营商出口前提下，很早以前的策略路由是通过标记一段地址走一条线路，如内网 IP 地址段是 192.168.10.0/24，将 192.168.10.2-192.168.10.127 前半个 C 走线路 A，剩下的 IP 地址则走线路 B，这样的策略路由在一定的情况下利用效率不高的问题，用户上线虽然是随机的，但不一定均匀分布在各个地址范围，这样可能导致 A 或 B 线路流量不太均衡（不考虑 PCC 负载均衡策略），这样就不会起到流量分配的作用。

为了解决这样的问题，我们通过 RouterOS 的 address-list 建立一个地址列表，分别将奇数和偶数的 IP 地址分开，即奇数的 IP 地址走线路 A，而偶数的 IP 地址走线路 B，这样的策略路由便提高了双线的使用效率。同时这个实例，也为大家提供源地址策略路由配置的一种方法介绍。

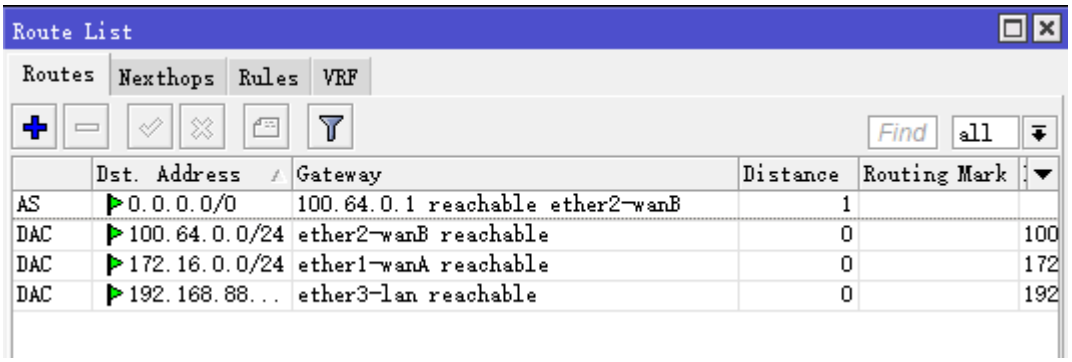
操作步骤如下：

1、 配置好网络的 IP 地址和路由；
2、 在 ip firewall address-list 列表中建立奇数或者偶数地址列表；
3、 进入 ip firewall mangle 通过 src-address-list 标记数据报；
4、 在 ip route 中调用标记好的地址策略，配置路由。

步骤 1：我们首先进入路由器配置 IP 地址，假设我们有两条线路，分别是 wanA 和 wanB，A 的 IP 地址是 172.16.0.2/24，网关是 172.16.0.1；B 的 IP 地址是：100.64.0.2/24，网关是 100.64.0.1。



在 ip route 中配置以线路 B 为默认网关，设置默认路由 100.64.0.1：

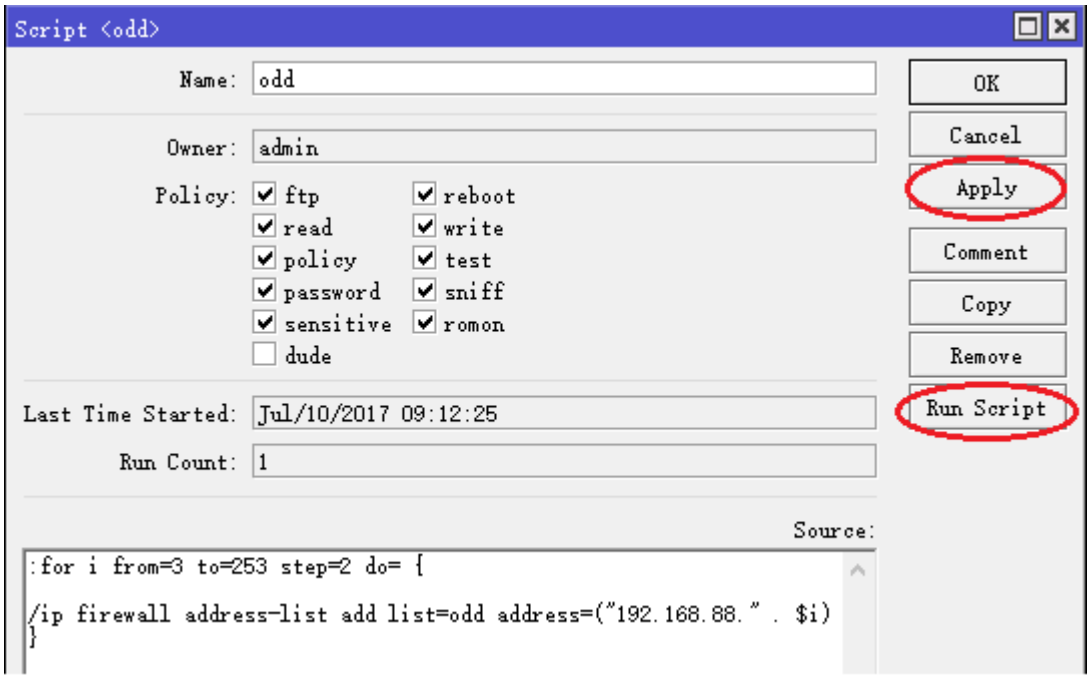


步骤 2：配置好 IP 地址和路由后，接下在 ip firewall address-list 中添加奇数的地址列表，因为是双线路由，我们只需要配置一条奇数的列表，而偶数的列表可以不用配置，直接使用线路 B 的默认路由，因为奇数被标记后，剩下的就为偶数地址。

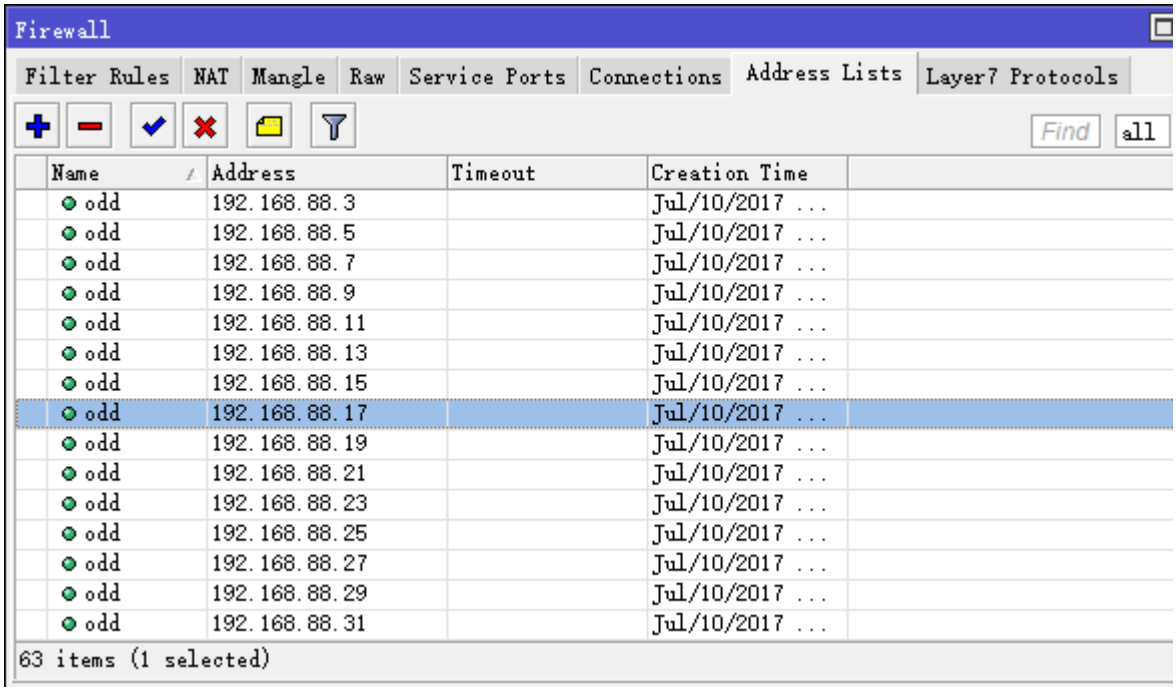
如果你觉得要添加这么多地址很麻烦，可以使用脚本

```
:for i from=3 to=253 step=2 do= {
/ip firewall address-list add list=odd address=("192.168.88." . $i)
}
```

下面是进入/system scripts 里添加的脚本，添加完成后，点 Apply 应用，然后点 Run Script 执行



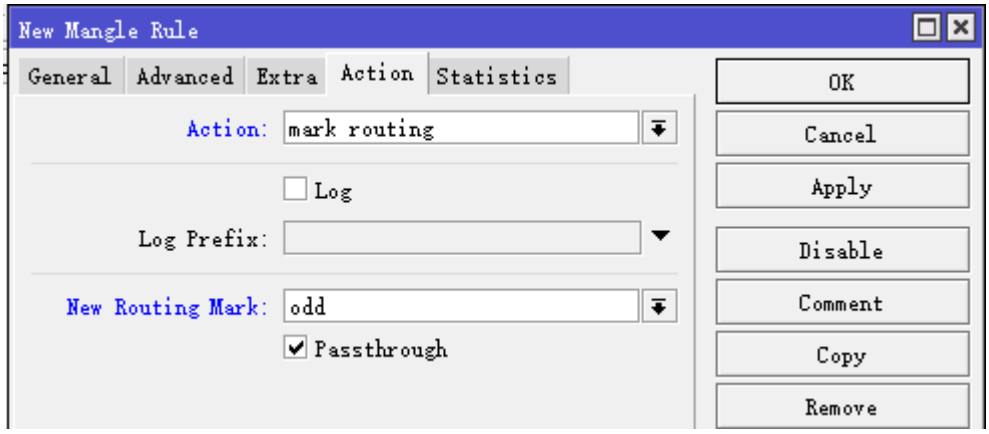
将奇数地址取名为 odd，添加到 address-list 地址列表：



步骤 3：当奇数 IP 地址添加完成后，我们进入 ip firewall mangle 标记路由规则，我们选择链表为 prerouting，下面是 CLI 的执行操作：

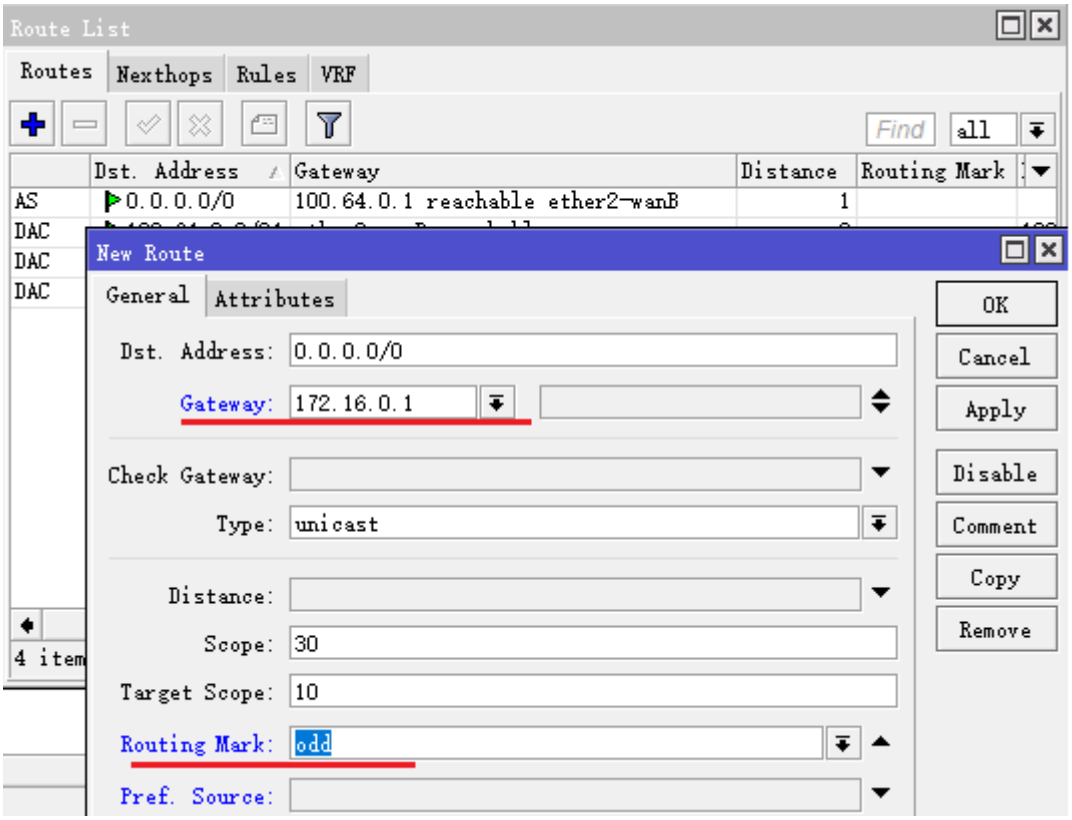
```
[admin@MIKROTIK] /ip firewall mangle> add chain=prerouting action=mark-routing new
-routing-mark=odd src-address-list=odd
[admin@MIKROTIK] /ip firewall mangle> print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=prerouting action=mark-routing new-routing-mark=odd passthrough=yes
     src-address-list=odd
```

下面是 winbox 操作：

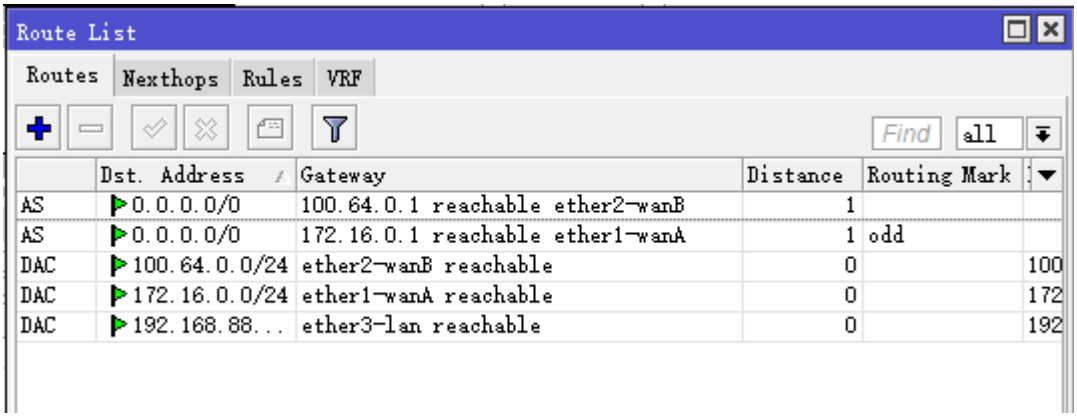


上面的路由标记，只标记了奇数的 ip 地址，剩下的便是偶数的，所以不用再做配置。

步骤 4：配置完标记后，我们进入 ip route 配置路由，我们只需要将奇数的 IP 地址标记到线路 A 的网关即可，操作如下配置地址如下：



配置只需要添加 gateway=172.16.0.1 和 routing-mark=odd，点确认：



奇数的 IP 地址便从 A 线路的 172.16.0.1 的网关出去，剩下的偶数 IP 地址从默认的线路 B 的 100.64.0.1 的网关出去。

6.10 光纤和 ADSL 静态路由

基本情况：假设用户有两条 Internet 线路，一条是使用固定地址的 Un 光纤 2M，另一条是使用 Tel 拨号的 ADSL 通用为 2M。使用 NAT 伪装让局域网共享上网。在路由器上共有 3 块网卡，WAN1 用于 Un 光纤，WAN2 用于 ADSL 拨号，LAN 用于连接内网终端。

首先我们设置 WAN1 与 WAN2 的 IP 地址：ADSL 拨号大致如下：具体参考 PPPoE 设置说明

```
配置 ADSL 线路
/interface pppoe-client 配置 ADSL 拨号信息。
```

```
/interface pppoe-client add name=abcd123456 password=123 interface=WAN2 use-peer-dns=yes
```

注: 设置 pppoe-client 时当得到 ADSL 默认网关后，将 pppoe-client 中的 add-default-route=yes，修改为 **add-default-route=no** 避免自动添加默认的 Tel 路由。

```
[admin@MikroTik] ip address> add address 61.193.77.77/24 interface WAN1
 [admin@MikroTik] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #   ADDRESS            NETWORK        BROADCAST      INTERFACE
  0   61.193.77.77/24   61.193.77.0     61.193.77.255    WAN1
D 1   218.88.32.10/24   218.88.32.1     0.0.0.0          pppoe-out1
[admin@MikroTik] ip address>
```

下面配置内网地址为 192.168.0.1/24：

```
[admin@MikroTik] ip address> add address 192.168.0.1/24 interface LAN
[admin@MikroTik] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #   ADDRESS            NETWORK        BROADCAST      INTERFACE
  0   61.193.77.77/24   61.193.77.0     61.193.77.255    WAN1
D 1   218.88.32.10/24   218.88.32.1     0.0.0.0          pppoe-out1
  2   192.168.0.1/24    192.168.0.0     192.168.0.255    LAN
[admin@MikroTik] ip address>
```

下面我们需要配置一个默认网关，在这里我们以 Un 的 61.193.77.1 网关为默认网关，Tel 的作为静态路由：

```
[admin@MikroTik] ip route> add gateway=61.193.77.1
[admin@MikroTik] ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf
 #     DST-ADDRESS       PREFSRC        G GATEWAY        DISTANCE INTERFACE
 0 ADC 61.193.77.0/24    61.193.77.77                                WAN1
1 ADC 218.88.32.1/32    218.88.32.10                               pppoe-out1
 2 ADC 192.168.0.0/24    192.168.0.1                                LAN
 3 A S 0.0.0.0/0                      r 61.193.77.1                 WAN1
 [admin@MikroTik] ip route>
```

据说明要求设置，Tel 和 Un 双线路由脚本操作方式：

将你的正确的 Tel 或 Un 的网关，使用用编辑-替换掉脚本里的“网关”，然后打开 winbox，点击 Terminal（控制终端）然后复制脚本，并在 Terminal（控制终端）中点右键选择“paste”粘贴脚本，粘贴完后敲回车，也可以生产.rsc 的档上传到路由器的 files 根目录下，通过 import 命了完成操作。

这里我们将 Tel 的网关 218.88.32.1 在“Tel IP 脚本”文本文件中使用替换操作将所有含“网关”的关键词替换为 218.88.32.1，然后复制并在 Terminal 控制台中粘贴脚本。这样 Tel 脚本即可导入。

```
[admin@MikroTik] ip route> prin
Flags: X - disabled, A - active, D - dynamic,
```

```
C - connect, S - static, r - rip, b - bgp, o - ospf
 #     DST-ADDRESS       PREFSRC        G GATEWAY        DIS    INTERFACE
0 ADC 61.193.77.0/24    61.193.77.77                            WAN1
1 ADC 218.88.32.1/32    218.88.32.10                         pppoe-out1
2 ADC 192.168.0.0/24    192.168.0.1                             LAN
3 A S 0.0.0.0/0                      r 61.193.77.1             WAN1
4 A S 218.4.0.0/15                    r 218.88.32.1       pppoe-out1
5 A S 218.6.0.0/16                    r 218.88.32.1       pppoe-out1
6 A S 218.13.0.0/16                   r 218.88.32.1       pppoe-out1
7 A S 218.14.0.0/15                   r 218.88.32.1       pppoe-out1
8 A S 218.16.0.0/14                   r 218.88.32.1       pppoe-out1
9 A S 218.20.0.0/16                   r 218.88.32.1       pppoe-out1
10 A S 218.21.0.0/17                  r 218.88.32.1       pppoe-out1
11 A S 218.22.0.0/15                  r 218.88.32.1       pppoe-out1
12 A S 218.30.0.0/15                  r 218.88.32.1       pppoe-out1
13 A S 218.62.128.0/17                r 218.88.32.1       pppoe-out1
14 A S 218.63.0.0/16                  r 218.88.32.1       pppoe-out1
15 A S 218.64.0.0/15                  r 218.88.32.1       pppoe-out1
16 A S 218.66.0.0/16                  r 218.88.32.1       pppoe-out1
.....
```

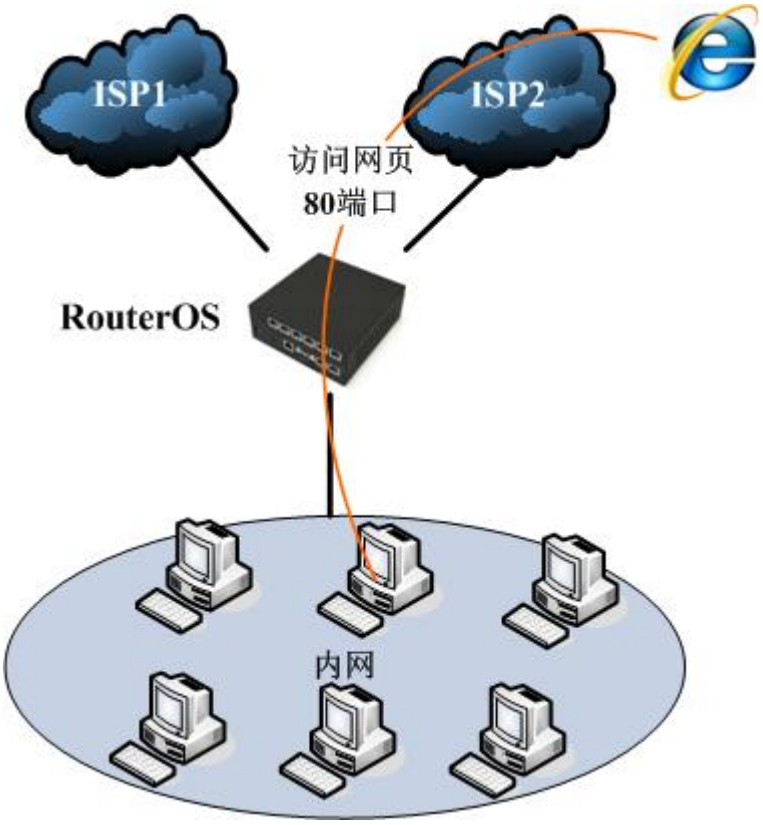
除了使用 check-gateway 监测网关外，还可以使用 netwatch 判断，当一条线路断线时，在/tool netwatch 中设置主机网关监控（具体设置参考 Network 监控），并配置脚本，如当你使用静态路由指定 Un 或 Tel 线路的时候，其中一条线路出现故障，需要切换到另外一条线路时我们需要设置以下脚本，如 Tel 的线路出现故障，需要禁用掉 Tel 网关的静态路由策略，让所有的数据走默认的 Un 线路，Tel 网关为：222.212.48.1。脚本设置如下

```
当 Tel 线路出现故障的时候，禁用掉所有到 Tel 网关的策略
:foreach i in=[/ip route find gateway=218.88.32.1] do={/ip rout disable $i}
当 Tel 线路正常后，启用所有 Tel 策略
:foreach i in=[/ip route find gateway=218.88.32.1] do={/ip rout enable $i}
```

以上脚本分别写入 netwatch 框的 UP 和 DOWN 中。

6.11 HTTP 端口的策略路由

MikroTik RouterOS 可以支持多种策略路由，如我们常见的源地址、目标地址，同样支持端口的策略路由，多种规则可以根据用户情况配合使用，如下图：

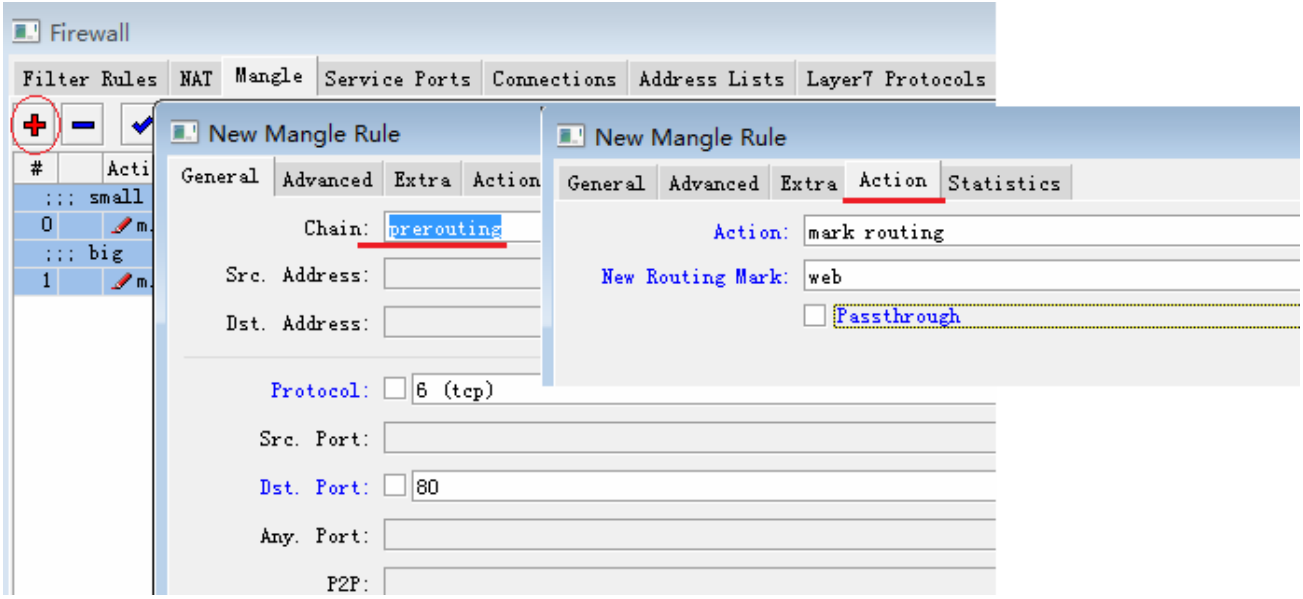


网络情况：

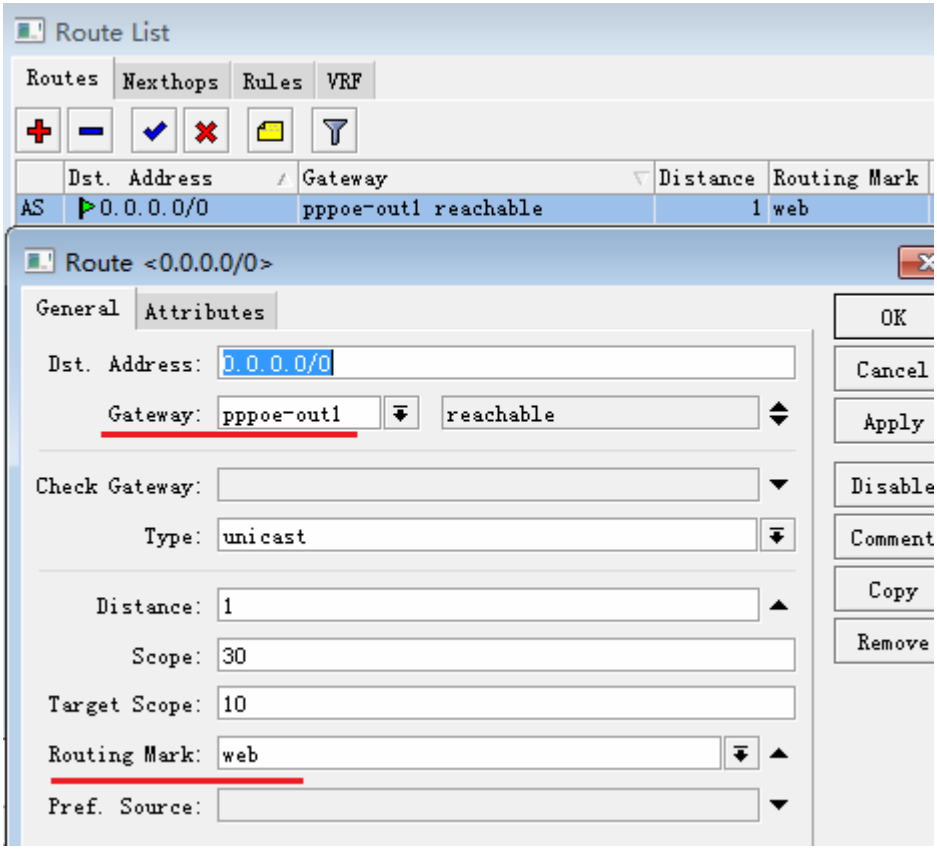
我们有两个 ISP 接入的线路，一个是 ISP1 通过光纤接入，另外一个 ISP2 的 PPPoE 拨号的 ADSL 连接，我们需要通过将访问网页的数据都转移到 PPPoE 拨号上，其他的的数据默认走 ISP1 的光纤（注意，转移 80 端口网页到 ADSL 上，也要注意 DNS 的配置，因为 DNS 解析关系到网站关联的 IP 地址，最好设置 ADSL 的分配的 DNS）。

实际配置

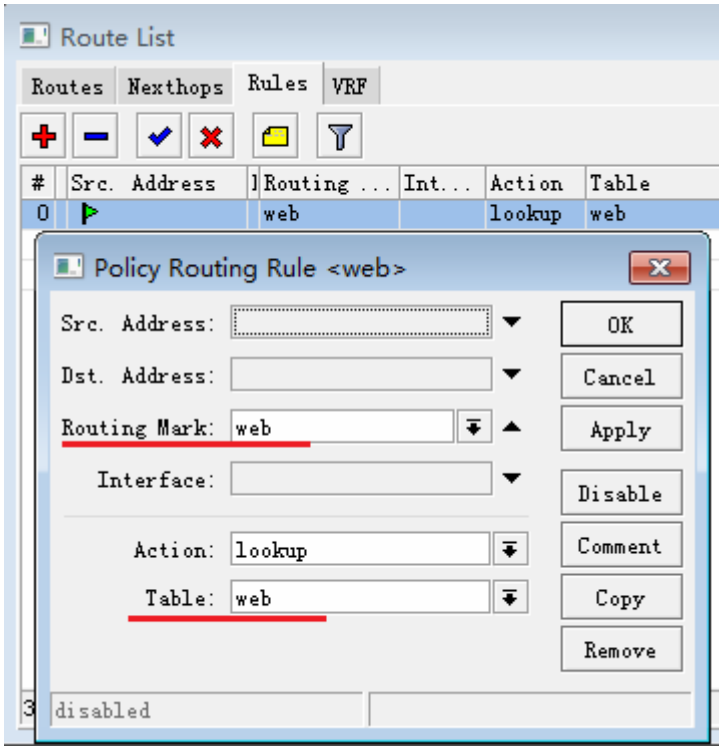
现在我们定义访问网页的端口，访问网页的端口是 TCP 80 端口，我们进入/ip firewall mangle 中做数据标记，从标记中提取路由标记，命名为“web”，因为我们在前面的连接标记中做过了 passthrough 的设置，在这里就不用在重复设置。



然后我们进入/ip route，配置路由我们让标记好的 80 端口通过 pppoe-out1 出去：



在这里，如果在 ip route rule 里有其他的策略规则出现，我们最好是在/ip route rule 里再次定义 80 端口的规则：



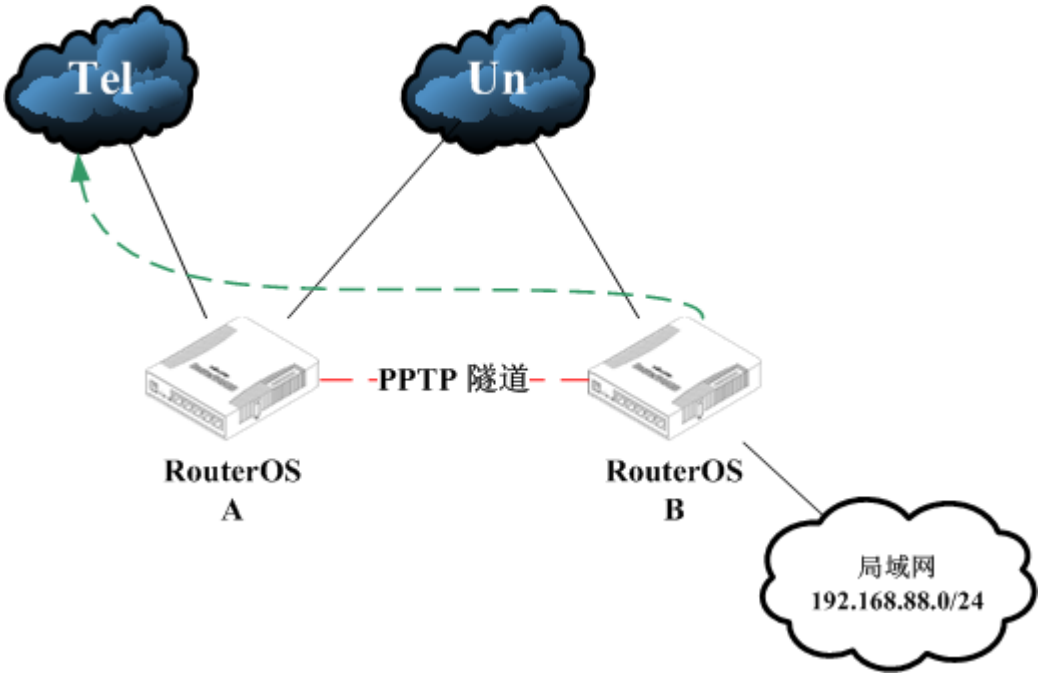
在 ip route rules 定义的 web 标记在 web 路由表中去查找路由。

注：当配置运营商 T 或运营商 U 的双线，通过策略路由将 TCP/80，即网页指定到其中一条线路后出现网页打开慢的问题，需要注意以下几点：

1. 如果你设置 TCP/80 的网页走运营商 T 线路，请将用户请求的 DNS 设置为运营商 T 的 DNS，因为 DNS 用运营商 T 会解析返回的 IP 是运营商 T 的服务节点 IP，如果用运营商 U 的 DNS 则会导致解析 IP 地址是运营商 U，但却走到运营商 T 线路，导致延迟过大或掉包。
2. 确认 DNS 的 IP 地址是否走到正确的运营商 T 或运营商 U 线路。

6.12 PPTP VPN 借线路由操作

假设如下面的网络拓扑，有两台 RouterOS 设备 A 和 B，RouterOS A 是双线路由器，有运营商 Tel 和 Un 两条线路，并做了以运营商 Un 为主，运营商 Tel 为静态路由策略设置。而另一台 RouterOS B 接入的单线路由器接入了运营商 Un 的线路，并且想通过 PPTP VPN 隧道的方式借用双线路由器的 Tel 线路。

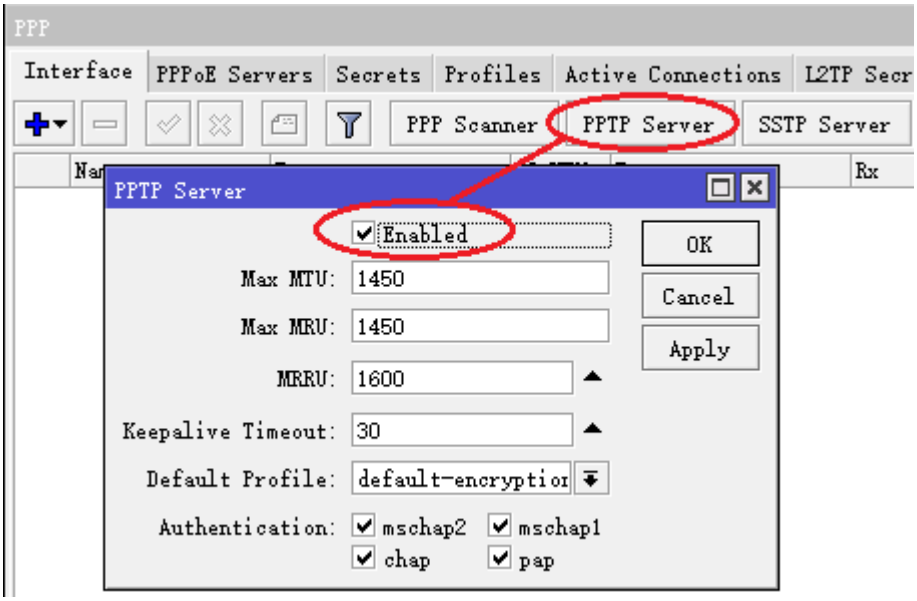


根据上面的案例，接入点 A 和 B 他们都是共同使用了 Un 的线路，本示例中在 Un 网络中两个点之间的延迟小于 10ms，且有足够的上行带宽，低延迟才能保证良好的网络给 B 做 Tel 的访问，作为 A 点点 Un 网络尽可能为固定 IP，如果无法保障，确保 Un 网络分配的是公网 IP，可以采用 Cloud 做 DDNS 服务。

首先建立从接入点 B 到 A 的 PPTP 隧道，在接入点 A 设置 PPTP 服务器，在接入点 B 设置 PPTP 客户端。这里假设接入 A 点的 Un IP 地址为 192.168.10.10，B 点 Un 地址为 192.168.10.12。A 路由器的默认网关为 Tel 线路，B 路由器下局域网 ip 地址为 192.168.88.0/24，需要使用 A 路由器的 Tel 线路

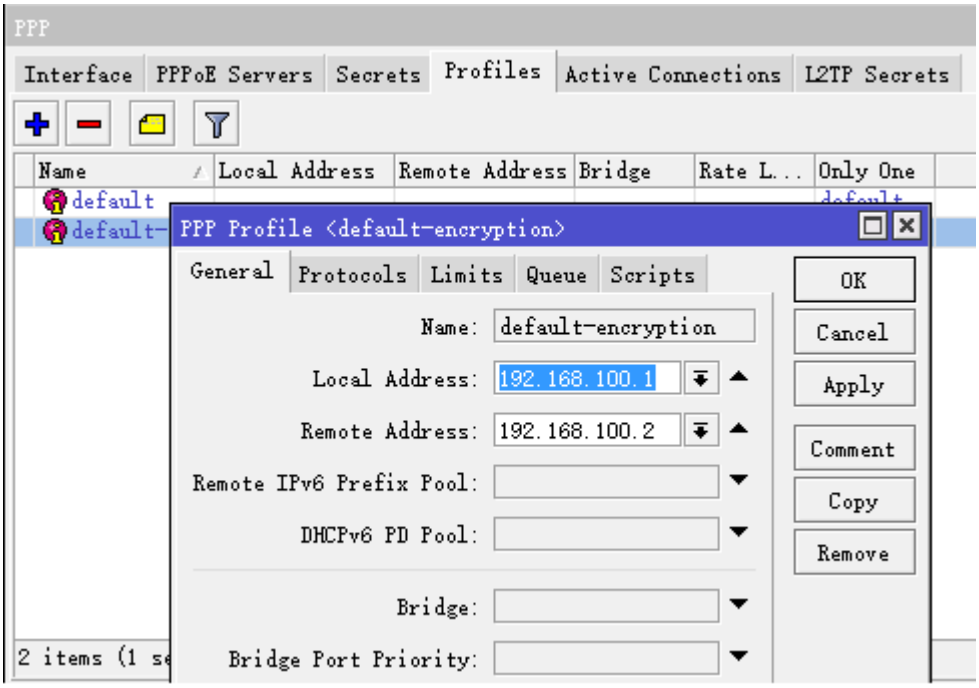
配置 PPTP-Server

在接入点 A 启用 PPTP-Server



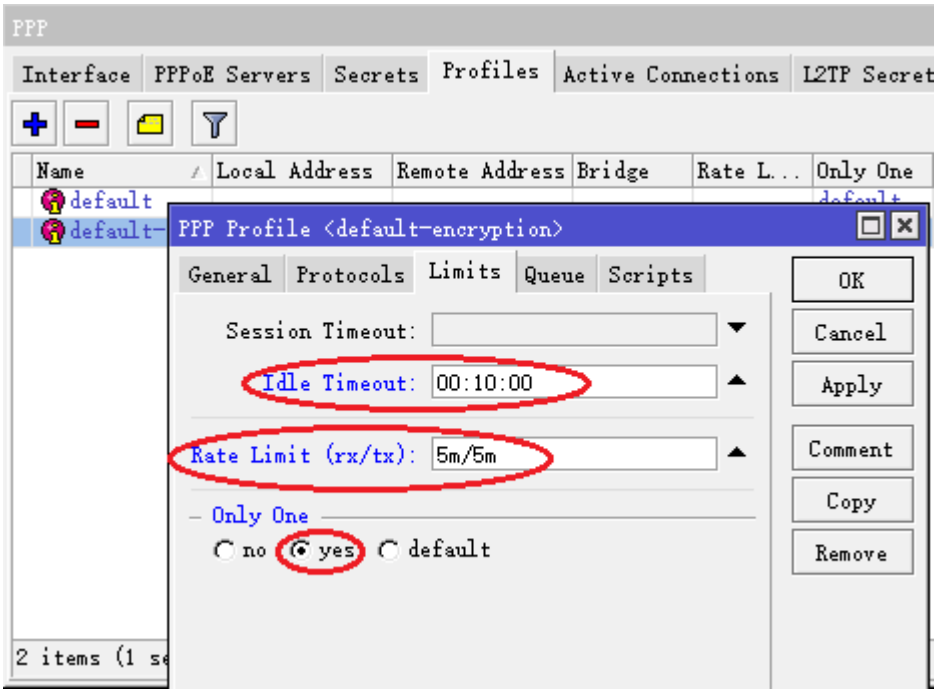
在这里 Default-Profile 我们采用 default-encryption，同样你也可以在 PPTP-Server 的 profiles 中创建自己的规则。Keepalive-Timeout 是 PPTP-Server 主动使用 ICMP 协议探测客户端是否在线，如果客户端使用了防火墙或禁止 ICMP 探测，那无法探测到客户端，Server 就会主动断开该客户端的连接，这个设置需要用户自己根据网络情况判断是否开启 Keepalive-Timeout。

设置 Profile 定义客户和主机的访问地址：



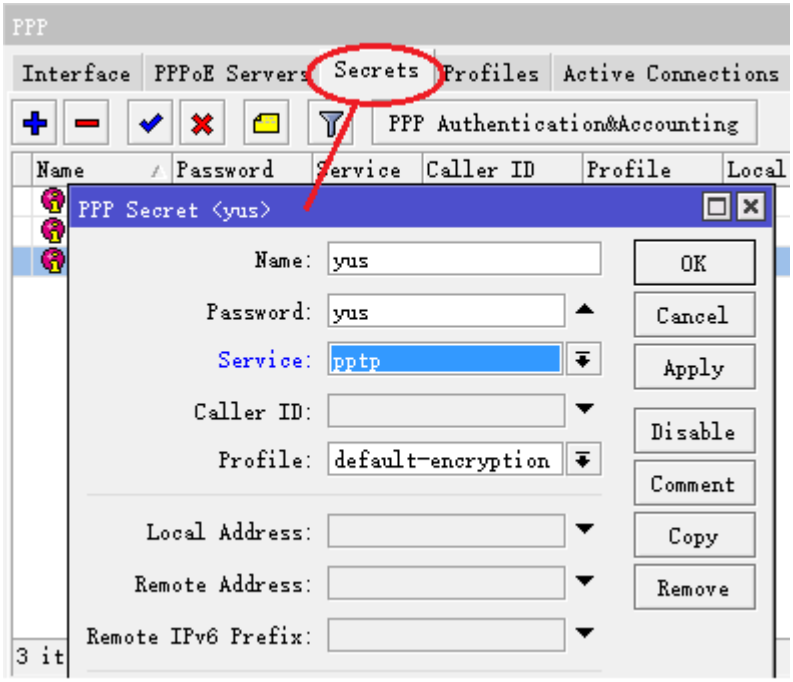
在这里我们给 PPTP-Server 分配的 IP 地址为 192.168.100.1(local-address) ，给客户端分配的地址为 192.168.100.2(remote-address) 。分配 IP 地址也可以通过账号设置 Secrets 进行，在这里我们只有一个客户端所有可以直接通过 profile 中的规则设置，如果有多个客户端也可以通过/ip pool 中的地址池做多 IP 的动态分配。

如果你需要控制每个拨入线路的带宽，可以配置 limit 参数：



在 limit 参数中，idle-timeout 设置用户空闲超时时间为 10 分钟，用户没有任何连接数据就断开。Rate-limit 是对该类用户的流量控制这里设置的上行为 5Mb，下行 5Mb 的带宽。最后是 only-one 该账户是否为唯一，这里设置为 yes。

进入 Secrets 菜单，设置客户端的账号密码：



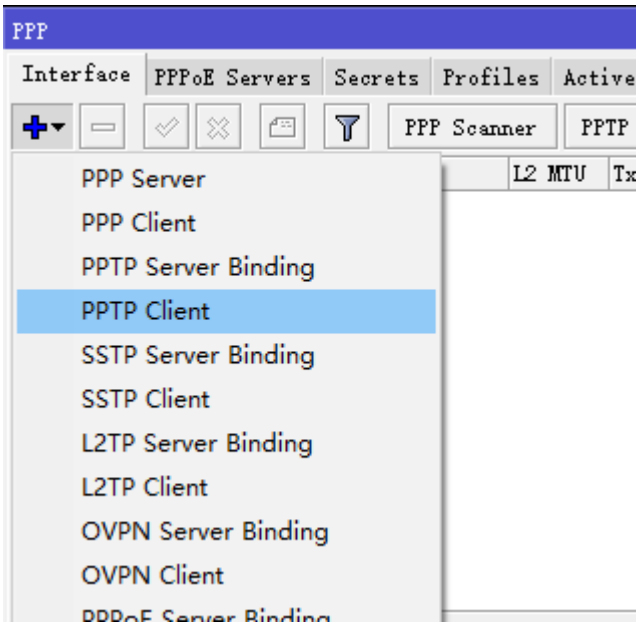
进入 secret 设置账号和密码以及相关信息，设置好 name 和 password 后，选择 service 服务类型为 pptp，profile 规则为 default-encryption。这样 PPTP-Server 就已经设置完成。

确保 PPTP-Server 的 nat 配置正确，最简单的 nat 配置可以通过创建默认 masquerade 规则：

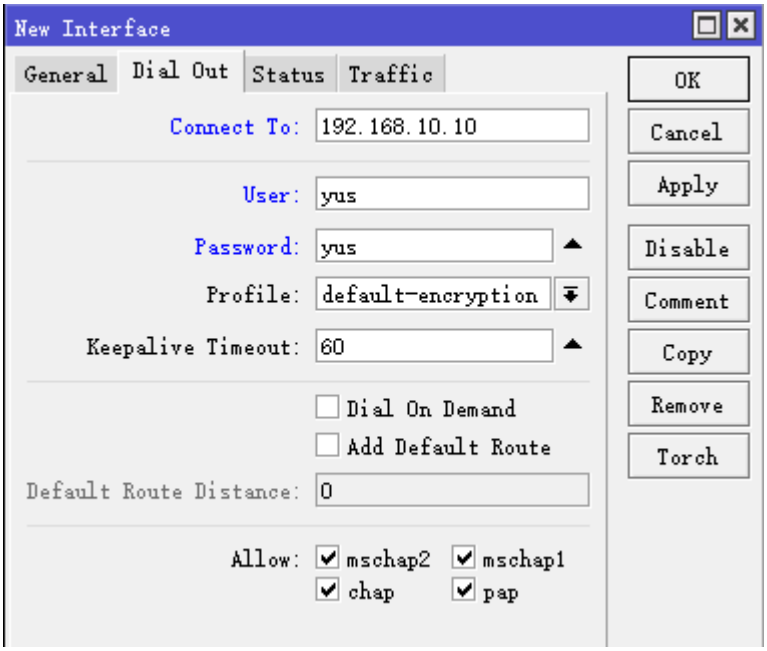
```
/ip firewall nat add chain=srcnat action=masquerade
```

配置 PPTP-Client

完成 PPTP 服务设置后，现在开始设置接入点 B 的 PPTP-Client，进入 PPP 菜单，点击加号添加 PPTP-Client：



进入带宽 PPTP-client 对话框后，选择 dial-out 设置 PPTP 拨号信息，在 server-address 的地址为 192.168.10.10 级接入点 A 的 Un 地址：



设置账号和密码分别为 yus，记住不能勾选 add-default-route 参数，以上 PPTP 连接完成，这样即可连接到 A 点的 PPTP-Server 连接。

路由配置

完成 PPTP VPN 隧道连接后，前提确保路由和 nat 规则正确，如果你对静态路由不熟悉，建议在 A 和 B 路由器上都配置默认的 nat 规则，确保 A 路由器的默认路由为 Tel 线路。如果你需要做静态路由回指，那就需要将 B 路由器下局域网 IP 在 A 路由器上做回指，那这样只需在 A 路由器配置 nat 规则，而 B 路由器不用。

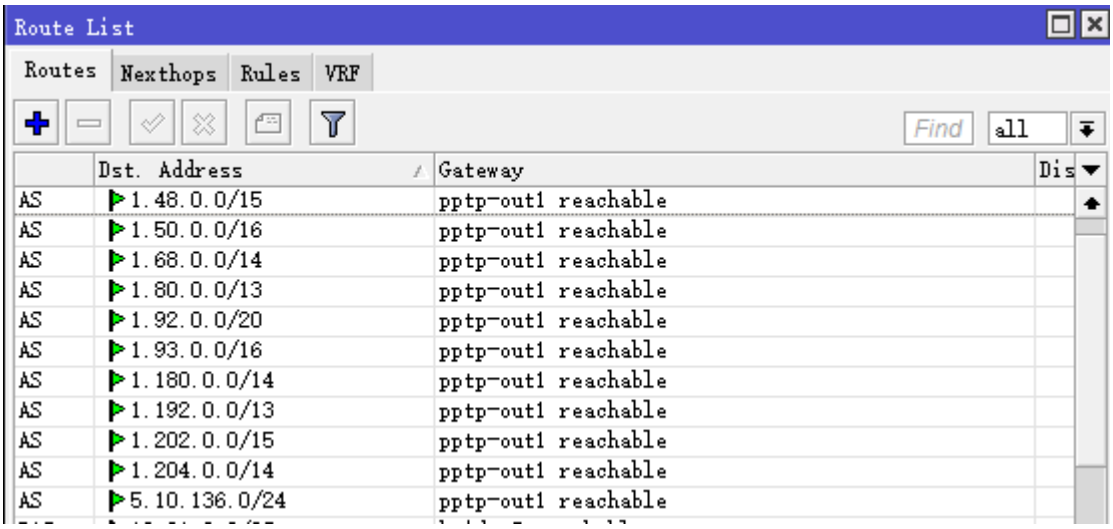
基本互联配置完成后，我们可以从 B 路由器上去调用 A 路由器的 Tel 线路。调用方式我们可以选择静态路由，配置运营商 Tel 路由表，或者通过策略路由指定内网某 IP 主机走 Tel 线路。

静态路由配置

下面是 Tel 的静态路由脚本配置，我们设置网关为 pptp-out1

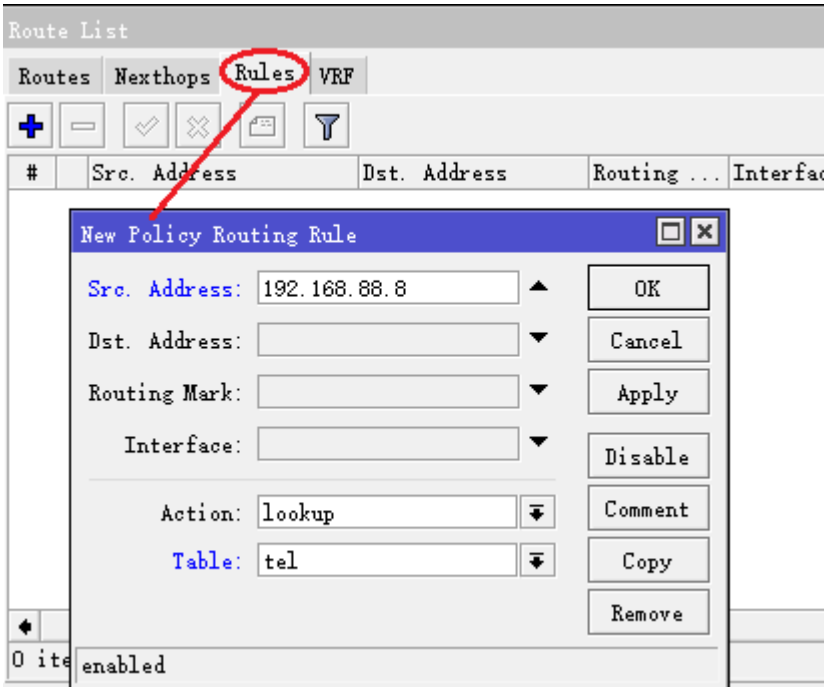
```
/ip route
add   disabled=no dst-address=1.48.0.0/15 gateway=pptp-out1
add   disabled=no dst-address=1.50.0.0/16 gateway=pptp-out1
add   disabled=no dst-address=1.68.0.0/14 gateway=pptp-out1
add   disabled=no dst-address=1.80.0.0/13 gateway=pptp-out1
add   disabled=no dst-address=1.92.0.0/20 gateway=pptp-out1
add   disabled=no dst-address=1.93.0.0/16 gateway=pptp-out1
add   disabled=no dst-address=1.180.0.0/14 gateway=pptp-out1
add   disabled=no dst-address=1.192.0.0/13 gateway=pptp-out1
add   disabled=no dst-address=1.202.0.0/15 gateway=pptp-out1
add   disabled=no dst-address=1.204.0.0/14 gateway=pptp-out1
add   disabled=no dst-address=5.10.136.0/24 gateway=pptp-out1
add   disabled=no dst-address=14.16.0.0/12 gateway=pptp-out1
…
```

通过命令行导入后，在 winbox 生成路由表：

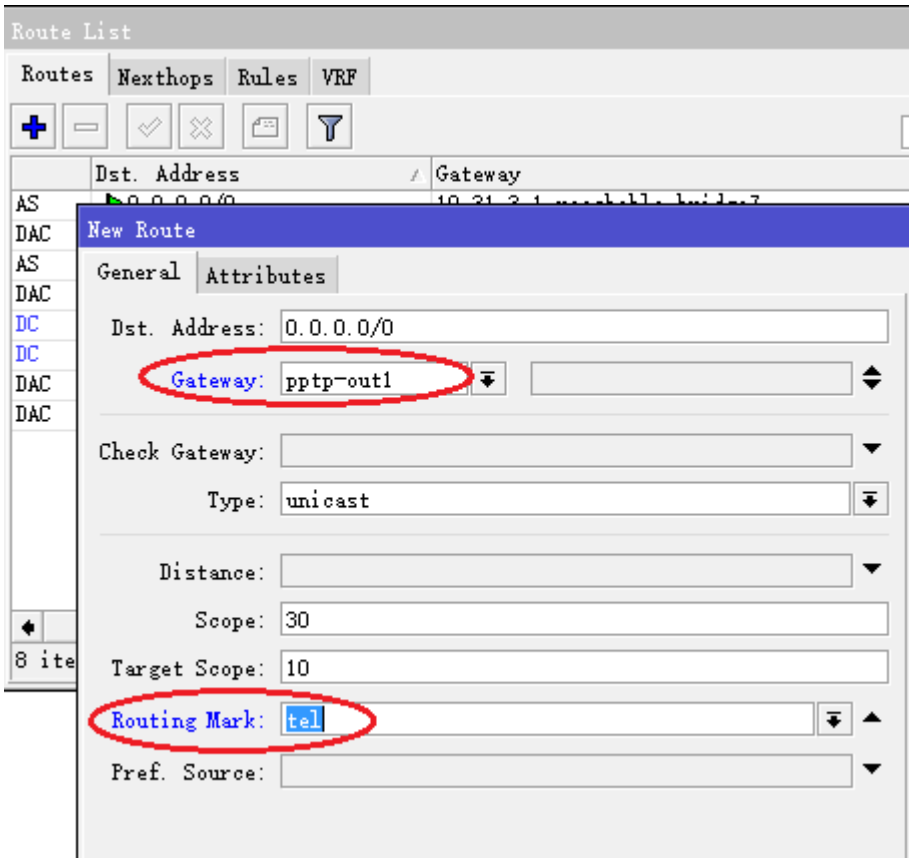


源主机策略路由

我们指定路由器 B 下的内网一台主机 IP 192.168.88.8 连接到 A 路由的运营商 Tel，我们进入 ip route rules 创建策略路由，设置 src-address=192.168.88.8,创建 table 为 tel



在 ip route 里添加并应用策略路由



6.13 RouterOS PCC 负载均衡

RouterOS 是从 v3.24 开始支持 PCC（per-connection-classifier）功能，对于 RouterOS 的负载均衡我总结了以下类型，并总结了他们的特点：

• **ECMP -** Equal Cost Multi-Path Routing(等价多路径) 对IP地址负载均衡，每隔10分钟路由列表会自动更新，仅用于路由交换网络，不宜用于nat多线路会造成当前连接中断。
• **Nth负载均衡** – Nth负载均衡是早期较稳定的负载均衡，通过对连接进行分组，定义每连接数进行负载均衡，但不稳定。Nth我们将在后面单独讲解，因为还涉及到其他一些功能。
• **混合自定义模式** – 这种解决方法基于多种模式，例如通过调整路由、配置策略路由和定义脚本达到负载均衡的目的，但配置较复杂，不易快速部署和扩展。
• **PCC负载均衡** - Per Connection Classifier 每次连接分类，基于源地址/端口、目标地址/端口或不同组合的分类映射为哈希值，并对这些哈希值作取余算法，实现负载均衡功能。通过PCC可以更好的实现路由权重比例调配。

PCC 是 RouterOS 主要的负载均衡策略：

- 基于源地址/端口、目标地址/端口或不同组合的分类映射为哈希值，并作取余算法；
- 再使用 mangle 分类数据报和路由标记；
- 通过定义新的路由表指定分类的数据走相应的网关出口。

这里我们要知道 packet 数据报是将数据封装在内，connections 用于连接运输数据报，如同数据报是火车的每节车厢，connections 则是铁轨，Connections 可以选择不同的连接方式（TCP/UDP）到达目的地。

PCC 原理

PCC 从 IP 报头中获取选定的字段，并通过哈希算法将选定的字段转换为 32 位值。然后将此值除以指定的分母，将余数与指定的余数进行比较，如果相等，则标记该数据包。PCC 支持从数据包头选定的字段包括 src-address、dst-address、src-port、dst-port 进行哈希算法操作.

在 RouterOS 中 PCC 选择数据报头字段如下：

- both-addresses
- both-ports
- dst-address-and-port
- src-address|src-port
- both-addresses-and-ports
- dst-address|dst-port
- src-address-and-port

IP 数据报的包头包含了各种字段，其中有两个字段分别是 IP 的源和目标地址，TCP 和 UDP 数据报，在 TCP 和 UDP 包头中包含有源端口和目标端口。

下面是 PCC 的函数，计算方式是将指定的 IP 包头字段选取后，计算哈希值，作为分子，而要分配的线路数量作为分母进行取余的算法

```
per-connection-classifier=
PerConnectionClassifier ::= [!]ValuesToHash:Denominator/Remainder
  Remainder ::= 0..4294967295      (integer number)
  Denominator ::= 1..4294967295      (integer number)
  ValuesToHash ::= both-addresses|both-ports|dst-address-and-port|
src-address|src-port|both-addresses-and-ports|dst-address|dst-port|src-address-and-port
```

当哈希值取得后，我们举例下面的一个实例，下面有三条常用的 PCC 规则：

```
/ip firewall mangle add chain=prerouting action=mark-connection \
new-connection-mark=1st_conn per-connection-classifier=both-addresses:3/0

/ip firewall mangle add chain=prerouting action=mark-connection \
new-connection-mark=2nd_conn per-connection-classifier= both-addresses:3/1

/ip firewall mangle add chain=prerouting action=mark-connection \
new-connection-mark=3rd_conn per-connection-classifier= both-addresses:3/2
```

依次看看三条规则

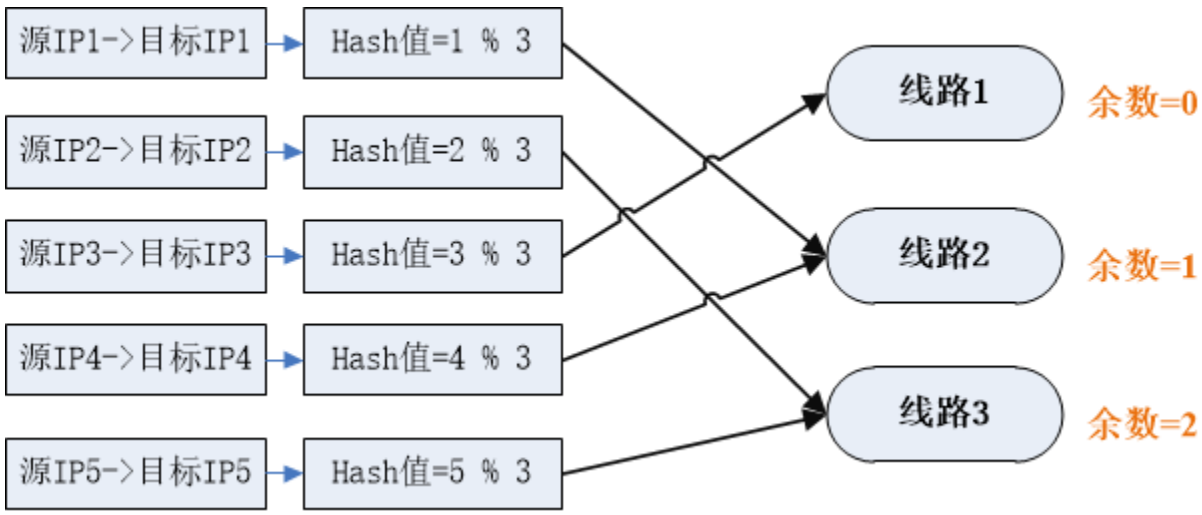
- 第一条：取源和目标 IP 地址字段（both-addresses）计算 hash 值，得到的 hash 值除以 3，余数如果为 0，action 操作标记为 1st_conn。
- 第二条：同理，hash 值除以 3 ，余数如果为 1，action 标记为 2nd_conn，
- 第三条：hash 值除以 3，余数如果为 2，action 标记为 3rd_conn。

以上是三条规则可以理解为 RouterOS 有 3 条线路，我们对这 3 条线路做 PCC 负载均衡的连接标记

上面是最经典的 IP hash 取余算法，这种算法是任给一个整数 A，将自然数 1,2,3,4,…依次除以 A，所得的余数总是循环出现，呈周期性变化，所以，我们可以取关键词被某个不大于散列表表长 m 的数 p 除后所得的余数为散列地址。即我们通常所说的两个数相除得到的余数。通常我们使用%来表示取余算法，

IP Hash 取余算法

IP_hash 是根据用户请求的 IP 地址和端口，映射成哈希值，并做取余算法，根据余数进行的策略分配。使用 IP_hash 这种负载均衡以后，可以保证用户相同请求的 IP 会话保证走同一条线路。IP_hash 算法在网络通信方面应用非常广泛 ，很多 CDN 缓存技术通过对 url 进行哈希算法，完成内容访问和存储的负载均衡。



上图是 3 条线路的负载均衡，根据源和目标 IP 做哈希取余算法，3 作为分母。如果 4 线就用 4 作为分母，以此类推。可以看到，同一组源和目标 IP 地址请求，哈希值是一样的，取余数的结果也会是一样，因此相同的源和目标 IP 会话始终会走相同线路。

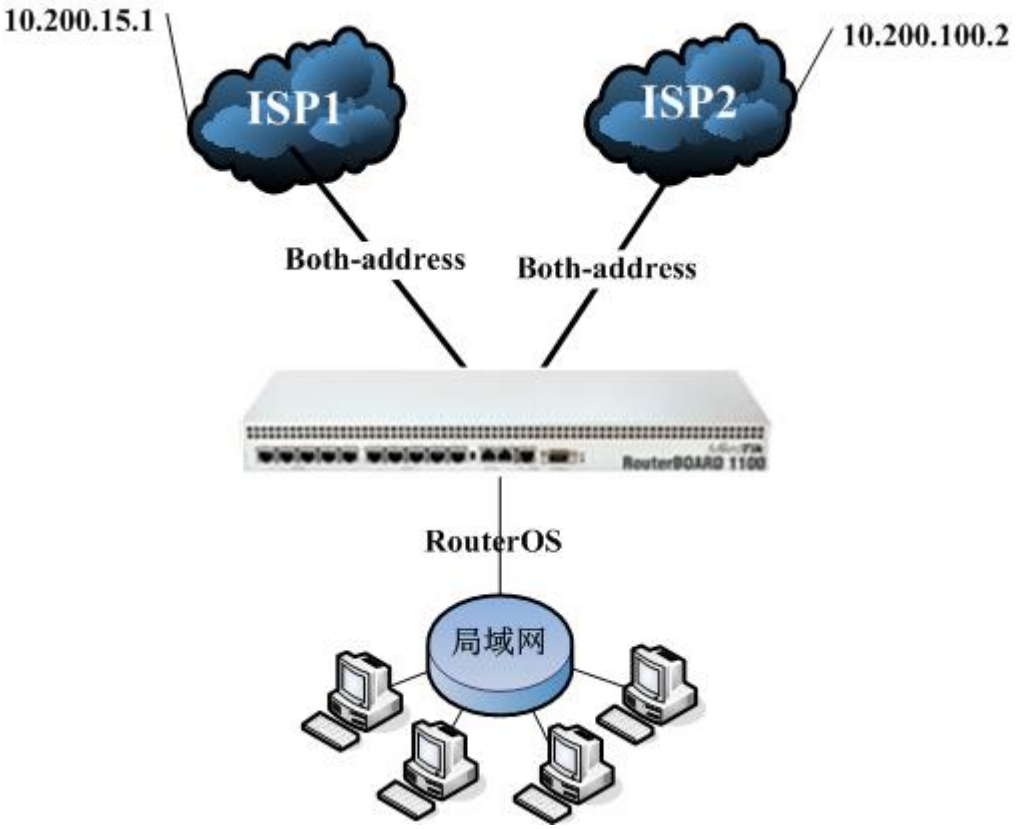
我们这里举一个例子，假如我们选择了 both-addresses 字段，即目标 IP 地址和源 IP 地址，在我们实际案例中如 192.168.10.8 访问了 114.114.114.114 和 8.8.8.8，这个两个不同的访问连接，但 192.168.10.8 访问 114.114.114.114 的 hash 值始终不变，每次取余数是相同的，即每次都会被分配到相同的线路上，同理 192.168.10.8 访问 8.8.8.8 始终有相同的 hash 值，也是会分配到相同连线路上，这样 PCC 将相同会话的连接分配的同一条线路保持不变。

PCC 虽然能用于负载均衡，但 PCC 自身是无法直接用于路由，或 routing marks，PCC 仅仅只能用于数据报匹配，不能直接用于数据报标记。

注：PCC 从 RouterOS v3.24 开始支持，这个功能解决了多网关的负载均衡问题。

PCC 的负载均衡事例

下面的事例，是一个双线的 PCC 负载均衡策略事例，这里有两条 WAN 出口分别是 wan1 和 wan2，网络结构如下：

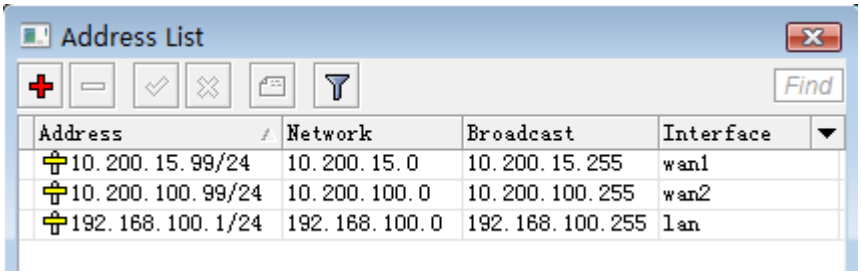


警告：PCC 负载均衡对于国内，仅适用于相同运营商的出口，不同的运营商之间 DNS 解析结果不同，而且到同一目标 IP 路径不同会导致网络故障！

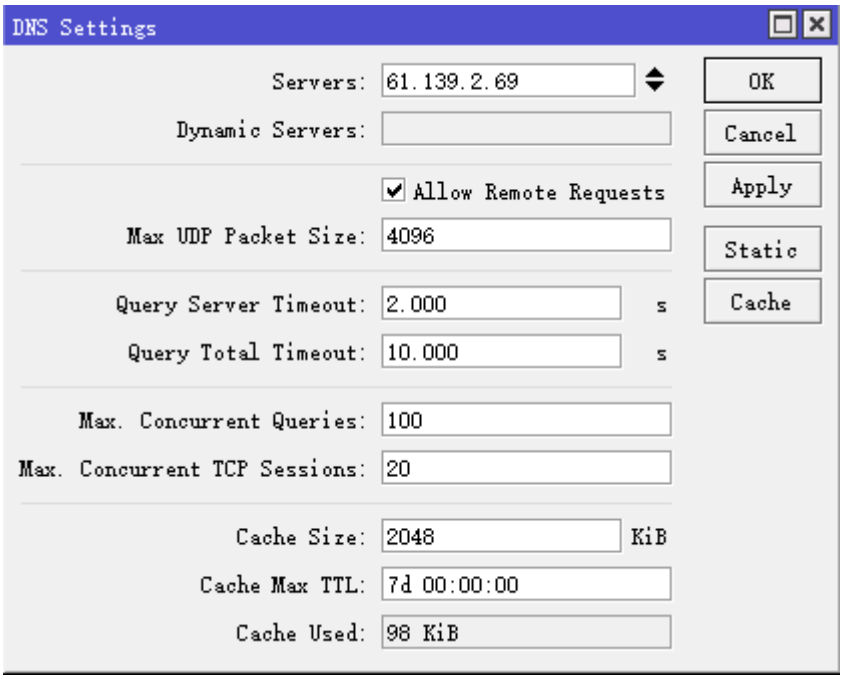
- ISP1 地址 10.200.15.99/24，网关：10.200.15.1；
- ISP2 地址 10.200.100.99/24，网关：10.200.100.2；
- 内网 IP 地址 192.168.100.1/24；
- 启用 DNS 缓存功能，用 192.168.100.1 作内网 DNS 解析；

基本配置

首先进入 ip address 添加相应接口的 IP 地址：



在 ip dns 中配置 DNS 为：61.139.2.69，并选择 allow-remote-requests，启用 dns 本地解析。

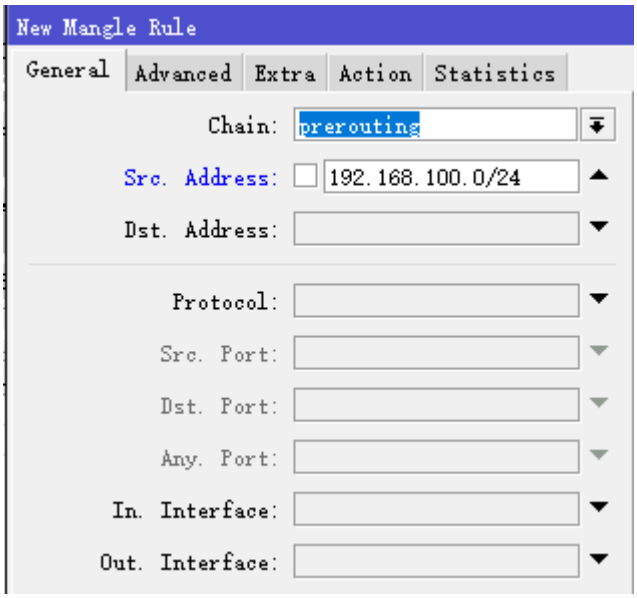


Mangle 标记配置

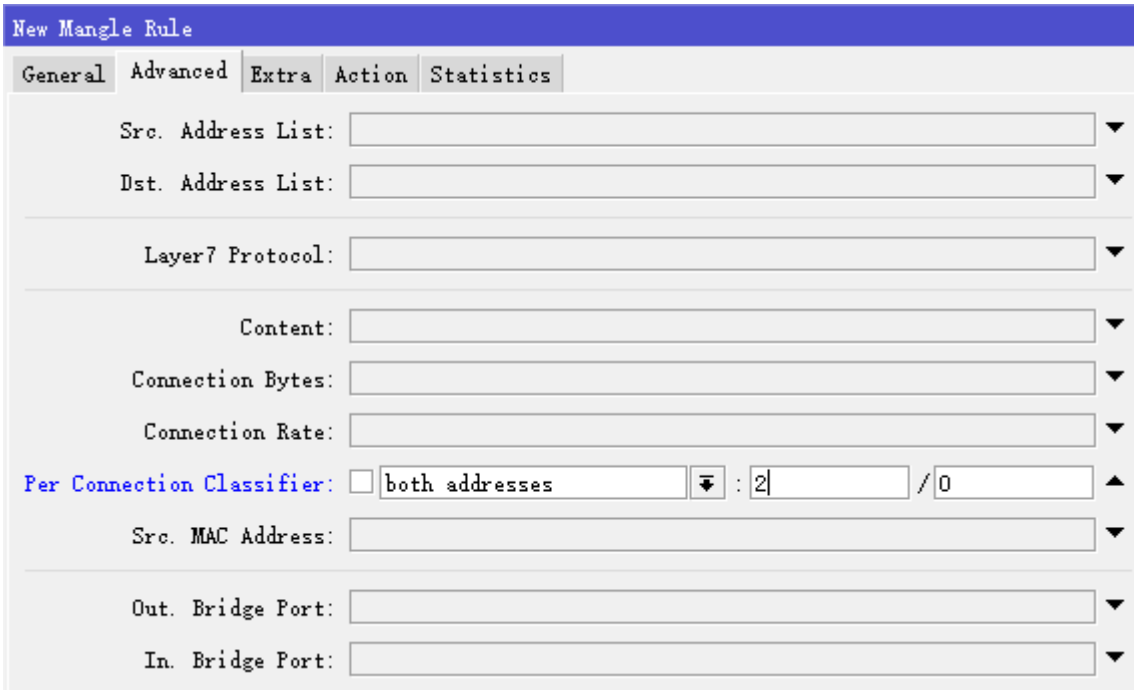
接下来我们进入 ip firewall mangle 标记连接和路由，使用 per-connection-classifier 双向地址进行分类做连接分类标记。一共需要创建 4 条 mangle 规则，2 条 PCC 连接标记和 2 条关联的路由标记。

第一组 PCC 标记

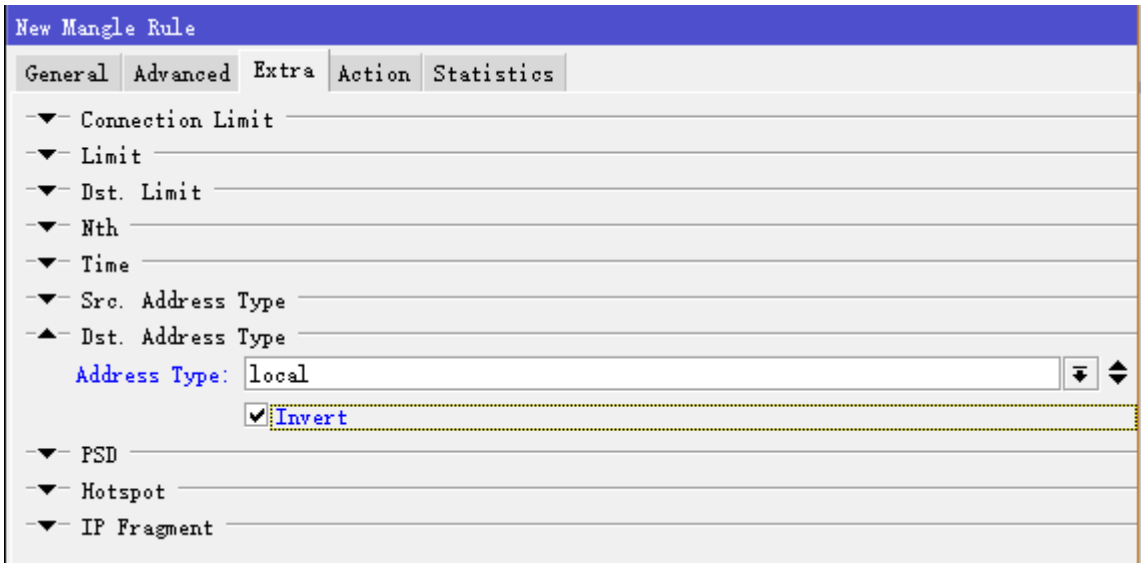
首先，在 prerouting 链表，标记第一条 PCC 连接标记策略，设置 src-address=192.168.100.0/24



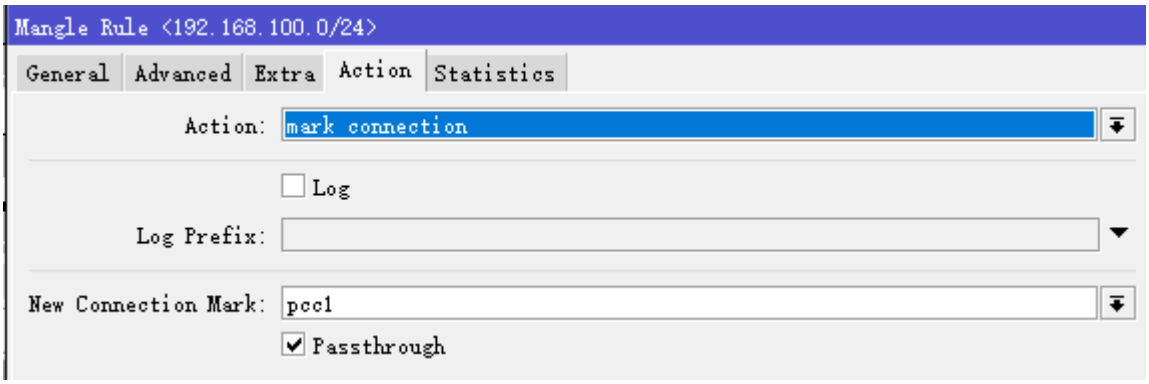
然后进入 advanced 菜单下，可以看到 per-connection-classifier 分类器，选择 both-addresses，设置参数为 2/0(both-addresses，代表取目标 IP 和源 IP 地址的哈希值，2/0 意思是，2 代表两条线路，那么引入一个除法算式"哈希值/2"余数为 0 的标记一条线路，余数为 1 标记第二条)，线路数增加就改变除数值，依次类推：



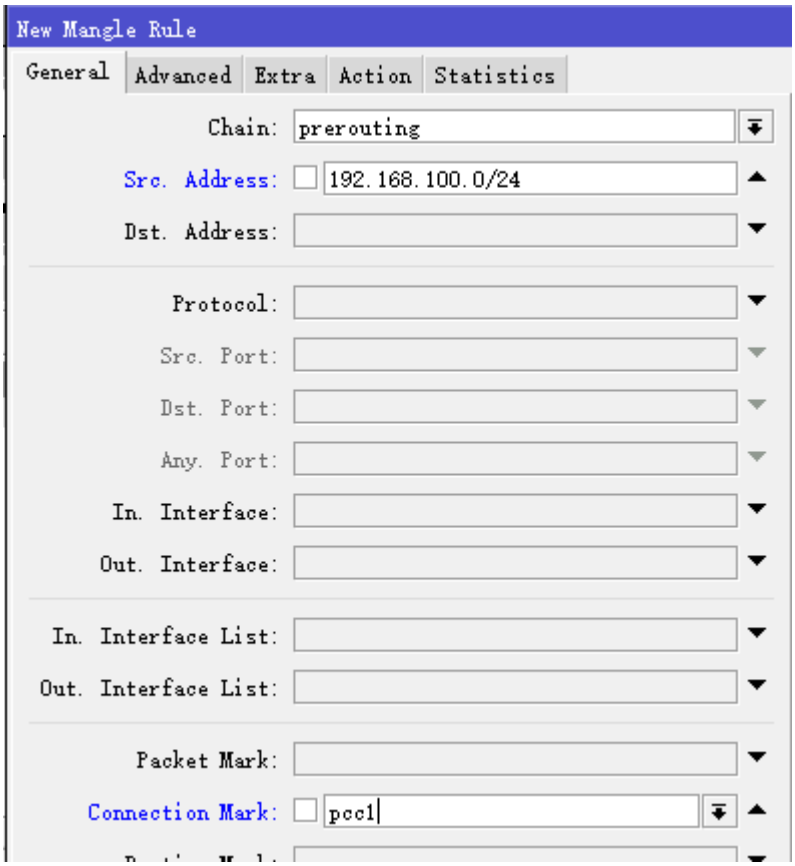
接着，进入 extra 菜单下，选择 dst-address-type=!local，即排除目标是本地地址：



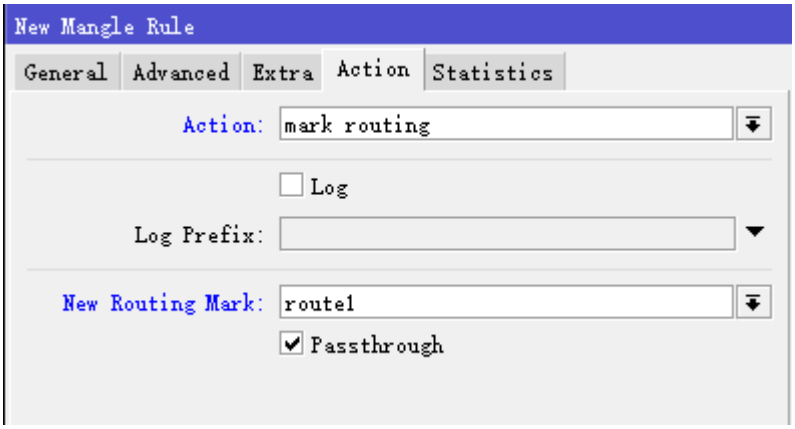
最后，进入 action 菜单设置 action=mark-connection，并取名 new-connection-mark=pcc1



紧跟着，添加第一条关联的路由标记，仍然是在 perrouting 链表添加规则，设置 src-address=192.168.100.0/24，connection-mark=pcc1



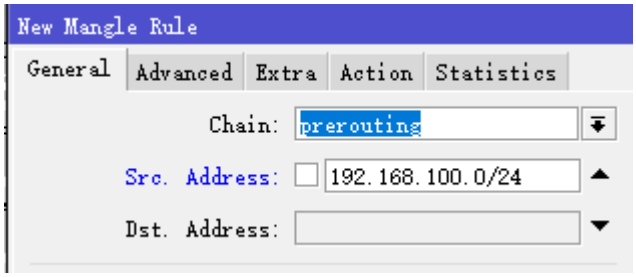
设置该规则的 action=mark-routing，new-routing-mark=route1

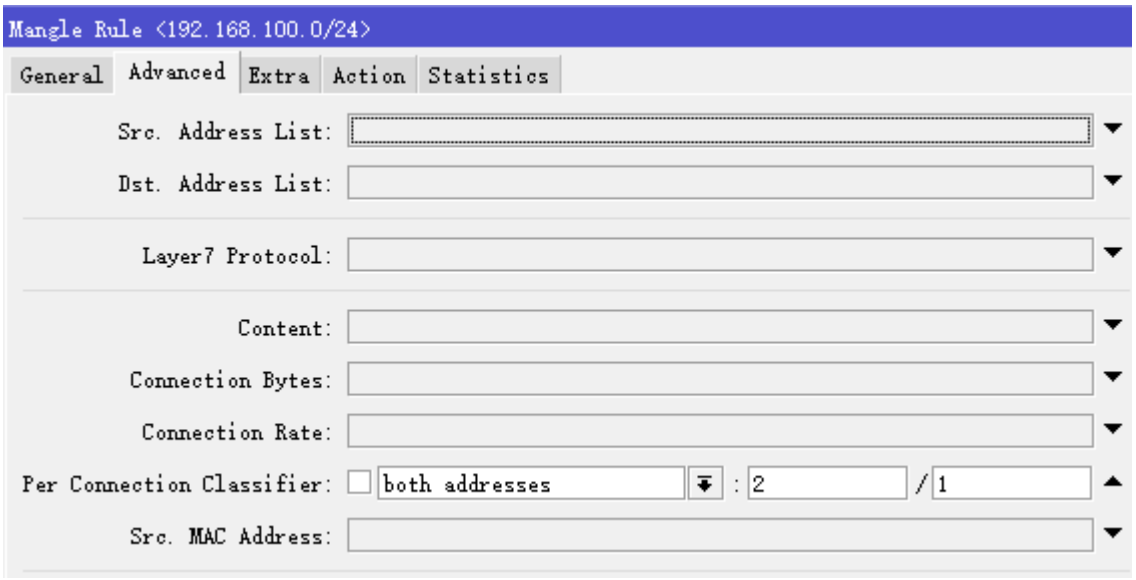


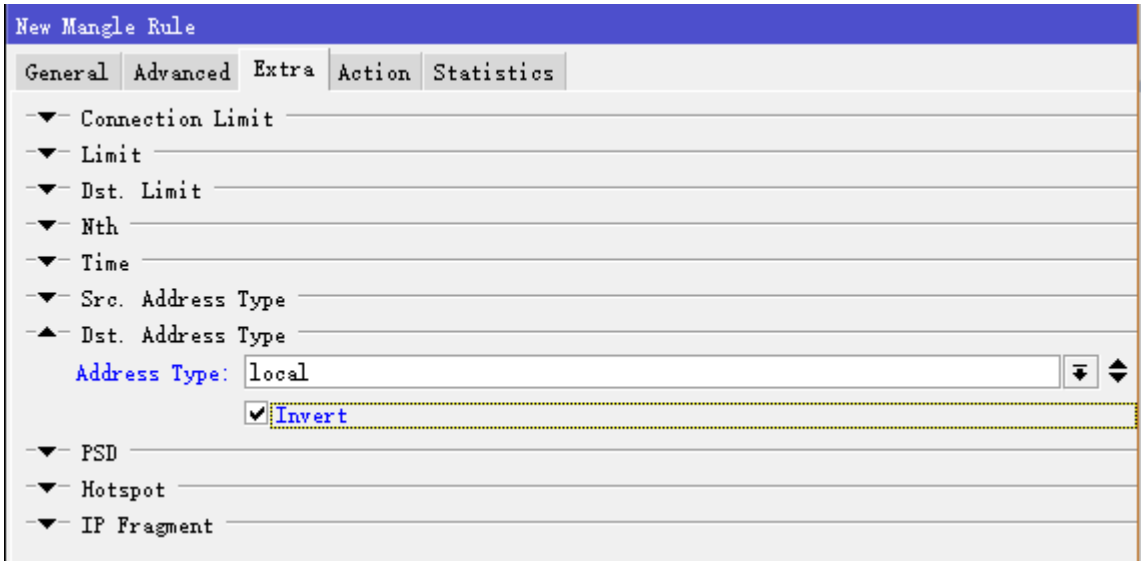
第一组线路的 PCC 规则设置完成

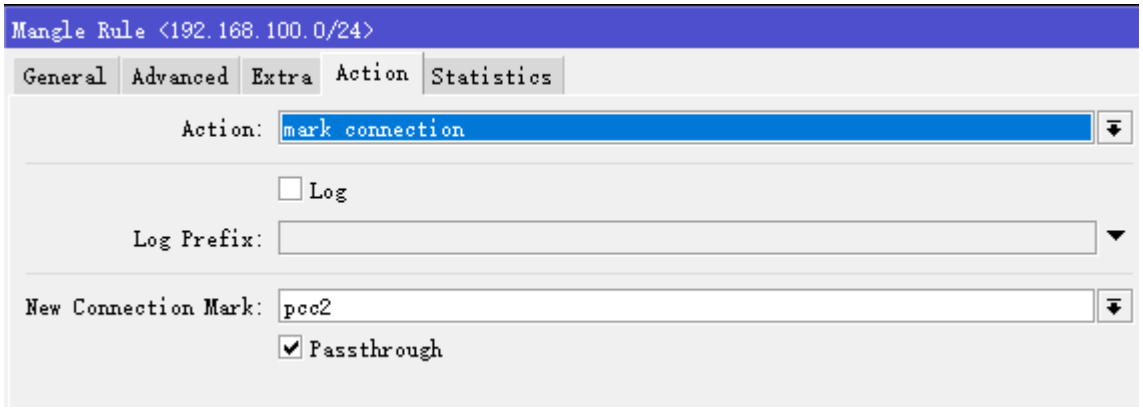
第二组 PCC 标记

双线的 per-connection-classifier 分类是第一条线为 2/0，第二条为 2/1，第二条规则也是类似方式添加，只是创建 PCC 策略时，设置 2/1，继续创建第二组的 pcc 连接标记，修改 new-connection-mark 为 pcc2，以及 new-routing-mark 为 route2

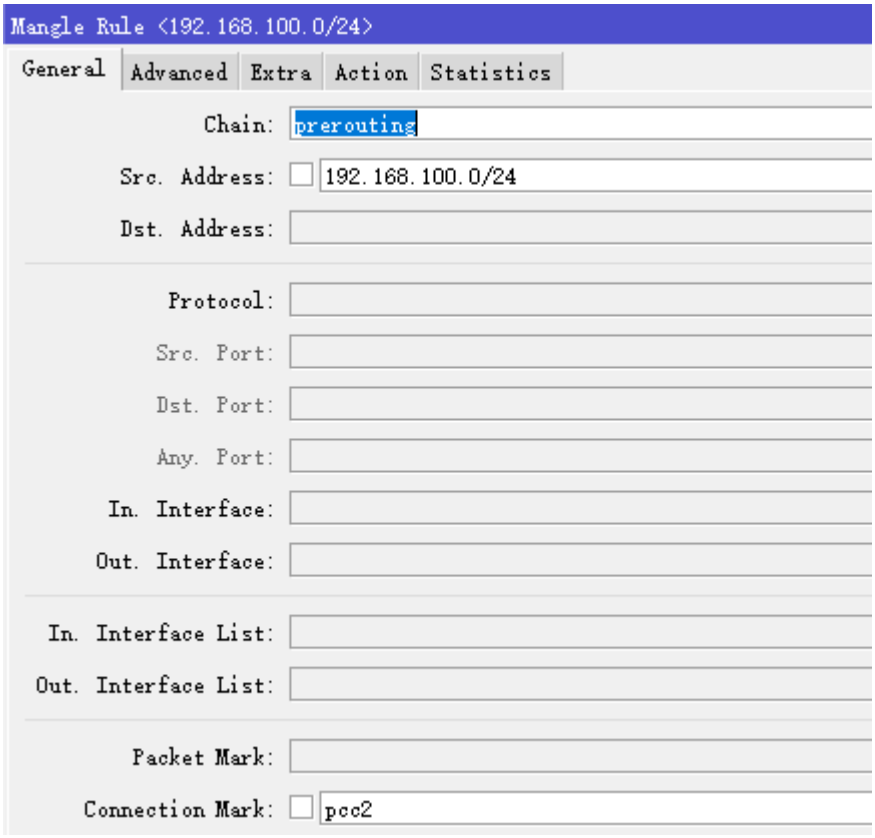

New Mangle Rule
General
Advanced
Extra
Action
Statistics
Chain:
prerouting
Src. Address:
192.168.100.0/24
Dst. Address:

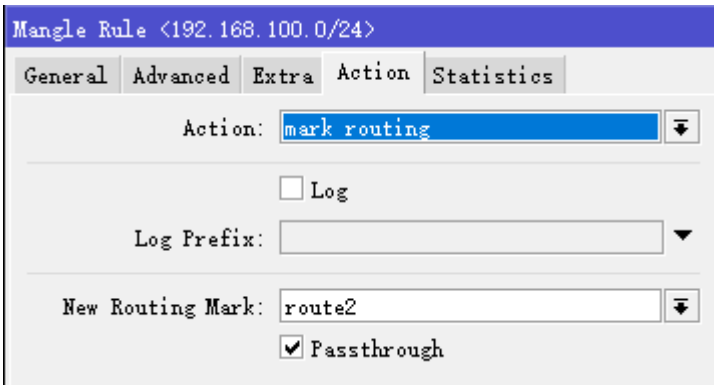

Mangle Rule <192.168.100.0/24>
General
Advanced
Extra
Action
Statistics
Src. Address List:
Dst. Address List:
Layer7 Protocol:
Content:
Connection Bytes:
Connection Rate:
Per Connection Classifier:
both addresses
2
1
Src. MAC Address:


New Mangle Rule
General
Advanced
Extra
Action
Statistics
Connection Limit
Limit
Dst. Limit
Nth
Time
Src. Address Type
Dst. Address Type
Address Type:
local
Invert
PSD
Hotspot
IP Fragment

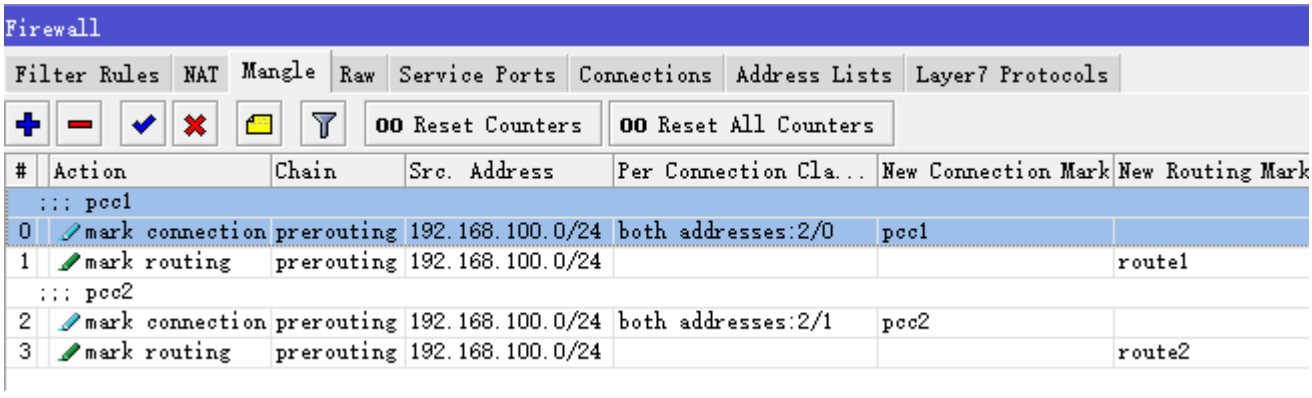

Mangle Rule <192.168.100.0/24>
General
Advanced
Extra
Action
Statistics
Action:
mark connection
Log
Log Prefix:
New Connection Mark:
pcc2
Passthrough

创建关联的路由标记规则





最后添加完成如下：



下面命令行配置，第一条线路的连接标记取名为 **pcc1**，并从连接里提取路由标记名位 **route1**，设置：per-connection-classifier=both-addresses:2/0。

```
/ip firewall mangle
add action=mark-connection chain=prerouting comment="" disabled=no \
    src-address=192.168.100.0/24 new-connection-mark=pcc1 passthrough=yes \
    per-connection-classifier=both-addresses:2/0
add action=mark-routing chain=prerouting comment="" connection-mark=pcc1 \
    disabled=no src-address=192.168.100.0/24 new-routing-mark=route1
passthrough=yes
```

第二条线路的连接标记取名为 **pcc2**，并从连接里提取路由标记名位 **route2**，设置：per-connection-classifier=both-addresses:2/1。

```
/ip firewall mangle
add action=mark-connection chain=prerouting comment="" disabled=no \
    src-address=192.168.100.0/24 new-connection-mark=pcc2 passthrough=yes \
    per-connection-classifier=both-addresses:2/1
add action=mark-routing chain=prerouting comment="" connection-mark=pcc2 \
    disabled=no src-address=192.168.100.0/24 new-routing-mark=route2
passthrough=yes
```

返程路由设置

定义 IP 数据报从那个接口进入，就按原路从那个接口回去，即保证每个外网口的数据能得到正确的路由。

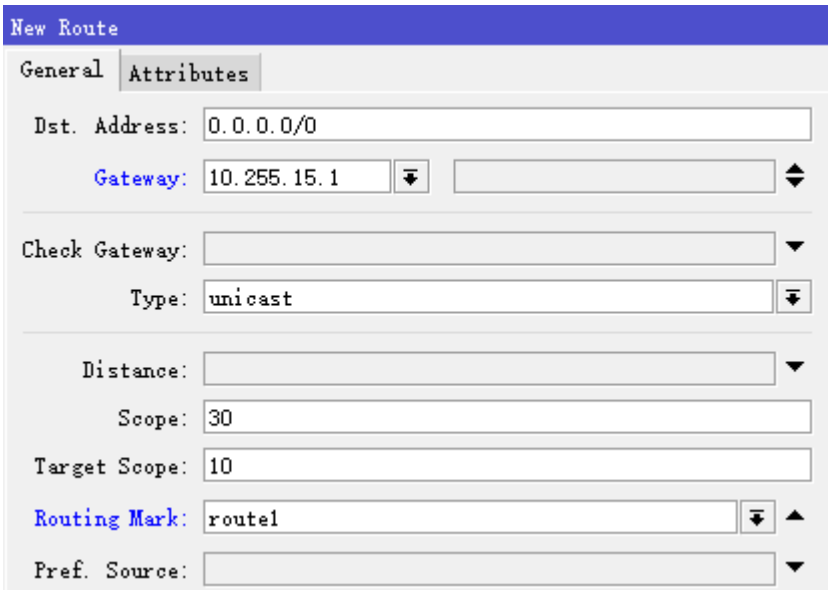
```
/ ip firewall mangle
add chain=input in-interface=wan1 action=mark-connection new-connection-mark=pcc1
add chain=input in-interface=wan2 action=mark-connection new-connection-mark=pcc2
```

标记完进入接口的连接后，将这些连接接指定到相应的路由标记上：

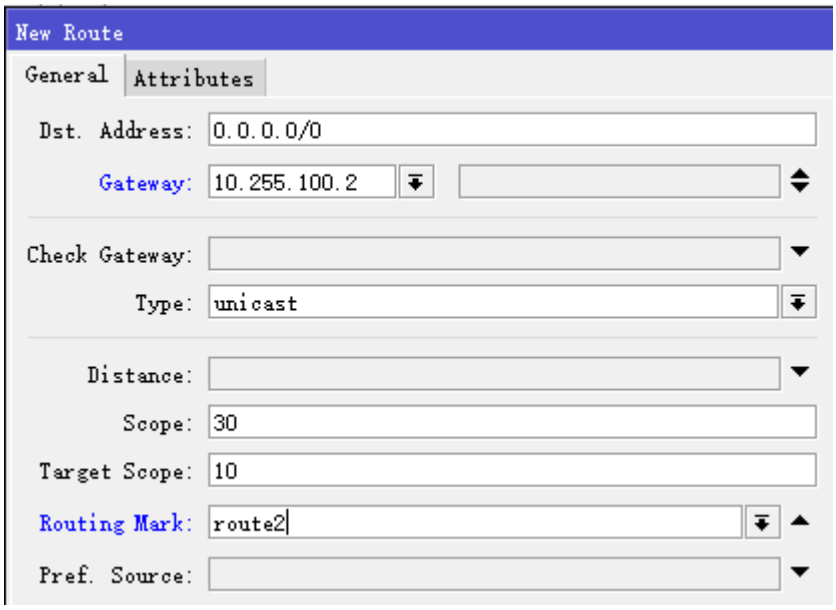
```
add chain=output connection-mark=pcc1 action=mark-routing new-routing-mark=route1
add chain=output connection-mark=pcc2 action=mark-routing new-routing-mark=route2
```

路由配置

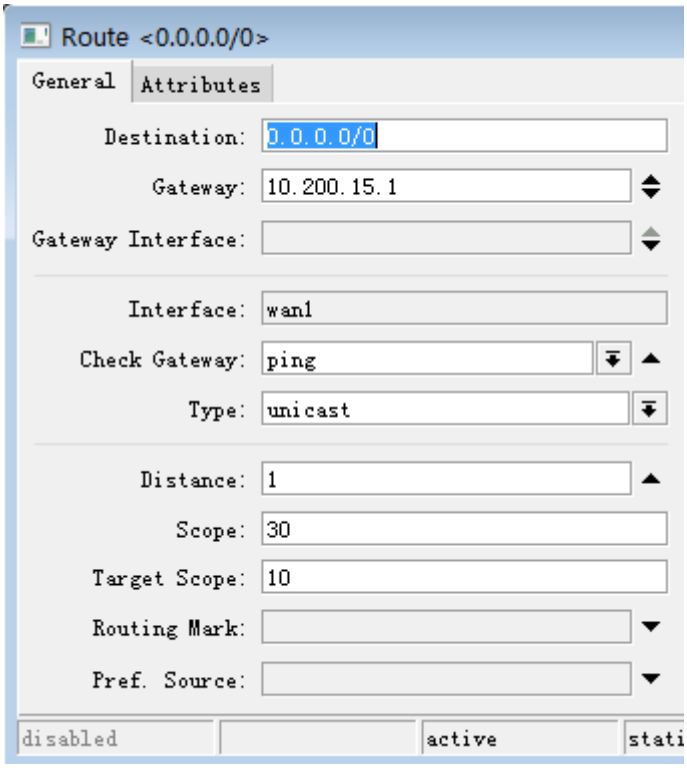
配置完标记路由后，我们进入 ip route 配置路由，首先设置负载均衡的标记路由，首先设置第一条线路的路由标记，设置 routing-mark=route1：



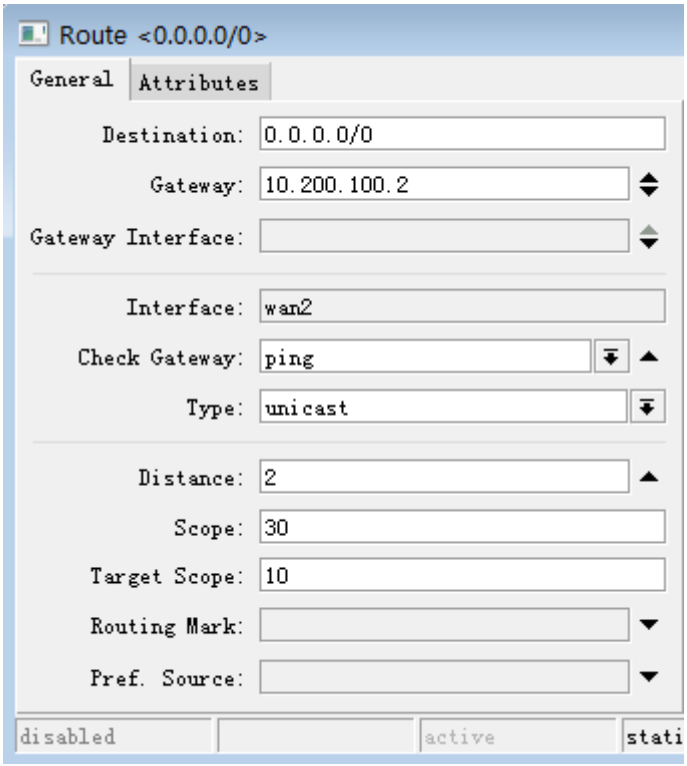
设置第二条线路的路由标记，设置 routing-mark=route2：



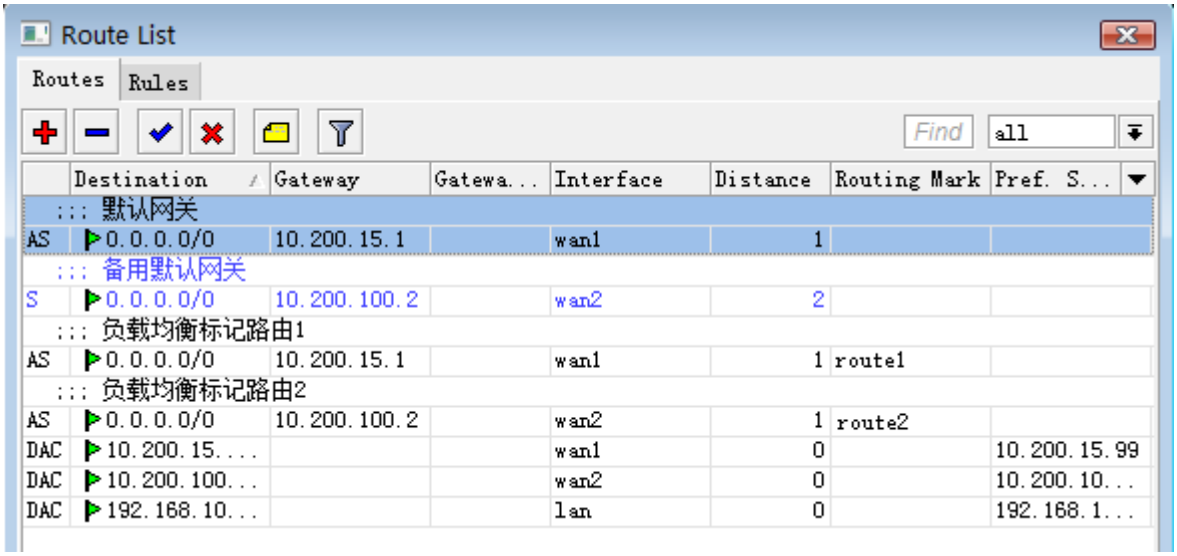
配置默认网关和备份网关，默认网关的 **distance** 设置为 1，并设置 check-gateway=ping，通过 ping 监测网关状态：



备份网关的 **distance** 设置为 2，并设置 check-gateway=ping，通过 ping 监测网关状态：



配置完成后的路由标如下图：

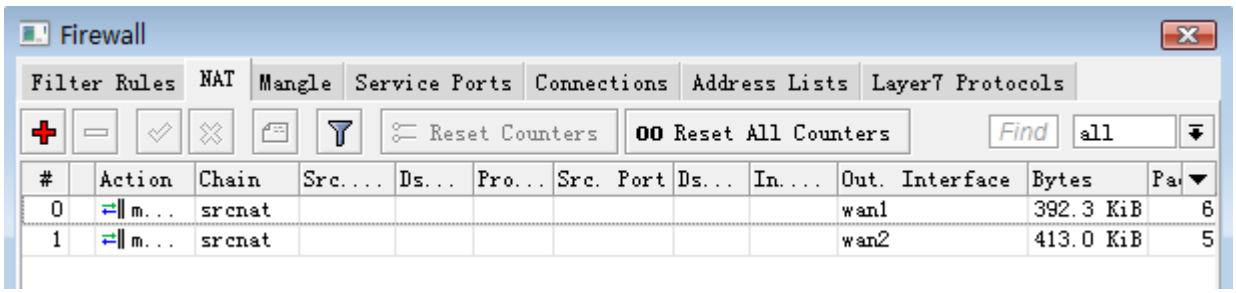


注意：如果是 PPPoE 拨号，gateway 可以直接选择 pppoe-out 的接口作为网关。

配置 nat

最后配置 nat 转换规则，进入 ip firewall nat 中配置 action=masquerade，分别对 2 条线路做伪装：

```
/ip firewall nat
add action=masquerade chain=srcnat out-interface=wan1
add action=masquerade chain=srcnat out-interface=wan2
```



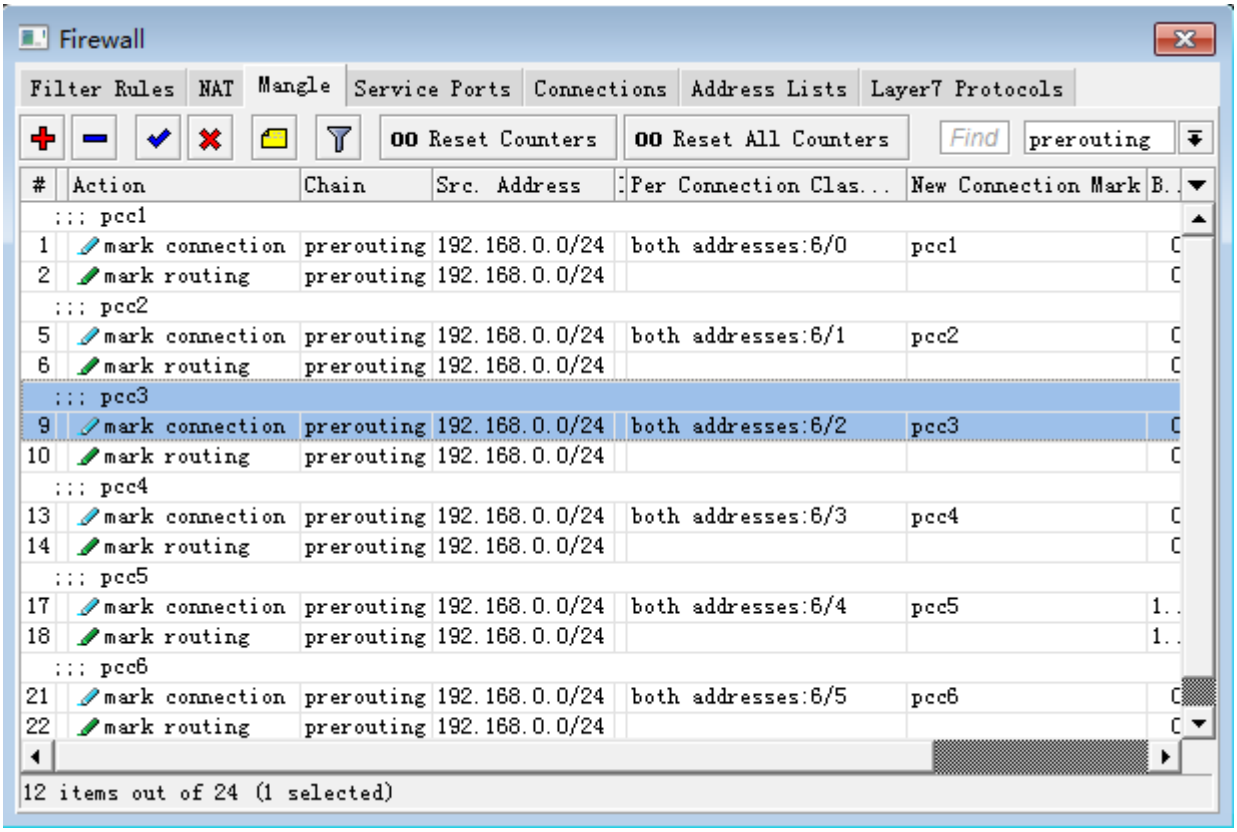
以上 PCC 配置完成

注意，如果在 RouterOS 的内网有多个 IP 地址段（如 192.168.10.0/24，192.168.20.0/24 和 1921.68.30.0/24 等等），在做 PCC 后，会导致他们之间无法互通，解决方法请参考策略路由导致内网段互访失效

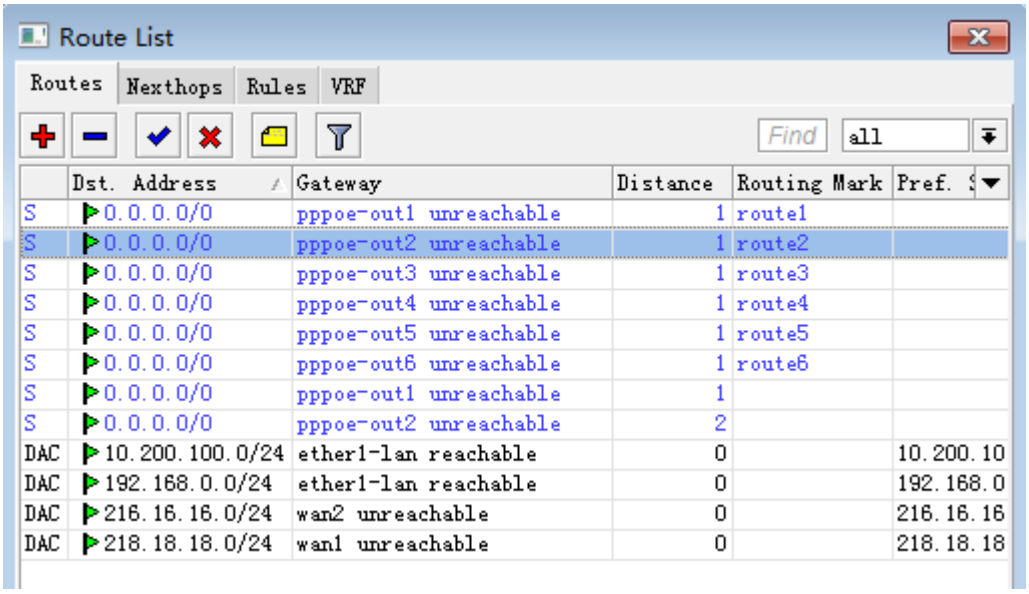
六线的 PCC 配置

在许多网络中会出现相同运营商，相同带宽多线路的接入，比如下面是一个 6 条线路的接入，我们通过 PCC 解决带宽的均衡，通过分类源地址和目标地址的负载均衡，即双向地址（both addresses），设置于之前的操作基本相同，只是变成了 6 组规则

下面是 6 条 ADSL 的情况，mangle 标记的 prerouting 规则：



进入 ip route 设置路由，这个是 PPPoE 拨号上网的策略路由配置：



PCC 多线比例权重路由

假设网络有这样一个需求，同时拥有两条相同运营商的出口，一条 8M，一条是 25M，为最大化有效利用线路，将两条线路按照流量比例实现权重的路由策略，我们可以通过 PCC 来实现。

警告：PCC 负载均衡对于国内，仅适用于相同运营商的出口，不同的运营商之间 DNS 解析结果不同，而且到同一目标 IP 路径不同会导致网络故障！

平常我们都是用 PCC 做多条相同带宽出口的负载均衡，而这次则通过实现比例分配到每条线路上，虽然 Nth 也可以实现，但 Nth 缺乏对每条会话的 hash 算法处理，不能做到相同会话走相同线路，出现不稳定情况。

实现原理比较简单，一条 8M，一条是 25M，后者大约是前者的 3 倍出口，所以约等于 1:3，那就是要按照 1:3 的比例分配路由，策略将 PCC 策略看成 4 份，然后指定按照 1:3 的路由策略规则分配。

先定义用户 IP 地址，通过 src-address-list 定义

```
/ip firewall address-list
add address=192.168.88.0/24 list=userip
add address=192.168.80.0/24 list=userip
```

配置 PCC 规则，即把 2 条出口，看成 4 份数据进行 PCC 的策略配置，即我们在 mangle 中配置 4 组 PCC 的标记规则，和配置 4 条负载均衡的规则一样，下面是命令行配置：

```
/ip firewall mangle
add action=mark-connection chain=prerouting dst-address-type=!local
new-connection-mark=pcc1 passthrough=yes per-connection-classifier=both-addresses:4/0
src-address-list=userip

add action=mark-routing chain=prerouting connection-mark=pcc1 new-routing-mark=r1
passthrough=yes src-address-list=userip

add action=mark-connection chain=prerouting dst-address-type=!local
new-connection-mark=pcc2 passthrough=yes per-connection-classifier=both-addresses:4/1
src-address-list=userip

add action=mark-routing chain=prerouting connection-mark=pcc2 new-routing-mark=r2
passthrough=yes src-address-list=userip

add action=mark-connection chain=prerouting dst-address-type=!local
```

```
new-connection-mark=pcc3 passthrough=yes per-connection-classifier=both-addresses:4/2
src-address-list=userip

add action=mark-routing chain=prerouting connection-mark=pcc3 new-routing-mark=r3
passthrough=yes src-address-list=userip

add action=mark-connection chain=prerouting dst-address-type=!local
new-connection-mark=pcc4 passthrough=yes per-connection-classifier=both-addresses:4/3
src-address-list=userip

add action=mark-routing chain=prerouting connection-mark=pcc4 new-routing-mark=r4
passthrough=yes src-address-list=userip
```

路由配置

其实比例分配关键就在路由设置上，这里我们把网关命名为 8M 和 25M 以示区分。将分配好的路由标记按照 1:3 的比例分配到各条线路上

```
/ip route
add check-gateway=ping gateway=8M routing-mark=r1
add check-gateway=ping gateway=25M routing-mark=r2
add check-gateway=ping gateway=25M routing-mark=r3
add check-gateway=ping gateway=25M routing-mark=r4
```

nat 配置

配置 nat 规则类似的操作

```
/ip firewall nat
add chain=srcnat action=masquerade out-interface=wan-8M
add chain=srcnat action=masquerade out-interface=wan-25M
```

配置完成后流量几乎按照预想的方式运行 ，这样的操作建议使用到相同类型的出口，比如不同带宽的线路都是 Tel，或者都是 Un。不建议在不同运营商出口上采用这样的规则，避免延迟和 dns 解析等问题。

PCC 在 IDC 应用

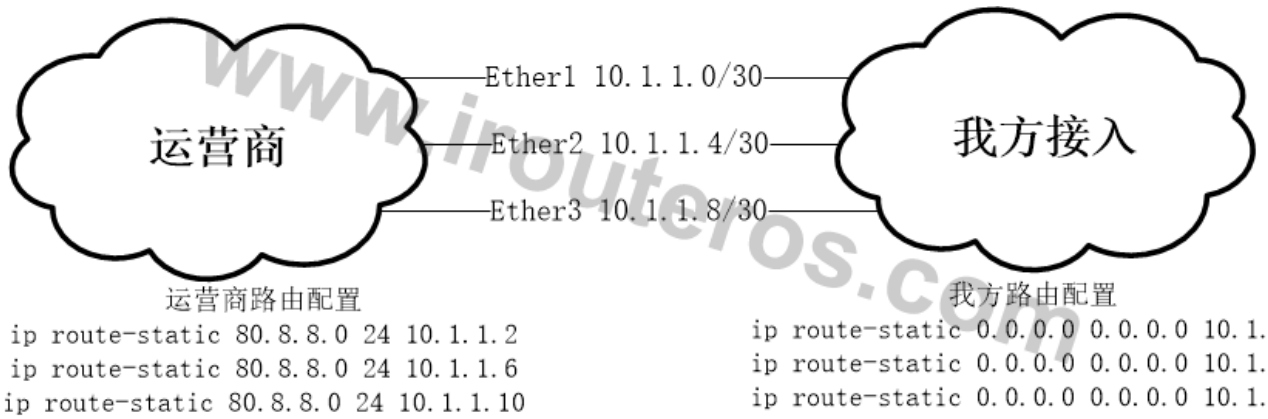
这里探讨的不是 RouterOS 在 IDC 机房的 VPS 或虚拟化的应用，而是作为一个 IDC 客户与运营商接入三层路由的应用，或是与集群服务器多条回指路由的应用。当然这种环境并非只出现在 IDC 机房网络，这个应用是本人在 IDC 工作中总结的，仅供参考。

IDC 接入的应用

通过一个实例，比如我在运营商的 IDC 机房开通了 30G 的 IDC 带宽，那运营商给我开通肯定是 3 个 10GE 的接口，分配了 256 个公网 IPv4 地址（暂不讨论 IPv6 网络），IPv4 地址接入采用路由模式，而非二层接入。

运营商 30G 接入配置，可以是分配 3 组/30 子网互联 IPv4 地址，也可以是配置 3 个端口的链路聚合口，例如华为的 Eth-trunk 接口，分配一组/30 子网互联 IPv4 地址。

这里我们排除链路聚合口的配置，选择 3 组/30 子网互联地址，将一个 C 段的 IPv4 地址，假如 IPv4 地址是：80.8.8.0/24，运营商会配置 3 条静态路由，指向 3 个互联接口的 IPv4 地址，我方也会配置 3 条默认路由到 3 个互联接口的 IPv4 网关，这个是最常见的 ECMP 协议，保证 3 个接入端口流量均衡，IDC 网络中应用非常普片。互联配置如下：



www.irouteros.com

通过上面的互联图，有网络基础的朋友，大概就明白了 IDC 网络路由模式接入配置

基本的接入说完了，现在要说下 PCC 如何应用其中，这个应用非 nat 网络，仅是三层路由，因此关闭连接跟踪：

```
[admin@MikroTik] > ip/firewall/connection/tracking/set enabled=no
```

3 条 mangle 直接标记路由包，因为不涉及 nat 的连接跟踪，所以没有 mark-connection 的标记规则：

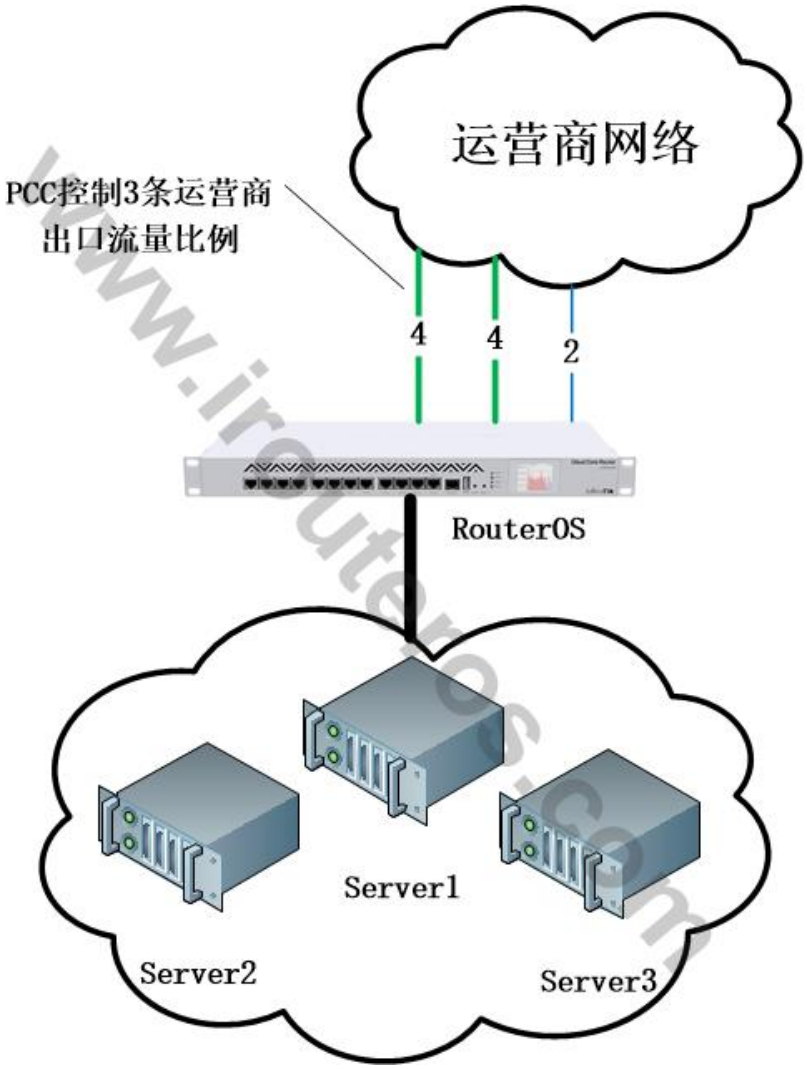
```
/ip firewall mangle
add action=mark-routing chain=prerouting src-address=80.8.8.0/24 \
new-routing-mark=route1 passthrough=yes per-connection-classifier= both-addresses:3/0

add action=mark-routing chain=prerouting src-address=80.8.8.0/24 \
new-routing-mark=route2 passthrough=yes per-connection-classifier= both-addresses:3/1

add action=mark-routing chain=prerouting src-address=80.8.8.0/24 \
new-routing-mark=route3 passthrough=yes per-connection-classifier= both-addresses:3/2
```

路由配置：

```
/ip route
add gateway=10.1.1.1 routing-mark=route1
add gateway=10.1.1.5 routing-mark=route2
add gateway=10.1.1.9 routing-mark=route3
```

上面的配置其实和 ECMP 没什么区别，就是用 PCC 复现了 ECMP 的配置，也就是 PCC 做的 3 条负载均衡

现在我们要实现 4：4：2，也就是 2：2：1，需要标记 5 条路由规则



标记 5 条规则配置：

```
/ip firewall mangle
add action=mark-routing chain=prerouting src-address=80.8.8.0/24 \
new-routing-mark=route1 passthrough=yes per-connection-classifier= both-addresses:5/0

add action=mark-routing chain=prerouting src-address=80.8.8.0/24 \
new-routing-mark=route2 passthrough=yes per-connection-classifier= both-addresses:5/1

add action=mark-routing chain=prerouting src-address=80.8.8.0/24 \
new-routing-mark=route3 passthrough=yes per-connection-classifier= both-addresses:5/2

add action=mark-routing chain=prerouting src-address=80.8.8.0/24 \
new-routing-mark=route4 passthrough=yes per-connection-classifier= both-addresses:5/3

add action=mark-routing chain=prerouting src-address=80.8.8.0/24 \
new-routing-mark=route5 passthrough=yes per-connection-classifier= both-addresses:5/4
```

路由配置：

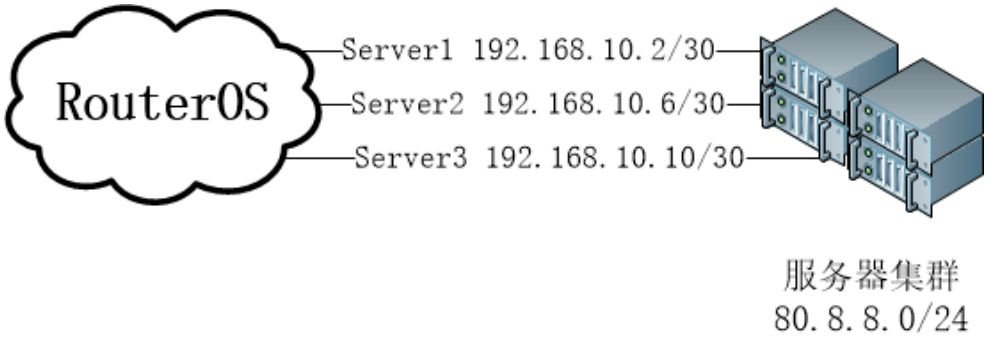
```
/ip route
```

```
add gateway=10.1.1.1 routing-mark=route1
add gateway=10.1.1.1 routing-mark=route2
add gateway=10.1.1.5 routing-mark=route3
add gateway=10.1.1.5 routing-mark=route4
add gateway=10.1.1.9 routing-mark=route5
```

这样 3 条线路比例调配，这个比例还可以修改 PCC 规则调整，同时 RouterOS 的 script 可以参与实时的流量调整比例分配，至于这种配置有什么用，搞 IDC 的朋友他们自然明白！

服务器集群应用

考虑另外一个场景，方向是去一个集群服务器，也就是和之前的案例反过来，RouterOS 需要面向客户，让集群服务器需要确保流量调度均衡，也需要按比例分配一定流量，同样是 4：4：2，需要标记 5 条路由规则，下图是服务器集群是三台和 RouterOS 都有一个/30 子网互联：



标记 5 条规则，对于 RouterOS 来说，是去 80.8.8.0/24，修为 dst-address

```
/ip firewall mangle
add action=mark-routing chain=prerouting dst-address=80.8.8.0/24 \
new-routing-mark=route1 passthrough=yes per-connection-classifier= both-addresses:5/0

add action=mark-routing chain=prerouting dst-address=80.8.8.0/24 \
new-routing-mark=route2 passthrough=yes per-connection-classifier= both-addresses:5/1

add action=mark-routing chain=prerouting dst -address=80.8.8.0/24 \
new-routing-mark=route3 passthrough=yes per-connection-classifier= both-addresses:5/2

add action=mark-routing chain=prerouting dst -address=80.8.8.0/24 \
new-routing-mark=route4 passthrough=yes per-connection-classifier= both-addresses:5/3

add action=mark-routing chain=prerouting dst -address=80.8.8.0/24 \
new-routing-mark=route5 passthrough=yes per-connection-classifier= both-addresses:5/4
```

在写静态路由同时加上 routing-mark 标记

```
/ip route
add dst-address=80.8.8.0/24 gateway=192.168.10.2 routing-mark=route1
add dst-address=80.8.8.0/24 gateway=192.168.10.2 routing-mark=route2
add dst-address=80.8.8.0/24 gateway=192.168.10.6 routing-mark=route3
add dst-address=80.8.8.0/24 gateway=192.168.10.6 routing-mark=route4
```

```
add dst-address=80.8.8.0/24 gateway=192.168.10.10 routing-mark=route5
```

到目标地址的路由，PCC 的 hash 算法能确保相同的源 IP 命中到同一台服务器，而不会随机连接到其中一台。

ECMP 会每 10 秒调整一次算法，导致路由线路变动，对于三层路由而言并不影响使用，因为多条路径下，都可以到达目标网络，DNS 和普通 web 服务采用相同 loopback 的 IP 地址用 ECMP 没问题，对于一些特殊的内容连接服务，有时候要求源 IP 始终落到同一台服务器的 loopback IP 上能保持一直的路径连接，PCC 为一些特殊应用提供可用性。之前的一篇文章大家可以参考下：探讨下 RouterOS PCC 的工作原理与应用延伸

注意：RouterOS 的当前的 CRS3 系列交换机，ECMP 是在 v7 版本后可以下发给交换机芯片处理的，但 PCC 无法下发交换芯片，就需要 CPU 来抗了（可以选择 CCR1072，或者多核心，高频率的 x86 服务器），能跑多少流量我不清楚，只是闲聊下。

6.14 单网卡多次拨号 PPPoE 配置

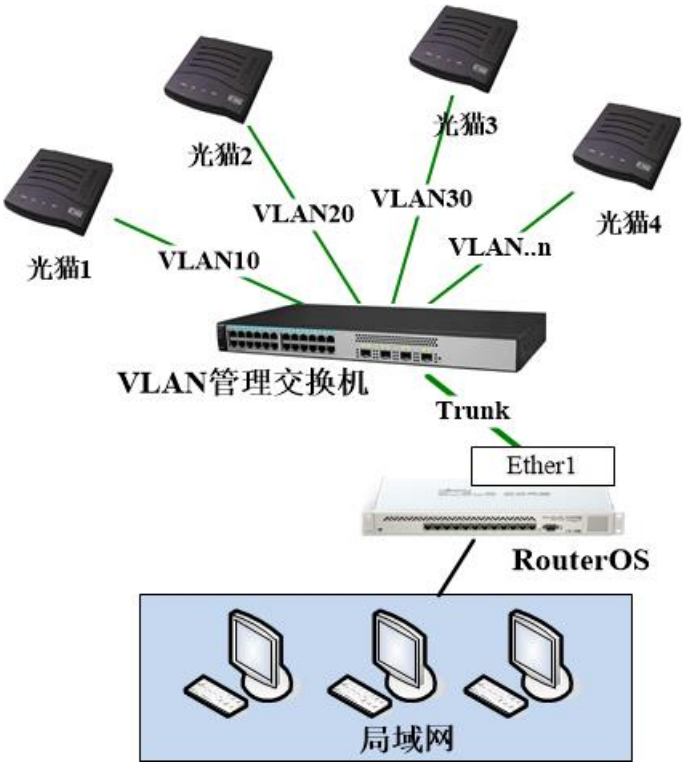
现在 RouterOS 在使用 PCC 负载均衡已经很成熟了，通常我们接入多条线路，需要 RouterOS 路由器安装更多的以太网卡，以便接入更多的线路，如果 10 条左右的线路还能基本解决，但超过 10 条时候，路由器扩展网卡受到局限性，当前 CCR2004-16G-2S+有 16 个千兆电口的路由器，也只能接入 16 条线路。特别是多条光猫接入的 PPPoE 拨号方式，如果是一台 PC 的 RouterOS 路由器，可能希望安装一个万兆网卡解决 10-20 条线路的接入，或者是有一条光猫可以重复拨号三次，也需要在一张网卡上完成。

首先我们要了解运营商的 BAS 认证设备，如果接入是在同一台运营商的 BAS 下（PPPoE 认证设备），在该设备下每个拨号设备的网卡 MAC 地址是唯一的，不能重复。作为拨号的路由器 MAC 地址不能相同，所以才要求使用不同的网卡，解决 MAC 不同的问题，避免 PPPoE 拨号失败。

主流的单网卡多次拨号，大概有两种方式，一种是 Bridge vlan 方式，一种是 VRRP 方式，两种方式各有优缺点，VRRP 方式我更推荐用于单线多拨的方式。

Bridge VLAN 多线路拨号

早期民间高手想到了通过二层管理交换机实现了划分 VLAN 的方式来进行拨号，通过 24 口、48 口管理交换机划分多个 vlan id，Trunk 给 RouterOS 一个网卡，RouterOS 在 interface vlan 下建立对应的 VLAN id，但每个 VLAN 接口的 MAC 地址是从属于实际物理网卡的，VLAN 接口的 MAC 还是相同，所以还要将每个 VLAN 加入到一个独立的 bridge 方式，利用 bridge 可以设定 admin-mac 的特点，将每组拨号的 MAC 区分开，达到多线拨号能在一张网卡上实现。如下图：



通过简单的一个配置说下：

华为交换机接入多条光猫，每个交换机的电口划分一个 VLAN id，这里我们仅用 2 条线路举例，分别接入交换机的 1，2 口，24 口连接 RouterOS 的 ether1

```
interface GigabitEthernet0/0/1
port link-type access
port default vlan 10

interface GigabitEthernet0/0/2
port link-type access
port default vlan 20

interface GigabitEthernet0/0/24
 description RouterOS_ether1
 port link-type trunk
 undo port trunk allow-pass vlan 1
port trunk allow-pass vlan 10  20
```

RouteOS 配置，在 ether1 网卡下，创建 VLAN10，20 接口

```
/interface vlan
add interface=ether1 name=vlan10 vlan-id=10
add interface=ether1 name=vlan20 vlan-id=20
```

创建 bridge 接口，用于添加对应的 VLAN 接口，并设置 admin-mac 地址，每个 bridge 的 MAC 要不同

```
/interface bridge
```

```
add admin-mac=E4:8D:8C:1D:24:CC auto-mac=no fast-forward=no name=bridge-vlan10 \
    protocol-mode=none
add admin-mac=E4:8D:8C:1D:24:C1 auto-mac=no fast-forward=no name=bridge-vlan20 \
    protocol-mode=none
```

将对应的 VLAN 接口加入到对应的 Bridge 下

```
/interface bridge port
add bridge=bridge-vlan10 interface=vlan10
add bridge=bridge-vlan20 interface=vlan20
```

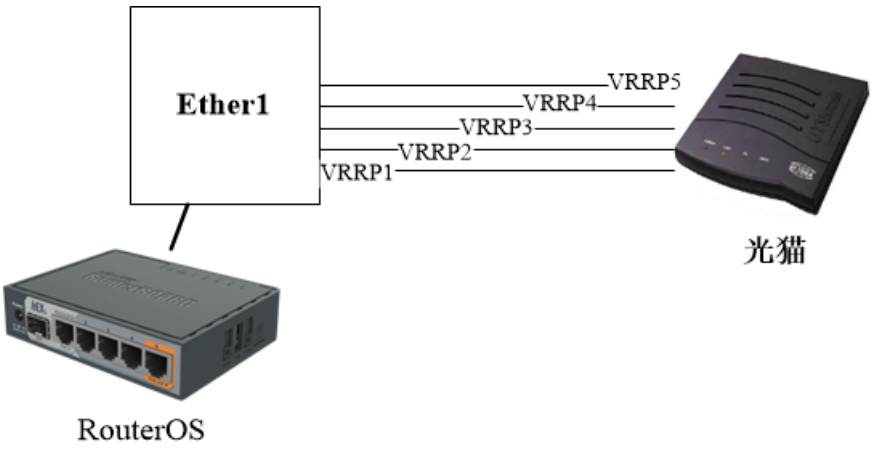
将 bridge-vlan10 和 bridge-vlan20 添加到对应的 pppoe-out 拨号接口，设置账号密码完成拨号。

```
/interface pppoe-client
add interface=bridge-vlan10 name=pppoe-out1 password=yus user=yus
add interface=bridge-vlan20 name=pppoe-out2 password=yus user=yus
```

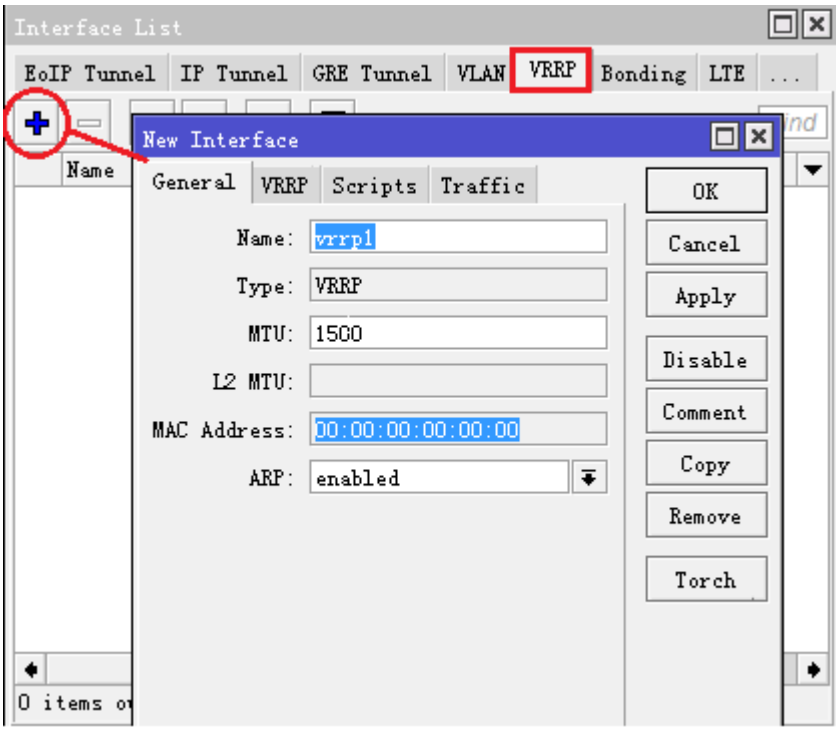
剩下的 PCC 负载均衡配置，请参阅 6.13 章节

单网卡 VRRP 多拨配置

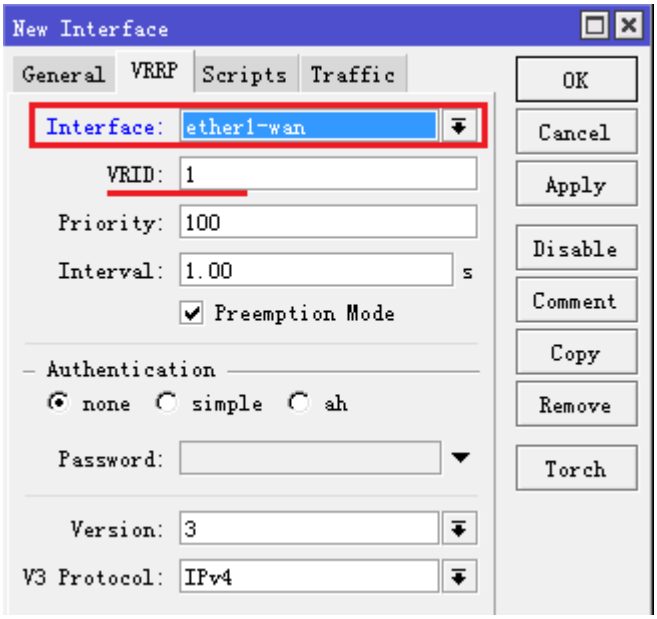
下图，一个光猫能同时 5 次拨号(仅仅是一个假设，大多只能同时 3 次拨号)，用来演示如何进行 VRRP 的拨号，这样偷个懒把多光猫拨号配置实例一起讲解了，因为配置是一样的。



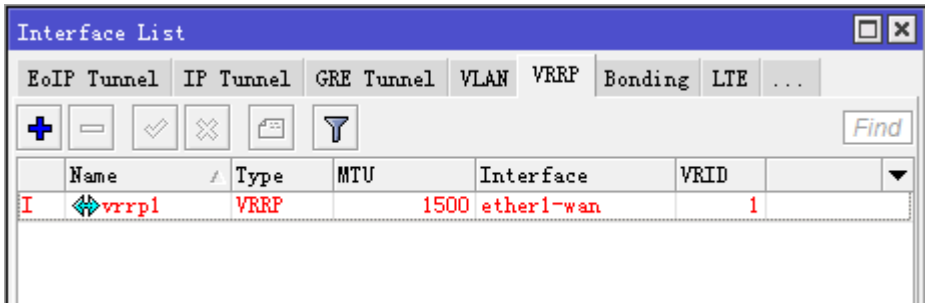
首先，进入 interface vrrp 菜单下，创建 vrrp 接口，



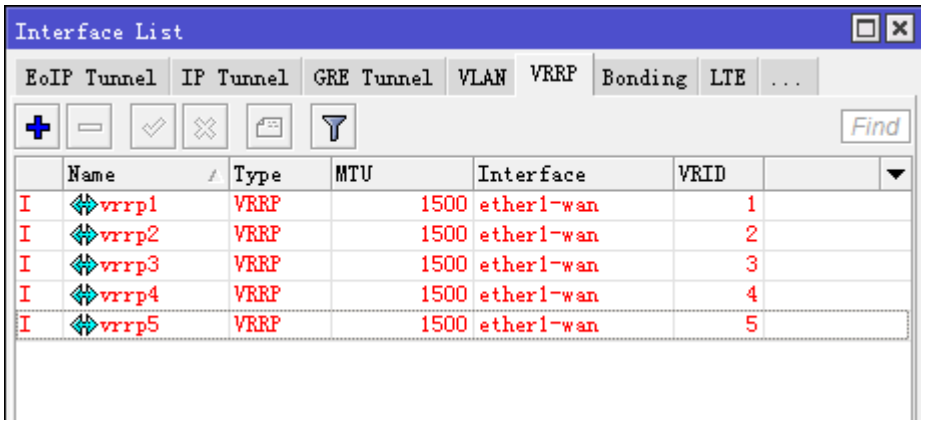
在添加项中，进入 VRRP 项设置 interface 和 VRID，这里 interface 是设置对应拨号的物理网卡，VRID 用于区分多个 VRRP 虚拟接口的身份，即每个用于拨号的 VRRP 虚拟接口 VRID 都不同，其他参数默认，这样 vrrp1 接口就在 ether1-wan 上生成了一个虚拟机口（如果你想仔细了解 VRRP，可以参考教材的 VRRP 章节）。



添加完成后，vrrp1 状态是红色，因为 vrrp1 接口和 ether1-wan 没有设置 IP 地址，之后我们会说明



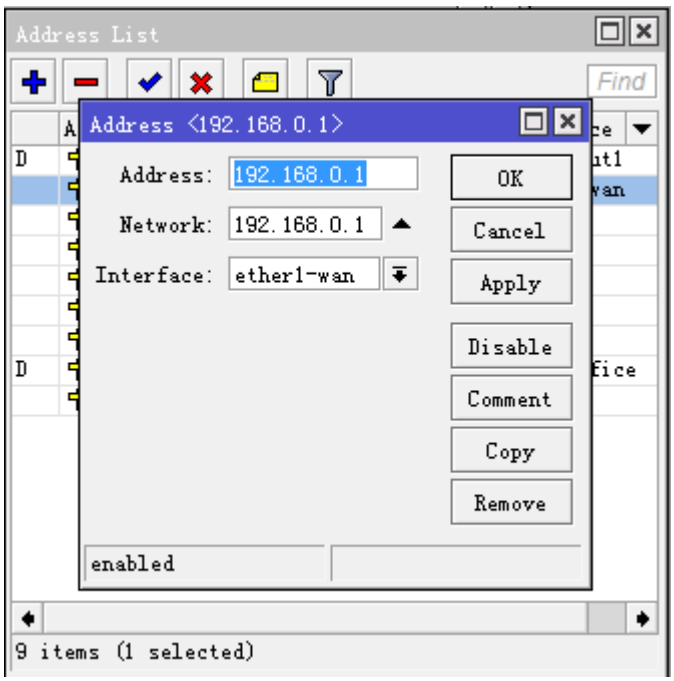
假设当前光猫能重复拨号是 5 次，这样需要建立 5 条 vrrp 虚拟接口，他们只是 VRID 不同，其他参数一样，5 条线路的 VRID 分别是 1～5，对于 5 条光猫线路也是如此配置。



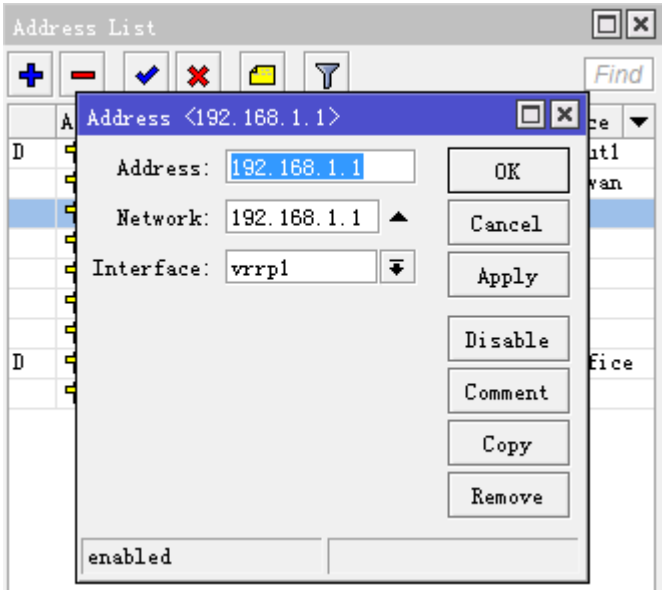
5 口虚拟的 VRRP 接口添加完成后，下面需要将它们启动，当前他们都不能正常使用，因为 VRRP 接口虚拟路由冗余协议，即对 2 台以上设备实现三层的冗余，当前 5 口接口都没有设置 IP 地址，即是不能生效使用的。

这里我们要进入/ip address 添加 ip 地址，不仅要添加 5 个 vrrp 虚拟接口的地址，还要添加 ether1-wan 接口 ip，至于设置什么 ip 地址，就随意了！目的是让 vrrp 接口生效。

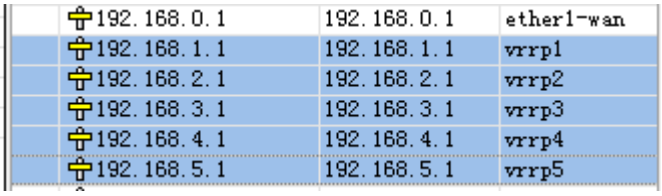
首先我们设置 ether1-wan 接口的 ip，192.168.0.1，这里我直接使用主机 IP 是可以生效的



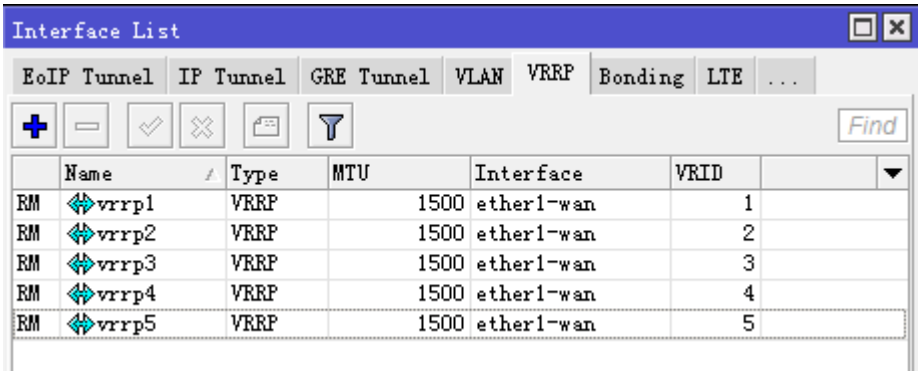
接着添加 vrrp1 虚拟接口的 IP 地址，同样设置为主机 IP，192.168.1.1，



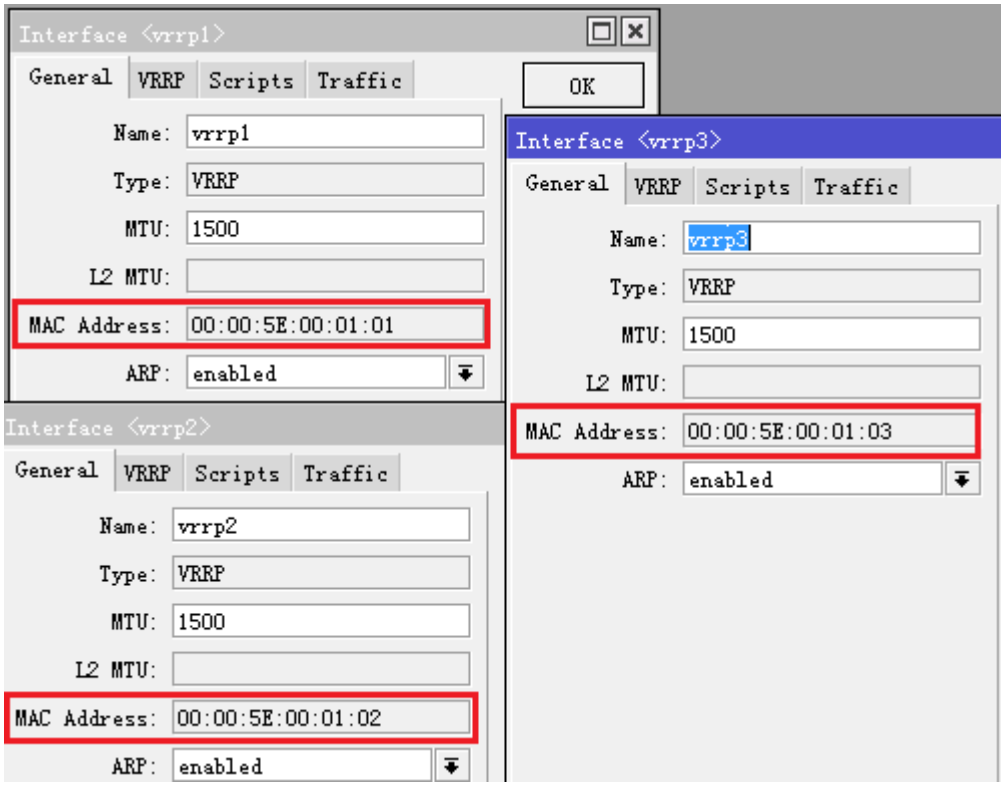
之后的配置以此类推 192.168.2.1-192.168.5.1，添加剩下的 4 个 vrrp 接口



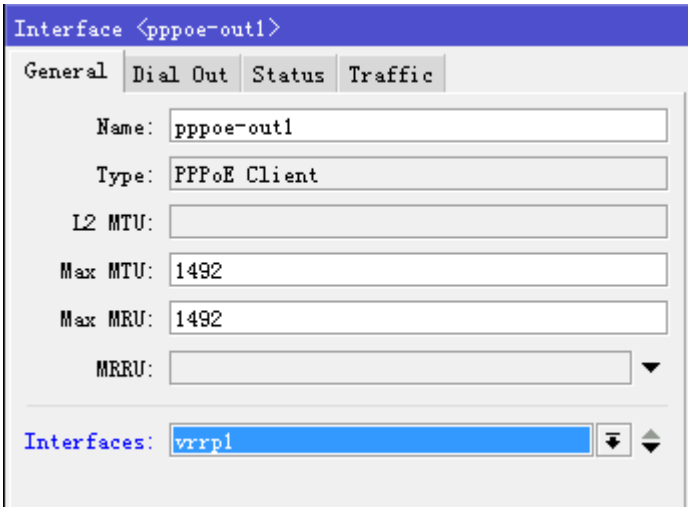
现在我们返回到 interface vrrp 菜单下，所有接口都进入了 RM 状态，即 VRRP 协议的 Master 状态



我们来对比下 VRRP 接口的 MAC 地址，都是不同的



现在我们可以将 vrrp 接口分别设置到 5 个不同的 pppoe-client 接口，实现 5 个不同账号的拨号了！



剩下建立 5 个对应 vrrp 接口的的 PPPoE 客户端拨号：

R	pppoe-out1	PPPoE Client		27.2 kbps
	pppoe-out2	PPPoE Client		0 bps
	pppoe-out3	PPPoE Client		0 bps
	pppoe-out4	PPPoE Client		0 bps
	pppoe-out5	PPPoE Client		0 bps

剩下的操作就 PCC 配置，参见 6.13 章节。

下面对比下两种多线拨号的优缺点：

方式	实现方式	优势	缺点
Bridge vlan	VLAN id 隔离，加入 Bridge，并修改 MAC 地址	每条拨号线路隔离	配置繁琐，vlan 和拨号账号需关联
VRRP	创建虚拟 VRRP 接口，修改 MAC	配置简单，vrrp 接口与账号无需关联	适合单线多拨

6.15 多线接入的路由问题

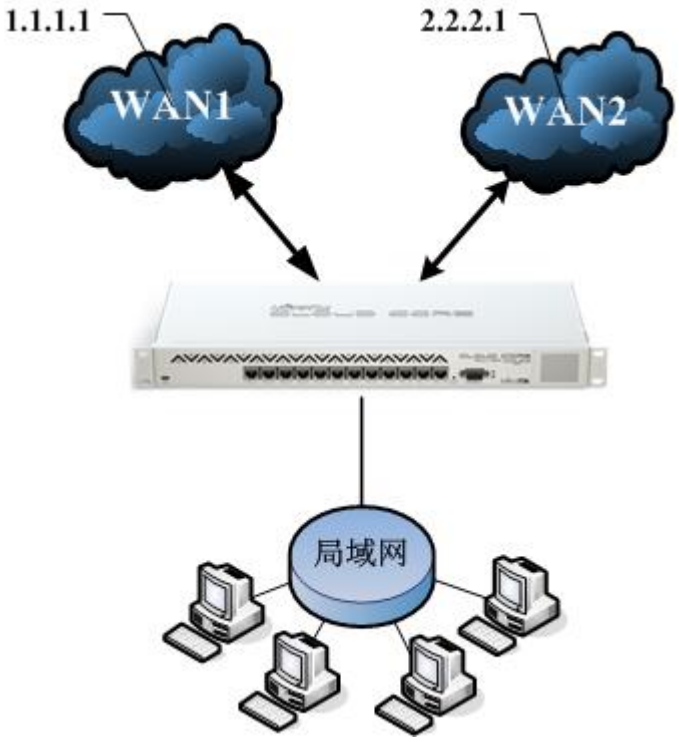
当做 RouterOS 的 PCC 多线路负载均衡的时候，我们会发现一组规则，在 ip firewall mangle 的 input 和 output 链表中出现的，这条规则用于非默认路由的返程，即从那个三层口到路由器，就从那个三层接口返回。通常路由器默认路由只有一条（ECMP 除外），返程路由可以保证数据不会被拒绝。

默认路由（Default Route）：即网关设备的预设路由出口，如果要讲就需要把三层网络和路由长篇大论一番，但这里就不细说，可以去网上查查。

PCC 返程路由配置

这里再次单独把 PCC 的返程路由配置列出，更详细点说明返程的配置，假设我们有两条线路，线路 A 和线路 B，做 PCC 负载均衡，PCC 的具体配置就不细说，来看看这两条返程路由的规则和默认路由的区别

- 线路 A：1.1.1.2/24，网关 1.1.1.1，接口 wan1
- 线路 B：2.2.2.2/24，网关 2.2.2.1，接口 wan2



首先配置 PCC mangle 标记（配置省略），主要看看 PCC 中的 mangle 标记中有以下两组规则

```
/ ip firewall mangle
```

```
add chain=input in-interface=wan1 action=mark-connection new-connection-mark=1st_conn
add chain=input in-interface=wan2 action=mark-connection new-connection-mark=2nd_conn
```

input 链表配置的是进入 wan1 口的数据标记为 1st_conn，进入 wan2 接口的标记为 2nd_conn

```
add chain=output connection-mark=1st_conn action=mark-routing new-routing-mark=pcc1
add chain=output connection-mark=2nd_conn action=mark-routing new-routing-mark=pcc2
```

在 output 链表中将 1st_conn 做路由标记到 pcc1，2nd_conn 标记到 pcc2

当然我们会把对应的标记指定到相同的网关上

```
ip route add gateway=1.1.1.1 routing-mark=pcc1
ip route add gateway=2.2.2.1 routing-mark=pcc2
```

注意正常情况下，我们会配置一条默认路由，例如：

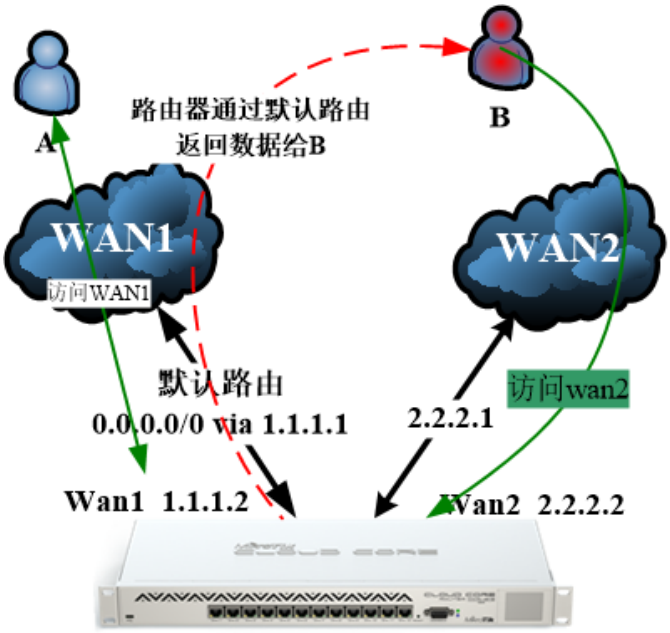
```
ip route add gateway=1.1.1.1
```

路由表的情况如下：

```
#      DST-ADDRESS                GATEWAY         DISTANCE     ROUTING-MARK
 0 A S  0.0.0.0/0                  1.1.1.1            1          -
 1 A S  0.0.0.0/0                  1.1.1.1            1         pcc1
 2 A S  0.0.0.0/0                  2.2.2.1            1         pcc2
```

返程路由原理

原理上面的实例来说明，双线接入的路由器，有 WAN1 和 WAN2，路由器只能配置一条默认路由，这里配置的是 WAN1 的默认路由，



上面场景告诉大家双线只有一条默认路由，无法同时访问两条线路的原因，如果 A 用户要访问路由器的 WAN1 接口，是没有问题的，因为 WAN1 进入，默认路由走 WAN1 出去，但如果 B 要访问 WAN2，由于默认路由设置的是 WAN1，因此进入 WAN2 的数据包不是按照从 WAN2 进入，再从 WAN2 出去，而是，从 WAN2 进入，WAN1 出去，因此返回给 B 用户的源 IP，不是 2.2.2.2，而是 1.1.1.2，那么数据包是非法的，会被丢弃。

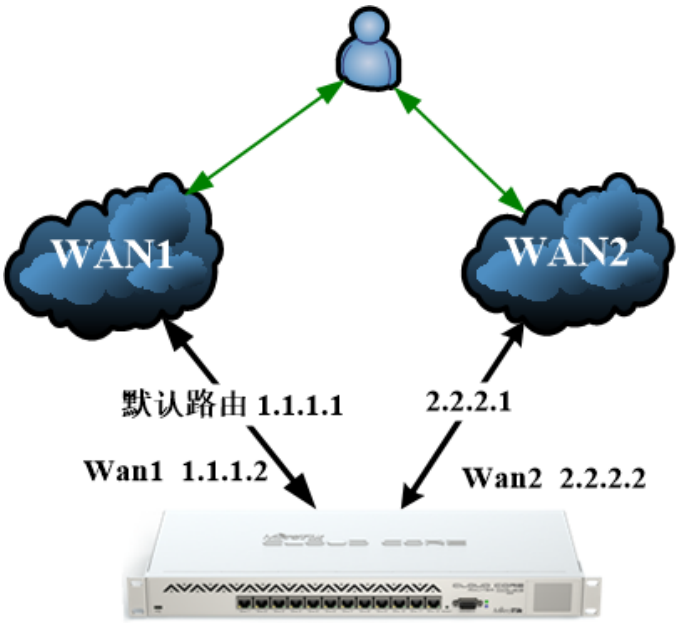
以上说明了失败的原因，那么解决此类问题需要通过策略路由完成。

返程路由的三种场景

返程路由问题，对于 RouterOS 多线接入情况下，遇到的问题最多，而且在不同的环境下，解决方案也有不同，这里我归纳了三种环境的返程路由问题，分别是路由器本地的返程路由，PCC 环境的端口映射返程路由，多线端口映射

环境 1：路由器本地返程路由

首先多线路接入到路由器，在使用多线的情况下，不涉及 nat 端口映射，仅仅是通过多线访问路由器进行管理，那只需要做 input 和 output 的 mangle 标记策略，如下图



RouterOS 有两条线路 WAN1 和 WAN2，分别是 1.1.1.2 和 2.2.2.2，默认网关为 1.1.1.1，也就是访问路由器只能通过 WAN1 进入，而 WAN2 没有配置路由，无法通过 2.2.2.2 访问路由器

要实现通过 WAN2 的 2.2.2.2 访问到路由器，需要配置返程路由，即标记从 WAN2 接口进入的连接，从 WAN2 接口出去，配置如下：

首先添加两条线路的 IP 地址

```
/ip address
add address=1.1.1.2/24 interface=wan1
add address=2.2.2.2/24 interface=wan2
```

需要添加 WAN1 的默认路由：路由器的默认路由走 WAN1，外部访问路由器默认 WAN1 线路是正常

```
ip route add gateway=1.1.1.1
```

配置 WAN2 线路的 mangle 标记， mangle 在 input 链表标记进入 WAN2 接口的连接为 wan2_con，WAN1 有默认路由，因此无需在配置策略

```
/ ip firewall mangle
add chain=input in-interface=wan2 action=mark-connection new-connection-mark=wan2_con
```

在 output 链表中将 wan2_con 做路由标记到 wan2_route

```
add chain=output connection-mark=wan2_con action=mark-routing new-routing-mark=wan2_route
```

把对应的标记指定到相同的网关上

```
ip route add gateway=2.2.2.1 routing-mark=wan2_route
```

路由表的情况如下：

```
#      DST-ADDRESS                GATEWAY         DISTANCE     ROUTING-MARK
```

```
 0 A S   0.0.0.0/0                    1.1.1.1             1           -
1 A S   0.0.0.0/0                    2.2.2.1             1           wan2_route
```

路由配置完成了，从外网能同时通过 WAN1 和 WAN2 访问到路由器。该配置也适用用于 PCC 负载均衡策略的路由器

注意：这个配置仅仅能实现访问到路由器本身，但如果需要做 nat 映射到内网设备，这个配置是无法实现的

环境 2：多线路的 nat 端口映射

现在我们要把问题延申到内网的服务器，网络结构仍然不变，默认路由还是在 WAN1 上，我们需要映射 TCP/3389 到内网服务器 192.168.1.8，通常端口映射配置为

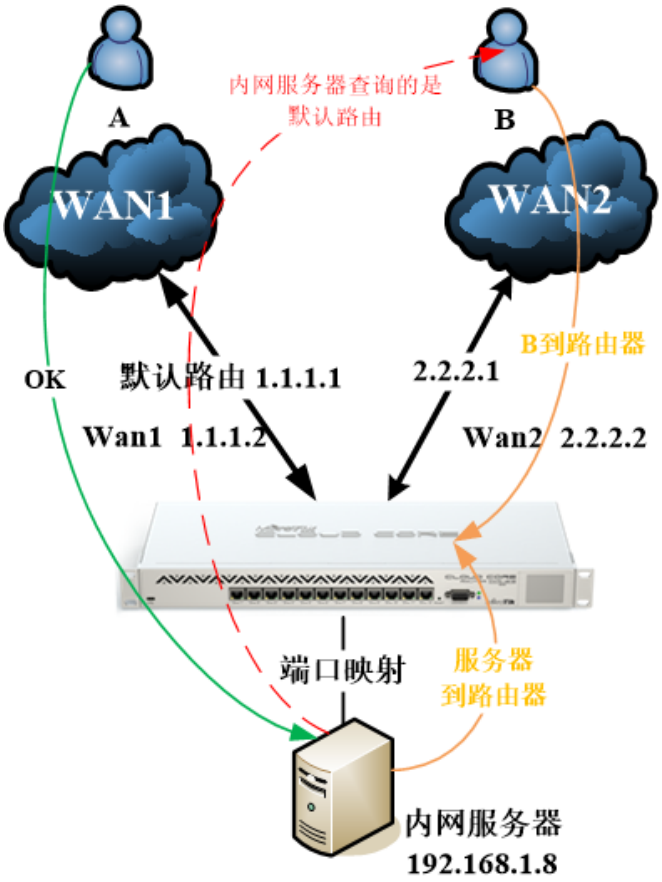
首先要配置默认的 src-nat 的伪装规则隐藏内网地址

```
/ip firewall nat
add chain=srcnat action=masquerade
```

配置相应的端口映射规则

```
/ip firewall nat
add chain=dstnat dst-address=1.1.1.2 dst-port=3389 protocol=tcp action=dst-nat
to-addresses=192.168.1.8   to-ports=3389

add chain=dstnat dst-address=2.2.2.2 dst-port=3389 protocol=tcp action=dst-nat
to-addresses=192.168.1.8 to-ports=3389
```

但结果是 A 用户，通过 WAN1 访问映射的服务器是 ok 的，但 WAN2 线路是失败的，之前不是配置了路由器的返程路由了，为什么会失败。这里我们要把到 WAN2 的路径分为两段，如下图



从 B 用户到路由器是 ok，因为有 input 和 output 策略，但从服务器到路由器这部分出问题了，内网服务器查询的是默认路由。因为 input 和 output 链表只管入和出路由器本地数据，而不管通过路由器的转发数据，因此我们要把标记策略进行修改

接下来需要在 mangle 标记 wan2 线路的策略，注意这里我们使用链表不再是 input，而是 prerouting，即路由之前，标记连接为 wan2_con

```
/ ip firewall mangle
add chain=prerouting in-interface=wan2 action=mark-connection new-connection-mark=wan2_con
```

prerouting 链表配置的是进入 wan2 口的数据标记的 wan2_con，通过 output 链表标记返回的路由，和之前的 PCC 一样，取名 wan2_route

```
/ip firewall mangle
add chain=output connection-mark=wan2_con action=mark-routing new-routing-mark=wan2_route
```

这样的配置并没有完，之前说过，由于 output 链表不会理会转发到互联网的数据，只是处理路由器本地路由，因此我们需要补充一条内网映射主机到外网的路由策略，也要返回到 wan2 口，配置如下：

```
/ip firewall mangle
add chain=prerouting connection-mark=wan2_con src-address=192.168.1.8/32 action=mark-routing
new-routing-mark=wan2_route
```

再把对应的标记指定到相同的网关上

```
ip route add gateway=2.2.2.1 routing-mark=wan2_route
```

环境 3，PCC 环境的端口映射

该环境是一个综合题，我们在做 PCC 负载均衡的时候，如果不涉及端口映射和多线访问路由器本地，我们可以忽略返程路由的配置，PCC 环境的路由器多线情况下只是需要访问路由器自身，用环境 1 的配置即可，但如果涉及到 nat 端口映射，那么我们又要重新考虑，考虑两种情况：

1. 被映射主机会匹配到 PCC 负载均衡策略，使用环境 1 的配置
2. 被映射主机不会匹配 PCC 负载均衡策略，使用环境 2 的配置

如果匹配到 PCC 策略，自然映射主机的路由会跟随 PCC 策略走到指定出口（input 和 output 策略配置正确），看看前面章节的 PCC 的策略标记原理。如果没有匹配到 PCC 策略，就会走默认路由出去，这里大家多思考下。

估计很多朋友看了，可能会被绕晕，不同的场景配置方法的确有差别，因为网络需求是在不断的变化的，适应不同的需求，就需要灵活的解决方案，这里我告诉了方法和原理，结果就需要大家去实践，喜欢 RouterOS 就说因为解决问题的方法的确很多。

策略路由导致外网 nat 端口映射内网访问失效

这个是很典型的问题，大多朋友都回遇到，都出现在配置策略路由，包括 PCC 负载均衡策略环境下。大家必须了解 RouterOS 在多线路下会创建新的路由表用于策略路由配置，被匹配到策略路由表的用户，无法查询到 main 表的路由表，路由器接口添加的 IP 或拨号接口地址，自动生成的直连路由都放在 main 表中，而对于新创建的策略路由表是不会有直连路由，只有一条网关路由。

公网的 nat 端口映射，访问到的是路由器本地接口 IP，因此大多数情况是在 main 表查询，例如一台路由器的公网 IP 地址是 10.20.10.2/24 接口在 ether1 上，并在/ip firewall nat 配置了 tcp/3389 的映射

```
/ip firewall nat
add action=dst-nat chain=dstnat dst-port=3389 dst-address=10.20.10.2 protocol=tcp
to-addresses=192.168.88.88 to-ports=3389
```

路由器的内网 IP 地址是 192.168.88.1/24，配置在 ehter3 接口，用户 IP 地址是 192.168.88.8，通过 mangle 标记或者/ip firewall mangle 策略配置到了新的路由表 route1 查询新的出口网关，

```
[admin@MikroTik] /ip firewall mangle> print
Flags: X - disabled, I - invalid, D - dynamic
 1    chain=prerouting action=mark-routing new-routing-mark=route1
      passthrough=yes src-address=192.168.88.8 log=no
```

无法在 main 表中找到 10.20.10.0/24 的直连路由，这样当内网用户要访问路由器本地 10.20.10.2 的 IP 是只能被直接转发到另外一个出口，无法在本地查询路由。解决这个方法是在 mangle 中允许从 ether3 接口进入到目标 IP 是 10.20.10.2 的任何源地址接受并通过

```
[admin@MikroTik] /ip firewall mangle> add action=accept chain=prerouting
dst-address=10.20.10.2 in-interface=ether3
```

查看 manlge 规则顺序

```
[admin@MikroTik] /ip firewall mangle> print
Flags: X - disabled, I - invalid, D - dynamic
 0    chain=prerouting action=mark-routing new-routing-mark=route1
      passthrough=yes src-address=192.168.88.8 log=no log-prefix=""

 1    chain=prerouting action=accept dst-address=10.20.10.2 in-interface=ether3
      log=no log-prefix=""
```

需要将规则移动到顶部

```
[admin@MikroTik] /ip firewall mangle> move 1 0
```

这样就解决了因为策略路由导致内网访问公网映射的问题

对于 PPPoE 拨号或者 dhcp 获取的公网 IP 来说，因为公网 IP 是动态变换的，如果要用此规则，需通过脚本随时判断 IP 是否变动，并替换，这样操作毕竟麻烦，我们可以改进配置，如下：

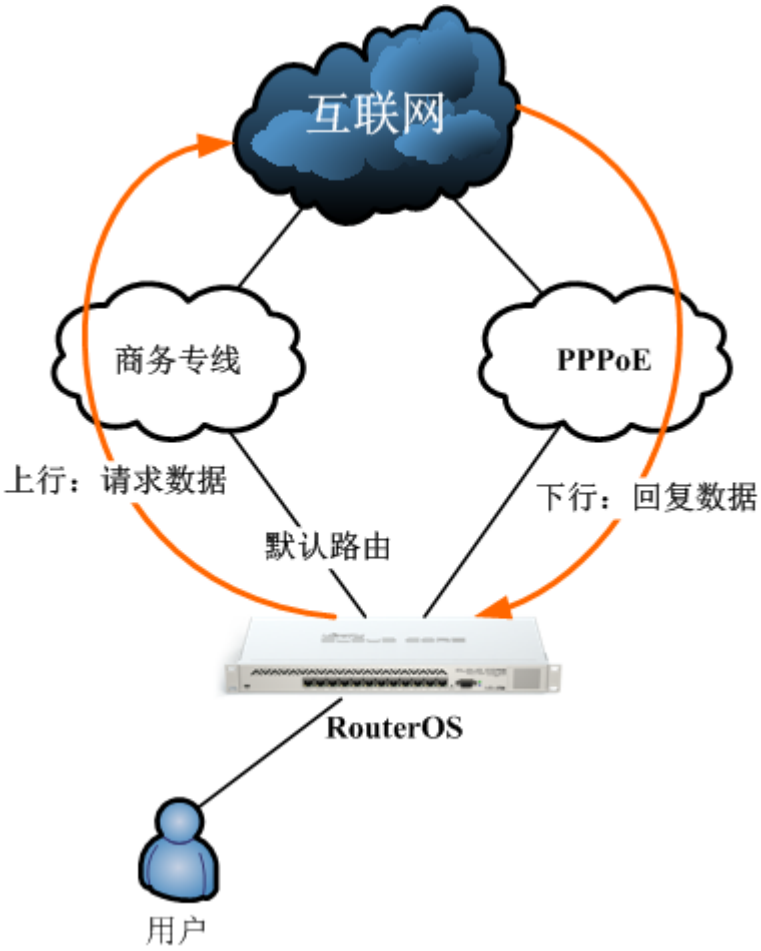
```
[admin@MikroTik] /ip firewall mangle>add action=accept chain=prerouting
dst-address-type=local in-interface=ether3
```

规则含义是从 ether3 内网端口进入，到目标地址类型是 local 本地的数据接受通过，记住此规则仍然要保持在 mangle 序列顶部。

6.16 PPPoE 走下行，商务专线走上行

由于受到运营商 PPPoE 拨号上下行不对称问题，限制了上行带宽，导致部分网络应用无法获得足够的上行带宽，而光纤（商务光纤）的上下行带宽是对等，如果两种接入方式在一定程度上相互补充，光纤走上行，PPPoE 走下行就非常完美了。关于这个问题，网络技术上已经流传很久，实现方式主要涉及路由和 nat 两个功能。

基本原理是将默认网关设置为光纤固定 IP，在做 nat 的时候将内网伪装为 PPPoE 的 IP 地址（要求获取公网 IP 地址，且专线和 PPPoE 是相同运营商），但这里有一个前提运营商接入光纤的设备不检测源地址是否合法，因为是将 PPPoE 获取的 IP 地址走光纤出口，当目标网站收到你的请求看到的 IP 是 PPPoE 的（上行请求），目标网站回复请求，将数据返回给 PPPoE 的 IP，但路径是按照正常的路由选择到达 PPPoE 的接口（下行数据）。这样实现了上下行出口的分离，因此这种操作网络上属于不对称访问。下面的实例仅供大家参考：



我们通下面一个接入 PPPoE 和光纤固定 IP 的实例：

- 光纤专线 IP 地址：172.18.200.2/24，网关是 172.18.200.1
- PPPoE 拨号：账号 yus，密码 yus，自动获取 IP 地址
- 内网用户 IP 地址：192.168.88.1/24

首先我们创建 PPPoE 拨号

```
[admin@MikroTik] > /interface pppoe-client
[admin@MikroTik] /interface pppoe-client>add interface=ether2-pppoe name=pppoe-out1 password=yus
user=yus
```

添加商务专线 IP 地址

```
[admin@MikroTik] > /ip address
[admin@MikroTik] /ip address> add address=172.18.200.2/24 interface=ether3-wan
```

添加商务专线的默认路由

```
[admin@client] >/ ip route
[admin@client] /ip route> add gateway=172.18.200.1
```

查看当前 PPPoE 获取的 IP 地址

```
[admin@MikroTik] /ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #     ADDRESS                  NETWORK              INTERFACE
 0     192.168.88.1/24     192.168.11.0            ether1-lan
 1     172.18.200.2/24      172.18.200.0           ether3-wan
 2 D 172.20.0.2/32          172.20.0.1            pppoe-out1
```

通过上面可以看到 pppoe-out1 拨号获取的 IP 地址是 172.20.0.2，

查看下路由表

```
[admin@MikroTik] >/ip route
[admin@MikroTik] /ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
  #        DST-ADDRESS            PREF-SRC            GATEWAY                DISTANCE
0 A S   0.0.0.0/0                                  172.18.200.1                1
1 ADC   172.18.200.0/24      172.18.200.1       ether3-wan                    0
2 ADC   192.168.88.0/24      192.168.88.1       ether1-lan                    0
3 ADC   172.20.0.2/32        172.20.0.1         pppoe-out1                     0
```

配置 nat 规则，这里我们假设运营商接入点商务专线，对源地址没有校验，即 PPPoE 拨号 IP 地址能通过光纤专线请求到目标地址，即利用商务专线的上行。

```
[admin@MikroTik] /ip firewall> /ip route
[admin@MikroTik] /ip route> /ip firewall nat
[admin@MikroTik] /ip firewall nat>add chain=srcnat action=src-nat src-address=192.168.88.0/24
to-addresses= 172.20.0.2   comment="nat"
```

配置完成后，用户的上行请求会走商务专线出去，返回的下行数据会走 PPPoE 接口回来，在 src-nat 规则上添加一个 comment 注释，目的是在后面的脚本查询时使用。

对于 PCC 的负载均衡的朋友，mangle 策略配置不变，在/ip route 的策略路由不需要配置，需要添加多个 src-nat 策略，并匹配 connection-mark 标记，以上面的 srcnat 配置为例，加入 pcc1 的连接标记，其他 srcnat 规则以此类推，如下：

```
add chain=srcnat action=src-nat connection-mark=pcc1 src-address=192.168.88.0/24 to-addresses=
172.20.0.2   comment="nat"
```

由于做 src-nat 规则，需要填写 PPPoE 获取的 IP 地址，由于 PPPoE 获取 IP 地址是动态，所以需要通过脚本修改，下面脚本配置参考

```
:local newip
:local status
:local natip
:set status [/interface get [find name="pppoe-out1"] running]
:log info ("get pppoe status" . $status)
```

```
:if ($status=true) do={
  :set newip [/ip address get [ find dynamic=yes interface=("pppoe-out" . $i)] address]
  :set newip [:pick $newip 0 [:find $newip "/"]]
  :set natip [/ip firewall nat get [ find comment="nat"] to-addresses]
  :if ($natip != $newip) do={
  :log info ("pppoe ip change " . $newip)
  /ip fir nat set [ find comment=”nat”] to-addresses=$newip
}
}
```

脚本添加到计划任务执行，具体可以参考 39 章节

第七章 DHCP 操作配置

DHCP(动态主机分配协议)，负责分配和接收网络中的 IP 地址信息，能让网络内的主机动态获取 IP 地址，连接指定的网络。RouterOS 支持服务端（Server）和客户端 （Client）,同时支持 DHCP-relay 接力传输功能和客户端的静态绑定。

7.1 DHCP-Client 设置

操作路径: ***/ip dhcp-client***

MikroTik RouterOS DHCP-client 可以启用在任意类以太网网络接口，client 能接受一个 IP 地址、子网掩码、默认网关和两个 DNS 服务器地址。DHCP client 获取的 IP 将会自动添加到 ip address，默认网关也会自动添加到 ip route 中，如果 DHCP client 规则被禁用，这些 IP 和网关都会自动消失，如果在 ip route 中已经存在一条静态的默认路由，优先级高于 DHCP-client 默认路由，DHCP-client 路由将会是非法状态

RouterOS DHCP cilent 会向 DHCP 服务器询问以下 option 代码：

- option 1 - SUBNET_MASK 设置子网掩码
- option 3 - GATEWAY_LIST 设置网关地址选项
- option 6 - TAG_DNS_LIST 设置 DNS 服务器
- option 33 - STATIC_ROUTE 设置静态路由选项，如果存在 Option 121 则忽略该选项
- option 42 - NTP_LIST 设置 NTP 对时
- option 121 - CLASSLESS_ROUTE 设置无类别路由 classless route 选项

属性描述

属性	描述
add-default-route (*yes* | *no* | *special-classless*; 默认: **yes**)	是否接受 DHCP 服务器分配的默认路由。RouterOS 的 DHCP-Client 遵循 RFC，当接收到 Option 121 时，会忽略 Option 3。如果要强制 DHCP-Client 不忽略 Option 3，请设置 special-classless。该参数在 v6rc12+后添加。 • **yes** – 如果收到 option 121 classless route，会添加路由表，如果未收到 option 121 则默认添加 option 3 的默认路由。 • **special-classless** – 同时添加 classless route 和默认路由
default-route-distance (*integer: 0..255*; 默认:)	当 `add-default-route` = `yes` ，可以设置 Distance 属性用于路由距离判断
interface (*string*; 默认:)	DHCP client 运行的网络接口。
script (*script*; Default:)	在状态改变时执行脚本。脚本可以直接访问的变量如下： • `bound` - 1 – lease 租约被添加和改变; 0 – lease 租约被删除 • `server-address` – 服务器地址

	• lease-address - lease address provided by server • interface – 配置的网络接口 • gateway-address – 服务器提供的网关 IP 地址 • vendor-specific – 接收到 DHCP 服务器 option43 值 • lease-options – 接收到 options 队列值 实例 >>
use-peer-dns (*yes* | *no*; 默认: **yes**)	是否接受 DHCP server 分配的 DNS 服务器，将自动写入/ip dns 配置
use-peer-ntp (*yes* | *no*; 默认: **yes**)	是否接受 DHCP 服务器分配的 NTP 对时设置，将自动写入 /system ntp client 配置

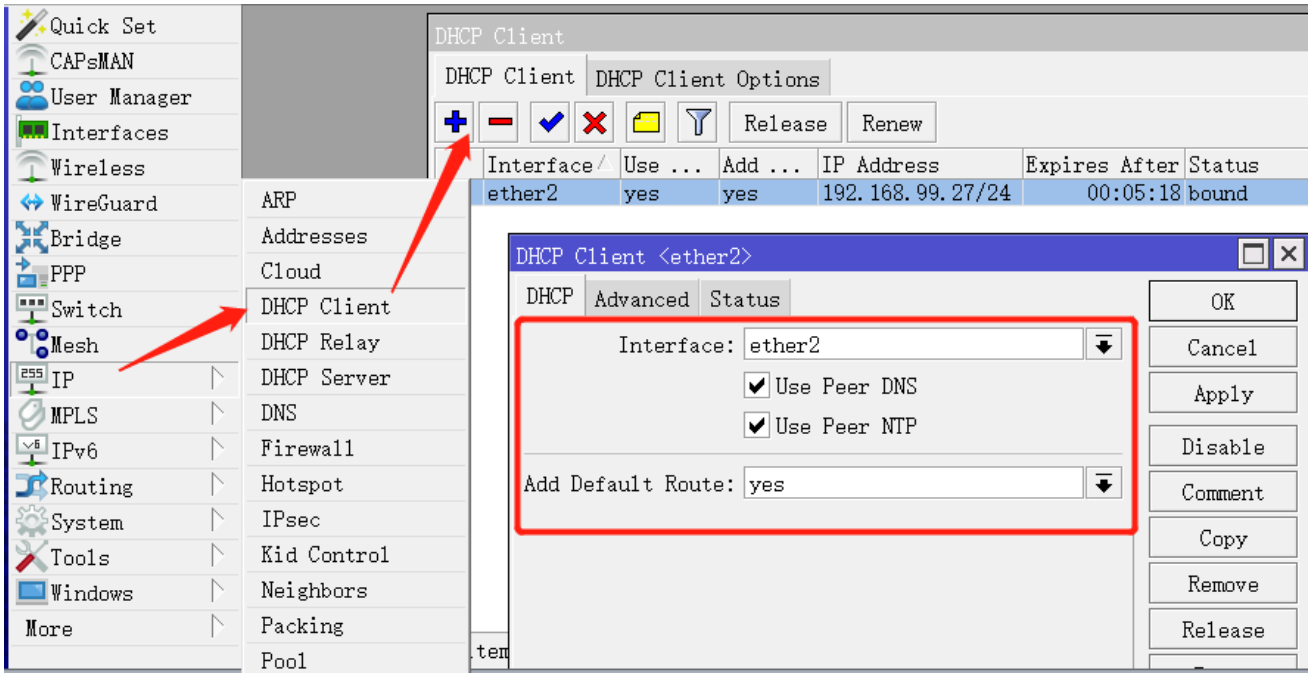
命令描述

renew（id） – 更新当前的租约，如果更新操作没有成功，客户端将试着初始化租约。
release (id) – 释放当前 DHCP 绑定，并重启 DHCP 客户端

事例：在 **ether1** interface 启用 DHCP-client：

```
/ip dhcp-client add interface=ether1 disabled=no
[admin@MikroTik] ip dhcp-client> print detail
Flags: X - disabled, I - invalid
 0    interface=ether1 add-default-route=yes use-peer-dns=yes use-peer-ntp=yes
      status=bound address=192.168.0.65/24 gateway=192.168.0.1
      dhcp-server=192.168.0.1 primary-dns=192.168.0.1 primary-ntp=192.168.0.1
      expires-after=9m44s
[admin@MikroTik] ip dhcp-client>
```

Winbox 操作配置 ether2 接口自动获取 IP 地址：



- User-Peer-DNS：代表获取对端分配的 DNS 服务器
- User-Peer-NTP：代表使用对端分配的 NTP 网络对时服务
- Add-Default-Route=yes：代表自动添加默认网关到/ip route 下

当 status 状态是 bound 时，表示已经获取 IP 地址

Lease 租约脚本实例

从 RouterOS v6.39rc33 开始，当 DHCP client 获得新的租约或失去现有租约时，都可以执行脚本。自动添加一个带有 routing-mark=WAN1 的默认路由，并在租约改变时，对路由进行操作：

该脚本添加在 dhcp-client 规则下的 script 中，可以直接获取 bound，gateway-address 变量

```
/ip dhcp-client
add add-default-route=no dhcp-options=hostname,clientid disabled=no interface=ether2 script="{\r\
    \n      :local rmark \"WAN1\"\r\
    \n      :local count [/ip route print count-only where comment=\"WAN1\"]\r\
    \n      :if (\$bound=1) do={\r\
    \n          :if (\$count = 0) do={\r\
    \n              /ip route add gateway=\$\"gateway-address\" comment=\"WAN1\"
routing-mark=\$rmark\r\
    \n          } else={\r\
    \n              :if (\$count = 1) do={\r\
    \n                  :local test [/ip route find where comment=\"WAN1\"]\r\
    \n                  :if ([/ip route get \$test gateway] != \$\"gateway-address\") do={\r\
    \n                      /ip route set \$test gateway=\$\"gateway-address\"\r\
    \n                  }\r\
    \n              } else={\r\
    \n                  :error \"Multiple routes found\"\r\
    \n              }\r\
    \n          }\r\
    \n      } else={\r\
    \n          /ip route remove [find comment=\"WAN1\"]\r\
    \n      }\r\
    \n}\r\
    \n"
```

7.2 DHCP-Server 设置

操作路径：***/ip dhcp-server***
关联操作：***/ip pool***

属性描述

dhcp server interface (*名称*) – 选择 DHCP 服务的网络接口
dhcp address space (*IP 地址/子网*; 默认：**192.168.0.0/24**) – DHCP 服务器将出租给客户端的网络地址段

gateway (*IP 地址*; 默认: **0.0.0.0**) – 分配给客户端的网关地址
dhcp relay (*IP 地址*; 默认: **0.0.0.0**) – 在 DHCP 服务器与 DHCP 客户端的 DHCP 接力的 IP 地址
addresses to give out (*文本*) – DHCP 服务器分配给客户端的 IP 地址池
dns servers (*IP 地址*) – 分配给 DHCP 客户端的 DNS 服务器地址
lease time (时间; 默认: **3d**) – 使用的租期时间

在命令行中，DHCP server 提供 setup 命令的向导配置，首先需要在对应的接口配置 IP 地址，确保接口下的 IP 网络正常通信，接口配置到 ether3，IP 地址 192.168.88.1/24 作为内网网关。

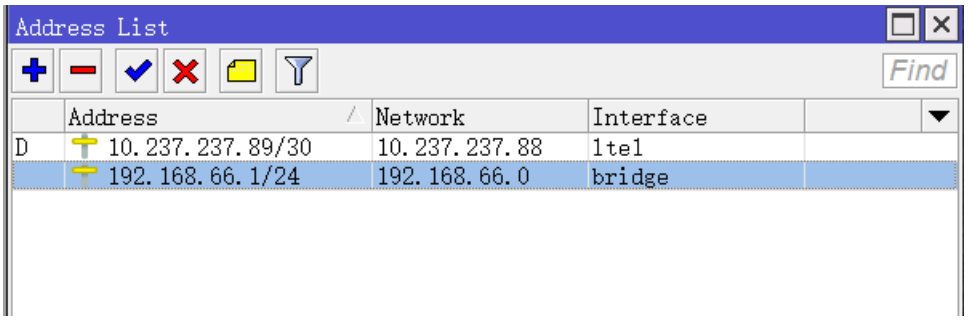
```
[admin@MikroTik] > /ip address add address=192.168.88.1/24 interface=ether3 disabled=no
```

IP 地址配置完成后，可以通过 setup 命令进行向导配置：DNS 服务器 IP 地址是 10.155.126.1 和 10.155.0.1

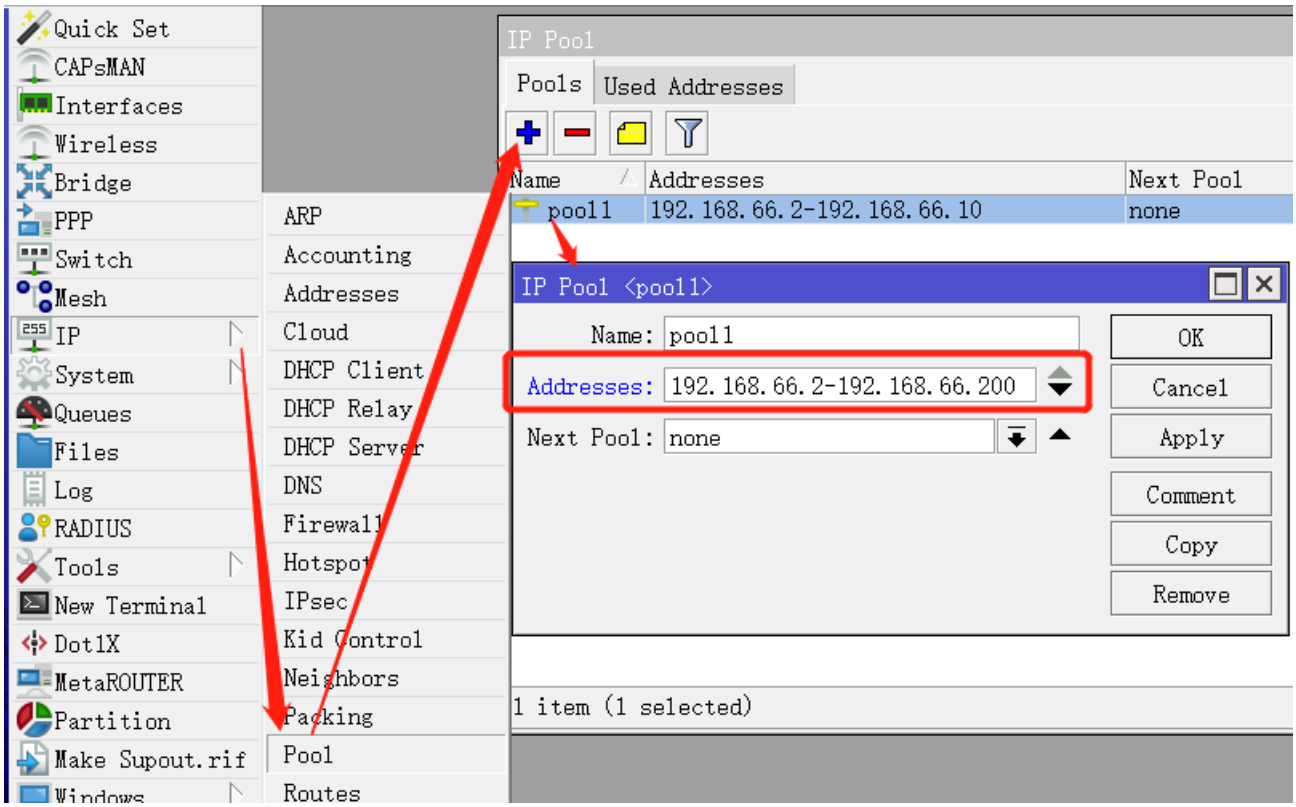
```
[admin@MikroTik] > /ip dhcp-server setup
Select interface to run DHCP server on

dhcp server interface: ether3
Select network for DHCP addresses

dhcp address space: 192.168.88.0/24
Select gateway for given network

gateway for dhcp network: 192.168.88.1
Select pool of ip addresses given out by DHCP server

addresses to give out: 192.168.88.2-192.168.88.254
Select DNS servers

dns servers: 10.155.126.1,10.155.0.1,
Select lease time

lease time: 10m
```

以上是命令行操作，下面的实例通过 Winbox 演示：

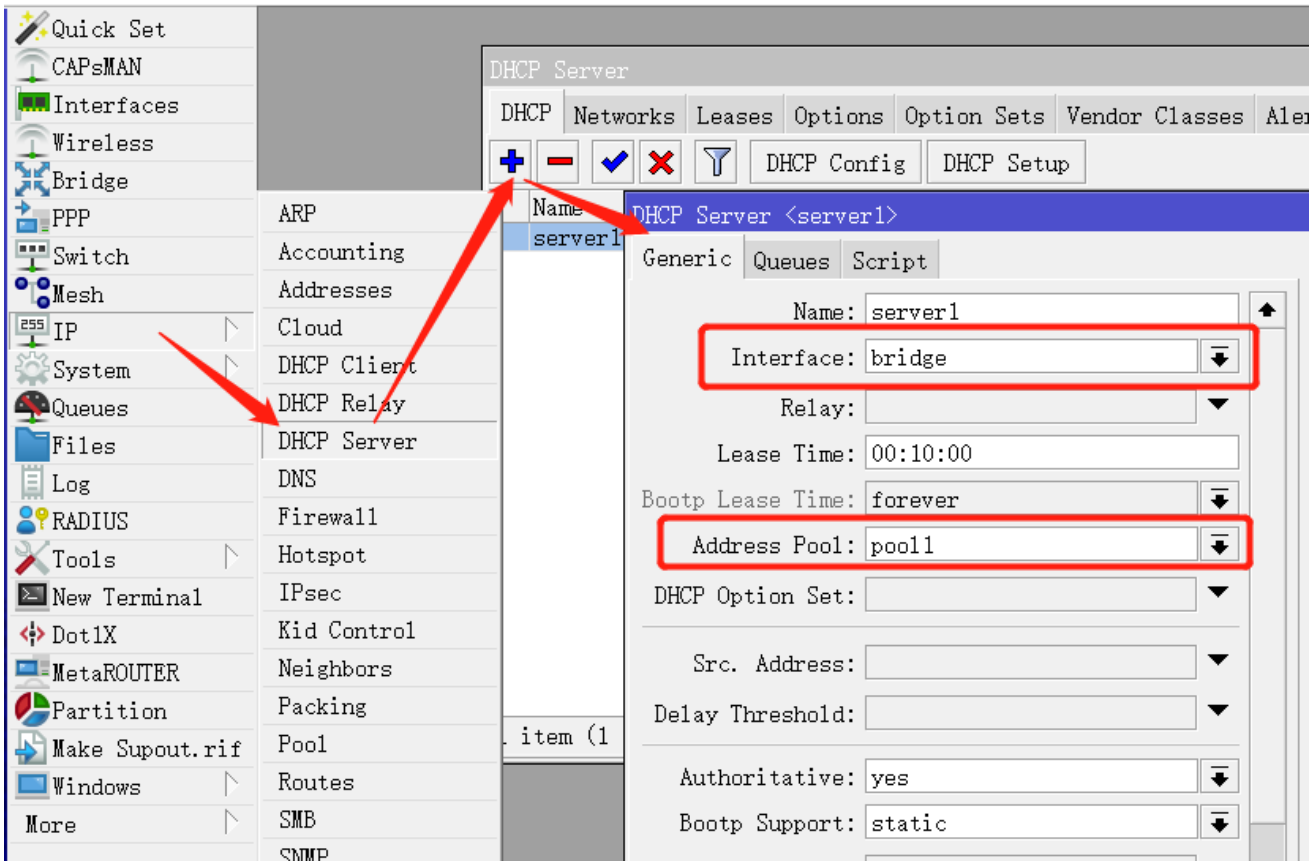
这里的内网 IP 我们配置到 bridge 接口，/ip address 配置 IP 地址 192.168.66.1 到 bridge 接口



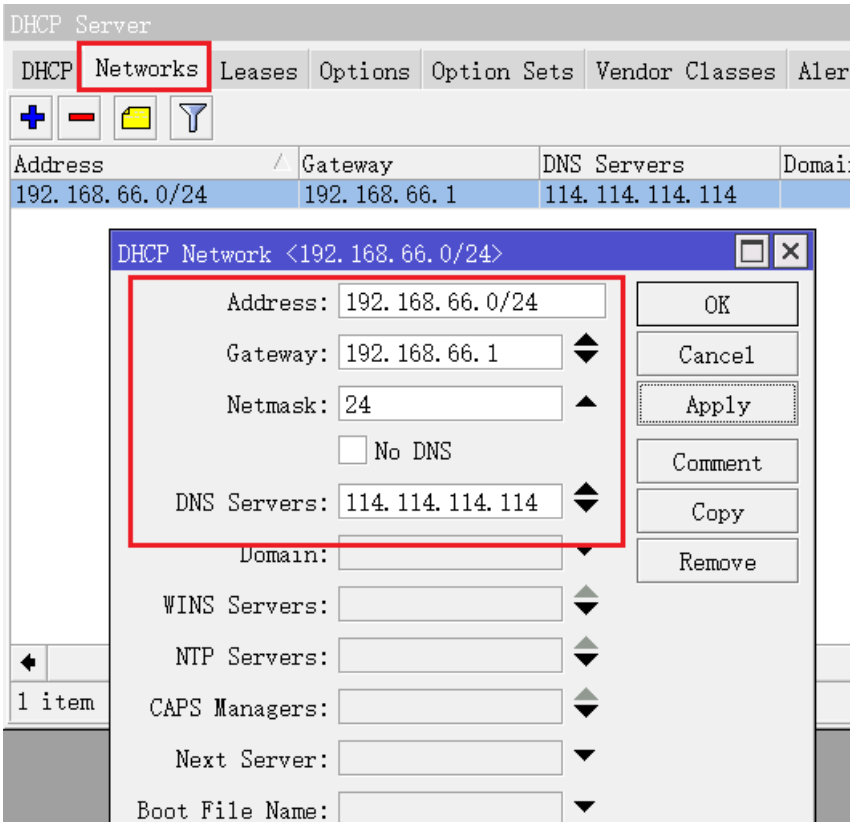
添加 DHCP 服务，首先进入/ip pool 中分配地址池范围，注意添加时要排除网关地址：



进入/ip dhcp-server 添加 DHCP 服务接口到 bridge 和对应的 pool 地址池



在/ip dhcp-server network 中新建规则，address 参数限制该 bridge 接口只能是网段 192.168.66.0/24（该参数在多网卡下配置多个 DHCP 服务器起作用，用于区分不同网卡下的不同 IP 地址），然后分配网关、子网和 DNS 参数：



注意：如果 DHCP-server 规则显示为红色，可能原因是所属接口没有连接“no-link”状态，或者在/ip address 里没有正确配置该接口的 IP 地址。

DHCP Lease 租约

DHCP 服务开启后，都会分配给每个客户端一个 IP 地址，并分配一个获取该地址的租约时间，默认是 3 天，在之前的 DHCP server 配置可以看到，已经分配的客户端地址可以在 lease 租约菜单下看到：

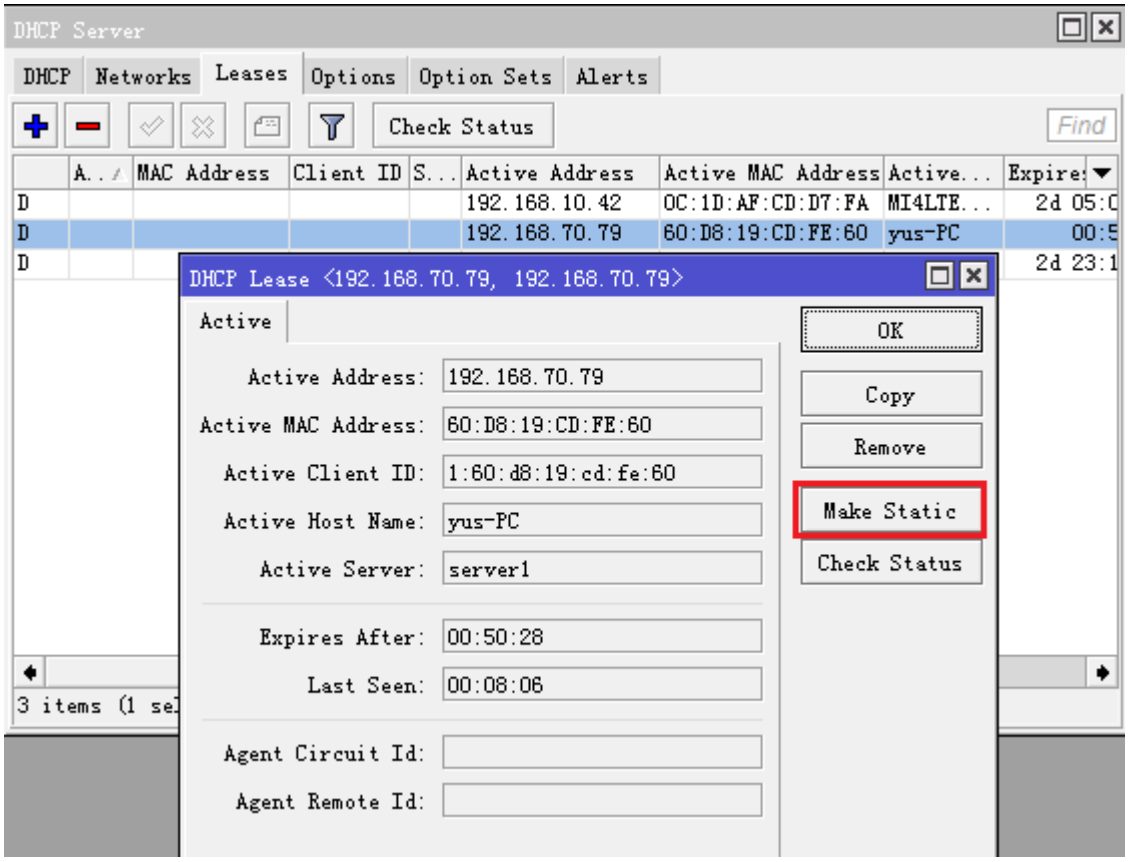
```
[admin@MikroTik] /ip dhcp-server lease> print
Flags: X - disabled, R - radius, D - dynamic, B - blocked
 #     ADDRESS         MAC-ADDRESS       HOST-NAME      SERVER     RATE-LIMIT     STATUS
 0 D 192.168.10.42     0C:1D:AF:CD:D7:FA MI4LTE-bu...    server2                   bound
 1 D 192.168.70.79     60:D8:19:CD:FE:60 yus-PC          server1                   bound
 2 D 192.168.10.44     14:F6:5A:99:90:37 android-2...    server2                   bound
```

当某个客户端要求固定获取一个 IP 地址时，可以设置静态租约，让客户端固定获取 IP 地址，还需要一个条件即知道客户端的 MAC 地址，即将 IP 和 MAC 做绑定，设置静态租约有两种方式，一种是在当前获取的客户端设置为静态，一种是手动添加静态

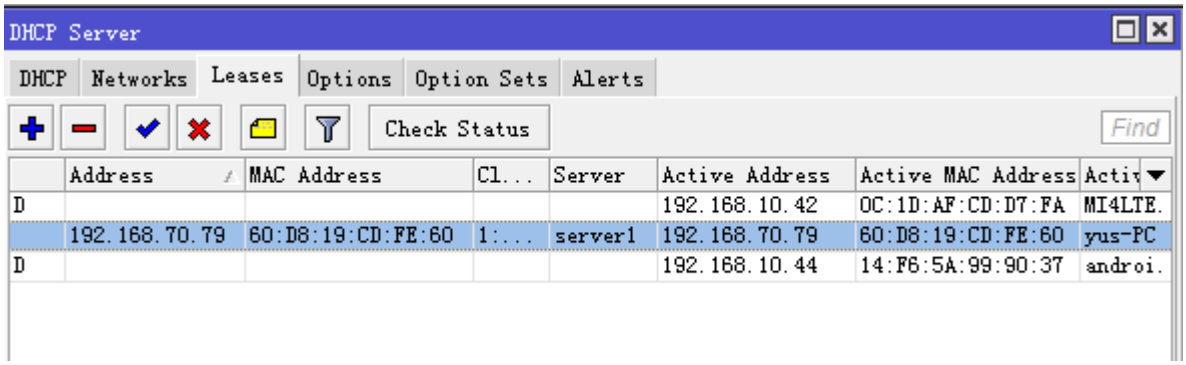
下面通过 winbox 操作来演示如何通过 DHCP Lease 将一个客户端 MAC 和 IP 做静态绑定，假设我们有一台主机 192.168.70.79 和 MAC 地址为 60:D8:19:CD:FE:60，要求做静态绑定

方法一、Make static

在 winbox 中可以使用 Make Static 功能将动态的客户端租约变为静态

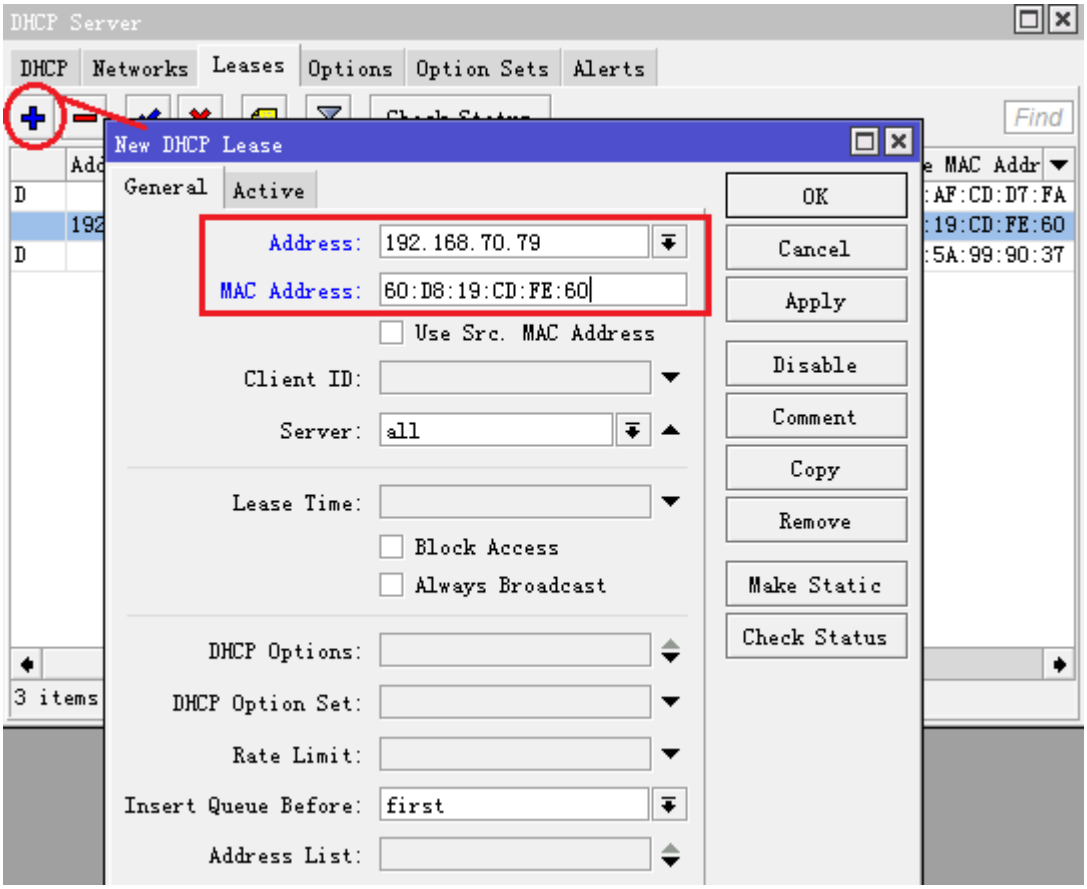


配置后可以看到租约静态绑定的效果：



方法二、手动添加

手动添加，直接将 IP 和 MAC 地址添加到对应栏目中



注意：RouterOS v6 在 DHCP lease 中新增了 insert queue Before 菜单，即当指定客户端获取 IP 后，可以自动添加带宽策略到 queue simple 中，在 Insert queue before 可以选择在某条队列规则前还是后。

7.3 DHCP Options

操作路径: /ip dhcp-server option

DHCP 报文中的一个选项，该选项在 DHCP 报文中为可变长的字段，option 选项中包含了部分租约信息、报文类型等，option 选项中最多可以包括 255 个 option。

根据 DHCP 协议，一个参数返回到 DHCP 客户端，只有在他请求这个参数时。指定各自的代码中 DHCP 请求参数列表（Parameter-List code55），如果代码没有包含着参数列表，DHCP 服务器将不会发送到 DHCP 客户端

Classless static Route

Classless static route 无类静态路由会添加到 DHCP 客户端的路由表中，下面实例中将会添加静态路由 dst-address=160.0.0.0/24 gateway=10.1.101.1，由于 Option 的值中我们需要使用十六进制格式，添加静态路由可以使用 code 249 和 121，下面以 code 121 为例，因为 RouterOS DHCP-client 只支持 121

首先需要掌握如何配置 option 值，根据 RFC3442 对格式定义如下：

子网段	子网掩码	目标路由格式

0	0	0
10.0.0.0	255.0.0.0	8.10
10.0.0.0	255.255.255.0	24.10.0.0
10.17.0.0	255.255.0.0	16.10.17
10.27.129.0	255.255.255.0	24.10.27.129
10.229.0.128	255.255.255.128	25.10.229.0.128
10.198.122.47	255.255.255.255	32.10.198.122.47

因此 dst-address=160.0.0.0/24，目标路由格式为 24.160.0.0，网关为 10.1.101.1，

整个格式为：24.160.0.0.10.1.101.1，现在我们要将以上格式换算为十六进制：

十进制	24	160	0	0	10	1	101	1
十六进制	**18**	**A0**	**00**	**00**	**0A**	**01**	**65**	**01**

结果是:18A000000A016501

如果 DHCP-Server 设置了 option code 121，RouterOS 的 DHCP-client 只识别 121 的路由，默认网关 code 3 会忽略，所以我们需要在 code 121 值中添加一条默认路由，假设默认网关为 10.1.101.1，换算为 000A016501，

所以两组路由结合，按照十六进制的写法是 0x18A000000A016501000A016501(0x 为十六进制格式)，配置如下：

```
/ip dhcp-server option
add code=121 name=classless value=0x18A000000A016501000A016501
/ip dhcp-server network
set 0 dhcp-option=classless
```

RouterOS 的 DHCP-client 获取情况

```
[admin@MikroTik] /ip route> print
Flags: X - disabled, A - active, D - dynamic, C - connect, S - static, r - rip, b - bgp, o - ospf,
m - mme, B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 0 ADS  0.0.0.0/0                          10.1.101.1         0
 1 ADS  160.0.0.0/24                       10.1.101.1         0
```

Option-set

RouterOS 的 DHCP-client 不支持 249，只支持 121，而 Windows XP 和 Windows 2003 仅支持 option 249， Windows vista、Windows 7 和 Windows 2008 对 option 249 和 option 121 都支持。如果在一个网络中即有支持 249，又支持 121 的主机或网络设备，需要设置复合型的 option 参数，这里可以利用 option-set 完成

```
/ip dhcp-server option
add code=121 name=classless121 value=0x18A000000A016501000A016501
add code=249 name=classless249 value=0x18A000000A016501000A016501
```

设置 option-set 参数，取名 set1

```
/ip dhcp-server option sets
add name=set1 options=classless121, classless249
```

设置 network 的 dhcp-option-set 属性

```
/ip dhcp-server network
set 0 dhcp-option-set=set1
```

7.3 Alerts 警报

操作路径: /ip dhcp-server alert

Alerts 功能是用于查找存在于当前网络中流氓 DHCP 服务器，Alerts 工具启用后会监控当前接口下所有 DHCP 响应信息，“流氓 DHCP 探测器”不会接收其他 DHCP 客户端的请求，流氓 DHCP 探测器会作为一个 DHCP 客户端，会每分钟发送 DHCP 探测请求。并检查 DHCP 服务器响应信息是否有效，如果一个响应来至未知服务器，报警将被触发。

```
[admin@MikroTik] ip dhcp-server alert>/log print
00:34:23 dhcp,critical,error,warning,info,debug dhcp alert on Public:
      discovered unknown dhcp server, mac 00:02:29:60:36:E7, ip 10.5.8.236
[admin@MikroTik] ip dhcp-server alert>
```

当系统发现流氓 DHCP 服务器发出报警，可以通过执行自定义脚本完成相应操作，可以设置 on-alert 设置脚本发送邮件提醒管理员等

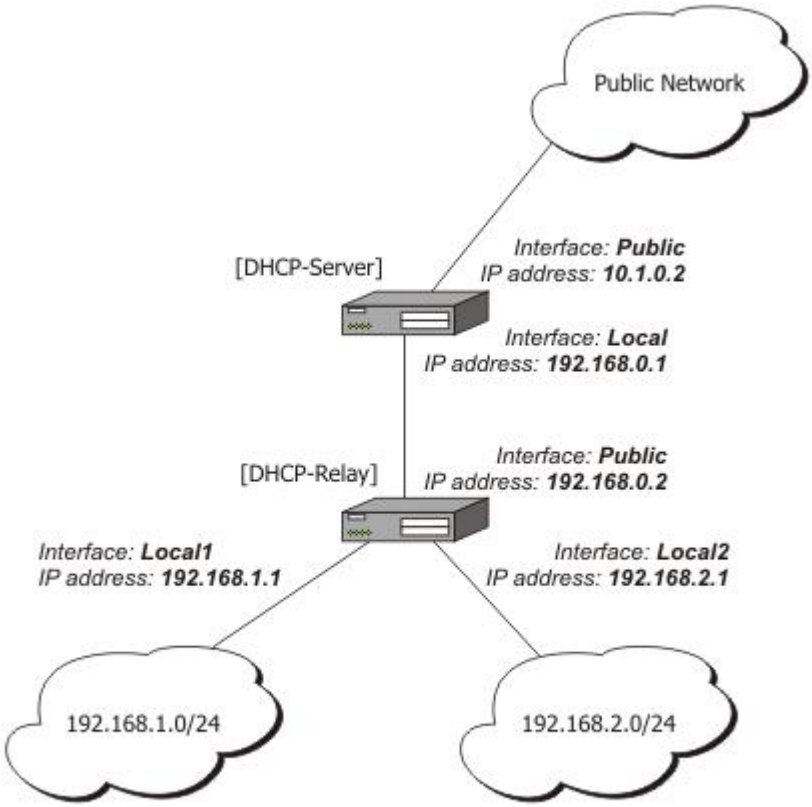
DHCP 响应单播方式，“流氓 DHCP 探测器”不会接收其他 DHCP 客户端的请求，为解决这个问题流氓 DHCP 探测器会作为一个 DHCP 客户端，会每分钟发送 DHCP 探测请求。

属性	描述
alert-timeout (*none* | *time*; 默认: **none**)	时间一过，警告将被遗忘。如果在此之后检测到局域网其他的 DHCP 服务器，将生成新的警报。如果设置为 none, timeout 将不会过期。
interface (*string*; 默认:)	允许探测流氓 DHCP 服务器的网络接口
on-alert (*string*; 默认:)	当探测到其他 DHCP 服务器触发的脚本
valid-server (*string*; 默认:)	列出流氓 DHCP 服务器的 MAC 地址

7.4 DHCP-Relay 配置

让我们考虑，你有几个 IP 网络在另外一个路由器的网络内，但你想保持所有的 DHCP 服务都在一台路由器上，这样你需要建立 DHCP-Relay，即 DHCP 中继。

这个实例将显示如何配置 **DHCP-Server** 和 **DHCP-Relay**。这里有 2 个 IP 段 **192.168.1.0/24** 和 **192.168.2.0/24** 都在一个路由器后面。



DHCP-Server 的 IP 地址：

```
[admin@DHCP-Server] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #    ADDRESS            NETWORK         BROADCAST       INTERFACE
 0    192.168.0.1/24     192.168.0.0     192.168.0.255   To-DHCP-Relay
 1    10.1.0.2/24  10.1.0.0  10.1.0.255 Public
[admin@DHCP-Server] ip address>
```

配置 Relay 的 IP 地址：

```
[admin@DHCP-Relay] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #    ADDRESS            NETWORK         BROADCAST       INTERFACE
 0    192.168.0.1/24     192.168.0.0     192.168.0.255   To-DHCP-Server
 1    192.168.1.1/24     192.168.1.0     192.168.1.255   Local1
 2    192.168.2.1/24     192.168.2.0     192.168.2.255   Local2
[admin@DHCP-Relay] ip address>
```

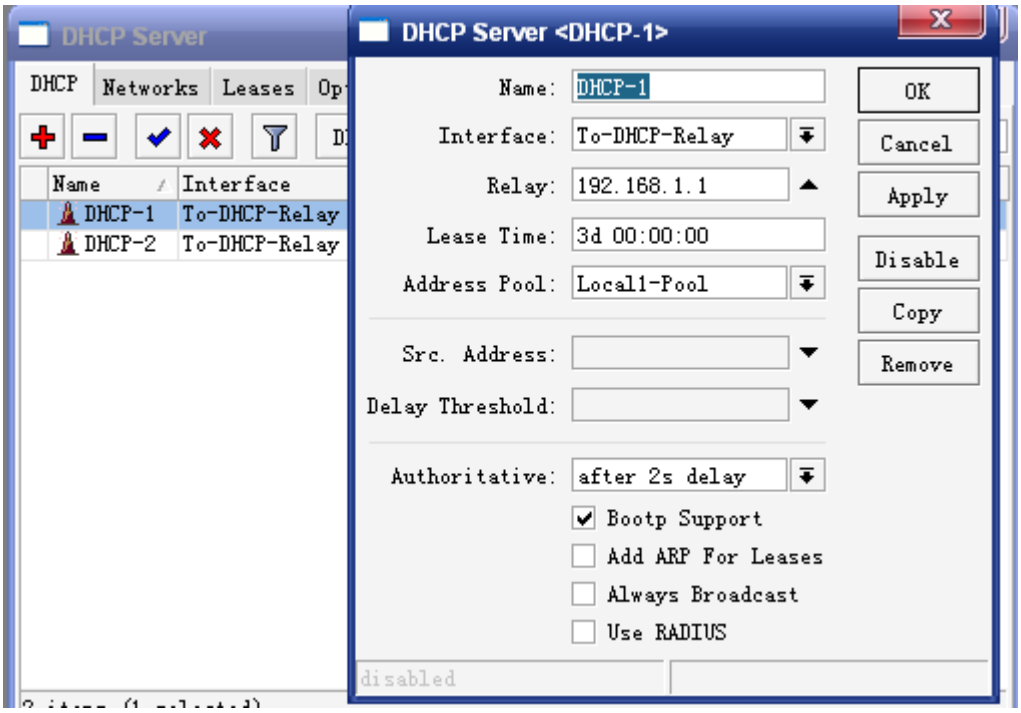
设置两个 DHCP 服务器在 **DHCP-Server** 的路由器上，这里分别添加两个 IP 地址段 **192.168.1.0/24** 和 **192.168.2.0/24**，如下：

```
/ip pool add name=Local1-Pool ranges=192.168.1.11-192.168.1.100
/ip pool add name=Local1-Poo2 ranges=192.168.2.11-192.168.2.100
```

```
[admin@DHCP-Server] ip pool> print
 # NAME                                          RANGES
 0 Local1-Pool                                   192.168.1.11-192.168.1.100
 1 Local2-Pool                                   192.168.2.11-192.168.2.100
[admin@DHCP-Server] ip pool>
```

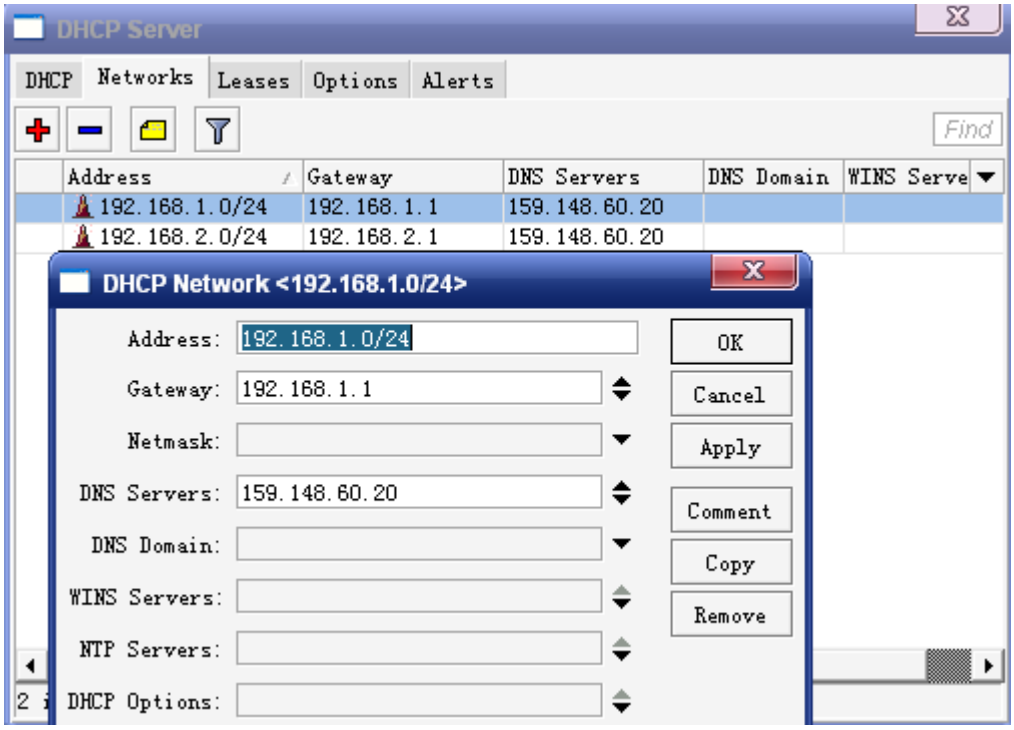
创建 DHCP 服务：

```
/ip dhcp-server add interface=To-DHCP-Relay relay=192.168.1.1 \
    address-pool=Local1-Pool name=DHCP-1 disabled=no
/ip dhcp-server add interface=To-DHCP-Relay relay=192.168.2.1 \
    address-pool=Local2-Pool name=DHCP-2 disabled=no
[admin@DHCP-Server] ip dhcp-server> print
Flags: X - disabled, I - invalid
 #    NAME            INTERFACE          RELAY           ADDRESS-POOL LEASE-TIME ADD-ARP
 0    DHCP-1          To-DHCP-Relay 192.168.1.1      Local1-Pool    3d00:00:00
 1    DHCP-2          To-DHCP-Relay 192.168.2.1      Local2-Pool    3d00:00:00
[admin@DHCP-Server] ip dhcp-server>
```



配置各自的网络：

```
/ip dhcp-server network add address=192.168.1.0/24 gateway=192.168.1.1 \
    dns-server=159.148.60.20
/ip dhcp-server network add address=192.168.2.0/24 gateway=192.168.2.1 \
    dns-server 159.148.60.20
[admin@DHCP-Server] ip dhcp-server network> print
 # ADDRESS            GATEWAY         DNS-SERVER      WINS-SERVER     DOMAIN
 0 192.168.1.0/24     192.168.1.1     159.148.60.20
 1 192.168.2.0/24     192.168.2.1     159.148.60.20
[admin@DHCP-Server] ip dhcp-server network>
```



DHCP-Server 路由器配置完成，现在配置 **DHCP-Relay** 路由器：

```
/ip dhcp-relay add name=Local1-Relay interface=Local1 \
    dhcp-server=192.168.0.1 local-address=192.168.1.1 disabled=no
/ip dhcp-relay add name=Local2-Relay interface=Local2 \
    dhcp-server=192.168.0.1 local-address=192.168.2.1 disabled=no
[admin@DHCP-Relay] ip dhcp-relay> print
Flags: X - disabled, I - invalid
 #    NAME                      INTERFACE         DHCP-SERVER       LOCAL-ADDRESS
 0    Local1-Relay              Local1            192.168.0.1       192.168.1.1
 1    Local2-Relay              Local2            192.168.0.1       192.168.2.1
[admin@DHCP-Relay] ip dhcp-relay>
```

7.5 Vendor Classes（厂商 ID 分类）

从 v6.45beta6 开始，RouterOS 的 DHCP Server 支持对 vendor class id 进行配匹，可通过 DHCP 客户端的厂商 ID 选择指定的地址池。下面的实例，基于三星 Android 手机提供一个指定 IP 地址池。

首先，路由器已经有 192.168.88.1/24 的 IP 配置到 bridge 接口，需区分不同厂商设备 class id，创建一个新的 IP 地址段，在/ip addres 下为 bridge 接口添加一个 IP 地址 172.16.16.1/24

```
/ip address
add address=172.16.16.1/24 interface=bridge
```

创建普通客户端 pool 的地址池

```
/ip pool
add name=default-dhcp ranges=192.168.88.10-192.168.88.254
```

新建一个 pool-for-VID 地址池，分配 IP 地址 172.16.16.10-172.16.16.120

```
/ip pool
add name=pool-for-VID ranges=172.16.16.10-172.16.16.120
```

配置普通客户端的 DHCP 规则

```
/ip dhcp-server
add address-pool=default-dhcp disabled=no interface=bridge name=defconf
/ip dhcp-server network
add address=192.168.88.0/24 comment=defconf gateway=192.168.88.1 dns-server=192.168.88.1
```

根据三星手机 vendor-class-id ，创建一个新的规则配匹 vid=android-dhcp-9，并调用 pool-for-VID 地址池

```
/ip dhcp-server vendor-class-id
add address-pool=pool-for-VID name=samsung server=defconf vid=android-dhcp-9
```

为三星手机创建新的 network 策略

```
/ip dhcp-server network
add address=172.16.16.0/24 comment=defconf gateway=172.16.16.1 dns-server=172.16.16.1
```

当三星手机连接到 DHCP 后，会从 172.16.16.0 网段中分配 IP

```
[admin@mikrotik] > /ip dhcp-server lease print detail
Flags: X - disabled, R - radius, D - dynamic, B - blocked
 0 D address=172.16.16.120 mac-address=30:07:4D:F5:07:49 client-id="1:30:7:4d:f5:7:49" address-lists=""
server=defconf dhcp-option=""
      status=bound expires-after=8m55s last-seen=1m5s active-address=172.16.16.120
active-mac-address=30:07:4D:F5:07:49
      active-client-id="1:30:7:4d:f5:7:49" active-server=defconf host-name="Galaxy-S8"
```

如果你不知道各个厂商设备的 Vendor Class ID，可以开启 DHCP debug，在日志中查看，创建 DHCP debug 日志配置如下命令：

```
/system logging add topics=dhcp
```

这是当客户端获取 DHCP 服务器 IP 地址时，会在/logs 显示信息，通过查看 **Class-ID** 得到 Vendor Class ID

```
10:30:31 dhcp,debug,packet defconf received request with id 4238230732 from 0.0.0.0
10:30:31 dhcp,debug,packet      secs = 3
10:30:31 dhcp,debug,packet      ciaddr = 0.0.0.0
10:30:31 dhcp,debug,packet      chaddr = 30:07:4D:F5:07:49
10:30:31 dhcp,debug,packet      Msg-Type = request
10:30:31 dhcp,debug,packet      Client-Id = 01-30-07-4D-F5-07-49
10:30:31 dhcp,debug,packet      Address-Request = 172.16.16.120
10:30:31 dhcp,debug,packet      Server-Id = 192.168.88.1
10:30:31 dhcp,debug,packet      Max-DHCP-Message-Size = 1500
```

```
10:30:31 dhcp,debug,packet      Class-Id = "android-dhcp-9"
10:30:31 dhcp,debug,packet      Host-Name = "Galaxy-S8"
```

以上是官方三星 Galaxy-S8 手机的 class-Id 为 **android-dhcp-9**

下面是我从 log 中获取华为 mate30 的 class-id

```
10:43:14 dhcp,debug,packet tsecs = 4
10:43:14 dhcp,debug,packet tciaddr = 0.0.0.0
10:43:14 dhcp,debug,packet tMsg-Type = request
10:43:14 dhcp,debug,packet tAddress-Request = 192.168.80.19
10:43:14 dhcp,debug,packet tServer-Id = 192.168.80.1
10:43:14 dhcp,debug,packet tMax-DHCP-Message-Size = 1500
10:43:14 dhcp,debug,packet tClass-Id = "HUAWEI:android:TAS"
10:43:14 dhcp,debug,packet tHost-Name = "HUAWEI_Mate_30_5G-7203a0c"
```

7.6 Rate limit（流量控制）

DHCPv4 server 支持在 Lease（租约）栏下添加静态规则时配置 IPv4 流量控制属性，将为 DHCPv4 Lease 会将对应的 IPv4 地址添加为一个动态的 simple queue 规则。这样做使得客户端在获取 IP 后，会自动创建一条动态的流量控制规则，客户端下线后流控规则会自动删除。

注意：特别提醒，如果防火墙开启了 FastTracked，流控规则将失效，请确保您的防火墙没有启用 FastTracked

首先，进入 DHCP Server 的 lease，查看当前获取 IP 的主机情况

```
[admin@MikroTik] > /ip dhcp-server lease print
Flags: X - disabled, R - radius, D - dynamic, B - blocked
 #     ADDRESS         MAC-ADDRESS       HOST-NAME        SERVER        RATE-LIMIT      STATUS
 0 D 192.168.88.254    6C:3B:6B:7C:41:3E       MikroTik         DHCPv4_Server                 bound
```

通过 make-static 命令，将主机 192.168.88.254 设置为静态规则

```
[admin@MikroTik] > /ip dhcp-server lease make-static 0
[admin@MikroTik] > /ip dhcp-server lease print
Flags: X - disabled, R - radius, D - dynamic, B - blocked
 #     ADDRESS         MAC-ADDRESS       HOST-NAME        SERVER          RATE-LIMIT      STATUS
 0    192.168.88.254   6C:3B:6B:7C:41:3E    MikroTik           DHCPv4_Server                  bound
```

然后修改，通过 set 命令，设置这条 Lease 规则的 rate-limit 带宽为 10M/10M（上行 10m，下行 10m）

```
[admin@MikroTik] > /ip dhcp-server lease set 0 rate-limit=10M/10M
```

查看/queue simple 下是否动态添加了流控规则

```
[admin@MikroTik] > /queue simple print
Flags: X - disabled, I - invalid, D - dynamic
```

```
 0   D name="dhcp-ds<6C:3B:6B:7C:41:3E>" target=192.168.88.254/32 parent=none packet-marks=""
priority=8/8 queue=default-small/default-small limit-at=10M/10M max-limit=10M/10M burst-limit=0/0
burst-threshold=0/0 burst-time=0s/0s
        bucket-size=0.1/0.1
```

缺省情况下，allow-dual-stack-queue 是启用的，可以将 DCHPv6 和 DHCPv4 的 Lease 规则添加到单个 simple queue 规则里，没有启用此选项，将为 IPv6 和 IPv4 添加单独的 simple queue 规则。

如果 *allow-dual-stack-queue* 启用，IPv4 和 IPv6 地址将添加到一条 simple queue 规则中：

```
[admin@MikroTik] > /queue simple print
Flags: X - disabled, I - invalid, D - dynamic
 0   D name="dhcp-ds<6C:3B:6B:7C:41:3E>" target=192.168.88.254/32,fdb4:4de7:a3f8:418c::/66
parent=none packet-marks="" priority=8/8 queue=default-small/default-small limit-at=10M/10M
max-limit=10M/10M burst-limit=0/0 burst-threshold=0/0
        burst-time=0s/0s bucket-size=0.1/0.1
```

第八章 DNS

DNS 是域名系统（Domain Name System）的缩写，是因特网的一项核心服务，它作为可以将域名和 IP 地址相互映射的一个分布式数据库，能够使人更方便的访问互联网，而不用去记住能够被机器直接读取的 IP 地址数字串。

需要功能包：**system**
需要等级：**Level1**
操作路径：**/ip dns**

8.1 DNS 配置

属性描述

属性	描述
allow-remote-requests (*yes | no; 默认: no*)	指定是否启用 DNS 服务，响应用户端的 DNS 请求
cache-max-ttl (*时间; 默认: 1w*)	指定缓存记录的最大存活时间，即 cache-max-ttl 时间到后记录将不再保留。对于 DNS 服务器推荐使用较短的 TTL 设置
cache-size (*整型: 512..10240; 默认: 2048KiB*)	指定 DNS 缓存大小，单位 KiB
cache-used (*read-only: 整型*)	显示当前缓存使用大小，单位 KiB
servers (*IPv4/IPv6 地址; 默认: 0.0.0.0*)	通过逗号间隔设置 DNS 服务器 IP

注意：

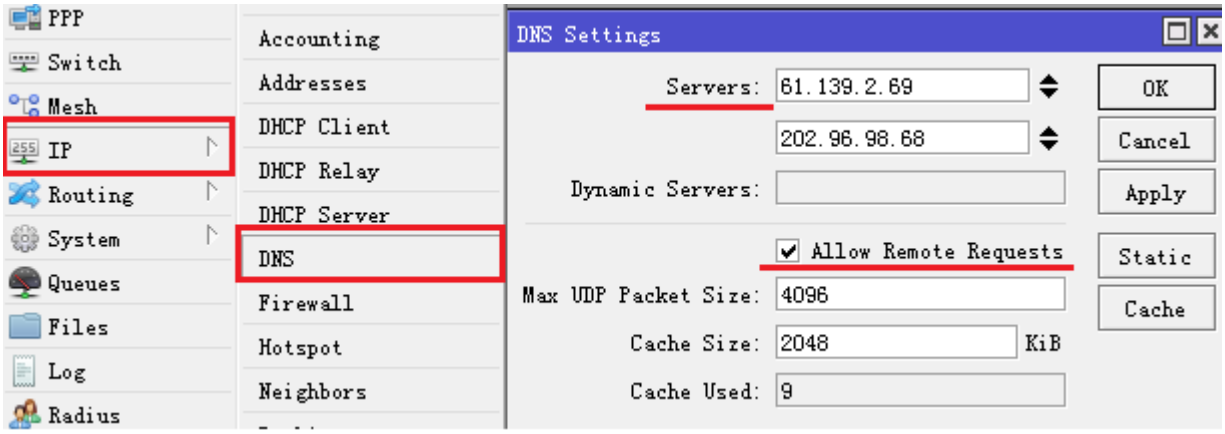
- 早先的 RouterOS v4.6 版本 DNS 服务器 IP 配置，在 CLI 下使用 primary-dns 和 secondary-dns 字段，在此之后的版本被 servers 字段替代，即所有 DNS 服务器 IP 地址配置，都通过该字段列表设置
- 如果在/ip dhcp-client、pppoe-client 或 pptp-client 使用属性 use-peer-dns ，即/ip dns 的 servers 字段将自动填写为对端服务器 DNS 地址
- 如果 **allow-remote-requests** 启用，确保 DNS 的 TCP 和 UDP 53 端口不受限制，能正常访问路由器做 DNS 解析。

事例：设置首选 DNS 服务器为 61.139.2.69 和 218.6.200.139：

```
[admin@MikroTik] /ip dns> print
                  servers:
 allow-remote-requests: yes
    max-udp-packet-size: 512
             cache-size: 2048KiB
          cache-max-ttl: 1w
             cache-used: 40KiB
[admin@MikroTik] ip dns> set servers 61.139.2.69,218.6.200.139
[admin@MikroTik] ip dns> print
```

```
                servers: 61.139.2.69,218.6.200.139
  allow-remote-requests: yes
    max-udp-packet-size: 512
            cache-size: 2048KiB
        cache-max-ttl: 1w
            cache-used: 40KiB

[admin@MikroTik] ip dns>
```

在 4.6 版本后增加了多个 DNS 服务器的支持，操作如下：



这里的 allow remote requests 为启用 DNS 缓存功能，cashe size 设定缓存存储大小

8.2 Cache 监测

操作路径: /ip dns cache

该菜单下主要提供 DNS 解析的记录和缓存信息

属性描述

属性	描述
address (只读: IP 地址)	主机的 IP 地址
name (只读: 名称)	主机的 DNS 域名
ttl (只读: 时间)	记录域名缓存剩下的存活时间

操作命令： **/ip dns cache all**

该命令是显示完整的 DNS 记录和缓存信息

属性描述

属性	描述
data (只读: 文本)	DNS data 字段。Type A 记录的 IP 地址，其他记录类型有不同的数据字段内容。

name (只读: 名称)	主机的 DNS 域名
ttl (只读: 时间)	记录域名缓存剩下的存活时间
type (只读: 文本)	DNS 记录类型

可以通过 flush Cache 命令清空当前缓存

8.3 静态 DNS 域名解析

操作路径: ***/ip dns static***

MikroTik RouterOS 嵌入了 DNS 解析服务器功能，允许你设置 RouterOS 为一台 DNS 服务器，并能在 RouterOS 指定静态域名解析到指定的 IP 主机。此功能能提供“伪造”的 DNS 解析给网络用户，即做本地 A 记录的设置，例如部分企业需要在指定域名解析到本地服务器。

属性描述

属性	描述
address (IP 地址)	解析域名返回的 IP 地址
name (文本)	需要设置的 DNS 域名
regex (文本)	DNS 域名正则表达式
ttl (时间)	DNS 记录存活时间
type (time)	DNS 记录类型，可以获得以下值： A, AAAA, CNAME, FWD, MX, NS, NXDOMAIN, SRV, TXT

注意：

在正则表达式里“.”代表任意一个字符，因此需要避免与域名的点混淆。例如：你需要匹配任意前缀为“example.com”的域名，如 www.another-example.com ，使用配置 name=".*\\.example\\.com"。这里命令行使用“\\.”表示“.”而 winbox 中表达为“\.”

正则表达式匹配速度要比普通的文本记录慢，因此尽量减少正则表达式的规则或者优化表达式。

例如，下面添加一个静态 www.example.com 的 DNS A 记录，解析 IP 是 10.0.0.1

```
[admin@MikroTik] ip dns static> add name=www.example.com address=10.0.0.1
[admin@MikroTik] ip dns static> print
Flags: D - dynamic, X - disabled, R - regexp
 #       NAME                  ADDRESS                                  TTL
 0       www.example.com       10.0.0.1                                 1d
[admin@MikroTik] ip dns static>
```

下面是用正则表达式匹配方式添加：

```
[admin@MikroTik] ip dns static> add regexp="[*example*]" address=10.0.0.2
```

8.4 清除 DNS cache（缓存）

命令操作: /ip dns cache flush

命令属性

属性	描述
flush	清除内部 DNS 缓存

DNS 缓存在实际的工作环境中会遇到，外部 DNS 已经更新，而 RouterOS 的缓存仍然解析的是老旧信息，造成访问失败，因此如果出现 DNS 解析故障，可以尝试清楚 DNS 缓存。

```
[admin@MikroTik] ip dns> cache flush
[admin@MikroTik] ip dns> print
              servers: 61.139.2.69,218.6.200.139
  allow-remote-requests: yes
    max-udp-packet-size: 512
             cache-size: 2048KiB
          cache-max-ttl: 1w
             cache-used: 4KiB
[admin@MikroTik] ip dns>
```

8.5 DoH（DNS over HTTPS)

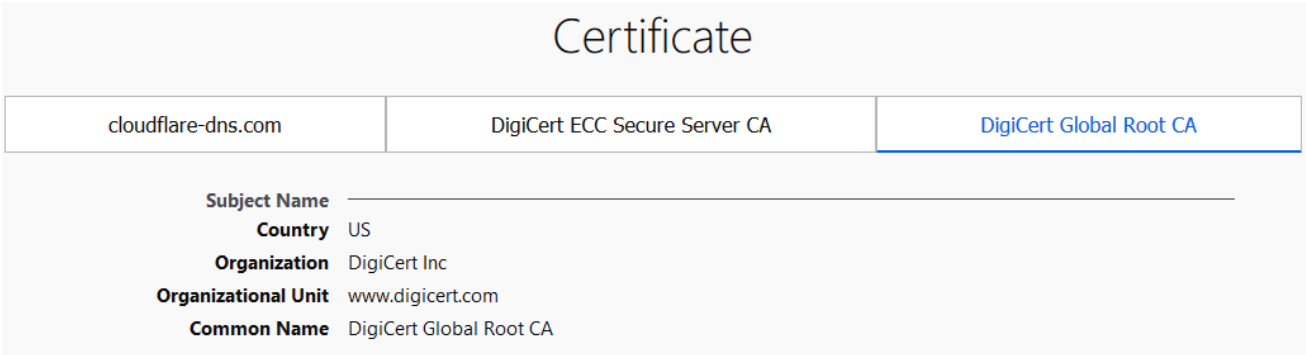
从 RouterOS v6.47 开始，可以在 HTTPS (DoH)上使用 DNS 服务。DoH 使用 HTTPS 协议发送和接收 DNS 请求以获得更好安全和数据完整性。它的主要目标是通过消除中间人攻击(MITM)来提供隐私保护。目前 DoH 不兼容 FWD 类型的静态项，如果要使用 FWD 功能，不能启用 DoH。

实例

首先要导入选择使用的 DoH 服务器的根 CA 证书，以提高安全性。

警告: 我们强烈建议不要使用第三方下载链接来获取证书。使用认证机构自己的网站。

有各种方法确定需要根 CA 证书。最简单的方法是使用 WEB 浏览器，导航到 DoH 站点并检查网站的安全性。使用 web 浏览器可以看到，CloudFlare DoH 服务器使用的是 DigiCert Global Root CA。您可以直接从浏览器下载证书，或者导航到 DigiCert 网站，通过 fetch 下载可信证书。



通告 fetch 下载并导入证书：

```
/tool fetch url="https://cacerts.digicert.com/DigiCertGlobalRootCA.crt.pem"
/certificate import file-name=DigiCertGlobalRootCA.crt.pem
```

配置 DoH 服务器：

```
/ip dns set use-doh-server=https://cloudflare-dns.com/dns-query verify-doh-cert=yes
```

请注意，至少需要为路由器配置一个常规 DNS 服务器来解析 DoH 主机名本身。如果你没有任何动态或静态 DNS 服务器，可以参考下面的配置：

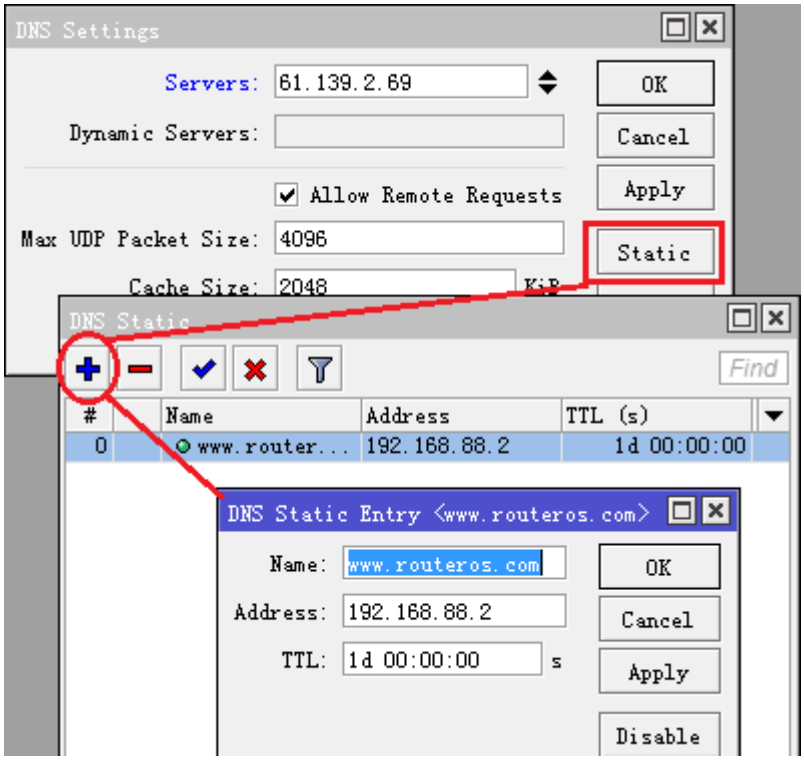
```
/ip dns set servers=1.1.1.1
```

8.6 DNS 劫持

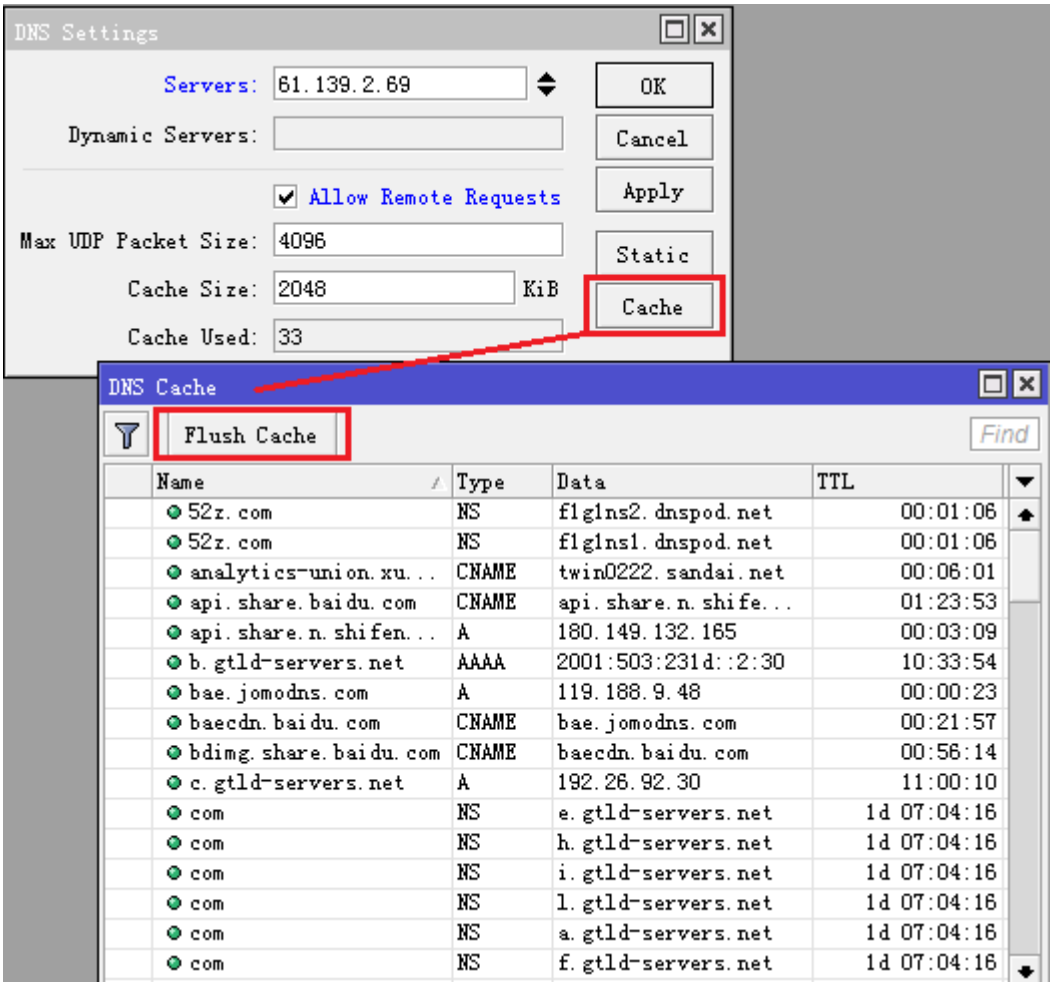
DNS 劫持是当用户请求某一网站时，将网站的 DNS 解析强制到指定的主机，即等同于 A 记录。注意：要实现 RouterOS 路由网关的域名 A 记录查询，必须满足：

1. 开启 DNS 缓存功能
2. 客户端设置 RouterOS 路由器 IP 为 DNS（RouterOS 上配置的任意 IP 地址，路由可达即可）
3. 如果不设置 RouterOS 上配置的 IP，就需要做 dst-nat 重定向，即将所有 TCP 或 UDP 的 53 端口重定向到 RouterOS 本地（具体操作可以参考 nat 章节的 dst-nat）

做 A 记录的作用是，将特定的域名解析指定到某一台服务器获取，例如企业网络的办公 OA、CRM 系统等通过域名访问，或者一些指定网站或页面的缓存等。下面是将所有请求 www.routeros.com 的 dns 请求都指定到 192.168.88.2 的主机上



添加完成后，我们需要打开 Cache 菜单，清空 DNS 缓存，避免遗留缓存无法让 A 记录生效，我们进入 Cache 菜单后，点击 Flush Cache 清空缓存：

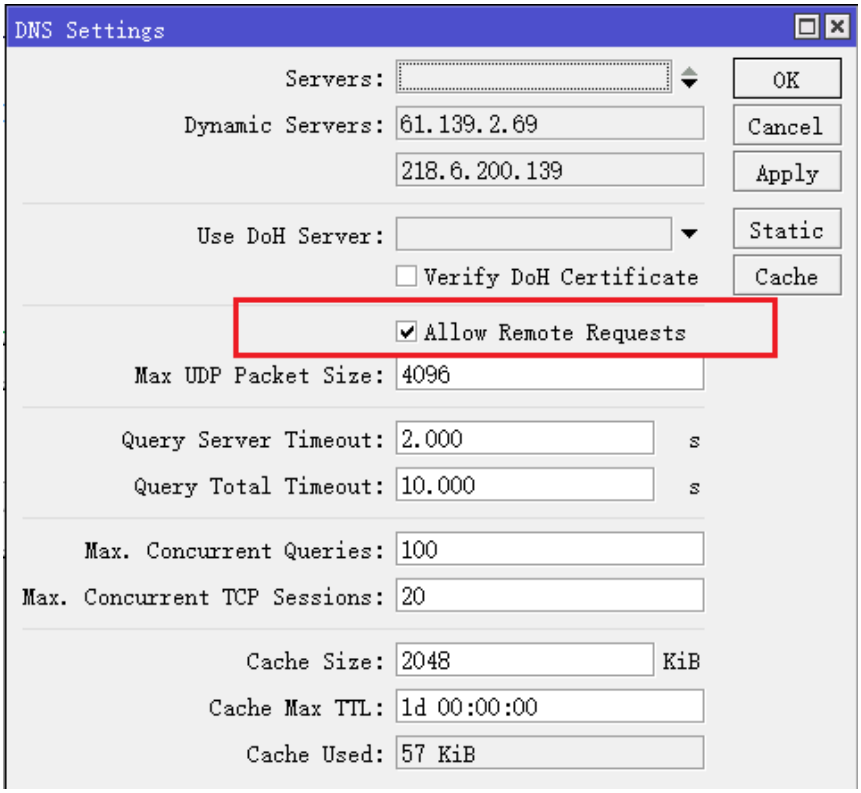


8.7 DNS forward 转发

RouterOS v6.47 后，DNS 静态条目规则中增加了 type 属性，增加了各种 DNS 记录参数包含：A, AAAA, CNAME, FWD, MX, NS, NXDOMAIN, SRV, TXT。A 记录在 RouterOS 早期就支持，特别是 FWD（forward 转发功能），可以将指定域名转发到指定的 DNS 服务器，对于多线路需求的用户来说非常有用，因为不同运营商的 DNS 不同，返回的服务器 IP 地址也不同，因此多线路调整不仅涉及网关路由，还涉及 DNS 解析。

用 www.taobao.com 做一个实例，RouterOS 默认 DNS 服务器为电信，转发 DNS 服务器用联通，电脑 DNS 服务器指向 RouterOS 的 192.168.88.1。

首先确保 DNS 配置，启用了本地缓存



解析 www.taobao.com 如下：

```
C:\Users\yus>nslookup www.taobao.com
服务器:  router.lan
Address:  192.168.88.1

非权威应答:
名称:    www.taobao.com.danuoyi.tbcache.com
Addresses:  240e:d9:a001:0:3::3fa
          240e:d9:a400:700:3::3fa
          240e:d9:c202:300:3::3fa
          240e:d9:a003:a00:3::3f9
          240e:d9:a001:0:3::3f9
          240e:d9:a400:700:3::3f9
          240e:d9:c202:300:3::3f9
          240e:d9:a003:a00:3::3fa
          182.140.139.108
          182.140.130.232
          182.140.143.232
          182.140.130.233
          118.112.252.234
          118.112.19.99
          118.123.2.243
Aliases:  www.taobao.com
```

当前返回的是电信 IP 地址

添加 www.taobao.com 的域名转发配置如下：

```
/ip dns static
add disabled=no forward-to=119.6.6.6 name=www.taobao.com type=FWD
```

配置完成后，需要执行一次 flush，清空缓存

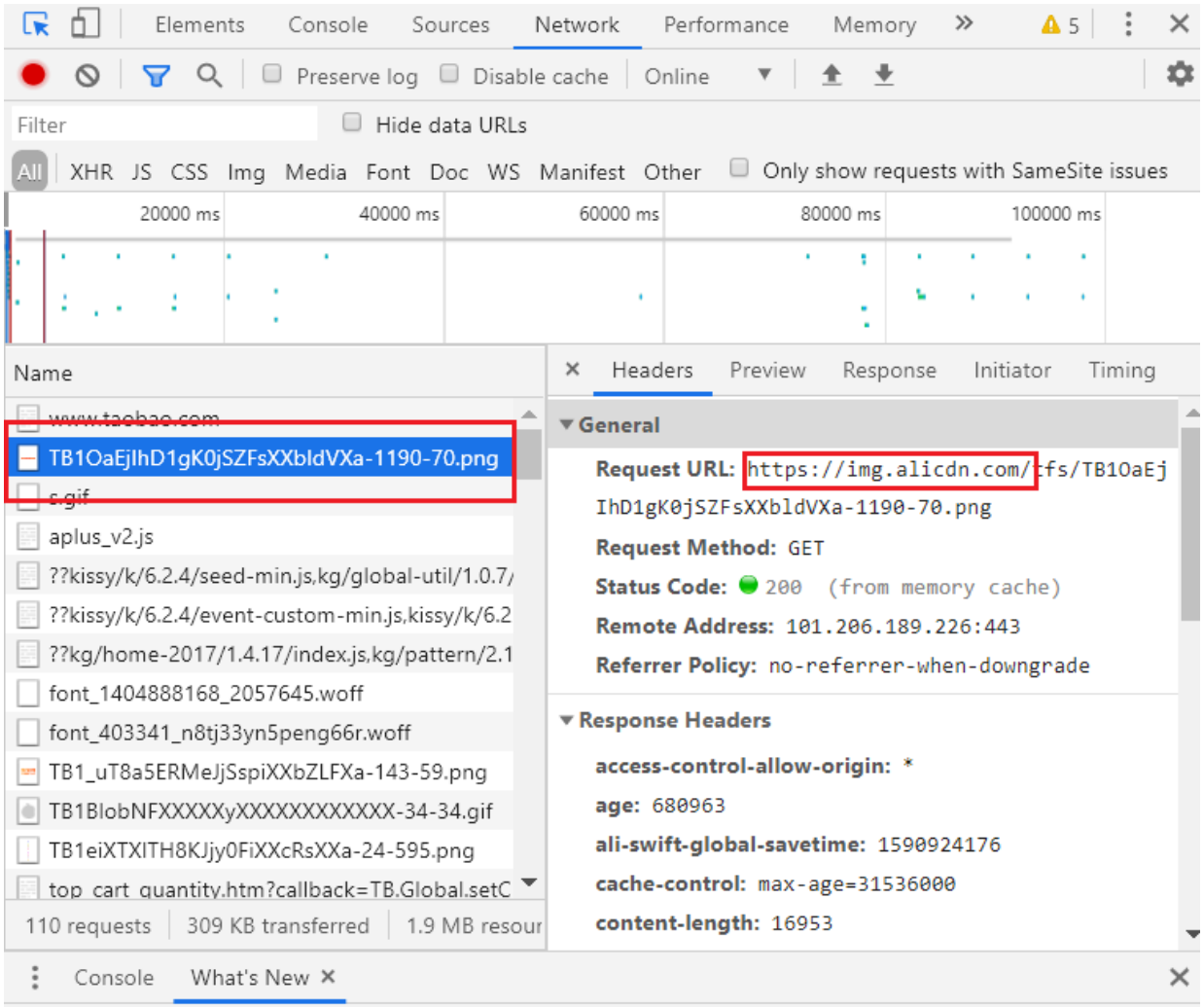
```
[admin@MikroTik] > /ip dns cache flush
```

配置完成后解析结果如下：

```
C:\Users\yus>nslookup www.taobao.com
服务器:  router.lan
Address:  192.168.88.1

非权威应答:
名称:    www.taobao.com.danuoyi.tbcache.com
Addresses:  2408:8766:0:1004:3::3f9
          2408:8766:0:1004:3::3fa
          101.207.252.145
          101.207.252.146
Aliases:  www.taobao.com
```

已经成功转发到联通 DNS，返回的是联通服务器 IP 地址

这个操作仅仅是转发了淘宝主站的一个域名，淘宝网的域名组成有很多个，不仅仅是这些，万里长征才迈出第一步，如果想把淘宝网站都转发到指定的 DNS 服务器，那么需要进行网站分析，可以通过 chrome 浏览器的 F12 开发者模式，打开淘宝页面，进行查看，会显示所有页面浏览记录



我们随便抓取一个 png 图片文件，他的域名是 img.alicdn.com，当然 windows cmd 下也可以查看

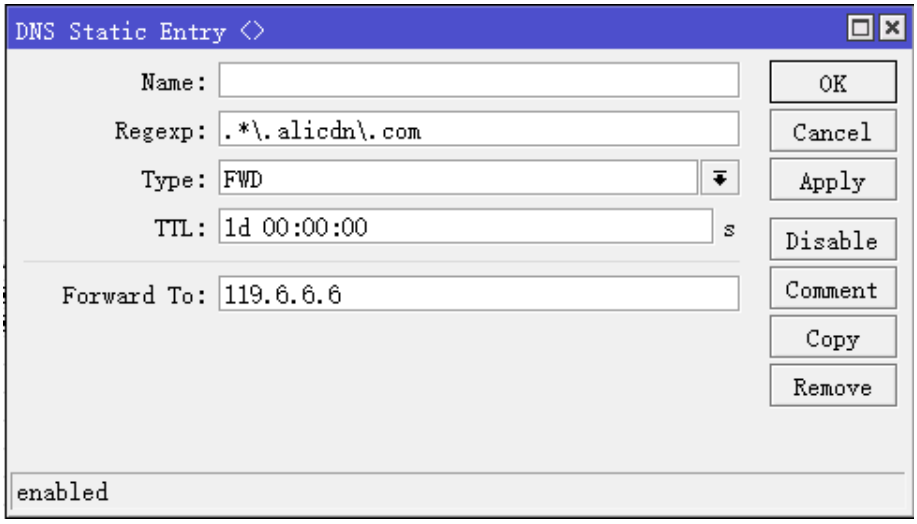
```
C:\Users\yus>nslookup img.alicdn.com
服务器:  router.lan
Address:  192.168.88.1

非权威应答:
名称:    img.alicdn.com.danuoyi.alicdn.com
Addresses:  182.140.139.124
          110.185.124.224
          118.123.2.229
          118.123.2.254
Aliases:  img.alicdn.com
```

因此我们要转发的是 img.alicdn.com，后面还有很多域名，都需要逐个分析，这个是告诉大家方法。

可以用则表达式去匹配主域名，如 img.alicdn.com，这个只是其中一个,那么我们可以配匹为

```
/ip dns static
 add regexp=".*\\.alicdn\\.com" forward-to=119.6.6.6 type=FWD
```

正则表达式的写法在命令行如上，涉及字符转译需要加"\\"，不要直接复制命令行的内容到 winbox 中，在 winbox 中写法如下：



如果对网站结构不太了解，只能逐步分析其域名的关联性，不能偷懒的。

对于转发还有一点需要补充，我使用的 BIND DNS 应用中 forward 有两个类型分别是：

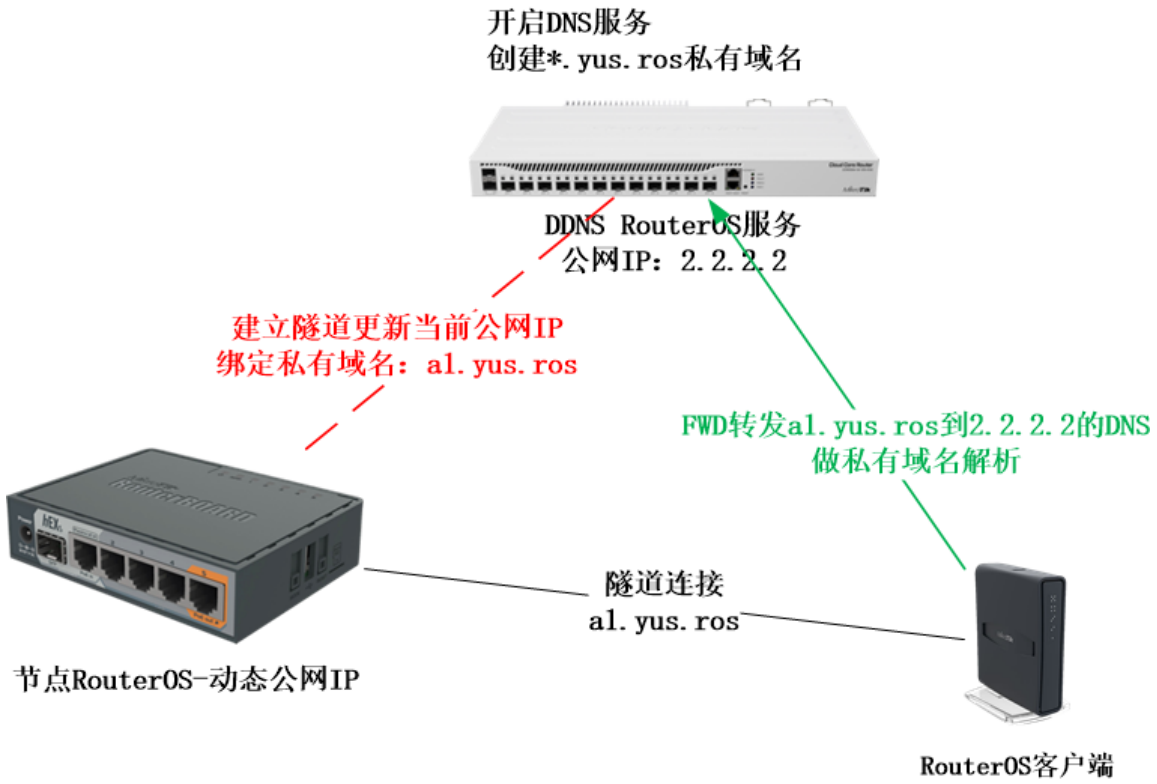
- forward frist：只转发一次，如果失败别则使用主 DNS 解析
- forward only，只做转发操作，及时失败也继续转发

我测试了 RouterOS 的 DNS 转发，类型是 forward only，也就是转发的 DNS 服务器失败了，也继续转发到该 DNS 服务器，会导致转发域名解析失败。这个你要记住，如果转发 DNS 服务器中断，要用脚本监测及时修补。

8.8 RouterOS 搭建私有 DDNS 服务

DNS 实际就是将域名和 IP 地址绑定，DDNS 就是动态 DNS，需实时自动的更新域名和 IP 的绑定关系，通常动态获取 IP 的主机需要实时的向 DDNS 服务器上报自己的 IP，如果 IP 变动，就更新该主机域名的 IP 地址。

按照这个思路，首先 DDNS 服务器，需要为绑定域名做 A 记录解析，RouterOS 的 DNS 是支持的是 static 下配置 A 记录的，那我们要求节点的 RouterOS 上报自己的 IP 地址，如果变动就修改 DNS 下的 A 记录 IP。下面是通过 RouterOS 完成一个私有域名的 DDNS 解析操作拓扑：



上面拓扑是节点 RouterOS 是动态公网 IP，客户端 RouterOS 需要通隧道连接到节点，假设在没有其他 DDNS 服务器的情况下，建立私有的 DDNS 域名解析，私有域名为 yus.ros，通过搭建一台 RouterOS 作为 DDNS 服务器。

在设计时，上报 IP 我最先想到的是通过 fetch 提交，因为 ddns 提交都用这个，但这个方式需要 RouterOS 打开 FTP 服务，并上传文本，感觉不安全，操作繁琐。要让 DDNS 知道节点 RouterOS 的 IP 地址，其实可以建立一个 pptp、l2tp 或者 sstp 隧道，在 DDNS RouterOS 上搭建隧道服务器，通过 caller-id 来判断节点 RouterOS 的 IP 是否变动，而且 ppp profile 有现成的登录脚本触发功能，caller-id 变了就去改 A 记录，每个用户在 ppp 下创建一个账号，并给账号加上 comment 注释记录 IP。

节点 RouterOS 和 DDNS RouterOS 之间沟通解决了，有人会说这个不是公共的 DNS 服务器，如何完成公网的域名解析，域名我采用的是私有域名，只能在这套 RouterOS 网络完成，客户端的 RouterOS 需要将私有域名转发到 DDNS RouterOS 上，仅仅需要通在 IP DNS 下配置 FWD 转发就 ok。

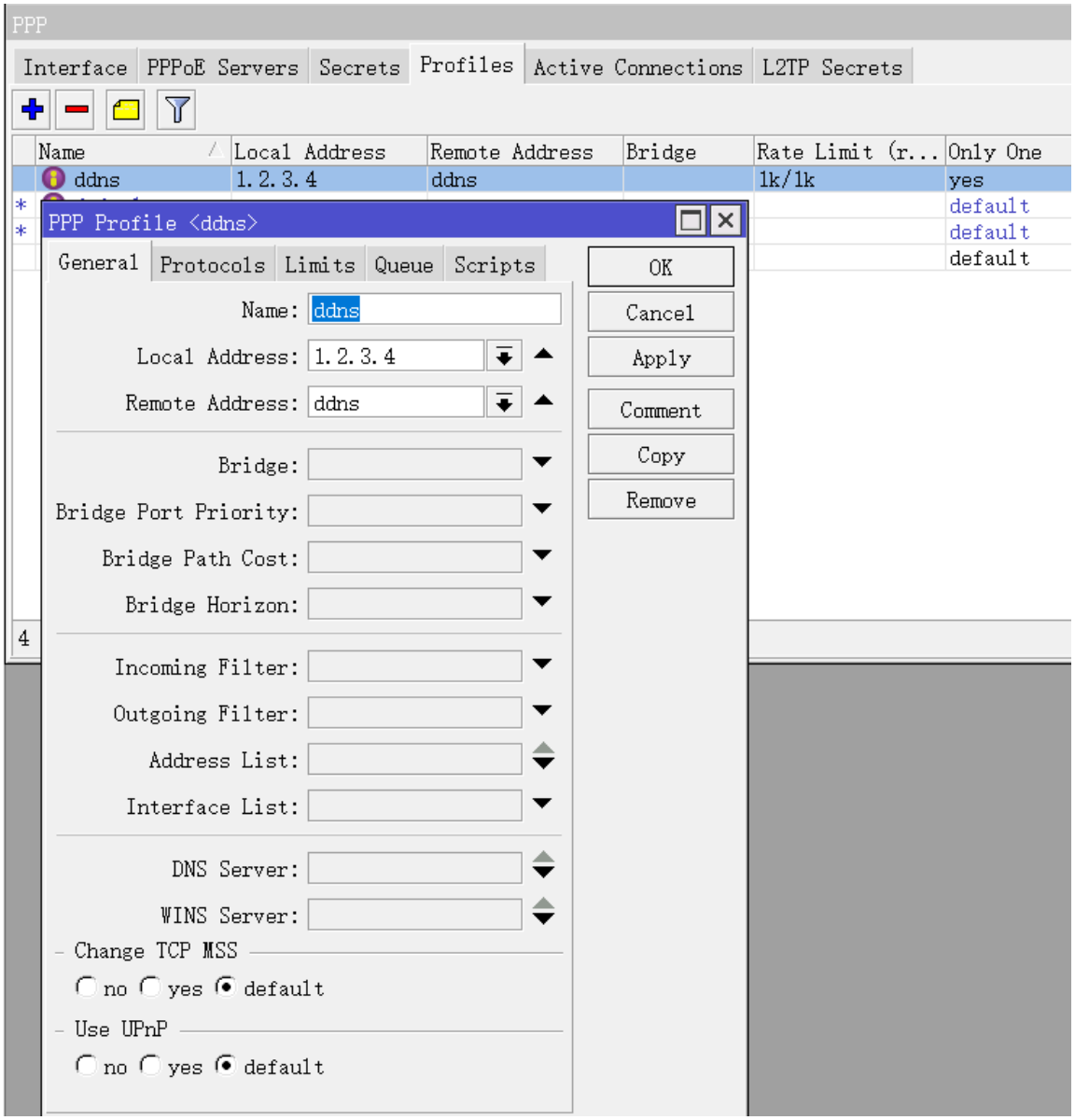
特别说明：

虽然有 MikroTik 官方的 cloud 功能，但有时候还是需要一套自己的解决方案，这套 DDNS 解析设计目的建立一个区域网络的私有服务上，仅仅是私有域名解析，非公共域名解析，请大家一定要明白是私有 DDNS，例如一个企业多个异地办公之间的 DDNS 服务，建立一个内部私有域名解析，也就是建立一个内部的 DDNS 服务，当然也可以继续扩展就看各位如何发挥。

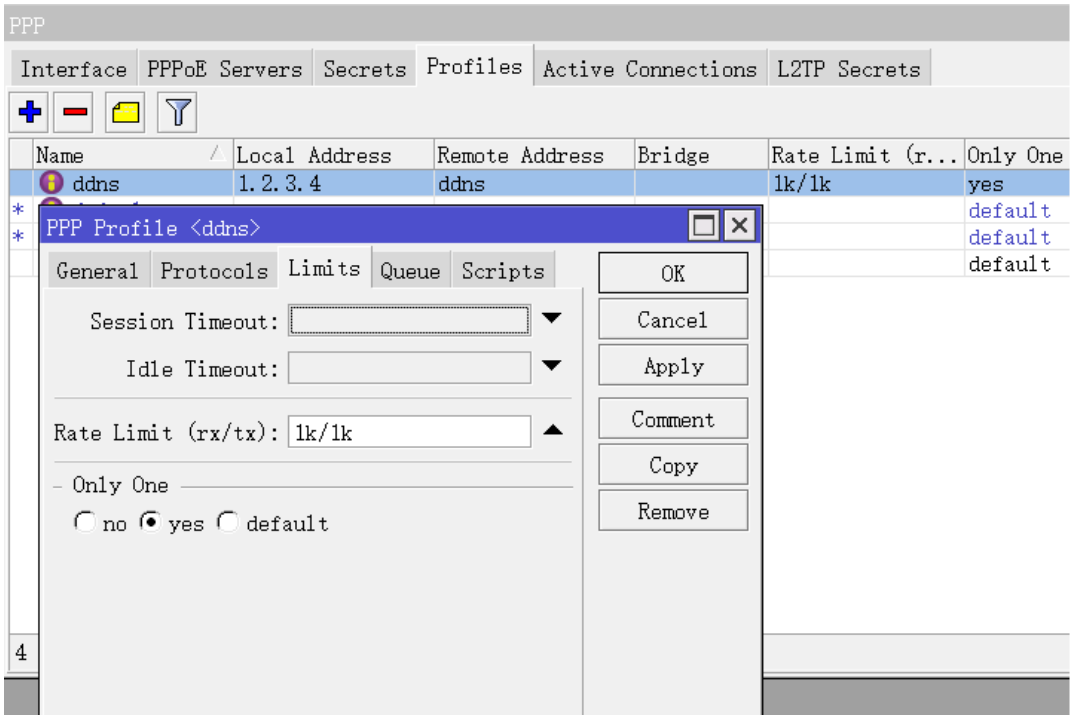
DDNS RouterOS 配置

作为 DDNS 服务的 RouterOS，首先需要一个固定的公网 IP 地址，建立隧道服务，这里我选择的是 SSTP，这个实例中 DDNS 的 IP 假设是 2.2.2.2，在实际应用中可以找一台 VPS。

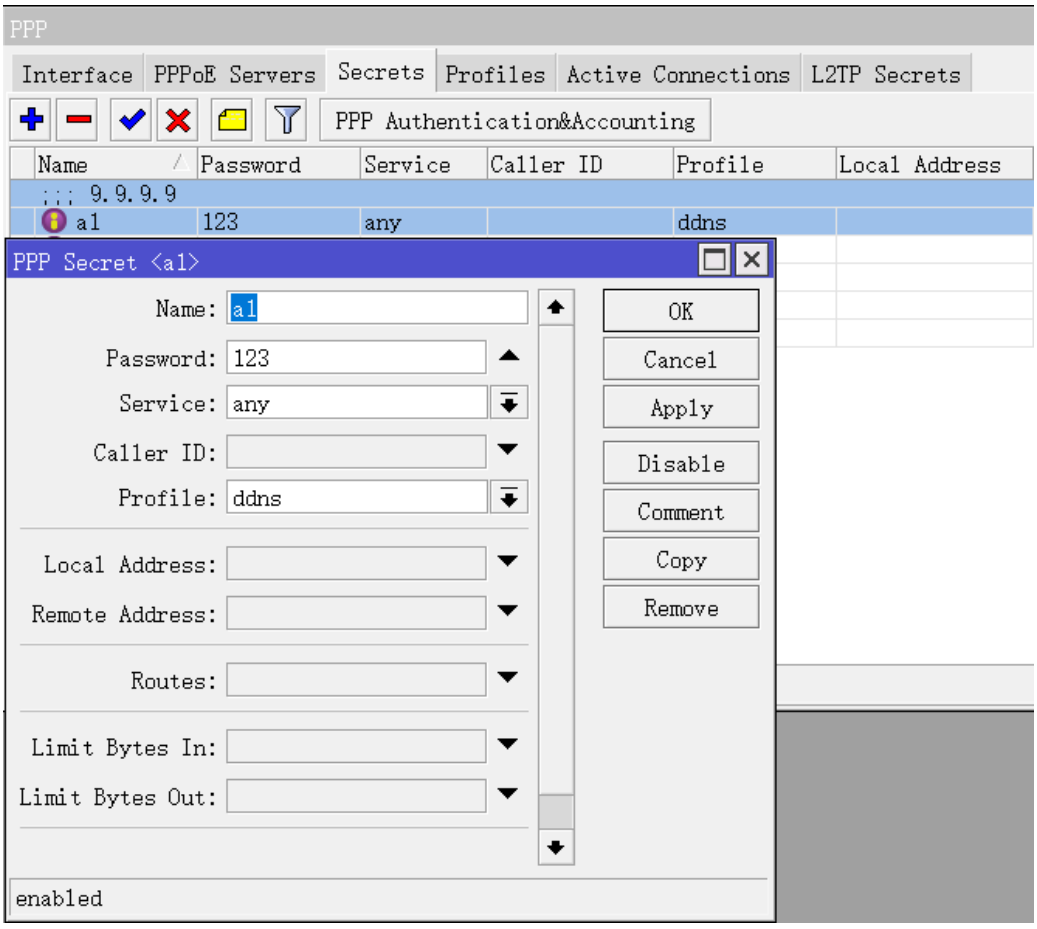
配置 sstp 服务器，用户客户端上报连接 IP，先创建一个 profile，local address 随便写，remote-address 也可以随便创建一个 pool 的地址池（/ip pool 下创建），能不能上网不是重要，只要连通到 RouterOS 就行，上报下对端的 IP。



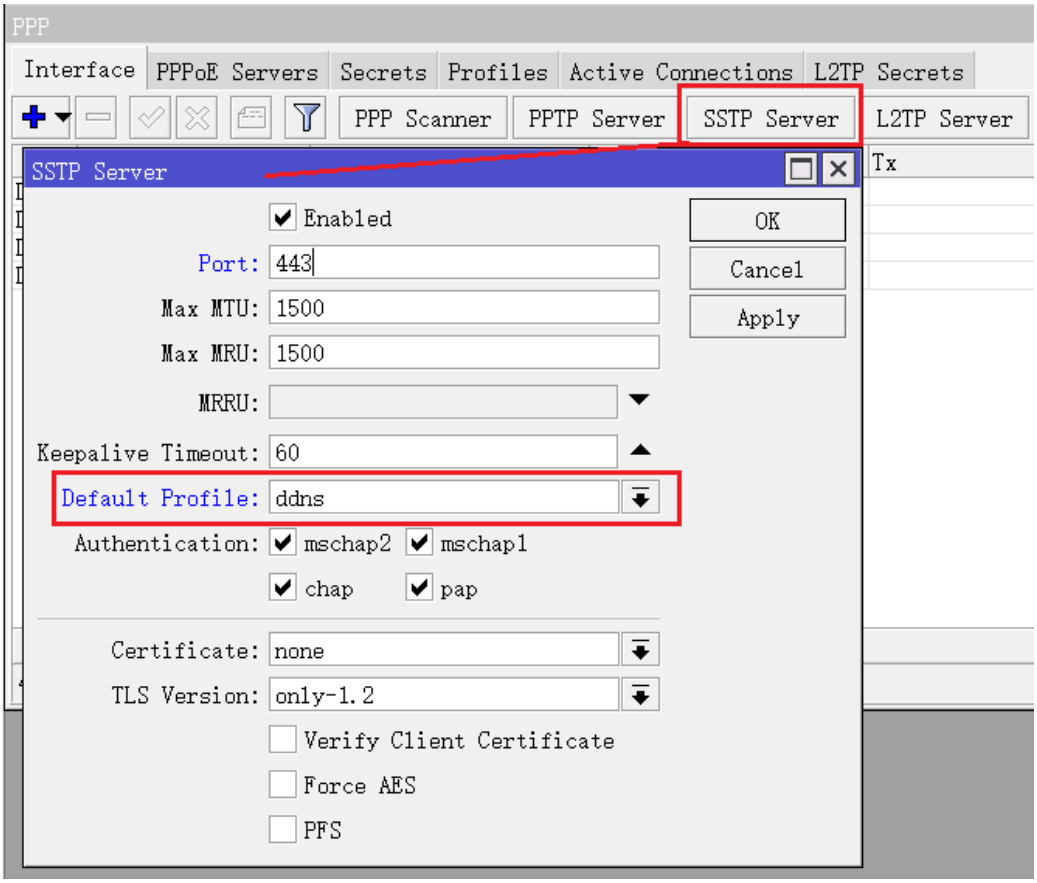
限制账号仅能同时登录一次和带宽



在 ppp secret 创建账号 a1（账号名称可以和后面使用到的域名对应），设置密码 123，并添加一个 comment 注释预设值 IP 地址 9.9.9.9：

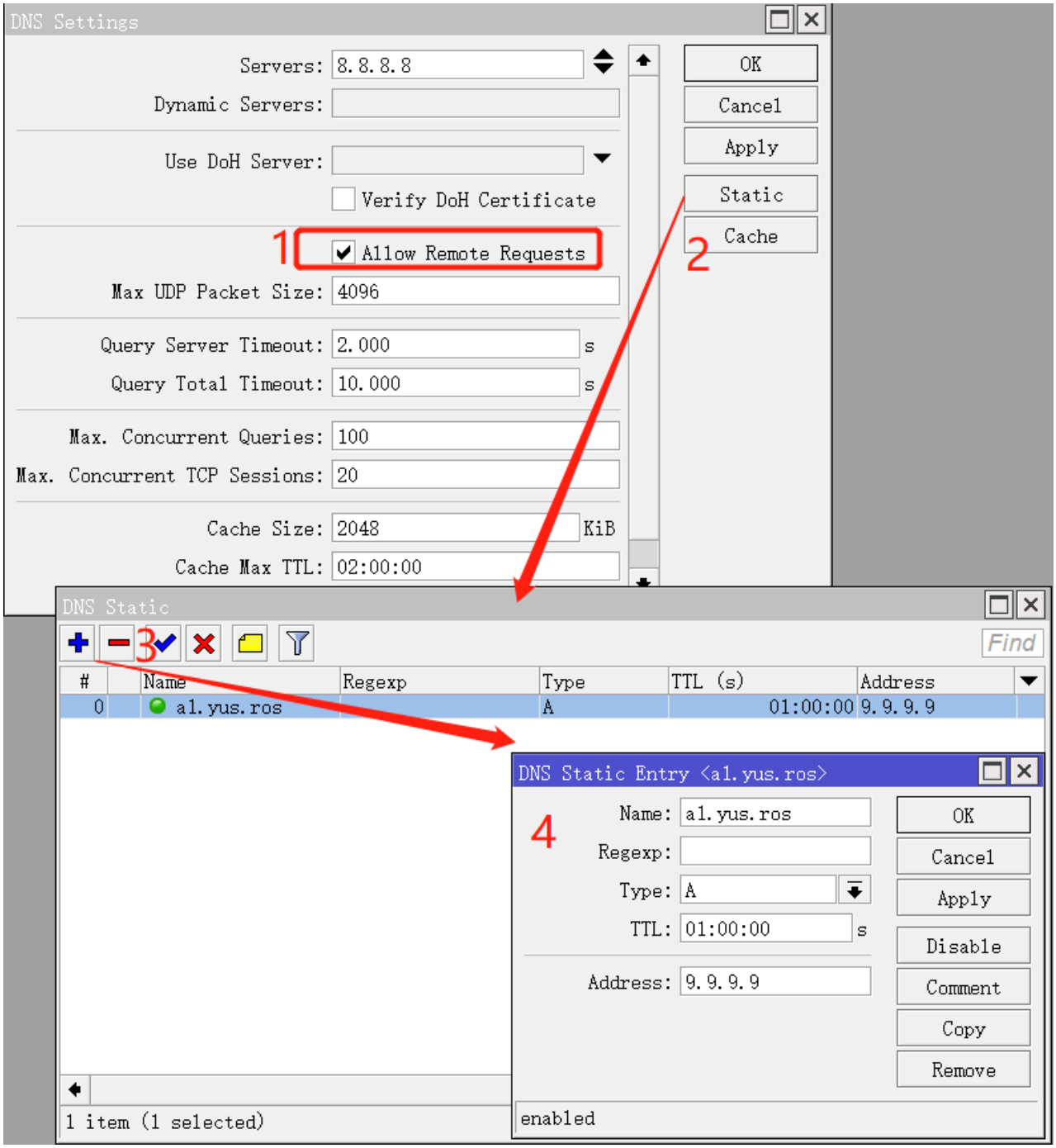


启用 SSTP 服务，当然可以选择其他隧道 PPTP、L2TP 等，根据自己网络环境配置

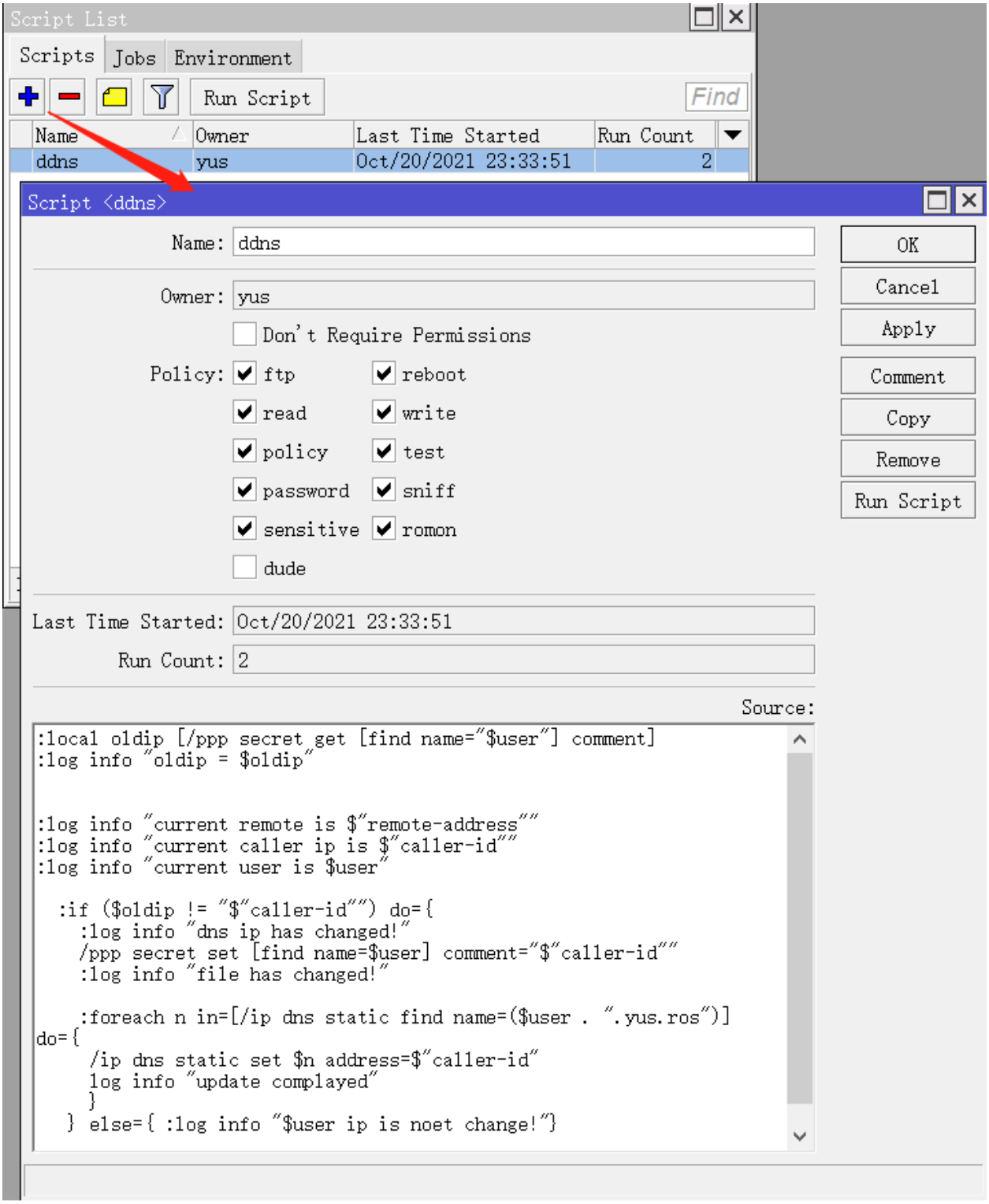


隧道服务搭建完成后，进入 IP DNS 配置 DNS 缓存和添加 A 记录

- 开启 allowed remote request
- 进入 static
- 点加号添加一个记录，与账号 a1 对应，设置 a1.yus.ros 的私有域名，如果有多个账号可以依次与域名对应。
- 设置 a1.yus.ros 的 A 记录 IP 地址为 9.9.9.9，TTL 值我修改为 1 小时，尽量缩短时间。



DNS 配置完成后，就是编写脚本，在 system script 下，创建一个 ddns 脚本

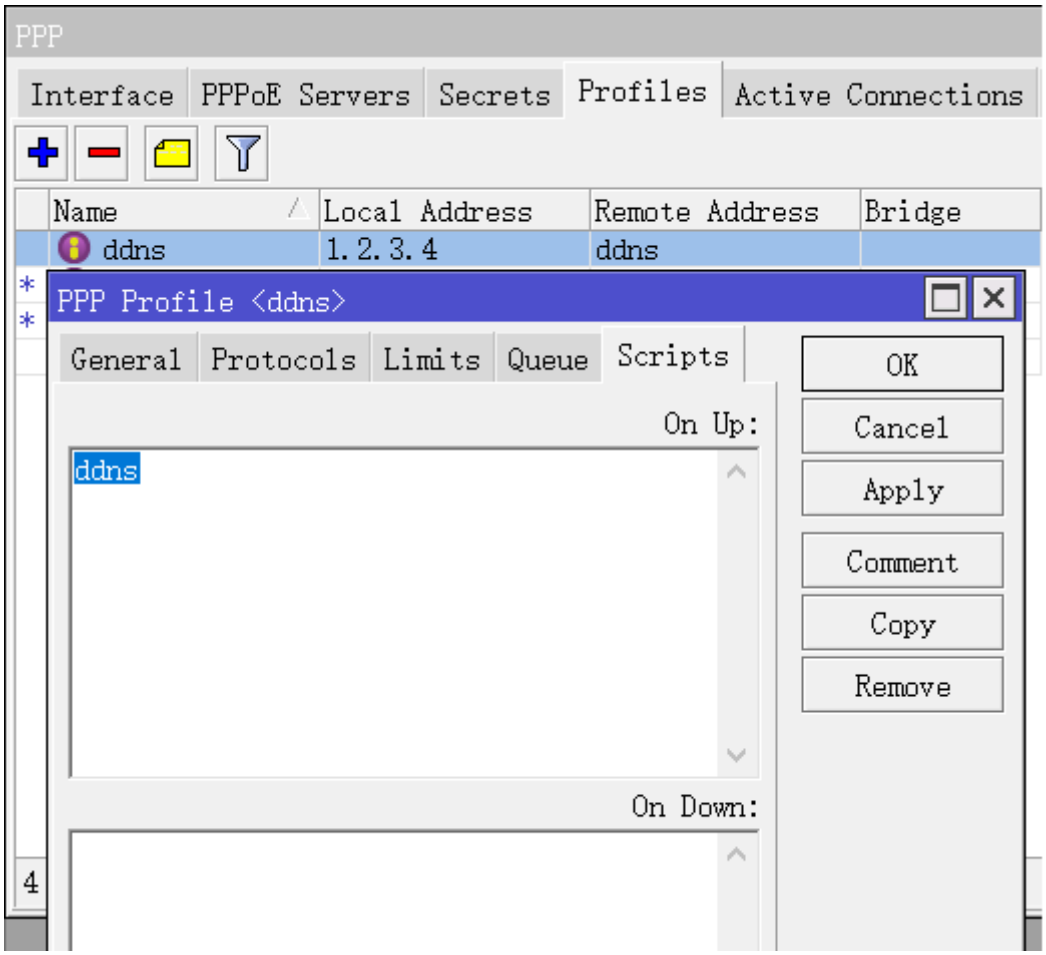


脚本如下：

```
:local oldip [/ppp secret get [find name="$user"] comment]
:log info "oldip = $oldip"

:log info "current remote is $"remote-address""
:log info "current caller ip is $"caller-id""
:log info "current user is $user"

:if ($oldip != "$"caller-id"") do={
```

```
:log info "dns ip has changed!"
/ppp secret set [find name=$user] comment="$"caller-id""
:log info "file has changed!"

:foreach n in=[/ip dns static find name=($user . ".yus.ros")] do={
/ip dns static set $n address=$"caller-id"
log info "IP updated"
}
} else={ :log info "$user's ip is not change!"}
```

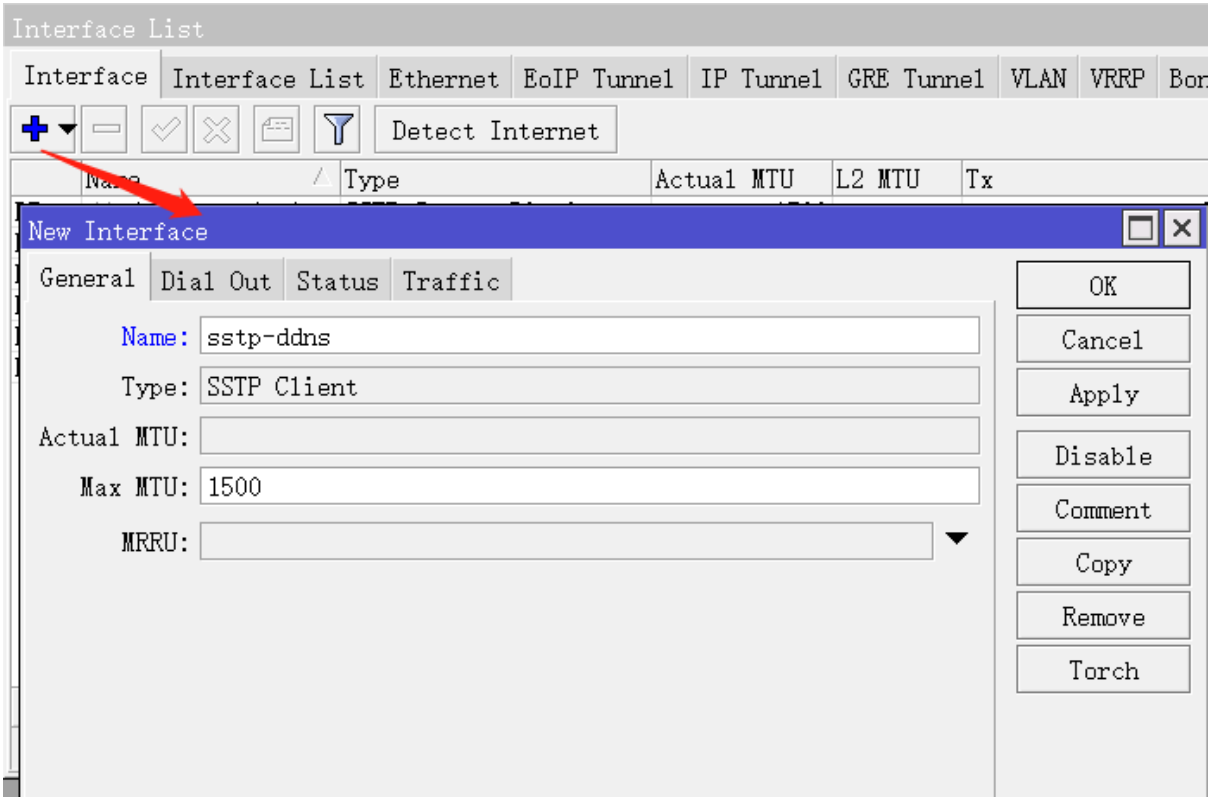
然后把这个脚本添加到 ppp profile 的 ddns 规则的 on-up 脚本下



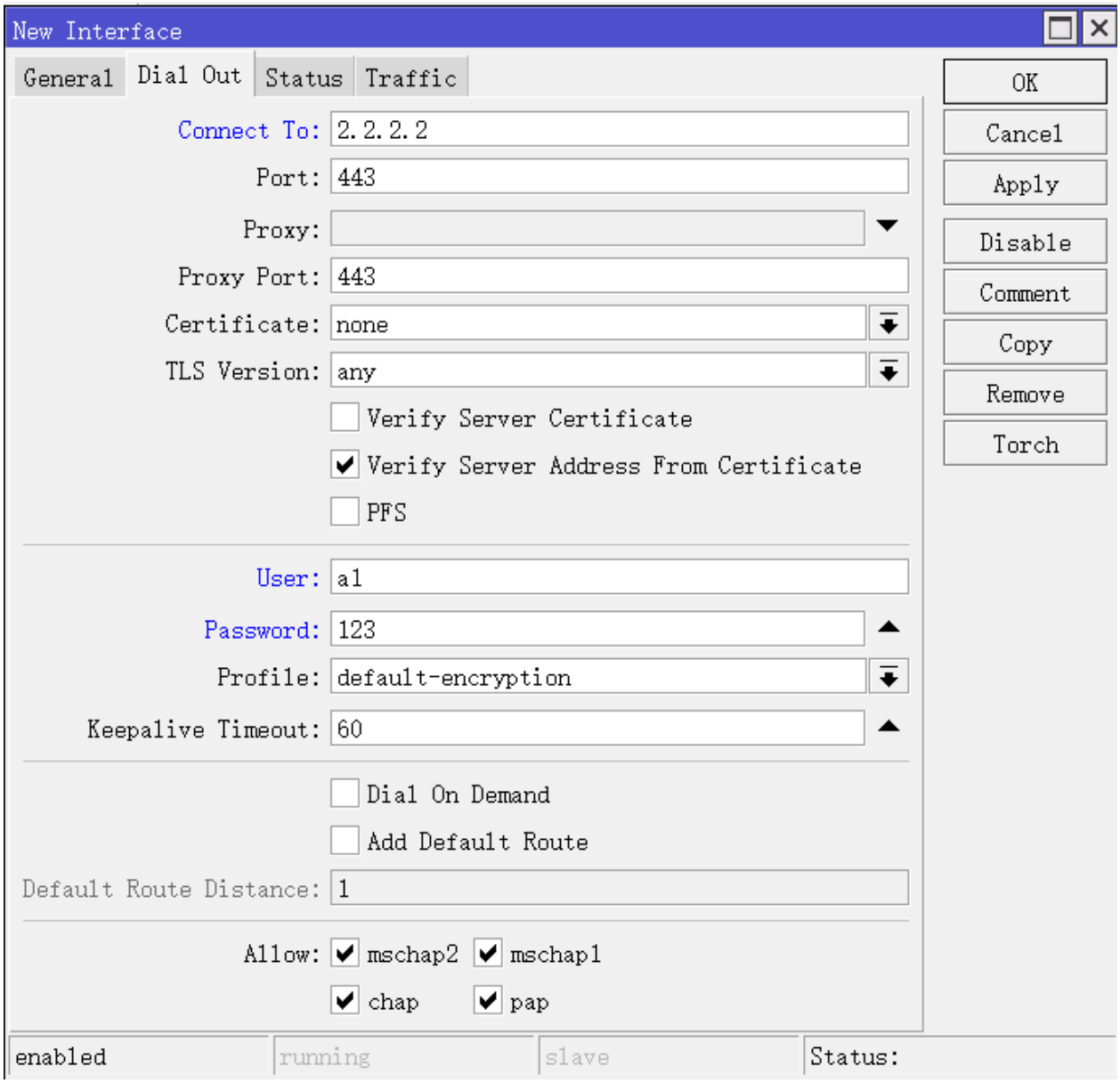
DDNS 的 RouterOS 配置完成。

节点 RouterOS

节点 RouterOS 只需要配置连接到 DDNS RouterOS 的隧道客户端即可。如果你是多线出口的话，需要在 ip route 下配置静态路由，将 2.2.2.2 的 DDNS 服务器 IP 走指定出口访问，以便 DDNS RouterOS 记录到正确的出口 IP。



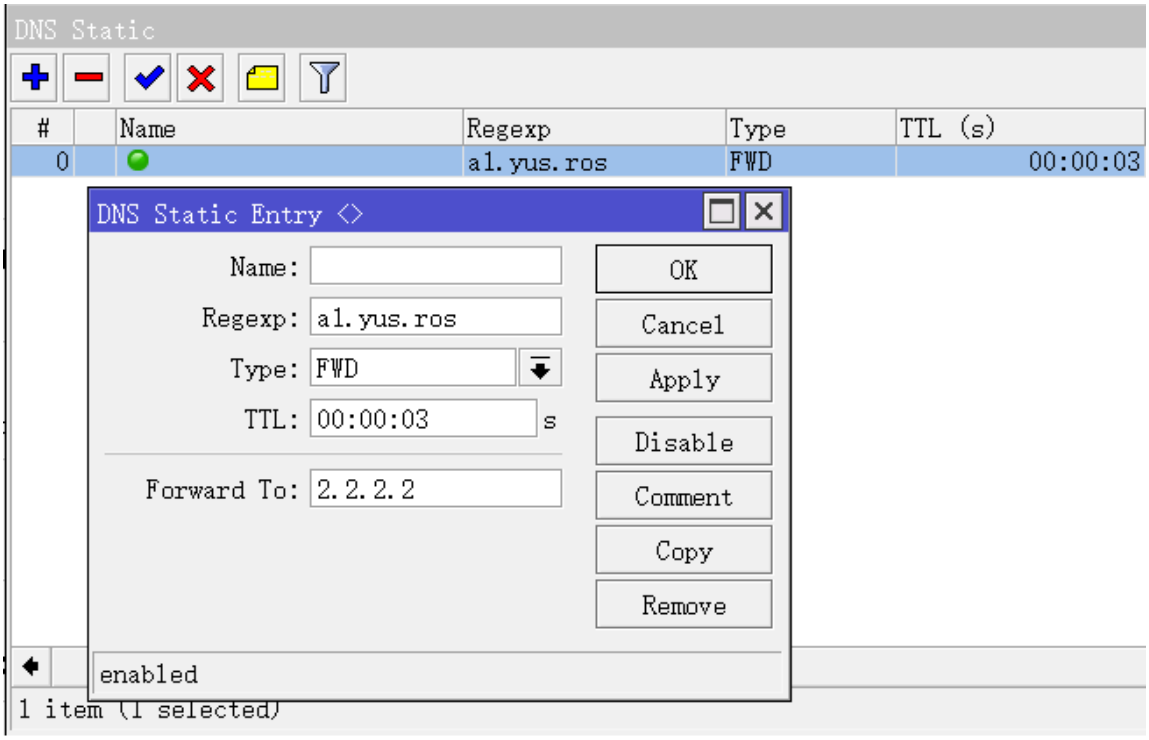
连接 DDNS 服务器，并配置账号和密码分别是 a1 和 123



节点 RouterOS 配置完成

客户端 RouterOS

客户端 RouterOS 只需要将域名 a1.yus.ros 解析转发到 2.2.2.2 服务器，同样需要客户端 RouterOS 在 ip dns 下启用 allowed-remote-request

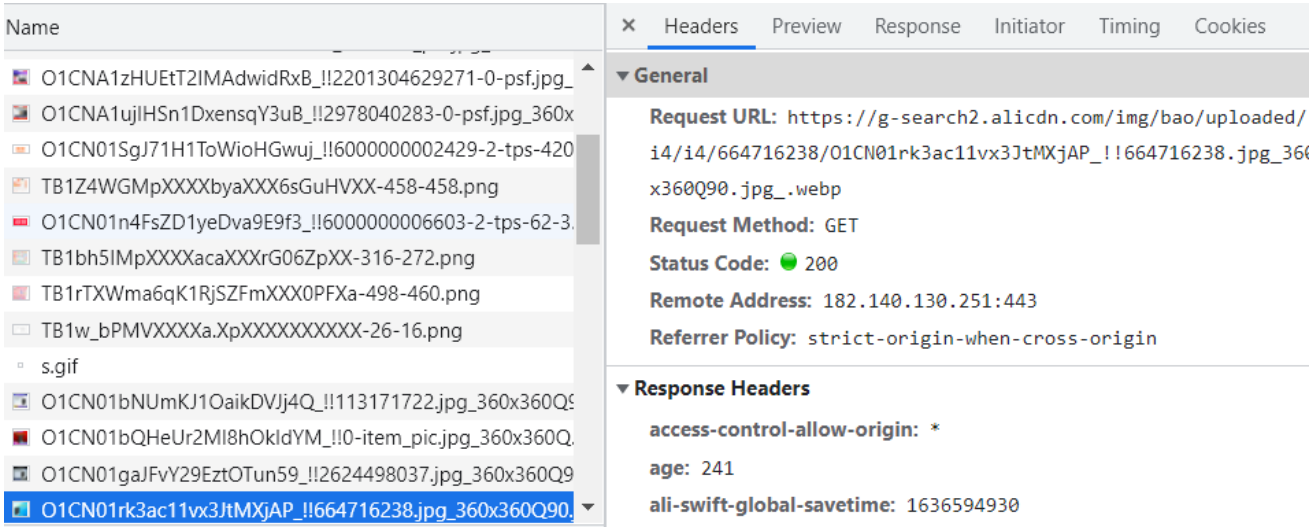


由于节点 RouterOS 是动态 IP，客户端需要连接节点 RouterOS 的隧道协议，就可以使用 a1.yus.ros 这个域名进行访问，

这个算不算建立了软件定义网络的概念，个人感觉也很模糊，但至少 RouterOS 提供的这些功能是可以组合在一起，达到我们想要的结果。

8.9 DNS cache 捕获域名 IP 地址

RouterOS 的/ip firewall address-list 支持填写域名，并自动解析返回 IP 地址，但仅支持明文域名，无法通过正则表达式去匹配泛域名。一般大型网站，图片、下载、视频和 css 都是单独的域名，如某宝网站可以通过浏览器的 F12 抓取域名，如下图：



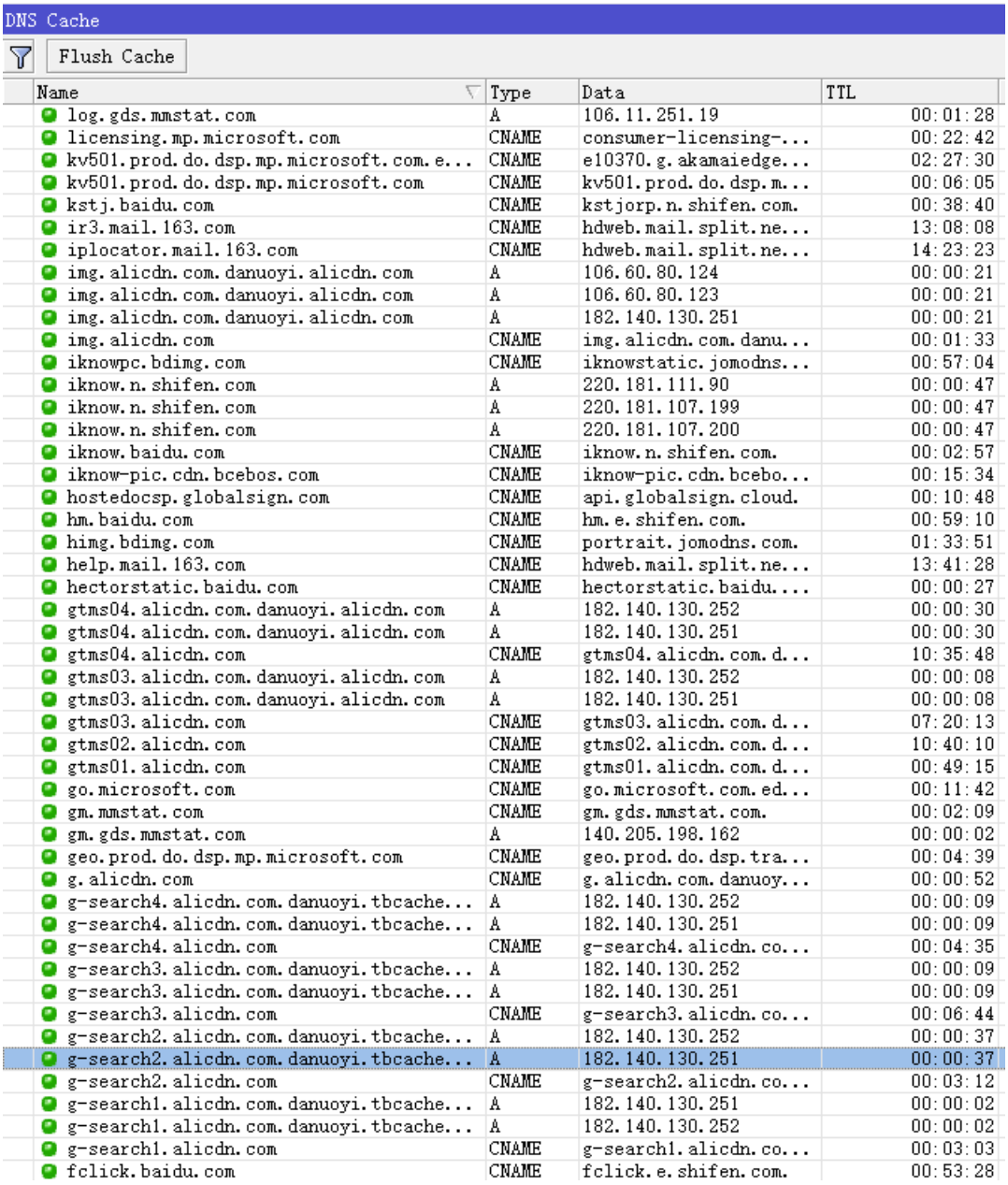
可以看到域名 g-search2.alicdn.com

然后在 windows 的 cmd 下通过 nslookup 命令解析，可以看到 g-search2.alicdn.com 是 CNAME 到了 img.alicdn.com.danuoyi.alicdn.com 域名，解析的 IPv4 和 IPv6 地址

```
C:\Users\yus>nslookup g-search2.alicdn.com
服务器:  UnKnown
Address:  192.168.80.1

非权威应答:
名称:    g-search2.alicdn.com.danuoyi.tbcache.com
Addresses:  182.140.130.252
 Aliases:  g-search2.alicdn.com

C:\Users\yus>nslookup gw.alicdn.com
服务器:  UnKnown
Address:  192.168.80.1

非权威应答:
名称:    gw.alicdn.com.danuoyi.tbcache.com
Addresses:  240e:d9:a003:1400:3::3fb
          240e:d9:c202:300:3::3f3
          118.112.14.5
          182.140.130.251
          182.242.89.225
          219.144.108.251

Aliases:  gw.alicdn.com
```

还有 g-search1.alicdn.com 、g-search3.alicdn.com 和 gw.alicdn.com，同样在 cmd 命令下操作解析，可以看到通过 CNAME 分别转发到了 gw.alicdn.com.danuoyi.tbcache.com 和 g-search1/2/3.alicdn.com.danuoyi.tbcache.com。由于域名过多，如果通过/ip firewall address-list 一个个添加非常繁琐。可以选择 DNS cache 使用正则表达式去抓取这些域名的 IP

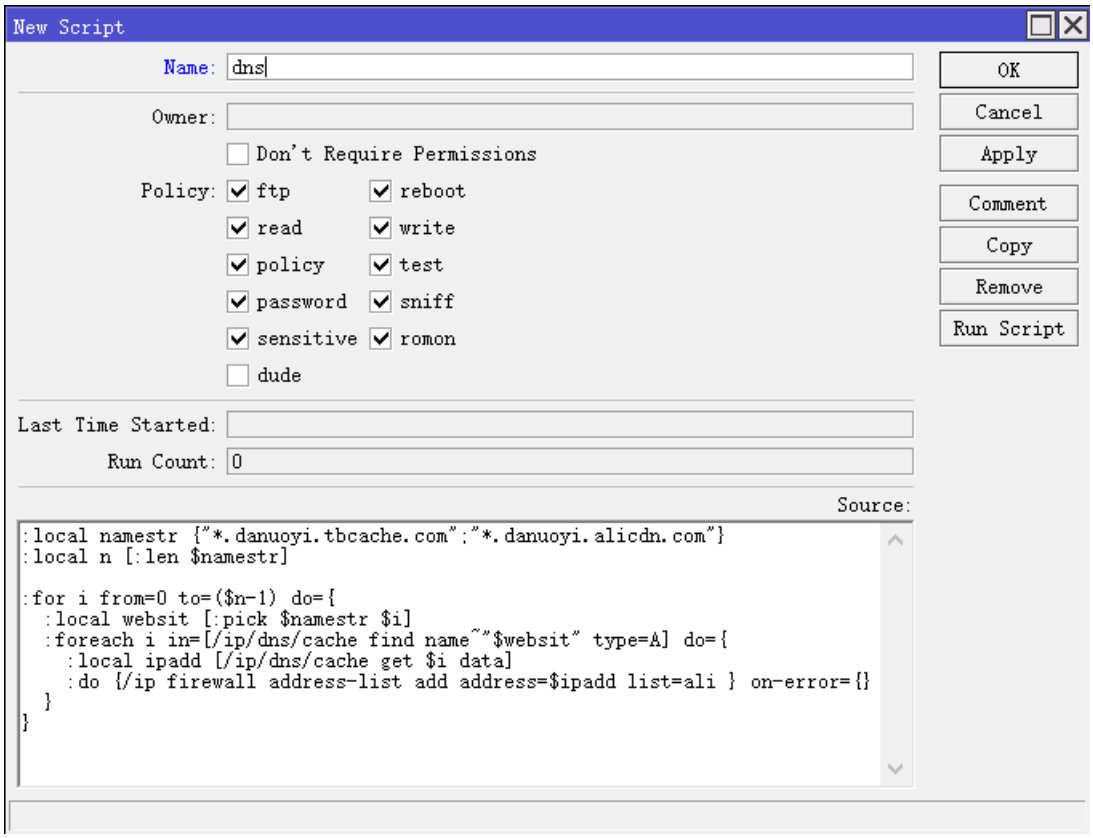
首先需要开启 DNS 服务，在 cache 列表中看到这些域名解析信息如域名、IP 和类型等：



Name	Type	Data	TTL
log.gds.mmstat.com	A	106.11.251.19	00:01:28
licensing.mp.microsoft.com	CNAME	consumer-licensing-...	00:22:42
kv501.prod.do.dsp.mp.microsoft.com.e...	CNAME	e10370.g.akamaiedge...	02:27:30
kv501.prod.do.dsp.mp.microsoft.com	CNAME	kv501.prod.do.dsp.m...	00:06:05
kstj.baidu.com	CNAME	kstjorp.n.shifen.com.	00:38:40
ir3.mail.163.com	CNAME	hdweb.mail.split.ne...	13:08:08
iplocator.mail.163.com	CNAME	hdweb.mail.split.ne...	14:23:23
img.alicdn.com.danuoyi.alicdn.com	A	106.60.80.124	00:00:21
img.alicdn.com.danuoyi.alicdn.com	A	106.60.80.123	00:00:21
img.alicdn.com.danuoyi.alicdn.com	A	182.140.130.251	00:00:21
img.alicdn.com	CNAME	img.alicdn.com.danu...	00:01:33
iknowpc.bdimg.com	CNAME	iknowstatic.jomodns...	00:57:04
iknow.n.shifen.com	A	220.181.111.90	00:00:47
iknow.n.shifen.com	A	220.181.107.199	00:00:47
iknow.n.shifen.com	A	220.181.107.200	00:00:47
iknow.baidu.com	CNAME	iknow.n.shifen.com.	00:02:57
iknow-pic.cdn.bcebos.com	CNAME	iknow-pic.cdn.bcebo...	00:15:34
hostedocsp.globalsign.com	CNAME	api.globalsign.cloud.	00:10:48
hm.baidu.com	CNAME	hm.e.shifen.com.	00:59:10
himg.bdimg.com	CNAME	portrait.jomodns.com.	01:33:51
help.mail.163.com	CNAME	hdweb.mail.split.ne...	13:41:28
hectorstatic.baidu.com	CNAME	hectorstatic.baidu....	00:00:27
gtms04.alicdn.com.danuoyi.alicdn.com	A	182.140.130.252	00:00:30
gtms04.alicdn.com.danuoyi.alicdn.com	A	182.140.130.251	00:00:30
gtms04.alicdn.com	CNAME	gtms04.alicdn.com.d...	10:35:48
gtms03.alicdn.com.danuoyi.alicdn.com	A	182.140.130.252	00:00:08
gtms03.alicdn.com.danuoyi.alicdn.com	A	182.140.130.251	00:00:08
gtms03.alicdn.com	CNAME	gtms03.alicdn.com.d...	07:20:13
gtms02.alicdn.com	CNAME	gtms02.alicdn.com.d...	10:40:10
gtms01.alicdn.com	CNAME	gtms01.alicdn.com.d...	00:49:15
go.microsoft.com	CNAME	go.microsoft.com.ed...	00:11:42
gm.mmstat.com	CNAME	gm.gds.mmstat.com.	00:02:09
gm.gds.mmstat.com	A	140.205.198.162	00:00:02
geo.prod.do.dsp.mp.microsoft.com	CNAME	geo.prod.do.dsp.tra...	00:04:39
g.alicdn.com	CNAME	g.alicdn.com.danuoy...	00:00:52
g-search4.alicdn.com.danuoyi.tbcache...	A	182.140.130.252	00:00:09
g-search4.alicdn.com.danuoyi.tbcache...	A	182.140.130.251	00:00:09
g-search4.alicdn.com	CNAME	g-search4.alicdn.co...	00:04:35
g-search3.alicdn.com.danuoyi.tbcache...	A	182.140.130.252	00:00:09
g-search3.alicdn.com.danuoyi.tbcache...	A	182.140.130.251	00:00:09
g-search3.alicdn.com	CNAME	g-search3.alicdn.co...	00:06:44
g-search2.alicdn.com.danuoyi.tbcache...	A	182.140.130.252	00:00:37
g-search2.alicdn.com.danuoyi.tbcache...	A	182.140.130.251	00:00:37
g-search2.alicdn.com	CNAME	g-search2.alicdn.co...	00:03:12
g-search1.alicdn.com.danuoyi.tbcache...	A	182.140.130.251	00:00:02
g-search1.alicdn.com.danuoyi.tbcache...	A	182.140.130.252	00:00:02
g-search1.alicdn.com	CNAME	g-search1.alicdn.co...	00:03:03
fclick.baidu.com	CNAME	fclick.e.shifen.com.	00:53:28

分别转发到了 gw.alicdn.com.danuoyi.tbcache.com 和 g-search2.alicdn.com.danuoyi.tbcache.com，他们的共同点是相同二级域名 danuoyi.tbcache.com，这样使用脚本进行匹配，匹配原则是包含域名，且 type 为 A 记录的 IP 地址加入到一个 address-list 中

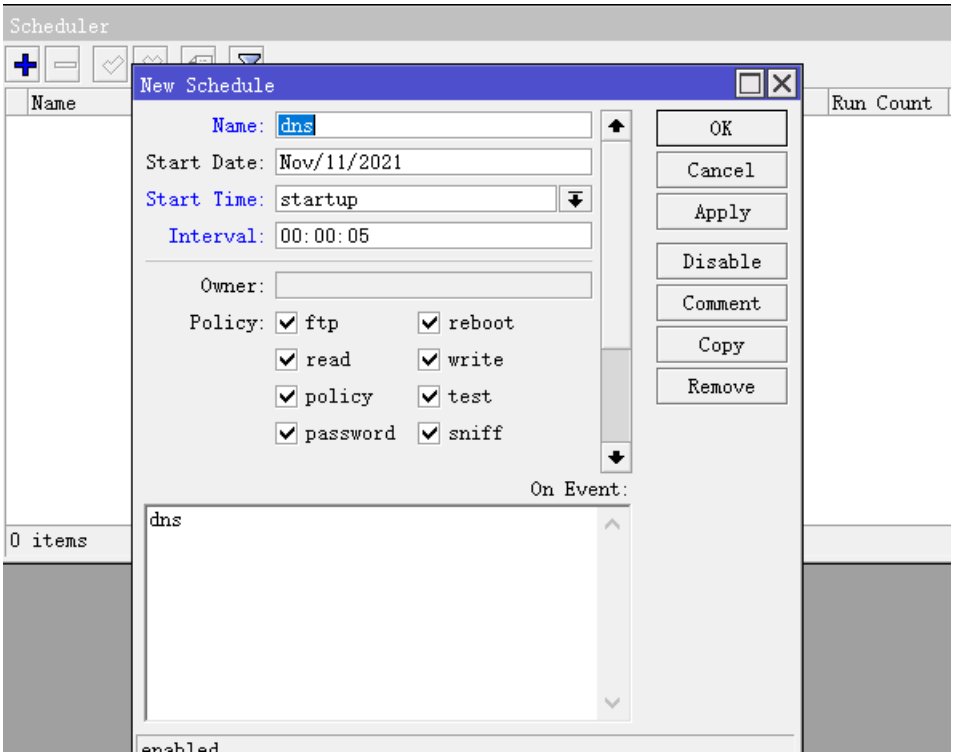
在/system scripts 编写以下脚本，这里还包含了另外一个域名 danuoyi.alicdn.com，脚本基于 RouterOS v7 运行



脚本文本如下（兼容 v6 和 v7 版本）：

```
:local namestr {"*.danuoyi.tbcache.com";"*.danuoyi.alicdn.com"}
:local n [:len $namestr]

:for i from=0 to=($n-1) do={
  :local websit [:pick $namestr $i]
  :foreach i in=[/ip dns cache find name~"$websit" type=A] do={
    :local ipadd [/ip dns cache get $i data]
    :do {/ip firewall address-list add address=$ipadd list=ali } on-error={}
  }
}
```

然后在/system scheduler 计划任务中加入在 script 中创建的脚本名称，interval 是间隔 5 秒执行，也就是每 5 秒抓取一次 dns cache 列表的域名



涉及相关域名的 IP 地址将会被加入到 ali 地址列表中（/ip firewall address-list 可以看到），如何调用这个地址列表，就是管理员自己发挥了。

第九章 Firewall filter（防火墙过滤）

在 RouterOS 通过 ip firewall filter 能对 IP 数据报、P2P 协议、源和目标 IP、端口、IP 协议、协议（ICMP、TCP、MSS 等）、网络接口、对内部的数据报和连接作标记、ToS 字节、关键内容、时间控制和包长度进行过滤控制

RouterOS 是基于 iptables，如果你熟悉 Linux 的 iptables 操作，那 RouterOS 防火墙就能很快上手，与 iptables 类似的分为 4 个基础表 Filter、Mangle、NAT 和 RAW。这里介绍的是 RouterO 防火墙 Filter：分为 input、foreward 和 output 三种链（chain）， RouterOS 的 Filter 支持的扩展模块包括了 address-list 和 L7-protocol 等调用。

本章节主要讲解/ip firewall filter 过滤策略，下面我们看看一些简单设置：

- 添加一条 firewall 规则，将所有通过路由器到目标协议为 TCP 端口为 135 的数据报全部丢弃掉：

```
/ip firewall filter add chain=forward dst-port=135 protocol=tcp action=drop
```

- 拒绝通过 Telnet 访问路由器(协议 TCP, 端口 23)：

```
/ip firewall filter add chain=input protocol=tcp dst-port=23 action=drop
```

注意：RouterOS 防火墙 input、forward 和 output 链表默认都是 accept

9.1 Firewall filter

操作路径：***/ip firewall filter***

在互联网时代，总是会有某些人试图嗅探和攻击别人的网络，窃取数据和破坏网络，通过适当的配置防火墙可以有效的保护自身网络的安全。

MikroTik RouterOS 是功能非常强大的防火墙，包括以下特征：

- IP 包过滤功能
- P2P 协议过滤
- 7 层协议过滤
- IPv6 防火墙过滤（需要在独立 IPv6 包调用）
- 数据传输分类：
    - 源 MAC 地址
    - IP 地址（网段或列表）和地址类型（广播、本地、组播）
    - 端口或端口长度
    - IP 协议
    - 协议选择选项(ICMP 类型和代码字段、TCP 标记、IP 选项和 MSS)
    - Interface 的数据报从那里到达或通过那里里去
    - 内部数据流与连接标记
    - ToS (DSCP)
    - 数据报内容
    - Connection-rate 连接速率

- PCC 分离器
- 数据包大小
- 包到达时间

属性描述

属性	描述
action (*action 指令名称*; 默认: **accept**)	如果数据包匹配此规则执行下列操作： • accept – 接受数据包，不再传递给后面的防火墙规则 • add-dst-to-address-list -通过 address-list 参数，添加目标地址到指定的地址列表中 • add-src-to-address-list -通过 address-list 参数，添加源地址到指定的地址列表中 • drop – 直接丢弃掉数据包 • jump -通过 jump-target 参数跳转到指定的自定义链表 • log -添加 nat 规则信息到系统日志，日志信息包括：in-interface,out-interface,src-mac, protocol, src-ip:port->dst-ip:port 和数据包长度，当该 log 规则匹配并处理对应数据包后，会传递给后面的规则，类似与 passthrough • passthrough -如果数据包配置该规则，将作为对其统计并不做任何处理传递给下一条规则，常用于数据统计。 • reject – 丢弃数据包，并发送 ICMP reject 信息 • return -返回控制，让数据包回到被 jump 跳转策略规则位置 • tarpit – 抓取并压制 TCP 连接(回复 SYN/ACK 入方向的 SYN 数据包)
address-list (字符; 默认:)	调用地址列表（address-list）的规则名称
address-list-timeout (时间; 默认: **00:00:00**)	Address-list 地址列表记录超时时间，当 Address-list 记录 IP 条目超过指定时间，将其删除。该参数与 add-dst-to-address-list 或 add-src-to-address-list 操作关联，当值设定为 00:00:00 IP 条目将用于被记录
chain (名称; 默认:)	指定添加的链表。如果输入的链表名称不存在，则将创建新的链表。
comment (字符; 默认:)	填写规则备注
connection-bytes (整型-整型; 默认:)	匹配给定数据包连接的传输字节（bytes）。0-表示无穷大，例如 connection-bytes=2000000-0 意思是该规则匹配超过 2Mbytes 的相关传输流量
connection-limit (整型,掩码; 默认:)	限制每 IP 地址或 IP 地址段的连接数量
connection-mark (no-mark | 字符; 默认:)	通过 mangle 标记指定的数据包做连接匹配，如果选择 **no-mark**，规则将匹配未标记连接的任何数据包

connection-nat-state (*srcnat* | *dstnat*; 默认:)	用于匹配连接为 srcnatted、dstnatted 或两种类型。用于源和目标 nat 转换的会话处理，例如记录 nat 会话日志
connection-rate (整型 0..4294967295; 默认:)	连接数率一个防火墙匹配器，能用于捕获当前连接的传输流量（具体使用可以参考 12.3 章节）
connection-type (ftp | h323 | irc | pptp | quake3 | sip | tftp; 默认:)	使用连接跟踪助手匹配相关连接的数据包，一个相关连接助手信息必须在/ip firewall service-port 下启用
connection-state (*estabilished* | *invalid* | *new* | *related* | *untracked*; 默认:)	用于匹配连接跟踪数据状态 • `established` – 匹配一个数据包属于已建立的连接 • `invalid` – 一个数据包在连接跟踪列表中无法判断状态，由于这个原因非法的数据包将不会加入到 NAT 转换，但当进行路由时，仍然会包含源地址，这里强烈建议使用 connection-state=invalid ，在 filter 的 forward 和 input 链表丢弃这类数据包。 • `new` – 一个数据包开始建立一个新的连接。 • `related` – 一个数据包与之相关联的连接，但并不是当前存在连接的一部分，例如：ICMP errors 信息或一个 FTP 数据包的数据连接 • `untracked` –在 firewall raw 设置数据包设置为绕过连接跟踪（connection tracking）
content (*字符*; 默认:)	匹配数据包包含的文本字段
dscp (整型: 0..63; 默认:)	匹配 IP 包头的 DSCP 字段信息
dst-address (IP/掩码 | IP 范围; 默认:)	匹配数据包的目标地址或指定目标地址的范围
dst-address-list (名称; 默认:)	使用 address-list 设置的 IP 地址，匹配数据包的目标地址
dst-address-type (unicast | local | broadcast | multicast; 默认:)	匹配目标地址类型： • `unicast` – IP 地址点对点传输 • `local` – 目标地址属于本地路由器接口 IP • `broadcast` – 数据包发送到所有设备的子网域 • `multicast` – 数据包转发到指定的组播设备
dst-limit (整型[/时间],整型,目标地址 | 目标端口| 源地址[/时间]; 默认:)	匹配数据包直到超出给定的 pps 值。与 limit 属性不同的是每个目标 IP 地址/目标端口有自己的 limit 属性，参数格式：`count[/time],burst,mode[/expire]`. • **count** –测量指定“time”周期最大的数据包转发率均值 • **time** –指定时间周期，用于测量数据包转发率(可选项，如果未指定，默认为 1 秒) • **burst** –设置一个突发值，多少数据包不会被 count 限制 • **mode** – 数据包率分类

	• **expire** – 指定一个记录 IP 地址/端口的到期时间，指定期到后自动删除
dst-port (整型[-整型]: 0..65535; 默认:)	仅协议配置为 TCP 或 UDP 协议时，可设置目标端口或目标端口范围值
fragment (yes|no; 默认:)	匹配分段数据包。第一个分段不会被记录。如果连接跟踪被启用，将不会有分段，系统会自动组装每一个数据包
hotspot (auth | from-client | http | local-dst | to-client; 默认:)	匹配 Hotspot 类型
icmp-options (整型：整形; 默认:)	匹配 ICMP 类型：代码字段
in-bridge-port (名称; 默认:)	当实际接口加入 bridge 后，数据包进入路由器的 bridge 接口
in-interface (名称; 默认:)	数据包进入路由器的网络接口
ingress-priority (整型: 0..63; 默认:)	匹配数据包的入口优先级，该优先级来至于 VLAN, WMM 或 MPLS EXP 字段位。具体阅读
ipsec-policy (in | out, ipsec | none; 默认:)	匹配用于 IPsec 的策略，命令行参数值书写采用格式：`direction, policy`， Direction 是用于选择匹配策略的放向用于解包或策略的封包 • `in` – 在 PREROUTING、INPUT 和 FORWARD 链表生效 • `out` – 在 POSTROUTING、OUTPUT 和 FORWARD 链表生效 • `ipsec` – 匹配属于 IPsec 数据包的处理 • `none` – 匹配不属于 IPsec 数据包的处理，如 IPsec 传输数据包。 例如：如果路由器收到 IPsec 封装的 Gre 数据包，此时 `ipsec-policy=in,ipsec`，将匹配 Gre 数据包，但 `ipsec-policy=in,none`，将匹配 ESP 数据包
jump-target (名称; 默认:)	当 action=jump，将会指定跳转的目标链表名称
layer7-protocol (名称; 默认:)	选择已经定义的 7 层协议规则名称
limit (整型,时间,整型; 默认:)	匹配数据包直到超出给定的 pps 值。参数格式：`count[/time],burst`. • **count** – 测量指定“time”周期最大的数据包转发率均值 • **time** – 指定时间周期，用于测量数据包转发率(可选项，如果未指定，默认为 1 秒) • **burst** – 设置一个突发值，多少数据包不会被 count 限制
log-prefix (字符; 默认:)	当 action=log，为指定的 log 信息前加上文本前缀
nth (整型,整型; 默认:)	设置 nth 数据包规则，具体参考第十八章节

out-bridge-port (名称; 默认:)	当实际接口加入 bridge 后，数据包从路由器的 bridge 出接口
out-interface (名称; 默认:)	数据包从路由器本地网络发出的接口
packet-mark (no-mark | 字符; 默认:)	通过 mangle 标记指定的数据包，如果选择 **no-mark**，规则将匹配未标记的任何数据包
packet-size (整型[-整型]:0..65535; 默认:)	通过字节（byte）匹配指定的数据包长度或范围
per-connection-classifier(哈希值：分母/余数; 默认:)	PCC 匹配器，根据指定的选项对传输流量进行平均分配
port (整型[-整型]: 0..65535; 默认:)	当 protocol=TCP 或 UDP，允许配置指定的任何方向端口（源或目标端口）或指定的端口访问
protocol (名称或协议 ID; 默认: **tcp**)	通过协议名称或代码，匹配指定的 IP 协议
psd (整型,时间,整型,整型; 默认:)	探测可能存在的 TCP 和 UDP 端口扫描。参数采用以下格式： **WeightThreshold, DelayThreshold, LopPortWeight, HighPortWeight**。（触发 PSD 是机制是在 **DelayThreshold** 值时间范围里，**WeightThreshold** >= **LowPortWeight** + **HighPortWeight**） • **WeightThreshold** – 权重阀门值，当前 TCP/UDP 数据包收到从相同主机发送的不同目标端口被视为端口扫描，该值指定总权重值 • **DelayThreshold** – 延迟阀门值，从相同主机发送到路由器的不同的目标端口延迟时间间隔 • **LowPortWeight** – 低端口权重值，设置小于等于 1024 端口的目标端口权重值。 • **HighPortWeight** – 高端口权重值，设置高于 1024 端口的目标端口权重值
random (整型: 1..99; 默认:)	通过给定的概率范围随机匹配数据包
reject-with (*icmp-admin-prohibited* | *icmp-net-prohibited* | *icmp-protocol-unreachable* | *icmp-host-prohibited* | *icmp-network-unreachable* | *tcp-reset* | *icmp-host-unreachable* | *icmp-port-unreachable*; 默认: icmp-network-unreachable)	当 action=reject 时，数据包被拒绝，指定 ICMP 返回的错误信息
routing-mark (字符; 默认:)	通过 mangle 标记指定的数据包做路由匹配
src-address (IP/子网, IP 范围; 默认:)	匹配数据包的源地址或指定源地址的范围

src-address-list (名称; 默认:)	使用 address-list 设置的 IP 地址，匹配数据包的源地址
src-address-type (unicast | local | broadcast | multicast; 默认:)	匹配源地址类型： • unicast – IP 地址点对点传输 • local – 目标地址属于本地路由器接口 IP • broadcast – 数据包发送到所有设备的子网域 • multicast – 数据包转发到指定的组播设备
src-port (整型[-整型]: 0..65535; 默认:)	仅协议配置为 TCP 或 UDP 协议时，可设置源端口或源端口范围值
src-mac-address (MAC 地址; 默认:)	匹配一个数据包的源 MAC 地址
tcp-flags (*ack* | *cwr* | *ece* | *fin* | *psh* | *rst* | *syn* | *urg*; 默认:)	匹配指定的 TCP Flags • ack – 确认应答 • cwr - 拥塞窗口减少 • ece - ECN-echo flag (显式拥塞提醒回应) • fin - 结束会话 • psh – 推送数据包 • rst - 复位，并中断一个连接 • syn - 同步请求，表示开始会话请求 • urg - 紧急
tcp-mss (integer: 0..65535; Default:)	配置一个 IP 数据包的 TCP MSS 值
time (时间-时间,sat | fri | thu | wed | tue | mon | sun; 默认:)	允许创建该规则的时间执行计划，根据匹配的时间范围和日期执行。
tls-host (*字符*; 默认:)	允许过滤匹配基于 TLS SNI 主机名的 https 传输，接受基于 GLOB syntax 的通配符匹配，请注意，如果 TLS 握手被分割成多个 TCP 段（数据包），那么该功能将无法匹配主机名。
ttl (整型: 0..255; 默认:)	匹配数据包 TTL 值

Chains 过滤链表

防火墙操作是借助于防火墙的策略，一个策略规则是告诉路由器如何处理一个 IP 数据报，每一条策略都由两部分组成，一部份是传输状态配置和定义如何操作数据包。数据链（Chains）是为更好的管理和组织策略。

过滤功能有三个默认的数据链（chains）：**input**, **forward** 和 **output** 他们分别负责从哪里进入路由器的、通过路由器转发的与从路由器发出的数据。用户也可用自定义添加链，当然这些链没有默认的传输配置，需要在三条预设的链中对 **action=jump** 策略中相关的 **jump-target** 进行配置。

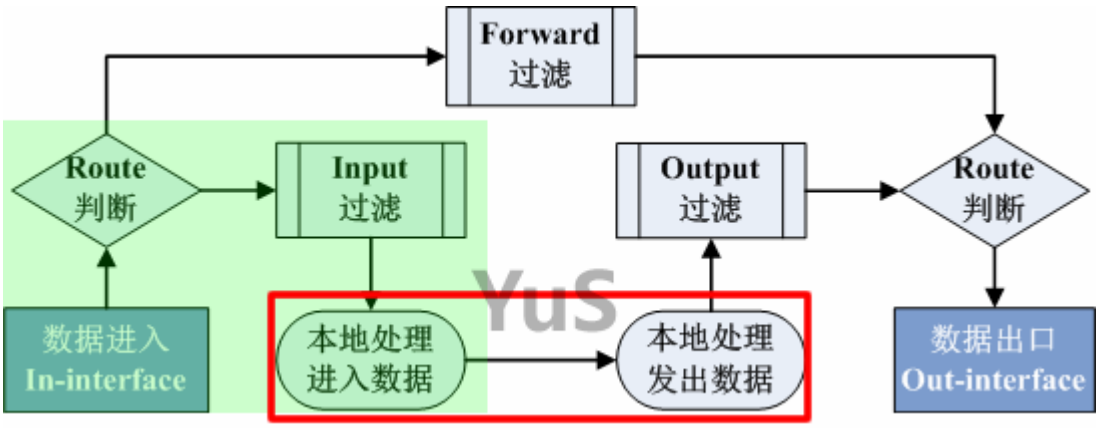
下面是三条预先设置好了的 chains，他们是不被能删除的：

- **input** – 用于处理进入路由器的数据报，即数据报目标 IP 地址是到达路由器一个接口的 IP 地址，经过路由器的数据报不会在 input-chains 处理。
- **forward** – 用于处理通过路由器的数据报

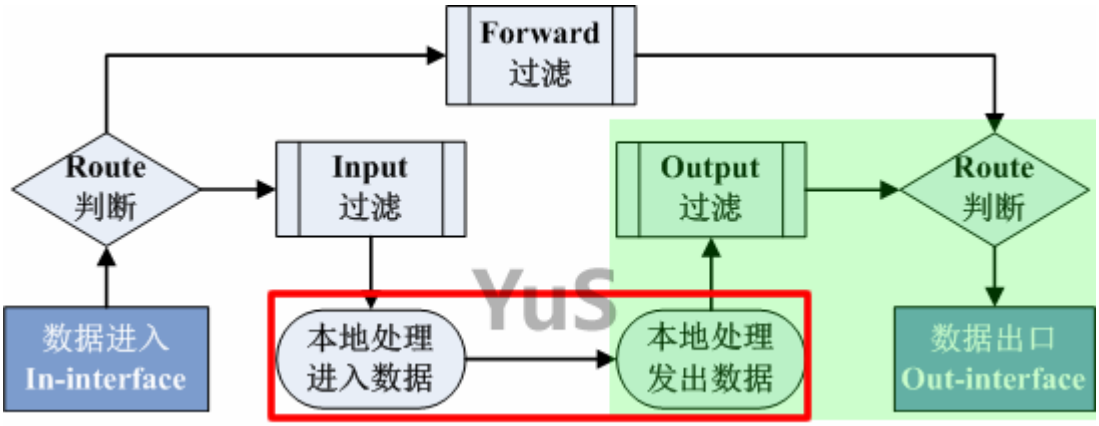
- **output** – 用于处理源于路由器并从其中一个接口出去的数据报。

他们具体的区别如下：

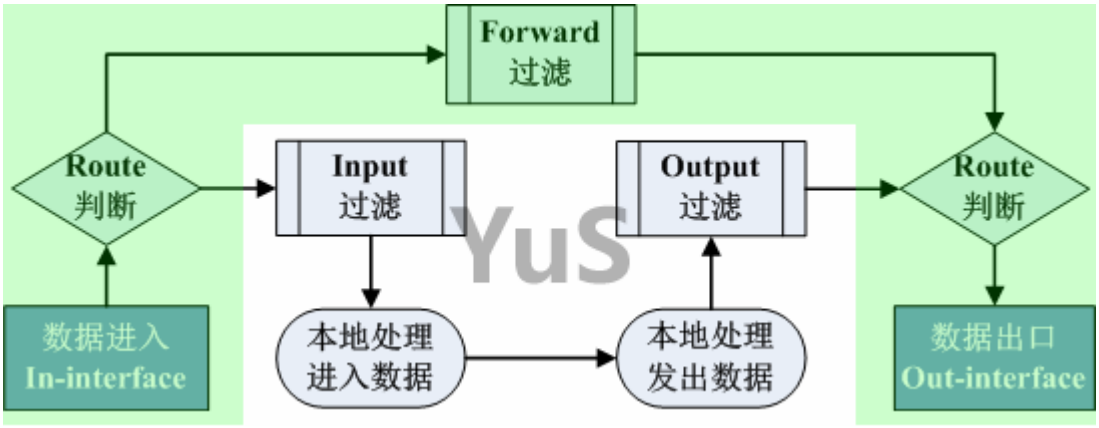
IP 数据报进入 input 链表的数据工作流程，阴影部分代表经过的处理部件：



IP 数据报进入 output 链表的流程，阴影部分代表经过的处理部件：



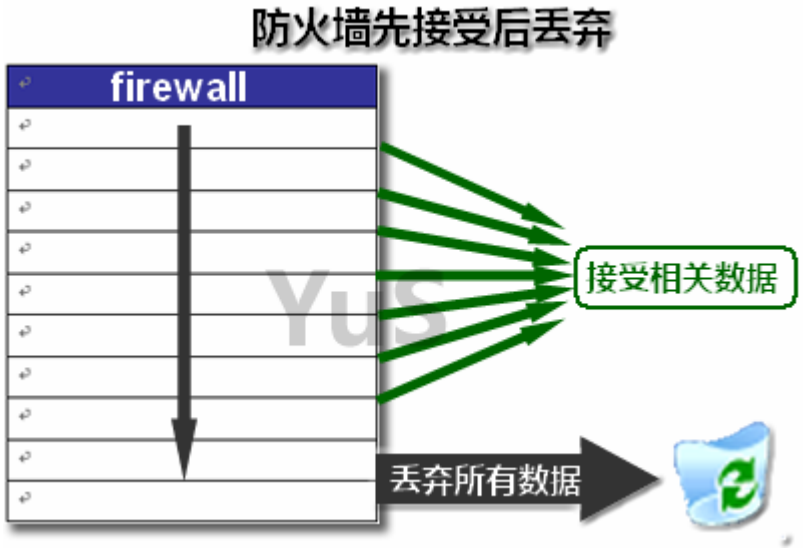
IP 数据进入 forward 链表的流程，阴影部分代表经过的处理部件

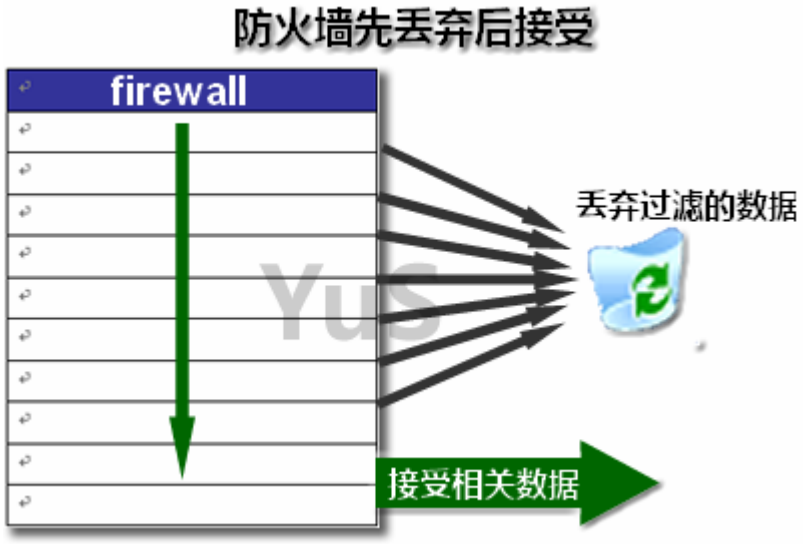


当处理一条 chain（数据链），策略是从 chain 列表的顶部从上而下执行的。如果一个数据报满足策略的条件，这时会执行该操作。

防火墙过滤方式

我们明白 input、forward 和 output 三个链表针对不同方向数据处理功能后，来看看防火墙过滤方式，一般来说只有两种方式，即先接受后丢弃或先丢弃后接受：



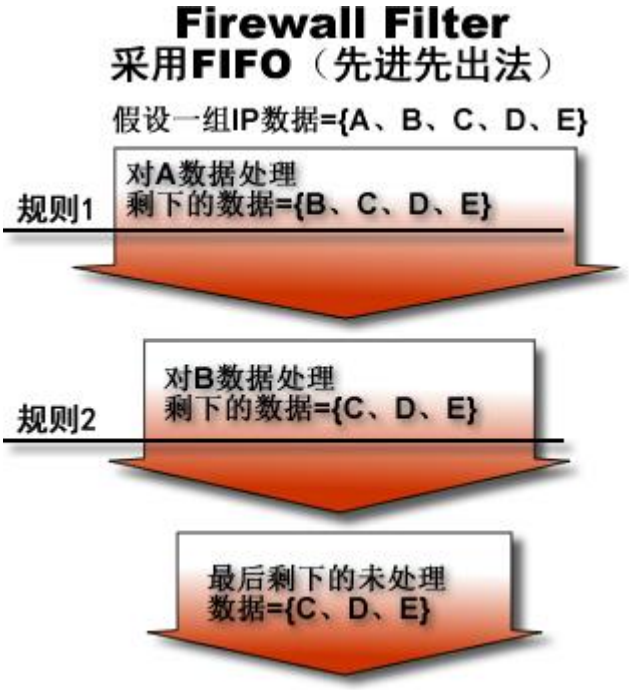


以上两种方式通过图可以很好的理解，在整个 filter 中，当规则为空时，默认是接受所有数据通过，我们通常情况下最多的方式是加入拒绝那些数据通过，即先丢弃后接受。

先接受后丢弃或先丢弃后接受，都是建立防火墙 ACL 过滤方式的选择，例如可以配合 address-list 创建地址列表，对黑白名单进行过滤。

防火墙过滤流程

在 firewall 中，不管是 filter、nat、mangle 等都遵循一个处理原则 FIFO（First In First Out）先进先出算法：

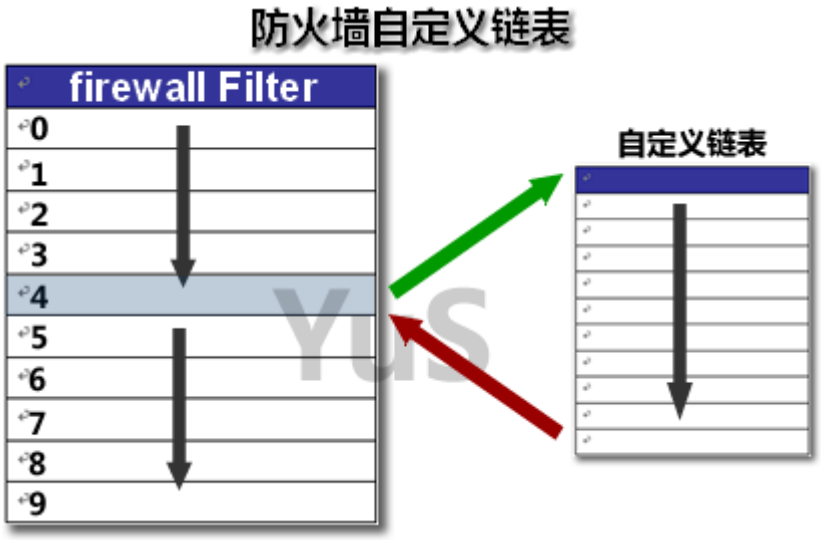


假设我们有 A、B、C、D、E 等 5 个数据，当第一条规则处理了 A 数据，继续向下传递时，第二条规则中不会再出现 A 数据，第二条规则处理 B 数据，当然在后面的规则也不会出现 B 数据。整个流程先处理先通过，规则越靠前，就越优先处理，即从上往下的一个执行过程。

注意：不过在 firewall 中 action 有一个命令是 passthrough，即让该规则直接通过，后面的规则也能对之前处理过的数据再一次处理。

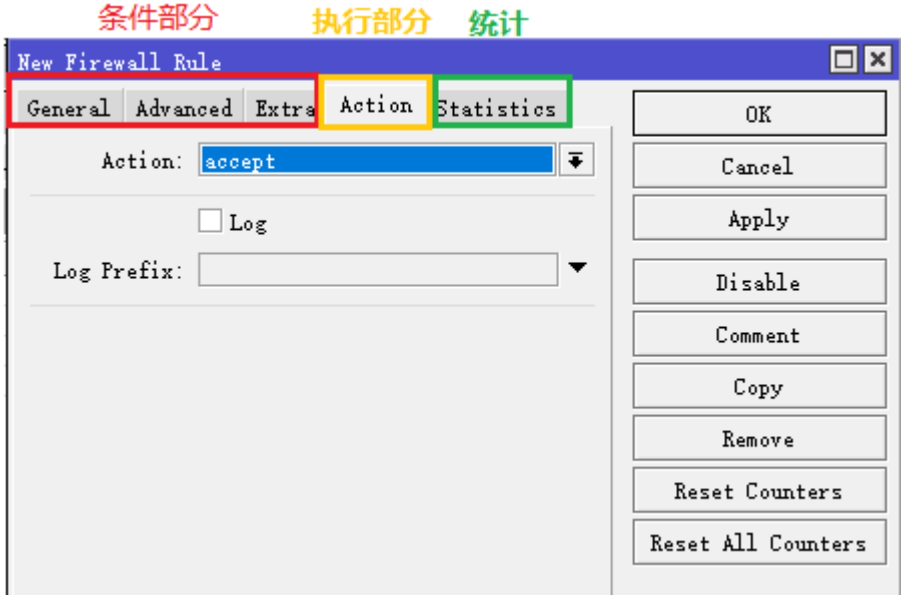
自定义链表

RouterOS 防火墙能自定义链表，即在 filter 中，可以由网络管理员自定义防火墙策略，然后通过 jump 命令调用自定义链表的过滤策略，这样有利于对过滤策略的分类和管理。



规则条件执行

RouterOS 防火墙规则构成是 **if – then** 的方式：**if**=（条件）then **action**=（执行），通过前面的各种条件参数定义规则，最后选择执行方式



RouterOS 的 Action 执行方式有以下几种

- Accept – 接受数据报，例如接受允许数据通过；
- Add-dst-to-address-list – 根据规则条件，将 IP 数据报的目标地址 IP 添加到指定 address-list；
- Add-src-to-address-list – 根据规则条件， 将 IP 数据报的源地址 IP 添加到指定的 address-list；
- Drop – 丢弃掉数据报（不会发送 ICMP 拒绝信息）；
- fasttrack-connection – 加速指定的连接跟踪条目的数据报处理
- Jump – 跳转到指定的链表；
- Log – 与之匹配的操作将会被记录到系统 log 中；
- Passthrough – 忽略该规则，并继续执行下一条规则；
- Reject – 拒绝数据报，并发送 ICMP 拒绝信息；
- Return – 通过返回操作，返回到上一跳转链表；
- Tarpit – 捕捉并控制进入的 TCP 连接。

Connection states（连接状态）

为了更加完全理解防火墙规则，需要掌握数据包在 TCP/IP 协议的各种状态。对于 TCP 传输建立有一个著名的三步握手协议，是学习网络入门的基础，大家可以去百度下。

RouterOS 的防火墙提供了五种连接状态：

- **NEW** – TCP/IP 在传输层建立通信时，创建的会话建立的第一个数据包。在 conntrack 模块在连接中看到请求会话的第一个数据包，并进行匹配。例如，TCP 协议发出的一个 SYN 包，它是我们看到的连接中的第一个包。
- **ESTABLISHED -** 该状态表明两端已经建立连接，双方正在进行数据传输，已建立的连接是很容易理解的，例如：进入 ESTABLISHED 状态的前提是，一台主机发送一个 SYN 包，然后它从另一台主机得到一个 SYN+ACK 回复。当收到 SYN+ACK 回复后，NEW 状态将改变为 ESTABLISHED 状态；

- **RELATED** - 当一个连接与另一个已经建立的连接相关时，它被认为是 RELATED 关联的连接会话。对于一个被认为是 RELATED 的连接，必须首先有一个被认为是 ESTABLISHED 的连接。已建立的连接将在主连接之外生成一个连接。然后，新生成的连接将被视为 RELATED，例如，一个启动 FTP 数据连接的包;
- **INVALID** – 该状态表示数据包无法被识别或没有任何状态。建议在这种状态下执行 drop 丢弃掉;
- **UNTRACKED** -在 firewall raw 链表中被设置为绕过连接跟踪（connection tracking）的数据包。

下面通过一个 input 链表实例来理解下，首先在 filter 的各个链表中规则为空的情况下，防火墙将接受所有的数据通过，通过创建需要的过滤规则完成防火墙对 RouterOS 自身的保护。

```
/ip firewall filter
add chain=input connection-state=invalid action=drop comment="丢弃 INVALID 连接"
add chain=input connection-state=established,related,untracked action=accept comment="允许 Established/Related/Untracked 连接通过"
add chain=input protocol=icmp action=accept ;comment="允许 ICMP 通过"
add chain=input protocol=tcp port=8291,22 action=accept comment="允许 Winbox 和 SSH 端口通过"
add chain=input action=drop comment="丢弃所有数据"
```

在判断一个数据包是 NEW 状态，有两种不同连接情况，一种是在无状态连接(UDP 协议)，Connection 表中无法判断状态。另一种是有状态协议(TCP 协议)，TCP 协议在这种情况下，TCP 判断 NEW 状态是通过的新包总是带有 SYN 标志的 TCP 连接，因此 NEW 状态仅对 TCP 协议有效。

连接状态是 notrack 是使用 RAW 防火墙规则将连接排除在 Connection tracking 连接跟踪之外的一种特殊情况。RAW 规则将使匹配的转发流量绕过 Connection tracking 连接跟踪引擎，加速通过设备的数据包处理。即 NAT 无法在跟踪连接匹配到的 RAW 规则数据，NAT 规则不再生效。

9.2 防火墙 filter 事例

RouterOS 的 ip firewall filter 菜单是主要的防火墙过滤规则，能对三层，四层以及应用层进行过滤，提供了丰富的控制参数（二层防火墙配置，需要启用 bridge 桥接功能后，在 bridge filter 配置）。

基本配置事例

如何使用 filter 过滤指定的端口

添加一条 firewall 规则，将所有通过路由器到目标协议为 TCP，端口为 445 和 135 的数据包丢弃掉：

```
/ip firewall filter add chain=forward dst-port=135 protocol=tcp action=drop
/ip firewall filter add chain=forward dst-port=445 protocol=tcp action=drop
```

拒绝通过 Telnet 访问路由器(协议 TCP，端口 23)：

```
/ip firewall filter add chain=input protocol=tcp dst-port=23 action=drop
```

如何禁止内网 IP 地址上网

进入 ip firewall filter 中，选择 chain=forward 链表，设置内网源地址为 192.168.0.11，action=drop

```
[admin@cdnat] > ip firewall filter
[admin@cdnat] /ip firewall filter> add chain=forward src-address=192.168.0.11 action=drop
```

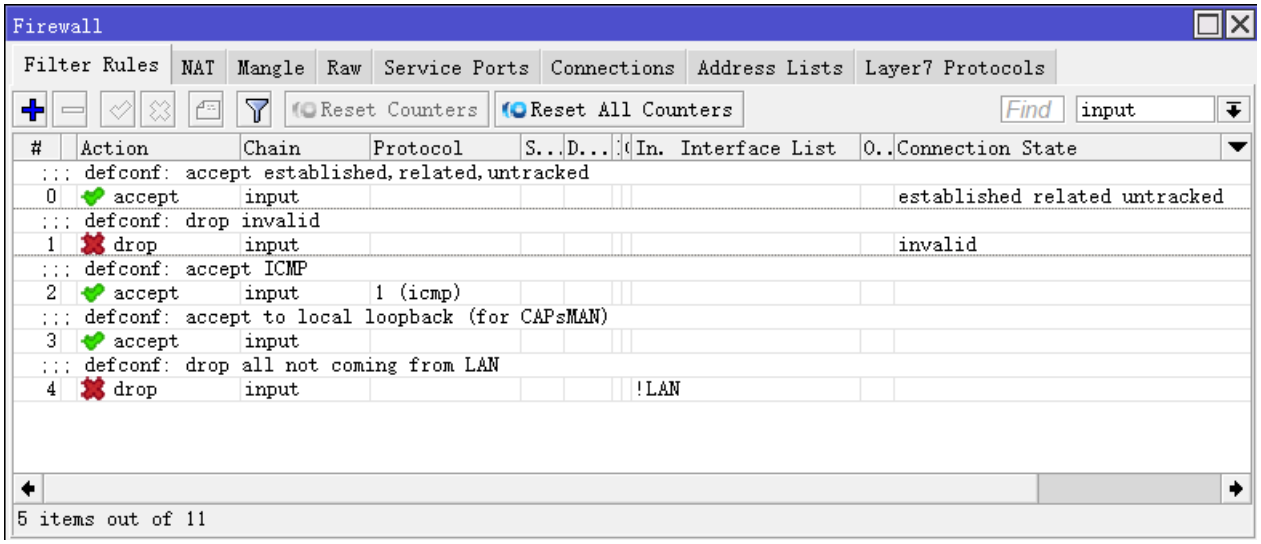
如何禁止访问目标 IP 地址

进入 ip firewall filter 中，选择 chain=forward 链表，创建禁止访问目标 IP 地址.8.8.8.8，设置 action=drop

```
[admin@cdnat] > ip firewall filter
[admin@cdnat] /ip firewall filter> add chain=forward dst-address=8.8.8.8 action=drop
```

Input 链表

先从 input 链表开始，这里是对所有访问路由的数据进行过滤和处理，方向是进入路由器本地的数据，下面是一个对 RouterOS 默认配置规则（部分 RB 设备 hAP ac 系列、hEX 系列、RB3011、RB4011 等有防火墙默认配置），规则的顺序执行是从上往下，序列不能乱。



从 input 链表的第一条开始执行，这里一共有 7 条规则，配置命令如下：

```
[admin@MikroTik] /ip/firewall/filter> print chain=input
Flags: X - disabled, I - invalid; D - dynamic
 0     ;;; defconf: 接受连接状态是 established,related,untracked 的数据
       chain=input action=accept connection-state=established,related,untracked log=no
log-prefix=""

 1     ;;; defconf: 丢弃连接状态是 invalid 的数据
       chain=input action=drop connection-state=invalid log=no log-prefix=""

 2     ;;; defconf: 接受 ICMP 协议通过（包括 ping 和 tracert）
       chain=input action=accept protocol=icmp log=no log-prefix=""

 3     ;;; defconf: 接受本地环回地址 (用于 CAPsMAN)
```

```
      chain=input action=accept dst-address=127.0.0.1 log=no log-prefix=""

 4    ;;; defconf: 丢弃所有不是从 LAN 接口列表进入的数据，代表拒绝 WAN 口数据
      chain=input action=drop in-interface-list=!LAN log=no log-prefix=""
```

关于 in-interface-list=!LAN，需要在 interface list 定义，部分 RB 默认会有配置，定义了 WAN 和 LAN 列表，入下配置

```
[admin@MikroTik] /interface/list> print
Flags: *, D - DYNAMIC
Columns: NAME
#    NAME
;;; contains all interfaces
0 * all
;;; contains no interfaces
1 * none
;;; contains dynamic interfaces
2 * dynamic
;;; contains static interfaces
3 * static
;;; defconf
4    WAN
;;; defconf
5    LAN
```

在 member 定义列表成员，WAN 列表成员是 ether1，LAN 列表成员是 bridge（例如 hAP ac 系列的 bridge 接口包括 ether2-ether5）

```
[admin@MikroTik] /interface/list> member/
[admin@MikroTik] /interface/list/member> print
Flags: X, D - DYNAMIC
Columns: LIST, INTERFACE
#    LIST   INTERFACE
;;; defconf
0    LAN    bridge
;;; defconf
1    WAN    ether1
```

这里我们有一条 ICMP 的自定义链表，用于对 ICMP 数据进行过滤，在后面我们会单独讲解。

RouterOS 默认的 input 防火墙规则，拒绝了所有 WAN 口的请求，如果需要从外网通过 winbox 和 SSH 进行管理，可以添加以下规则

```
[admin@MikroTik] /ip/firewall/filter> add chain=input protocol=tcp   port=8291,22
action=accept comment=" Allow Winbox and SSH "
```

查看当前配置，添加规则在序列 5，最下方，按照 filter 的 FIFO 算法，8291 和 22 端口在规则序列 4 丢弃 WAN 口规则之后，无法生效。

```
[admin@MikroTik] /ip/firewall/filter> print chain=input
Flags: X - disabled, I - invalid; D - dynamic
 0     ;;; defconf: accept established,related,untracked
       chain=input action=accept connection-state=established,related,untracked log=no
log-prefix=""

 1     ;;; defconf: drop invalid
       chain=input action=drop connection-state=invalid log=no log-prefix=""

 2     ;;; defconf: accept ICMP
       chain=input action=accept protocol=icmp log=no log-prefix=""

 3     ;;; defconf: accept to local loopback (for CAPsMAN)
       chain=input action=accept dst-address=127.0.0.1 log=no log-prefix=""

 4     ;;; defconf: drop all not coming from LAN
       chain=input action=drop in-interface-list=!LAN log=no log-prefix=""

 5     ;;; Allow Winbox and SSH
       chain=input action=accept protocol=tcp port=8291,22
```

通过 move 命令将序列 5，移动到序列 4，winbox 操作可以通过拖放方式完成

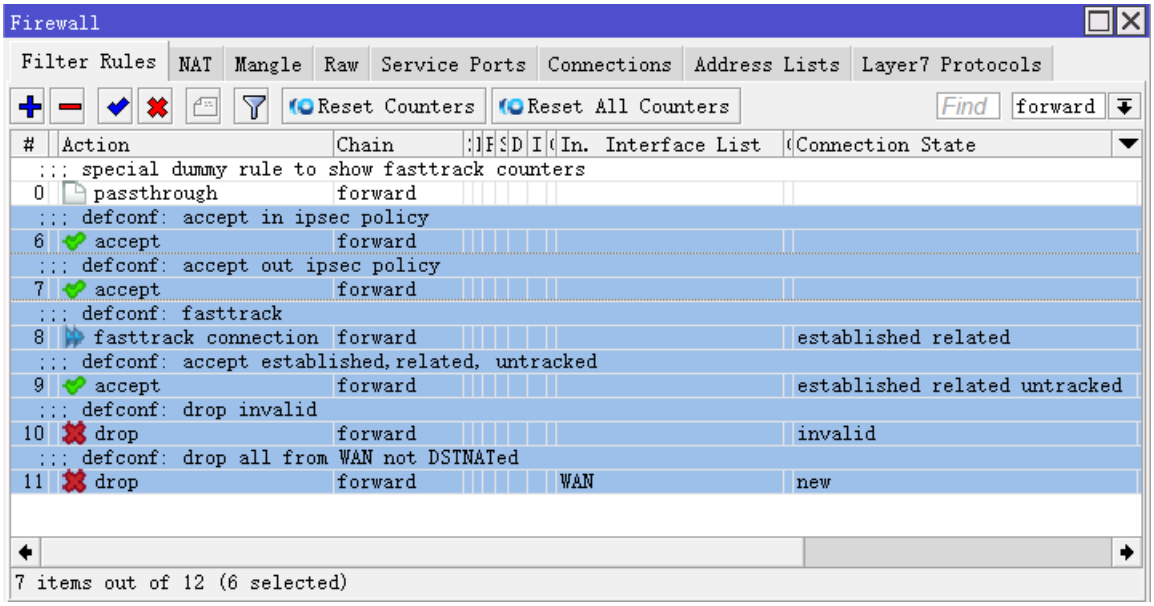
```
[admin@MikroTik] /ip/firewall/filter> move 5 4
```

移动完成后，print 命令查看

```
[admin@MikroTik] /ip/firewall/filter> print chain=input
Flags: X - disabled, I - invalid; D - dynamic
 0     ;;; defconf: accept established,related,untracked
       chain=input action=accept connection-state=established,related,untracked log=no
log-prefix=""

 1     ;;; defconf: drop invalid
       chain=input action=drop connection-state=invalid log=no log-prefix=""

 2     ;;; defconf: accept ICMP
       chain=input action=accept protocol=icmp log=no log-prefix=""

 3     ;;; defconf: accept to local loopback (for CAPsMAN)
       chain=input action=accept dst-address=127.0.0.1 log=no log-prefix=""

 4     ;;; Allow Winbox and SSH
       chain=input action=accept protocol=tcp port=8291,22

 5     ;;; defconf: drop all not coming from LAN
       chain=input action=drop in-interface-list=!LAN log=no log-prefix=""
```

Forward 链表

Input 是对进入路由器方向数据处理，即 input 的作用是在保护路由器做配置，而 forward 链表，则是在对由外向内或由内向外的过滤，下面通过 RouterOS 默认 filter 过滤做介绍（部分 RB 设备 hAP ac 系列、hEX 系列、RB3011、RB4011 等有防火墙默认配置）



通过命令查看：

```
[admin@MikroTik] /ip/firewall/filter> print chain=forward
Flags: X - disabled, I - invalid; D - dynamic
 0   D ;;; 动态规则用于显示 fasttrack 数据统计
       chain=forward action=passthrough

 1     ;;; defconf: 接受 IPsec policy 入策略通过
       chain=forward action=accept log=no log-prefix="" ipsec-policy=in,ipsec

 2     ;;; defconf: 接受 IPsec policy 出策略通过
       chain=forward action=accept log=no log-prefix="" ipsec-policy=out,ipsec

 3     ;;; defconf: fasttrack 规则
       chain=forward action=fasttrack-connection hw-offload=yes
connection-state=established,related log=no log-prefix=""

 4     ;;; defconf: 接受连接状态为 established,related, untracked 的数据
       chain=forward action=accept connection-state=established,related,untracked log=no
log-prefix=""

 5     ;;; defconf: 丢弃连接状态为 invalid 的数据
       chain=forward action=drop connection-state=invalid log=no log-prefix=""
```

6 ;;; defconf: 丢弃来至 WAN 接口列表，连接状态为 new 的非 DSTNAT 数据，用于拒绝外部通过静态路由访问内网，如果你要 WAN 口上联路由配置静态路由访问内网 IP，此规则可以删除。
chain=forward action=drop connection-state=new connection-nat-state=!dstnat in-interface-list=WAN log=no log-prefix=""

如果需要对内网每个主机连接数限制，可以使用 connection-limit 这个参数，限制每个主机 TCP/UDP 连接数为 150 条：

```
[admin@MikroTik] /ip/firewall/filter>add chain=forward action=drop connection-limit=150,32
```

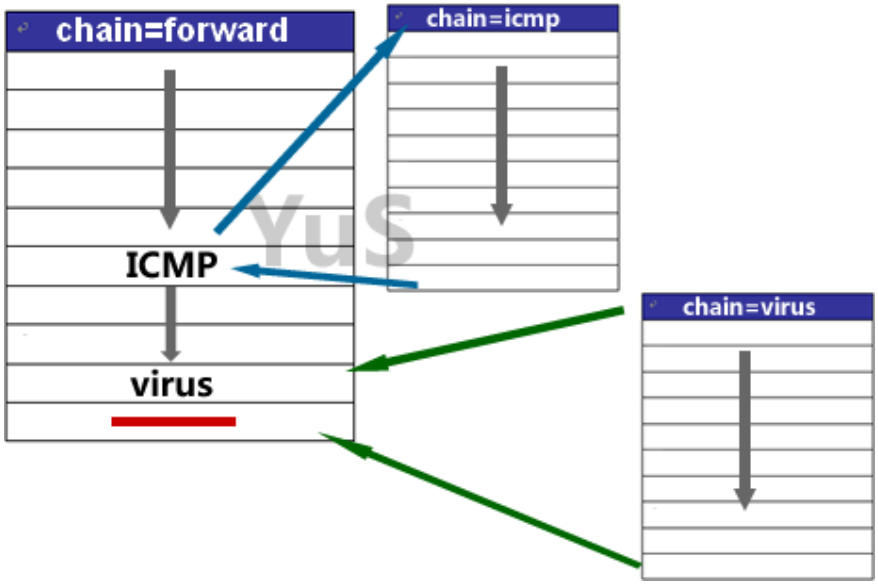
connection-limit=150,32，即每个主机用 32 表示，所有主机用 0 表示。

注：RouterOS 从 v5.7 开始支持 connection-limit 对 UDP 连接的限制，在此之前 RouterOS 只能对 TCP 协议做连接数限制。

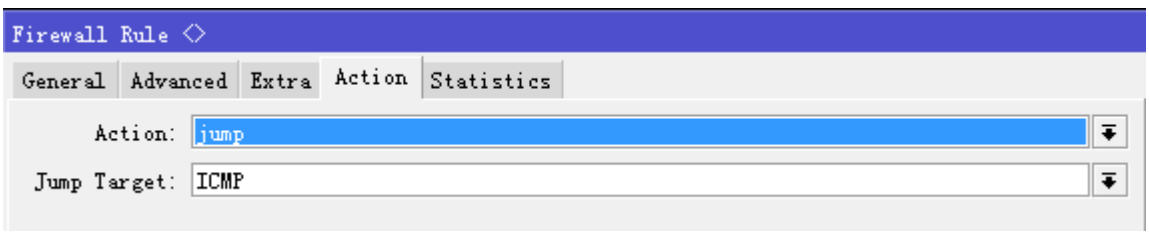
Output 是对由路由器主动发出的数据进行过滤，这类数据需要限制的很少，所以在此就不具体讲解，操作与 input 较类似，读者可以自行实验。

Jump 规则的使用

在 filter 规则总我们多次使用到 jump 指令，该指令可以让我们将指定的数据转向我们自定义的链表中，进行过滤，下面我们举例 forward 链表中通过 jump 工作过程：

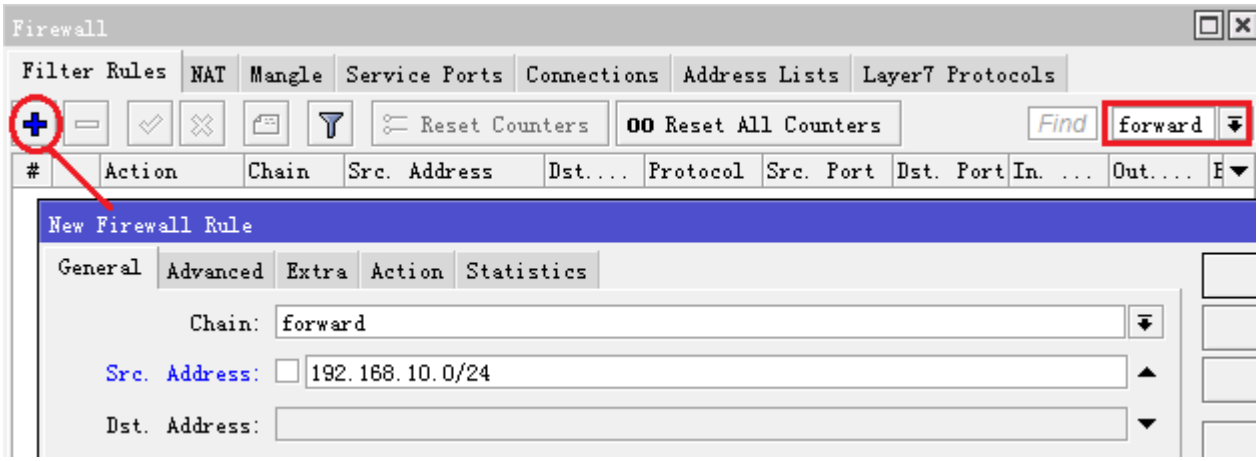


在 winbox 中 jump 的设置

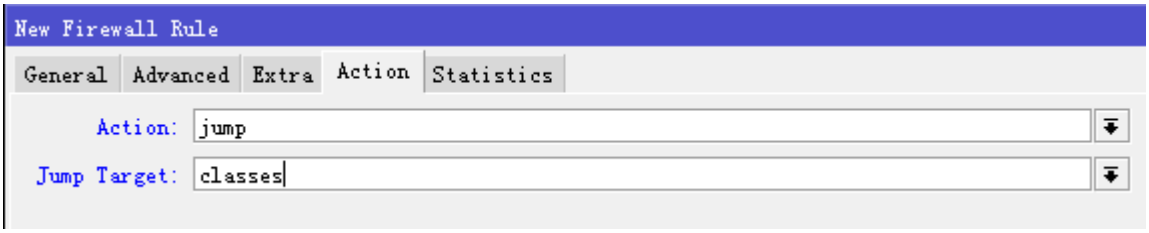


虽然我们可以一次将所有规则在 forward 或 input 等链表中配置，但对于分类管理和查找非常不便，Jump 操作让我们可以将各类过滤规则分类，如我们企业网络，可以将员工 IP 地址段和经理 IP 地址段区分开；在校园网络可以将学生 IP 段和教师 IP 段分开；在 ISP 网络中可以将应用数据进行特点分类等等...

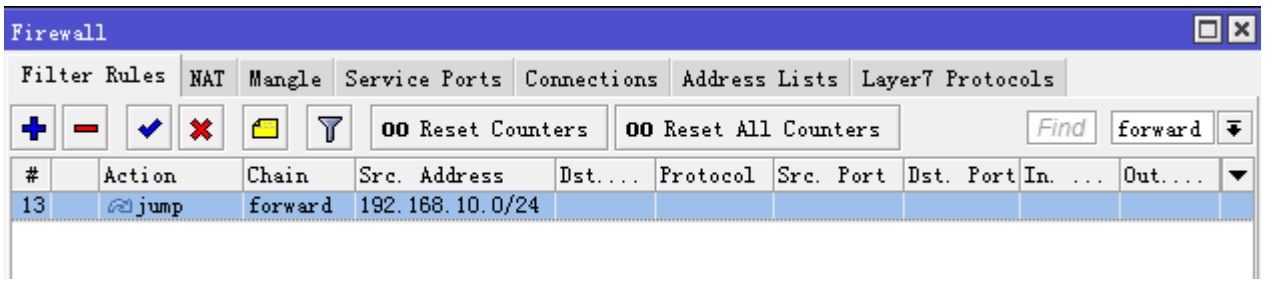
例如我们定义一个 classes 分类的过滤链表，下面是通过 winbox 配置，我们对学生的 ip 地址段 192.168.10.0/24 进行分类跳转：



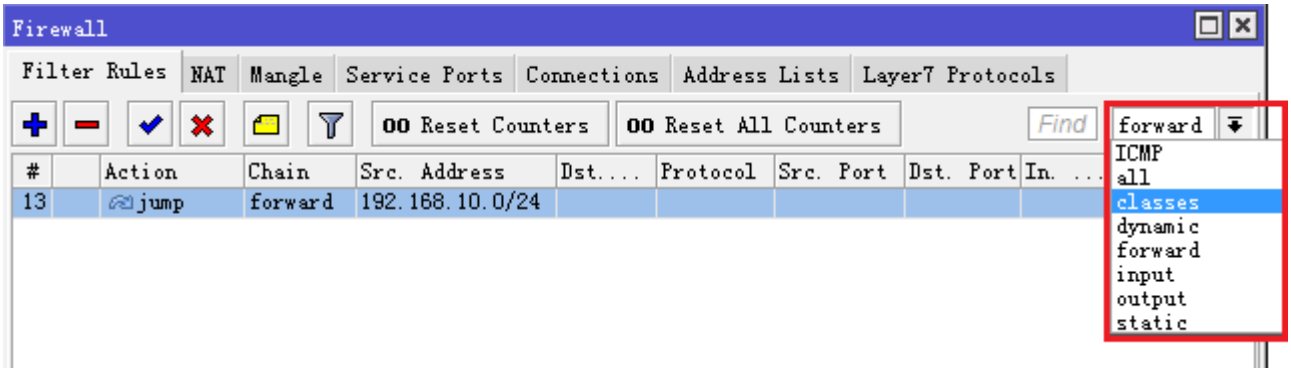
设置 action 为 jump，jump-target 用于指定链表，也可以用于创建一个新链表，这里取名 classes



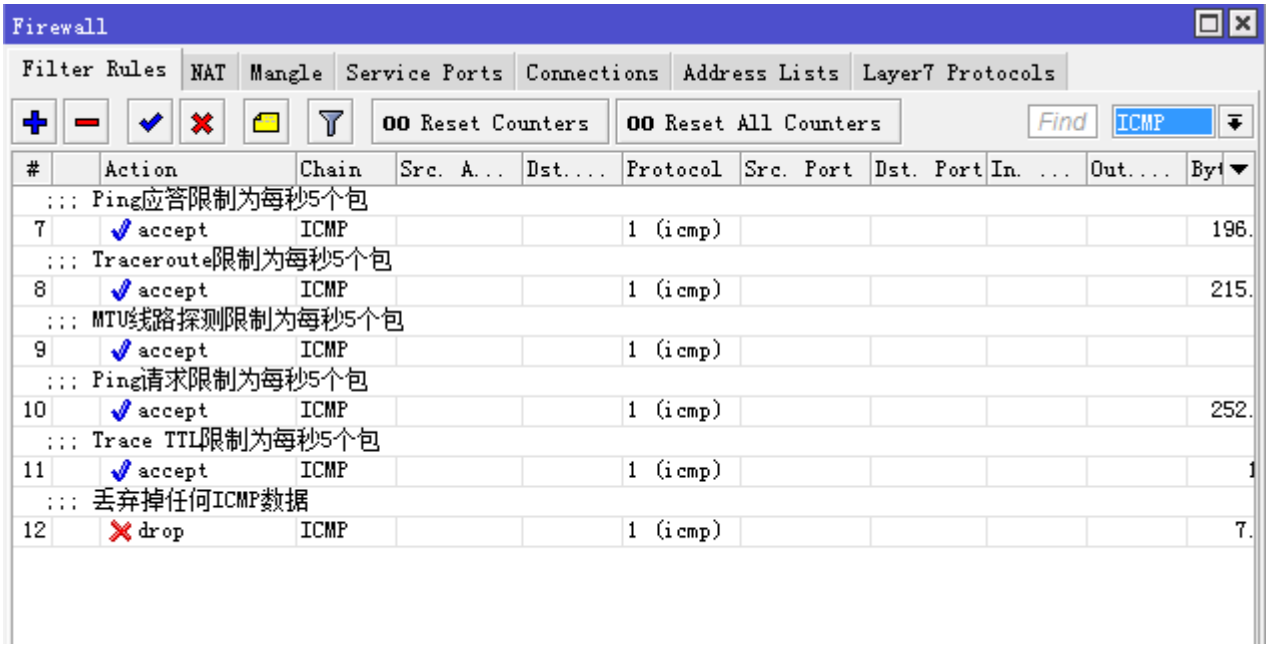
这样我们可以进入 classes 里设置所需的规则，由于 jump 规则已经选择了 192.168.10.0/24 的用户地址，在 classes 链表中，我们无需设置 src-address 的 ip 地址，简化了配置。



通过点击下拉菜单进入 classes 链表



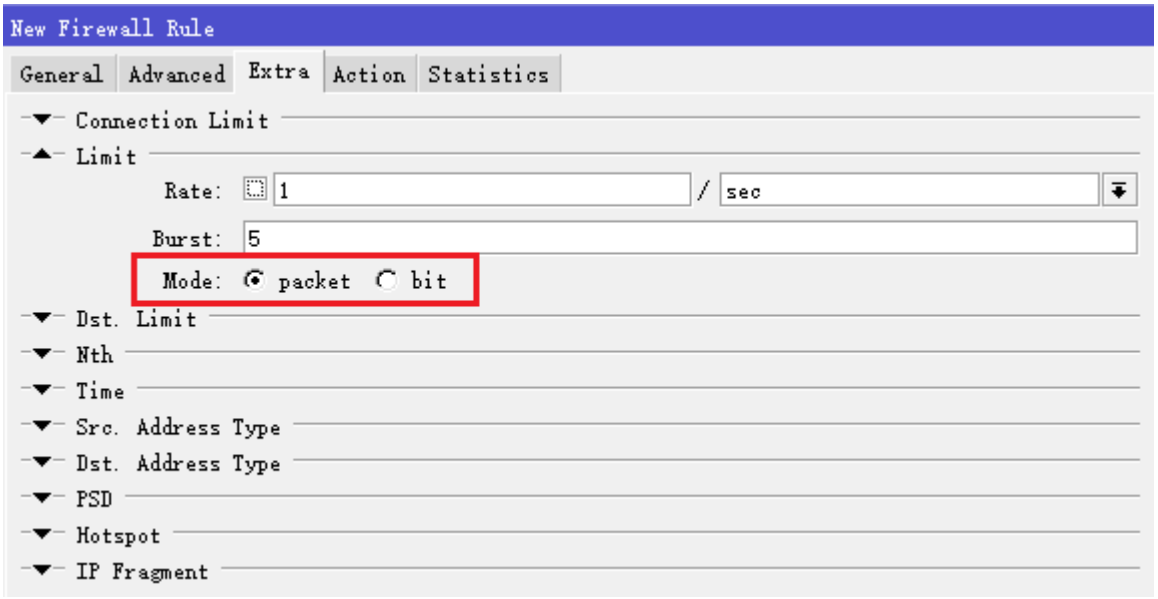
在自定义链表 ICMP 中，定义所有 ICMP（Internet 控制报文协议），例如：ping、traceroute、trace TTL 等。我们通过 ICMP 链表来过滤所有的 ICMP 协议，限制 ICMP 的连接数，当然根据你需要也可以拒绝掉 ICMP：



ICMP 链表操作过程（v6.34 版本前配置）：

```
0   ;;;  Ping 应答限制为每秒 5 个包
    chain=ICMP protocol=icmp icmp-options=0:0-255 limit=5,5 action=accept
1   ;;;  Traceroute 限制为每秒 5 个包
    chain=ICMP protocol=icmp icmp-options=3:3 limit=5,5 action=accept
2   ;;;  MTU 线路探测限制为每秒 5 个包
    chain=ICMP protocol=icmp icmp-options=3:4 limit=5,5 action=accept
3   ;;;  Ping 请求限制为每秒 5 个包
    chain=ICMP protocol=icmp icmp-options=8:0-255 limit=5,5 action=accept
4   ;;; Trace TTL 限制为每秒 5 个包
    chain=ICMP protocol=icmp icmp-options=11:0-255 limit=5,5 action=accept
5   ;;;  丢弃掉任何 ICMP 数据
    chain=ICMP protocol=icmp action=drop
```

注意，在 RouterOS v6.34 后，limit 参数新增了 bit 选项，即能选择 packet 和 bit 参数对数据进行限制



因此，这里的 ICMP 过滤配置脚本，在 v6.34 后修改为：

```
0    ;;;  Ping 应答限制为每秒 5 个包
     chain=ICMP protocol=icmp icmp-options=0:0-255 limit=5,5:packet action=accept
 1   ;;;  Traceroute 限制为每秒 5 个包
     chain=ICMP protocol=icmp icmp-options=3:3 limit=5,5:packet action=accept
 2   ;;;  MTU 线路探测限制为每秒 5 个包
     chain=ICMP protocol=icmp icmp-options=3:4 limit=5,5:packet action=accept
 3   ;;;  Ping 请求限制为每秒 5 个包
     chain=ICMP protocol=icmp icmp-options=8:0-255 limit=5,5:packet  action=accept
 4   ;;; Trace TTL 限制为每秒 5 个包
     chain=ICMP protocol=icmp icmp-options=11:0-255 limit=5,5:packet action=accept
 5   ;;;  丢弃掉任何 ICMP 数据
     chain=ICMP protocol=icmp action=drop
```

基于协议的自定义链表

建立自定义链表是通过 action=jump 创建，如下面我们将 TCP、UDP 和 ICMP 三个协议分类建立三个不同的自定义链表，对各类连接进行防火墙控制，我们通过 jump 命令创建三个链表 tcp、udp 和 icmp

```
add chain=forward protocol=tcp action=jump jump-target=tcp
add chain=forward protocol=udp action=jump jump-target=udp
add chain=forward protocol=icmp action=jump jump-target=icmp
```

三个链表创建完成后，选择 chain=tcp，并拒绝指定的 tcp 端口：

```
add chain=tcp protocol=tcp dst-port=69 action=drop comment="deny TFTP"
add chain=tcp protocol=tcp dst-port=111 action=drop comment="deny RPC portmapper"
add chain=tcp protocol=tcp dst-port=135 action=drop comment="deny RPC portmapper"
add chain=tcp protocol=tcp dst-port=137-139 action=drop comment="deny NBT"
add chain=tcp protocol=tcp dst-port=445 action=drop comment="deny cifs"
add chain=tcp protocol=tcp dst-port=2049 action=drop comment="deny NFS"
```

```
add chain=tcp protocol=tcp dst-port=12345-12346 action=drop comment="deny NetBus"
add chain=tcp protocol=tcp dst-port=20034 action=drop comment="deny NetBus"
add chain=tcp protocol=tcp dst-port=3133 action=drop comment="deny BackOriffice"
add chain=tcp protocol=tcp dst-port=67-68 action=drop comment="deny DHCP"
```

选择 chain=udp，拒绝非法的 udp 端口:

```
add chain=udp protocol=udp dst-port=69 action=drop comment="deny TFTP"
add chain=udp protocol=udp dst-port=111 action=drop comment="deny PRC portmapper"
add chain=udp protocol=udp dst-port=135 action=drop comment="deny PRC portmapper"
add chain=udp protocol=udp dst-port=137-139 action=drop comment="deny NBT"
add chain=udp protocol=udp dst-port=2049 action=drop comment="deny NFS"
add chain=udp protocol=udp dst-port=3133 action=drop comment="deny BackOriffice"
```

选择 chain=icmp，允许需要的 icmp 连接通过，其他的丢弃：

```
add chain=icmp protocol=icmp icmp-options=0:0 action=accept
        comment="drop invalid connections"
add chain=icmp protocol=icmp icmp-options=3:0 action=accept \
        comment="allow established connections"
add chain=icmp protocol=icmp icmp-options=3:1 action=accept \
        comment="allow already established connections"
add chain=icmp protocol=icmp icmp-options=4:0 action=accept \
        comment="allow source quench"
add chain=icmp protocol=icmp icmp-options=8:0 action=accept \
        comment="allow echo request"
add chain=icmp protocol=icmp icmp-options=11:0 action=accept \
        comment="allow time exceed"
add chain=icmp protocol=icmp icmp-options=12:0 action=accept \
        comment="allow parameter bad"
add chain=icmp action=drop comment="deny all other types"
```

ICMP 类型：代码值

ICMP 协议包含 ping、traceroute、trace TTL 等网络探测参数，你可以通过配置防火墙限制或拒绝 ICMP 协议的传输。当我们需要对 ICMP 某一类参数区分限制时，需要使用到 ICMP 类型代码。

下面是 ICMP 类型列表：通常下面的 ICMP 传输建议被允许通过

Ping

- **8:0** – 回应请求
- **0:0** – 响应答复

Trace

- **11:0** – TTL 超出
- **3:3** – 端口不可到达

路径 MTU 探测

- **3:4** – 分段存储 Fragmentation-DF-Set

一般 ICMP 过滤建议：

- 允许 ping—ICMP 响应请求向外发送和响应答复进入
- 允许 traceroute—TTL 超出和端口不可到达信息进入
- 允许路径 MTU—ICMP Fragmentation-DF-Set 信息进入
- 阻止其他任何数据

源 IP 地址与目标 IP 地址

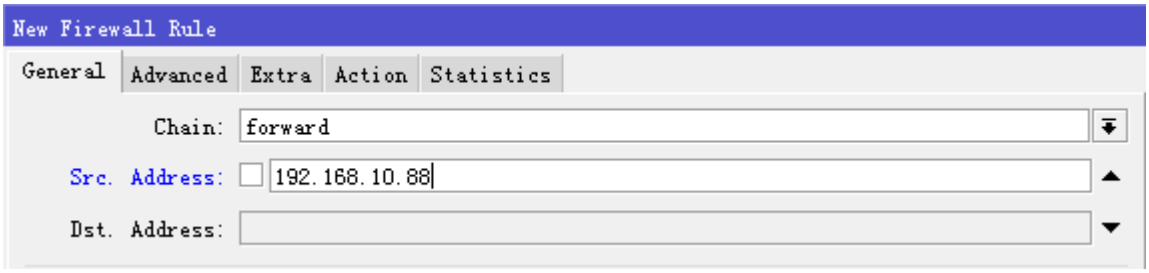
通常我们拒绝一个 IP 访问或者端口访问，仅仅过滤到他源地址/端口或目标地址/端口即可，因为单向通行已经被阻断，但如果我们是先接受后丢弃的方式，就会涉及到如何判断源地址/端口和目标地址/端口，与他们在 ip firewall filter 的链表，我们先看看下面的图：

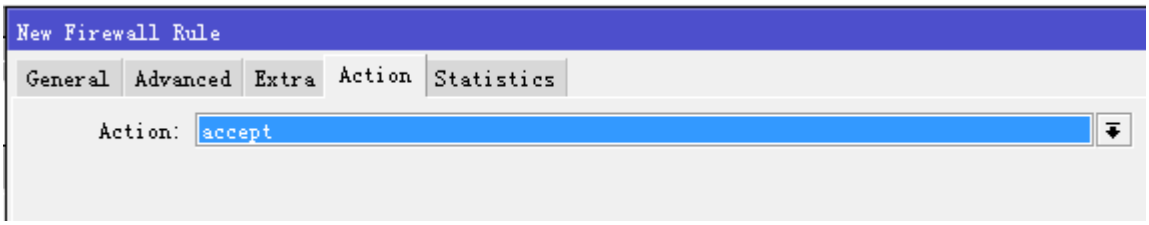
我们从该图上可以看到，内网主机 192.168.10.88 与 web 服务器 218.88.88.88 通信的情况，内网主机 192.168.10.88 向路由器请求连接，不同情况下源目标 IP 地址的转变。

当主机 192.168.10.88 请求向 web 服务器 218.88.88.88 连接，这时站在 web 服务器角度而言 192.168.10.88 是源 IP 地址，web 服务器目标地址，这个请求是主机发出的，通过路由器 forward 链表转发，web 服务器收到后会响应主机 192.168.10.88，这时 web 服务器响应发送数据变成了 web 服务器是源地址，主机是目标地址。所以在这里要记住任何通信是双向的，而不仅只有源到目标一条链路。

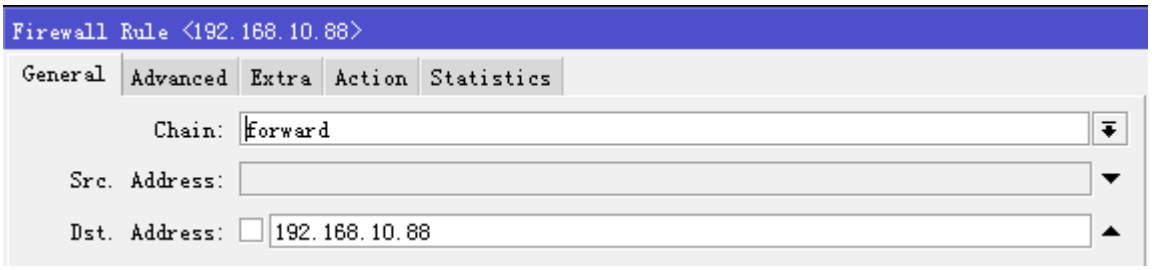
如果我们在配置先接受后丢弃方式时，我们需要允许 192.168.10.88 访问路由器，其他数据都拒绝，按照之前的双向通信的原则，我们需要配置两条 accept 规则，一条 drop 规则。

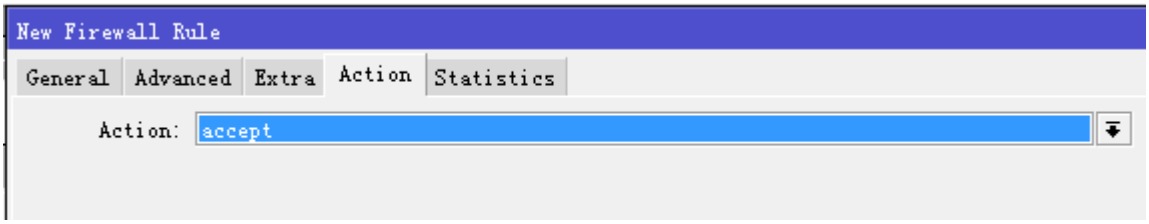
第一条接受 src-address=192.168.10.88



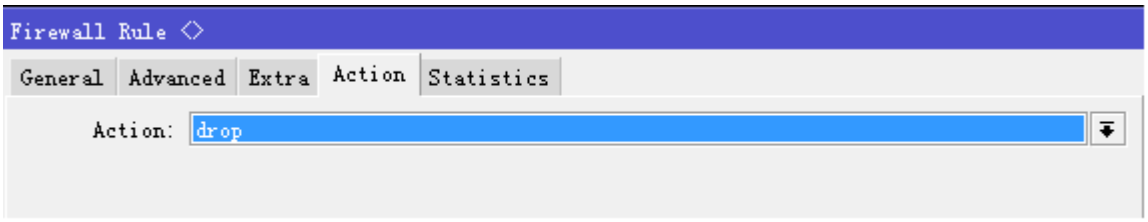


第二条规则，接受 dst-address=192.168.10.88

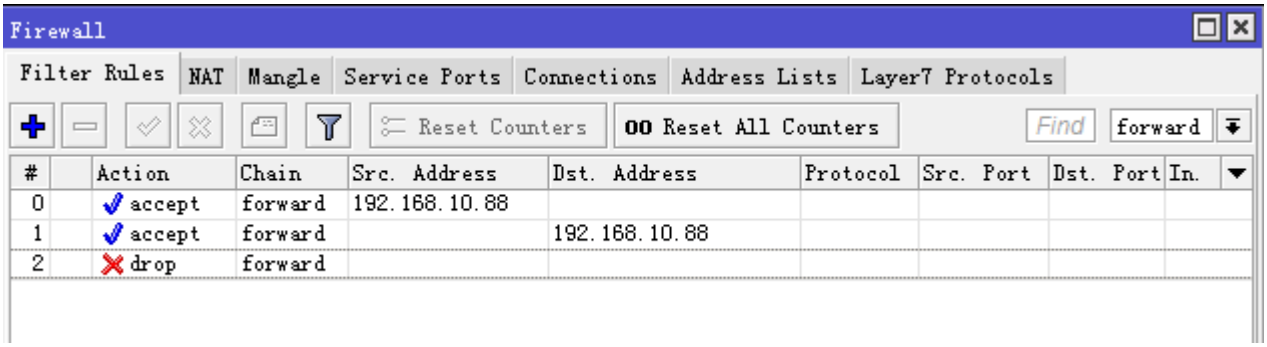




第三条规则，丢弃所有的数据，这里我们直接配置一条 action=drop 规则



规则如下：



脚本配置如下：

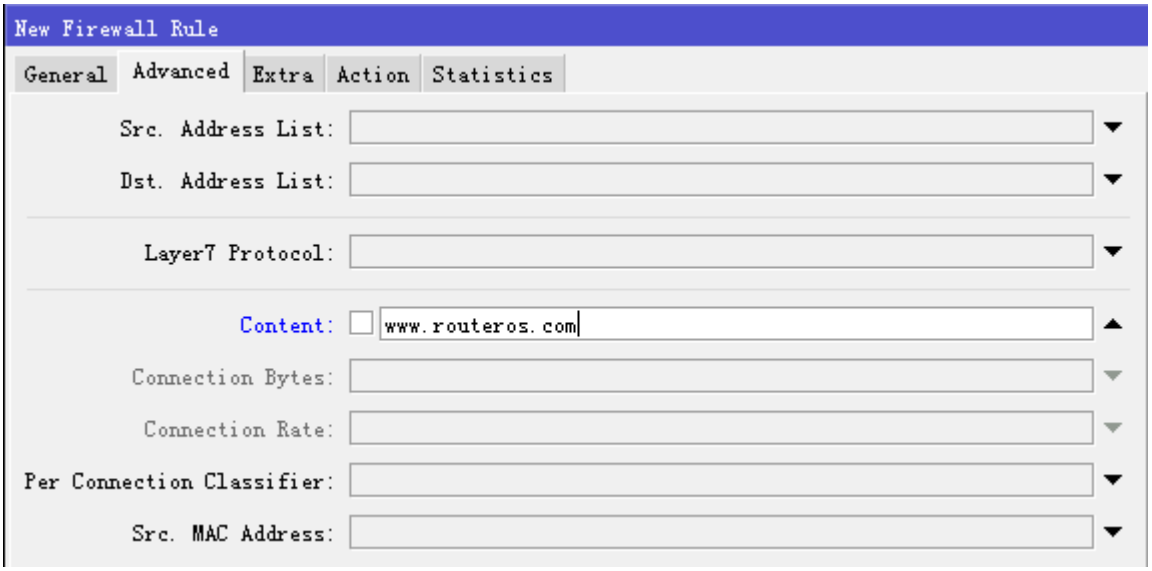
```
/ip firewall filter
add chain=forward src-address=192.168.10.88
```

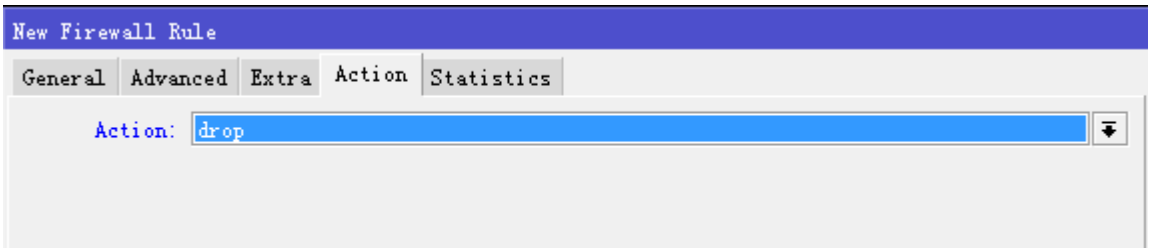
```
add chain=forward dst-address=192.168.10.88
add action=drop chain=forward
```

文本过滤

在 RouterOS 中能实现文本的内容过滤，即 content 属性设置，对一些明文传输的字符进行过滤，特别是 web 中的内容，我们可以通过 content 过滤掉一个域名或者 web 页面中的一个数字或字母的关键词。

如下我们过滤掉访问 www.routeros.com 的域名，我们添加一条规则选择链表是 forward，在 advanced 菜单下设置 content 为 www.routeros.com，设置 action=drop





配置脚本如下：

```
/ip firewall filter
add action=drop chain=forward content=www.routeros.com
```

上面是一个过滤 www.routeros.com 的域名过滤

过滤指定域名

上一篇内容提到了 content 的文本内容过滤，例如过滤 www.mikrotik.com 的域名

```
/ip firewall filter
add action=drop chain=forward content=www.mikrotik.com
```

但 content 内容过滤进对 http 内容有效，且非常简单粗暴的，因为报文中涵盖的明文内容也会过滤掉，也无法对 https 域名进行过滤。如果要过滤 https 网站域名，需要使用 tls-host 的过滤功能，配置如下：

```
/ip firewall filter
add action=drop chain=forward protocol=tcp tls-host=www.mikrotik.com
```

如果你的网络配置了 IPv6，则需要在 ipv6 防火墙下添加同样的配置

```
/ipv6 firewall filter
add action=drop chain=forward protocol=tcp tls-host= www.mikrotik.com
```

address-list 过滤方法

域名过滤还有另外一种方式，通过 dns 域名解析到的 IP 进行过滤，这个操作需要通 DNS 和 firewall address-list 来完成，要求 RouterOS 版本在 v6.36 以上。

这种方法是通过 firewall address-list 对域名进行解析，返回该域名在当前 DNS 下解析的 IP 地址，然后通过 filter 过滤 address-list 的地址列表，完成对网站的过滤访问，配置如下：

```
/ip firewall address-list
add address=www.mikrotik.com list= blockweb
add address=www.routeros.com list= blockweb
```

在 filter 下过滤网站的目标 IP 地址

```
/ip firewall filter
add action=drop chain=forward protocol=tcp dst-address-list=blockweb
```

如果有 IPv6 网络配置如下：

```
/ipv6 firewall address-list
add address=blockweb list=www.mikrotik.com
add address=blockweb list=www.routeros.com

/ipv6 firewall filter
add action=drop chain=forward protocol=tcp dst-address-list=blockweb
```

注意：配置域名到 address-list 操作虽然简单，但关键是 DNS 解析的 IP，必须和客户端的 DNS 解析的 IP 一直，所以对客户端的 DNS 管控是非常重要的，可以通过重定向客户的 DNS 请求到本地 DNS 解析。

9.3 Address-list

操作路径: /ip firewall address-list

RouterOS 在防火墙中提供了创建 IP 地址列表分组功能，address-list 可以用于 firewall filter、Mangle 和 nat 等的规则属性。提高管理对 IP 地址策略的管理和操作性，例如创建不信任的黑名单，配置源或目标路由条目的地址列表分组等。

Address-list 也可以通过动态记录到地址列表组里，如者 NAT、Mangle 和 Filter 等菜单下提供了 action=add-src-to-address-list 或 action=add-dst-to-address-list，用于动态添加地址组。

属性描述

属性	描述
address (*IP 地址/子网* | *IP-IP*; 默认:)	该属性可以设置一个独立的 IP 地址/*子网*或 IP 地址长度范围，例如'192.168.0.0-192.168.1.255'，也支持 192.168.0.0/23。
list (*字符*; 默认:)	IP 地址列表的组名称

下面事例是创建一个动态地址列表用于记录连接路由器 23 端口（telnet）的用户，并限制他们在 5 分钟内的流量。另外，添加一个静态 IP 地址 192.0.34.166/32 到列表中，永远不能访问 23 端口:

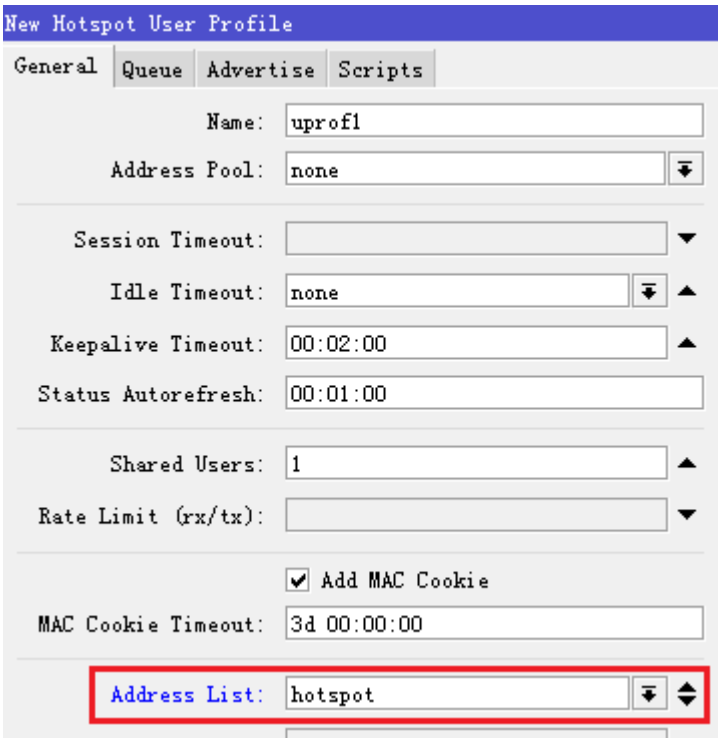
```
/ip firewall address-list add list=drop_traffic address=192.0.34.166/32
/ip firewall address-list print
Flags: X - disabled, D - dynamic
 #    LIST           ADDRESS
 0    drop_traffic 192.0.34.166

/ip firewall mangle add chain=prerouting   action=add-src-to-address-list
address-list=drop_traffic \
    address-list-timeout=5m dst-port=23 protocol=tcp

/ip firewall filter add action=drop chain=input src-address-list=drop_traffic
```

通过 address-list print 命令可以查看当前地址列表情况，如下有两个新的动态地址出现，动态 IP 地址条目前面会加上‘D‘前缀。

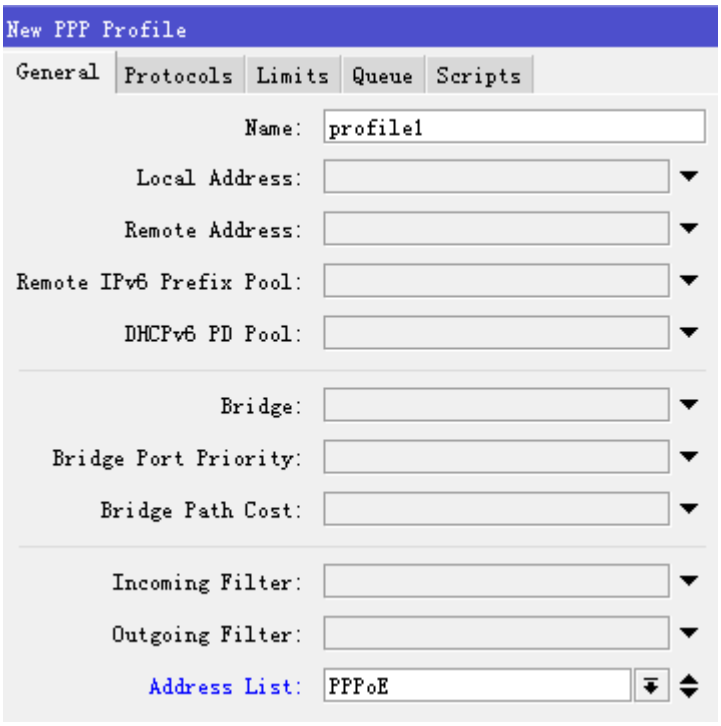
```
/ip firewall address-list print
Flags: X - disabled, D - dynamic
 #    LIST           ADDRESS
 0    drop_traffic 192.0.34.166
 1 D drop_traffic 1.1.1.1
 2 D drop_traffic 10.5.11.8
```

Address-list 还可以应用于 PPP profiles 和 Hotspot user Profiles，例如当一个用户认证成功后，会将用户获取的 IP 地址自动添加到指定的 address-list 中，用于不同的策略应用

如下面在 Hotspot User Profile 中添加一个 hotspot 地址列表的属性



下面是添加 PPP Profile 的 PPPoE 地址列表属性



在后面的章节如路由和 Queue 都会涉及到 address-list 的应用。

使用 address-list 创建黑白名单事例

在 RouterOS 通过 address-list 创建黑白名单有效的进行地址分组和管理，例如建立上网黑白名单

白名单事例

在内网有三个 IP 地址段 192.168.10.0/24、192.168.20.0/24 和 192.168.30.0/24，在 ip address 查看配置

```
[admin@MikroTik] /ip/address> print
Flags: X, D - DYNAMIC
Columns: ADDRESS, NETWORK, INTERFACE
#    ADDRESS            NETWORK         INTERFACE
0    192.168.10.2/24   192.168.10.0      ether2
1    192.168.20.2/24   192.168.20.0      ether3
2    192.168.30.2/24   192.168.30.0      ether4
```

网络中仅允许 192.168.10.0/24、192.168.20.88/32、192.168.30.88/32 的主机能访问互联网，其他 IP 地址拒绝访问，首先在 address-list 创建白名单列表

```
[admin@MikroTik]>/ip firewall address-list
[admin@MikroTik] /ip/firewall/address-list>add address=192.168.10.0/24 list="whitelist "
[admin@MikroTik] /ip/firewall/address-list>add address=192.168.10.88 list="whitelist "
[admin@MikroTik] /ip/firewall/address-list>add address=192.168.30.88 list="whitelist "
```

在 filter 下配置过滤策略，内部局域网到互联网访问是 forward 链表，首先匹配允许白名单列表通过

```
[admin@MikroTik] >/ip firewall filter
[admin@MikroTik] /ip/firewall/filter> add action=accept chain=forward \
src-address-list="whitelist "
```

再添加拒绝所有 IP 通过，注意添加顺序，防火墙是按照顺序执行，规则在优先级越高。

```
[admin@MikroTik] /ip/firewall/filter>add action=drop chain=forward
```

黑名单事例

在之前的白名单事例反过来配置，即拒绝不能上网的 IP 地址，就是黑名单的方式

禁止 192.168.20.0/24 和 192.168.30.0/24 访问互联网，创建一个 blacklist 黑名单

```
[admin@MikroTik] /ip/firewall/address-list>add address=192.168.20.0/24 list="blacklist "
[admin@MikroTik] /ip/firewall/address-list>add address=192.168.30.0/24 list="blacklist "
```

在 filter 下配置过滤策略，内部局域网到互联网访问是 forward 链表，配匹 blacklist 的 ip 地址禁止通过

```
[admin@MikroTik] /ip/firewall/filter> add action=drop chain=forward src-address-list="blacklist "
```

注意：默认情况下所有 filter 链表规则是 accept 接受通过，因此只需要添加 drop 丢弃规则，不匹配的 IP 将会被接受通过

之前我们的策略允许 192.168.20.88 和 192.168.30.88 可以访问互联网，那我们需要补充两条规则

```
[admin@MikroTik] /ip/firewall/filter>add action=accept chain=forward \
  src-address=192.168.20.88
```

```
[admin@MikroTik] /ip/firewall/filter>add action=accept chain=forward \
    src-address=192.168.30.88
```

使用 print 命令查看 filter 列表，规则 1，2 是 192.168.20.88 和 192.168.30.88 的允许规则，在规则 0 后面，肯定不会优先处理

```
[admin@ MikroTik] /ip/firewall/filter> print
Flags: X - disabled, I - invalid; D - dynamic
 0     chain=forward action=drop src-address-list=blacklist log=no
       log-prefix=""

 1     chain=forward action=accept src-address=192.168.20.88 log=no
       log-prefix=""

 2     chain=forward action=accept src-address=192.168.30.88 log=no
       log-prefix=""
```

将规则 1，2 移动到 0 之前优先处理，通过 move 命令调整顺序

```
[admin@MikroTik] /ip/firewall/filter> move 1,2 0
```

查看修改情况：

```
[admin@ MikroTik] /ip/firewall/filter> print
Flags: X - disabled, I - invalid; D - dynamic

 0     chain=forward action=accept src-address=192.168.20.88 log=no
       log-prefix=""

 1     chain=forward action=accept src-address=192.168.30.88 log=no
       log-prefix=""
  2     chain=forward action=drop src-address-list=blacklist log=no
       log-prefix=""
```

当然，192.168.20.88 和 192.168.30.88 可以创建一个 whitelist 中，添加一条 src-address-list 规则完成，这样适合较多的白名单 IP 地址情况。

v6.36 允许 address-list 添加域名

Address-list 从 RouterOS v6.36 后， 允许 address 属性支持域名，即方便我们对域名网站的 IP 控制，下面我们通过一个实例为具体配置情况

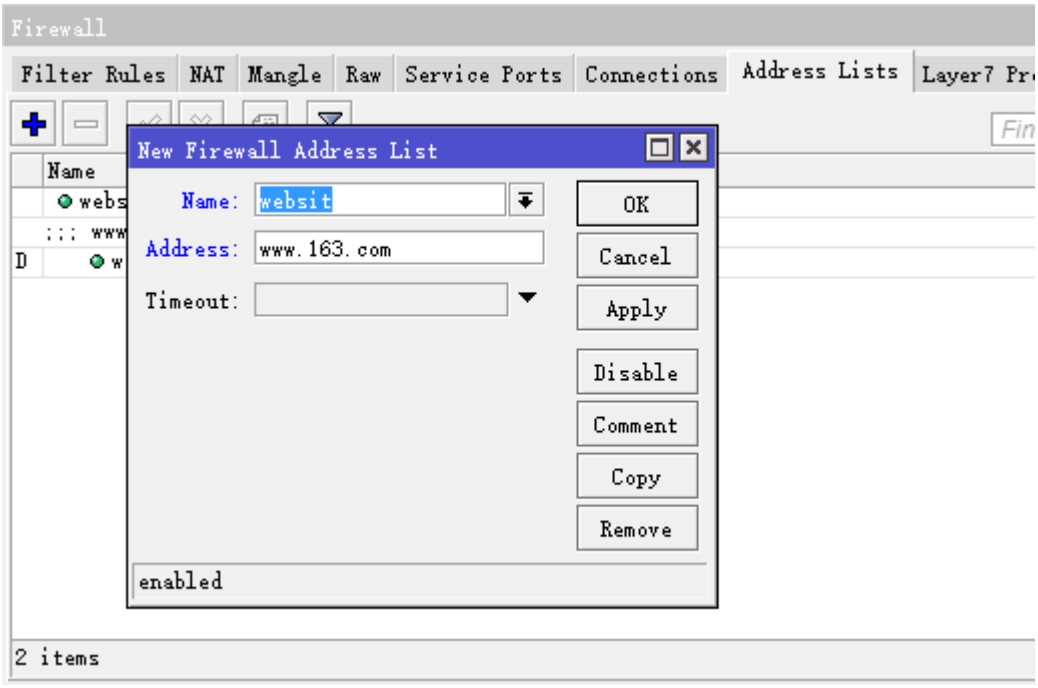
下面我们创建一个 websit 的列表，并添加 address=www.mikrotik.com

```
[admin@MikroTik] /ip firewall address-list> add list=websit address=www.mikrotik.com
[admin@MikroTik-J] /ip firewall address-list> print
Flags: X - disabled, D - dynamic
```

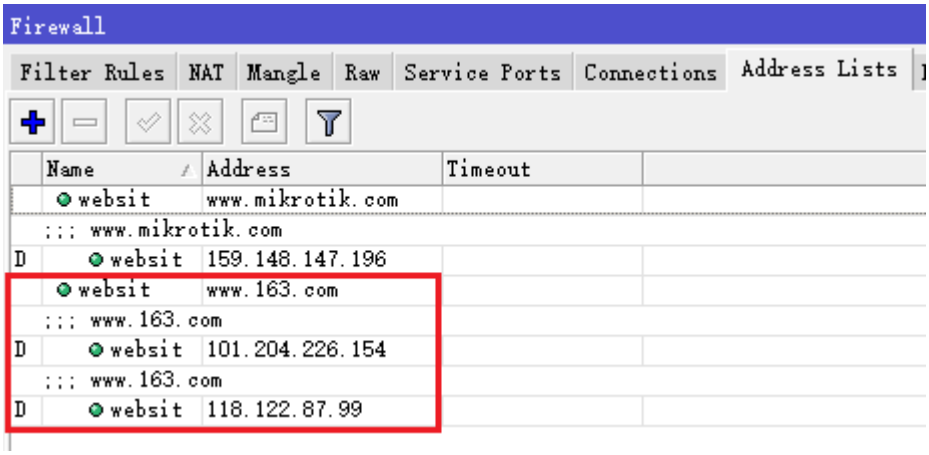
```
#     LIST                              ADDRESS                 TIMEOUT
0     websit                            www.mikrotik.com
1 D ;;; www.mikrotik.com
      websit                            159.148.147.196
```

如上面操作所示，我们添加完成域名后，在 address-list 查看到两条规则一条是我们配置的静态 www.mikrotik.com 域名，一条是动态解析 www.mikrotik.com 网站域名的 IP 地址

下面我们通过 winbox 添加一个 www.163.com 的域名，



在添加完成后，www.163.com 会自动解析两个动态条目的 IP：



注意：设置地址列表域名条目的前提是你的 RouterOS 已经配置 DNS 服务器（操作路径/ip dns）

9.4 Interface list 事例

在 v6.36 版本新增了一个属性 interface list，即接口列表，允许我们在 RouterOS 的 interface 中做一个接口的列表分类，用于 firewall 规则的调用，简化了多接口规则的配置。

如下面我们需要创建一个 lan 接口的列表分类，首先我们进入 interface list 创建一个新的列表名 lan

```
[admin@MikroTik-J] /interface list> add name=lan
[admin@MikroTik-J] /interface list> print
 # NAME
 0 all
 1 lan
```

列表名创建完成后，我们进入 member 菜单下，添加 lan 列表的接口成员

```
[admin@MikroTik-J] /interface list> member
[admin@MikroTik-J] /interface list member> add list=lan interface=ether4
[admin@MikroTik-J] /interface list member> add list=lan interface=ether5
```

查看接口成员添加情况

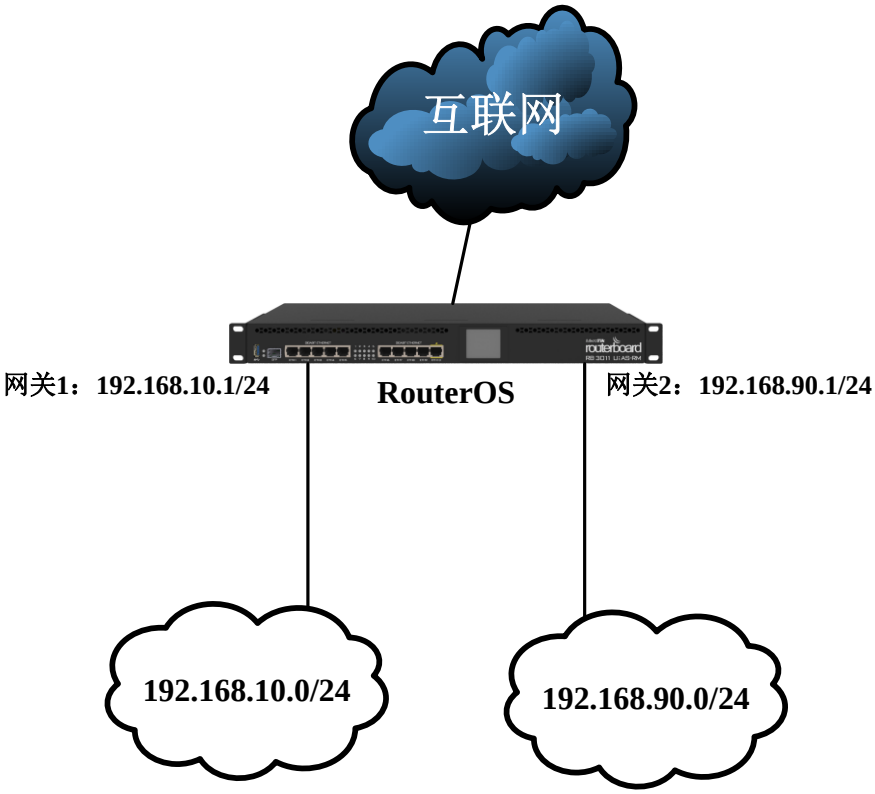
```
[admin@MikroTik-J] /interface list member> print
Flags: X - disabled, D - dynamic
 #    LIST                                INTERFACE
 0    lan                                 ether4
 1    lan                                 ether5
```

接着对 lan 接口列表下的成员限制 icmp 访问，进入/ip firewall filter

```
[admin@MikroTik-J] /interface list member> /ip firewall filter
[admin@MikroTik-J] /ip firewall filter> add chain=input in-interface-list=lan protocol=icmp
action=drop
[admin@MikroTik-J] /ip firewall filter> print
Flags: X - disabled, I - invalid, D - dynamic
 0     chain=input action=drop protocol=icmp in-interface-list=lan
```

9.5 内网多 IP 段访问控制

当 RouterOS 作为网关，并在 RouterOS 的内口配置多个 IP 地址段（通常用于 VLAN 区分不同的办公段、核心设备或区域划分）连接多个 IP 子网段，当我们需要控制多个 IP 网段之间的访问时，需通过防火墙 filter 来完成。通过下面一个简单实例介绍：



RouterOS 内网有 192.168.10.0/24 和 192.168.90.0/24 两个网段，网关都配置在 RouterOS 内口 ether2 和 ether3，IP 地址配置：

```
/ip address
add address=192.168.10.1/24 interface=ether2
add address=192.168.90.1/24 interface=ether3
```

由于两个网段涉及特殊业务，不能相互访问，需要通过防火墙 filter 控制两个段的互访问

控制 IP 段访问，进入 ip firewall filter 中添加一条 forward 规则为：

```
/ip firewall filter add chain=forward src-address=192.168.10.0/24 dst-address=192.168.90.0/24
action=drop
```

但在网络里希望 192.168.10.8/32 的主机访问 192.168.90.0/24 的网络

```
/ip firewall filter add chain=forward src-address=192.168.10.8/32 dst-address=192.168.90.0/24
action=accept
```

查看 fliter 的配置：

```
[admin@MikroTik] /ip firewall filter> print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=forward action=drop src-address=192.168.10.0/24 dst-address=192.168.90.0/24
log=no log-prefix=""
```

```
 1   chain=forward action=accept src-address=192.168.10.8/32 dst-address=192.168.90.0/24
log=no log-prefix=""
```

通过 move 命令将允许 192.168.10.8 主机访问 192.168.90.0/24 段规则移动到序列最上，即”0”，优先执行

```
[admin@MikroTik] /ip firewall filter> move 1 0
[admin@MikroTik] /ip firewall filter> print
Flags: X - disabled, I - invalid, D - dynamic
0      chain=forward action=accept src-address=192.168.10.8/32 dst-address=192.168.90.0/24
log=no log-prefix=""

1      chain=forward action=drop src-address=192.168.10.0/24 dst-address=192.168.90.0/24
log=no log-prefix=""
```

两个 IP 网络段单向访问控制

根据前面的拓扑，网络要求 192.168.10.0/24 能访问 192.168.90.0/24，但 192.168.90.0/24 不能访问 192.168.10.0/24，这样的设置涉及到连接状态控制

首先允许已经建立连接和相关连接通过

```
/ip firewall filter add action=accept chain=forward connection-state=established,related
```

创建允许 192.168.10.0/24 网段能访问任意网络

```
/ip firewall filter add action=accept chain=forward src-address=192.168.10.0/24
```

关键的配置在这里，禁止 192.168.90.0/24 的网络向 192.168.10.0/24 网络发起新的网络请求

```
/ip firewall filter add action=drop chain=forward connection-state=new
src-address=192.168.90.0/24 dst-address=192.168.10.0/24
```

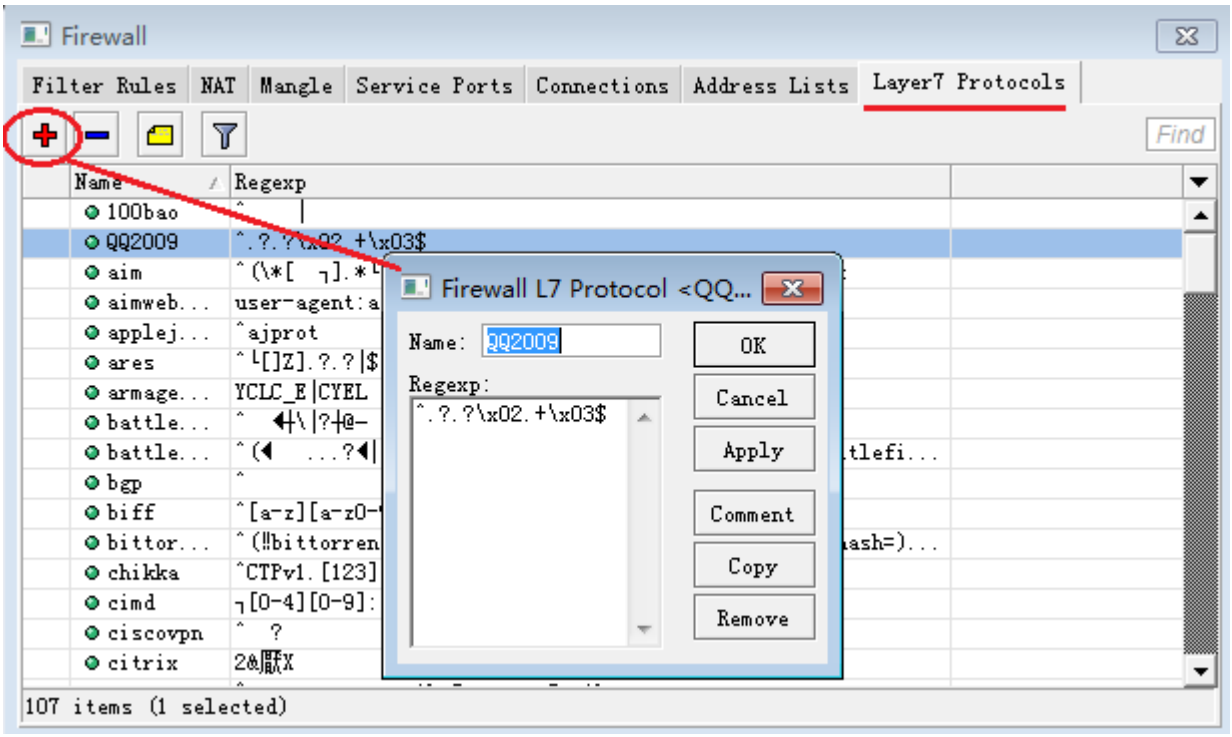
注意：该配置仅对 TCP 连接生效。

9.6 RouterOS L7 协议

RouterOS 在防火墙中增加了一个新得功能——7 层协议过滤。针对一些应用程序如 skype、MSN、魔兽世界等等提供了脚本。

在防火墙 Layer7-protocol 目录下，你可以添加正则表达式字符串协议，定义他们的名称，并在防火墙 filter 目录下丢弃他们的数据。这个功能将不会查看单独的数据报，会查看整个数据的相关连接，收集数据直到第 10 个数据报或者 2kb 数据，来执行第一次操作

下面介绍一下具体方法的使用：7 层协议过滤增加在 ip firewall 中 Layer7 Protocols，我们可以在下面的图中看到：



7 层协议通过 Regexp 脚本编写相应应用程序的过滤代码，Regexp 可以通过网上搜索相关数据了解。在这里我们已经提供了一些常用程序的 7 层协议脚本：

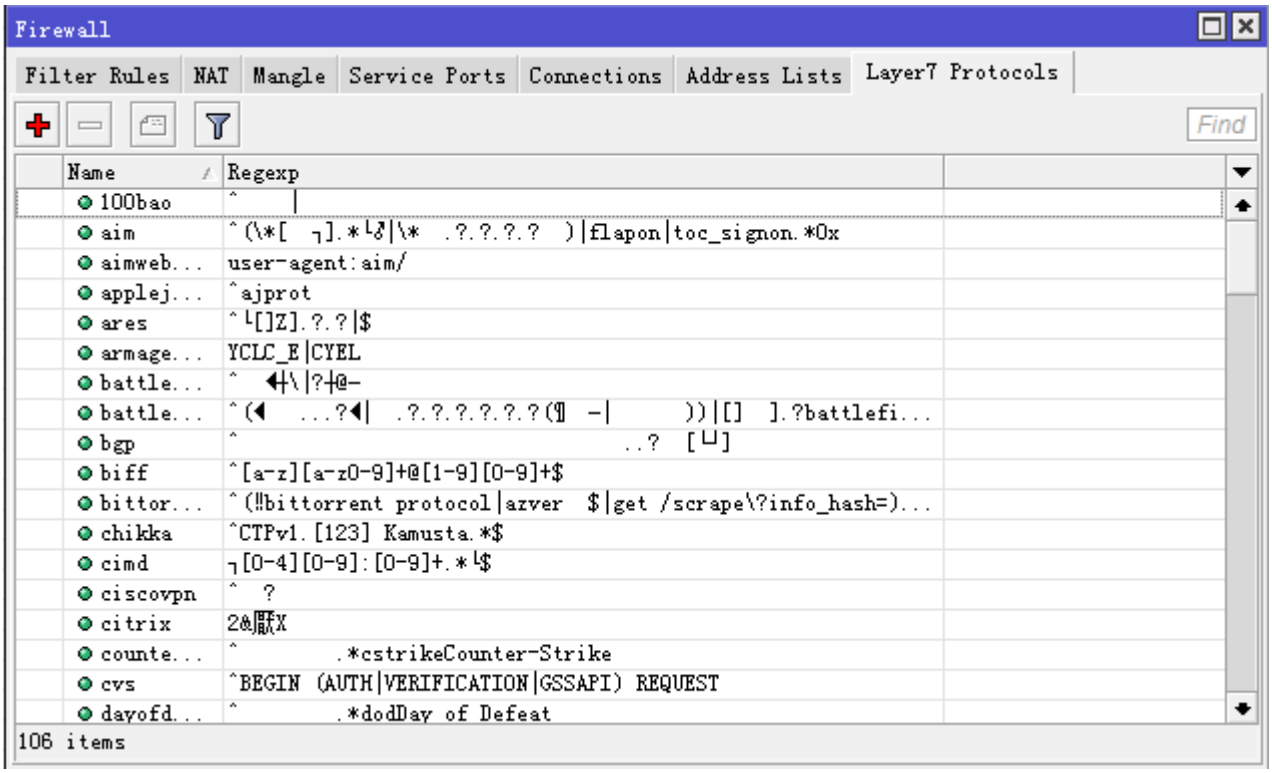
通过在 http://wiki.mikrotik.com 下载 L7 层协议过滤脚本。然后我们可以通过 FTP 上传或者直接拖放到 Files 对话框中。

之后我们在命令行(Terminal)中导入 7 层协议脚本，用 import l7-protos.rsc 命令来导入脚本

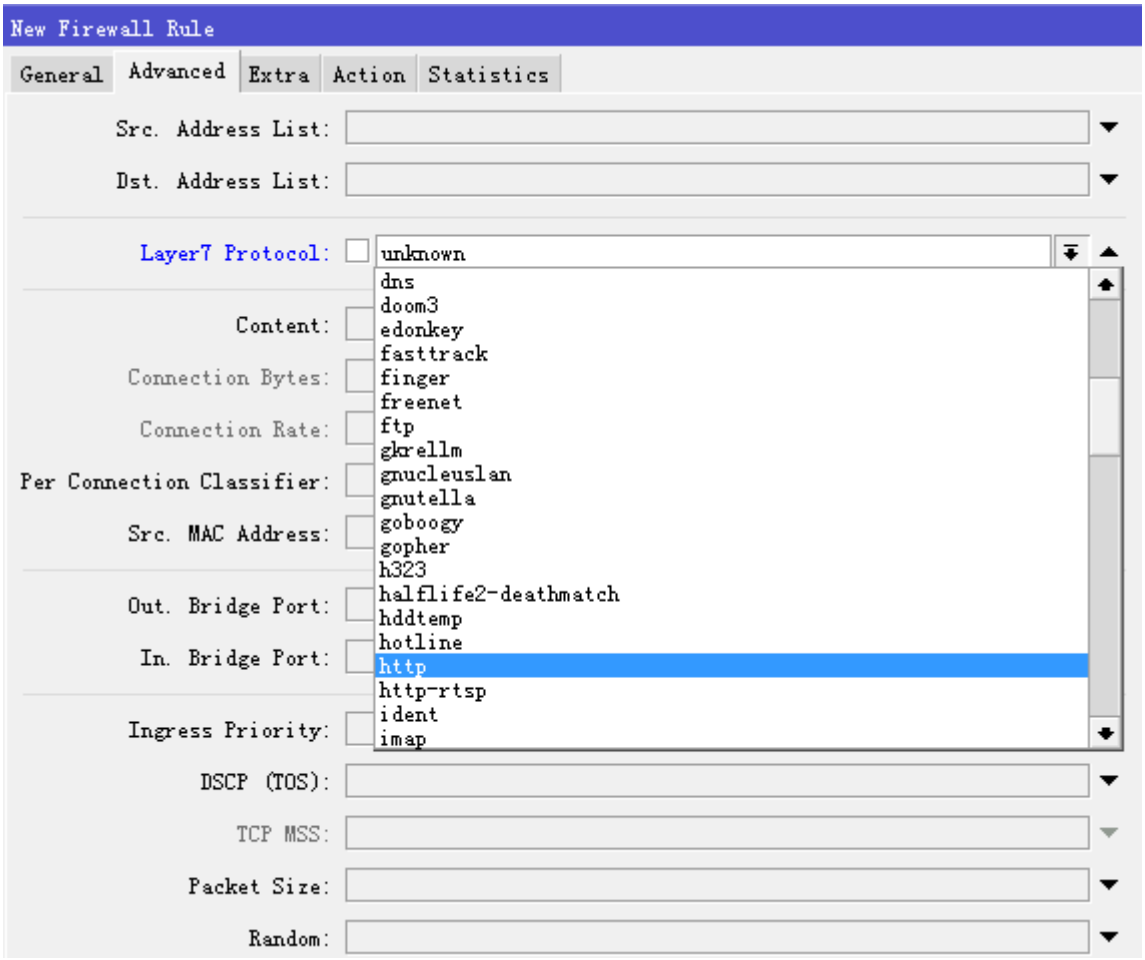
```
[admin@MikroTik] > import l7-protos.rsc
Opening script file l7-protos.rsc
Script file loaded and executed successfully
[admin@MikroTik] >
```

当系统提示 Script file loaded and executed successfully，说明脚本成功导入。

导入脚本后，我们可以在 Layer7 Protocols 中看到



导入后，我们就可以在 ip firewall 中通过 Layer7 Protocols 参数调用，并做相应的规则处理，下面是一个在防火墙得 Filter Rules 里面调用 L7 脚本



在调用 http 的 L7 代码后，根据你的需要选择 action 的操作方式，其他的操作也同以上设置类似，如果需要对 IP 地址做控制，可以设置 src-address 或者 dst-address 等参数。

L7 代码控制某聊天软件登录

例如企鹅聊天软件，登录通常默认采用 UDP/8000 端口，当 UDP/8000 端口被禁止，会选择 TCP 的 SSL，也就是 TCP 的 443 端口连接，禁用该聊天软件必须两种方式都启用，下面是 UDP 和 TCP 两种登录方式的 L7 代码（仅能命令行导入，不能直接复制到 winbox　L7 代码区域）

```
/ip firewall layer7-protocol
add name=OICQ_udp regexp="^\\x02\\x36\\x61.+\\x03\$"
add name=OICQ_ssl regexp="^.\?.\?\\x02\\x36\\x61.+\\x03\$"
```

添加 L7 代码后，应用到防火墙的 filter 过滤规则的 forward 链表

```
/ip firewall filter
add action=drop chain=forward disabled=no layer7-protocol=OICQ_ssl
add action=drop chain=forward disabled=no layer7-protocol=OICQ_udp
```

以上代码仅供测试和参考使用

9.7 使用 wireshark 分析网页视频

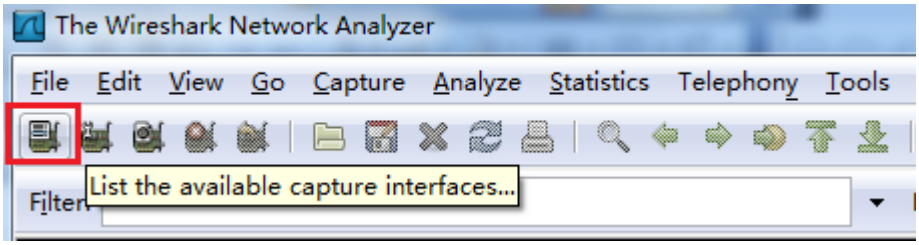
Wireshark 是非常著名的数据分析软件，我们首先需要下载 wireshark 并安装，具体 wireshark 的安装和详细操作请 baidu 或 google，下面简单介绍下 wireshark 如何抓包分析网页视频特征，并写出 L7 的代码。

注意：该分析内容仅供参考，可能会涉及后期相关内容变更。

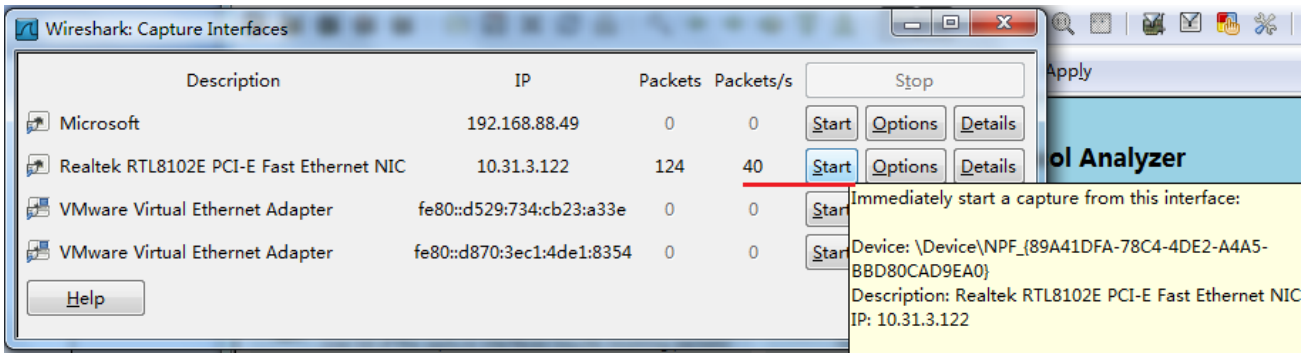
操作流程：

1、 启动 wireshark，并找到需要抓取数据报的网卡，准备抓取。
2、 启动浏览器，并打开需要浏览的视频页面
3、 在 wireshark capture interface 点击 start 开始抓取
4、 视频浏览一定时间后，停止 wireshark 抓取，并分析数据，输入 http.request　get 查找请求获取视频的连结，一般都是以.swf 结尾
5、 分析该类型连接的数据，多次对比，找到相同点，并编写 L7 正则表达式

第一步、启用 Wireshark，并选择抓取数据的接口



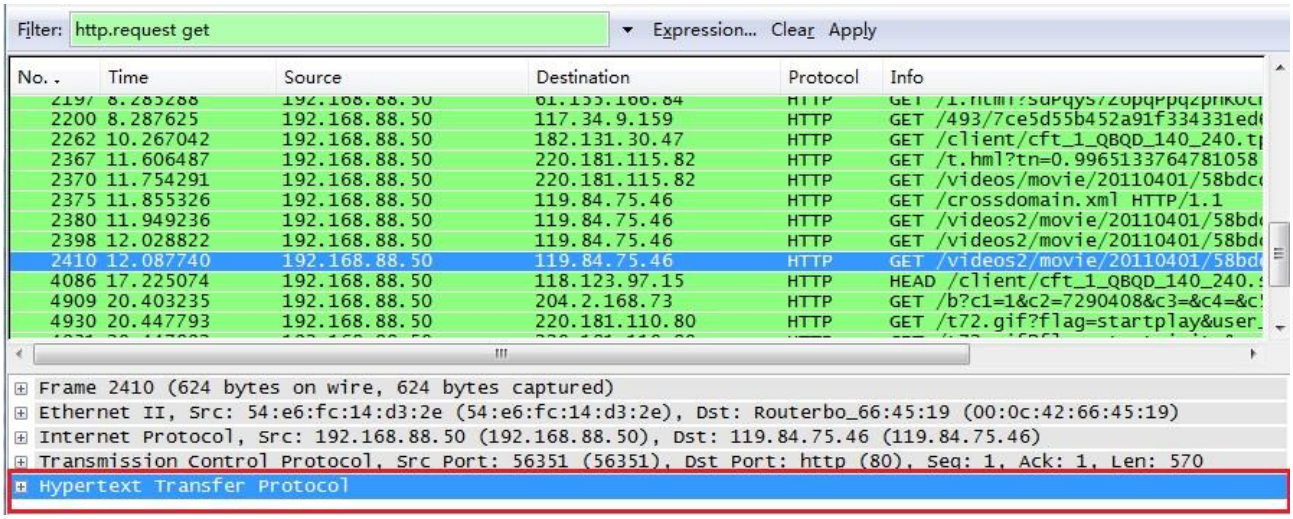
通过清单可以选择我们需要抓取数据报的网卡



第二步、开启浏览器，但不要播放视频，这里我们以 www.qiyi.com 奇艺视频网站作为案例（该案例仅供如何学习抓包分析参考，不代表实际能使用），

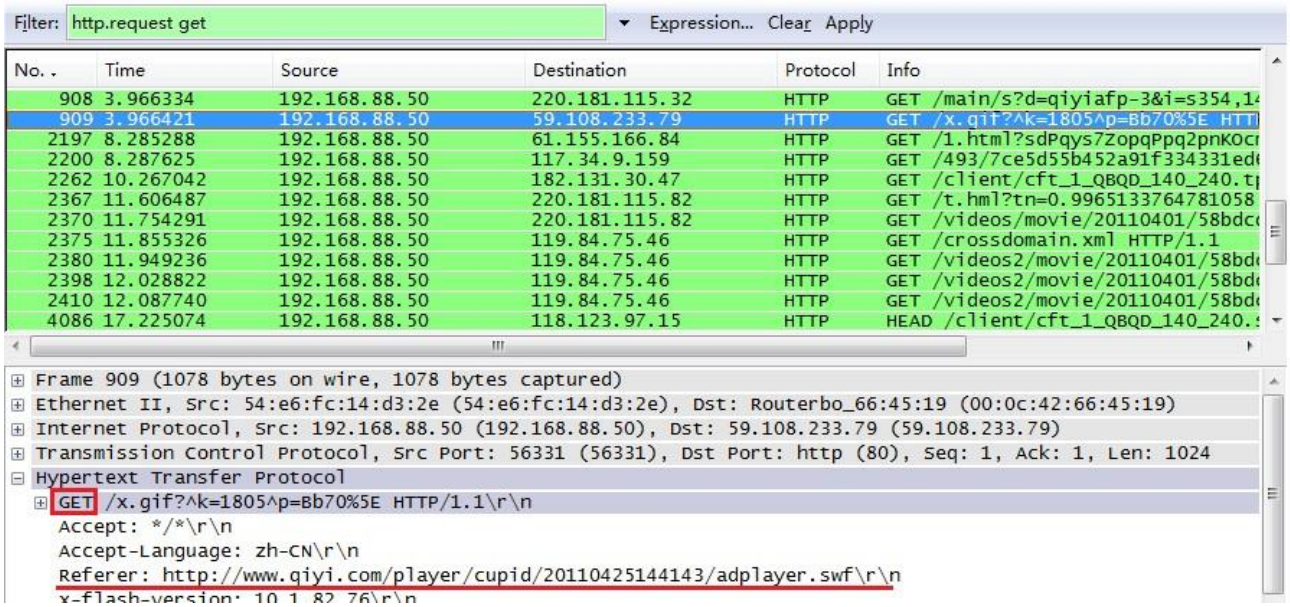
第三步、先点击 start 按钮开始抓取，然后点击播放网页视频

第四步、视频播放到一段时间后，我们可以停止抓取数据报，分析数据报最主要的是刚开始主机向视频服务器申请获取视频的 get 或 post 参数，我们可以使用 http.request get 来查找，如下图



我们分析 L7 数据是从应用层开始，不管是游戏、下载还是视频，都不用考虑他们的 Frame、Ethernet、Internet Protocol 和传输方式（TCP 或者 UDP），只从传输协议后分析，如视频，我们分析的是 HTTP 传输协议

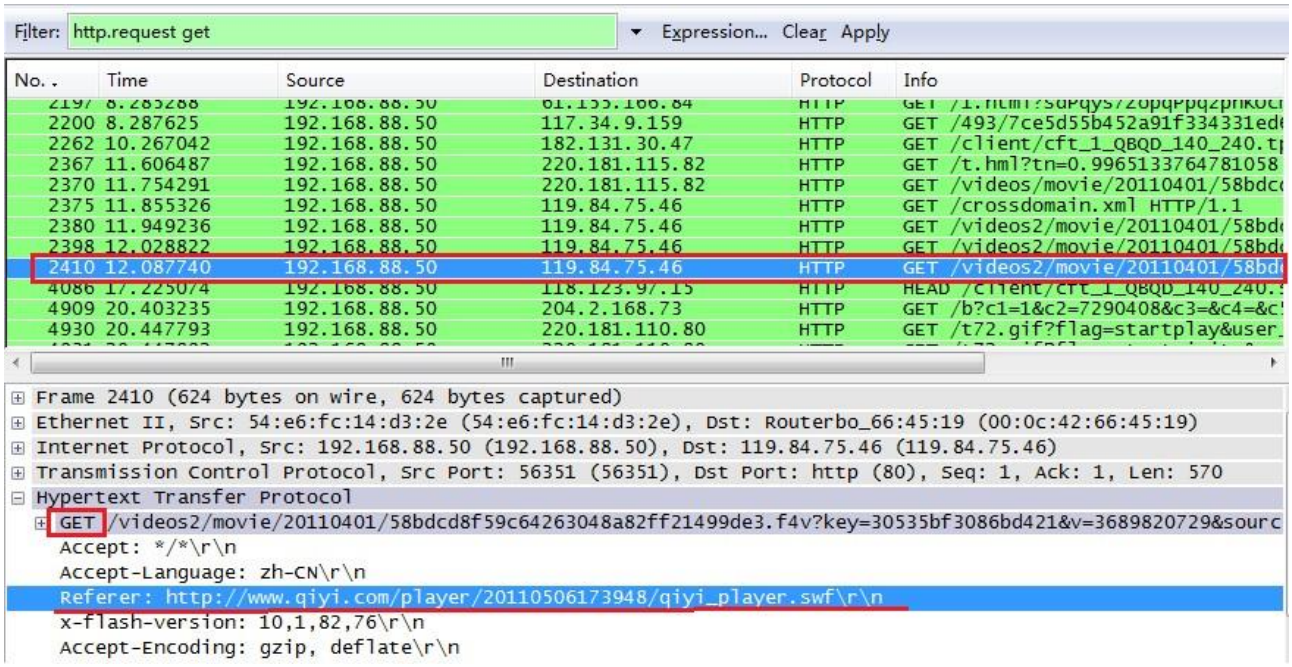
当然我们要找的是视频档，如.swf 结尾的内容（有些可能是 mp4 之类），其他 jpg、js、png 这些都不用考虑。上图，我们找到了一个相关匹配的 GET 值，我们看到主机向 119.84.75.46 的服务器获取一个.swf 的 flash 视频档，



内容如下：

```
Referer: http://www.xxxxx.com/player/cupid/20110425144143/adplayer.swf\r\n
```

继续找到相关的内容：



这次我们获取到的有所不同

```
Referer: http://www.xxxxx.com/player/20110506173948/qiyi_player.swf\r\n
```

我们在用这样的方法对比多次这种视频的 GET 信息，几乎他们的连接内容都有以上的共同点，

这样我们就可以开始抓取关键词，www.xxxxx.com/player 这个字段是固定的，然后是 adplayer.swf 或者 qiyi_player.swf，这里我们只取.swf，那我们要包含的关键词有

```
GET + www.xxxxx.com/player + .swf
```

最后我们写出以下 L7 的正则表达式代码

```
^(get|post).+\www\.\xxxxx\.\com\/\player.+\.swf
```

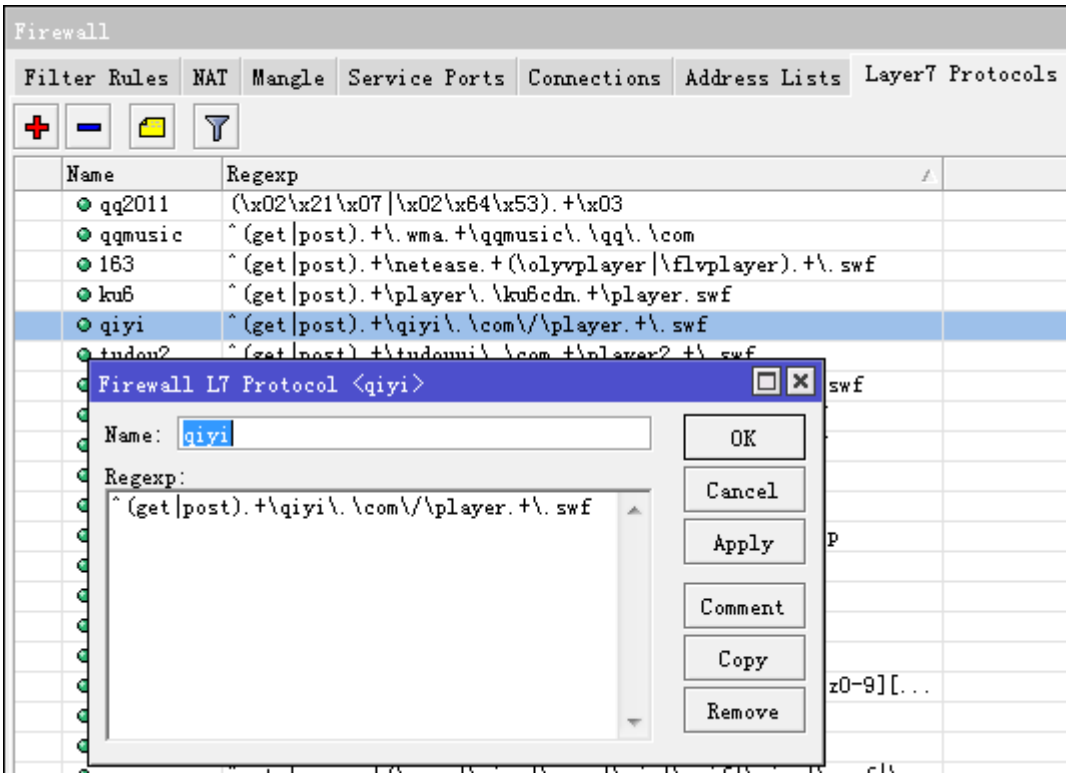
代码分析：

```
^ ：                标示内容的起始
(get|post) ：  中间的“|”标示 get 或者 post
.+ ：               中间任意字符
.? ：               任意一个字符
\ ：                将内容转意为字符
```

最后我将代码简化为

```
^(get|post).+\xxxxx\.\com\/\player.+\.swf
```

这里没有包含 www.是因为可能服务器会更换其他的二级域名的可能，我们将代码添加入/ip firewall layer7-protocol



剩下的就是你需要在 ip firewall filter 里做防火墙过滤，还是在 ip firewall mangle 里做流控等操作了

9.8 DMZ 配置事例

DMZ 是英文“demilitarized zone”的缩写，中文名称为“隔离区”，也称“非军事化区”。它是为了解决安装防火墙后外部网络不能访问内部网络服务器的问题，而设立的一个非安全系统与安全系统之间的缓冲区，这个缓冲区位于

企业内部网络和外部网络之间的小网络区域内，在这个小网络区域内可以放置一些必须公开的服务器设施，如企业 Web 服务器、FTP 服务器和论坛等。另一方面，通过这样一个 DMZ 区域，更加有效地保护了内部网络，因为这种网络部署，比起一般的防火墙方案，对攻击者来说又多了一道关卡。

路由器一般需要 3 张网卡（Public 公网，Local 本地网络，DMZ-Zone 非军事区）：

```
[admin@gateway] interface> print
Flags: X - disabled, D - dynamic, R - running
 #     NAME                TYPE            RX-RATE     TX-RATE     MTU
 0   R Public              ether           0           0           1500
 1   R Local               ether           0           0           1500
 2   R DMZ-zone            ether           0           0           1500
[admin@gateway] interface>
```

- 给相应的 Interface 添加对应的 IP 地址，如下：

```
[admin@gateway] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
#    ADDRESS             NETWORK         BROADCAST       INTERFACE
0    192.168.0.2/24      192.168.0.0     192.168.0.255   Public
 1    10.0.0.254/24       10.0.0.0        10.0.0.255      Local
 2    10.1.0.1/32         10.1.0.2        10.1.0.2        DMZ-zone
 3    192.168.0.3/24      192.168.0.0     192.168.0.255   Public
[admin@gateway] ip address>
```

- 添加静态默认路由到本地路由器上

```
[admin@MikroTik] ip route> print
Flags: X - disabled, I - invalid, D - dynamic, J - rejected,
C - connect, S - static, r - rip, o - ospf, b - bgp
 #     DST-ADDRESS         G GATEWAY          DISTANCE INTERFACE
 0   S 0.0.0.0/0           r 10.0.0.254       1        ether1
 1 DC 10.0.0.0/24          r 0.0.0.0          0        ether1
[admin@MikroTik] ip route>
```

- **配置 DMZ 服务器的 IP 地址为 IP 地址 10.1.0.2，网络地址段 10.1.0.0/24，以及网关 10.1.0.1**
- **配置能从因特网访问 DMZ 服务的 dst-nat 规则，将地址 192.168.0.3 配置给 DMZ 服务器：**

```
[admin@gateway] ip firewall nat> add chain=dst-nat action=dst-nat \
\... dst-address=192.168.0.3 to-dst-address=10.1.0.2
[admin@gateway] ip firewall dst-nat> print
Flags: X - disabled, I - invalid, D – dynamic
0   Chain=dst-nat dst-address=192.168.0.3 action=dst-nat to-dst-address=10.1.0.2
[admin@gateway] ip firewall nat>
```

9.9 RouterOS ip settings

操作路径: /ip settings

RouterOS v6 版本新增了三层的设置菜单，**ip settings** 允许设置 TCP/IP 相关的核心参数，RouterOS 内核基于 Linux，所以这些参数都来至于 Linux，即与 Linux 下的/etc/sysctl.conf 配置文件相似。

属性	描述
accept-redirects (*yes* | *no*; 默认: **no**)	是否接受 ICMP 重定向信息，通常情况下在主机端启用，而路由器禁用
accept-source-route (*yes* | *no*; 默认: **no**)	是否接受数据报的 SRR 选项，通常情况下是在路由器上启用
allow-fast-path (*yes* | *no*; 默认: **yes**)	是否开启 fast path
arp-timeout (时间周期; 默认: 30s)	ARP timeout 属性设置是应用于全局接口的 ARP 设置，能控制所有接口的 APR 超时时间（6.36 后 arp timeout 属性可以单独控制所有接口）参数可以单位可以选 ms, s, m, h, d，即毫秒，秒，分，小时和天，如果没有设置单位，默认为 s（秒）。
ip-forwarding (*yes* | *no*; 默认: **yes**)	启用或禁止 IP 路由数据报在接口之间转发。默认开启该功能。如果需要关闭，请谨慎考虑是否需要做路由转发。
rp_filter (*loose* | *no* | *strict*; 默认: **no**)	禁用或启用源地址效验，即是否打开反向路由过滤功能，用于对源地址欺骗的网络攻击进行防御。 • no – 不启用源地址效验 • strict – Strict 模式被定义在 RFC3704，每个进入的数据包都要根据 FIB 进行测试，如果接口不是最好的反向路由，数据包检查就会失败。默认情况下，失败的包被丢弃 • loose – Loose 模式被定义在 RFC3704 ，即宽松逆向路由。每个进入的数据包源地址会根据 FIB 进行检查，如果通过任何接口都无法访问源地址，则包检查将失败。 RFC3704 中推荐是启用 strict 模式来防止 DDoS 攻击中的 IP 欺骗。如果使用非对称路由或其他复杂的路由或 VRRP，则建议使用 loose 模式。
secure-redirects (*yes* | *no*; 默认: **yes**)	允许在网关情况下重定向 ICMP 信息
send-redirects (*yes* | *no*; 默认: **yes**)	是否发送 ICMP 重定向，建议在路由器上启用。
tcp_syncookies (*yes* | *no*; 默认: **no**)	当 syn 缓存队列的接口溢出，将发送 syncookies 。该功能是防御 syn 洪水攻击。
max-neighbor-entries *(整型 [0..4294967295]; 默认:)*	ARP 表中允许的最大主机数。从 RouterOS 7.1 版本开始，默认值取决于 RAM 的大小。可以设置比默认更高的值，但这会增加内存不足的风险。 主机数量与 RAM 系统默认:

	• 64 MB - 2048 • 128 MB - 4096 • 256 MB- 8192 • 512 MB 或更高 - 16384
route-cache (yes | no; 默认: yes)	禁用或启用 linux 路由缓存

改动这些参数时涉及到 IP 核心协议的修改，请谨慎操作！也可以参考相关的 Linux sysctl.conf 档对比。

A maximum number of allowed neighbors in the ARP table. Since RouterOS version 7.1, the default value depends on the installed amount of RAM. It is possible to set a higher value than the default, but it increases the risk of out of memory condition.

The default values for certain RAM sizes:

- 2048 for 64 MB,
- 4096 for 128 MB,
- 8192 for 256 MB,
- 16384 for 512 MB or higher.

只读属性

属性	描述
ipv4-fast-path-active (*yes* | *no*)	显示是否成功启用 fast-path
ipv4-fast-path-bytes (*整型*)	统计 fast-path 的字节
ipv4-fast-path-packets (*整型*)	统计 fast-path 的数据报
ipv4-fasttrack-active (*yes* | *no*)	显示是否成功启用 fasttrack
ipv4-fasttrack-bytes (*整型*)	统计 fasttrack 的字节
ipv4-fasttrack-packets (*整型*)	统计 fasttrack 第数据报

9.10 DoS 防御

DoS (Denial of Service),即拒绝服务，这种攻击是利用 TCP/IP 协议不断发生无效的 syn 请求，让路由器或服务器超负荷， 导致 CPU 使用率 100%，路由器回应变得非常缓慢，甚至无响应。由于攻击者发送了大量的 syn 数据报，都属于小包（在 RouterBOARD Throughput 章节介绍了路由器对包的处理），数据流都会经过 firewall（filter, NAT, mangle)，这样每秒到达路由器的数据报造成 CPU 处理超载，攻击者还可以利用分散的被感染主机做 DDoS（Distributed Denial of Service）分布式拒绝服务攻击。

通常我们没有完美的解决方法免受 DoS 攻击，但我们尽量减少攻击对路由器的影响，对于 RouterOS 而言，x86 平台数据处理完全靠 CPU，如果 CPU 处理能力越强，抗攻击能力也会增强。对于 RB 或 CCR 等路由器则是吞吐量指标，也是衡量其处理性能的一点。下面是几点建议：

- 使用性能更强的路由器或服务器
- 配置性能更好的以太网卡，有足够的上联带宽
- 减少防火墙、queue 和其他数据报处理规则
- 跟踪攻击源数据路径，并阻止攻击或关闭攻击源（找上级网络提供商协助处理）
- v6.36 后，RouterOS 新增 RAW 功能，指定 IP 不经过连接跟踪降低 CPU 消耗，参考 10.8 章节

诊断 SYN 攻击

攻击正是用最多的是 TCP SYN 洪水攻击，这种利用 TCP 协议的三步握手协议发起攻击，对于问题诊断如下

通过 connection 查看 syn-sent 状态是否大量出现

```
/ip firewall connection print
```

查看 CPU 利用率是否比平常高，或达到 100%，如果遭受攻击可能无法使用远程连接到路由器，请使用本地终端或显示器查看

```
/system resource monitor
```

通过 torch 查看可疑连接，如果有必要可以在来源方向使用交换机做端口镜像分析抓包

```
/tool torch
```

9.11 保护路由器

限制连接

一个 IP 地址出现太多的连接可以添加到黑名单（black-list），可以使用 connection-limit 配合 address-list 完成，如下限制每台主机的连接数，超过后将攻击者主机 IP 添加到 blocker-addr 清单，保存一天：

```
/ip firewall filter add chain=input protocol=tcp connection-limit=（100 或更高）,32  \
action=add-src-to-address-list  address-list=blocked-addr address-list-timeout=1d
```

注意该规则的 connection-limit 没有设置固定值，因为需要考虑到一些多连接请求，如 HTTP、BT、迅雷和其他 P2P 连接问题，所以需要根据自己的实际环境考虑，该规则默认连接数为 100，也可以设置更高。

Action=tarpit(压制)

利用 tarpit 参数压制攻击，代替简单 drop 攻击者数据报（action=drop），如果一台处理性能强大的路由器抓住这些连接，并 hold 住，让其连接数不超过 3 个。

```
/ip firewall filter add chain=input protocol=tcp src-address-list=blocked-addr \
connection-limit=3,32 action=tarpit
```

SYN 过滤

另外一种方式是限制先建立的 syn 请求，对新建的 syn 请求，通过 jump 建立一个自定义规则组 SYN-Protect，判断是 tcp-flags=syn，且 conneciton-state=new 的。

```
/ip firewall filter add chain=forward protocol=tcp tcp-flags=syn connection-state=new \
action=jump jump-target=SYN-Protect comment="SYN Flood protect"

/ip firewall filter add chain=SYN-Protect protocol=tcp tcp-flags=syn limit=400,5
connection-state=new action=accept

/ip firewall filter add chain=SYN-Protect protocol=tcp tcp-flags=syn connection-state=new \
action=drop
```

SYN-Protect 第一条规则是限制每 5 秒通过新建的 syn 数为 400，第二条规则是丢弃超过的部分

SYN cookies

SYN Cookie 是对 TCP 服务器端的三次握手协议作一些修改，专门用来防范 SYN Flood 攻击的一种手段。它的原理是，在 TCP 服务器收到 TCP SYN 包并返回 TCP SYN+ACK 包时，不分配一个专门的数据区，而是根据这个 SYN 包计算出一个 cookie 值。在收到 TCP ACK 包时，TCP 服务器在根据那个 cookie 值检查这个 TCP ACK 包的合法性。如果合法，再分配专门的数据区进行处理未来的 TCP 连接。

RouterOS v6.x 设置如下：

```
/ip settings set tcp-syncookies=yes
```

6.x 以前的配置：

```
/ip firewall connection tracking set tcp-syncookie=yes
```

对于大量的 DoS 攻击，RouterOS 是很难防御的，建议寻找上级网络提供商协助处理，限制攻击源或更换 IP 地址等。

第十章 nat（网络地址翻译)

网络地址翻译(NAT)是当 IP 包通过路由器时取代其源和(或)目标地址的路由协议。它通常被用来启用专用网络的多个主机使用一个公用 IP 地址访问互联网，如我们常见的 192.168.0.0/24 局域网地址，通过一个拨号获取一个公网地址，通过这个公网将内网局域网地址转换上网。

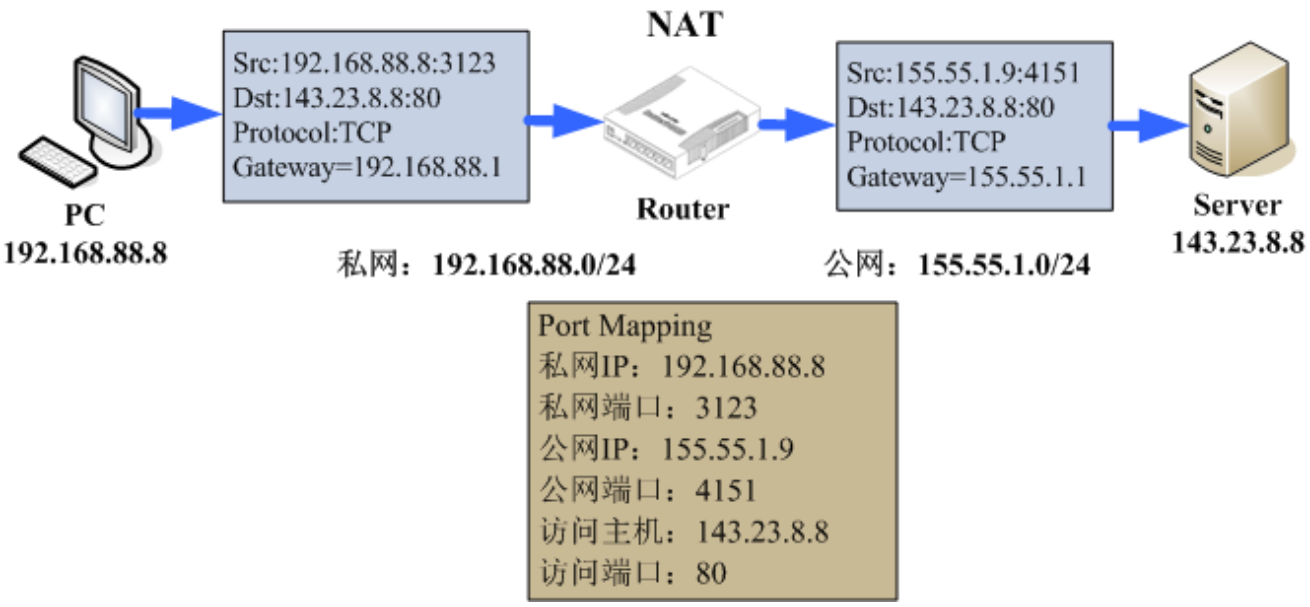
规格说明

功能包要求: ***system***
等级要求: *Level1*，*Level3*
操作路径: ***/ip firewall nat***
硬件使用：提升 CPU 和内存有助于 NAT 规则的处理

10.1 nat 介绍

网络地址翻译是一种允许本地网络主机使用一段 IP 地址进行本地通信，使用另一段 IP 地址进行外部通信的因特网标准。一个使用网络地址翻译的局域网就被称为 natted 网络。为了使网络地址翻译进行工作必须在每个 natted 网络都有一个 nat 网关路由器。nat 网关的作用就是在数据报进/出局域网时重写 IP 地址的作用，并在路由器建立 natted 端口映射表（Port Mapping）。

下图是一个私网 PC 访问公网服务器的 natted 过程，路由器会建立一个 Port Mapping 关系用于记录每个内网主机+端口访问外网目标主机+端口：



两种 NAT 类型：

- **源 NAT 或 srcnat.** 该 NAT 类型表现为从路由器发出时替换一个数据包的源 IP 地址，即一个 NAT 路由器将发到公网的私网 IP 地址替换为路由器的公网 IP，同样反向的操作也可以应用于内网方向。
- **目标 NAT 或 dstnat.** 该 NAT 类型表现为从进入路由器时替换一个数据包的目标 IP 地址，即一个 NAT 路由器从公网访问路由器的 IP 替换为内网访问私网地址。

基于 iptables 的 RouterOS 是按照目的地址来选择的，因此 dst-nat 是在 PREROUTING 链表上进行的，而 src-nat 是在数据包发送出去的时候才进行，因此是在 POSTROUTING 链表上进行的。

当主机在一个启用 NAT 的路由器后面时，无法实现端到端的互联互通。因此一些互联网协议无法工作在 NAT 环境下。一些 TCP 协议请求连接初始化从外网到内网或者如无连接协议 UDP，可能被破坏，例如 P2P 下载和 IPsec 的 AH 协议等 ，这样的问题可以通过 RouterOS 提供的 NAT Helpers 解决，NAT 透明协议。

Masquerade

Nat 设置中 **masquerade** 一种独特的 srcnat 类型，应用于公网 IP 地址随机变动的环境，用发送数据的网卡上的 IP 来替换源 IP，例如 PPPoE 拨号动态获取 IP 和 DHCP 动态获取 IP 等

每次路由器接口连接中断或 IP 地址变动，将清除发出接口的 masquerade 连接跟踪，该方式有助于尽快恢复公网 IP 地址变动

注意：选择 **masquerade** 后，会导致一些问题，如多线路情况下，线路中断后造成连接会话不稳定，在中断后，所有相关连接跟踪会被清除；例如线路闪断会导致会有连接跟踪会话被清楚，如果这样的情况出现在 **action=srcnat**，中断会保存连接，如果在超时时间内恢复。

重定向与伪装

重定向和伪装分别是目的 nat 和源 nat 的特殊形式。重定向类似与普通的目的网络地址翻译就好比伪装类似与源网络地址翻译——伪装是一种不需要指定 **to-addresses** 的源网络地址翻译的特殊形式——对外接口地址将被自动使用。重定向同理——进入接口地址将被使用。注意，**to-ports** 对于重定向规则来说很有意义——这就是在路由器起上处理这些请求的的服务端口。（比如：web 代理）

当数据报进行了目的网络地址翻译（dst-nat）时（不论 action=nat 或者 action=redirect），目的地址都将改变。有关地址翻译的任何信息（包括初始的目的地址）将被保存在路由器的内部维护表。当 web 请求被复位性到路由器的代理端口时，工作在路由器上的透明 web 代理将访问从内部表这个信息并从其中取得 web 服务器的地址。如果你正在对几个不同的代理服务器进行目的网络地址翻译，那你将不会从 IP 包头找到 web 服务器的地址，因为 IP 包的目的地址之前是 web 服务器的地址但现在已经变成了代理服务器的地址。从 HTTP/1.1 开始在 HTTP 请求中出现了特殊的可以告知 web 服务器地址的包头，于是代理服务器使用它取代了 IP 包的目的地址。如果没有这样的包头（如：老版本的 HTTP），代理服务器将不能确定 web 服务器地址也将无法工作。

这也就是说，对 HTTP 流从一个路由器到其他一些透明代理服务器进行正确的透明的重定向是有可能的。只有在路由器本身添加透明代理并配置才是正确的方法，因此你的“真实的”代理就是上级代理。这种情况下你的“真实的”代理再也不用是透明的，因为在路由器上的代理将成为透明的并将向“真正的”代理转交代理方式请求（根据标准，这些请求包括了所有必须的 web 服务器信息）。

属性描述

属性	描述
action (action 指令名称; 默认: **accept**)	匹配策略的数据包执行该规则的指令 • accept – 接受该数据包，并不再传递给后面的 nat 规则。 • add-dst-to-address-list –通过 address-list 参数，添加目标地址到指定的地址列表中

	• add-src-to-address-list –通过 address-list 参数，添加源地址到指定的地址列表中 • dst-nat – 通过 to-addresses 和 to-port 参数替换 IP 数据的目标地址或目标端口 • jump – 通过 jump-target 参数跳转到指定的自定义链表 • log – 添加 nat 规则信息到系统日志，日志信息包括：in-interface,out-interface,src-mac, protocol, src-ip:port->dst-ip:port 和数据包长度，当该 log 规则匹配并处理对应数据包后，会传递给后面的规则，类似与 passthrough • masquerade – 依据路由设置而定，将一个数据包的源地址替换为相应 IP，是一种自动化的 snat • netmap – 创建一个静态 1:1 的 IP 映射关系，即指定一个 IP 映射给另外一个主机，常用于分配公网 IP 地址给私网一台主机 • passthrough – 如果数据包配置该规则，将作为对其统计并不做任何处理传递给下一条规则，常用于数据统计。 • redirect – 通过 to-ports 参数，替换一个目标 IP 的端口，并重定向到路由器本地 IP • return – 返回控制，让数据包回到被 jump 跳转策略规则位置 • same – 为客户端指定一个连续范围的源地址 nat 转换，常用于使用连续的公网 IP 地址为内网用户做 nat 转换 • src-nat – 通过 to-addresses 和 to-ports 参数，替换 IP 数据包的源地址给指定的 IP 和端口
address-list (字符; 默认:)	调用地址列表（address-list）的规则名称
address-list-timeout (时间; 默认: **00:00:00**)	Address-list 地址列表记录超时时间，当 Address-list 记录 IP 条目超过指定时间，将其删除。该参数与 add-dst-to-address-list 或 add-src-to-address-list 操作关联，当值设定为 00:00:00 IP 条目将用于被记录
chain (名称; 默认:)	指定添加的链表。如果输入的链表名称不存在，则将创建新的链表。
comment (字符; 默认:)	填写规则备注
connection-bytes (整型-整型; 默认:)	匹配给定数据包连接的传输字节（bytes）。0-表示无穷大，例如 connection-bytes=2000000-0 意思是该规则匹配超过 2Mbytes 的相关传输流量
connection-limit (整型,掩码; 默认:)	限制每 IP 地址或 IP 地址段的连接数量
connection-mark (no-mark | 字符;	通过 mangle 标记指定的数据包做连接匹配，如果选择 **no-mark**，

默认:)	规则将匹配未标记连接的任何数据包
connection-rate (整型 0..4294967295; 默认:)	连接数率一个防火墙匹配器，能用于捕获当前连接的传输流量（具体使用可以参考 12.3 章节）
connection-type (ftp | h323 | irc | pptp | quake3 | sip | tftp; 默认:)	使用连接跟踪助手匹配相关连接的数据包，一个相关连接助手信息必须在/ip firewall service-port 下启用
content (字符; 默认:)	匹配数据包指定的文本内容
dscp (整型: 0..63; 默认:)	匹配 IP 包头的 DSCP 字段信息
dst-address (IP/掩码 | IP 范围; 默认:)	匹配数据包的目标地址或指定目标地址的范围
dst-address-list (名称; 默认:)	使用 address-list 设置的 IP 地址，匹配数据包的目标地址
dst-address-type (unicast | local | broadcast | multicast; 默认:)	匹配目标地址类型： • `unicast` – IP 地址点对点传输 • `local` – 目标地址属于本地路由器接口 IP • `broadcast` – 数据包发送到所有设备的子网域 • `multicast` – 数据包转发到指定的组播设备
dst-limit (整型[/时间],整型,目标地址 | 目标端口| 源地址[/时间]; 默认:)	匹配数据包直到超出给定的 pps 值。与 limit 属性不同的是每个目标 IP 地址/目标端口有自己的 limit 属性，参数格式：`count[/time],burst,mode[/expire]`. • **count** –测量指定“time”周期最大的数据包转发率均值 • **time** –指定时间周期，用于测量数据包转发率(可选项，如果未指定，默认为 1 秒) • **burst** –设置一个突发值，多少数据包不会被 count 限制 • **mode** – 数据包率分类 • **expire** – 指定一个记录 IP 地址/端口的到期时间，指定期到后自动删除
dst-port (整型[-整型]: 0..65535; 默认:)	仅协议配置为 TCP 或 UDP 协议时，可设置目标端口或目标端口范围值
fragment (yes|no; 默认:)	匹配分段数据包。第一个分段不会被记录。如果连接跟踪被启用，将不会有分段，系统会自动组装每一个数据包
hotspot (auth | from-client | http | local-dst | to-client; 默认:)	匹配 Hotspot 类型
icmp-options (整型：整形; 默认:)	匹配 ICMP 类型：代码字段
in-bridge-port (名称; 默认:)	当实际接口加入 bridge 后，数据包进入路由器的 bridge 接口
in-interface (名称; 默认:)	数据包进入路由器的网络接口
ingress-priority (整型: 0..63; 默	匹配数据包的入口优先级，该优先级来至于 VLAN, WMM 或

认:)	MPLS EXP 字段位。具体阅读
ipsec-policy (in | out, ipsec | none; 默认::)	匹配用于 IPsec 的策略，命令行参数值书写采用格式： `direction, policy`， Direction 是用于选择匹配策略的放向用于解包或策略的封包 • in – 在 PREROUTING、INPUT 和 FORWARD 链表生效 • out – 在 POSTROUTING、OUTPUT 和 FORWARD 链表生效 • ipsec – 匹配属于 IPsec 数据包的处理 • none – 匹配不属于 IPsec 数据包的处理，如 IPsec 传输数据包。 例如：如果路由器收到 IPsec 封装的 Gre 数据包，此时 `ipsec-policy=in,ipsec`，将匹配 Gre 数据包，但 `ipsec-policy=in,none`，将匹配 ESP 数据包
jump-target (名称; 默认:)	当 action=jump，将会指定跳转的目标链表名称
layer7-protocol (名称; 默认:)	选择已经定义的 7 层协议规则名称
limit (整型,时间,整型; 默认:)	匹配数据包直到超出给定的 pps 值。参数格式： `count[/time],burst`. • **count** – 测量指定“time”周期最大的数据包转发率均值 • **time** – 指定时间周期，用于测量数据包转发率(可选项，如果未指定，默认为 1 秒) • **burst** – 设置一个突发值，多少数据包不会被 count 限制
log-prefix (字符; 默认:)	当 action=log，为指定的 log 信息前加上文本前缀
nth (整型,整型; 默认:)	设置 nth 数据包规则，具体参考第十八章节
out-bridge-port (名称; 默认:)	当实际接口加入 bridge 后，数据包从路由器的 bridge 出接口
out-interface (名称; 默认:)	数据包从路由器本地网络发出的接口
packet-mark (no-mark | 字符; 默认:)	通过 mangle 标记指定的数据包，如果选择 **no-mark**，规则将匹配未标记的任何数据包
packet-size (整型[-整型]:0..65535; 默认:)	通过字节（byte）匹配指定的数据包长度或范围
per-connection-classifier(哈希值：分母/余数; 默认:)	PCC 匹配器，根据指定的选项对传输流量进行平均分配
port (整型[-整型]: 0..65535; 默认:)	当 protocol=TCP 或 UDP，允许配置指定的任何方向端口（源或目标端口）或指定的端口访问

protocol (名称或协议 ID; 默认: **tcp**)	通过协议名称或代码，匹配指定的 IP 协议
psd (整型,时间,整型,整型; 默认:)	探测可能存在的 TCP 和 UDP 端口扫描。参数采用以下格式： **WeightThreshold, DelayThreshold, LopPortWeight, HighPortWeight**。（触发 PSD 是机制是在 **DelayThreshold** 值时间范围里，**WeightThreshold** >= **LowPortWeight** + **HighPortWeight**） • **WeightThreshold** – 权重阀门值，当前 TCP/UDP 数据包收到从相同主机发送的不同目标端口被视为端口扫描，该值指定总权重值 • **DelayThreshold** – 延迟阀门值，从相同主机发送到路由器的不同的目标端口延迟时间间隔 • **LowPortWeight** – 低端口权重值，设置小于等于 1024 端口的目标端口权重值。 • **HighPortWeight** – 高端口权重值，设置高于 1024 端口的目标端口权重值
random (整型: 1..99; 默认:)	通过给定的概率范围随机匹配数据包
routing-mark (字符; 默认:)	通过 mangle 标记指定的数据包做路由匹配
src-address (IP/子网, IP 范围; 默认:)	匹配数据包的源地址或指定源地址的范围
src-address-list (名称; 默认:)	使用 address-list 设置的 IP 地址，匹配数据包的源地址
src-address-type (unicast | local | broadcast | multicast; 默认:)	匹配源地址类型： • unicast – IP 地址点对点传输 • local – 目标地址属于本地路由器接口 IP • broadcast – 数据包发送到所有设备的子网域 • multicast – 数据包转发到指定的组播设备
src-port (整型[-整型]: 0..65535; 默认:)	仅协议配置为 TCP 或 UDP 协议时，可设置源端口或源端口范围值
src-mac-address (MAC 地址; 默认:)	匹配一个数据包的源 MAC 地址
tcp-flags (ack | cwr | ece | fin | psh | rst | syn | urg; 默认:)	匹配指定的 TCP 标识 • ack – 确认数据 • cwr - 拥塞窗口减小 • ece - ECN-echo flag (显示拥塞通知) • fin – 关闭连接 • psh – push 标识 • rst – 丢弃连接

	• syn – 新建连接 • urg – 紧急数据
tcp-mss (整型: 0..65535; 默认:)	配置一个 IP 数据包的 TCP MSS 值
time (时间-时间,sat | fri | thu | wed | tue | mon | sun; 默认:)	允许创建该规则的时间执行计划，根据匹配的时间范围和日期执行。
to-addresses (IP 地址[-IP 地址]; 默认: **0.0.0.0**)	如果 action 设置为 dst-nat, redirect, netmap, same, src-nat ，替换源地址为指定 ip 地址
to-ports (整型[-整型]: 0..65535; 默认:)	如果 action 设置为 dst-nat, redirect, netmap, same, src-nat，则替换源端口为指定端口
ttl (整型: 0..255; 默认:)	匹配数据包 TTL 值

10.2 src-nat

如果你想在 ISP 给你的 10.5.8.109 地址后“隐藏”你的 192.168.0.0/24 的专用局域网，需使用源网络地址翻译。当数据包通过路由器时，伪装将 192.168.0.0/24 产生的源 IP 地址和包端口改变成路由器的 10.5.8.109 地址，并执行伪装“masquerade”配置：

```
/ip firewall nat add chain=srcnat action=masquerade src-address=192.168.0.0/24
out-interface=Public
```

所有从 192.168.0.0/24 出去的向外连接都将使用路由器的外网接口的 10.5.8.109 作为源地址代理访问互联网，由于伪装规则隐藏了内网地址，互联网将不能访问内网的 192.168.0.0/24 地址。如果需要让外网访问到内网某台服务器的端口，需要使用目的网络地址翻译（dstnat）。

RouterOS 支持两种隐藏私有网络方式，‘masquerade’与’src-nat’都是改变源 IP 地址或一个数据包的端口为外网 IP，Masquerade 和 source nat 典型的应用都是将私有网络隐藏在一个或多个外网后，设置一个新的源地址 nat

- **‘masquerade’**使用的是路由器默认的 IP 地址
- **‘src-nat’**需要明确指定转换的对外 IP 地址，即‘to-address’

Masquerade 操作

```
add chain=srcnat src-address=192.168.0.0/24 action=masquerade out-interface=WAN
```

Src-nat 操作，需要明确指定外网接口 IP 地址

```
add chain=srcnat src-address=192.168.0.0/24 action=src-nat to-address=10.5.8.109
out-interface=WAN
```

Same nat

在 srcnat 的 action 选项中，有一项 same 参数，应用场景主要是当运营商给你分配了一段同一子网掩码的 IP，你需要通过它们来对私网用户做 nat 转换，普通情况下我们用 src-nat 和 masquerade 只能将私网地址转换到一个运营商 IP 地址上，当我们从运营商那里分配到多个或者一段 IP 地址时，我们需要使用 same 的 nat 规则，即将所有 IP 地址都应用到私网 IP 地址的 nat 转换中，由于单个 IP 的 nat 端口只有 65535-1024=64511 个，但私网地址过多后，运营商提供一个 IP 地址端口将无法满足使用，因此需要运营商提供多个 IP 地址解决端口不足的问题。

例如运营商提供了 11.22.33.0/29 的 ip 地址，网关是 11.22.33.1，运营商分配可用 IP 地址为 11.22.33.2-11.22.33.6

首先在 ip address 里配置

```
[admin@MikroTik] /ip address>add address=11.22.33.2/29 interface=wan
[admin@MikroTik] /ip address>add address=11.22.33.3/29 interface=wan
[admin@MikroTik] /ip address>add address=11.22.33.4/29 interface=wan
[admin@MikroTik] /ip address>add address=11.22.33.5/29 interface=wan
[admin@MikroTik] /ip address>add address=11.22.33.6/29 interface=wan
```

设置路由网关：

```
[admin@MikroTik] /ip route>add gateway=11.22.33.1
```

配置 nat 规则

```
[admin@MikroTik] /ip firewall nat>add chain=srcnat src-address=192.168.0.0/24 action=same
to-addresses=11.22.33.2-11.22.33.6
```

PCC src-nat 负载均衡

按照 same 功能，我们可以解决连续 IP 地址的 src-nat 负载均衡，但无法解决不连续 IP 地址，下面介绍一种不连续 IP 地址，如果实现 src-nat 负载均衡的方法。这里要用到 PCC 功能，我们将之前多线路负载均衡的 PCC 用到 src-nat。利用 PCC 的原理将 src-nat 翻译实现负载均衡。

运营商提供了 11.22.33.0/24 的 ip 地址，网关是 11.22.33.1，运营商分配可用 IP 地址为 11.22.33.2，11.22.33.16，11.22.33.39，11.22.33.99

首先在 ip address 里配置

```
[admin@MikroTik] /ip address>add address=11.22.33.2/29 interface=wan
[admin@MikroTik] /ip address>add address=11.22.33.16/29 interface=wan
[admin@MikroTik] /ip address>add address=11.22.33.39/29 interface=wan
[admin@MikroTik] /ip address>add address=11.22.33.99/29 interface=wan
```

设置路由网关：

```
[admin@MikroTik] /ip route>add gateway=11.22.33.1
```

首先我们创建用户地址列表，将内网用户 IP 段设置到 ip firewall address-list

```
[admin@MikroTik] /ip firewall address-list
[admin@MikroTik] /ip firewall address-list> add address=192.168.0.0/24 list=userip
```

创建 mangle 的 PCC 规则，在 mangle 下创建内网用户 192.168.0.0/24 的 4 条 PCC 规则，将内网用户连接划分为 4 分，即分成对应 4 个运营商 IP 地址，配置脚本如下：

```
/ip firewall mangle
add action=mark-connection chain=prerouting comment=nat1 dst-address-type=\
    !local log-prefix="" new-connection-mark=nat1 passthrough=yes \
    per-connection-classifier=both-addresses:4/0 src-address-list=userip

add action=mark-connection chain=prerouting comment=nat2 dst-address-type=\
    !local log-prefix="" new-connection-mark=nat2 passthrough=yes \
    per-connection-classifier=both-addresses:4/1 src-address-list=userip

add action=mark-connection chain=prerouting comment=nat3 dst-address-type=\
    !local log-prefix="" new-connection-mark=nat3 passthrough=yes \
    per-connection-classifier=both-addresses:4/2 src-address-list=userip

add action=mark-connection chain=prerouting comment=nat4 dst-address-type=\
    !local log-prefix="" new-connection-mark=nat4 passthrough=yes \
    per-connection-classifier=both-addresses:4/3 src-address-list=userip
```

创建 nat 规则，我们采用 src-nat，将每条 PCC 链接标记设置到指定的运营商 IP 地址，即每一个运营商 IP 对应一条 PCC 标记，配置脚本如下：

```
    /ip firewall nat
    add action=src-nat chain=srcnat connection-mark=nat1 log-prefix="" \
        src-address-list=userip to-addresses=11.22.33.2
    add action=src-nat chain=srcnat connection-mark=nat4 log-prefix="" \
        src-address-list=userip to-addresses=11.22.33.99
    add action=src-nat chain=srcnat connection-mark=nat3 log-prefix="" \
        src-address-list=userip to-addresses=11.22.33.39
    add action=src-nat chain=srcnat connection-mark=nat2 log-prefix="" \
        src-address-list=userip to-addresses=11.22.33.16
```

这样配置完成后，就通过 PCC 规则将 192.168.0.0/24 的用户请求，平均负载到 4 个运营商相同网段，不连续 IP 地址上。

10.3 dst-nat

目标 nat 是常见的 nat 规则，通常端口映射、数据重定向、一对一地址映射等都会使用到目标 nat，即 dst-nat 功能。

这里有一个简单的一对一地址映射事例，如你想使用公网 IP 地址 10.5.8.200 访问本地地址 192.168.0.109，需要使用目标 nat 和原 nat 翻译。

添加公网和内网 IP 地址：

```
/ip address add address=10.5.8.200/24 interface=Public
/ip address add address=192.168.0.1/24 interface=Local
```

添加允许外部网络访问本地服务器的规则：

```
/ip firewall nat add chain=dstnat dst-address=10.5.8.200 action=dst-nat \
        to-addresses=192.168.0.109
```

添加规则使本地服务器能够与外部网络通信，并将其源地址翻译为 10.5.8.200

```
/ip firewall nat add chain=srcnat src-address=192.168.0.109 action=src-nat \
        to-addresses=10.5.8.200
```

端口映射配置

将外网访问 10.5.8.200 的 80 端口的数据映射到内网的主机 192.168.0.18

```
/ip firewall nat add action=dst-nat chain=dstnat dst-address=10.5.8.200 \
    dst-port=80 protocol=tcp to-addresses=192.168.0.18 to-ports=80
```

dst-nat 数据转移

通过使用 dst-nat 操作转移 IP 数据或端口到指定的主机上，如我们可以将内网所有访问 tcp/80 端口的数据转移到另外一个主机的 192.168.0.100，这样的操作可以实现对 80 端口的重定向到 proxy 一类的服务器

```
/ip firewall nat chain=dstnat protocol=tcp  dst-port=80 action=dst-nat
to-address=192.168.0.100
```

dst-nat 数据重定向

Redirect 是改变目标 IP 地址或一个目标 IP 数据的端口，指定访问数据转移到本地。与 dst-nat 不同的是 Redirect 不需要指明"to-address"，一个将 tcp/80 端口重定向到本地的测试

```
 /ip firewall nat add chain=dstnat  action=redirect protocol=tcp dst-port=80
```

DNS 重定向

当我们需要对所有用的 DNS 请求转移到本地进行解析时，我们可以使用 redirect，将所有来至内网的 dns 请求重定向到本地（必须确保你的 ip dns setting 的 DNS 缓存开启）

```
/ip firewall nat add action=redirect chain=dstnat dst-port=53 in-interface= \
     Local protocol=udp to-ports=53
```

1:1 nat 实例（netmap）

可以理解为 1 对 1 的 nat 配置，如下面的配置想从公用 IP 子网 11.11.11.1/32，访问本地的 192.168.88.2/32，应该使用目的地址翻译以及源地址翻译特性设置 **action=netmap**。

```
/ip firewall nat add chain=dstnat dst-address=11.11.11.1 action=netmap \
 to-addresses=192.168.88.2

/ip firewall nat add chain=srcnat src-address=192.168.88.2 action=netmap \
  to-addresses=11.11.11.1
```

可以使用连续的地址表示法，但仍然必须与所描述的内外 IP 网络地址长度相匹配

```
/ip firewall nat add chain=dstnat dst-address=11.11.11.0-11.11.11.255 \
        action=netmap to-addresses=192.168.88.0-192.168.88.255

/ip firewall nat add chain=srcnat src-address=192.168.88.0-192.168.88.255 \
        action=netmap to-addresses=11.11.11.0-11.11.11.255
```

非对称端口映射

非对称端口映射是指，外网目标访问的端口和原始映射端口可以不一样。即当我需要映射 80 端口时，我向外网开放的映射端口可以是非 80，可以选择其他端口，这样的目的是：

- 可以对内网多台相同服务的端口设备，修改外网为多个不同的端口进行映射
- 保证访问安全性，避免实际使用端口映射造成的攻击，攻击方不轻易发现映射端口是多少
- 解决部分端口被运营商封后，使用其他端口映射。

使用非对称的端口映射 src-nat 配置如下：

```
/ip firewall nat add chain=srcnat action=masquerade src-address=192.168.0.0/24
out-interface=wan
```

例如配置 192.168.0.100 的 80 端口映射，我们配置 dst-port 可以不写 80 端口，而写为 60000 端口

```
/ip firewall nat chain=dst-nat protocol=tcp   dst-address=10.5.8.200 \
    dst-port=60000 action=dst-nat to-address=192.168.0.100 to-port=80
```

这样来至公网通过 60000 端口访问会映射到内网主机 192.168.0.100 的 80 端口。

正确配置 nat 规则接口参数

RouterOS 管理员通常会不在意为 nat 规则配置接口参数，当然 nat 规则在没有 interface 接口参数的情况下也能正常工作，但它们也可能导致部分问题，例如，多内网 IP 的互访被伪装，路由环路和资源消耗等等。

对于初学者来说配置一条能上网的 nat 规则，默认设置 action= masquerade 能用就好，但对于网络管理员来说，注意为 srcnat 规则指定 out-interface 参数，为 dstnat NAT 规则指定 in-interface 参数。如有多个出接口，那么需要配置多个 NAT 规则，这些还能减少 CPU 的负载。

下面是一个不太推荐的配置：

```
/ip firewall nat
add action=masquerade chain=srcnat

add chain=dstnat dst-port=1234 action=dst-nat protocol=tcp to-address=192.168.88.2
to-port=80
```

上面的 srcnat 规则是没有指定源地址和需要伪装的外网接口，会对任意 IP 地址，内口和外网口都做 srcnat 伪装，同时会导致 srcnat 更消耗 CPU 资源。

下面是推荐的配置：

```
/ip firewall nat
add action=masquerade chain=srcnat src-address=192.168.88.0/24 out-interface=WAN

add chain=dstnat dst-port=1234 action=dst-nat protocol=tcp to-address=192.168.88.2
to-port=80 in-interface=WAN
```

精确的参数指定，能减少故障问题和 CPU 开销。多接口情况下，可以自定义 interface list（接口列表）来简化 nat 规则配置

10.4 Stats

在命令行里可以通过/ip firewall nat print stats 查看每条规则的状态

通过 pirnt stats 可以查看静态规则的状态

```
[admin@dzeltenais_burkaans] /ip firewall mangle> print stats
Flags: X - disabled, I - invalid, D - dynamic
 #    CHAIN              ACTION                BYTES          PACKETS
 0    prerouting         mark-routing          17478158       127631
 1    prerouting         mark-routing          782505         4506
```

查看所有规则包括动态规则 print all stats。

```
[admin@dzeltenais_burkaans] /ip firewall mangle> print all stats
Flags: X - disabled, I - invalid, D - dynamic
 #    CHAIN              ACTION                BYTES          PACKETS
 0    prerouting         mark-routing          17478158       127631
 1    prerouting         mark-routing          782505         4506
 2 D forward             change-mss            0              0
 3 D forward             change-mss            0              0
 4 D forward             change-mss            0              0
 5 D forward             change-mss            129372          2031
```

仅查看动态规则 print stats dynamic

```
[admin@dzeltenais_burkaans] /ip firewall mangle> print stats dynamic
Flags: X - disabled, I - invalid, D - dynamic
```

```
#    CHAIN          ACTION          BYTES        PACKETS
0 D forward         change-mss      0            0
1 D forward         change-mss      0            0
2 D forward         change-mss      0            0
3 D forward         change-mss      132444       2079
```

10.5 Connection 连接状态

操作路径: ***/ip firewall connection***

连接追踪用于维护连接状态信息，例如源目的 IP 地址和端口，连接状态，协议类型和超时。特定连接的状态包含：**estabilished** 意思即数据报是已知连接的一部分，**new** 意思为数据报开启了一个新连接，**related** 为在已有连接状态下数据开始了一个新连接，与之前的连接关联，如 FTP 数据传输或 ICMP 错误信息，**invalid** 意为数据报不属于任何一个已建立的连接。

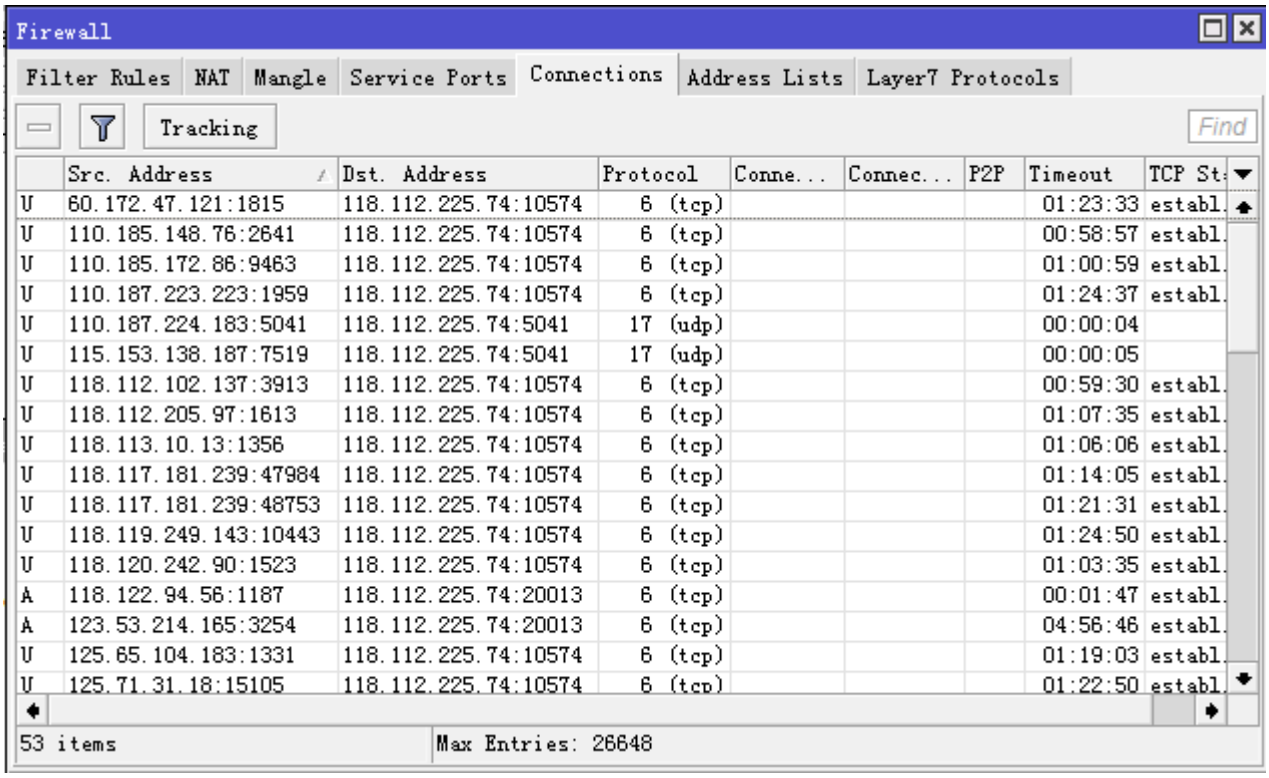
注： 连接追踪是对本地产生的数据报在 **prerouting** 链或者 **output** 链完成的。

另一个不能被过高估计的连接追踪功能是 nat 对其的需要。你应该清楚除非你启用了连接追踪否则 NAT 是不能完成的，对 P2P 协议识别也一样。连接追踪也在进一步处理前会从碎片中收集 IP 包。

/ip firewall connection 状态列表包含的最大数连接是由路由器的初始物理内存大小决定的。因此，例如一个 64M RAM 的路由器可以容纳最多 65536 连接的信息，128M RAM 的路由器就可以增加到 130000 以上。因此请确定你的路由器配置了足够量的内存以便可以适宜地处理所有连接。

属性描述

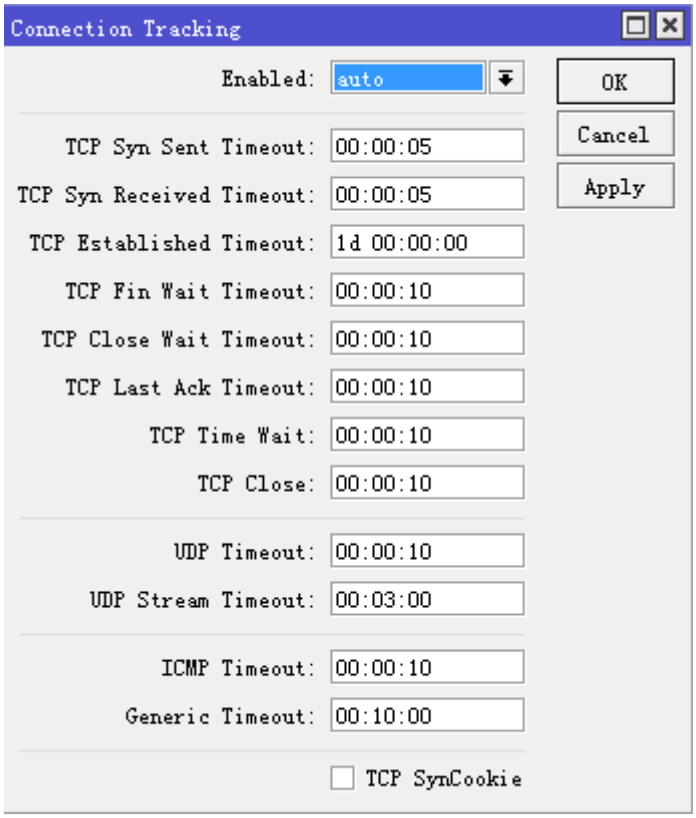
connection-mark (*只读: 文本*) - mangle 中设置的连接标记
dst-address (*只读: IP address:port*) - 连接建立到的目的地址和端口
protocol (*只读: 文本*) – IP 协定名和序号
p2p (*只读: 文本*) – P2P 协定
reply-src-address (*只读: IP address:port*) - 从源地址和端口建立的响应连接
reply-dst-address (*只读: IP address:port*) - 连接建立到的目的地址和端口
src-address (*只读: IP address:port*) - 从源地址和端口建立的连接
tcp-state (*只读: 文本*) - TCP 连接状态
timeout (*只读: 时间*) - 直到连接超时的时间量
assured (*只读:* true | false) - 显示是否看到对该条登记的最后一个包的回应
icmp-id (*只读: 整型*) - 每个 ICMP 包都会在被发送时得到一个为其设定的 ID，并且当接收器收到了 ICMP 信息时，它会在新的 ICMP 信息内设定同样的 ID 以使发送器能识别响应并能够用适当的 ICMP 请求连接它。
icmp-option (*只读: 整型*) - ICMP 类型和代码域
reply-icmp-id (*只读: 整型*) - 包含已接收包的 ICMP ID
reply-icmp-option (*只读: 整型*) - 已接收包的 ICMP 类型和代码域
unreplied (*只读:* true | false) - 显示请求是否未被响应



10.6 Tracking 连接跟踪

操作路径: ***/ip firewall connection tracking***

连接追踪为 nat 地址转换提供每条 TCP/UDP 连接的转换状态维护，提供了连接超时（timeout）参数，当在指定的超时时间过后，相应的条目将会从连接状态清单中删除。下面是 tracking 的配置，在 RouterOS 6.0 后开始新增 FastPath 功能，Enabled 由原来的双选，变为 auto、no 和 yes（具体参见 Fastpath 章节）：



属性描述

count-curent (*只读: 整数*) - 在连接状态列表中记录的当前连接数
count-max (*只读: 整数*) - 取决于总内存量的连接状态列表，自动计算出最大连接数
enable (yes | no|auto; 默认: **auto|yes**) –允许或禁止连接追踪，nat 被使用的情况下必须开启；根据硬件平台不同，x86 不具备 Fastpath，默认是 yes，而 RouterBOARD 具备 Fastpath 功能，默认是 auto。
generic-timeout (*时间*; 默认: **10m**) - 连接列表中追踪既非 TCP 又非 UDP 包的条目的最大时间量将会在看到匹配此条目最后一个包之后存活
icmp-timeout (*时间*; 默认: **10s**) - 连接追踪条目将在看到 ICMP 请求后存活最的大时间量
tcp-close-timeout (*时间*; 默认: **10s**) – TCP 连接追踪条目在看到连接复位请求（RST）或来自连接释放初始化机连接终端请求确认通知（ACK）之后存活的最大时间
tcp-close-wait-timeout (*时间*; 默认: **10s**) – 当来自应答器的终端请求（FIN）之后连接追踪条目存活的最大时间
tcp-established-timeout (*时间*; 默认: **1d**) – 当来自连接初始化机的确认通知后连接追踪条目存活的最大时间
tcp-fin-wait-timeout (时间; 默认: **10s**) – 当来自连接释放初始化机的连接终端请求（FIN）后存后连接追踪条目存活的最大时间
tcp-syn-received-timeout (*时间*; 默认: **1m**) – 当匹配连接请求（SYN）之后连接追踪条目存活的最大时间
tcp-syn-sent-timeout (*时间*; 默认: **1m**) – 当来自连接初始化机的连接请求（SYN）后连接追踪条目存活的最大时间
tcp-time-wait-timeout (*时间*; 默认: **10s**) – 当紧随连接请求（SYN）的连接终端请求（FIN）之后或在看到来自连接释放初始化机的其他终端请求（FIN）之后连接追踪条目存活的最大时间
udp-timeout (*时间*; 默认: **10s**) – 当匹配此条目的最后一个包之后连接追踪条目存活的最大时间
udp-stream-timeout (*时间*; 默认: **3m**) - 在匹配此连接（连接追踪条目是确定的）的最后一个包的响应被看到之后连接追踪条目存活的最大时间。它用于增加对 H323，VoIP 等连接的超时。

注:最大超时值取决于在连接状态清单中的连接数量。如果在列表中连接数量大于：

- 连接的最大数量的 1/16，超时值将为 1 天
- 连接的最大数量的 3/16，超时值将为 1 小时

- 连接的最大数量的 1/2，超时值将为 10 分钟
- 连接的最大数量的 13/16，超时值将为 1 分钟

如果超时值超过了上面列出的值，那么将使用更小的值。如果连接追踪超时值小于数据报率，比如：在下一个包到达之前超时就过期了，那么 nat 和 statefull-firewalling 将停止工作。

在判断一个数据包是 NEW 状态，有两种不同连接情况，第一个是在无状态连接(UDP 协议)，Connection 表中无法判断状态。另一个是有状态协议(TCP 协议)，TCP 协议在这种情况下，TCP 判断 NEW 状态是通过的新包总是带有 SYN 标志的 TCP 连接，因此 NEW 状态仅对 TCP 协议有效。

连接状态是 notrack 是使用 RAW 防火墙规则将连接排除在 Connection tracking 连接跟踪之外的一种特殊情况。RAW 规则将使匹配的转发流量绕过 Connection tracking 连接跟踪引擎，加速通过设备的数据包处理。即 NAT 无法在跟踪连接匹配到的 RAW 规则数据，NAT 规则不再生效。

注：tracking 功能被关闭，nat 功能也将会失效；如果你在不考虑启用 nat 功能情况，可以关闭掉 tracking。RouterOS 在经过一些大型的 nat 网络应用后，通过 Xeon 服务器平台能实现大概 20~25 万左右的 nat 连接转发，我们看一台 nat 设备的性能主要看他的连接转发能力，不管你多少流量 nat 连接转发能力是关键，如果我们将 RouterOS 的 nat 转换关闭，即 Tracking 关闭，RouterOS 在高性能路由设备上能实现 2Gbps~3Gbps 的吞吐量。但随着 RouterBOARD 支持 Fastpath 功能后，会根据不同平台的 RouterBOARD 提升转发性能。可惜 x86 平台无法得到这样的提升，因为芯片无法支持。

注：这里为大家提供一个参考一个路由网络，没有 nat 需求，采用 CCR1036-8G-2S+路由器，2 个万兆接口对网络做路由连接，connection tracking 的 enable 设置为 auto，峰值流量可以达到 1.4Gbps，峰值时 CPU 负载平均为 10%左右，而当我设置 connection tracking 的 enable 设置为 no，关闭连接跟踪，CPU 平均负载基本为 0，原因很简单，CCR 系列路由器在对路由包转发时，不做会话的连结跟踪意味着数据报不需要发给 CPU 处理，而直接通过交换芯片完成，后面 Fastpath 会讲到。

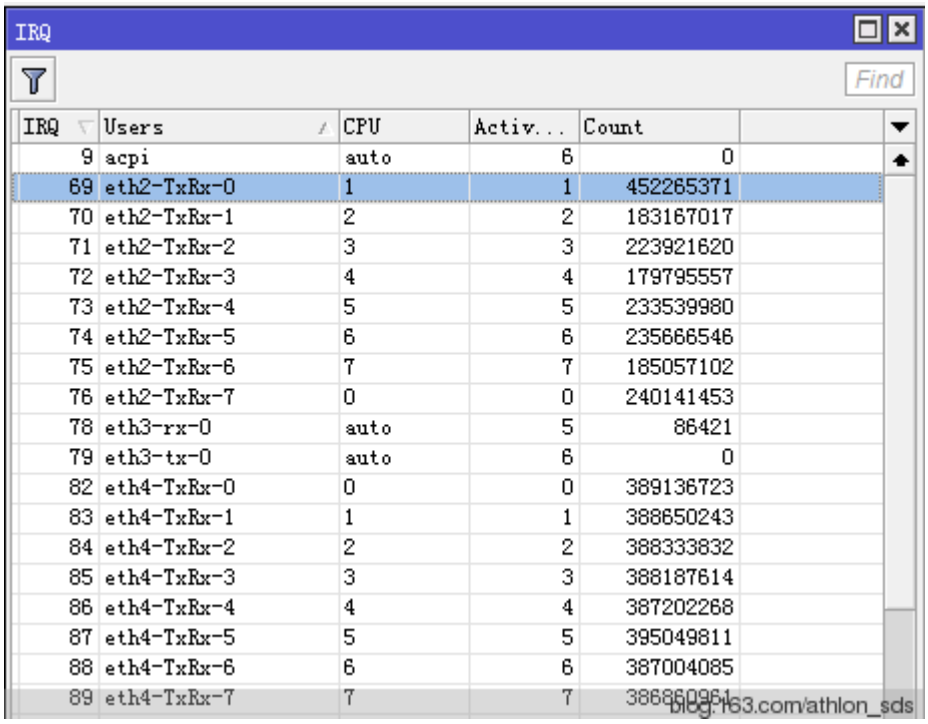
10.7 高性能 nat 实践

这个部分是 RouterOS 对大流量 nat 网关的实践做介绍，RouterOS v6 配合 Intel 至强处理器搭建了一个 nat 网关设备，单台处理超过 1.5G 的网络流量，前期采用 18 条多线路接入，每条 100M，PCC 负载均衡，两个千兆以太网卡连接华为 5700 交换机，这个实践配置供大家参考：

系统配置：

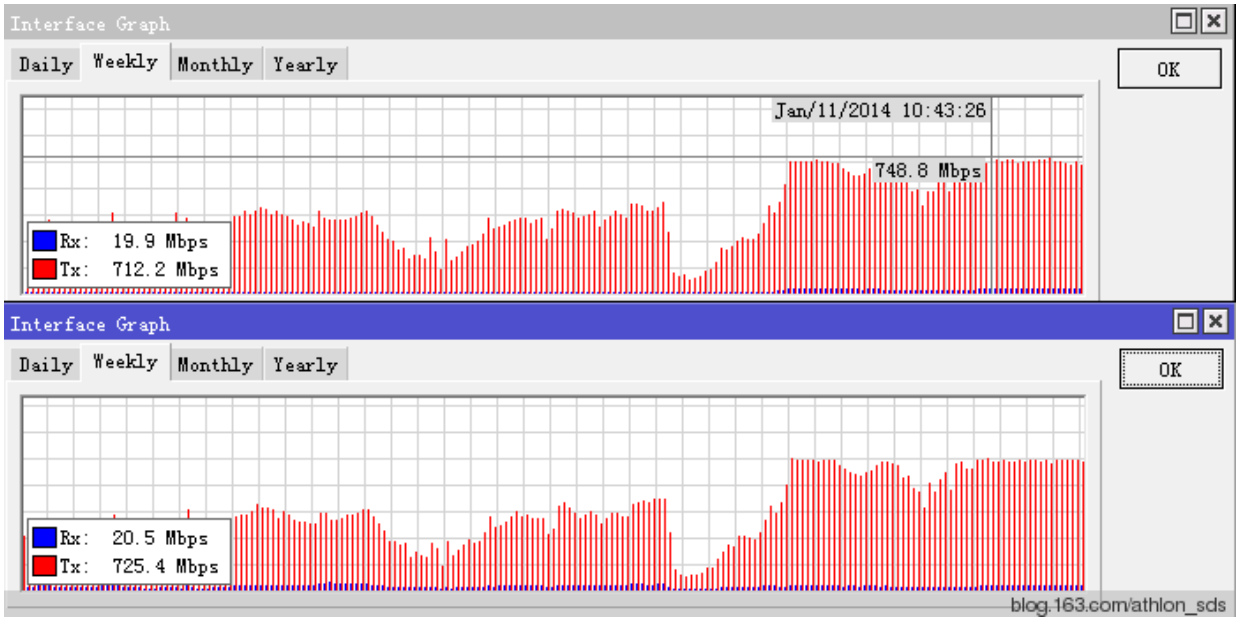
- Intel Xen 5606 × 2 （未打开超线程，一共使用 8 核）+ Intel 芯片组（别人的主机，主板只知道是 ASUS，型号没有记）
- 内存： DDR3 ECC 内存 2G ×2 （双信道需要配置 2 根 2G 内存，当然 RouterOS 只能识别 2G）
- 硬盘：1G Flash
- 网卡：Intel 82580 4 口网卡，每个网口中断 8
- RouterOS v6.6，功能包：仅安装 system、ppp 和 advanced-tools

使用中，手动调整了 IRQ，指定网卡每个终端负载到相应的 CPU：

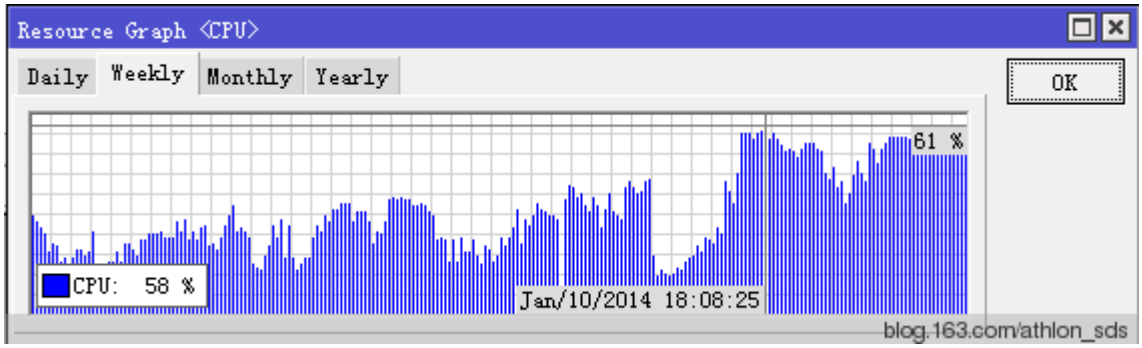


该配置主要用于跑 nat，其配置均省略，但出现过配置 simple queue 做整体流控后自动重启现象，后取消流控配置后，运行均无出现死机情况，这点估计是 RouterOS 在 Queue 还存在 bug。

共计 18 条，总 nat 处理流量达到 1.6G 左右，下面是两张网卡的流量截图：



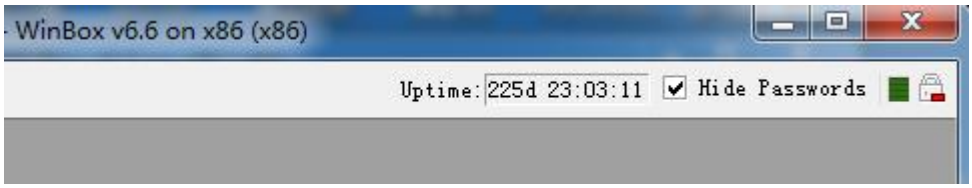
CPU 负载情况：



CPU 保持在 70%以内，运行无异常，session 数大约 24k 左右 。 个人更多希望是能多跑点会话，这样知道 RouterOS v6 版本的 nat 会话性能如何， 但有人说如果上行流量上去了，CPU 会比较高，但我更关心的是会话数，因为衡量 nat 转发性能会话数很非常重要的。

经过进一步测试，这套系统在 22 条 100M 线路 PCC 负载均衡后，流量只能达到 1.7G，且 CPU 在 83%左右，无法跑到 2 条千兆链路的 95% 。但如果拆分为 2 台服务器，每台 11 条 100M 线路做 PCC，链路使用率不仅能到 95%，且 CPU 都维持在 30%以内，2 台 CPU 加起来不到 60%，远低于一台的 83%，因此流量、nat 和 PCC 规则增加对系统负载也成倍增长。

该平台已经稳定运行了 225 天：



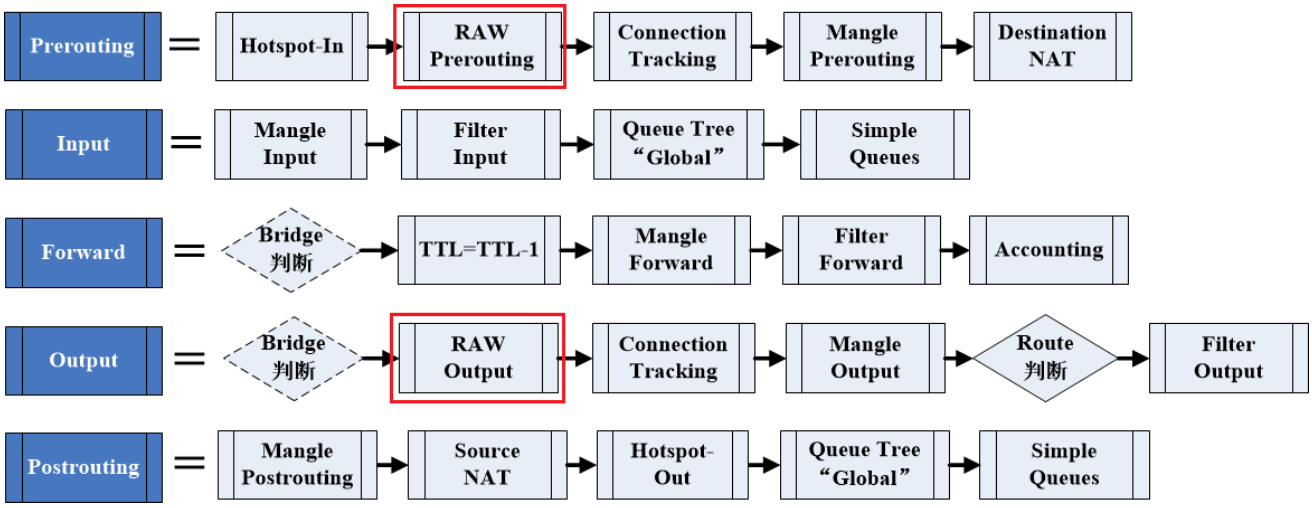
在此案例中网卡的中断数量对 CPU 均衡起到较大作用，所以在针对大流量的网络处理上，配合多 CPU 必须选择中断数高的网卡，以上内容供大家参考！

10.8 Raw

操作路径： **/ip firewall raw**

Raw 是 RouterOS v6.36 新增的功能属性，firewall Raw 列表允许选择性在 connection tracking 之前绕过或者丢弃数据包，这样能大大降低 CPU 负载。该功能非常有助于防御 DOS 攻击，RAW 链表不依赖于连接跟踪（connection-tracking）的匹配器，如连接状态和 Layer7 等，也就是在 RAW 里面无法使用这两个功能。

如下图，Raw 出现在 Prerouting 和 output 链表，并在 connection-tracking 前执行：



Raw 目的是对指定的 IP 访问可以不经过连接跟踪（connection-tracking），有助于减少 CPU 的开销，但对于私有网络做 nat 转换来说，connection-tracking 是必须的，那 Raw 又如何在实际网络应用中发挥作用。例如，在企业或 ISP 网络内网路由访问不需要做 nat 的连接跟踪，或者我们遭到 DDoS 攻击，我们可以通过 Raw 规则关闭掉攻击源 IP 的连接跟踪，使其不再 RouterOS 本地建立连接会话，不予回应，减小 CPU 开销。

在 Raw 中有两个链表，分别是 prerouting 和 output:

- **prerouting** – 用于处理任何进入路由器的数据报
- **output** – 用于处理数据报源于路由器，通过其中一个网络接口发出，即由路由器自身发出的数据

例如:我们遭到 11.22.33.4 的 IP 通过 syn 的 DDoS 攻击，我们可以拒绝对该 IP 做连接跟踪，减少 CPU 开销

```
[yus@MikroTik] >/ip firewall raw
[yus@MikroTik] /ip firewall raw>add chain=prerouting src-address=11.22.33.4 action=notrack
```

另一种情况，如较大的网络环境，涉及内网和外网请求，在访问外网时需要做 nat 转换，而内网则不需要，因此流量和会话增加的情况下，避免 CPU 负载过高，可以通过 Raw 关闭掉访问内网的连接跟踪。

例如，在一个大型网络，内网 IP 地址段是 192.168.0.0/16，关闭掉内网的连接跟踪，配置如下：

```
[yus@MikroTik] >/ip firewall raw
[yus@MikroTik] /ip firewall raw>add chain=prerouting dst-address=192.168.0.0/16
action=notrack
```

这样配置后，我们在 connection-tracking 列表将不会看到内网访问 192.168.0.0/16 的连接会话条目。

RAW 在 connection-tracking 之前处理数据，因此如果你在配置防火墙过滤策略时，想尽量降低 PCU 开销，可以在 RAW 里创建 IP 过滤规则。

第十一章 RouterOS Fastpath

RouterOS 基于开发的硬件产品大部分支持 Fast path，该功能允许数据报转发不在经过 Linux 内核处理，直接由 IC 芯片处理后转发，进一步提升转发速度。RouterOS 将添加更多关于 Fastpath 的功能更新，例如桥接和 forward 过滤，当前的 Fastpath 还有较多的限制条件（v6.0rc10）。

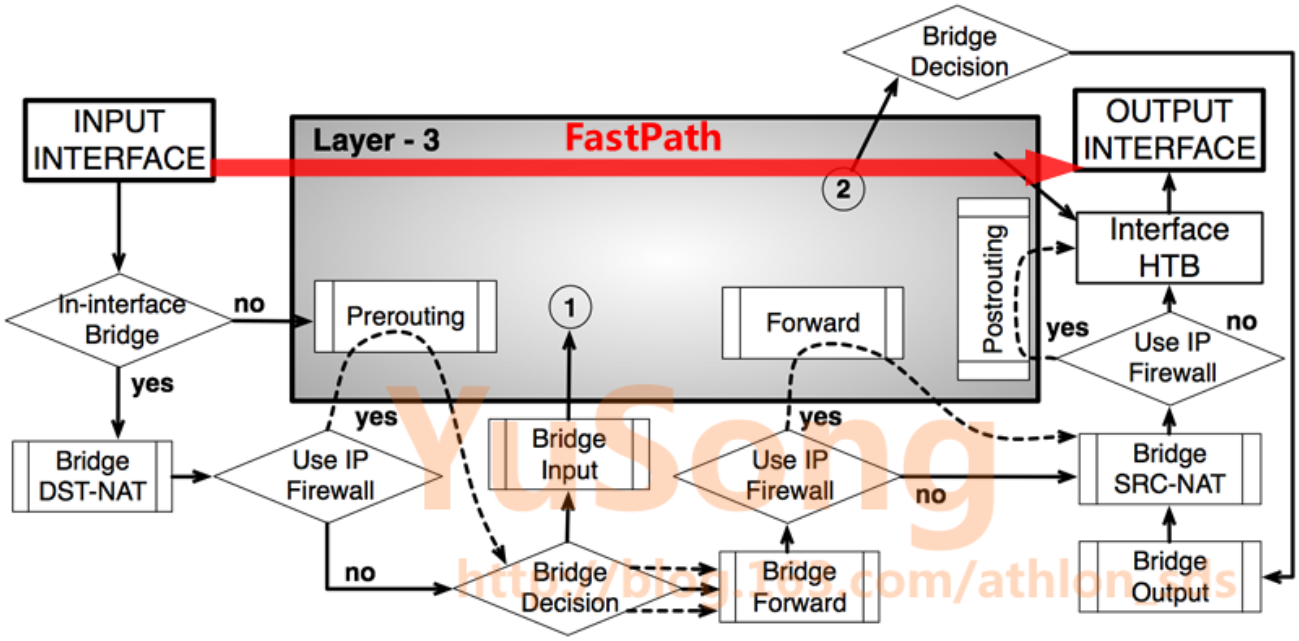
Fastpath 功能是基于 RouterBOARD 的 RouterOS 发展的必然结果，否则软件和硬件不能结合，并针对性的对硬件做优化，那将很浪费硬件所提供的资源，比较 Cisco、Juniper 等 IOS 和 JunOS 都是针对自己的硬件做了优化。

由于 RouterOS v6 开启 Fastpath 有部分功能限制，但当所有条件都满足时，Fastpath 会自动工作。当你没有防火墙规则（在将来的版本还会支持多种防火墙规则），如 nat 等，connection tracking 将自动关闭，Fastpath 自动启动生效，这样将使得 RouterOS 的吞吐量倍增。因此在新的版本中加入了 connection tracking 默认设置为 auto。

从 RouterOS v6rc7 版本开始将在 ip settings 目录中看到 Fastpath 配置，基于 IPv4 协议的 Fastpath 会在下面情况下才能开启，以下限制是 RouterOS v6.45 后的要求：

- 防火墙规则不能配置;
- /firewall address-list 不能配置
- simple 和 queue trees 不能配置 parent=global;
- mesh 和 metarouter 网络不能配置;
- sniffer, torch 和 traffic generator 不能运行;
- connection tracking 连接跟踪不能激活;
- IP accounting 需要禁用;
- VRF 不能配置(`/ip route vrf` 配置为空);
- Hotspot 热点认证不能启用(`/ip hotspot` 下无接口配置);
- IPSec policies 不能配置；
- `/tool mac-scan` 不能使用;
- `/tool ip-scan` 不能使用。

“/ip firewall connection tracking set enabled”参数添加了新的功能“auto”值，即默认情况下连接跟踪功能是禁用状态，当 firewall 添加规则后，连接跟踪才会启用。



如上图，Fastpath 将绕过多个操作流程，直接从入接口快速将 IPv4 的数据转发到目标接口上，高速的转发功能，仅能有相应的硬件支持，软硬结合是必然的，对于通用计算的 x86 的构架将不能获得这样的提升，其实对于 RouterOS 发展来说是必然的，以后 x86 平台在部分行业看起来是一个廉价的 DIY 平台。
看看以下 RouterBOARD Fastpath 比较

RB2011 pps throughput	64 byte	512 byte	1518 byte
bridging RouterOS v5	151000	145300	102500
bridging RouterOS v6 fastpath	270000	232000	*122000 (port max)*
times faster	**1.79x**	**1.60x**	**> 1.19x**
routing v5 RouterOS v5	128800	126500	96100
routing v6 RouterOS v6 fastpath	227000	210000	*122000 (port max)*
times faster	**1.76x**	**1.66x**	**> 1.27x**

RB750GL pps throughput	64 byte	512 byte	1518 byte
bridging RouterOS v5	97000	90400	78200
bridging RouterOS v6 fastpath	194000	178000	*81200 (port max)*
times faster	**2x**	**1.97x**	**> 1.04x**
routing v5 RouterOS v5	66400	65000	52000
routing v6 RouterOS v6 fastpath	183700	167000	*81200 (port max)*
times faster	**2.77x**	**2.57x**	**> 1.56x**

以上是开启 Fastpath 功能性能提升非常明显。

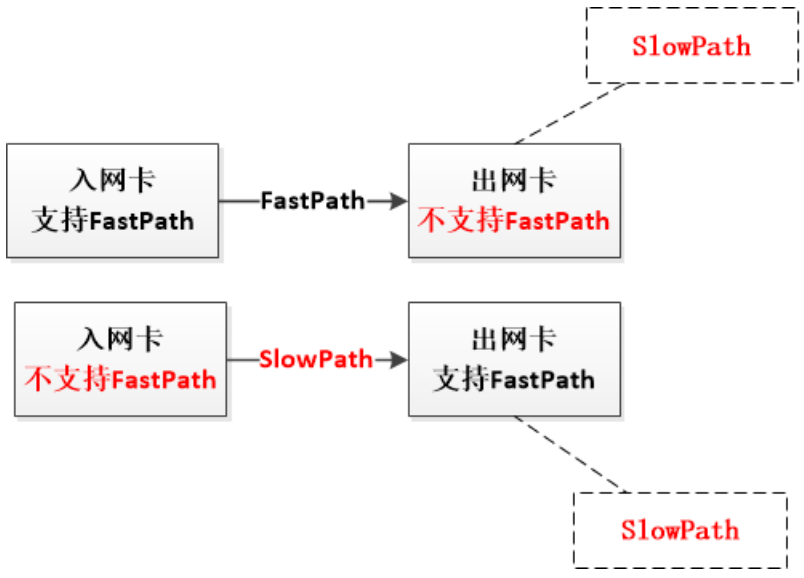
11.1 FastPath 支持列表

FastPath 功能仅支持部分网卡的快速转发，在下面的表中，列出了支持的 RouterBOARD 设备和其所属的网卡

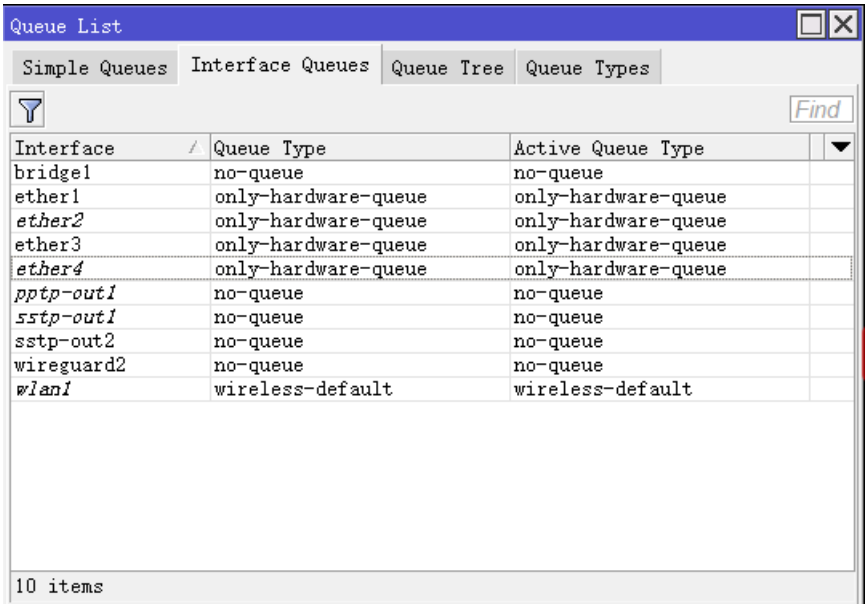
设备型号/网卡	支持网络接口
RB6xx series	ether1,2
RB7xx series	所有接口
RB800	ether1,2

设备型号/网卡	支持网络接口
RB9xx series	所有接口
RB1000	所有接口
RB1100 series	ether1-11
RB2011 series	所有接口
RB3011 series	所有接口
CRS series routers	所有接口
CCR series routers	所有接口
All devices	Wireless 接口
	Bridge 接口
	VLAN, VRRP 接口
	bonding 接口(RX only)
	PPPoE, L2TP 接口
	EoIP, GRE, IPIP 接口

通过从 FastPath 切换到 SlowPath，数据包可以进入半 FastPath 处理，但不能反过来。例如，如果接收网口支持 FastPath，但输出网卡不支持，那么路由器将尽可能地通过 FastPath 处理程序处理数据包，然后继续使用 SlowPath。如果接收网卡不支持 FastPath，但输出网卡支持 FastPath，数据包将通过路由器一直由 SlowPath 处理。



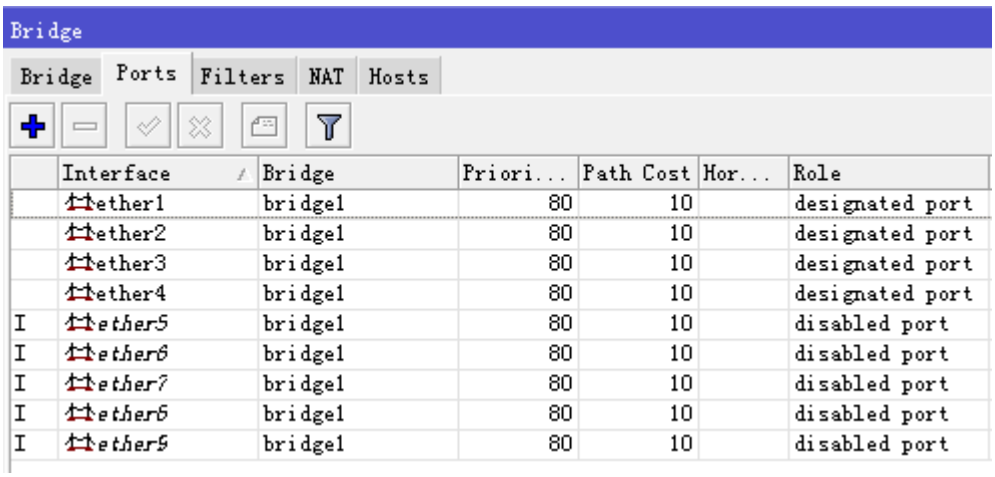
Queue 流控需要查看 interface queue 栏下，支持 only-hardware-queue 的接口才能保证 FastPath，如果你需要 interface queue，非 hardware，那么数据包将无法完全使用 FastPath，但这对性能没有很大的影响，因为"interface queue"是数据包处理的最后一步。



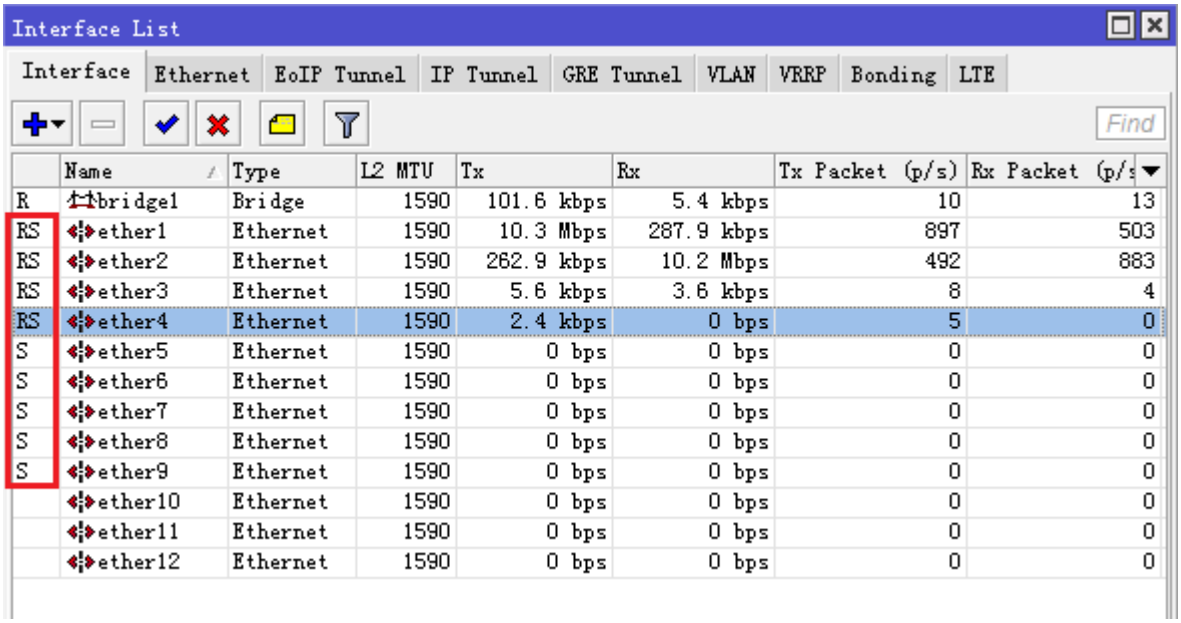
11.2 CCR 系列的 Fastpath

由于 CCR 系列采用的 Tile 构架的处理器，CPU 核心包括 9、16、36 和 72 核四个版本，但在该系列机型上不会找到 switch 功能菜单，除了早期的 CCR1009，有一个独立的 Atheros 交换芯片，但 Atheros 交换芯片到 CPU 只有 1Gbit 通道，导致 4 个千兆接口共享 1Gbit，因此在后面的 CCR1009 取消了 Atheros 芯片，所有网络接口直连到 CPU。

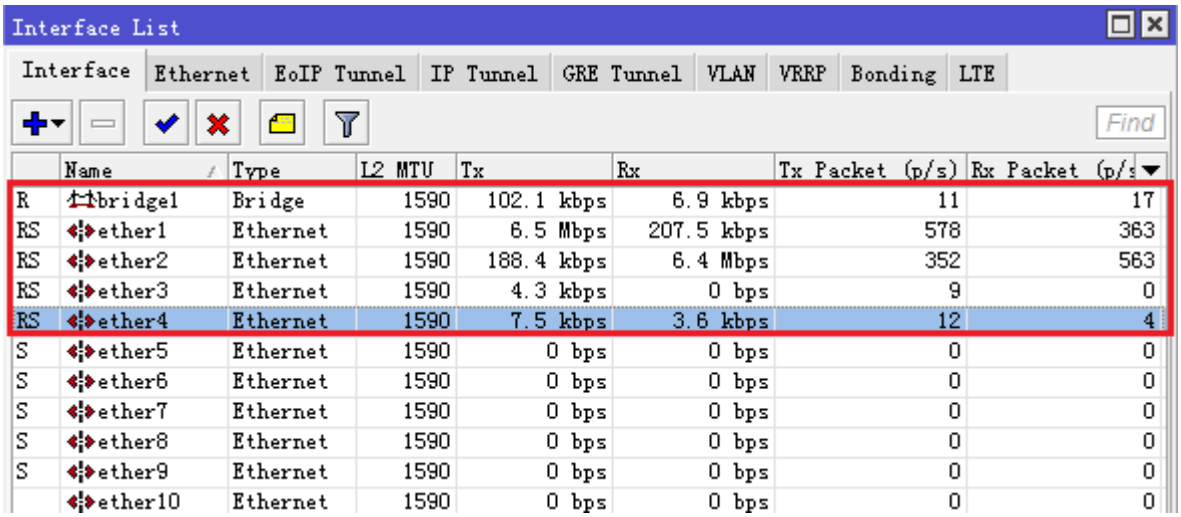
关于 Fastpath 在 CCR 系列上测试，我们可以通过将几个以太网接口加入 bridge，通过二层网络转发来测试，在 bridge port 中将指定的网卡加入 bridge1 后，会发现在 interface 中网卡前缀自动加上了 S 标记，即 Slave，与其他 RouterBOARD 配置 switch 功能一样的前缀标示。



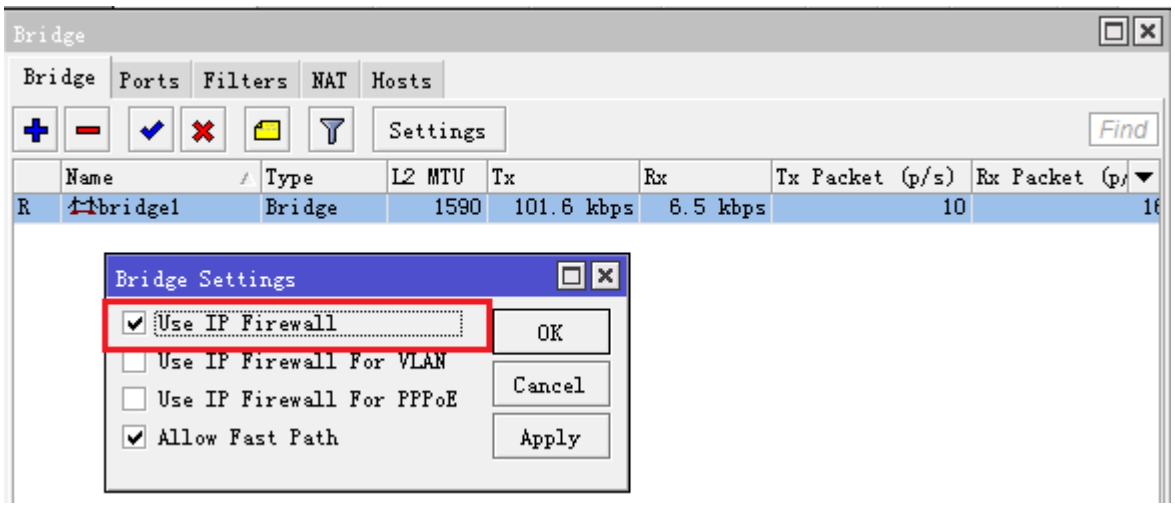
Interface 中看到 Slave 的前缀



我们需要注意到一个细节，那就是 bridge1 接口上并没有产生流量，按照正常情况下会是所有被桥接的网卡流量 tx 和 rx 总和与 bridge1 相同，但在这里我们并没有看到这样的情况，即说明作为 bridge 后 CCR 系列的 RouterOS 会自动将 bridge 识别为 Switch，但我个人认为并非完全的 Switch，而是 Fastpath 的功能。



即使当我打开 Use IP Firewall 后，流量也未有变化



如果是 Switch 功能，bridge 的 Filter 过滤将不会生效，但我测试 Filter 功能后发现依然没有问题，且当配置 Filter 为接收所有的二层数据或禁止某一网卡数据通过后，bridge1 的网卡流量仍然没有和其他网卡总和相等，即说明 Fastpath 功能起到了一定的作用，这样 CCR 系列的二层交换直接给了硬件处理，不会经过 RouterOS 内核，提高了转发率。

11.3 IPv4 FastTrack

从 v6.29rc9，引入新的功能 FastTrack，根据官方解释可以使你的路由器 Firewall/NAT 性能提升 5 倍。FastTrack 基于 IPv4，Fasttrack 会自动处理标记的连接会话，当前支持 SNAT 和 DNAT，协议支持 TCP 和 UDP 会话。

Fasttrack 可以理解为 FastPath + Connection Tracking。它允许将连接标记为" Fast Tracked"，标记属于 FastTrack 连接的数据包将以 FastTrack 的方式发送。这样的连接在 connection 列表将有一个 FastTrack 标志，主要特点为：

1. FastTrack 加速指定的连接跟踪条目的数据报处理
2. FastTrack 具备 NAT 加速支持
3. 与普通的连接跟踪和 NAT 性能提升 5 倍
4. firewall filter/mangle 菜单下新增 fasttrack-connection 选项
5. 支持 IPv4 TCP/UDP 连接跟踪条目
6. 工作方式类似于“mark-connection” 添加连接跟踪标记，并允许数据报被定义为 FastTrack
7. 并非所有的数据报都能被 FastTrack 处理，有一些数据报将仍然使用普通的 Conntrack（连接跟踪）

注意，fasttrack 启用后，将直接绕过 firewall、simple queue、queue tree（parent=global）、ip traffic-flow，ip accouting、ipsec、hotspot 和 vrf 等应用，因此路由器配合应确保应用 fasttrack 的接口没有其他配置，IPv4 FastTrack 启用有以下要求

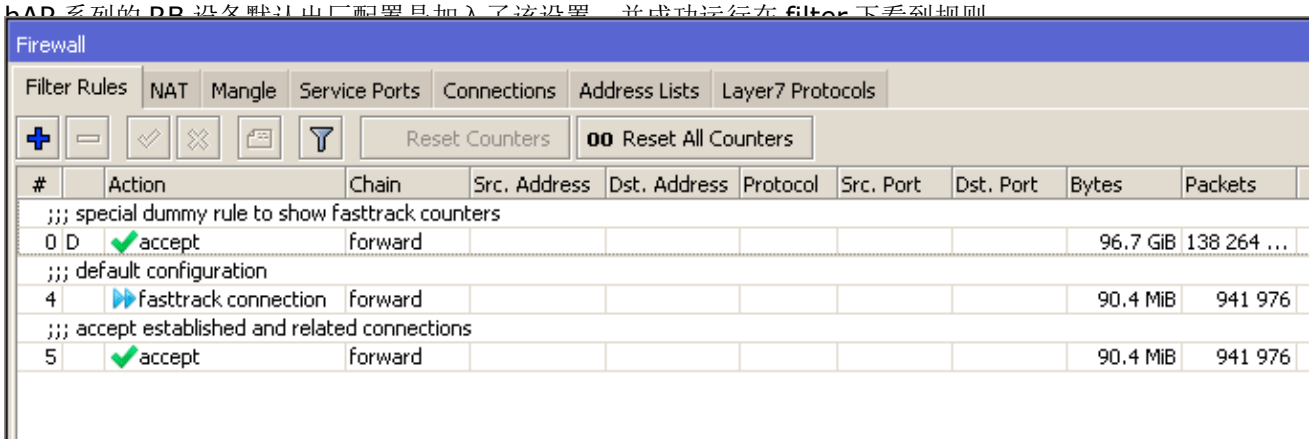
- Mesh、MetaRouter 接口没有配置；
- sniffer, torch and traffic generator 未运行；
- ~~mac-ping,mac-telnet 或 mac-winbox 会话连接未启用~~，6.33 移除；
- /tool mac-scan 未被使用;
- /tool ip-scan 未被使用;

由于 FastTrack 是基于 FastPath 上运行，因此也继承了许多运行 FastPath 的限制条件，包括不支持 VLAN、PPP 和 Bonding 的配置（从 6.30 开始 Fastpath 支持 VLAN 和 VRRP 接口，同时支持 Bonding 接口的 rx 方向）

启用 FastTrack，进入 ip firewall filter 配置如下，并将该规则放在规则顶端

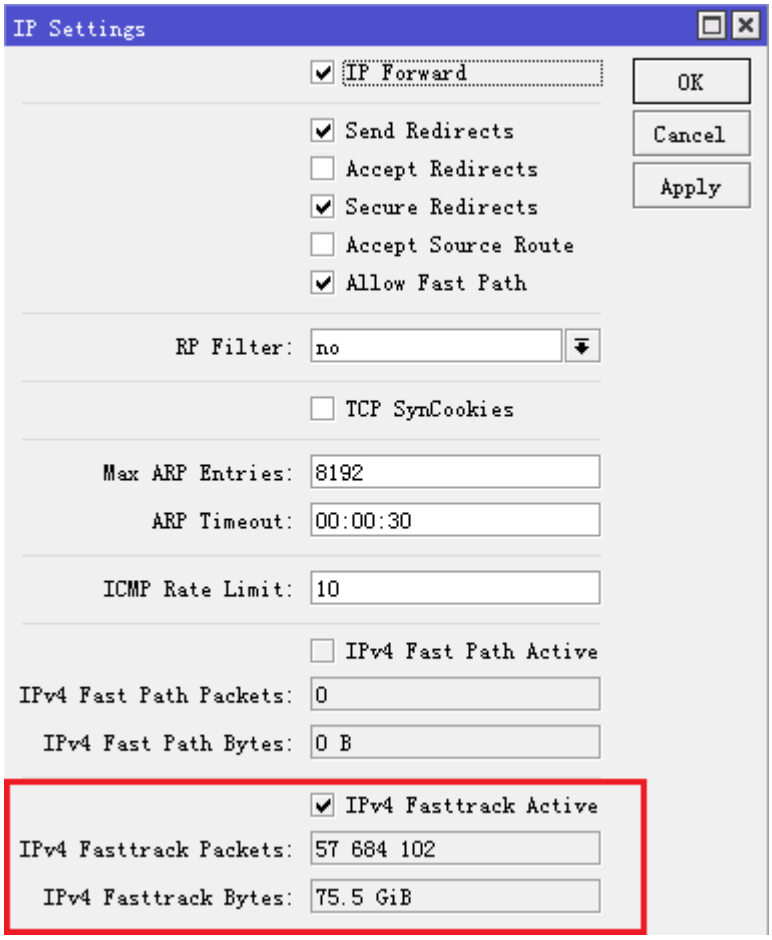
```
/ip firewall filter add chain=forward action=fasttrack-connection
connection-state=established,related
/ip firewall filter add chain=forward action=accept connection-state=established,related
```

hAP 系列的 RB 设备默认出厂配置是加入了该设置，并成功运行在 filter 下看到规则



如果要关闭 Fasttrack 规则，首先要禁用/ip firewall filter 的 fasttrack 规则，然后重启路由器才生效。

可以在 ip setting 菜单下，看到 ipv4 fasttrack active 启用状态，并显示统计信息



由于 FastTrack 在各个功能配置的限制，如 VLAN、PPP、Bonding 和 Queue 开启时不能启用，因此对于路由器性能提升的同时功能限制较多。

警告： Fasttrack 启用后，Queues 流控功能，除 Queue Trees 的接口作为父级外，将无法使用，并且 firewall filter 和 mangle 规则，将无法匹配到 Fasttrack 的流量，也会导致失效。

根据官方的测试，在 RB2011 上配置 nat 规则，启用 FastTrack ，基于 TCP 建立一条会话的带宽测试情况如下：

FastTrack	
关闭	开启
360Mbps	**890Mbps**
CPU 100%	**CPU 86%**
防火墙占用 CPU 44%	**防火墙占用 CPU 6%**

第十二章 Queue（带宽控制）

Queue（队列）是 RouterOS 针对数据流的 QoS 功能菜单，包括了 simple queue 和 queue tree 两个主要数据流带宽控制功能，对于 MikroTik RouterOS 主要支持队列类型：

- **PFIFO** - 包先进先出
- **BFIFO** – 字节先进先出
- **SFQ** - 随机公平队列
- **RED** – 随机早先探测
- **PCQ** – 每次连接队列
- **HTB** – 等级令牌桶

功能包要求: ***system***
等级要求: *Level1 (限 1 条规则)* ， *Level3*
操作路径: ***/queue***

服务质量(QoS)即路由器应该优先考虑保证数据流的质量，并形成新的网络数据流。 QoS 并非是只是对流量的控制，它更多的是与提供优良质量的服务相关。以下是一些 RouterOS 带宽控制机制的特征：

- 对指定 IP 地址，子网，协议，端口以及其他参数限制数据率
- 限制 P2P 流量
- 数据流的优先处理
- 为 Web 浏览提供队列脉冲
- 设置指定时间间隔执行队列
- 每个用户连接队列，实现平等共享可用流量
- 队列应用在通过路由器真实接口的数据报上(比如:队列应用在向外的接口，像业务流)，或者三个添加的虚拟接口中的任何一个或几个(global-in、global-out、global-total，v6 版本后只有 global)。

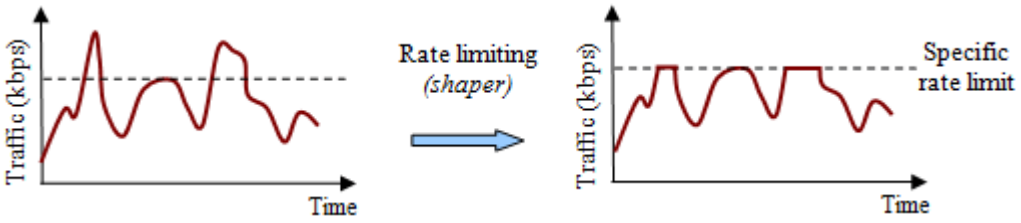
12.1 流控原理

流控用于对网络数据流发送和接收进行控制，流量被控制在指定的范围值内，即传输的流量只能小于或等于这个值，反之超过的流量将会被丢弃或延迟发送。

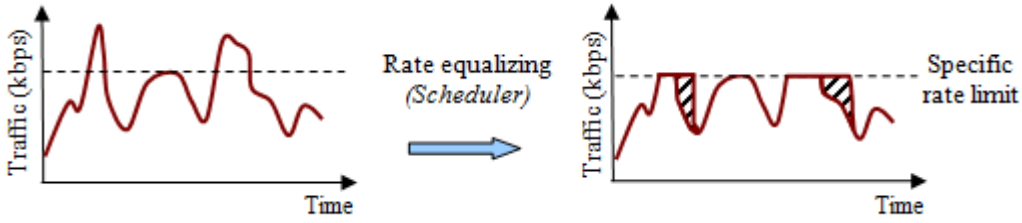
流控执行的两种方式：

- 丢弃所有超出的流量限制的数据 – **rate limiting** (丢弃)，当 queue-size=0 超出部分流量被全部丢弃。
- 超出指定限制流量加入到队列中– **rate equalizing** (计划任务) ，当 queue-size=unlimited（无限制），队列中的数据按照比例均衡发送。

下面两组图片进一步理解 **rate limiting** 和 **rate equalizing** 的区别：



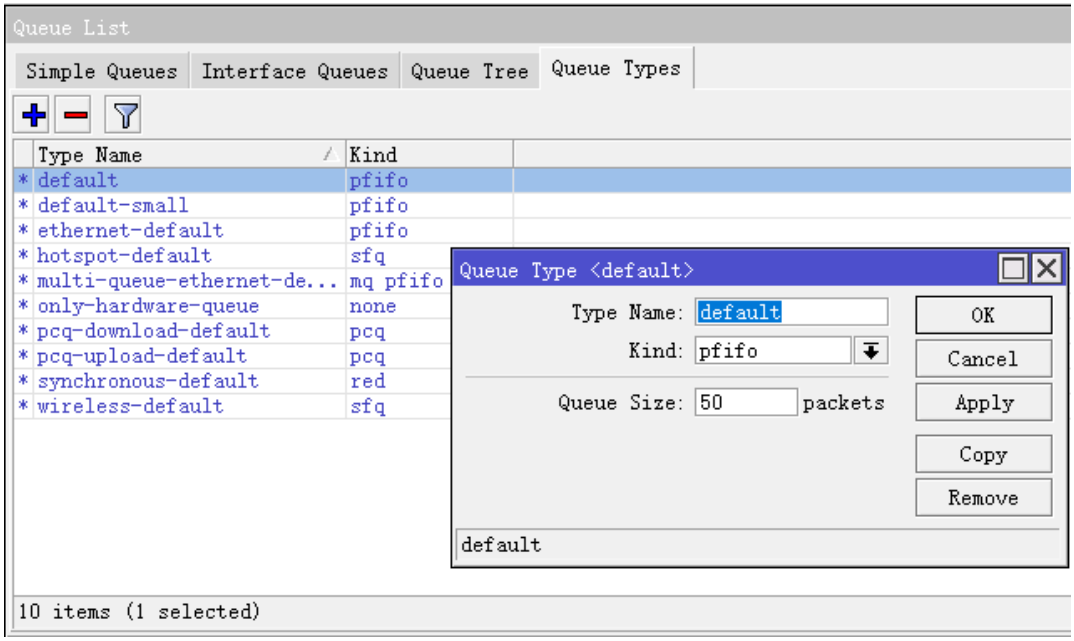
上图采用 rate limiting，（**queue-size=0**）显示了传输流量超出了指定限速带宽的那部分被直接丢弃掉。网络 ping 会出现丢包



上图显示了传输流量超出了指定限速带宽的那部分，将进入队列容器（**queue-size=unlimited**），超出部分的数据会延迟发送。网络 ping 会出现延迟增大。

注意：当指定了 queue-size 的大小后，数据包延迟发送只会在队列容器没有满的情况下，如果队列容器没有多余的空间缓存数据，数据包同样会被丢弃，但需要注意增加 queue-size 会增加路由器的内存消耗。

在 RouterOS 队列容器值（queue-size）可以通过/queue type 指定，每种类型的 queue type 有不同的队列长度大小，队列的类型包括：(pfifo-limit, bfifo-limit, pcq-limit, pcq-total-limit, red-limit，但所有的类型原则上是一样的，即 queue-size 决定数据报是被丢弃还是延迟发送。



每个队列都有 2 个速率限制：

- **CIR** (约定信息速率 Committed Information Rate) – (在 RouterOS 中的参数为 **limit-at**) 最坏的情况下，无论如何都会将得到给定的的 CIR 传输量(假设我们能发送那么多的数据量)。

- **MIR** (最大信息速率 Maximal Information Rate) – (在 RouterOS 中的参数为 **max-limit**) 最好的情况下，如果有剩余带宽，才能获得这个的带宽。

队列执行在 RouterOS 基于等级令牌桶 Hierarchical Token Bucket (HTB)，HTB 允许创建等级队列结构并能指定队列直接的关系，在 RouterOS v6.0 之前等级结构能被指定在 4 个不同的位置

- **global-in** - 代表了所有输入接口(INGRESS 队列)。请注意在数据报过滤前与 **global-in** 相关的队列应用到路由器接的数据流。 **global-in** 排序就是在 mangle 和 dst-nat 之后执行。
- **global-out** - 代表了所有普通的输出接口。附属于它的队列会在附属于特定接口的队列之前应用。
- **global-total** - 表了一个流经路由器的数据都能通过的虚拟接口。当把一个 qdisc 附属到 global-total 时，限制需要在两个方向起作用。例如，如果我们设置一个为 **total-max-limit** 256000 限制，我们将得到 upload+download=256kbps(最大值)
- **<interface name>** - 明确指定的网络接口，在流量从这个接口发送出去时将被放入 HTB 队列

注意 v6.0 后取消了 global-in 和 global-out 接口，使用 global 代替。

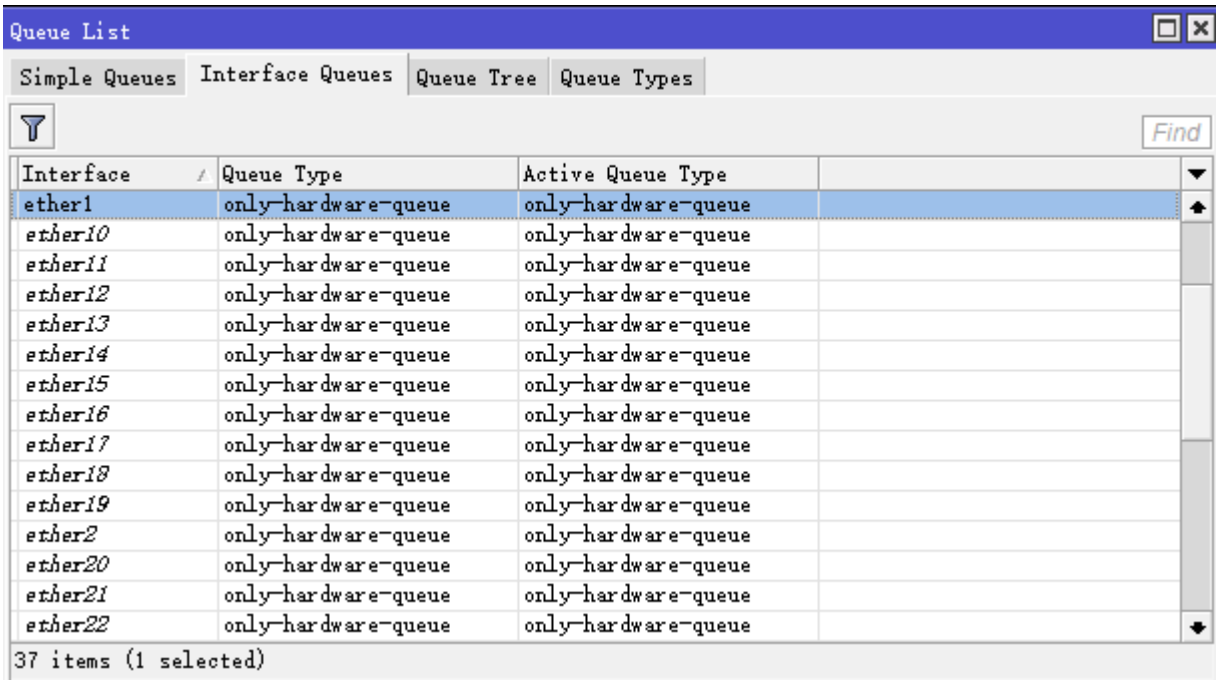
RouterOS 中有两种方式配置队列：

- **/queue simple** – 用于简单的队列配置，如直接对单个用户的上下行带宽控制，队列的时间计划任务。
- **/queue tree** – 为执行高级的队列任务，如全局的优先策略，用户组带宽控制，需从/ip firewall mangle 标记数据报中调用

12.2 Interface Queue（接口队列）

操作路径：/queue interface

数据流在发送到接口之前，会被 queue 处理，该菜单下将显示 RouterOS 当前所有可以获得的网卡接口，并允许修改网卡接口的队列类型，但不能创建接口，因为该列表是自动生成的。



描述

属性	描述
interface (*字符*)	接口名称，只读参数
queue (字符; 默认:)	接口的 **Queue type**（队列类型），可以为特殊接口配置需要的队列类型

12.3 Queue Type（队列类型）

操作路径: ***/queue type***

Queue type 定义了队列类型，在该配置目录下，可以创建自己的队列类型，并可以被**/queue tree**，**/queue simple** 或 **/queue interface** 使用

下面是 RouterOS 默认的队列类型：

```
[admin@MikroTik] > /queue type print
Flags: * - default
 0 * name="default" kind=pfifo pfifo-limit=50

 1 * name="ethernet-default" kind=pfifo pfifo-limit=50

 2 * name="wireless-default" kind=sfq sfq-perturb=5 sfq-allot=1514

 3
* name="synchronous-default" kind=red red-limit=60 red-min-threshold=10 red-max-t
hreshold=50 red-burst=20 red-avg-packet=1000

 4 * name="hotspot-default" kind=sfq sfq-perturb=5 sfq-allot=1514

 5
* name="pcq-upload-default" kind=pcq pcq-rate=0 pcq-limit=50KiB pcq-classifier=sr
c-address pcq-total-limit=2000KiB pcq-burst-rate=0 pcq-burst-threshold=0 pcq-burs
t-time=10s pcq-src-address-mask=32
     pcq-dst-address-mask=32 pcq-src-address6-mask=128 pcq-dst-address6-mask=128

 6
* name="pcq-download-default" kind=pcq pcq-rate=0 pcq-limit=50KiB pcq-classifier=
dst-address pcq-total-limit=2000KiB pcq-burst-rate=0 pcq-burst-threshold=0 pcq-bu
rst-time=10s pcq-src-address-mask=32
     pcq-dst-address-mask=32 pcq-src-address6-mask=128 pcq-dst-address6-mask=128

 7 * name="only-hardware-queue" kind=none

 8 * name="multi-queue-ethernet-default" kind=mq-pfifo mq-pfifo-limit=50

 9 * name="default-small" kind=pfifo pfifo-limit=10
```

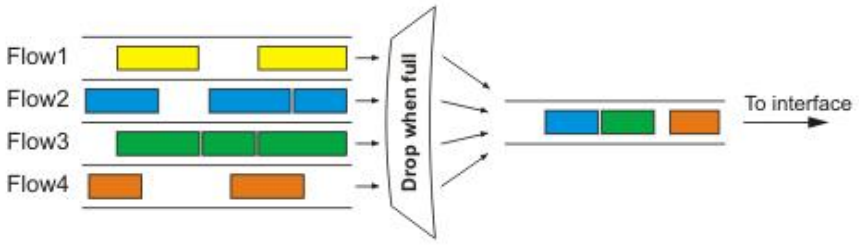
从 v5.8 开始，新的队列类型 **only-hardware-queue**，将 kind 定义为 none，所有 RouterBOARD 将默认配置该类型到接口上，即交给硬件驱动处理。在 interface queue 章节已经提到，数据流在发送到接口之前，会被 queue 处理，事实上基于硬件驱动完成队列处理，效率比软件更快（特别是在 SMP 硬件中）。

only-hardware-queue 通过指定接口的 queue type 为 none，所有数据包控制直接下发给 RouterBOARD 硬件驱动，绕过软件处理。这样能更快，使用更少的资源消耗处理数据流，但同样需要硬件驱动支持，因此该功能只被部分 RouterBOARD 接口驱动支持。

multi-queue-ethernet-default 有助于 SMP 系统（Symmetrical Multi-Processing，对称多处理）下以太网接口支持多传输队列，即在/system resource 下查看网卡的 IRQ 中断数量，参考 IRQ 中断。

PFIFO/BFIFO

该队列类型是基于先进先出算法(FIFO：First-In First-Out)。PFIFO 和 BFIFO 的区别在于一个是以数据包为单位，另一个是以字节为单位。用来定义一个 FIFO 队列可以容纳多少数据的，当该队列容纳数据满后，不能再加入排队，包/字节会被丢弃，队列长度可以增加，但过大会增加处理时间。

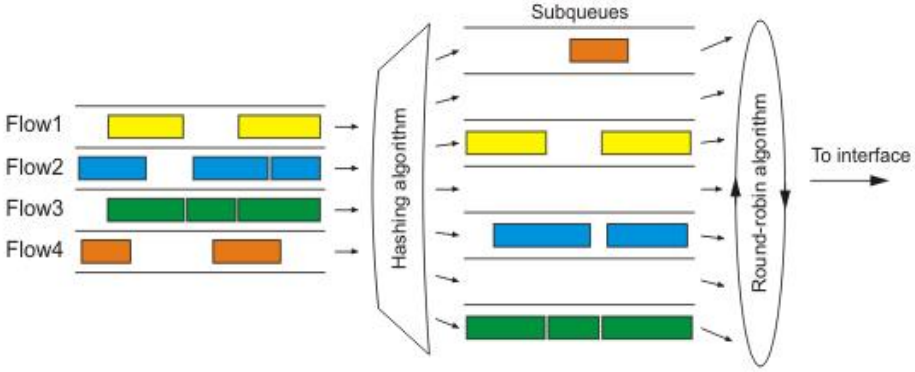


mq-pfifo ，即上节提到的 **multi-queue-ethernet-default**，应用在 SMP 系统，要求系统满足多 CPU 且网卡支持多队列

SFQ

随机公平排序（SFQ）不会一开始就对流量限制。它的主旨是当你的连接完全满的时候均衡业务流（TCP 会话或者 UDP 流）。

SFQ 的公平性是由散列法和 round-robin 算法保证的。散列算法把会话流分成一个有限数量的子队列。在 **sfq-perturb** 时间之后散列算法改变并划分会话流为其他子队列。Round-robin 算法把从每个子队列的 **pcq-allot** 字节按照顺序出队列。

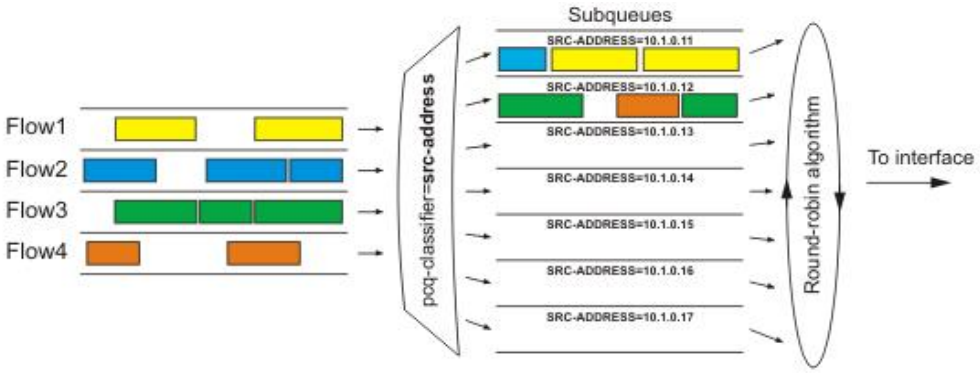


整个 SFQ 队列可以容纳 128 个数据报并且对这些包有 1024 个子队列可用。对拥挤的连接使用 SFQ 可以保证一些连接不至于空等待（starve）。

PCQ

为了解决 SFQ 的不完美，每次连接排序 Per Connection Queuing (PCQ)便产生了。它是唯一一种能限流的无等级排序类型。是一种去掉了随机特性的进化版 SFQ。PCQ 也会根据 **pcq-classifier** 参数产生子队列。每个子队列都有一个 **pcq-rate** 的数据率限制和 **pcq-limit** 大小的数据报。PCQ 队列的总大小不能大于 **pcq-total-limit** 包。

以下实例说明了 PCQ 对数据报的用法，以它们的源地址分类。

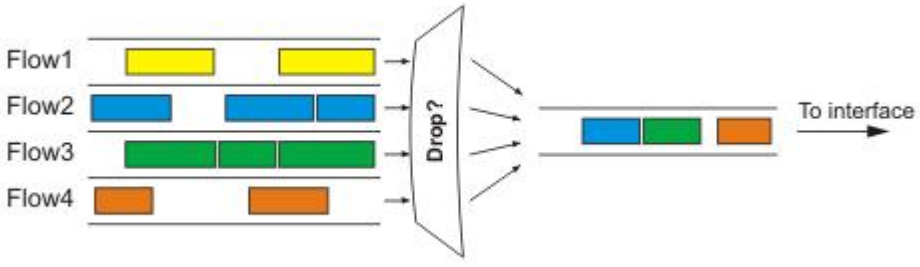


如果你以 **src-address** 对包分类那么所有带有不同源 IP 地址的包将被集合在不同的子队列中。现在你可以使用 **pcq-rate** 参数对每一个子队列进行限制或均衡。或许最重要的部分是决定我们到底应该把这个队列附属到哪个接口上。如果我们把它依附在本地接口上，那么所有来自公网接口的数据流都将以 src-address（很可能这不是我们想要的）地址分组；相反地如果我们把它依附到公共接口，所有来自我们客户的数据都会以 src-address 分组——于是我们可以很容易的限制或者均衡客户的上载。

用 **pcq-classifier** 分类后为了在子队列中均衡速率，设置 **pcq-rate** 为 **0** 几乎不限制，PCQ 也可以用来对多用户动态均衡或者形成流量，

RED

随机早先探测（RED）是一种通过控制平均队列长度避免网络拥塞的排序机制。当平均队列长度达到 **red-min-threshold** 时，RED 随机选择该丢弃哪个包。当平均队列长度变长时，堆砌多少包数的可能性会增加。如果平均队列长度达到 **red-max-threshold**，则丢弃该包。尽管如此，也存在真实队列长度（非平均的）远大于 **red-max-threshold** 时，丢弃所有超过 **red-limit** 的数据报的情况。



注：RED 应用在高数据率的拥挤的连接上，它在 TCP 协议上工作的很好，但在 UDP 上就没那么理想了。

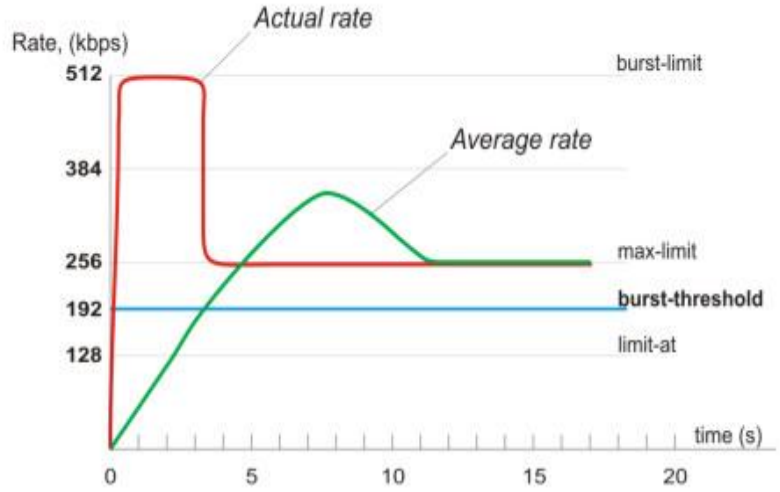
属性描述

bfifo-limit (整数; 默认: **15000**) - BFIFO 队列可以容纳的最大字节数
kind (bfifo | pcq | pfifo | red | sfq) – 选择队列控制类型
bfifo - *字节先进先出*
pcq – *每次连接队列*
pfifo – *数据报先进先出*
red – *随机早先探测*
sfq - *随机公平队列*
name (名称) - 队列类型相关名称
pcq-classifier (dst-address | dst-port | src-address | src-port; 默认: **""**) - PCQ 对其子队列进行分组的分类器。可以同时被数个分类器使用。例如：src-address，src-port 可使用不同源地址和源端口把所有包分为独立的子队列
pcq-limit (*整数*; 默认: **50**) - 可以容纳一个单个 PCQ 子队列的包的数目
pcq-rate (*整数*; 默认: **0**) - 对每个子队列允许的最大数据率。 **0** 值指的是没有任何限制
pcq-total-limit (*整数*; 默认: **2000**) - 可以容纳整个 PCQ 队列的包的数目
pfifo-limit (*整数*) - PFIFP 队列可以容纳包的最大数目
red-avg-packet (整数; 默认: **1000**) - 被 RED 用来对平均队列长度计算
red-burst (*整数*) - 用来决定平均队列长度被真实队列长度影响的快慢的字节值。较长的值将减慢 RED 的计算速度——较长的脉冲串也是允许的
red-limit (*整数*) - 以字节计算。如果真实队列长度（非平均值）超过了这个值那么所有大于这个值的包都将被丢弃。
red-max-threshold (整数) - 以字节计算。数据报标记概率最高的平均队列长度
red-min-threshold (*整数*) - 当平均 RED 队列长度达到这个值时，数据报标记才有可能
sfq-allot (*整数*; 默认: **1514**) - 在一个 round-robin 循环中从子队列发出的字节数
sfq-perturb (*整数*; 默认: **5**) - 以秒计时。指定改变 SFQ 的散列算法的频率

Bursts

脉冲串用来在一段很短的时间允许更高数据率。每 1/16 **burst-time** 时间，路由器都会计算每个类在上一个 **burst-time** 时间的平均数据率。如果这个平均数据率小于 **burst-threshold**，脉冲串就会被启用且实际数据率达到 **burst-limit** bps，否则实际数据率将跌至 **max-limit** 或 **limit-at**。

让我们考虑如果我们有个 **max-limit**=256000， **burst-time**=8， **burst-threshold**=192000 以及 **burst-limit**=512000 的设置情况。当一个用户通过 HTTP 下载一个档，我们可以观察到这样的现象：



在最开始的 8 秒中平均数据率是 0bps 因为在应用队列规则前没有流量通过。由于这个平均数据率小与 **burst-threshold** (192kbps)，所以脉冲串会被使用。在第一秒之后，平均数据率为(0+0+0+0+0+0+0+512)/8=64kbps，低于 **burst-threshold**。在第二秒后，平均数据率为(0+0+0+0+0+0+512+512)/8=128kbps。在第三秒之后达到临界点此时平均数据率变得大于 **burst-threshold**。这个时候脉冲串将被禁用且当前数据率降至 **max-limit** (256kbps)。

12.4 Burst 突发值

Burst 原理

Burst 允许满足队列需要增加的带宽，甚至要求速率在有限的时间内大于 MIR（max-limit），Burst 发生仅当队列的 average-rate 在 burst-time 时间内小于 burst-threshold。Burst 停止当队列的 average-rate 在 burst-limit 时间内大于或者等于 burst-threshlod

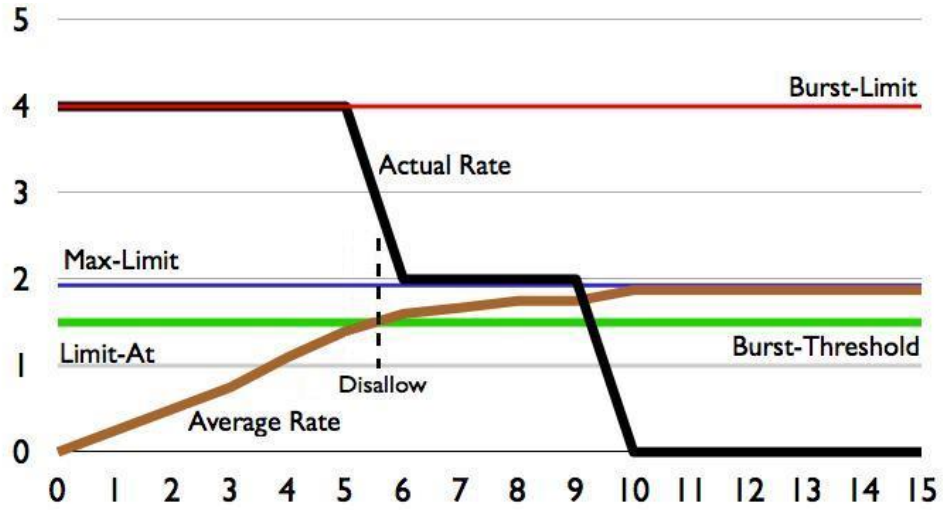
Burst 原理是，通过计算如果允许用 burst-limit 代替 max-limit，当 burst 时间结束后 max-limit 恢复

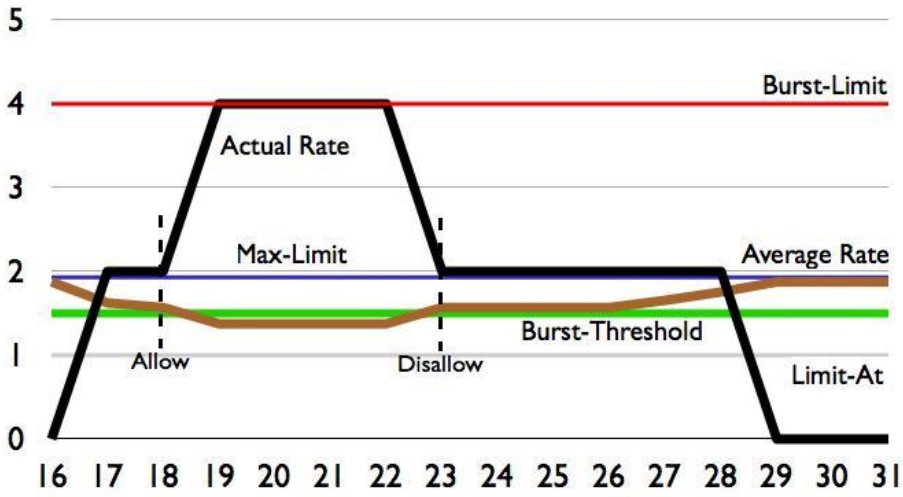
1. **burst-limit** (整型)：能被 burst 允许达到的最大上传和下载数据
2. **burst-time** (时间)：一段时间，单位秒，用于平均速率的计算 (并非实际的 burst 时间长度)
3. **burst-threshold** (整型)：这个参数是通过计算后比对，并开关 burst 功能
4. **average-rate** (隐含只读参数)：路由器计算平均速率根据 **burst-time** 划分为 16 份，每份都会计算出一个平均速率进行比对
5. **actual-rate** (隐含只读参数)：队列的实际传输带宽

Burst 事例

我们设置的 Queue 速率参数：**limit-at=1M** , **max-limit=2M** , **burst-threshold=1500k** , **burst-limit=4M Burst-time=16s**

客户将会下载一个4MByte（32Mbit，队列单位是bit）数据，下载将从0秒开始，第二次下载将开始于第17秒，最后一分钟传输将停止。





如同我们看到的客户要求的带宽 burst 在 6 秒钟能达到 4Mbps。这个最长的 burst 时间具有一个值（*最长突发时间 = burst-threshold * burst-time / burst-limit*）。很快 burst 用完突发时间，剩下的数据下载将到 2Mbps。在 9 秒钟后数据被下载完，一段时间没有流量，Burst 有 7 秒钟的空闲，并重新计算，第 16 秒开始将有新的下载开始。

注:从这个事例我们证明可以发生在下载的中间部分，Burst 持续了 4 秒钟。

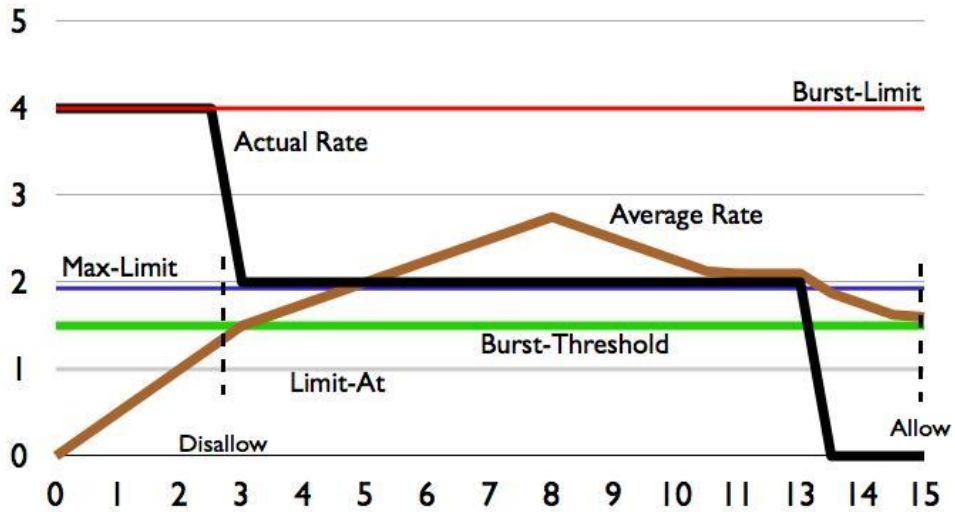
每个 Average rate（平均速率）是根据 burst time 的 1/16，因此这个事例是 1 秒钟 计算一次平均速率

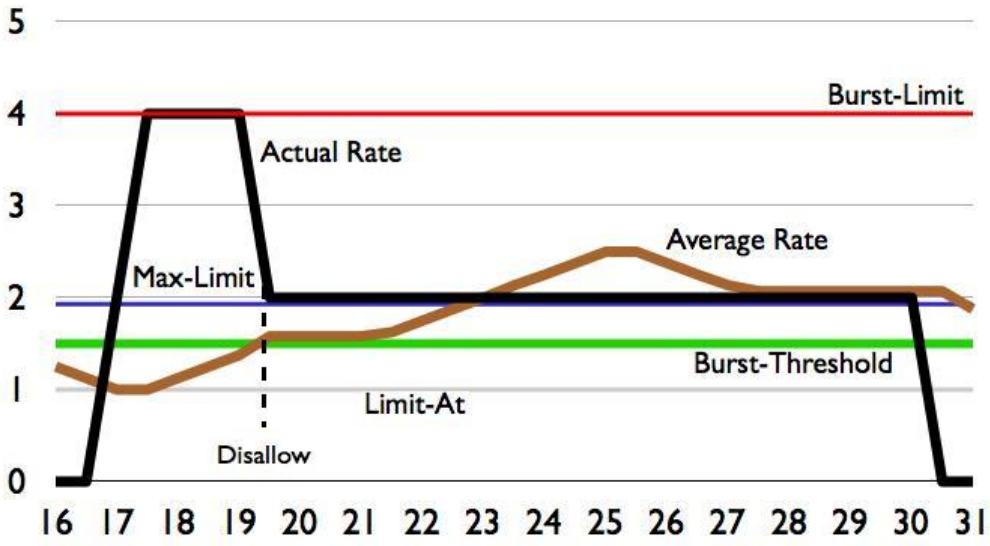
时间	average-rate	burst	实际速率
0	(0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0)/16=0Kbps	average-rate < burst-threshold → Burst 开启	4Mbps
1	(0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+4)/16=250Kbps	average-rate < burst-threshold → Burst 开启	4Mbps
2	(0+0+0+0+0+0+0+0+0+0+0+0+0+0+4+4)/16=500Kbps	average-rate < burst-threshold → Burst 开启	4Mbps
3	(0+0+0+0+0+0+0+0+0+0+0+0+0+4+4+4)/16=750Kbps	average-rate < burst-threshold → Burst 开启	4Mbps

4	(0+0+0+0+0+0+0+0+0+0+0+4+4+4+4)/16=1000Kbps	average-rate < burst-threshold → Burst 开启	4Mbps
5	(0+0+0+0+0+0+0+0+0+0+0+4+4+4+4+4)/16=1250Kbps	average-rate < burst-threshold → Burst 开启	4Mbps
6	(0+0+0+0+0+0+0+0+0+0+4+4+4+4+4+4)/16=1500Kbps	average-rate = burst-threshold → Burst 关闭	2Mbps
7	(0+0+0+0+0+0+0+0+0+4+4+4+4+4+4+2)/16=1625Kbps	average-rate = burst-threshold → Burst 关闭	2Mbps
8	(0+0+0+0+0+0+0+0+4+4+4+4+4+4+2+2)/16=1750Kbps	average-rate = burst-threshold → Burst 关闭	2Mbps
9	(0+0+0+0+0+0+0+4+4+4+4+4+4+2+2+2)/16=1750Kbps	average-rate = burst-threshold → Burst 关闭	2Mbps
10	(0+0+0+0+0+0+4+4+4+4+4+4+2+2+2+2)/16=1875Kbps	average-rate = burst-threshold → Burst 关闭	0Mbps
11	(0+0+0+0+0+4+4+4+4+4+4+2+2+2+2+0)/16=1875Kbps	average-rate = burst-threshold → Burst 关闭	0Mbps
12	(0+0+0+0+4+4+4+4+4+4+2+2+2+2+0+0)/16=1875Kbps	average-rate = burst-threshold → Burst 关闭	0Mbps
13	(0+0+0+4+4+4+4+4+4+2+2+2+2+0+0+0)/16=1875Kbps	average-rate = burst-threshold → Burst 关闭	0Mbps
14	(0+0+4+4+4+4+4+4+2+2+2+2+0+0+0+0)/16=1875Kbps	average-rate = burst-threshold → Burst 关闭	0Mbps
15	(0+4+4+4+4+4+4+2+2+2+2+0+0+0+0+0)/16=1875Kbps	average-rate = burst-threshold → Burst 关闭	0Mbps
16	(4+4+4+4+4+4+2+2+2+2+0+0+0+0+0+0)/16=1875Kbps	average-rate = burst-threshold → Burst 关闭	0Mbps
17	(4+4+4+4+4+2+2+2+2+0+0+0+0+0+0+0)/16=1625Kbps	average-rate = burst-threshold → Burst 关闭	2Mbps
18	(4+4+4+4+2+2+2+2+0+0+0+0+0+0+0+2)/16=1500Kbps	average-rate = burst-threshold → Burst 关闭	2Mbps
19	(4+4+4+2+2+2+2+0+0+0+0+0+0+0+2+2)/16=1375Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps
20	(4+4+2+2+2+2+0+0+0+0+0+0+0+2+2+4)/16=1375Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps
21	(4+2+2+2+2+0+0+0+0+0+0+0+2+2+4+4)/16=1375Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps
22	(2+2+2+2+0+0+0+0+0+0+0+2+2+4+4+4)/16=1375Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps

23	(2+2+2+0+0+0+0+0+0+0+2+2+4+4+4+4)/16=1500Kbps	average-rate = burst-threshold → Burst not allowed	2Mbps
24	(2+2+0+0+0+0+0+0+0+2+2+4+4+4+4+2)/16=1500Kbps	average-rate = burst-threshold → Burst not allowed	2Mbps
25	(2+0+0+0+0+0+0+0+2+2+4+4+4+4+2+2)/16=1500Kbps	average-rate = burst-threshold → Burst not allowed	2Mbps
26	(0+0+0+0+0+0+0+2+2+4+4+4+4+2+2+2)/16=1500Kbps	average-rate = burst-threshold → Burst not allowed	2Mbps
27	(0+0+0+0+0+0+2+2+4+4+4+4+2+2+2+2)/16=1625Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps
28	(0+0+0+0+0+2+2+4+4+4+4+2+2+2+2+2)/16=1750Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps
29	(0+0+0+0+2+2+4+4+4+4+2+2+2+2+2+2)/16=1875Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps
30	(0+0+0+2+2+4+4+4+4+2+2+2+2+2+2+0)/16=1875Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps
31	(0+0+2+2+4+4+4+4+2+2+2+2+2+2+0+0)/16=1875Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps

当 Burst-time=8s





如果我们减少 burst-time 为 8 秒，我们能看到在这个事例中 burst 仅在下载开始

每个 Average rate（平均速率）是根据 burst time 的 1/16，因此这个事例是 0.5 秒钟计算一次平均速率

时间	average-rate	burst	实际速率
0.0	(0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0)/8=0Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
0.5	(0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+2)/8=250Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
1.0	(0+0+0+0+0+0+0+0+0+0+0+0+0+0+2+2)/8=500Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
1.5	(0+0+0+0+0+0+0+0+0+0+0+0+0+2+2+2)/8=750Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
2.0	(0+0+0+0+0+0+0+0+0+0+0+0+2+2+2+2)/8=1000Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
2.5	(0+0+0+0+0+0+0+0+0+0+0+2+2+2+2+2)/8=1250Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
3.0	(0+0+0+0+0+0+0+0+0+0+2+2+2+2+2+2)/8=1500Kbps	average-rate = burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
3.5	(0+0+0+0+0+0+0+0+0+2+2+2+2+2+2+1)/8=1625Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
4.0	(0+0+0+0+0+0+0+0+2+2+2+2+2+2+1+1)/8=1750Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
4.5	(0+0+0+0+0+0+0+2+2+2+2+2+2+1+1+1)/8=1875Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
5.0	(0+0+0+0+0+0+2+2+2+2+2+2+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold	2Mbps (1Mb per

		→ Burst not allowed	0,5sek)
5.5	(0+0+0+0+0+2+2+2+2+2+2+1+1+1+1+1+1)/8=2125Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
6.0	(0+0+0+0+2+2+2+2+2+2+1+1+1+1+1+1+1)/8=2250Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
6.5	(0+0+0+2+2+2+2+2+2+1+1+1+1+1+1+1+1)/8=2375Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
7.0	(0+0+2+2+2+2+2+2+1+1+1+1+1+1+1+1+1)/8=2500Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
7.5	(0+2+2+2+2+2+2+1+1+1+1+1+1+1+1+1+1)/8=2625Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
8.0	(2+2+2+2+2+2+1+1+1+1+1+1+1+1+1+1+1)/8=2750Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
8.5	(2+2+2+2+2+1+1+1+1+1+1+1+1+1+1+1+1)/8=2625Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
9.0	(2+2+2+2+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2500Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
9.5	(2+2+2+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2375Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
10.0	(2+2+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2250Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
10.5	(2+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2125Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
11.0	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
11.5	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
12.0	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
12.5	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
13.0	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps (0Mb per 0,5sek)
13.5	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+0)/8=1875Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps (0Mb per 0,5sek)
14.0	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+0+0)/8=1750Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps (0Mb per 0,5sek)

14.5	(1+1+1+1+1+1+1+1+1+1+1+1+1+0+0+0)/8=1625Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps (0Mb per 0,5sek)
15.0	(1+1+1+1+1+1+1+1+1+1+1+1+0+0+0+0)/8=1500Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps (0Mb per 0,5sek)
15.5	(1+1+1+1+1+1+1+1+1+1+1+0+0+0+0+0)/8=1375Kbps	average-rate < burst-threshold → Burst **is** allowed	0Mbps (0Mb per 0,5sek)
16.0	(1+1+1+1+1+1+1+1+1+1+0+0+0+0+0+0)/8=1250Kbps	average-rate < burst-threshold → Burst is allowed	0Mbps (0Mb per 0,5sek)
16.5	(1+1+1+1+1+1+1+1+1+0+0+0+0+0+0+0)/8=1125Kbps	average-rate < burst-threshold → Burst is allowed	0Mbps (0Mb per 0,5sek)
17.0	(1+1+1+1+1+1+1+1+0+0+0+0+0+0+0+0)/8=1000Kbps	average-rate < burst-threshold → Burst is allowed	**2Mbps** (1Mb per 0,5sek)
17.5	(1+1+1+1+1+1+1+0+0+0+0+0+0+0+0+1)/8=1000Kbps	average-rate < burst-threshold → Burst is allowed	**4Mbps** (2Mb per 0,5sek)
18.0	(1+1+1+1+1+1+0+0+0+0+0+0+0+0+1+2)/8=1125Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
18.5	(1+1+1+1+1+0+0+0+0+0+0+0+0+1+2+2)/8=1250Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
19.0	(1+1+1+1+0+0+0+0+0+0+0+0+1+2+2+2)/8=1375Kbps	average-rate < burst-threshold → Burst is allowed	4Mbps (2Mb per 0,5sek)
19.5	(1+1+1+0+0+0+0+0+0+0+0+1+2+2+2+2)/8=1500Kbps	average-rate = burst-threshold → Burst **not** allowed	2Mbps (1Mb per 0,5sek)
20.0	(1+1+0+0+0+0+0+0+0+0+1+2+2+2+2+1)/8=1500Kbps	average-rate = burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
20.5	(1+0+0+0+0+0+0+0+0+1+2+2+2+2+1+1)/8=1500Kbps	average-rate = burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
21.0	(0+0+0+0+0+0+0+0+1+2+2+2+2+1+1+1)/8=1500Kbps	average-rate = burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
21.5	(0+0+0+0+0+0+0+1+2+2+2+2+1+1+1+1)/8=1625Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
22.0	(0+0+0+0+0+0+1+2+2+2+2+1+1+1+1+1)/8=1750Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
22.5	(0+0+0+0+0+1+2+2+2+2+1+1+1+1+1+1)/8=1875Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
23.0	(0+0+0+0+1+2+2+2+2+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
23.5	(0+0+0+1+2+2+2+2+1+1+1+1+1+1+1+1)/8=2125Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)

24.0	(0+0+1+2+2+2+2+1+1+1+1+1+1+1+1+1)/8=2250Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
24.5	(0+1+2+2+2+2+1+1+1+1+1+1+1+1+1+1)/8=2375Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
25.0	(1+2+2+2+2+1+1+1+1+1+1+1+1+1+1+1)/8=2500Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
25.5	(2+2+2+2+1+1+1+1+1+1+1+1+1+1+1+1)/8=2500Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
26.0	(2+2+2+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2375Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
26.5	(2+2+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2250Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
27.0	(2+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2125Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
27.5	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
28.0	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
28.5	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
29.0	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
29.5	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
30.0	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	2Mbps (1Mb per 0,5sek)
30.5	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1)/8=2000Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps (0Mb per 0,5sek)
31.0	(1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+0)/8=1875Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps (0Mb per 0,5sek)

12.5 Simple Queue 简单队列

操作路径: /queue simple

Simple Queue简单队列是对IP地址或子网段进行流量限制,主机流控最简单方法就是使用/queue simple，simple queue 采用的 FIFO 先进先出算法，序列越靠前越优先执行。simple queue 支持建立高级 QoS 应用，如 mangle 标记和等级队列。

在 v5 版本的/queue simple 创建一个流控配置项目，会分别有三个独立的队列，分别是 global-in, global-out 和 global-total。如果在/queue simple 创建一个默认队列规则（无流控限制、queue type 为默认），并且该队列没有子队列，即这样的队列实际上没有创建。如果队列只配置了 upload/download 流控属性，global-total 队列可以被忽略。如果仔细观察，当建立一条 queue simple 规则同时在 queue tree 可以瞬间看到 3 条规则的建立，然后被隐藏到后台，即 queue simple 被建立在 queue tree 下。

V6 版本改进后，不会再有 global-in 和 global-out，由“global”代替，位于“input”链表后。

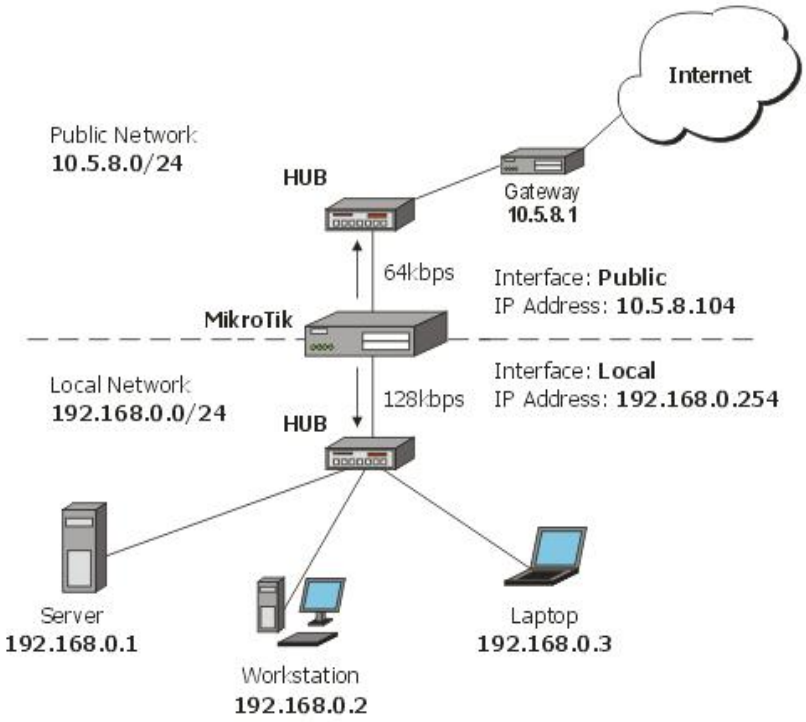
Simple queues 是有序对队列即 FIFO，每个数据报都必须经过每一个队列处理，直到最后一条队列规则，即如果有 1000 条队列，匹配的队列规则是排列在第 1000 条，那么数据报过经过前面 999 条后，才能到达该规则。因此 Simple queue 在出现大量队列规则后，处理效率会降低，v6 之前的版本，数据流会被 queue 重复处理两次，造成性能消耗更大（参考 12.5 章节）。

在 v6 大改动后，simple queues 优化了 RouterOS 在流控处理性能，全新的算法和编译到 linux 内核处理，且不再出现数据流两次被重复处理的问题，使得性能更好更快。Simple queues 仍然是顺序的 FIFO 执行，经过优化后，在创建 32 条流控规则，且 simple queue 在多 CPU 路由器下，性能比 v5 版本快 9 倍。

- P2P 流量队列
- 计划时间任务执行队列规则
- 优先级队列
- 从 */ip firewall mangle* 使用多重包标记
- 双向流控（对上行和下行的带宽限制）

Simple queue 应用举例（仅适用于 v5 和之前版本）

下面假设我们想要对网络 192.168.0.0/24 流量限制为：下行 1Mb 上行 512kb，这里我们需要让服务器 192.168.0.1 不受流量控制。网络的基本设置如图：



这里我们使用（simple queue）简单队列，首先我们配置 RouterOS 的 IP 地址、网关和 NAT 等基本网络参数：

```
 [admin@MikroTik] ip address> print
Flags: X – disabled, I – invalid, D - dynamic
 #    ADDRESS              NETWORK          BROADCAST        INTERFACE
 0    192.168.0.254/24      192.168.0.0      192.168.0.255    Local
 1    10.5.8.104/24        10.5.8.0         10.5.8.255       Public
[admin@MikroTik] ip address>
```

路由配置：

```
[admin@MikroTik] ip route> print
Flags: X – disabled, A – active, D – dynamic,
C – connect, S – static, r – rip, b – bgp, o - ospf
 #       DST-ADDRESS          G GATEWAY           DISTANCE INTERFACE
 0 ADC 10.5.8.0/24                                    Public
 1 ADC 192.168.0.0/24                                 Local
 2 A S 0.0.0.0/0            r 10.5.8.1                Public
[admin@MikroTik] ip route>
```

最后不要忘记在 ip firewall nat 中配置 src-nat 的伪装或 nat，做地址转换操作。

为网络 192.168.0.0/24 的所有客户端添加一个限制下载流量为 2Mb 上传流量 1Mb 的简单队列规则。

```
[admin@MikroTik] queue simple> add name=Limit-Local target-address=192.168.0.0/24
max-limit=1000000/2000000
[admin@MikroTik] queue simple> print
Flags: X - disabled，  I - invalid，  D - dynamic
 0      name="Limit-Local" target-addresses=192.168.0.0/24 dst-address=0.0.0.0/0
        parent=none priority=8 queue=default/default   limit-at=0/0 max-limit=1000000/2000000
total-queue=default
[admin@MikroTik] queue simple>
```

max-limit 限制了最大可用带宽，从客户的角度看，参数 **target-addresses** 定义限制带宽的目标网络或者主机（也可以用逗号分隔开网络段或主机地址）。

这里不想让服务器受到我们添加上面规则的任何流量限制，我们可以通过添加一个没有任何限制的规则（**max-limit=0/0** 代表没有任何限制）并把它移到列表的顶部：

```
[admin@MikroTik] queue simple> add name=Server target-addresses=192.168.0.1/32
[admin@MikroTik] queue simple> print
Flags: X - disabled，  I - invalid，  D - dynamic
 0      name="Limit-Local" target-addresses=192.168.0.0/24 dst-address=0.0.0.0/0
        parent=none priority=8 queue=default/default limit-at=0/0 max-limit=65536/131072
total-queue=default

 1      name="Server" target-addresses=192.168.0.1/32 dst-address=0.0.0.0/0
```

```
      parent=none priority=8 queue=default/default limit-at=0/0 max-limit=0/0
total-queue=default
```

使用 move 命令将第二条规则移动到第一条，即从编号 1，移动到编号 0，编号越小优先级越高。

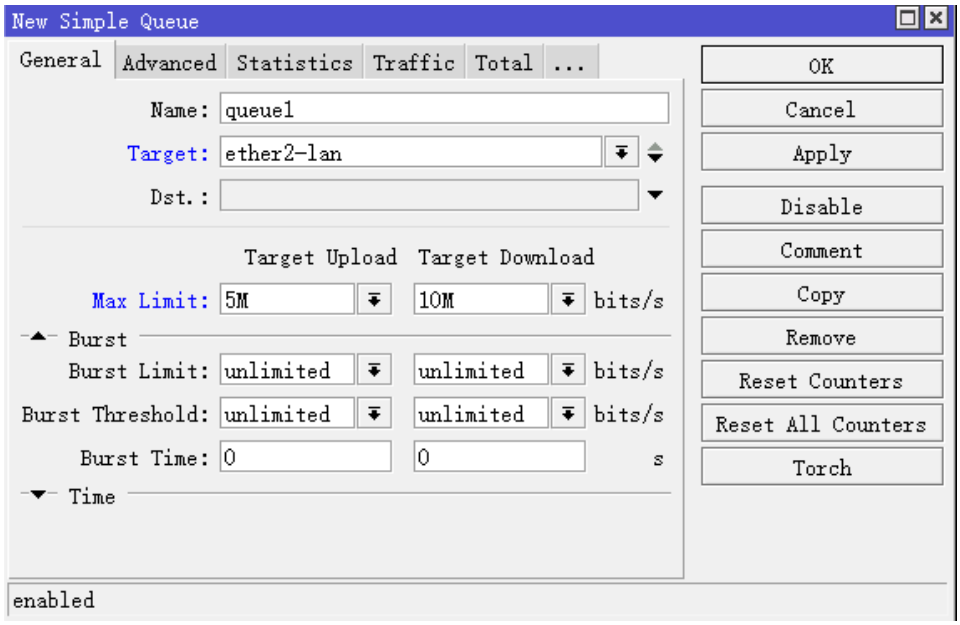
```
[admin@MikroTik] queue simple> move 1 0
[admin@MikroTik] queue simple> print
Flags: X - disabled,  I - invalid,  D - dynamic
 0    name="Server" target-addresses=192.168.0.1/32 dst-address=0.0.0.0/0
      parent=none priority=8 queue=default/default
      limit-at=0/0 max-limit=0/0 total-queue=default

 1    name="Limit-Local" target-addresses=192.168.0.0/24 dst-address=0.0.0.0/0
      parent=none priority=8 queue=default/default
      limit-at=0/0 max-limit=65536/131072 total-queue=default
[admin@MikroTik] queue simple>
```

Simple Queue 接口流控配置（适用于 v6）

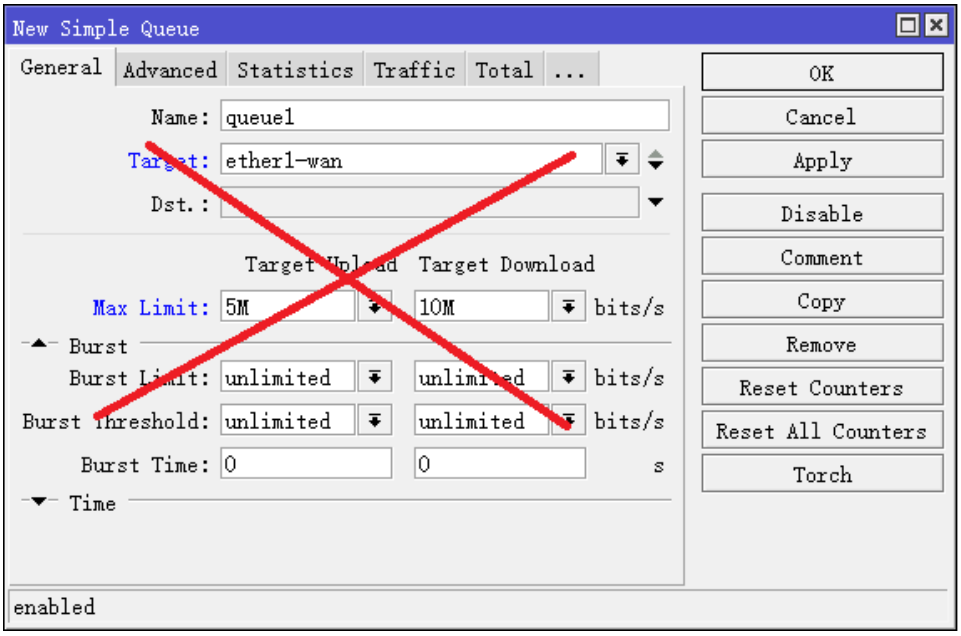
在 Simple Queue 菜单下做流控，我们可以选择接口，但新手对接口的流控可能认识不够，WAN 和 LAN 配置接口流控是有区别的

我们举例来说明，假如一条单线接入的网络，要求对带宽控制，上行 5M，下行 10M，在外网和内网接口的 simple Queue 是不一样的。

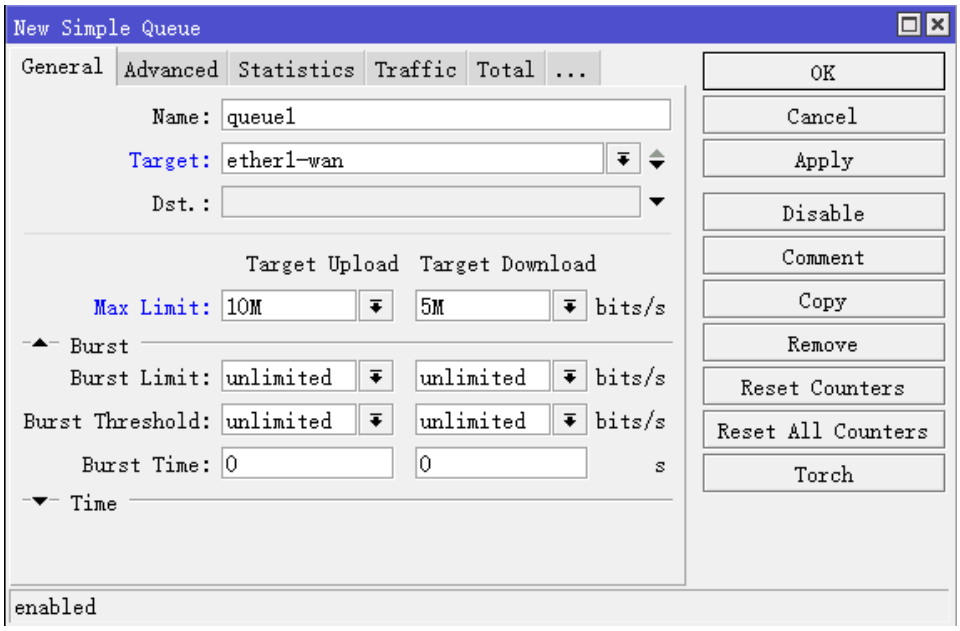
首先我们用 ether2-lan 内网口的带宽规则，target-upload=5M，target-download=10M，即上行 5M，下行 10M，这样能正确的控制 ether2-lan 接口下的带宽为上行 5M 和下行 10M



如果我们将接口选择外网接口 ether1-wan 上，实现同样的带宽控制策略，下面的配置是错误的，因为对于外网接口和内网接口的流量是相反的



下面是正确的配置，upload 是上行带宽 10M，download 是下行带宽 5M：



Simple Queue 配置父级流控无效问题

如果在 RouterOS v6 你使用 simple queue 配置 HTB 的规则，需要注意父级队列规则匹配后在子队列同样会被重复匹配检查。因此你必须在父级队列下包含同样的子队列策略，保证队列匹配完全生效。

例如，下面的实例如何正确的控制在 simple queue 的父级队列和子队列的流控，该配置基于 v6.34.3。下面是一个错误的实例，因为只有 192.168.88.2 主机会被流控，其他主机流控则会失效。

```
/queue simple
add max-limit=100M/100M name=GLOBAL target=192.168.88.0/24
add max-limit=10M/10M name=child1 parent=GLOBAL target=192.168.88.2/32
```

修正这个问题，需要添加与父级队列相同的子队列规则

```
/queue simple
add max-limit=100M/100M name=GLOBAL target=192.168.88.0/24
add max-limit=10M/10M name=child1 parent=GLOBAL target=192.168.88.2/32
add name=child2 parent=GLOBAL target=192.168.88.0/24
```

simple queue 的 CPU 负载问题

随着多核心 CPU 和超线程的发展，当前 RouterOS v6 后 simple queue 是基于多 CPU 负载均衡，即每条 simple queue 负载到一个 CPU。因此在配置 simple queue 时，需要注意配置规则与多 CPU 负载问题，下面实例告诉你配置 simple queue 的注意事项

假设我们硬件是多 CPU，下面的配置基于 v6.34.3，这个 simple queue 配置将 192.168.88.2 和 192.168.88.3 主机写入一条 simple queue 规则，这样被识别为一条线程，被分配到一个 CPU 处理。

```
/queue simple
add max-limit=100M/100M name=queue1 target=192.168.88.2/32,192.168.88.3/32
```

下面将两个主机分别写成两条规则的流控，修改后，可以同时负载到两个 CPU 处理，当然你可以根据自己需要调整规则保障 CPU 负载均衡

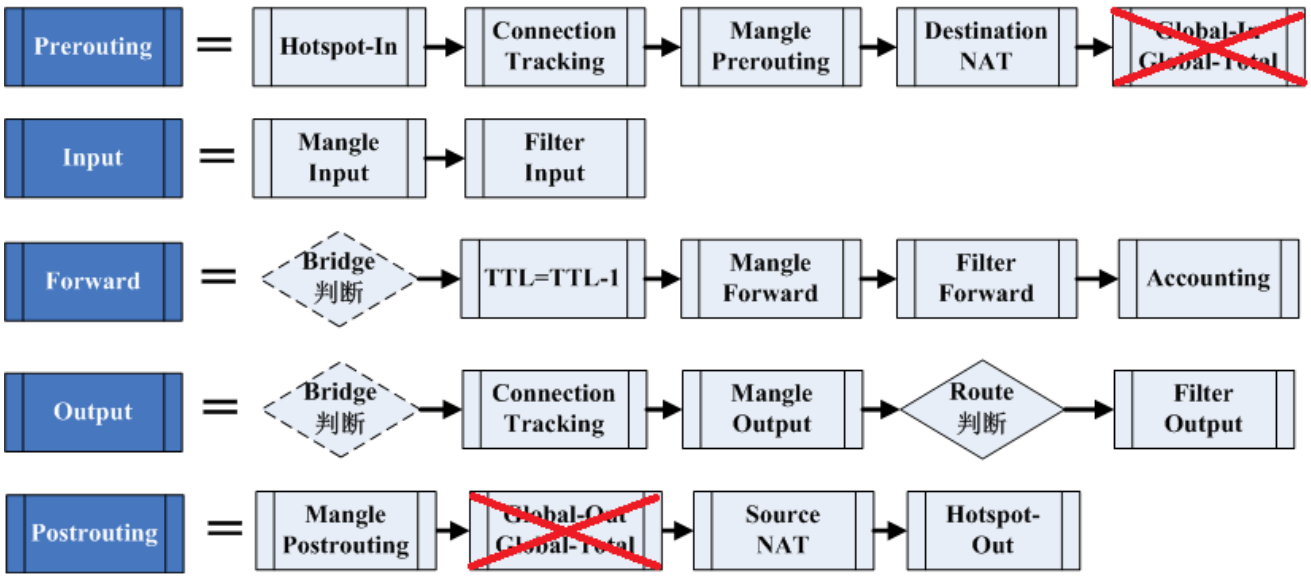
```
/queue simple
add max-limit=50M/50M name=queue1 target=192.168.88.2/32
add max-limit=50M/50M name=queue2 target=192.168.88.3/32
```

12.6 RouterOS v6.0 Queue 变动

为了提升 Queue 的处理性能，V6.0 对 Queue 流控进行了大改动，即将 simple queue 从 queue tree 中独立出来，我们需要关心的是 simple queue 和 queue tree 在分离后，实际网络环境中的处理流程，由于 v6 版本对 QoS 的重新设计，从 v3，v4，v5 等版本如果直接升级到 v6，会导致 simple queue 和 queue tree 的一些参数将无法识别，因此从 v6 之前的版本升级，涉及 queue 配置时需谨慎，可能会重新配置 queue 参数。

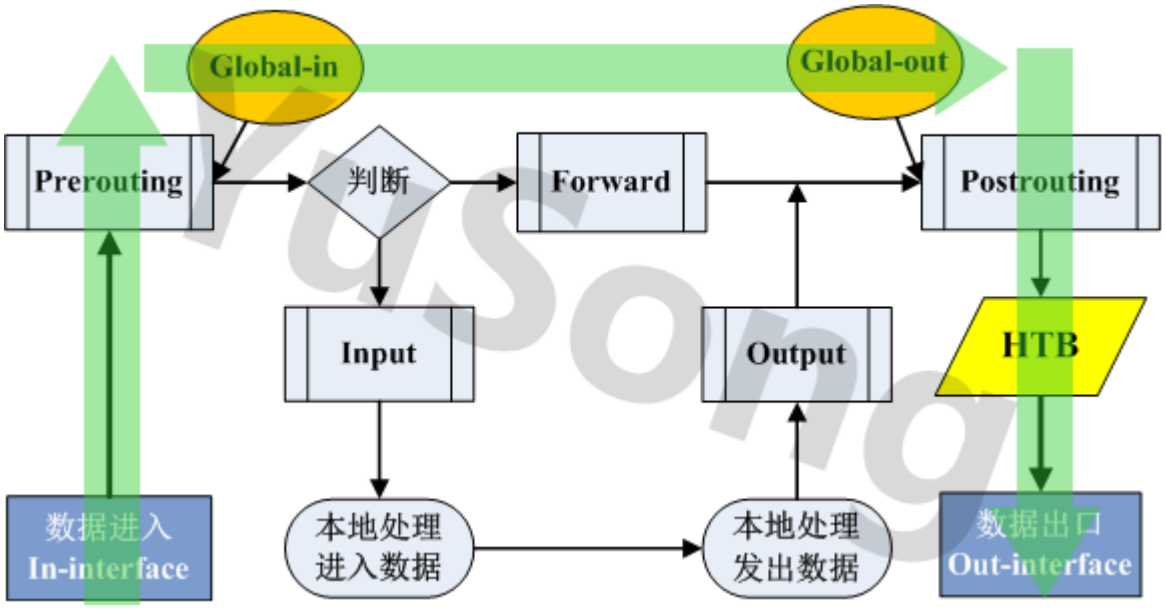
V6.0 前

我们看看在 v6.0 之前的链表结构图，可以看到 global-in 属于 prerouting 链表，global-out 属于 postrouting，但 v6.0 重新设计后被取消。



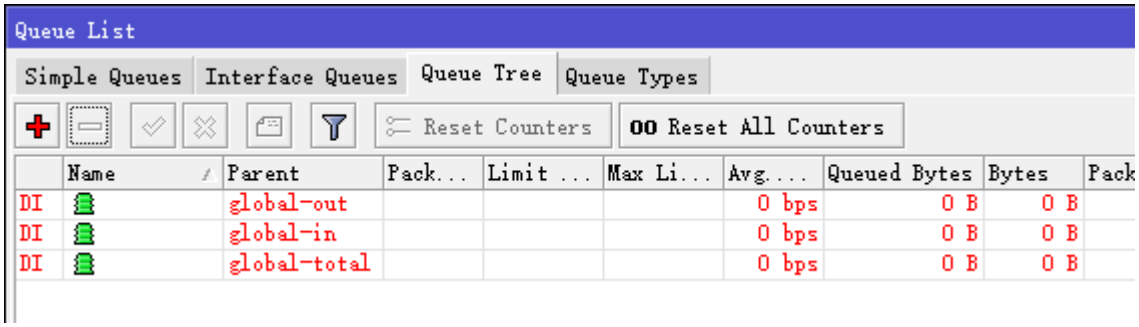
再看看，下面的 v6.0 前的流程图，global-in 包含在 prerouting 中，global-in 会处理通过路由（forward）和进入路由器（input）的数据，对于处理进入路由器（input）的数据被笼统的放在 Prerouting 链表中，造成 Prerouting 中的 Queue 模块究竟负责处理进入路由器（input）的数据，还是处理经过路由器（forward）的数据，并不明确。

global-out 会在 Postrouting 中，再一次处理通过路由器（forward）后的数据，Queue 队列会把一个 IP 数据重复处理 2 次。



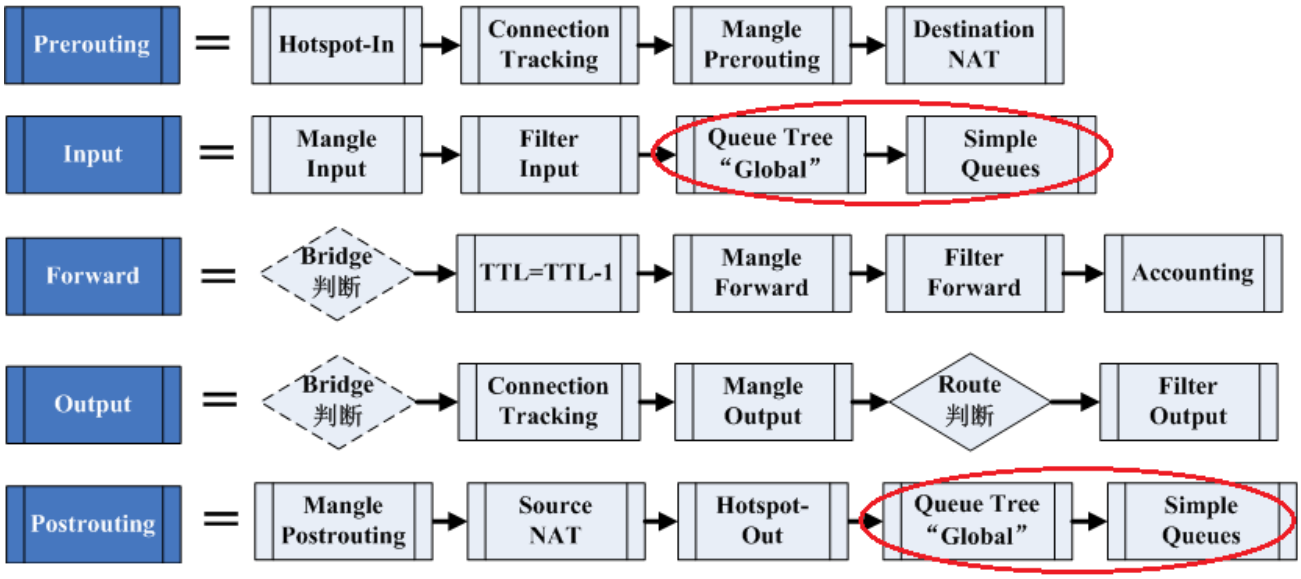
如上图，我们可以看到，当数据流通过路由器的 in-interface 进入后，在从 out-interface 转发出去，会被 Global-in 和 Global-out 先后处理两次，造成 CPU 过多的开销。

当我们在创建一条 simple queue 规则，刚生成规则时，可以在 queue tree 里瞬间看到三条规则出现，global-out，global-in 和 global-total，也就是 v6 前的版本在生成一条 simple queue 规则是会在内部的 queue tree 里面创建 3 条规则



V6.0 后

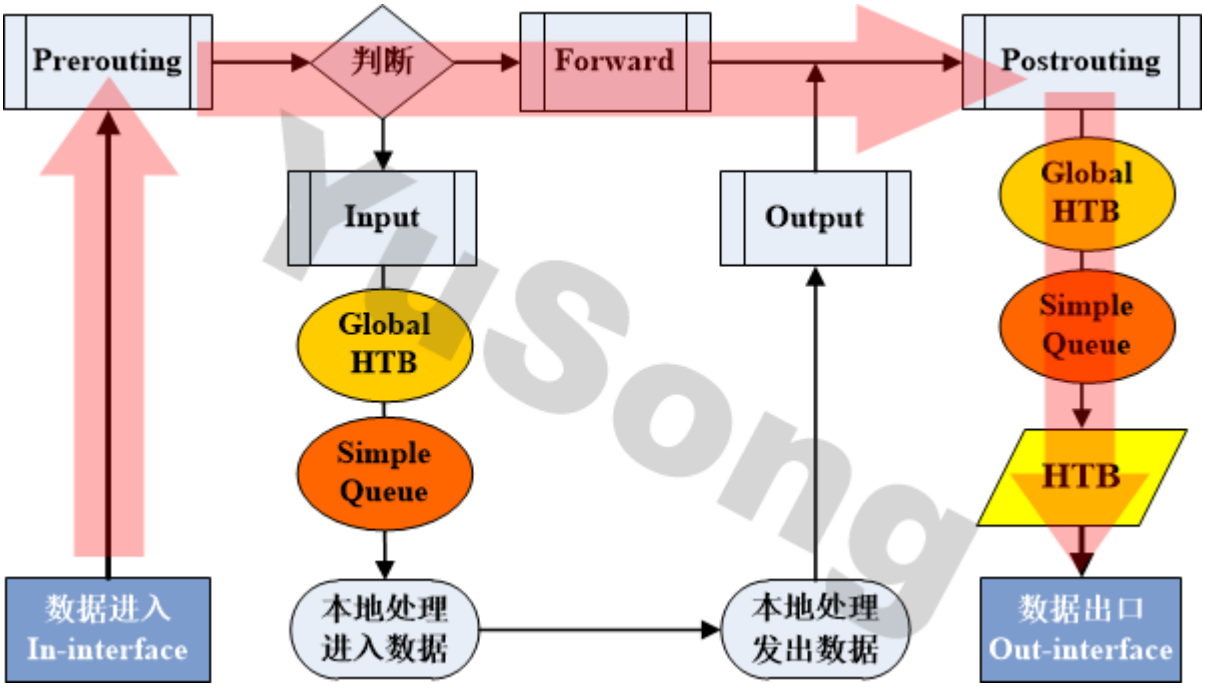
v6.0 后的数据流，将 global-in 和 global-out 合并为 global，并将 prerouting 的 queue 位置调整到 input 链表：



在 RouterOS 新的处理架构上 Queue tree 和 simple queue 会出现在 Input 和 Postrouting 两个链表中，且 Queue tree 会首先获得 IP 数据流，处理后再给 Simple Queue，但他们之间互相独立没有联系。

这样 simple queue 完全从 queue tree 分离，对于 simple queue 和 Global queue tree 传输流量能被两者分别独立的获取到，这样提供了建立双重 QoS 策略方案。

如下图所见，通过路由器的流量不会再被 Queue 重复处理，因为 Queue 模块被放到了 input 链表，而不是 Prerouting 链表中，明确了是控制进入路由器的 IP 数据流，而通过路由器的 IP 数据流会被 Postrouting 链表处理，这样减少了 CPU 的开销，这就是为什么 RouterOS v6 的流控处理性能比 v5 版本的高。



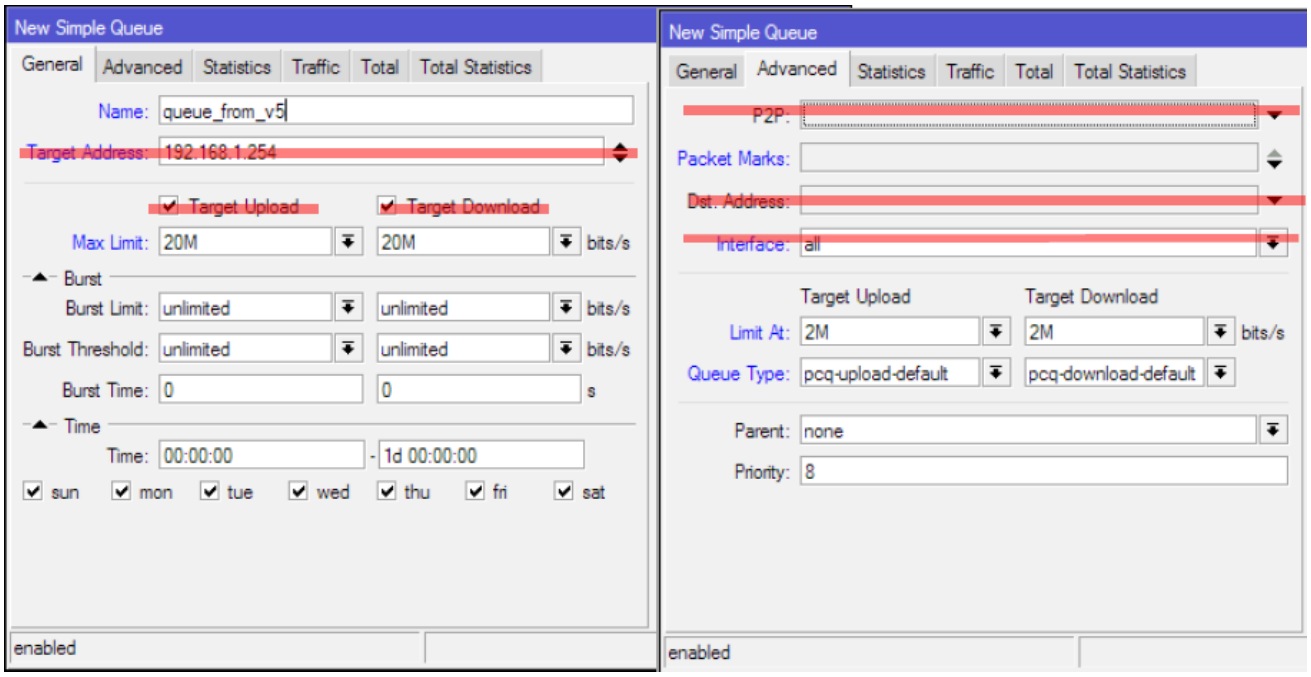
现在 Queue 模块被放置到了 src-nat 之后，对于 PCQ 策略在 src-nat 后更明确，能从 connection tracking 进行匹配。

可以总结以下：

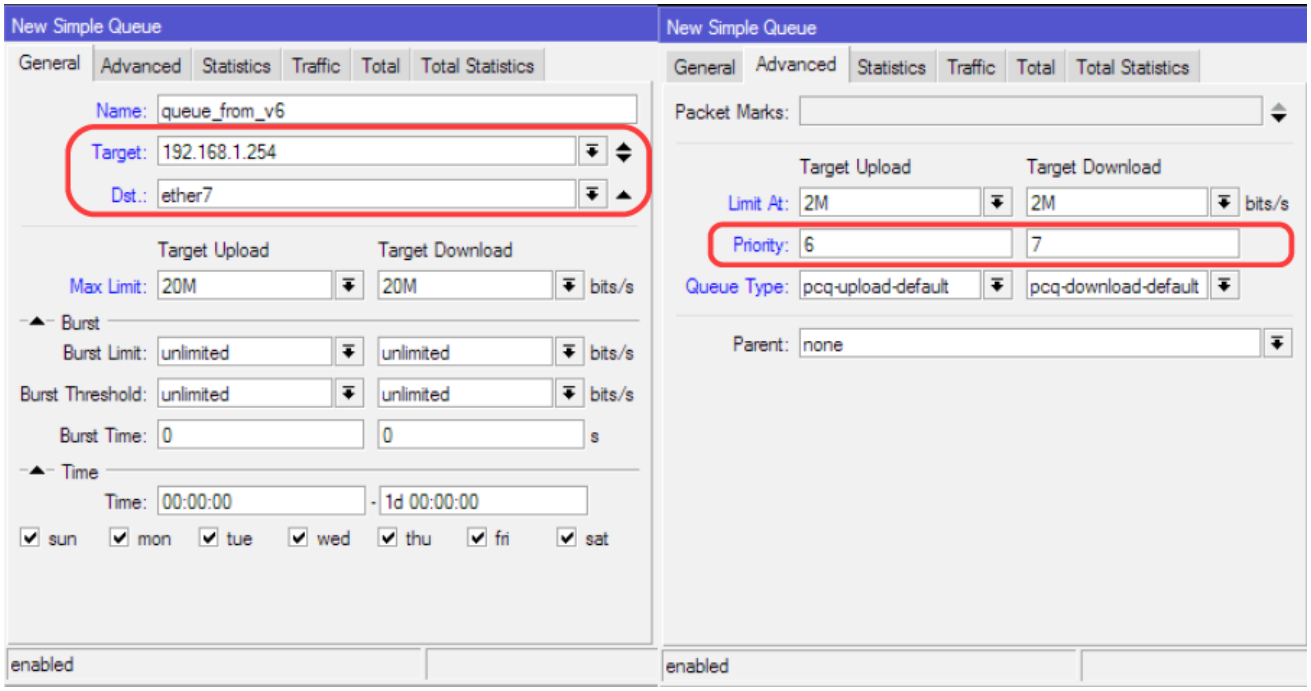
- Simple queues 现在完全从 queue tree 中分离，因此我们可以称他为另一个 queue tree “global-2”。
- Simple queues 你可以同样建立 queue tree 结构，父级和子级实现 HTB，但仍然采用 FIFO 顺序执行的优先级处理。
- 虽然 Queue tree 和 simple queue 被分离，但 IP 数据流首先经过 Queue tree 被处理，接下来是 simple queue，你可以只选择一个模块进行流控设置。

V6.0 配置变动

下面对比 6.0 之前和 6.0 之后的 simple queue 配置接口，下面是 v5.0 的接口与调整内容：



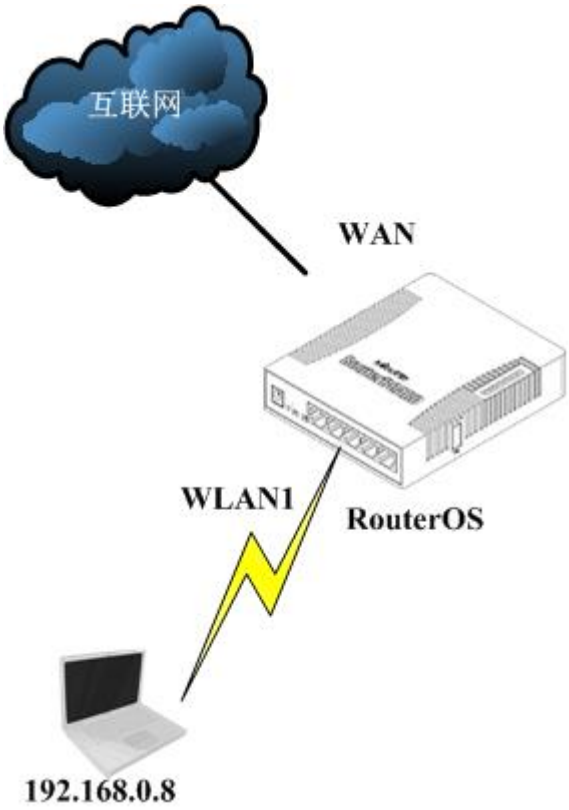
RouterOS 6.0 版本的 simple queue 接口，General 接口 6.0 把 dst-address 从 advanced 栏放到了 general 下(dst)，原有 interface 属性合并到 target 里



6.0 版本的 P2P 选项以及被删除掉，这个功能的确是个鸡肋，interface 项也没有了，因为 interface 选项已经被集成到了 general 栏的 target 中。

V6.0 Queue 事例

我们举例一个主机 IP 的流控，我们同时在 simple queue 和 queue tree 添加相同 IP 的主机流控规则。我们以 192.168.0.8 主机 IP 为例，如下网络结构：



注：以下环境是采用下载数据为测试标准，该环境以结果为主，规则配置简化请谅解。

环境 1：Queue tree 设置 2M，simple queue 设置 6M

我们首先配置 queue tree 中的流控规则，限制该主机带宽为 2M，我们配置 queue tree 需要通过 mangle 标记 IP 数据流。所以我们进入 ip firewall mangle 标记 foreword,首先将标记 src-address=192.168.0.8 的连接，并取名为 new-connection-mark=ip1_connetion，然后从 ip1_connetion 的连接标记中提取数据报，并取名为 packet1，如下配置脚本

```
/ip firewall mangle
add action=mark-connection chain=forward new-connection-mark=ip1_connection
src-address=192.168.0.8
add action=mark-packet chain=forward connection-mark=ip1_connection new-packet-mark=packet1
```

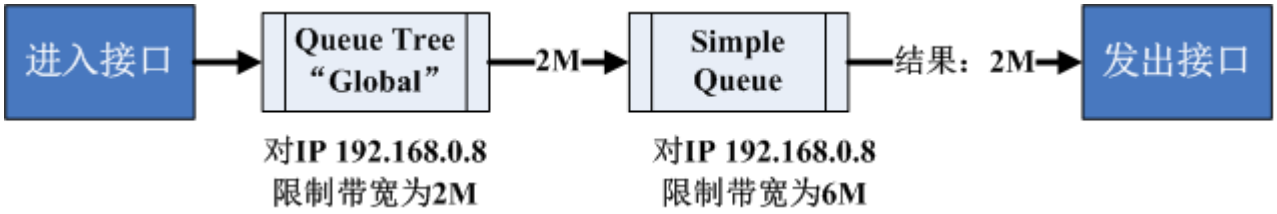
然后进入 queue tree 中添加一条流控规则，parent=wlan1，下载数据流向 wlan1 网卡，并设置为 2M

```
/queue tree
add max-limit=2M name=ip8 packet-mark=packet1 parent=wlan1 queue=default
```

设置完成 queue tree 的流控规则，下面我们设置 simple queue 流控规则，simple queue 规则配置相对简单，无需标记数据流，直接设置为：

```
/queue simple
add disabled=yes max-limit=6M/6M name=ip8 target=192.168.0.8/32
```

最后我们得到的结果是：



虽然我们将 Simple Queue 设置了 6M 带宽，但从上级的 Queue Tree 流程专递下来的带宽只有 2M，所以即使 Simple Queue 也只能获得 2M 带宽的数据流，最终结果 192.168.0.8 也只能获得 2M 下载带宽。

环境 2：Queue tree 设置 6M，simple queue 设置 2M

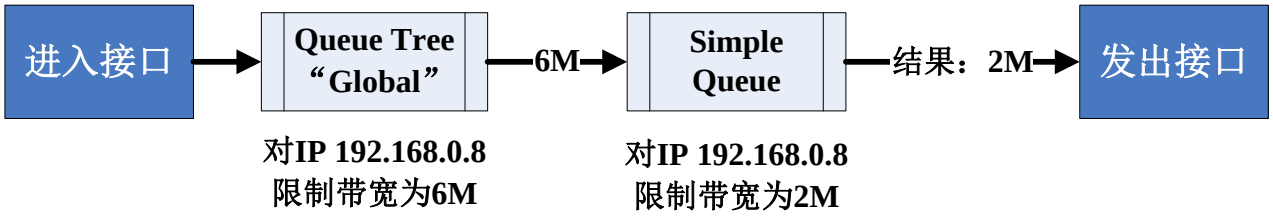
我们将两个规则的带宽调整下，即 Queue Tree 设置 6M

```
/queue tree
add max-limit=6M name=ip8 packet-mark=packet1 parent=wlan1 queue=default
```

将 queue simple 规则设置为 2M

```
/queue simple
add disabled=yes max-limit=2M/2M name=ip8 target=192.168.0.8/32
```

最后我们得到的结果是：



结果仍然是 2M，因为 Queue Tree 虽然设置了 6M 带宽，但到了后面的 Simple Queue 被限制为 2M，所以结果是 Queue Tree 比 Simple Queue 先获取流量，但两个同时使用时对同一属性参数都会起作用。

其实对于一套系统来说出现两个流控模块完全有点多余，可能会有人说 Queue Tree 可以实现 HTB 功能，其实 HTB 在现在的 Simple Queue 同样可以实现。

但现在的这个结构也不能说没有用，例如我们可以用 Queue Tree 做 IP 的流控，再用 Simple Queue 做基于一些协议和端口的流控，这样当 IP 被限制一定带宽后，每个 IP 中的协议和端口可以在后面的 Simple Queue 再一次被处理，递属于这个 IP 带宽的端口只能被限制在 Queue Tree 控制带宽范围内做一次 simple queue 的流控。

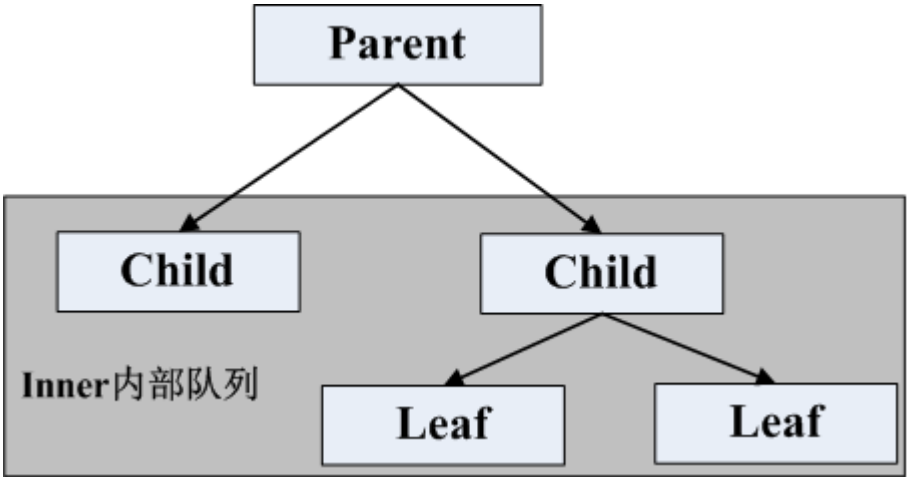
12.7 HTB 等级令牌桶介绍

HTB (Hierarchical Token Bucket) 是一种流量分类的的队列算法，可有效提高带宽利用率和限制各种网络流量，并能根据 IP 地址、协议和端口等参数对数据流进行优先级设置，然后针对不同类别的数据流进行带宽控制。

指定主机或服务预留带宽、限制最高带宽，也能实现平均分配带宽，并进行优先级管理，特别适合语音视频和数据混合的网络。配置 HTB 的流程如下：

HTB 等级令牌桶允许创建一个等级队列结构，并确定队列之间的关系，就像“父亲与儿子”或“兄弟之间”。

一旦队列添加了一个 Child（子队列）将会变为 inner(内部队列)，所有向下没有 Children（子队列）称为 Leaf 队列（叶队列）,内部队列仅负责传输的分配,所有 Leaf 队列对符合的数据进行处理。 在 RouterOS 必须指定 Parent(父级)选项并指定一个队列为子队列。



双重限制

每条队列规则，都有 2 个速率限制参数：

- **CIR** (约定信息速率 Committed Information Rate) – (在 RouterOS 中的参数为 **limit-at**) 最坏的情况下，无论如何都会将得到给定的的 CIR 传输量(假设我们能发送那么多的数据量)。
- **MIR** (最大信息速率 Maximal Information Rate) – (在 RouterOS 中的参数为 **max-limit**) 最好的情况下，如果父级有剩余带宽，才能获得这部分剩下的带宽。

换句话说，首先 Limit-at（CIR）都会被满足，仅当子队列尝试借调父级剩余带宽时，才可以达到最大的带宽 **max-limit** (**MIR**).

在 HTB 中，无论如何 CIR 带宽都将会得到满足 (即使父级的 max-limit 满载) ，MIR 和 CIR 双重限制功能，我们建议坚持这些规则：

- **CIR** 约定速率之和，即所有子级速率必须小于或等于可获得父级传输量。

CIR(parent)* ≥ CIR(child1) +...+ CIR(childN)

如果父级下有子队列和叶队列，子队列可以设置为 CIR(parent)=MIR(parent)

- 任何子队列的 MIR 最大速率必须小于或者等于父级的 MIR 最大速率

MIR (parent) ≥ MIR(child1) & MIR (parent) ≥ MIR(child2) & ... & MIR (parent) ≥ MIR(childN)

在 winbox 中队列的颜色变化：

- 0% - 50% 使用情况 - 绿色
- 51% - 75% 使用情况 - 黄色
- 76% - 100% 使用情况 - 红色

优先级

这里我们知道，所有队列的 **limit-at (CIR)**都有可能被耗尽，优先级则主要负责分配父级队列剩余的带宽给 **Child**（子队列）达到 max-limit。队列高的优先级最优先达到 max-limit，优先级低的则不会。8 是最低优先级 ，1 则最高。

优先级工作环境:

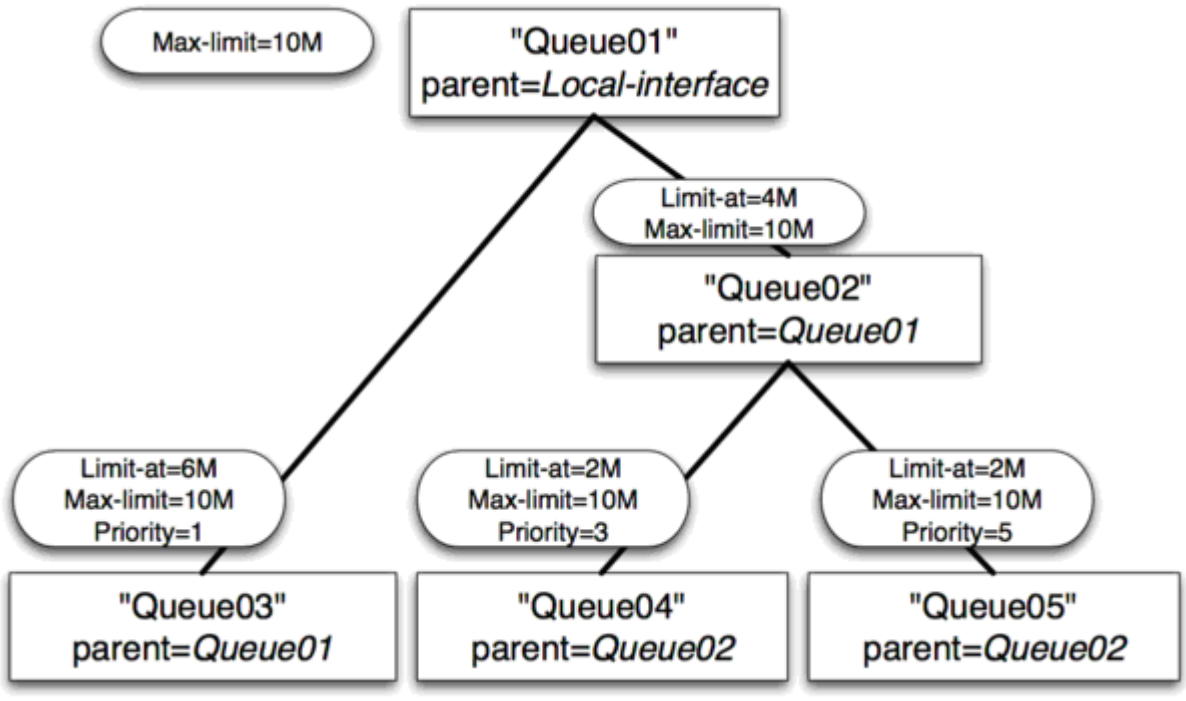
- 对于 **leaf** 叶队列的优先级对于自己 **inner**（内部队列）没有任何意义，即 **inner** 内部队列与其所属的 **leaf**（叶队列）的优先级是不可比较。
- 如果 **max-limit** 被设定（非 0）

下面这部分我们将分析 **HTB** 的操作，将演示一个 HTB 结构并将涵盖可能出现的所有情况和功能，我们的 HTB 结构由下面 5 个队列构成：

- **Queue01** 内部队列有 2 个子级 - **Queue02** 和 **Queue03**
- **Queue02** 内部队列有 2 个子级 - **Queue04** 和 **Queue05**
- **Queue03** 叶队列
- **Queue04** 叶队列
- **Queue05** 叶队列

Queue03，Queue04 和 **Queue05** 的需要 **10Mbps，我们接口处理能力在 10Mbps** 的流量

事例 1：普通事例

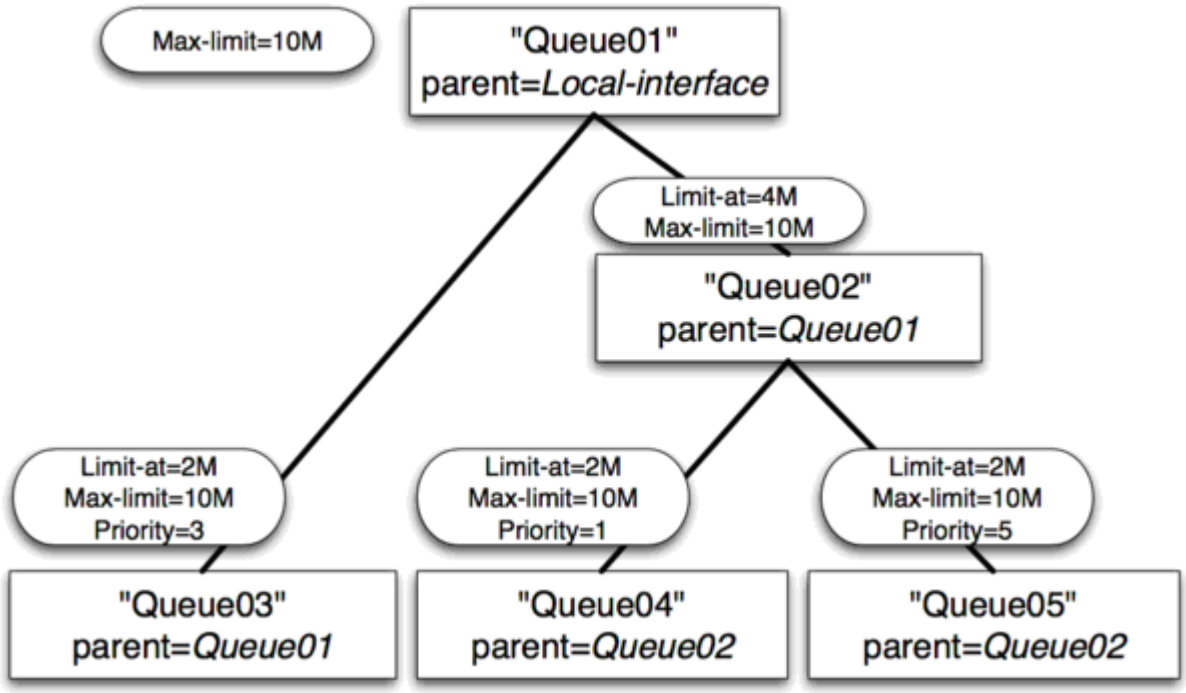


- **Queue01 limit-at=0Mbps max-limit=10Mbps**
- **Queue02 limit-at=4Mbps max-limit=10Mbps**
- **Queue03 limit-at=6Mbps max-limit=10Mbps priority=1**
- **Queue04 limit-at=2Mbps max-limit=10Mbps priority=3**
- **Queue05 limit-at=2Mbps max-limit=10Mbps priority=5**

事例 1 结果：

- **Queue03** 得到 **6Mbp**s
- **Queue04** 得到 **2Mbps**
- **Queue05** 得到 **2Mbps**
- **结论：HTB** 建立在一种方式上，通过满足所有的 **limit-at**，主队列已没有带宽进行分发。

事例 2：max-limit 事例

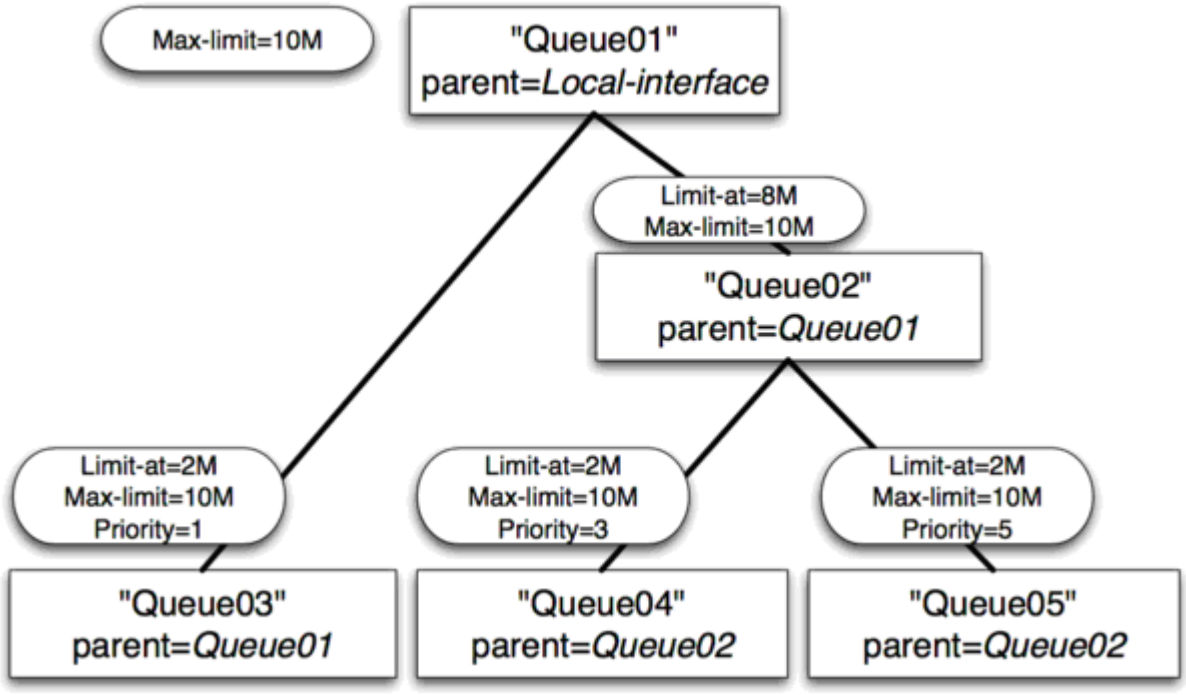


- **Queue01 limit-at=0Mbps max-limit=10Mbps**
- **Queue02 limit-at=4Mbps max-limit=10Mbps**
- **Queue03 limit-at=2Mbps max-limit=10Mbps priority=3**
- **Queue04 limit-at=2Mbps max-limit=10Mbps priority=1**
- **Queue05 limit-at=2Mbps max-limit=10Mbps priority=5**

事例 2 结果

- **Queue03** 得到 **2Mbps**
- **Queue04** 得到 **6Mbps**
- **Queue05** 得到 **2Mbps**
- **结论：** 在满足所有的 **limit-at** 后，**HTB** 将把剩余的带宽分配给优先级高的队列。

事例 3: inner 队列 limit-at



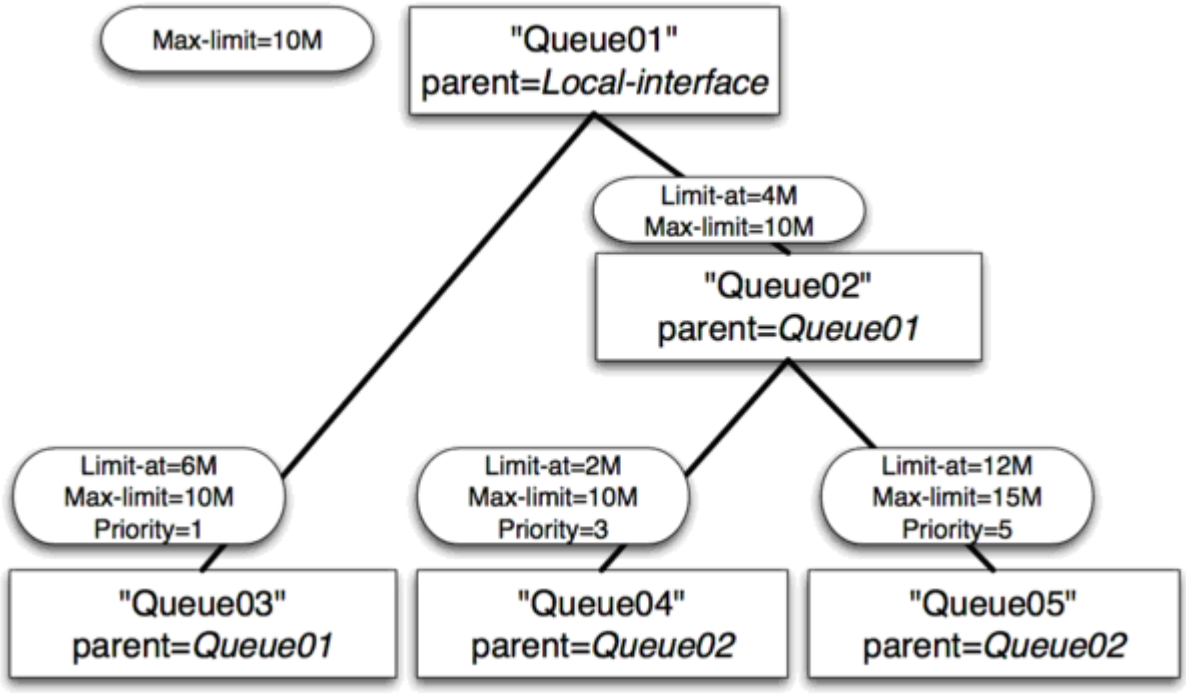
- **Queue01 limit-at=0Mbps max-limit=10Mbps**
- **Queue02 limit-at=8Mbps max-limit=10Mbps**
- **Queue03 limit-at=2Mbps max-limit=10Mbps priority=1**
- **Queue04 limit-at=2Mbps max-limit=10Mbps priority=3**
- **Queue05 limit-at=2Mbps max-limit=10Mbps priority=5**

事例 3 结果

- **Queue03** 得到 **2Mbps**
- **Queue04** 得到 **6Mbps**
- **Queue05** 得到 **2Mbps**

- **结论：**在满足所有的 **limit-at** 后，**HTB** 将分配剩余带宽给优先级高的，但在这个事例中，内部对列 **Queue02** 指定了 **Limit-at**，这样他会保留 **8Mbps** 的流量给 **Queue04** 和 **Queue05**，**Queue04** 有更高的优先级，那就是为什么会得到更高的带宽。

事例 4：leaf 队列的 Limit-at



- **Queue01 limit-at=0Mbps max-limit=10Mbps**
- **Queue02 limit-at=4Mbps max-limit=10Mbps**
- **Queue03 limit-at=6Mbps max-limit=10Mbps priority=1**
- **Queue04 limit-at=2Mbps max-limit=10Mbps priority=3**
- **Queue05 limit-at=12Mbps max-limit=15Mbps priority=5**

事例 4 结果

- **Queue03** 得到 **3Mbps**
- **Queue04** 得到 **1Mbps**
- **Queue05** 得到 **6Mbps**

- 结论：为了满足所有的 Limit-at，HTB 被强迫分配 20Mbps，Queue03 为 6Mbps ，Queue04 为 2Mbps，Queue05 为 12Mbps，但我们的接口只能处理 10Mbps，因此接口队列通常 FIFO 带宽发分配将保持比例 6:2:12，即 3:1:6。

RouterOS 中的 HTB

在 RouterOS 中有 4 个 HTB 树：

- global-in
- global-total
- global-out
- interface queue

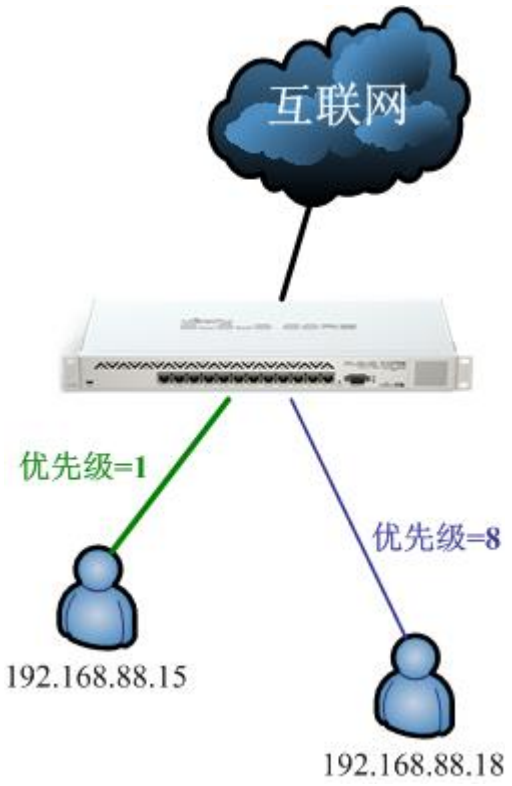
当添加一个简单队列时，将产生 3 个 HTB 类(in global-in， global-total and global-out)，但在接口队列中不添加任何类。

当数据报通过路由器时，它将穿过所有 4 个 HTB 树——global-in， global-total， global-out 和 interface queue。如果是指向路由器的它将穿过 global-in 及 global-total HTB 树，如果数据报是从路由器发出的，它们将穿过 global-total，global-out 及 interface 队列。

12.8 v6.0 Simple Queue 等级优先

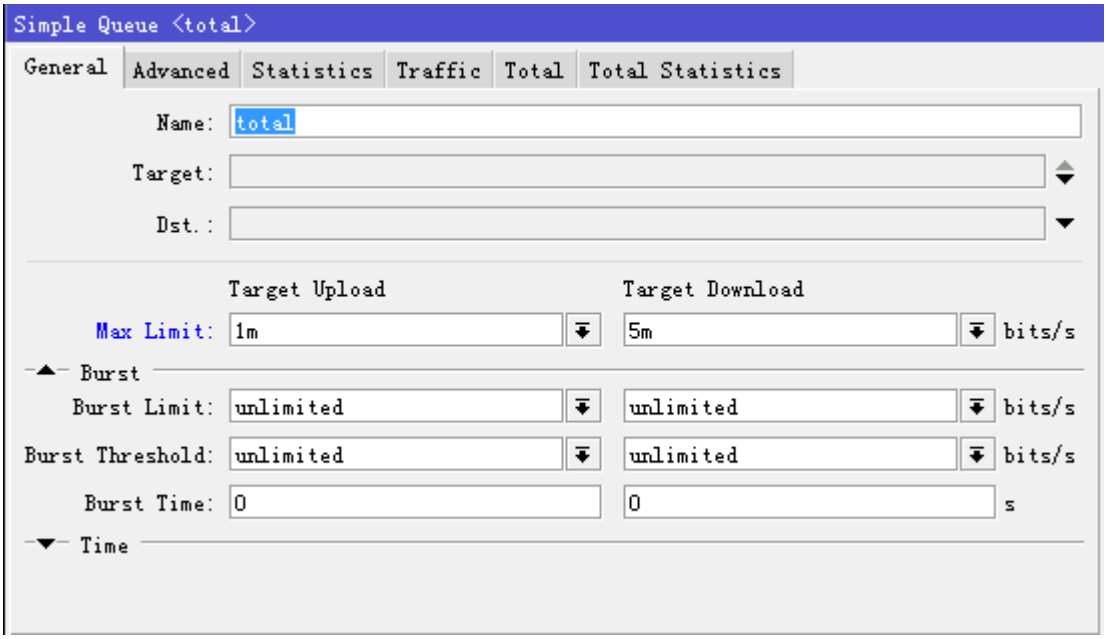
下面是一个 queue simple 下的等级优先流控规则实例，出口带宽是 5M，我们需要为内网主机设置优先级流控，如果在过去我们需要使用到 queue tree 的 HTB 功能，即需要通过 mangle 标记 ip 数据报进行操作，而现在我们可以不通过 mangle 来完成，直接在 simple queue 中操作。

例如在网络内有两台主机 192.168.88.15 和 192.168.88.18，192.168.88.15 优先级最高能优先获取带宽，而 192.168.88.18 则有低优先级，在 192.168.88.15 需要带宽时自动释放出带宽。



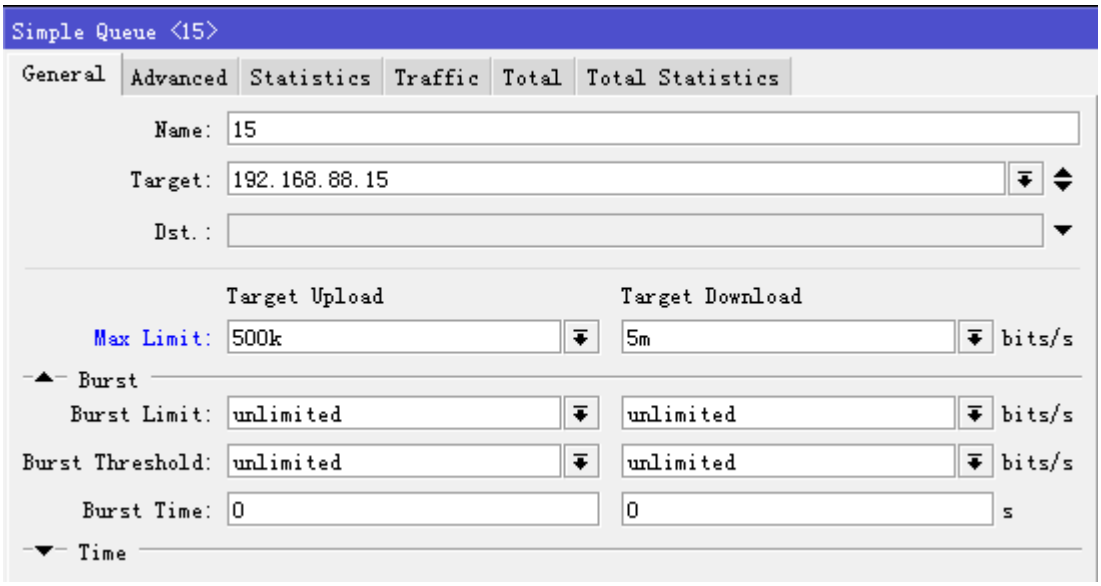
注意：simple queue 的等级优先规则同样需要按照 HTB 的原则进行配置

首先我们进入 simple queue 中，添加一条父级总带宽规则 total



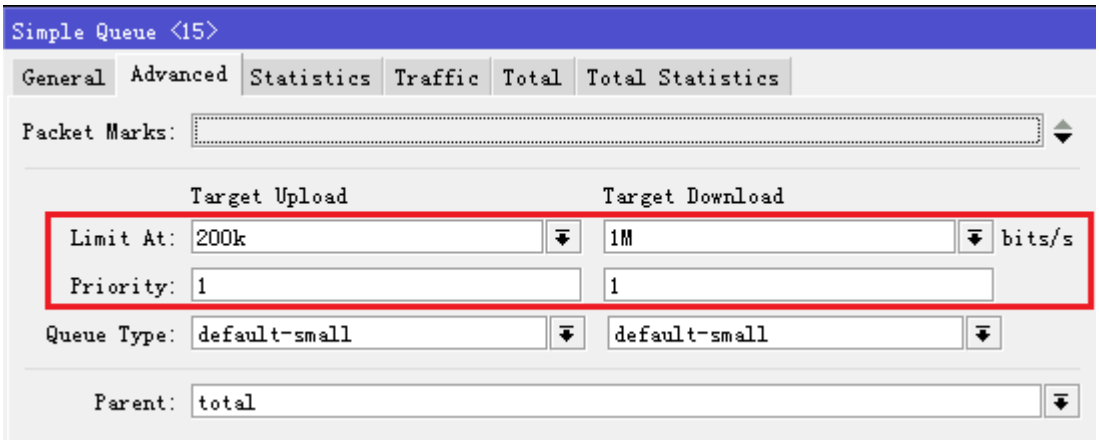
Max-limit 设置整个网络上行带宽为 1m，下行带宽为 5m

配置第二条规则取名 15，定义主机 192.168.88.15 的带宽规则：

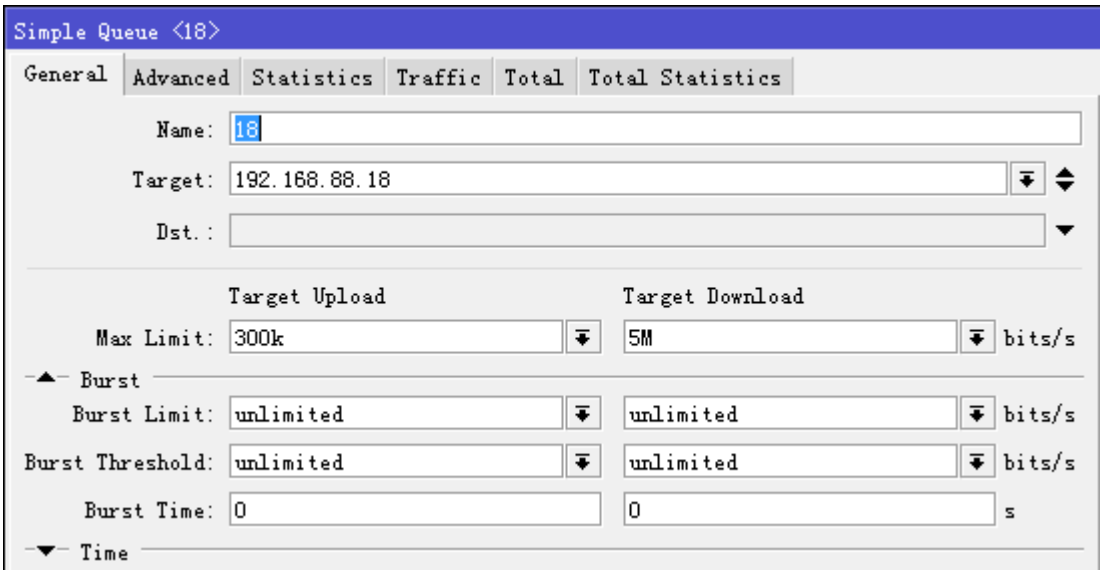


Max-limit 设置目标主机 192.168.88.15 上行带宽为 500k，下行带宽为 5m

设置 advanced 栏下配置，这里注意 limit-at 设置了上行 200k，下行 1m，即表示 192.168.88.15 主机最低能保证 200k 上行和 1m 下行带宽，是谁也不能拿走的，当然优先级 priority 最高是 1，parent 父级是 total

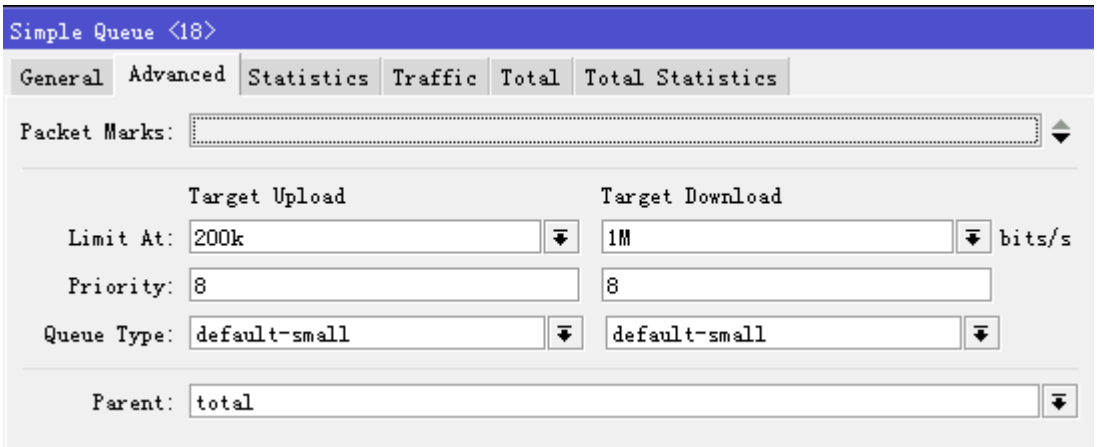


配置第三条规则取名 18，定义主机 192.168.88.18 的带宽规则：

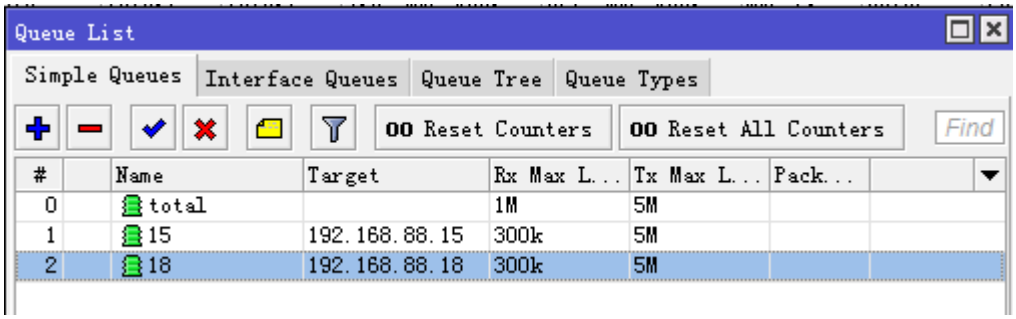


Max-limit 设置目标主机 192.168.88.18 上行带宽为 500k，下行带宽为 5m，与之前没有区别

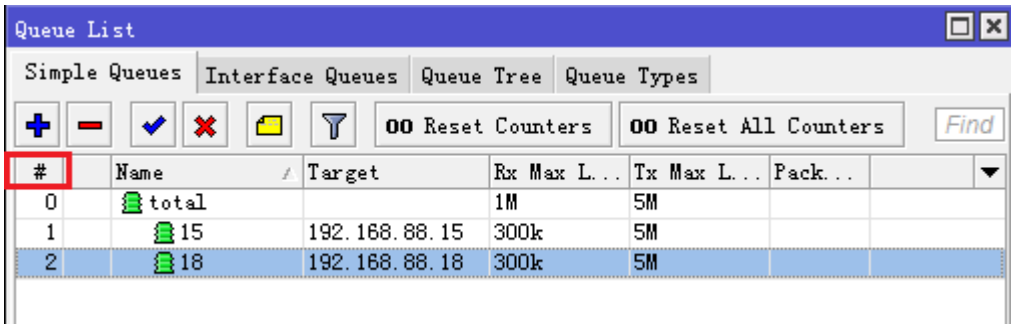
Advanced 栏中 limit-at 也相同，只是 priority 优先级设置为 8 最低，parent 父级是 total



两台主机的流控优先级配置完成后，看到 simple queue 菜单下队列情况：



我们可以点击“#”来重新排列队列等级：



这样基于 simple queue 中的 ip 的流控等级优先配置完成，这样你可以测试两台主机的优先级带宽控制。

12.9 Queue tree 队列树

操作路径: ***/queue tree***

Queue tree 队列树规则只能创建一个方向的流控队列，即一条队列规则只能控制上行或者下行，当一个 HTB 建立，一条 queue tree 规则只能限制一个方向的流量。因此在这样的条件下允许在一个队列规则里设置一个独立的接口，这种方式方便 mangle 配置，你不需要在一个 mangle 标记规则中去区分上行或下行，例如使用 out-interface 可设置 wan 口获取上行流量，设置 LAN 口获取下行流量。

注意：Queue tree 不是有序的队列，不同于 queue simple，所有流量将会被 HTB 等级令牌桶同时处理 ，queue tree 必须使用**/ip firewall mangle** 下标记数据报流进行匹配，建立流控队列。

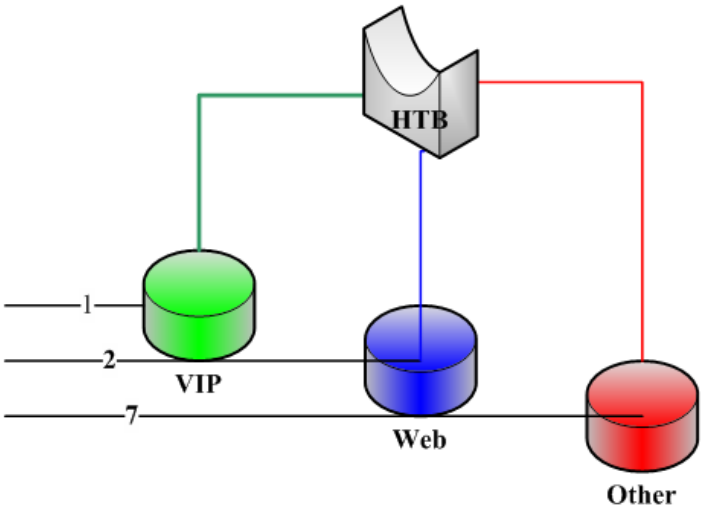
属性描述

burst-limit (*整数*) - 当突发值启动时可以达到的最大数据率
burst-threshold (*整数*) - 用于计算是否允许突发流量。如果上一次突发时间的平均数据率低于 *burst-threshold* 则实际数据率可能达到 *burst-limit*。
burst-time (*整数*) - 用于计算平均数据率。
flow (*文本*) - 在**/ip firewall mangle** 下标记的数据报流。 当前队列参数仅应用于用这个数据流标记标识了的数据报。
limit-at (*整数*) - 这个队列的约定流量
max-limit (*整数*) - 在有足够带宽可用的情况下可达到的流量
name (*文本*) - 队列的描述性名称
parent (*文本*) - 父队列的名称。顶级的父队列是可用的接口（实际上是主 HTB）。低级点的父队列可能是其他的队列。
priority (*整数*: 1..8) - 队列的优先级。 1 是最高级等级， 8 为最低。

queue (*文本*) - 队列类型名称。类型是在**/queue type** 下定义的。这个参数仅应用于树等级制中的子队列。

简单的 Queue Tree 实例

这个事例中，设定 3 类数据 VIP、Web 和 Other，这三类数据中 VIP 为网络内的重要用户优先级最高为 1，访问网页的数据 web 其次为 2，而剩下的数据 Other 级别最低为 7，假设我们的网络是 1M 的 ADSL，我们通过配置 HTB 策略来保证网络内的优先数据。



通过用 new-connection-mark 标记向外的连接，并采取 mark-connection 动作。当这个完成时你可以使用 new-packet-mark 标记属于这个连接的所有数据报并采用 mark-packet。

首先 VIP 数据标记，我们通过 ip firewall address-list 定义 VIP 用户的地址列表，定义完成后通过 src-address-list 调用：

```
[admin@Office] /ip firewall mangle> print
Flags: X - disabled, I - invalid, D - dynamic
 0   ;;; vip
     chain=forward action=mark-connection new-connection-mark=vip
     passthrough=yes src-address-list=vip

 1   chain=forward action=mark-packet new-packet-mark=vip passthrough=no
     connection-mark=vip
```

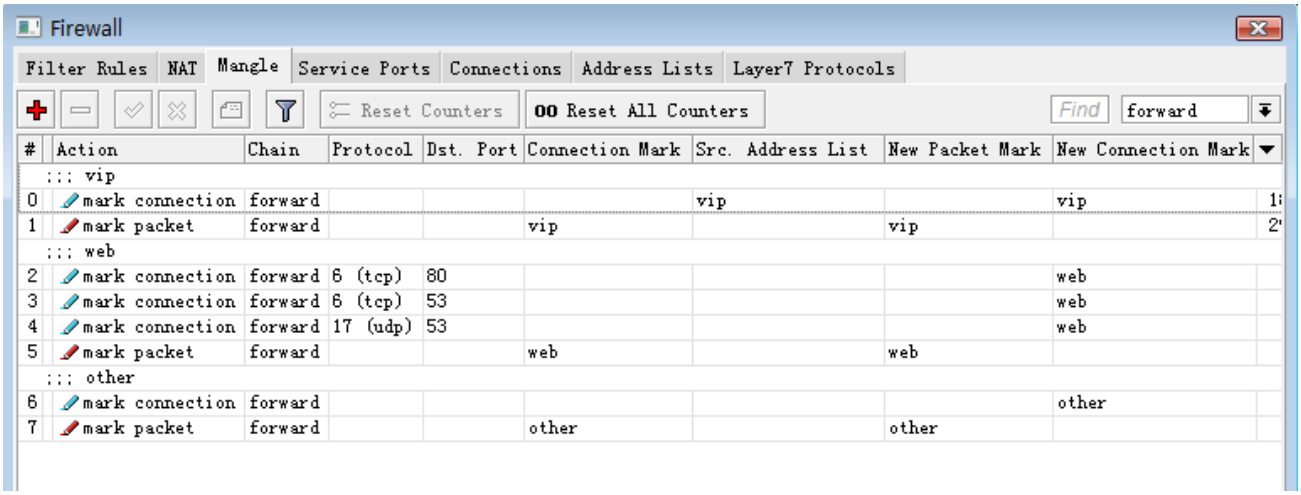
跟着定义 web 数据，这里我们需要针对访问网页的 tcp/80 端口和域名解析的 DNS 端口 tcp/53 和 udp/53 端口标记：

```
2   ;;; web
     chain=forward action=mark-connection new-connection-mark=web
     passthrough=yes protocol=tcp dst-port=80

 3   chain=forward action=mark-connection new-connection-mark=web
     passthrough=yes protocol=tcp dst-port=53

 4   chain=forward action=mark-connection new-connection-mark=web
```

```
    passthrough=yes protocol=udp dst-port=53

 5  chain=forward action=mark-packet new-packet-mark=web passthrough=no
    connection-mark=web
```

最后对剩下的 Other 数据进行标记，因为前面已经标记了 VIP 和 Web 的数据报，所有剩下数据就是其他的 Other 数据：

```
6   ;;; other
    chain=forward action=mark-connection new-connection-mark=other
    passthrough=yes

 7  chain=forward action=mark-packet new-packet-mark=other passthrough=no
    connection-mark=other
```



#	Action	Chain	Protocol	Dst. Port	Connection Mark	Src. Address List	New Packet Mark	New Connection Mark
	;;; vip							
0	mark connection	forward				vip		vip
1	mark packet	forward			vip		vip	
	;;; web							
2	mark connection	forward	6 (tcp)	80				web
3	mark connection	forward	6 (tcp)	53				web
4	mark connection	forward	17 (udp)	53				web
5	mark packet	forward			web		web	
	;;; other							
6	mark connection	forward						other
7	mark packet	forward			other		other	

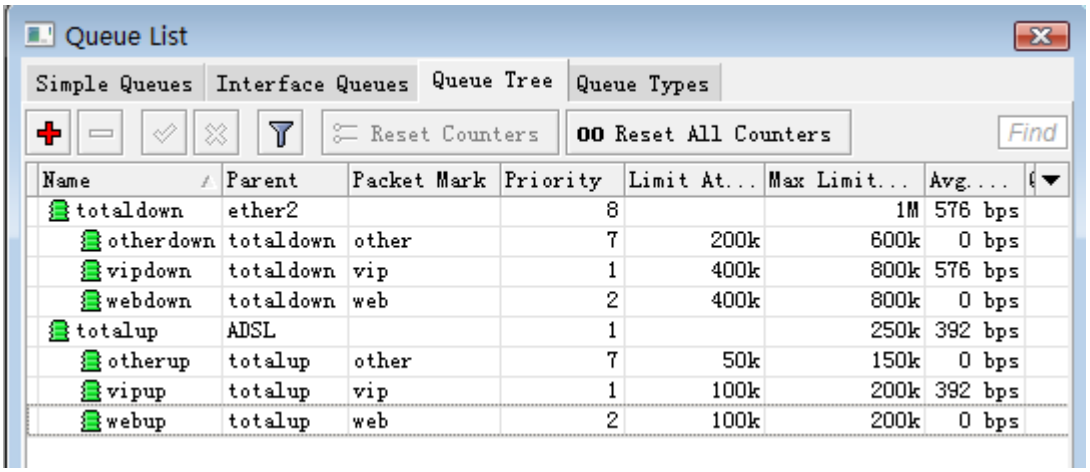
标记数据完成后，我们进入 queue tree 中，对数据进行优先级的配置，ADSL 总带宽为 1Mbps 下行，250kps 的上行，给三类数据带宽分配如下

- **VIP**：下行 Max-limit=800k limit-at=400k，上行 Max-limit=220k limit-at=200k，优先级 1
- **Web**：Max-limit=800k limit-at=400k，上行 Max-limit=200k limit-at=200k，优先级 2
- **Other**：Max-limit=600k limit-at=200k，上行 Max-limit=150k limit-at=50k，优先级 7

根据以上参数，我们在 queue tree 中配置队列优先级：

```
[admin@Office] /queue tree> print
Flags: X - disabled, I - invalid
 0   name="totalup" parent=ADSL packet-mark="" limit-at=0 priority=1
    max-limit=250000

 1   name="totaldown" parent=ether2 packet-mark="" limit-at=0 priority=8
    max-limit=1000000

 2   name="vipdown" parent=totaldown packet-mark=vip limit-at=400000
    priority=1 max-limit=800000
```

```
3   name="vipup" parent=totalup packet-mark=vip limit-at=100000
    priority=2 max-limit=200000

4   name="otherdown" parent=totaldown packet-mark=other limit-at=200000
    priority=8 max-limit=600000

5   name="otherup" parent=totalup packet-mark=other limit-at=50000
    priority=8 max-limit=150000

6   name="webup" parent=totalup packet-mark=web limit-at=100000
    priority=1 max-limit=150000

7   name="webdown" parent=totaldown packet-mark=web limit-at=400000
    priority=1 max-limit=800000
```



Name	Parent	Packet Mark	Priority	Limit At...	Max Limit...	Avg....
totaldown	ether2		8		1M	576 bps
otherdown	totaldown	other	7	200k	600k	0 bps
vipdown	totaldown	vip	1	400k	800k	576 bps
webdown	totaldown	web	2	400k	800k	0 bps
totalup	ADSL		1		250k	392 bps
otherup	totalup	other	7	50k	150k	0 bps
vipup	totalup	vip	1	100k	200k	392 bps
webup	totalup	web	2	100k	200k	0 bps

Queue tree v6 实例

RouterOS v6 对 Queue 的改动，导致 simple queue 从 global queue 分离，在 RouterOS 内部出现了两条独立的流控模块。由于 simple queue 的独立不再需要通过 global-out 和 global-in 双重流控，从而大大提升了处理效率和 RouterOS 在流控方面的性能。而对于 Queue Tree 来说，global-in 和 global-out 已经合并为 global 模块，因此我们 mangle 标记数据流时，就不用考虑在那个链表使用 global-in 还是 global-out 模块，统一都是 gloabl。具体的 RouterOS v6 改动请参考 12.5 章节。

下面引入一个参考实例，供大家学习：

- 有一个小型网络 192.168.0.0/24，
- 通过 Queue tree 建立 HTB 流控，
- 区分 HTTP 视频，网页和其他流量，流控顺序是网页优先，其次视频，最后是其他流量

按照上面的要求，实现以下操作步骤：

步骤一

进入 mangle 标记对应的数据流，需要区别标记 HTTP 视频，网页和剩下的其他流量，首先标记 HTTP 视频流量，这里会调用 L7 协议

识别大部分 HTTP 视频正则表达式如下：

```
[admin@MikroTik] /ip firewall layer7-protocol> print
 # NAME                                REGEXP
 0 video                               ^(get|post).+\.(flv|mp4|f4v)
```

配置完成 L7 协议后，继续在 mangle 完成对 HTTP 视频流量的标记，选择 forward 链表，先创建 mark-connecion 标记 HTTP 视频的链接，然后从 HTTP 视频链接标记中提取 HTTP 视频数据报 mark-packet 的下行数据，注意，我使用了 dst-address=192.168.0.0/24，表明数据方向是发给用户的下载，同时 mark-packet 规则的 passthrough=no，即数据报到此不再向下传递处理。

```
/ip firewall mangle
add action=mark-connection chain=forward comment=http_video layer7-protocol=video
log-prefix="" new-connection-mark=video_c passthrough=yes
add action=mark-packet chain=forward connection-mark=video_c
dst-address=192.168.0.0/24 log-prefix="" new-packet-mark=video_p passthrough=no
```

继续配置 HTTP 网页的识别，这里没有采用 L7 协议，直接通过标记 TCP 的 80,443 端口，同样先标记连接，在标记数据报，通过目标 IP 确定数据方向，passthrough=no。

```
/ip firewall mangle
add action=mark-connection chain=forward comment=h dst-port=80,443 log-prefix=""
new-connection-mark=http_c passthrough=yes protocol=tcp
add action=mark-packet chain=forward connection-mark=http_c
dst-address=192.168.0.0/24 log-prefix="" new-packet-mark=http_p passthrough=no
```

最后，标记剩下的流量，这里我没有再标记连接，而是直接标记数据报，通过 IP 地址源和目标区分上行和下行

```
/ip firewall mangle
add action=mark-packet chain=forward comment=l log-prefix="" new-packet-mark=UP
passthrough=yes src-address=192.168.0.0/24
add action=mark-packet chain=forward dst-address=192.168.0.0/24 log-prefix=""
new-packet-mark=DOWN passthrough=yes
```

通过 mangle 标记，将三类数据区分完成，接下来将建立 Queue Tree 的 HTB 流控

步骤二

进入 Queue tree，创建 HTB 流控规则，HTB 创建需要设置两个父级，总上行和总下行，假设我们总带宽为 10M，为三类分配带宽如下：

- HTTP 视频最大获得 9Mb（Max-limit），最低保障 2Mb（Limit-at），优先级 7
- HTTP 网页最大获得 9Mb（Max-limit），最低保障 5Mb（Limit-at），优先级 5
- 剩下其他流量最大获得 9Mb（Max-limit），最低保障 2Mb（Limit-at），优先 8

首先，我们创建总上行和总下行的 total_down 和 total_up 规则

```
/queue tree
add max-limit=10M name=total_down parent=global queue=default
add max-limit=8M name=total_up parent=global queue=default
```

然后创建 HTTP 视频下行规则，父级 parent=total_down，packet-mark 设置为 video_p

```
/queue tree
add limit-at=2M max-limit=9M name=video packet-mark=video_p parent=total_down
priority=7 queue=default
```

再创建 HTTP 网页下行规则，父级 parent=total_down，packet-mark 色或者为 http_p

```
/queue tree
add limit-at=5M max-limit=9M name=http packet-mark=http_p parent=total_down
priority=5 queue=default
```

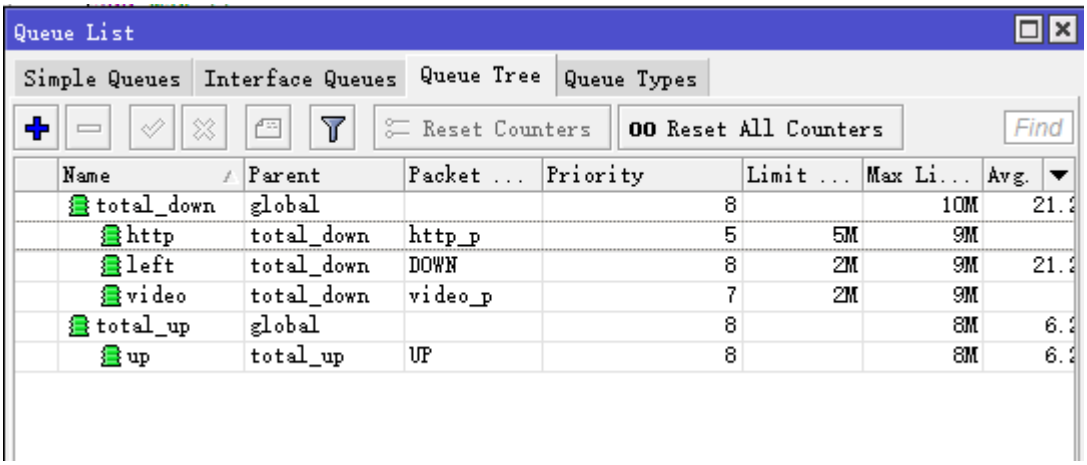
最后创建其他流量下行规则，父级 parent=total_down，packet-mark 设置为 DOWN

```
/queue tree
add limit-at=2M max-limit=9M name=left packet-mark=DOWN parent=total_down
queue=default
```

设置总的上行规则，带宽 8Mb，父级为 total_up

```
/queue tree
add max-limit=8M name=up packet-mark=UP parent=total_up queue=default
```

配置完成后在 winbox 显示如下：



实际测试中，当网络中存在 HTTP 视频流量时，如果没有其他流量使用，视频流量最大可以达到 9M，这时有人打开网页优先级高于 HTTP 视频流量，HTTP 视频流量会被压制，并把流量让给网页，且会保障 HTTP 视频流量最低有 2Mb。该实例供大家参考，如果要限制单个用户流量，请引入 PCQ 到 Queue-type。

Queue tree HTB 配置实例

这里演示 **Parent 定义**为网络接口，区分上行和下行流量，外网 WAN 口作为出方向，定义为上行，内网 LAN 接口则作为下行，实例仍然通过 HTB 在 Queue tree 的配置标记不同类型的数据流，实现优先级的带宽控制。

注意：Parent 基于网络接口配置，仅适用于单线接入和一个内网接口的网络！

首先查看接口上的 IP 地址：

```
[admin@MikroTik] /ip address> print
Flags: X, D - DYNAMIC
Columns: ADDRESS, NETWORK, INTERFACE
#   ADDRESS            NETWORK        INTERFACE
;;; defconf
0   192.168.88.1/24    192.168.88.0   bridge
1 D 100.64.100.119/32  100.64.100.1   pppoe-out1
```

在/ip address 下看到有两个网络接口，一个是内网的 bridge，一个是外网的 pppoe-out1 拨号接口。

基于协议的 HTB

Queue tree 里建立 HTB 等级流控，在 mangle 标记流量，首先提取 TCP 的 80 和 443 端口标记为 web 流量，然后将剩下的流量标记为 other。

先是连接标记，匹配源地址是 192.168.88.0/24 的内网 IP 地址段，并提取 tcp 协议的 80 和 443 端口连接标记为 web，然后 web 连接标记中提取数据包，标记为 web，passthrough 设置为 no，不再下行传递。

```
/ip firewall mangle
add action=mark-connection chain=forward dst-port=80,443 new-connection-mark=\
    web passthrough=yes protocol=tcp src-address=192.168.88.0/24
add action=mark-packet chain=forward connection-mark=web new-packet-mark=web \
    passthrough=no
```

最后是标记 192.168.88.0/24 剩下的连接，选择 connection-mark=no-mark 的连接，并从中提取数据包标记为 other

```
/ip firewall mangle
add action=mark-connection chain=forward connection-mark=no-mark \
    new-connection-mark=other passthrough=yes src-address=192.168.88.0/24
add action=mark-packet chain=forward connection-mark=other new-packet-mark=\
    other passthrough=no
```

在 queue tree 创建 HTB 配置，首先创建父级队列 down 和 up，分别对应上行的外网口 pppoe-out1 和下行的内网口 bridge，出口总带宽是 150Mbps，上行带宽 80Mbps，上行和下行都预留一部分带宽，用于缓冲。

```
/queue tree
add max-limit=140M name=down parent=bridge
add max-limit=60M name=up parent=pppoe-out1
```

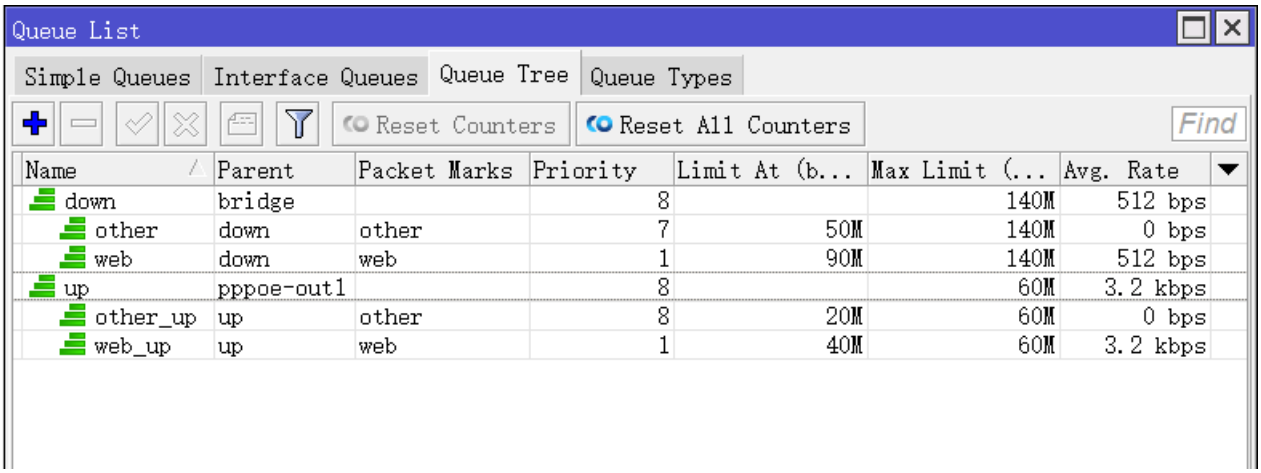
web 流量优先于其他流量 priority 设置 1，其他流量 priority 优先级设置 8，

```
/queue tree
add max-limit=140M limit-at=50M name=web packet-mark=web parent=down priority=1
add max-limit=140M limit-at=50M name=other packet-mark=other parent=down priority=8
add max-limit=60M limit-at=40M  name=web_up packet-mark=web parent=up priority=1
add max-limit=60M limit-at=20M name=other_up packet-mark=other parent=up priority=8
```

- down 作为下行父级总带宽 140M，子级包含 web 和 other，
- up 作为上行父级总带宽 60M，子级包含 web_up 和 other_up
- web：定义 max-limit 最大可以获取 140M，limit-at 为 90M（保证最少带宽 90M，其他子级无法抢占），优先级 1，即最高
- other：定义 max-limit 最大可以获取 140M，limit-at 为 50M（保证最少带宽 50M，其他子级无法抢占），优先级 8，最低

下行子级的 web_up 和 other_up 配置同理

配置完成后通过 winbox 查看



Name	Parent	Packet Marks	Priority	Limit At (b...	Max Limit (...	Avg. Rate
down	bridge		8		140M	512 bps
other	down	other	7	50M	140M	0 bps
web	down	web	1	90M	140M	512 bps
up	pppoe-out1		8		60M	3.2 kbps
other_up	up	other	8	20M	60M	0 bps
web_up	up	web	1	40M	60M	3.2 kbps

下载的运行结果：

1、 other 流量达到 140M 时，web 流量需求不断增加，other 优先级低于 web，会释放 90M 带宽给 web，由于 limit-at 的设置保证了 other 保留 50M 带宽，这时 web 流量是 90M，other 流量是 50M

2、 当 web 流量达到 140M 时，other 流量需求 20M，虽然 web 优先级高于 other，但 limit-at 要求最低保证 50M 带宽，因此 web 会释放 20M 给 other，满足 limit-at 的需求。

配置 HTB 时需要注意：所有子级的 Limt-at 总和必须小于等于父级的 Max-limit，子级的 max-limit 必须小于等于父级的 max-limit。当然实际情况中可以根据网络需求，限制低等级的 limit-at，从而能让高等级的队列获得更大流量。

基于 IP 的 HTB

网络情况如上，假如网络中有两台主机需要进行优先级的设置，如 192.168.88.8 和 192.168.88.9 两台主机进行流量控制，当 192.168.88.8 获取带宽的优先级高于 192.168.88.9

```
/ip firewall mangle
add action=mark-connection chain=forward new-connection-mark=host8 passthrough=\
    yes src-address=192.168.88.8
```

```
add action=mark-packet chain=forward connection-mark=host8 new-packet-mark=host8 \
    passthrough=no
add action=mark-connection chain=forward connection-mark=no-mark \
    new-connection-mark=host9 passthrough=yes src-address=192.168.88.9
add action=mark-packet chain=forward connection-mark=host9 new-packet-mark=\
    host9 passthrough=no
```

在 queue tree 配置主机有些的 HTB 流控

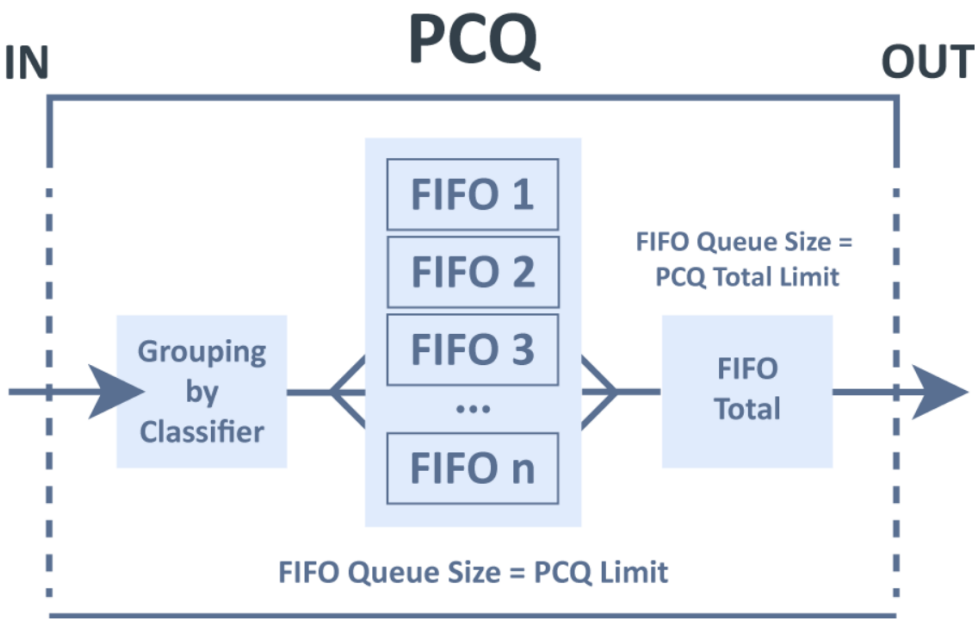
```
/queue tree
add max-limit=140M name=down parent=bridge
add max-limit=140M limit-at=100M name=host8 packet-mark=host8 parent=down priority=1
add max-limit=140M limit-at=10M name=host9 packet-mark=host9 parent=down priority=7
add max-limit=60M name=up parent=pppoe-out1
add max-limit=60M name=host8_up packet-mark=host8 parent=up priority=1
add max-limit=60M name=host9_up packet-mark=host9 parent=up priority=7
```

运行情况是 host8 优先级高于 host9，如果 host8 和 host9 同时进行大流量下载数据，host8 最大可以获得 130M 流量，因为 host9 的 limit-at 设置为 10M，需要保证 host9 的最低带宽。

12.10 PCQ 介绍

PCQ（Per Connection Queue）算法比较简单，首先利用 **pcq-classifier**（分类器）从相应数据流中区分一个 sub-stream（子数据流），然后在每一个 sub-stream 上建立独立的 FIFO 队列长度和限制，再归类所有的 sub-stream 在一起，通过全局的 Total FIFO 队列作总体的流控。以下是 PCQ 参数:

- **pcq-classifier** (dst-address | dst-port | src-address | src-port; 默认: "")：选择 sub-stream 分类类型。
- **pcq-rate (数字)：**每个 sub-stream 可获得的最大数据带宽。
- **pcq-limit (数字)：**在数据报中一个 sub-stream 的队列长度
- **pcq-total-limit (数字)：**全局 FIFO 队列的队列长度



如上图，假设有 100 个队列需要限制 1000kbps 下载时，可以使用 1 个 PCQ 队列和该 PCQ 队列包含的 100 子数队列进行处理。

PCQ 在 v5.0rc4 被重新编写，以优化在 Mbps 和 pps 的高吞吐量。有效的利用了新 Linux 内核特性，这使得 PCQ 更快，更少的资源需求。V5.0rc5 后又加入了一项新特性，一旦新的数据流被激活，它将获得 1/4 的最高优先级，但如果 rate 设置为"0"，sub-stream 子流将不具有获取 1/4 优先级特性(因为算法是“rate*1/4”，即"0"的 1/4 为"0")

假设 sub-stream 的 rate 是 10Mbps，那么当新的 sub-stream 请求流量时，将无限制地获得前 2.5Mbps 的流量。如果在使用类似 Speedtest.net 这样的带宽测试网站，得到的结果比预期更高。

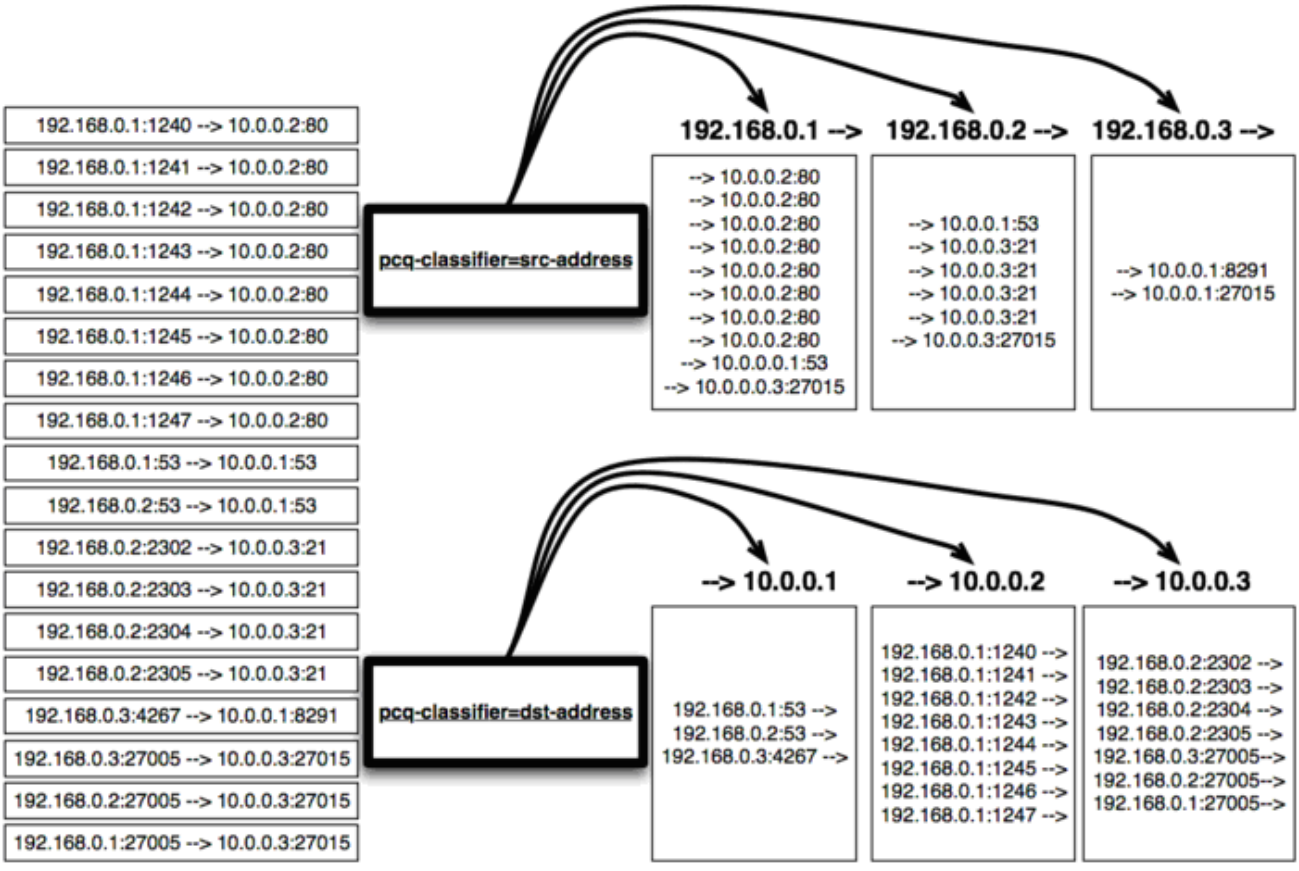
PCQ Sub-stream 的突发流量。PCQ 将有与简单队列和队列树相同的突发功能，PCQ 突发值参数:

- **pcq-burst-rate** (数值)：允许 sub-stream 突发流量时所能达到的最大上传/下载速率
- **pcq-burst-threshold** (数值)：该参数为突发值的开关
- **pcq-burst-time** (时间)：计算平均数据速率的时间周期，以秒为单位。 (并非实际的突发流量时间)

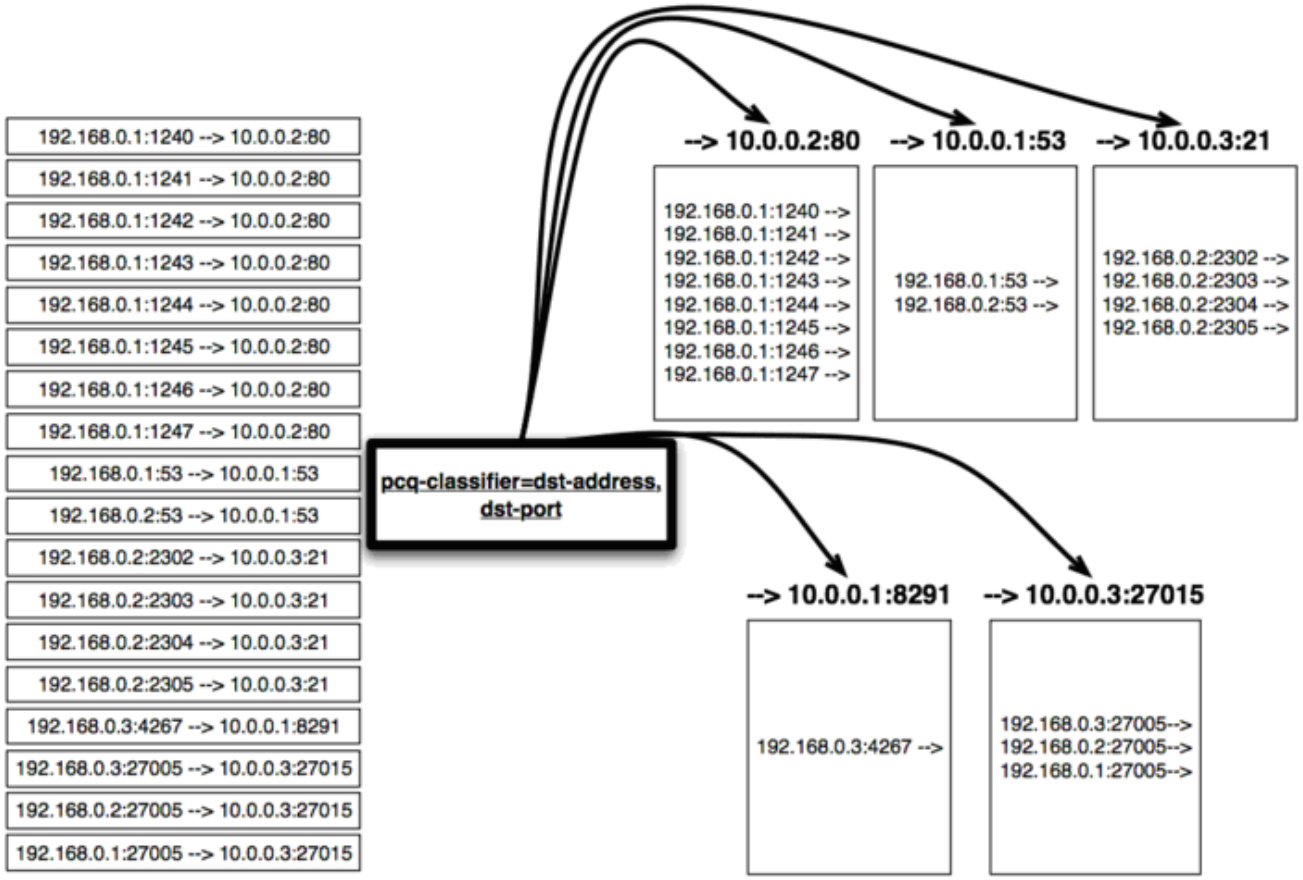
pcq-classifier 分类器

为更好的理解 PCQ 工作，需要了解分类器，用一组 IP 地址和端口到对应的地址和端口数据流的实例，这时将选择一种分类器，并通过 PCQ 将 18 个数据流从中分离到 PCQ 的子数据流中进行分类。

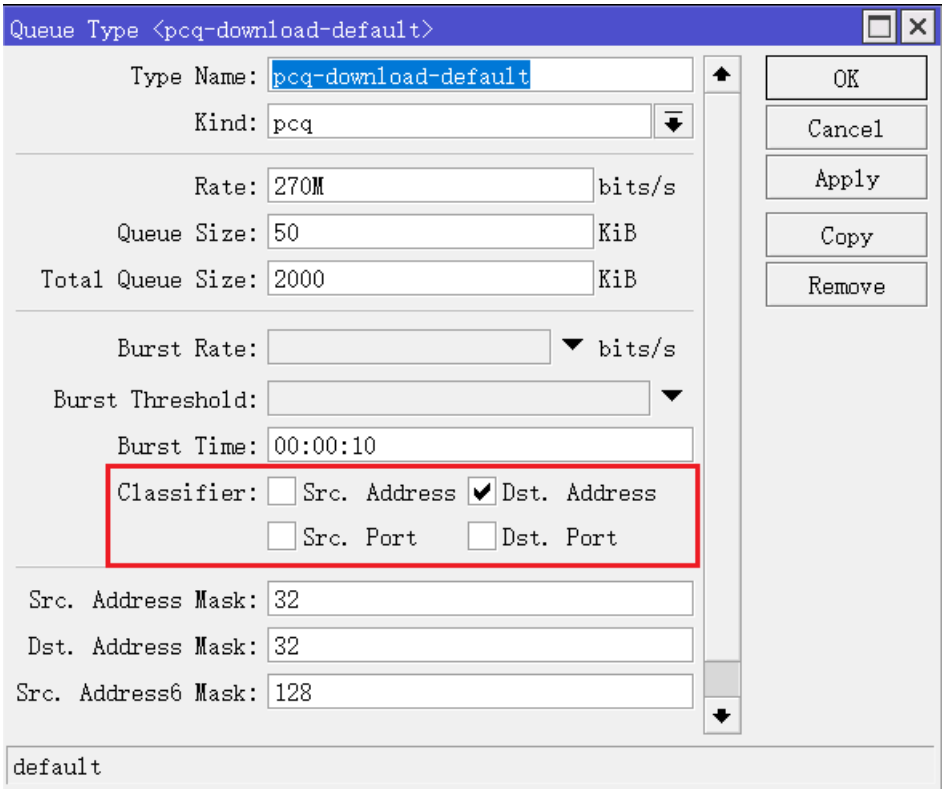
PCQ 的目标和源地址分类原理图：



端口分类

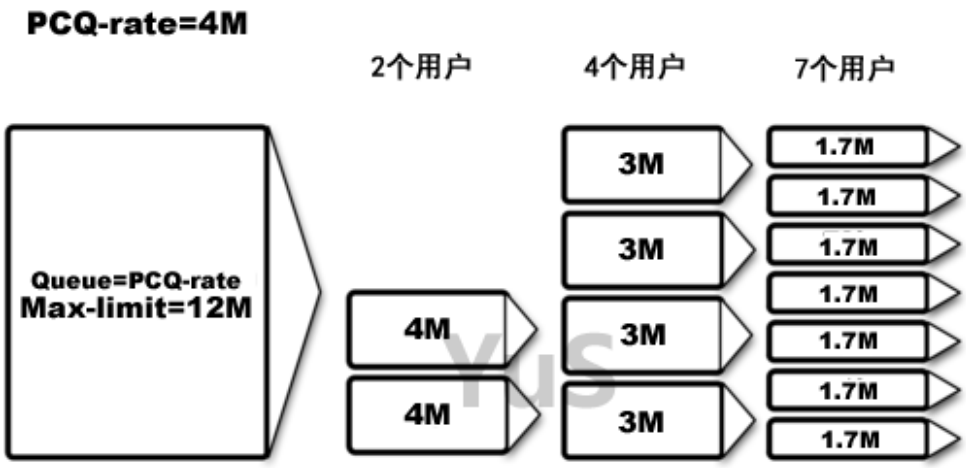


Classifier 分类器会根据参数设置来匹配连接会话中的源和目标 IP 或者是源和目标端口，如图的配置界面：

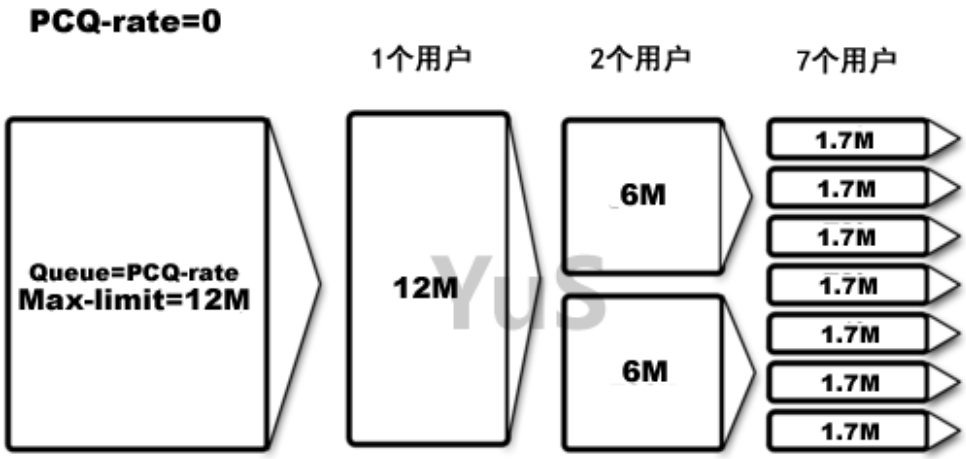


这样就把相同类型的数据归纳为一类，分配到一个 sub-stream 的 FIFO 队列中，然后所有的 sub-stream 子队列放到一个 Total 队列中进行总的流控。

在局域网中因为网络带宽的问题，需要对网络流做控制，但又因为做固定的流量控制时，会造成空闲带宽的浪费，这里我们可以同 RouterOS 的 PCQ 算法完成对内部局域网流量的动态分配，如下图所示：



如上图，最大带宽 Max-limit=12M，我们可以看到当 PCQ-rate 设定为 4M 的时候，小于等于 3 个用户在线时会分配固定 4M 带宽给每个用户，一旦大于 3 个用户时，PCQ 会做带宽平均分配的操作，会对在线用户进行平均的带宽分配。（Max-limit 就是 Total FIFO 全局队列带宽，PCQ-rate 就是 sub-stream 的 FIFO 子队列带宽）



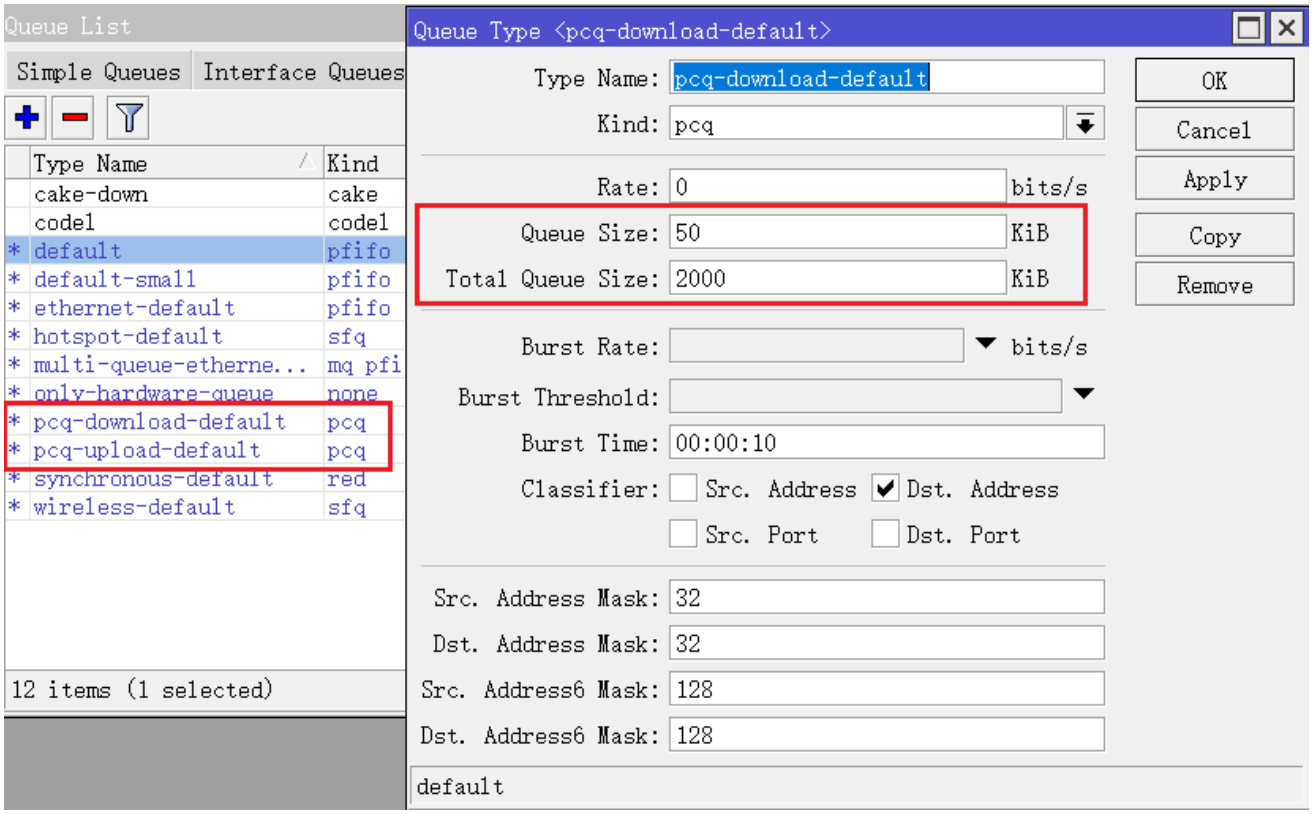
如上图，最大带宽 Max-limit=12M，PCQ-rate 设定为 0，这样在一个用户的时候就可以得到全部带宽，之后是 2 个用户平均分配，依次类推。

注意：Max-limit 总带宽是一定被设置的，这样 PCQ 能知道全局用户的带宽容量，并能为每个用户进行带宽分配。在实际使用中 Max-limit 最好略小于总带宽，例如总带宽是 100Mbps，Max-limit 设置为 90-95Mbps 预留一部分作为突发缓冲。

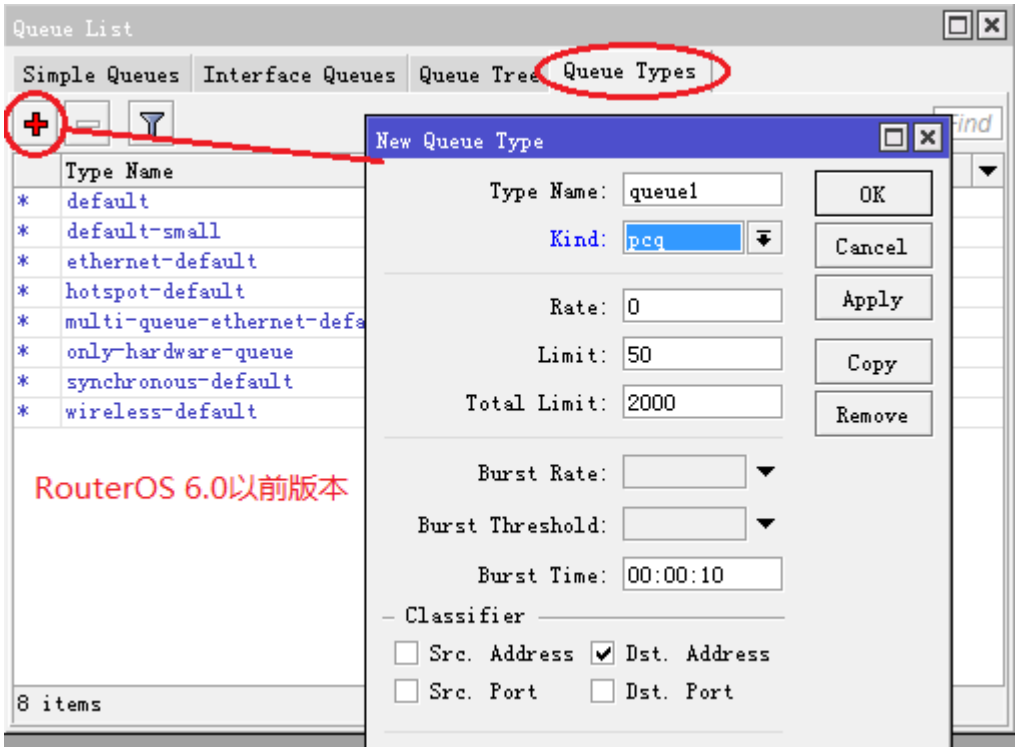
V5.0 后，PCQ 还允许使用不同大小的 IPv4 和 IPv6 子网或前缀作为 sub-stream 标识。之前它被锁定为一个/32 子网的主机 IP 地址。这主要是为 IPv6 在运营商的角度考虑，因为运营商分配给客户的 IPv6 地址可能是/64 前缀，也存在/128 的主机 IP 地址，参数如下：

- **pcq-dst-address-mask** (数值)：用于目标地址的 IPv4 子网长度
- **pcq-src-address-mask** (数值)：用于源地址地址的 IPv4 子网长度
- **pcq-dst-address6-mask** (数值)：用于目标地址的 IPv6 前缀长度
- **pcq-src-address6-mask** (数值)：用于源地址地址的 IPv6 前缀长度

RouterOS v6.0 版本后在 Queue Types 里增加了默认的 PCQ 类型 pcq-download-default 和 pcq-upload-default，



V6 之前版本需要自己手动添加：



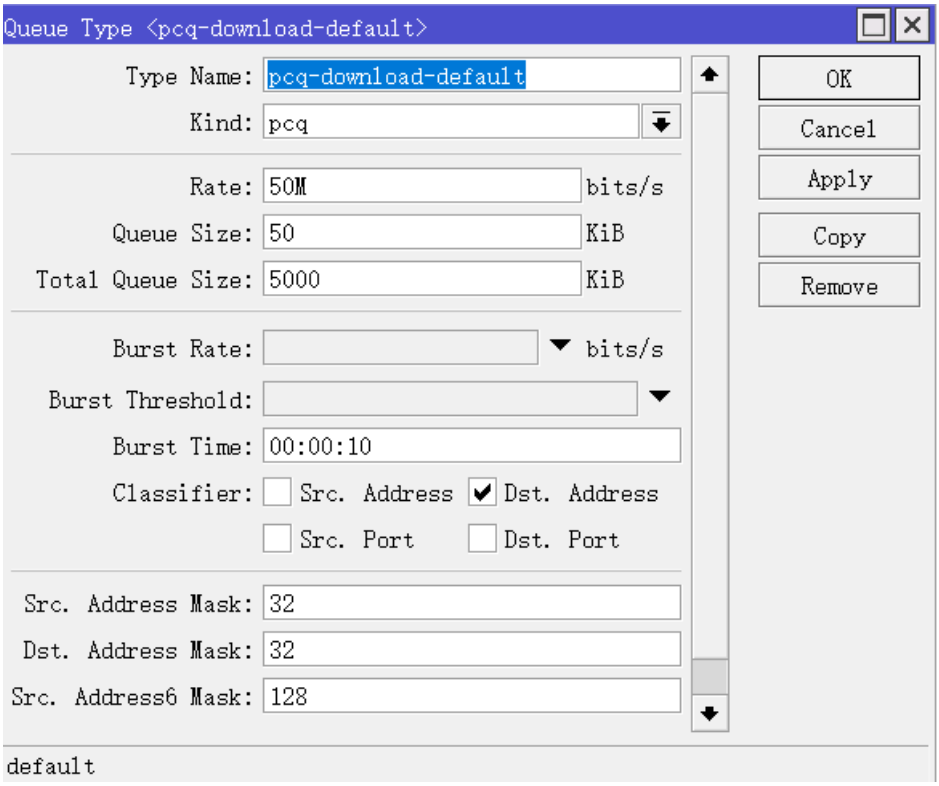
注： winbox 中的 Limit 和 Total-Limit 改为了 queue size 和 total queue size，调整为了 KB 单位，在命令行操作时属性名称为 pcq-limit 和 pcq-total-limit，它们具体关系：

- 默认情况下 total queue size（pcq-total-limit）是 2000，该规则仅能容纳 40 个用户，计算方法 total queue size（pcq-total-limit）/queue size（pcq-limit），即 2000/50=40
- 用户数量变化，可以通过调整 total queue size（pcq-total-limit）或者 queue size（pcq-limit）来满足需求
- 但必须保证每个用户队列 queue size（pcq-limit）有 10-20KiB，默认情况下建议 queue size（pcq-limit）值不变，调整 total queue size（pcq-total-limit）即可

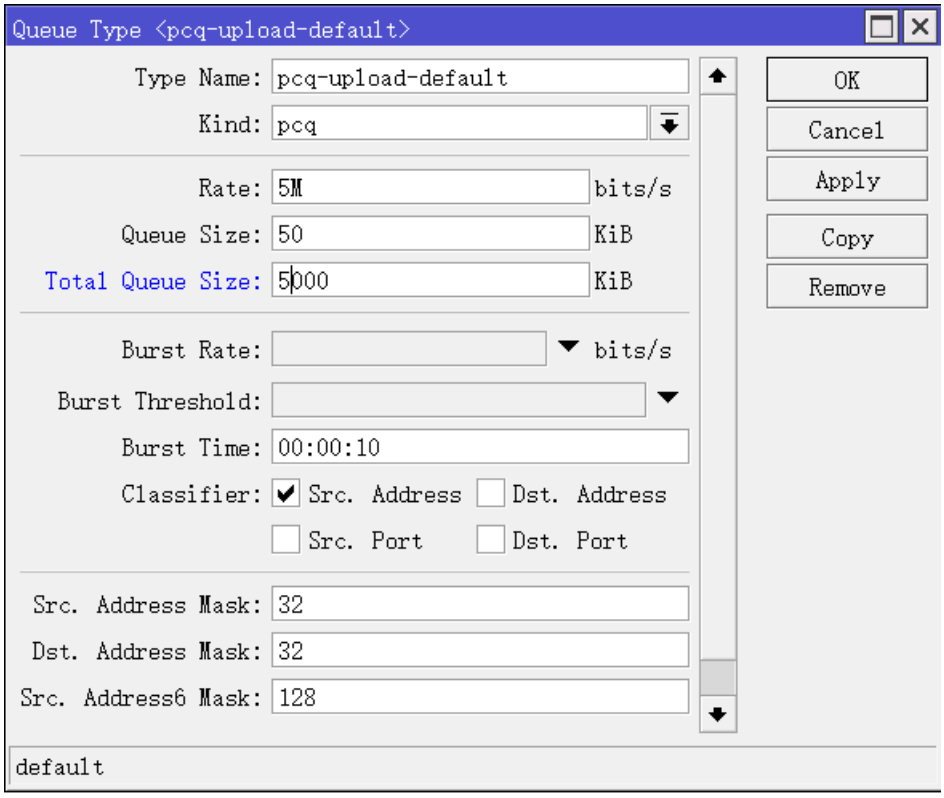
PCQ 动态流量分配实例

有一个网段为 192.168.10.0/24，需要通过 PCQ 流量控制，总带宽是 100M 下行/70M 上行，估计最高有 100 个用户在线，给每个用户分配上行和下行是 5mbps 和 50mbps，当在线每用户使用带宽超过可分配的 50M 时，会使用**"总带宽/在线用户数量"**，进行带宽平均分配。

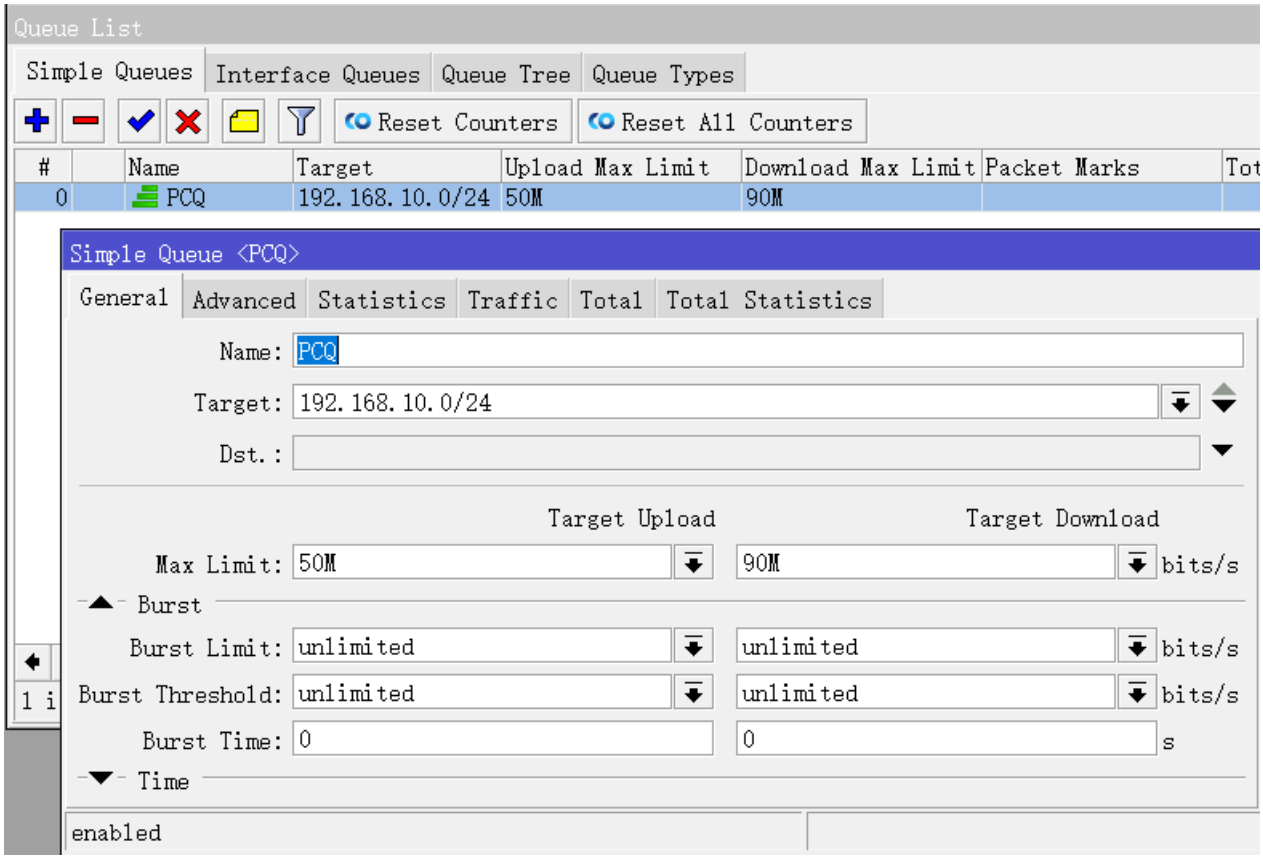
首先进入 Queue Type 中配置 PCQ 的下行，设置 pcq-download-default 为 50M，在线用户数 100 人，为保证队列容量，我们增加 Total queue size 到 5000，即 queue size*用户数量，因此是 50×100=5000，下行分类为 dst-address。（Total queue size 和 queue size 在命令行操作时属性名称为 pcq-limit 和 pcq-total-limit）



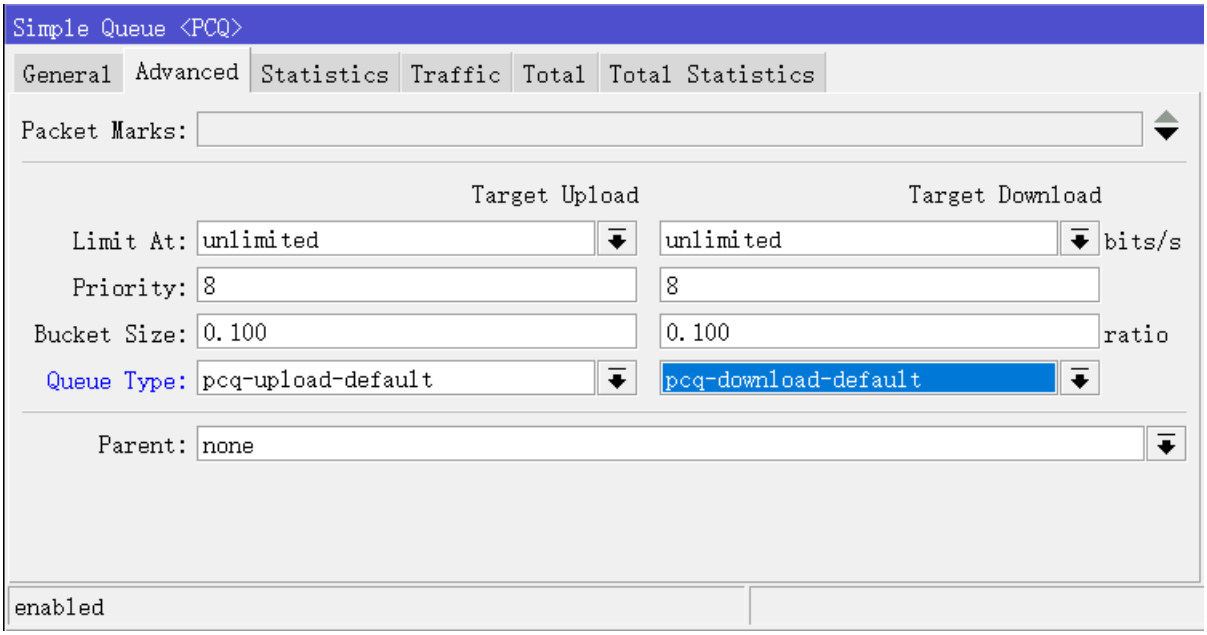
上行分类选择 src-address，并配置 5M 的上行流量配置如下：



在配置好 Queue Type 后，进入 Simple Queue 中配置流量控制规则，这里在 General 中配置总下行带宽为 90M（预留 10M 作为缓冲），总上行 70M，上行设置为 50M，内网地址段为 192.168.10.0/24：



进入 advanced 菜单，配置 Queue-type 类型，对应 Target-upload 和 Download 为 pcq-upload-default 和 pcq-download-default：



这样 PCQ 配置就完成，当只有 1 个用户在线时，可分配总带宽富裕，1 个用户能获得 50M 最大带宽，当有 2 个用户时，90M 带宽不再支持 2 个用户获取 100M，会平分为 45M，用户数量继续增加，**每个用户带宽=总带宽 90M/用户数量**，即 PCQ 会根据用户数量平均分配带宽 。

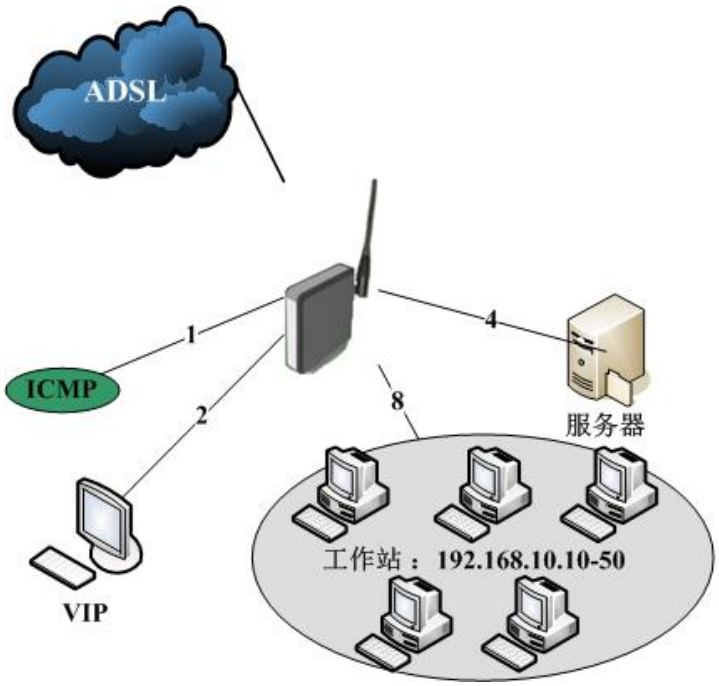
命令配置如下：

```
/queue type
set 5 pcq-rate=8M pcq-total-limit=5000KiB
set 6 pcq-rate=50M pcq-total-limit=5000KiB
/queue simple
add max-limit=50M/90M name=PCQ queue=pcq-upload-default/pcq-download-default
target=192.168.10.0/24
```

12.11 HTB 与 PCQ 流量控制

使用 MikroTik RouterOS 通过 PPPoE 连接到互联网（基于 ADSL 拨号），为了了内网的用户得到优质的网络环境，通过对流量上下行流量进行优先控制，保证特定的网络用户得到优质的网络带宽。

局域网采用以太网有线和 WiFi 无线接入方式，通过一个 bridge 将有线口与无线连接起来，网络拓扑图如下：



注：

- ADSL 是一个 PPPoE 客户端接口.运行在 ether1 的网卡上，ether1 连接到一个 ADSL-modem
- 内网 IP 地址 192.168.10.1/24，内网通过以太网和 WiFi 无线接入方式
- 通过 NAT/Masquerad 隐藏内网用户。
- ADSL 连接速度：3Mbps

客户主机

主机	IP	优先级	备注
服务器	192.168.10.6	优先级 4	公司服务器
VIP	192.168.10.7	优先级 2	重要上网人员，优先级最高
工作站	192.168.10.0/24	等级低 8，但除 icmp 协议	员工工作计算机

服务端口

协议	端口协议	优先级	目标
ICMP	icmp	最高 1	所有主机

如何获得优先的互联网带宽和流量控制，通过下面的步骤来实施：

- 由于 ADSL 有较小的缓冲空间，并当带宽满载下载速度会变慢。所以 RouterOS 配置上传或者下载不能超过 90%；
- 当 VIP 想与外面通信，将得到最优先带宽；
- ICMP 协议优先通过，得到较小的延迟；
- 需要考虑 CIR（约定带宽）即 Limit-at，MIR（最大带宽）即 Max-limit，每个流量控制需要考虑他们的 CIR 与 MIR 值

基本配置

下面我可以看到通过 ether2-wan 拨号的 ADSL 外网拨号接口，桥接 ether1-lan 与 wlan1 的 bridge1 的接口

```
[admin@MikroTik] /interface> print
Flags: D - dynamic, X - disabled, R - running, S - slave
 #      NAME                                TYPE                MTU    L2MTU
 0   R  ADSL                                pppoe-out           1480
 1   R  bridge1                             bridge              1500    65535
 2   R  ether1-lan                          ether               1500    1526
 3   R  ether2-wan                          ether               1500    1524
 4      ether3                              ether               1500    1524
 5   R  wlan1                               ether               1500    1524
 [admin@MikroTik] /interface>
```

定义 HTB 流量控制

HTB 里，我们需要考虑到父级、子级等关系，首先定义父级（parents），即上下行的总带宽，即定义整个 HTB 的总带宽

经过带宽测试后，计算出 ADSL 带宽为 2850/420kbps,我们需要在 queue tree 添加带宽限制，使用 90%的实际带宽分配给下载和上传，这里我们定义总的上下行带宽，parent 定义接口，bridge1 对应内网的下行数据（这里我们将），ADSL 则对应发出的上行数据

```
/queue tree add name=Download parent=bridge1 max-limit=2600k
/queue tree add name=Upload parent=ADSL max-limit=360k
```

ICMP 协议

对 ICMP 协议进行标记和流量控制，ICMP 协议我们需要首先满足，让所有用户得到较低 ICMP 延迟。进入 mangle 标记连接和数据报

```
/ip firewall mangle add protocol=icmp action=mark-connection new-connection-mark=icmp-con
chain=forward
/ip firewall mangle add connection-mark=icmp-con action=mark-packet new-packet-mark=icmp
chain=forward
```

我们进入 Queue tree,我们考虑到 ICMP 协议主要是网络监测，对带宽需求不大，CIR 定义为 100kbps，最大 MIR 带宽为 500kbps，保证正常的 ICMP 通信就可以了

```
/queue tree add name=icmp-down parent=Download packet-mark=icmp limit-at=100k max-limit=500k priority=1
/queue tree add name=icmp-up parent=Upload packet-mark=icmp limit-at=100k max-limit=500k priority=1
```

VIP 优先级高于其他主机

192.168.10.7 为 VIP 需要得到更多的带宽，但需要考虑到 CIR 保证使用到最低带宽，这里我们为 VIP 分配最低下行 800kbps，上行 200kbps 带宽，当然 MIR 最大可以获取到 2600kbps

标记 VIP 的连接传输与数据：

```
/ip firewall mangle add src-address=192.168.10.7/32 action=mark-connection new-connection-mark=vip-con chain=forward
/ip firewall mangle add connection-mark=vip-con action=mark-packet new-packet-mark=vip chain=forward
```

接下来进入 Queue tree 对 VIP 配置带宽规则：

```
/queue tree add name=vip-down parent=Download limit-at=1024 packet-mark=vip max-limit=5000k priority=2
/queue tree add name=vip-up parent=Upload limit-at=512 packet-mark=vip max-limit=100k priority=2
```

服务器规则

我们将 192.168.10.6 的服务器标记，并定义他们的 HTB 带宽规则

```
/ip firewall mangle add src-address=192.168.10.6/32 action=mark-connection new-connection-mark=server-con chain=forward
/ip firewall mangle add connection-mark=server-con action=mark-packet new-packet-mark=server chain=forward
```

进入 Queue tree 对服务器带宽规则：

```
/queue tree add name=server-down parent=Download limit-at=1024 packet-mark=server max-limit=2600k priority=4
/queue tree add name=server-up parent=Upload limit-at=512 packet-mark=server max-limit=300k priority=4
```

工作主机最低级别

剩下工作主机需要标记所有的传输，所有传输来至 192.168.10.0/24，因此我们使用 src-address 获取，通过标记连接（users-con），然后从连接中提取数据报（users）。

```
/ip firewall mangle add chain=forward src-address=192.168.10.0/24 action=mark-connection new-connection-mark=users-con
/ip firewall mangle add connection-mark=users-con action=mark-packet new-packet-mark=users chain=forward passthrough=no
```

这时我们需要添加 2 条新的 PCQ 规则，第一条为 ADSL-down，定义组的 dst-address 分类，即 ADSL 的下载，将 pcq-rate 设置为 0，这样将建立每个主机的动态带宽。第二条为 ADSL-up，即 ADSL 的上行，定义组为 src-address 分类，pcq-rate=100kbps，限制每台主机的上行带宽（因为 ADSL 上行相对较小）。

```
/queue type add name=ADSL-down kind=pcq pcq-classifier=dst-address
/queue type add name=ADSL-up kind=pcq pcq-rate=100k pcq-classifier=src-address
```

在 queue tree 定义

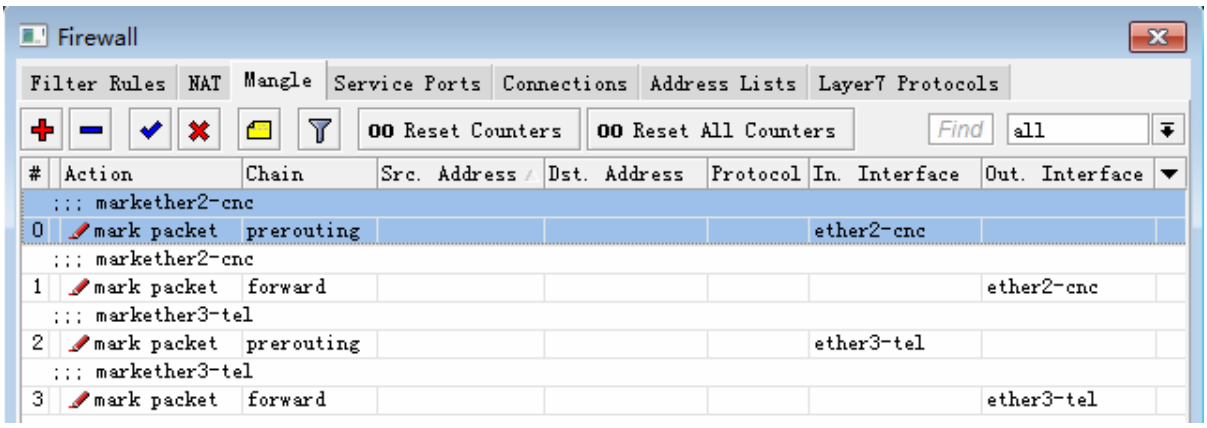
```
/queue tree add parent=Download queue=users-down packet-mark=users
/queue tree add parent=Upload queue=users-up packet-mark=users
```

12.12 网吧的 PCQ 与 HTB

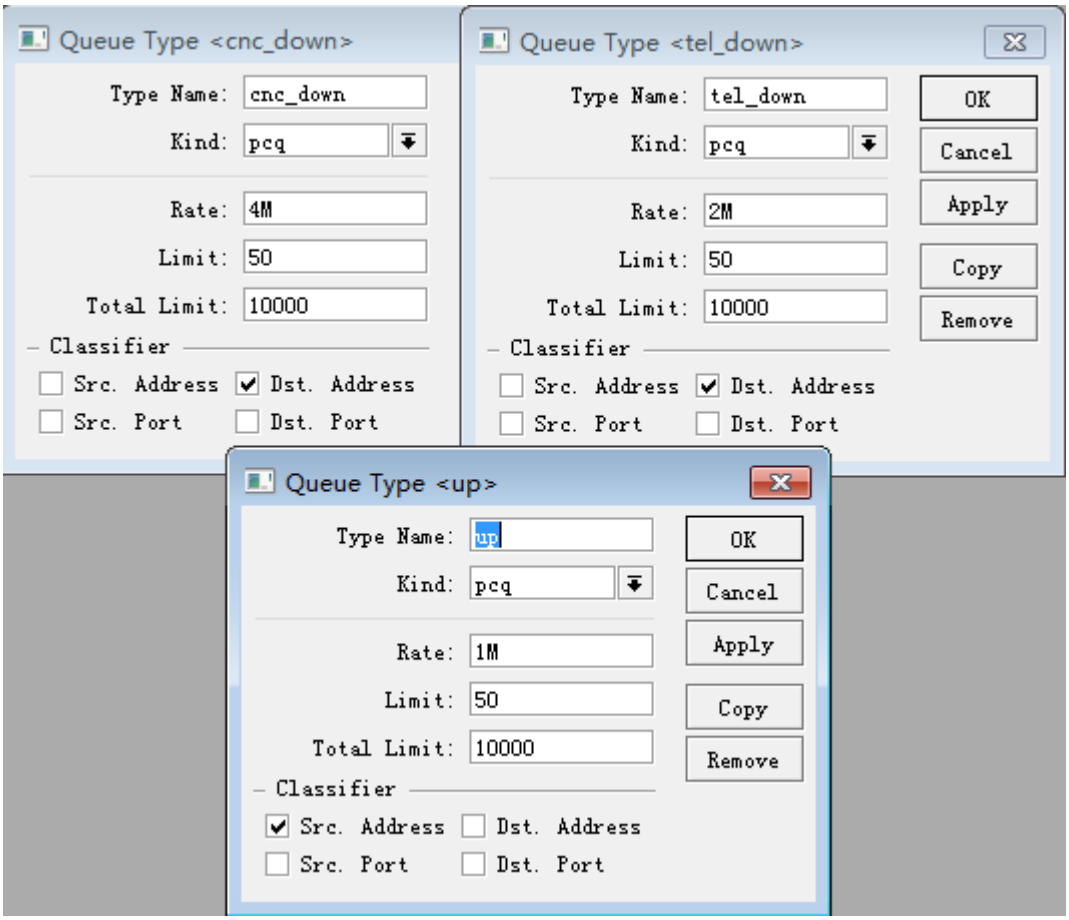
假设这样一个网络环境，我们需要实现对带宽的动态分配：Tel 带宽为 6M，Un 带宽为 12M，该实例基于 RouterOS v5 版本。

配置步骤：
1、在 ip firewall mangle 标记上下行数据流
2、进入 queue type 定义单机带宽
3、在 queue tree 定义总带宽和流量控制规则

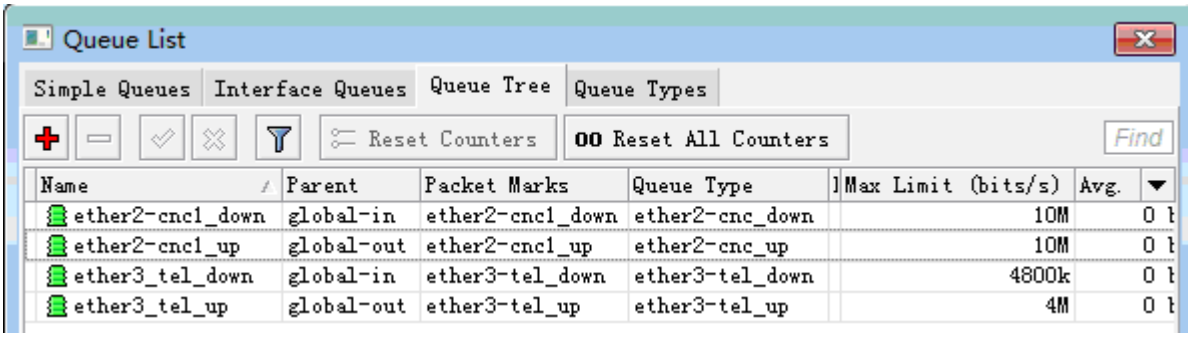
步骤 1：在 Mangle 标记上下行的标记，这里我们使用的下载标记链表为 prerouting，上传标记链表用的是 forward（为什么要用这两个链表处理数据报，可以参考第十章 RouterOS 对数据流的处理）。



步骤 2：在 Queue Type 里按照 200 台主机的数量，定义 PCQ 规则：



步骤 3：建立 Queue Tree 规则，记住保留一定带宽为缓冲，Un 我们保留 2M，Tel 我们保留 1.2M 带宽，这里下载使用的是 global-in，上传使用的是 global-out，记住 prerouting 和 input 链表标记的数据选择 global-in，其他两个链表 forward 和 output 则选择 global-out。

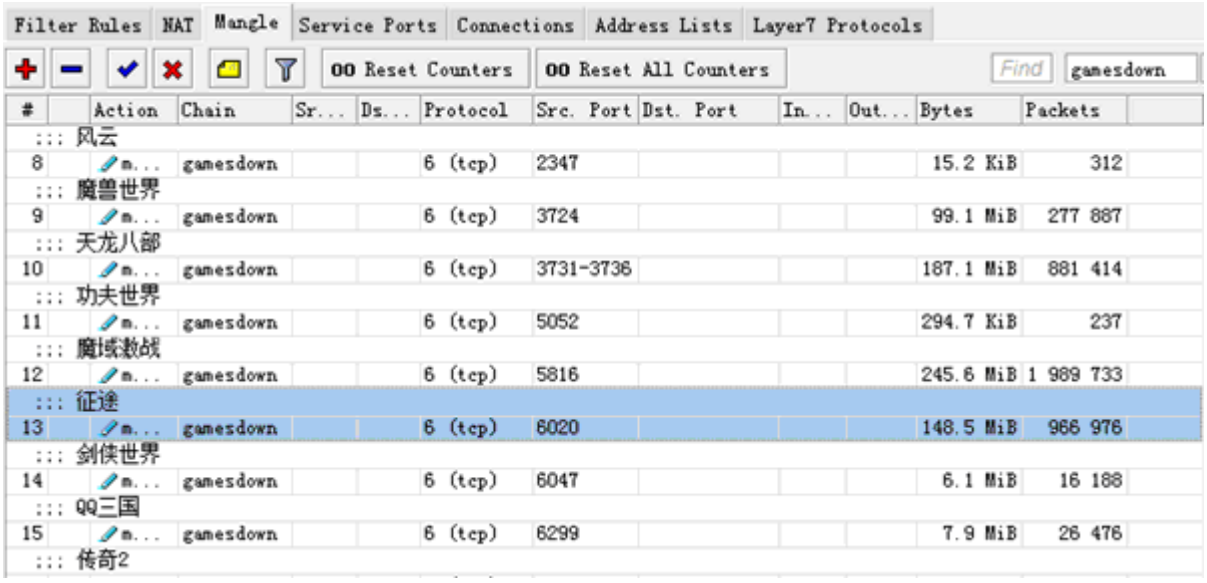


Name	Parent	Packet Marks	Queue Type	Max Limit (bits/s)	Avg.
ether2-cnc1_down	global-in	ether2-cnc1_down	ether2-cnc_down	10M	0
ether2-cnc1_up	global-out	ether2-cnc1_up	ether2-cnc_up	10M	0
ether3_tel_down	global-in	ether3-tel_down	ether3-tel_down	4800k	0
ether3_tel_up	global-out	ether3-tel_up	ether3-tel_up	4M	0

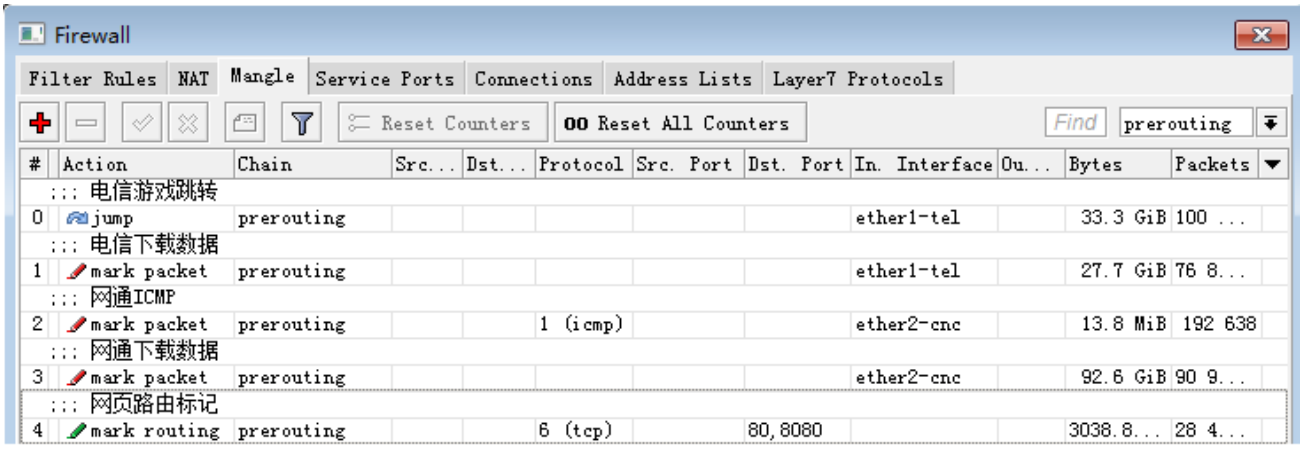
HTB 游戏优先

通过 HTB 为游戏预留带宽，保证在下载和视频情况下，游戏照样流畅，HTB+PCQ 组合实现，我们根据上面的实例配置，做以下配置调整：

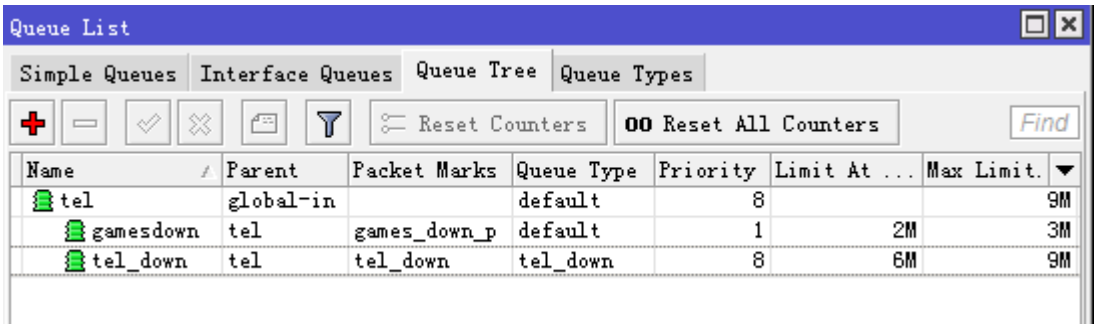
步骤 1：在原有的动态的 PCQ 流控规则上进行改进，首先导入游戏端口，建立新的 gamesdown 链表，将游戏与其他数据区分出来



通过将指定的数据转移到游戏链表进行过滤和数据报处理：



假设 Tel 带宽是 11M，预留 2M 为缓冲带宽，最大带宽为 9M，Tel 线路下行的 HTB 设置，游戏优先级为 1 最高，其他下行数据为 8 最低；这里游戏只分配了 3M 最大带宽，最低保证 2M，对于游戏带宽较小不需要那么大；其他下行数据最低保证 6M。



如果需要也可以为游戏流量配置 PCQ 规则，定义一个游戏的 PCQ 队列类型 Queue-type 对每个用户进行带宽控制。

12.13 Connection Rate 流量控制

Connection Rate 是一个防火墙标记器，允许捕获在当前传输的连接速度

Connection Rate 原理

每个连接项目在 connection tracking 表中是双向通讯。每次得到相关的数据报到特定的项目，数据报的长度值（包括 IP 数据报头）被添加到“Connection-bytes”值，换句话说，Connection-bytes 包括两部分上行和下行。

Connection Rate 计算连接的速度基于"connection-bytes"的变化。Connection Rate 每秒会被重新计算，且没有任何平均值。

```
ConnectionRate ::= [!]From-To
  From,To ::= 0..4294967295       (integer number)
```

两个选项 "connection-bytes" 和"connection-rate" 工作只能在 TCP 和 UDP 传输。(你需要指定协议启动这些选项) 在 "connection-rate"您可以指定速度，和你想捕获范围。

例如：这个规则是捕获当连接速度低于 100kbps 通过路由器的 TCP/UDP 传输

```
/ip firewall filter
add action=accept chain=forward connection-rate=0-100k protocol=tcp
add action=accept chain=forward connection-rate=0-100k protocol=udp
```

注：Connection Rate 从 3.30 才能获得，这个选项是用于捕获传输密集的连接

传输优先级

Connection-rate 能被使用在各种方式，通常的方式是使用队列树进行 HTB 的优先级控制，检测并设置低优先级给“heavy connections”（连接在一段时间内保持较快速率，例如：P2P、HTTP、FTP 下载）通过这样做，你可以区分所有其它传输的优先次序，通常包括 VOIP、HTTP 浏览和在线游戏

connection-rate 选项没有任何平均值，需要确定识别"heavy connections"的差额。如果我们假设正常的 HTTP 浏览连接小于 500kB(4Mb，即 connection-bytes 值)长度，VOIP 需要不超过 200kbps 的流量，那么每次连接当超过 500kB 后，仍然有 200kbps 的流量将被认为是"heavy connections"

(对于 HTTP 浏览和 VOIP 可能有不同的"connection-bytes"在你的网络环境中，所以请你在实际操作时，这个实例仅做参考。)

实例脚本

下面实例让我们假设，我们有 6Mbps 上传和下载

```
/ip firewall mangle
add chain=forward action=mark-connection connection-mark=!heavy_traffic_conn
new-connection-mark=all_conn
add chain=forward action=mark-connection connection-bytes=500000-0 \
    connection-mark=all_conn connection-rate=200k-100M \
    new-connection-mark=heavy_traffic_conn protocol=tcp
add chain=forward action=mark-connection connection-bytes=500000-0 \
```

```
    connection-mark=all_conn connection-rate=200k-100M \
    new-connection-mark=heavy_traffic_conn protocol=udp
add chain=forward action=mark-packet connection-mark=heavy_traffic_conn \
    new-packet-mark=heavy_traffic passthrough=no
add chain=forward action=mark-packet connection-mark=all_conn \
    new-packet-mark=other_traffic passthrough=no

/queue tree
add name=upload parent=public max-limit=6M
add name=other_upload parent=upload limit-at=4M max-limit=6M \
    packet-mark=other_traffic priority=1
add name=heavy_upload parent=upload limit-at=2M max-limit=6M \
    packet-mark=heavy_traffic priority=8
add name=download parent=local max-limit=6M
add name=other_download parent=download limit-at=4M max-limit=6M \
    packet-mark=other_traffic priority=1
add name=heavy_download parent=download limit-at=2M max-limit=6M \
    packet-mark=heavy_traffic priority=8
```

脚本说明

在 mangle 中，我们需要分离所有连接到 2 个组中，这样数据报标记从 2 个组取得。我们讨论的客户的传输理论上大多标记在 forward 链表中。

请记住，“heavy”连接将有低的优先级，队列将打压 max-limit—heavy 连接将被限制速度。这样引起改变高优先级连接获取更多的带宽，当在一次产生 connection-rate 将上升，并导致其改变为较低优先级。为了避免这一点，我们必须确保，一旦发现“heavy”连接，剩下的标记在所有时间仍然是"heavy"连接

Mangel 规则配置

```
/ip firewall mangle
add chain=forward action=mark-connection connection-mark=!heavy_traffic_conn
new-connection-mark=all_conn
```

这个规则将确定"heavy"连接，连接将只剩下“heavy”

```
add chain=forward action=mark-connection connection-bytes=500000-0 \
    connection-mark=all_conn connection-rate=200k-100M \
    new-connection-mark=heavy_traffic_conn protocol=tcp
add chain=forward action=mark-connection connection-bytes=500000-0 \
    connection-mark=all_conn connection-rate=200k-100M \
    new-connection-mark=heavy_traffic_conn protocol=udp
```

这两个规则将根据我们的标准标记所有 heavy 连接，每次连接在第一次超过 500KB 流量后，仍然保持 200kbps 以上速度被认为“heavy”。

```
add chain=forward action=mark-packet connection-mark=heavy_traffic_conn \
    new-packet-mark=heavy_traffic passthrough=no
```

```
add chain=forward action=mark-packet connection-mark=all_conn \
    new-packet-mark=other_traffic passthrough=no
```

最后 2 条规则在 mangle 中将标记数据报传输从相应的连接中。

队列配置

这个是一个简单的队列树被放到接口的 HTBThis is a simple queue tree that is placed on the Interface HTB – 你的 ISP 连接的“wan”接口，“lan“连接你的内网客户。如果你有多个 wan 接口或者多个 lan，你将需要标记上行和下行分离标记

```
/queue tree
add name=upload parent=public max-limit=6M
add name=other_upload parent=upload limit-at=4M max-limit=6M \
    packet-mark=other_traffic priority=1
add name=heavy_upload parent=upload limit-at=2M max-limit=6M \
    packet-mark=heavy_traffic priority=8
add name=download parent=local max-limit=6M
add name=other_download parent=download limit-at=4M max-limit=6M \
    packet-mark=other_traffic priority=1
add name=heavy_download parent=download limit-at=2M max-limit=6M \
    packet-mark=heavy_traffic priority=8
```

Connection-rate HTB 事例

通过 Connection-rate 配合 HTB 分离正常的网页浏览和网页视频，方法是标记网页的 tcp/80 端口，并区分他们连接速率，将高速率的流量认为是网页视频、剩下的则是正常的网页浏览，通 HTB 将正常的网页浏览设置为优先。

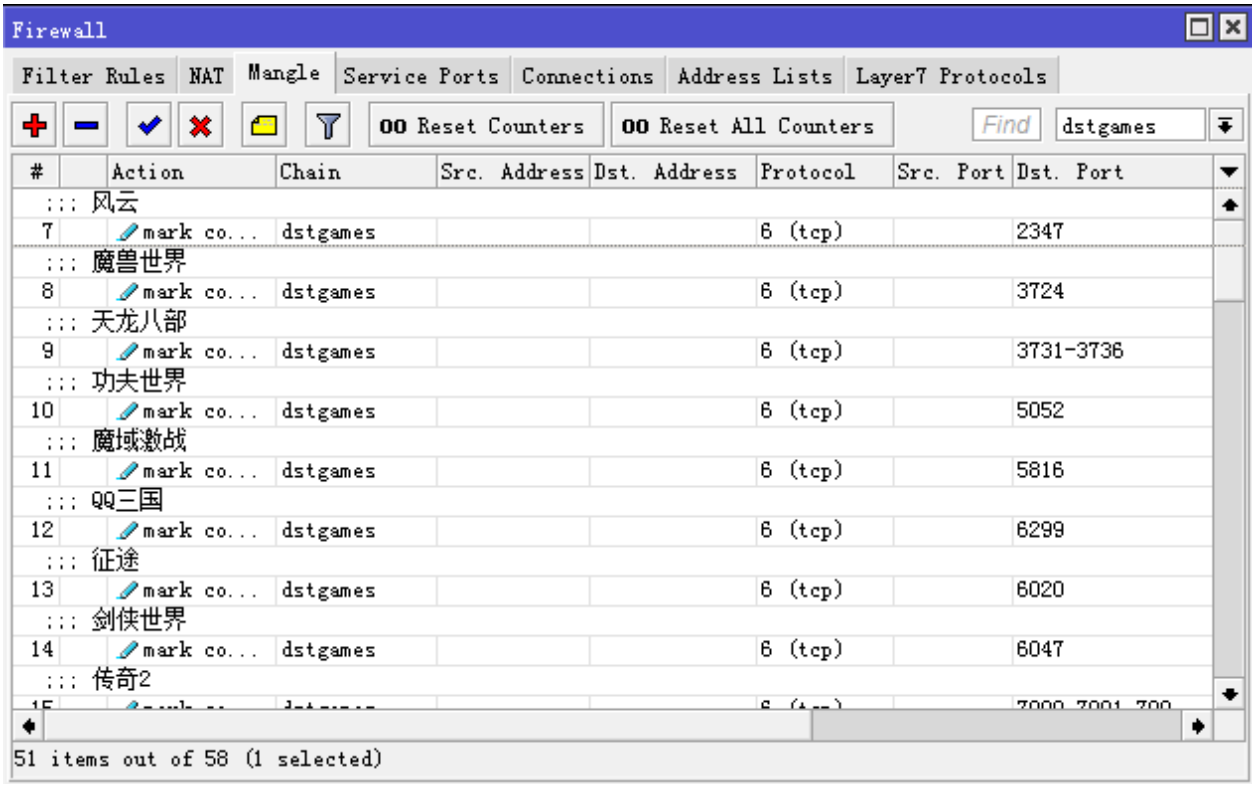
通过 Mangle 标记

我们需要考虑一个网络的优先处理结构，首先是游戏带宽最优先、其次是正常浏览网页带宽、然后是网页视频的带宽，最后在才是其他的下载数据，根据这一的结构我们在 mangle 标记也是从游戏开始到最后的下载数据。

首先我们需要导入游戏优先的标记，将游戏优先文件 games_v2.rsc 放入 RouterOS 的 files 根目录下，然后在命令行下使用 import 命令，导入脚本文件，如下图输入以下命令，并回车执行提示：脚本文件加载，并执行成功：

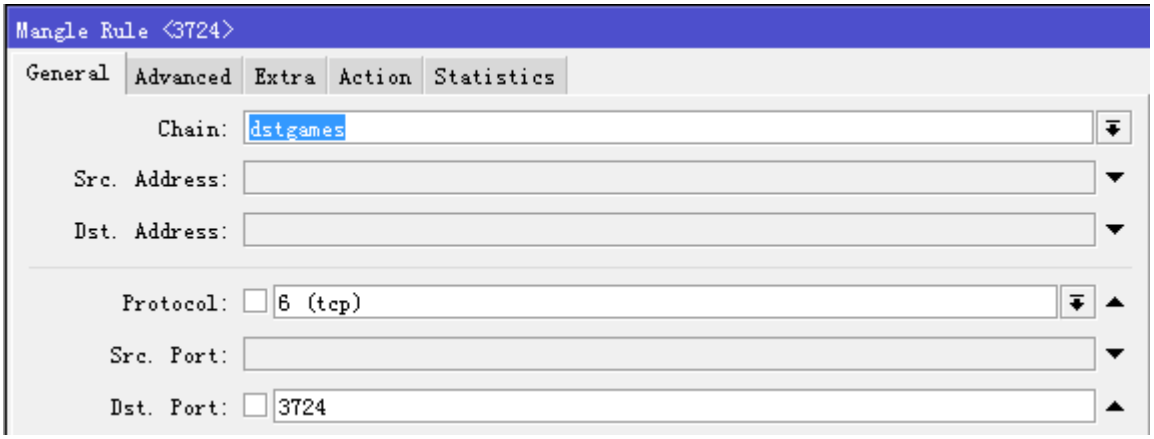
```
[admin@MikroTik] > import games_v2.rsc
Opening script file games_v2.rsc
Script file loaded and executed successfully
[admin@MikroTik] >
```

我们进入 ip firewall mangle 中可以找到导入游戏标记规则，我们选择右上角 dstgames 的自定义链表，可以看到各种游戏的标记，你也可以在此链表添加自己的游戏标记

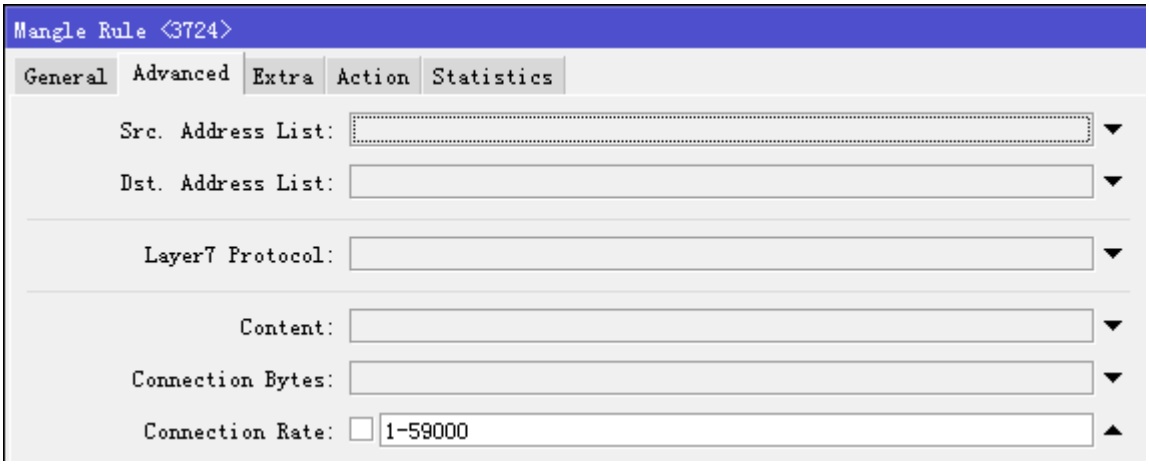


每条规则都标记的是目标端口，因为这里我将使用 forward 链表跳转数据到这个游戏标记，如果使用 prerouting 链表注意接口方向。

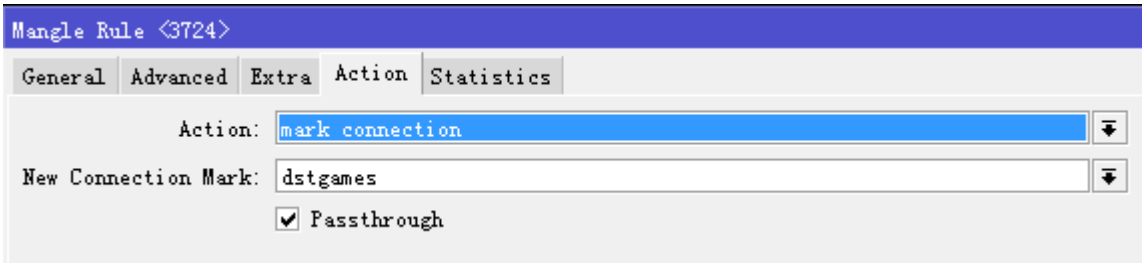
下面是魔兽世界的端口 tcp/3724，标记服务器的端口



在每个游戏标记规则里，都使用了 connection-rate 的参数，参数值为 1-59k，即带宽使用在 1-59kps 的数据认为是该端口的游戏流量，否则认为是其他数据，这样的目的是避免该端口被下载所占用，导致游戏带宽控制失效，当然这个值可以根据你自己的需要调整和修改



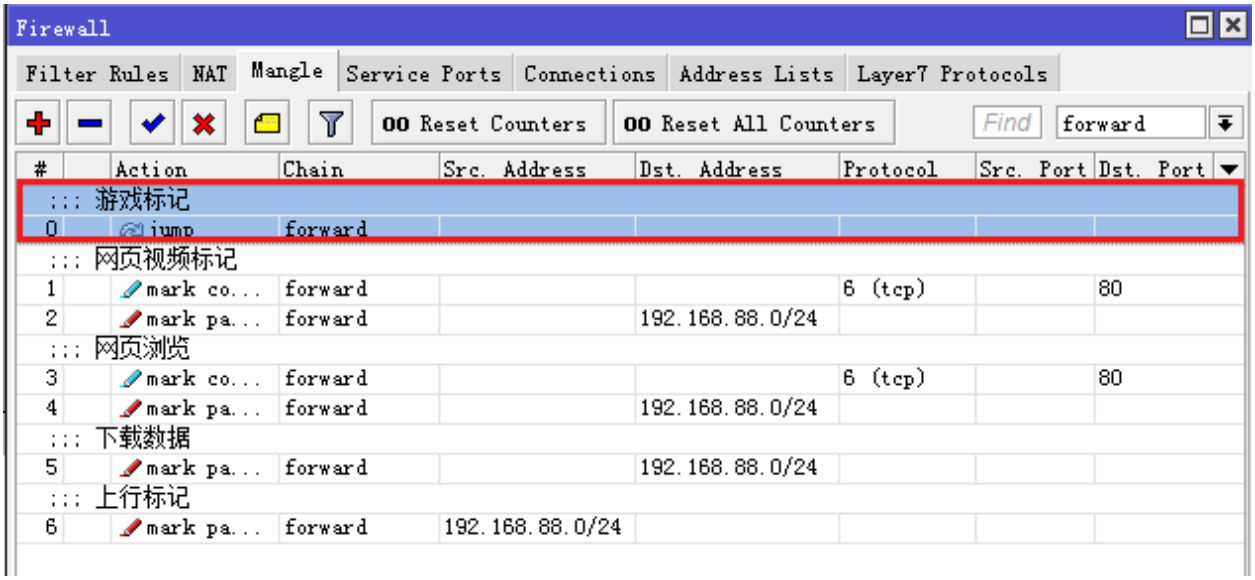
最后是在 action 里标记为 mark-connection，并填写标记名称 dstgames，passthrough=yes 要传递给后面的数据报标记规则



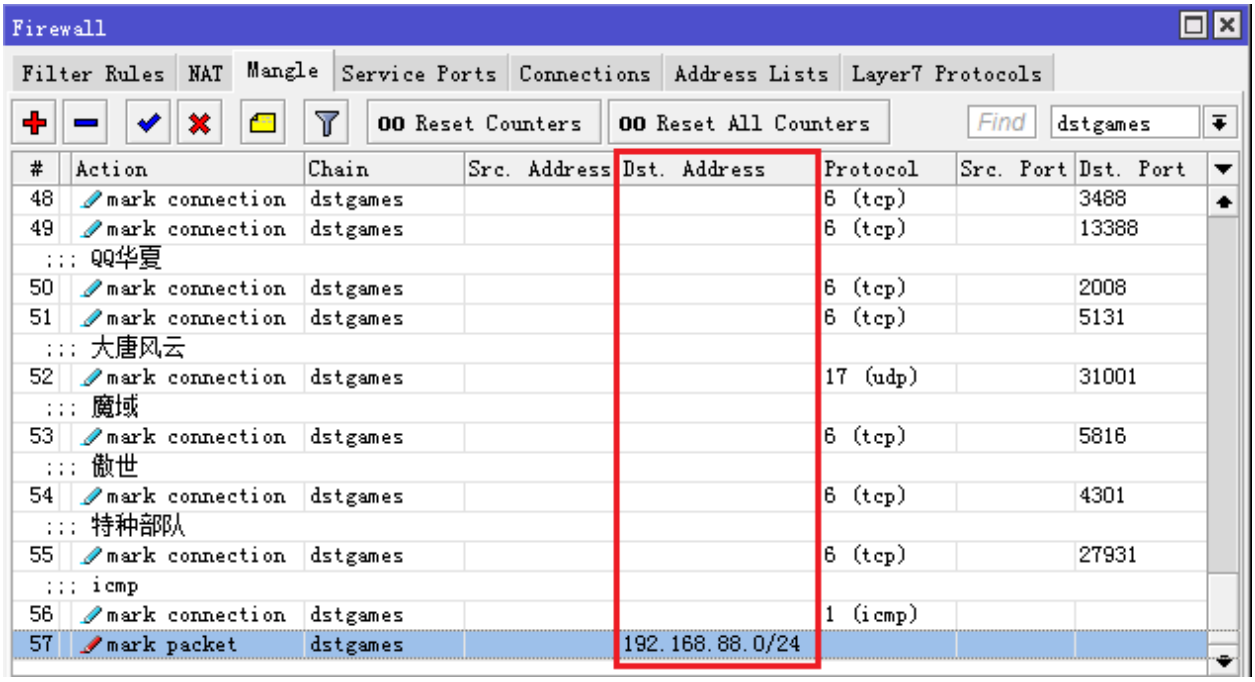
Forward 标记

我们进入 forward 链表，添加一条跳转的规则，将 forward 中所有数据首先进入 dstgames 的链表过滤一次，将游戏分离出来

```
/ip firewall mangle add chain=forward action=jump jump-target=dstgames
```



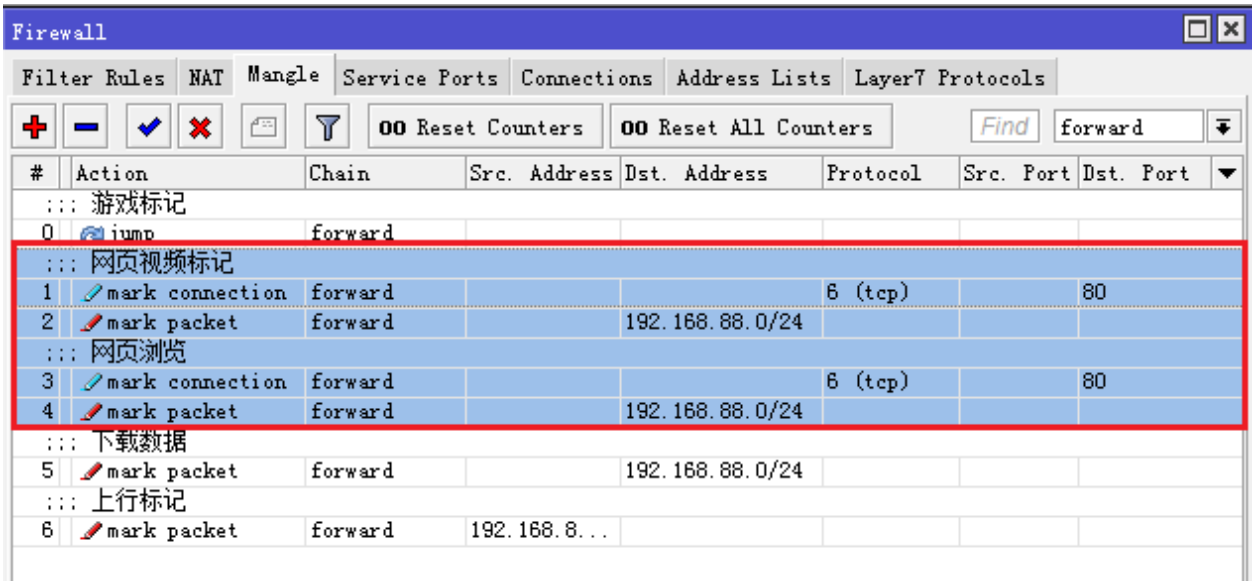
这条规则在 forward 链表里排在最前面，因为是首先处理的数据。这里我们的内网地址段是 192.168.88.0/24，我们需要通过地址来区分下载和上传，所以我们需要返回 dstgames 链表里，将数据报标记规则修改下



在 dstgames 里最后一条规则说 mark-packet，即汇总和标记之前所有游戏连接的数据，并生成可以被 queue 流控规则使用的 packet，之前的游戏连接是没有区分上行和下行的，所以我们这里我们通过 dst-address 将到内网（即下行）的数据标记出来，这样可以得到游戏的下行数据，该规则的 passthrough=no。注意：你的内网地址根据你的设置修改。

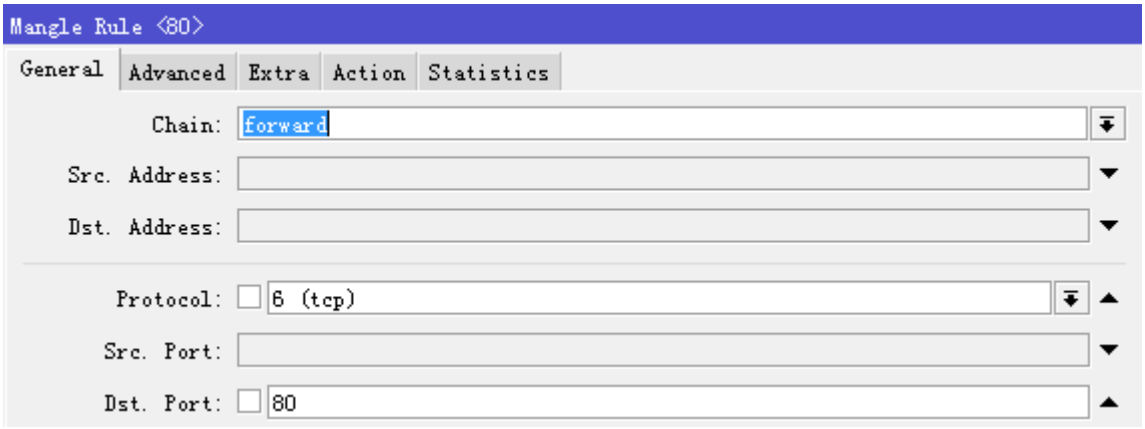
网页视频和浏览区分

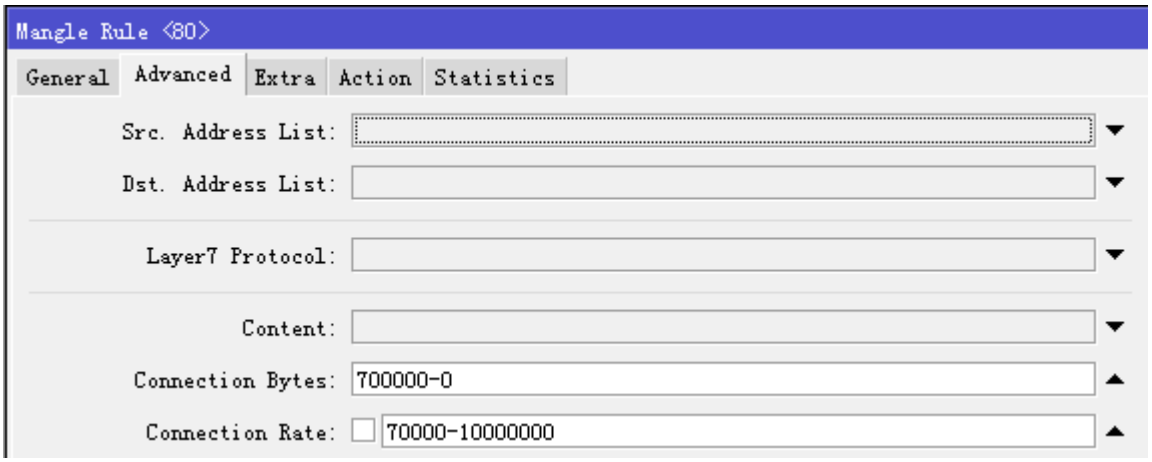
网页视频和网页浏览的区分，仍然采用 connection-rate 参数，区分网页的规则一共 4 条，如下图：

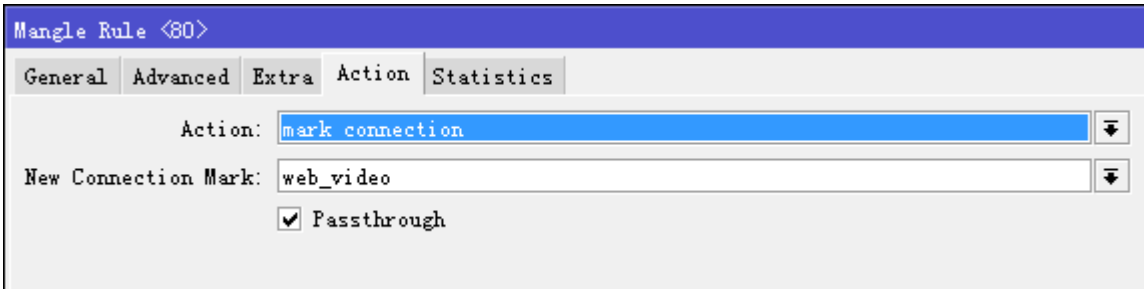


4 条规则里关键是在第一条连接标记，用于区分网页视频的速率，

```
chain=forward action=mark-connection new-connection-mark=web_video passthrough=yes
protocol=tcp dst-port=80 connection-bytes=700000-0 connection-rate=70k-10M
```

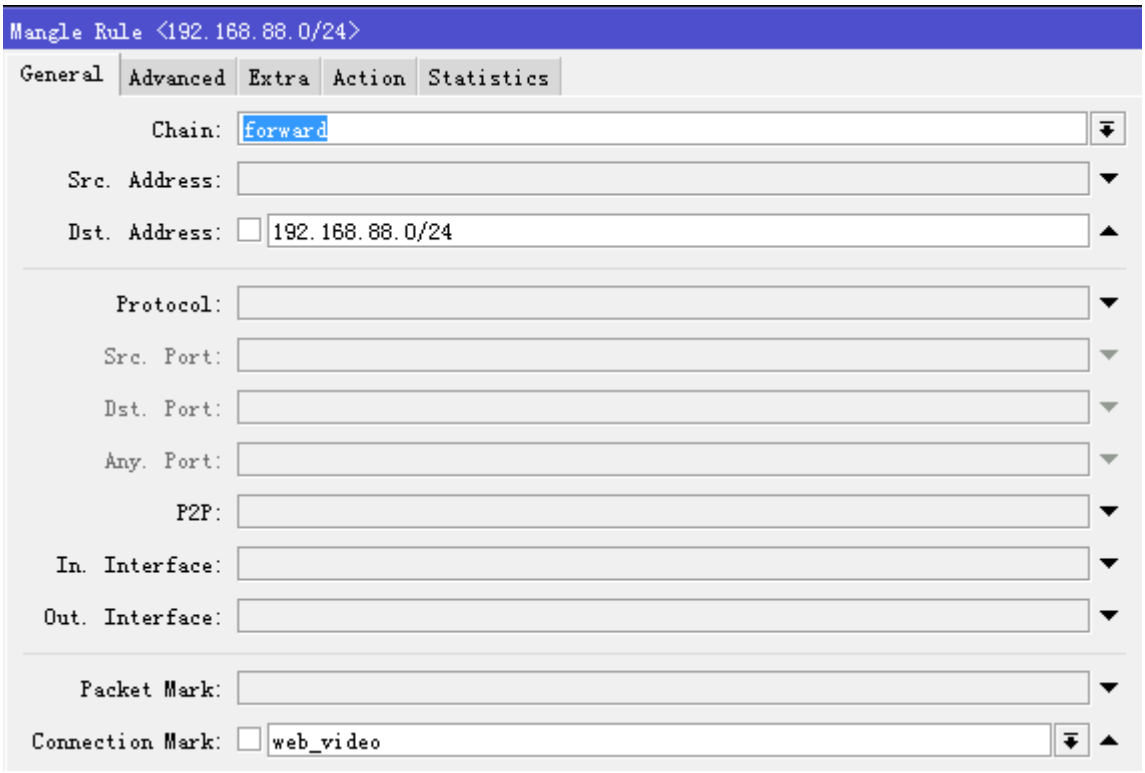


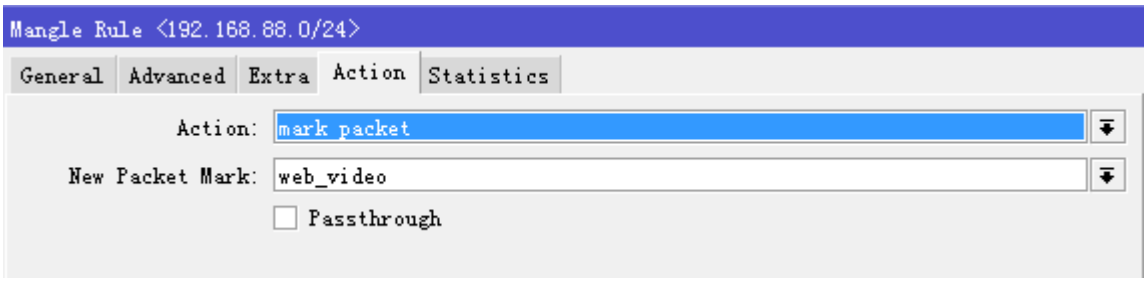




标记所有 tcp/80 端口，并设置 connection-bytes 为 700k 范围内，即初次使用了 700K 的字节，之后仍然保持 70kbps~10Mbps 的速率，就认为是网页视频，并标记名称为 web_video

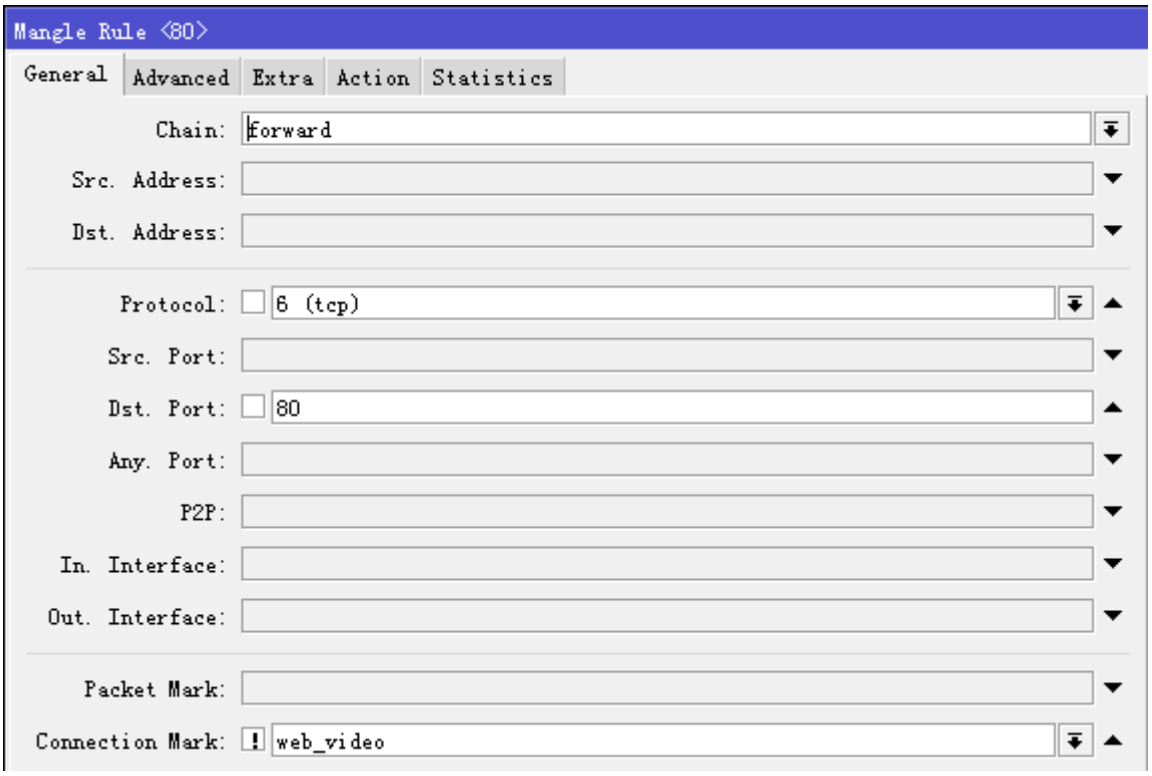
接下来的一条规则就只需要从这个连接里提取数据报，并指明目标地址是 192.168.88.0/24 的下行数据，action=mark-packet，取名标记为 web_video，注意这个 web_video 和连接的 web_video 并不相同，一个是连接一个是数据，这里的 passthrough=no，因为已经被数据标记处理，不需要交给后面的规则

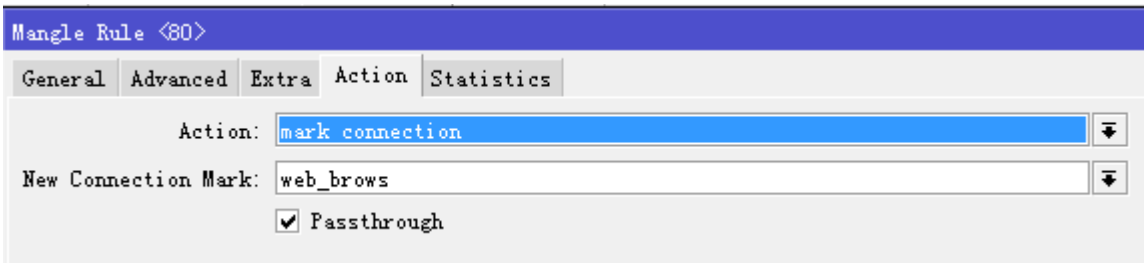




接下来我们标记网络浏览连接，仍然是是 tcp/80，但这里的 connection-mark 设置为非 web_video 的连接

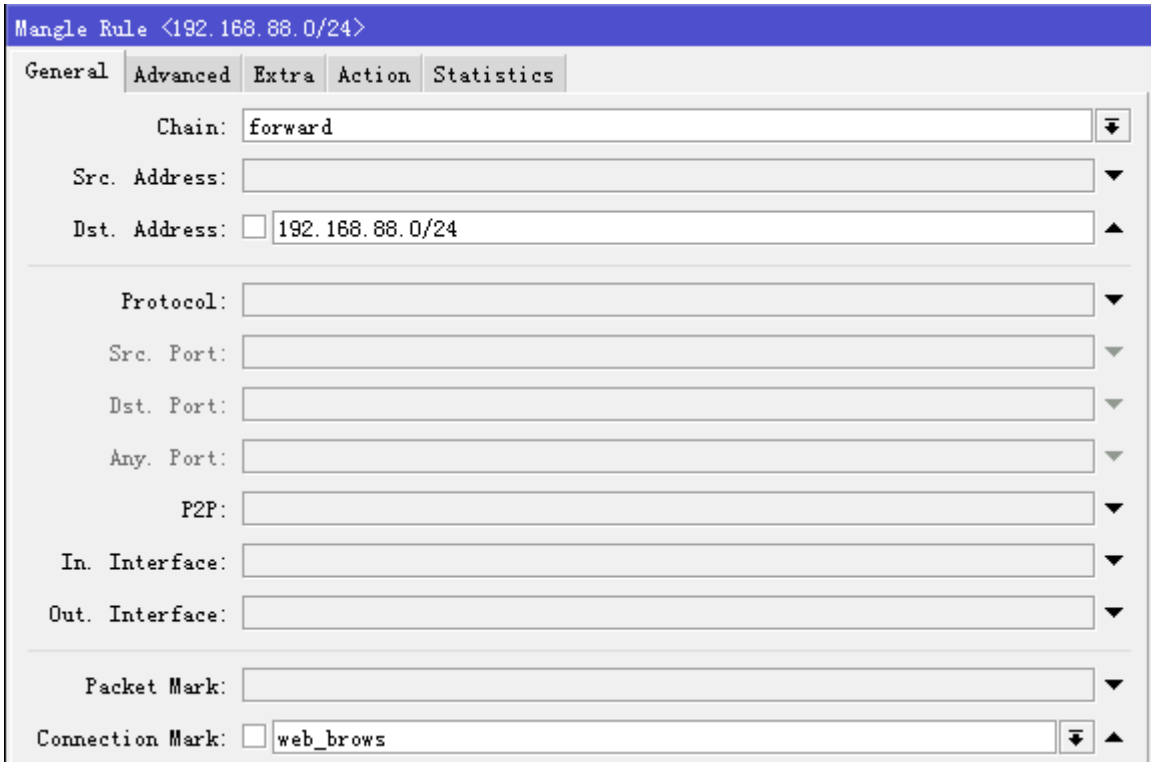
```
chain=forward action=mark-connection new-connection-mark=low_web passthrough=yes
protocol=tcp dst-port=80 connection-mark=!web_video
```

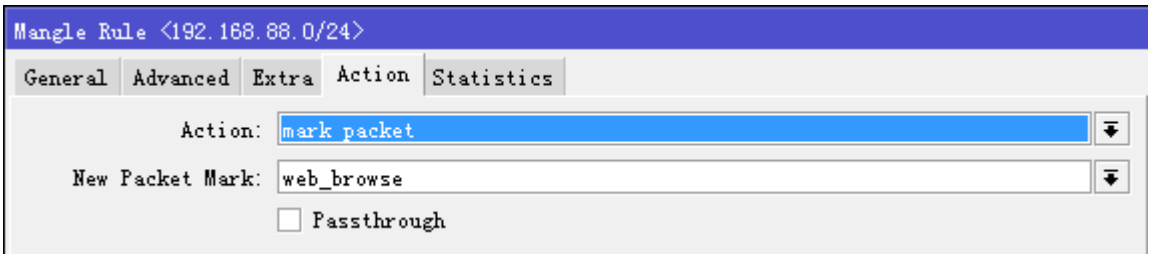




标记连接名称为 web_brows，并传递给后面的数据报标记规则

同样的方式，提取 web_brows 的数据报标记，选择目标地址并同样取名为 web_brows，passthrough=no

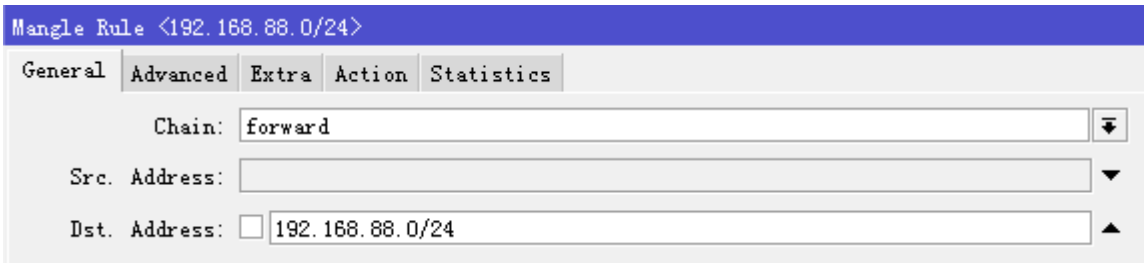


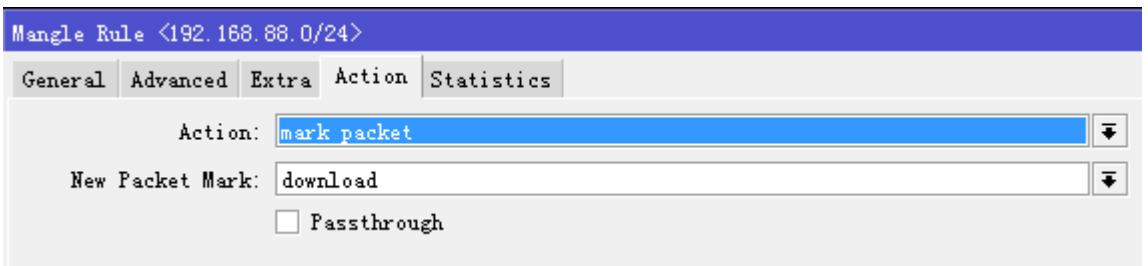


其他下载数据标记

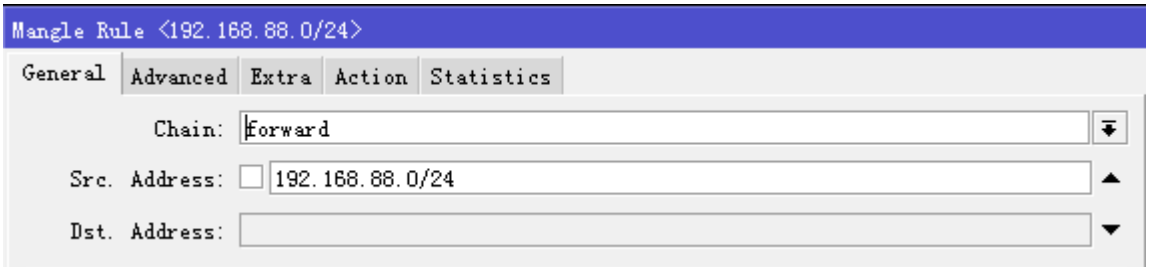
这样我们已经将部分游戏、网页视频和网页浏览区分出来，剩下的被认为是其他下载数据，直接做数据报标记

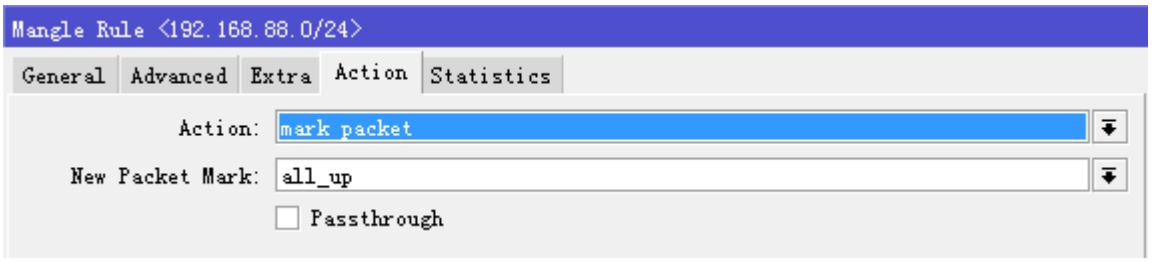
```
chain=forward action=mark-packet new-packet-mark=download passthrough=no
dst-address=192.168.88.0/24
```





最后一个规则则是标记上行数据，我们主要处理数据优先是下行，上行需要的带宽相对较小，我们只需要做一次性的标记，取名标记为 all_up，通过后面的 HTB 的 PCQ 给一个适合的带宽即可。

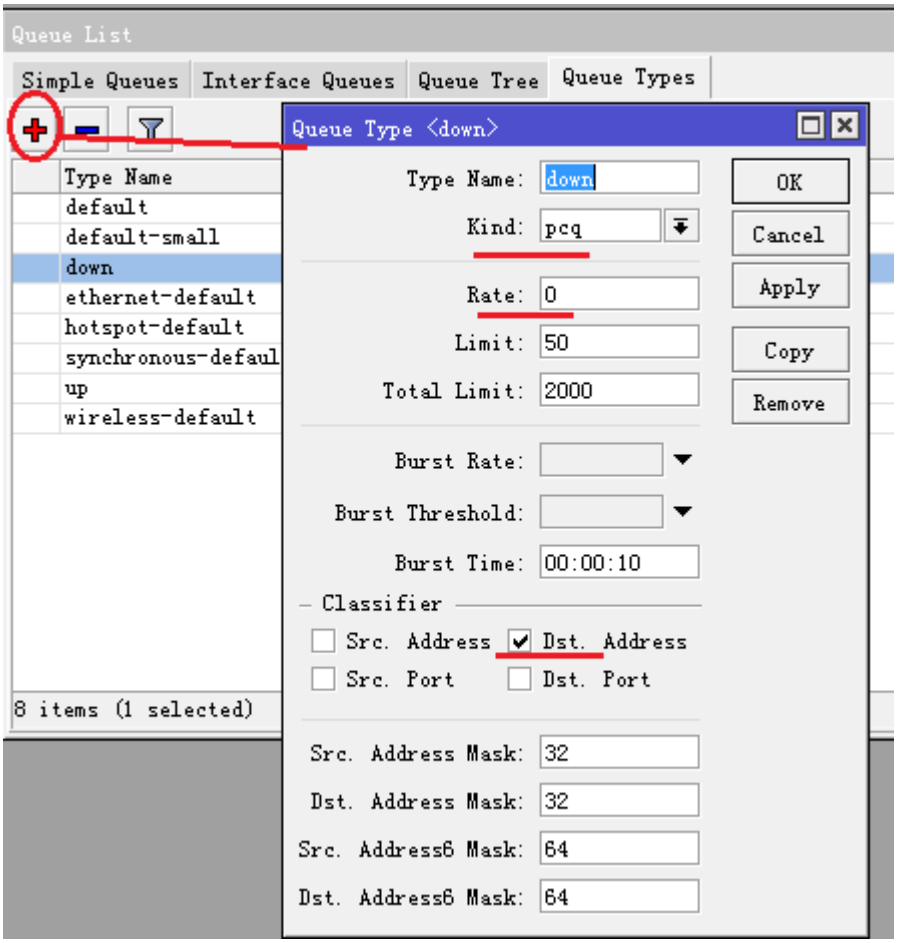




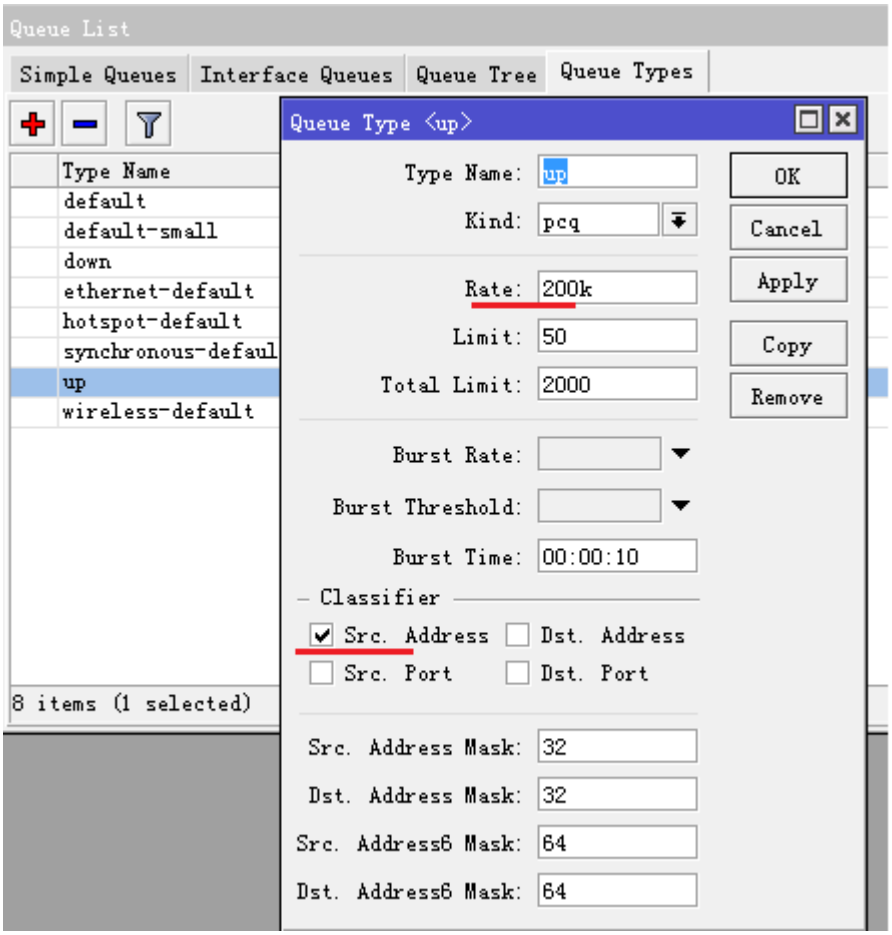
HTB 和 PCQ 设置

HTB 令牌桶方式控制各个数据的优先级，并通过 PCQ 动态分配带宽，我们进入 queue 的 tree 设置 HTB，但我们首先还是要先设置 PCQ 参数

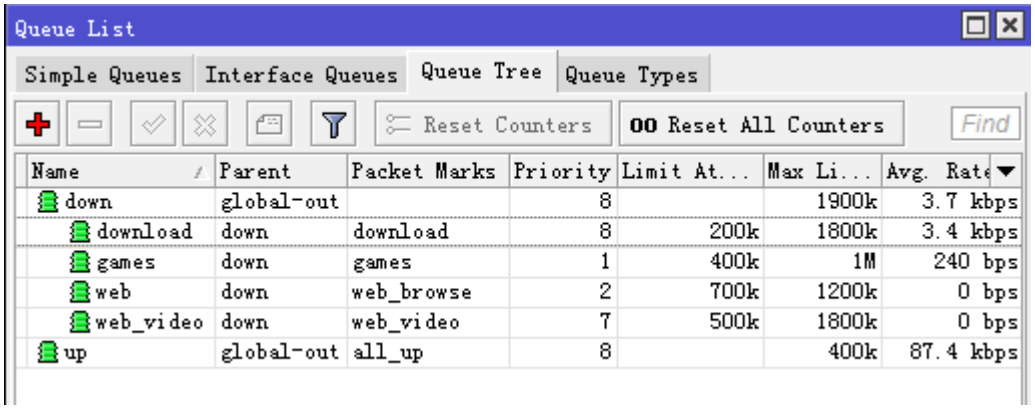
进入 queue type 添加 PCQ 规则，我们添加一个 down 的 PCQ，kind=pcq，rate 设置每台主机的带宽，这里我们选择 0，即动态分配带宽，你也可以设置一个固定值，以 k 或者 m 为单位，选择 dst-address 为下行的规则



添加上行带宽控制规则，这里我们将上行的 PCQ 带宽参数，rate=200k，即每台主机的上行为 200k（由于本人使用的是 2M ADSL，上行只有 512kbps，所以上行设置较小），上行带宽类型为 src-address



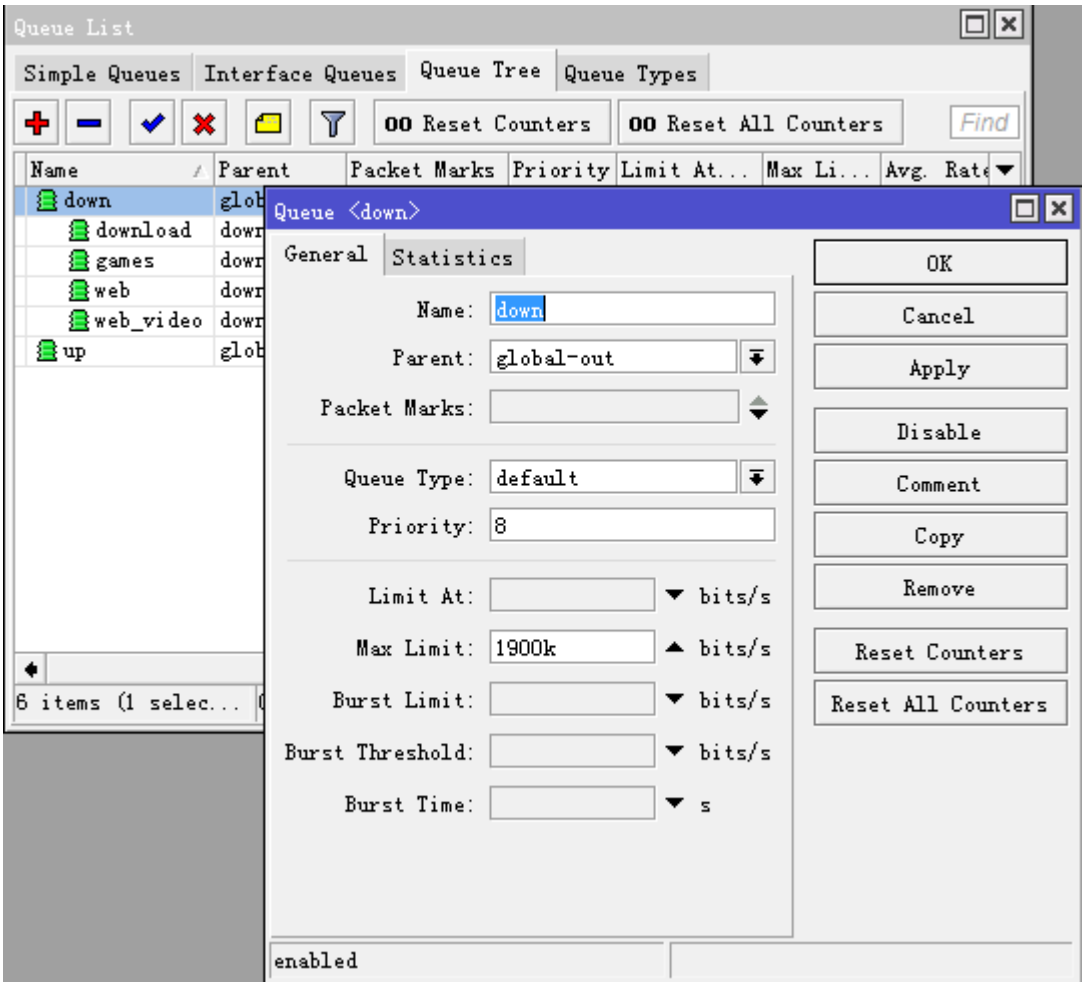
PCQ 设置完成后，我们接下来就需要配置 HTB，HTB 是由父级队列和子队列组成，我们只对下行做 HTB，上行数据则采用普通的 PCQ 限速（没有子队列的规则不能称为 HTB），如下图



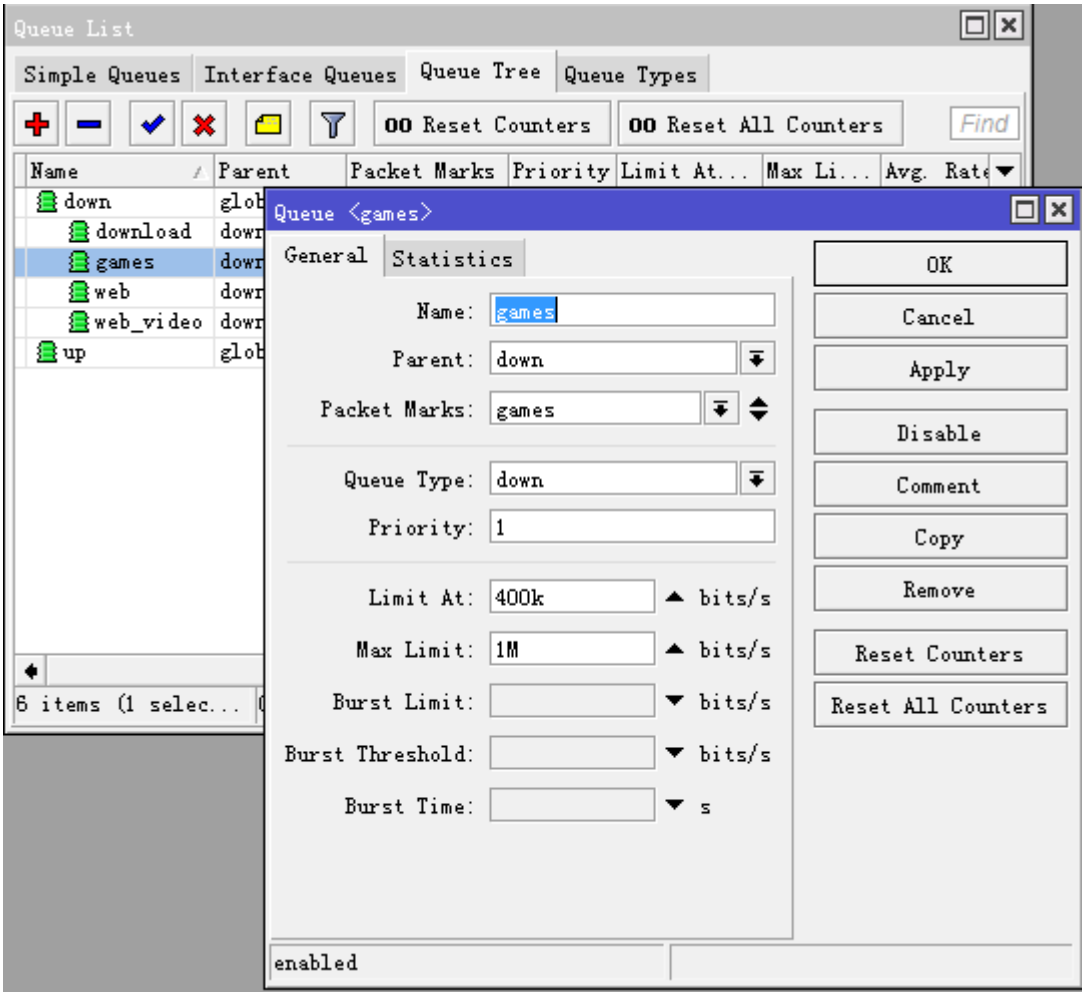
配置原则说明：

1、down 是下行父级队列，负责总带宽分配，up 是上行队列，仅控制上行。
2、down 和 up 都是通过 forward 链表标记的，属于 global-out，所以我们使用的 parent 为 global-out
3、down 父级队列下有 4 个子队列，分配是从属于 down，从父级获取带宽，子队列自己有各自的优先级 priority，1 为最高，8 最低，游戏优先级最高 1，web 优先级其次 2，web 视频 7，下载数据最低 8.
4、子队列的优先级与父级队列是不能比较，max-limit 是最大能获取的带宽，limit-at 为保障带宽，所有子队列 limi-at 之和小于等于父级 max-limit 带宽
5、这里是 2M 的 ADSL，我们分配父级带宽不能将 2Mbps 分配完，并须预留一部分作为缓冲，这里下行是 1900kbps，上行 400kbps（如果你是 10Mbps 的带宽建议，预留 2M 作为缓冲）

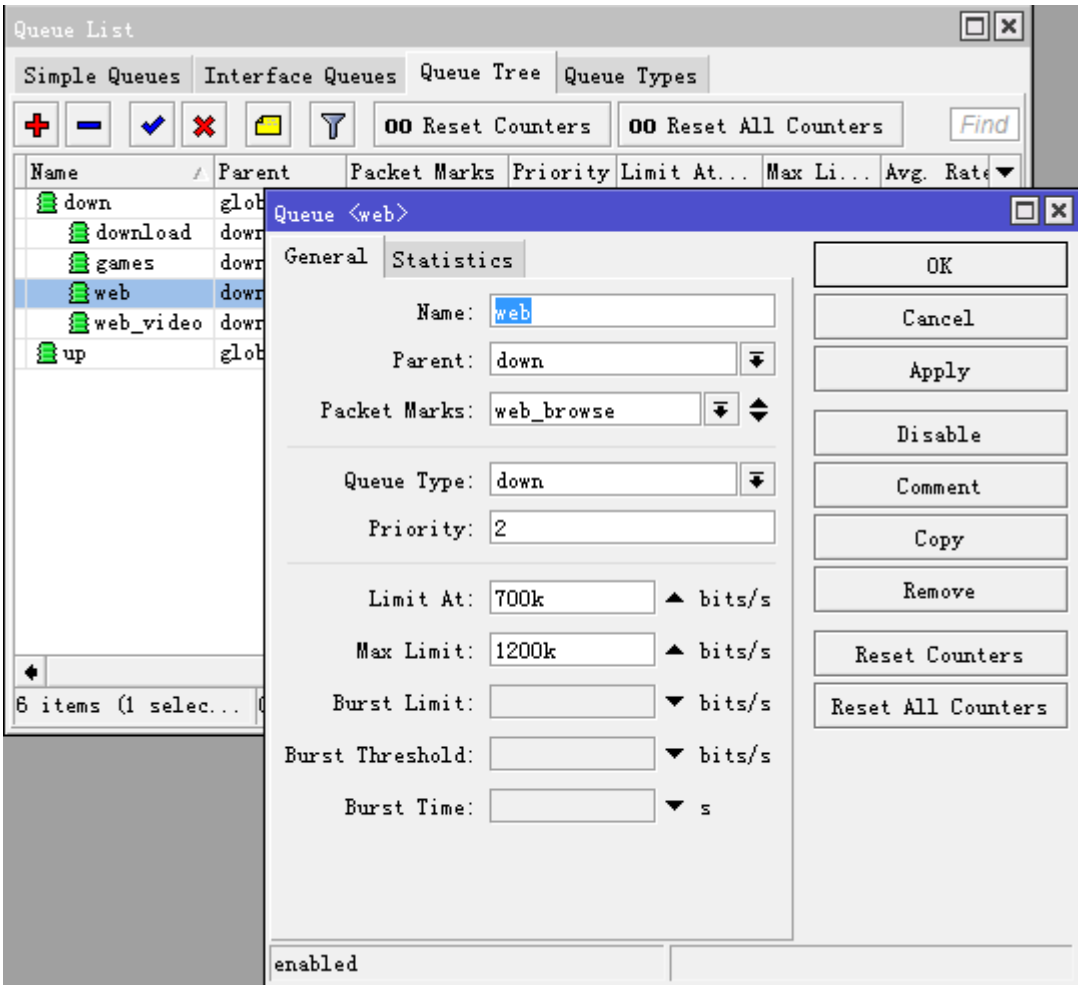
根据以上原则，添加 HTB 的规则的思路就明确了，首先添加父级队列，这里我们不设置任何 packet marks 标记，仅作为父级带宽分配给下面的子队列



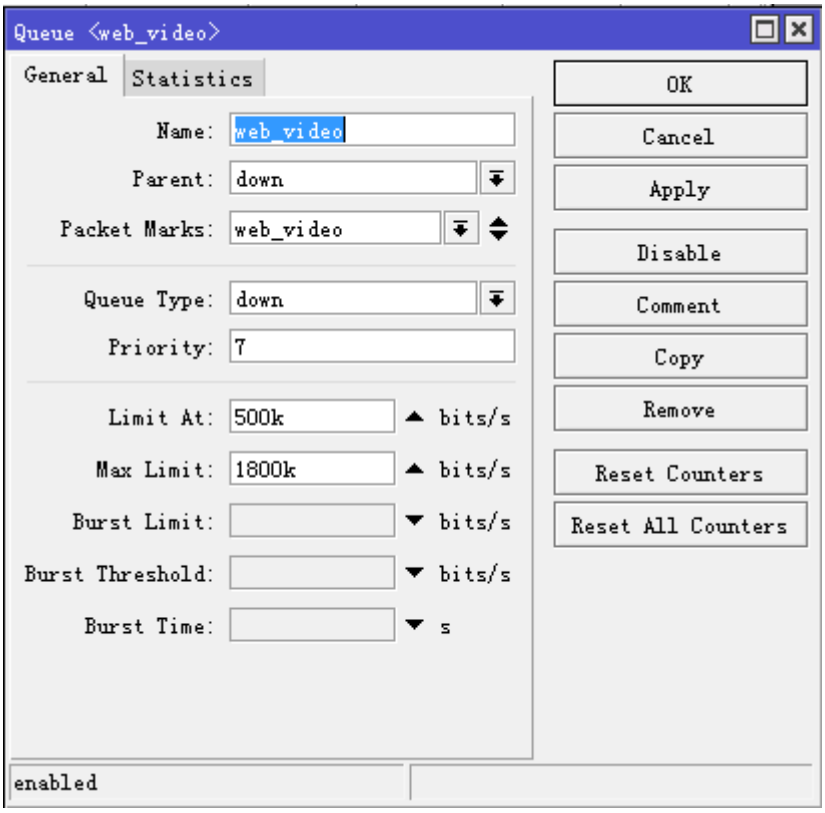
我们添加子队列，首先添加游戏子队列，我们需要设置父级 parent=down，需要从 down 获取带宽，packet-mark 为之前标记的 games，选择 queue type 为刚才的 PCQ 规则 down（给每个主机动态分配带宽），priority 优先级为 1 最高，我们给游戏分配 1M 带宽，最低保证 400kbps 带宽，如果有剩余带宽可以从父级获取得到最大 1Mbps



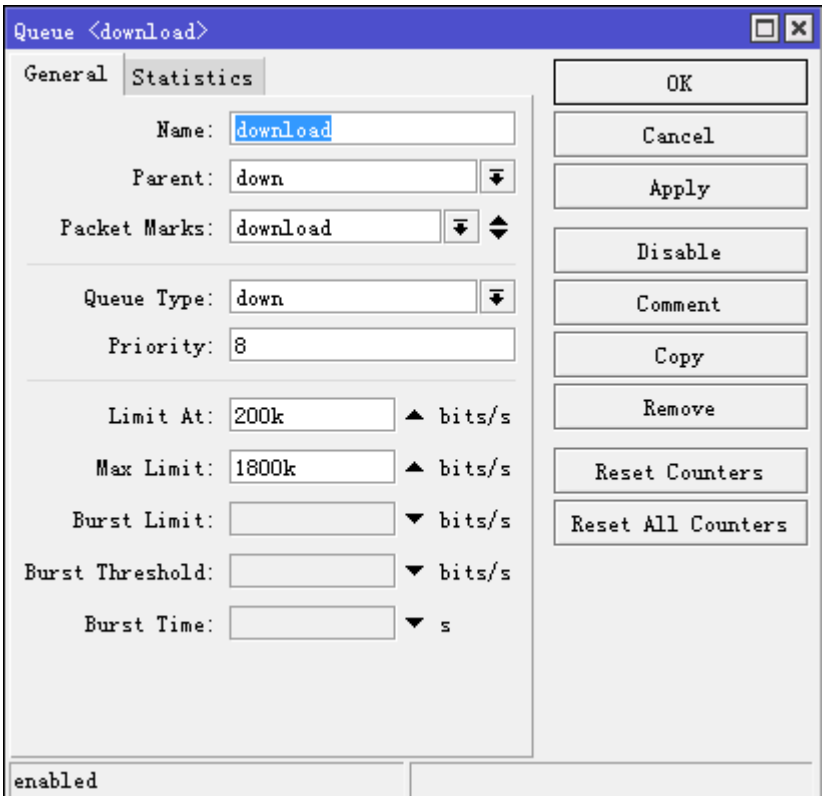
其次我们添加网页浏览的流控，使用同样的方法，priority 为 2，max-limit 为 1200k，保证最低获得 700k



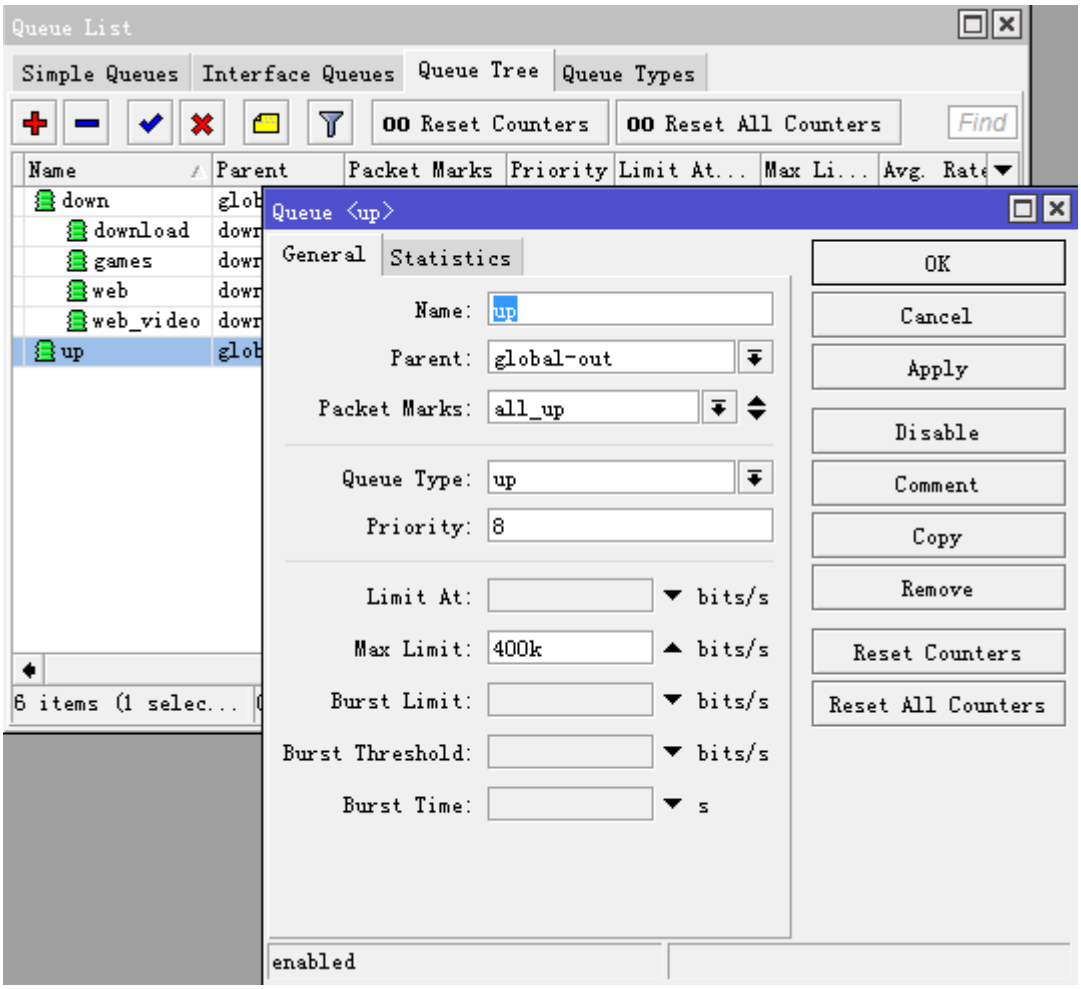
网页视频的优先级为 7，最大可以获得 1800k，最低保证 500k



其他下载数据优先级最低 8，最大带宽为 1800k，最低保证 200k



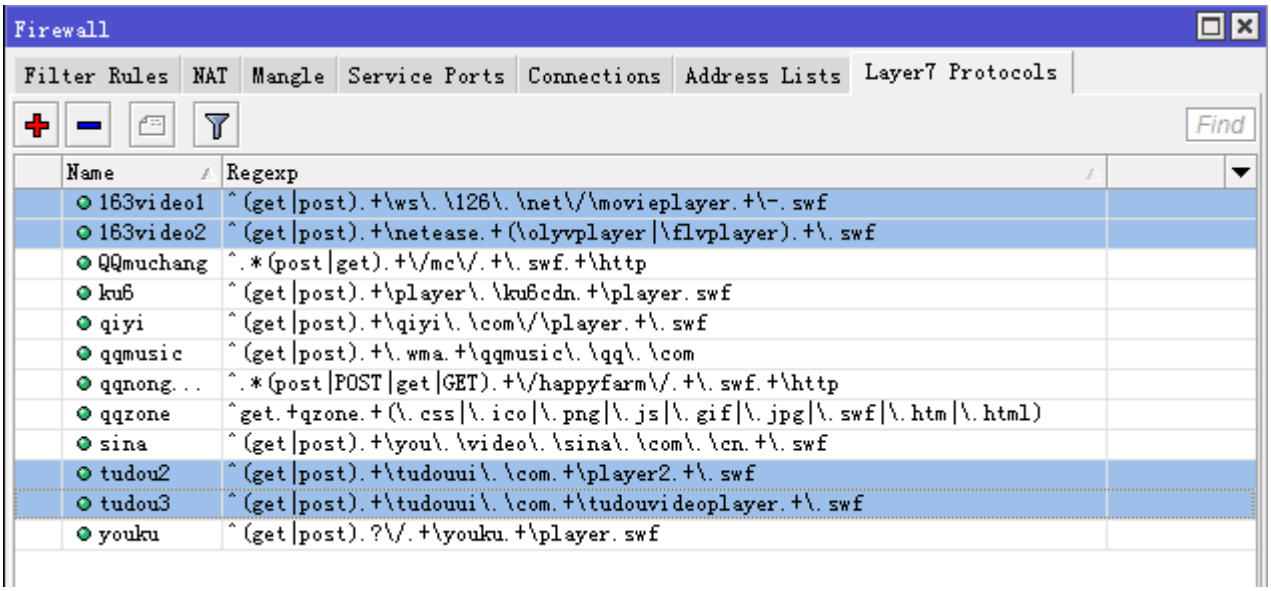
上行带宽的 PCQ 规则，取名为 up，因为上行是独立的 PCQ 规则，所以父级是 global-out，选择 packet-mark 为 all_up，queue-type 为刚才的 PCQ 规则 up，每台主机 200kbps 带宽，max-limit 为 400k（因为 2M 的 ADSL 上行带宽为 512k，保留一部分带宽）



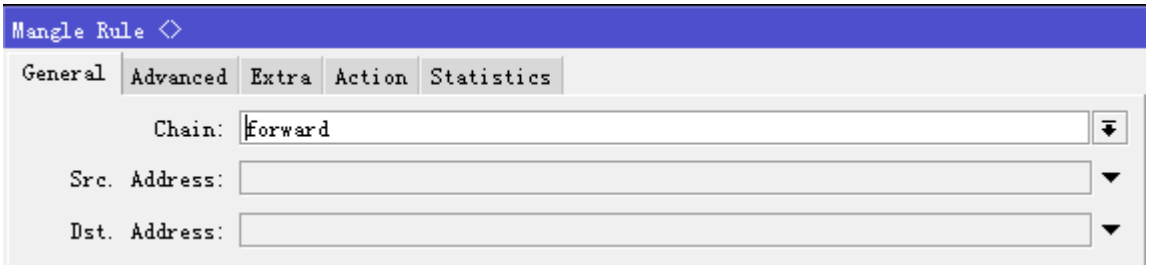
12.14 L7 流控

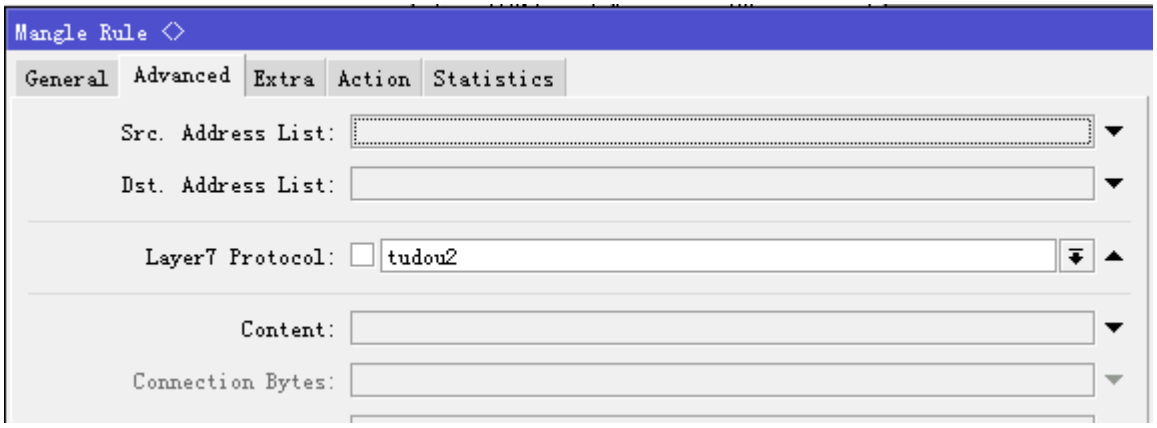
我们能通过 RouterOS 集成的 L7 协议识别一些网络软件，并对其进流量控制，例如我们对网页视频进行流控，通过 L7 协议抓取的网页视频比使用 Connection-rate 更精确，例如我们对网易和土豆网站的视频进行流控

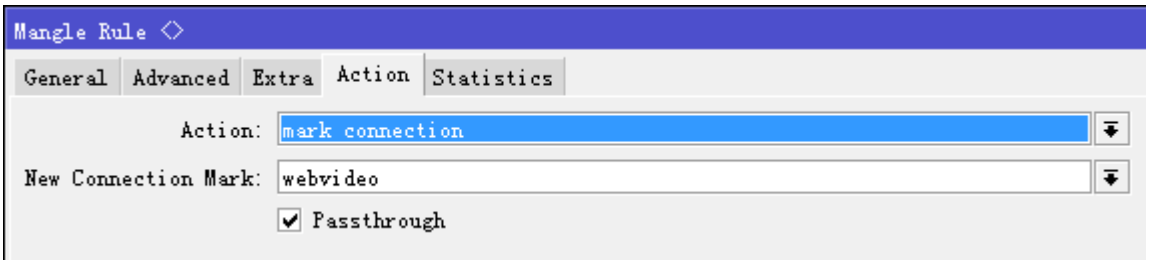
首先我们需要将网易和土豆的 L7 表达式写入/ip firewall layer7-protocol（具体的 L7 表达式，请参见防火墙章节的 L7 协议），如下图，我们选择 163 和 tudou 的视频 L7 规则（下面 L7 代码仅供参考）



进入 ip firewall mangle，我们首先标记 163 和 tudou 视频的连接，我们链表为 chain=forward 链表，添加 Layer7 protocol=tudou2，执行 action=mark-connection，并取名 new-connection-mark=webvideo，并设置 passthrough=yes，配置如下图：



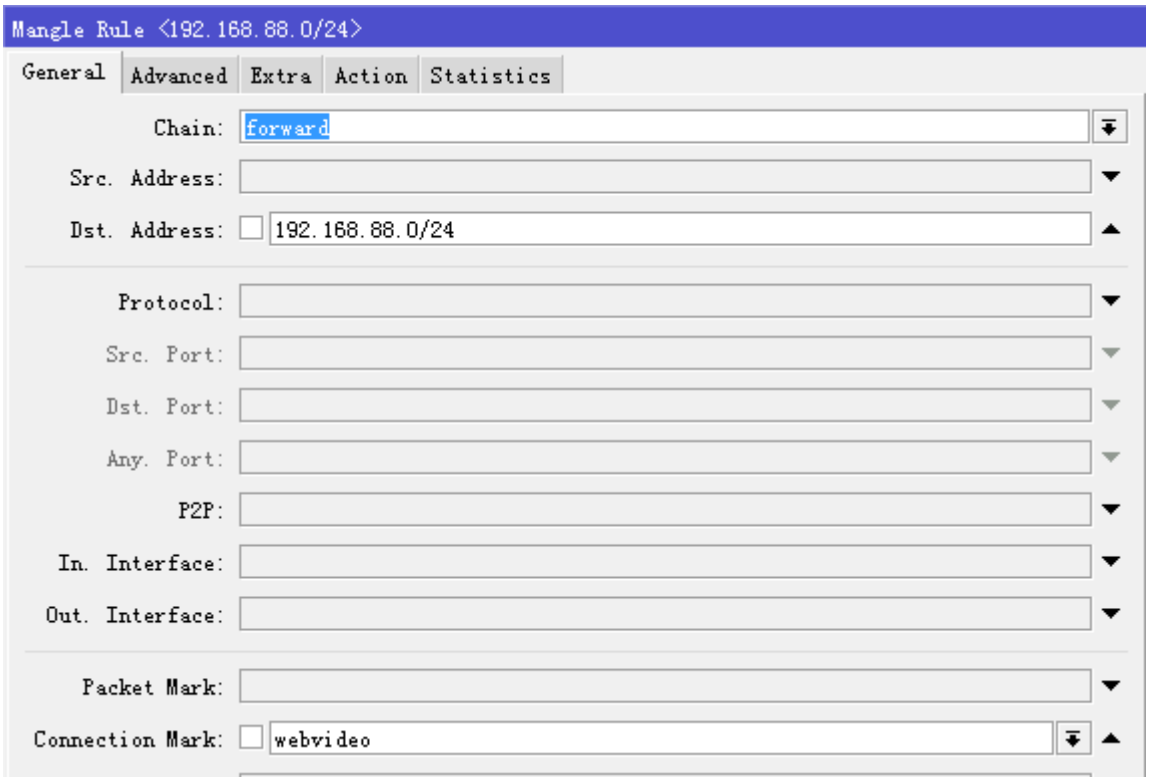


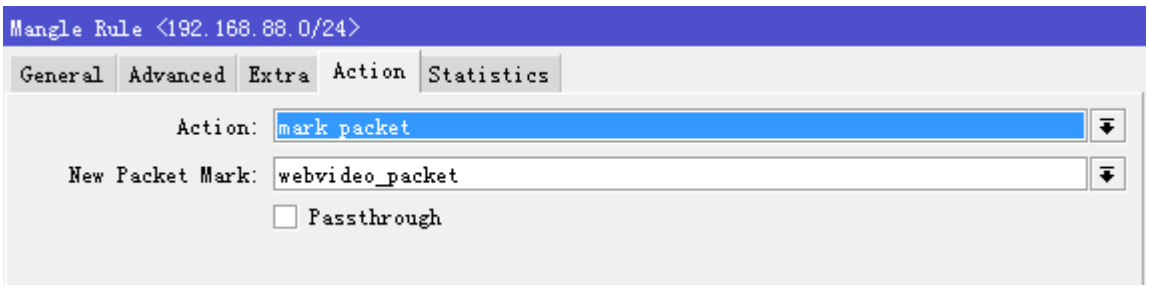


我们分别将其他几条，下面是连接标记的 mangle 脚本：

```
/ip firewall mangle
add action=mark-connection chain=forward disabled=no layer7-protocol=tudou2
new-connection-mark=tudou passthrough=yes

add action=mark-connection chain=forward disabled=no layer7-protocol=tudou3
new-connection-mark=tudou passthrough=yes

add action=mark-connection chain=forward disabled=no layer7-protocol=163video2
new-connection-mark=tudou passthrough=yes

add action=mark-connection chain=forward disabled=no layer7-protocol=163video1
new-connection-mark=tudou passthrough=yes
```

接下来我们将标记的这几条链表，汇总给一个数据标记，仍然选择的 forward 链表，提取 webvideo 的连接标记，我们需要指定数据报的方向，我的内网 IP 地址是 192.168.88.0/24，即发向目的地址是 192.168.88.0/24 地址段的是下载，然后我们选择 action=mark-packet，new-packet-mark=webvideo_packet，passthrough=no 停止向下传递：



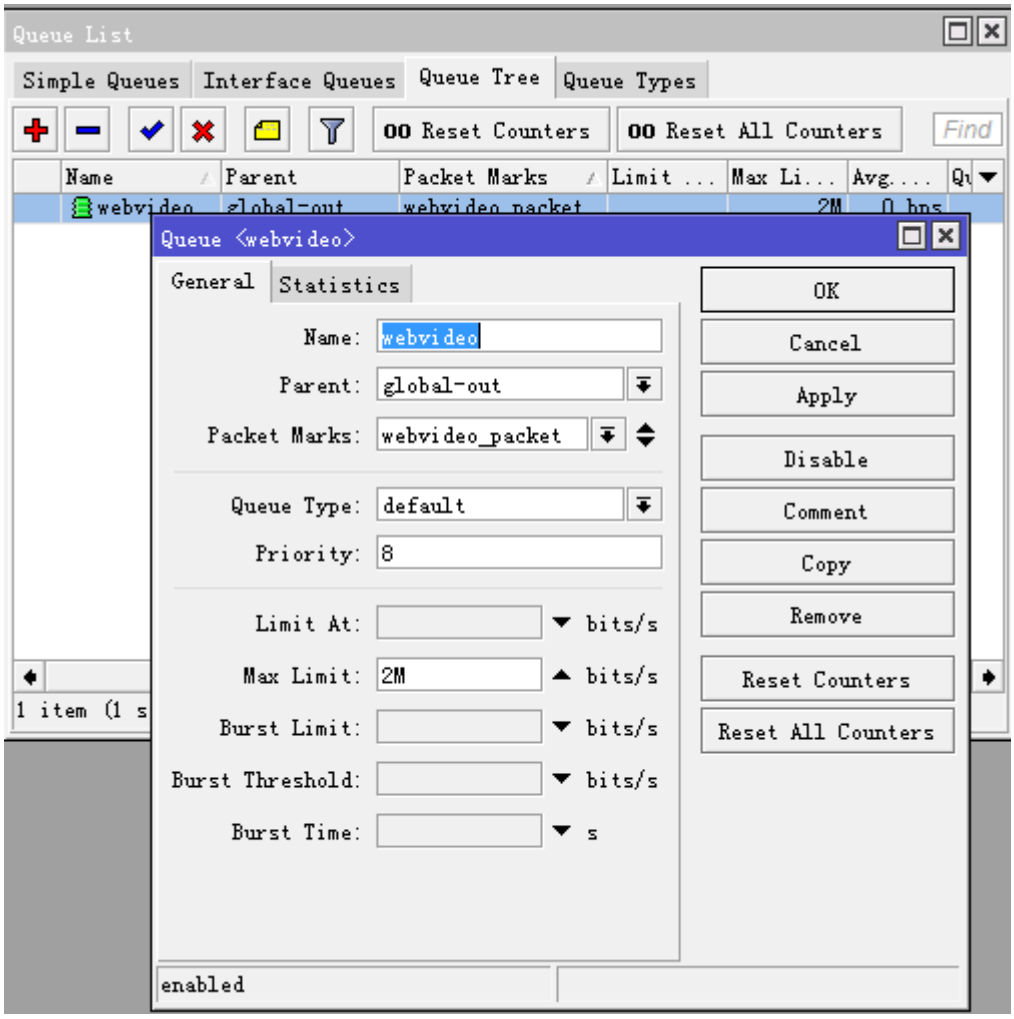


我们脚本如下：

```
/ip firewall mangle
add action=mark-packet chain=forward connection-mark=tudou disabled=no
dst-address=192.168.88.0/24 new-packet-mark=tudou passthrough=no
```

queue tree 设置

我们进入 tree 里，设置控制网页视频的带宽，我们给网页视频频宽为 2M，在这里我们使用的是 forward 链表，parent 选择 global-out，具体配置如下：



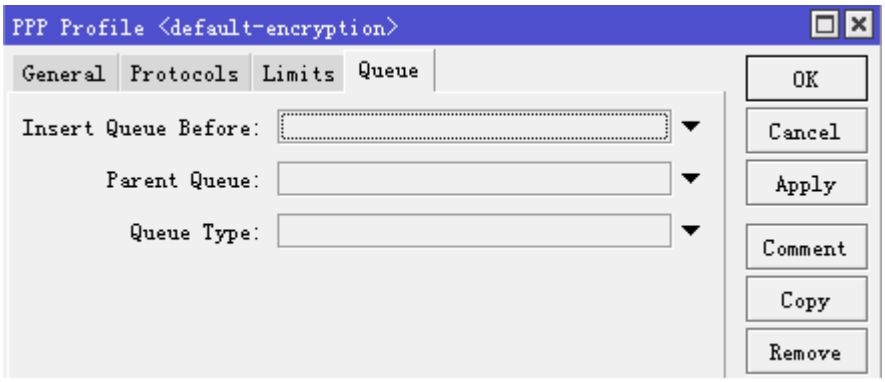
Queue tree 脚本如下

```
/queue tree
add max-limit=2M name=webvideo packet-mark=webvideo_packet parent=global-out
```

12.15 PPP Profile QoS

RouterOS v6 版本对 Queue 的改动较大，直接让 simple queue 在功能应用上与 queue tree 几乎相同，并且 simple queue 取消了 FIFO 的先进先出算法，直接采用等级优先的策略，取消 FIFO 算法有助于提高路由器对 simple 流控的处理性能。所以我们对于一条策略的上下结构不再关心，而指定策略优先通过 priority 属性。

在 PPP profile 的中增加了 queue 菜单，这样管理者直接可以定义 PPP 客户端在 simple 的流控属性。



这里有三个属性：

Insert Queue Before – 指定插入该 ppp profile 用户流控规则到 simple 的位置：

Bottom – 底部之前，即加入到 simple queue 中最后一条

First – 首部之前，即加入到 simple queue 中第一条

自选 – 选择插入当前 simple queue 中任意一条规则前

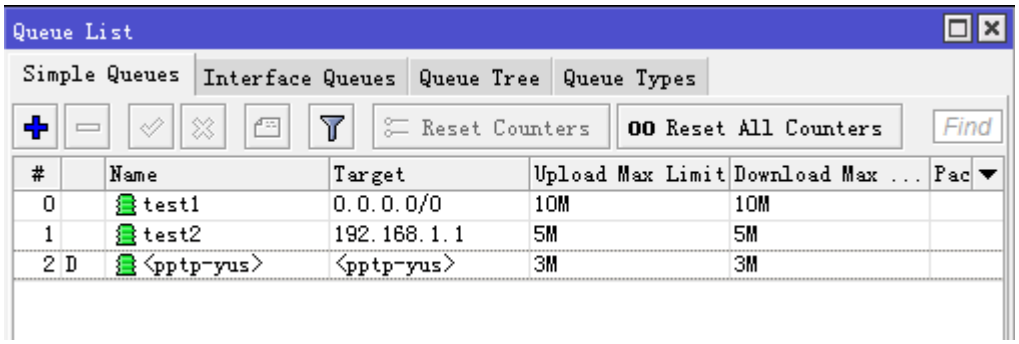
Parent Queue – 指定从属于父级的 simple queue 规则，有助于管理和优先级处理

Queue Type – 指定 queue 流控类型

其实大家如果对 queue tree 足够属性的话，这些功能一看就知道如何使用，但这里为大家更好的理解和操作，我举例介绍下：在这个实例下建立一个 PPTP server，设置账号 yus 登录，限制带宽为 3M

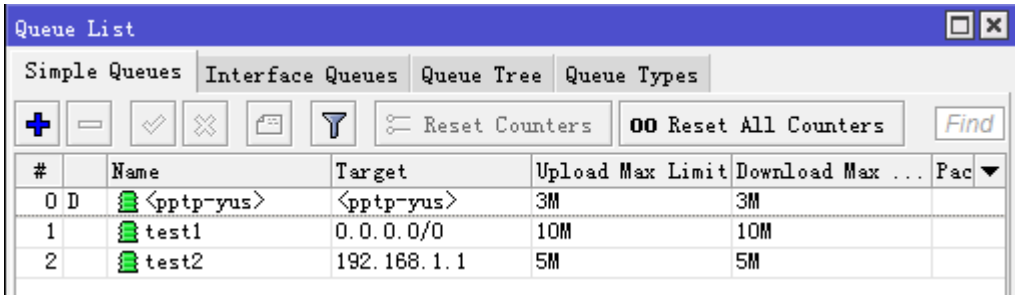
1、Bottom

当我将 Insert Queue Before 设置为 bottom 时，PPTP 拨号的流控规则会自动放到 simple queue 的最后，我们可以看到 pptp-yus 流控规则在 test2 之后



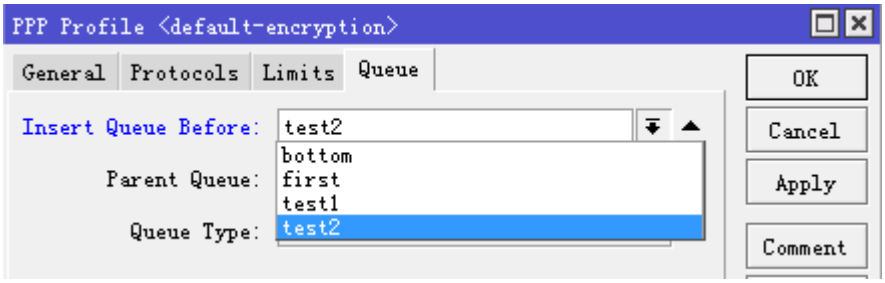
2、First

当我将 Insert Queue Before 设置为 first 时，我拨号的规则会自动放到 simple queue 的第一

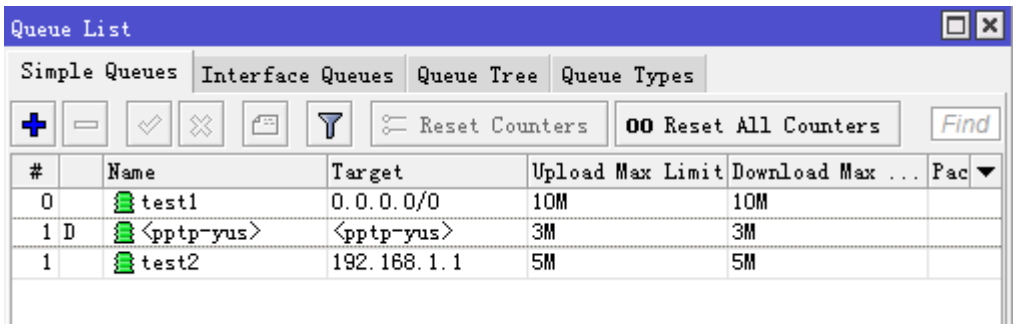


自定义：

自定义即，选择指定插入某条规则，如我插入 test2 之前



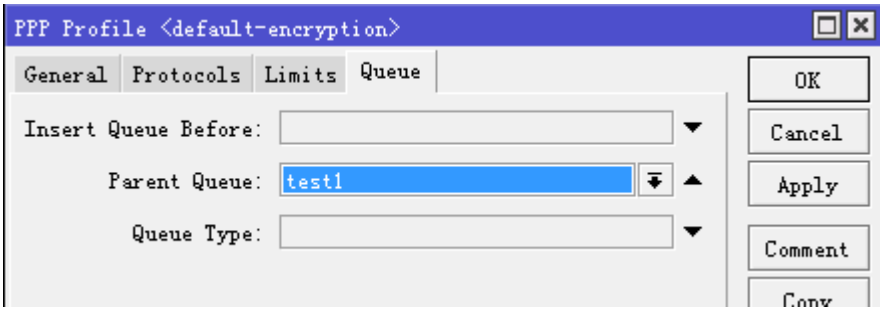
这样，pptp-yus 被放在 test2 之前



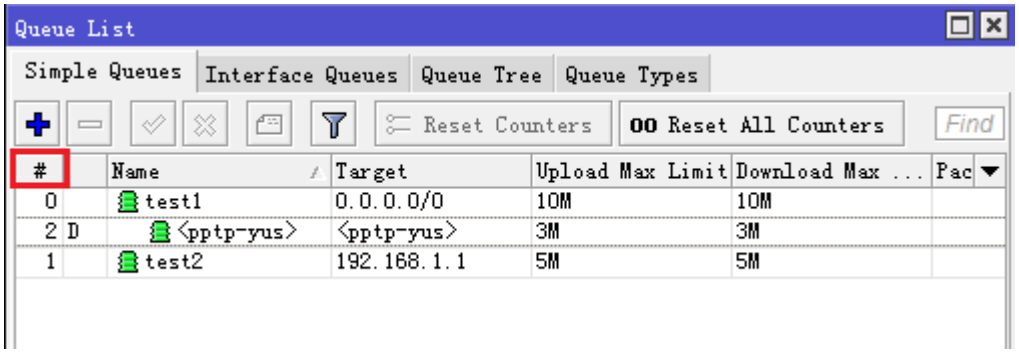
3、Parent 属性

Parent queue 属性是值得父级队列，这个在 HTB 里常用到，当 ppp 动态 queue 规则从属于某个父级规则下，将被定义为这个父级下 HTB QoS 流控规则，具体的 HTB 原理可以参见介绍，

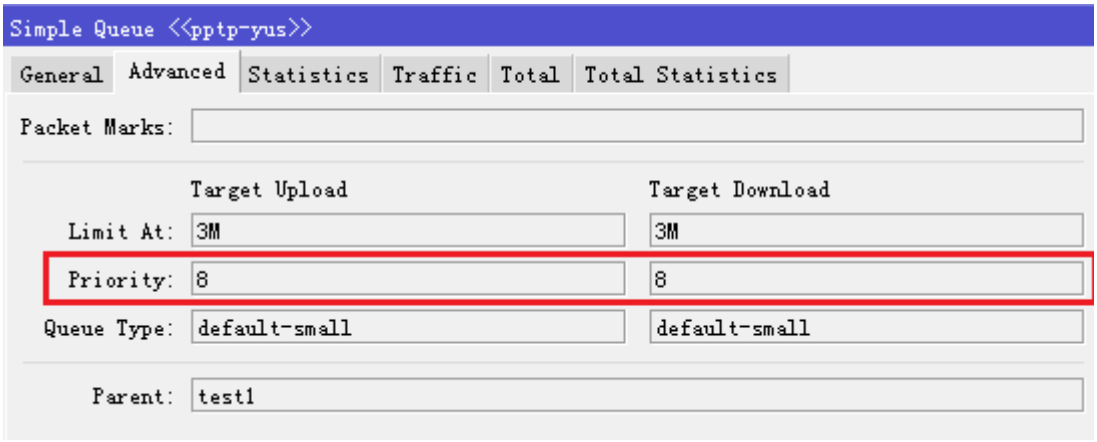
这里我们定义下面 PPP profile 的 Parent Queue 属性为 test1，即从属于 test1 规则下



Simple queue 正常情况下无法看到父子级结构，需要点击“#”查看，这里可以看到 pptp-yus 已经从属于 test1 下



当然默认的 Priority 为 8，即最低，至于在 test1 父级下的 HTB 如何设定，根据管理员需要来控制



第十三章 Mangle 分类标记

mangle 允许对 IP 数据报做特殊的标记，mangle 是通过修改指定的 IP 数据报头字段，去标记 IP 数据报的特征 能标记端口、IP、协议、TCP 协定和相应的 IP 数据流。 Mangle 属于综合性功能，所以在路由、流量控制和其他相应功能中都会涉及到，本章仅对 Mangle 做简单讲解，主要应用集中在路由和 QoS 章节。

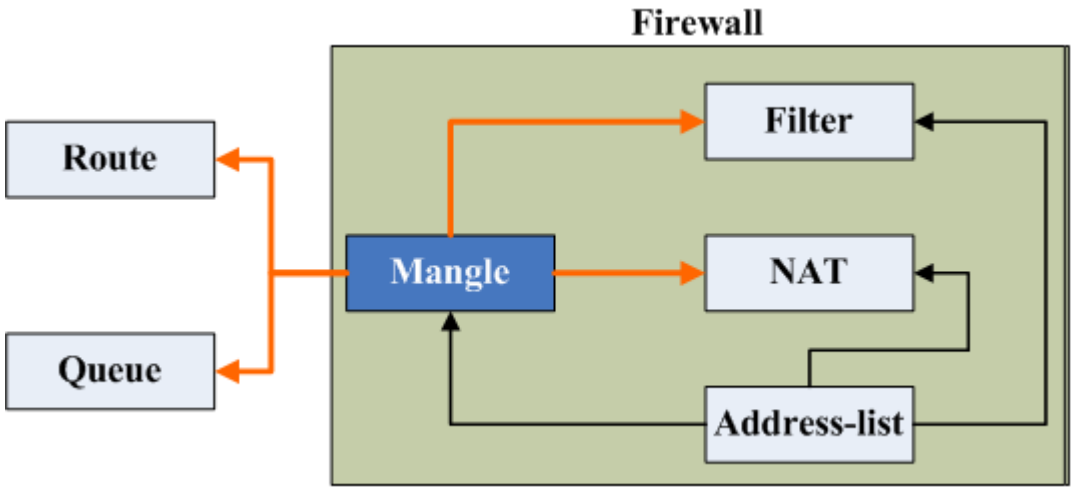
需要功能包: ***system***
需要等级: *Level1*
操作路径: ***/ip firewall mangle***
协议标准: IP

13.1 Mangle 介绍

Mangle 是一种标记器，标记特殊的数据包后待将来处理。在 RouterOS 中许多其他的功能组件会使用到他，如 queue（流控）、路由策略和 nat，他们识别到一个数据报了标记的便会做相应的处理。Mangle 标记仅存在于当前路由器中，他们无法传输到网络中去。根据数据传输方式不同可以选择：

- **Prerouting：路由前，常用于标记策略和端口路由**
- **Input：进入路由器的数据**
- **Foreward：通过路由转发，用于修改 TTL、TCP-MSS 和流量控制规则**
- **Output：数据输出**
- **Postrouting：路由后**

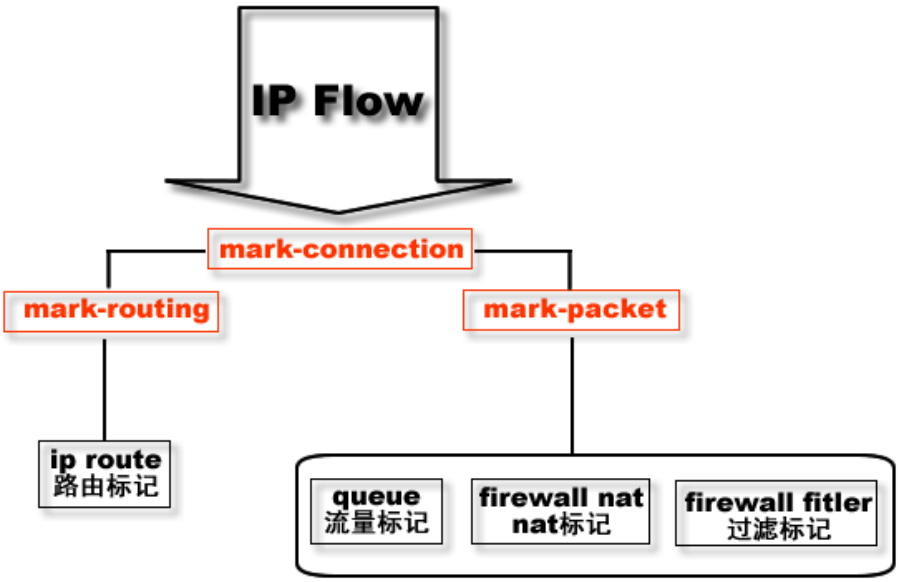
RouterOS 中的 IP firewall 主要由 3 个规则部分组成 Mangle、Filter、NAT ，而 Address-list 常用于地址列表分类。Mangle 通过标记特定的 IP 数据流后，为 Filter、NAT 和、路由、Queue 提供标记后的 IP 数据流



标记 IP 数据流的三种类型，这三种类型会在各种应用中多次出现，特别是 Queue 的流量控制和 ip route 的路由：

- **Mark-connection：标记所有 IP 流的连接**
- **Mark-packet：标记 IP 流中数据报**
- **Mark-routing：标记 IP 流中 IP 数据报的路由信息**

三种类型的关系，所有的在 IP 数据报传输前，首先需要通过建立 TCP/UDP 连接，进行传输。所以当数据通过 IP 流进入 Mangle 后，建立相应的连接标记，并从连接标记中提取数据报，做处理。图示如下：



13.2 Mangle 应用

Peer-to-Peer 传输标记

保证优质的网络连接，如 VoIP 和 HTTP 等为最优先级，将 P2P 的优先级设置为最低 RouterOS QOS 操作首先使用 mangle 标记不同类型的传输，然后把它们放入的 queues 做不同的限制。下面的事例是强迫 P2P 的总的传输不能超过 1Mbps，其他的传输连接则扩大连接带宽和优先级：

```
[admin@MikroTik] > /ip firewall mangle add chain=forward  p2p=all-p2p
action=mark-connection new-connection-mark=p2p_conn
[admin@MikroTik] > /ip firewall mangle add chain=forward connection-mark=p2p_conn
action=mark-packet new-packet-mark=p2p
[admin@MikroTik] > /ip firewall mangle add chain=forward  packet-mark=!p2p_conn
action=mark-packet new-packet-mark=other
[admin@MikroTik] > /ip firewall mangle print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=forward p2p=all-p2p action=mark-connection new-connection-mark=p2p_conn

 1   chain=forward connection-mark=p2p_conn action=mark-packet new-packet-mark=p2p

 2   chain=forward packet-mark=!p2p_conn action=mark-packet new-packet-mark=other
[admin@MikroTik] >
[admin@MikroTik] > /queue tree add parent=Public packet-mark=p2p limit-at=1000000
max-limit=100000000 priority=8
[admin@MikroTik] > /queue tree add parent=Local packet-mark=p2p limit-at=1000000
max-limit=100000000 priority=8
[admin@MikroTik] > /queue tree add parent=Public packet-mark=other  limit-at=1000000
max-limit=100000000 priority=1
```

```
[admin@MikroTik] > /queue tree add parent=Local packet-mark=other  limit-at=1000000
max-limit=100000000 priority=1
```

Mangle 限制 2 级代理

通过 mangle 限制 2 级代理，思路是修改 TTL 值，让路由级数减少，但对端口的 http 代理无效，进入 forward 链表指定 in-interface 或者指定目标数据到内往的 IP 地址，即 dst-address 或 dst-address-list 等参数来修改到达目标的 TTL 值为 1

```
[admin@MikroTik] /ip firewall mangle> add chain=forward out-interface=lan action
=change-ttl new-ttl=set:1
[admin@MikroTik] /ip firewall mangle>print chain=forward
Flags: X - disabled, I - invalid, D - dynamic
8   chain=forward action=change-ttl new-ttl=set:1 out-interface=lan
```

13.3 Nth 的 mangle 应用

Nth 是 RouterOS 曾经路由负载均衡的一个重要的工具，可以实现基于 IP 的多线路负载均衡，还能实现对端口负载均衡和 nat 的有序指定访问，是 RouterOS v3.0 解决路由路由负载均衡的一个方式，但存在很多问题。直到 v3.24 的 PCC 出现。

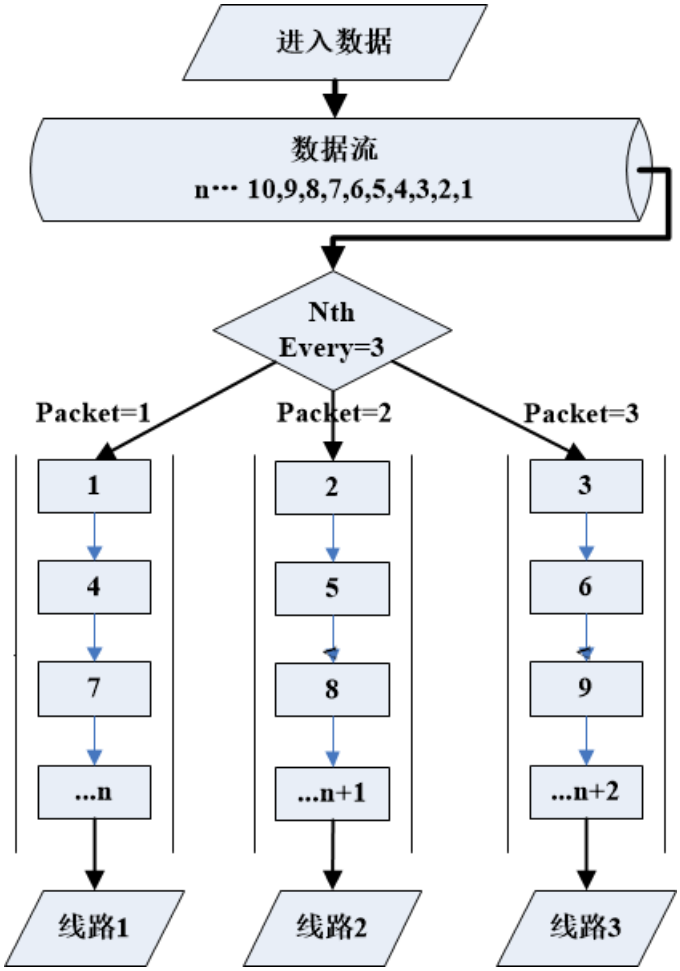
Nth 原理介绍

在 v3.0 后 Nth 功能做了一点修改，仅只有两个参数“every”和“packet”。Nth 两个参数“every”和“packet”，Nth 通过计数器方式处理。当规则收到数据包，该规则的计数器会增加 1，如果计数器匹配值“every”与 packet 值相等，计数器将重新设置为 0，重新计数。 使用 Nth 将连续的数据包通过计数器分组，比如可以将连续的数据包分配为多个组，重新排列连接序列。

nth 匹配特定的第 N 次收到的数据报的规则，一个计数器最多可以计数 16 个数据包，包含两个属性：

Every – 匹配每 every 数据报，同时指定 **Counter**（计数器值）

Packet – 匹配给定的包，如：Nth=3,1（every=3，packet=1），每 3 个数据包，取第 1 个

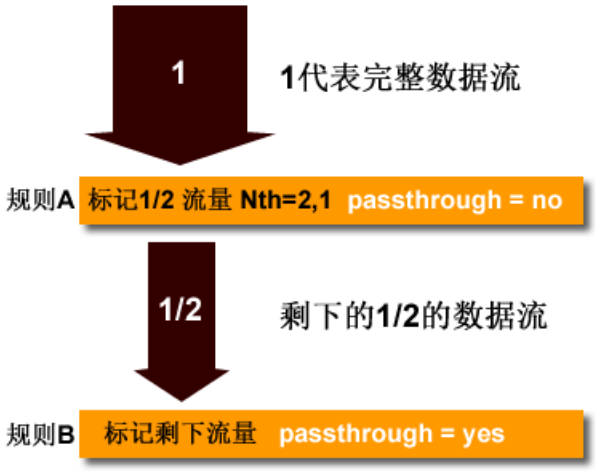


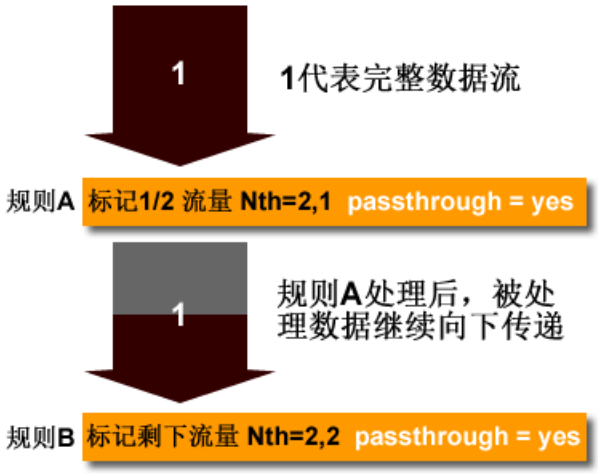
上图，可以看到数据流被 Nth 分为 3 个计数器，并根据 Packet 重新排列数据流的队列。建立一个数组形式的容器，按照数组序列来分发数据包，计数器满 3 后，将重新设置为 0，重新计数。按照要去走到指定 3 条外网线路出口。

Nth 没有采用哈希算法，仅仅是按照序列分组排队，会出现源目标 IP 相同会话的数据包无法走同一条线路，nat 后多线的出口 IP 是不一样的，这样会出现要求对源 IP 校验的站点无法访问的情况，例如，淘宝验证，银行网银验证等等，线路越多出错的概率越大，因为序列分组后，指定的 IP 数据包走之前线路的几率越小，当时通过指定 443 端口走一条固定线路来解决，那时 https 网站还不流行，勉强能解决。

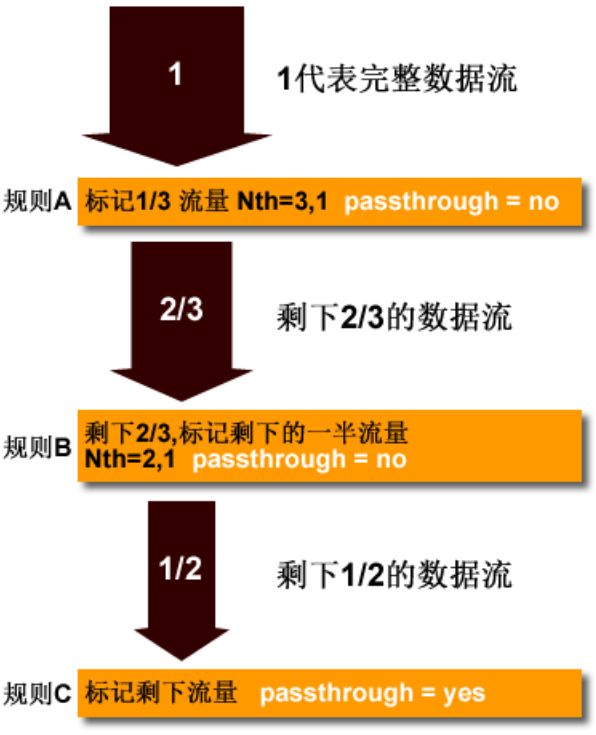
Passthrough 对 Nth 的控制

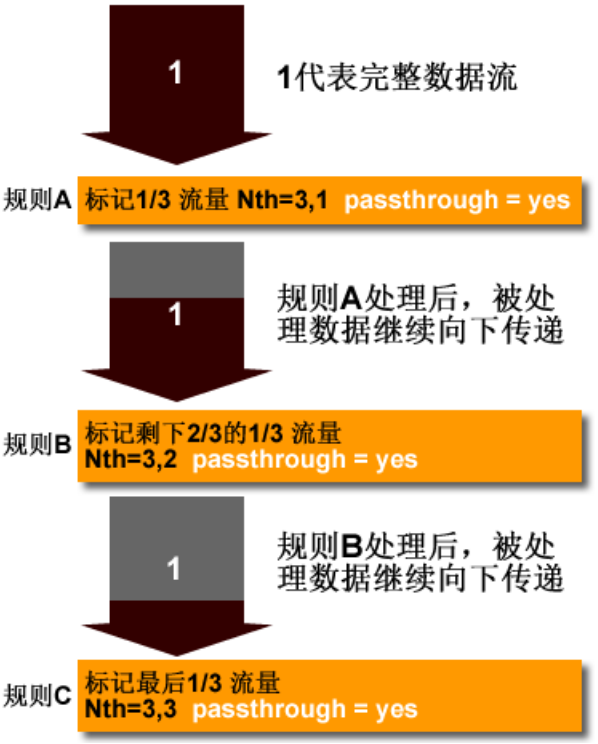
实现相同的 Nth 结果时，改变 Passthrough 参数（Passthrough 为是否将该规则数据继续向下传递，**no** 为停止向下传递，**yes** 则相反，具体参考 Mangle 章节）会得到不同的规则配置，首先要知道 Mangle 标记捕获数据是先进先出算法，即从上往下执行，我们在配置 Mangle 的 Nth 规则，需要注意前后顺序。如我们把数据流标记为两个组，即一条为 1/2，另一条也为 1/2，把一个数据流看成“1”，而我们把可以通过两种方法配置：





当我们需要将数据流标记为 3 组时，即每条规则为 1/3。配置方法同样有两种，如下图





如从上面的图上看到，使用和不使用 Passthrough 的区别，在于流量是否继续向下传递。

例如，有双线接入，并采用 Nth 的双线负载均衡。首先我们需要在 mangle 里标记连接，如果配置 Passthrough=no 参数，Nth 参数配置仅需要一条规则，即标记置 50%流量，首先我们需要标记连接：

```
/ip firewall mangle
add chain=prerouting new-connection-mark=AAA nth=2,1 action=mark-connection
passthrough=no;
```

抓取完前 50%的数据后，剩下的流量只需要做一个默认的标记剩下的数据即可。

```
add chain=prerouting new-connection-mark=BBB action=mark-connection
```

当变成 3 条线路时，第一条规则标记所有数据报并对比所有流量的 1/3，第二条规则标记剩下 2/3 数据报的 50%，第三条规则标记和对比所有剩下的数据报（所有数据报的 1/3）

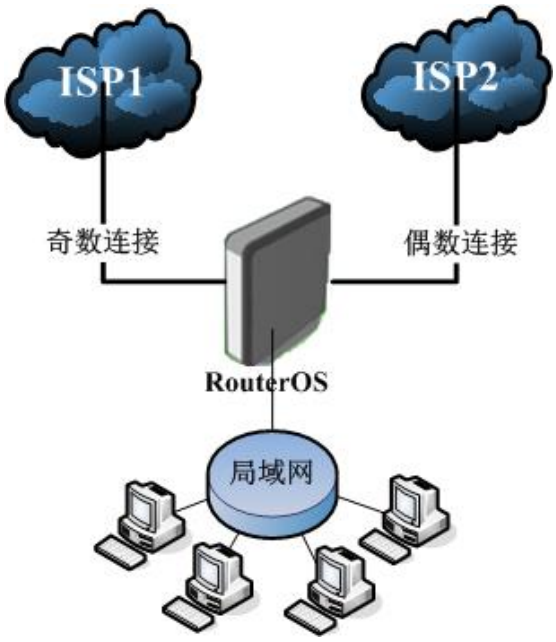
```
/ip firewall mangle
add action=mark-connection chain=prerouting new-connection-mark=AAA nth=3,1
passthrough=no;
add action=mark-connection chain=prerouting new-connection-mark=BBB nth=2,1
passthrough=no;
add action=mark-connection chain=prerouting new-connection-mark=CCC ;
```

同样我们有的数据报并且每个规则对比每 3 个数据报。

```
/ip firewall mangle
add action=mark-connection chain=prerouting new-connection-mark=AAA nth=3,1
passthrough=yes;
add action=mark-connection chain=prerouting new-connection-mark=BBB nth=3,2
passthrough=yes;
add action=mark-connection chain=prerouting new-connection-mark=CCC nth=3,3
passthrough=yes;
```

Nth 在负载均衡的应用

下面我看一个实际的双线接入的 Nth 应用事例，假设我们有两条 ISP 的线路，我们通过 Nth 的方法实现负载均衡，让 2 条同样 ISP 线路达到合并带宽的作用。



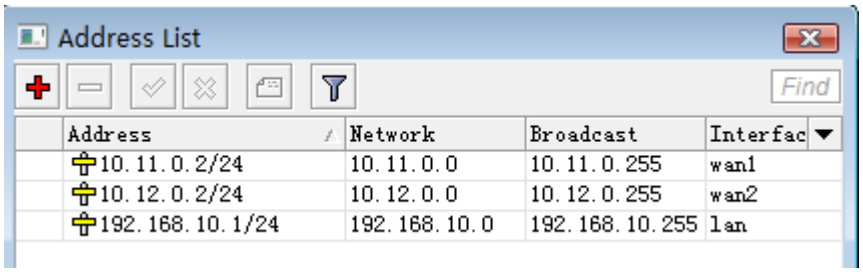
根据 Nth 的原理我们可以将来至内网的联接分为两组，即一组为奇数连接、一组为偶数连接，即奇数走一条线路，偶数走另外一条线路。因为我们定义的是连接状态为 new，即新建立的连接，对正常的访问没有任何影响，每个新建立所产生的后续数据都会按照原来的线路连接运行。

我们从所有的连接中，提取每次新建立的连接 connection=new，并对他们做 Nth 的标记，将这些连接中相关的奇数（odd）包和偶数（even）包分离开，并走两个不同的网关（ISP1 与 ISP2）出去。这样就能保持每次连接的持续性。

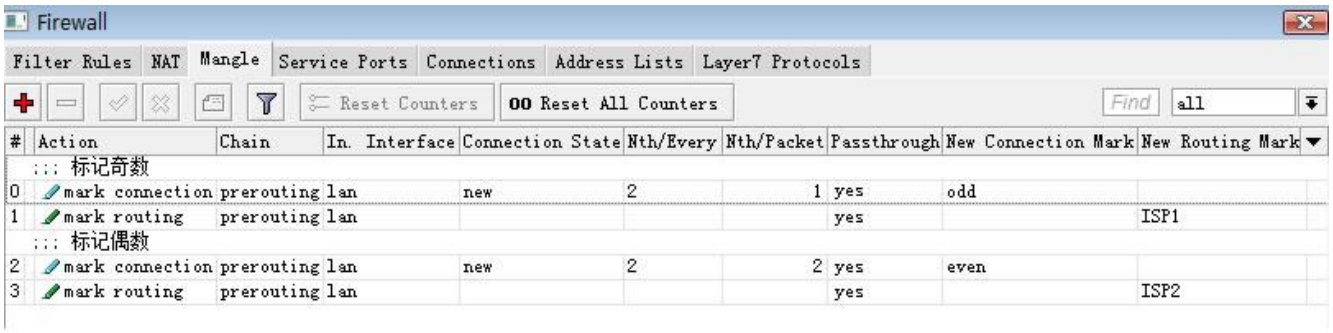
网络参数如下：

- wan1: ip 地址 10.11.0.2/24，网关 10.11.0.1
- wan2：ip 地址 10.12.0.2/24，网关 10.12.0.1
- lan：192.168.10.1/24

首先配置 IP



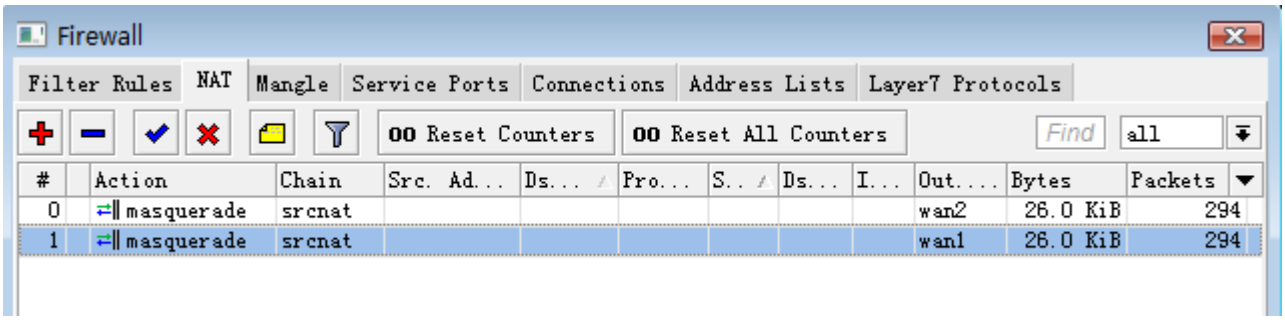
接下来在 ip firewall mangle 中标记奇数和偶数的 Nth，并配置路由标记，奇数 Nth 连接标记取名为 odd，偶数连接标记取名为 even，将奇数的路由标记取名为 ISP1，将偶数的路由标记取名为 ISP2，如下：



命令行配置如下：

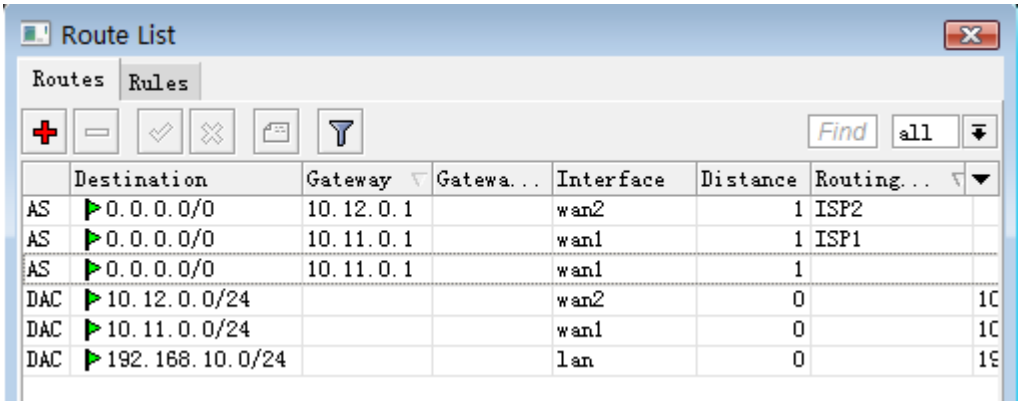
```
[admin@MikroTik] /ip firewall mangle> print
Flags: X - disabled, I - invalid, D - dynamic
 0 chain=prerouting action=mark-connection new-connection-mark=odd passthrough=yes
connection-state=new in-interface=lan nth=2,1

 1   chain=prerouting action=mark-routing new-routing-mark=ISP1 passthrough=yes
in-interface=lan connection-mark=odd

 2 chain=prerouting action=mark-connection new-connection-mark=even passthrough=yes
connection-state=new in-interface=lan nth=2,2

 3   chain=prerouting action=mark-routing new-routing-mark=ISP2 passthrough=yes
in-interface=lan  connection-mark=even
```

NAT 配置



路由配置

进入 ip route 中配置路由规则，配置 10.12.0.1 对应 ISP2 的路由标记，10.11.0.1 对应 ISP1 的路由标记，我们用 10.11.0.1 作为路由器本身的默认网关。



命令行配置如下

```
/ ip route
add gateway=10.11.0.1 routing-mark=ISP1
add gateway=10.12.0.1 routing-mark=ISP2
```

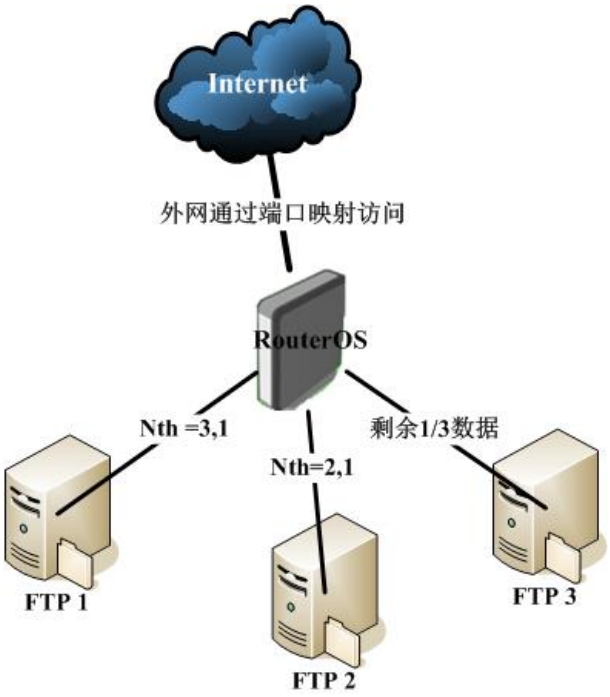
这样双线的 Nth 负载均衡就配置完成，建议这样的负载均衡使用在相同 ISP 的线路上，并且带宽接近。

注：在 Nth 时我们需要将 TCP 443 和 8443 端口指定到固定的一条线路，避免一些要求固定 IP 验证的网站，如网上银行等。

Nth 在最多支持 16 个计数器，如果我们允许接入 12 条相同带宽的线路，并采用 **passthrough=yes**，即 Nth 规则应是[every,packet]=[12,1]，[12,2]，[12,3]，[12,4]，[12,5]，[12,6]，[12,7]，[12,8]，[12,9]，[12,10]，[12,11]，[12,12]

Nth 在端口映射的应用

通过 Nth 的原理我可以实现一些特定的应用，比如应用于 FTP 服务的端口映射，当我们有大量信息需要向互联网共享时，可能我们一台 FTP 服务器无法承担所有的数据流量，我们可以通过建立多台服务器来分担流量，在不必修改 FTP 端口的情况下，通过 Nth 均衡分流数据到 3 台 ftp 服务器上，如下图内网的 3 个 FTP 服务器：



我们通过建立 3 条 nat 规则，区别 3 个不同的服务器连接，在 nat 中没有同时做 Passthrough 的选项，而且在 nat 规则中采用的是先进先出算法，所以我们只能采用先标记 1/3，在标记 1/2，最后标记剩下的数据的方法处理 3 条线路的均衡操作。

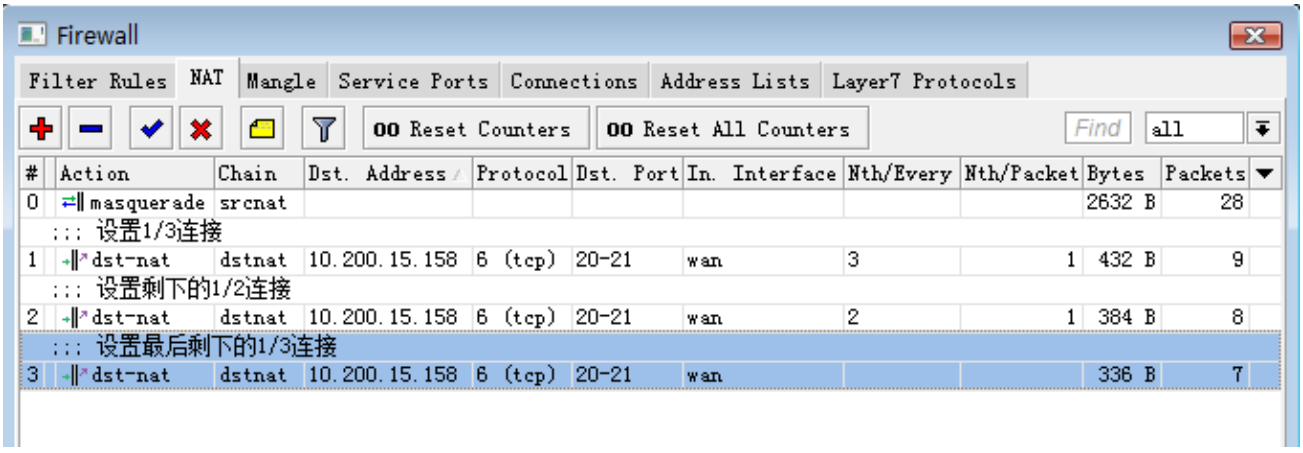
我们的网络环境如下

- Wan：ip 地址为 10.200.15.158/24 网关为 10.200.15.1
- Lan：ip 地址为 192.168.10.1/24
- 内网的 3 个 FTP 服务器的 IP 地址分别是 192.168.10.2，192.168.10.3，192.168.10.4

在配置完 IP 地址后，我们进入 ip firewall nat 配置 nat 规则， 首先我们需要配置基本的 nat 伪装规则，将内网的私有 IP 地址转换为公网 IP。

```
/ip firewall nat
add action=masquerade chain=srcnat disabled=no out-interface=wan
```

接着，设置端口映射，FTP 使用的是 TCP，20-21 端口，我们配置 3 条 nat 的 Nth 端口映射的规则，分别指向 192.168.10.2、192.168.10.3 和 192.168.10.4 三个服务器的 IP 地址：



通过命令行配置如下：标记前 1/3 的端口映射

```
add action=dst-nat chain=dstnat dst-address=10.200.15.158 dst-port=20-21
in-interface=wan nth=3,1 protocol=tcp to-addresses=192.168.10.2 to-ports=20-21
```

标记剩下 1/2 的端口映射

```
add action=dst-nat chain=dstnat dst-address=10.200.15.158 dst-port=20-21
in-interface=wan nth=2,1 protocol=tcp to-addresses=192.168.10.3 to-ports=20-21
```

标记最后 1/3 的端口映射

```
add action=dst-nat chain=dstnat dst-address=10.200.15.158 dst-port=20-21
in-interface=wan protocol=tcp to-addresses=192.168.10.4 to-ports=20-21
```

这样通过 Nth 分流的端口映射配置完成，这样的 Nth 操作仅适合于一次性提交和访问的数据连接。如果是带登陆验证的访问，不建议使用这种方式，会出现连接后在不同服务器上的重复认证。

第十四章 Bridge 网桥（二层）

网桥（Bridge），是工作在 OSI 网络模型的第二层数据链路层，二层设备间通过 MAC 地址进行通信，对网络数据帧进行转发的过程。简单的说就是实现二层交换机功能，把两个不同的物理局域网连接起来，通过 MAC 地址通信，进行学习、存储和转发的设备。

RouterOS 支持以太网 MAC 层桥接，如以太网卡、EoIP（Ethernet over IP）、WLAN 协议和 PPP 协议等，还支持桥接的防火墙过滤功能，能有效对二层网络 MAC 和协议进行控制管理。

交换机原理与 RouterOS Bridge 二层交换机配置视频

Bridge 主要特征：

- 生成树协议(STP)
- 快速生成树协议（RSTP）
- 支持多网桥接口
- MAC 地址可以被实时监控
- 为三层访问分配 IP 地址
- 支持基于桥数据报过滤器

功能包: ***system***
等级: ***Level3***
操作路径: ***/interface bridge***

在 RouterOS 的 bridge 能实现以太网二层交换的功能外，还能实现 WLAN 的桥接和 WDS 的 MESH 组网等，还能实现虚拟接口类以太网的交换网络，如 EoIP（Ethernet over IP）、PPP tunnel bridging protocol (BCP)

网桥在功能上和交换机同样的网络设备，除了连接两个二层网络基本功能外，RouterOS 网桥还具备防火墙过滤和（M/R）STP（Spanning Tree Protocol）生成树协议，逻辑上断开环路，防止二层网络的广播风暴的产生。在复杂的网络拓扑出现（有意或无意）。如果没有特殊的处理，环路将造成网络无法正常工作，因为环路会导致雪崩一样的广播数据报倍增。生成树协议不仅能避免环路造成的广播风暴导致网络无法正常工作，还可以建立一个冗余的二层环网络，实现设备的冗余保护。

注：关于无线的 RSTP MESH 请参阅《RouterOS 脚本编写与 WLAN 无线教程》

14.1 Bridge 配置

操作路径: ***/interface bridge***

通过将多个局域网络连接到一个网桥上，即交换机的功能，实现学习、存储和转发，在 host 列表中可以看到各个接口学习到的 MAC 地址，并存储在 bridge 中用于查找需要通信的 MAC 地址。

属性描述

ageing-time (*时间*; 默认: **5m**) - 一个主机信息可以被保存在桥数据库的时间
arp (disabled | enabled | proxy-arp | reply-only; 默认: **enabled**) - 地址解析协议设置

forward-delay (*时间*; 默认: **15s**) - 在桥接口初始化阶段（例如：在路由器启动或起用接口之后）桥正常工作之前监听/学习状态所用的时间
garbage-collection-interval (*时间*; 默认: **4s**) - 丢弃桥数据库中老的（过期的）主机词条的频率。无用存储单元收集过程消除比 **ageing-time** 属性定义的更老的词条。
hello-time (*时间*; 默认: **2s**) - 给其他桥发送 hello 包的频率
mac-address (*只读: MAC 地址*) – 接口的 MAC 地址
max-message-age (*时间*; 默认: **20s**) - 保留从其他桥接受 hello 信息的时间长短
mtu (整型; 默认: **1500**) - 最大传输单元
name (名称; 默认: **bridgeN**) - 桥接口的描述性名称
priority (整型: 0..65535; 默认: **32768**) - 桥接口优先级。STP 使用优先级参数决定如果最后两个端口形成了环路应保留哪个
stp (no | yes; 默认： **no**) - 是否启用生成树协议。桥环路仅在这个属性启用是才会被阻止。
rstp (no | yes; 默认： **no**) - 是否启用快速生成树协议。

快速配置

下面的简单实例，是在一台 RouterOS 上，把三个独立的网口，设置为 3 口交换机，我们把三个网卡 ether1、ether2 和 ether3，配置为 3 个交换端口，首先要创建一个 bridge

1. 添加一个桥接口，命名为 **Bridge1**：

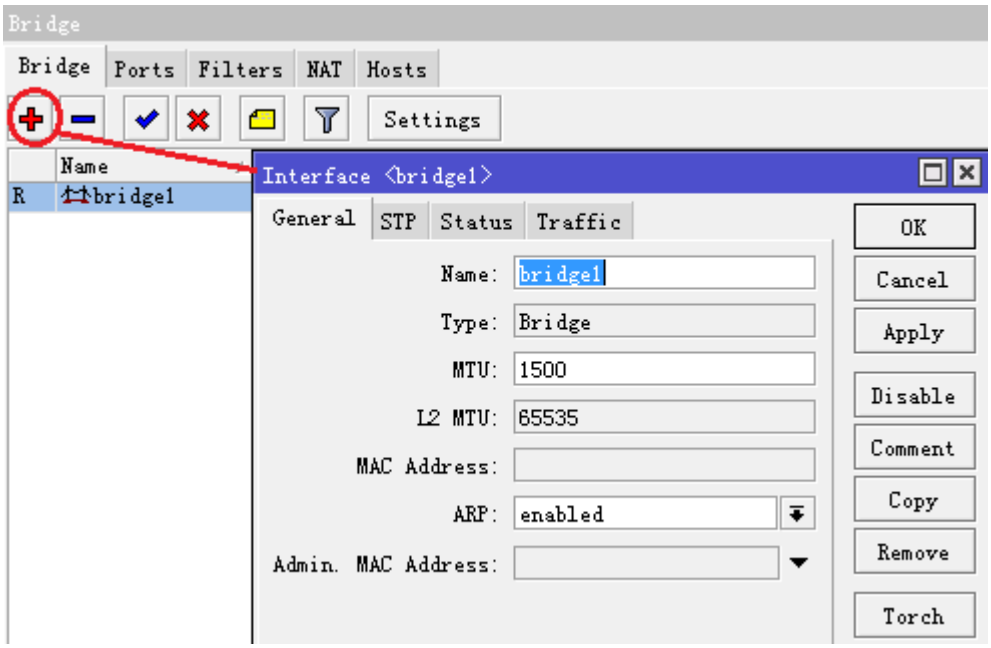
```
/interface bridge add name="Bridge1"   protocol-mode=none disabled=no
```

2. 把 **ether1、ether2 和 ether3** 添加到 **Bridge1** 接口：

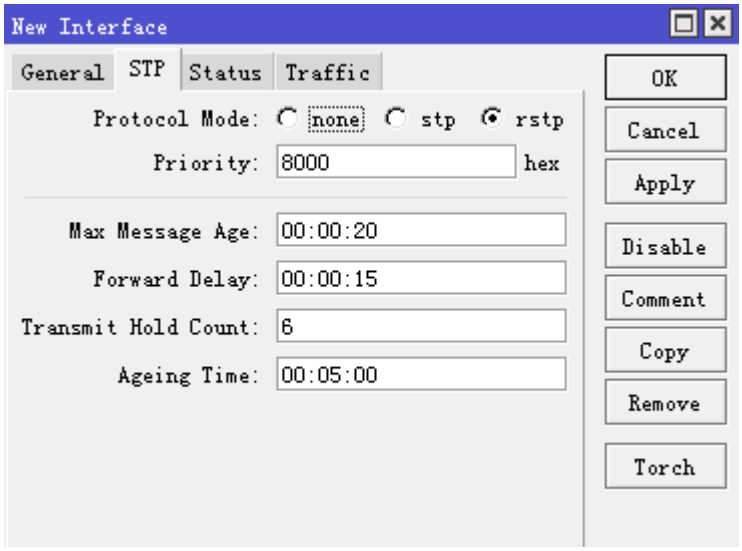
```
/interface bridge port add interface=ether1 bridge=Bridge1
/interface bridge port add interface=ether2 bridge=Bridge1
/interface bridge port add interface=ether3 bridge=Bridge1
```

整个配置完成，相当于创建了一个网桥 1 的分组取名为 Bridge1，将 ether1、ether2 和 ether3 加入到这个网桥组里。

用 winbox 进入 bridge 菜单，添加并启用一个网桥：



当创建一个 bridge 时，在菜单中可以选择 STP 配置，根据网络需要是否选择启用 STP 或 RSTP，初次配置建议关闭 stp 协议，设置为 none：



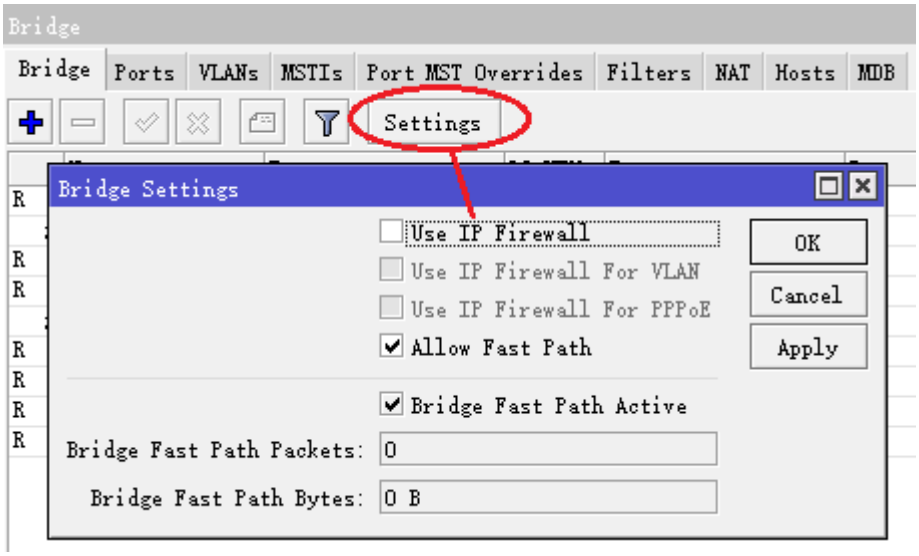
Bridge setting

操作路径: **/interface bridge settings**

属性	描述
allow-fast-path (*yes* | *no*; 默认:**yes**)	允许开启 fast path（参考第十一章节）
use-ip-firewall (*yes* | *no*; 默认:**no**)	强制将二层桥接流量发送到三层 prerouting, forward 和 postrouting 链表处理，即启用三层过滤，但无法应用到路由传输处理。具体参考

	(http://wiki.mikrotik.com/wiki/Manual:Packet_Flow_v6).
use-ip-firewall-for-pppoe (*yes* | *no*; 默认:**no**)	将二层桥接未加密 PPPoE 流量发送给三层"IP firewall"处理，该参数工作前提是 use-ip-firewall=yes
use-ip-firewall-for-vlan(*yes* | *no*; 默认: **no**)	将二层桥接 VLAN 流量发送给三层"IP firewall"处理，该参数工作前提是 use-ip-firewall=yes

use-ip-firewall 参数选择是否启用 ip firewall 的三层过滤规则，这个功能有别于 2.9 的桥接功能，如果关闭 **use-ip-firewall** 过滤，可以大大提高 RouterOS 的网桥转发性能，特别在 WLAN 桥接和一些纯二层的网桥应用中非常有用。



当你需要开启三层的 Bridge 过滤，启用包括 ip firewall filter、mangle、nat 和 queue 流控等功能，就需要打开以下设置：

```
[admin@MikroTik] /interface bridge settings>set use-ip-firewall=yes
[admin@MikroTik] /interface bridge settings>print
                 use-ip-firewall: yes
    use-ip-firewall-for-vlan: no
  use-ip-firewall-for-pppoe: no
[admin@MikroTik] /interface bridge settings>
```

由于打开了三层过滤，转发性能会受到一定的影响，但你可以实现对三层数据的过滤和流控。通过启用三层过滤我们可以实现许多透明桥的流量整形，如 IP 地址流控、P2P 和基于 80 端口的 HTTP 流控等；也可以实现基于 IP 和协议端口的过滤。

注意：如果桥接的网络中通过的是 vlan 封装，即 tagg，即两端交互设备使用 trunk 模式，请选择 **use-ip-firewall-for-vlan**，这样才能在 ip firewall 对 vlan 中的数据报进行处理，如果通过是 pppoe 协议选择 **use-ip-firewall-for-pppoe**

Bridge Port

操作路径：**/interface bridge port**

当我们创建一个 bridge 接口后，需要将指定的网络接口加入 bridge 中，通过 Port 菜单选项添加指定网络接口，即那些网络接口归属于指定的一个 bridge 中。

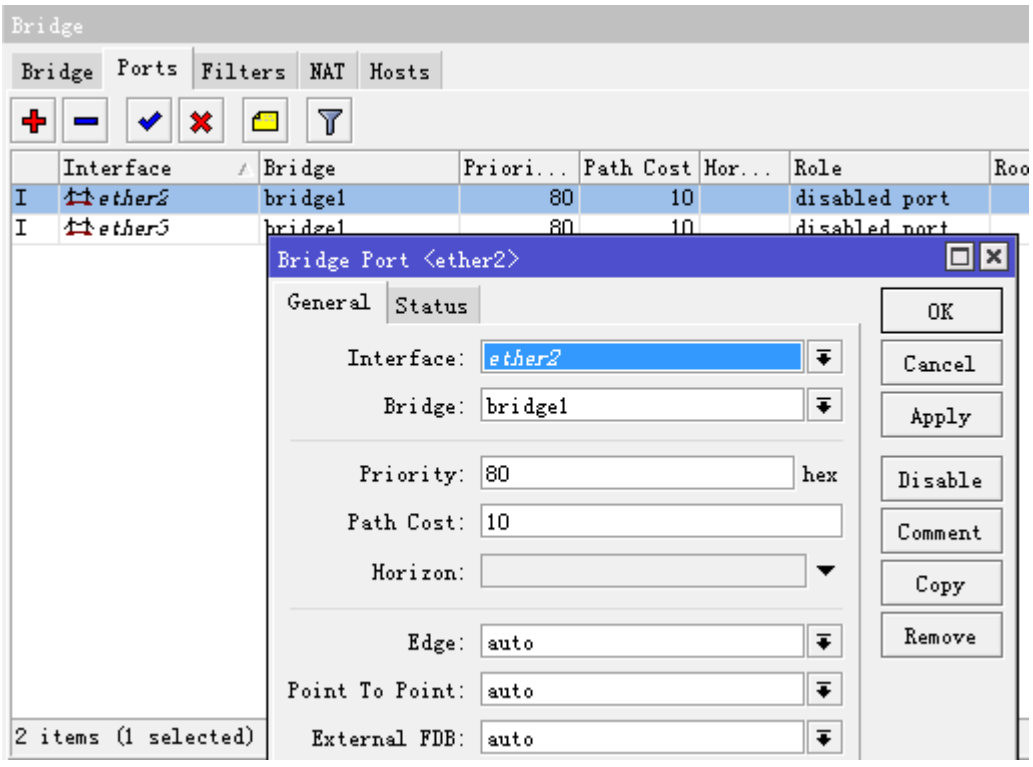
属性描述

bridge (*名称*; 默认: **none**) – 那些接口被定义为 bridge 接口
none – *接口没有被定义到任何桥中*
interface (*只读: 名称*) - 接口名，包含在一个桥内
path-cost (*整型*: 0..65535; 默认: **10**) - STP 使用的用以决定最佳路径代价
priority (*整型*: 0..255; 默认: **128**) - 同一网络中相比较于其他接口的接口优先级

注：从 V2.9.9 版本起，列表中的端口应被添加（add）而非设置(set)，这里都以 2.9.9 以后的配置介绍为主

把 ether2 和 ether3 分到已创建的桥 bridge1 中（V2.9.9 后）：

```
[admin@MikroTik] interface bridge port> add interfae=ether2 bridge=bridge1
[admin@MikroTik] interface bridge port> add interfae=ether3 bridge=bridge1
[admin@MikroTik] interface bridge port> print
 # INTERFACE    BRIDGE PRIORITY PATH-COST        HORIZON
 0 ether2       bri... 0x80      10                  none
 1 ether3       bri... 0x80       10                  none
[admin@MikroTik] interface bridge port>
```



Bridge monitor

命令名: ***/interface bridge monitor***

用于监听一个桥的当前状态，通常在命令行使用。

属性描述

bridge-id (*文本*) – 桥 ID，以如下形式 bridge-priority，bridge-MAC-address
designated-root (文本) - 根桥的 ID
path-cost (*整型*) - 到根桥所需总代价
root-port (*名称*) -根桥连接的端口

监听一个桥：

```
[admin@MikroTik] /interface bridge> print
Flags: X - disabled, R - running
 0   R name="bridge3" mtu=1500 l2mtu=1584 arp=enabled
       mac-address=D4:CA:6D:FA:4B:2A protocol-mode=rstp priority=0x8000
       auto-mac=yes admin-mac=00:00:00:00:00:00 max-message-age=20s
       forward-delay=15s transmit-hold-count=6 ageing-time=5m

 1   R name="bridge7" mtu=1500 l2mtu=1584 arp=enabled
       mac-address=D4:CA:6D:FA:4B:42 protocol-mode=rstp priority=0x8000
       auto-mac=yes admin-mac=00:00:00:00:00:00 max-message-age=20s
       forward-delay=15s transmit-hold-count=6 ageing-time=5m
[admin@MikroTik] /interface bridge> monitor 0
                     state: enabled
    current-mac-address: D4:CA:6D:FA:4B:2A
              root-bridge: yes
         root-bridge-id: 0x8000.D4:CA:6D:FA:4B:2A
         root-path-cost: 0
               root-port: none
              port-count: 1
  designated-port-count: 1
```

Bridge host

命令名: /interface bridge host

Bridge 主机列表显示学习到的 MAC 地址，当 vlan-filtering 启用也能显示 VLAN ID 信息

```
[admin@MikroTik] > interface bridge host print where !local
Flags: L - local, E - external-fdb
  BRIDGE                    VID MAC-ADDRESS       ON-INTERFACE            AGE
  bridge1                   200 D4:CA:6D:77:2E:F0 ether3                   7s
  bridge1                   200 E4:8D:8C:1B:05:F0 ether2                   2s
  bridge1                   300 D4:CA:6D:74:65:9D ether4                   3s
  bridge1                   300 E4:8D:8C:1B:05:F0 ether2                   2s
  bridge1                   400 4C:5E:0C:4B:89:5C ether5                   0s
  bridge1                   400 E4:8D:8C:1B:05:F0 ether2                   0s
[admin@MikroTik] >
```

14.2 Bridge 防火墙

操作路径: ***/interface bridge filter***,

bridge 防火墙功能提供对数据报过滤，能有效的管理经过 bridge 的数据流。

注：在桥接接口之间的数据报就像其他 IP 流一样，也要经过类属的**/ip firewall** 规则（但桥过滤器总是在 IP 过滤器/NAT 之前应用，除了在 IP 防火墙输出之后执行的 **output**）。这些规则可以同真实的物理接收/发送接口一起使用，也可以和简单对桥接在一起的接口划分的桥接口同时使用。

有三种桥过滤器列表：

- **filter** - 有三个预先设定的桥防火墙链表：
  - **input** -其目的地是桥（进入桥设备的数据报，无论什么情况下以本地桥 MAC 地址为目标的数据）。
  - **output** - 来自于桥（由桥设备本身处理发出的数据）
  - **forward** – 通过桥转发（即有桥设备转发到另外网络的数据）。
- **nat** - 桥网络地址翻译提供了改变遍历桥的数据报的源/目的 MAC 地址的方法。它有连条内置的链：
  - **srcnat** - 用于在一个不同的 MAC 地址后“隐藏”一个主机或者一个网络。这个链适用于通过一个桥接口离开路由器的数据报
  - **dstnat** -用于把一些包重定向到另一个目的地址
- **broute** - 使一个桥变为一个桥路由器 — 一种在一些包上起路由作用而在其他包起桥作用的路由器。它有一个预定义链：**brouting**，当一个包进入一个受控接口后它便进行遍历（在“Bridging Decision”之前）。

注：桥的目标网络地址翻译在桥接判定之前执行。当需要涉及到三层过滤时或者流量控制，需要将桥的 **use-ip-firewall** 启用，否则三层过滤和流量控制将无法工作。

你可以在桥防火墙（filter, broute and NAT）中设置数据报标记，就像用 mangle 在 IP 防火墙中设置数据报标记一样。所以用桥防火墙设置的包标记可以在 IP 防火墙中使用，反之亦然。普通桥防火墙属性在这部分描述。一些在 nat,，broute 和 filter rules 之间有区别的参数将在后面的部分描述。

属性描述

802.3-sap (整型) - DSAP（目的档服务访问点） 和 SSAP（源端业务接入点）是两个 1 字节域，它们识别使用链路层服务的网络协议实体。这些字节总是相等的。两个十六进制数字可以在这里指定以匹配 SAP 字节。
802.3-type (整型) - 以太网协议类型，放置在 IEEE 802.2 帧标题后面。仅当 802.3-sap 为 0xAA（SNAP——子网连接点标题）时才生效。例如：AppleTalk 可以由跟随在 0x8098 SNAP 类型码后面的 0xAA SAP 码说明。
arp-dst-address (*IP 地址*; 默认: **0.0.0.0/0**) - ARP 目的地址
arp-dst-mac-address (*MAC 地址*; 默认: **00:00:00:00:00:00**) - ARP 目的 MAC 地址
arp-hardware-type (*整型*; 默认: **1**) - ARP 硬件类型
arp-opcode (arp-nak | drarp-error | drarp-reply | drarp-request | inarp-request | reply | reply-reverse | request | request-reverse) - ARP opcode (数据报类型)
arp-nak – 消极 ARP 应答 (很少使用，主要在 ATM 网络中使用)
drarp-error – 动态 RARP 错误代码, saying that an IP address for the given MAC address can not be allocated 表明一个给定 MAC 地址的 IP 地址不能分配
drarp-reply – 动态 RARP 应答，带有一个主机临时地址分配
drarp-request - 动态 RARP 请求一个对给定 MAC 地址的临时 IP 地址
reply - 带有一个 MAC 地址的标准 ARP 应答
reply-reverse - 带有一个以分配 IP 地址的反向 ARP（RARP）应答
request - 向一个已知 IP 地址询问未知 MAC 地址的标准 ARP 请求
request-reverse - reverse ARP (RARP) request to a known MAC address to find out unknown IP

向已知MAC地址询问未知IP地址的凡响ARP（RARP）请求(intended to be used by hosts to find out their own IP address 主机有意用来查明其本身 IP 地址，类似于 DHCP 服务)

arp-src-address (*IP 地址*; 默认: **0.0.0.0/0**) – ARP 源 IP 地址

arp-src-mac-address (*MAC 地址*; 默认: **00:00:00:00:00:00**) – ARP 源 MAC 地址

chain (文本) - 过滤器工作其中的桥防火墙链（内置或用户定义的）

dst-address (*IP 地址*; 默认: **0.0.0.0/0**) – 目的 IP 地址(仅当 MAC 协议设置为 IPv4 时)

dst-mac-address (*MAC 地址*; 默认: **00:00:00:00:00:00**) – 目的 MAC 地址

dst-port (整型: 0..65535) - 目标端口号或范围（仅对 TCP 或 UDP 协议）

in-bridge (名称) - 数据报进入的桥接口

in-interface (名称) - 数据报进入的物理接口（例如：桥端口）

ip-protocol (ipsec-ah | ipsec-esp | ddp | egp | ggp | gre | hmp | idpr-cmtp | icmp | igmp | ipencap | encap | ipip | iso-tp4 | ospf | pup | rspf | rdp | st | tcp | udp | vmtp | xns-idp | xtp) – IP 协定(仅当 MAC 协定设置为 IPv4)

- **ipsec-ah** - IPsec AH 协议
- **ipsec-esp** - IPsec ESP 协议
- **ddp** - 数据报投递协议
- **egp** - 外部网关协定
- **ggp** – 网关-网关协定
- **gre** - 通用路由压缩
- **hmp** - 宿主监督协议
- **idpr-cmtp** - idp 控制报文传输
- **icmp** - 因特网控制报文协议
- **igmp** - 因特网分组管理协议
- **ipencap** - ip 压缩至 ip
- **encap** - ip 压缩
- **ipip** - ip 压缩
- **iso-tp4** - iso 传输协议类型 4
- **ospf** - 开放式最短路径优先
- **pup** - parc 通用包协定
- **rspf** - 广播最短路径优先
- **rdp** - 靠数据报协议
- **st** - st 数据报模式
- **tcp** - 传输控制协议
- **udp** - 用户数据报协议
- **vmtp** - 通用信息传输
- **xns-idp** - xerox ns idp
- **xtp** – xpress 传输协议

jump-target (名称) – 如果指定 **action=jump**，那么指定用户定义的防火墙链来处理数据报

limit (*整型*/时间{0,1},*整型*) - 以给定值限制包匹配率，有助于减少日志消息的总量

Count - 除非跟随在 **Time** 选项之后否则以包每秒（pps）衡量最大平均包率

Time - 指定包率测量的时间间隔

Burst - 要匹配的脉冲串中的包数量 8

log-prefix (文本) - 在日志信息之前定义用于打印的前缀

mac-protocol (整型 | 802.2 | arp | ip | ipv6 | ipx | rarp | vlan) - 以太网有效负载类型(MAC 等级协议)

mark-flow (*名称*) - marks existing flow

packet-type (broadcast | host | multicast | other-host) – MAC 帧类型:

broadcast – 广播 MAC 包
host -目的为桥本身的数据报
multicast – 多重 MAC 包
other-host - 定位到其他联合广播地址而非到桥本身的数据报
src-address (*IP 地址*; 默认: **0.0.0.0/0**) – 源 IP 地址(仅当 MAC 协议设置为 IPv4 时)
src-mac-address (*MAC 地址*; 默认: **00:00:00:00:00:00**) – 源 MAC 地址
src-port (整型: 0..65535) - 端口号或范围 (仅对 TCP 或 UDP 协议)
stp-flags (topology-change | topology-change-ack) - BPDU (网桥协议数据单元)标志。桥之间为阻止环路定期地互相交换名为 BPDU 的配置信息。
topology-change - 拓扑变化标志是当一个桥检测到端口状态改变时设置，它命令所有其他桥丢弃它们的主机列表并重新计算网络拓扑
topology-change-ack - 拓扑变化确认标志是作为通告数据报响应而设置的
stp-forward-delay (*time*: 0..65535) - forward delay timer 转发延迟定时器
stp-hello-time (*time*: 0..65535) - stp hello 数据报时间
stp-max-age (*time*: 0..65535) – 最大 STP 信息年龄
stp-msg-age (*time*: 0..65535) – STP 信息年龄
stp-port (*整型*: 0..65535) – stp 端口识别
stp-root-address (*MAC 地址*) – 根桥 MAC 地址
stp-root-cost (*整型*: 0..65535) – 根桥代价
stp-root-priority (*时间*: 0..65535) – 根桥优先级
stp-sender-address (*MAC 地址*) – stp 信息发射机 MAC 地址
stp-sender-priority (*整型*: 0..65535) – 发射机优先级
stp-type (config | tcn) - BPDU 类型
config – 配置 BPDU
tcn - 拓扑变化通告
vlan-encap (802.2 | arp | ip | ipv6 | ipx | rarp | vlan) -压缩在 VLAN 帧中的 MAC 协议类型
vlan-id (*整型*: 0..4095) – VLAN 识别域
vlan-priority (*整型*: 0..7) – 用户优先级域

注：仅当目的 MAC 地址为 01:80:C2:00:00:00/FF:FF:FF:FF:FF:FF (桥组地址)时, stp 匹配器才有效，同时 stp 应被启用。仅当 mac-protocol 为 arp 或 rarp 时 ARP 匹配器才有效。VLAN 匹配器仅对 vlan 以太网协议有效。IP 相关匹配器仅当 mac-protocol 被设置为 ipv4 时才有效

如果实际帧和 IEEE 802.2 和 IEEE 802.3 标准一致时，802.3 匹配器就会被询问（注意：它并不是在全世界网络使用的工业标准以太网帧格式）。这些匹配器对其他包会被忽视。

Bridge Filter

操作路径: ***/interface bridge filter***

这部分描述的是桥数据报过滤器详细的过滤选项，在一般的防火墙描述中这部分通常都被省略掉了。

属性描述

action (accept | drop | jump | log | mark | passthrough | return; default: **accept**) - 如果数据报匹配了其中一个规则就采取动作：
accept - 接受包，无动作。例如：数据报通过而没有任何动作，并且没有其他规则会在相关列表/链中处理。
drop – 悄然地丢弃包(不发送 ICMP 拒绝信息)

jump - 跳转到由 jump-target 变数指定的链
log - 记录数据报
mark – 标记数据报以便后面使用
passthrough - 忽视这条规则并到下一个。除了对包计数外像一个被禁用的规则一样动作
return - 从跳转发生的地方回到前一个链
out-bridge (*名称*) - 流出桥的接口
out-interface (*名称*) - 数据报离开桥的接口

过滤一个主机 MAC 地址 00:0c:11:23:00:0a 通过网桥，

```
[admin@MikroTik] /interface bridge filter
[admin@MikroTik] /interface bridge filter> add action=drop chain=forward src-mac-address=
00:0C:11:23:00:0A/FF:FF:FF:FF:FF:FF
[admin@MikroTik] /interface bridge filter> print
Flags: X - disabled, I - invalid, D - dynamic
 0    chain=forward action=drop
      src-mac-address=00:0C:11:23:00:0A/FF:FF:FF:FF:FF:FF log=no log-prefix=""
[admin@MikroTik] /interface bridge filter>
```

Bridge nat

操作路径: ***/interface bridge nat***

Bridge nat 可以定义 mac 地址的 nat 规则，即可以指定一个 mac 代理访问其他网络，类似 arp-proxy。

属性描述

action (accept | arp-reply | drop | dst-nat | jump | log | mark | passthrough | redirect | return | src-nat; 默认: **accept**) - 如果数据报匹配了其中一个规则就采取动作：
accept - 接受包，无动作。例如：数据报通过而没有任何动作，并且没有其他规则会在相关列表/链中处理。
arp-reply - 发送一个带有指定 MAC 地址的 ARP 应答(任何其他包都会被这条规则忽略，仅在 **dstnat** 链内有效)
drop – 悄然丢弃数据报 (不发送 ICMP 拒绝信息)
dst-nat - 改变一个包的目的 MAC 地址 (仅在 **dstnat** 链有效)
jump - 跳转到由 jump-target 变数指定的链
log - 记录数据报
mark – 标记数据报以便后面使用
passthrough - 忽视这条规则并到下一个。除了对包计数外像一个被禁用的规则一样动作
redirect - 把数据报重新定位到桥本身（仅在 **dstnat** 链中有效）
return - 从跳转发生的地方回到之前的链
src-nat - 改变包的源 MAC 地址 (仅在 **srcnat** 链中有效)
out-bridge (*名称*) - 流出桥接口
to-arp-reply-mac-address (*MAC 地址*) - 当选中 **action=arp-reply** 时，把源 MAC 地址加入以太网帧及 ARP 有效负载
to-dst-mac-address (*MAC 地址*) - 当选中 **action=dst-nat** 时，把目的 MAC 地址加入以太网帧
to-src-mac-address (*MAC 地址*) - 当选中 **action= src-nat** 时，把源 MAC 地址加入以太网帧

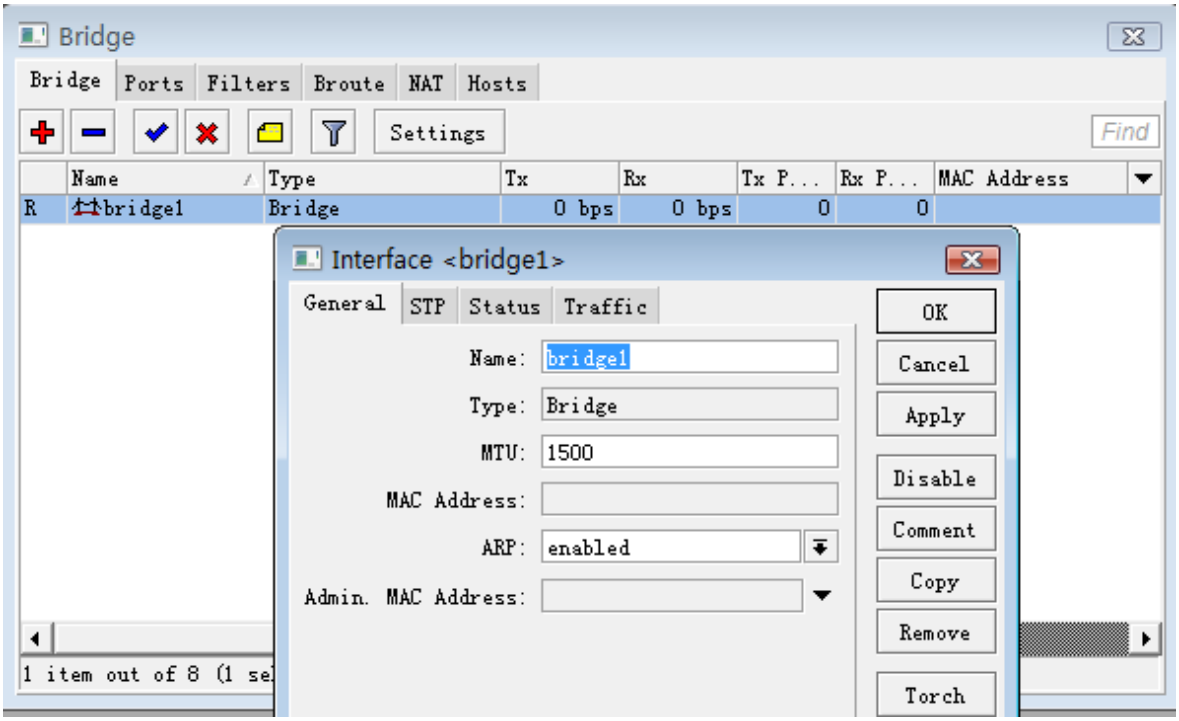
故障分析

- **路由器显示我的规则不合法**
  - in-interface, in-bridge (或 in-bridge-port) 被指定，但并不存在这样的接口
  - 有一条 action=mark-packet 的动作，但没有 new-packet-mark
  - 有一条 action=mark-connection 的动作，但没有 new-connection-mark
  - 有一条 action=mark-routing 的动作，但没有 new-routing-mark

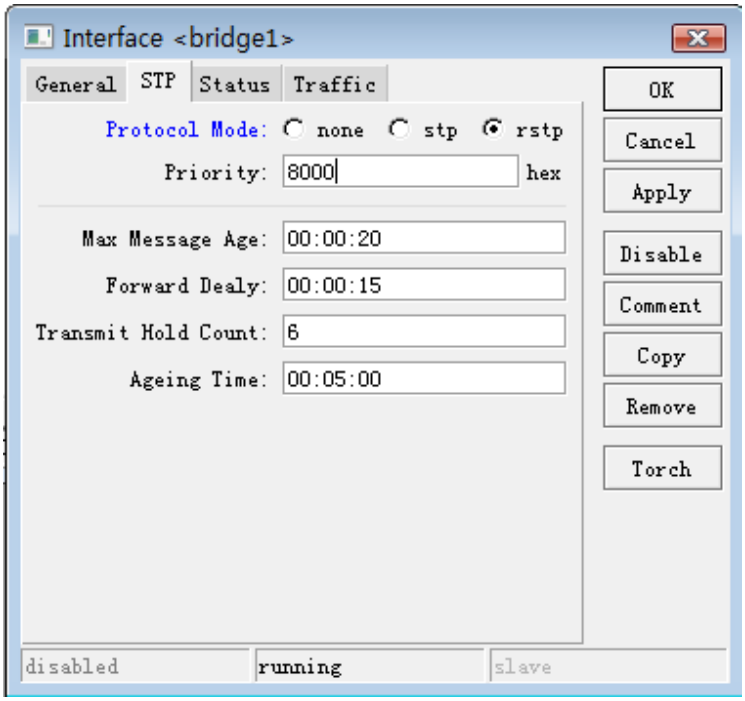
14.3 Bridge 实现二层端口隔离

RouterOS 具有 Bridge 的桥接功能，在配置多网口的情况下可以实现二层数据的转发，即可以实现交换机功能，加上 RouterOS 支持 birdge filter 的过滤，同样也支持对二层数据的管理，通过配置 Bridge 的防火墙规则实现多网口的端口隔离，v6.19 以前的版本采用 bridge filter 隔离，v6.19 后引入了自动隔离属性。

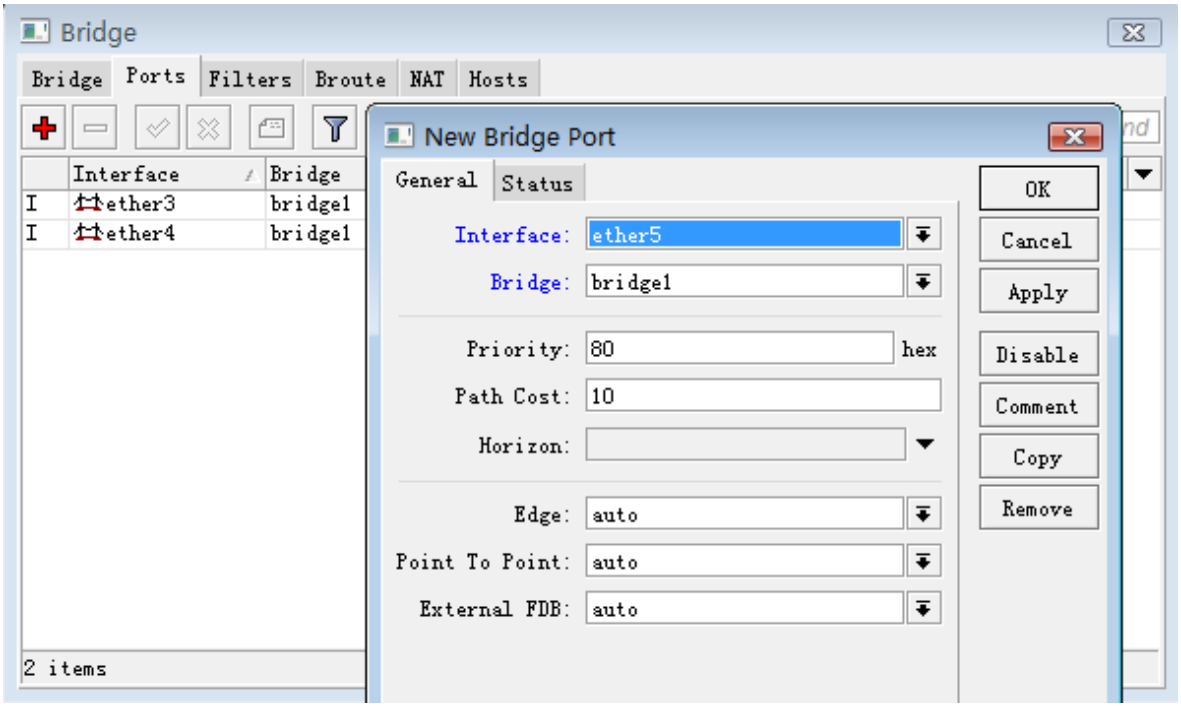
在这里我们通过 RB450 的操作为实例，配置二层端口隔离。首先我们在 Bridge 中添加一个网桥 bridge1：



在 bridge 中启用 rstp 快速生成树协议，防止二层的回环出现，同样也是支持二层的冗余功能，在这里我们选择 rstp：

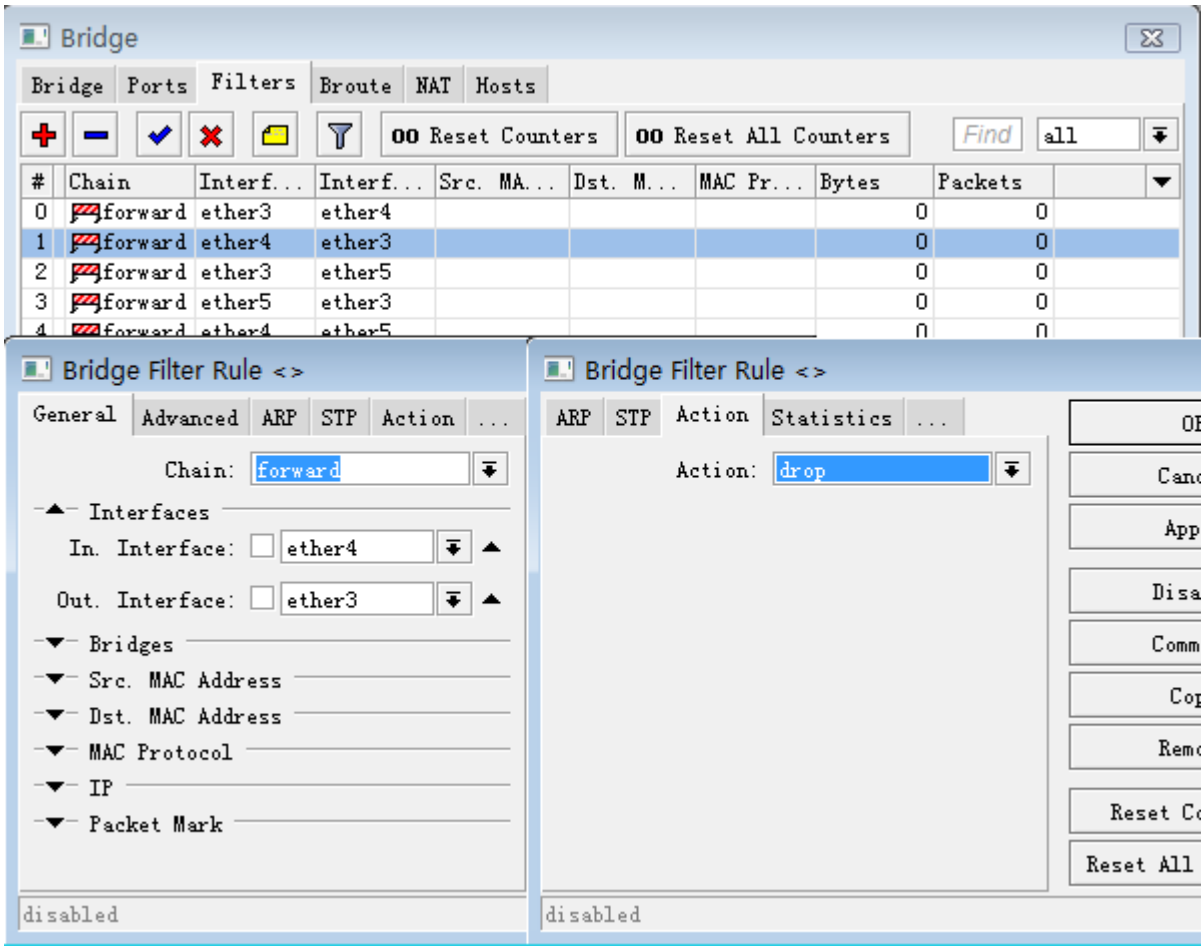


添加完桥接功能后，需要将对应的网卡添加入 bridge1 中，进入 Port 中设置，我们将 3 个网卡 ether3、ether4 和 ether5 一个一个添加到 bridge1 中：

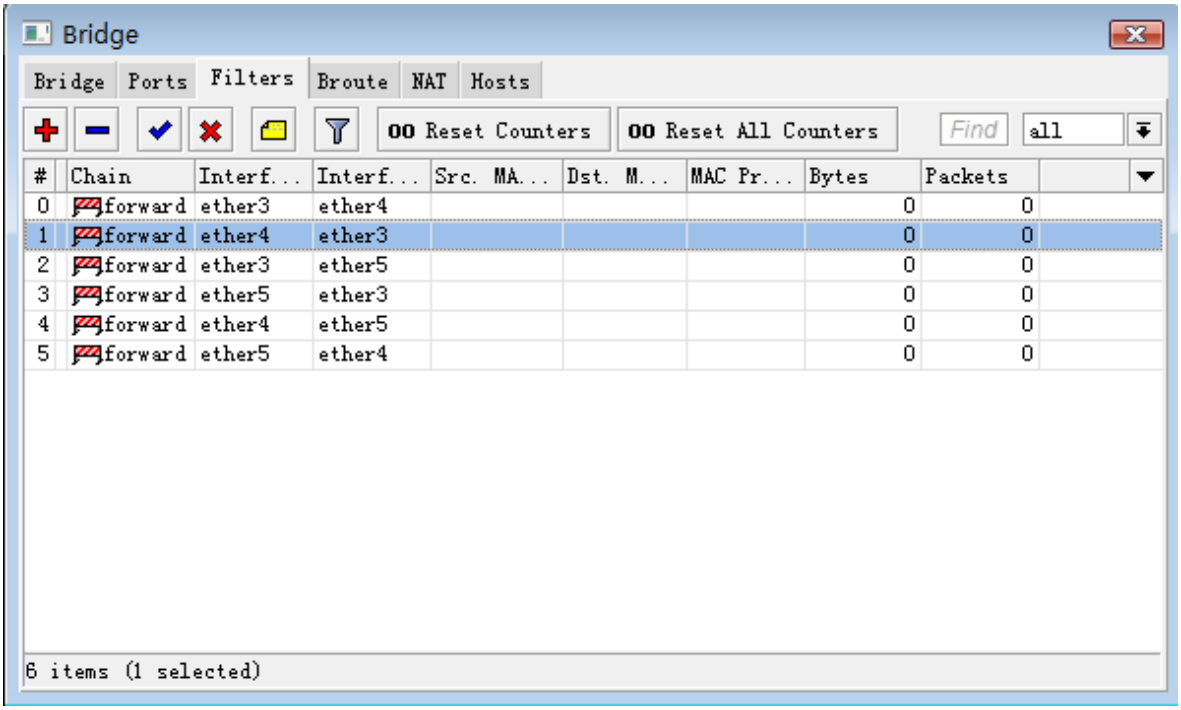


添加完每个端口后，现在 RB450 的 3 个以太网口，就完成了桥接的设置，这样 3 个口就实现了二层的交换功能。

这里我们禁止 ether3、ether4 和 ether5 进行通信， 我们进入 filter 中设置防火墙过滤规则，我们首先配置 ether3 与 ether4 的数据隔离我们在 interface 选项中设置 In-interface 和 Out-interface（In-interface 为数据进入的网口，Out-interface 为数据出去的网口，数据是双向传输的，两个接口需要做两条规则），然后选择 action 设置 action 参数为 drop，丢弃数据：



下面是设置好的状态：



14.4 创建二层透明流控过滤

RouterOS 的桥架设置是允许将二层数据转发到三层防火墙过滤，即可以实现透明的三层数据处理，包括防火墙策略和流控。即实现一台透明的二层流控过滤设备。下面的事例将两个网络中间加入一台 RouterOS 设备，并配置两张网卡。

启用并修改以太网卡，连接到内部网络的网卡命名为 int，连接到上级路由器的网卡为 ext ：

```
/interface set ether1,ether2 disabled=no
/interface set ether1 name=int
/interface set ether2 name=ext
```

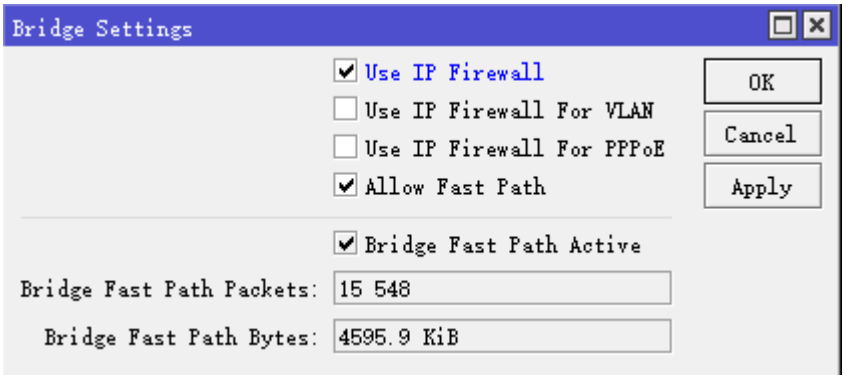
创建一个桥接口，并将两个物理网卡 int 和 ext 做桥接：

```
/interface bridge add name=bridge
/interface bridge port add interface=ext bridge=bridge
/interface bridge port add interface=int bridge=bridge
```

我们用 10.0.0.2 作为管理接口，并设置网关为 10.0.0.1，通过 ping 测试，确认配置正确，如果不能通，你可以换一下网线（例如：将插在 ext 网卡上的线换到 int 上，看是否网卡设置反了）

```
/ip address add interface=bridge1 address=10.0.0.2/24
/ip route add gateway=10.0.0.1
```

打开 bridge setting，设置 use-ip-firewall，让二层数据交给三层防火墙处理，但会增加 CPU 性能



注意：如果你的二层流量在交换机的接口上设置了 VLAN 标签（trunk），请选择 use-ip-firewall-for-vlan。

以上配置完成后，现在可以在 simple queue 设置期望的流控策略，你可以在队列中使用真实的网卡名称。例如，限制所有下载为 10Mbps 和所有上传为 5Mbps，仅需要添加两条队列就可以了：

```
/queue simple add max-limit=5000000 interface=ext
/queue simple add max-limit=10000000 interface=int
```

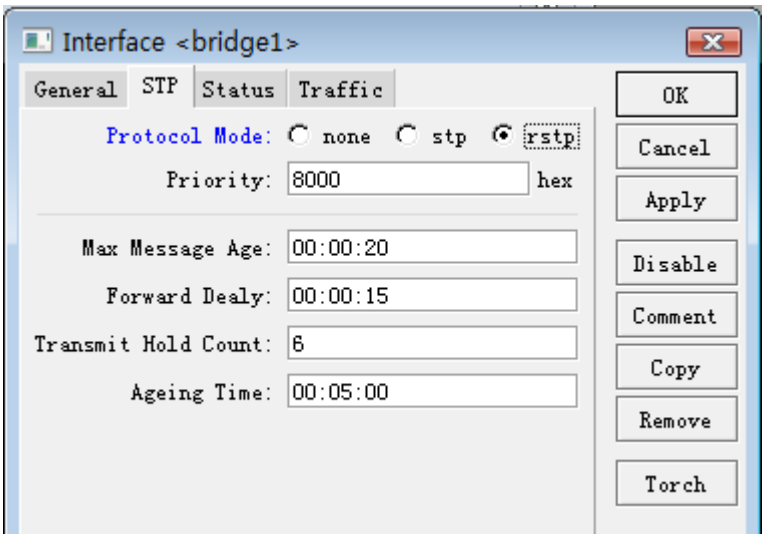
也可以对指定 IP 地址做流控，或者使用 PCQ 策略，以及调用 mangle 做标记策略

14.5 通过 Bridge Filter 控制 MAC 地址

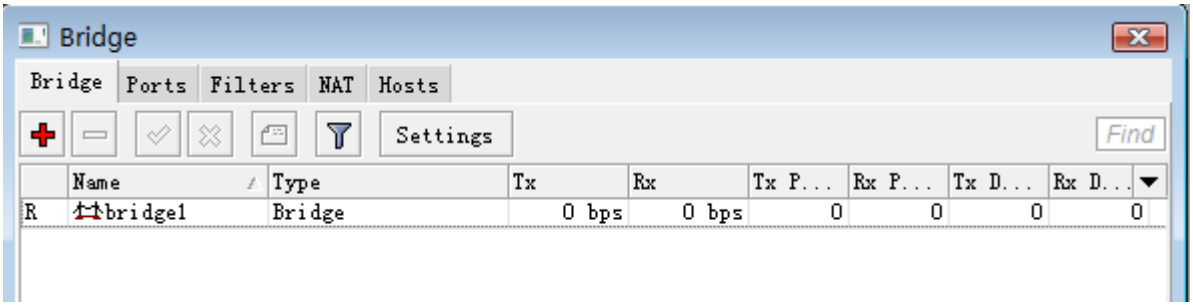
通过 bridge filter 控制 MAC 地址，如当我们把 RouterOS 设置为透明桥时，可以控制网络内的主机 MAC 通讯，这样我们可以从二层上控制客户端 PC。

我们通过 bridge 过滤 MAC 地址，必须启用 bridge，并指定相应网络接口到 bridge port 中，至少需要设定一个网络接口到 Port 中，设置 bridge 的操作如下

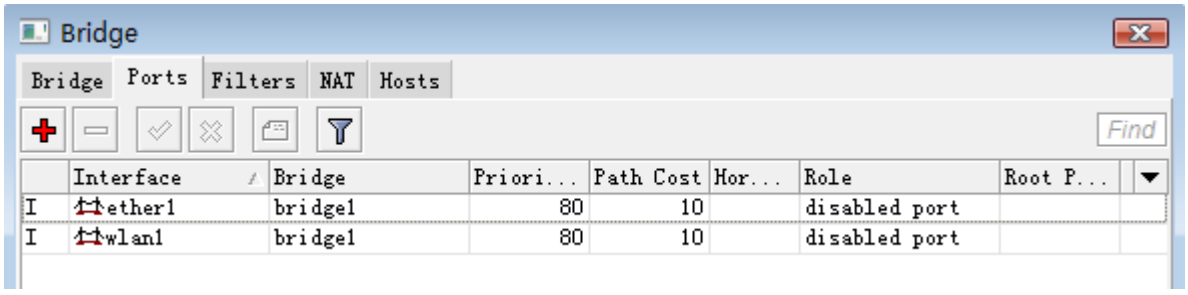
1、添加一个 bridge，默认的 bridge 的名称为 bridge1，并设置 RSTP（快速生成树协议）模式：



添加完 bridge 后，我们可以在 bridge 列表中查看到：



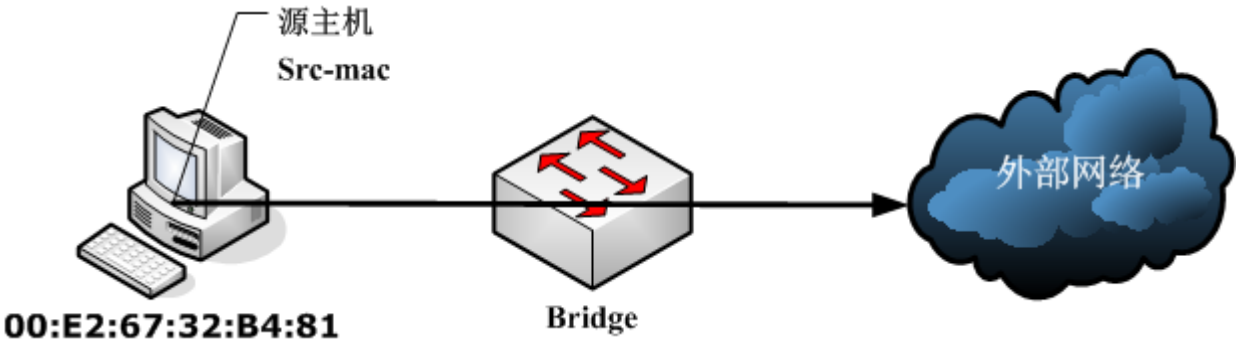
2、我们将 ether1 和 wlan1 网络接口添加到 bridge1 里，这样 2 个网络接口就实现了桥接功能



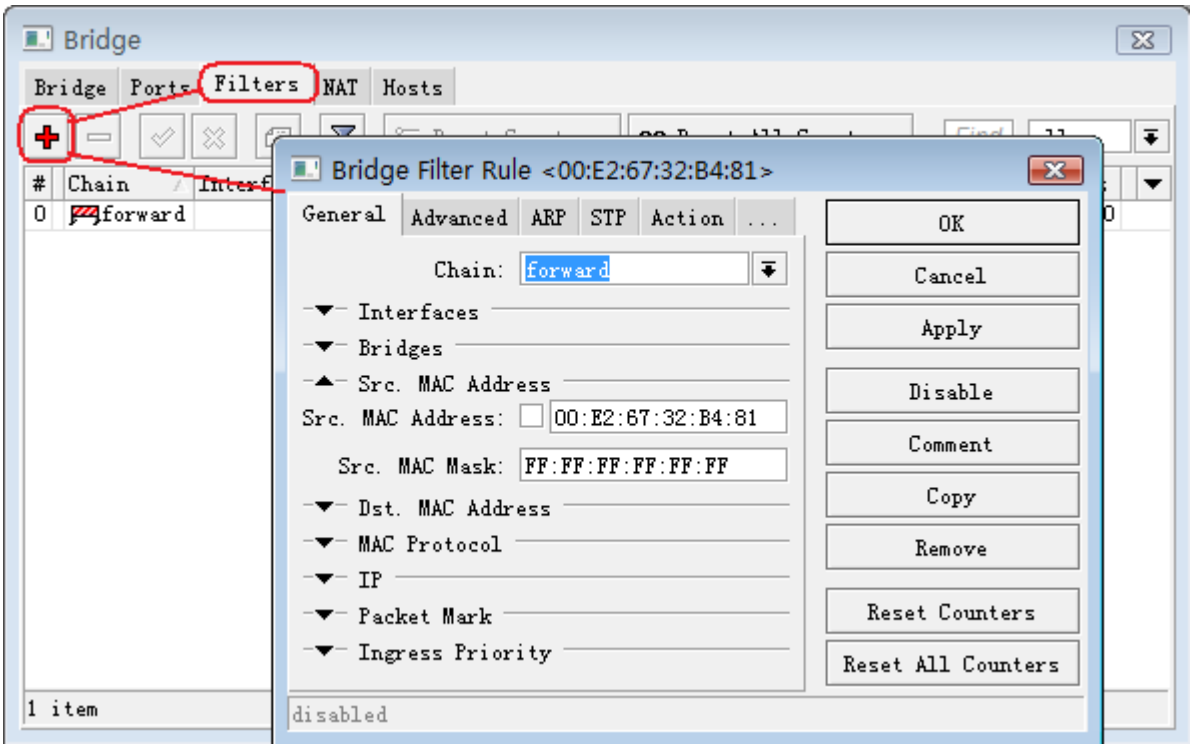
过滤源和目标 MAC 地址

1、源 MAC 地址过滤

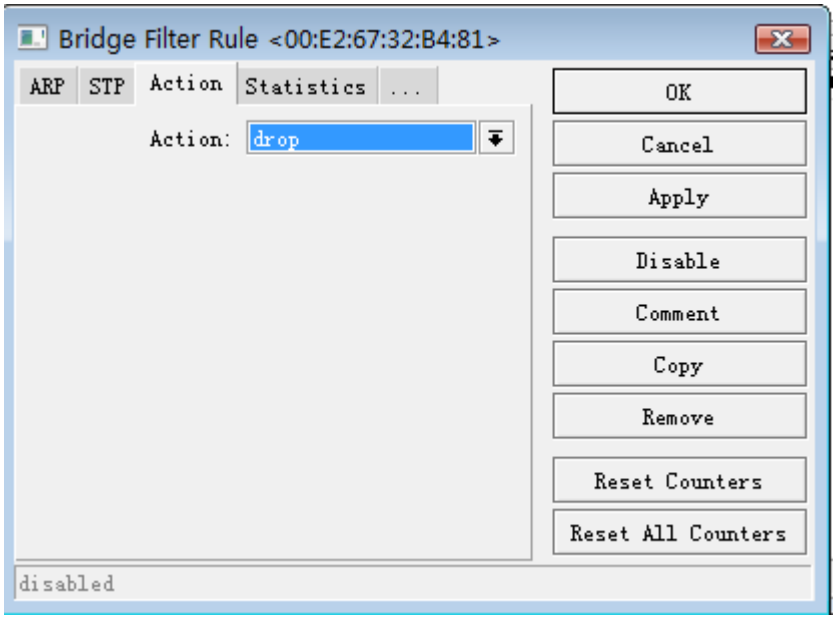
在设置完基本的 bridge 后，我们进入 bridge filter 中配置桥防火墙过滤，首先我们需要对指定的一台 PC 的 MAC：00:E2:67:32:B4:81 地址做过滤，不允许与 bridge 的外部网络连接，如下图：



这个 MAC 是发起源，选择 src-mac-address，由于这里拒绝访问 bridge 以外的网络，选择 chain=forward，，设定 action=drop。RouterOS Winbox 配置如下：



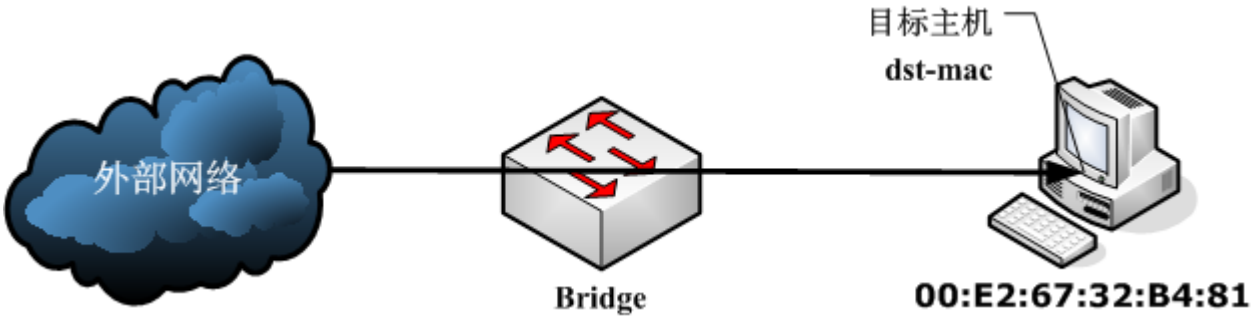
接下来选择 Action 为 drop，丢弃该 MAC 地址发出的数据：



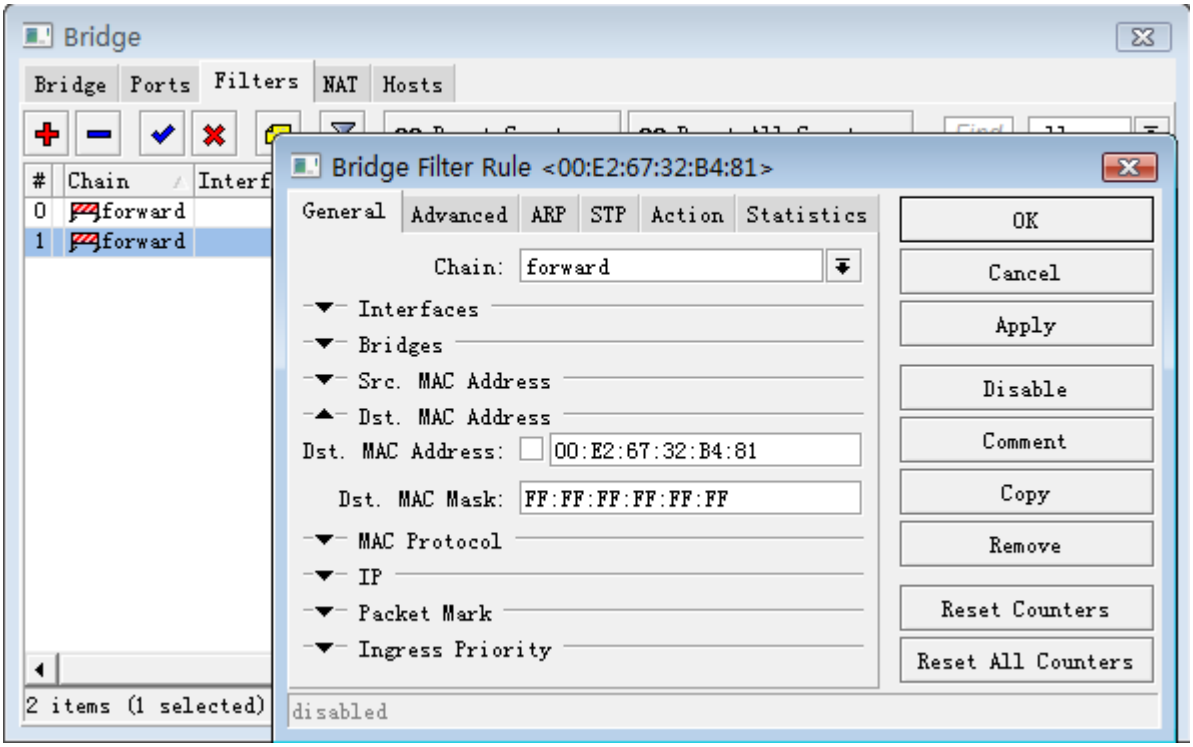
注：我们设置 src-mac-address 时，后面跟着 MAC 子网，这个子网和我们 IP 层的子网掩码类是，只是 MAC 子网是按照十六进制换算，十六进制的 FF 与 IP 子网的 255 是相同，规定网络范围，因为这里是过滤一个台主机的 MAC 地址，所以我们设置 MAC 子网掩码为 FF:FF:FF:FF:FF:FF。

2、目标 MAC 地址

反过来从外网访问一个该主机，则是目标 MAC 过滤，只是之前我们设置的是 scr-mac-address，反过来填写目标的 MAC，即 dst-mac-address，我们还是用之前的 MAC 地址做事例

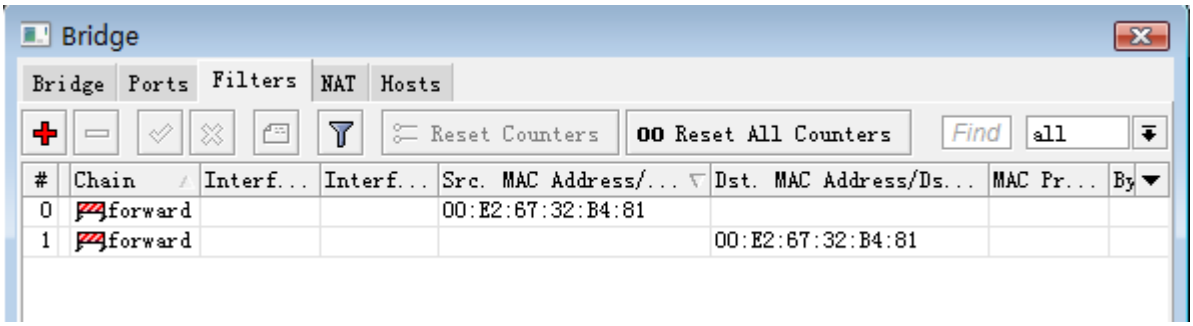


我们添加目标 MAC 地址过滤规则，选择 dst-mac-address=00:E2:67:32:B4:81，dst-mac-address 默认为全 FF。



Action 同样选择 drop，丢弃到该目标 MAC 的数据。

下面我们可以在 filter 中看到 2 条规则，分别是控制从源地址和目标地址的数据，这样设置后，我们可以理解为对 00:E2:67:32:B4:81 主机数据的双向过滤。



过滤指定厂商的 MAC 地址

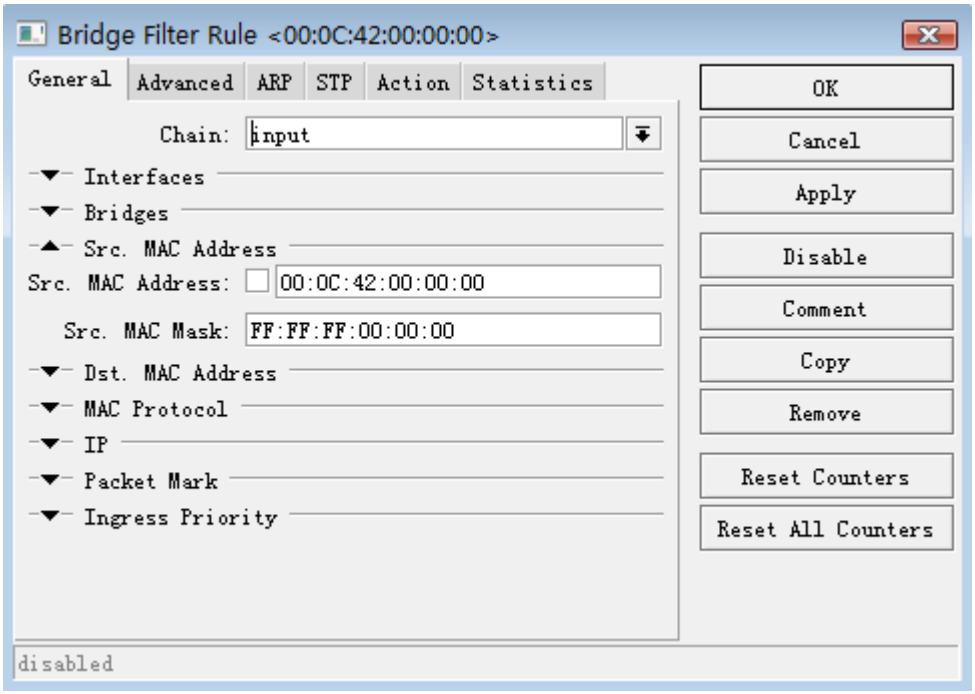
我们知道所有的网络设备都有一个 6 位的 MAC 地址，前 3 位为生产厂商标示，后 3 位为设备编号，当我们在做无线网桥的时候，只允许特定某一厂商的网卡连接到 RouterOS，可以通过 Bridge 的防火墙控制 MAC 地址，限制某一类的 MAC 不能连接到 RouterOS 设备，或者通过 RouterOS 设备。

例如，我们的一台 RouterBOARD 设备要求只能允许其他 RouterBOARD 的设备连接，可以通过 brigde fiter 控制，由于每个 RouterBOARD 的以太网卡 MAC 地址都是前 3 位都是以 00:0C:42 开头，我们只需要允许前 3 位 MAC 为 00:0C:42 的 MAC 通过就可以。

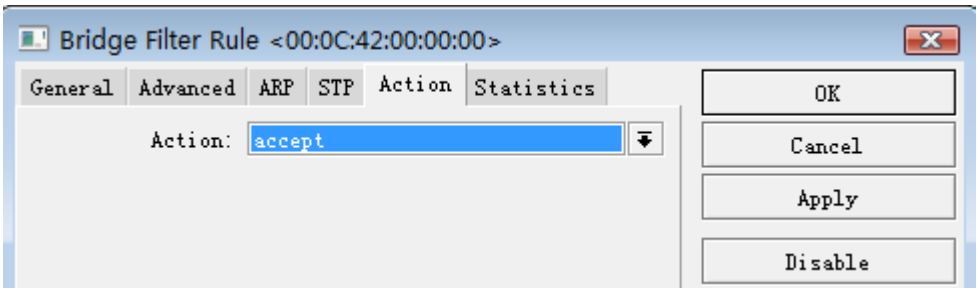
在设置为 bridge 的接口参数后，我们在 filter 中配置 2 条 input 规则，限制除了 MAC 地址前 3 位是 00:0C:42 能连接 RouterOS，其他的都拒绝掉。

根据 RouterOS 防火墙原理，分别需要设置两条规则，一条是接受 MAC 地址前 3 位是 00:0C:42 的 MAC 地址，第二条是丢弃其他所有的 MAC 数据。

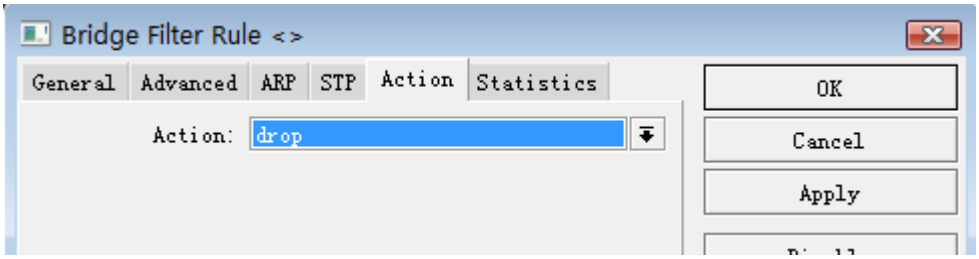
第一条规则，设置 src-mac-address=00:0C:42:00:00:00/src-mac-mask=FF:FF:FF:00:00:00



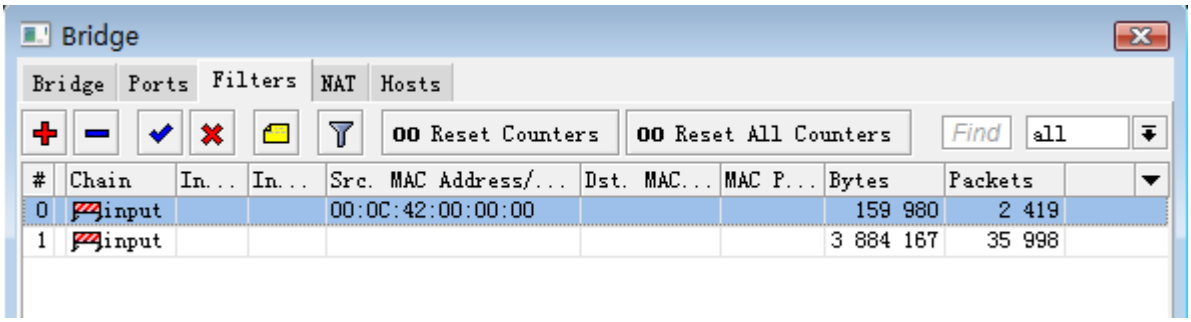
在 action 中选择 accpet 接受，数据通过



第二条规则，是丢弃其他所有的 MAC 数据



配置完成后，如下：



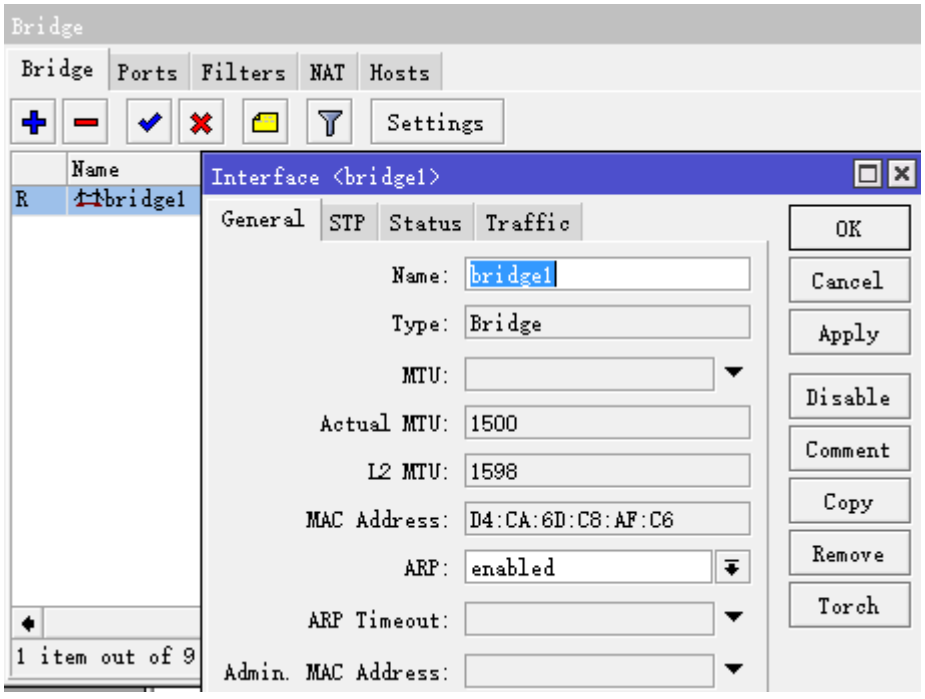
14.6 Bridge nat 修改 VRRP 接口 MAC

关于 VRRP 在同一广播域创建 PPPoE 拨号，会涉及到一个问题，就是任何一个版本的 RouterOS 创建 VRRP 的 MAC 地址是一样，造成一个运营商 BRAS 下如果出现两个人都使用 RouterOS 的 VRRP 拨号，会导致拨号失败，因此解决这个问题在网络上已经流传很久，即通过 bridge 的 nat 功能对 VRRP 接口 MAC 地址修改。

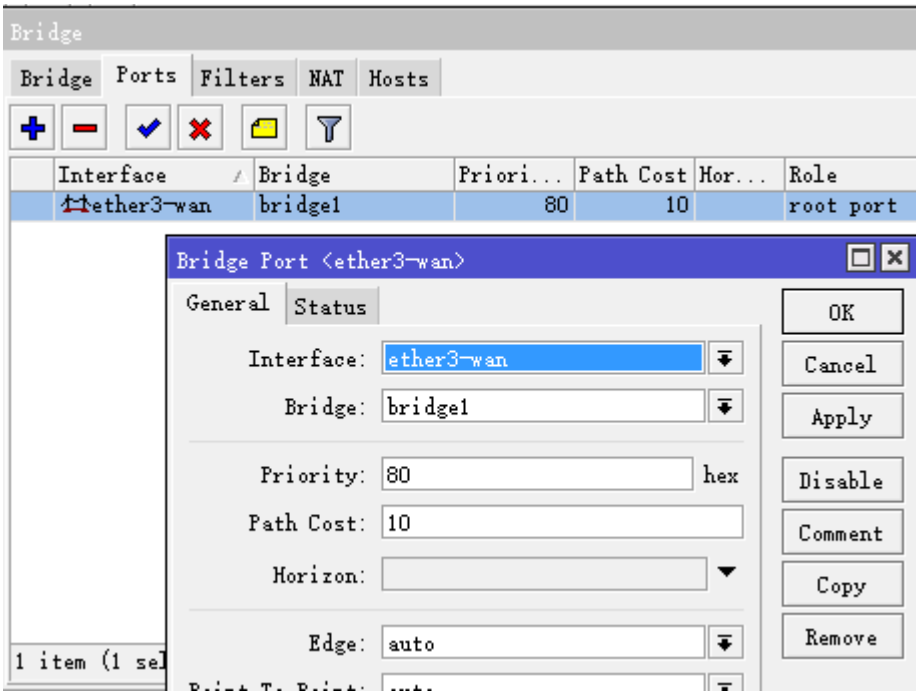
二层的 MAC 地址和三层的 IP 地址可以作为相同的理解方式，MAC 作为二层的物理地址识别主机，而 IP 地址则作为三层路由来识别主机，MAC 地址也可以通过 nat 进行伪装，映射一类的操作，proxy-arp 也就是 MAC 地址被 nat 的一个实例。

使用 Bridge 对 VRRP 接口的 MAC 地址进行，修改需要先将 VRRP 从属的物理网卡，加入到 bridge 里，例如，我们使用 ether3-wan 的接口作为 VRRP 承载接口，创建 VRRP 接口并拨号。

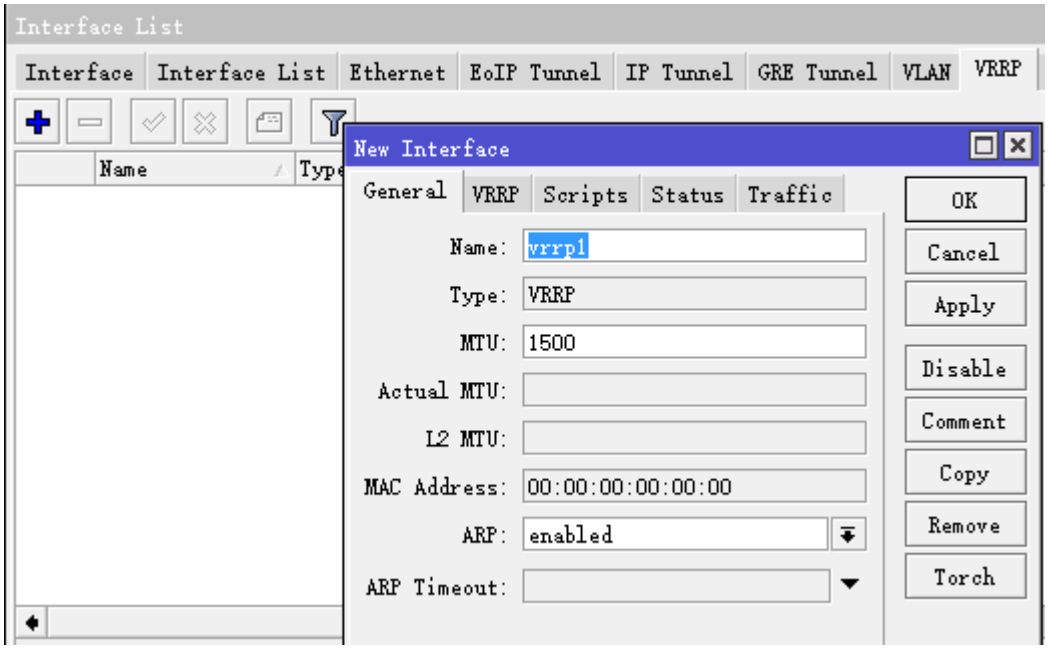
首先，我们将 ether3-wan 加入 bridge1 接口，下面是通过 winbox 在 bridge 菜单下创建 bridge1：



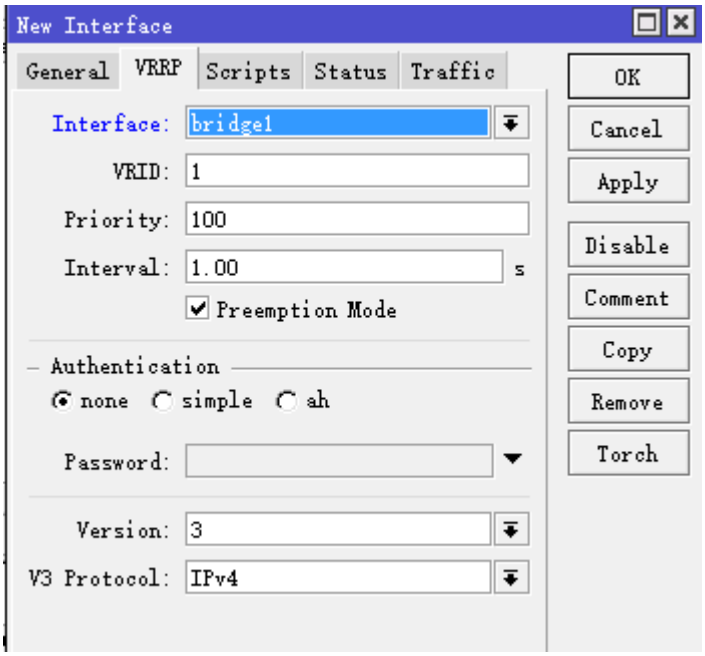
进入 bridge port 将 ether3-wan 接口加入 bridge1



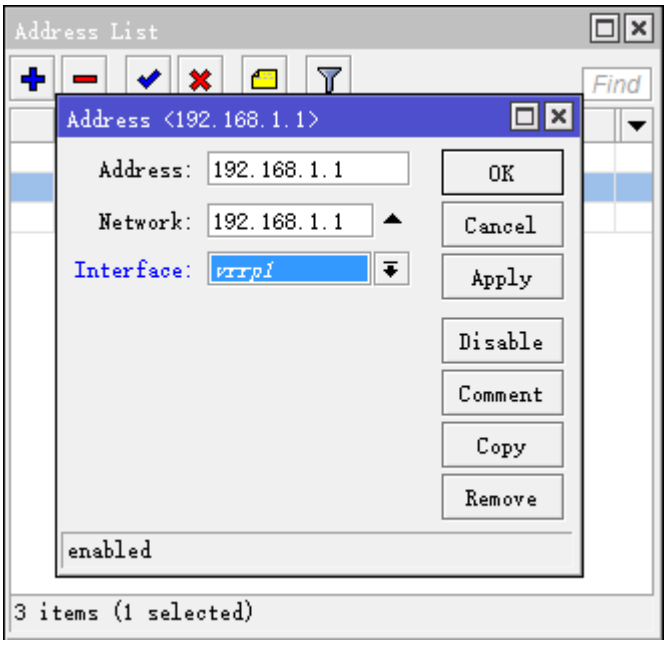
进入 interface vrrp 菜单下创建 vrrp1 接口



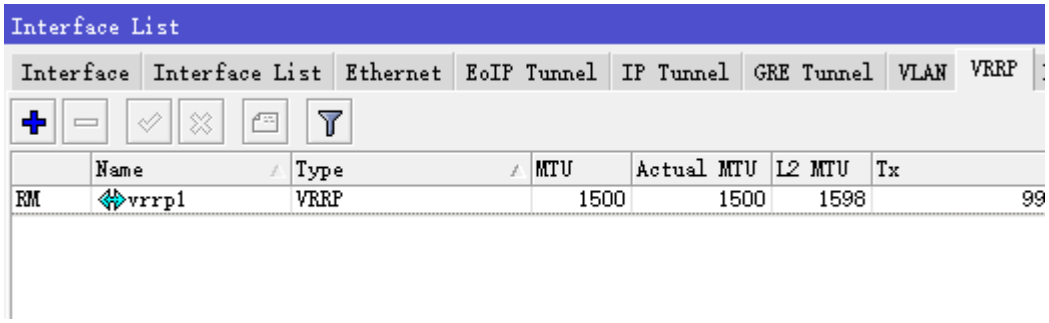
设置 vrrp1 的 interface 为 bridge1，设置 VRID 为 1，后续的 VRRP 接口 VRID 依次设置 2，3，4，5 等等，只要不一样即可：



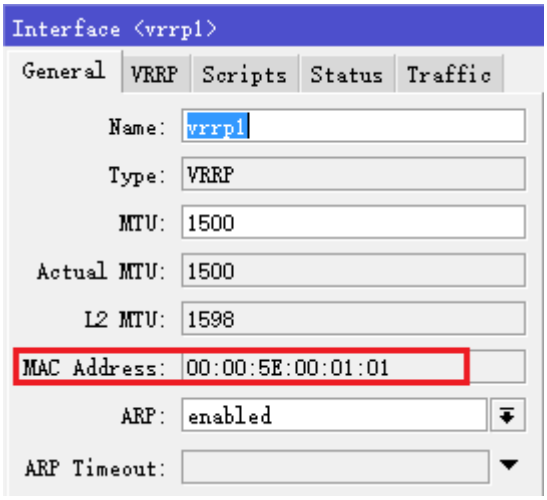
进入 ip address 下，为 VRRP 创建一个 IP 地址（对 IP 地址没有要求，目的是让 VRRP 接口生效）



IP 地址添加完成后，可以看到 VRRP 列表下单 vrrp1，由红色变为黑色，即生效，状态显示为“RM”



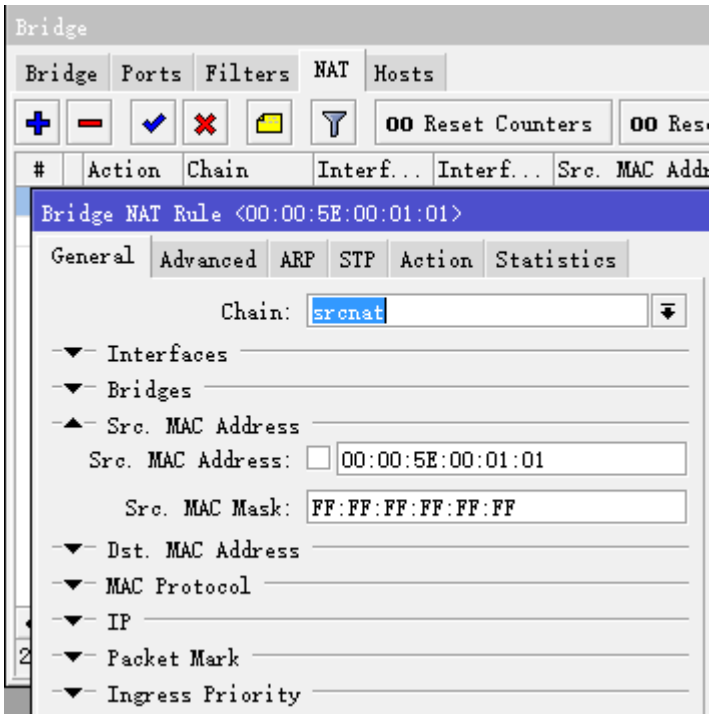
我们在 winbox 双击 vrrp1 接口，可以看到 MAC address 地址为 00:00:5E:00:01:01，需要通过 bridge 来完成 vrrp1 去拨号的 MAC 地址



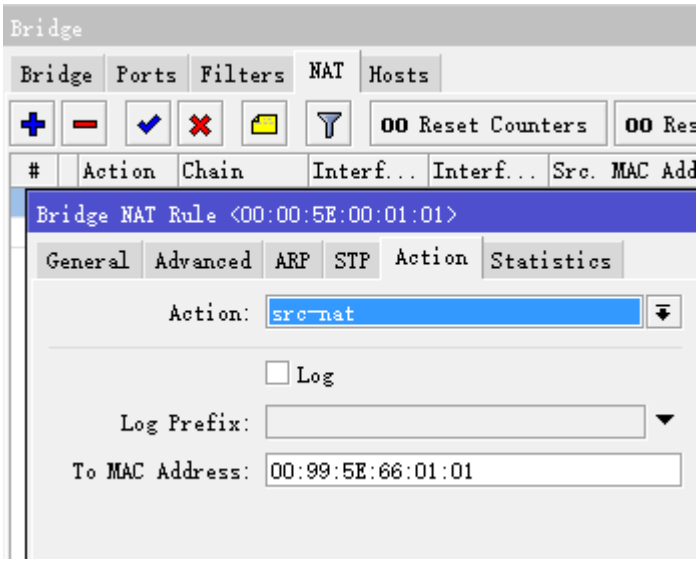
下面我们进入 bridge 菜单，选择 NAT 窗口，对 vrrp1 接口的 MAC 地址做 nat 转换，注意，我们不是真正的去修改了 vrrp1 的接口 MAC 地址，而是利用 vrrp1 加入 bridge1 后，在从接口发出时，使用 bridge 的 nat 功能截获 vrrp1 的 MAC 00:00:5E:00:01:01，修改发向 BRAS 的源 MAC 源地址和 BRAS 发给 RouterOS 的目标 MAC，达到 MAC 被修改的目的。

这个方法和 IP 地址的端口映射一个道理，因此我很佩服发明这个方法的牛人，对网络协议掌握到如火纯清地步，而且能灵活应用和扩展，因此网络高手就是知其然，还能知其所以然，不管学习 RouterOS，思科或华为，学习基础的网络协议才是入门的前提。

我们进入 bridge nat 接口，添加一个 nat 规则，设置 chain 为 srcnat，src-mac-address= 00:00:5E:00:01:01/FF: FF: FF: FF: FF: FF，MAC 地址也有子网掩码的全 FF，就代表 IP 地址子网掩码 255.255.255.255，表示一个主机地址。

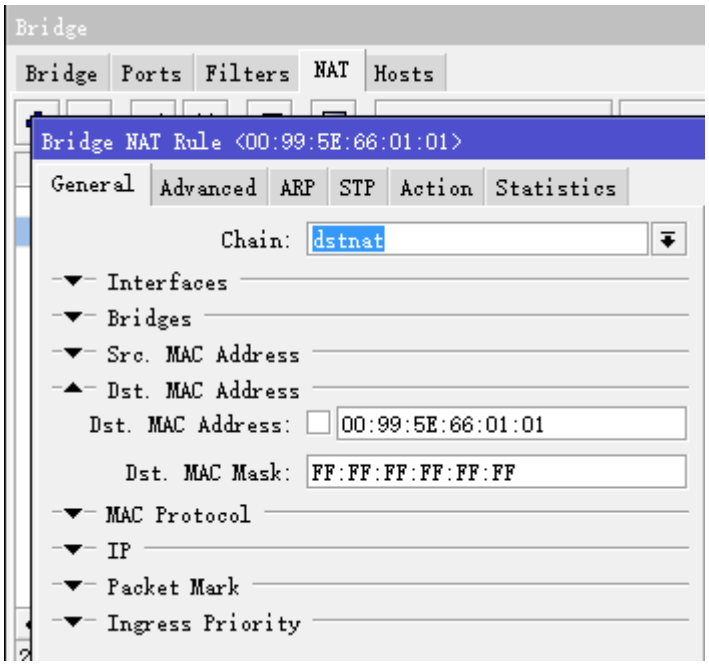


选择 action 菜单，设置 action=src-nat，to-mac-address 参数就是我们想要修改的 MAC 地址，设置我们设置为 00:99:5E:66:01:01

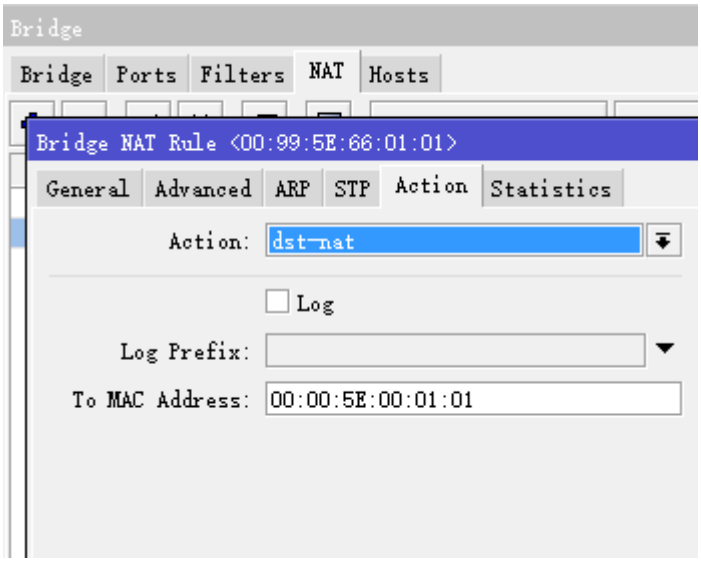


设置完成后点 OK

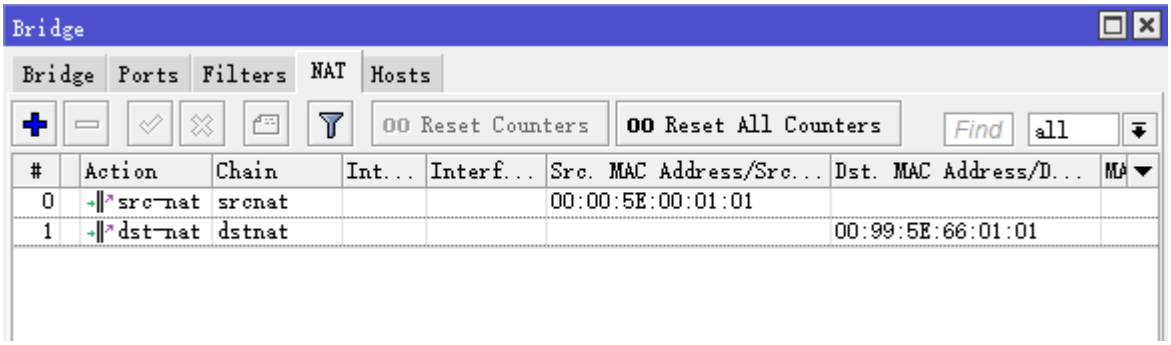
然后再创建第二条规则，选择 chain 为 dstnat，设置 dst-mac-address=00:99:5E:66:01:01/FF: FF: FF: FF: FF: FF



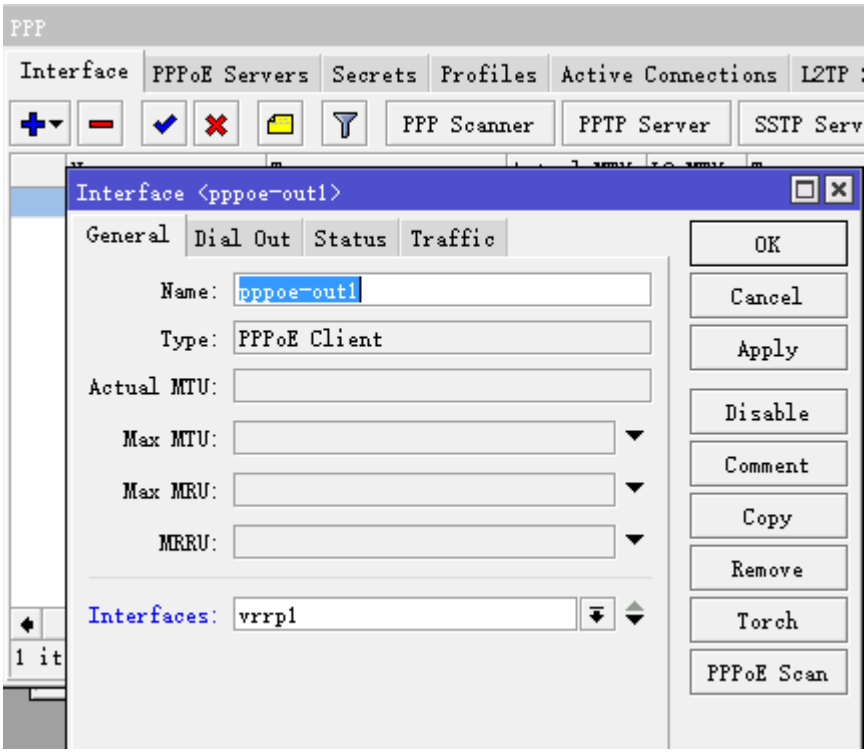
选择 action 菜单，设置 to-mac-address=00:00:5E:00:01:01



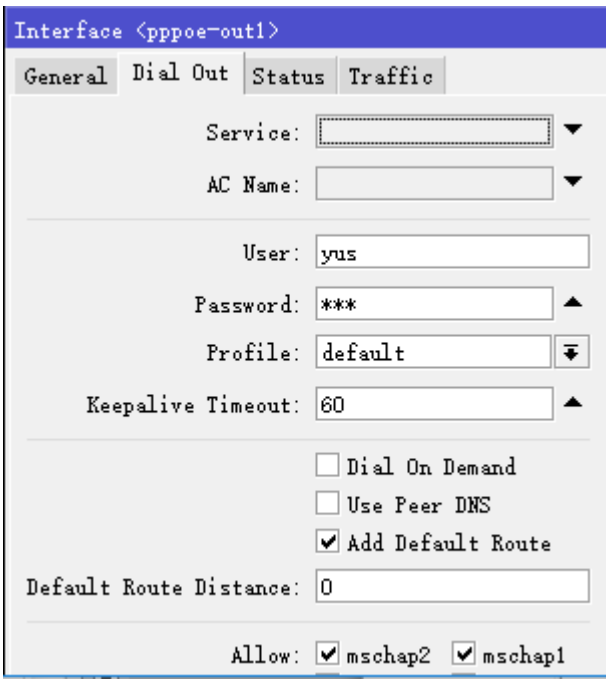
这样 vrrp1 接口的 MAC 修改设置完成，一共两条规则，srcnat 负责发向 BRAS 的 MAC 修改，dstnat 负责 BRAS 回给 RouterOS 的 MAC 修改



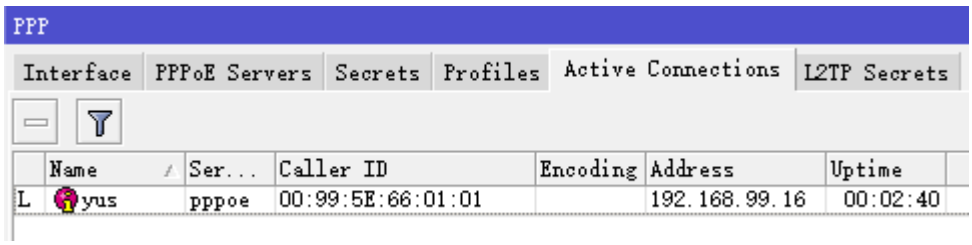
最后我们完成 PPPoE 拨号，创建 pppoe-out1 接口



设置账号密码：



用RouterOS模拟了一台PPPoE server做认证，可以看到caller ID已经是我修改的MAC地址00:99:5E:66:01:01，



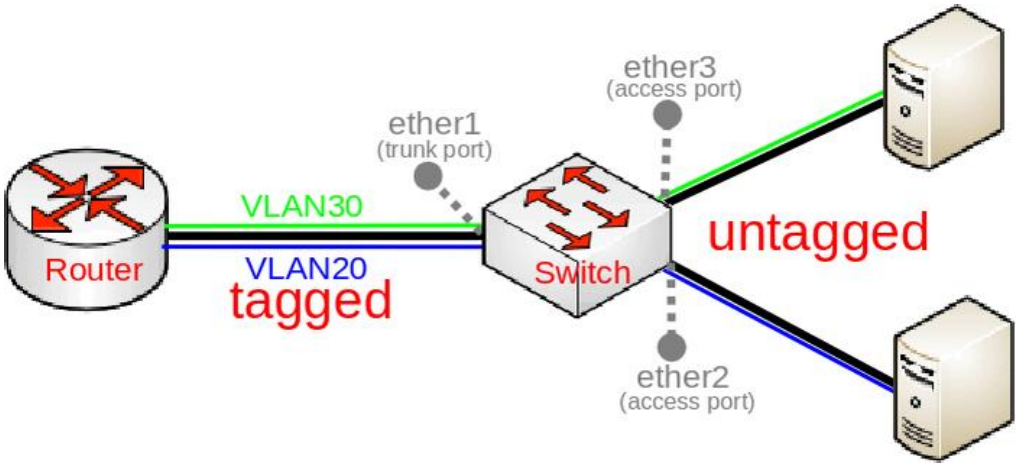
后续添加到vrrp接口也按照这个方面，一条条添加bridge nat规则（相关VRRP多PPPoE拨号配置参考第17.3章节）。

14.7 Bridge VLAN（access/trunk）

特别注意：在RouterOS VLAN配置包括两部分 /interface vlan 和/bridge vlan，区分如下：

- **/interface vlan**：是三层vlan，需要在VLAN上配置IP地址。如果你是把RouterOS看作一台路由器，下方的交换机通过trunk将VLAN透给RouterOS路由器，那么需要在/interface vlan创建对应物理接口的vlan子接口即可，无需到bridge vlan配置。（参见第十五章节 Interface VLAN）
- **/bridge vlan**：二层vlan，即把RouterOS看作交换机，在bridge vlan下配置二层vlan交换，如果二层VLAN需要进行三层处理时，在配置interface vlan与二层VLAN对接，完全类似华为和H3C交换机的interfac vlan id。

根据官方的文档，为大家梳理下Brdige vlan的一些配置要点，下图是一个基本的VLAN实例，一个trunk端口和多access端口：



这样的 VLAN 实例设置是非常常见，通过将交换机网络划分为多个 VLAN，用于分隔多个区域。使用 VLAN，可根据 VLAN ID 划分不同的 IP 段，并创建不同的 DHCP 服务器，还能提供更灵活的防火墙规则和 QoS 策略。

例如 ether2 和 ether3 连接了两个工作区域，需要将两个工作区域隔离开，必须向进入 ether2 或 ether3 的数据帧添加 VLAN tag，需要使用一个称为 Port based VLANs(基于端口的 VLAN)。

在这个实例中，ether1、ether2 和 ether3 加入到 bridge1 下，数据帧会根据连接设备的 Bridge 端口获得带有 VLAN ID 的 tag 标签。例如，ether2 上的设备进入交换机（ingress），将获得带有 VLAN20 的 VLAN 标签，而 ether3 上的设备将获得带有 VLAN30 的 VLAN 标签，当流量从 ether2/ether3 发出后（egress），将删除 VLAN 标签。

确定了未做标签的端口后，现在需要确定打上标签的端口。即 trunk 口(携带多个 vlan 标签的端口)，这个实例中 ether1 就作为 trunk 端口。

PVID 属性需特别说明，这个参数应该使用在 access 端口上，也能同时用于 trunk 端口(称为 Hybrid 的混合模式)。在 Bridge port 的 PVID 参数中指定 VLAN ID，接收的所有不带 VLAN 标签的数据。PVID 对带 tag 的报文不做任何影响，例如在 ether2 的 PVID 为 20，但接收到 VLAN 40 的报文，不会对 VLAN tag 修改，并且会根据 Bridge 的 VLAN 列表，查询 VLAN 40 进行转发。

具体配置，在进行 trunk/access 端口配置前，首先要创建一个 Bridge，配置如下：

```
/interface bridge
add name=bridge1
```

警告：创建 bridge 的时候，VLAN filtering 先不要启用，如果你 RouterOS 调试的管理端口包含在 Bridge 中，会导致 winbox 无法管理 RouterOS（管理端口需要指定相应的 vlan id 通过 trunk 到 CPU 处理）），通常在配置完成后，启用 VLAN filtering。

添加 Bridge port，并指定每个端口的 PVID 值：

```
/interface bridge port
add bridge=bridge1 interface=ether1
add bridge=bridge1 interface=ether2 pvid=20
add bridge=bridge1 interface=ether3 pvid=30
```

注意：PVID 需要在 bridge 的 VLAN filtering 启用后才能生效

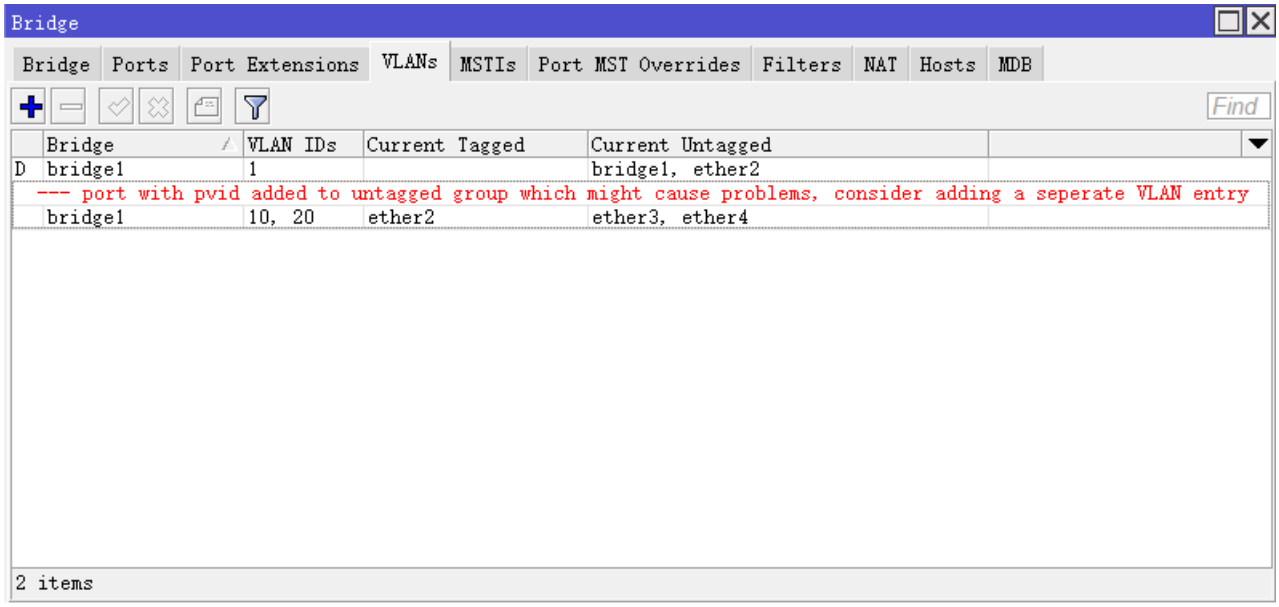
在 Bridge vlan 列表中添加相应的规则：

```
/interface bridge vlan
add bridge=bridge1 tagged=ether1 untagged=ether2 vlan-ids=20
add bridge=bridge1 tagged=ether1 untagged=ether3 vlan-ids=30
```

对于 Bridge 的 trunk 配置，可能会认为可以简化配置，就像下面这样：

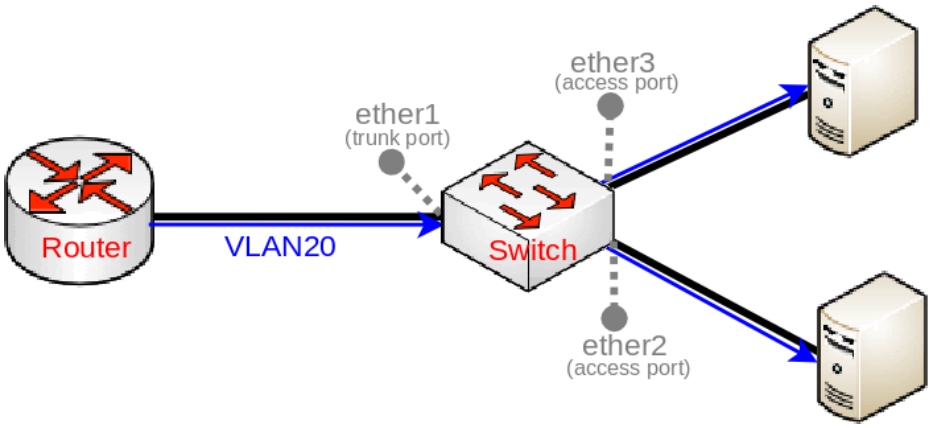
```
/interface bridge vlan
add bridge=bridge1 tagged=ether1 untagged=ether2,ether3 vlan-ids=20,30
```

RouterOS 会用红色的错误警告提示，如下图



这将无意中允许 VLAN20 和 VLAN30 同时出现在 ether2 和 ether3 两个 access 端口上，例如 ether3 在对进入的数据帧插入一个 VLAN30 标签，但当 VLAN 通过该端口发送时，并不限制该端口允许的 VLAN id，VLAN20 也可以从 ether3 发出。

Bridge VLAN 列表负责决定是否允许一个 VLAN 通过特定的端口发送，上面的规则却让 VLAN20 和 VLAN30 都允许通过 ether2 和 ether3 不带 VLAN 标签发送（以 untagged 的形式发送）。因此，造成了 VLAN 间数据穿透问题，见下图：



这样错误的配置，导致本来 ether3 只允许通过 VLAN30，却同时让 VLAN20 通过，而且这样的情况在 ether2 端口上也出现。

最后，启用 vlan filtering， 让 VLAN 配置生效

```
/interface bridge set bridge1 vlan-filtering=yes
```

注意：启用 vlan filtering，如果管理接口包含在 bridge 中，可能会导致 winbox 管理失效，在启用时需要确保配置正确，相关配置参阅后面内容。

三层接口（管理口）配置

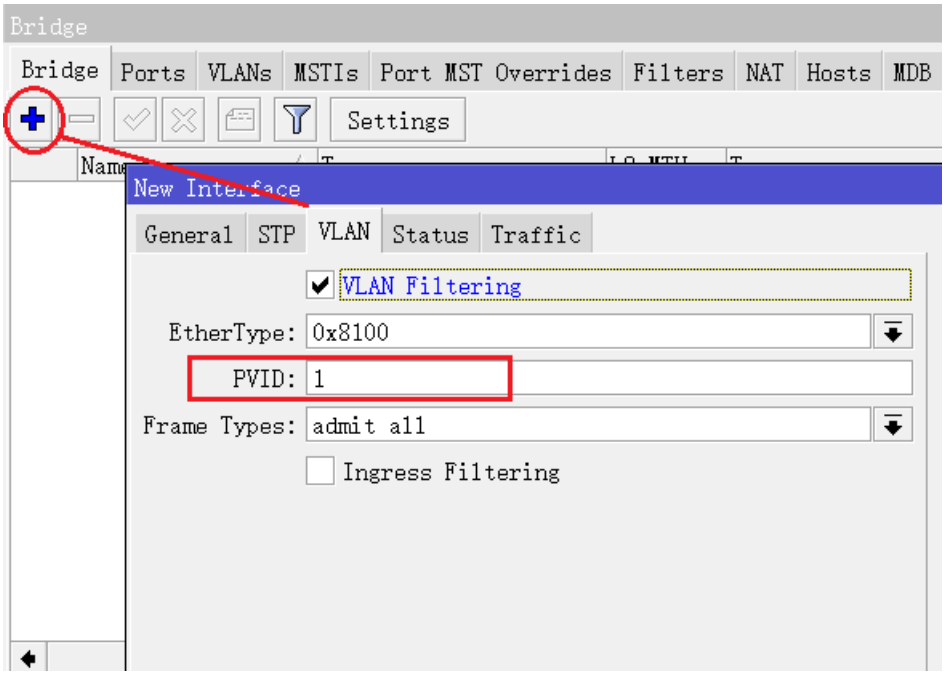
在官方介绍中称为管理口（management），我倾向于称为 VLAN 三层接口，因为将二层的数据发送到 CPU（Bridge 端口）意味着需要进行三层以上的数据处理。

如果你使用 Winbox 通过 MAC 管理交换机，管理接入交换机的端口，正好包含在 Bridge VLAN 中，可能会导致失去对设备的管理。这是因为 RouterOS 将 VLAN 配置下发到了交换机芯片处理，对应的 VLAN 没有发送到运行 RouterOS 系统的 CPU，造成管理中断。

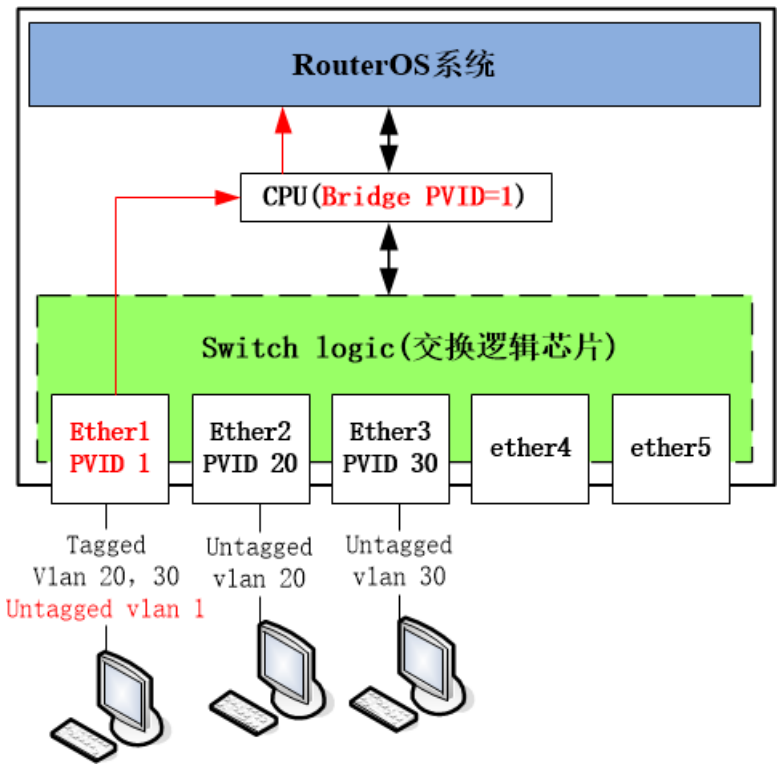
这样的情况可以选择使用 console 控制台通过命令行配置 VLAN，如果要解决 Winbox 管理问题，需要指定接入端口的 VLAN，访问到 CPU（即 bridge 端口），当 vlan filtering 启用后，通过 print 命令查看 vlan 列表状态，可以看到以下信息：

```
[admin@MikroTik] > /interface bridge vlan print
Flags: X - disabled, D - dynamic
 #   BRIDGE                    VLAN-IDS  CURRENT-TAGGED       CURRENT-UNTAGGED
 0   bridge1                   20        ether1               ether2
 1   bridge1                   30        ether1               ether3
 2 D bridge1                   1                              bridge1
                                                              ether1
```

从上面的状态可以看到，bridge1 接口默认 PVID 为 1，会在 VLAN 列表中自动创建 vlan 1，因为创建 Bridge 端口时，PVID=1 是默认设置(包括我们的 trunk 端口 ether1)，端口的 PVID 值是会自动添加到 bridge vlan 列表中



由于该功能，可能会使得 trunk 端口无意中访问到 RouterOS 系统。例如，如果根据默认 bridge1 的 PVID=1，而 ether1 设置为 trunk 端口，对于 VLAN 1 允许使用不带标签的流量通过 ether1 访问设备，通过上面的 Bridge VLAN 列表显示，反应出来的情况如下图：



这样如果管理电脑接入到 ether1 的 trunk 端口，未打上标签的 VLAN 1（untagged），无意间允许 Winbox 能通过 MAC 管理访问， 如果你不允许某些 VLAN 或未打上 VLAN 标签的流量进入 CPU 端口 (bridge1)，一定要检查 Bridge VLAN 列表。

有一种简单的方法可以防止 Bridge(CPU 端口)被添加为一个 untagged 端口，可将 trunk 端口上的 PVID 设置为与 Bridge 的 PVID 不同，但这种方式并非最佳，还有另一种选择对数据帧类型过滤，也更推荐使用。采

用 ingress-filtering 与 frame-type 一起过滤未知的标签，但为了充分理解 ingress-filtering 的行为，首先理解 VLAN 访问到 RouterOS 系统的细节，

Untagged 到 CPU

交换机的三层接入，需在启用 VLAN filtering 后，将二层交换芯片到 CPU 的通道连接，也就是和 RouterOS 系统实际对接。如果指定的 VLAN 未能连接到 CPU，Winbox 的邻居发现协议是无法扫描到 RouterOS，这里必须要理解交换逻辑芯片和 RouterOS 的关系。

假设 ether3 连接到一台电脑，并能够通过 VLAN 访问到 RouterOS 系统，只需将 bridge1 和 ether3 指定为 untagged 端口，如下面的配置实例，查找 vlan-ids=30，并修改 untagged 对应的端口为 bridge1 和 ether3：

```
/interface bridge vlan set [find vlan-ids=30] untagged=bridge1,ether3
```

上面的配置将以 untagged 去掉标签的形式发送到 bridge1（运行 RouterOS 系统的 CPU 端口），这样可以通过 VLAN 30 访问到 RouterOS 系统。

注意：untagged 的 access 端口更适用于用户接入，而如果连接路由器或是 CPU 端口（Bridge），采用 untagged 的 access 口，虽然也能用于三层网络连接，但从扩展和安全方面来说并不推荐，因为采用 tag 的 trunk 连接，更能灵活的扩展用户 VLAN 接入，即使管理 VLAN 被攻击的情况下，trunk 端口更换 VLAN 也非常容易。

Tag 到 CPU

使用 tag 的 trunk 接入三层的管理 VLAN，需要将 vlan 30 透传给 CPU（bridge1），需要做如果下配置

```
/interface bridge vlan
add bridge=bridge1 tagged=bridge1 vlan-ids=30
```

需要注意到，在 RouterOS 使用 VLAN30 的访问还不能直接工作，需要创建一个三层 VLAN 来接收来至交换芯片带 tag 标签数据，这样 winbox 可以通过 mac 登录 RouterOS 系统

```
/interface vlan
add interface=bridge1 name=VLAN30 vlan-id=30
```

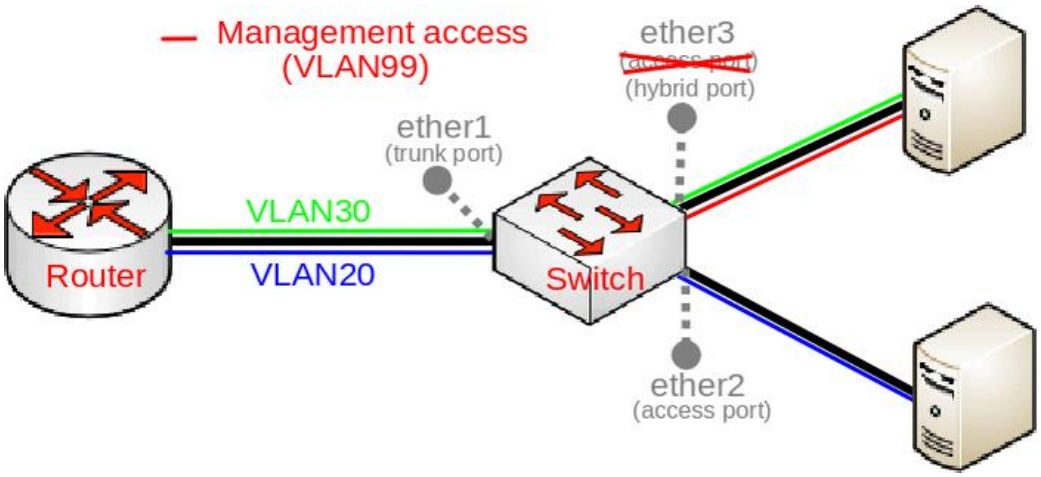
如果需要 IP 管理，可以为该 VLAN 接口设置一个 IP 地址：

```
/ip address
add address=192.168.30.2/24 interface=VLAN30
```

hybrid 配置

如下面的实例，通过 VLAN 99 作为管理端口，为了允许从 ether3 访问使用 VLAN99，需在 Bridge VLAN 列表中，配置一条规则允许 VLAN30 和 VLAN99 同时出现在 ether3，但 vlan 99 采用的是 tag 的 trunk 模式

```
/interface bridge port
add bridge=bridge1 interface=ether3 pvid=30

/interface bridge vlan
add bridge=bridge1 tagged=bridge1,ether3 vlan-ids=99
```



本来 ether3 是一个 access 端口，但现在定义为一个 hybrid 混合端口，并允许 VLAN 99 通过 trunk 到 bridge1，此时在 ether3 下面的电脑需要创建一个 vlan99 的子接口才能访问到 RouterOS

创建一个三层 VLAN 来接收带 tag 标签数据，这样 winbox 可以通过 mac 登录 RouterOS 系统

```
/interface vlan
add interface=bridge1 name=VLAN99 vlan-id=99
```

如果需要 IP 管理，可以为该 VLAN 接口设置一个 IP 地址：

```
/ip address
add address=192.168.99.2/24 interface=VLAN99
```

ingress-filtering 和 frame-type

以上三层 VLAN 配置了解完后，现在我们在来看看 ingress-filtering 和 frame-type，首先，将关注 frame-type，它限制了数据帧的类型(tag 带标签和 untagged 无标签，以及两者都有的情况)，为了让 frame-type 能工作，ingress-filtering 必须启用，否则不会生效。

在上面的实例中，我们允许来至 ether3 的流量打上 VLAN99 的标签，并能访问 VLAN99 网络的其他设备。对于 ether1 和 ether2 接口希望只接收指定的 VLAN tag，保障网络的通信安全。ether1 作为 trunk 端口，仅允许带 tag 标签的数据帧通过，而 ether2 作为 access 端口，只允许不带 tag 标签的数据。根据以上的结论，可以配置 frame-tyes 丢弃不符合要求的数据，配置如下：

```
/interface bridge port
set [find where interface=ether1] ingress-filtering=yes frame-types=admit-only-vlan-tagged
set [find where interface=ether2] ingress-filtering=yes frame-types=admit-only-untagged-and-priority-tagged
```

假设你忘记在 ether1 启用 ingress-filtering 和修改 frame-type 属性，bridge1 的 PVID 和 ether1 接口配置相同的 PVID，这会无意中将该 VLAN 透传到了路由器的 CPU，但实际情况是你只希望 tage 标签的数据才能访问到 CPU，因此配置如下：

```
/interface bridge
set bridge1 frame-types=admit-only-vlan-tagged ingress-filtering=yes
```

这样可以禁止 ether1 的 PVID 动态添加到 Bridge VLAN 列表中。如果通过 print 命令，查看当前的 Bridge VLAN 列表，会注意到 bridge1 不会动态添加 VLAN 1 到当前 untagged 中：

```
[admin@MikroTik] > /interface bridge vlan print
Flags: X - disabled, D - dynamic
 #    BRIDGE      VLAN-IDS  CURRENT-TAGGED       CURRENT-UNTAGGED
 0    bridge1      20        ether1
 1    bridge1      30        ether1                ether3
 2 D bridge1       1                               ether1
 3    bridge1      99        bridge1
                              ether3
```

注意:当 frame-type=admit-only-vlan-tagged ，这时该端口将不会在/bridge vlan 动态添加 untagged 的 PVID，但手动是可以添加 untagged 的 VLAN。

frame-type 可以用于丢弃指定类型的数据帧，ingress-filtering 可以用于在数据帧被发送出去之前过滤掉。为了充分理解 ingress-filtering 的作用，考虑以下场景:ether3 和 bridge1 允许 VLAN99，但是仍然可以将 VLAN99 的流量从 ether1 发送到 ether3，这是因为 bridge VLAN 列表会检查一个端口是否只允许在出口端口携带某个 VLAN 标签 。在例子中，ether3 允许携带 VLAN99 标签的数据被转发，为了防止这种情况发生，必须使用 ingress-filtering。

也可以在 CPU 端口(bridge)上使用 ingress-filtering，这可以用来防止一些可能的攻击，并限制访问 CPU 的 vlan。最好是在入口端口丢弃数据包，而不是在出口端口，当使用二层的 hardware offloading 时，是非常关键的设置，能减少了 CPU 负载。

注意： ingress-filtering 属性只对输入流量有影响，但 frame-type 对输出流量和输入流量都有影响。.

其实整个 trunk/access 主要讨论的问题：

1. 在配置端口为 trunk 时，此端口的 PVID=1 可能存在的问题；
2. 管理口的配置，和 untagged VLAN 进入 CPU（Bridge 端口）问题。

关于问题 1，其实在华为、H3C 和 Cisco 交换机有类似的问题，就是如何避免 VLAN 1 的透传，华为交换机在 trunk 端口的操作是 undo port trunk allowed vlan 1，通常在实际网络中 vlan 1 是被禁止使用的，因为网络结构一旦扩展变大后，很容易造成 VLAN 1 的环路。

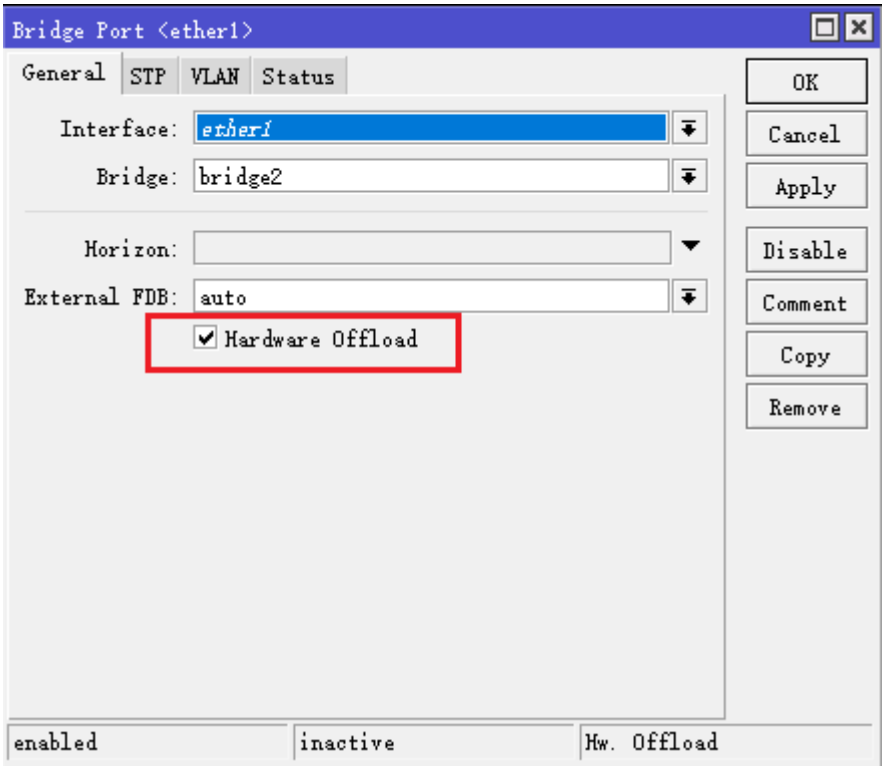
关于问题 2，管理口，其实就是三层接入，在实际网络中，三层接入可以是多个 VLAN 存在，官方的文章主要还是把 RouterOS 交换看作二层使用，需要配置一个管理 IP，CRS3xx 系列的部分型号在 RouterOS v7 支持三层 HW-offloading 的硬件转发。

14.8 Bridge HW-Offloading(v6.41)

从 RouterOS v6.41 版本开始，RouterBOARD 设备的接口 master-port 二层配置操作改变，该配置被迁移到 bridge port 菜单的 Hardware-offloading。hw-offloading 即代表硬件设备的交换芯片提供处理。在 v7beta6 版本开始，新加入了 L3 Hardware-offloading 功能，为三层转发提供硬件加速。

L2 Hardware-offloading 介绍

从 v6.41 该版本开始 RB/CRS 等产品的二层交换配置不再由 Master-Port 完成，而改为 bridge 方式，如果交换芯片支持，即能实现二层交换转发硬件加速。RouterOS switch 菜单其他功能保持不变。默认创建新的 bridge port 时 hw 参数都设置为 yes。如果对该功能无需求，可以选择 hw=no，禁用硬件加速，这样 bridge 的二层交换处理完全交由软件控制，但 x86 平台是完全无法实现 Hardware-offloading 的功能。



注意：如果你要降级 RouterOS 到之前版本，将不会恢复 master-port 的配置，也不会有 hw-offload 功能在 bridge port 出现，因此必须重新配置 master-port 参数。.

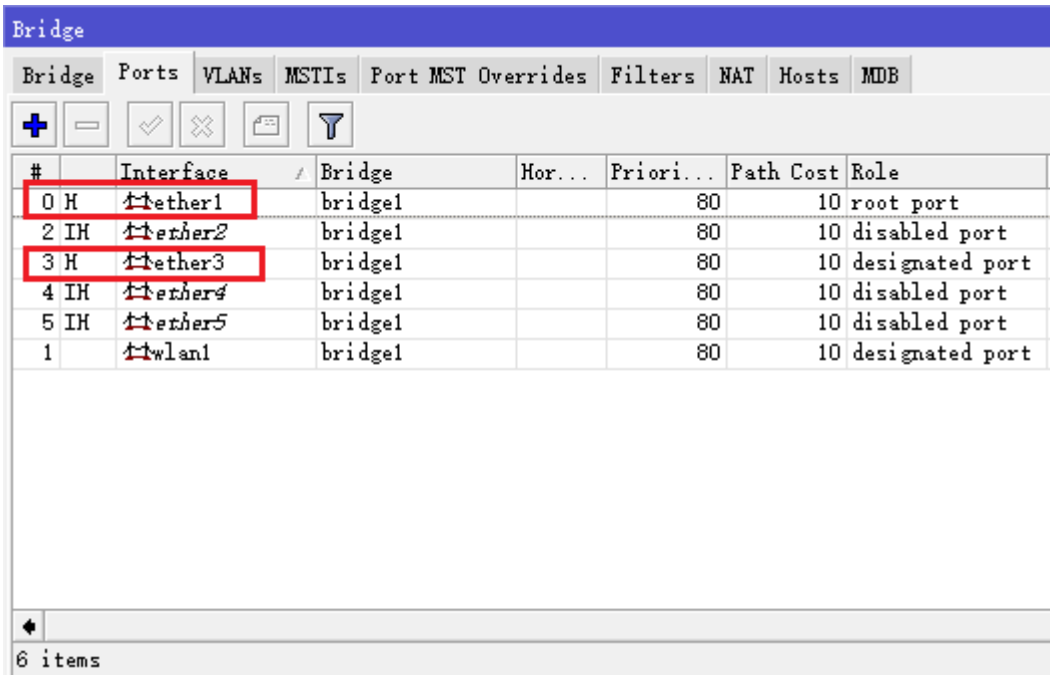
交换配置在 v6.41rc 版本之前，进入 interface master-port 设置

```
[admin@MikroTik] > interface ethernet export
/interface ethernet
set [ find default-name=ether3 ] master-port=ether2
set [ find default-name=ether4 ] master-port=ether2
set [ find default-name=ether5 ] master-port=ether2
[admin@MikroTik] >

[admin@MikroTik] > interface ethernet print
Flags: X - disabled, R - running, S - slave
 #     NAME        MTU   MAC-ADDRESS        ARP        MASTER-PORT        SWITCH
 0 R   ether1    1500 D4:CA:6D:E2:64:64 enabled        none          switch1
 1 R   ether2    1500 D4:CA:6D:E2:64:65 enabled        none          switch1
 2 RS ether3     1500 D4:CA:6D:E2:64:66 enabled        ether2        switch1
 3 RS ether4     1500 D4:CA:6D:E2:64:67 enabled        ether2        switch1
 4 RS ether5     1500 D4:CA:6D:E2:64:68 enabled        ether2        switch1
[admin@MikroTik] >
```

从 v6.41rc 版本开始，交换端口配置在 bridge port，添加接口时默认开启 hw-offload

```
[admin@MikroTik] > interface bridge export
/interface bridge
add name=bridge1 igmp-snooping=no    protocol-mode=none
/interface bridge port
add bridge=bridge1 interface=ether2
add bridge=bridge1 interface=ether3
add bridge=bridge1 interface=ether4
add bridge=bridge1 interface=ether5
[admin@MikroTik] >

[admin@MikroTik] > interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic, H - hw-offload
 #   INTERFACE   BRIDGE   HW PVID PRIORITY   PATH-COST INTERNAL-PATH-COST   HORIZON
 0   H ether2    bridge1  yes     1    0x80         10                 10   none
 1   H ether3    bridge1  yes     1    0x80         10                 10   none
 2   H ether4    bridge1  yes     1    0x80         10                 10   none
 3   H ether5    bridge1  yes     1    0x80         10                 10   none
[admin@MikroTik] >
```

在 winbox 的操作界面，可以看到 ports 菜单下，加入桥接的网卡前缀会出现 H 或 IH，H 代表 Hw.offload 启用硬件加速，IH 表示 inactive Hw.offload，网卡未连接，无法激活硬件加速，而在下图的 wlan1 上没有出现 H 或 IH 代表无线网卡不支持硬件加速：



Bridge hw-offloading 支持的 RouterBOARD 设备和交换芯片型号，以及功能，

+ 代表支持 hw-offloading 加速，
- 代表不支持，开启会关闭 hw-offloading 加速

RB/[交换芯片]	Switch	STP/RSTP	MSTP	IGMP	DHCP	VLAN	Bonding

	路径			Snooping	Snooping	Filtering	
CRS3xx 系列	+	+	+	+	+	+	+
CRS1xx/CRS2xx	+	+	-	+ [2]	+ [1]	-	-
[QCA8337]	+	+	-	-	+ [2]	-	-
[Atheros8327]	+	+	-	-	+ [2]	-	-
[Atheros8227]	+	+	-	-	-	-	-
[Atheros8316]	+	+	-	-	+ [2]	-	-
[Atheros7240]	+	+	-	-	-	-	-
[MT7621]	+	-	-	-	-	-	-
[RTL8367]	+	+	+	-	-	+ [3]	-
[ICPlus175D]	+	-	-	-	-	-	-

1. 该功能在 VLAN 交换设置中无法正常工作。在单个 VLAN 中 snoop DHCP 报文是可能的，但这需要使用 ACL 规则给这些 DHCP 报文打上正确的 VLAN 标签，例如：

   /interface ethernet switch acl add dst-l3-port=67-68 ip-protocol=udp mac-protocol=ip new-customer-vid=10 src-ports=switch1-cpu

2. 该功能在 VLAN 交换设置中无法正常工作
3. HW vlan-filtering 功能在 RouterOS 7.1rc1 开始支持，但不支持 ether-type 为 0x88a8 和 0x910，仅支持 0x8100，并不能使用 QinQ 的 tag-stacking，开启此功能会导致 HW offload 关闭

在上面表格所示，CRS3xx 系列所有功能都支持，使用 RTL8367 的 RB4011 和 RB1100AHx4 在 v7.1rc1 后获得 HW vlan-filtering 支持。其他型号如果你要使用 VLAN 配置，请在这些 RB 设备上选择 **Switch** 菜单操作。

L3 Hardware Offloading 介绍

L3 Hardware Offloading（L3-HW），又可以称为三层 IP 硬件卸载或硬件路由加速 ，允许将三层路由转发卸载给交换芯片，能实现三层路由线速转发，不再消耗 CPU。

从 RouterOS v7.0 beta6 开始支持 Layer3/4 hardware offloading，而 RouterOS v7.1beta4 开始 l3hw=yes 和 l3hw=fw 两个模式进行了合并，允许单独指定接口的路由硬件转发，并兼容部分 Firewall/NAT 配置，将 l3hw 属性被重命名为 l3-hw-offloading。

```
/interface/ethernet/switch set switch1 l3-hw-offloading=yes
/interface/ethernet/switch/port set [find] l3-hw-offloading=yes
```

现在 L3-HW 能分别配置到每个物理交换接口，当启用 **l3-hw-offloading** 设置更改配置，不影响正在使用的会话连接。如果禁用 **l3-hw- offloading** 仍然会有路由条目在硬件内存中保留，因此建议修改 **l3-hw-offloading** 配置后重启设备。

v7.1 beta4 调整内容如下：

L3-HW 设置同时存在于/interface/ethernet/switch 和/interface/ethernet/switch/port 下，但它们有所不同， 在 switch 下设置 **l3-hw- offloading=no** ，将完全禁用硬件转发，所有数据包将通过 CPU 处理。但在 port 下设置 **l3-hw- offloading=no**，将禁用该指定接口的三层路由转发，而 **fasttrack** 将继续下发到硬件处理 。但需要注意，数据包在路由转发下发到硬件时，要求源和目标端口同时设置

l3-hw-offloading=yes，如果仅一侧启用，数据将会通过 CPU 处理，仅 Fasttrack 的连接会被下发到硬件。

默认情况下，所有路由条目都启用了 HW 硬件加速。为了进一步调整需要硬件处理的流量，每条路由都有一个选项 suppress-hw-offload（压制 hw-offload）来禁用/启用，是否进行硬件转发。

例如，当我们知道某条路由涉及服务器网络的流量非常大，我们可以只启用到该特定静态路由硬件加速：

```
/ip/route set [find where static && dst-address!="192.168.3.0/24"] suppress-hw-offload=yes
```

仅目标静态路由 192.168.3.0/24 启用硬件加速（注意规则是 dst-address!=”192.168.3.0/24″，有“！” 代表非），其他路由开启压制，可以在下面看到 AsH 前缀，表明该路由条目选择 HW-offloading 的硬件加速（H-flag 并非表示该路由实际上已经在 HW-offloading，而是表示该路由可被选择为 HW-offloading）

```
[admin@MikroTik] > /ip route/print where static
Flags: A - ACTIVE; s - STATIC, y - COPY; H - HW-OFFLOADED
Columns: DST-ADDRESS, GATEWAY, DISTANCE
#      DST-ADDRESS     GATEWAY          D
0 As   0.0.0.0/0       172.16.2.1        1
1 As   10.0.0.0/8       10.155.121.254   1
2 AsH 192.168.3.0/24    172.16.2.1       1
```

如果只启用 fasttrack 功能，请在 port 上禁用 l3-hw-offloading 功能，而在 switch 上启用全局 l3-hw-offloading 功能。

```
/interface/ethernet/switch set switch1 l3-hw-offloading=yes
/interface/ethernet/switch/port set [find] l3-hw-offloading=no
```

另一个例子是，当 sfp-sfpplus16 作为外网 WAN 端口时，需要开启 NAT 转换，但其他端口作为内网开启 L3-HW 加速，进出 sfp-sfpplus16 的数据包会进入 CPU 处理，即被防火墙/NAT 处理。

```
/interface/ethernet/switch set switch1 l3-hw-offloading=yes
/interface/ethernet/switch/port set [find] l3-hw-offloading=yes
/interface/ethernet/switch/port set sfp-sfpplus16 l3-hw-offloading=no
```

需要注意 CRS3xx 系列交换机的 L3-HW 在 v7.1 beta6 后开始被完全支持，但需要区分 Marvell DX8000 和 DX3000/2000 系列芯片，它们之间的功能有所区别。

DX8000 系列芯片

下列设备基于 Marvell 98DX8xxx 或 98DX3257 交换芯片

型号	版本	路由条目	下一跳	ECMP	Fasttrack [1,2]	NAT
CRS317-1G-16S+	7.1beta1	150 - 240K	8K	4K	4500 / 3750 [3]	8K [4]
CRS309-1G-8S+	7.1beta2	50 - 80K	8K	4K	4500 / 3750	8K
CRS312-4C+8XG	7.1beta2	50 - 80K	8K	4K	2250 / 1500	8K
CRS326-24S+2Q+	7.1beta2	50 - 80K	8K	4K	2250 / 1500	8K
CRS354-48G-4S+2Q+	7.1beta4	50 - 80K	8K	4K	2250 / 1500	8K

备注说明：

- *1 当达到 Fasttrack 或 NAT 的 HW 会话限制时，超出的会话连接将回落到 CPU，MikroTik 的智能连接算法确保了流量最大的连接被卸载到硬件处理
- *2 Fasttrack 连接与 ACL 规则共用 HW 内存。根据复杂度的不同，一条 ACL 规则可能会占用 3-6 条 Fasttrack 连接的内存。
- *3 (MPLS 和 Bridge Port Extender 同时被禁用) / (MPLS 和 Bridge Port Extender 两者都启用)。MPLS 与 Fasttrack 连接共用 HW 内存。此外，启用 MPLS 需要分配整个内存区域，否则可以存储多达 750 个 Fasttrack 连接。这同样适用于 Bridge Port Extender（桥接端口扩展器）。然而，MPLS 和 Bridge Port Extended 可能使用相同的内存区域，所以同时启用它们不会使 Fasttrack 连接的限制加倍。
- *4 如果 Fasttrack 启用，NAT 条目无法被全部使用因为受限于 Fasttrack 连接数量。

DX3000/2000 系列

下面的设备列表为 Marvell 98DX224S、98DX226S 和 98DX3236 交换芯片，这些芯片不支持 Fasttrack 和 NAT 连接下发，而 98DX3257 除外，其功能与 DX8000 相同，所以有别于 DX3000 系列。

这类交换芯片使用不同的方法下发 IPv4 路由，而不是 DX8000 系列芯片使用的 IP rangs，DX3000/2000 使用的 IP prefixs

型号	版本	IPv4 路由前缀 [1]	下一跳	ECMP per prefix [2]
CRS305-1G-4S+	7.1beta6	13312	4K	8
CRS318-1Fi-15Fr-2S	7.1beta6	13312	4K	8
CRS318-16P-2S+	7.1beta6	13312	4K	8
CRS326-24G-2S+	7.1beta6	13312	4K	8
CRS328-24P-4S+	7.1beta6	13312	4K	8
CRS328-4C-20S-4S+	7.1beta6	13312	4K	8

- *1 可用于加速的路由总量非常有限，更高的子网掩码会被优先做硬件转发(如：/32，/30，/29 等高位子网，最后才是 0.0.0.0/0，如果超出限制数量 0.0.0.0/0 的默认路由会被放到 CPU 处理，对于其他任何前缀不适合 HW 表将由 CPU 处理。

- *2 如果到相同目的路由，存在多路径，即 ECMP 路由，超过限制数量(8 条)，则只加速第一组 8 条 EMCP 路由，多出的会由 CPU 处理。

使用 interface list

如果交换机的接口较多，需要配置 l3-hw-offloading，每次调整比较麻烦，可用通过 interface list 统一管理，但有些接口列表中可能包含虚拟接口(如 VLAN)，而 l3-hw-offloading 设置只能应用于交换机的物理端口

需要注意 interface list 只是用作脚本的端口选择。以下示例演示了如何在 LAN 端口(属于"LAN"接口列表的端口)上启用硬件路由，并在 WAN 端口上禁用硬件路由：

```
:foreach i in=[/interface/list/member/find where list=LAN] do={
/interface/ethernet/switch/port set [/interface/list/member/get $i interface] l3-hw-offloading=yes
}

:foreach i in=[/interface/list/member/find where list=WAN] do={
/interface/ethernet/switch/port set [/interface/list/member/get $i interface] l3-hw-offloading=no
}
```

特别注意：interface list 并不是直接用于 L3-HW 的控制，仅仅是脚本借助 interface list 去完成 L3-HW 的接口修改。例如，将 CRS3xx 交换机的端口添加到"LAN"接口列表中，并不会自动在该端口上启用 l3-hw-offloading，需要管理员运行上面的脚本来应用更改。

2021 年 5 月功能支持情况列表：

功能	状态	描述
IPv4 单播	HW	根据路由表中路由的复杂度，最大 HW 加速路由数可在 150-250k 之间，对于 IP 子网/8，/16，/24 等要比其他子网（如/22）占用更少的 HW 空间。当到达 HW 路由限制时，新的路由将交给 CPU 处理，除非添加的路由与已经在处理的路由重叠。
IPv6 单播	CPU	
IPv4 多播	CPU	
IPv6 多播	CPU	
ECMP 等价路由	HW	参考不同设备参数
"blackhole"路由	HW	通过配置黑洞路由，允许以线速处理 DDOS 攻击
"prohibit" 路由	CPU	
"unreachable" 路由	CPU	
gateway=<interface>	HW/CPU	这只适用于直接连接的网络。由于 HW 不知道如何发送 ARP 请求，由 CPU 发送 ARP 请求，并等待一个答复，当接收到第一个匹配的目标 IP 数据包后，从中找出目标 MAC 地址，在目标 MAC 确定后，添加 HW 条目，这样实现接口作为网关的数据包由交换机芯片处理。
Bridge	HW	从 bridge 接口方向的路由处理
VLAN	HW	从 VLAN 接口方向的路由处理

功能	状态	描述
LACP/bonding	HW	/interface bonding
Firewall	FW	仅 Fasttrack 连接会话能被 HW 处理，意味着三层路由数据包由 CPU 处理，意味着 HW 不参与三层包转发，只处理 Fasttrack 连接
NAT	FW	应用于 Fasttrack 的 NAT 规则，会被 HW 处理。
QoS	N/A	

描述说明：

- **CPU** – 此功能仅支持 CPU 处理
- **HW** – 此功能支持 L3 HW offloading
- **FW** - 此功能当在 switch 全局下 l3-hw-offloading=yes 开启 L4 加速
- **N/A** – 当前不支持，如果开启，将会禁用 L3 HW offloading 功能运行

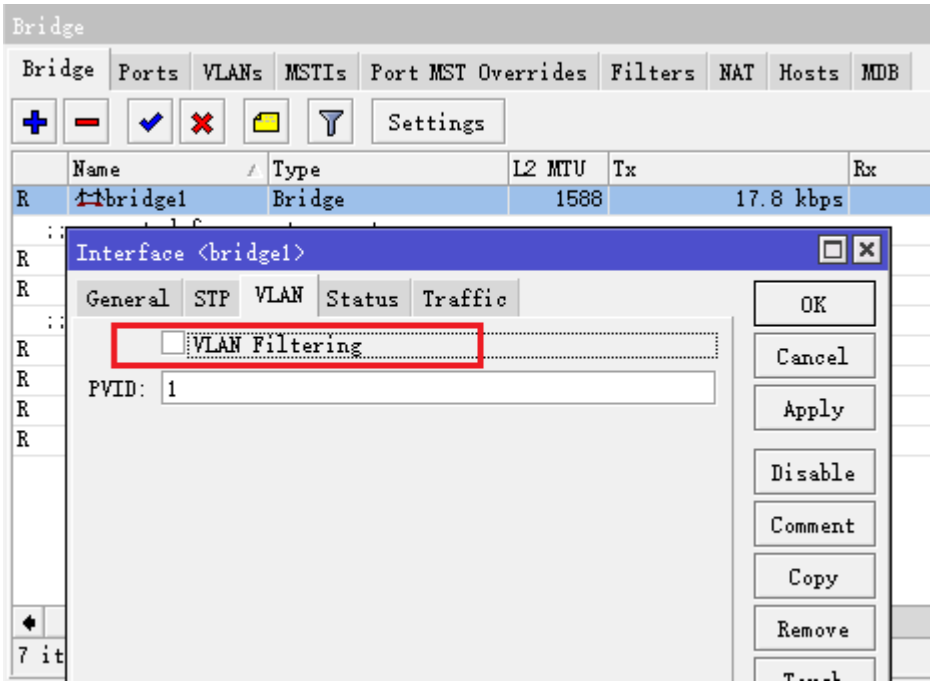
14.9 Bridge VLAN 过滤(v6.41)

操作路径: /interface bridge

创建 Bridge 时，可以选择 VLAN 的设置， **vlan-filtering** 是用于桥接下全局的 VLAN 识别和 VLAN tags 标记处理。如果 vlan-filtering=no, Bridge 将忽略 VLAN tag 标记。只能工作在 shared-VLAN-learning (SVL) 模式，也不能修改 VLAN 数据包的 tags。设置为 yes，相关的 VLAN 功能和 independent-VLAN-learning (IVL) 模式将启用。当 Bridge 加入端口，实现二层转发，bridge 自身也将会成为一个接口，并拥有自己的 PVID（Port VLAN ID）。

属性	描述
vlan-filtering (*yes* | *no*; 默认:**no**)	启用和禁用 bridge 下的 VLAN 功能
pvid (*1..4094*; 默认:**1**)	指定 Bridge 接口下的 Port VLAN ID (pvid) ，用于分配 VLAN 无标记入口

在 winbox 中创建一个 bridge1，可以在 vlan 菜单下找到 vlan-filtering 选项



在 6.41 版本后，设置 Bridge port 下新增了相关 vlan 过滤参数：

属性	描述
frame-types (*admit-all* | *admit-only-untagged-and-priority-tagged* | *admit-only-vlan-tagged*; 默认:**admit-all**)	指定一个 Bridge 端口准许进入的针类型
ingress-filtering (*yes* | *no*; 默认:**no**)	启用或禁用过滤查找一个入口在 Bridge VLAN 表中的匹配
pvid (*1..4094*; 默认:**1**)	指定 Bridge 接口下的 Port VLAN ID (pvid) ，用于分配 VLAN 无标记入口

操作路径: /interface bridge vlan

Bridge 的 VLAN 表是对每个 VLAN 端口的映射关系做出方向的 VLAN 标记处理。Trunk（tagged）接口标记学习到的 VLAN ID 发送出去，而 access（untagged）接口则在发送前删除 VLAN ID 标记

属性	描述
bridge (*名称*)	VLAN 配置属于那个一个 Bridge 接口下
disabled (*yes* | *no*; 默认:**no**)	启用或禁用 Bridge VLAN
tagged (*接口*; 默认:**none**)	设置 Trunk 的接口列表，用于选择 VLAN 标记（tagged）出方向的操作，配置可以多接口例如：`tagged=ether1,ether2`.
untagged (*interfaces*; Default:**none**)	设置 access 的接口列表，用于选择删除出方向的 VLAN 标记（untagged），配置可以多接口例如：`tagged=ether3,ether4`.
vlan-ids (*1..4094*)	VLAN ID 列表，设置确定的 VLAN ID 范围或通过逗号隔离相互独立的 VLAN，例如：

	`vlan-ids=100-115,120,122,128-130`.

VLAN -ids 参数可用于指定 VLAN 的集合或范围，但在单个 Bridge VLAN 表条目中指定多个 VLAN 应该仅用于 trunk 端口。如果为 access 端口指定了多个 vlan，那么不管 PVID 值如何，标记的包可能作为未标记包通过错误的 access 端口发送出去。

操作路径: /interface bridge host

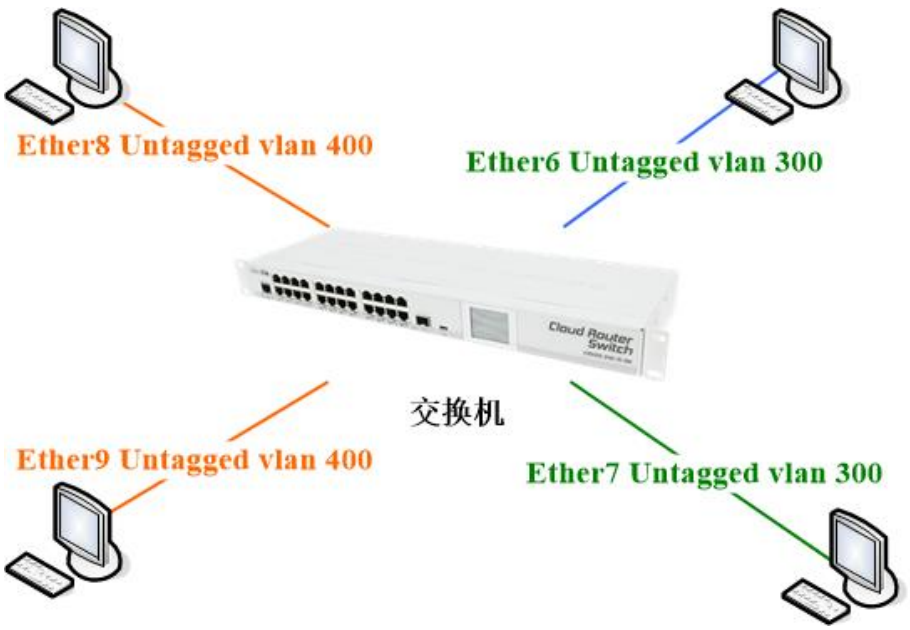
Bridge host 是主机列表，可以监测学习到的 MAC 地址，当 vlan-filtering 启用，会显示学习到的 VLAN ID

```
[admin@MikroTik] > interface bridge host print where !local
Flags: L - local, E - external-fdb
  BRIDGE                VID  MAC-ADDRESS       ON-INTERFACE                AGE
  bridge1               200  D4:CA:6D:77:2E:F0  ether3                       7s
  bridge1               200  E4:8D:8C:1B:05:F0  ether2                       2s
  bridge1               300  D4:CA:6D:74:65:9D  ether4                       3s
  bridge1               300  E4:8D:8C:1B:05:F0  ether2                       2s
  bridge1               400  4C:5E:0C:4B:89:5C  ether5                       0s
  bridge1               400  E4:8D:8C:1B:05:F0  ether2                       0s
[admin@MikroTik] >
```

注意：在使用 Bridge VLAN Filtering 时，请确保将所有需要的二层端口添加到 Bridge Port 表中，如果某些 VLAN 需要开启三层路由功能，需要从这些 VLAN 端口设置允许访问 CPU，操作可以参考管理端口部分

VLAN 实例 1 (不同 VLAN 配置 Access 接口)

这个实例介绍没有 trunk 端口的情况下，常见二层不同 VLAN 的 access 接口配置，



创建一个 bridge 接口，并禁用 vlan-filtering ，在配置完成 VLAN 之前，避免丢失路由器的管理

```
/interface bridge
```

```
add name=bridge1 vlan-filtering=no
```

添加 bridge ports 的接口成员，并指定每个 access 接口的 PVID。

```
/interface bridge port
add bridge=bridge1 interface=ether2
add bridge=bridge1 interface=ether6 pvid=300 frame-types=admit-only-untagged-and-priority-tagged \
    ingress-filtering=yes
add bridge=bridge1 interface=ether7 pvid=300 frame-types=admit-only-untagged-and-priority-tagged \
    ingress-filtering=yes
add bridge=bridge1 interface=ether8 pvid=400 frame-types=admit-only-untagged-and-priority-tagged \
    ingress-filtering=yes
add bridge=bridge1 interface=ether9 pvid=400 frame-types=admit-only-untagged-and-priority-tagged \
    ingress-filtering=yes
```

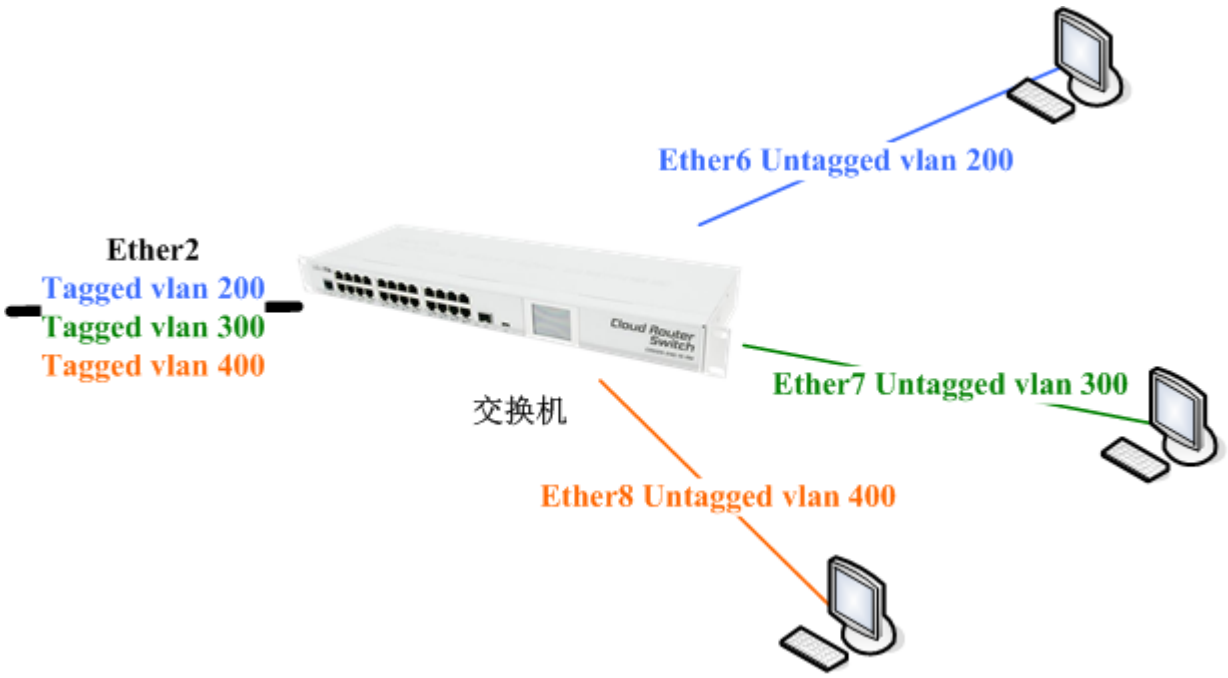
以上配置完成后，RouterOS 会在 bridge vlan 下自动生成端口和 vlan id 对应关系

最后，当 VLAN 配置完成，启用 vlan-filtering

```
/interface bridge set bridge1 vlan-filtering=yes
```

VLAN 实例 2 (配置 Trunk 和 Access 接口)

这个实例介绍常见二层 VLAN 的 trunk 和 access 接口配置



创建一个 bridge 接口，并禁用 vlan-filtering ，在配置完成 VLAN 之前，避免丢失路由器的管理

```
/interface bridge
add name=bridge1 vlan-filtering=no
```

添加 bridge ports 的接口成员，并指定每个 access 接口的 PVID，trunk 接口不用指定。

```
/interface bridge port
add bridge=bridge1 interface=ether2 frame-types=admit-only-vlan-tagged ingress-filtering=yes
add bridge=bridge1 interface=ether6 pvid=200 frame-types=admit-only-untagged-and-priority-tagged \
    ingress-filtering=yes
add bridge=bridge1 interface=ether7 pvid=300 frame-types=admit-only-untagged-and-priority-tagged \
    ingress-filtering=yes
add bridge=bridge1 interface=ether8 pvid=400 frame-types=admit-only-untagged-and-priority-tagged \
    ingress-filtering=yes
```

添加 bridge VLAN，并指定 tagged 和 untagged 接口

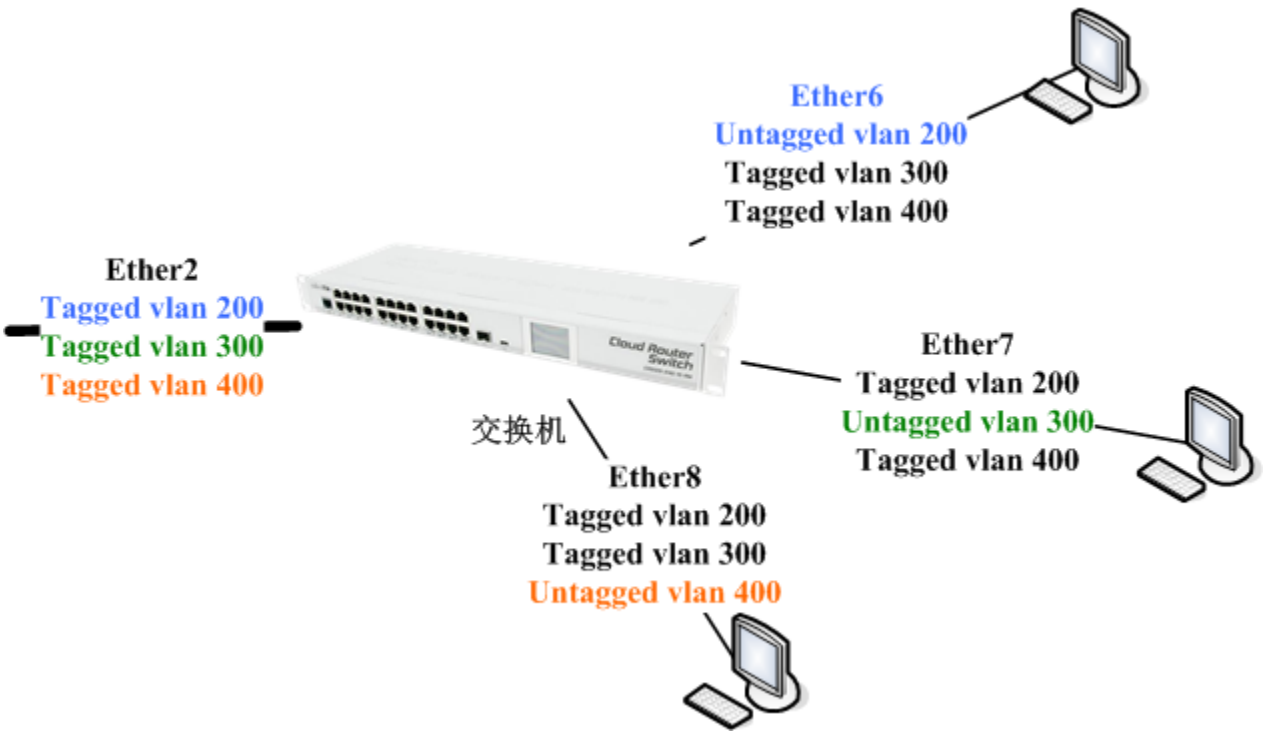
```
/interface bridge vlan
add bridge=bridge1 tagged=ether2 untagged=ether6 vlan-ids=200
add bridge=bridge1 tagged=ether2 untagged=ether7 vlan-ids=300
add bridge=bridge1 tagged=ether2 untagged=ether8 vlan-ids=400
```

最后，当 VLAN 配置完成，启用 vlan-filtering

```
/interface bridge set bridge1 vlan-filtering=yes
```

VLAN 实例 3 (配置 Trunk 和 Hybrid 接口)

该实例介绍二层 VLAN 混杂模式的配置



创建一个 bridge 接口，并禁用 vlan-filtering ，在配置完成 VLAN 之前，避免丢失路由器的管理

```
/interface bridge
```

```
add name=bridge1 vlan-filtering=no
```

添加 bridge ports 的接口成员，并指定每个 access 接口的 PVID，trunk 接口不用指定。

```
/interface bridge port
add bridge=bridge1 interface=ether2 frame-types=admit-only-vlan-tagged ingress-filtering=yes
add bridge=bridge1 interface=ether6 pvid=200 frame-types=admit-only-untagged-and-priority-tagged \
    ingress-filtering=yes
add bridge=bridge1 interface=ether7 pvid=300 frame-types=admit-only-untagged-and-priority-tagged \
    ingress-filtering=yes
add bridge=bridge1 interface=ether8 pvid=400 frame-types=admit-only-untagged-and-priority-tagged \
    ingress-filtering=yes
```

添加 Bridge VLAN，并指定 tagged 和 untagged 端口，设置 ether6.ether7 和 ether8 为 Hybrid 端口

```
/interface bridge vlan
add bridge=bridge1 tagged=ether2,ether7,ether8 untagged=ether6 vlan-ids=200
add bridge=bridge1 tagged=ether2,ether6,ether8 untagged=ether7 vlan-ids=300
add bridge=bridge1 tagged=ether2,ether6,ether7 untagged=ether8 vlan-ids=400
```

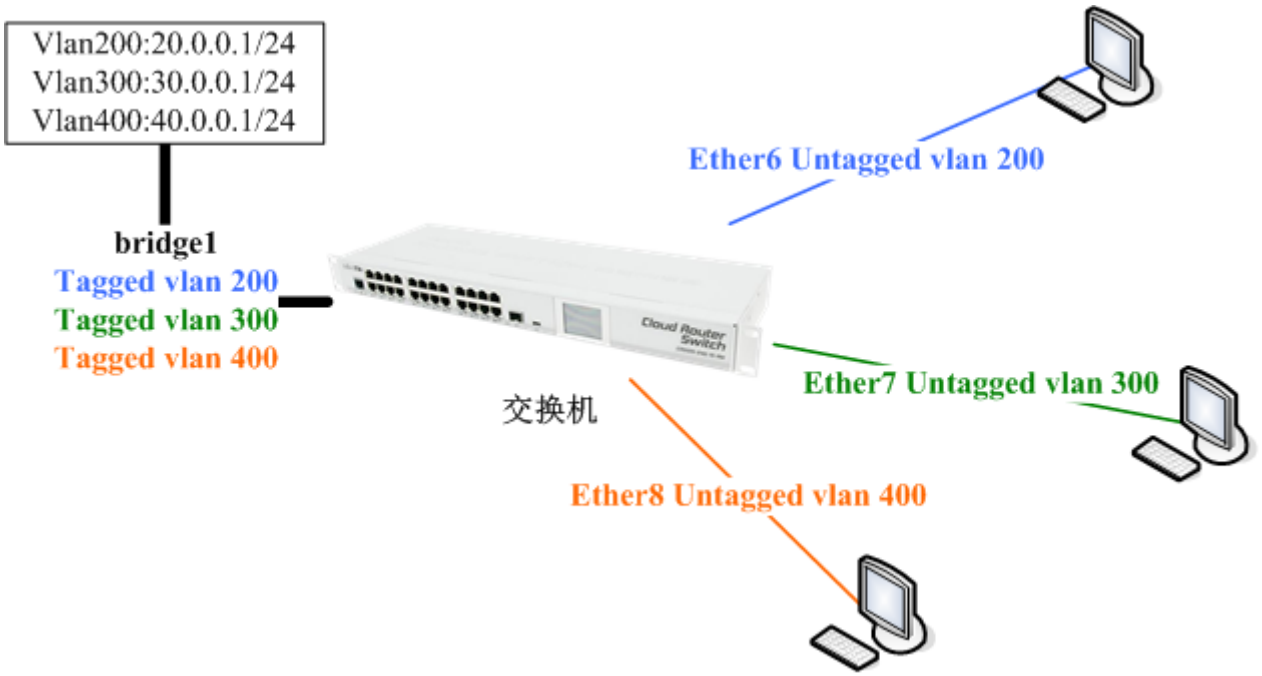
最后，当 VLAN 配置完成，启用 vlan-filtering

```
/interface bridge set bridge1 vlan-filtering=yes
```

注意：在 bridge vlan 下无需为 access 端口指定 untagged 值，在 bridge port 下的端口指定 PVID 后会自动创建 VLAN 下添加 untagged 的 access 端口，仅需指定 tagged 端口。自动创建时具有相同 PVID 值的所有端口将作为 untaged 添加到单个条目中的情况发生，这里必须考虑 bridge 本身也是一个端口，因为 bridge 也有一个 PVID 值，这也意味 brdige 端口将被添加到相同 PVID 的 untagged 端口，可以通过在端口上设置不同的 PVID（包括 trunk 端口和 bridge 本身），或者将帧类型（frame-type）设置为 admit-only-vlan-tagged 来避免这样的情况。

VLAN 实例 4(配置 VLAN 三层接口)

该实例介绍，如何二层 VLAN 上创建三层接口



创建一个 bridge 接口，并禁用 vlan-filtering ，在配置完成 VLAN 之前，避免丢失路由器的管理

```
/interface bridge
add name=bridge1 vlan-filtering=no
```

添加 bridge ports 的接口成员，并指定每个 access 接口的 PVID，trunk 接口不用指定。

```
/interface bridge port
add bridge=bridge1 interface=ether6 pvid=200 frame-types=admit-only-untagged-and-priority-tagged \
    ingress-filtering=yes
add bridge=bridge1 interface=ether7 pvid=300 frame-types=admit-only-untagged-and-priority-tagged \
    ingress-filtering=yes
add bridge=bridge1 interface=ether8 pvid=400 frame-types=admit-only-untagged-and-priority-tagged \
    ingress-filtering=yes
```

添加 bridge VLAN，并指定 tagged 和 untagged 接口，在这个实例中设置 tagged 为 bridge1，即将 VLAN trunk 到路由器内部

```
/interface bridge vlan
add bridge=bridge1 tagged=bridge1 untagged=ether6 vlan-ids=200
add bridge=bridge1 tagged=bridge1 untagged=ether7 vlan-ids=300
add bridge=bridge1 tagged=bridge1 untagged=ether8 vlan-ids=400
```

进入/interface vlan 菜单，创建 bridge1 接口下对应的 vlan id，允许处理来之二层的 vlan 数据

```
/interface vlan
add interface=bridge1 name=vlan200 vlan-id=200
add interface=bridge1 name=vlan300 vlan-id=300
add interface=bridge1 name=vlan400 vlan-id=400
```

在/ip address 菜单下添加对应 VLAN 的 IP 地址

```
/ip address
add address=20.0.0.1/24 interface=vlan200 network=20.0.0.0
add address=30.0.0.1/24 interface=vlan300 network=30.0.0.0
add address=40.0.0.1/24 interface=vlan400 network=40.0.0.0
```

最后，当 VLAN 配置完成，启用 vlan-filtering

```
/interface bridge set bridge1 vlan-filtering=yes frame-types=admit-only-vlan-tagged ingress-filtering=yes
```

VLAN 实例 5 (配置 VLAN 三层管理 IP)

该实例如何是二层 VLAN 配置三层管理 IP 和网关

创建一个 bridge 接口，并禁用 vlan-filtering ，在配置完成 VLAN 之前，避免丢失路由器的管理

```
/interface bridge
add name=bridge1 vlan-filtering=no
```

添加 bridge ports 的接口成员，并指定每个 access 接口的 PVID，trunk 接口不用指定。

```
/interface bridge port
add bridge=bridge1 interface=ether2 frame-types=admit-only-vlan-tagged ingress-filtering=yes
add bridge=bridge1 interface=ether3 pvid=300 frame-types=admit-only-untagged-and-priority-tagged \
    ingress-filtering=yes
```

添加 bridge VLAN，并指定 tagged 和 untagged 接口，在这个实例中设置 tagged 为 bridge1，即将 VLAN trunk 到路由器内部

```
/interface bridge vlan
add bridge=bridge1 tagged=ether2,bridge1 vlan-ids=200
add bridge=bridge1 tagged=ether2 untagged=ether3 vlan-ids=300
```

进入/interface vlan 菜单，创建 bridge1 接口下对应的 vlan id，允许处理来之二层的 vlan 数据

```
/interface vlan
```

```
add interface=bridge1 name=vlan200 vlan-id=200
```

在/ip address 菜单下添加对应 VLAN 的 IP 地址

```
/ip address
add address=20.0.0.2/24 interface=vlan200
```

在/ip route 菜单下添加对应网关

```
/ip route
add gateway=20.0.0.1
```

最后，当 VLAN 配置完成，启用 vlan-filtering

```
/interface bridge set bridge1 vlan-filtering=yes frame-types=admit-only-vlan-tagged ingress-filtering=yes
```

注意：配置 bridge 下启用 vlan-filtering，并在端口下配置 vlan id 后，无法通过 winbox 扫描到 RouterOS，Winbox 扫描 RouterOS 设备需要识别 mac 地址，需要通过邻居发现协议进行扫描，但二层的 vlan 配置，并未将数据发送到 RouterOS 系统，只是在交换芯片进行 vlan 转发处理，所以 winbox 无法通过二层扫描到 RouterOS 系统，解决这个问题，需要将该 VLAN id 传递给 RouterOS 系统，也就是传递给 CPU 处理。操作和配置三层管理 IP 一样

如果你仅仅是通过二层协议（如 MAC-telnet）连接 CRS 设备，那可以跳过添加 IP 地址的步骤，仅把 VLAN 传递给 CPU。

14.10 不同 MikroTik 硬件的二层 VLAN 配置

大多的 MikroTik 硬件设备都集成交换芯片，因此可以实现硬件级的二层转发，在这些硬件中部分可以实现线速的二层 VLAN 交换，但不同硬件配置又有所不同，下面介绍下不同硬件的二层交换配置 ：

首先通过一个案例，来对比不同交换机的配置，ether1 作为 trunk 接口透传 vlan20、vlan30 和 vlan99，ether2 和 ether3 作为 access 接口，vlan99 作为管理口配置 192.168.99.1/24 的管理 IP 地址。

提示：以下配置都是基于 RouterOS v6.41 后的版本，该版本后对 bridge 做了大改动。

CRS3xx 系列

```
/interface bridge
add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether1 hw=yes
add bridge=bridge1 interface=ether2 hw=yes pvid=20
add bridge=bridge1 interface=ether3 hw=yes pvid=30
/interface bridge vlan
add bridge=bridge1 tagged=ether1 untagged=ether2,ether3 vlan-ids=20,30
add bridge=bridge1 tagged=ether1,bridge1 vlan-ids=99
```

```
/interface vlan
add interface=bridge1 vlan-id=99 name=MGMT
/ip address
add address=192.168.99.1/24 interface=MGMT
/interface bridge
set bridge1 vlan-filtering=yes
```

提示：CRS3xx 系列交换机介绍单独整理了一份教程，可以下载 **MikroTik CRS3xx 系列操作配置手册**

CRS1xx/CRS2xx 系列

由于 CRS1 系列和 CRS2 系列芯片问题，因此需要通过 switch 菜单下完成部分配置

```
/interface bridge
add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether1 hw=yes
add bridge=bridge1 interface=ether2 hw=yes
add bridge=bridge1 interface=ether3 hw=yes
/interface ethernet switch ingress-vlan-translation
add ports=ether2 customer-vid=0 new-customer-vid=20 sa-learning=yes
add ports=ether3 customer-vid=0 new-customer-vid=30 sa-learning=yes
/interface ethernet switch egress-vlan-tag
add tagged-ports=ether1 vlan-id=20
add tagged-ports=ether1 vlan-id=30
add tagged-ports=ether1,switch1-cpu vlan-id=99
/interface ethernet switch vlan
add ports=ether1,ether2 vlan-id=20 learn=yes
add ports=ether1,ether3 vlan-id=30 learn=yes
add ports=ether1,switch1-cpu vlan-id=99 learn=yes
/interface vlan
add interface=bridge1 vlan-id=99 name=MGMT
/ip address
add address=192.168.99.1/24 interface=MGMT
/interface ethernet switch
set drop-if-invalid-or-src-port-not-member-of-vlan-on-ports=ether1,ether2,ether3
```

其他集成交换芯片设备

警告： 不是所有 MikroTik 硬件设备都支持 VLAN 的硬件级转发，具体的支持列表可以参见 https://wiki.mikrotik.com/wiki/Manual:Switch_Chip_Features#Introduction ，如果设备支持 VLAN table，即可以通过集成芯片处理 VLAN 转发，查看你硬件的交换芯片命令通过**/interface ethernet switch print**

下面是查看 RB750Gr3 的交换芯片型号：

```
[admin@MikroTik] > /interface ethernet switch print
Flags: I - invalid
 #    NAME      TYPE              MIRROR-SOURCE     MIRROR-TARGET     SWITCH-ALL-PORTS
 0    switch1   MediaTek-MT7621   none              none
```

下面的配置可以应用于 RouterBOARD 系列，包括 RB4xx, RB9xx, RB2011, RB3011, RB4011,hAP, hEX, cAP 等

```
/interface bridge
add name=bridge1 protocol-mode=none
/interface bridge port
add bridge=bridge1 interface=ether1 hw=yes
add bridge=bridge1 interface=ether2 hw=yes
add bridge=bridge1 interface=ether3 hw=yes
/interface ethernet switch vlan
add ports=ether1,ether2 switch=switch1 vlan-id=20
add ports=ether1,ether3 switch=switch1 vlan-id=30
add ports=ether1,switch1-cpu switch=switch1 vlan-id=99
/interface vlan
add interface=bridge1 vlan-id=99 name=MGMT
/ip address
add address=192.168.99.1/24 interface=MGMT
/interface ethernet switch port
set ether1 vlan-mode=secure vlan-header=add-if-missing
set ether2 vlan-mode=secure vlan-header=always-strip default-vlan-id=20
set ether3 vlan-mode=secure vlan-header=always-strip default-vlan-id=30
set switch1-cpu vlan-header=leave-as-is vlan-mode=secure
```

其他没有集成交换芯片的设备

如果没有集成交换芯片，如基于 PC 的 x86 平台，处理二层转发和 VLAN 只能通过 CPU 完成，虽然有几种方法可以实现，但推荐处理 VLAN 使用以下配置：

```
/interface bridge
add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether1 hw=no
add bridge=bridge1 interface=ether2 hw=no pvid=20
add bridge=bridge1 interface=ether3 hw=no pvid=30
/interface bridge vlan
add bridge=bridge1 tagged=ether1 untagged=ether2,ether3 vlan-ids=20,30
add bridge=bridge1 tagged=ether1,bridge1 vlan-ids=99
/interface vlan
add interface=bridge1 vlan-id=99 name=MGMT
/ip address
add address=192.168.99.1/24 interface=MGMT
/interface bridge
set bridge1 vlan-filtering=yes
```

14.11 IGMP Snooping(v6.41)

从 RouterOS v6.41 开始，在 bridge 中支持 ICMP Snooping，用于控制组播数据流。在 bridge 菜单下设置组播参数，并独立工作在各个 bridge 接口下，所有 RouterOS 的设备都支持软件方式处理，只有 CRS1xx/2xx/3xx 系列支持 IGMP Snooping 的硬件处理。

操作路径: /interface bridge, /interface bridge mdb

在 Bridge 下启用 IGMP Snooping

```
/interface bridge set bridge1 igmp-snooping=yes
```

在 interface bridge mdb 下查看组播数据

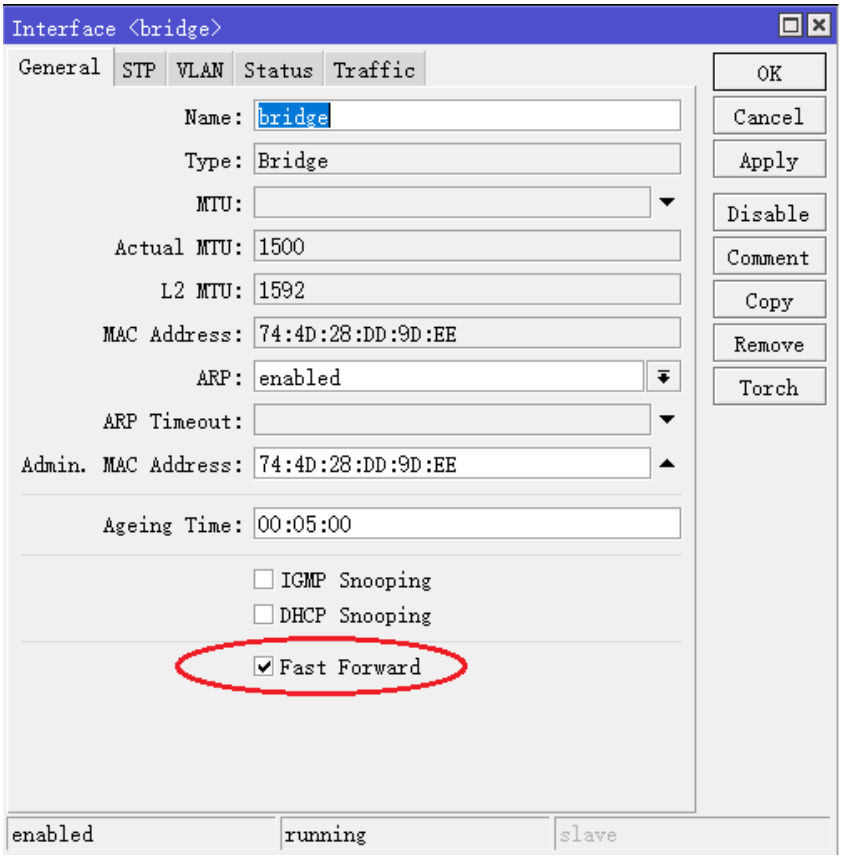
```
[admin@MikroTik] > interface bridge mdb print
BRIDGE                VID GROUP                                  PORTS
bridge1               200 229.1.1.2                              ether3
                                                                 ether2
                                                                 ether1
bridge1               300 231.1.3.3                              ether4
                                                                 ether3
                                                                 ether2
bridge1               400 229.10.10.4                            ether4
                                                                 ether3
bridge1               500 234.5.1.5                              ether5
                                                                 ether1
[admin@MikroTik] >
```

IGMP 端口成员报告仅转发到连接到组播路由器或另一个启用 IGMP 监听的桥接端口。如果没有端口被标记为组播路由器，那么 IGMP 端口成员将不会被转发到任何端口。

CRS 系列交换机的 hardware offloading 支持运行 IGMP snooping，但是 CRS1xx 和 CRS2xx 系列交换机在配置了 VLAN 过滤功能后，IGMP Snooping 的 hardware offloading 功能将无法启用，即无法由交换芯片处理 IGMP Snooping 功能，只能交给系统 CPU 处理。

14.12 Fast Forward

Fast Forward 在指定的条件下可以快速转发数据包，当 Fast Forward 功能被启用，这时 Bridge 能快速处理数据包转发，会跳过 Bridge 下多个关联端口查询，包括 MAC 学习。



下面是启用 Fast Forward 相关的条件清单：

- Bridge 设置 fast-forward= yes
- Bridge 中只有 2 个运行端口
- Bridge 中的 2 个端口，支持 Fast Path ，并启用 Fast Path 功能。
- Bridge Hardware Offloading 关闭
- protocol-mode 设置为 none
- Bridge VLAN Filtering 功能关闭
- unknown-multicast-flood 设置为 yes
- unknown-unicast-flood 设置为 yes
- broadcast-flood 设置为 yes
- horizon 参数在两个端口配置为 none

Fast Forward 的设计概念是禁止 MAC 学习，这实现更快的包转发。MAC 学习可以防止流量发送到多个接口，但是当一个包只能通过一个接口发送时，就不需要 MAC 学习了

启用 hardware offloading 时 Fast Forward 会禁用。hardware offloading 可以实现线速转发，即使用内置的交换芯片处理

Fast Forward 使用 CPU 转发数据，RouterOS 调用硬件转发处理上看会有这样的结果：

Hardware offloading > Fast Forward > Fast Path > Slow Path

通下面的命令可以查看 Fast Forward 启用状态下，处理了多少数据：

```
[admin@MikroTik] > /interface bridge settings print
                   use-ip-firewall: no
```

```
      use-ip-firewall-for-vlan: no
    use-ip-firewall-for-pppoe: no
                  allow-fast-path: yes
      bridge-fast-path-active: yes
     bridge-fast-path-packets: 0
       bridge-fast-path-bytes: 0
bridge-fast-forward-packets: 1279812
  bridge-fast-forward-bytes: 655263744
```

注意：如果数据被 Fast Path 处理，这时 Fast Forward 不会被激活 i。数据包统计可以用来衡量 Fast Forward 是否被激活。

从 RouterOS 6.44beta28 可以查看 Fast Forward 状态，如下：

```
[admin@MikroTik] > /interface bridge monitor bridge1
                         state: enabled
    current-mac-address: D4:CA:6D:E1:B5:82
                root-bridge: yes
           root-bridge-id: 0x8000.00:00:00:00:00:00
           root-path-cost: 0
                root-port: none
               port-count: 2
 designated-port-count: 0
             fast-forward: yes
```

警告:禁用或启用fast-forward将临时禁用所有桥接端口,以便设置生效。每当在实际生产环境中更改此属性时，都必须考虑到这一点，这样会导致桥接网络被临时中断。

14.13 (R)STP

RouterOS Bridge 接口支持 STP 生成树协议，通过冗余拓扑避免二层环路出现。对于只有 2 个二层交换节点的小型网络来说，STP 不会带来太多好处，但是对于较大的网络，正确配置 STP 是非常重要的。如果将 STP 相关值设为默认值可能会导致二层网络出现异常,甚至网络完全不可达,为了实现冗余拓扑,需要正确设置 Bridge 的优先级、端口路径成本和端口优先级。

STP 协议有多版本，目前 RouterOS 支持 STP、RSTP 和 MSTP。CRS3xx 完全支持 hardware offloading 对该协议加速。

STP 协议现在几乎被 RSTP 所取代，在所有的网络拓扑中，RSTP 是向后兼容 STP 的。对于依赖于 VLAN 的网络拓扑，建议使用 MSTP，MSTP 支持 VLAN 协议，并且能够对每个 VLAN 组进行负载均衡。在设计支持 STP 的网络时需要考虑很多问题，更详细的案例研究请自行查阅 STP 生成树协议。

RouterOS 在配置 STP 时，通常建议手动设置每个 Bridge 的优先级、端口优先级和端口路径成本，以确保二层网络转发正常。对于启用(R/M)STP 的由 1 到 2 个交换节点组成的网络，将 STP 相关值保留为默认值是可以接受的，但是强烈建议为较大的网络手动设置。由于 STP 通过检查二层网络上的交换节点的 STP 相关值来选择 root（根），如果将 STP 设置为 automatic（自动），可能出现选择到不正确的 root 节点，出现一些不确定的网络故障，导致网络无法正常通信。

在 RouterOS 上，可以通过/interface bridge monitor ，查看 STP 状态：

```
/interface bridge monitor bridge
                     state: enabled
   current-mac-address: 64:D1:54:D9:27:E6
             root-bridge: yes
        root-bridge-id: 0x3000.64:D1:54:D9:27:E6
        root-path-cost: 0
             root-port: none
            port-count: 5
 designated-port-count: 5
```

查看接口的 STP 状态，通过命令 /interface bridge port monitor ，如下查看 ether3 端口

```
/interface bridge port monitor 2
                interface: ether3
                   status: in-bridge
              port-number: 3
                     role: root-port
               edge-port: no
    edge-port-discovery: yes
    point-to-point-port: yes
            external-fdb: no
            sending-rstp: yes
                learning: yes
              forwarding: yes
         root-path-cost: 10
      designated-bridge: 0x3000.64:D1:54:D9:27:E6
        designated-cost: 0
 designated-port-number: 4
       hw-offload-group: switch1
```

注意，root-bridge-id 由 bridge 的优先级和 bridge 的 MAC 地址组成，对于非 root bridge，将显示为 designated-bridge。在启用 STP 的网络中，每个端口会有一个角色状态，下面是端口可能的角色列状态表：

- 根端口（Root Port）：root Port 存在于非 root bridge 上，该端口具备到 root bridge 的最佳路径（Path Cost）。
- 指定端口（Designated Port）：Designated Port 存在于 root Bridge 和非 root Bridge，通常 root Bridge 上的所有端口都是 Designated Port，而对于非 root Bridge 来说，Designated Port 是指根据需要接收数据或向 root bridge 转发数据的端口。
- 替代端口（Alternate Port）：该类端口为当前"root 端口"到"root bridge"提供一条替代切换路径。作为 root 端口的备份端口
- 备份端口（Backup Port）：作为指定端口的备份， "备份端口"仅当一个 Bridge 节点的两个端口在一个点对点的出现环路时，或者当交换机有两个或多个到达一个 LAN 网络的连接时可以存在。
- 禁用端口（Disabled Port）：该类端口在生成树操作中没有担当任何角色，不参与 RSTP 运算。

当启用 bridge，并设置为 802.1Q 协议，它们将发送 BPDUs 到 01:80:C2:00:00:00，这些 BPDUs 被 MSTP、RSTP 和 STP 使用。当使用 802.1ad 作为 bridge VLAN 协议时，BPDUs 与 802.1Q 网桥不兼容，并发送到 01:80:C2:00:00:08。如果在第二层网络上存在不同的 VLAN 协议，STP 将无法正常工作。

STP 和 RSTP

STP 和 RSTP 在许多网络中被广泛使用，当前主流网络都在使用 RSTP。STP 是一种非常古老的协议，它的收敛时间(了解网络拓扑变化和继续正确转发流量所需的时间)甚至可以达到 50 秒，这在 20 世纪 80 年代发明 STP 时这样的情况是能被接受，但随着网络的发展已经无法满足需要。

RSTP 有更小的收敛时间，几秒甚至几毫秒)，更适网络发展的要求，因此建议使用 RSTP 而不是 STP，因为 RSTP 收敛时间更快，而且与 STP 兼容。RSTP 更快的原因之一是端口状态减少，以下 STP 端口状态列表：

- **Forwarding** -端口参与流量转发，正在学习 MAC 地址，正在接收 BPDUs。
- **Listening** -端口不参与流量转发，不学习 MAC 地址，正在接收 BPDUs。
- **Learning** -端口不参与流量转发，但正在学习 MAC 地址
- **Blocking** -端口被阻塞，因为它正在导致循环，但正在接收
- **Disabled** -端口被禁用或不活动。

在 RSTP 中，disabled、listening 和 blocking 端口状态被替换为 Discarding 状态:

- **Forwarding** -端口参与流量转发，正在学习 MAC 地址，正在接收 BPDUs (forwarding=yes).
- **Learning** -端口不参与流量转发，但正在学习 MAC 地址(learning=yes).
- **Discarding** -端口不参与流量转发，不学习 MAC 地址，正在接收 BPDUs (forwarding=no).

在 STP 中，Bridge 之间的连接是通过在相邻 Bridge 间发送和接收 BPDU 来确定的。指定的端口将 BPDU 发送到 root 端口。如果 BPDU 连续 3 次未接收到 HelloTime，则认为连接不可用，网络拓扑将开始收敛。STP 可以通过减少 forward-delay 值，来减少特定场景下的收敛时间，该值用于处理端口 learning/listening 状态。

在 RouterOS 中，可以指定哪些 Bridge 端口是边缘端口（Edge port）。边缘端口是不应该接收任何 BPDU 的端口，这样允许 STP 跳过 learning/listening 状态，直接进入 forward 状态。该特性有时被称为 PortFast，此参数保留为默认值，即 auto，但也可以手动指定。手动将端口设置为边缘端口（Edge port），通常这些是 access 端口。

BPDU（Bridge Protocol Data Unit，桥协议数据单元）：用于网桥之间传递 BPDU 来交互协议信息。BPDU 分为配置 BPDU 和 TCN BPDU。配置 BPDU 用来进行生成树的计算和维护生成树拓扑的报文，TCN BPDU 是当拓扑结构改变时候，用来通知相关桥设备网络拓扑结构发生变化的报文。

BPDU 包含类型：

- Flags：由 8 位组成，最低位为 TC 标志位，最高位为 TCA 标志位，其他 6 位保留。当拓扑结构变化时候，下游 Bridge 将会从根端口发送 TCN BPDU 报文，TC 标志位置为 1，上游 Bridge 收到后进行相应处理，回复配置 BPDU 报文，TCA 标志位置为 1.
- Root Identifier：根桥 ID，包含优先级和 MAC 地址，标识网络中的根桥。
- Root Path Cost：根路径成本，指从发送该配置 BPDU 的 Bridge 到根 Bridge 的最小路径开销，总路径成本是所有途径链路开销之和。
- Bridge Identifier：发送该配置 BPDU 的 Bridge ID，即该指定 Bridge 的 ID。
- Port Identifier：发送该配置 BPDU 的 Bridge 发送端口 ID。

- Message Age：从根 Bridge 生成配置 BPDU 开始，到当前时间为止配置 BPDU 的存活时间，配置 BPDU 报文经过一个 Bridge，Message Age 增加 1。
- Max Age：配置 BPDU 存活的最大时间。
- Hello Time：根 Bridge 成并发送配置 BPDU 的周期。
- Forward Delay：配置 BPDU 传播到全网的最大时延。

当创建一个 Bridge 或向 Bridge port 添加端口时，RouterOS 分配的 BPDU 默认值如下：

- bridge priority: **32768 / 0x8000 （默认的 bridge 优先级）**
- bridge port path cost: **10 （默认的 bridge 端口路径成本）**
- bridge port priority: **0x80 （默认的 bridge 端口优先级）**
- BPDU message age : **1**（BPDU 消息的生存时间，如果配置 BPDU 是直接来自根 Bridge 的，则 Message Age 为 0，如果是其他桥转发的，则 Message Age 是从根桥发送到当前桥接收到 BPDU 的总时间，包括传输延时等。实际实现中，配置 BPDU 报文经过一个 Bridge，Message Age 增加 1）
- HelloTime: **2** （发送两个相邻 BPDU 的时间间隔）
- max message age: **20** （BPDU 消息的最大生存时间）
- forward-delay：**15s**

RouterOS 不会根据当前端口的链路协商速率改变路径成本值（**Path Cost**），即 10M、100M、1000M 和 10000M 速率的以太网端口被添加到 bridge port 时，默认的路径成本值是 10（wlan 端口默认是 100）。BPDU 的生成周期取决于 BPDU 通过 Bridge 的数量乘以 message age 值，因为 RouterOS 使用 1 作为 message age 值增量，所以 BPDU 包可以通过 max-message-age 参数中指定 bridge 的数量。默认情况下，这个值设置为 20,这意味着经过 20 个 Bridge 后，BPDU 报文后会被丢弃。如果出现下一个 Bridge 将成为一个新的 STP 节点网络。如果在使用 bridge 过滤规则时，确保允许 DST-MAC 地址为 01:80:C2:00:00:00 的数据包通过，因为这些数据包携带 STP 正常工作的 BPDU 信息。

选举过程

网络中如何正确配置 STP，需要了解 STP 节点的选举过程，在 RouterOS 中，root 节点将根据最小优先级值和最小 MAC 地址，按如下顺序选出：

1. Bridge priority (最小值)
2. Bridge MAC 地址 (最小值)

在 RouterOS 中，root 端口是根据最低的端口路径成本、最低的 Bridge 标识符和最低的 Bridge 端口 ID 来选择的，顺序如下：

1. Port path cost（端口路径成本，最小值)
2. Bridge 标识 (最小值)
3. Bridge port ID (最小值)

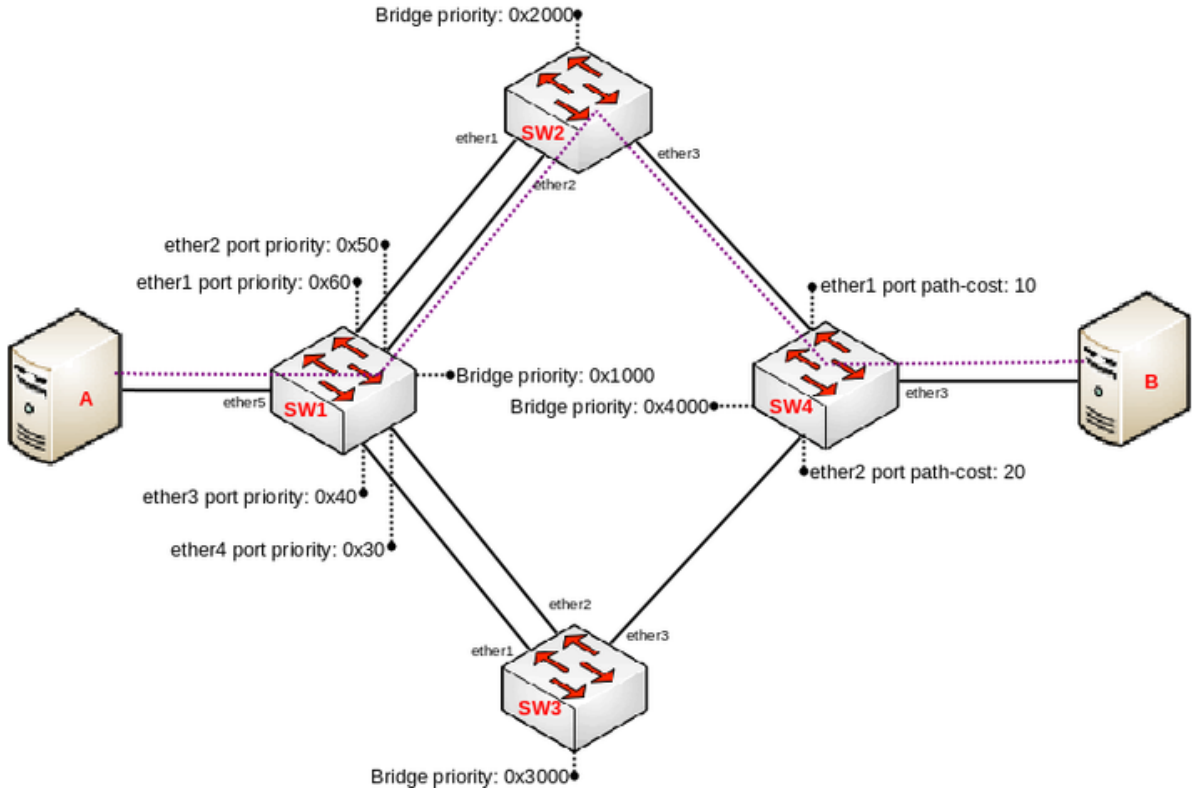
注意：确保在正确的端口上使用了路径成本和优先级。例如，在 root 节点中的端口上设置路径成本没有影响，只有端口优先级对其有影响。路径成本对面向 root 节点的端口有影响，而端口优先级对面向 root 节点的端口有影响

当选择一个 root 端口时，将首先检查路径成本。如果多条路径的路径成本相同，则检查 bridge 的标识符。如果 bridge 标识符相同，则检查 bridge port ID，选择值最低的端口作为 root 端口。在设计启用 STP 的网络时，一定要考虑到选举过程。

警告：在 RouterOS 中，Bridge 的 STP 优先级设置 0 到 65535 之间的任何值。根据 IEEE 802.1W 标准规定 Bridge 的优先级步进必须是 4096，如果随意设置，可能导致不支持这些值的设备出现不兼容性问题。为了避免兼容性问题，建议只使用以下优先级:0、4096、8192、12288、16384、20480、24576、28672、32768、36864、40960、45056、49152、53248、57344、61440

路径成本（Path cost）可在"/interface bridge port"下配置。网桥标识符（Bridge identifier）是"bridge priority"和"bridge MAC"的组合，可在"/interface bridge"下配置，在默认值情况下，MAC 最低的网桥路径将成为 root 端口。网桥端口 ID 是"唯一 ID"和"bridge port priority"的组合，该唯一 ID 添加到网桥后自动分配给网桥端口，无法编辑。可以在"Bridge Port"" Port Number"栏下的 WinBox 中看到，也可以在"/interface bridge port monitor"中看到，作为"port-number"。

实例



在上面的实例中，4 台交换机组成了一个环线网络，并开启(R)STP，从 ServerA 到 ServerB 的连接的需要建立二层冗余。当一个端口连接到一个不运行(R)STP 的设备时，这个端口被认为是一个边缘端口（edge port），在这种情况下，ServerA 和 ServerB 连接到一个边缘端口。下面是每个交换机的配置示例。

SW1 配置：

```
/interface bridge
add name=bridge priority=0x1000
/interface bridge port
add bridge=bridge interface=ether1 priority=0x60
```

```
add bridge=bridge interface=ether2 priority=0x50
add bridge=bridge interface=ether3 priority=0x40
add bridge=bridge interface=ether4 priority=0x30
add bridge=bridge interface=ether5
```

SW2 配置：

```
/interface bridge
add name=bridge priority=0x2000
/interface bridge port
add bridge=bridge interface=ether1
add bridge=bridge interface=ether2
add bridge=bridge interface=ether3
```

SW3 配置：

```
/interface bridge
add name=bridge priority=0x3000
/interface bridge port
add bridge=bridge interface=ether1
add bridge=bridge interface=ether2
add bridge=bridge interface=ether3
```

SW4 配置：

```
/interface bridge
add name=bridge priority=0x4000
/interface bridge port
add bridge=bridge interface=ether1
add bridge=bridge interface=ether2 path-cost=20
add bridge=bridge interface=ether3
```

在本例中，SW1 是 root 节点，它具有最低的 Bridge 优先级。SW2 和 SW3 的 ether1、ether2 连接到 root Bridge，ether3 连接到 SW4。当所有交换机正常工作时，流量将从 ServerA 通过 SW1_ether2、SW2、SW4 流向 ServerB。

在 SW1 中断的情况下，较低优先级的 SW2 将接替 root 节点。下面是每个交换机的端口及其状态列表：

- **root-port** - SW2_ether2, SW3_ether2, SW4_ether1
- **alternate-port** - SW2_ether1, SW3_ether1, SW4_ether2
- **designated-port** - SW1_ether1, SW1_ether2, SW1_ether3, SW1_ether4, SW1_ether5, SW2_ether3, SW2_ether3, SW4_ether3

注意：根据 802.1W 建议，应该按照步进 4096 使用 Bridge 优先级。Bridge 优先级使用的是十六进制表示法，例如，0 是 0x0000, 4096 是 0x1000, 8192 是 0x2000。因此在 RouterOS 设置时，winbox 看到 Bridge priority 默认值是 8000，值是十六进制，十进制为 32768。优先级最小是 0(0x0000)，最大为 61440（0xF000），按照 4096 的步进最多 15 个节点。

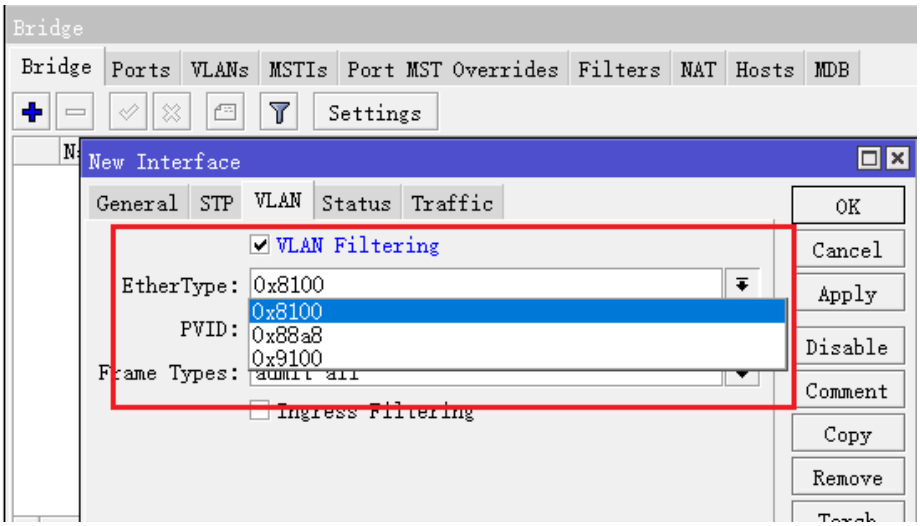
14.14 QinQ

关于 802.1ad 和 802.1q，通常在 QinQ 中设备的内外层标签 TPID 的 Etype 值均采用协议规定的 0x8100，但是某些厂商的设备将 QinQ 报文外层 Tag 的 TPID 中 Etype 值设置为 0x9100 或 0x88a8，RouterOS 提供了 QinQ 报文 TPID 值可调功能

Etype 长度为 2 字节，表示帧类型。取值为 0x8100 时表示 802.1Q 帧。对于内层 VLAN tag，该值设置为 0x8100；对于外层 VLAN tag，有下列几种类型

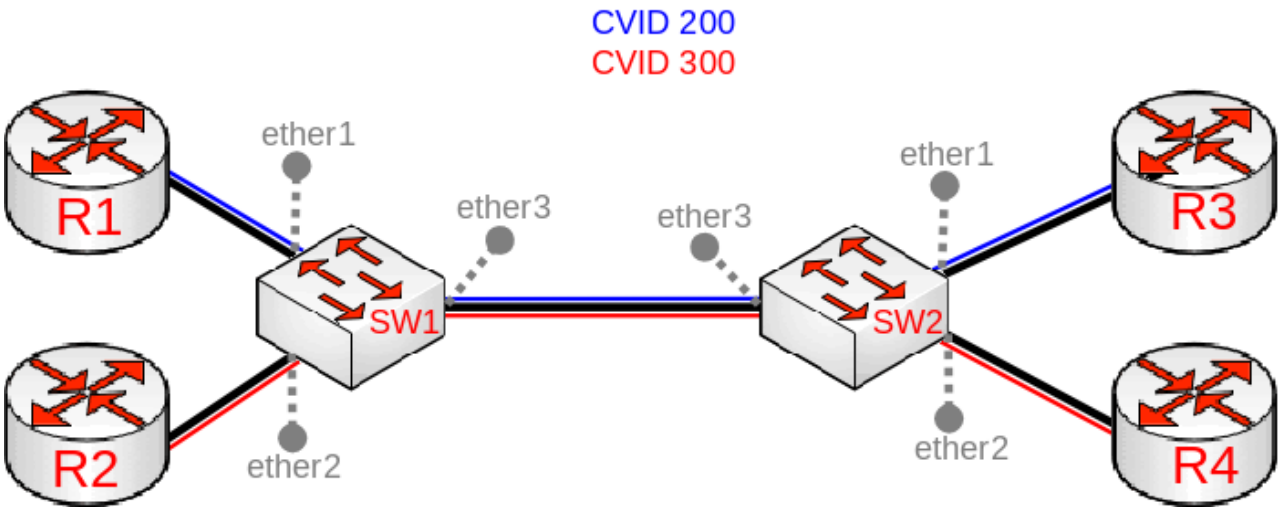
- **0x8100**：SVID,IEEE802.1ad，主流厂商路由交换设备使用，如华为，思科等；
- **0x88A8**：CVID，IEEE802.1Q，802.1ad 协议的路由交换机使用；
- **0x9100**：双重标签，Juniper 设备或其他厂商，不常用。

从 RouterOS v6.43 开始，bridge 完全支持 IEEE 802.1ad，可以根据 Service VLAN ID (0x88A8)过滤 VLAN ID，默认是通过 Customer VLAN ID (0x8100)。以下内容涉及 bridge 功能的操作，请参阅第十四章节 Bridge 网桥



标签堆叠（Tag stacking）

从 v6.43 开始，RouterOS 可以加入新的 VLAN 标记到现有的 VLAN 标记中，此功能可以使用一个 CVID 的堆叠配置，如下面的网络拓扑实例：



上面的实例有 **R1, R2, R3** 和 **R4** 四台路由器，它们可能传输任何 VLAN 标签类型（802.1ad 和 802.1Q），但 **SW1** 和 **SW2** 在 4 台路由器中间，因此需要选择合适的 EtherTpye 值，用于 VLAN 标签的透传，所以 **SW1** 和 **SW2** 选择 **0x8100** 类型（当前主流 QinQ 网络采用的方式）：

```
/interface bridge
add name=bridge1 vlan-filtering=no ether-type=0x8100
```

在 SW1 和 SW2 交换机上 bridge port，配置 ether1 和 ether2 将配置指定的 VLAN 标签，并允许标签堆叠 **tag-stacking=yes**：

```
/interface bridge port
add interface=ether1 bridge=bridge1 pvid=200 tag-stacking=yes
add interface=ether2 bridge=bridge1 pvid=300 tag-stacking=yes
add interface=ether3 bridge=bridge1
```

在 SW1 和 SW2 的 bridge vlan 中指定 tagged 和 untagged 端口

```
/interface bridge vlan
add bridge=bridge1 tagged=ether3 untagged=ether1 vlan-ids=200
add bridge=bridge1 tagged=ether3 untagged=ether2 vlan-ids=300
```

当相应 vlan 配置完成后，启用 SW1 和 SW2 的 vlan 过滤功能，在开启 vlan-filtering 前，确保 VLAN 流量传输给了 CPU，避免丢失 winbox 和三层网络管理，在此之前参考管理口配置

```
/interface bridge set bridge1 vlan-filtering=yes
```

三层标签堆叠实例

通过 802.1q（0x8100）标签堆叠方式，完成三层路由的配置，下面的实例有两台 RouterOS 设备分别是 CRS326 和 RB951,网络拓扑如下：

CRS326 配置

将 ether9 口添加到 bridge port 中，并设置 pvid 为 300，启用 tag-stacking

```
/interface bridge port
add bridge=bridge interface=ether9 pvid=300 tag-stacking=yes
```

将 vlan 300 分别 tagged 给 bridge（传递给 CPU，用于三层配置），untagged 给 ether9

```
/interface bridge vlan
add bridge=bridge tagged=bridge untagged=ether9 vlan-ids=300
```

进入 interface vlan 菜单，创建 vlan id 为 300 接口，取名为 vlan300

```
/interface vlan
add interface=bridge1 name=vlan300 vlan-id=300
```

在 vlan300 接口下，添加内层 vlan 30

```
/interface vlan
add interface=vlan300 name=vlan30 vlan-id=30
```

进入 ip address 添加 IP 地址

```
/ip address
add address=10.1.1.1/24 interface=vlan300
add address=10.2.2.1/24 interface=vlan30
```

最后启用 vlan-filtering

```
/interface bridge set bridge1 vlan-filtering=yes
```

RB951 配置

将 ether4 口添加到 bridge port 中，并设置 pvid 为 300，启用 tag-stacking

```
/interface bridge port
add bridge=bridge1 interface=ether4 pvid=300 tag-stacking=yes
```

将 vlan 300 分别 tagged 给 bridge（传递给 CPU，用于三层配置），untagged 给 ether4

```
/interface bridge vlan
add bridge=bridge1 tagged=bridge1 untagged=ether4 vlan-ids=300
```

进入 interface vlan 菜单，创建 vlan id 为 300 接口，取名为 vlan300

```
/interface vlan
add interface=bridge1 name=vlan300 vlan-id=300
```

在 vlan300 接口下，添加内层 vlan 30

```
/interface vlan
add interface=vlan300 name=vlan30 vlan-id=30
```

进入 ip address 添加 IP 地址

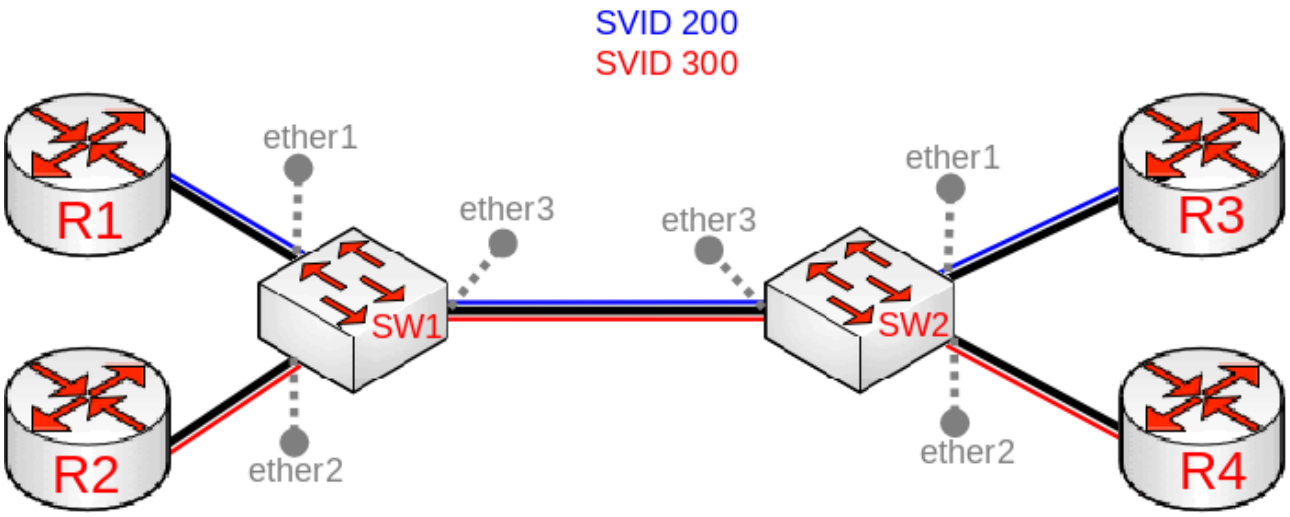
```
/ip address
add address=10.1.1.2/24 interface=vlan300
add address=10.2.2.2/24 interface=vlan30
```

最后启用 vlan-filtering

```
/interface bridge set bridge1 vlan-filtering=yes
```

802.1ad 实例

在某些场景下，其他厂商设备会使用 0x88a8 的 Etype 值，RouterOS 需要调整 VLAN 过滤配置，仍然通过上面的标签堆叠实例来说明基于如何配置 802.1ad 的 QinQ 配置：



上面的实例，R1, R2, R3 和 R4 可通过 802.1Q (CVID Customer VID)发送任何 VLAN 标记，但 SW1 和 SW2 需要在路由器之间隔离流量，R1 只能与 R3 通信，R2 只能与 R4 通信。为此，可以使用 SVID 标记所有进入流量，并且只允许这些 vlan 在某些端口上运行。

首先在网桥上启用 802.1ad VLAN 协议（0x88a8），在 SW1 和 SW2 上使用以下命令：

```
/interface bridge
add name=bridge1 vlan-filtering=no ether-type=0x88a8
```

在 SW1 和 SW2 交换机 ether1 和 ether2 将配置为 access 端口（untagged）：

```
/interface bridge port
add interface=ether1 bridge=bridge1 pvid=200
add interface=ether2 bridge=bridge1 pvid=300
add interface=ether3 bridge=bridge1
```

在 VLAN 列表中指定 tagged 和 untagged 端口，在 SW1 和 SW2 配置命令如下：

```
/interface bridge vlan
add bridge=bridge1 tagged=ether3 untagged=ether1 vlan-ids=200
add bridge=bridge1 tagged=ether3 untagged=ether2 vlan-ids=300
```

当 bridge VLAN 列表配置完成，则在 SW1 和 SW2 启用 VLAN filtering 功能，在开启 vlan-filtering 前，确保 VLAN 流量传输给了 CPU，避免丢失 winbox 和三层网络管理，在此之前参考管理口配置

```
/interface bridge set bridge1 vlan-filtering=yes
```

ether-type 为 0x88a8（SVID Service VLAN ID)，只有 CRS3xx 系列交换机支持 hardware offloading

警告: 当 ether-type=0x8100, 然后网桥检查外部 VLAN 标记是否使用 EtherType= 0x8100。如果网桥接收到一个不同 EtherType 的外部标记的包，它将把该包标记为 untagged。由于 RouterOS 的 VLAN 过滤只检查包的外部标签，所以在使用 802.1ad（ether-type=0x88a8）协议时，是无法过滤 802.1Q（ether-type=0x8100）的数据，因此需要特别注意 VLAN 过滤类型的选择。

14.15 Bridge 下二层错误配置

单交换芯片的多 Bridges 配置

大多数 RB 和 CRS 设备是单交换芯片，是无法支持多 bridge 的 hw-offloading 功能。考虑以下情况，希望在 RB 或 CRS 设备建立两个或两个以上的 Bridge，用于两个独立的二层广播域网络，在使用一台单交换芯片的 RB 或 CRS 设备时，启用 hw-offloading，可能存在 CPU 过高的问题。这里说明下，Linux Bridge（网桥）是工作于二层的网络设备，功能等同于物理的交换机，可以参考下:交换机原理和 RouterOS Bridge 二层交换配置（视频）。

配置：

```
/interface bridge
add name=bridge1
add name=bridge2
/interface bridge port
add bridge=bridge1 interface=ether1
add bridge=bridge1 interface=ether2
```

```
add bridge=bridge2 interface=ether3
add bridge=bridge2 interface=ether4
```

存在问题

经过运行测试后，可能会注意到 bridge1 能够以线速转发流量，而第二个 Bridge2 的转发性能就很差，而另外一个问题是路由转发时出现巨大的延迟，同时 CPU 处于高负载，这是因为 hw-offloading 不是在所有 bridge 上都能使用，只能在第一个 Bridge 上生效，通过在 bridge port 下查看，可以得知当前 hw 激活端口：

```
[admin@MikroTik] > /interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic, H - hw-offload
 #      INTERFACE                  BRIDGE                HW
 0    H ether1                     bridge1               yes
 1    H ether2                     bridge1               yes
 2      ether3                     bridge2               yes
 3      ether4                     bridge2               yes
```

上面可以看到只有 Bridge1 有 HW-offloading 的标识“H “，而 Bridge2 只能由 CPU 处理

症状

这种情况的错误配置症状表现如下：

- 在 bridge port 下无 "H" 标识
- 转发吞吐量低
- 高 CPU 利用率

解决方法

并不是所有的设备支持多 Bridge，目前只有 CRS1xx/CRS2xx 系列设备同时支持 7 组 hw-offloading 的 Bridge，其他设备仅支持 1 个的 Bridge。对于多 Bridge 可以通过 hw 属性开关其他 Bridge 的 hw-offloading，你可以用 hw=yes 标志控制哪个网桥支持 hw-offloading，例如:

```
/interface bridge port set [find where bridge=bridge1] hw=no
/interface bridge port set [find where bridge=bridge2] hw=yes
```

其实多 Bridge 方案设计是很特殊的情况，大多的网络解决方案都是在一个 Bridge 下完成，通过 VLAN 和端口隔离，将多个二层广播域划分开。华为和 H3C 交换机采用 Linux 内核，也是将所有端口划分在一个 Bridge 下，并封装在一个配置界面操作。而 RouterOS 提供了开放的配置，容易造成一些意外的操作，虽然有些功能可以实现，但会导致系统负载过高。

错误的 VLAN interface

考虑以下场景 ether1 和 ether2 被创建到了一个 bridge 下，其中一个网络接入到 ether1 接口，此时该网络对接的交换机通过 VLAN tagged 透传了一个 VLAN id（交换机配置 trunk 接口到路由器），而且需要为这个 VLAN 创建 DHCP 服务器，这样的场景需要正确选择 VLAN 的主接口，否则会导致网络无法连通。

配置

配置 bridge

```
/interface bridge
add name=bridge1
/interface bridge port
add interface=ether1 bridge=bridge1
add interface=ether2 bridge=bridge1
```

创建 vlan（错误配置）

```
/interface vlan
add name=VLAN99 interface=ether1 vlan-id=99
```

创建 dhcp 服务器

```
/ip pool
add name=VLAN99_POOL range=192.168.99.100-192.168.99.200
/ip address add address=192.168.99.1/24 interface=VLAN99
/ip dhcp-server
add interface=VLAN99 address-pool=VLAN99_POOL disabled=no
/ip dhcp-server network
add address=192.168.99.0/24 gateway=192.168.99.1 dns-server=192.168.99.1
```

存在问题

当创建一个 Bridge，此 bridge 接口将变为 master 接口，而所有加入到 bridge port 下的端口都变为从属的 slave 端口，这意味着经过该 bridge 的所有流量都会通过 bridge 接口转发到 CPU，因此，在 slave 接口上的 VLAN 永远不会被捕获。通常情况是一些 DHCP 客户端接收 IP 地址，但无法连接网络，因为数据无法被 CPU 处理。

症状

这种情况的错误配置症状表现如下：

- DHCP Client/Server 无法正常工作 not working properly;
- 网关设备无法到达;
- 在 bridge 后的设备无法打上 tagged 标签

解决方法

更改 VLAN 接口的主接口为 bridge1：

```
/interface vlan set VLAN99 interface=bridge1
```

interface vlan 加入 bridge

但在本例中，只是在不同以太网物理接口上创建的两个或多个 interface VLAN，并桥接在一起，这样的错误配置会导致网络故障。这是一种非常常见的错误操作，值得探讨下：

配置

直接在两个 ether1 和 ether2 接口创建 interface vlan

```
/interface vlan
add name=ehter1_vlan10 interface=ether1 vlan-id=10
add name=ether2_vlan10 interface=ether2 vlan-id=10
```

并将创建的 vlan 加入 bridge，bridge 默认是开启 STP 协议

```
/interface bridge
add name=bridge1
/interface bridge port
add bridge=bridge1 interface=ether1_vlan10
add bridge=bridge1 interface=ether2_vlan10
```

存在问题

以上操作从逻辑层面没有问题，但可能会发现部分网络无法访问或某些链接一直在超时。这是由于(R)STP 的原因，这种配置会强制设备发送带有标签的 BPDU 报文，可能是其他设备(包括 RouterOS)不支持的。由于设备接收到一个畸形的 BPDU 包(当运行(R)STP 时，带标签的 BPDU 不应该存在于您的网络中，这违反了 IEEE 802.1W 和 IEEE 802.1Q)，设备将不能正确地解释带有标签的包，导致网络异常。

症状

这种情况的错误配置症状表现如下：

- 端口可能被 RSTP 阻止;
- 端口数据异常;
- 网络不可达;

解决方法

最简单的解决方案是在 Bridge 上禁用(R)STP:：

```
/interface bridge
set bridge1 protocol-mode=none
```

但还是仍然建议重新配置，使用 Bridge 的 vlan-filtering 方式，将 ether1 和 ether2 打上 vlan 标签

```
/interface bridge
add name=bridge1 vlan-filtering=yes
/interface bridge port
add bridge=bridge1 interface=ether1
add bridge=bridge1 interface=ether2
/interface bridge vlan
add bridge=bridge1 tagged=ether1,ether2 vlan-ids=10
```

第十五章 Interface VLAN（三层）

Virtual Local Area Network (VLAN)工作在 OSI 参考模型第二层，可以在一张独立的物理网卡上拥有多个虚拟 LAN 接口（例如：以太网卡和无线网卡），能有效的隔离各个二层广播域。 使用 MikroTik RouterOS 与华为、H3C、Linux 和其他路由系统一样，都能处理 VLAN 标签的数据，同样能通过 Bridge 将 VLAN 透传到 wlan 无线连接，并在一张无线网卡上设置多个 VLAN 接口。

注意：/interface vlan 配置主要应用在 VLAN 三层 IP 地址的配置，类似华为和 H3C 等交换机/路由器的 intreface vlan id，也类似于其他品牌路由器的 vlan 子接口，如：interface ether1.id。对于二层 VLAN 配置，则由/bridge vlan 或/interface ethernet switch 菜单配置。

规格

功能包要求: ***system***
等级要求: *Level1 (限制 1 个 vlan 规则)* , *Level3*
子目录要求: ***/interface vlan***
标准与技术: VLAN (IEEE 802.1Q)

属性

属性	描述
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; 默认: **enabled**)	地址解析协议的模式
interface (name; 默认:)	需要进入 VLAN 的物理接口的名称
l2mtu (*整型*; 默认:)	二层 MTU. 对于 VLAN 这个参数不需要配置
mtu (*整型*; 默认: **1500**)	三层最大传输单元
name (*字符*; 默认:)	自定义接口的名称
use-service-tag (*yes* | *no*; 默认: no)	兼容 802.1ad 标签
vlan-id (整型; *4095*; 默认: **1**)	虚拟 LAN 验证或者 tag 标签用于 VLAN 通信，必须设置与对方相同的 VLAN

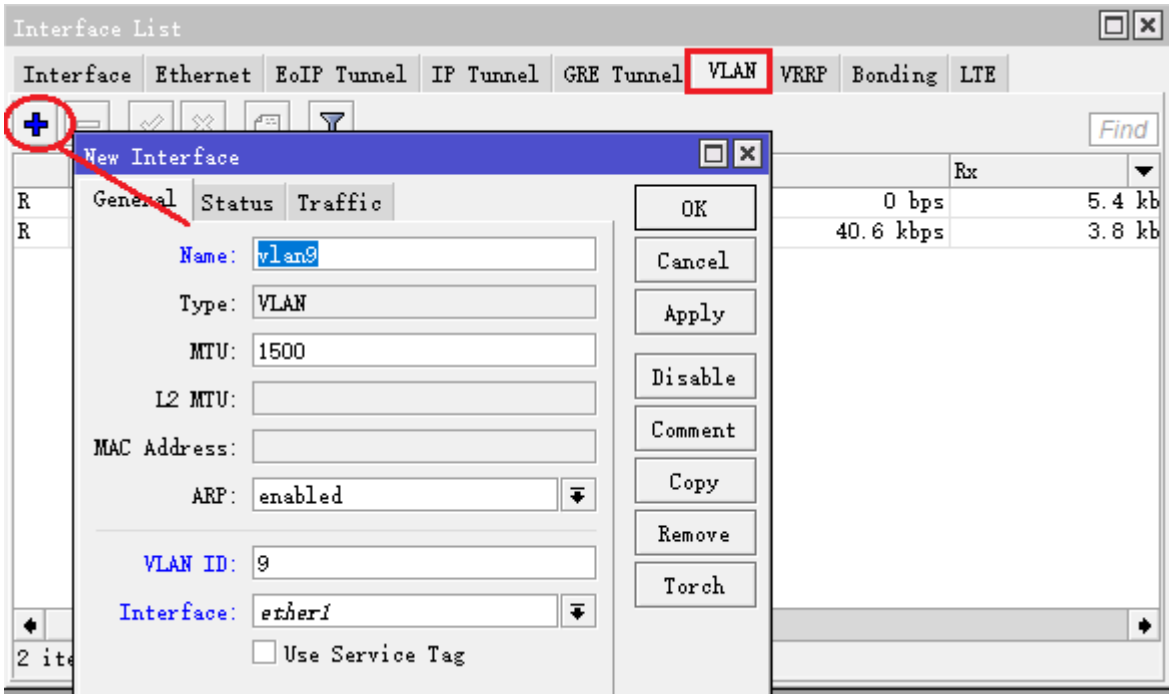
注：MTU 必须像在以太网接口那样设置为 1500 字节。但这样也可能不能与一些不支持接受/传输满长度带有 VLAN 标题的以太网数据报的以太网卡一起工作（1500 字节数据+4 字节 VLAN 标题+14 字节以太网标题）。这种情况下使用 MTU1496，但要注意如果较长的数据报要在接口发送的话这会引起数据报的分割。同时要记得如果路径 MTU 搜索在源和目的间不能正常工作，MTU1496 可能引起一些问题。

在接口 ether1 添加并启用名为 vlan9 且 vlan-id=9 的 VLAN：

```
[admin@MikroTik] interface vlan> add name=vlan9 vlan-id=9 interface=ether1
[admin@MikroTik] interface vlan> print
Flags: X - disabled, R - running
 #     NAME                      MTU   ARP          VLAN-ID INTERFACE
```

```
 0 X   vlan9                    1500 enabled          9          ether1
[admin@MikroTik] interface vlan> enable 0
[admin@MikroTik] interface vlan> print
Flags: X - disabled, R - running
 #      NAME                    MTU   ARP          VLAN-ID INTERFACE
 0   R vlan9                    1500 enabled          9          ether1
[admin@MikroTik] interface vlan>
```

Winbox 中添加如下图：



15.1 802.1Q 协议

IEEE802.1Q 虚拟 LAN 是一个通用协议，标准的封装协议定义了如何插入 4 个字节的 VLAN 信息到以太网包头。下图是标准的以太网协议和 802.1Q 的对比

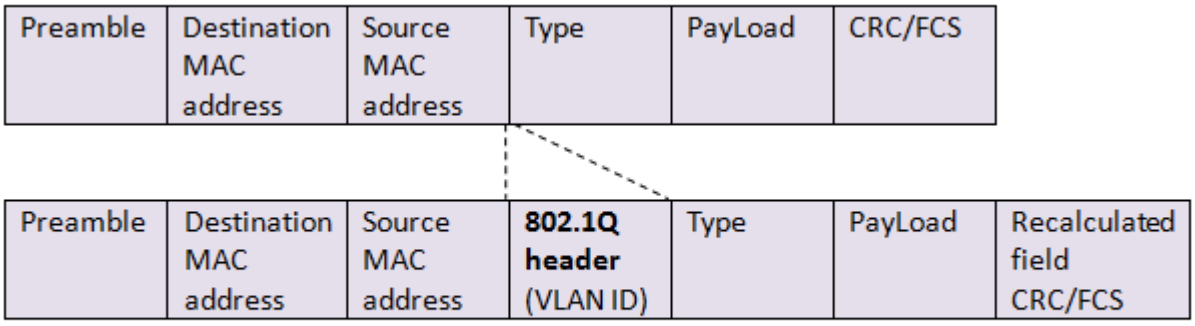
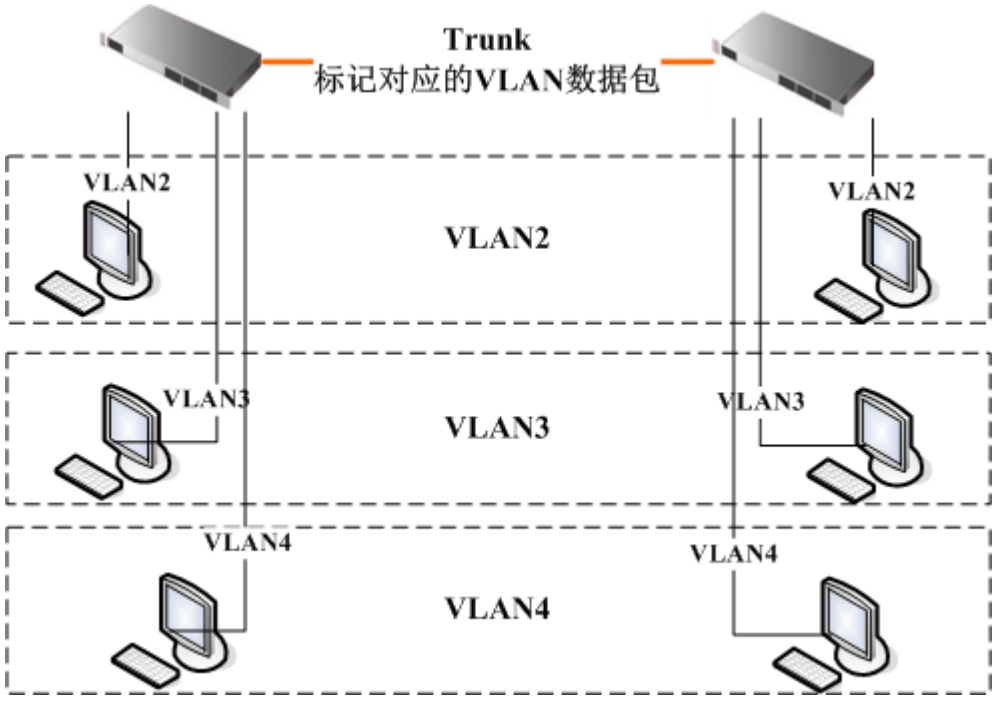


Figure 12.1. Insertion of 802.1Q Tag (VLAN ID) in Ethernet-II frame

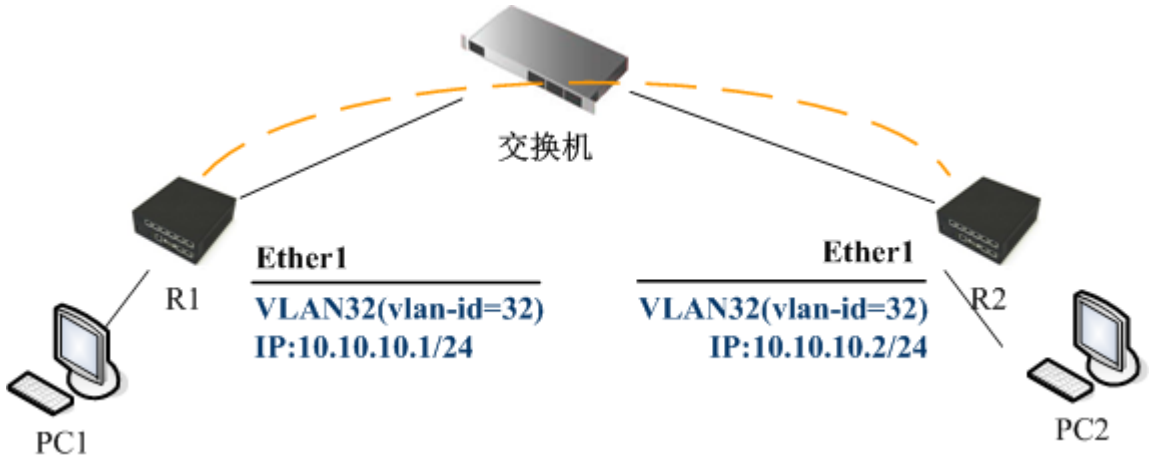
每个 VLAN 被当做独立的子接口，当主机被指定到一个 VLAN 后，虽然他们在同一个交换机下，但不能连接其他 VLAN 里的主机。

当多个 VLAN 分部在多个交换机时，交换机之间需要通过 trunk（tagged 加上 VLAN 标签）方式连接，数据报将被打上属于自己 VLAN 标签，当交换机把对应的 VLAN 传递给下面的主机时，会去掉标签（untagged），即我们通常说的 access 口。



15.2 简单 VLAN 事例

我们假设有两个 RouterOS 的路由器通过交换机或者 hub 连接。我们需要通过 VLAN 将他们连接起来，这里他们的网卡都是 **ether1**，如下图。



我们首先创建 VLAN，R1 和 R2 的配置一样

```
[admin@MikroTik] interface vlan> add name=vlan32 vlan-id=32 interface=ether1
[admin@MikroTik] interface vlan> print
Flags: X - disabled, R - running
  #     NAME                    MTU    ARP        VLAN-ID INTERFACE
  0   R vlan32              1500 enabled        32        ether1
```

```
[admin@MikroTik] interface vlan>
```

如果 VLAN 接口成功的创建，这时 vlan 通过二层完成连接。接下来我们需要为 vlan32 配置 IP 地址，启用三层通信

在 R1 上：

```
[admin@MikroTik] ip address> add address=10.10.10.1/24 interface=vlan32
[admin@MikroTik] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #    ADDRESS          NETWORK       BROADCAST          INTERFACE
  0    10.0.0.204/24        10.0.0.0          10.0.0.255         ether1
  1    10.20.0.1/24         10.20.0.0         10.20.0.255        pc1
  2    10.10.10.1/24        10.10.10.0        10.10.10.255       vlan32
[admin@MikroTik] ip address>
```

在 R2 上：

```
[admin@MikroTik] ip address> add address=10.10.10.2/24 interface=vlan32
[admin@MikroTik] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #    ADDRESS                NETWORK             BROADCAST          INTERFACE
  0    10.0.0.201/24        10.0.0.0          10.0.0.255         ether1
  1    10.10.10.2/24        10.10.10.0        10.10.10.255       vlan32
[admin@MikroTik] ip address>
```

如果设置得正确，那么从 R1 可以 ping 通 R2，否则你的网络可能没正确连接：

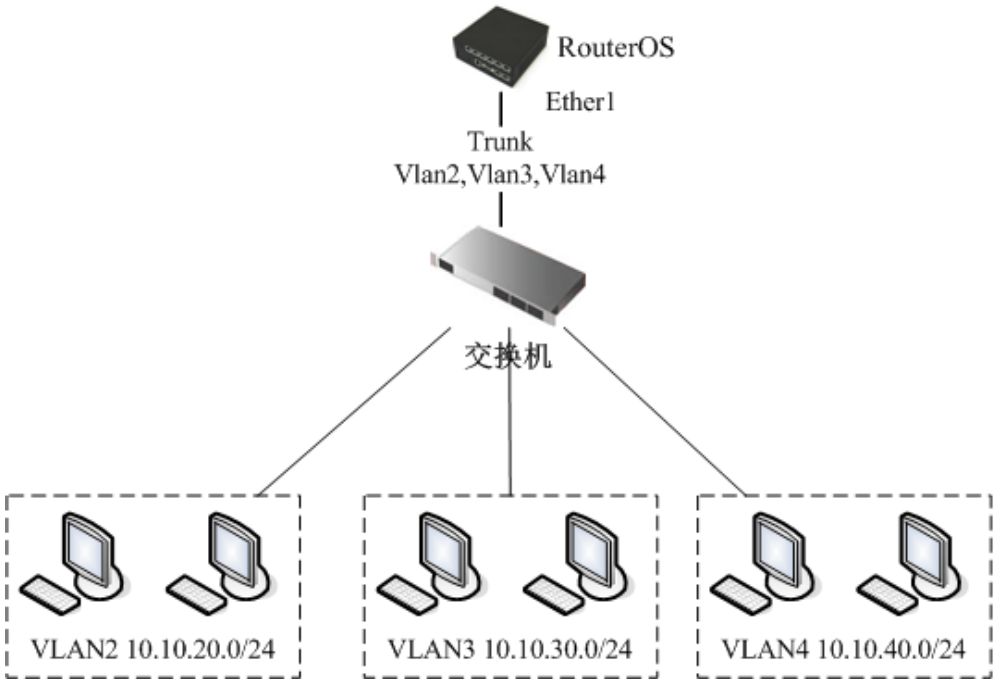
```
[admin@MikroTik] ip address> /ping 10.10.10.1
10.10.10.1 64 byte pong: ttl=255 time=3 ms
10.10.10.1 64 byte pong: ttl=255 time=4 ms
10.10.10.1 64 byte pong: ttl=255 time=10 ms
10.10.10.1 64 byte pong: ttl=255 time=5 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 3/10.5/10 ms
[admin@MikroTik] ip address> /ping 10.10.10.2
10.10.10.2 64 byte pong: ttl=255 time=10 ms
10.10.10.2 64 byte pong: ttl=255 time=11 ms
10.10.10.2 64 byte pong: ttl=255 time=10 ms
10.10.10.2 64 byte pong: ttl=255 time=13 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 10/11/13 ms
[admin@MikroTik] ip address>
```

15.3 Trunk 连接

如果 VLAN 是在一个交换机或者多个交换机上创建，这时路由器需要加入 VLAN 的连接，而交换机工作在 OSI 的第二层转发二层以太网数据，并不会检查 IP 数据报，而我们需要将每个 VLAN 建立一个三层通信，即每个 VLAN 下的主机能通过 IP 地址访问的路由器，并通过路由器作为网关，访问外部网络。这样的网络方案被广泛使用到我们的网络中，是非常重要的一种 VLAN 解决方案

为了让交换机之间建立的 VLAN 通信能传递到路由器，我们需要在交换机上启用 trunk 模式，把二层网络内的 VLAN 透传给路由器，同时每个 VLAN 在二层上是相互独立开，这样 VLAN 下的用户访问只能通过路由器的 IP 方式访问

如下图，我们有 3 个 VLAN 汇聚到一个交换机上，通过一个 trunk 接口连接到 Router，RouterOS 能识别 VLAN 的 trunk 模式中的 vlan id，并能在每个 VLAN 上启用三层管理



如图，每个 VLAN 都有自己独立的子网（独立的广播域），如下：

- VLAN 2 - 10.10.20.0/24;
- VLAN 3 - 10.10.30.0/24;
- VLAN 4 - 10.10.40.0/24

VLAN 的配置已经的交换机上完成，我们只需要在 RouterOS 添加对应的 VLAN，至于交换机如果配置 VLAN 参数各种各种交换机有所不同，这里不再具体介绍，只需要确定交换机与 RouterOS 对接的口是 trunk 模式，并允许了相应的 VLAN 通过即可

创建 VLAN 接口：

```
/interface vlan
add name=VLAN2 vlan-id=2 interface=ether1 disabled=no
add name=VLAN3 vlan-id=3 interface=ether1 disabled=no
add name=VLAN4 vlan-id=4 interface=ether1 disabled=no
```

添加每个 VLAN 的 IP 地址:

```
/ip address
add address=10.10.20.1/24 interface=VLAN2
add address=10.10.30.1/24 interface=VLAN3
add address=10.10.40.1/24 interface=VLAN4
```

当然这种 VLAN 方式也可以应用于 Hotspot 认证，只是需要在每个 vlan 上设置 IP 地址。

15.4 三层 QinQ 配置

QinQ 通过在以太帧中堆叠两个 802.1Q 包头（802.1Q in 802.1Q），有效地扩展了 VLAN 数目，使 VLAN 的数目最多可达 4096x4096 个，原本 802.1Q 仅允许一个 VLAN 包头，Q-in-Q 则允许在一个 VLAN 中包含多个 VLAN 包头通过。

简单实例：

在 RouterOS 配置是在一个现有的 VLAN 接口上，再添加一个 VLAN 标签，如下外层 VLAN 是标签是 11，内层 VLAN 标签是 12

```
/interface vlan
add name=vlan11 vlan-id=11 interface=ether1   disable=no
add name=vlan11.12 vlan-id=12 interface=vlan11 disable=no
```

如果数据报发向 vlan2 接口，2 个 vlan 卷标将会被添加到以太网的包头里 "11" 和 "12"。

第十六章 Bonding

Bonding 是通过汇聚多个网口到一个虚拟的连接上，实现二层的链路负载均衡和冗余，这种方式可以获得更高的带宽或提供容错转移，与路由中提到的多线路负载均衡类似，只是这个功能是基于二层网络实现。

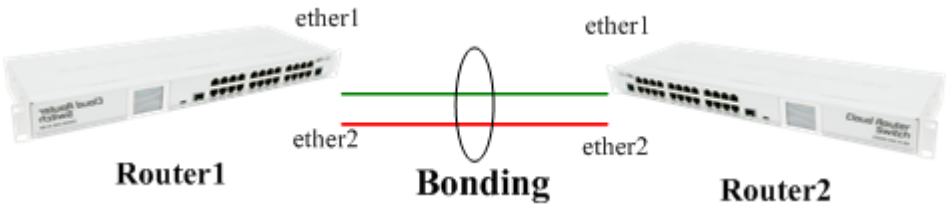
规格

需要功能包: ***system***
需要等级: *Level1*
操作路径: ***/interface bonding***

关于 Bonding 功能，其实在运营商网络是很常见的一项功能应用，通常是交换机之间进行端口聚合和冗余使用，通常称为 link-aggregate，在 linux 中被称为 bond，其实是 linux 服务器与交换机实现端口聚合功能，对于 RouterOS 是 linux 内核当然是按照 bond 配置实现与交换设备实现端口聚合功能。除了链路的冗余外，bond 最主要的目的是多个 100M 或 1G 以太网口汇聚提升带宽，特别是万兆网卡价格昂贵情况下，通过多张 1G 以太网卡实现汇聚。

16.1 Bonding 基本操作

假设两台路由器（Router1 和 Router2）通过 1 个 100M 的以太网接口互联，但由于流量超过 100M，小于 150M，为了扩容两台路由器互联，必须使用千兆接口，但没有千兆接口情况下，就需要采用 2 张 100M 网卡做 bonding 实现 200M 的互联，使得路由器之间得到最大的传输速率。



如下配置：

1. 确定你没有 IP 地址在相应的接口，这将被从属到 bonding 接口上！
2. 在 Router1 上添加 **bonding** 接口（默认 **mode=balance-rr**）：

```
[admin@Router1] interface bonding> add slaves=ether1,ether2
```

在 Router2 上添加：

```
[admin@Router2] interface bonding> add slaves=ether1,ether2
```

3. 添加 IP 地址到两台路由器的 bonding 接口上：

```
[admin@Router1] ip address> add address=172.16.0.1/24 interface=bonding1
[admin@Router2] ip address> add address=172.16.0.2/24 interface=bonding1
```

4. 在 Router1 上测试连接：

```
[admin@Router1] interface bonding> /pi 172.16.0.2
172.16.0.2 ping timeout
172.16.0.2 ping timeout
172.16.0.2 ping timeout
172.16.0.2 64 byte ping: ttl=64 time=2 ms
172.16.0.2 64 byte ping: ttl=64 time=2 ms
```

bonding 接口需要几秒钟时间的连通时间。

16.2 Bonding 的七种模式

如果经常用 linux 服务器的朋友来说，bond 模式不会陌生，做 Linux 服务器时，受到单网卡速率限制，如 100M 或者 1000M，要求扩容时，考虑成本或者物理接口原因，会使用 bond 来做负载均衡，这个操作和交换机的链路聚合一个道理。

在 linux 下，bond 一共有 7 种模式，由于 RouterOS 采用 linux 内核，也同样是这七种

1、balance-rr（Round-robin policy）

对应 linux 的 mod=0，采用轮询负载均衡的方式，即：第一个包走 ether0，第二个就走 ether1，依次轮询发送，直到最后一个发送完毕，该模式提供负载均衡和容错能力，并且要求交换机配置的支持。该模式也是 Linux 服务器最常用的一种方式。

2、 active-backup

对应 linux 的 mod=1，只有一个网卡处于活动状态，另一个处于备份，当一个故障或意外中断，另一个马上由备份转换为主。mac 地址是外部可见得，从外面看来，bond 的 MAC 地址是唯一的，以避免 switch(交换机)发生混乱。此模式只提供了容错能力；但该模式利用率低，并极少被用到。

3、balance-xor

对应 liunx 的 mod=2，采用指定的 HASH 算法传输数据报，实现对流量的负载均衡。和 balance-rr 一样，交换机端口需要能配置端口聚合。这模式是通过源和目标 mac 做 hash 因子来做 xor 算法来选路的，当所有流量是通过单个路由器（比如“网关”型网络配置，只有一个网关时，源和目标 mac 都固定了，那么这个算法算出的线路就一直是同一条，那么这种模式就没有多少意义了）

4、broadcast

对应 linux 的 mod=3，该模式的特点是一个数据会复制两份往 bond 下的两个或多个接口分别发送出去,当有对端交换机一个接口失效，我们感觉不到任何延迟，但由于将数据复制到两个接口上发送，过于浪费网络接口资源，但该模式有极好的容错机制，因此该模式适用于金融或对数据传输要求极高的行业，因为他们需要高可靠性的网络，不允许出现任何问题。

5、802.3ad（IEEE 802.3ad 动态链接聚合）

对应 linux 的 mod=4，802.3ad 模式是 IEEE 标准，802.3ad 协议包括聚合的自动配置，要求交换机支持 802.3ad，才能使用。802.3ad 标准也要求帧按顺序（一定程度上）传递，因此通常单个连接不会看到包的乱序。802.3ad 也有些缺点：标准要求所有设备在聚合操作时，要在同样的速率和双工模式，而且，和除了 balance-rr 模式外的其它 bonding 负载均衡模式一样，任何连接都不能使用多于一个接口的带宽。此外，linux bonding 的 802.3ad 实现通过对端来分发流量（通过 MAC 地址的 XOR 值），因此在"网关"型配置下，所有外出（Outgoing）流量将使用同一个设备。进入（Incoming）的流量也可能在同一个设备上终止，这依赖于对端 802.3ad 实现里的均衡策略。因此要求每个 slave 使用相同速率和双工模式，交换机支持 IEEE 802.3ad（Dynamic link aggregation）

RouterOS 的 802.3ad 配置：

```
/inteface bonding add slaves=ether1,ether2 mode=802.3ad lacp-rate=30secs link-monitoring=mii-type1
transmit-hash-policy=layer-2-and-3
```

6，balance-tlb（Adaptive transmit load balancing）

对应 linux 的 mod=5，该模式不需要交换机对 bonding 的配置支持。在每个 slave 上根据当前的负载（根据速度计算）分配外出流量。 不像 802.3ad，该模式的接口可以有不同的速率，而且不需要特别的交换机配置。不利的一面在于，该模式下所有进入的（incoming）流量会到达同一个接口，而且 ARP 监控不可用。

7， balance-alb（ Adaptive load balancing）

对应 linux 的 mod=6，该模式包含了 balance-tlb 模式，同时加入针对 IPV4 流量的接收负载均衡(receive load balance, rlb)，而且不需要任何交换机的支持。接收负载均衡是通过 ARP 协商实现的。Bonding 驱动器截获本机发送的 ARP 应答，并把源硬件地址改写为 bond 中某个 slave 的唯一硬件地址，从而使得不同的对端使用不同的硬件地址进行通信。该模式也是 Linux 服务器最常用的，在不依赖于交换机配置的情况下，交换机只需要将对应的接口配置在相同 VLAN 或广播域，即可完成数据的负载均衡。

16.3 Link monitoring 连接监测

Link Monitoring 对于 bonding 运行链路监控的重要手段，在没有启用监控的情况下，如果 bonding 其中一条链路失效，bonding 仍然会继续向失效的链路发送数据，这样造成了网络故障。当前 bonding 在 RouterOS 中连接状态监测支持两种方案：MII 和 ARP，但这两种模式不能同时在 bonding 驱动中使用。

ARP 监控

ARP 监测通过向在同一广播域内指定的网关或主机 IP 发送 ARP 请求，并通过 ARP 响应确认该链路运行正常。

启用 ARP 监控，假设两台 RouterOS 做了 bond，IP 地址是 172.16.0.1/24 和 172.16.0.2/24，相互配置 ARP 监控

```
[admin@Router1] interface bonding> set 0 link-monitoring=arp arp-ip-targets=172.16.0.2
[admin@Router2] interface bonding> set 0 link-monitoring=arp arp-ip-targets=172.16.0.1
```

无法修改 arp-interval 值，RouterOS 设置默认为 100ms

拔掉网线后，测试如果链路探测工作正常，ARP 探测将发现链路异常，并决定故障链路是否关闭

```
[admin@Router1] interface bonding> /pi 172.16.0.2
172.16.0.2 ping timeout
172.16.0.2 64 byte ping: ttl=64 time=2 ms
172.16.0.2 ping timeout
172.16.0.2 64 byte ping: ttl=64 time=2 ms
172.16.0.2 ping timeout
172.16.0.2 64 byte ping: ttl=64 time=2 ms
172.16.0.2 64 byte ping: ttl=64 time=2 ms
172.16.0.2 64 byte ping: ttl=64 time=2 ms
```

MII 监控

MII 监控，在 RouterOS 可以配置以下两种方式：

- **MII Type 1** – 网卡适配器决定链路状态 UP 或 DOWN，如果网卡驱动器不支持该选项，链路将默认 UP 状态
- **MII Type 2** – 通过系统内核处理方式判断链路是否 UP 或 DOWN，该模式效率低，但能适用于任何网卡适配器，该模式仅作 MII type 1 不支持的情况下使用

主要缺点是 MII 监控不能做链路监测（数据报通过该链路状态到达网关是否正常），也就是只能监测网口是否连接（网口是否亮）；当然 Bond 也支持 ARP 协议的链路监测

MII 监控配置属性在 ***link-monitoring*** 和 ***mii-interval***，默认的 mii-interval 值为 100ms，下面是启用 MII Type2 的监控

```
[admin@Router1] interface bonding> set 0 link-monitoring=mii-type-2
[admin@Router2] interface bonding> set 0 link-monitoring=mii-type-2
```

属性描述

arp (disabled | enabled | proxy-arp | reply-only; 默认: **enabled**) – 接口的地址解析协议
  disabled – 关闭 ARP 协定
  enabled – 启用 ARP 协议
  proxy-arp – 使用 ARP 代理功能
  reply-only – 将只响应/ip arp 的静态 MAC 地址
arp-interval (*time*; 默认: **00:00:00.100**) – 通过定义多少毫秒监测 ARP 请求。
arp-ip-targets (*IP 地址*; 默认: **""**) – IP 目标地址，如果 **link-monitoring** 被设置 **arp** 目标 IP 地址将会被监视。你也可以指定多个 IP 地址。
down-delay (*时间*; 默认: **00:00:00**) – 如果一个连接失效被探测到，bonding 接口通过 **down-delay** 时间禁用配置。
lacp-rate (1sec | 30secs; 默认: **30secs**) – 连接聚合控制协议速率是指定多久将 bonding 端的 LACPDUs 进行交换。被用于确定是否连接或进行其他变化。LACP 试着适应这些变化并提供失效管理。
link-monitoring (arp | mii-type1 | mii-type2 | none; 默认: **none**) – 连接监视是否使用 (是否设置启用)

arp – 使用地址解析协议，探测远程地址是否到达。

mii-type1 – 使用 MII type1 协议确认连接状态。连接状态探测依赖设备驱动。如果 bonding 显示状态为 up，但运行时并未启动，说明该卡可能不支持 bonding 功能。

mii-type2 – 使用 MII type2 探测连接状态（被用于如果 **mii-type1** 不支持 NIC）

none – 没有任何模式监测，如果一个连接失效，不会被关闭（但没有传输通过）。

mac-address (*只读: MAC address*) – bonding 接口的 MAC 地址

mii-interval (*时间*; 默认: **00:00:00.100**) – 多久监测一次连接失效(此参数被用于在 **link-monitoring** 设置为 **mii-type1** 或 **mii-type2**)

mode (802.3ad | active-backup | balance-alb | balance-rr | balance-tlb | balance-xor | broadcast; 默认: **balance-rr**) – 接口绑定模式，如下：

802.3ad - IEEE 802.3ad 动态连接聚合，提供容错和负载平衡。在这个模式下，接口被聚合到一个组里，每个 slave 共享同样的速度。如果你在两个 bonding 路由器之间使用一个交换机，必须确定这个交换机支持 IEEE 802.3ad。**active-backup** – 提供连接备份。在同一时间仅一个 slave 可以运行。如果一个失效，另外一个 slave 自动连接。

balance-alb – 自适应负载均衡。该模式包含 **balance-tlb**，通过接收传输负载均衡。设备驱动应支持设置 MAC 地址，不需要指定的交换机支持

balance-rr – 轮询负载均衡。在 bonding 接口里 Slaves 将依次序的传输和接收。提供负载均衡和容错

balance-tlb – 输出传输同分布式方式分配负荷到当前的每个 slave 上，传入数据被接收通过当前 salve。如果接收 slave 失败，这时另外一个 slave 带走实效的 MAC 地址。不需要任何特殊的交换机支持

balance-xor – 为传输使用 XOR 策略。仅提供失效管理，但不支持负载均衡

broadcast – 同样的数据在所有接口广播一次。这样提供失效容错，但在一些慢的机器上降低了传输吞吐量。

mtu (*整型*: 68..1500; 默认: **1500**) – 最大传输单元，单位 btyes

name (*名称*) – bonding 接口的名称

primary (*名称*; 默认: **none**) – 接口被涩混浊主要的输出媒体。如果主接口失效，从属接口会被自动启用。该参数仅能使用于 **mode=active-backup**

slaves (*名称*) – 至少 2 个 ethernet 接口被用于 bonding 接口

up-delay (*时间*; 默认: **00:00:00**) – 如果一个链路已经连接，bonding 接口被 **up-delay** 时间禁用，在这个时间过后 bonding 接口启用。

16.3 RouterOS 与交换机做 Bond 配置

实际的生产环境中，RouterOS 最常与交换机做 bond 配置，即链路聚合，对于网络交换机链路聚合被称为 LACP（Link Aggregation Control Protocol）。这里介绍下如何与华为交换机实现链路聚合，配置采用两种方式，一种是依赖于交换机配置，一种是无需依赖交换机配置。

下面事例假设 RouterOS 有两张 1G 网卡 ether1 和 ether2，我们连接的是华为 5700 交换机。

方法一、balance-rr

基于 balance-rr 的 bond 配置，要求与交换机配置匹配，即交换机要配置 LACP 参数，下面在 RouterOS 上添加 bonding 接口，设置 mode=balance-rr：

```
[admin@Router1] interface bonding> add slaves=ether1,ether2
```

添加地址到 bonding 接口上：

```
[admin@Router1] ip address> add address=172.16.0.1/24 interface=bonding1
```

RouterOS 配置设置完成

配置创建 vlan 8 后，进入 interface vlan8 设置三层接口，并添加 IP 地址

```
[5700]interface vlan 8
[5700-vlanif8] ip address 172.16.0.2 24
[5700-vlanif8]quit
```

华为 5700 创建 eth-trunk 链路汇聚口

```
[5700]interface Eth-Trunk1
[5700-Eth-Trunk1]port link-type access
[5700-Eth-Trunk1]port default vlan 8
[5700-Eth-Trunk1]quit
```

将 GigabitEthernet 0/0/1 和 GigabitEthernet 0/0/2 加入 Eth-Trunk1 的链路汇聚接口

```
[5700] interface GigabitEthernet 0/0/1
[5700-GigabitEthernet0/0/1]eth-trunk 1
[5700-GigabitEthernet0/0/1]interface GigabitEthernet 0/0/2
[5700-GigabitEthernet0/0/2]eth-trunk 1
[5700-GigabitEthernet0/0/2]
```

方法二、balance-alb

在 RouterOS 上添加 bonding 接口，设置 mode=balance-alb：

```
[admin@Router1] interface bonding> add slaves=ether1,ether2 mode=balance-alb
```

添加地址到 bonding 接口上：

```
[admin@Router1] ip address> add address=172.16.0.1/24 interface=bonding1
```

RouterOS 配置设置完成

配置创建 vlan 8 后，进入 interface vlan8 设置三层接口，并添加 IP 地址

```
[5700]interface vlan 8
[5700-vlanif8]ip address 172.16.0.2 24
[5700-vlanif8]quit
```

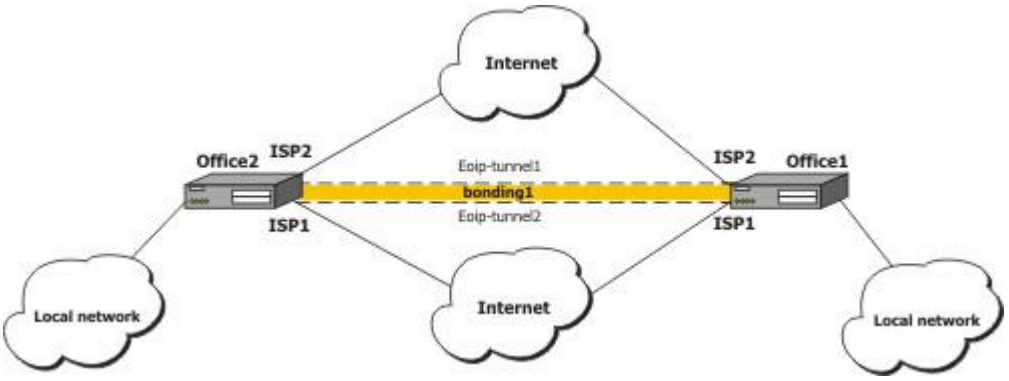
将 GigabitEthernet 0/0/1 和 GigabitEthernet 0/0/2 安装普通的 access 口配置，加入 vlan8，无需仍和特殊配置

```
[5700] interface GigabitEthernet 0/0/1
[5700-GigabitEthernet0/0/1]port link-type access
```

```
[5700-GigabitEthernet0/0/1]port default vlan 8
[5700-GigabitEthernet0/0/1]interface GigabitEthernet 0/0/2
[5700-GigabitEthernet0/0/2]port link-type access
[5700-GigabitEthernet0/0/2]port default vlan 8
[5700-GigabitEthernet0/0/2]
```

16.4 官方 EoIP 隧道的 Bonding

假设你需要通过 MikroTik 路由器配置以下的网络设置，你有 2 个办公室，并同时接入了相同的 2 个 ISP 线路，你想绑定 2 条线路，得到双倍的带宽速度，并提供失效冗余。



两个路由器直接通过 2 个 ISP 连接到 Internet，并配置这两个路由器连接上网。

- 配置 **office1** 路由器：

```
[admin@office1] > /interface print
Flags: X - disabled, D - dynamic, R - running
 #      NAME                                                TYPE               MTU
 0   R   isp1                                               ether             1500
 1   R isp2                                                ether             1500

[admin@office1] > /ip address print
Flags: X - disabled, I - invalid, D - dynamic
 #    ADDRESS             NETWORK         BROADCAST       INTERFACE
 0    1.1.1.1/24            1.1.1.0         1.1.1.255       isp2
 1    10.1.0.111/24          10.1.0.0        10.1.0.255      isp1
```

配置 **Office2** 的路由器

```
[admin@office2] interface> print
Flags: X - disabled, D - dynamic, R - running
 #      NAME                                                TYPE               MTU
 0   R isp2                                                ether             1500
 1   R isp1                                                ether             1500
[admin@office2] interface> /ip add print
Flags: X - disabled, I - invalid, D - dynamic
 #    ADDRESS             NETWORK         BROADCAST       INTERFACE
```

```
0    2.2.2.1/24              2.2.2.0          2.2.2.255        isp2
1    10.1.0.112/24           10.1.0.0         10.1.0.255       isp1
```

通过 EoIP 隧道连接，实现一个虚拟的二层网络连接，用于 bonding 的连接（由于 bonding 基于二层链路层的链路聚合，所以必须使用 2 层接口）。先配置 **Office1** 通过 ISP1 连接的 EoIP 隧道：

```
[admin@office1] > interface eoip add remote-address=10.1.0.112 tunnel-id=2
\... mac-address=FE:FD:00:00:00:04
[admin@office1] > interface eoip print
Flags: X - disabled, R - running
0   R name="eoip-tunnel2" mtu=1500 mac-address==FE:FD:00:00:00:04 arp=enabled
\... remote-address=10.1.0.112 tunnel-id=2
```

在 **Office2** 路由器上配置 ISP1 线路的 EoIP

```
[admin@office2] > interface eoip add remote-address=10.1.0.111 tunnel-id=2
\... mac-address=FE:FD:00:00:00:02
[admin@office2] > interface eoip print
Flags: X - disabled, R - running
0   R name="eoip-tunnel2" mtu=1500 mac-address=FE:FD:00:00:00:02 arp=enabled
\... remote-address=10.1.0.111 tunnel-id=2
```

在 **Office1** 路由器上配置 ISP2 的 EoIP 隧道

```
[admin@office1] > interface eoip add remote-address=2.2.2.1 tunnel-id=1
\... mac-address=FE:FD:00:00:00:03
[admin@office1] interface eoip> print
Flags: X - disabled, R - running
 0   R name="eoip-tunnel1" mtu=1500 mac-address=FE:FD:00:00:00:03 arp=enabled
     remote-address=2.2.2.1 tunnel-id=1

1   R name="eoip-tunnel2" mtu=1500 mac-address=FE:FD:00:00:00:04 arp=enabled
 remote-address=10.1.0.112 tunnel-id=2
```

在 **Office2** 路由器上配置 ISP2 的 EoIP 隧道

```
[admin@office2] > interface eoip add remote-address=1.1.1.1 tunnel-id=1
\... mac-address=FE:FD:00:00:00:01
[admin@office2] interface eoip> print
Flags: X - disabled, R - running
 0   R name="eoip-tunnel1" mtu=1500 mac-address=FE:FD:00:00:00:01 arp=enabled
       remote-address=1.1.1.1 tunnel-id=1

 1   R name="eoip-tunnel2" mtu=1500 mac-address=FE:FD:00:00:00:02 arp=enabled
       remote-address=10.1.0.111 tunnel-id=2
```

设置 Bonding，在 **Office1**

```
[admin@office1] interface bonding> add slaves=eoip-tunnel1,eoip-tunnel2
[admin@office1] interface bonding> print
Flags: X - disabled, R - running
 0   R name="bonding1" mtu=1500 mac-address=00:0C:42:03:20:E7 arp=enabled
slaves=eoip-tunnel1,eoip-tunnel2 mode=balance-rr primary=none   link-monitoring=none
arp-interval=00:00:00.100 arp-ip-targets=""   mii-interval=00:00:00.100 down-delay=00:00:00
up-delay=00:00:00   lacp-rate=30secs
[admin@office1] ip address> add address=3.3.3.1/24 interface=bonding1
[admin@office1] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #    ADDRESS             NETWORK          BROADCAST        INTERFACE
 0    1.1.1.1/24          1.1.1.0          1.1.1.255        isp2
 1    10.1.0.111/24       10.1.0.0         10.1.0.255       isp1
 2    3.3.3.1/24          3.3.3.0          3.3.3.255        bonding1
```

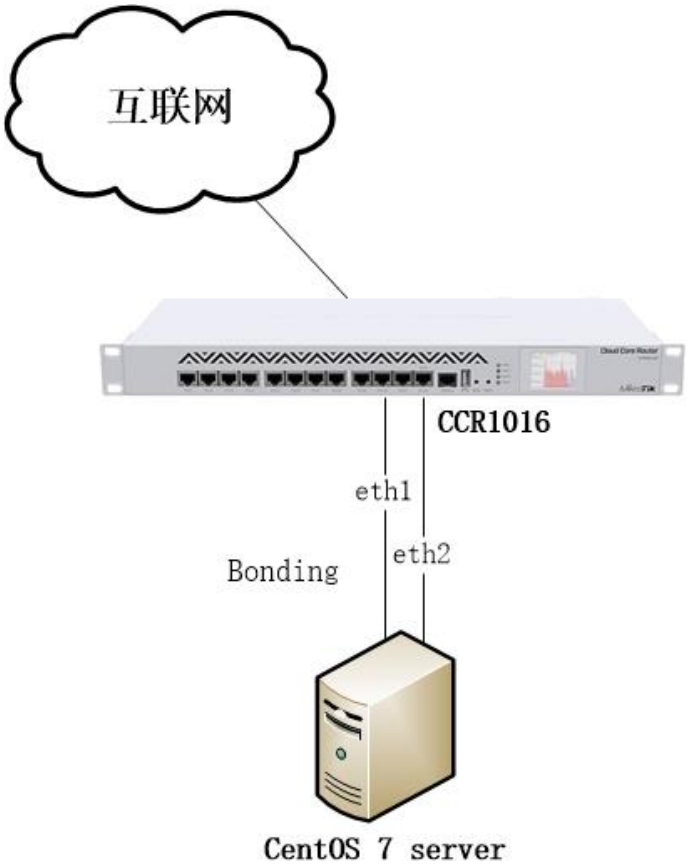
在 **Office2** 上配置

```
[admin@office2] interface bonding> add slaves=eoip-tunnel1,eoip-tunnel2
[admin@office2] interface bonding> print
Flags: X - disabled, R - running
 0   R name="bonding1" mtu=1500 mac-address=00:0C:42:03:20:E7 arp=enabled
       slaves=eoip-tunnel1,eoip-tunnel2 mode=balance-rr primary=none
       link-monitoring=none arp-interval=00:00:00.100 arp-ip-targets=""
       mii-interval=00:00:00.100 down-delay=00:00:00 up-delay=00:00:00
       lacp-rate=30secs
[admin@office2] ip address> add address=3.3.3.2/24 interface=bonding1
[admin@office2] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #    ADDRESS             NETWORK          BROADCAST        INTERFACE
 0    2.2.2.1/24          2.2.2.0          2.2.2.255        isp2
 1    10.1.0.112/24       10.1.0.0         10.1.0.255       isp1
 2    3.3.3.2/24          3.3.3.0          3.3.3.255        bonding1
[admin@office2] ip address> /ping 3.3.3.1
3.3.3.1 64 byte ping: ttl=64 time=2 ms
3.3.3.1 64 byte ping: ttl=64 time=2 ms
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 2/2.0/2 ms
```

关于无线 bonding 设置 RouterOS 无线内容的介绍

16.5 RouterOS 与 CentOS 7 配置 bond

在 IDC 网络应用场景中，服务器没有配置万兆网卡，且流量超过 1G 的时候，我们需要使用多个千兆网卡完成二层汇聚，采用 bond 方式完成两个或两个以上的千兆网卡汇聚，下面的事例将介绍如何用 RouterOS 与 CentOS7 服务器完成两张千兆网卡的 bond



RouterOS 和 CentOS 都是基于 Linux 内核，所以他们的 bond 配置方式很相似，

RouterOS 配置：

首先配置网卡 IP 地址

```
/ip address
add address=10.200.1.2/28 interface=wan
add address=172.31.0.1/27 interface=bonding1
```

配置默认路由

```
/ip route
add gateway=10.200.1.1
```

在 interface bonding 下创建 ether1 和 ether2 两张网卡的 bond 配置，哈希算法我使用了 layer-2-and-3，也可以选择默认

```
/interface bonding
add mii-interval=200ms name=bonding1 slaves=ether1,ether2 mode=balance-rr
```

这里配置 mode 默认是 balance-rr，在 linux 里面 bond 对应的配置 mode 为 0

CentOS 7 服务器配置

首先创建 bond0 网卡配置，通过 vi 在/etc/sysconfig/network-scripts/下创建 ifcfg-bond0 文件

```
[root@localhost ~]# vi /etc/sysconfig/network-scripts/ifcfg-bond0
TYPE=bond
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
NAME=bond0
DEVICE=bond0
ONBOOT=yes
BONDING_MASTER=yes
BONDING_OPTS="miimon=200 mode=0"
IPADDR=172.31.0.2
NETMASK=255.255.255.192
GATEWAY=172.31.0.1
```

修改 ifcfg-em1 配置

```
[root@localhost ~]# vi /etc/sysconfig/network-scripts/ifcfg-em1
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=none
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=em1
UUID=2be61eaa-a773-472d-952d-a1ef739f570c
DEVICE=em1
ONBOOT=yes
MASTER=bond0
SLAVE=yes
```

修改 ifcfg-em2 配置

```
[root@localhost ~]# vi /etc/sysconfig/network-scripts/ifcfg-em2
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=none
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
```

```
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=em2
UUID=3fb532e6-939a-476e-b563-ad2dacc5ae01
DEVICE=em2
ONBOOT=yes
MASTER=bond0
SLAVE=yes
```

配置完成后，重启 CentOS7 的网络服务

```
[root@localhost ~]#Service network restart
```

通过命令检查 bond 情况

```
[root@localhost ~]#cat /proc/net/bonding/bond0
```

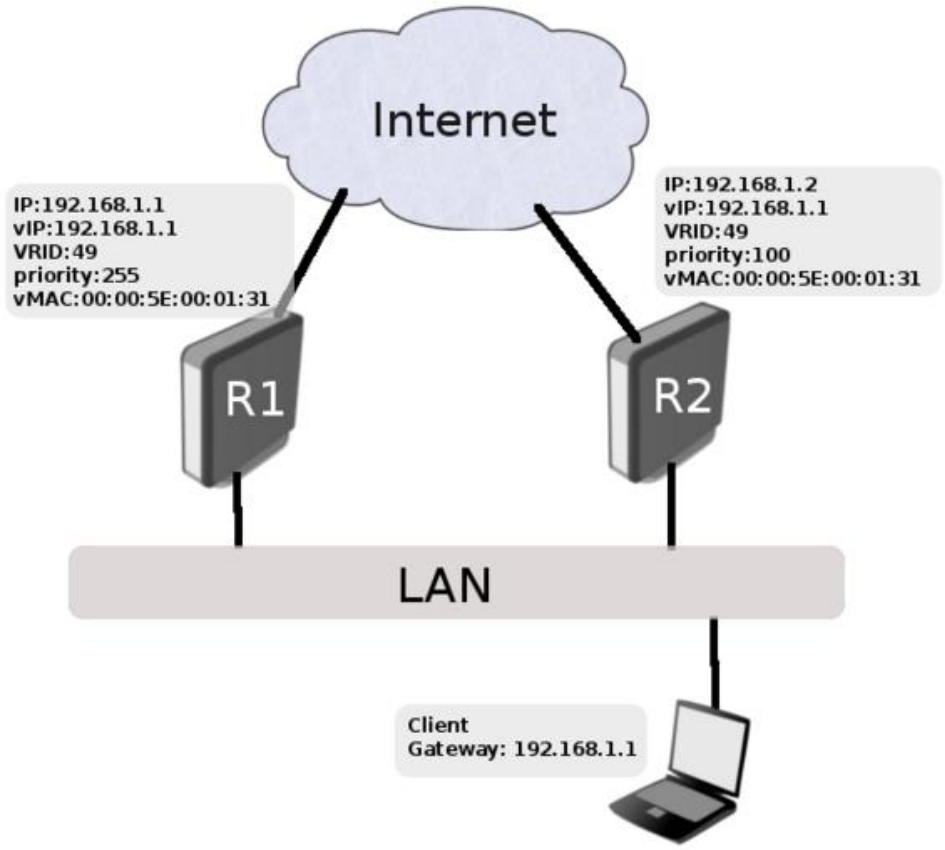
第十七章 VRRP (虚拟路由冗余协议)

虚拟路由冗余协议 Virtual Router Redundancy Protocol (VRRP)，MikroTik RouterOS VRRP 协议遵循 RFC2338。VRRP 协议是保证访问指定资源不会中断，即通过多台路由器组成一个网关集合，如果其中一台路由器出现故障，会自动启用另外一台。两个或多个路由器建立起一个动态的虚拟集合，每一个路由器都可以参与处理数据，这个集合最大不能超过 255 个虚拟路由器(可参考虚拟路由协议)。一般现在的路由器都支持该协议。

在 IPv6 网络中，主机需要学习到周围路由器是通过接收路由器广播的邻居发现协议(Neighbor Discovery 简称 ND)。ND 已经内置了不可到达路由器的检查机制，但需要花费多达 38 秒探测时间 ，虽然可以通过修改参数让探测更快，但如果在大量主机网络中会增加 ND 在网络中的探测流量。VRRP 能实现在 3 秒内探测到不可达路由器，且不用增加流量开销 。

虚拟路由器冗余协议(VRRP)提供的解决方案是将多台路由器的组合，变成虚拟路由器(VR)的逻辑组，即在网络中多台路由器被看着为一台。RouterOS 的 VRRP 符合 VRRPv2 RFC 3768 和 VRRPv3 RFC 5798 标准。

17.1 协议介绍



VRRP 协议设计思路是两台或两台以上的路由器，设置了 VRRP 的虚拟路由 ID，他们之间建立 VRRP 虚拟路由器通信，并通过 VRRP 路由器之间优先选择来支持路由器的冗余。RouterOS 的 VRRP 有以下特点：

所有通信消息都是由 IPv4 或 IPv6 多播包的协议 112 (VRRP)完成的，IPv4 的目标 IP 地址为 224.0.0.18 ，IPv6 是 FF02:0:0:0:0:0:0:12。数据包的源地址始终是发送数据包的接口的主 IP 地址，在 IPv6 网络中则是接口的 link-local 地址

这些数据包总是用 TTL=255 发送，不被路由器转发（路由协议中 TTL=255 表示数据包周期最大值）。如果因任何原因其他路由器收到一个 TTL=255 的数据包，将不再转发。

每个虚拟路由节点都有一个指定的 MAC 地址。此 MAC 地址被用于同一网络段中的源，通过 Master 路由器定期发送消息。

虚拟路由器识别通过 VRID 定义，并映射一组 IPv4 或 IPv6 地址。Master 路由器向网络宣告映射的 IPv4/IPv6 地址的所有者。对于 IPv4 和 IPv6 使用相同的 VRID 没有限制，但是将是两种不同的虚拟路由器。

只有 Master 路由器定期发送宣告信息，以减少流量，只有在 Backup 路由器有更高的优先级，且不禁止抢占的情况下，Backup 路由才会尝试取代 Master 路由器。

所有属于同一 VR（虚拟路由器）的 VRRP 路由器必须配置相同的宣告间隔时间。如果间隔不匹配，路由器将丢弃接收到的宣告数据包。

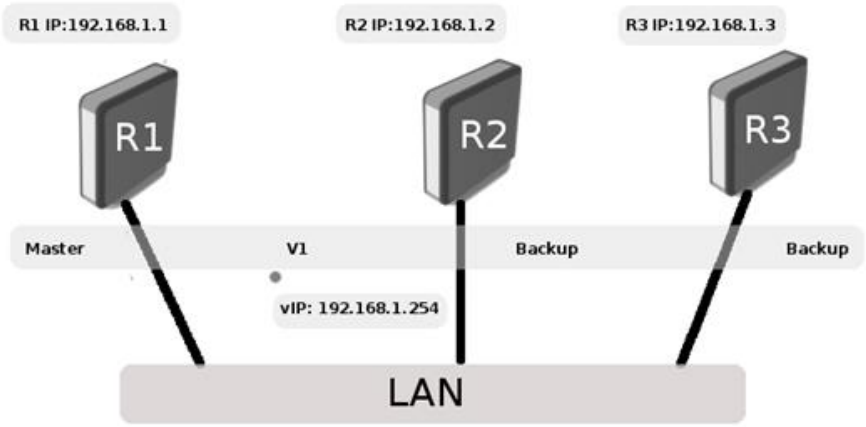
警告： 如果 VRRP 与反向路由过滤一起使用，那么建议将 rp-filter 设置为 loose，否则可能无法访问 VRRP 接口。

Virtual Router (VR)

虚拟路由器 (VR)组成由一个主路由器和一个或多个备份路由器组成，并在相同的网络地址段中。VR 配置包括：

- VRID 配置到每台 VRRP 路由器
- 每台路由器配置相同的虚拟 IP 地址
- 主和备份都会在每台路由器上设置，但在 VR 网络中，只有一台主路由器工作

在 VR 网络中，默认会有一台 Master 路由器，并对当前子网段中的设备提供网关服务。一台 Mater 路由器的优先级必须是最高（最高为 255），例如：网络中的 R1 路由器作为 Master，优先级设置为 255，虚拟 IP 地址（virtual IP）与网络中的真实 IP 地址相同，即作为网络中的网关 IP：192.168.1.254



在虚拟路由器网络成员中，推荐虚拟 IP 地址和物理接口实际 IP 不同，便于网络中虚拟地址可以任意修改，在上图中，R1，R2 和 R3 物理接口 IP 地址分别为 192.168.1.1，192.168.1.2，192.168.1.3，而作为网关的虚拟 IP 地址是 192.168.1.254，这样虚拟地址可以被称为浮动或纯虚拟 IP 地址。这种设置的优点是为网络管理员提供灵活性。由于虚拟 IP 地址不是任何参与路由器的真实物理接口地址，所以管理员可以变更物理路由器或它们的 IP 地址，而不需要重新配置虚拟路由网络的本身

虚拟 MAC 地址（Virtual MAC address）

VRRP 自动分配 MAC 地址到 VRRP 的接口，VRRP 的 MAC 分配基于标准 MAC 地址前缀和 VRID 编号。前五位是 00:00:5E:00:01 ，最后一位是 VRID，例如 VRID 为 9，那么虚拟 MAC 地址是 00:00:5E:00:01:09.

注意：虚拟 MAC 地址不能手动修改。

Master（主路由器）

Master 路由器在虚拟路由器网络中作为物理网关配置的，选举 Master 路由器是通过优先级参数控制。 Master 状态 内容中描述了 Mastre 路由器如何工作，在上图中 R1 是 Master 路由器，当 R1 中断或无法工作时，R2 将接替 R1，作为 Master 路由器。

Backup（备份路由器）

VR 虚拟路由器网络中必须包含一个 Backup 路由器，且 Backup 路由器需配置与 Master 路由器相同的虚拟 IP 地址，RouterO VRRP 中默认优先级为 100，即 Backup 路由器默认的优先级。如果当前 Master 路由器不再工作，Backup 路由器就拥有最高优先级变更为 Mater 路由器。当更高优先级的路由器恢复时，会变为 Master 路由器，但有时这样的操作没有必要，这样需要把 preemption mode（抢占模式）设置为禁用。

Virtual Address

虚拟路由器的网络中，所有节点路由器虚拟机 IP 地址必须是相同的，例如：Master 路由器的真实 IP 地址和虚拟 IP 地址为 192.168.1.1，在 Backup 路由器虚拟 IP 地址为 192.168.1.1，但真实地址是 192.168.1.2，这样的配置是允许的，但不推荐之前已经提到过。

如果在虚拟路由器网络中关联了多个 IP 地址，Master 路由器配置多个虚拟 IP 地址，同样在 Backup 路由器也需要配置相同的虚拟 IP 地址。如果 Master 和 Backup 路由器虚拟 IP 地址配置不同，这样 VRRP 广播数据包将会被丢弃，导致 VRRP 无法正常工作。.

注意：不建议将 MikroTik 路由器的物理接口 IP 设置为实际网关 IP，即 VRRP 网络的虚拟 IP 地址和实际 IP 地址不要相同

在 IPv6 网络中，首选地址总与虚拟机路由网络相关的 link-local 地址。如果配置了多个 IPv6 地址，那么它们将被添加到广播数据包中的 link-local 地址之后。

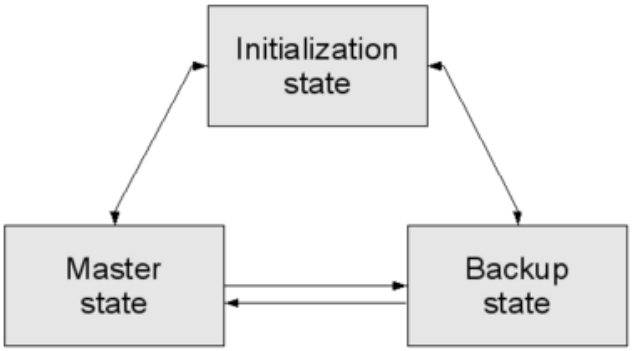
IPv4 ARP

给定虚拟路由的主路由器使用虚拟路由分配的 MAC 地址响应 ARP 请求。虚拟 MAC 地址用于虚拟路由的源 MAC 地址进行广播。对于非虚拟 IP 地址的 ARP 请求，路由器用系统 MAC 地址响应。备份路由器对虚拟 IP 的 ARP 请求不做响应。

IPv6 ND

在 IPv6 网络中，没用 ARP 协议，路由器的探测通过邻居发现协议，当路由器变为主路由，路由器通过主动 ND 邻居宣告发送到与虚拟路由相关联的每个 IPv6 地址

17.2 VRRP 状态器



从上图可以看到，每个 VRRP 节点，一共有三种状态：

- Init state（初始状态）
- Backup state（备份状态）
- Master state（主机状态）

初始状态

该状态目的是等待启动事件，当路由器 VRRP 事件被启动，这时会有下面的操作流程：

- 如果优先级是 255,
- * 对于 IPv4 发送宣告数据包和广播 ARP 请求
- * 对于 IPv6 发送一个主动 ND 邻居宣告与虚拟路由器光联的每一个 IPv6 地址，并且将目标地址设置为与 VR 相关的链路本地地址（link-local address）。
- * 转变为 MASTER 状态;
- 否则转变为 BACKUP 状态。

Backup 状态

当路由器在 Backup 状态时：

- 在 IPv4 网络中，Backup 节点不会响应 ARP 请求，也不会转发与虚拟路由相关的 IP 流量
- 在 IPv6 网络中，Backup 节点不会应 ND 邻居的请求，也不会发送 ND 路由器的宣告消息.

路由器的主要任务是接收广播的数据包，并检查 Mater 节点是否可用。在两种情况下，Backup 路由器将转换成 Mater 状态

- 如果优先级在宣告数据包被设置为 0;
- 当 Preemption_Mode 设置为 yes 时，宣告中的优先级低于本地优先级

在转换到 Master 状态后：

- 在 IPv4 网络中广播 ARP 请求；
- IPv6 网络中为每个相关的 IPv6 地址，发送一个主动的 ND 邻居广播为。

在其他情况下，宣告数据包将被丢弃。当收到关闭事件时，传送到初始化状态。

Master 状态

当 Master 状态启用，节点路由器将转发与之相关的 IPv4 和 IPv6 转发网络数据。在 IPv4 网络中，Master 节点将响应来至相关 IPv4 地址的 ARP 请求，在 IPv6 网络 Master 节点操作如下：

- 响应相关联的 IPv6 网络 ND 邻居请求信息；
- 发送相关联的 IPv6 网络 ND 路由器宣告信息

如果宣告信息数据包被 Master 接口接收到：

- 如果优先级为 0，会立刻发送宣告数据包，告诉其他虚拟路由器;
- 如果接收到的宣告数据包中的优先级大于本节点优先级，则转换到 Backup 状态
- 如果接收到的宣告数据包中的优先级等于节点优先级，则需参考发送方的主 IP 地址，如果大于本地主 IP 地址，则转换为 Backup 状态
- 如果是其他情况则忽略接收到的宣告数据包

当关闭操作接收到后，发送宣告数据包设置 priority=0，并转换到初始化状态

17.3 简单配置 VRRP

操作路径: /interface vrrp

IPv4

启用 VRRP 功能，只需要两部操作，创建 VRRP 接口和添加虚拟路由器 IP 地址，例如，添加 vrrp 到 ether1 接口，并设置虚拟路由器地址 192.168.1.1

```
/interface vrrp add name=vrrp1 interface=ether1
/ip address add address=192.168.1.1/32 interface=vrrp1
```

以上操作紧配置了 interface 参数到 vrrp 中，其他参数默认，即 vrid=1, priority=100 和 authentication=none，这些参数可以根据需要手动配置

在 VRRP 正确启用前，ether1 接口需要配置正确的 IP 地址，例如，ether1 配置 192.168.1.2/24

注意：如果在 VRRP 接口上配置的地址与路由器对应接口上的地址来自相同的子网，则 VRRP 接口上的地址必须设置为/32 掩码。

IPv6

确保 VRRP 能正确工作在 IPv6 网络中，有几个参数需要启用，v3 协议需求启用，并且协议要选择 ipv6，如下设置:

```
/interface vrrp add name=vrrp1 interface=ether1 version=3 v3-protocol=ipv6
```

当 VRRP 接口设置，可以添加 IPv6 地址，并启用 ND 宣告：

```
/ipv6 address add address=FEC0:0:0:FFFF::1/64 advertise=yes interface=vrrp1
```

不需要像在 IPv4 中那样在必须在对应接口上添加 IP 地址配置。IPv6 使用链路本地地址在节点之间进行通信。

属性描述

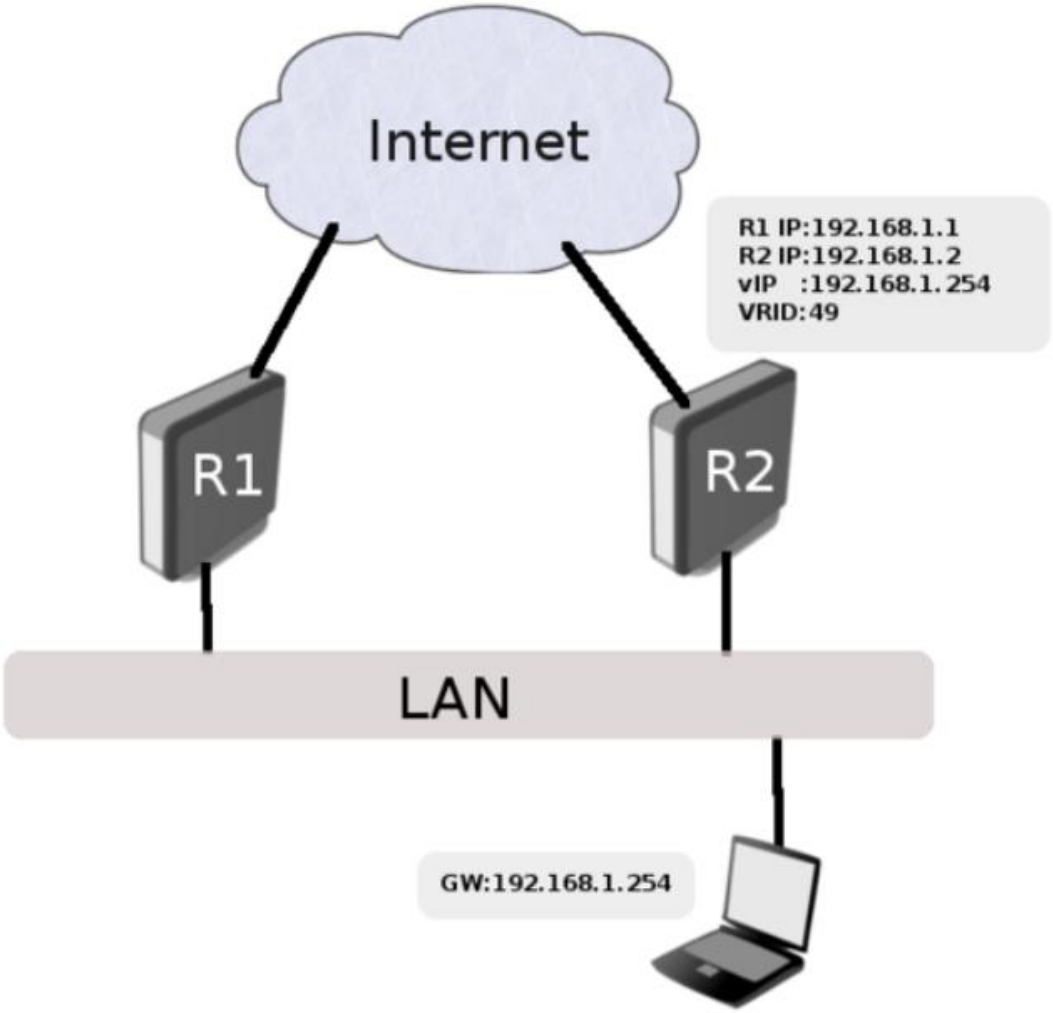
属性	描述
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; 默认: **enabled**)	ARP 协议模式
authentication (*ah* | *none* | *simple*; 默认: **none**)	验证模式用于 VRRP 协议之间的宣告数据包加密 • none – 一般用于安全要求不高的网络，例如两个 VRRP 节点在同一局域网内 • ah – IP 数据包头验证，该算法提供了强大的保护，防止配置错误，以及重复攻击和数据包损坏/修改。如果对 LAN 网络管理有要求，可以选择该模式 • simple – 使用简单的明文密码，较普通的安全防护和配置保护
interface (*字符*; 默认:)	加入并运行 VRRP 实例的接口名称
interval (*时间[10ms..4m15s]*; 默认: **1s**)	VRRP 更新事件周期，定义 Master 发送宣告数据包的更新频率。
mtu (*整型*; 默认: **1500**)	三层 MTU 值
name (*字符*; 默认:)	VRRP 接口名称
on-backup (*字符*; 默认:)	当节点切换到 Backup 状态时执行的脚本
on-master (*字符*; 默认:)	当节点切换到 Master 状态时执行的脚本
password (*字符*; 默认:)	认证密码。如果不使用身份验证，可以忽略。
preemption-mode (*yes* | *no*; 默认: **yes**)	Master 节点是否始终具有优先级。当设置为“no”时，即使 Backup 节点的优先级高于当前主节点，在当前 Master 节点失败之前也不会被选为 Master 节点。
priority (*整型: 1..254*; 默认: **100**)	VRRP 的优先节点选择算法，数值越高代表优先级越高。“255”保留给当前的虚拟路由器，而“0”则保留给 Master 路由器，用于释放权限
v3-protocol (*ipv4* | *ipv6*; 默认: **ipv4**)	选择 IP 协议，该属性仅适用于 VRRPv3 版本
version (*integer [2, 3]*; Default: **3**)	选择 VRRP 版本。
vrid (*integer: 1..255*; Default: **1**)	虚拟路由器的识别编号，每个虚拟路由器必须有一个唯一的 ID 编号。

17.4 VRRP 配置实例

下面的实例包含多 VRRP 配置应用

基本配置实例

下图是是一个基本的 VRRP 配置案例



如上图，R1 路由器为 Master，所有访问流量将通过 R1 到外网，在该实例中路由器 R2 始终为备份状态。但 R1 路由器故障或中断，R2 将接替 Master，处理并转发流量。

R1 配置:

```
/ip address add address=192.168.1.1/24 interface=ether1
/interface vrrp add interface=ether1 vrid=49 priority=254
/ip address add address=192.168.1.254/32 interface=vrrp1
```

R2 配置:

```
/ip address add address=192.168.1.2/24 interface=ether1
/interface vrrp add interface=ether1 vrid=49
```

```
/ip address add address=192.168.1.254/32 interface=vrrp1
```

测试

首先查看两台路由器的 vrrp 接口标识是否正确，在路由器 R1 应该会查看到“RM”标识，如下命令显示：

```
/interface vrrp print detail
 0   RM name="vrrp1" mtu=1500 mac-address=00:00:5E:00:01:31 arp=enabled
     interface=ether1 vrid=49 priority=254 interval=1 preemption-mode=yes
     authentication=none password="" on-backup="" on-master="" version=3
     v3-protocol=ipv4
```

在路由器 R2 上标识为“B”：

```
/interface vrrp print detail
 0    B name="vrrp1" mtu=1500 mac-address=00:00:5E:00:01:31 arp=enabled
     interface=ether1 vrid=49 priority=100 interval=1 preemption-mode=yes
     authentication=none password="" on-backup="" on-master=" version=3
     v3-protocol=ipv4
```

如同你看到的，vrrp 接口 MAC 地址在两台路由器都识别，现在检查 vrrp 工作是否正常，通过在客户端 ping 虚拟地址 192.168.1.254，查看 arp 信息：

```
[admin@client] > /ping 192.168.1.254
192.168.1.254 64 byte ping: ttl=64 time=10 ms
192.168.1.254 64 byte ping: ttl=64 time=8 ms
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 8/9.0/10 ms
[admin@client] /ip arp> print
Flags: X - disabled, I - invalid, H - DHCP, D - dynamic
 #   ADDRESS        MAC-ADDRESS      INTERFACE
 ...
 1 D 192.168.1.254   00:00:5E:00:01:31 bridge1
```

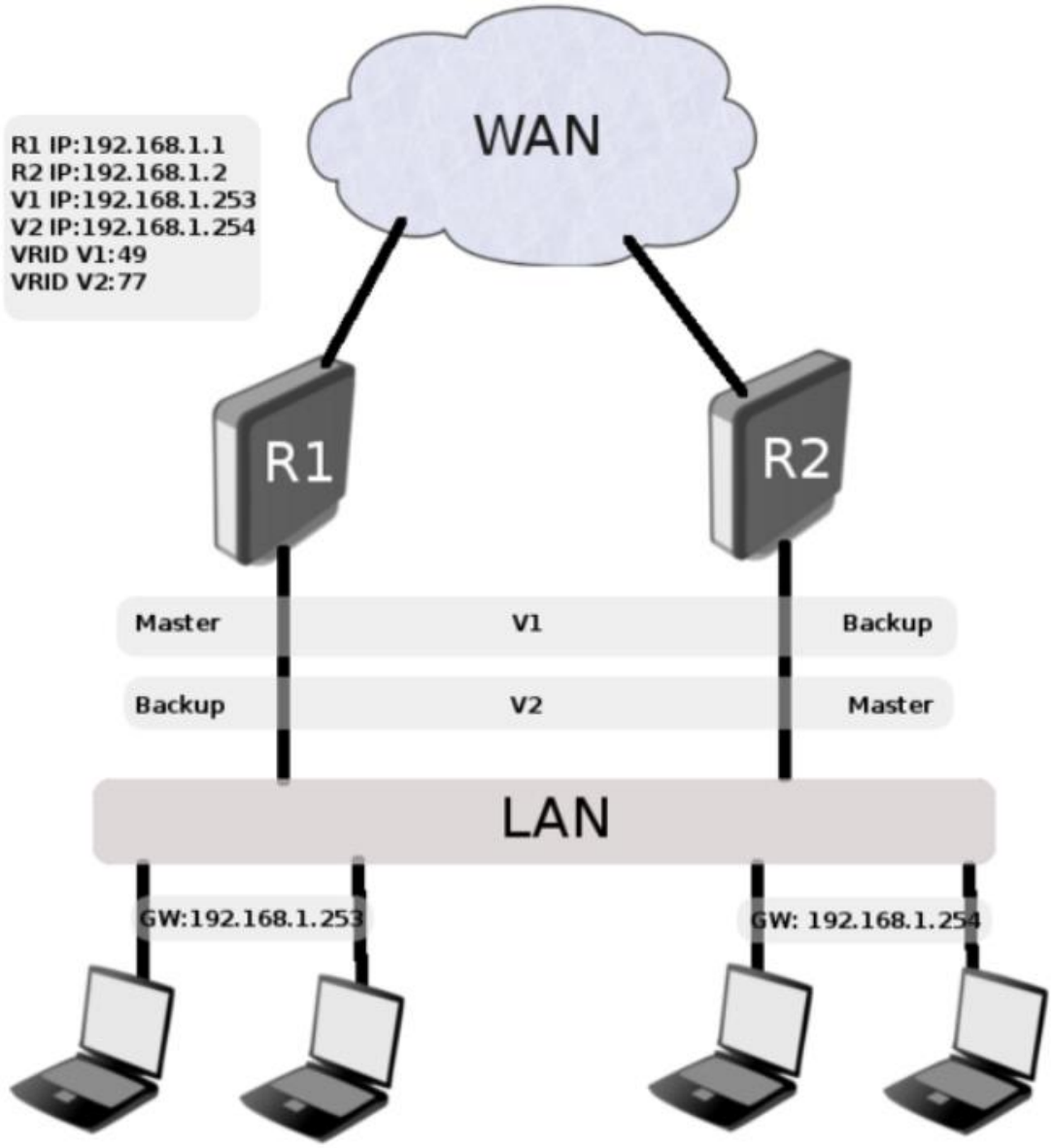
测试 VRRP 备份是否成功，可以拔掉路由器 R1 的 ether1 网线，R2 将接替 Master，ARP 中的 MAC 地址和 IP 将不会改变，但流量将从 R2 路由器转发

注意: 如果 VRRP 与反向路径过滤一起使用，那么建议将 rp-filter 设置为 loose，否则可能无法访问 VRRP 接口

负载共享(Load sharing)

在之前的基本配置实例中，R2 完全处于备份状态，这样的方式可能会被认为浪费了资源。为此我们也可以将 R2 设置为网关接入部分用户

这种配置的明显优点是建立了一个负载共享方案。但是这样做，R2 路由器不受当前 VRRP 设置的保护，为避免这样的情况，需要配置 2 组虚拟路由，如下：



在该实例，需要创建两组 VRRP 网络 V1 和 V2，两组建立了 2 个网关，分别是 192.168.1.253 和 192.168.1.254。一部分用户走 192.168.1.253，另一部分走 192.168.1.254。

配置 V1 虚拟路由与之前基本配置一样，R1 为 Master，R2 为 Backup，V2 虚拟路由则是 R2 为 Master，R1 为 Backup。通过这种配置，我们可以在 R1 和 R2 之间建立负载共享;此外，通过让两个路由器作为彼此的备份来创建一个保护设置。

R1 配置:

```
/ip address add address=192.168.1.1/24 interface=ether1
/interface vrrp add interface=ether1 vrid=49 priority=254
/interface vrrp add interface=ether1 vrid=77
/ip address add address=192.168.1.253/32 interface=vrrp1
/ip address add address=192.168.1.254/32 interface=vrrp2
```

R2 配置:

```
/ip address add address=192.168.1.2/24 interface=ether1
```

```
/interface vrrp add interface=ether1 vrid=49
/interface vrrp add interface=ether1 vrid=77 priority=254
/ip address add address=192.168.1.253/32 interface=vrrp1
/ip address add address=192.168.1.254/32 interface=vrrp2
```

VRRP 无抢占模式（no Preemption）

默认情况下每当优先级高的路由器可用时它就成为主路由器，即当优先级高的路由器恢复时，备份路由器会回切到高优先级的路由器，但有时候并不希望这样，我们可以关闭该模式，设置 preemption-mode=no。

我们仍然使用基本实例的配置作为参考，把 preemption-mode 设置为 no，修改操作如下：

```
/interface vrrp set [find] preemption-mode=no
```

我们可以测试关闭 R1 路由器，R2 将变为 Master，然后我们在恢复优先级高的 R1 路由器，这时 R2 路由器将继续作为 Master

第十八章 RouterOS 数据流(Packet Flow)

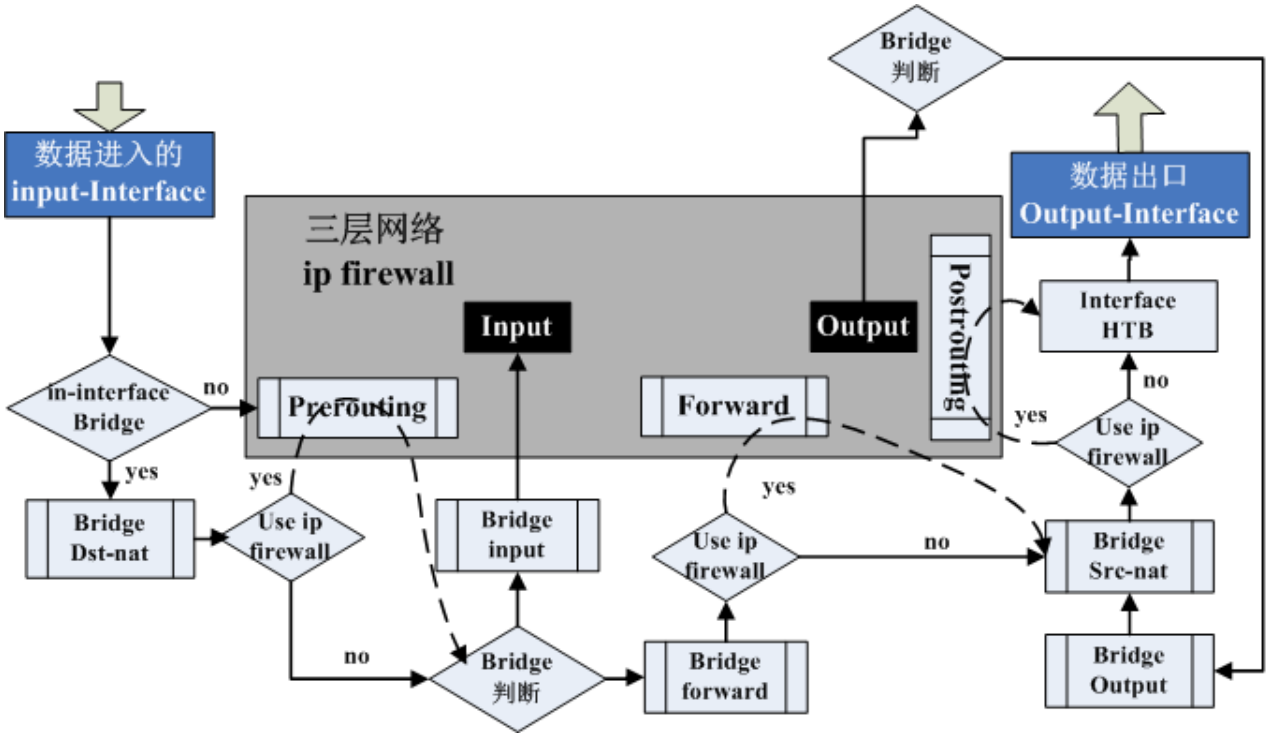
了解 RouterOS 的核心就需要对 IP 数据流进入 RouterOS 后是如何被处理的，这里将通过各种流程图来介绍 RouterOS 是如何处理 IP 数据流的过程，其实 RouterOS 本质是基于 Linux 的 netfilter 框架，因此属性 netfilter 的朋友能很快掌握 RouterOS 的数据流处理。需要注意的是 RouterOS 在 3.x 和 6.x 对数据流在内核升级后，有重大调整。

18.1 IP 数据流流程图

下面是 RouterOS 3.0 以上版本数据流原理图，RouterOS 3.0 以前这里不可能将所有的原理放到一个图中，所以我们将原理图分解成 2 部分，在 RouterOS3.0 后将二层网络和三层网络的流进行了分离，即使用 use-ip-firewall 属性选择是否分离。在 6.0 后的版本则重点对 Queue 进行调整。

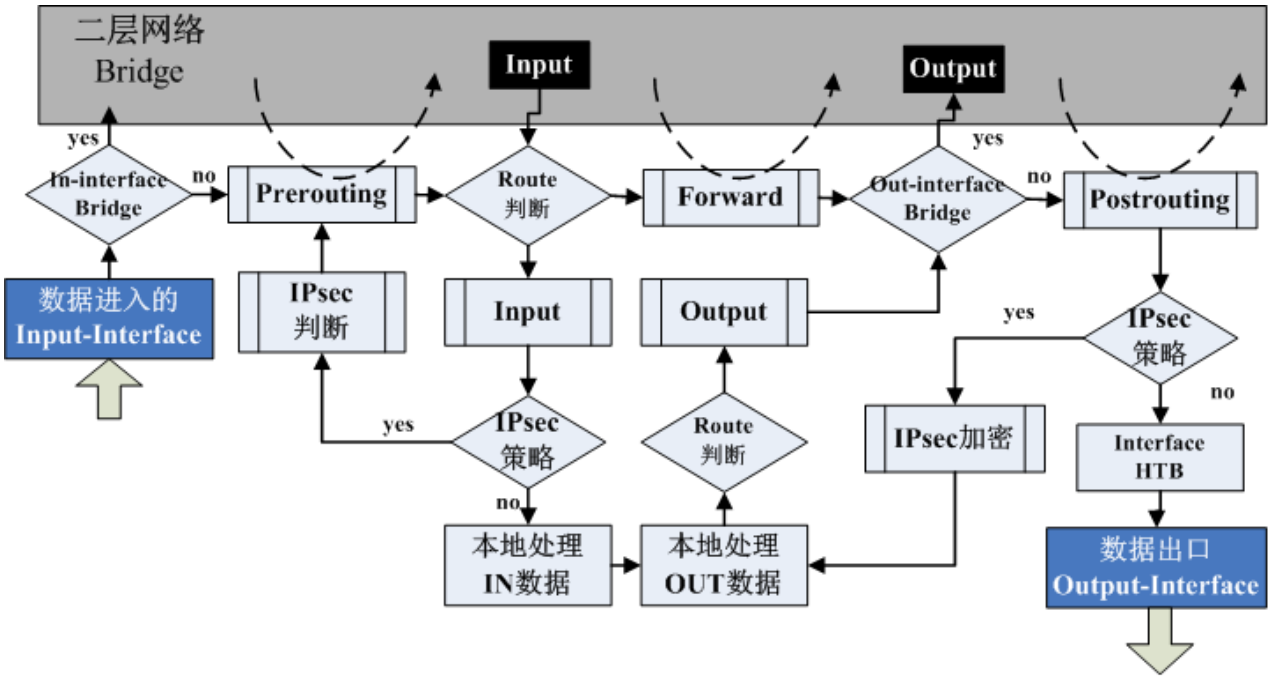
二层网络

在下面这个图中三层路由部分简化为一个“三层网络”的框内



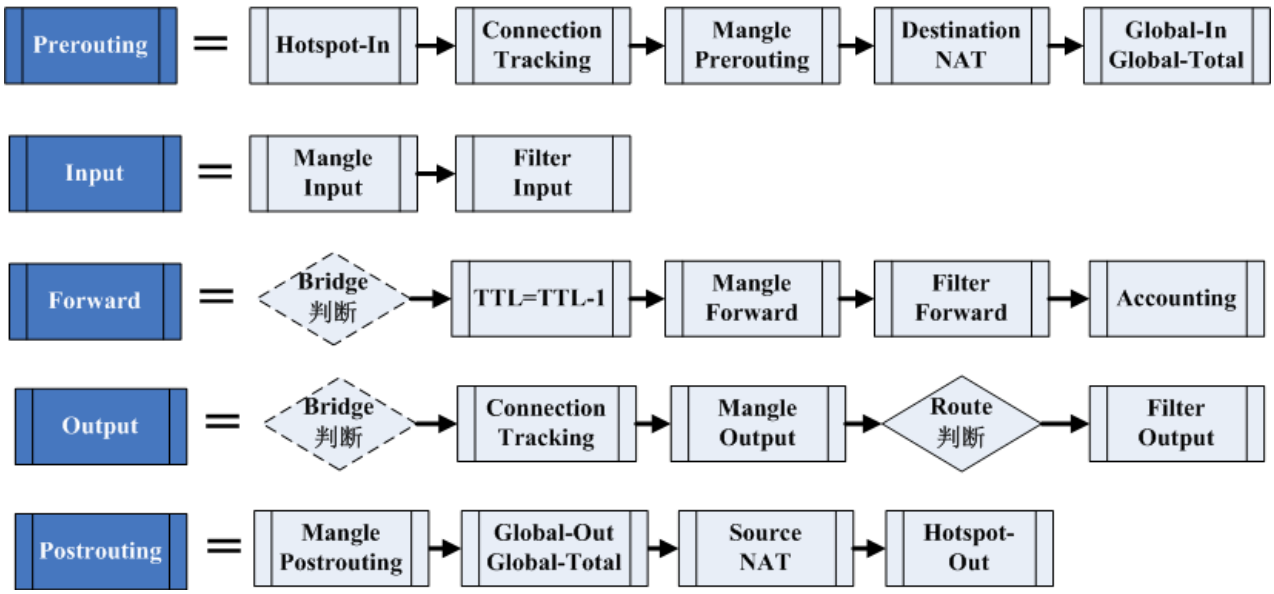
三层网络

在下面的图中，将二层网络或“Bridging”简化到框内



- **Input Interface** - 数据报进入路由器的起点，不论什么样的接口（物理接口或虚拟接口）数据报都会从这里开始进入路由器。
- **Output Interface** - 数据报离开路由器的终点，在之前经过或路由自发的数据报被发送出路由器。
- **Local Process IN** - 最终目的地是到达路由器自身的数据报。
- **Local Process OUT** - 由路由器自身发出的数据报。

下面的图是 mangle 中 prerouting、input、forward、output 和 postrouting 链表所包含的各个功能组件，该流程是 v6.0 版本前：



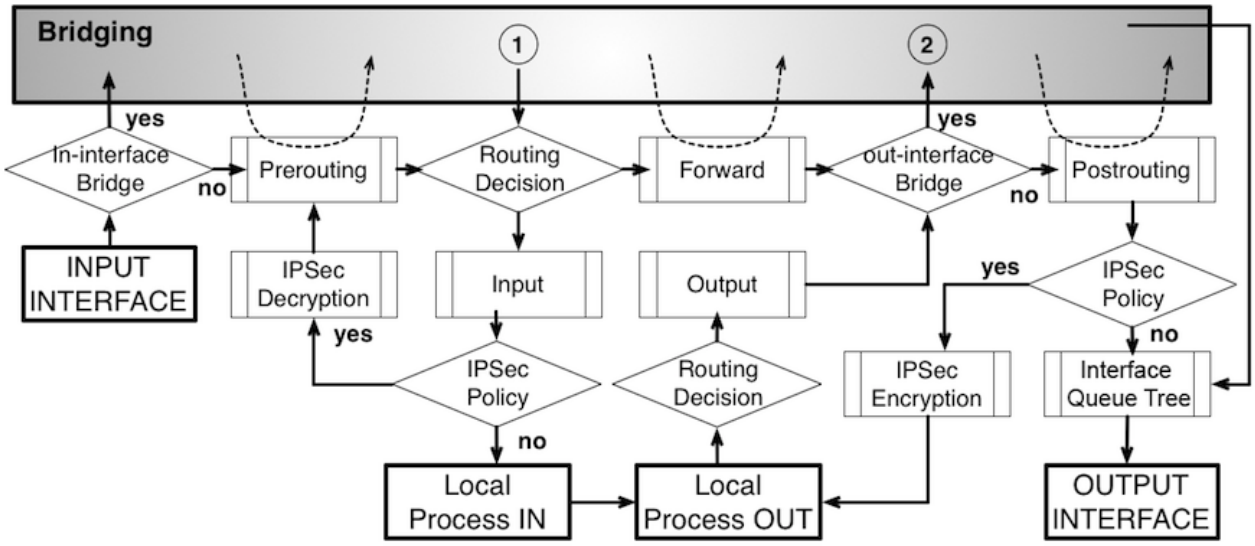
Mangle 中有 5 条链表：

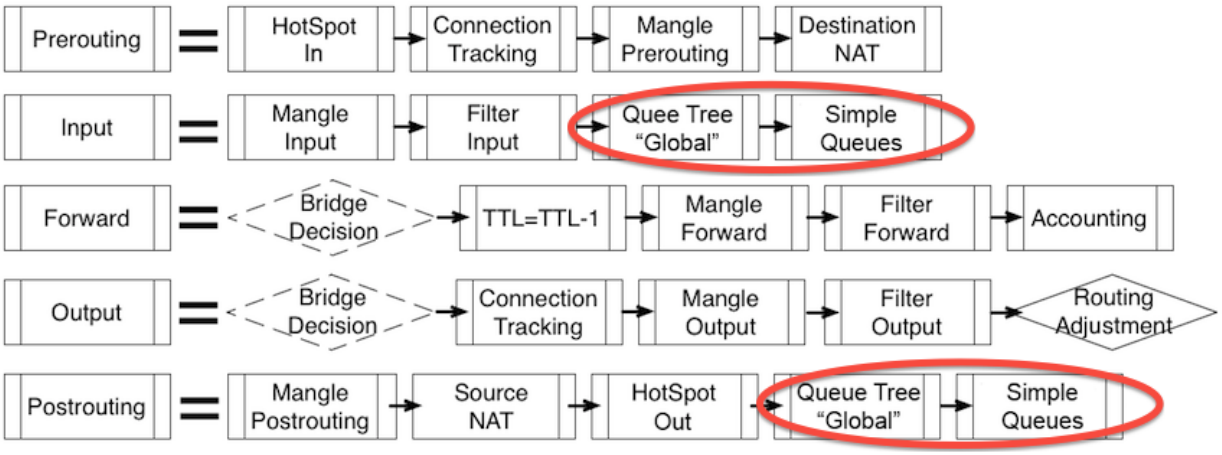
- Prerouting：数据包进入路由器后，在路由判断之前处理数据包。

- Postrouting：数据包确定了目标路由后，即路由判断后改变数据包。
- Output：在数据包被确定目的地址前改变数据包
- Input：在数据包进入本机后，应用程序接受之前改变数据包。
- Forward：第一次路由判断之后，最后一次路由判断前改变数据包。

18.2 RouterOS 6.0 对 Queue 调整后的数据流

注意，在 RouterOS6.0 后，对 Queue 做出来重大的修改，使得 Queue Simple 和 Queue Tree 形成双重 Queue，即处理数据流在队列中会被处理两次，具体配置介绍可以 Queue 章节

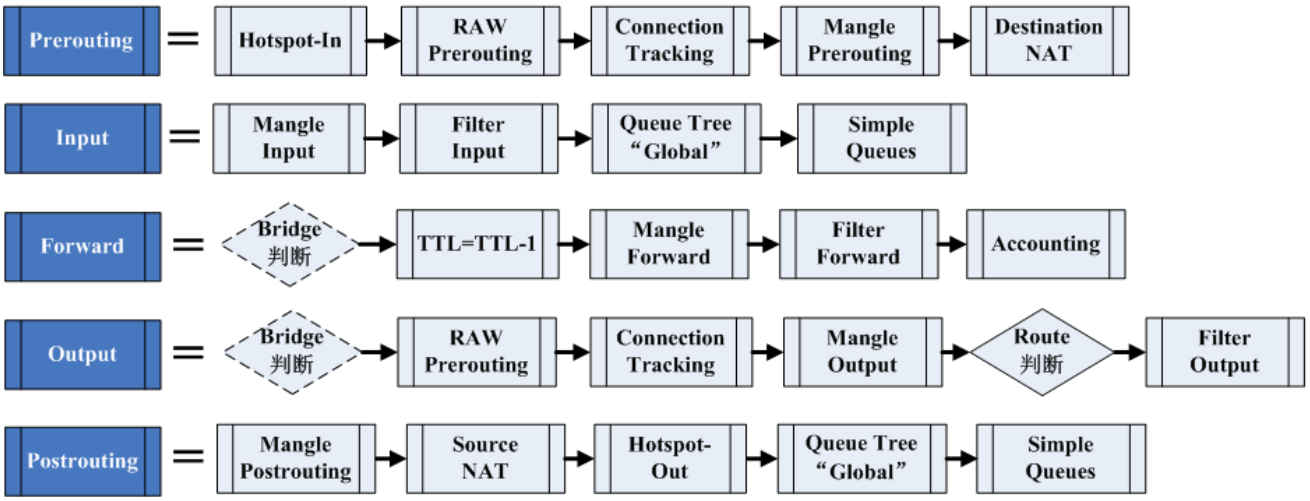




在流程图上可以看到，input（原来在 prerouting 链表）和 postrouting 下 simple queues 和 queue tree Global 从 Global-in/out 和 Global-total 中分离出来，官方的解释如下：

对于 simple queue 和 Global queue tree 传输流量能被两者分别独立的获取到，这样能给你建立双重 QoS 策略：一种是通过 mangle 标记流量，并应用到 queue tree 中对流量进行限制，即 HTB 流控；另一种是 PPP、Hotspot、RADIUS 等动态建立的 simple queues，或手动设置 simple queues，以及对每个用户流量限制的 PCQ 规则，也能允许"target"和"dst"选项建立每个用户限制，具体可以参考 12.5 章节。

在 6.36 版本在 ip firewall 新增了 Raw 列表，用于对 connection-tracking 控制，即能控制连接会话是否被跟踪，具体可以参考 10.8 章节



18.3 功能模块与结构

每一个功能模块在 RouterOS 中指定的目录下对应不同的功能，这些模块可以对应到相应的 RouterOS 操作路径。

- **Connection Tracking** - /ip firewall connection tracking 连接跟踪
- **Filter Input/Forward/Output** - /ip firewall filter 防火墙过滤
- **Souce NAT 和 Destination NAT** - /ip firewall nat 地址转换策略
- **Mangel Input/Forward/Output/Postrouting** - /ip firewall mangle 标记策略
- **Global-in/out/Total 和 interface HTB** - /queue simple 和 /queue tree 流控策略
- **IPsec policy** - /ip ipsec IPSec 策略
- **Accounting** - /ip accounting 访问日志记录
- **Use IP firewall** - /interface bridge settings 当启用桥接后，该功能才能生效，如果 Use IP Firewall 设置为 Yes，则数据流将进入 Layer-3 层处理。
- **Bridge Input/Forward/Output** - /interface bridge filter 二层桥接过滤
- **Bridge Dst-nat/Src-nat** - /interface bridge nat 二层桥接 nat 策略

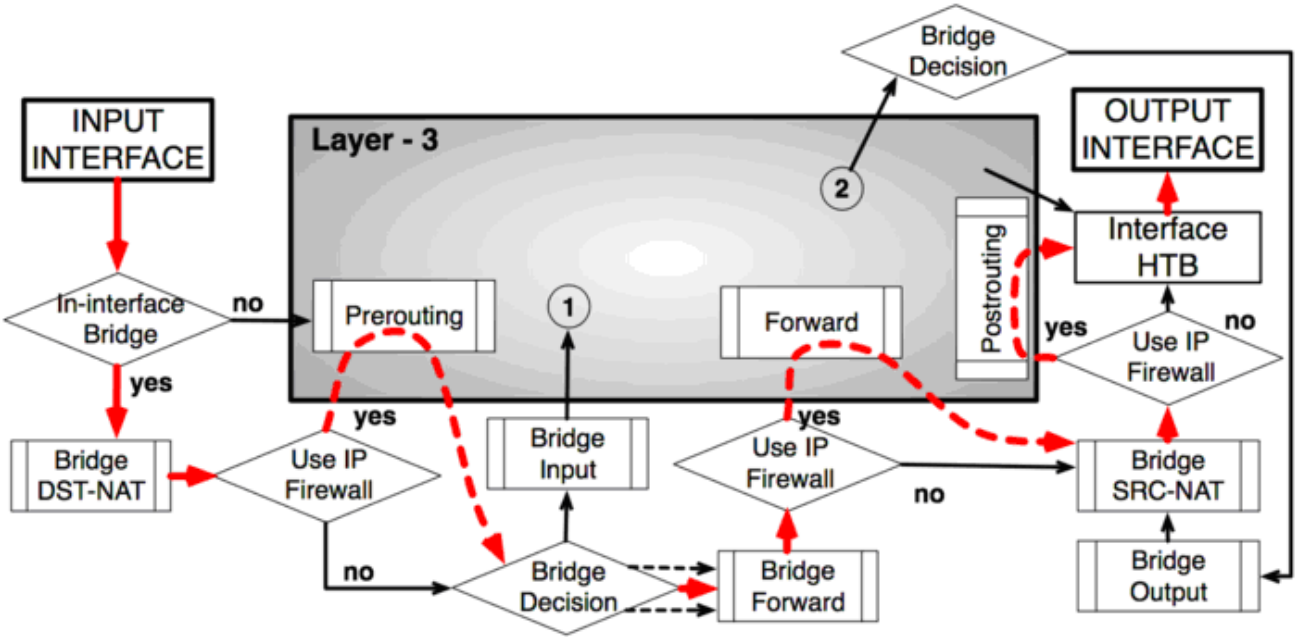
判断处理

- **In-interface Bridge** - 判断进入接口是否为桥接类型。
- **HotSpot-in** - 运行抓取传输 Hotspot 特征数据，否则从连接跟踪丢弃掉。
- **Forward Bridge 判断（Decision）** - 桥通过 MAC 地址列表查找数据报所匹配的目标 MAC 地址，当目标被找到，数据报将会被发送到相应的桥接口，如果没有匹配则会将数据报复制多份放送到所有的桥接端口。
- **Output Bridge 判断（Decision）** - 桥接下目标 MAC 属性判断，是否允许指向“out-bridge-port”

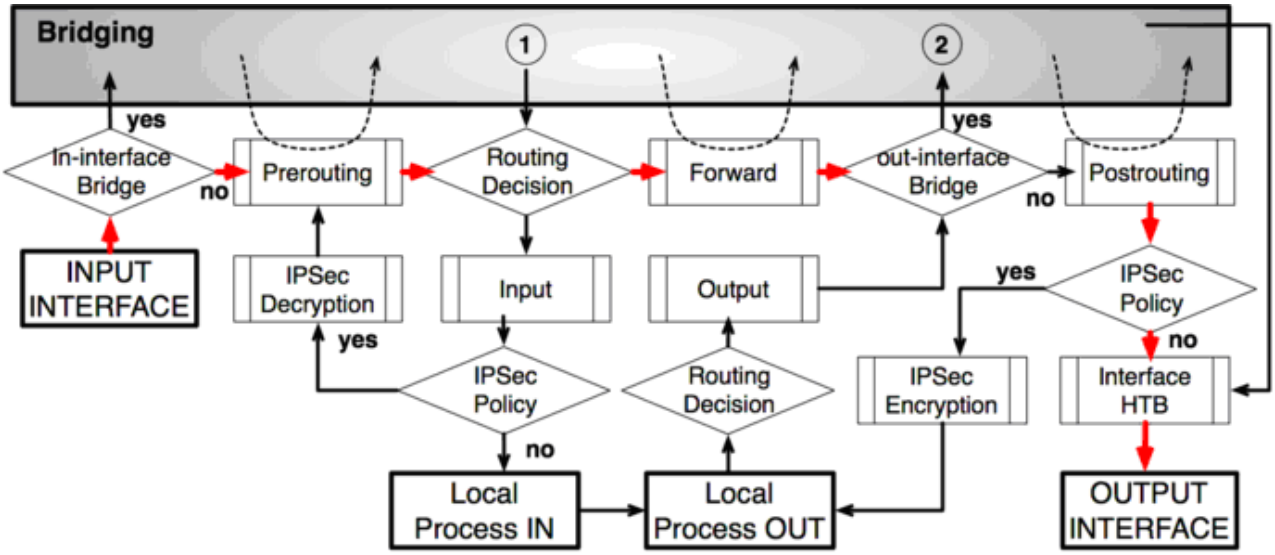
- **Route 判断（Routing Decision）** - 路由器通过路由命令查找一个匹配数据报的目标 IP 地址。当查找到后，数据报将发送到对应的端口或者路由器本身。如果在该事件中没有找到匹配路径，数据报将会被丢弃。
- **TTL-1** – 当路由得到准确的目的地，TTL 值被减少 1。如果 TTL 值变为 0，IP 包将会被丢弃掉。
- **IPSec Decryption/Encryption** - IPsec 判断，解密和加密。
- **Out-Interface Bridge** - 判断实际输出接口是否为桥接端口或判断输出接口是否为桥。
- **HotSpot-Out** - 撤销所有通过 Hotspot-in 的数据报操作，并发送回客户端。

18.4 功能处理流程

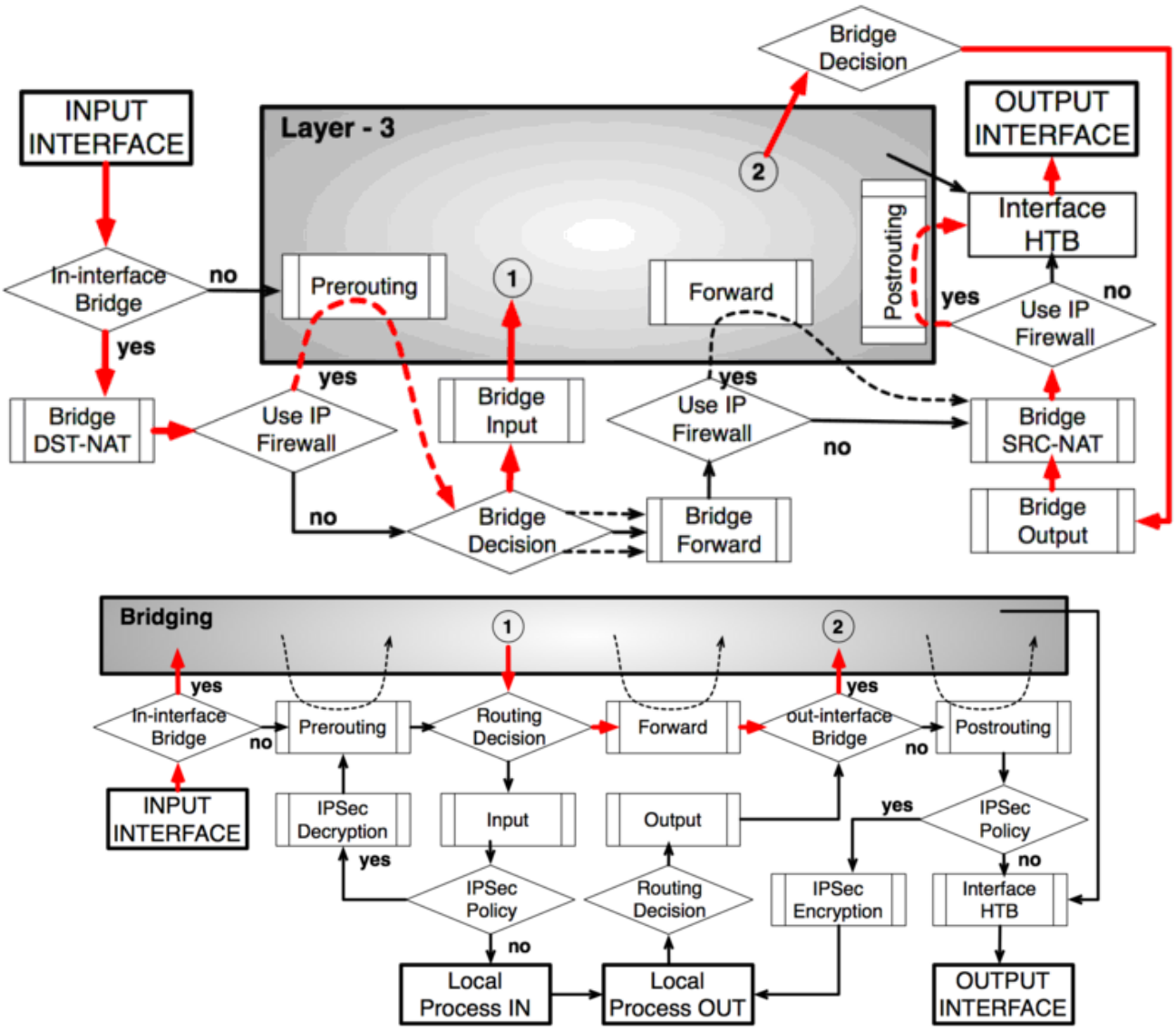
Bridge 设置 use-ip-firewall=yes



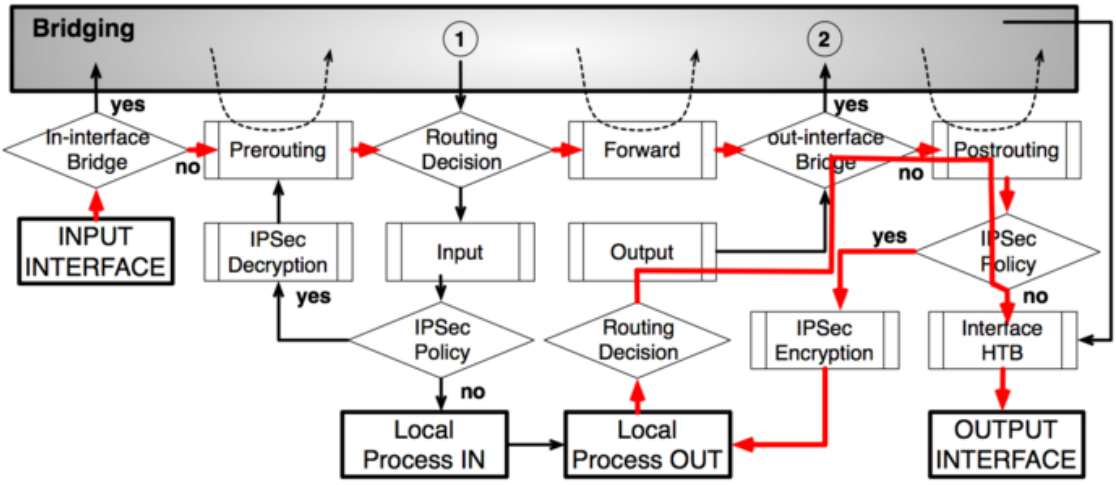
路由 – 从 Ethernet 到 Ethernet 接口



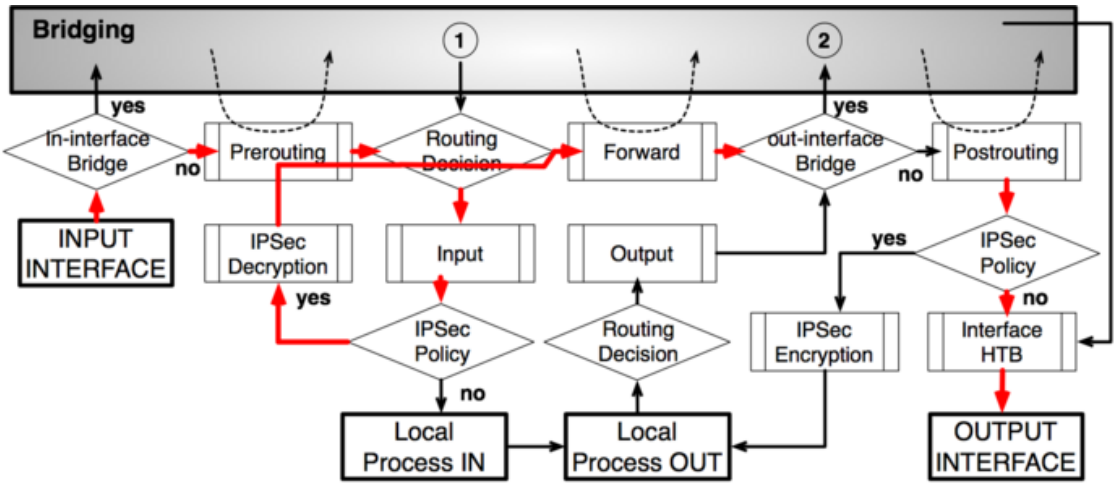
路由 - 从一个桥接口道不同的桥接口



IPsec 加密



IPsec 解密


Bridging
1
2
yes
In-interface Bridge
no
Prerouting
Routing Decision
Forward
yes
out-interface Bridge
no
Postrouting
INPUT INTERFACE
IPSec Decryption
yes
Input
Output
yes
IPSec Policy
no
IPSec Policy
no
Routing Decision
IPSec Encryption
Interface HTB
Local Process IN
Local Process OUT
OUTPUT INTERFACE

第十九章 RADIUS

RADIUS 是（Remote Authentication Dial-In User Service）的简称，翻译是远程认证拨号用户服务。即由远程服务器提供各种类型的验证和账号，是一个客户与服务器协议和软件，它使远程访问服务器能够与中心服务器通信，以鉴别拨号用户并且授权他们访问请求的系统和服务。（PS：从单词翻译 Radius 为“半径”，无线里面会涉及到，因此需要区分下）。RADIUS 提供了验证和账号服务为 ISP 或网络管理员对大型网络的用户访问和认证提供了方便。

当 RADIUS 服务器启用，就作为 NAS（Net Access Server）服务器，任何运行 RADIUS 客户端应用的网络设备都可以成为 RADIUS 的客户端。RADIUS 协议认证机制灵活，可以采用 PAP、CHAP 或者 Unix 登录认证等多种方式。例如 RouterOS 作为 PPPoE BAS 认证设备接入一台 RADIUS 服务器的 NAS，NAS 使用 Access-Require 数据报向 RADIUS 服务器提交用户信息，包括用户名、密码等相关信息。

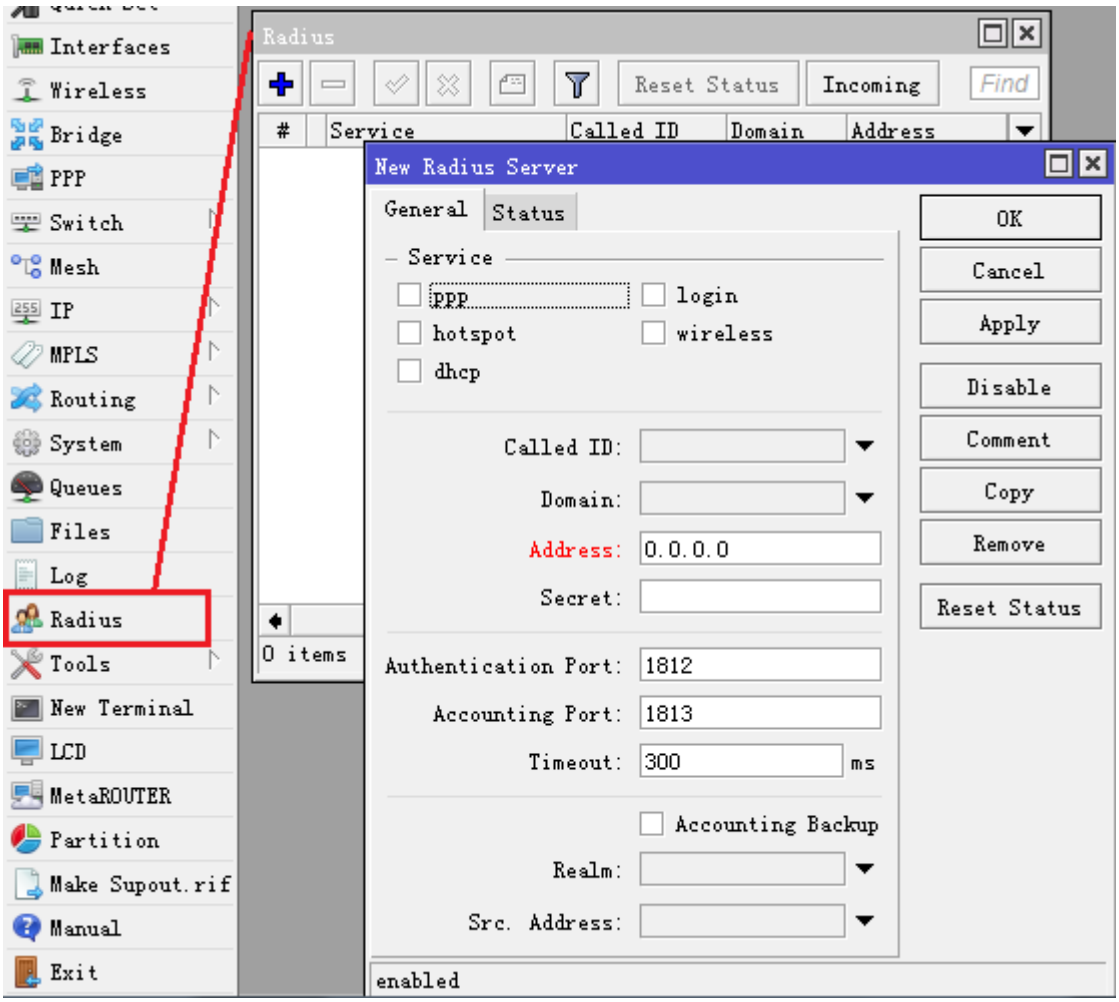
MikroTik RouterOS 提供了一个 RADIUS 客户端，能对接 RADIUS 服务器，并为 HotSpot, PPP、PPPoE、PPTP 和 L2TP 连接。 RADIUS 服务器通过本地数据库存储用户信息和策略参数，例如 RouterOS PPP 中的 Profile 配置有用户认证的相关信息和策略，当认证信息传输访问到 RADIUS 服务器，会从 RADIUS 数据库中查询相关信息，并返回。在 RouterOS 当本地数据库没有匹配到用户记录，RADIUS 服务器数据库才会被访问

19.1 RADIUS 客户端

MikroTik RouterOS 提供了一个 RADIUS 客户端，能对接 RADIUS 服务器，并为 HotSpot, PPP、PPPoE、PPTP、L2TP 和 ISDN 连接。 RADIUS 服务器通过本地数据库存储用户信息和策略参数，例如 RouterOS PPP 中的 Profile 配置有用户认证的相关信息和策略，当认证信息传输访问到 RADIUS 服务器，会从 RADIUS 数据库中查询相关信息，并返回。在 RouterOS 当本地数据库没有匹配到用户记录，RADIUS 服务器数据库才会被访问。

操作路径: /radius

在 winbox 中，可以找到 Radius 菜单，这里可以添加和删除 RADIUS 服务器配置



如上图可以看到 RADIUS 客户端能指定对应 RADIUS 服务器的服务类型，如 ppp、login、hotspot、wireless 和 dhcp 等

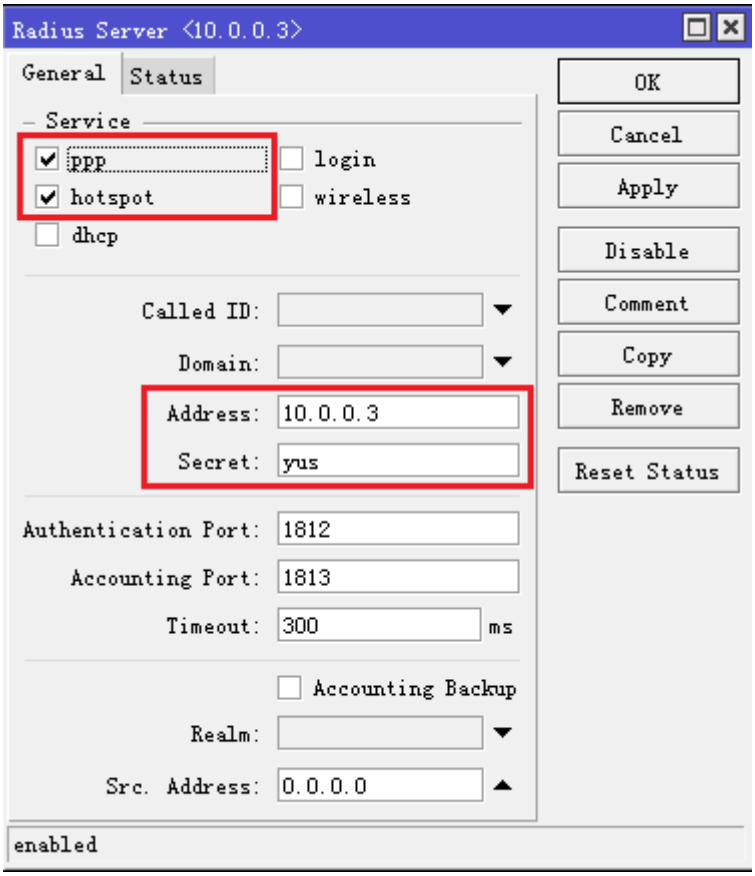
RADIUS 客户端属性

属性	描述
accounting-backup (*yes* | *no*; 默认: **no**)	是否配置备份 RADIUS 服务器
accounting-port (*整型 [1..65535]*; 默认: **1813**)	RADIUS 服务器计费端口
address (*IPv4/IPv6 地址*; 默认:**0.0.0.0**)	连接 RADIUS 服务器的 IPv4 或 IPv6 地址
authentication-port (*整型 [1..65535]*; 默认: **1812**)	RADIUS 服务器验证端口
called-id (字符串; 默认:)	该值应用于 PPP 协议，PPPoE 为服务名（service name），PPTP 为服务器 IP 地址，L2TP 为服务器 IP 地址
comment (*字符串*; 默认:)	注释
disabled (*yes* | *no*; 默认: **no**)	禁用或启用
domain (*字符串*; 默认:)	设置 Microsoft Windows 客户端域

realm (*字符串*; 默认:)	明确的声明范围 (用户域)
secret (*字符串*; 默认:)	共享密钥用于访问 RADIUS 服务器
service (*ppp|login|hotspot|wireless|dhcp*; 默认:)	将用于 RADIUS 服务器的路由器服务: ▪ hotspot – 热点认证服务 ▪ login – 账号登录路由器认证 ▪ ppp - Point-to-Point 客户端认证 ▪ wireless – 无线客户端 MAC 地址认证 ▪ dhcp – DHCP 客户端 MAC 地址认证
src-address (*ipv4/ipv6 地址*; 默认:**0.0.0.0**)	发送给 RADIUS 服务器的源 IP/IPv6 地址
timeout (*time*; 默认: **100ms**)	超时重发时间

注意:当 RADIUS 服务器验证用户的 CHAP、MS-CHAPv1 和 MS-CHAPv2,将不会使用共享 secret。Secret 仅被用于验证回复,路由器会验证用户的 CHAP、MS-CHAPv1 和 MS-CHAPv2。如果你有错误的共享 secret, RADIUS 服务器将接受请求,但路由不会接受回复,你可以通过/radius monitor 命令,"bad-replies"的数据会增加,代表有人在不断尝试连接

下面是设置 Hotspot 和 PPP 的用户认证,连接 RADIUS 服务器配置。假设 RADIUS 服务器 IP 为 10.0.0.3, secret 安全为 yus,配置如下:



如上图可以看到 RADIUS 客户端能指定对应 RADIUS 服务器的服务类型,如 ppp、login、hotspot、wireless 和 dhcp 等

RADIUS 连接终止

操作路径: /radius incoming

此功能支持从 RADIUS 服务器发送的非请求报文，非请求报文是 RADIUS 协议的扩展命令，允许终止 RADIUS 服务器上已经认证在线账号的会话。即发出 DM（Disconnect-Messages），强制用户下线。

注意：RouterOS 不支持其他类型 RADIUS 类似的 Disconnect Messages 请求，如 POD (Packet of Disconnect)

属性	描述
accept (*yes* | *no*; 默认: **no**)	是否接受非请求报文
port (*整型*; 默认: **1700**)	设置监听的端口

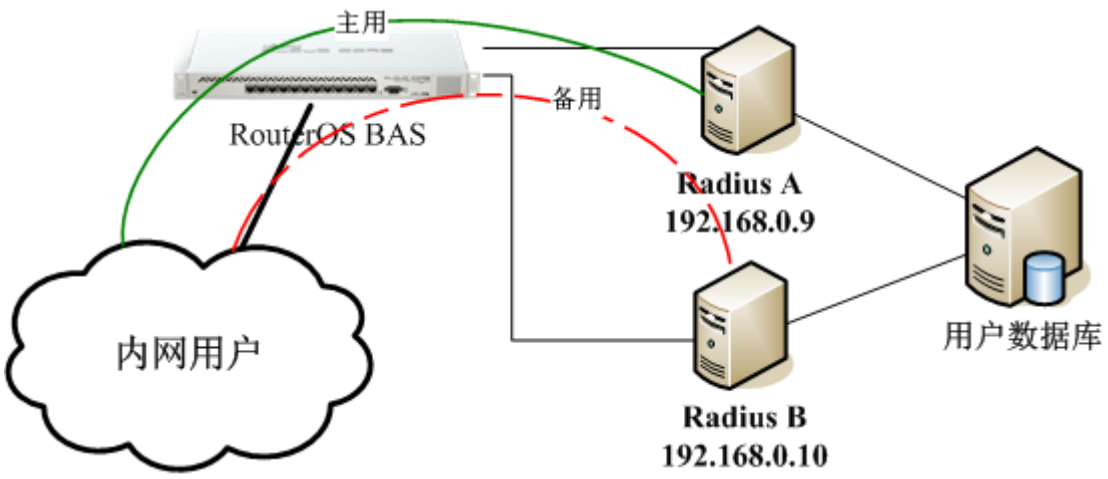
19.2 RouterOS 连接 RADIUS 备份

RADIUS 服务器对用户名和密码的合法性进行检验，必要时可以提出一个疑问，要求进一步对用户认证，也可以对 NAS 进行类似的认证；如果合法，给 NAS 返回 Access-Accept 数据报，允许用户进行下一步工作，否则返回 Access-Reject 数据报，拒绝用户访问；如果允许访问，NAS 向 RADIUS 服务器提出计费请求 Account-Require，RADIUS 服务器响应 Account-Accept，对用户的计费开始，同时用户可以进行自己的相关操作。RADIUS 是从数据库中读取用户相关信息，验证成功后，返回给 RouterOS BAS，RADIUS 连接的数据库，可以是 Mssql、Mysql 或 oracle 等。数据库建立可以和 RADIUS 在同一服务器，也可以是通过网络连接。

这里如何实现 RouterOS BAS 认证的 RADIUS 备份，在 RouterOS 中配置很简单，但后端的 RADIUS 和数据库信息需要保证一致。

RADIUS 备份模式

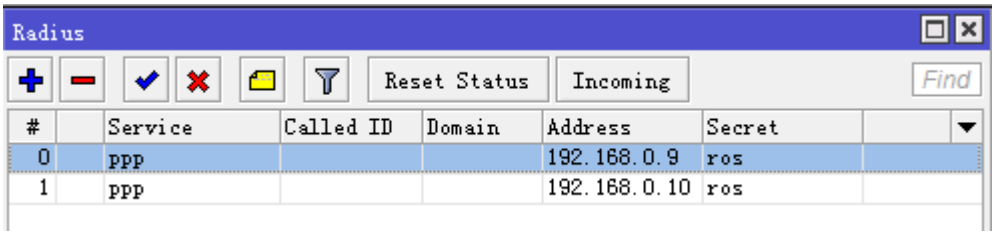
即 2 台或 2 台以上的 RADIUS 组成的 RADIUS 备份集群，如下图



在这里我们有 2 台 RADIUS，分别是 A：192.168.0.9 和 B：192.168.0.10，两台 RADIUS 分别连接到用户数据库。

RouterOS 连接 RADIUS 是顺序执行，当第一台 RADIUS 没有对用户请求响应或通过，会继续向后面的规则发出请求。这样实现 RADIUS 请求的备份功能。

当然 RouterOS 配置也很简单，仅仅加入两个 RADIUS 连接即可：



当然这个 RADIUS 集群还存在数据库的单点故障，可以通过数据库同步和配置备份数据完成。

第二十章 HotSpot 热点认证网关

20.1 HotSpot 介绍

HotSpot 是一种通过要求用户认证来访问某些网络资源的方法。用户可以使用几乎任何网页浏览器（HTTP 或 HTTPS 协议）登陆，所以他们不需要安装任何附加的插件。RouterOS 会自动计算正常运行时间以及每个客户使用的流量，并且也能把这个信息发送到 RADIUS 服务器。HotSpot 系统可以限制每个特定用户的比特率，总流量，运行时间以及涉及的其他参数。

Hotspot 热点 web 服务认证是一种友好的 web 方式的认证系统，在此种认证方式中，系统将自动要求未认证用户打开认证网页，验证通过后，便可连接到因特网，未认证用户无论输入任何一个网站地址，都会被强制到一个认证接口，要求用户进行认证。配置了 Walled Garden 特性后，允许用户不需要提前认证就可以访问一些网页。

获取地址

首先，一个客户必须先获得一个 IP 地址。它可以通过设置被静态 IP 地址，或者获取一个 DHCP 服务器的所分配的 IP 地址。如果需要的话，DHCP 服务器可以提供绑定分发 IP 地址到客户 MAC 地址的途径。

此外，HotSpot 服务器能自动分配给任何客户端的虚拟 IP 地址，分配来自 Hotspot 建立的 IP 池。这个特性对那些不愿意修改 IP（或不清楚，缺乏网络技术）。如果用户和 Hotspot 网关不在相同子网，Hotspot 通过 ARP 广播的方式强迫分配一个 IP 给用户，用户不会注意到这个转变（例如：在用户配置不会有任何改变），但路由器本身则看到完全不同（在 hotspot host 中可以看到被转化了的地址），这项技术叫做一对一 NAT，但它也以 RouterOS 2.8 版本中叫做的“即插即用”。

一对一 NAT 接收来自已连接网络接口的任何向内地址，并完成一个网络地址翻译。客户可以使用任何预先配置的地址（注意要求配置网关）。如果一对一 NAT 特性被设置为翻译一个客户的地址为一个公网 IP 地址，那么这个客户就甚至可以运行一个服务器或任何其他需要公网 IP 地址的服务。这个 NAT 将在数据报被路由器接收后就立即改变包的源地址。

注： 使用一对一 NAT 时，必须在该接口上启用 **arp** 模式。

认证之前

当在一个接口上启用 HotSpot 时，系统自动配置对所有未登陆用户显示登陆页面。这个是通过添加动态目的 NAT 规则完成的，你可以在一个运行中的 HotSpot 系统上观察的到。这些规则是用来把未认证用户的所有 HTTP 及 HTTPS 请求重定向到 HotSpot servlet（认证过程，例如：登陆页面）。其他一些规则将在该章后面专门部分进行讲述。

在配置好 Hotspot 后，打开任何 HTTP 页面都会产生 HotSpot servlet 登陆页面（可以通过自行定义登陆页面），所有访问 Hotspot 网关以外的资源，都会跳转到登陆页面，因此必须在 HotSpot 网关配置一个合法的 DNS。

Walled Garden

有时希望对某些目标应用不做认证（例如让客户不需要认证就访问公司的服务器和 OA 系统），或者指定服务要求认证（例如，用户访问一个内部文件服务器或其他限制区域）。这些都可以通过 Walled Garden 系统实现。

当一个未登陆用户请求 Walled Garden 中允许的服务时，HotSpot 网关不会阻拦它，或者是把 HTTP 请求重定向到原来的目的（或定向到一个指定的父级代理）。

为了执行 Walled Garden 对 HTTP 请求的特性，专门设计了一个嵌入的 web 代理服务器，所有来自未认证用户的请求是从这个代理通过。注意嵌入的代理服务器还没有高速缓存功能。还要注意这个嵌入代理服务器是在 **system** 软件功能包里并不需要 **web-proxy** 功能包。它是在**/ip proxy** 下面配置的。

认证

现在有 5 种不同的认证方法。你可以同时使用一个或多个：

- **HTTP PAP** - 最简单的方法。显示 HotSpot 登陆页并以纯文本格式获取认证信息（如：用户名和密码）。注意当在网络传输时，密码是没有加密的。
- **HTTP CHAP** - 标准方式，在登陆页包含了 CHAP 询问。CHAP MD5 散列询问与用户密码一起使用来计算将被发送到 HotSpot 网关的字符串。散列结果（作为一个密码）与用户名一起通过网络发送到 HotSpot 服务器（所以，密码是从来不以纯文本格式通过 IP 网络发送的）。在客户端，MD5 算法通过 JavaScript applet 执行，所以如果一个浏览器不支持 JavaScript（比如，Internet Explorer 2.0 或一些 PDA 浏览器），将不能认证用户。可以允许未加密密码，即打开 HTTP PAP 认证方式被接受，但并不推荐使用这个特性（出于安全考虑）。
- **HTTPS** - 与 HTTP PAP 一样，但对加密传输使用了 SSL 协议。HotSpot 用户只发送没有附加散列的密码（没有必要担心明文密码在网络上的泄露，因为传输本身是加密的）。另一种情况，HTTP POST 方法（如果不可能，那么用 HTTP GET 方法）用于向 HotSpot 网关发送数据。
- **HTTP cookie** - 在每次成功登陆之后，会有一个 cookie 发送到 web 浏览器，同时被添加到活动 HTTP cookie 列表。这个 cookie 将与存储在 HotSpot 网关的相比较，并仅当源 MAC 地址及随机生成的 ID 与存储在网关的相匹配。这个方法只可以与 HTTP PAP， HTTP CHAP 或 HTTPS 方法一起使用，不然的话没有其他方式可以产生 cookie。
- **MAC address** - 将用客户端的 MAC 地址与用户账号同时作为用户名。

HotSpot 可以通过询问本地用户数据库或 RADIUS 服务器认证用户（本地数据库会被先询问，然后是 RADIUS 服务器）。如果通过 RADIUS 服务器认证 HTTP cookie，那么路由器将在 cookie 被第一次产生时发送相同的信息到服务器。如果认证在本地完成，那么符合该用户的信息将会被调用， 否则将会调用 RADIUS 中的参数。如果要知道更多关于 RADIUS 服务器工作的信息，请参见其相应的 RADIUS 手册。

HTTP PAP 方法也使得通过请求页 `/login?username=username&password=password`。如果你想使用 telnet 连接登陆，准确的 HTTP 请求应该这样：**GET /login?username=username&password=password HTTP/1.0**

配置菜单

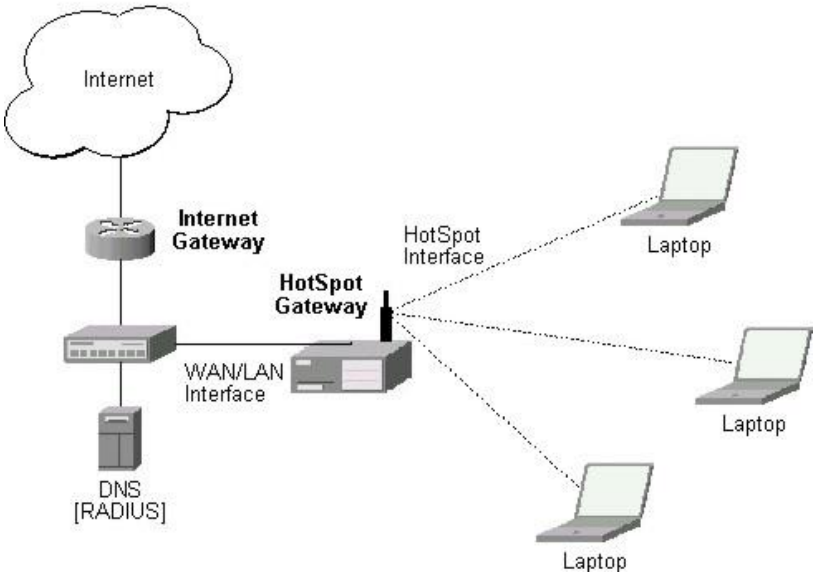
- **/ip hotspot** - HotSpot 上的特定接口（每个接口一个服务器）。HotSpot 服务器必须添加在这个目录中，HotSpot 系统才能够在一个接口上工作。
- **/ip hotspot profile** - HotSpot 服务器概要。影响 HotSpot 客户登陆过程的设置在这里进行。多个 HotSpot 服务器可以使用同样的概要信息。
- **/ip hotspot host** - 所有 HotSpot 接口上的活动网络主机的动态列表。在这里你可以找到 IP 地址与一对一 NAT 的绑定
- **/ip hotspot ip-binding** - 将 IP 地址绑定到主机 HotSpot 接口的规则
- **/ip hotspot service-port** - 一对一 NAT 地址翻译助手
- **/ip hotspot walled-garden** – HTTP 等级的 Walled Garden 规则(域名， HTTP 请求的 URL)
- **/ip hotspot walled-garden ip** – IP 等级的 Walled Garden 规则 (IP 地址，IP 协议)

- **/ip hotspot user** –本地 HotSpot 系统用户
- **/ip hotspot user profile** – 本地 HotSpot 系统用户组规则
- **/ip hotspot active** - 所有已认证 HotSpot 用户的动态列表
- **/ip hotspot cookie** - 所有合法的 HTTP cookie 动态列表

下面是一个简单的 Hotspot 事例，HotSpot 网关应该至少有两个网络接口：

1. HotSpot 接口，用于连接 HotSpot 客户
2. LAN/WAN 接口，用于访问网络资源。例如：DNS 和 RADIUS 服务器应该可达

下面的图表显示了一个简单的 HotSpot 设置。



HotSpot 接口应该分配一个 IP 地址。物理网络连接应该建立在 HotSpot 用户的计算机和网关之间。它可以是无线（无线网卡需要在 AP 上注册），或者有线的（NIC 网卡需要连接到一个集线器或一个交换机）。

当 ISP 需要在有线或者无线网络中建立 Hotspot 热点认证系统，如：小区、酒店、机场和其他公共场所。一个普通的 Hotspot 网络建立在一个外网接口和一个内部网络接口下，我们需要对内网用户作认证上网。

注：在 2.9 版本的 RouterOS Hotspot 功能包采用的是端口代理的方式连接，在启用 Hotspot 接口后 UpNp 即插即用功能自动开启，通过在/ip hotspot host 列表中可以查询相应的信息。

20.2 HotSpot Setup 向导配置

Hotspot 提供了一个向导命令 setup，可以根据向导提示配置 hotspot 服务，路由器通过/ip hotspot setup 命令，配置后向导会询问你设置 hotspot 服务的相关参数，一步一步的完成 Hotspot 服务器配置。注意：启用 Hotspot 认证前请配置好网络接口 IP 地址和网关，该实例本地接口 ether3 的 IP 地址为 10.5.50.1/24

```
[admin@MikroTik] /ip hotspot> setup
Select interface to run HotSpot on

hotspot interface: ether3
Set HotSpot address for interface
```

```
local address of network: 10.5.50.1/24
masquerade network: yes
Set pool for HotSpot addresses

address pool of network: 10.5.50.2-10.5.50.254
Select hotspot SSL certificate

select certificate: none
Select SMTP server

ip address of smtp server: 0.0.0.0
Setup DNS configuration

dns servers: 10.1.101.1
DNS name of local hotspot server

dns name: myhotspot
Create local hotspot user

name of local hotspot user: admin
password for the user:
[admin@MikroTik] /ip hotspot>
```

以下是使用向导创建策略后，查看各个功能菜单下的相关配置

```
[admin@MikroTik] /ip hotspot> print
Flags: X - disabled, I - invalid, S - HTTPS
 #    NAME          INTERFACE        ADDRESS-POOL        PROFILE        IDLE-TIMEOUT
 0    hotspot1      ether3           hs-pool-3           hsprof1        5m
[admin@MikroTik] /ip hotspot>
[admin@MikroTik] /ip pool> print
 # NAME                                        RANGES
 0 hs-pool-3                                   10.5.50.2-10.5.50.254
[admin@MikroTik] /ip pool> /ip dhcp-server
[admin@MikroTik] /ip dhcp-server> print
Flags: X - disabled, I - invalid
 #    NAME       INTERFACE     RELAY           ADDRESS-POOL     LEASE-TIME ADD-ARP
 0    dhcp1      ether3                        hs-pool-3        1h
[admin@MikroTik] /ip dhcp-server> /ip firewall nat
[admin@MikroTik] /ip firewall nat> print
Flags: X - disabled, I - invalid, D - dynamic
 0 X ;;; place hotspot rules here
     chain=unused-hs-chain action=passthrough

 1   ;;; masquerade hotspot network
     chain=srcnat action=masquerade src-address=10.5.50.0/24
[admin@MikroTik] /ip firewall nat>
```

向导参数说明：

参数	属性
hotspot interface (*字符*; 默认: **allow**)	选择运行 HotSpot 的网络接口，通常 HotSpot 接口配置到内网，当 HotSpot 运行到 bridge 接口时，确保公网接口不包含在 bridge 端口组中，也不能将 HotSpot 服务配置到 bridge 组中的网络端口，只能配置到 bridge 接口
local address of network (*IP*; 默认: **10.5.50.1/24**)	HotSpot 本地接口网关地址（通常是 HotSpot 接口的接口 IP 地址）
masquerade network (*yes* | *no*; 默认: **yes**)	是否伪装 HotSpot 网络，当选择 yes 将在 */ip firewall nat* 创建 *action=masquerade* 规则
address pool of network (*字符*; 默认: **yes**)	分配 HotSpot 网络的地址池，用于将用户动态 IP 地址分配，当客户端不愿意更改网络地址设置，通过 HotSpot 的 UPNP 功能分配用户当前地址池，也能实现网络访问。
select certificate (*none* | *import-other-certificate*; 默认:)	当 HTTPS 验证模式要求开启时 ，需选择 SSL 证书
ip address of smtp server (*IP*; 默认: **0.0.0.0**)	SMTP 服务器 IP 地址，是否重定向 HotSpot 网络用户的 SMTP 请求(TCP 25 端口)
dns servers (*IP*; 默认: **0.0.0.0**)	用于分配给 HotSpot 客户的 DNS 服务器地址，在/ip dns 菜单下开启 dns 缓存功能
dns name (*字符*; 默认: **""**)	为 HotSpot 服务器添加域名，需要完整域名配置，例如：www.example.com
name of local hotspot user (*字符*; 默认: **"admin"**)	创建一个 HotSpot 登录账号，不设置默认为 admin，添加到 */ip hotspot user* 菜单下
password for the user' (*字符*; 默认:)	设置创建 HotSpot 账号的密码

20.3 HotSpot 接口设置

操作路径: ***/ip hotspot***

HotSpot 系统建立在一个独立的网络接口，你可以在不同的网络接口（以太网卡、无线网卡等）上配置不同的 HotSpot 服务器。

属性描述

addresses-per-mac (*整型* | unlimited; 默认: **2**) - 允许与特定 MAC 地址绑定的 IP 地址数量（降低一个 IP 模拟多个 MAC 的攻击）
  unlimited - 每个 MAC 对应 IP 地址数量无限制
address-pool (*名称* | none; 默认: **none**) - 运行一对一 NAT 的 IP 地址池。你可以选择不使用一对一 NAT
  none - 对这个 HotSpot 接口的客户不使用一对一 NAT
HTTPS (*只读: flag*) - HTTPS 服务是否在这个接口上实际在运行（它在这个服务器概要中设置，并且在路由器中输入了一个合法的认证）

idle-timeout (*时间* | none; 默认: **00:05:00**) - 对未认证客户的空闲超时时间（非活动的最大时间）。它用于探测客户没有使用外部网络（因特网），例如，没有收到来自某个客户的流量也没有流出路由器的流量。达到超时时间后，用户将被主机注销清除，用户所使用的地址也将被释放

none – 不切断空闲用户

interface (*名称*) - 运行 HotSpot 的接口

ip-of-dns-name (*只读: IP address*) - HotSpot 接口概要中设置的 HotSpot 网关 DNS 名称的 IP 地址

keepalive-timeout (*时间* | none;默认: **none**) - 对认证客户进行在线存活探测，用于探测客户是活动，且是否可达的。如果在这个期间探测失败，那么用户将被剔除在线列表，并且释放获取的 IP 地址。

none – 不启用超时探测

profile (*名称*; 默认: **default**) - 接口的默认 HotSpot 概要

reset-html (*名称*) - 以原始的 HTML 档重新覆盖已有的 HotSpot servlet。它用于你改变 servlet 之后且它不工作。

注：addresses-per-mac - 只有当地址池定义后，属性才能生效。

为了把 HotSpot 系统添加到本地接口，允许系统对每个客户进行一对一 NAT（来自 **HS-real** 地址池的地址将被用于 NAT）：

```
[admin@MikroTik] ip hotspot> add interface=local address-pool=HS-real
[admin@MikroTik] ip hotspot> print
Flags: X - disabled, I - invalid, S - HTTPS
 #    NAME                          INTERFACE      ADDRESS-POOL PROFILE IDLE-TIMEOUT
 0    hs-local                      local          HS-real      default 00:05:00
[admin@MikroTik] ip hotspot>
```

20.4 HotSpot profile 策略

操作路径: ***/ip hotspot profile***

属性描述

属性	描述
dns-name (*字符*; 默认: **""**)	Hotspot 服务器的域名名称 (用于登录页面访问的域名). 设置该域名后，将自动配置到 DNS cache 菜单下创建一条静态 DNS 条目
hotspot-address (*IP*; 默认: **0.0.0.0**)	设置 Hotspot 服务的 IP 地址
html-directory (*字符*; 默认: **hotspot**)	HTML 页面文件存储目录名称(系统默认目录为 hotspot)，可根据需要修改 HTML 文件存储目录，使用自定义的 HTML 文件，如修改 Hotspot 的 login 文件。 • **v6.31 和早期版本:**对于官方硬件设备有 flash 路径，hotspot html 文件目录存储在 flash 中，路径如下:"/(hotspot_dir)"。由于 hotspot 将"flash"目录视为根目录，所以路径是/hotspot。 • **v6.32 和后期版本: html 文件路径修改为 "/flash/(hotspot_dir)"，即所有文件路径在/flash 下**

html-directory-override (*字符*; 默认: **none**)	Hotspot html 文件备用路径。只有当定义 Hotspot html 文件保存到外部存储设备时会被使用，如存放在 mircoSD、U 盘等。当配置后，Hotspot 会切换到外部路径，如果外部路径无法使用，Hotspot 回切回到 **html-directory**
http-cookie-lifetime (*时间*; 默认: **3d**)	HTTP cookie 有效时间，设置该参数与 Hotspot 登录模式 cookie 关联
http-proxy (*IP:Port*; 默认: **0.0.0.0:0**)	Hotspot 服务代理服务器的地址和端口，当使用默认值时，所有请求都重定向到本地/ip proxy 下访问
https-redirect (*yes* | *no*; Default: **yes**)	当然访问 https://url 是否重定向未认证用户到 Hotspot 登录页面。当 Hotspot 登录方式选择 HTTPs 时，导入的证书与域名不匹配，通常会导致错误，这样情况可以选择将此参数设置为“no”，所有 https 请求将被拒绝，用户只能 http 页面。
login-by (*cookie|http-chap|http-pap|https|mac|trial|mac-cookie*; 默认: **http-chap, cookie**)	设置 Hotspot 认证模式 • **mac-cookie** – 启用通过 mac cookie 登录方式 • **cookie** – 仅能与 HTTP 验证方式使用，当用户首次在 HotSpot 中进行身份验证时，将生成 HTTP cookie。在 cookie-lifetime 到期之前，不会询问用户的账号/密码并自动进行身份验证 • **http-chap** - 用户需要账号/密码才能在 HotSpot 中进行身份验证。采用基于 MD5 哈希算法对验证账号密码进行保护。 • **http-pap** -用户需要登录/密码才能在 HotSpot 中进行身份验证。但采用的是明文提交账号密码。 • **https** - 用户需要登录/密码才能在 HotSpot 中进行身份验证。用户和服务器之间采用 SSL 加密隧道交换认证信息，具体参考 HTTPs 认证事例 • **mac** – MAC 地址验证，用户端无需访问登录页面，即可进行身份验证。将客户端 mac 地址添加到 Hotspot 用户数据库中，客户端连接到 Hotspot 后立即进行身份验证。 • **trial** – 试用模式，允许用户在指定的时间内访问到互联网，无需进行 Hotspot 认证登录
mac-auth-password (*字符*; 默认:)	与 MAC 认证模式一起使用的字段，用于指定要通过其 MAC 地址认证账号的密码。当特定 RADIUS 服务器拒绝密码为空的用户进行认证时，需要使用该选项
name (*字符*; 默认:)	Profile 策略名称
nas-port-type (*字符*; 默认: **wireless-802.11**)	NAS-Port-Type 值用于发送到 RADIUS 服务器，该值描述 nas 端口类型，此值可选说明 Hotspot 服务器的物理端口的类型
radius-accounting (*yes* | *no*; 默认: **yes**)	选择 yes 时，将每个用户记账信息发送到 RADIUS 服务器

radius-default-domain (*字符*; 默认:)	用于 RADIUS 请求的默认域。允许通过配置*/ip hotspot profile* 多条策略分隔多个 RADIUS 域，相同域的 RADIUS 服务器需要在/radius 菜单下配置
radius-interim-update (*time* | *received*; 默认: **received**)	记账更新周期
radius-location-name (*字符*; 默认:)	当 RADIUS 服务被启用，该参数可以选择，作用是 Hotspot 服务器通过 RADIUS-Location-Id 向 RADIUS 服务器发送身份信息。
radius-mac-format (*"XX XX XX XX XX XX"|XX:XX:XX:XX:XX:XX|XXXXXX-XXXXXX|XXXXXXXXXXXX|XX-XX-XX-XX-XX-XX|XXXX:XXXX:XXXX|XXXXXX:XXXXXX*; 默认: **XX:XX:XX:XX:XX:XX**)	选择提交 RADIUS 服务器 MAC 地址格式
rate-limit (*字符*; 默认: **""**)	Hotspot 服务器带宽限制,具体格式 **rx-rate[/tx-rate] [rx-burst-rate[/tx-burst-rate] [rx-burst-threshold[/tx-burst-threshold] [rx-burst-time[/tx-burst-time]]] [priority] [rx-rate-min[/tx-rate-min]]**
smtp-server (*IP*; 默认: **0.0.0.0**)	重定向 Hotspot 用户的 SMTP 请求到指定的 SMTP 服务器
split-user-domain (*yes* | *no*; 默认: **no**)	当用户名以“user@domain”或 “domain\user”格式从 RADIUS 服务器出现时，是否将用户名与域名分离
ssl-certificate (*string* | *none*; 默认: **none**)	启用 HTTPs 认证后，选择 SSL 证书名称
trial-uptime (*time/time*; 默认: **30m/1d**)	trial 模式启用后可以设置，该参数有 2 个时间量，第一个是在试用模式下，由 MAC 地址标识的试用用户可以访问互联网的时长，第二个时间是间隔多长时间，用户能再次试用
trial-user-profile (*字符*; 默认: **default**)	试用模式下指定 Hotspot user profile 策略规则
use-radius (*yes* | *no*; 默认: **no**)	启用 RADIUS 协议验证 Hotspot 用户登录

注：可以在相同的 MAC 地址上有多重的 cookie。例如，在同一台计算机上对每个 web 浏览器都可以有一个单独的 cookie。

Cookie 有超时时间，默认的 cookie 合法时间为 3 天（72 小时），但 HotSpot 服务是可以修改的，例如：

```
/ip hotspot profile set default http-cookie-lifetime=1d
```

对于 Cookie 的使用或设置时间长短根据实际情况操作。

当路由器设置有多个本地 IP 或三层接口，Hotspot 的网关可以指定其中一个，通过 **hotspot-address** 参数配置，例如

```
[admin@MikroTik] /ip address> print
```

```
Flags: X - disabled, I - invalid, D - dynamic
 #    ADDRESS             NETWORK          INTERFACE
 0    192.168.88.1/24      10.31.3.0       vlan88
 1    192.168.70.1/24    192.168.70.0      wlan1
 2    111.111.88.2/24    111.111.88.0      ether1
```

我们在 wlan1 启用 hotspot 热点认证，但我们希望用户访问认证页面的 ip 地址是 111.111.88.2，我们可以配置 hotspot-address

```
[admin@MikroTik] /ip hotspot profile> set 0 hotspot-address=111.111.88.2
[admin@MikroTik] /ip hotspot profile> print
Flags: * - default
 0 * name="default" hotspot-address=111.111.88.2 dns-name=""
     html-directory=hotspot rate-limit="" http-proxy=0.0.0.0:0
     smtp-server=0.0.0.0 login-by=http-chap split-user-domain=no
     use-radius=no
```

这样设置后，所有认证用户都会重定向到 111.111.88.2 访问，该 IP 是路由器本地 IP，所以路由是可达。

20.5 Walled Garden 内院

操作路径: ***/ip hotspot walled-garden***

Walled garden 是在允许未认证下访问某些资源，同样能用于需要认证访问的其他资源。例如：访问一些 HotSpot 服务提供商的基本信息或账单选项。

这个目录只管理对 HTTP 和 HTTPS 协议的 Walled Garden。其他协议也可以包含进 Walled Garden，但要在其他地方配置（**/ip hotspot walled-garden ip**，参考本手册的下一部分）。

属性描述

action (allow | deny; 默认: **allow**) - 如果数据报和规则匹配则执行动作：
allow - 无需优先认证就允许访问页面
deny - 需要认证才能访问页面
dst-address (*IP 地址*) -目的 web 服务器的 IP 地址
dst-host (*wildcard*; 默认: **""**) - 目的 web 服务器的域名（这是一个通配符）
dst-port (*整型*; 默认: **""**) -客户发送请求的目的 TCP 端口
method (*文本*) – 请求的 HTTP 方法
path (*文本*; 默认: **""**) -请求的路径 (这是一个通配符)
server (*名称*) - 应用该规则的 HotSpot 服务器名
src-address (*IP 地址*) - 发送请求的用户 IP 地址

注： dst-host 和 **dst-path** 支持通配符匹配，如果仅输入"example"，不会匹配"example.com"，可用通配符'*' (匹配任意字符，如"example*"， '?' 匹配任何一个字符)。也能使用正则表达式，如果使用正则表达式，那么它应该以冒号(':')开始。

关于使用正则表达式：：

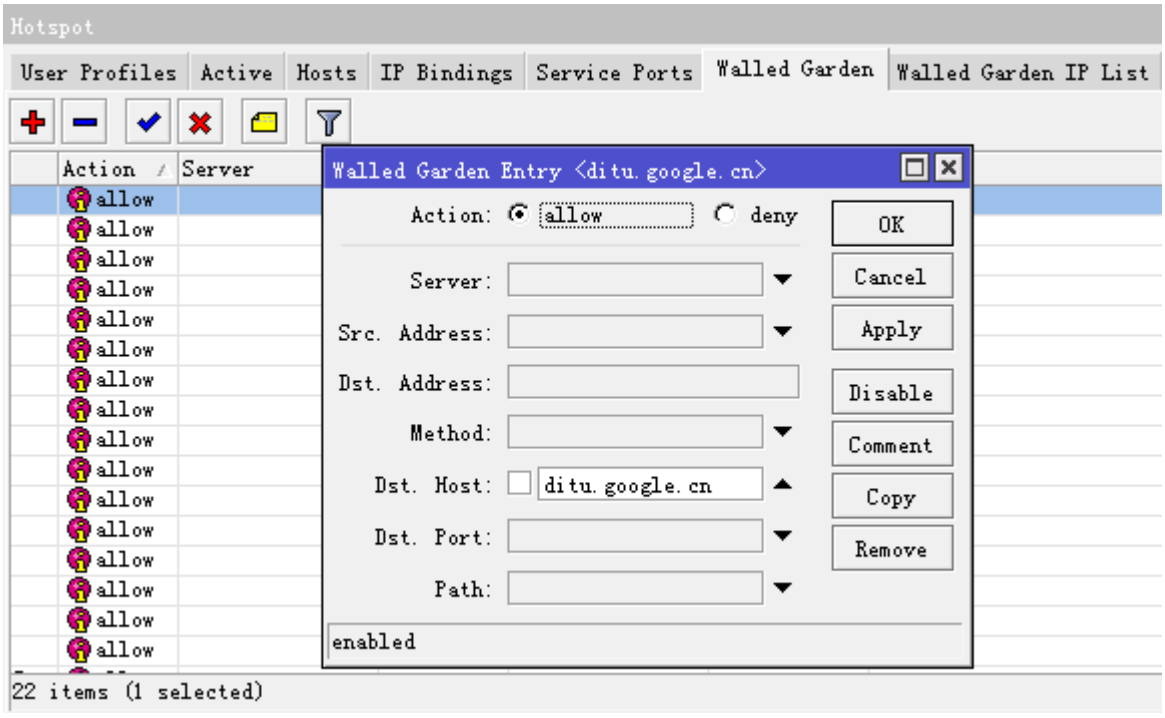
- **** 符号序列是用于在控制面板输入****字符的
- **\.** 样式的意思为只是 **.** (在正则表达式单独的点表示任何符号)
- 显示在给出样式之前任何符号都不允许，我们在样式开始使用**^**符号
- 指定在给出样式之后任何符号都不允许，我们在样式结束的地方使用符号**$**

由于路由器不能解密请求，你也就不能对 HTTPS 请求使用 **path** 属性（也不应该使用——这就是 HTTPS 协议被创造的目的）。

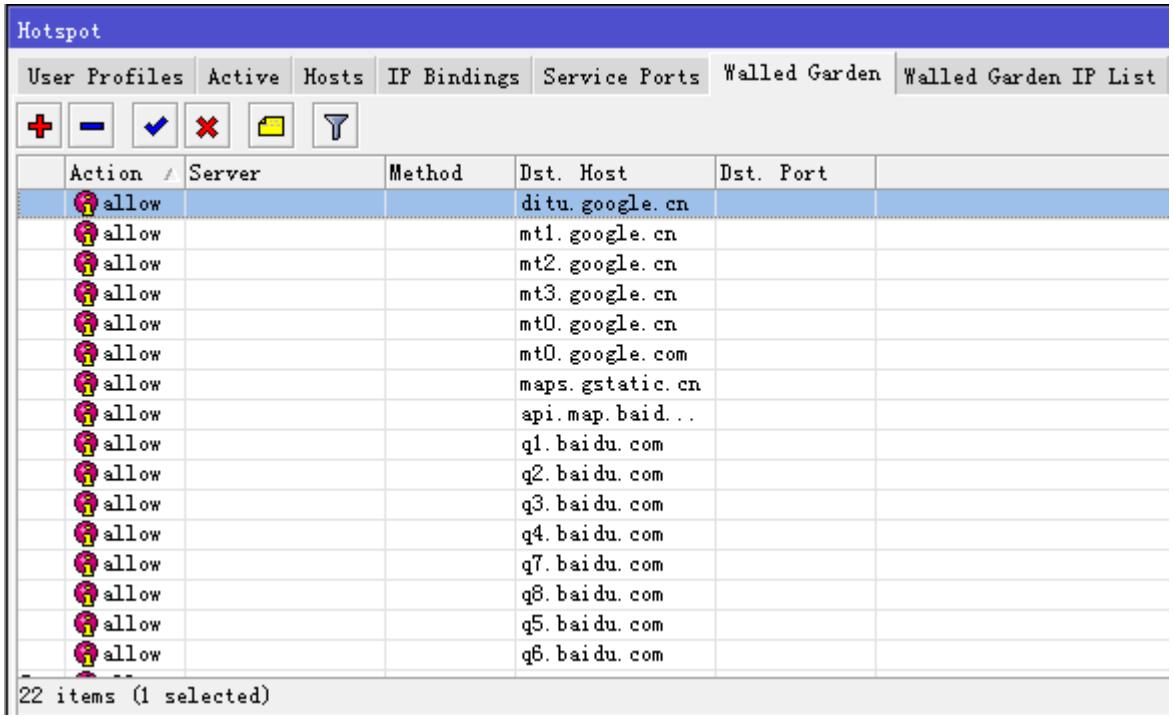
允许未认证用户到 **www.example.com** 域**/paynow.html** 页面的请求：

```
[admin@MikroTik] ip hotspot walled-garden> add path="/paynow.html" \
\... dst-host="www.example.com"
[admin@MikroTik] ip hotspot walled-garden> print
Flags: X - disabled, D - dynamic
 0    dst-host="www.example.com" path="/paynow.html" action=allow
[admin@MikroTik] ip hotspot walled-garden>
```

例如：我们让用户在没有认证的情况下免费访问 baidu 和 google 地图，我们将 baidu 和 google 的地图域名分析后添加到 Walled Garden



Baidu 和 google 需要访问的域名清单



IP 方式 Walled Garden

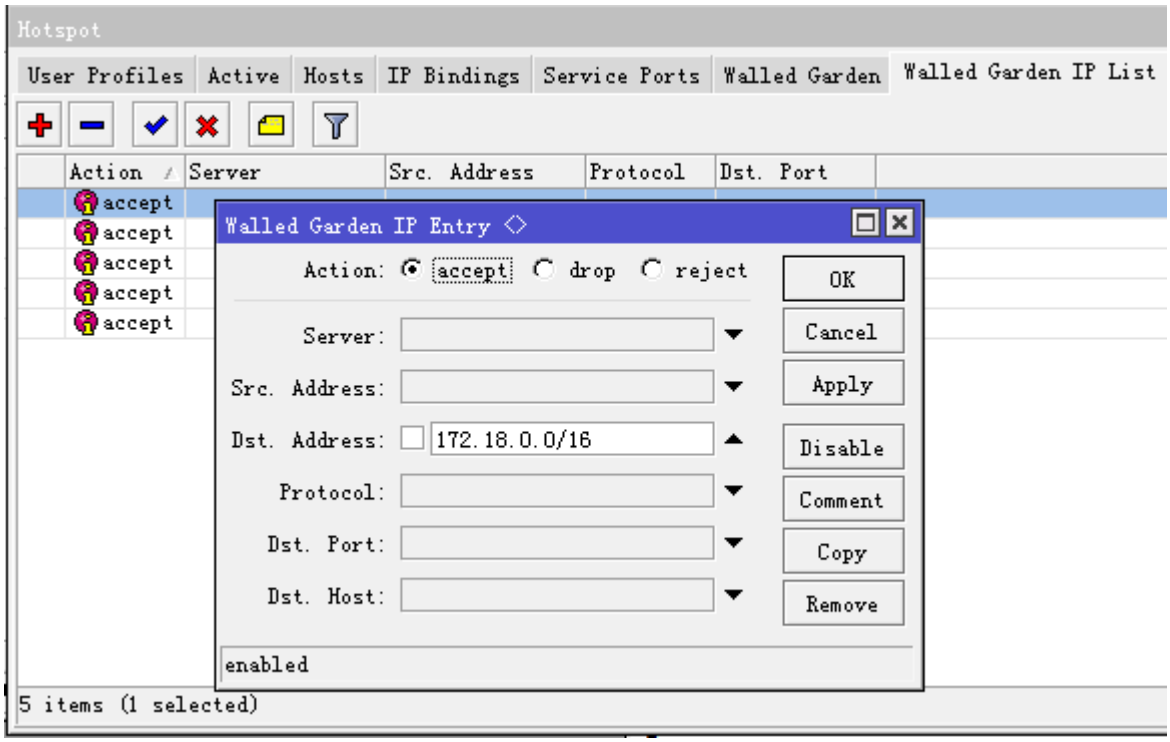
操作路径: ***/ip hotspot walled-garden ip***

IP 方式的 Walled Garden 与之前的 Walled Garden 相同，只是通过防火墙规则放行允许通过的 IP 地址段。

属性描述

action (allow | deny; default: **allow**) -如果数据报和规则匹配则执行动作：
allow - 无需认证就允许访问页面
deny - 需要认证才能访问页面
reject -需要认证才能访问该页面，以防页面会被没有认证的 ICMP 拒绝信息访问，主机不可达将被产生
dst-address (*IP 地址*) -目的 web 服务器的 IP 地址
dst-host (*wildcard*; 默认: **""**) - 目的 web 服务器的域名（这是一个通配符）
dst-port (*整型*; default: **""**) -客户发送请求的目的 TCP 端口
protocol (*整型* | ddp egp encap ggp gre hmp icmp idpr-cmtp igmp ipencap ipip ipsec-ah ipsec-esp iso-tp4 ospf pup rdp rspf st tcp udp vmtp xns-idp xtp) - IP 协议名
server (*名称*) - 应用该规则的 HotSpot 服务器名
src-address (*IP 地址*) - 发送请求的用户 IP 地址

我们允许一段 IP 地址 172.18.0.0/16 在没有认证的情况下能被用户访问，action=accept



基于IP的Walled Garden与Walled Garden不同在于Walled Garden用的是proxy方式，而Walled Garden IP方式通过 ip firewall filter 规则允许通过地址，我们可以在 ip firewall filter chain=hs-unauth 和 chain=hs-unauth-to 找到相应的规则，通过 action=return 让其返回不在执行后面的操作

在 hs-unauth 允许这些地址通过

```
 [admin@MikroTik] /ip firewall filter> print chain=hs-unauth
Flags: X - disabled, I - invalid, D - dynamic
 0 D chain=hs-unauth action=return dst-address=172.18.0.0/16

 1 D chain=hs-unauth action=return dst-address=10.0.0.0/8

 2 D chain=hs-unauth action=return dst-address=172.16.0.0/16

 3 D chain=hs-unauth action=return dst-address=172.17.0.0/16

 4 D chain=hs-unauth action=return dst-address=220.181.26.152

 5 D chain=hs-unauth action=reject reject-with=tcp-reset protocol=tcp

 6 D chain=hs-unauth action=reject reject-with=icmp-net-prohibited
```

在 hs-unauth-to 允许这些地址进行 ping

```
[admin@MikroTik] /ip firewall filter> print chain=hs-unauth-to
Flags: X - disabled, I - invalid, D - dynamic
 0 D chain=hs-unauth-to action=return src-address=172.18.0.0/16

 1 D chain=hs-unauth-to action=return src-address=10.0.0.0/8
```

```
 2 D chain=hs-unauth-to action=return src-address=172.16.0.0/16

 3 D chain=hs-unauth-to action=return src-address=172.17.0.0/16

 4 D chain=hs-unauth-to action=return src-address=220.181.26.152

 5 D chain=hs-unauth-to action=reject reject-with=icmp-host-prohibited
[admin@MikroTik] /ip firewall filter>
```

20.6 Hotspot binding

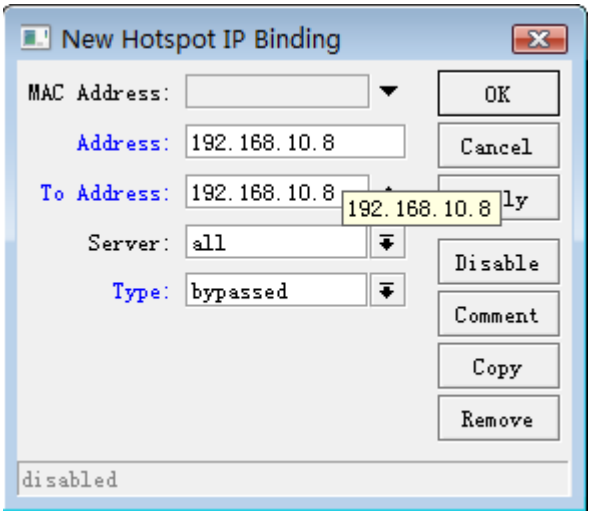
操作路径：***/ip hotspot ip-binding***

ip-binding 允许指定主机网络的静态 nat 规则，通过 binding 控制是否要求客户主机进行认证、绕过认证、阻止认证等。可以允许指定主机 IP 地址绕过 HotSpot 认证（如：它们可以不必认证登陆就能访问外部资源），并阻止指定的主机 IP 地址认证登陆。

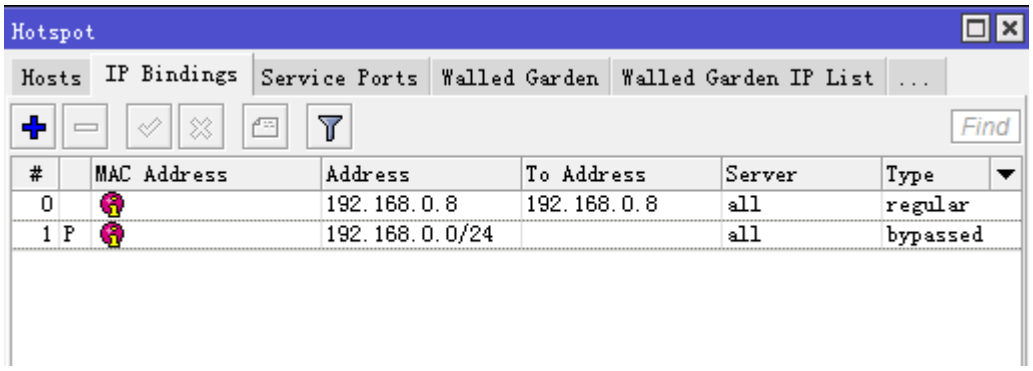
属性	描述
address (*IP 范围*; 默认: **""**)	客户主机的原始 IP 地址
mac-address (*MAC*; 默认: **""**)	客户主机的 MAC 地址
server (*字符* | *all*; 默认: **"all"**)	HotSpot 服务器的名称. • **all** – 将应于所有 Hotspot 服务器
to-address (*IP*; 默认: **""**)	客户主机新 IP 地址，用于在路由器做 nat 转换(客户端无法感知到这样的 nat 转换过程)
type (*blocked* | *bypassed* | *regular*; 默认: **""**)	IP-binding 类型 • **regular** – 根据规则执行一对一的 nat 转换 • **bypassed** – 执行 nat 转换，但不需要客户端登录 HotSpot 认证，直接访问网络。 • **blocked** – 不执行 nat 转换，并丢弃客户端访问数据

注：在 binding 规则列表中是一个有序列表，即按照 FIFO 队列执行，所以可以把更精确详细的主机规则放在该表的顶部，优先于其他规则执行。

下面是一台主机 192.168.10.8 在无需认证情况下上网，即使用 bypassed，绕过认证：



Binding 中我们设置做一个规则策略，类似于 firewall filter 中的 FIFO 算法，即指定某一主机要求认证，其他主机都无需认证，这样的策略配置可以在 hotspot 中灵活应用，如下面的策略规则 1 是允许 192.168.0.0/24 的网段可以绕过认证，直接上网，而 192.168.0.8 主机要求正常认证



20.7 Hotspot host 列表

操作路径: ***/ip hotspot host***

这个目录显示了所有连接到 HotSpot 服务下的活动主机，属性为只读，这个清单包含所有一对一 NAT 翻译。

属性描述

address (*只读: IP address*) - 客户的原始 IP 地址
authorized (*只读: flag*) - 客户是否成功地被 HotSpot 系统认证
blocked (*只读: flag*) - 如果访问在 walled-garden 中因为广告超时时间过期被阻止，则为真
bridge-port (*只读: 名称*) - 主机连接的真实物理接口。当 HotSpot 服务被放在一个桥接口以判定在桥中的主机实际的端口时，使用该值
bypass-hotspot (*只读: flag*) - 是否客户不需要 HotSpot 系统的认证
bytes-in (*只读: 整型*) - 路由器从客户接收的字节数
bytes-out (*只读: 整型*) - 路由器发送到客户的字节数
host-dead-time (*只读: 时间*) - 路由器没有从主机接收任何数据报（包括 ARP 响应，持活响应及用户流量）的时间。
idle-time (*只读: 时间*) – 闲置的时间

idle-timeout (*只读: 时间*) - 应用于用户的确切 **idle-timeout** 值。这个属性显示了用户空闲多久会被自动注销。
keepalive-timeout (*只读: 时间*) - 应用于用户的 **keepalive-timeout** 精确值。这个属性显示了用户的计算机在不可达状态多久会被自动注销
mac-address (*只读: MAC 地址*) - 实际的用户 MAC 地址
packets-in (*只读: 整型*) - 路由器接收客户的包数
packets-out (*只读: 整型*) - 路由器发送到客户的数据报数
server (*只读: 名称*) - 主机连接到的服务器名
static (*只读: flag*) - 翻译是否是来自静态 IP 绑定列表
to-address (*只读: IP 地址*) - 主机翻译成的原始 IP 地址
uptime (*只读: 时间*) - 用户的当前会话时间（如：用户在活动用户列表中已经多久了？）

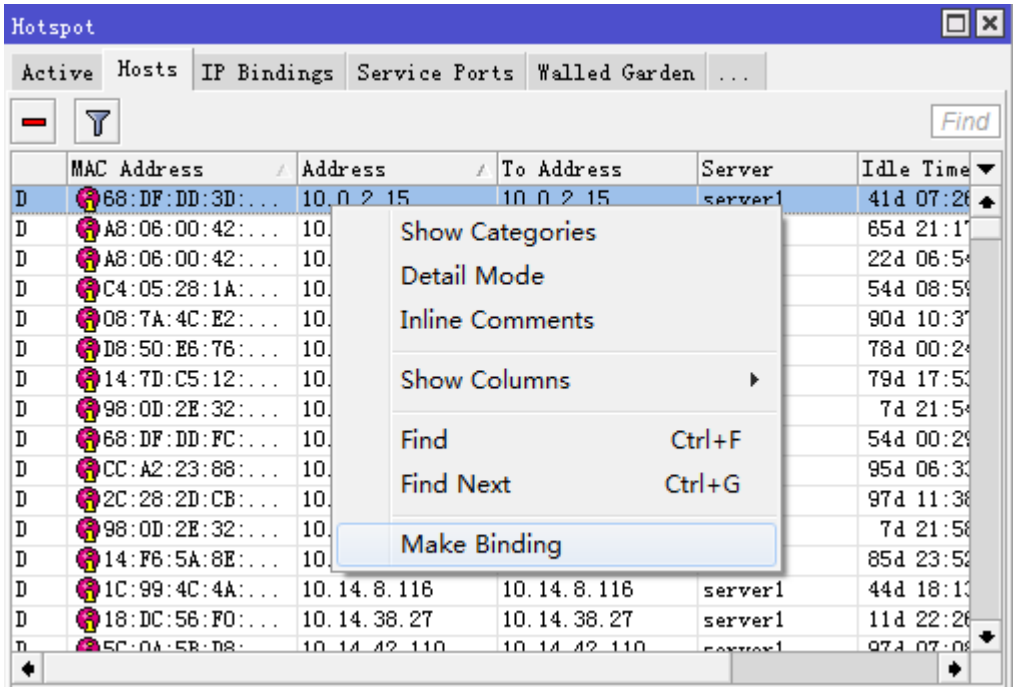
命令描述

make-binding - 把可以个动态项目从这个列表复制到静态 IP 绑定列表
unnamed (*名称*) – 项目编号
comment (*文本*) - 对规则的文本注释
type (regular | bypassed | blocked) - 静态项目的类型

该列表中，显示了在二层网络下，Hotspot 接口下学习到的所有 MAC 主机和 IP 等信息，在/ip hotspot host 下查看主机信息，前缀代表了主机的网络状态，H 代表 DHCP 获取，A 代表已经认证通过

```
[admin@MikroTik] /ip hotspot host> print
Flags: S - static, H - DHCP, D - dynamic, A - authorized, P - bypassed
 #       MAC-ADDRESS         ADDRESS          TO-ADDRESS       SERVER      IDLE-TIMEOUT
 0 D     1C:99:4C:4A:A9:29 10.36.132.140    10.36.132.140    server1
 1 D     78:F5:FD:8E:7E:D4 10.43.53.245     10.43.53.245     server1
 2 D     68:DF:DD:3D:FA:99 10.175.36.172    10.175.36.172    server1
 3 HD    68:DF:DD:3D:FA:99 10.162.170.200   10.162.170.200   server1
 4 AHD   14:F6:5A:AC:65:8F 10.162.39.75     10.162.39.75     server1
```

在 Winbox 中可以选择指定主机点击右键，选择 make-binding 设置指定主机的在该 hotspot 下的认证类型



20.8 HotSpot Users 管理

主要对 Hotspot 的用户账号、权限和用户参数分组进行策略管理，User 管理目录下有两个菜单，分别是 user 和 user profile，user 用于用户账号的建立，如账号的名称、密码、MAC 和 IP 绑定，以及策略绑定和相关流量和时间限制。User profile 则定义一组策略，便于对该组用户的策略定义，包括地址池、超时时间、带宽限制、匹配防火墙和 proxy 规则等。

Hotspot 热点用户管理用于普通用户分类设置，profile 用户组根据需要能将不同用户分类管理，下面是相关 profile 的属性介绍。

操作路径：***/ip hotspot user profile***

属性描述

属性	描述
add-mac-cookie (*yes*|*no*; 默认: **yes**)	为用户添加 mac cookie
address-list (*字符*; 默认:)	用于定义该策略下的用户 IP 地址被创建到指定的地址列表中，通常用于防火墙策略或标记用户地址应用到 Queue 流控等
address-pool (*字符*|*none*; 默认: **none**)	用户获取 IP 地址的地址池。当用户在电脑上有不正确的网络配置时，HotSpot 进行转换并从地址池中分配正确的 IP 地址，就是即插即用功能
advertise (*yes* | *no*; 默认: **no**)	启用强制广告。定义每间隔一段时间，Hotspot 用户就会看到指定的网页。广告页面可能被浏览器弹出窗口拦截器拦截。
advertise-interval (*time[,time[,..]]*; 默认: **30m,10m**)	设置两个广告间弹出的时间周期，在配置参数最后一个值用于所有后续广告，即 10 分钟

advertise-timeout (*time* | *immediately* | *never*; 默认: **1m**)	劫持 HotSpot 客户端访问网络，显示广告的时间长度。
advertise-url (*string[,string[,..]]*; 默认:)	URL 列表，填写弹窗广告的 URL 链接，允许多个 URL 设置.
idle-timeout (*时间* | *none*; 默认: **none**)	Hotspot 认证用户不活跃时间长度。通过时间计数器，记录客户没有流量产生时长。例如：当客户电脑关机，未正常注销操作，idle-timeout 计数时间到达后，Hotspot 自动将该客户账号剔除出在线列表
incoming-filter (*字符*; 默认:)	用于客户入方向数据包的防火墙链表的策略
incoming-packet-mark (*字符*; 默认:)	标记该客户入方向的数据包策略
keepalive-timeout (*时间*| *none*; 默认:)	Hotsport 认证客户保持超时时间，用于探测客户是否在线，探测客户未在线时间超时，将剔除客户。Keepalive 是主动探测，idle-timeout 是被动记录超时时间。
mac-cookie-timeout (*时间*; 默认: **3d**)	根据最后一次登录和注销选择 mac-cookie timeout 时间
name (*字符*; 默认:)	定义用户策略名称
on-login (*字符*; 默认: **""**)	脚本执行名称，当客户成功登录 Hotspot 时，可以获取多个变量值， 包括以下变量： • $user • $username (替换$user 为内部 var 名称) • $address • $"mac-address" • $interface 如果 hotspot 设置到 bridge 接口上，获取接口名称需要在 bridge settings 菜单下开启 **use-ip-firewall**
on-logout (*字符*; 默认: **""**)	脚本执行名称，当客户从 Hotspot 注销时，可以获取多个变量值， 包括以下变量： • $user • $username (alternative var name for $user) • $address • $"mac-address" • $interface • $cause 从 v6.34rc11 版本开始，新增以下变量： • $uptime-secs - final session time in seconds • $bytes-in - bytes uploaded • $bytes-out - bytes downloaded

	• $bytes-total - bytes up + bytes down • $packets-in - packets uploaded • $packets-out - packets downloaded • $packets-total - packets up + packets down 如果 hotspot 设置到 bridge 接口上，获取接口名称需要在 bridge settings 菜单下开启 **use-ip-firewall**
open-status-page (*always* | *http-login*; 默认: **always**)	认证用户显示状态页面选项，用于显示客户认证状态，也可以修改 alogin.html 文件显示广告内容 • **http-login** – 基于 HTTP 认证（包括 cookie 和 HTTPS 方式）打开状态页面 • **always** – 基于 mac 登录认证方式打开 HTTP 状态页面
outgoing-filter (*字符*; 默认:)	应用于该客户出方向的策略防火墙链表名称
outgoing-packet-mark (*string*; Default:)	标记该客户出方向的数据包策略
rate-limit (*string*; Default: **""**)	创建动态队列，即配置客户的 IP 流控策略，参数如下：**[rx-rate[/tx-rate] [rx-burst-rate[/tx-burst-rate] [rx-burst-threshold[/tx-burst-threshold] [rx-burst-time[/tx-burst-time] [priority] [rx-rate-min[/tx-rate-min]]]],**. 例如：1M 下行，512k 上行，格式为：rate-limit=512k/1M
session-timeout (*time*; 默认: **0s**)	设置 session-timeout 时间，当时间用完后，用户会被注销，0 代表不设置
shared-users (*整型*; 默认: **1**)	相同用户账号同时登录数量
status-autorefresh (*time* | *none*; 默认: **none**)	HotSpot 状态页面自动刷新间隔周期
transparent-proxy (*yes* |; 默认: **yes**)	对该用户策略规则启用 HTTP 透明代理

注：当 idle-timeout 或者 session-timeout 到时，对该用户的连接会话将会被从 Hotspot 认证中注销，减少用户闲置对系统的超载。

新建一个 profile 策略，取名 yus，设置空闲超时为 1 小时，带宽上线 1M，下行 2M，账号默认共享 1 人使用。

```
[admin@MikroTik] /ip hotspot user profile> add name=yus idle-timeout=1h rate-limit=1m/2m
[admin@MikroTik] /ip hotspot user profile> print
Flags: * - default
 0 * name="default" status-autorefresh=1m shared-users=1 add-mac-cookie=no address-list=""
     transparent-proxy=no

 1   name="yus" idle-timeout=1h keepalive-timeout=2m status-autorefresh=1m shared-users=1
     add-mac-cookie=no rate-limit="1m/2m" address-list="" transparent-proxy=no
```

在 RouterOS v6 版本后，Queue 的调整，使得 simple queue 获得了与 queue tree 同样的属性，可以实现 HTB 的令牌桶流控，在 PPP profile 和 hotspot profile 中都增加了 queue 的菜单设置，这样的设置不在像以前每个用户带宽规则被 hotspot 僵硬的添加到 simple queue 中。

如下面的事例，我们将 yus 策略组的用户都设置到 simple queue 中，并指定父级为 PCQ，queue 类型为 default，这样可以让 hotspot 的用户带宽策略可以动态的添加到相应的 HTB 流控中，具体参见 Queue 章节。

```
[admin@MikroTik] /ip hotspot user profile> set name=yus parent-queue=PCQ queue-type=default
[admin@MikroTik] /ip hotspot user profile> print
Flags: * - default
 0 * name="default" status-autorefresh=1m shared-users=1 add-mac-cookie=no parent-queue=PCQ
     queue-type=default address-list="" transparent-proxy=no

 1   name="yus" idle-timeout=1h keepalive-timeout=2m status-autorefresh=1m shared-users=1
     add-mac-cookie=no rate-limit="1m/2m" address-list="" transparent-proxy=no
```

用户账号建立都是 user 目录下，这里主要设置账号的名称、密码、MAC 和 IP 绑定，以及策略绑定和相关流量和时间限制。一般 hotspot 账号创建首先是建立 user profile，然后再建立 user 下的用户帐户信息。

操作路径：**/ip hotspot user**

属性描述

address (*IP 地址*; 默认: **0.0.0.0**) - 静态 IP 地址。如果不是 0.0.0.0，那么客户将总是得到相同的 IP 地址。也就是说，对该用户只允许一个同时的登陆。任何一个已存在的地址都将使用嵌入的一对一 NAT 被这个地址取代。
bytes-in (*只读: 整型*) - 接收用户的总字节数
bytes-out (*只读: 整型*) - 发送给用户的总字节数
limit-bytes-in (*整型*; 默认: **0**) - 用户可以传输的最大字节数（例如：从接收到的字节数）
0 – 无限制
limit-bytes-out (*整型*; 默认: **0**) - 用户可以接收的最大自己数（例如：发送给用户的字节数）
0 – 无限制
limit-uptime (*时间*; 默认: **0s**) - 用户的总正常运行时间限制
0s – 无限制
mac-address (*MAC 地址*; 默认: **00:00:00:00:00:00**) - 静态 MAC 地址。如果不是 **00:00:00:00:00:00**，那么用户仅能从该 MAC 地址登陆
name (*名称*) - 用户名
packets-in (*只读: 整型*) - 接收到用户的最大包数量
packets-out (*只读: 整型*) - 发送给用户的最大包数
password (*文本*) – 用户口令
profile (*名称*; 默认值: **defualt**) – 用户数据
routes (*文本*) - 当用户连接上后将在热点网关注册的路由器。路由格式为“目标地址 网关 距离”（例如：“10.1.0.0/24 10.0.0.1 1”）。数个路由应用逗号分开指定。
server (*名称* | all; 默认: **all**) - 该用户允许登陆的服务器
uptime (*只读: time*) - 用户登陆的总时间

注：如果 MAC 认证方法使用，客户的 MAC 地址可以被当作用户名使用（不需要口令）

limit-bytes-in/out 字节限制是对每个用户的流量限制，即类似移动通信中的流量套餐，例如：如果对一个用户的下载限制为 100MB，并且用户已经下载了 30MB，那么在**/ip hotspot active** 中的登陆后会话下载限制将为 100MB - 30MB = 70MB。如果一个用户达到了他的限制（bytes-in >= limit-bytes-in 或 bytes-out >= limit-bytes-out），他将无法登陆。

添加一个仅允许以 **01:23:45:67:89:AB** MAC 地址登陆的用户名和密码都为 **ex** 的用户，并限制 1 小时工作时间，并设置 profile 为 yus：

```
[admin@MikroTik] ip hotspot user> add name=ex password=ex \
\... mac-address=01:23:45:67:89:AB limit-uptime=1h profile=yus
[admin@MikroTik] ip hotspot user> print
Flags: X - disabled
 #    SERVER      NAME                          ADDRESS         PROFILE    UPTIME
 0                ex                                                   default   00:00:00
[admin@MikroTik] ip hotspot user> print detail
Flags: X - disabled
  0     name="ex" password="ex" mac-address=01:23:45:67:89:AB profile=yus
      limit-uptime=01:00:00 uptime=00:00:00 bytes-in=0 bytes-out=0
      packets-in=0 packets-out=0
[admin@MikroTik] ip hotspot user>
```

20.9 Hotspot active 在线管理

操作路径： ***/ip hotspot active***

Active 列表及在线管理，显示用户当前已登陆了的信息。这里不能修改任何信息，除了使用 **remove** 命令用于剔除在线用户。

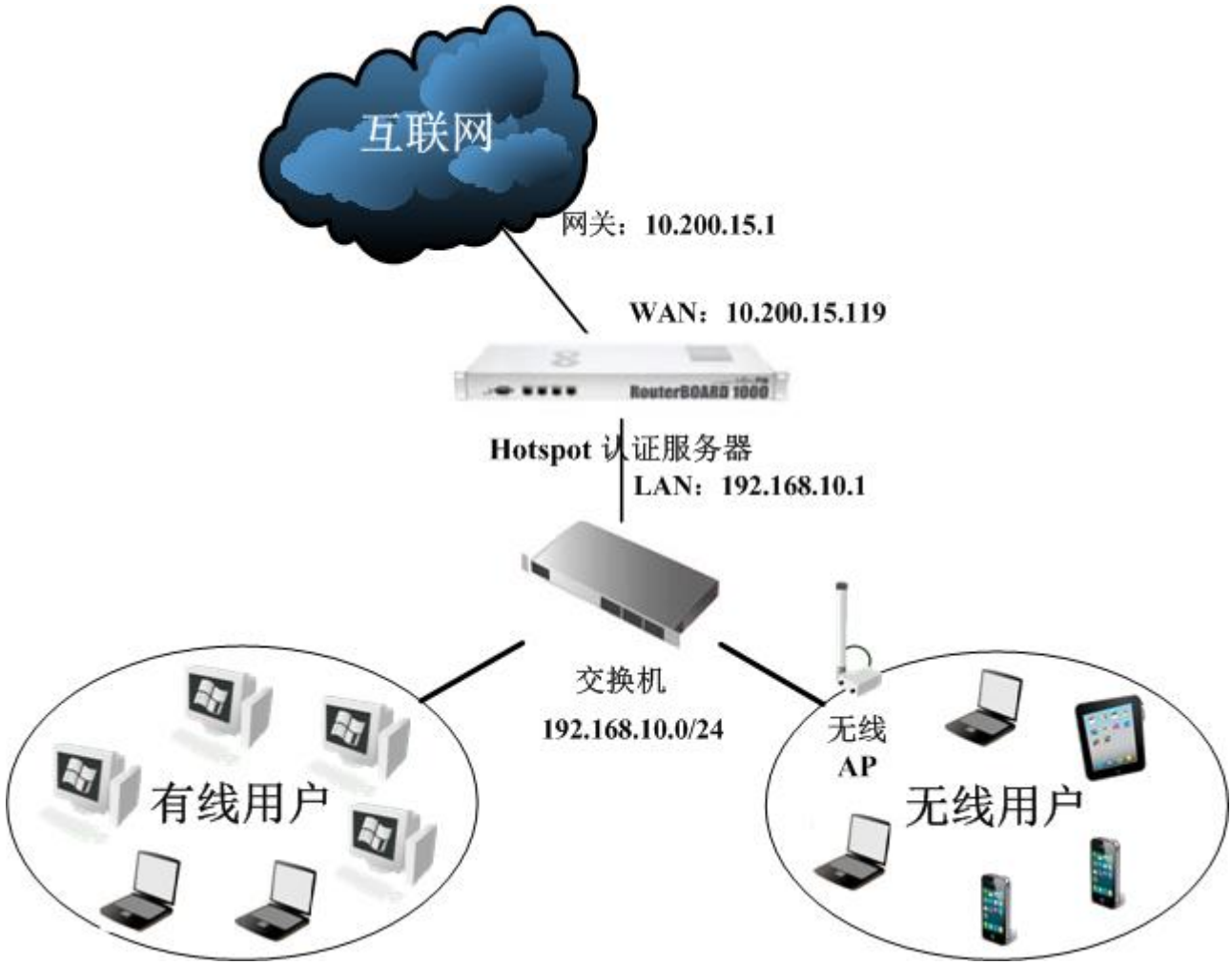
属性描述

address (*只读: IP 地址*) - 用户的 IP 地址
blocked (*只读: flag*) - 是否以广告将用户阻挡（例如：通常适当的广告未决）。
bytes-in (*只读: 整型*) - 路由器从客户收到的字节数
bytes-out (*只读: 整型*) - 由器发送到客户的字节数
domain (*只读: 文本*) - 用户范围（如果从用户名中分离出来）
idle-time (*只读: 时间*) - 用户被闲置的时间
idle-timeout (*只读: 时间*) -应用于该用户的 **idle-timeout** 精确值。这个属性显示他被自动注销的空闲时间
keepalive-timeout (*只读: 时间*) - 应用于该用户的 **keepalive-timeout** 精确值。该属性描述了用户的计算机不可达多久才会被自动注销
limit-bytes-in (*只读: 整型*) - 用户被允许发送给路由器的最大字节数
limit-bytes-out (*只读: 整型*) - 路由器被允许发送到客户的最大字节数
login-by (*多选项, 只读:* cookie | http-chap | http-pap | https | mac | trial) - 用户用的认证方法
mac-address (*只读: MAC 地址*) – 用户的实际 MAC 地址
packets-in (*只读: 整型*) - 路由器接受来自客户的包数量
packets-out (*只读: 整型*) - 路由器发送给客户的包数量
RADIUS (*只读:* yes | no) - 用户是否通过 RADIUS 认证
server (*只读: 名称*) - 用户登陆所指定的服务器

session-time-left (*只读: time*) - 应用于该用户的 **session-time-left** 精确值。这个属性显示了用户在自动被注销前保持的登入状态时间
uptime (*只读: time*) -当前用户的会话时间（例如：用户登入的时间)
user (*只读: 名称*) - 用户名

20.10 Hotspot 配置事例

通过以上的介绍后，根据下面的网络拓扑结构为例做一个 hotspot 的网络事例介绍：

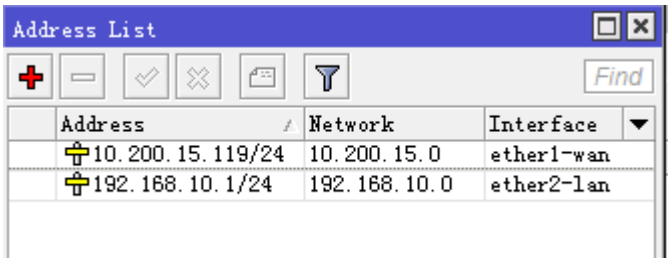


一个网关路由器的网络参数如下：

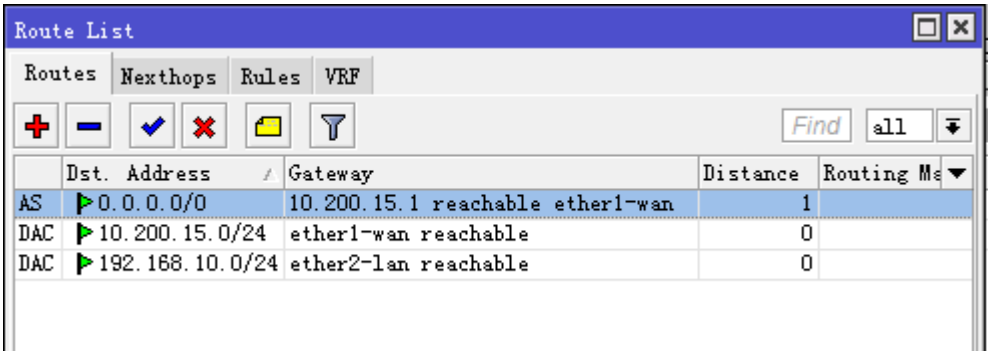
WAN 口对应外网 IP 为 10.200.15.119/24，网关为 10.200.15.1
LAN 口对应内网 IP 为 192.168.10.1/24
DNS：61.139.2.69

在根据这些参数我们需要先配置好 IP 地址、网关和 DNS，并打开 DNS 缓存等。

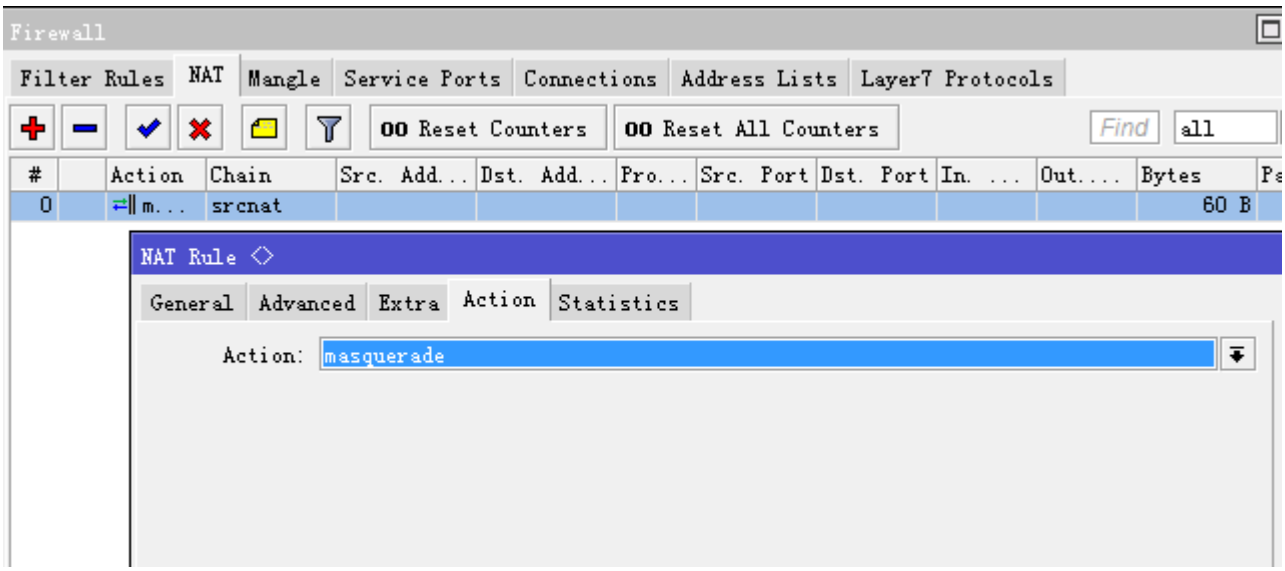
进入 ip address 配置 IP 地址：



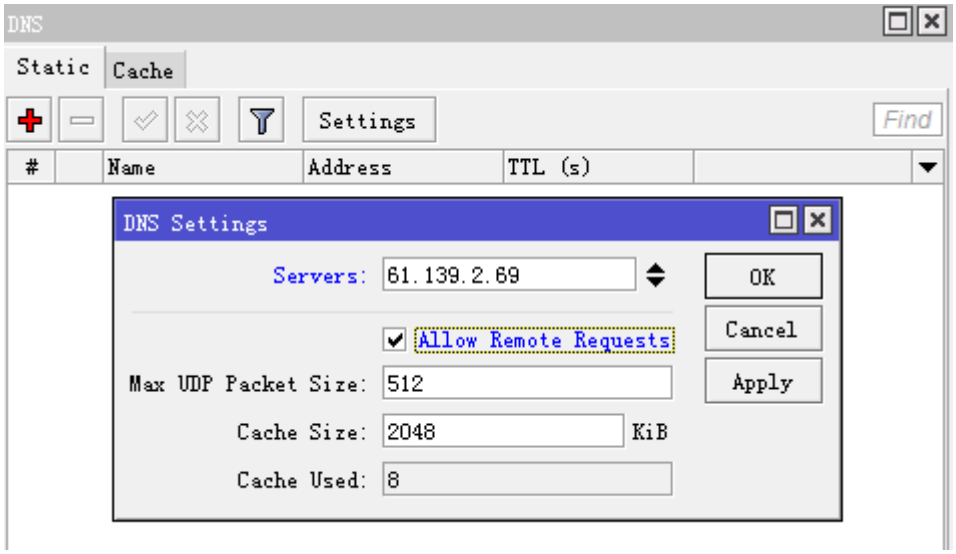
进入 ip route 配置网关



进入 ip firewall nat 设置好 NAT 伪装：



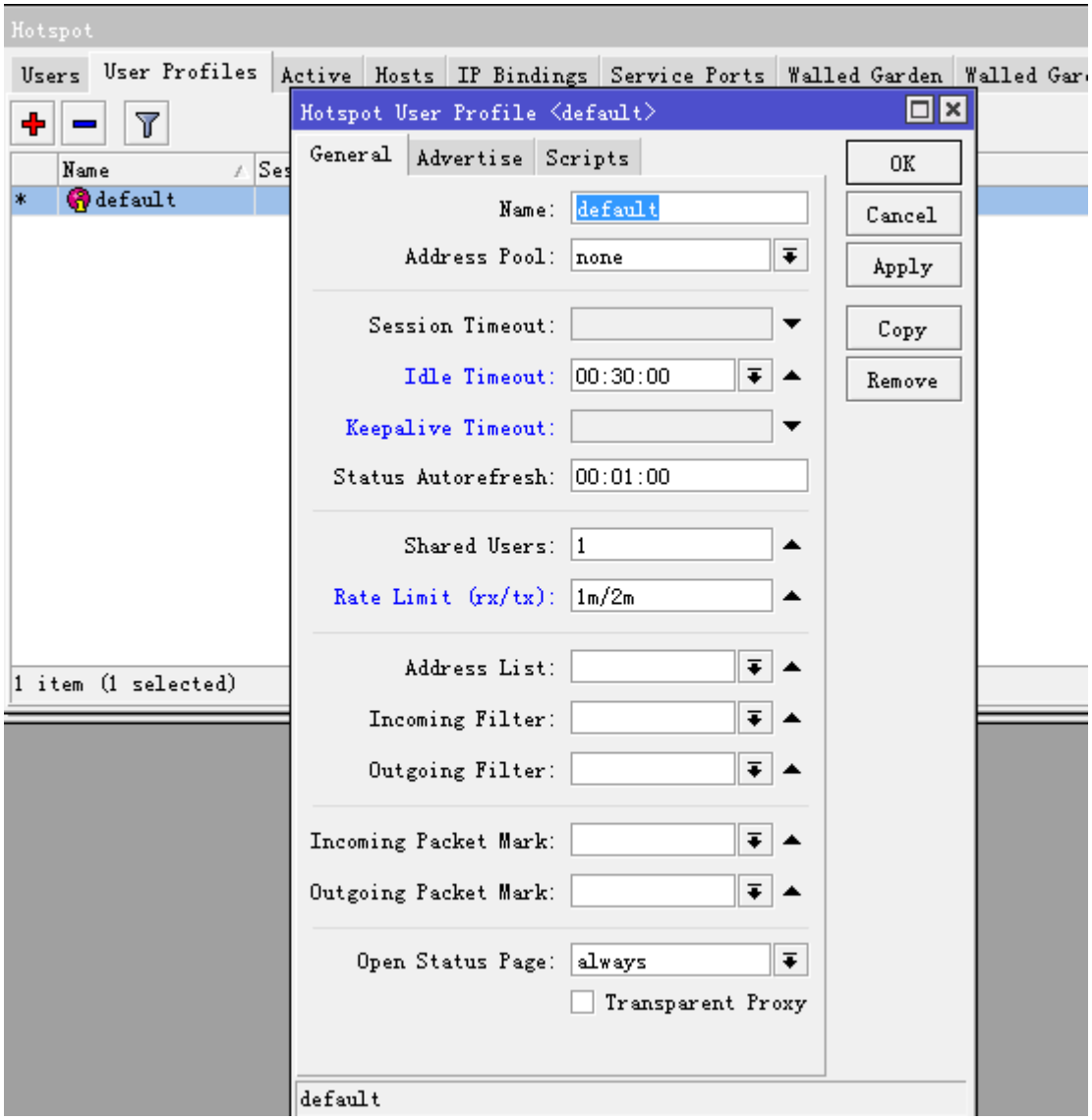
进入 ip dns 配置 DNS 缓存，DNS 和启用缓存对于 Hotspot 非常重要，如果 DNS 错误将导致用户认证跳转或无法浏览网页：



现在我们的基本参数已经配置完成，现在我们需要配置的 Hotspot 参数，配置 Hotspot 参数的基本流程是：

1、先进入 ip hotspot user profile 设置用户分组规则
2、然后在 ip hotspot user 添加用户的账号
3、进入 ip hotspot server profile 配置服务器规则
4、在 ip pool 中分配 IP 地址段，根据需要启用 DHCP 服务
5、在 ip hotspot server 添加并启用 hotspot 服务

现在我们进入 ip hotspot，并配置 ip hotspot use profile

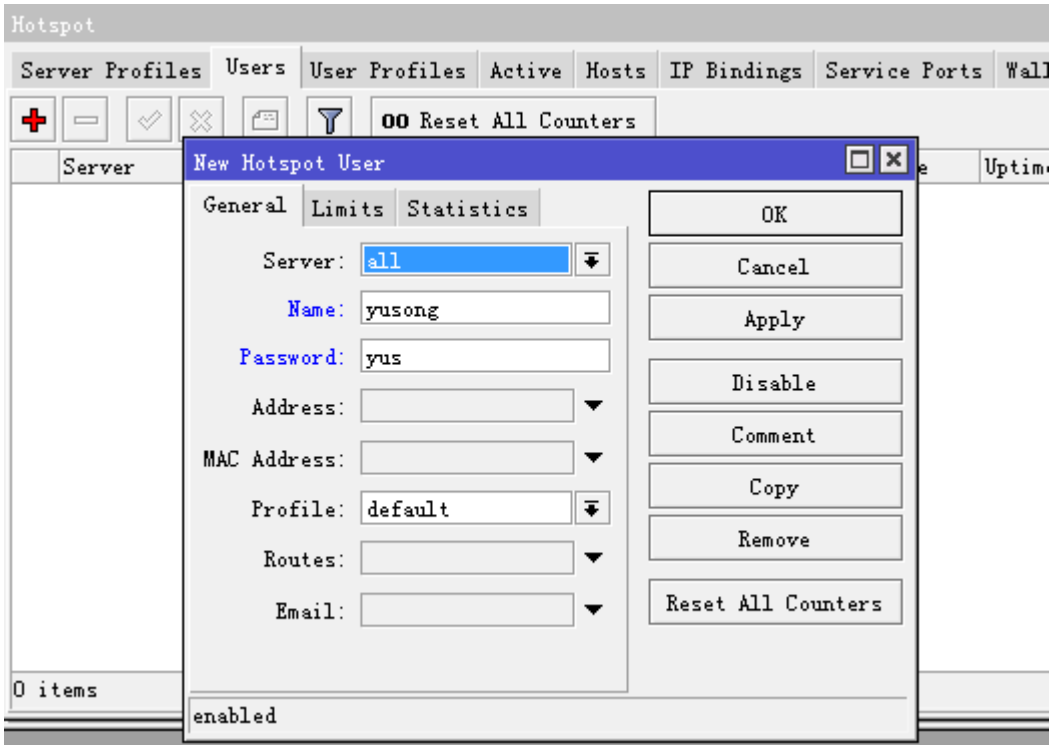


在 user profile 里面一般配置如下几个参数：

Idle-Timeout：用户在一定时间内没有任何流量发出后自动注销连接，这里我们设置 30 分
Keepalive-Timeout：路由器主动通过 ICMP 探测主机是否在线，如果在一定时间为探测到自动注销连接（如果用户机开启防火墙，路由器无法探测到）
Shared-users：账号的分享用户多少，默认为 1，即仅一个用户使用该账号。
Rate-Limit：分配每个账号带宽，格式为"上行 / 下行"，单位为 bit，只能使用整型，设置为 1m/2m
Transparent-proxy：透明代理功能是否开启，如果你启用了/ip proxy 透明代理,可以将该规则组的用户经过 proxy，并能作 web 缓存的功能
其他参数请参考具体 Hotspot 手册。

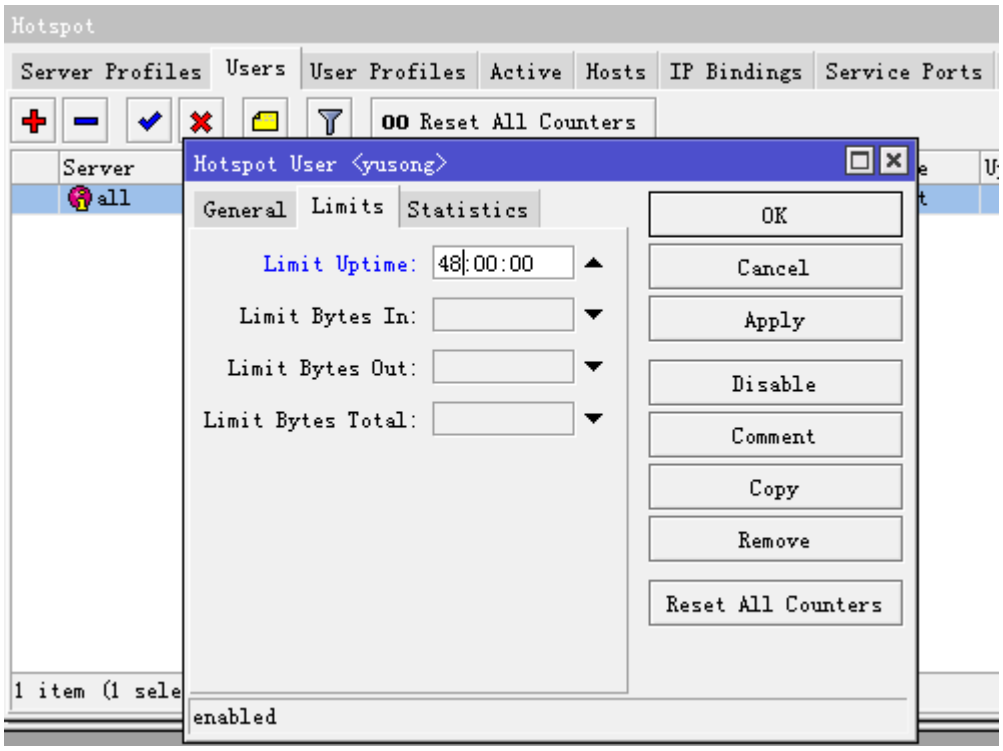
Address pool 这个是 DHCP 的地址池，给用户分配 IP，我们可以在 ip pool 中分配地址段，具体操作请参考 RouterOS 的 DHCP 操作。

在 user 配置用户登录账号和密码，以及所属的 profile 类型，这里默认 `server` 服务器为 `all`：

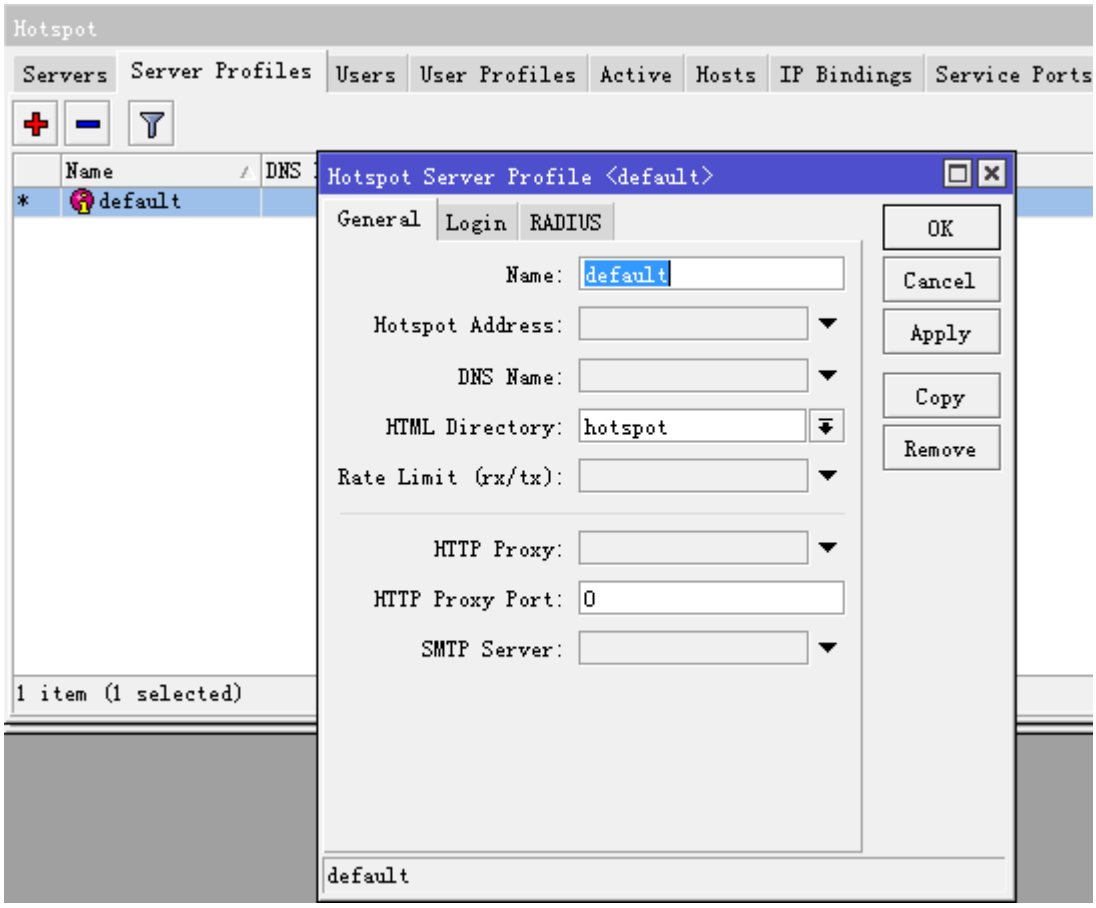


Name：用户名为 yusong
Password：密码为 yus
Profile：用户组规则，这里选择我们之前设置的 default 规则

在 Hotspot 我们可以对本地 User 做时间和流量的限制，我们可以选择 user 下的 limits 进行设置，例如我限制 yusong 的账号只能使用 48 小时



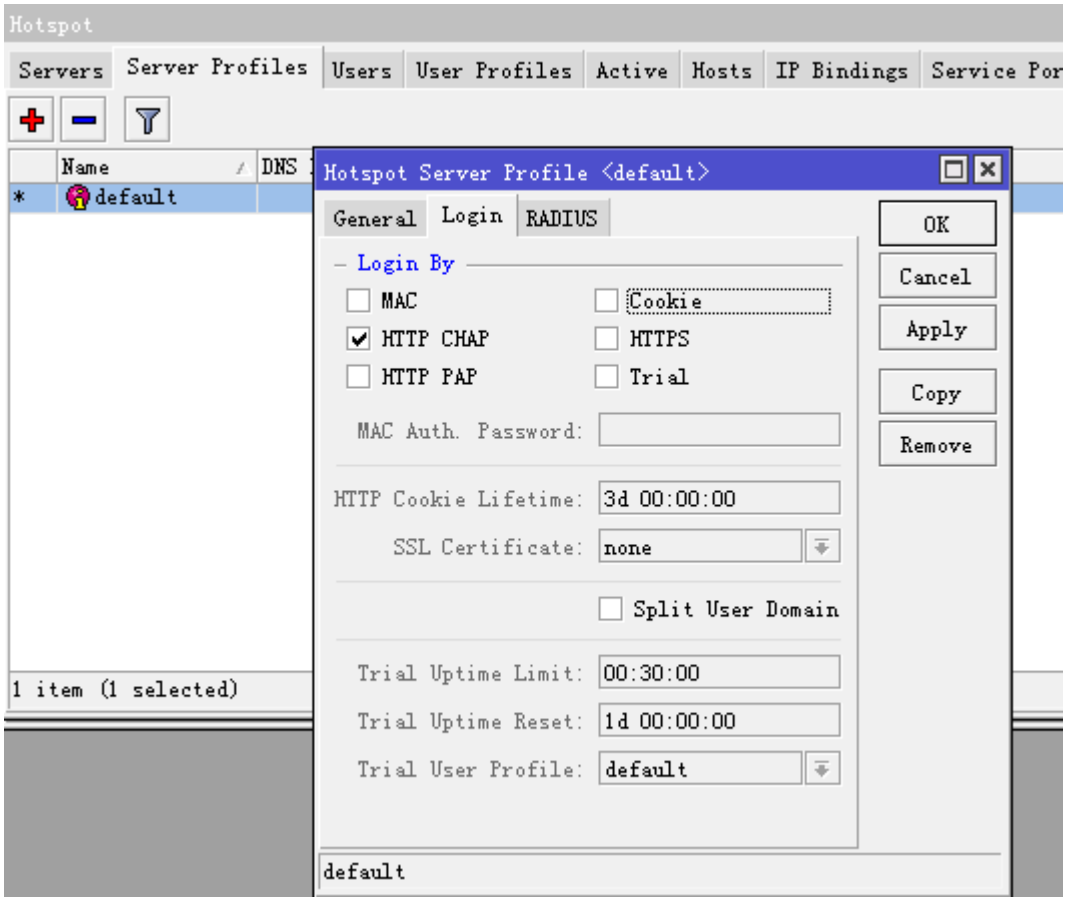
配置完用户规则后，进入 ip hotspot server profile，配置服务器器规则。



这里有几个参数我们需要说明：

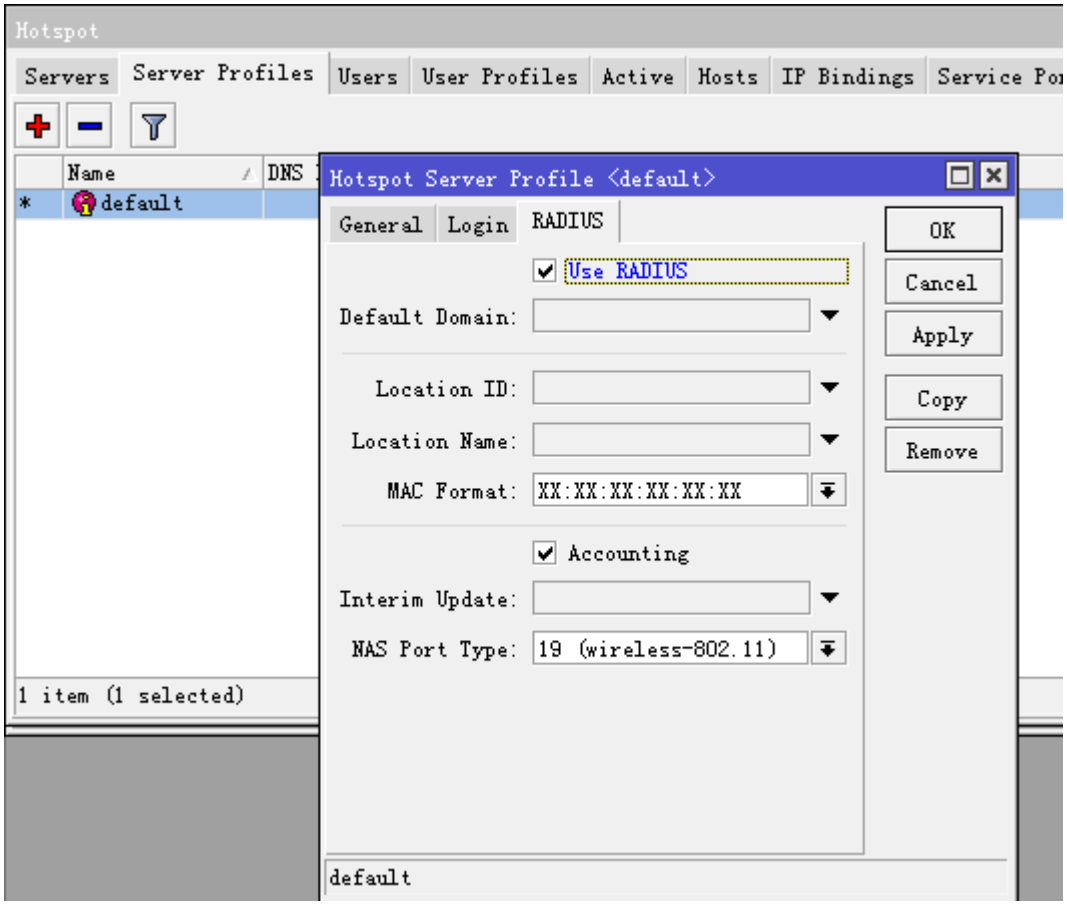
- **Hotspot address**：认证服务器的 IP 地址，如果你只有一个 hotspot 认证接口，这个参数可以预设不设置，但当你有多个接口的 hotspot 接口，且每个接口不同 IP 地址段，为了用户能都能同时访问一个 IP 地址认证，我们可以设置一个静态的地址，这样可以便于网络的管理。
- **HTML Directory**：该参数是指定 Hotspot 的认证也路径（在 Files 目录下可以找到），当你拥有多个 hotspot 认证接口时，你对不同接口的用户选择不同的认证页面，我们仅需要建立多个 server Profiles 规则。
- **Rate Limit**：该 hotspot 服务器规则下的总带宽，一般建议不用设置，由我们自定义用户的带宽。

配置 login 登录方式，即用户提交账号密码时采用的传输加密方式，一般预设启用 HTTP CHAP 即可，

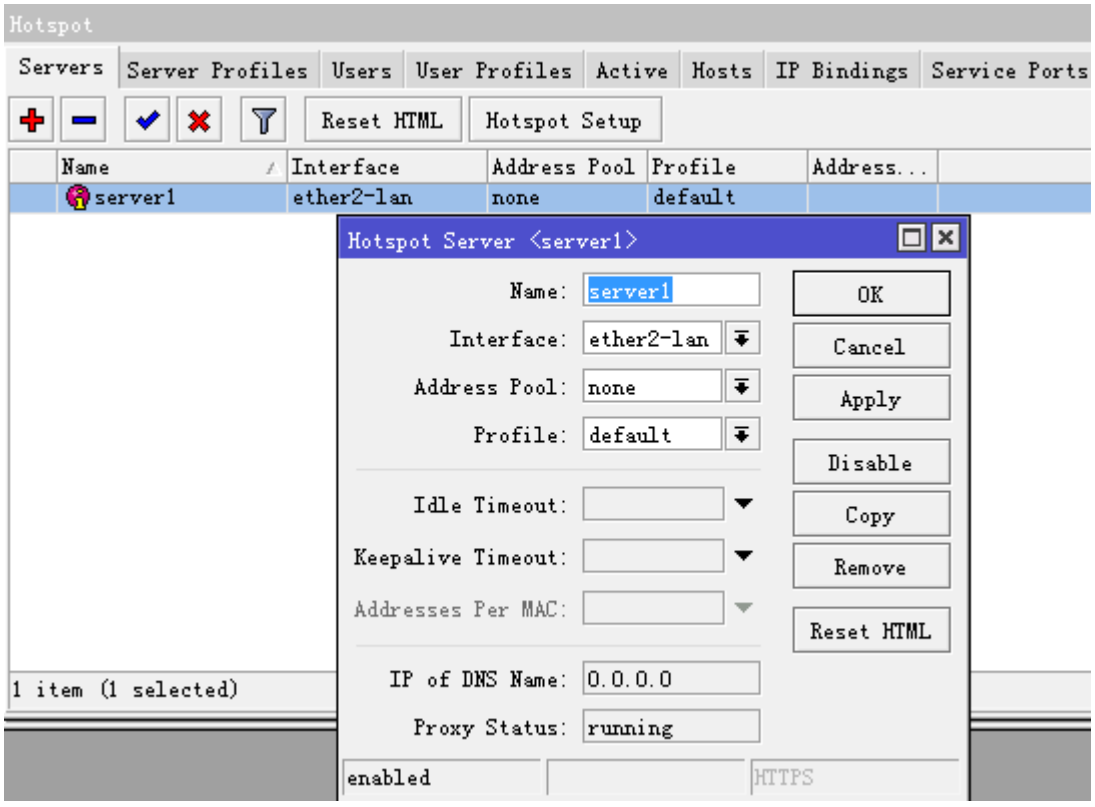


Cooke 等值一般不用选择，这里选择 HTTP CHAP 加密方式提交账号密码，HTTP PAP 是明文的账号密码提交。

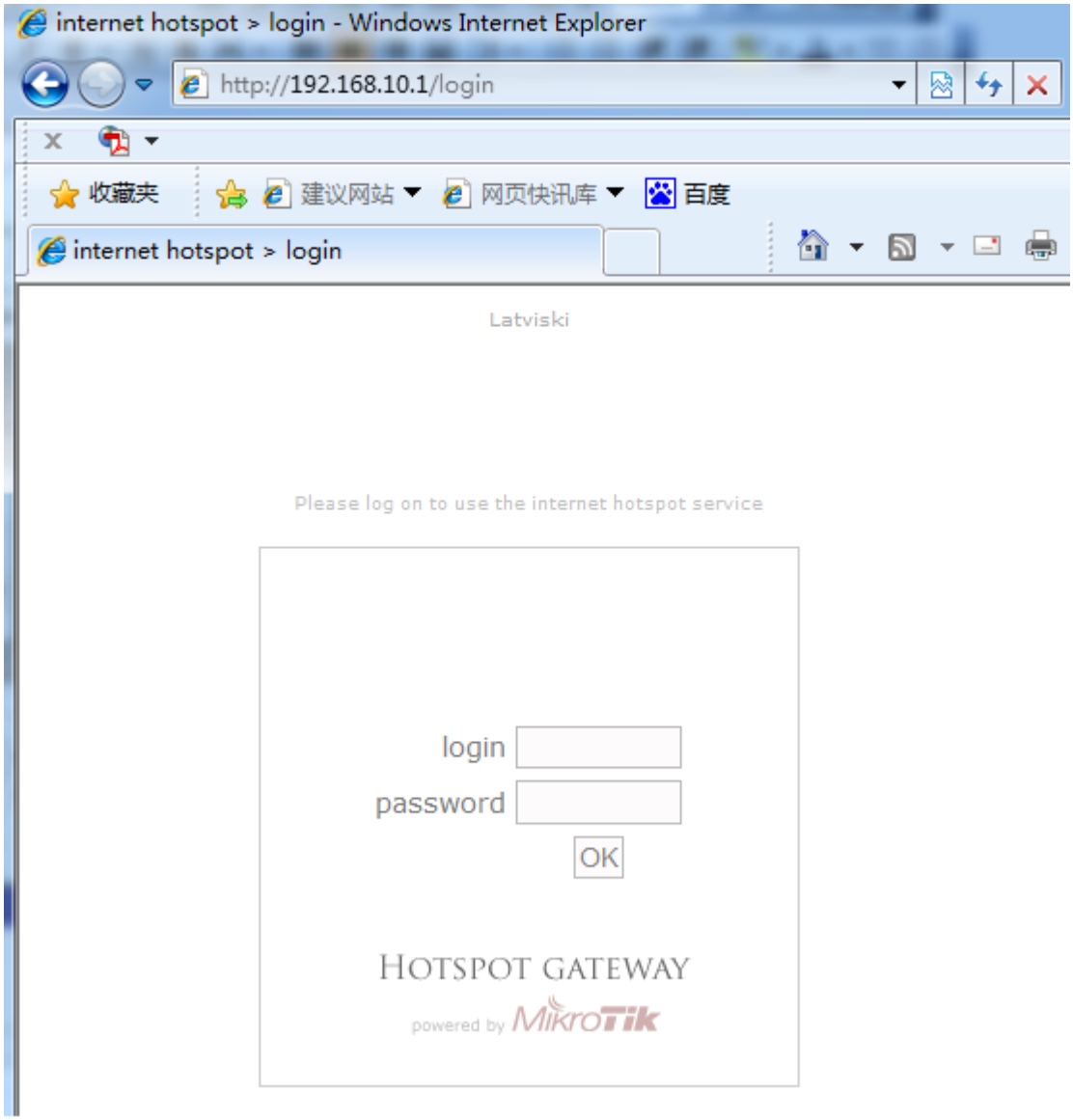
至于 RADIUS 根据需要开启，如果你有对应的 RADIUS 服务器，并要使用 RADIUS 计费，这里的 Use RADIUS 必须选择



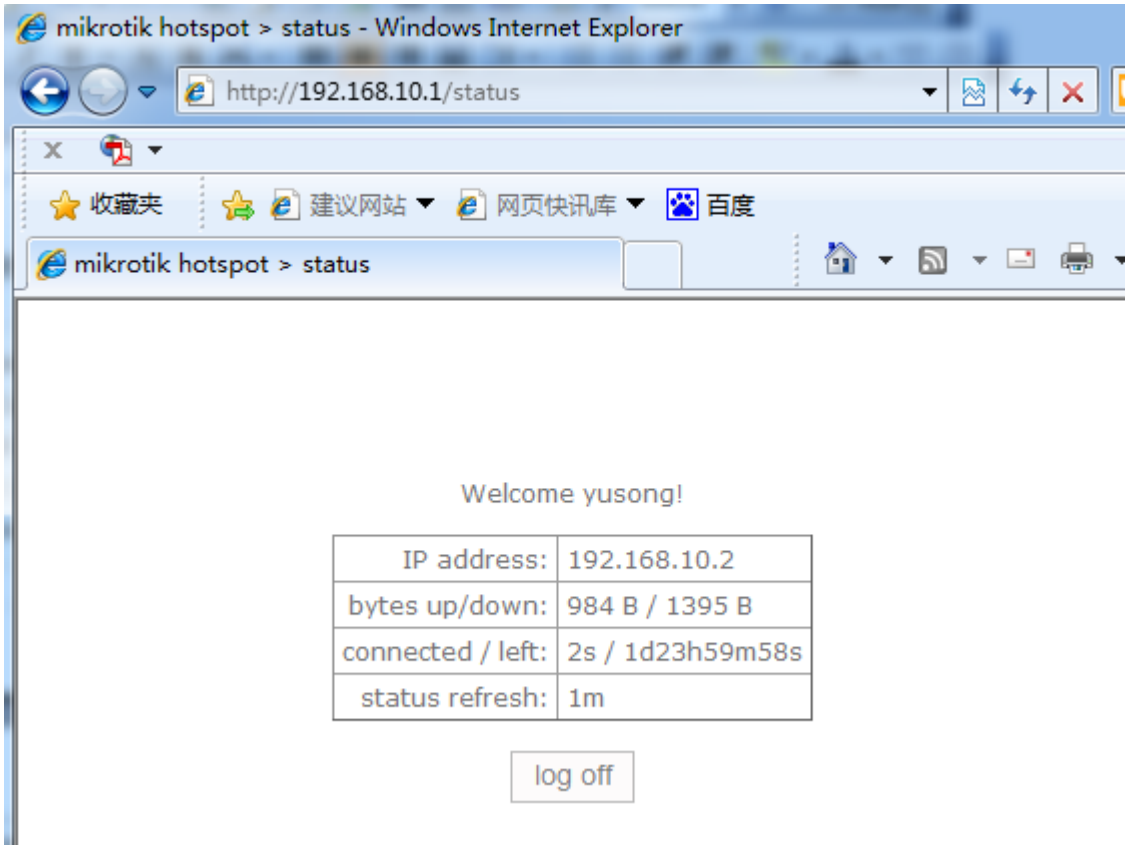
配置完成以上参数后，最后我们启用 Hotspot 服务器，并选择 interface 为我们的 lan 接口：



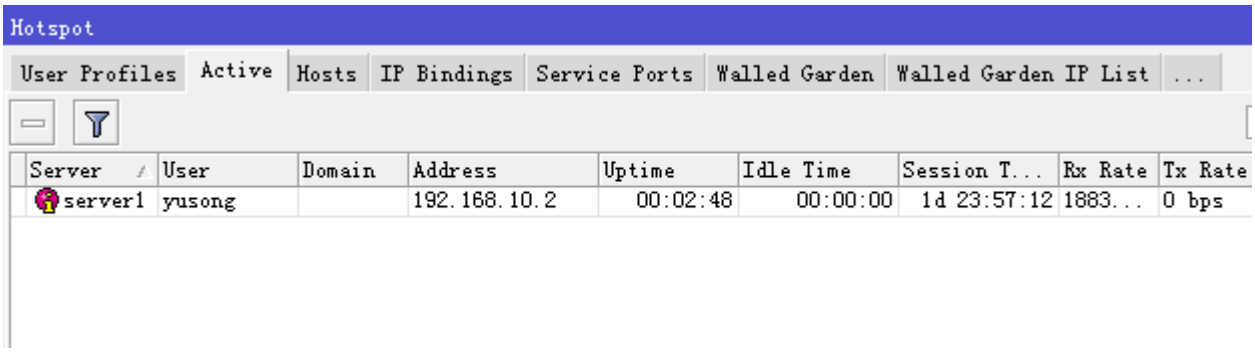
当启用完成后，所有对路由器或者外网访问都需要通过 web 认证，当用户随便输入一个网站都会跳转到认证页面，用户访问到的认证页面如下图：



我们输入账号和密码认证通过后，我们将跳转到以下页面：



这时我们可以在 ip hotspot active 中看到用户登录的在线情况：



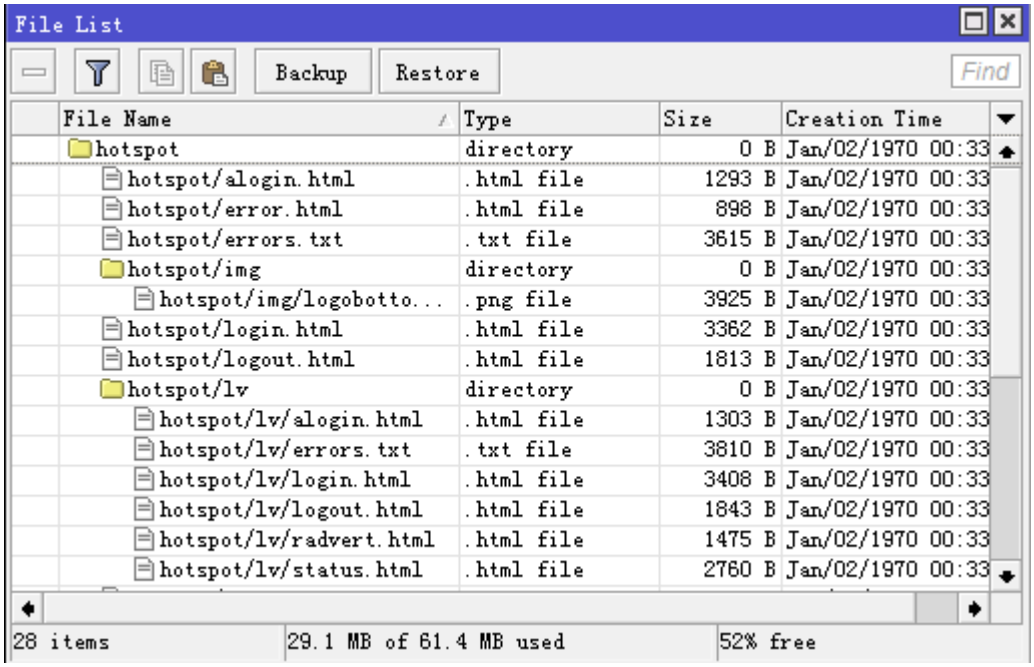
获取现时用户列表：

```
[admin@MikroTik] ip hotspot active> print
Flags: R - RADIUS, B - blocked
 #      USER              ADDRESS          UPTIME       SESSION-TIMEOUT IDLE-TIMEOUT
 0      yusong            192.168.10.2     4m17s        55m43s
[admin@MikroTik] ip hotspot active>
```

用户如果需要注销，通过在浏览器里输入 192.168.10.1 进入 Hotspot 认证网关，点击 log off 退出登录页面

认证页面

Hotspot 的认证登录页面是开放式的，即可以通过 RouterOS 的 files 目录下找到这些文件，Hotspot 在 files 中的默认文件名“Hotspot”，



认证页面我们可以通过各种网页制作软件修改 login.html、logout.html 和 status.html 的 web 接口得到你想要的网页画面或者 log。

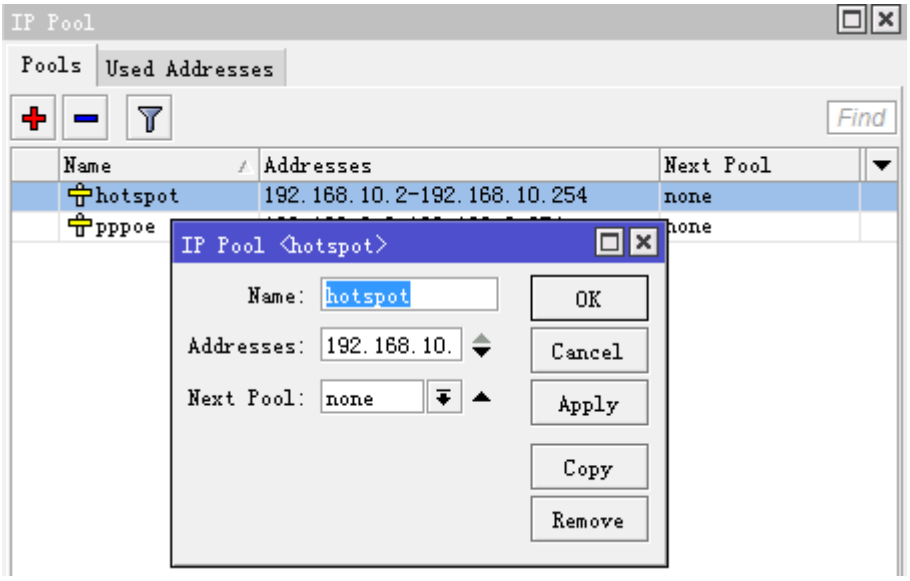
Hotspot 的 html 档提供了较开放的设置，能实现网页认证的要求，如首次登陆看见一个页面，然后在跳转到认证认页、认证后在到另外一个页面等，都可以通过修改 html 档代码实现。

20.11 Hotspot 即插即用功能

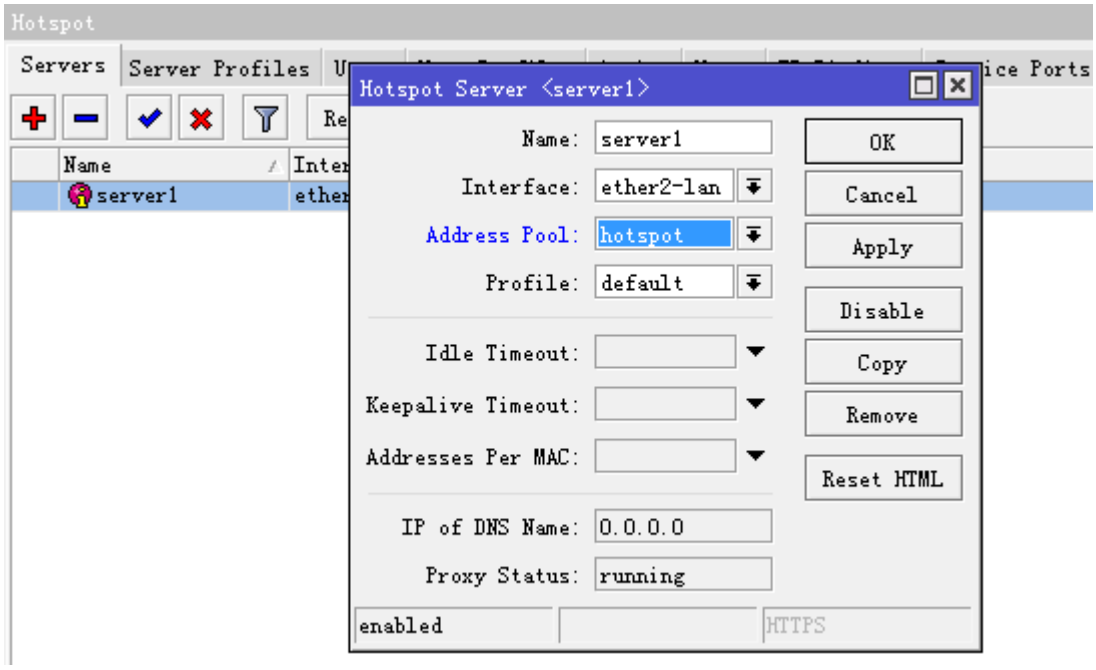
从 2.7 的版本就开始支持 upnp 的即插即用功能，即当用户和 Hotspot 认证服务器在同一局域网内，不管局域网用户设置任何的 IP 地址（前提是用户必须设置任意的 IP 地址、网关和 DNS）都可以被 Hotspot 认证服务器获取，并在 Hotspot 的 Host 中分配一个新的虚拟 IP 地址，并对用户作一对一的 NAT 转换。Hotspot 的即插即用方式分成适用于：流动性较强的公共场所，如机场、车站、公园，也可以应用到酒店和小区中。

Hotspot 服务器会在同一局域网内发送 ARP 广播，告诉局域网内的所有主机自己的网关设备，并为在线的主机分配一个虚拟的 IP 地址，这样客户主机在没有配置正确的 IP 地址情况下也能连接到 Hotspot 网关服务器，并认证上网。

在 2.9 以后的 Hotspot 启用 server 服务后，即插即用功能默认是打开的，但配置 Hotstop 需要在 hotspot server 中将 address pool 的地址池设置好，我们在 pool 中添加 192.168.10.2-192.168.10.254 的地址池，如图：

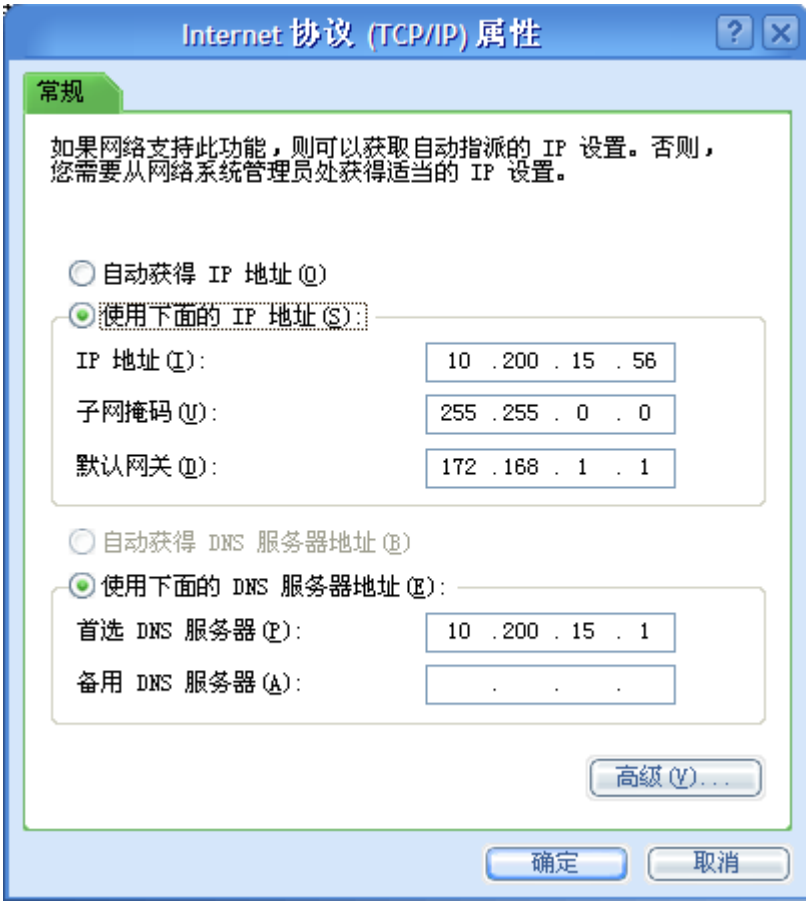


然后在 hotspot Severs 中选择 pool

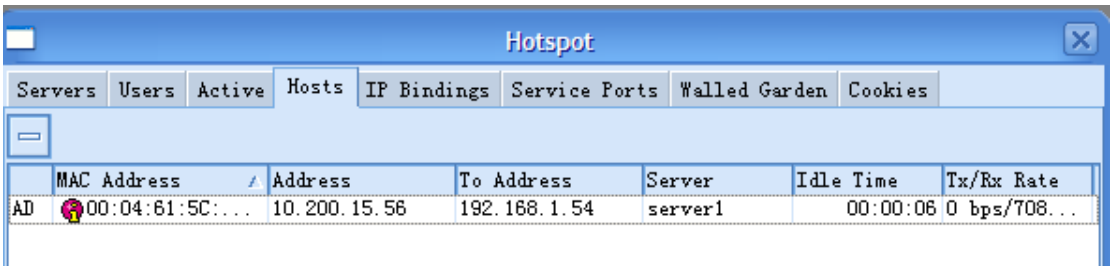


Addresses Per MAC 这个是每个 IP 对应的 MAC 地址，这里我们设置为 1，即一个 IP 对应一个 MAC 地址。

我们 windows 计算机的 IP 地址配置如下



在 Hotspot 的 host 列表中，我们可以看到，在同一局域网内的 windows 主机被 Hotspot 捕获后，自动为其分配 IP 地址，并做了对应关系



注： 如果 Hotspot 没有工作，可能的情况如下：

- 检查**/ip dns 是否**包含合法的 **DNS**，通过命令行或者 tools ping 中是否能解析域名，查看/ip dns setting 中的 allow- remote-request 已经启用。

```
/ping www.ex.com ，并确认 DNS 的缓存功能打开
```

- 确保连接追踪已经启用：**/ip firewall connection tracking set enabled=yes**

20.12 HotSpot 防火墙部分

除了在**/ip hotspot** 子目录本身的明显的动态规则（像主机及动态用户），一些附加的规则会在启动一个 HotSpot 服务时被动态添加到防火墙表中。不像 RouterOS 2.8 版本，添加规则为静态，非动态属性，且工作是有一对一 nat 算法完成的。

nat 规则

在**/ip firewall nat print dynamic** 命令你可以获取如下（在每条规则后跟有评注）：

```
0 D chain=dstnat hotspot=from-client action=jump jump-target=hotspot
```

把对数据报的所有 HotSpot 相关任务从 HotSpot 客户放到一个单独的链中：

```
1 D chain=hotspot protocol=udp dst-port=53 action=redirect to-ports=64872
2 D chain=hotspot protocol=tcp dst-port=53 action=redirect to-ports=64872
```

把所有 DNS 请求都重定向到 HotSpot 服务。64872 端口对所有 HotSpot 用户提供 DNS 服务。如果你想要 HotSpot 服务器也监听其他端口，在这里以同样方式添加规则，改变 **dst-port** 属性。

```
3 D chain=hotspot protocol=tcp dst-port=80 hotspot=local-dst action=redirect
    to-ports=64873
```

把所有 HTTP 登陆请求定向到 HTTP 登陆 servlet。64873 就是 HotSpot HTTP servlet 端口。

```
4 D chain=hotspot protocol=tcp dst-port=443 hotspot=local-dst action=redirect
    to-ports=64875
```

把所有 HTTPS 登陆请求定向到 HTTPS 登陆 servlet。64875 是 HotSpot HTTPS servlet 端口。

```
5 D chain=hotspot protocol=tcp action=jump hotspot=!auth jump-target=hs-unauth
```

所有其他的数据报除了 DNS 及来自未认证客户的登陆请求以外都应该通过 **hs-unauth** 链。

```
6 D chain=hotspot protocol=tcp action=jump hotspot=auth jump-target=hs-auth
```

来自认证用户的数据报通过 **hs-auth** 链

```
7 D ;;; www.mikrotik.com
    chain=hs-unauth dst-address=159.148.147.196 protocol=tcp dst-port=80
    action=return
```

首先在 **hs-unauth** 链中把所有影响 TCP 协议的东西都放到**/ip hotspot walled-garden ip** 子目录中。现在我们把 www.mikrotik.com 从重定向到登陆页面中排除。

```
8 D chain=hs-unauth protocol=tcp dst-port=80 action=redirect to-ports=64874
```

所有其他 HTTP 请求都被定向到监听 64874 的 Walled Garden 代理服务器。如果在**/ip hotspot walled-garden** 子目录有一个 HTTP 请求的 **allow** 条目，它将被转发到目的。否则，请求将会自动被重定向到 HotSpot 登陆 servlet（端口 64873）。

```
 9 D chain=hs-unauth protocol=tcp dst-port=3128 action=redirect to-ports=64874
10 D chain=hs-unauth protocol=tcp dst-port=8080 action=redirect to-ports=64874
```

默认设置的 HotSpot 假设只有这些端口才能用于 HTTP 代理请求。这两个条目用于“捕捉”客户到未知代理的请求。如：使的有可能让带有未知代理设置的客户与 HotSpot 系统能够一起工作。这个特性叫做“通用代理”。如果探测到一个客户正在使用某个代理服务器，系统将自动以 **http** hotspot 标志对数据报进行标记以便处理未知代理问题。注意已使用的端口（64874）与#8 规则中对 HTTP 请求的一样（所以 HTTP 和 HTTP 代理请求都由相同的代码处理）。

```
11 D chain=hs-unauth protocol=tcp dst-port=443 action=redirect to-ports=64875
```

HTTPS 代理监听 64875 端口

```
12 D chain=hs-unauth protocol=tcp dst-port=25 action=jump jump-target=hs-smtp
```

对 SMTP 协议的重定向也可以在 HotSpot 配置中定义。如果是这样，那么一个重定向规则将被放在 **hs-smtp** 链中。这个完成后以便带有未知 SMTP 配置的用户能通过服务提供商（你们的）的 SMTP 服务器发送邮件，而代替了用户在自己计算机配置的 SMTP 服务器。

```
13 D chain=hs-auth protocol=tcp hotspot=http action=redirect to-ports=64874
```

对认证用户提供 HTTP 代理服务。认证用户的请求可能需要透明的代理（“通用代理”技术以及广告特征）。**http** 标志会自动的放在被 HotSpot HTTP 代理探测到的服务器的 HTTP 代理请求（监听 64874 端口的）。这个完成后以便有代理设置的用户可以使用 HotSpot 网关代替用户在自己计算机上配置的代理服务器。这个标志也会被放在任何概要被配置为透明代理的用户所做的 HTTP 请求上。

```
14 D chain=hs-auth protocol=tcp dst-port=25 action=jump jump-target=hs-smtp
```

对授权用户提供 SMTP 代理（同#12 规则的一样）

包过滤规则

从**/ip firewall filter print dynamic** 命令，你可以获得：

```
 0 D chain=forward hotspot=from-client,!auth action=jump jump-target=hs-unauth
```

任何来自未认证且通过路由器的数据报都将被发送到 **hs-unauth** 链。**hs-unauth** 执行基于 IP 的 Walled Garden 过滤器。

```
 1 D chain=forward hotspot=to-client,!auth action=jump jump-target=hs-unauth-to
```

任何通过路由器到达客户的包都将被重定向到另一个叫做 **hs-unauth-to** 的链。这个链会拒绝到达客户的未认证请求。

```
 2 D chain=input hotspot=from-client action=jump jump-target=hs-input
```

任何从客户到达路由器本身的包将重定向到另一个叫 **hs-input** 的链。

```
 3 D chain=hs-input protocol=udp dst-port=64872 action=accept
```

```
4 D chain=hs-input protocol=tcp dst-port=64872-64875 action=accept
```

允许客户访问本地认证和代理服务。

```
5 D chain=hs-input hotspot=!auth action=jump jump-target=hs-unauth
```

所有其他来自未认证客户到路由器本身的数据流都将会与通过路由器的数据流一样的方式被处理。

```
6 D chain=hs-unauth protocol=icmp action=return
7 D ;;; www.mikrotik.com
    chain=hs-unauth dst-address=159.148.147.196 protocol=tcp dst-port=80
    action=return
```

不仅在 TCP 协议相关的 Walled Garden 条目被添加的 NAT 列表中，在包过滤器中 **hs-unauth** 链表也会添加在**/ip hotspot walled-garden ip** 目录中设置的东西。这就是为什么，尽管你只在 NAT 表中添加了一个条目却有两条规则的原因。

```
8 D chain=hs-unauth protocol=tcp action=reject reject-with=tcp-reset
9 D chain=hs-unauth action=reject reject-with=icmp-net-prohibited
```

任何没有被 Walled Garden 记录在表格上的都将被拒绝。注意拒绝 TCP 连接的 TCP 重启的使用。

```
10 D chain=hs-unauth-to action=reject reject-with=icmp-host-prohibited
```

用 ICMP 拒绝信息拒绝所有到达客户的包。

20.13 HotSpot HTTPS 认证登录

随着当前网络的发展，各大网站已经采用 HTTPs 加密 web 访问，因此在没有启用 HTTPs 登录的情况下使用 HotSpot 服务，如果请求的页面使用 HTTPs 协议，则导致无法将客户机重定向到 HotSpot 认证页面。为了避免这种情况，需启用"HTTPS 登录方式"

Hotspot HTTPs 登录提供以下优势：

- 提供 SSL 加密安全访问.
- 能为客户提供 HTTPs 的 URL 重定向.

通过下面的介绍，将帮忙大家使用 SSL 证书在 Hotspot 服务启用 HTTPs 登录，通过使用可信认证机构(CA)签名证书完成认证登录，同时也可以使用免费的自签名证书。

Self-signed certificates（自签证书）

自签名证书可以在没有公共 CA 参与的情况下免费生成。有多种免费的工具可以创建这样的证书。下面的示例将展示如何在 linux 机器上使用 OpenSSL 和 RouterOS CLI 来生成和签名自己的证书。

RouterOS 下创建证书事例

首先我们需要创建自己的 CA 签名证书

```
[admin@MikroTik] /certificate>add name=ca-template common-name=myCa
key-usage=key-cert-sign,crl-sign
[admin@MikroTik] /certificate>sign ca-template name=myCa
```

然后为 Hotspot 创建一份证书

```
[admin@MikroTik] /certificate>add name=Hotspot-template common-name=Hotspot
[admin@MikroTik] /certificate>sign Hotspot-template ca=myCa name=Hotspot
```

将 Hotspot 证书设置为信任

```
[admin@MikroTik] /certificate>set [find name=Hotspot] trusted=yes
```

以上操作完成后，将 Hotspot 证书导出

```
[admin@client] /certificate> print
Flags: K - private-key, L - crl, C - smart-card-key, A - authority, I - issued, R - revoked,
E - expired, T - trusted
 #           NAME       COMMON-NAME      SUBJECT-ALT-NAME
FINGERPRINT
 0 K   A   T ca         myca                                                  c14bb6bd7ae...
 1 K   A   T hotspot    hot                                                   c20a0e37560...
[admin@client] /certificate> export-certificate 1
```

导出证书可以直接在 file 下找到，并可以直接应用到 Hotspot 的 HTTPs 认证

基于 Linux 的 OpenSSL 事例

以下是基于 Linux 的 OpenSSL 创建证书事例，创建免费的自签证书，首先需要创建一个私有密钥，下面我通过树莓派 4b 上生成证书

首先需要创建一份 CA 签名证书，采用 1024 位加密，取名 hot.key，设置密码长度至少 4 位

```
root@raspberrypi:~# openssl genrsa -des3 -out hot.key 1024
Generating RSA private key, 1024 bit long modulus (2 primes)
...........+++++
.....................................+++++
e is 65537 (0x010001)
Enter pass phrase for hot.key:******
Verifying - Enter pass phrase for hot.key: ******
```

查看当前目录下生成的 ca 证书，

```
root@raspberrypi:~# ls
hot.key
```

用 hot.key 为 hotspot 生成一份证书，在生成过程中会要求填写相关信息，填写内容有向导提示，根据引导填写上既可

```
root@raspberrypi:~# openssl req -new -key hot.key -out hot.csr
Enter pass phrase for hot.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

最后导出前面证书和授权时间

```
root@raspberrypi:~# openssl x509 -req -days 365 -in hot.csr -signkey hot.key -out hot.crt
Signature ok
```

生成的 crt 文件，

```
root@raspberrypi:~# ls
hot.crt   hot.csr   hot.key
```

Linux 生成的证书文件，.crt 和.key 从 Linux 上下载后，上传到 RouterOS 的 files 目录下，然后通过以下命令导入证书

```
/certificate> import file-name=hot.crt
passphrase: ****
      certificates-imported: 1
      private-keys-imported: 0
             files-imported: 1
        decryption-failures: 0
  keys-with-no-certificate: 0
```

导入 key 文件

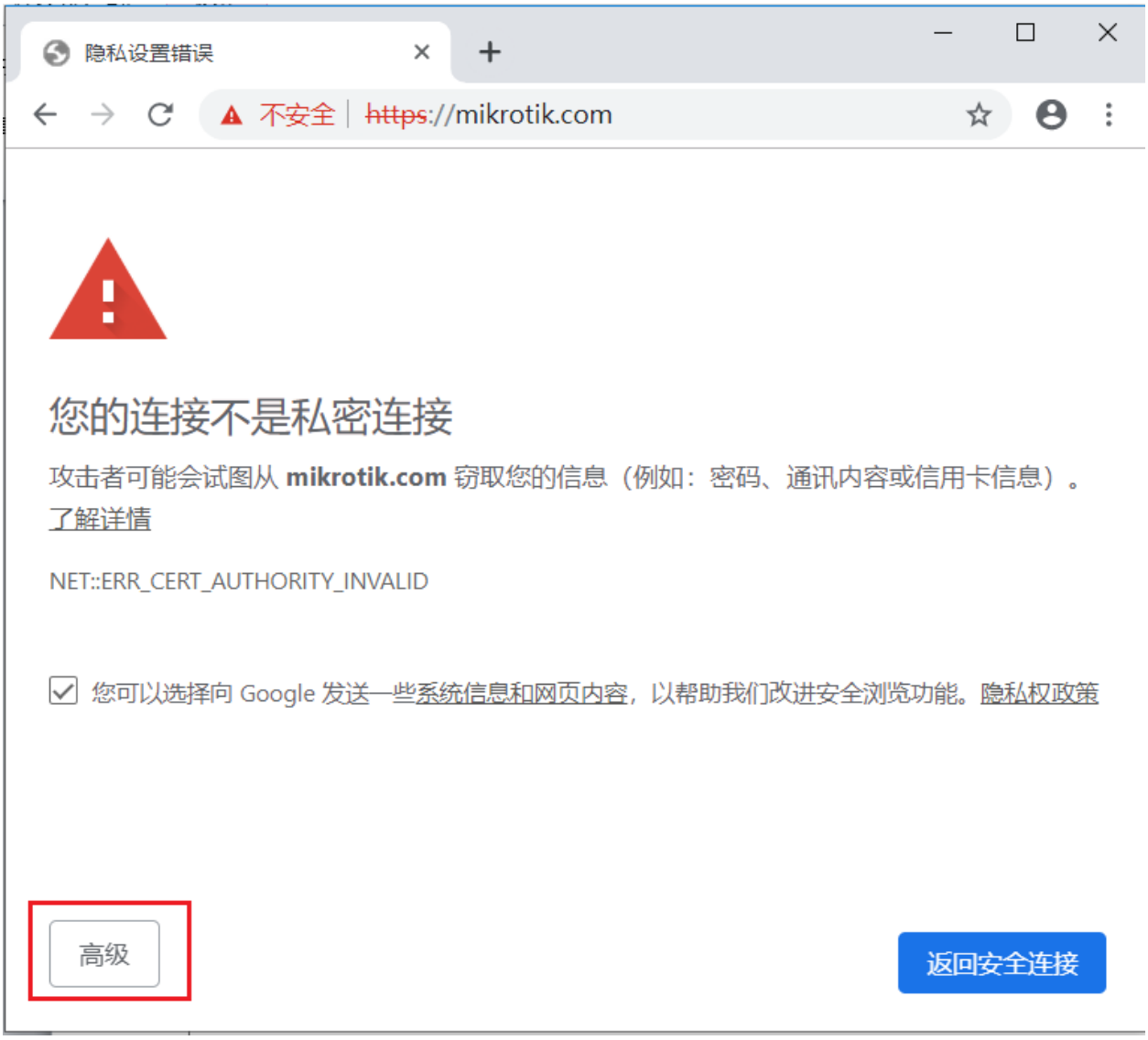
```
/certificate> import file-name=hot.key
```

```
passphrase: ****
      certificates-imported: 0
      private-keys-imported: 1
              files-imported: 1
         decryption-failures: 0
  keys-with-no-certificate: 0
```

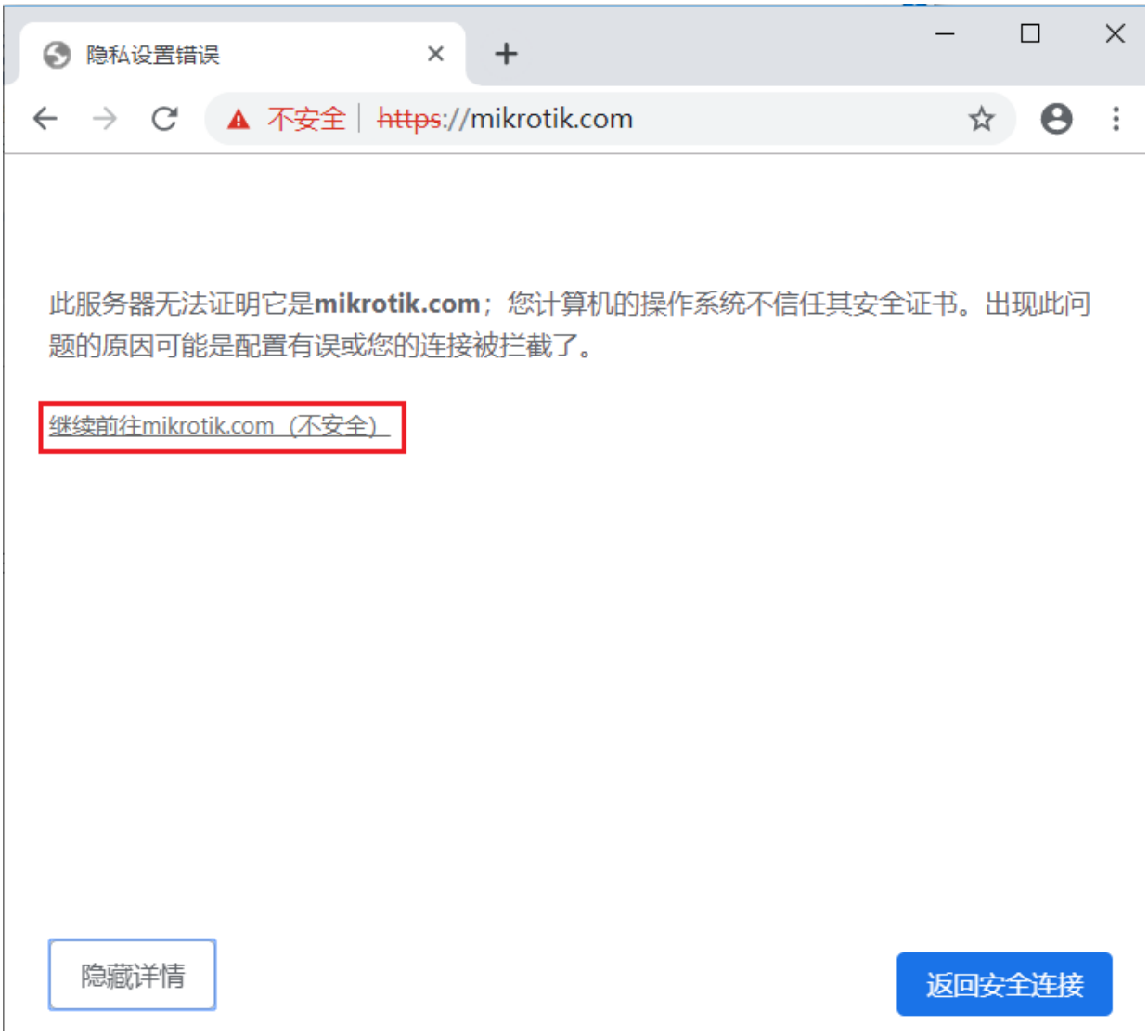
证书导入完成，并可以应用到 Hotspot 的 HTTPs 登录认证

注意：通过使用自签证书，在使用 SSL 的网页重定向会出现警告提示。由于 Hotspot 通过 SSL 协议，重定向 HTTPs 入口，RouterOS 的 SSL 接口将 被视作代理人进行转发。由于是非信任证书因此，浏览器会有 SSL 警告提示。

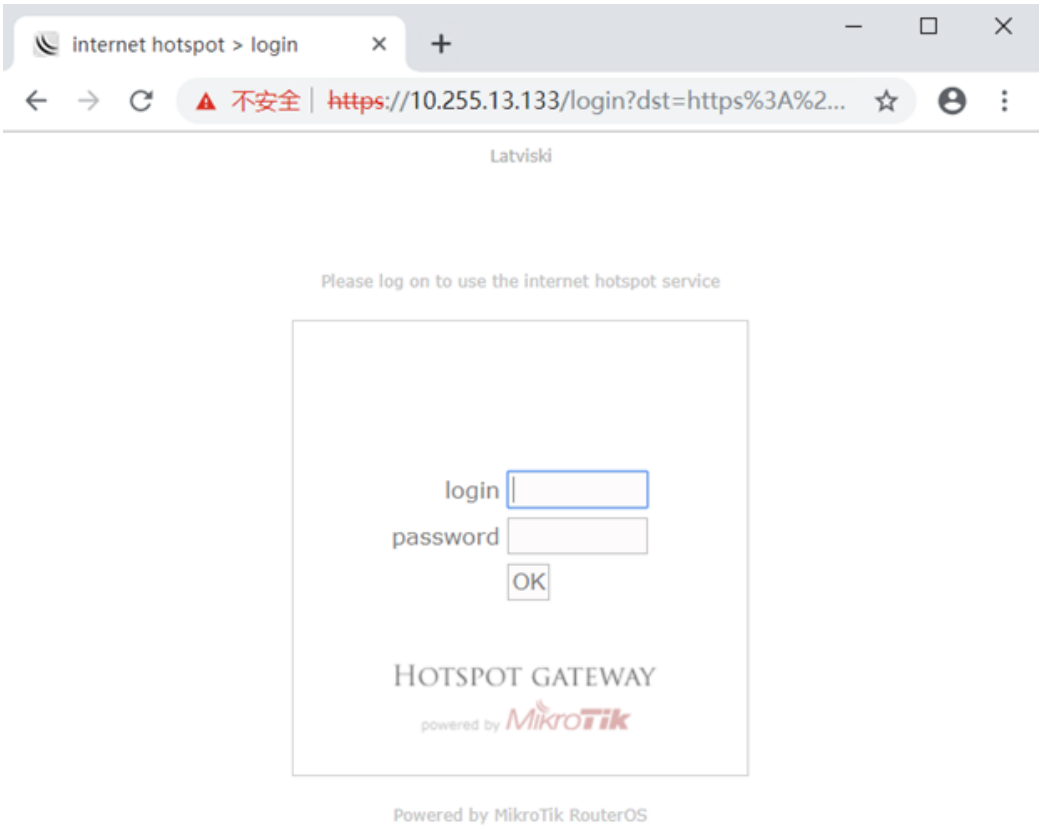
例如 chrom 浏览器，打开 mikrotik.com 网站，MikroTik 采用 https 打开时会提示：



点击高级，继续



点击继续前往访问的网站后，会出现登录页面



证书信任问题

要在客户端浏览器上使用 HTTPs 登录而不显示 SSL 警告，需要使用可信的 CA 签名证书。证书导入过程与前面的示例中描述的相同。

在使用不信任证书下，浏览器将警告终端用户关于重定向提示，即使有 CA 签名的证书，无法避免此警告消息。

Hotspot HTTPs 登录

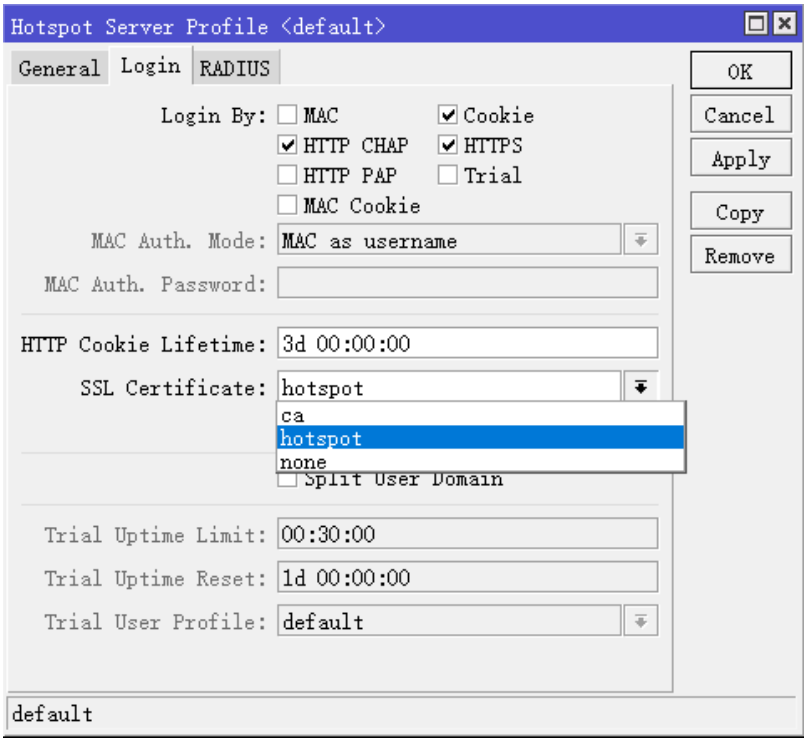
当你在路由器成功导入证书和私有密码后，首先需要启用 RouterOS 的 SSL 服务，进入/ip service 菜单下，设置启用，并选择对应的证书。

```
/ip service set www-ssl certificate=Hotspot disabled=no
```

下一步在 Hotspot 的 profile 下启用 HTTPs 登录.，并选择证书

```
/ip hotspot profile set hsprof1 login-by=https ssl-certificate=Hotspot
```

Winbox 下在 server profile 选择对应策略，点击 login 栏



现在所有来自未授权客户端的 HTTPs 请求，将被重定向到 Hotspot 登录页面，但如谷歌、facebook 等使用 SSL HSTS 的 HTTPS 站点将显示 SSL 错误，选择是否继续访问。在这种情况下，终端用户可以尝试访问不通的 web 站点。

SSL 证书密钥长度影响 CPU 负载

由于使用的是 443 端口，CPU 将大量处理 SSL 证书密钥数据，SSL 证书大小将影响 Hotspot 服务器上 www 服务的负载。这种情况，可从尚未认证的客户端主机上看到，例如 Dropbox（云端的文件服务，安装客户端软件情况下），都在试图联系它们的服务器，并不断地在 Hotspot 服务器上来回访问，通过实验在 60 台主机上只安装 Dropbox 软件，在较低端的 mipsbe 设备(如 RB951 或类似型号)上，CPU 负载最高可达 50-60%

解决问题的方法，

第一种：使用较小 SSL 密钥的长度，但只是降低了 CPU 负载。

第二个方法是在防火墙规则中禁用 HTTPs 重定向，副作用当然是 https 站点将不再被重定向到专属的登录认证页面，即完全不适用 HTTPs 重定向失效。

```
/ip firewall nat add chain=hs-unauth action=return protocol=tcp dst-port=443 place-before=0
```

第二十一章 PPPoE

PPPoE 基于以太网的点对点协议(Point to Point Protocol over Ethernet)，PPPoE 利用以太网组件的网络，建立 PPP 隧道，该协议具有用户认证及 IP 地址分发的功能。PPPoE 协议能对每一个主机实现控制、计费功能。极高的性能价格比使 PPPoE 在包括小区组网建设等一系列应用中广泛被使用。

RouterOS 支持 PPPoE 服务端（PPPoE 服务端也被称为访问集中器 Access Concentrator）和 PPPoE 客户端，同时支持 RADIUS 计费功能

支持的连接

- MikroTik RouterOS PPPoE 客户端到任何 PPPoE 服务器(Access Concentrator)
- MikroTik RouterOS PPPoE 服务器(Access Concentrator)到多个 PPPoE 客户端 (客户端包括几乎所有的操作系统和大部分路由器)

多连接 PPP 协议支持 MP，提供 MRRU 协议（能够传输 1500 和大数据报）和基于 PPP 连接的桥接 bridging（使用桥接控制协议 BCP，能发送基于 PPP 连接的原始以太网帧）这样能在没有 EoIP 协议的支持下，设置桥接。

规格

需要功能包: ***ppp***
需要等级： Level1 (限制 1 个连接) , Level3 (限制 200 个连接) , Level4 (限制 200 个连接) , Level5 (限制 500 个连接) , Level6 (无限制)
操作路径： ***/interface pppoe-server***, ***/interface pppoe-client***
硬件要求: PPPoE 服务器的需要增加 RAM 和提高 CPU 性能，每个连接使用 9KB（如果 Queue 流控开启，额外还需要增加 10KB），当前 PPPoE Server 最大支持 65535 个连接。

简单配置实例

下面我们通过两个事例，举例如何配置 PPPoE 客户端和 PPPoE 服务器：

这里通过命令行我们添加一个 PPPoE 拨号客户端操作，账号和密码都是 123，接口是 ether1，设置添加默认路由和获取对端 DNS，配置如下：

```
[admin@MikroTik] /interface pppoe-client> add user=123 password=123 interface=ether1
add-default-route=yes use-peer-dns=yes disabled=no
```

通过 print 命令查看配置情况：

```
[admin@MikroTik] /interface pppoe-client> print
Flags: X - disabled, R - running
 0 X   name="pppoe-out2" max-mtu=auto max-mru=auto mrru=disabled interface=ether1 user="123"
password="123" profile=default
       keepalive-timeout=60 service-name="" ac-name="" add-default-route=yes
default-route-distance=0 dial-on-demand=no
       use-peer-dns=yes allow=pap,chap,mschap1,mschap2
```

创建 MikroTIk RouterOS PPPoE 服务器，先看看以下的配置步骤和参数

1. 添加一个 IP 地址池，用户分配用户获取的地址从 10.1.1.62 到 10.1.1.72；
2. 添加 PPP profile 的配置策略组；
3. 添加 PPP secret，用户的账号密码和所属 profile；
4. 添加 PPPoE Server 到对应的网络接口。

1、配置地址池：

```
/ip pool
add name="pppoe-pool" ranges=10.1.1.62-10.1.1.72
```

2、配置 PPP 策略组：

```
/ppp profile
add name="pppoe-profile" local-address=10.1.1.1 remote-address=pppoe-pool
```

3、添加用户账号和密码

```
/ppp secret
add name=user password=passwd service=pppoe profile=pppoe-profile
```

4、添加 PPPoE server 到指定接口

```
/interface pppoe-server server
add service-name=internet interface=wlan1 default-profile=pppoe-profile disabled=no
```

21.1 PPPoE Client 设置

操作路径：**/interface pppoe-client**

属性	描述
ac-name (*字符*; 默认: **""**)	访问控制器名称，该字段可以为空，客户端可以连接到同一广播域的任何访问控制器（PPPoE Server）
add-default-route (*yes|no*; 默认: **no**)	客户端拨号完成后，是否自动添加默认路由。
allow (*mschap2|mschap1|chap|pap*; 默认:**mschap2,mschap1,chap,pap**)	选择允许的验证模式，默认配置下所有模式被允许
default-route-distance (*byte [0..255]*; 默认:**1**)	当 add-default-route 被启用，允许设置默认路由的路由距离值
dial-on-demand (*yes|no*; 默认: **no**)	连接 AC（PPPoE Server）只有在发出流量产生时进行。如果选择该属性，此时路由器会想路由表添加一条 10.112.112.0/24 路由，是否建立连接。
interface (*字符*; 默认:)	用于连接 PPPoE Server 的接口名称

keepalive-timeout (整型; 默认:**60**)	设置 keepalive 超时时间，单位秒。
max-mru (*整型*; 默认: **1460**)	最大接收单元
max-mtu (*整型*; 默认: **1460**)	最大传输单元
mrru (*整型：512..65535* | *disabled*; 默认: **disabled**)	在链路上能接收的最大数据报长度，如果一个数据报大小超过隧道的 MTU，将会分割到多个数据报中，允许全尺寸 IP 或以太网数据报发送到隧道中
name (*字符*; 默认: **pppoe-out[i]**)	PPPoE 接口名称，在没有指定名称的情况下，由 RouterOS 自动生成。
password (*字符*; 默认:)	用户认证密码
profile (*字符*; 默认: **default**)	连接后生成的策略配置，在 /ppp profiles 定义
service-name (*字符*; 默认: **""**)	指定服务名，也可以为空，即可以连接任何 PPPoE Server
use-peer-dns (*yes*|*no*; 默认: **no**)	是否获取 PPPoE Server 分配的 DNS
user (*字符*; 默认: **""**)	用户认证账号

监测 PPPoE 客户端

命令名称: **/interface pppoe-client monitor**

显示当前 PPPoE 客户端连接的状态信息，以下为可以获得的只读属性

属性	描述
ac-mac (*MAC 地址*)	客户端连接到访问集中器（PPPoE Server）的 MAC 地址
ac-name (*字符*)	访问集中器（PPPoE server）的名称
active-links (*整型*)	绑定 MLPP 连接的数量
encoding (*字符*)	当前连接使用的加密和编码信息
local-address (*IP 地址*)	分配给客户端的 IP 地址
remote-address (*IP 地址*)	服务端 IP 地址，即网关地址
mru (*integer*)	该连接有效的 MRU
mtu (*integer*)	该连接有效的 MTU
service-name (*string*)	使用的服务名称
status (*string*)	当前连结状态，包含如下状态： • dialing, • verifying password...,

	• connected, • disconnected.
uptime (*time*)	连接时长，显示方式：天，时，分，秒

实时查看 **pppoe-out1** 连接情况：

```
[admin@client] /interface pppoe-client> monitor pppoe-out1
           status: connected
           uptime: 1m59s
      active-links: 1
         encoding:
     service-name: service1
          ac-name: MikroTik
          ac-mac: D4:CA:6D:FA:4B:2F
              mtu: 1480
              mru: 1480
    local-address: 192.168.88.18
   remote-address: 192.168.88.1
-- [Q quit|D dump|C-z pause]
```

21.2 ADSL 拨号上网事例

```
ADSL 用户名: user@169
        密码: 1234
Service Name: CHN-Telecom
```

1：添加 PPPOE Clients

```
[admin@Router] interface pppoe-client>
[admin@Router] interface pppoe-client> add interface=ether1 mtu=1492 mru=1492
service-name=CHN-Telecom user= user@169 password=1234 add-default-route=yesuse-peer-dns=yes
[admin@ROUTER] interface pppoe-client> print
Flags: X - disabled, R - running
0   X name="pppoe-out1" mtu=1492 mru=1492 interface=ether1 user=user@169
      password=1234 profile=default service-name=CHN-Telecom ac-name=""
      add-default-route=yes dial-on-demand=no use-peer-dns=yes
```

PPPOE 拨号已经配置好，接下来将 ADSL MODEM 的网线连接好进行以下操作，即可连接完成：

```
[admin@Router] interface pppoe-client>enable 0
[admin@Router] interface pppoe-client> monitor pppoe-out1
           status: "connected"
           uptime: 10s
         encoding: "none"
     service-name: "CHN-Telecom"
          ac-name: ""
```

```
        ac-mac: 00:C0:DF:07:5E:E6
```

之后还需在 ip firewall mangle 中添加一条规则：

```
[admin@Router] ip firewall mangle> add chain=forward protocol=tcp tcp-flags=syn action=change-mss
new-mss=1440
[admin@Router] ip firewall mangle> print
Flags: X - disabled, I – invalid
0    chain=forward protocol=tcp tcp-flags=syn action=change-mss
     new-mss=1440
```

最后如果你要起用 nat 功能，不要忘了在 ip firewall nat 设置 IP 伪装。

21.3 Scanner（搜索器）

从 RouterOS v3.21 开始，新增 PPPoE Scanner 搜索器工具，允许你搜索在同一广播域里，所有运行中的 PPPoE 服务器，这样有助于对网络故障和对端设备的检测。

操作名利如下：

```
/interface pppoe-client scan <interface>
```

显示以下只读属性

属性	描述
service (*string*)	服务器配置的服务名
mac-address (*MAC*)	探测到服务器的 MAC 地址
ac-name (*string*)	访问控制器的名称

21.4 PPPoE Server 设置

操作路径: ***/interface pppoe-server server***

PPPoE server (access concentrator 访问集中器)支持在每一个接口上的多个 PPPoE 服务，通过设置 service-name 区分，当前 RouterOS 的 PPPoE Server 最大支持 65535 个连接。PPPoE server 是当前运营商和小区组网重要的组成部分，能对用户主机进行认证计费，管理和流量控制。

属性	描述
authentication (*mschap2* | *mschap1* | *chap* | *pap*; 默认:**"mschap2, mschap1, chap, pap"**)	验证协议
default-profile (*字 符* ; 默	选择 PPPoE Server 的默认策略，对应选择 User profile

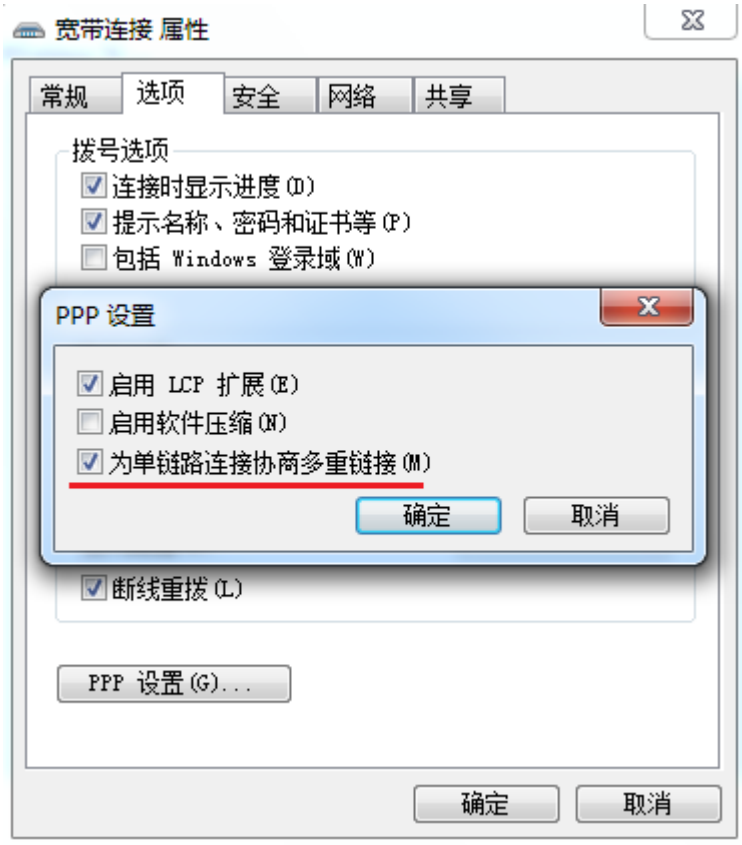
认: **"default"**)	策略条目，默认为里的 Default 策略
interface (*字符*; 默认: **""**)	接入 PPPoE server 的网络接口
keepalive-timeout (*时 间* ; 默认: **"10"**)	定义在路由器开始发送 keepalive 数据报的时钟周期，如果客户端没有流量或在 keepalive 时钟周期内响应，将宣告该客户端中断连接
max-mru (*整型*; 默认: **"1480"**)	最大接受单元，由于隧道协议原因，将原有 1500 以太网封包减少 20。
max-mtu (默认; 默认: **"1480"**)	最大传输单元，由于隧道协议原因，将原有 1500 以太网封包减少 20。
max-sessions (*整型*; 默认: **"0"**)	PPPoE server 允许客户端连接数量，如果设置'0' 代表无限制
mrru (*整型: 512..65535 | disabled*; 默认:**"disabled"**)	如果一个数据报大于隧道的 MTU，在该连接上能接收到最大数据报长度，通过分割数据报长度，允许全尺寸的 IP 或以太网包发到该隧道中。
one-session-per-host (*yes | no*; 默认: **"no"**)	仅允许一个主机一个会话，通过 MAC 地址限制，如果一个主机尝试建立新的连接，那该主机之前建立的会话，将被关闭。
service-name (*字符*; 默认: **""**)	PPPoE 服务名，服务器将接受客户端发送到 PADI 信息的 service-name，且满足 service-name 与 PPPoE 服务器的 service-name 匹配或 PAID 的 service-name 为空

注：keepalive-timeout 值通常情况下设置为 10。如果你设置为 0，路由器将不会断开客户端，直到他们自己注销或是路由器重启该用户账号才会断开。

安全提示：这个需要注意一个细节，PPPoE 是基于二层的隧道认证协议，因此不需要在 PPPoE 服务的接口上配置 IP 地址，因此建议管理与不要再 PPPoE 服务接口上配置 IP 地址（除非有特殊网络需要），避免出现用户可能不通过 PPPoE 验证即可上网的情况。

MRRU 介绍

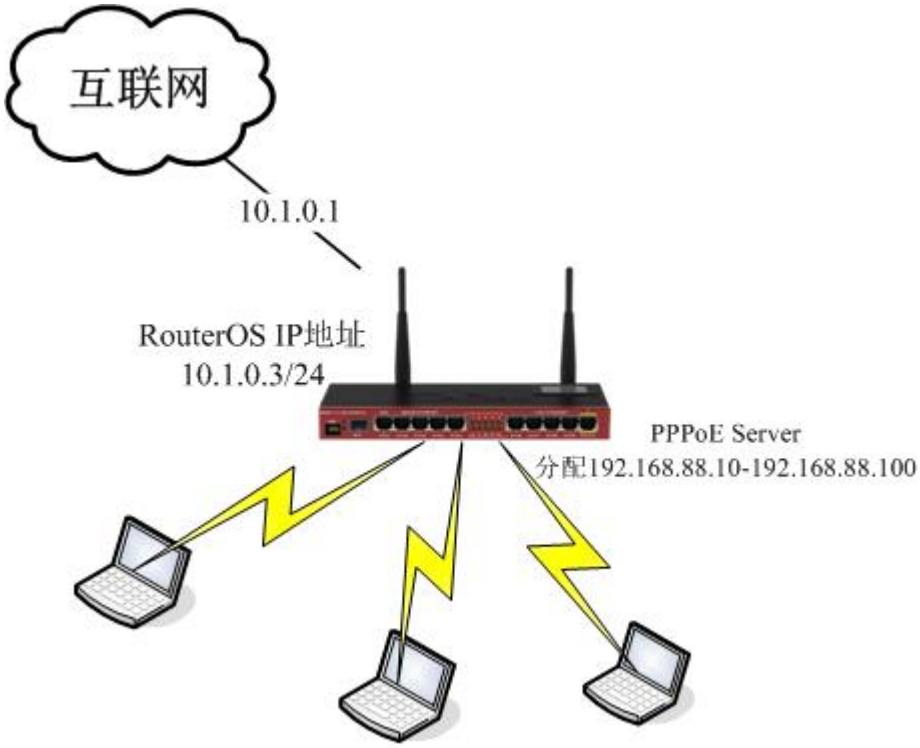
MRRU 意思为基于单连接的 MP，该协议被为拆分大数据包为更小的。在 windows 下在网络属性下，设置按钮中打开“为单链路连接协商多重连接”MRRU 是强行设置为 1614。这个设置有益于超载线路 MTU 探测失败。且 MP 协议应在双方都被启用。



21.5 基于 802.11g 无线网络的 PPPoE 服务

这里通过一个基于 WiFi 网络的 PPPoE 服务认证，RouterOS 可以将 PPPoE 服务器设置在一个 AP 访问节点（Access Point）,任意一个支持 PPPoE 认证的客户端都可以连接到 AP 的 PPPoE 认证。无线网卡的 MTU 可以设置为 1600，因此接口上的 MTU 设置为 1500，这可以充分利于 1500byte 传输数据报，并避免 MTU 比 1500 低出现的任何问题。

让我们参考下面的配置，MikroTik 无线 AP 能使无线客户端通过验证后访问到本地的网络：



首先，需要配置 RouterOS 的无线网卡，将无线网卡配置为 AP-bridge 模式：

```
[admin@PPPoE-Server] interface wireless> set 0 mode=ap-bridge   frequency=2442 band=2.4ghz-b/g
ssid=mt disabled=no
[admin@PPPoE-Server] interface wireless> print
Flags: X - disabled, R - running
 0      name="wlan1" mtu=1500 mac-address=00:01:24:70:53:04 arp=enabled
        disable-running-check=no interface-type=Atheros AR5211
        radio-name="000124705304" mode=station ssid="mt" area=""
        frequency-mode=superchannel country=no_country_set antenna-gain=0
        frequency=2412 band=2.4ghz-b scan-list=default rate-set=default
        supported-rates-b=1Mbps,2Mbps,5.5Mbps,11Mbps
        supported-rates-a/g=6Mbps,9Mbps,12Mbps,18Mbps,24Mbps,36Mbps,48Mbps,
                            54Mbps
        basic-rates-b=1Mbps basic-rates-a/g=6Mbps max-station-count=2007
        ack-timeout=dynamic tx-power=default tx-power-mode=default
        noise-floor-threshold=default periodic-calibration=default
        burst-time=disabled fast-frames=no dfs-mode=none antenna-mode=ant-a
        wds-mode=disabled wds-default-bridge=none wds-ignore-ssid=no
        update-stats-interval=disabled default-authentication=yes
        default-forwarding=yes default-ap-tx-limit=0 default-client-tx-limit=0
        hide-ssid=no security-profile=default disconnect-timeout=3s
        on-fail-retry-time=100ms preamble-mode=both
[admin@PPPoE-Server] interface wireless>
```

现在，配置以太网卡，添加默认 IP 地址和设置默认路由：

```
[admin@PPPoE-Server] ip address> add address=10.1.0.3/24 interface=Local
[admin@PPPoE-Server] ip address> print
```

```
Flags: X - disabled, I - invalid, D - dynamic
 #    ADDRESS              NETWORK          BROADCAST        INTERFACE
 0    10.1.0.3/24          10.1.0.0         10.1.0.255       Local
[admin@PPPoE-Server] ip address> /ip route
[admin@PPPoE-Server] ip route> add gateway=10.1.0.1
[admin@PPPoE-Server] ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf
 #       DST-ADDRESS          G GATEWAY           DISTANCE INTERFACE
 0 ADC 10.1.0.0/24                                    Local
 1 A S 0.0.0.0/0            r 10.1.0.1          1          Local
```

配置 nat 规则

```
[admin@PPPoE-Server] ip route> /ip firewall nat
[admin@PPPoE-Server]/ ip firewall nat >add action=masquerade
[admin@PPPoE-Server] /ip firewall nat > print
[admin@ PPPoE-Server] /ip firewall nat> print
Flags: X - disabled, I - invalid, D - dynamic
 0      chain=srcnat action=masquerade
```

添加 PPPoE server 到无线网卡上，设置 service-name 为 mt:

```
[admin@PPPoE-Server] interface pppoe-server server> add interface=wlan1 service-name=mt
one-session-per-host=yes disabled=no
[admin@PPPoE-Server] interface pppoe-server server> print
Flags: X - disabled
 0    service-name="mt" interface=wlan1 max-mtu=1480 max-mru=1480
      authentication=pap,chap,mschap1,mschap2 keepalive-timeout=10
      one-session-per-host=yes max-sessions=0 default-profile=default
[admin@PPPoE-Server] interface pppoe-server server>
```

配置用户地址池，并添加用户 profile

```
[admin@PPPoE-Server] ip pool> add name=pppoe ranges=192.168.88.10-192.168.88.100
[admin@PPPoE-Server] ip pool> print
 # NAME                                      RANGES
 0 pppoe                                     192.168.88.10-192.168.88.100
[admin@PPPoE-Server] ip pool> /ppp profile
[admin@PPPoE-Server] ppp profile> set default use-encryption=yes \
   local-address=10.1.0.3 remote-address=pppoe
[admin@PPPoE-Server] ppp profile> print
Flags: * - default
 0 * name="default" local-address=10.1.0.3 remote-address=pppoe
      use-compression=no use-vj-compression=no use-encryption=yes only-one=no
      change-tcp-mss=yes
```

```
1 * name="default-encryption" use-compression=default
    use-vj-compression=default use-encryption=yes only-one=default
    change-tcp-mss=default
```

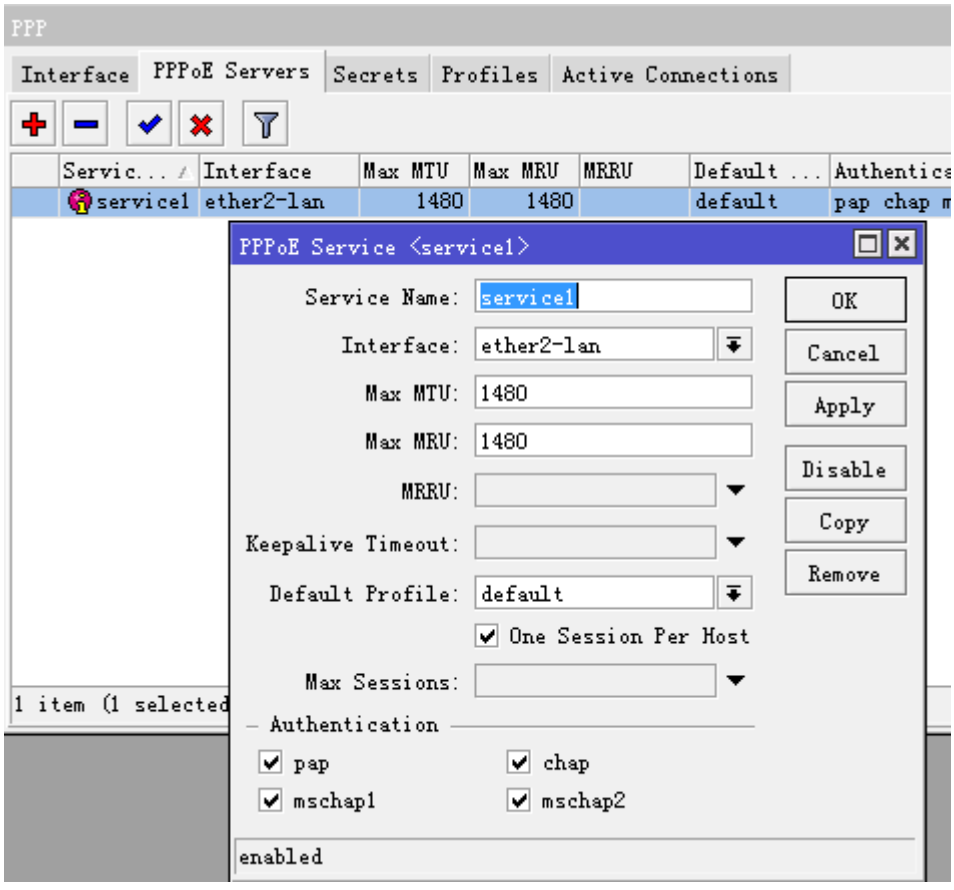
添加用户账号，并设置服务类型为 pppoe

```
[admin@PPPoE-Server] ppp profile> .. secret
[admin@PPPoE-Server] ppp secret> add name=w password=wkst service=pppoe
[admin@PPPoE-Server] ppp secret> add name=l password=ltp service=pppoe
[admin@PPPoE-Server] ppp secret> print
Flags: X - disabled
 #    NAME          SERVICE CALLER-ID PASSWORD   PROFILE             REMOTE-ADDRESS
 0    w               pppoe               wkst        default              0.0.0.0
 1    l               pppoe               ltp         default              0.0.0.0
[admin@PPPoE-Server] ppp secret>
```

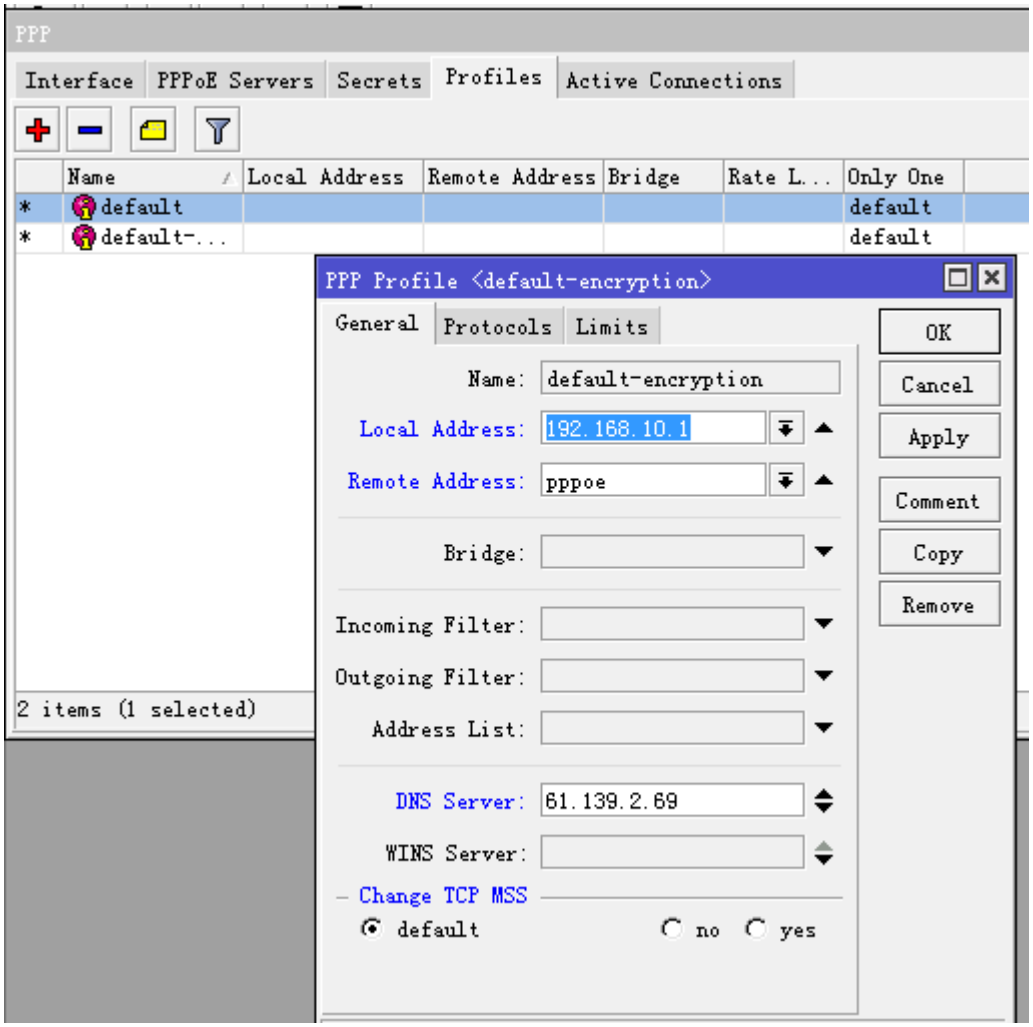
注： 在 Windows 中的 PPPoE 客户端内建加密功能，因此，如果计划不在支持比 Windows XP 老的 Windows 客户端，推荐在 **default** 规则配置把 **require-encryption** 值选择位 **yes**。在其他一些应用中，可以服务设置为接受为加密的数据。

21.6 Winbox 配置 PPPoE 服务

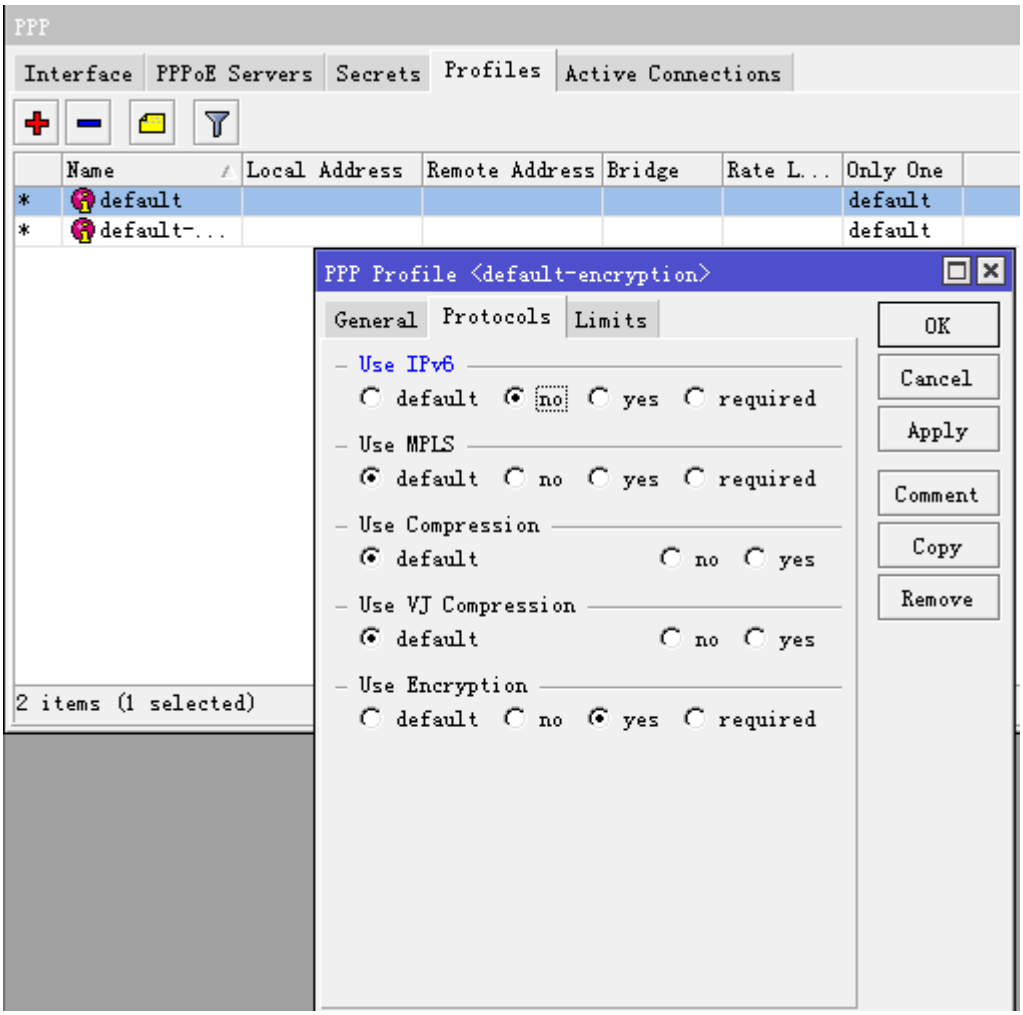
通过 Winbox 配置 PPPoE 服务器，这里我们首先通过进入 PPP 目录下的 PPPoE Server，配置 Service Name 为 MikroTik，用于 PPPoE 服务器名，并把 PPPoE 服务指向 ether2 的网卡上，其他参数如图所示：



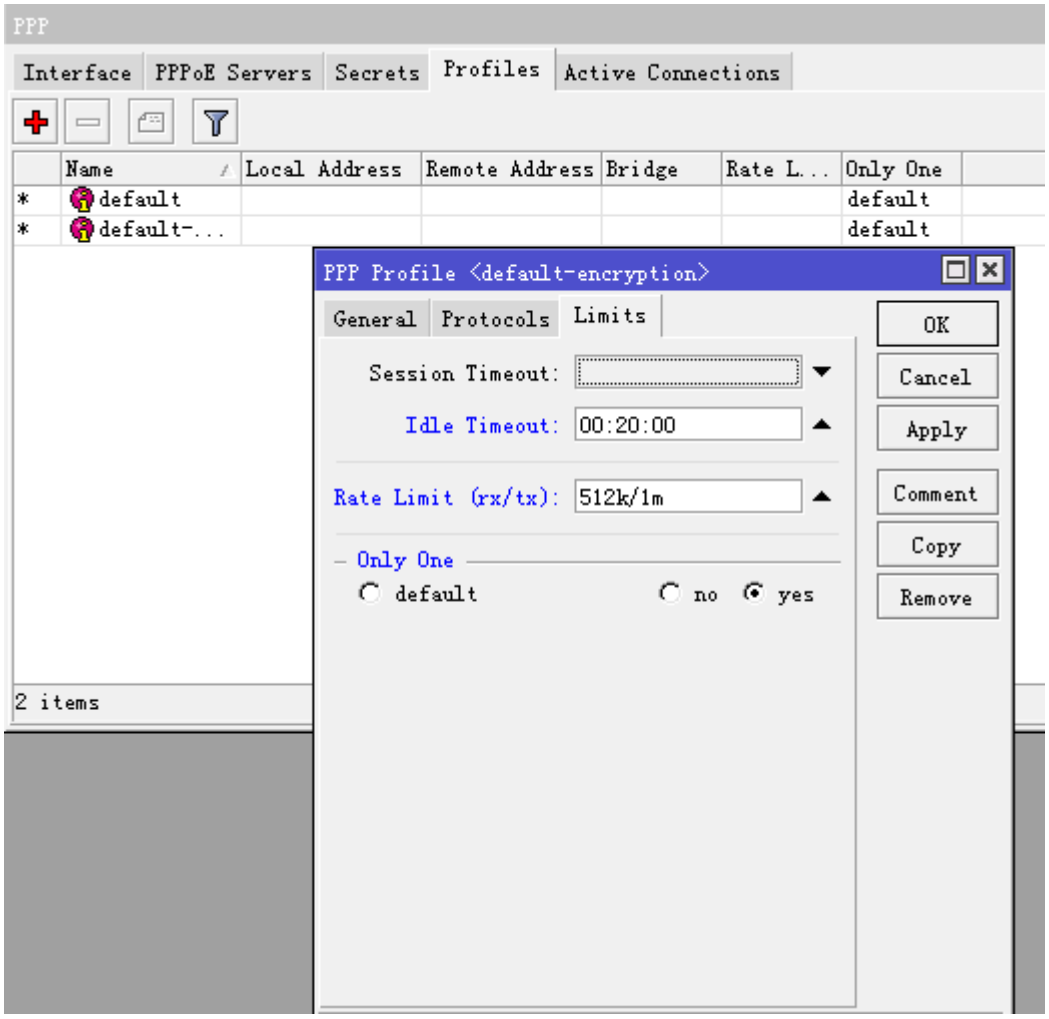
这里我们选择的是 default-encryption 的 profile 规则，所有我们需要进入 profiles 中配置该规则的参数，loacl-address 为本地路由器网关 IP，remote-address 则是远程客户端 IP 地址。这里我们设置 local-address 为 192.168.10.1，remote-address 添加在 ip pool 中设置好的地址池 pppoe，然后配置 DNS 参数，其他配置如图：



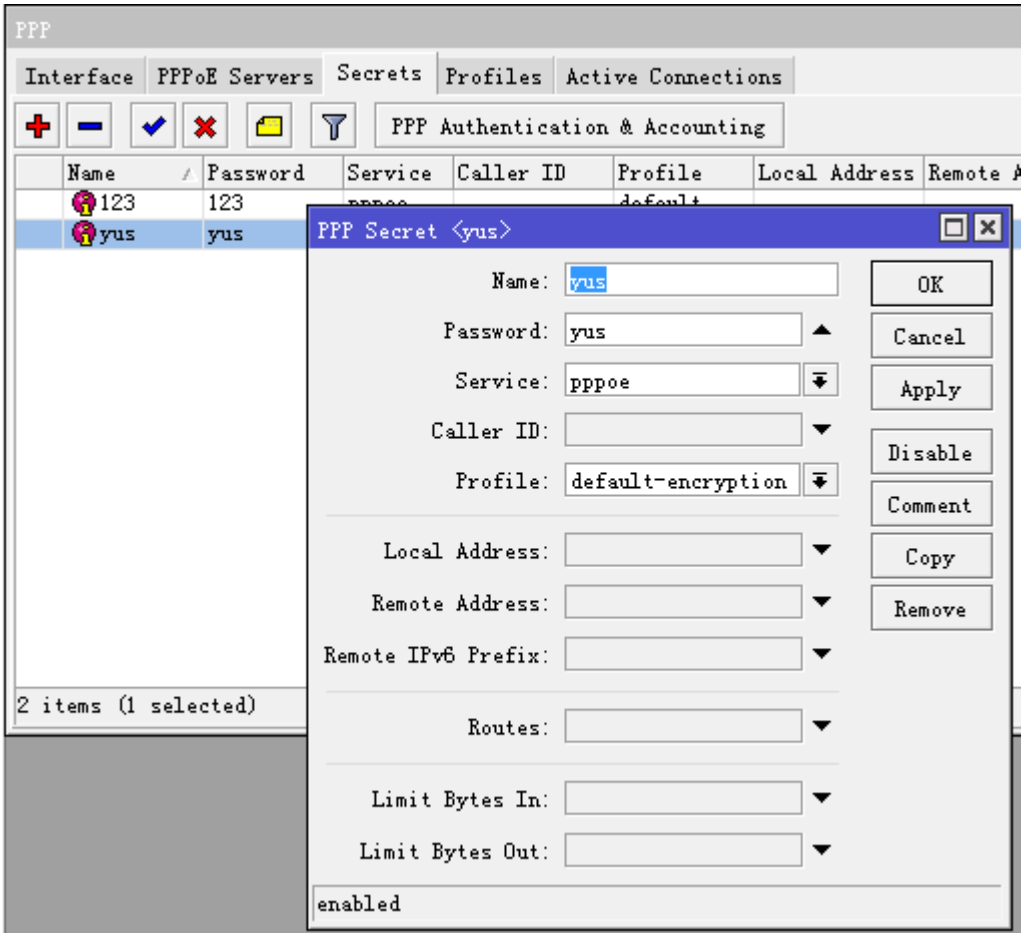
在 5.0 后增加了 IPv6 的支持，这里我们可以选择关闭 IPv6 的支持：



下面配置 Limits 参数，Idle-timeout 设置为 20 分钟，即当用户在 20 分钟内没有任何数据流量就注销该用户，每个用户带宽我们设置 512k 上行，1M 下行，only-one 参数只该 profile 下的账号只允许一个用户登陆：



这样用户的组规则配置完成，根据需要也可以增加其他的组规则到 profile 中。接下来配置每个用户信息，进入 ppp secrets 添加用户账号：

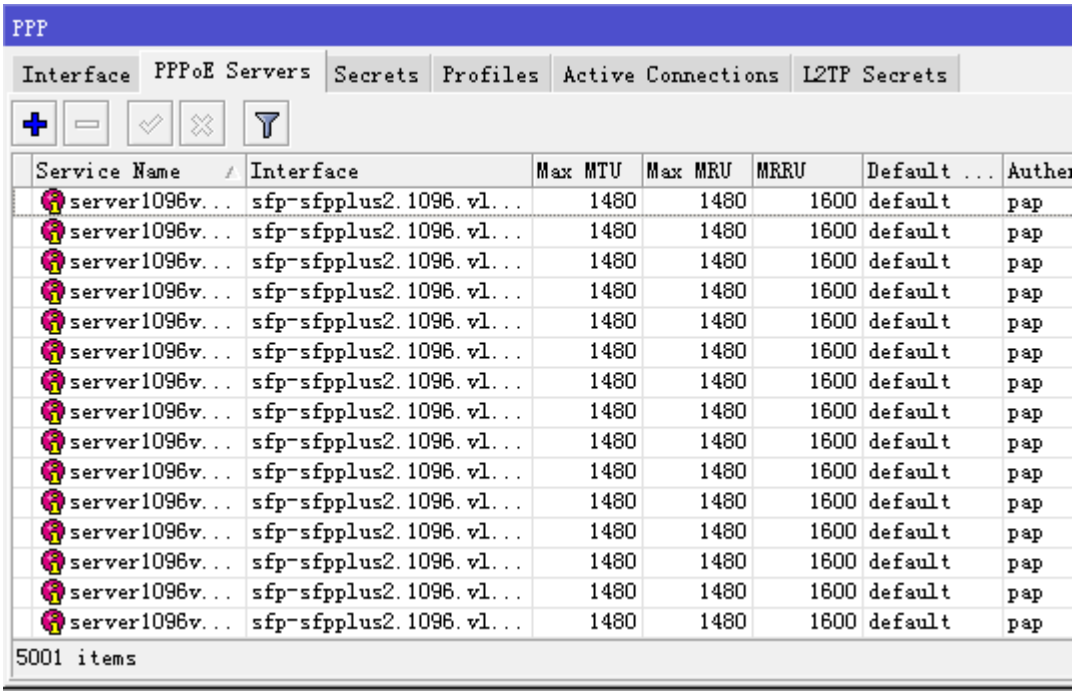


这里 Name 为用户账号名，Password 为用户密码。Profile 选择刚才设置好的 default-encryption，根据情况也可以调用其它相应的 profile 规则。配置完用户的账号和密码后，PPPoE 服务就可以启动。

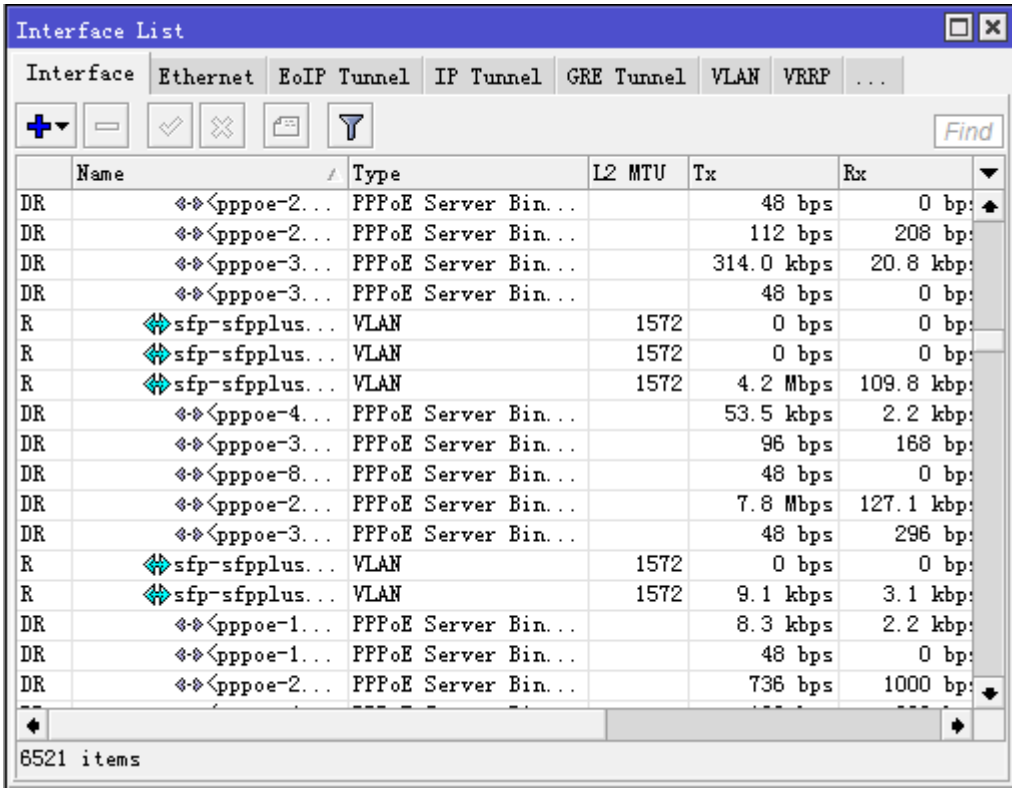
我们也可以选择让 PPPoE 认证与 RADIUS 对接，具体操作可以参加 User Manager 章节。

21.7 基于 VLAN 的 PPPoE 认证

在大型局域网络中 PPPoE 认证被大规模部署，为了保证网络的稳定，减小广播域，通过 vlan 交换机都创建多个 VLAN 隧道，并汇聚到一台汇聚交换机上，汇聚交换机通过一个 Trunk 口连接到 RouterOS 的内网口，此处配置和上一节的 trunk 配置一样，只是我们不需在每个 VLAN 上配置 IP 地址，而是创建 PPPoE 认证服务如下图，建立多个 VLAN 后，在 PPPoE-Server 中对每个 VLAN 建立一个 PPPoE 服务，下图是一台 RouterOS 设备配置 QinQ 后，创建的 5001 个 PPPoE Servers：



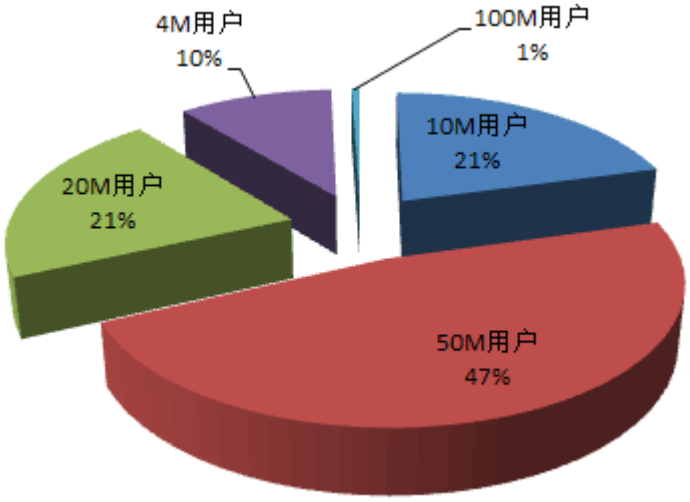
下面是在 interface 中独立 VLAN 下的 PPPoE 运行情况：



注：如果你网络启用了 PPPoE 认证，且有多级的交换机级联，最好使用具备功能 VLAN 的交换机，减小广播域，避免网络广播风暴或病毒产生造成的问题，之前发现未划分 VLAN 的二层网络，因为没有使用 vlan 隔离造成了用户在 PPPoE 认证后出现随机掉线的情况。

平台总结

以上的配置网络环境是运营商级别，用户账号带宽分布比例如下图：



在这样网络下基于三款设备，做相同配置的 PPPoE 认证，具体性能测试情况供大家参考：

	CCR1036-8G-2S+	CCR1072-1G-8S+	RouterOS x86
硬件配置	2 个 SFP+和 8 个 1G 电口	8 个 SFP+和 1 个 1G 电口	4 个 1G 电口，2 个 SFP+网卡
RouterOS	6.33.3	6.33.3	6.33.2
CPU	Tilera 1.2G 36 核心	Tilera 1G 72 核心	x86 Dell620 Intel E5-2609×2 2.4G 8 核心
登录测试	850 人同时登录完成，使用 5 分钟，CPU 无异常	-	-
QinQ 配置	QinQ：5000 条	QinQ：5000 条	QinQ：5000 条
路由协议	OSPF	OSPF	OSPF
SNMP	开启	开启，由于 VLAN 接口过多 snmp 响应延迟，甚至无响应	开启
RADIUS	账号验证	账号验证	账号验证
流控策略	通过 address-list 分类账号带宽做 PCQ	通过 address-list 分类账号带宽做 PCQ	通过 address-list 分类账号带宽做 PCQ
运用环境	PPPoE server 5000 条	PPPoE server 5000 条	PPPoE server 5000 条
	在线 571 人，CPU 平均 18%，流量 448Mb	在线 1380 人，CPU 平均 9%，流量 288Mb	在线 463 人，CPU 平均 30%，流量最高 412Mb
	在线 919 人，CPU 平均 15%，流量 285Mb	在线 1740 人，CPU 平均 40%（短时间 70%），流量 1.1Gb	
冗余电源	不支持	支持	支持
最大功耗	78w	125w	>180w
总结	转发和软件优化比较出色流量增长 CPU 变化不大（多 CPU 平均结果）	转发和软件优化比较出色流量增长 CPU 变化不大（多 CPU 平均结果）	IRQ 网卡对应与实际网卡有差别流量增长 CPU 增长较明显

以上对比是在实际环境得到，当时有大量 QinQ 配置，添加 cacti 监控时，都是在没有创建 QinQ 的情况下先添加接口，因为 5000 条 QinQ 在 cacit 获取接口 SNMP 时会卡死。还要初始化大量认证配置包括 5000 条 PPPoE 认证服务规则，整机达到了 1 万多条规则，基于 x86 服务器的 RouterOS，启动加载要比 CCR 慢，也是问题。

由于测试时间短只有一周多，而且大型运营商网络用户反馈问题，不如小宽带及时，也没有拿到一线情况，只能看到用户认证都正常，所以得到的结论也是不完整的。后续又对 RouterOS 自己开发的 CCR 系列和 x86 平台做

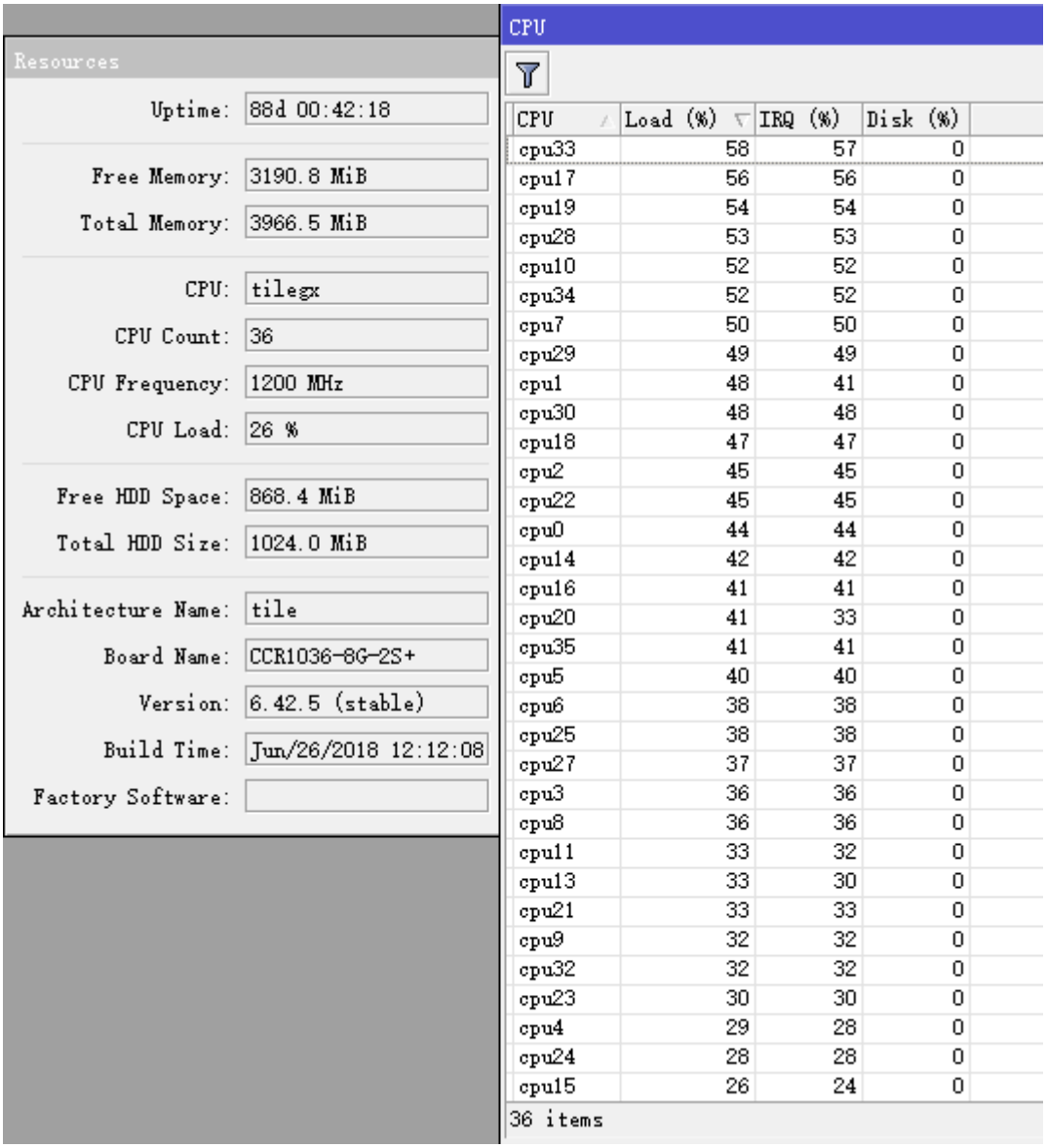
了评估，并不能说明 CCR 系列比 x86 平台占有多少优势，其实 CCR 存在的问题也比较多。（对于之前在 2018 年 MUM 提到的 CCR 更好，其实是片面的结论，对此表示抱歉）。

我再次总结下：

	CCR 平台	x86 平台
CPU	Tilera 9-72 核心 1.2G	x86 构架 2.0GHz，6 核心以上
三层转发	有 fastpath 优势	无 fastpath
处理能力	1、 在 CCR1036 上发现转发延迟大问题（ping 延迟不稳定），初步排除是多 CPU 负载不均衡导致，部分 CPU 负载 100% 2、 在做大量脚本处理时，几百用户的策略修改，个别 CPU 负载高，处理速度慢 3、 同时剔除上百 PPPoE 在线用户时，CPU 异常 90%	1、 转发延迟无异常 2、 CPU 性能高，脚本处理速度快 3、 同时剔除上百 PPPoE 在线用户，CPU 处理无异常

从以上对比，CCR 构架平台通过多 CPU（9~72 核心）频率在 1~1.2GHz，但单核性能是无法和 x86 平台媲美的，差距是非常明显。虽然有 fastpath 功能，但开启该功能受限制较多，当遇到大量数据处理时，很容易造成单个 CPU 负载 100%，会对正在转发的网络造成一定影响，因此 CCR 自身构架上存在一些弊端，要实现均衡的调度 CPU 很难，

通常我们查看到系统默认显示的 CPU 负载是平均值，当点开 Resource 下的 CPU 选项，可以看到每个 CPU 的利用率，每个 CPU 利用率并不均衡，部分应用程序还只能调用一个 CPU，如 bandwidth-test，做带宽测试只会应用一个 CPU。



Resources

Uptime:	88d 00:42:18
Free Memory:	3190.8 MiB
Total Memory:	3966.5 MiB
CPU:	tilegx
CPU Count:	36
CPU Frequency:	1200 MHz
CPU Load:	26 %
Free HDD Space:	868.4 MiB
Total HDD Size:	1024.0 MiB
Architecture Name:	tile
Board Name:	CCR1036-8G-2S+
Version:	6.42.5 (stable)
Build Time:	Jun/26/2018 12:12:08
Factory Software:	

CPU

CPU	Load (%)	IRQ (%)	Disk (%)
cpu33	58	57	0
cpu17	56	56	0
cpu19	54	54	0
cpu28	53	53	0
cpu10	52	52	0
cpu34	52	52	0
cpu7	50	50	0
cpu29	49	49	0
cpu1	48	41	0
cpu30	48	48	0
cpu18	47	47	0
cpu2	45	45	0
cpu22	45	45	0
cpu0	44	44	0
cpu14	42	42	0
cpu16	41	41	0
cpu20	41	33	0
cpu35	41	41	0
cpu5	40	40	0
cpu6	38	38	0
cpu25	38	38	0
cpu27	37	37	0
cpu3	36	36	0
cpu8	36	36	0
cpu11	33	32	0
cpu13	33	30	0
cpu21	33	33	0
cpu9	32	32	0
cpu32	32	32	0
cpu23	30	30	0
cpu4	29	28	0
cpu24	28	28	0
cpu15	26	24	0

36 items

当前 x86 平台的多 CPU 发展也非常快，开启超线程后也能到 72 核心，因此，如果基于大型的 PPPoE 认证，还是建议选择 x86 服务器平台，但 x86 平台 MikroTik 基本已放弃更新，转向 VM 平台，因此以上内容仅供大家参考。

QinQ 添加脚本

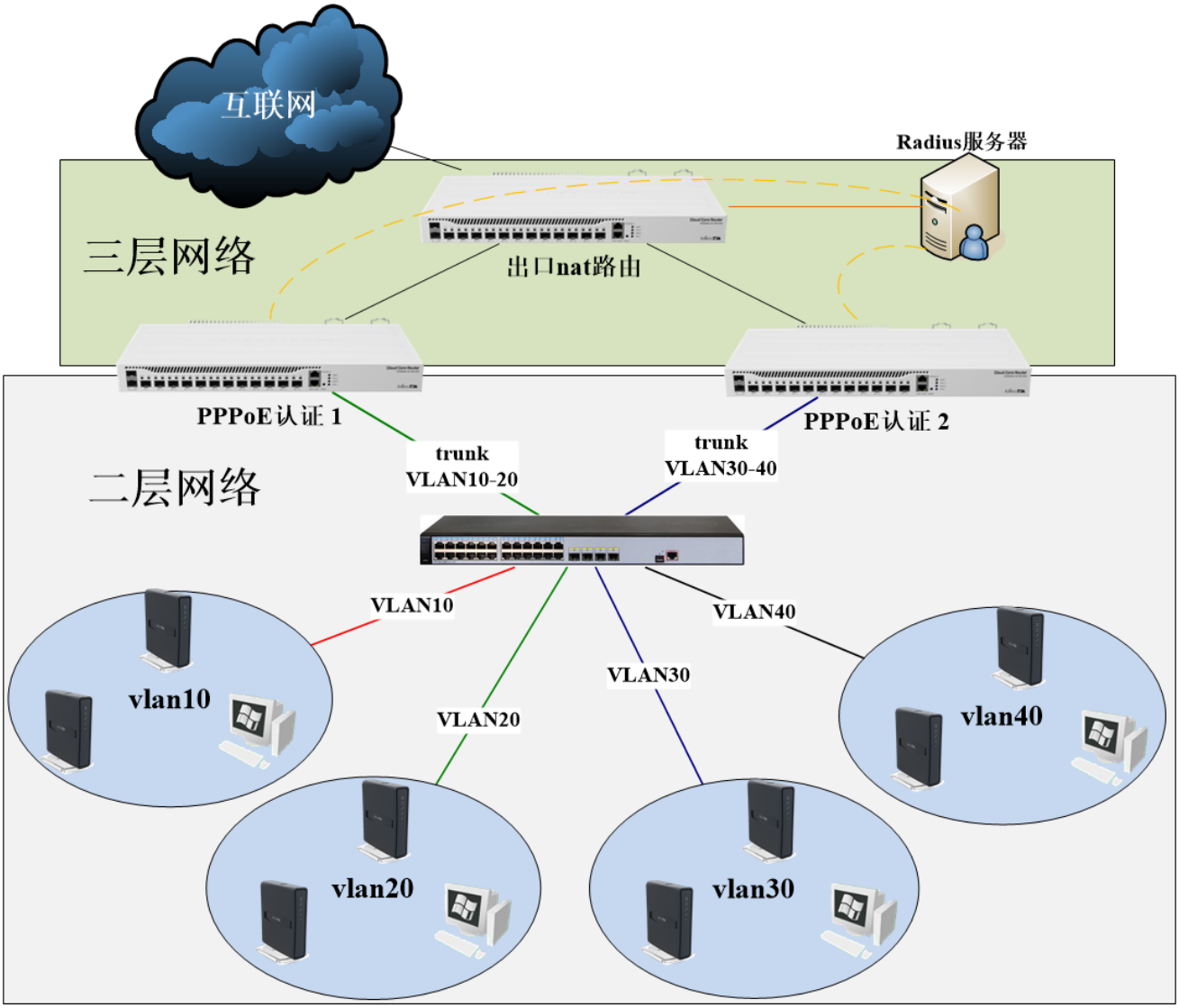
由于较大的网络规划，涉及到 QinQ 配置，创建时需配置大量的 vlan 规则，因此这个配置操作手动非常繁琐，下面提供一个添加 QinQ vlan 的脚本，该脚本包括外层和内层 VLAN 的脚本添加，外层 VLAN 是 2-1000，内层 VLAN 是 2-100

```
:for m from=2 to=1000 do={
    /interface vlan add name=("vlan" . $m) vlan-id=$i interface=sfp-sfpplus2 disabled=no
    :for i from=2 to=100 do={
        /interface vlan add name=("sfp-sfpplus2" . "." . $m . "." . "vlan" . $i) interface=("vlan" . $m) vlan-id=$i
disabled=no
  }
}
```

21.8 高并发 PPPoE 认证服务集群

运营商网络几乎都采用的 PPPoE 方式的拨号认证，PPPoE 认证由于是基于 OSI 七层参考模型第二层运行，所以不会受 IP 层影响，这样可以避免常见的 ARP 病毒攻击。PPPoE 是让每一个用户在二层 MAC 地址间建立一个虚拟的隧道，即保证了数据的安全，又保证了稳定，与通过 IP 方式的 hotspot Web 认证要更加稳定可靠。使用 Windows 操作系统都自带了 PPPoE 拨号功能，即用户不需要太复杂的操作，就可以建立一个 PPPoE 虚拟拨号连接，当然大多数家庭会使用路由器完成拨号上网。

高并发这里只是针对 RouterOS 自身的性能来说，与华为和思科这些大厂的 BAS 而言还是有很大的差异，MikroTik 可以选择 CCR1036 或者更高的 CCR1072。RouterOS 在配合高性能的 x86 服务器（如至强处理器一般采用大于 36 个 CPU 核心、频率 3GHz 以上）的认证用户数量的极限是多少，但预计在 2000-3000，流量 4-5Gbps（不启用 nat），这个参数仅供参考。下图，通过的拓扑简单的描述了一个中型 PPPoE 认证网络拓扑：



在 500 人左右的 PPPoE 认证网络，通常会把出口路由、nat 转换和 PPPoE 认证都做在一台 RouterOS 上，但对于人数更多的时候，出口路由和 nat 功能需要和 PPPoE 认证分离，组建一个分布式的网络，通各个功能分离进行组网，上图我通过三层网络和二层网络两个区块，来展现网络拓扑。我个人认为如果基于运营商的 PPPoE 认证网络，都应该把出口接入和 PPPoE 认证分离为两台设备，特别是在多线接入的情况下，有助于网络的维护和扩展。

二层网络：

- 通过 vlan 划分各个用户区域，分别用了 vlan10,20,30,40，接入到交换机，这样可以通过 vlan 隔离 4 个区域，避免广播域过大，如果出现故障不会影响另外一个区域。
- 交换机通过 trunk 口，将指定 vlan 透传给 PPPoE 认证服务 1 和 2，PPPoE 认证 1 负责 vlan10 和 20，PPPoE2 负责 vlan30 和 40，这样可以将网络用户均分

三层网络：

- 两台 PPPoE 认证，用户接入侧是二层网络，上联出口 nat 路由是三层网络，是二层和三层网络的一个交叉点
- Radius 服务器，用户两台 PPPoE 认证的用户账号验证，不建议使用 RouterOS 本地账号，因为无法做到账号完全同步。
- 出口 nat 路由主要负责各种出口的接入，如多线路由负载均衡和策略路由，然后完成内部私网 IP 地址到互联网的 nat 转换。

网络规划参数：

设备	类型	IP 地址	网关	接口
出口 nat 路由	出口 IP	172.16.0.2/27	172.16.0.1	ether1
	PPPoE 认证 1 互联地址	10.10.1.1/30		ether2
	PPPoE 认证 1 互联地址	10.10.2.1/30		ether3
	radius 互联	10.10.4.1/28		ether4
PPPoE 认证 1	出口 nat 路由互联 IP	10.10.1.2/30	10.10.1.1	ether1
	交换机接入以及管理 IP	10.10.3.1/28		ether2.vlan2
	用户地址池	10.20.0.0/16		ether2.vlan10,vlan20
PPPoE 认证 2	出口 nat 路由互联 IP	10.10.2.2/30	10.10.2.1	ether1
	交换机接入以及管理 IP	10.10.3.2/28		ether2.vlan2
	用户地址池	10.30.0.0/16		ether2.vlan30,vlan40
radius 服务器	对接 IP	10.10.4.2/28	10.10.4.1	eth0
华为 5700	对接 IP	10.10.3.3/28	10.10.3.1	xg0/0/1 vlan2
	PPPoE 认证 1 trunk			xg0/0/1 vlan10,20
	PPPoE 认证 2 trunk			xg0/0/2 vlan10,20
	用户接入 vlan10 access			g0/0/1
	用户接入 vlan20 access			g0/0/2
	用户接入 vlan30 access			g0/0/3
	用户接入 vlan40 access			g0/0/4

配置出口 nat 路由器

配置出口和互联 IP 地址

```
[admin@nat] >/ip address
[admin@nat] /ip address > add address=172.16.0.2/27 interface=ether1
[admin@nat] /ip address > add address=10.10.1.1/30   interface=ether2
[admin@nat] /ip address > add address=10.10.2.1/30   interface=ether3
```

配置 Radius 服务器互联 IP

```
[admin@nat] /ip address > add address=10.10.4.1/28   interface=ether4
```

配置默认路由和 PPPoE 用户 IP 段静态路由

```
[admin@nat] /ip address >/ip route
[admin@nat] /ip route > add gateway=172.16.0.1
[admin@nat] /ip route > add dst-address=10.20.0.0/16 gateway=10.10.1.2
[admin@nat] /ip route > add dst-address=10.30.0.0/16 gateway=10.10.2.2
```

配置 nat 规则

```
 [admin@nat] /ip route >/ip firewall nat
 [admin@nat] /ip firewall nat >add chain=srcnat out-interface=ether1 action=masquerade
```

配置 PPPoE 认证 1

创建 ether2 的 VLAN2，10，20 的接口

```
[admin@ PPPoE1] /ip address >/interface vlan
[admin@ PPPoE1] /interface vlan > add interface=ether2 name=vlan10 vlan-id=10
[admin@ PPPoE1] /interface vlan > add interface=ether2 name=vlan20 vlan-id=20
[admin@ PPPoE1] /interface vlan > add interface=ether2 name=vlan2 vlan-id=2
```

配置 IP 地址

```
[admin@PPPoE1] >/ip address
[admin@PPPoE1] /ip address > add address=10.10.1.2/30 interface=ether1
[admin@PPPoE1] /ip address > add address=10.10.3.1/28 interface=vlan2
```

配置默认网关

```
[admin@ PPPoE1] /ip address >/ip route
[admin@ PPPoE1] /ip route > add gateway=10.10.1.1
```

配置用户地址池，并添加用户 profile

```
[admin@PPPoE1] /ip pool> add name=pool1 ranges=10.20.0.0/16
[admin@PPPoE1] /ip pool>/ppp profile
[admin@PPPoE1] /ppp profile>add name=pppoe local-address=10.10.1.2 remote-address=pool1
dns-server=114.114.114.114
```

创建用户本地测试账号，并设置策略组为 pppoe

```
[admin@PPPoE1] ppp profile> .. secret
[admin@PPPoE1] ppp secret> add name=yus password=yus
```

配置 vlan10，vlan20 的 PPPoE Server

```
[admin@PPPoE1]/interface pppoe-server server
[admin@PPPoE1] interface pppoe-server server> add interface=vlan10 one-session-per-host=yes
default-profile=pppoe disabled=no
[admin@PPPoE1] interface pppoe-server server> add interface=vlan20 one-session-per-host=yes
  default-profile=pppoe disabled=no
```

连接 radius 服务器，所有用户账号的配置管理通过 radius 服务器完成

```
[admin@PPPoE1] > radius
[admin@PPPoE1] /radius> add address=10.10.4.2 secret=yus service=ppp
```

配置 PPPoE 认证 2

创建 ether2 的 VLAN2，30，40 的接口

```
[admin@ PPPoE2] /ip address >/interface vlan
[admin@ PPPoE2] /interface vlan > add interface=ether2 name=vlan30 vlan-id=30
[admin@ PPPoE2] /interface vlan > add interface=ether2 name=vlan40 vlan-id=40
[admin@ PPPoE2] /interface vlan > add interface=ether2 name=vlan2 vlan-id=2
```

配置 IP 地址

```
[admin@PPPoE2] >/ip address
[admin@PPPoE2] /ip address > add address=10.10.2.2/30 interface=ether1
[admin@PPPoE2] /ip address > add address=10.10.3.2/28 interface=vlan2
```

配置默认网关

```
[admin@ PPPoE2] /ip address >/ip route
[admin@ PPPoE2] /ip route > add gateway=10.10.2.1
```

配置用户地址池，并添加用户 profile

```
[admin@PPPoE2] /ip pool> add name=pool1 ranges=10.30.0.0/16
[admin@PPPoE2] /ip pool>/ppp profile
[admin@PPPoE2] /ppp profile>add name=pppoe local-address=10.10.2.2 remote-address=pool1
dns-server=114.114.114.114
```

创建用户本地测试账号，并设置策略组为 pppoe

```
[admin@PPPoE2] ppp profile> .. secret
[admin@PPPoE2] ppp secret> add name=yus password=yus
```

配置 vlan10，vlan20 的 PPPoE Server

```
[admin@PPPoE2]/interface pppoe-server server
[admin@PPPoE2] interface pppoe-server server> add interface=vlan30 one-session-per-host=yes
default-profile=pppoe disabled=no
```

```
[admin@PPPoE2] interface pppoe-server server> add interface=vlan40 one-session-per-host=yes
  default-profile=pppoe disabled=no
```

连接 radius 服务器，所有用户账号的配置管理通过 radius 服务器完成

```
[admin@PPPoE2] > radius
[admin@PPPoE2] /radius> add address=10.10.4.2 secret=yus service=ppp
```

华为 5700 交换机配置

创建相应的 VLAN

```
[S5700]vlan batch 2 10 20 30 40
```

配置 vlan2 管理 IP

```
[S5700]interface vlan 2
[S5700-Vlanif2]ip address 10.10.3.3 28
[S5700-Vlanif2]quit
```

配置默认路由，用 PPPoE 认证 1 作为默认网关

```
[S5700]ip route-static 0.0.0.0 0.0.0.0 10.10.3.1
```

配置万兆 1 口为 trunk，连接到 pppoe 认证 1

```
[S5700]interface xGigabitEthernet0/0/1
[S5700-interface xGigabitEthernet0/0/1] description to pppoe1
[S5700-interface xGigabitEthernet0/0/1] port link-type trunk
[S5700-interface xGigabitEthernet0/0/1] undo port trunk allow-pass vlan 1
[S5700-interface xGigabitEthernet0/0/1] port trunk allow-pass vlan 2 10 20
```

配置万兆 2 口为 trunk，连接到 pppoe 认证 2

```
[S5700-interface xGigabitEthernet0/0/1] interface xGigabitEthernet0/0/2
[S5700-interface xGigabitEthernet0/0/2] description to pppoe2
[S5700-interface xGigabitEthernet0/0/2] port link-type trunk
[S5700-interface xGigabitEthernet0/0/2] undo port trunk allow-pass vlan 1
[S5700-interface xGigabitEthernet0/0/2] port trunk allow-pass vlan 2 30 40
```

配置客户接入 VLAN10 到千兆 1 口

```
[S5700-interface xGigabitEthernet0/0/2] interface GigabitEthernet0/0/1
[S5700-interface GigabitEthernet0/0/1] description vlan10
[S5700-interface GigabitEthernet0/0/1] port link-type access
[S5700-interface GigabitEthernet0/0/1] port default vlan 10
```

配置客户接入 VLAN20 到千兆 2 口

```
[S5700-interface GigabitEthernet0/0/1] interface GigabitEthernet0/0/2
[S5700-interface GigabitEthernet0/0/2] description vlan20
 [S5700-interface GigabitEthernet0/0/2] port link-type access
[S5700-interface GigabitEthernet0/0/2] port default vlan 20
```

配置客户接入 VLAN30 到千兆 3 口

```
[S5700-interface GigabitEthernet0/0/2] interface GigabitEthernet0/0/3
[S5700-interface GigabitEthernet0/0/3] description vlan30
[S5700-interface GigabitEthernet0/0/3] port link-type access
[S5700-interface GigabitEthernet0/0/3] port default vlan 30
```

配置客户接入 VLAN40 到千兆 4 口

```
[S5700-interface GigabitEthernet0/0/3] interface GigabitEthernet0/0/4
[S5700-interface GigabitEthernet0/0/4] description vlan40
[S5700-interface GigabitEthernet0/0/4] port link-type access
[S5700-interface GigabitEthernet0/0/4] port default vlan 40
```

PPPoE 认证负载均衡的冗余配置

之前的配置是将两台 PPPoE 认证独立接入不同 VLAN 的用户，为提升 PPPoE 认证网络的冗余能力，可以将所有用户 VLAN 都同时透传给两台 PPPoE 认证用户， 用户会相对均衡的分配到 2 台 PPPoE 认证上，用户账号通过 radius 服务器验证

配置万兆 1 口 trunk vlan 10,20,30,40，连接到 pppoe 认证 1

```
[S5700]interface xGigabitEthernet0/0/1
[S5700-interface xGigabitEthernet0/0/1] description to pppoe1
[S5700-interface xGigabitEthernet0/0/1] port link-type trunk
[S5700-interface xGigabitEthernet0/0/1] undo port trunk allow-pass vlan 1
[S5700-interface xGigabitEthernet0/0/1] port trunk allow-pass vlan 2 10 20 30 40
```

配置万兆 2 口 trunk vlan 10,20,30,40，连接到 pppoe 认证 2

```
[S5700-interface xGigabitEthernet0/0/1] interface xGigabitEthernet0/0/2
[S5700-interface xGigabitEthernet0/0/2] description to pppoe2
[S5700-interface xGigabitEthernet0/0/2] port link-type trunk
[S5700-interface xGigabitEthernet0/0/2] undo port trunk allow-pass vlan 1
[S5700-interface xGigabitEthernet0/0/2] port trunk allow-pass vlan 2 30 40 10 20
```

在 PPPoE 认证 1 和 PPPoE 认证 2，同时配置 vlan10，vlan20，VLAN30，VLAN40 的 PPPoE Server

```
[admin@PPPoE1/2] interface pppoe-server server> add interface=vlan10
one-session-per-host=yes default-profile=pppoe disabled=no
[admin@PPPoE1/2] interface pppoe-server server> add interface=vlan20
one-session-per-host=yes default-profile=pppoe disabled=no
```

```
[admin@PPPoE1/2] interface pppoe-server server> add interface=vlan30
one-session-per-host=yes default-profile=pppoe disabled=no
[admin@PPPoE1/2] interface pppoe-server server> add interface=vlan40
  one-session-per-host=yes default-profile=pppoe disabled=no
```

注：关于 VLAN 的配置和事例请见第十五章 VLAN 介绍

故障分析

- **我能够连接到服务器，Ping 也能完全通过，但我仍然不能打开 web 页面？**

  确定你在路由器上指定了正确的 DNS 服务器（在**/ip dns** 或在**/ppp profile** 中的 **dns-server** 参数）

- **我能使 PPPoE 连接小点的数据报（例如 pings）**

  你需要改变所以经过 PPPoE 连接的 **mss** 数据报为 1440：

```
[admin@MT] interface pppoe-server server> set 0 max-mtu=1440 max-mru=1440
[admin@MT] interface pppoe-server server> print
Flags: X - disabled
 0   service-name="mt" interface=wlan1 max-mtu=1440 max-mru=1440
     authentication=pap,chap,mschap1,mschap2 keepalive-timeout=10
     one-session-per-host=yes max-sessions=0 default-profile=default
[admin@MT] interface pppoe-server server>
```

- **我想要记录连接建立的日志**

  在**/system logging facility** 中配置日志信息并启用 ppp 日志类型

提示：关于大型网络的 PPPoE 服务器配置和运行情况，请参考后面 21.8 章节，包括对三种硬件型号的 PPPoE 服务的对比测试。

第二十二章 PPTP

PPTP 是一个使用 PPP 传输 IP 通信的安全隧道，它通过将数据封装到 PPP 协议中，并基于 IP 网络建立虚拟链路的隧道。PPTP 结合了 PPP 和 MPPE (Microsoft Point to Point Encryption) 建立加密连接。一款非常受欢迎的隧道协议，可以建立在路由器与路由器，或路由器与终端设备，PPTP 客户被几乎所有操作系统支持，包括 Windows、Android 和 iOS 等，PPTP 隧道大致能实现以下的功能：

- 互联网常见的安全路由器/客户端-路由器隧道
- 连接（桥接）本地企业网或 LAN（当使用了 EoIP 时）
- 对移动或远程客户远程访问企业网/公司的 LAN（参见 Windows 的 PPTP 设置以获取更多信息）

Multilink PPP (MP)是为了支持 MRRU（可以传输完整的 1500byte 封装的数据包或更大的数据包），以及基于 PPP 协议的桥接（Bridge Control Protocol (BCP)，允许在 PPP 连接中建立桥接传输以太网帧），BCP 可以完全可以在没有 EoIP 下建立二层透传，在创出 PPP 的二层隧道时，如果只有 PPTP 接口需要为 bridge 分配 MAC 地址，或者将一个以太网接口加入到 bridge 中，完成 MAC 地址添加，因为 PPP 协议没有 MAC 物理地址概念。

PPTP 协议为每个连接提供 PPP 验证和记账功能，在 RouterOS 中支持连接 RADIUS 或本地验证和记账功能。

当前 RouterOS 支持 MPPE 128bit RC4 加密，PPTP 传输使用 TCP 1723 端口和 IP 协议中的 GRE（Generic Routing Encapsulation, IP 协议 ID 为 47），PPTP 可以与大多数防火墙和路由器一起使用，方法是允许 TCP 端口 1723 和 IP 协议 47 的数据通过防火墙或路由器

规格

功能包要求: ***ppp***
等级要求: *Level1 (限制 1 个在线)* , *Level3 (限制 1 在线)* , *Level4 (限制 200 在线)*，*Level5（无限制）*
操作路径: ***/interface pptp-server***, ***/interface pptp-client***
标准与技术: PPTP (RFC 2637)

快速设置向导

在两个 IP 地址为 **10.5.8.104**（PPTP 服务器）及 **10.1.0.172** （PPTP 客户）的 MikroTik 路由器之间创建一个 PPTP 隧道，参考下面的步骤：

- PPTP 服务器上的设置：

1. 添加一个用户：

```
[admin@PPTP-Server] ppp secret> add name=jack password=pass local-address=10.0.0.1
remote-address=10.0.0.2
```

2. 启用 PPTP 服务器：

```
[admin@PPTP-Server] interface pptp-server server> set enabled=yes
```

- PPTP 客户的设置：

1. 添加 PPTP 客户：

```
[admin@PPTP-Client] interface pptp-client> add user=jack password=pass
connect-to=10.5.8.104 disabled=no
```

22.1 PPTP 客户端

操作路径：***/interface pptp-client***

属性描述

属性	描述
add-default-route (*yes* | *no*; 默认: **no**)	是否为该 PPTP 隧道添加默认路由
allow (*mschap2* | *mschap1* | *chap* | *pap*; 默认: **mschap2, mschap1, chap, pap**)	选择密码验证模式
connect-to (*IP*; 默认:)	连接远端 PPTP 服务器 IP 地址或域名（域名支持在 v6.3 后支持）
default-route-distance (*byte [0..255]*; 默认: **1**)	当 add-default-route 被启用，设置动态添加路由距离值，默认为 1
dial-on-demand **(***yes|no***; 默认: no)**	连接 AC（PPPoE Server）只有在发出流量产生时进行。如果选择该属性，此时路由器会想路由表添加一条 10.112.112.0/24 路由，是否建立连接。
interface **(***字符***; 默认:)**	用于连接 PPPoE Server 的接口名称
keepalive-timeout **(整型; 默认:60)**	设置 keepalive 超时时间，单位秒。
max-mru **(***整型***; 默认: 1460)**	最大接收单元
max-mtu (*整型*; 默认: 1460)	最大传输单元
mrru (*整型: 512..65535 | disabled*; 默认: disabled)	在链路上能接收的最大数据报长度，如果一个数据报大小超过隧道的 MTU，将会分割到多个数据报中，允许全尺寸 IP 或以太网数据报发送到隧道中
name (*字符*; 默认: pppoe-out[i])	PPPoE 接口名称，在没有指定名称的情况下，由 RouterOS 自动生成。
password (*字符*; 默认:)	用户认证密码
profile (*字符*; 默认: default)	连接后生成的策略配置，在 /ppp profiles 定义
service-name (*字符*; 默认: "")	指定服务名，也可以为空，即可以连接任何 PPPoE Server
use-peer-dns (*yes|no*; 默认: no)	是否获取 PPPoE Server 分配的 DNS
user (*字符*; 默认: "")	用户认证账号

使用用户名为 **john** 密码为 **john**，设置 PPTP 名为 **test2** 的客户连接到 **10.1.1.12**PPTP 服务器并使用它作为默认网关：

```
[admin@MikroTik] interface pptp-client> add name=test2 connect-to=10.1.1.12 \
```

```
\... user=john add-default-route=yes password=john
[admin@MikroTik] interface pptp-client> print
Flags: X - disabled, R - running
 0 X  name="test2" mtu=1460 mru=1460 connect-to=10.1.1.12 user="john"
      password="john" profile=default add-default-route=yes

[admin@MikroTik] interface pptp-client> enable 0
```

通过 monitor 命令查看一个已建立连接状态：

```
[admin@MikroTik] interface pptp-client> monitor test2
     uptime: 4h35s
   encoding: MPPE 128 bit, stateless
     status: Connected
[admin@MikroTik] interface pptp-client>
```

22.2 PPTP 服务器

操作路径: ***/interface pptp-server server***

PPTP 服务器为每个连接的 PPTP 客户创建了一个动态的接口。PPTP 连接依靠你所有的证书登记从客户计数。Level1 许可允许一个 PPTP 客户，Level3 或 Level4 许可最多允许 200 客户，Level5 或 Level6 许可没有 PPTP 客户限制。

PPTP 服务器，用于配置登录用户的配置信息，每个用户连接到 PPTP 服务器都会创建一个新的接口，下面有两种接口存在于 PPTP 服务器中：

- **静态接口**-由管理员手动添加，静态接口常用于需要将该接口引用到指定配置（在 firewall rule 规则、ip route 指定路由网关或其他配置），例如将一个来至 test 账号的 PPTP 接口设置为一个路由网关，需要指定 gateway=pptp-test，在默认是动态接口状态下，连接中断后，gateway 参数将丢失，需要配置静态接口永久存在。
- **动态接口**-即 PPTP 客户端连接成功后，默认会自动添加到 interface 接口列表中，前缀为”D”

属性描述

属性	描述
authentication (*mschap2* | *mschap1* | *chap* | *pap*; 默认:**"mschap2, mschap1, chap, pap"**)	选择验证协议
default-profile (*字 符* ; 默认: **"default"**)	选择 PPTP Server 的默认策略，对应选择 User profile 策略条目，默认为里的 Default 策略
enabled (*yes* | *no*; 默认: **no**)	是否启用 PPTP 服务
keepalive-timeout (*时 间* ; 默认: **"10"**)	定义在开始发送 keepalive 数据报的时钟周期，如果客户端没有流量或在 keepalive 时钟周期内响应，将宣告该客户端中断连接

max-mru (*整型*; 默认: **"1480"**)	最大接受单元，由于隧道协议原因，将原有 1500 以太网封包减少 20。
max-mtu (默认; 默认: **"1480"**)	最大传输单元，由于隧道协议原因，将原有 1500 以太网封包减少 20。
mrru (*整型: 512..65535 | disabled*; 默认:**"disabled"**)	如果一个数据报大于隧道的 MTU，在该连接上能接收到最大数据报长度，通过分割数据报长度，允许全尺寸的 IP 或以太网包发到该隧道中。

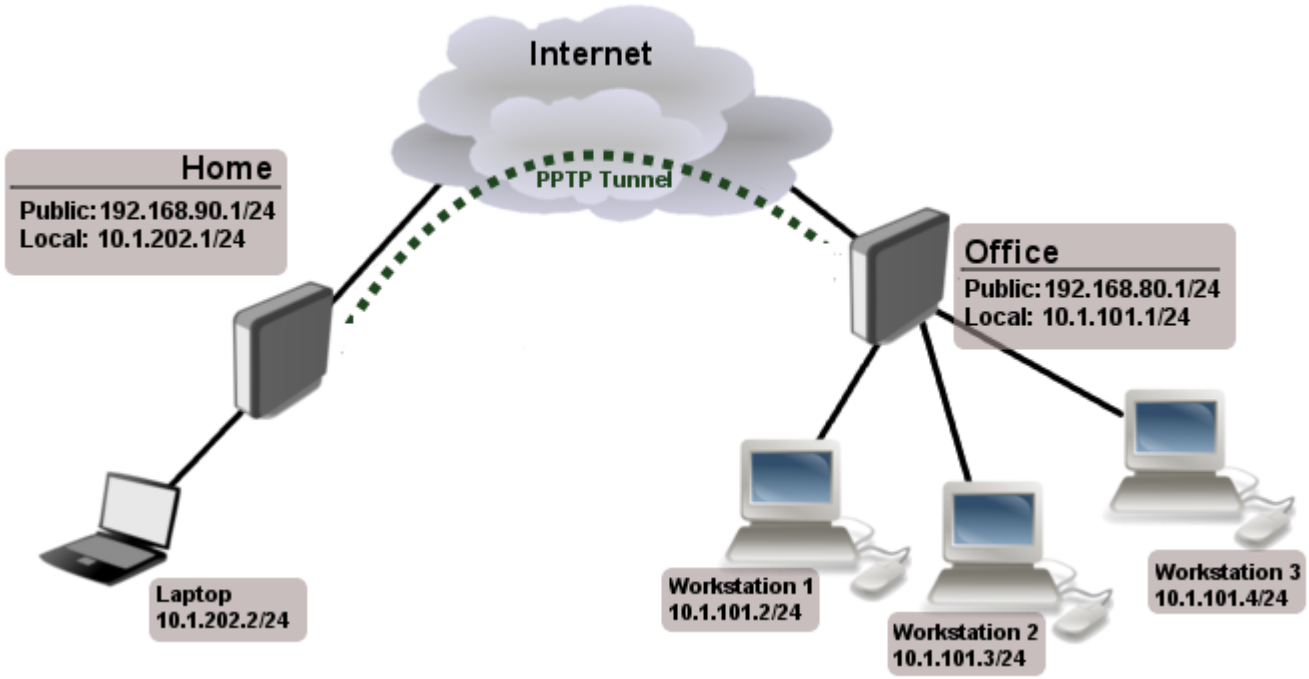
启用 PPTP 服务器：

```
[admin@MikroTik] interface pptp-server server> set enabled=yes
[admin@MikroTik] interface pptp-server server> print
            enabled: yes
                mtu: 1460
                mru: 1460
     authentication: mschap2,mschap1
   keepalive-timeout: 30
     default-profile: default
[admin@MikroTik] interface pptp-server server>
```

22.3 PPTP 应用事例

企业与家庭办公网络 PPTP 隧道配置事例

以下是一个使用互联网上的加密 PPTP 隧道实现家庭办公与企业域网连接的事例（该事例也可通用于 L2TP、SSTP 和 OVPN 等协议），通过下面的事例完成两台 RouterOS 之间的 VPN 隧道互联：



家庭办公路由器和企业路由器的公网连接都设置为 ether1,内网连接都使用 ether2，两个网络通过 PPTP 隧道连接路由连接，不在一个广播域，如果要实现相同广播域，需要使用 BCP 协议加上 bridge，将 PPTP 隧道和内网网卡桥接在一起。

在这个例子中有两个不同地区办公室的路由器，需要让两个办公局域网的主机之间实现互访：

- **[Home]**

Ether1 - 192.168.90.1/24
Ether2 - 10.1.202.1 /24

- **[Office]**

Ether1 - 192.168.80.1/24
Ether2 - 10.1.101.1/24

将 Office 路由器配置为 PPTP 服务器，每个路由器连接到当地的 ISP，任何一个路由器可以通过互联网访问到对端的路由器。

Office 配置

配置 IP 地址

```
[admin@Office] ip address >add address=192.168.90.1/24 interface=ether1
[admin@Office] ip address >add address=10.1.101.1/24 interface=ether2
```

添加默认网关，并设置 nat 转换

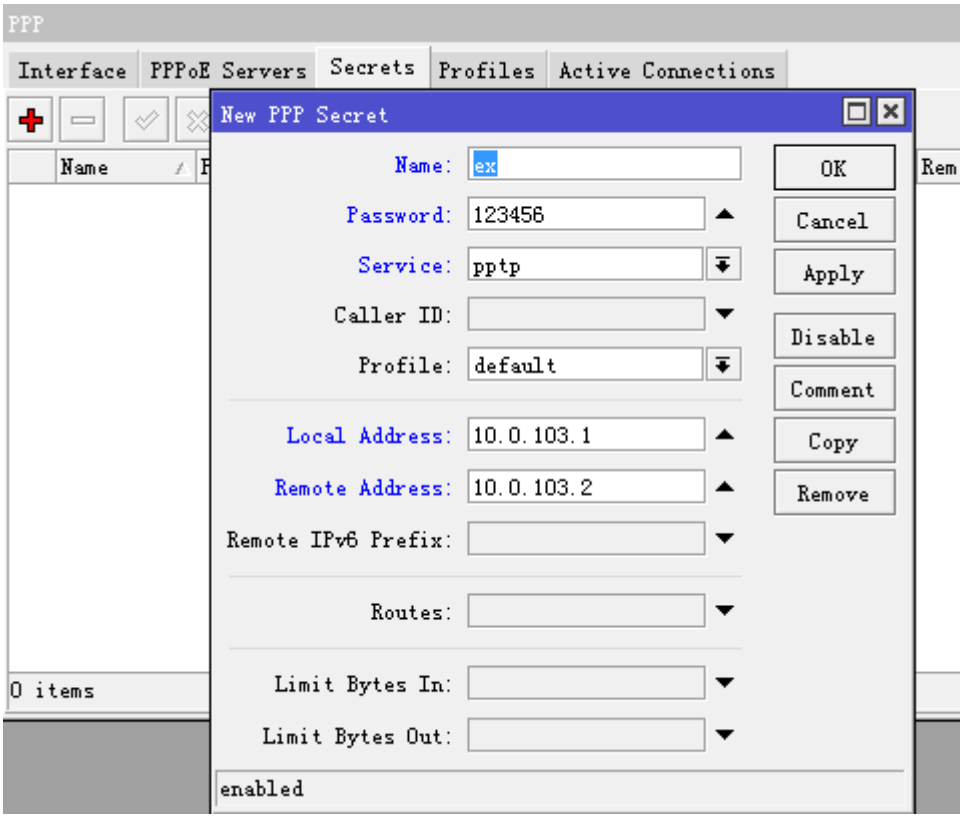
```
[admin@Office] /ip route> add gateway=192.168.90.254
[admin@Office] /ip route>/ ip firewall nat
[admin@Office] /ip firewall nat>add action=masquerade chain=srcnat
```

PPTP 创建连接

我们在 Office 端建立 PPTP 服务器，首先我们进入/ppp secret 目录下添加客户端账号，并分配给客户的隧道 IP 地址 10.0.103.2，使用 10.0.103.1 地址为客户的隧道网关：

```
[admin@Office] ppp secret> add name=ex service=pptp password=123456
local-address=10.0.103.1 remote-address=10.0.103.2
[admin@Home] ppp secret> print detail
Flags: X - disabled
  0    name="ex" service=pptp caller-id="" password="123456" profile=default
       local-address=10.0.103.1 remote-address=10.0.103.2

[admin@Office] ppp secret>
```

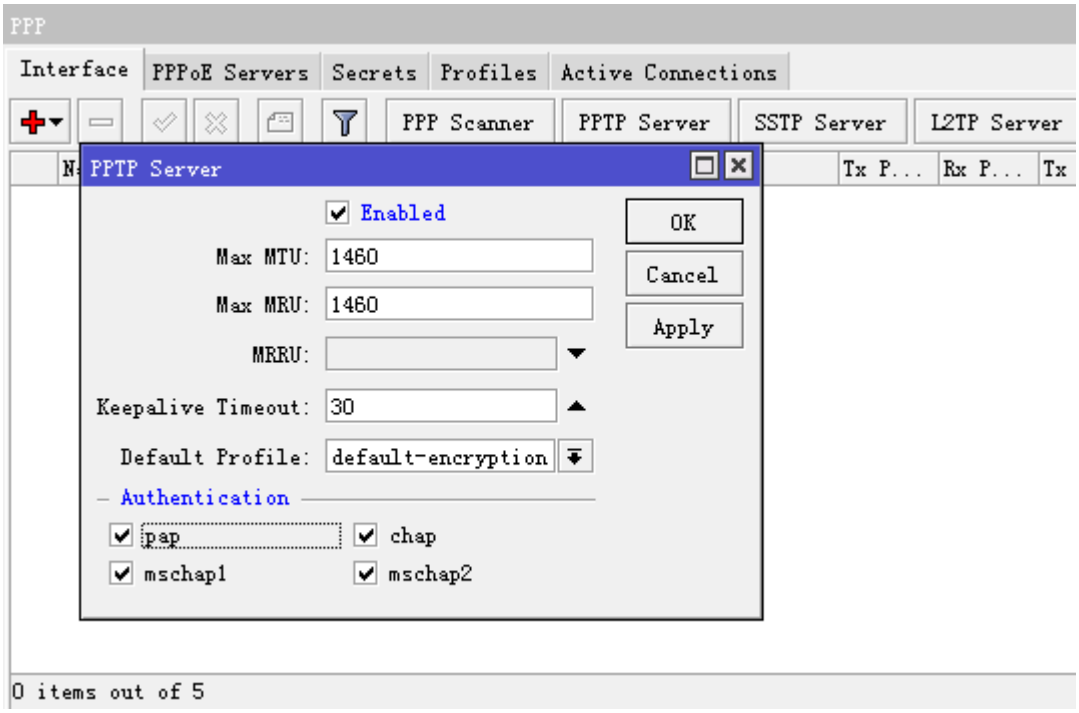
Winbox 操作如下：



在 interface pptp-server server 目录下，启用 pptp 服务器：

```
[admin@Office] interface pptp-server server> set enabled=yes
[admin@Office] interface pptp-server server> print
               enabled: yes
                   mtu: 1460
                   mru: 1460
      authentication: mschap2
     default-profile: default
[admin@HomeOffice] interface pptp-server server>
```

Winbox 下配置进入 ppp 目录下启用 pptp server：



Home 配置

配置 IP 地址

```
[admin@Home] ip address >add address=192.168.80.1/24 interface=ether1
[admin@Home] ip address >add address=10.1.202.1/24 interface=ether2
```

添加默认网关

```
[admin@Home] /ip route> add gateway=192.168.80.254
```

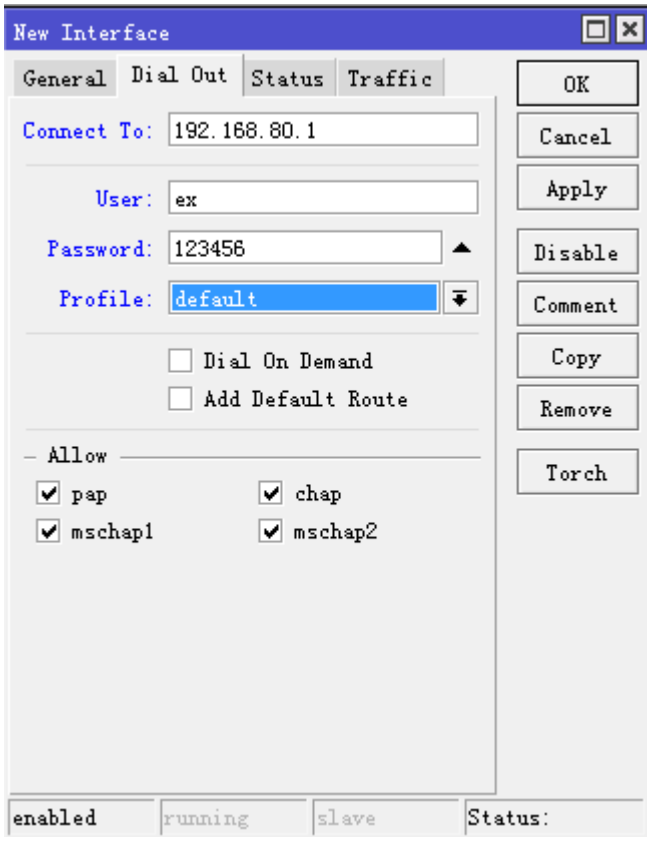
在 firewall nat 下添加 nat 地址转换

```
[admin@Home] /ip firewall nat>add action=masquerade chain=srcnat
```

在 Home 路由器添加一个 PPTP 客户：

```
[admin@Home] interface pptp-client> add connect-to=192.168.80.1 user=ex \
\... password=123456 disabled=no
[admin@RemoteOffice] interface pptp-client> print
Flags: X - disabled, R - running
  0   R name="pptp-out1" mtu=1460 mru=1460 connect-to=192.168.80.1 user="ex"
        password="123456" profile=default add-default-route=no

[admin@Home] interface pptp-client>
```

Winbox 在 interface 菜单中添加 pptp-client



在家庭路由器 pptp-client 连接状态显示为 R，表示连接成功，这样一个 PPTP 隧道就在路由器之间创建完成。

PPTP 隧道路由互访

pptp 隧道建立完成后，仅是路由器间可以互访，但两个企业间的局域网需要通过设置路由完成连接，根据以上事例，HomeOffice 路由器

为了在 PPTP 隧道上完成两个网络的互访，需要 Home 和 Office 的静态路由，

Home 路由器

```
[admin@Home] > ip route add dst-address=10.1.101.0/24 gateway=10.0.103.1
```

Office 路由器

```
[admin@Office] > ip route add dst-address=10.1.202.0/24 gateway=10.0.103.2
```

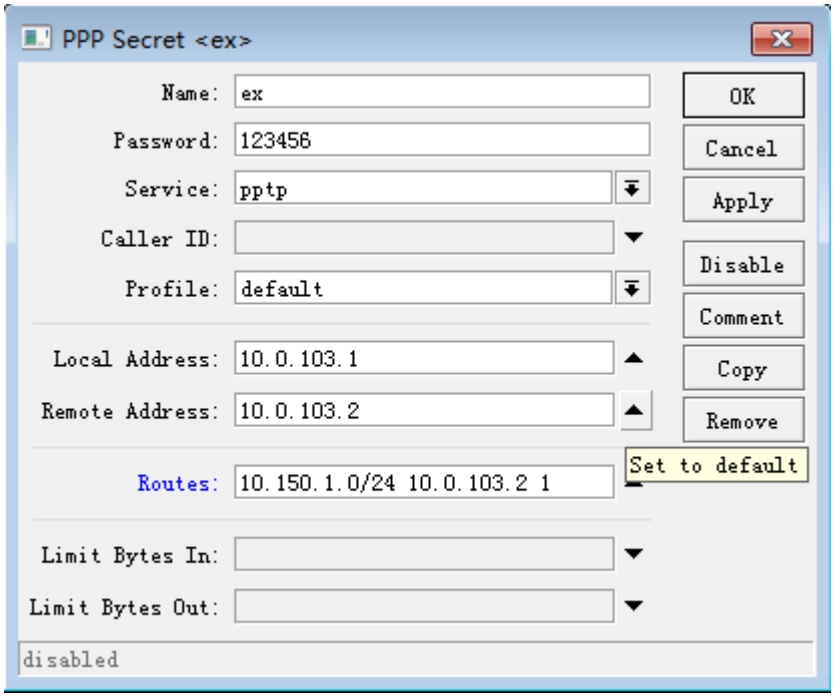
或者也可以在 PPTP 服务器，即 Office 端上通过用户配置的 routes 参数完成， Home 路由器还是需要在/ip route 中配置路由：

```
[admin@Office] ppp secret> print detail
Flags: X - disabled
 0    name="ex" service=pptp caller-id="" password="123456" profile=default
      local-address=10.0.103.1 remote-address=10.0.103.2 routes==""

[admin@Office] ppp secret> set 0 routes="10.1.202.0/24 10.0.103.2 1"
[admin@Office] ppp secret> print detail
```

```
Flags: X - disabled
  0    name="ex" service=pptp caller-id="" password="123456" profile=default
       local-address=10.0.103.1 remote-address=10.0.103.2
       routes="10.1.202.0/24 10.0.103.2 1"
```

代表：目的路由：10.1.101.0/24，经过 pptp 的网关：10.0.103.2，Distance 距离为 1。

在 Office 路由器的 Winbox 中修改 routes 参数

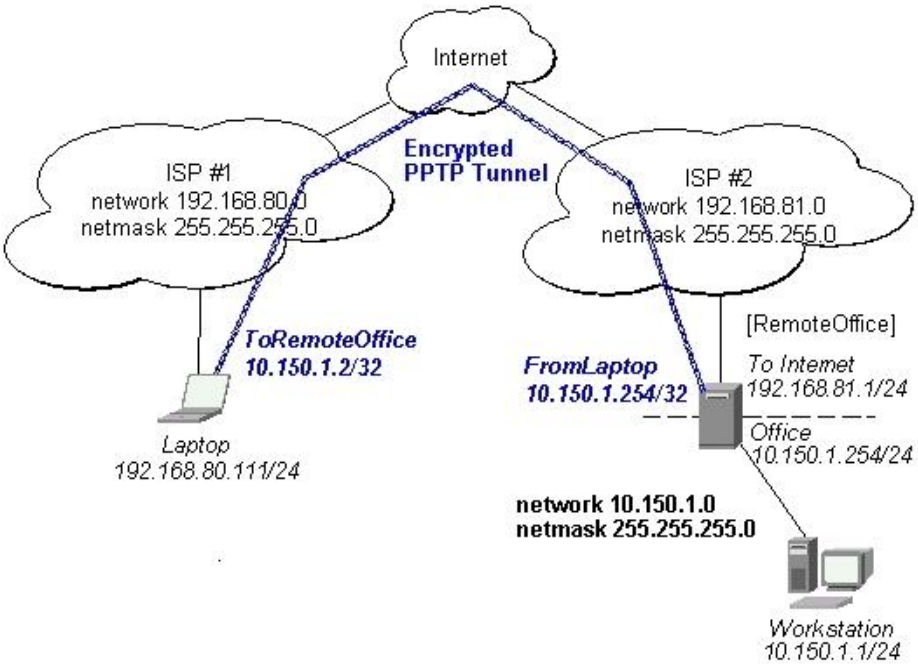


测试 PPTP 隧道路由连接：

```
[admin@Office]> /ping 10.1.202.2
10.1.202.2 pong: ttl=255 time=3 ms
10.1.202.2 pong: ttl=255 time=3 ms
10.1.202.2 pong: ttl=255 time=3 ms
ping interrupted
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 3/3.0/3 ms
```

通过 PPTP 隧道连接终端客户

下面的例子显示了通过终端计算机与远程办公网络进行 PPTP 加密隧道通信，如外地出差的同时，通过笔记本电脑连接会公司的网络进行远程信息管理和查询



这个例子中的路由器：

[RemoteOffice]

- 接口 WAN　192.168.81.1/24
- 接口 LAN　10.150.1.254/24

在 PPTP 服务器上设置用户账号：

```
[admin@RemoteOffice] ppp secret> add name=ex service=pptp password=123456
local-address=10.150.1.254 remote-address=10.150.1.2
[admin@RemoteOffice] ppp secret> print detail
Flags: X - disabled
 0    name="ex" service=pptp caller-id="" password="123456" profile=default
      local-address=10.150.1.254 remote-address=10.150.1.2 routes==""

[admin@RemoteOffice] ppp secret>
```

启用 pptp 服务：

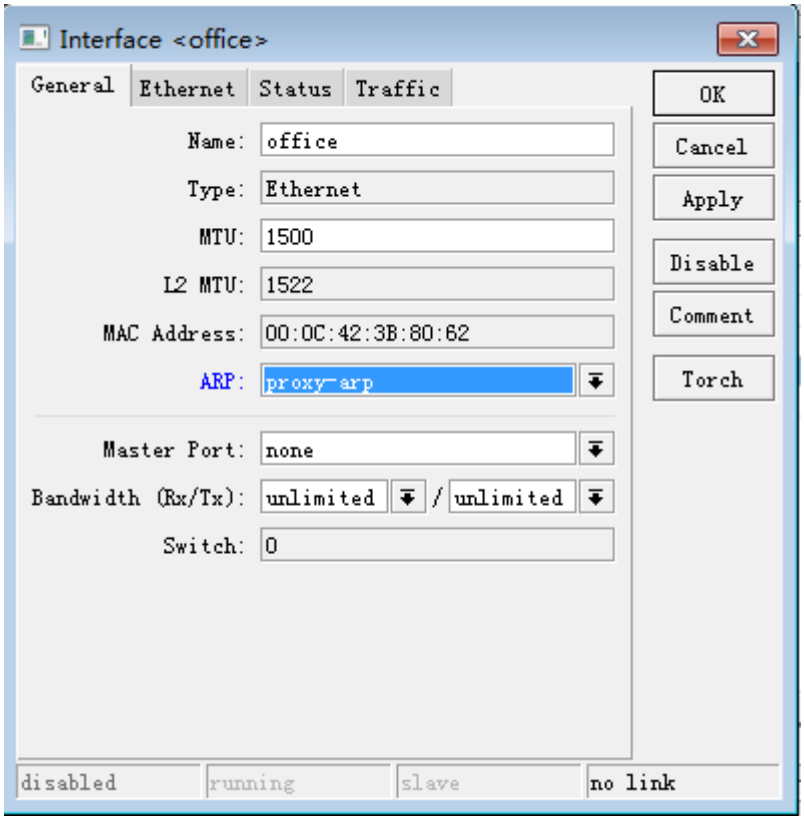
```
[admin@RemoteOffice] interface pptp-server server> set enabled=yes
[admin@RemoteOffice] interface pptp-server server> print
             enabled: yes
                 mtu: 1460
                 mru: 1460
      authentication: mschap2
     default-profile: default
[admin@RemoteOffice] interface pptp-server server>
```

当笔记本电脑远程访问公司内部资源时，需要配置相应规则才能确保获取资源正常的方法有以下两种：

局域网连接方法 1：代理 ARP 必须在'Office'接口上启用，这样可以通过代理 arp 访问，但有个缺点是内外的 DHCP 服务可能会受到影响：

```
[admin@RemoteOffice] interface ethernet> set Office arp=proxy-arp
[admin@RemoteOffice] interface ethernet> print
Flags: X - disabled, R - running
  #     NAME                    MTU    MAC-ADDRESS         ARP
  0   R ToInternet            1500   00:30:4F:0B:7B:C1 enabled
  1   R Office                1500   00:30:4F:06:62:12 proxy-arp
[admin@RemoteOffice] interface ethernet>
```

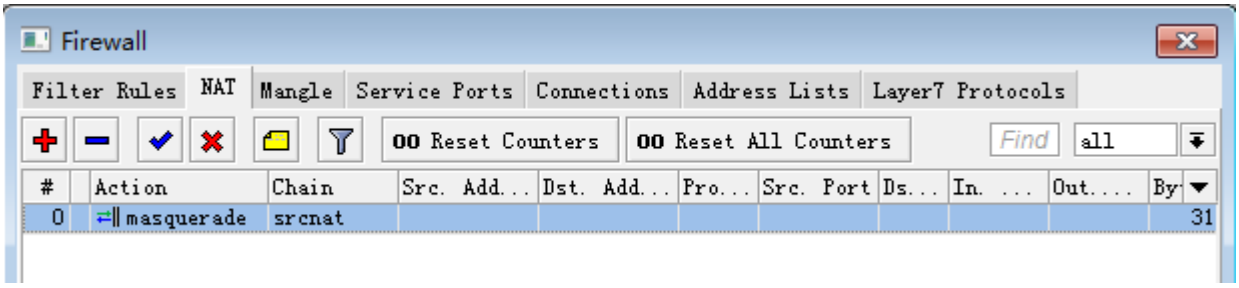
在 winbox 中进入 interface 目录下，选择 office 接口设置 arp 为 proxy-arp



局域网连接方法 2：通过 nat 设置 masquerade，规则要求对所有来访数据进行伪装，这样保证内外网通过转换通信

```
[admin@RemoteOffice] /ip firewall nat> add chain=srcnat action=masquerade
[admin@RemoteOffice] /ip firewall nat> print
Flags: X - disabled, R - running
  Flags: X - disabled, I - invalid, D - dynamic
 0    chain=srcnat action=masquerade
[admin@RemoteOffice] interface ethernet>
```

在 winbox 中添加 masquerade 规则：



Windows 的 PPTP 设置

对 Windows NT，2000，98SE 以及 98 支持 PPTP 客户。Windows 98SE，2000，以及 ME 包括 Windows 设置中的支持或者自动安装 PPTP。对 95，NT，及 98，安装需要从 Microsoft 下载。很多 ISP 都制作了说明页面以说明客户进行 Windows PPTP 安装。

PPTP (VPN)安装的简单说明及客户设置 - Windows 98SE

如果 VPN（PPTP）套件已经安装，选择'Dial-up Networking' 和 'Create a new connection'。创建一个 VPN 的选项应该选择。如果没有 VPN 选项，那么按照下面的安装说明进行。当询问 VPN 服务器主机名或 IP 地址时，输入路由器的 IP 地址。双击'new'图标并输入正确的用户名和密码（必须在路由器或用于认证的用户数据库中）。

连接的设置在选择了'connect'按钮后需要 9 秒钟。建议把连接属性进行编辑以便'NetBEUI', 'IPX/SPX compatible'，及'Log on to network'为未选择的。连接的设置时间为在'connect'按钮选择后 2 秒钟。

为了安装 Windows 98SE 的 VPN 套件，从'Start'主目录中选择'Setting'。选择'Control Panel'，选择 'Add/Remove Program'，选择'Windows setup'标签，选择 'Communications'软件安装以及'Details'。在软件列表的底部选择'Virtual Private Networking'安装。

故障分析

- **我使用了防火墙但我不能建立 PPTP 建立**

  确定 TCP 连接到 1723 端口可以通过你的两个网站。

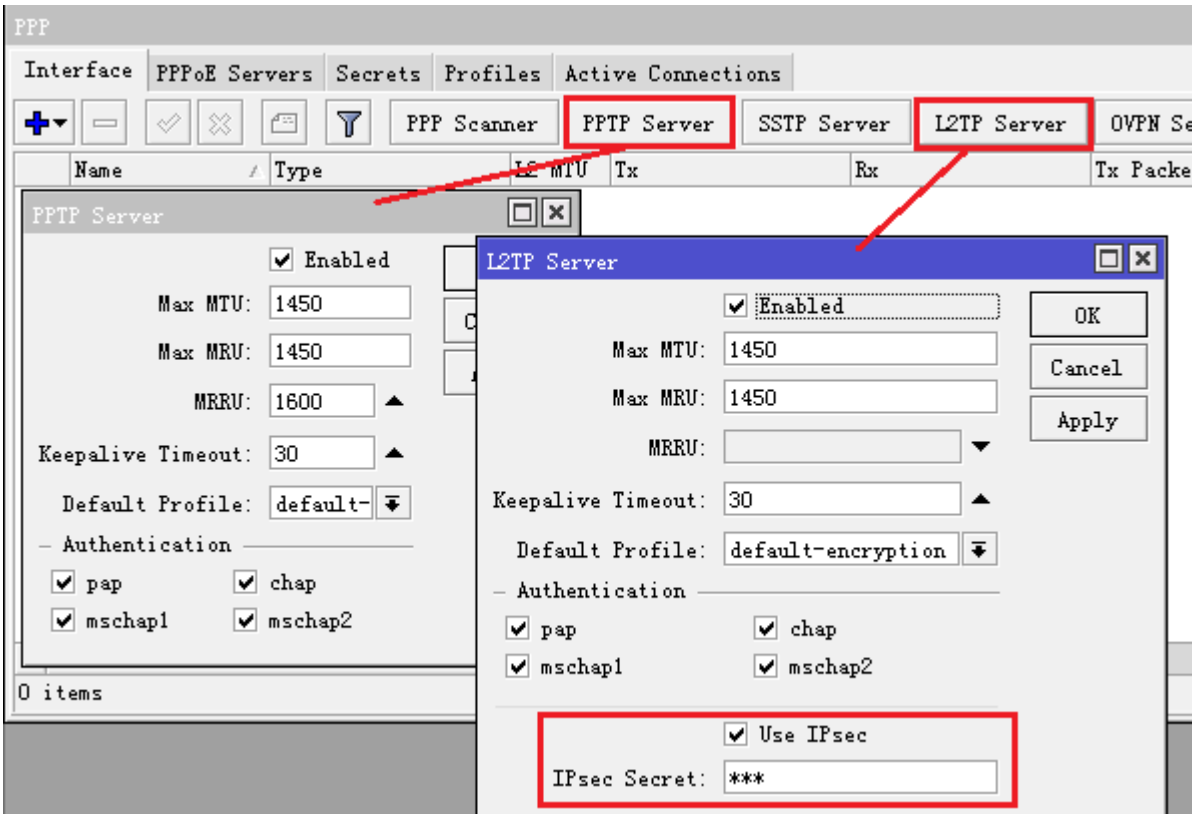
第二十三章 PPTP 与 L2TP 服务

PPTP 和 L2TP 都使用 PPP 协议对数据进行封装，然后添加附加包头用于数据在互联网络上的传输。尽管两个协议非常相似，但是仍存在以下几方面的不同：

PPTP 要求互联网络为 IP 网络。L2TP 只要求隧道媒介提供面向数据报的点对点的连接。PPTP 只能在两端点间建立单一隧道。L2TP 支持在两端点间使用多隧道。使用 L2TP，用户可以针对不同的服务质量创建不同的隧道。L2TP 可以提供包头压缩。当压缩包头时，系统开销（overhead）占用 4 个字节，而 PPTP 协议下要占用 6 个字节。L2TP 可以提供隧道验证，而 PPTP 则不支持隧道验证。但是当 L2TP 或 PPTP 与 IPSEC 共同使用时，可以由 IPSEC 提供隧道验证，不需要在第 2 层协议上验证隧道

23.1 同时建立 PPTP 和 L2TP 服务器

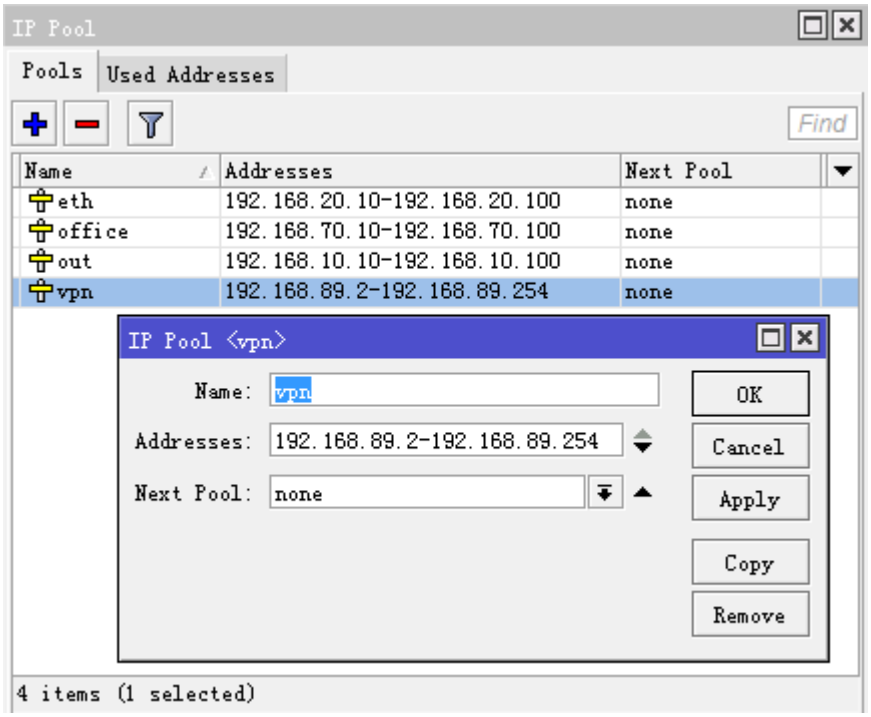
首先我看一下 PPTP 和 L2TP 的建立，同样是在 PPP 的目录下，interface 菜单中，选择 PPTP-Server 和 L2TP-Server 并启用服务：



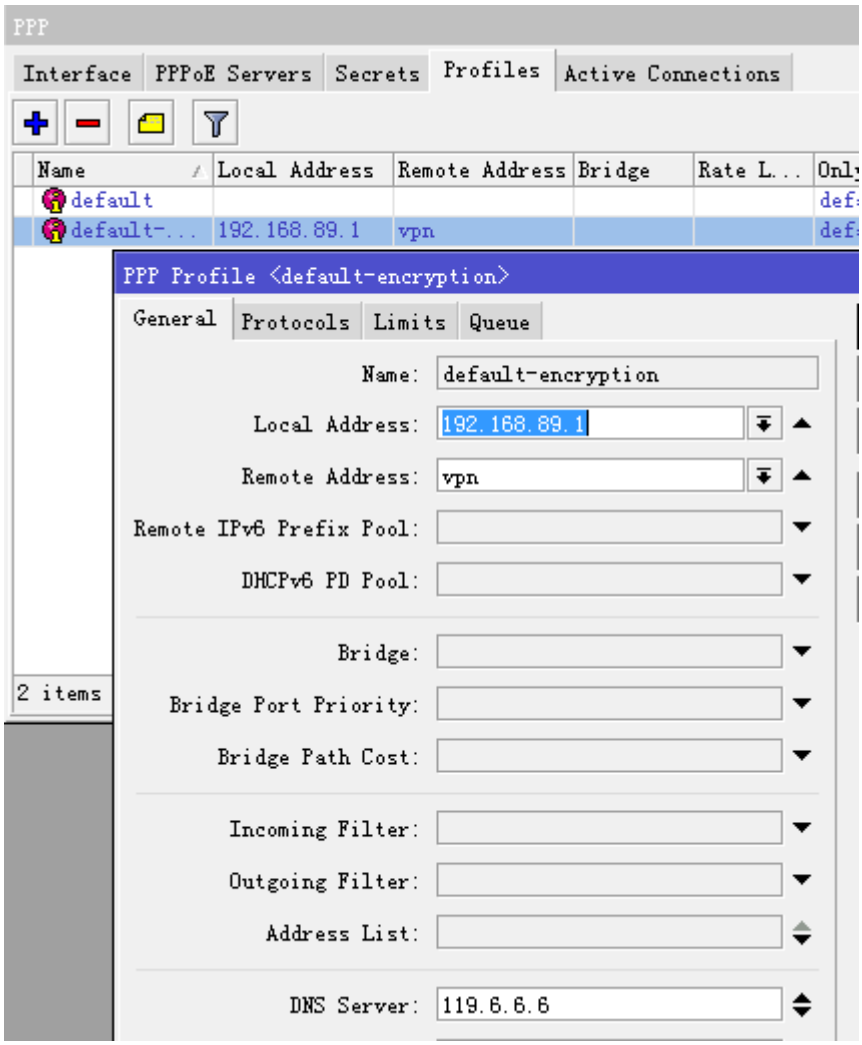
在 L2TP 服务器配置下多了 Use IPsec 选项，这是在 RouterOS 6.16 后加入的 IPsec 选项，方便 windows 客户端的连接，Default-Profile 类型选择 default-encryption，Autentication 认证方式也可以选择相同方式。

这里我们举一个实例，我们建立了一个主机的 VPN 服务，同时启用 PPTP 和 L2TP 方式，分配远程 IP 为 192.168.89.2-192.168.89.254 的地址池，我用 192.168.89.1 做为 VPN 隧道的本地 IP。

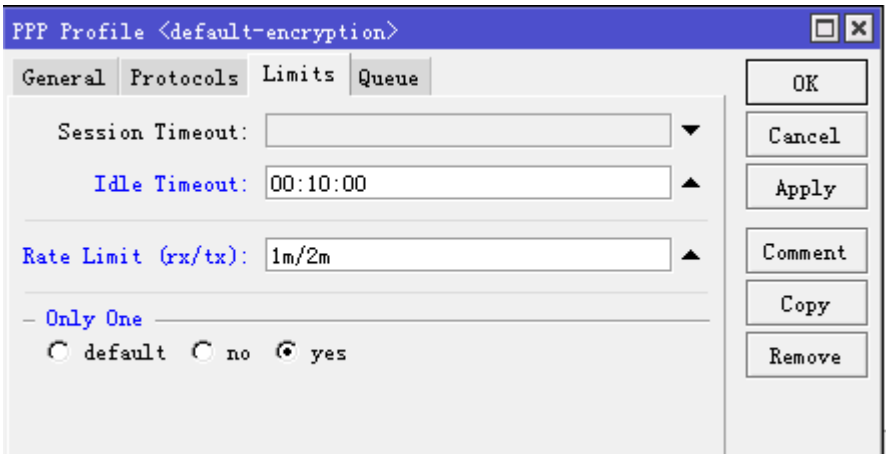
首先我进入 ip pool 中配置地址池：



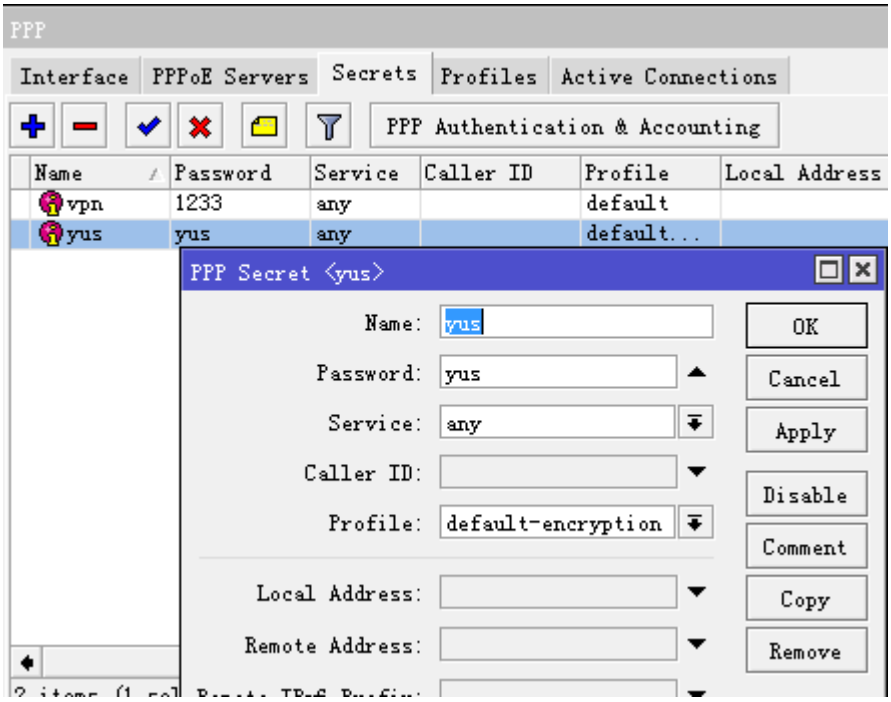
配置好地址池后，在 PPP Profiles 中使用默认的 default-encryption 组规则，配合本地 IP 地址 192.168.89.1，在远程 remote-address 种配置之前添加号的地址池 VPN，设置 DNS 为 119.6.6.6,其他配置参数如下：



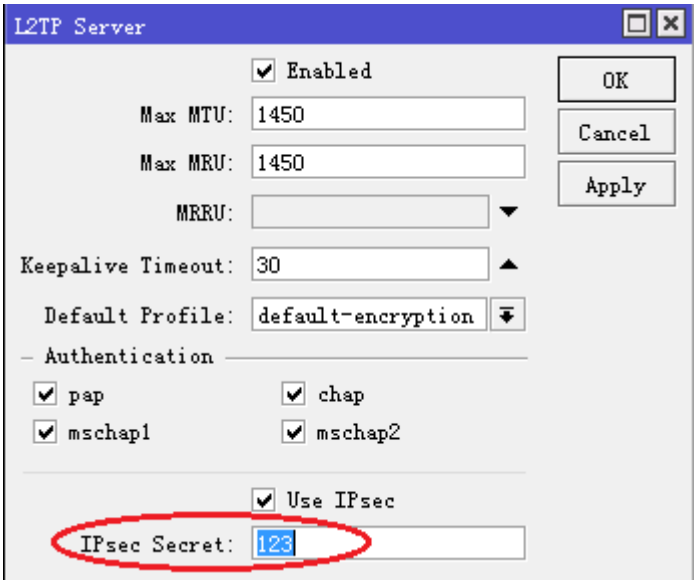
在 limits 选项中，配置相应的 Idle-timeout（空闲超时时间）为 10 分钟、Rate-limit（带宽）为 1M 上行和 2M 下行，配置 Only-one（账号是否唯一性）：



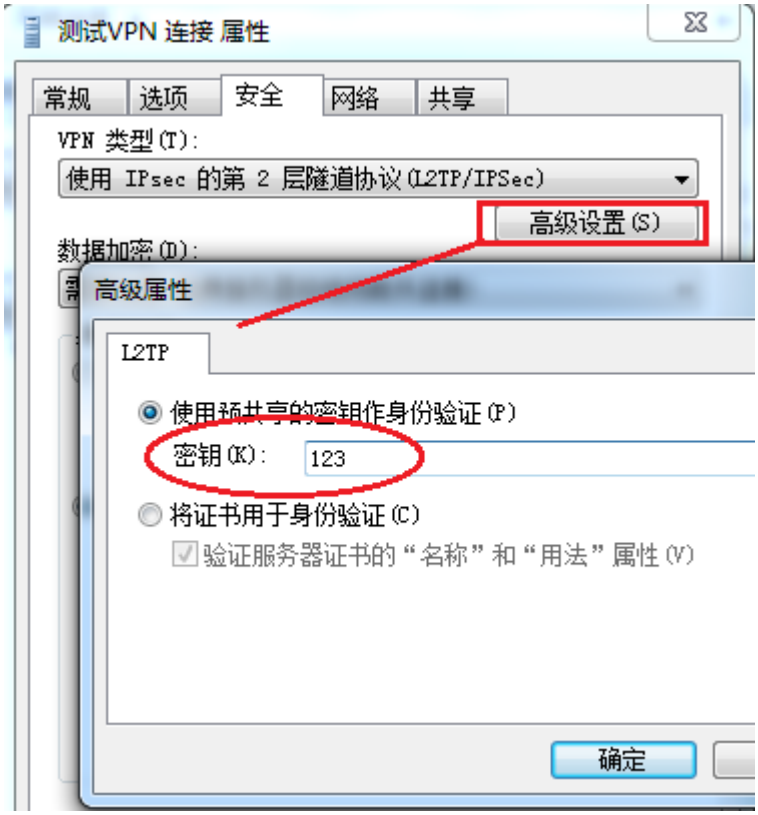
在 PPP secret 中配置用户，账号 yus，密码 yus，service 参数用于选择 VPN 类型，这里设置为默认 any 这样支持 L2TP 和 PPTP 登陆方式，profile 为 default-encryption，



配置完成后，我们便可以通过 PPTP 或者 L2TP 连接 RouterOS 的 VPN 服务，在 windows 下可以通过 PPTP 的方式直接连接 RouterOS 的 VPN 服务，由于 windows 要求 L2TP 进行 IPsec 的加密方式连接，所以需要在 L2TP 配置 IPsec 密钥，如果不考虑使用 IPsec 的 L2TP 连接，可以修改 windows 注册表。



Windows 配置在 VPN 属性中，选择“安全”，VPN 类型选择“使用 IPsec 的第 2 层隧道协议（L2TP/IPsec）”，点击“高级设置”，设置“使用预共享的密钥作为身份验证”



L2TP 的 Windows 注册表修改

这里介绍的是在 Windows 下不使用 IPsec 的 L2TP 连接，后面有提到如何配置 RouterOS 使用 L2TP/IPsec 的 Windows 连接。这里介绍修改 windows 的 L2TP 注册表，缺省的 Windows XP L2TP 传输策略不允许 L2TP 传输不使用 IPSec 加密。可以通过修改 Windows XP 注册表来禁用缺省的行为，手工修改：

1) 进入 Windows XP 的“开始”“运行”里面输入“Regedt32”，打开“注册表编辑器”，定位“HKEY_Local_Machine \ System \ CurrentControl Set \ Services \ RasMan \Parameters ”主键。

2) 为该主键添加以下键值：

- 键值：ProhibitIpSec
- 数据类型：reg_dword
- 值：1

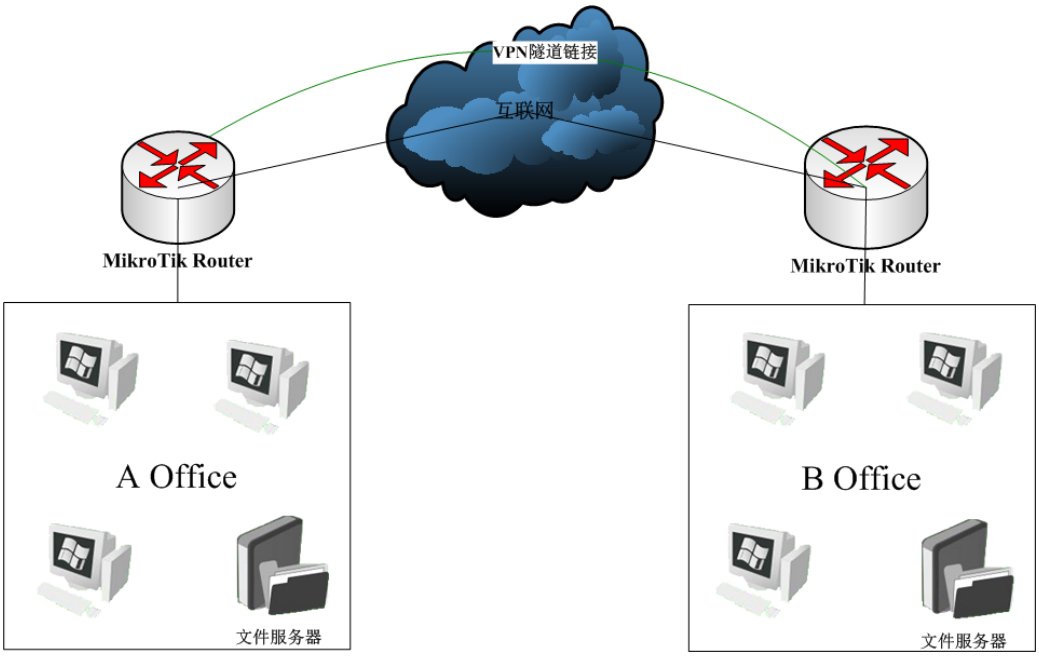
修改后即可通过 windows 正常连接到 L2TP 服务。在 RouterOS 6.16 后加入了 L2TP 服务中配置 IPsec 的选项，简化了 IPsec 配置操作，所以现在修改注册表会用到的人很少。

注：PPTP 和 L2TP 的端口都是固定的无法自定义，但 OVPN 和 SSTP 隧道 VPN 的端口可以自己定义，这有助于受到网络环境限制，采取灵活处理方法。

23.2 VPN 的几种应用方式

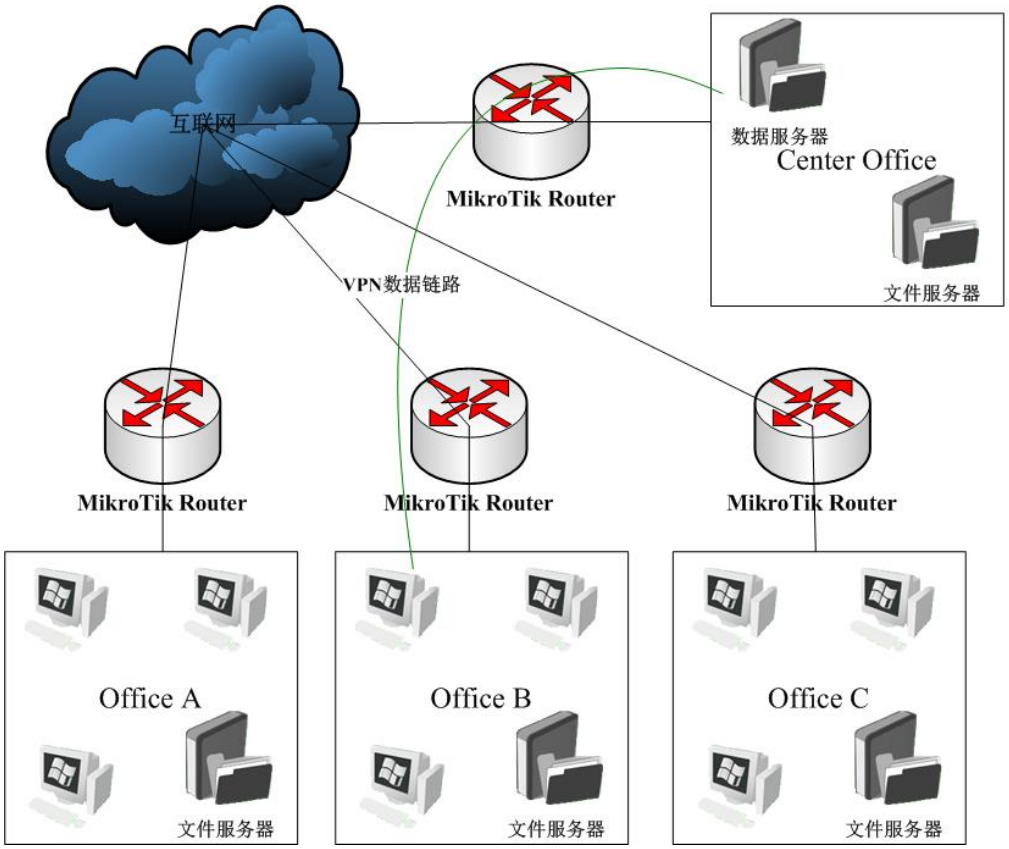
RouterOS 的 VPN 系统支持多种方式的应用，能实现企业对等访问、企业与多分支点、多点与移动办公和 VPN 数据转移等多种 VPN 连接方式。在 VPN 数据转移方面用处最多的方式是在 VoIP 方面，因为受某些 ISP 网络的限制，使得正常的 VoIP 通讯收到影响，所以可以通过 VPN 的方式实现数据的转移。

企业间对等互访



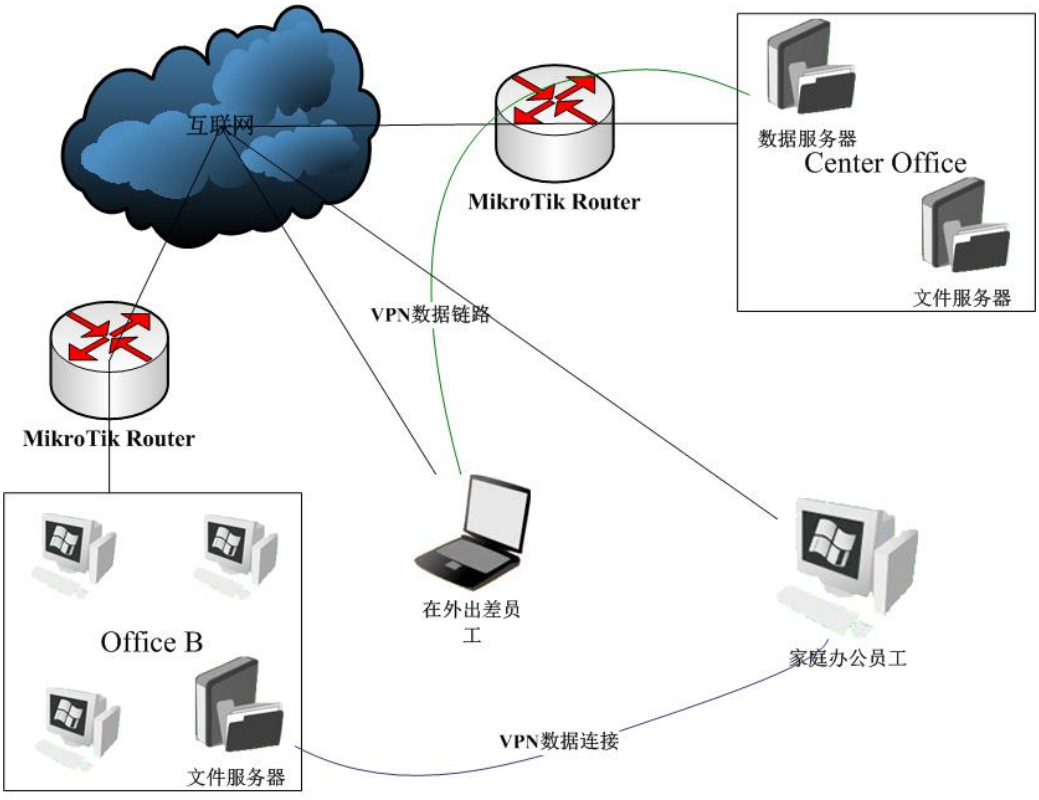
通过 VPN 隧道协议如 PPTP、L2TP 或者 IPIP 可以建立一个对等的隧道，使得两个办公室能互相访问公司数据和文件。这种方式我们首先建立 PPP 服务器，并给客户端分配账号和固定 IP 地址、建立 PPP 的拨号和分配 IP 地址，最后设置两个远程局域网的 IP 地址路由（操作可以参考 PPTP 章节）

企业与分支点总分连接



我们可以使用 PPTP 或者 L2TP 等隧道协议，建立多个客户端账号，使多个分公司能连接到总公司的中心服务器，并能通过公司管理各个点的数据和信息互联。这样总公司做到对分公司的有效管理，可以实时发布信息到各个分公司。并能实现数据的安全传输。

企业移动办公与综合应用



RouterOS 支持普通用户的 PPTP 和 L2TP 的 windows 的拨号连接，所以对于在外出差和家庭办公的用户就可以方便的连接到总部的中心服务器和分支点的文件服务器，特别对需要实时处理问题的公司员工最为适合。

23.3 BCP 协议

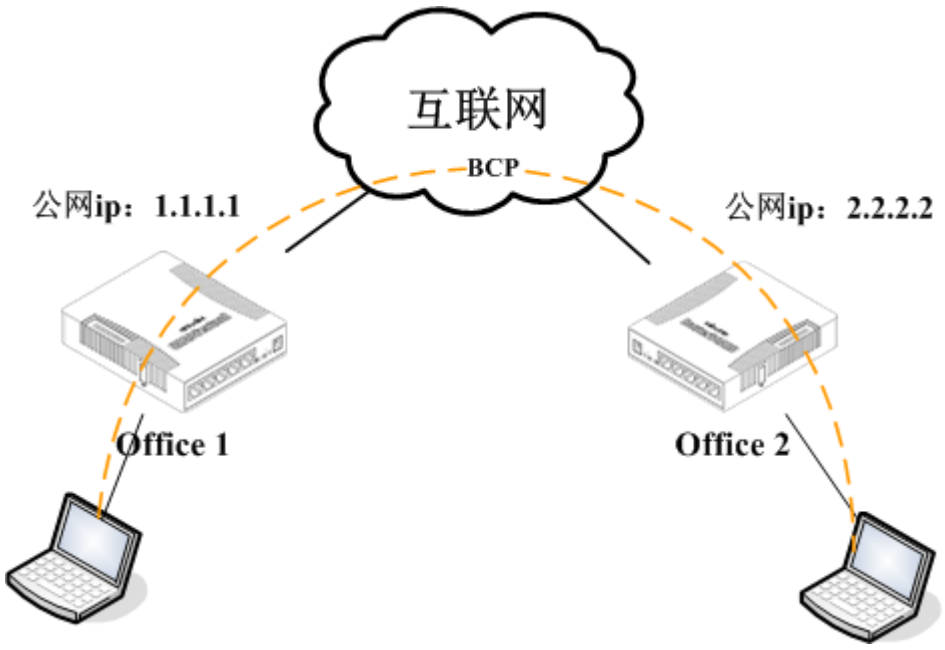
RouterOS 支持 BCP(Bridge Control Protocol)，即在 PPP、PPTP、L2TP 和 PPPoE 接口上的桥接。BCP 协议通过 PPP 协议，将两个远程的以太网数据链路打通。BCP 建立后独立于 PPP 隧道，将不会与任何 PPP 的 IP 地址接口有关系。BCP 能用于替代 EoIP+VPN 隧道，EoIP 要求对等的网络连接，而 BCP 为网络提供另一种方式的解决，特别是一端在 nat 网络内，经过测试能正常透过 nat 网络透传二层数据。

BCP (Bridge Control Protocol)需要在两边同时启用才能工作(PPP 服务器和 PPP 客户端)。MikroTik RouterOS 也可应用于其他的 PPP 设备，要求这个设备支持标准的 BCP 协议。

配置事例

我们需要相互连接 2 个远程办公室，并让他们在同一个以太网内工作。我们要求使用加密（encryption）保护 2 个办公室的数据交换。

如下图，我有 2 个办公室，办公室 1 设置为 PPTP 服务器，办公室 2 设置为 PPTP 客户端。下面通过 winbox 和 CLI 介绍配置：



Office1 配置

首先我们需要建立一个桥接口，并确保桥接将一直有 MAC 地址存在。原因很简单，当 BCP 被使用 PPP 桥接 port 中，不会有任何 MAC 地址生成。

```
/interface bridge add name=bridge_local protocol-mode=rstp
/interface bridge port add bridge=bridge_local interface=ether1_local
/interface bridge set bridge_local admin-mac=xx:xx:xx:xx:xx:xx
```

其中 xx:xx:xx:xx:xx:xx 是 ether1_local 的 MAC 地址

现在我们能分配本地和公网地址到相应的接口上：

```
/ip address add address=192.168.88.1/24 interface=bridge_local
/ip address add address=1.1.1.1/24 interface=ether2_public
```

在这个事例中，仅使用 PPP 做桥接，PPP profile 和 secret 的配置非常简单-仅分配用户名和密码，并指定 profile 的 bridge 选项。PPP 桥接不需要任何 IP 地址，但正常的 PPP 是必需的，所以要指定 local 和 remote 地址在服务器上。

```
/ppp profile add name=ppp_bridging bridge=bridge_local use-encryption=yes
/ppp secret add profile=ppp_bridging name=ppp1 password=ppp1
```

当桥接的 PPP 隧道需要通过二层（MAC）数据包头部信息，由于默认的接口 MTU（PPTP 是 1460）不能满足这个的通讯，所以为确保适用运用环境，建议不用考虑 MTU 值，通过在服务器的 MRRU 选项设置更高的值。

MRRU 允许启用支持单连接协商的多重链路，奋力数据报到多个通道，因此增加 MTU 和 MRU（支持 65535 字节）

```
/interface pptp-server server set enabled=yes mrru=1600
```

Office2 配置

首先我们需要建立桥，并确定桥将有 MAC 地址存在，原因如上提到。

```
/interface bridge add name=bridge_local protocol-mode=rstp
/interface bridge port add bridge=bridge_local interface=ether1_local
/interface bridge set bridge_local admin-mac=xx:xx:xx:xx:xx:xx
```

其中 xx:xx:xx:xx:xx:xx 是 ether1_local 的 MAC 地址。

现在我们能分配本地和公网地址到相应的接口上：

```
/ip address add address=192.168.88.254/24 interface=bridge_local
/ip address add address=2.2.2.2/24 interface=ether2_public
```

配置 PPP Profile，响应在服务器端的配置

```
/ppp profile add name=ppp_bridging bridge=bridge_local use-encryption=yes
```

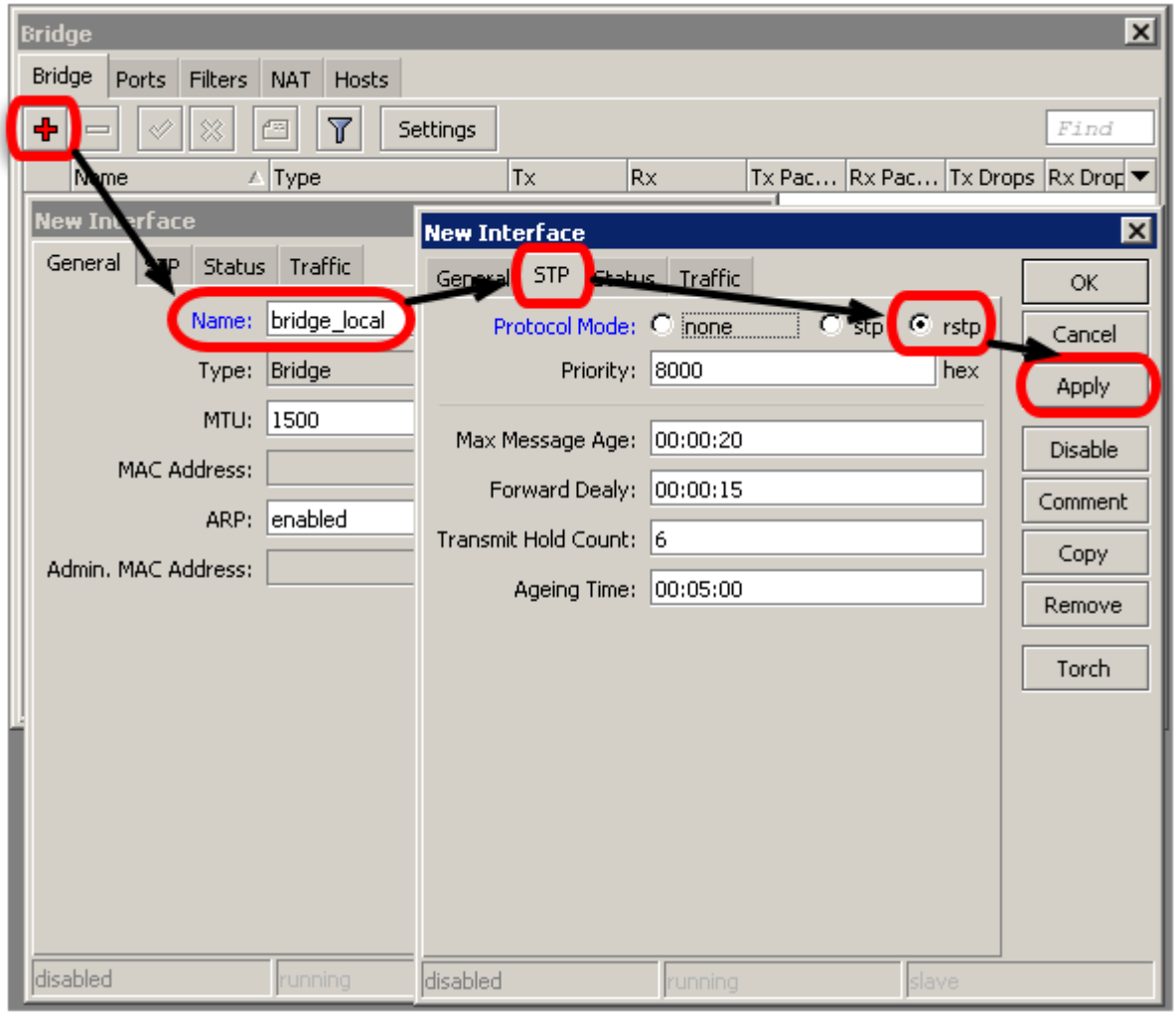
创建一个 pptp-client 接口，不要忘记配置 MRRU 选项，确保二层帧能通过 PPP 隧道。

```
/interface pptp-client add profile=ppp_bridging mrru=1600 connect-to=1.1.1.1 user=ppp1 password=ppp1
disabled=no
```

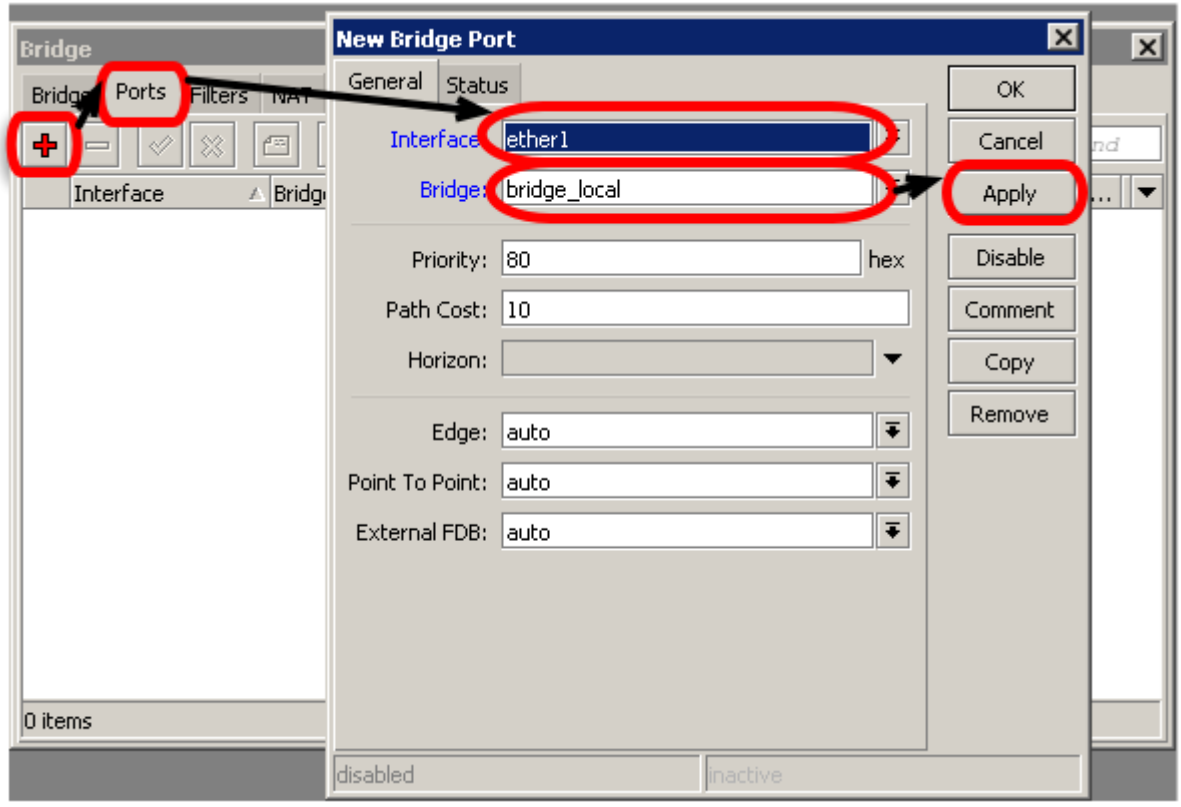
BCP winbox 配置

Office1 配置

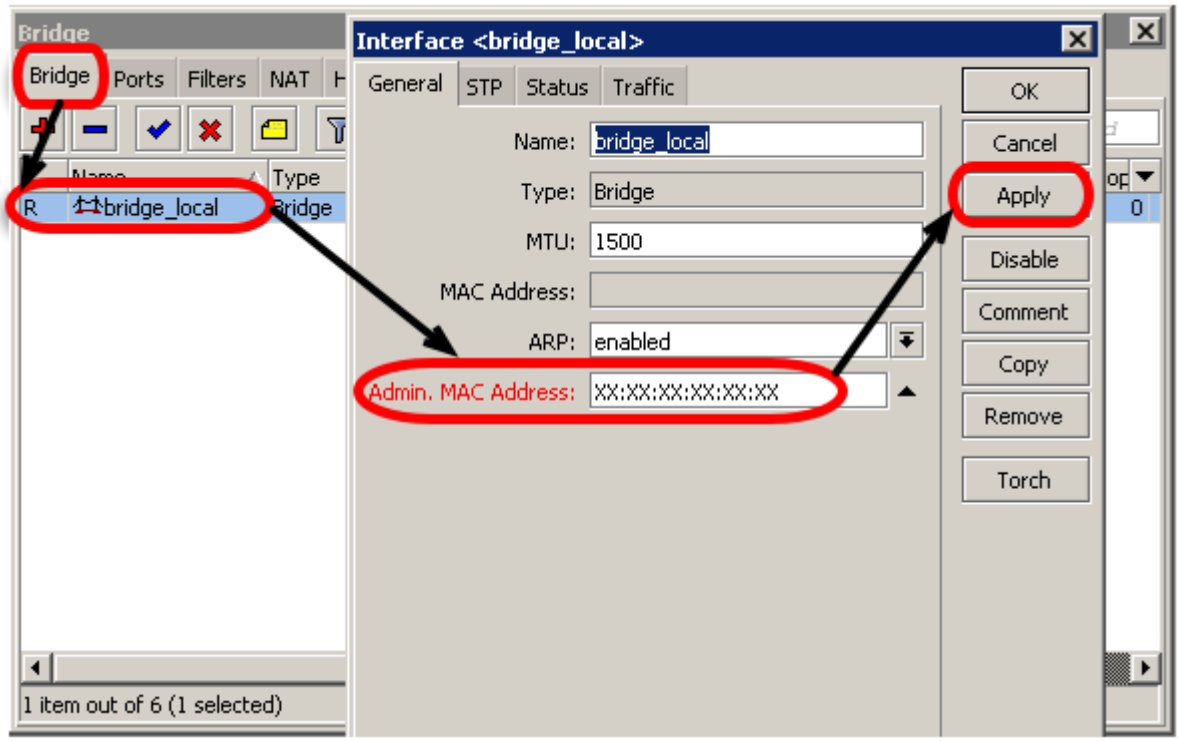
在 bridge 中添加一个桥，并设置 rstp：



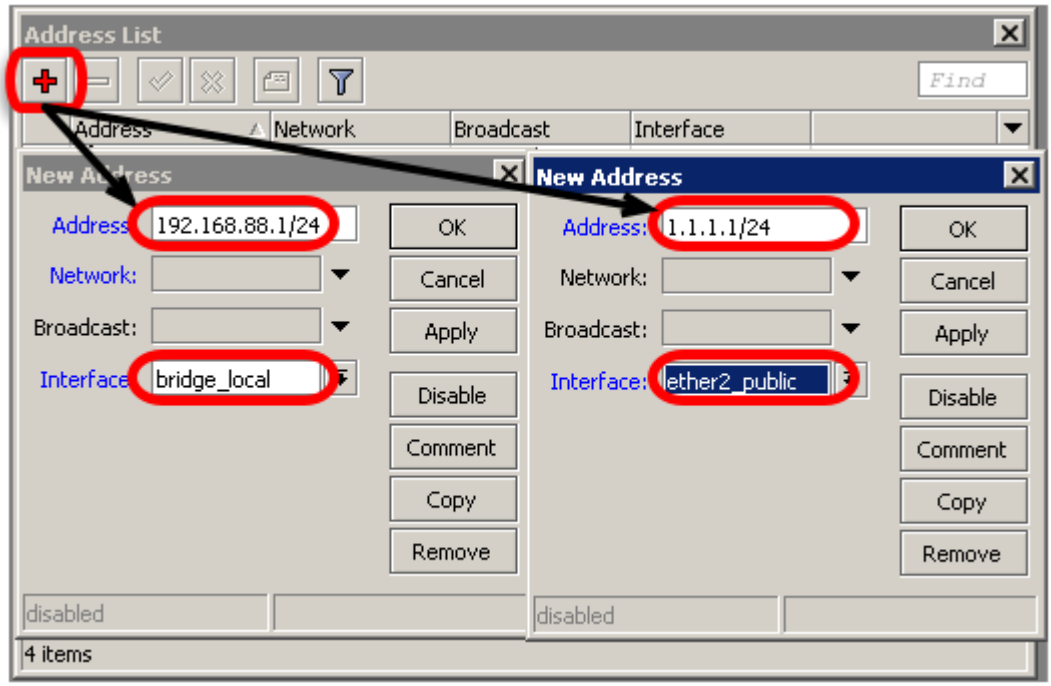
在 port 中，添加桥接的接口，我们添加 ether1 到 port 用于连接内网：



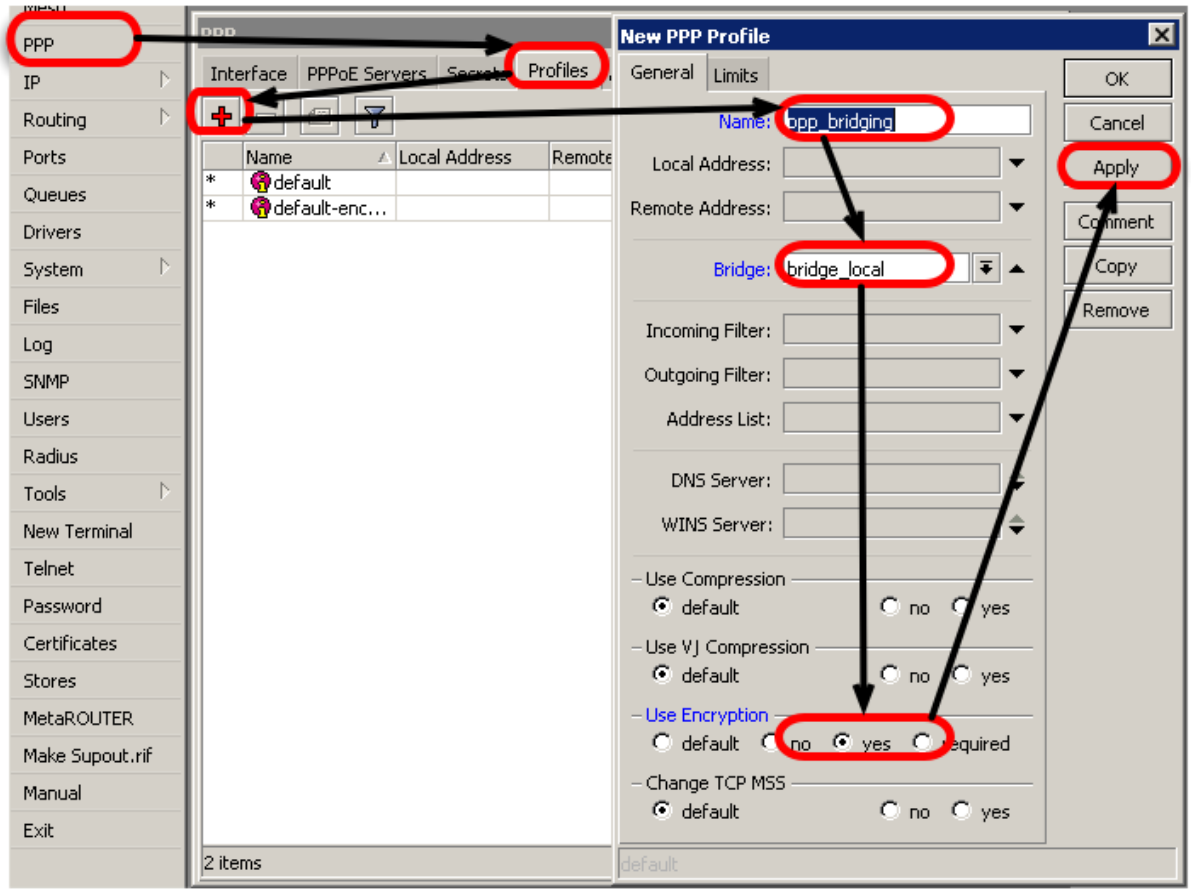
设置静态的 MAC-address：



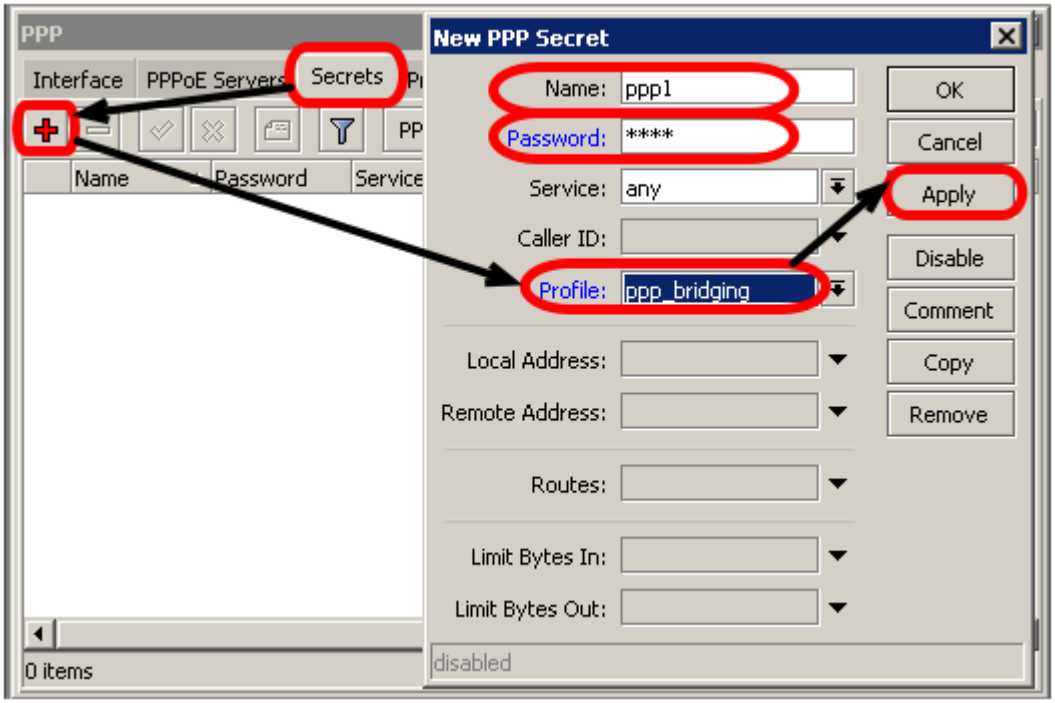
分配 IP addresses,



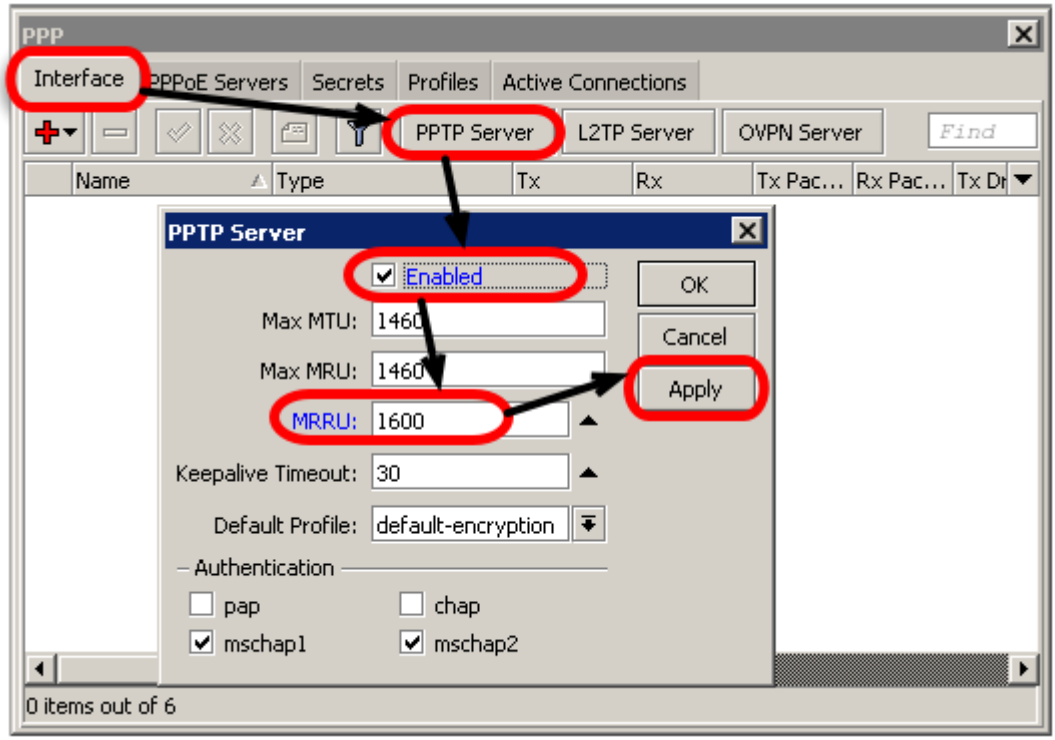
创建 PPP profile，并设置 bridge 参数，



添加 PPP 客户端

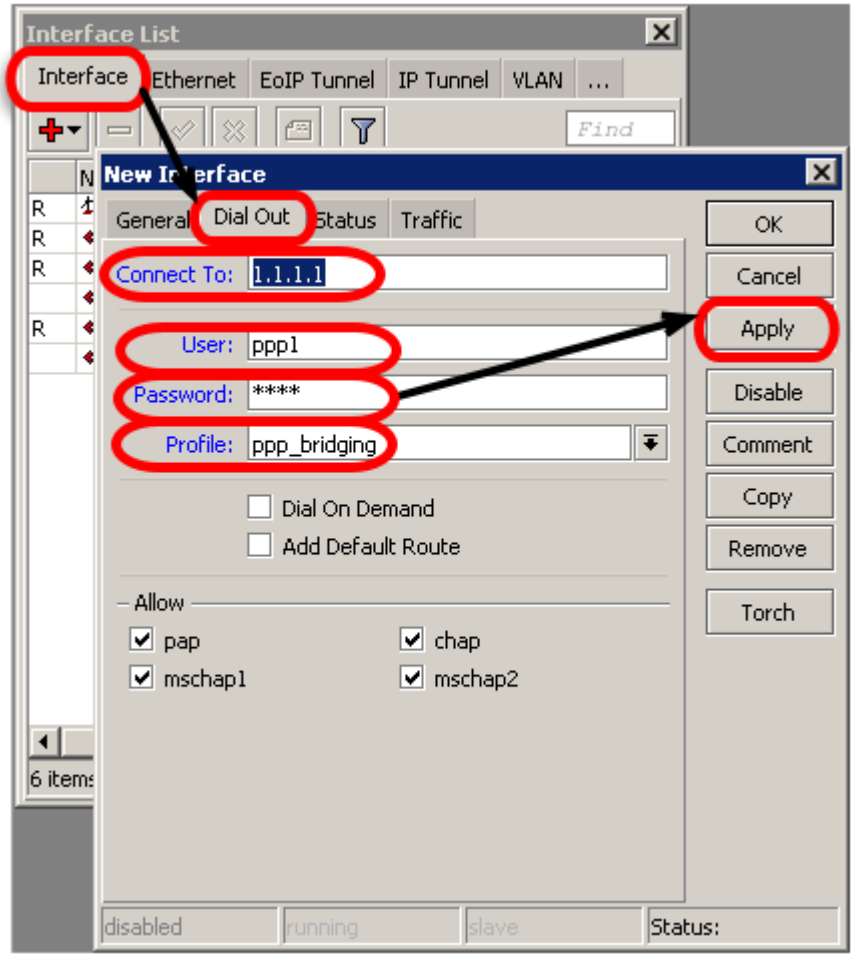


启用 PPTP-server，PPTP 服务器的 MRRU 一定要设置为 1600，否则会导致网页无法打开的情况。



Office2 配置

客户端路由器配置相同，只是你需要配置并启用 PPTP 客户端，添加 PPTP client，同样需要设置 MRRU=1600，然后配置 pptp 拨号信息：



在实际网络应用中，也可以将 vlan 和 ppp 做到一个 bridge 中，通过 vlan 来划分远程桥接的区域，桥接隧道的互联网应用有很多种，特别是互联网企业网络和运营网络涉及较多。

第二十四章 Open VPN

OpenVPN 是一个基于 OpenSSL 库的应用层 VPN 实现，和传统 VPN 相比，它的优点是简单易用。OpenVPN 允许参与建立 VPN 的单点使用共享密钥，电子证书，或者用户名/密码来进行身份验证。OpenVPN 已经被转移到各种平台，包括 Linux 和 Windows，**RouterOS** 在 v3.x 支持 OpenVPN ，你需要通过安装和启用 ppp 功能包。在 RouterOS 平台对 OpenVPN 仅支持 tcp 方式，udp 方式不支持。

在 **Windows** 你需要另外的 GUI，系统中允许你在 windows 系统中安装客户端，你可以到 OpenVPN GUI 下载 http://www.openvpn.se/download.html .

OpenVPN 要求与 SSL 证书一起工作，你需要生成一个证书，一般可以通过 linux 安装 OpenVPN 功能包后生成，或者通过 http://cacert.org 网站申请，在 RouterOS v6 版本后，可以通过 certificate 创建和管理证书，即可以自己做 RouterOS 生成和签发自己的证书。

当前 RouterOS 不支持以下 OVPN 功能：

- UDP 连接模式
- LZO 压缩 compression
- TLS 验证 authentication
- 无需账号/密码方式验证

24.1 OVPN Client

操作路径：/interface ovpn-client

属性描述

属性	描述
add-default-route (*yes* | *no*; 默认: **no**)	是否添加 OVPN remote address 作为路由器的默认网关
auth (*md5* | *sha1*; 默认: **sha1**)	设置验证模式
certificate (字符型 | *none*; 默认: **none**)	选择导入 certificate 列表中的客户端证书名称
cipher (*aes128* | *aes192* | *aes256* | *blowfish128*; 默认:**blowfish128**)	设置加密类型
comment (*字符型*; 默认:)	项目注释描述
connect-to (*IP 地址*; 默认:)	OVPN 服务器连接 IP 地址.
disabled (*yes* | *no*; 默认: **yes**)	是否禁用 OVPN 客户端网络接口，命令行添加后预设为禁用
mac-address (*MAC*; 默认:)	OVPN 接口的 MAC 地址，将自动生成无需指定
max-mtu (*整型*; 默认: **1500**)	OVPN 最大传输单元。
mode (*ip* | *ethernet*; 默认: **ip**)	选择二层或三层隧道模式

name (*字符型*; 默认:)	描述 OVPN 接口名称
password (*字符型*; 默认: **""**)	验证密码
port (整型; 默认: **1194**)	TCP 端口，暂不支持 UDP 模式
profile (*名称*; 默认: **default**)	使用 PPP profile 规则
user (字符型; 默认:)	验证用户名

以下为如何设置 OVPN 客户端实例，通过连接服务器 10.1.101.1，并设置用户名和密码为 test 和 123

```
[admin@bumba] /interface ovpn-client> add connect-to=10.1.101.1 user=test password=123 disabled=no
[admin@bumba] /interface ovpn-client> print
Flags: X - disabled, R - running
 0     name="ovpn-out1" mac-address=FE:7B:9C:F9:59:D0 max-mtu=1500 connect-to=10.1.101.1
       port=1194 mode=ip user="test" password="123" profile=default certificate=none auth=sha1
       cipher=blowfish128 add-default-route=no
```

24.2 OVPN Server

操作路径：/interface ovpn-server

在 ovpn-servers 目录会显示每个连接 OVPN 服务器的客户端

一个 server 接口可为每个隧道创建连接，OVPN 服务器可以创建如下两类接口

- 静态接口，由管理员手动添加，静态接口常用于需要将该接口引用到指定配置（在 firewall rule 规则、ip route 指定路由网关或其他配置），例如将一个来至 test 用户的 OVPN 接口设置为一个路由网关，需要指定 gateway=ovpn-test，在默认是动态接口状态下，连接中断后，gateway 参数将丢失，需要配置静态接口永久存在。
- 动态接口，即 OVPN 客户端连接成功后，默认会自动添加到 interface 接口列表中，前缀为”D”

动态接口出现在一个用户建立连接，中断后该用户接口将自动消失，因此无法引用到防火墙、路由等接口配置中，需要为这些配置提供静态接口

注意： PPP users 菜单下必须配置相关属性，静态接口参数无法代替 PPP 配置。

OVPN 服务器配置

操作路径：/interface ovpn-server server

属性描述

属性	描述
auth (*sha1* | *md5*; 默认: **sha1,md5**)	设置服务器的验证模式
certificate (*name* | *none*; 默认: **none**)	OVPN 服务器使用的证书名称

cipher (*aes128* | *aes192* | *aes256* | *blowfish128*; 默认:**aes128,blowfish128**)	设置加密类型
default-profile (*name*; 默认: **default**)	设置服务器的策略
enabled (*yes* | *no*; 默认: **no**)	设置 OVPN 服务器是否启用
keepalive-timeout (*整型* | *disabled*; 默认: **60**)	定义 keepalive 时间，路由器每秒会发生 keepalive 包，如果没有传输连接或 keepalive 响应在指定的时间范围内，客户端将会被中断连接
mac-address (*MAC*; 默认:)	自动生成服务器 MAC 地址
max-mtu (*整型*; 默认: **1500**)	最大传输单元
mode (*ip* | *ethernet*; 默认: **ip**)	设置二层或三层隧道模式
netmask (*整型*; 默认: **24**)	应用到客户端的子网掩码
port (*整型*; 默认: **1194**)	服务器运行的端口
require-client-certificate (*yes* | *no*; 默认: **no**)	如果设置为 yes，这时服务器会检查客户端的证书是否属于相同证书链

下面实例为启用 OVPN 服务器，并设置证书为 server

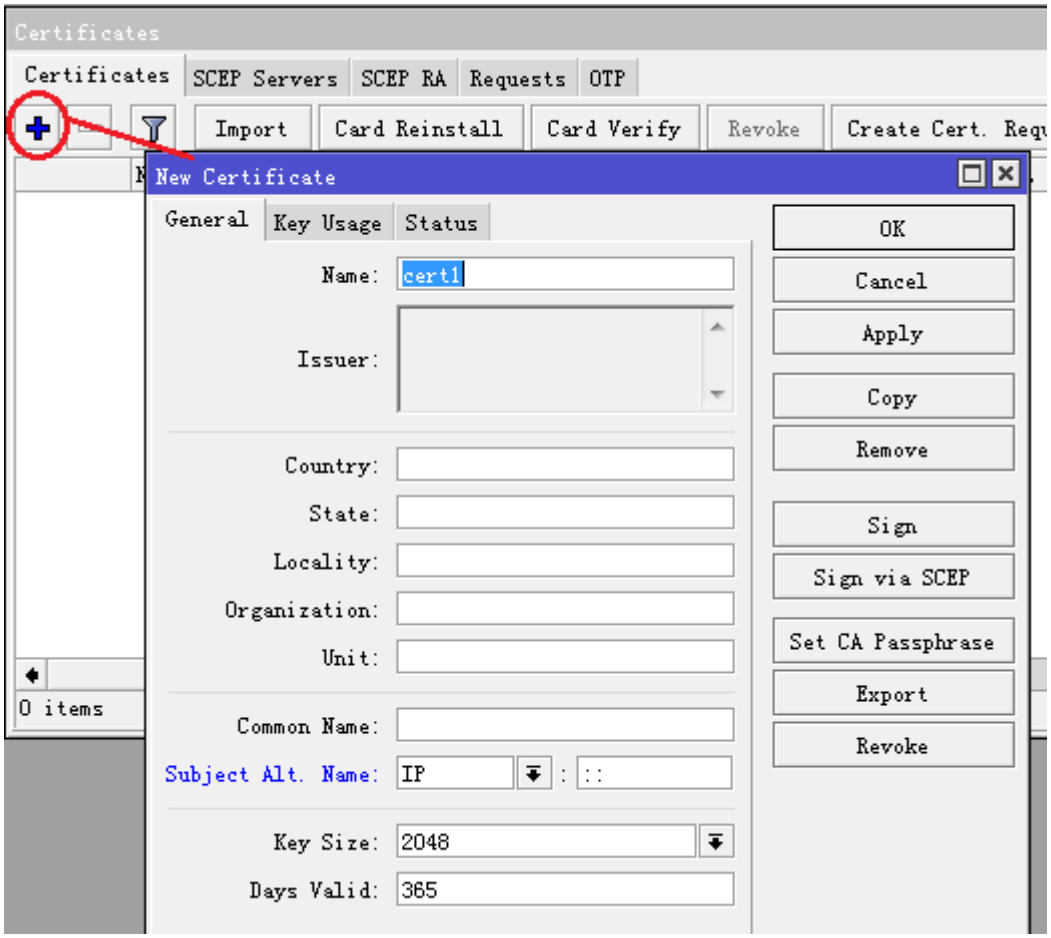
```
[admin@MikroTik] /interface ovpn-server server set enabled=yes
[admin@MikroTik] /interface ovpn-server server set certificate=server
[admin@MikroTik] /interface ovpn-server server print
                    enabled: yes
                       port: 1194
                       mode: ip
                    netmask: 24
                mac-address: FE:A5:57:72:9D:EC
                    max-mtu: 1500
          keepalive-timeout: 60
            default-profile: default
                certificate: server
 require-client-certificate: no
                       auth: sha1,md5
                     cipher: blowfish128,aes128
```

警告：路由器的时间日期非常重要，因为每份证书都有到期日，导入安装证书后，使用证书需要在规定的时间范围，如果路由器时间错误，将导致证书无法使用，建议开启 NTP 网络对时在服务器和客户端，保证时间正确。

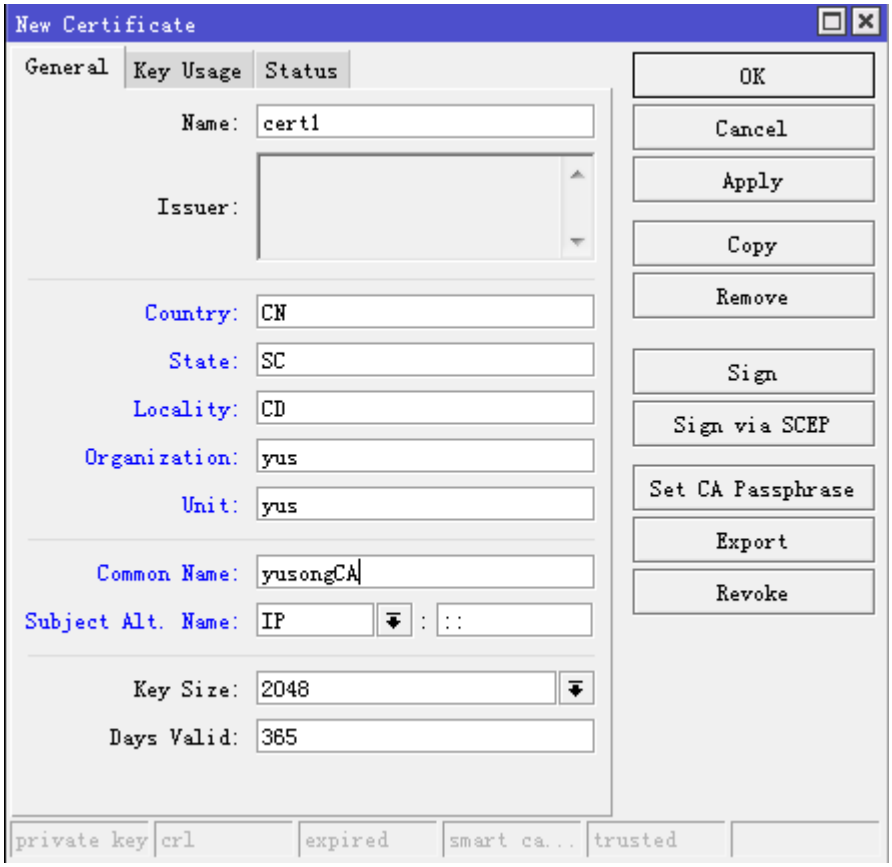
24.3 使用 certificate 创建 OVPN 证书

RouterOS v6 版本允许使用 Certificate 创建、存储和管理证书。命令行路径为/certificate，winbox 则需要进入 system->certificates。下面介绍使用 Certificate 为 OVPN 创建自己的证书。

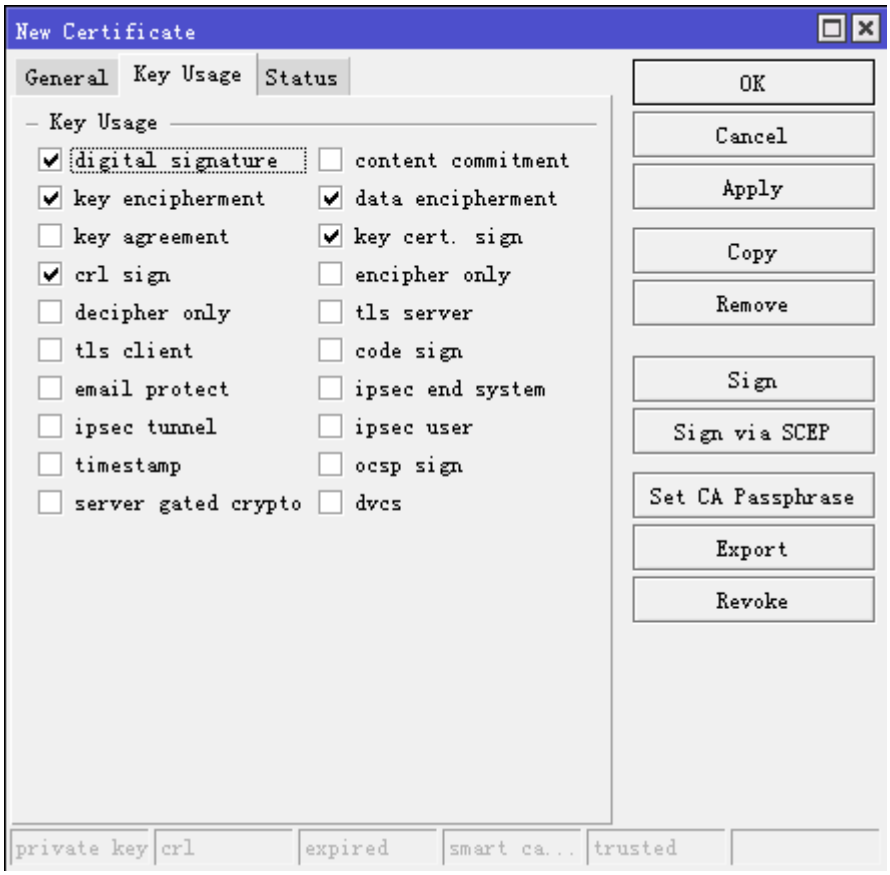
这里我们通过 CLI 命令行和 winbox 介绍，winbox 进入 system->certificates 后，点加号添加证书



默认取名为 cert1，然后设置自己证书的相关信息，大致内容如下，其他参数预设，Key-size 设置密钥的长度为 2048（Key 长度可以根据设备性能，如果处理性能较低可以选择 1024 长度），Days Valid 为有效日期长度，默认为 365 天。



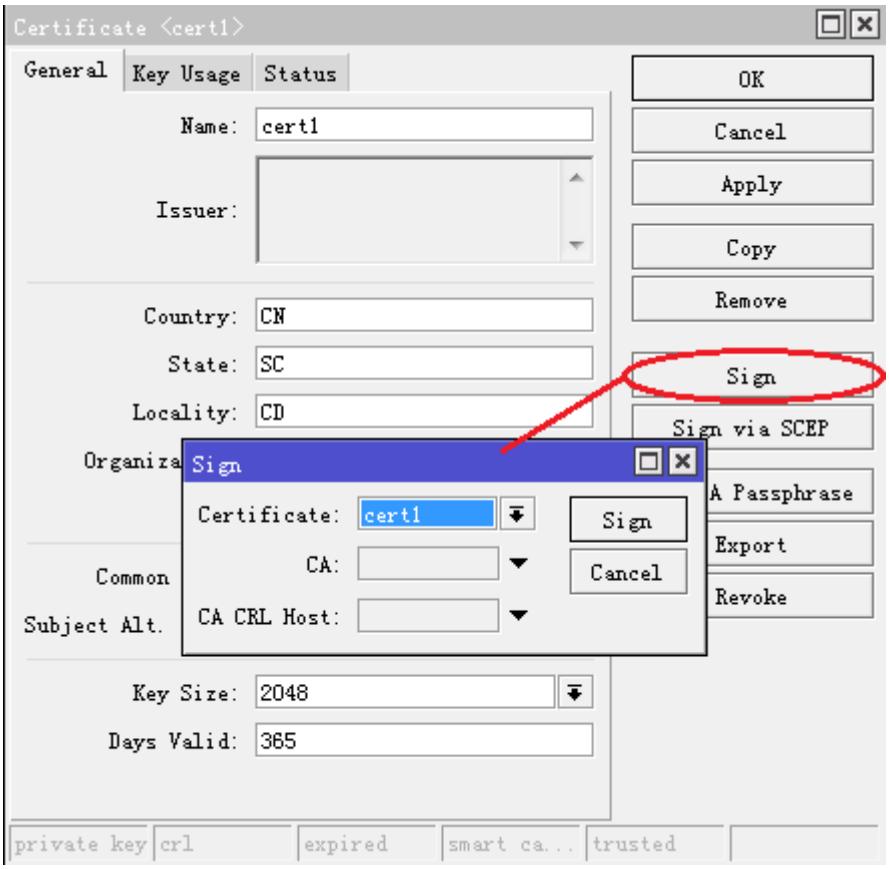
在 Key Usage 下选择默认即可，然后点击 OK 创建：



RouterOS 在 CLI 命令行添加证书操作如下：

```
[admin@MikroTik] /certificate> add name=cert1 country=CN state=SC locality=CD organization=yus
unit="yus" common-name="yusongCA"
[admin@MikroTik] /certificate> print detail
Flags: K - private-key, D - dsa, L - crl, C - smart-card-key, A - authority,
I - issued, R - revoked, E - expired, T - trusted
 0            name="cert1" country="CN" state="SC" locality="CD" organization="yus"
              unit="yus" common-name="yusongCA" key-size=2048 days-valid=365
              key-usage=digital-signature,key-encipherment,data-encipherment,key-
          cert-sign,crl-sign
```

添加证书后，需要签发证书，在 Certificate 列表中打开刚才添加到证书，点击 Sign 项，签发证书。弹出签发对话框选择 certificate 为 cert1，点击 Sign 签发

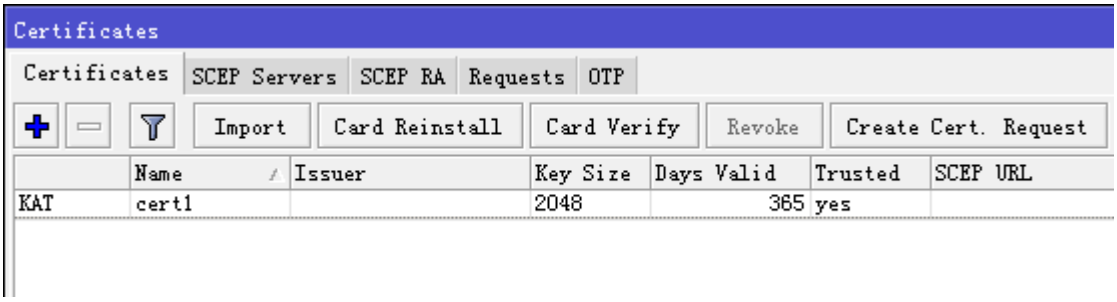


CLI 命令行签发操作如下：

```
[admin@MikroTik] /certificate> sign cert1
```

注意： 如果签发证书在 MIPS CPU 的设备（RB7xx,RB2011,RB9xx），根据指定的 key 长度大小，需花费一段时间处理。

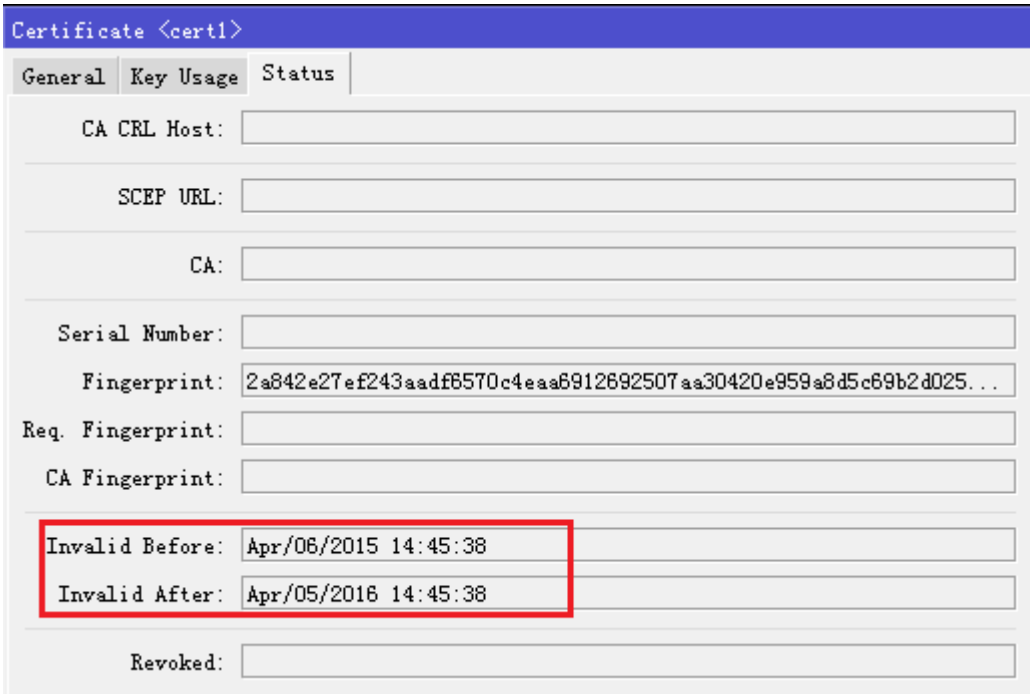
签发成功后，会显示 KAT 的前缀，



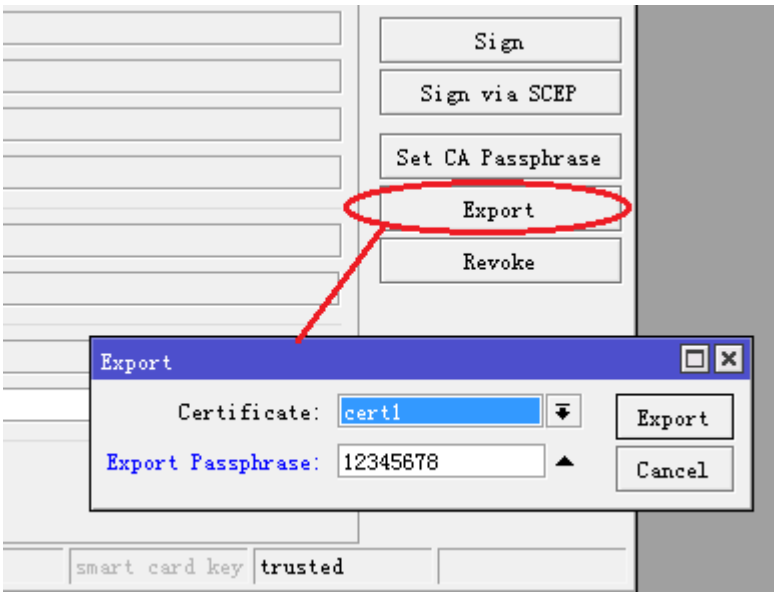
可以看到 Trusted 项为 yes，如果 Trusted 为 no 则代表不信任证书，创建证书后，我们可以通过 trusted 参数设置证书的信任

```
[admin@MikroTik] /certificate> set 0 trusted=yes
[admin@MikroTik] /certificate> print
Flags: K - private-key, D - dsa, L - crl, C - smart-card-key, A - authority,
I - issued, R - revoked, E - expired, T - trusted
 #            NAME    COMMON-NAME   SUBJECT-ALT-NAME       FINGERPRINT
 0 K   A   T  cert1       yusongCA
```

可以查看证书的状态，有效期从 2015 年 4 月 6 日-2016 年 4 月 5 日，导入路由器的时间日期需要正确配置



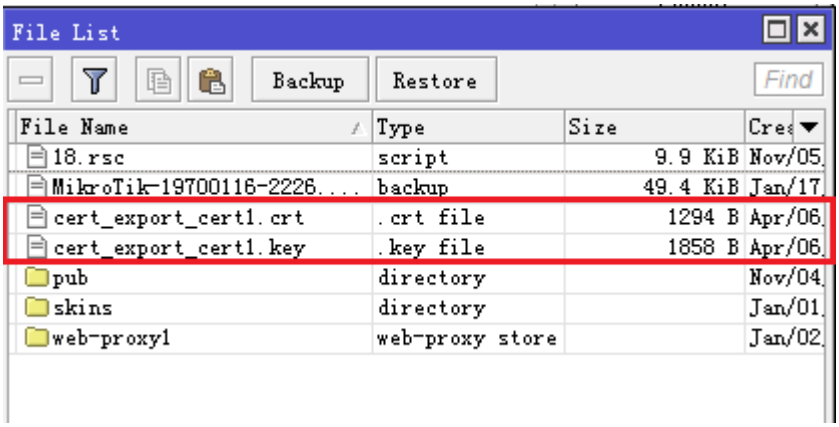
这样证书生成完成，可以使用 export 导出证书，并设置密码为 12345678



命令行操作如下：

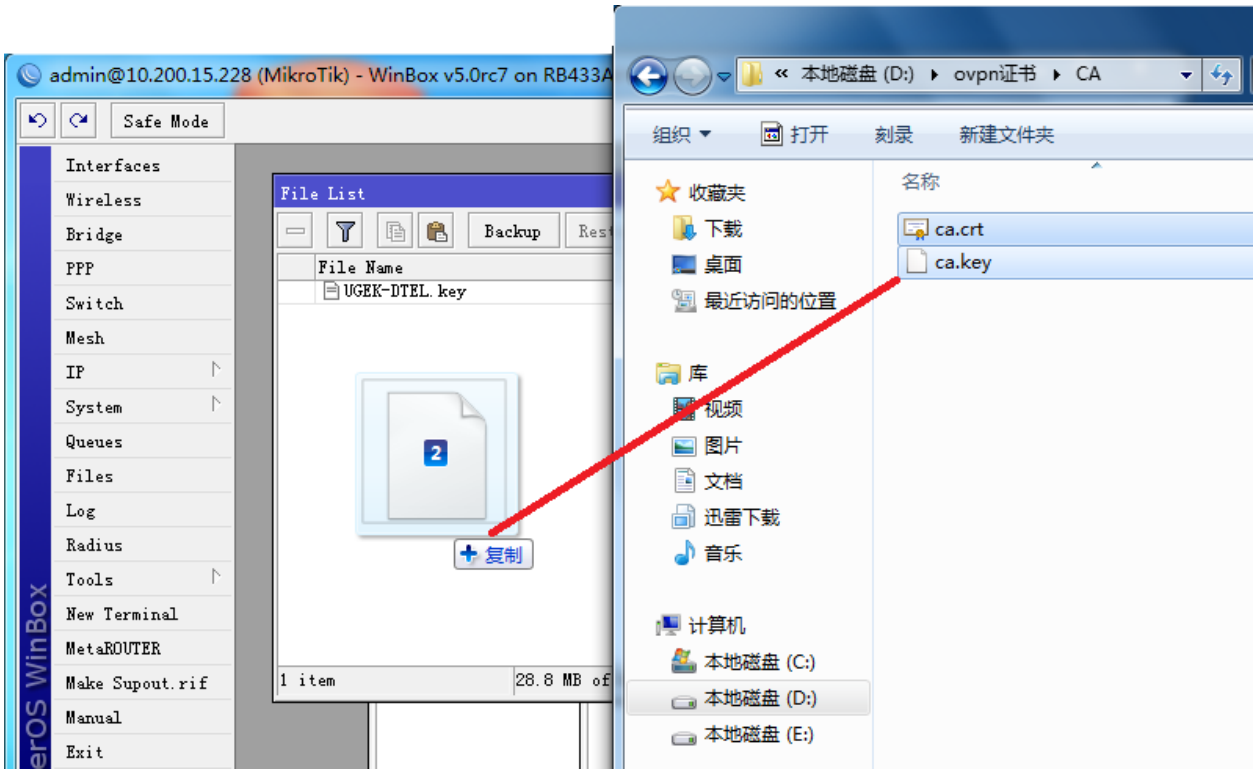
```
[admin@MikroTik] /certificate> export-certificate 0 export-passphrase=12345678
```

在 file 列表下，可以找到汇出的证书 crt 和 key 两个档，然后下载并上传到其他 OVPN 设备

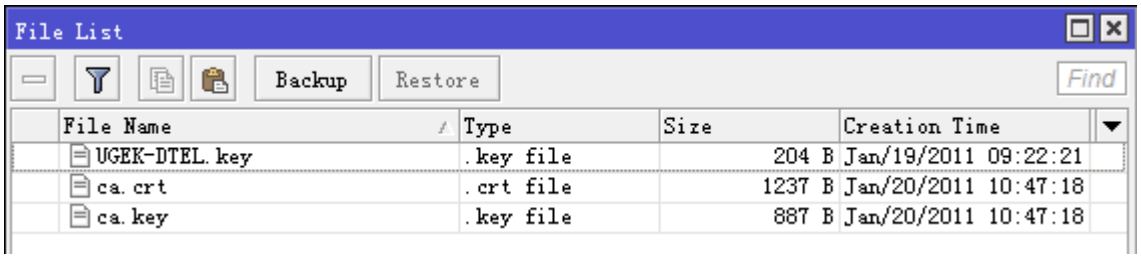


24.4 OVPN 配置

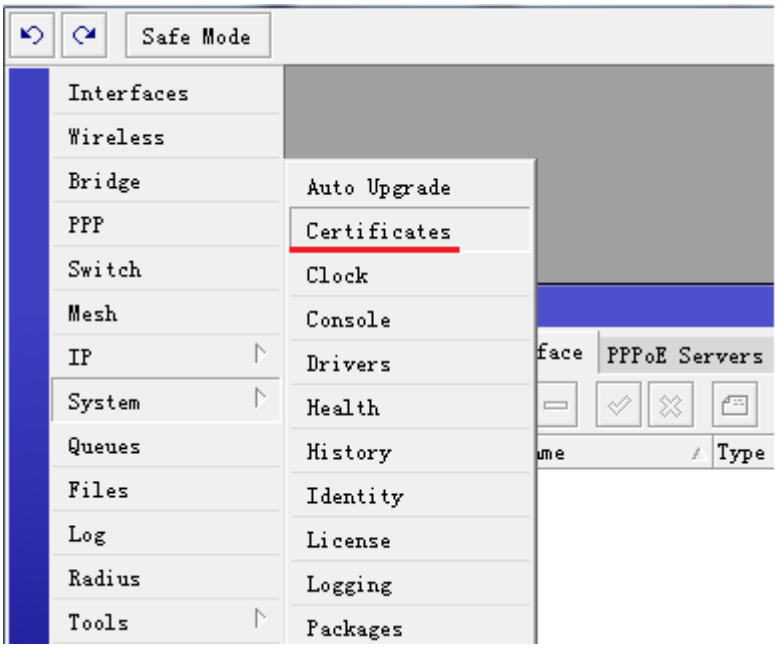
OVPN 服务器配置，首先要导入证书，否则 OVPN 与客户端是无法连接的，我们可以从网上下载已有的证书或者自己通过已经安装 OVPN 的 linux 系统生成新的证书，这里我们根据已有的证书进行操作，如下图：

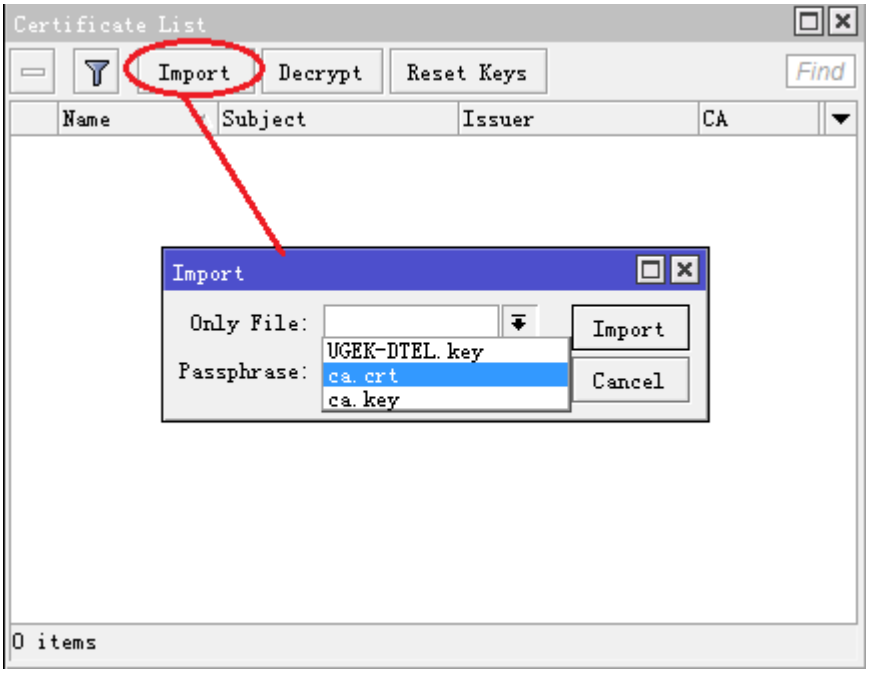


导入完成后，在 file list 可以看到证书

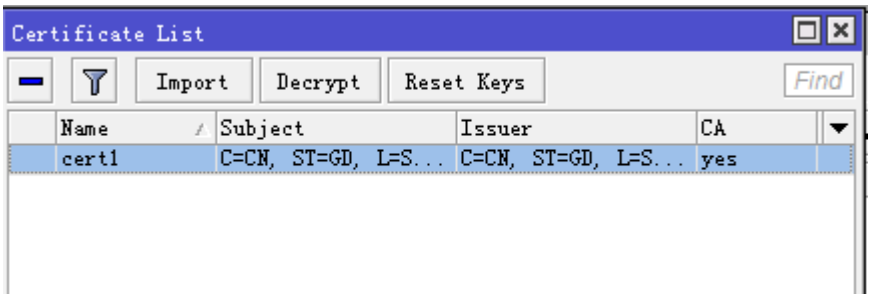


选择 system certificates 证书，并导入 crt 和 key 文件

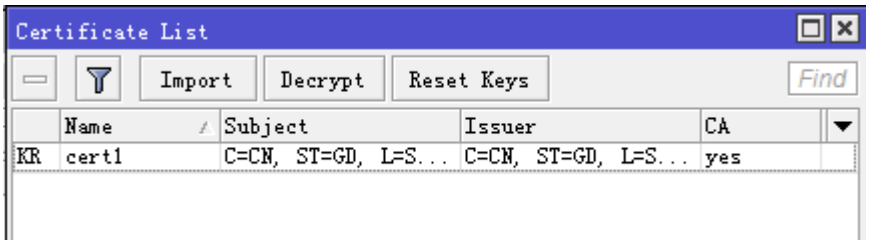




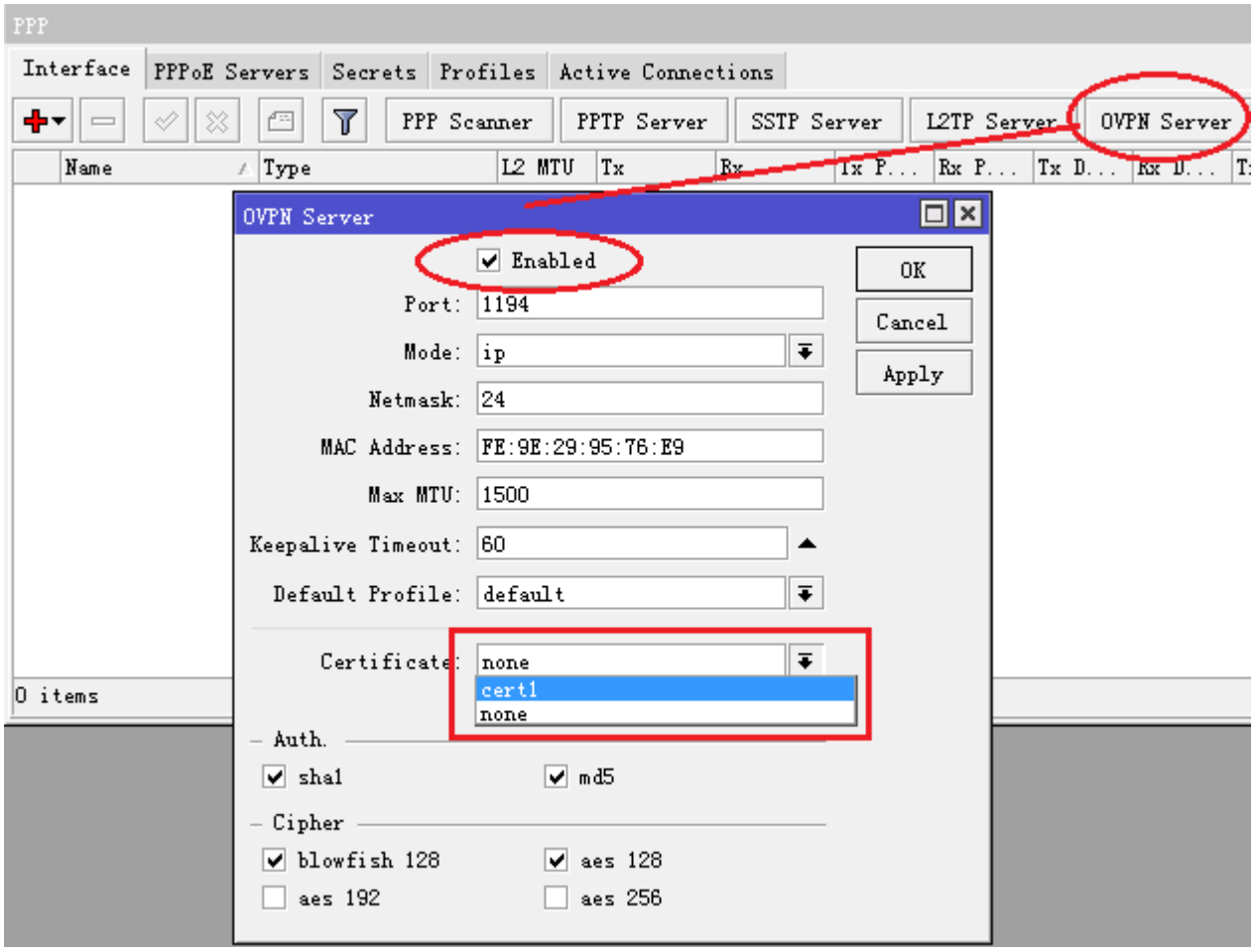
点击 import 导入 ca.crt 文件



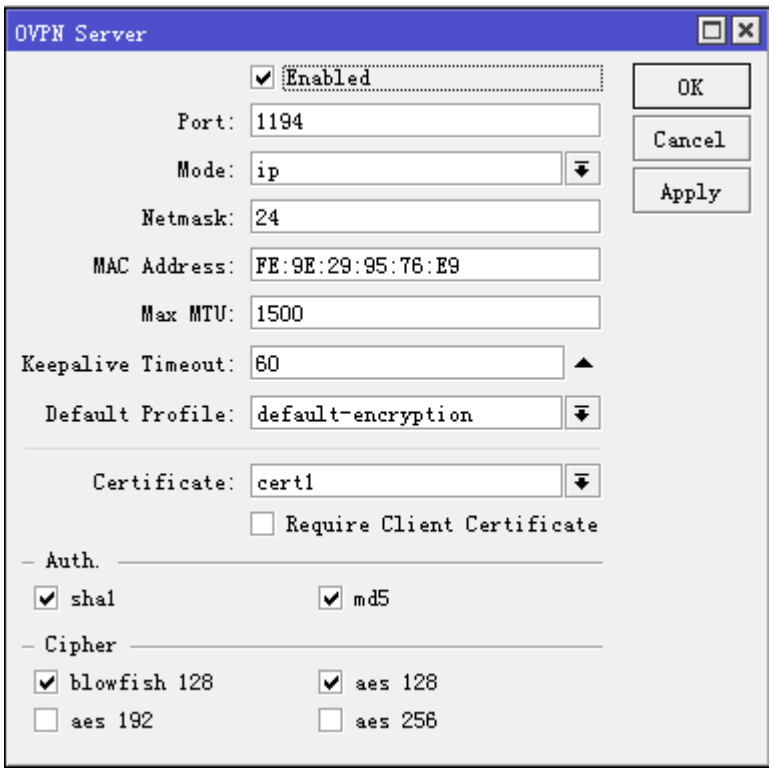
用同样的方式，导入 ca.key 文件，注意 ca.key 如果设置密码，需要在 Passphrase 选项输入密码，在当前目录下的项目出现了 KR 的前缀，表示已经导入 key 并运行



剩下的步骤就是启用 OVPN 服务，选择 enable 启用 OVPN 服务，并选择 certificate 为 cer1，其他参数默认配置，当然端口你可以自己选择，这里默认是 1194

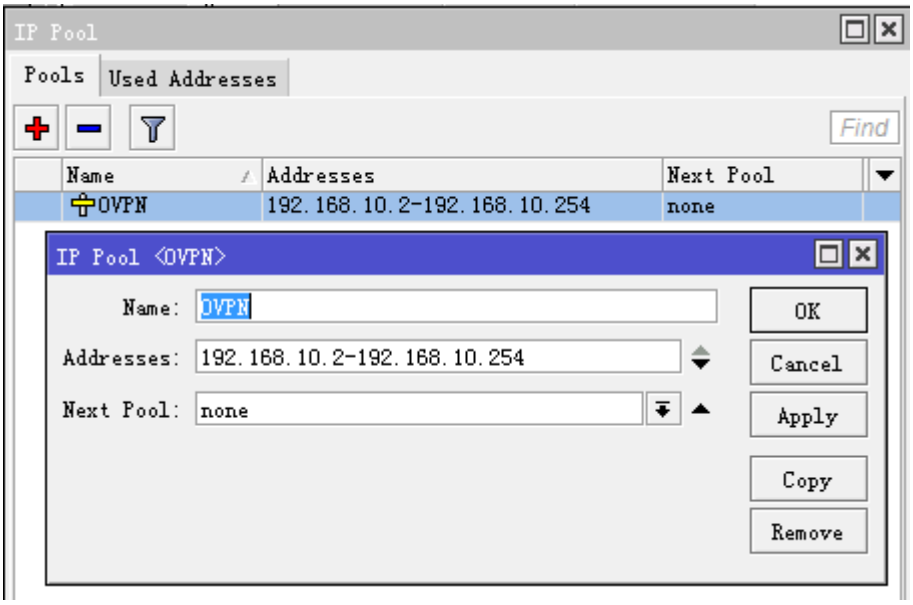


配置完成如下：

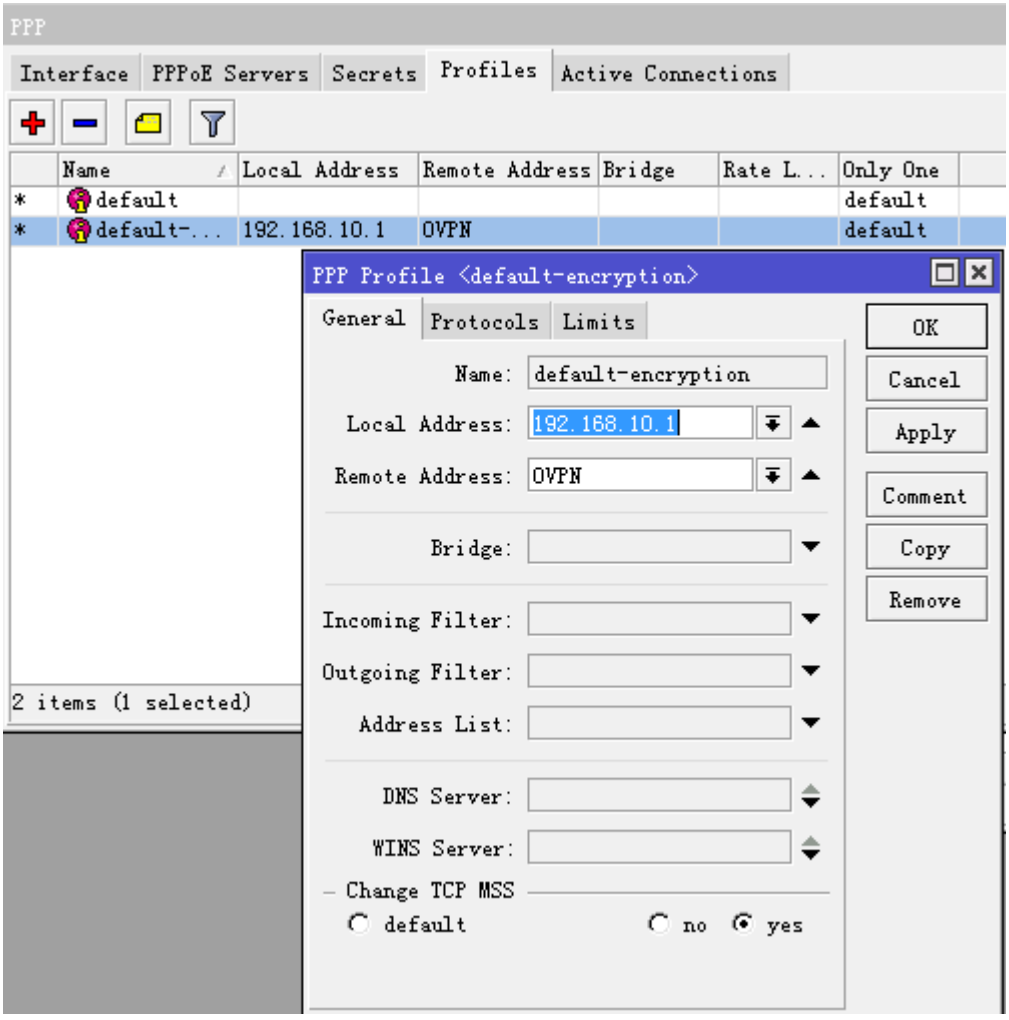


注： require-client-certificate 这里如果选择上了，客户端同样也要导入相同的证书

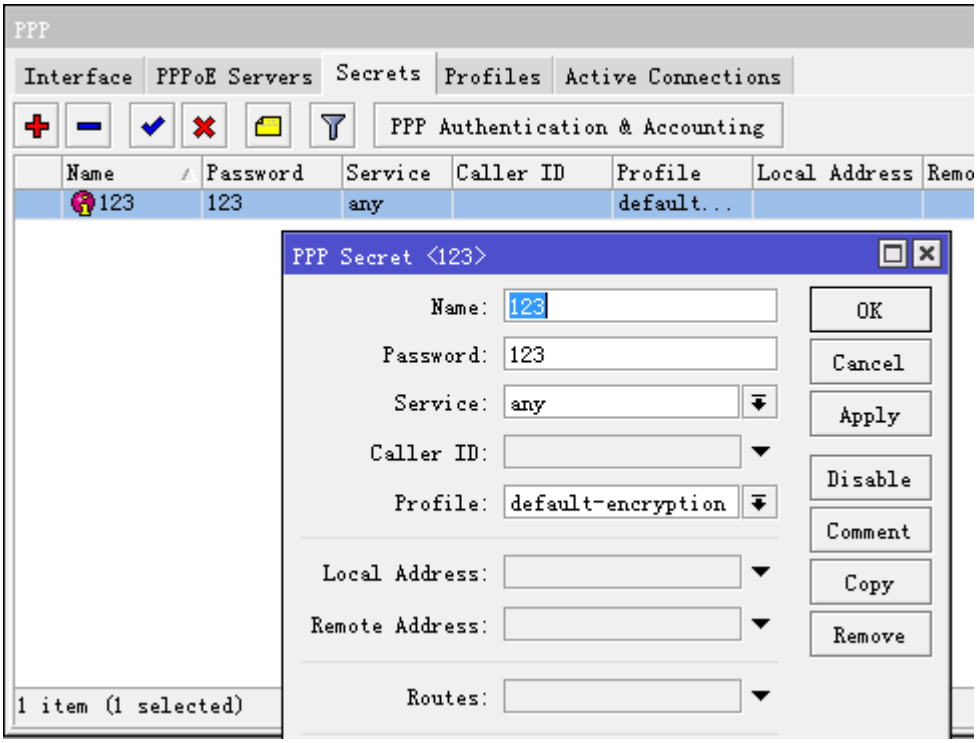
OVPN 服务器配置完成后，配置相应的规则，需要配置地址池 192.168.10.2-192.168.10.254，用于分配给用户的 IP 地址



进入 ppp profile 设置规则 local-address=192.168.10.1 和 remote-address=OVPN

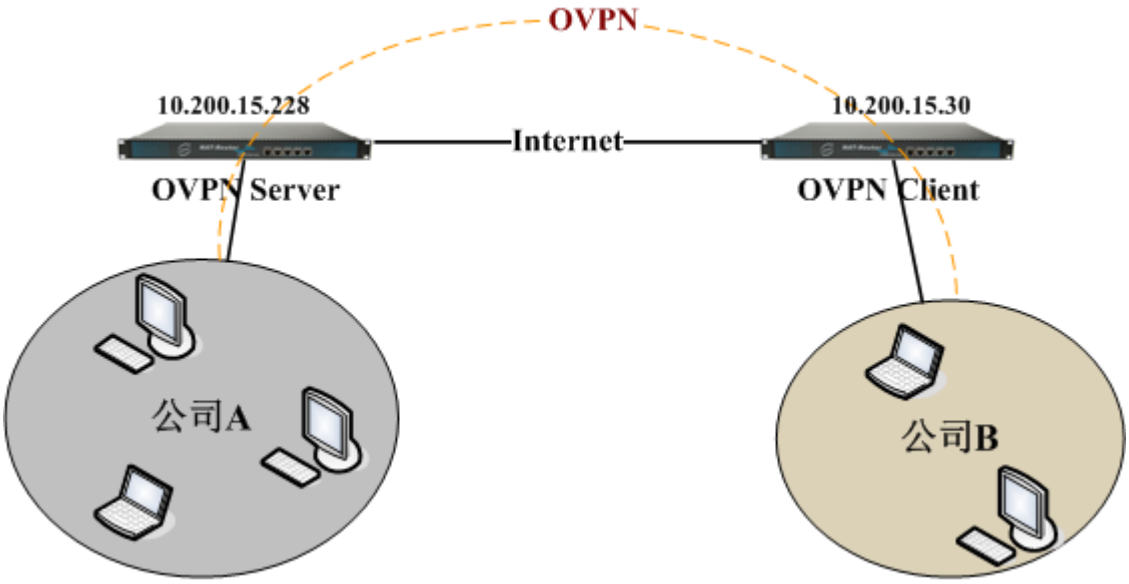


进入 ppp secret 添加用户账号



到此 OVPN 服务器配置完成

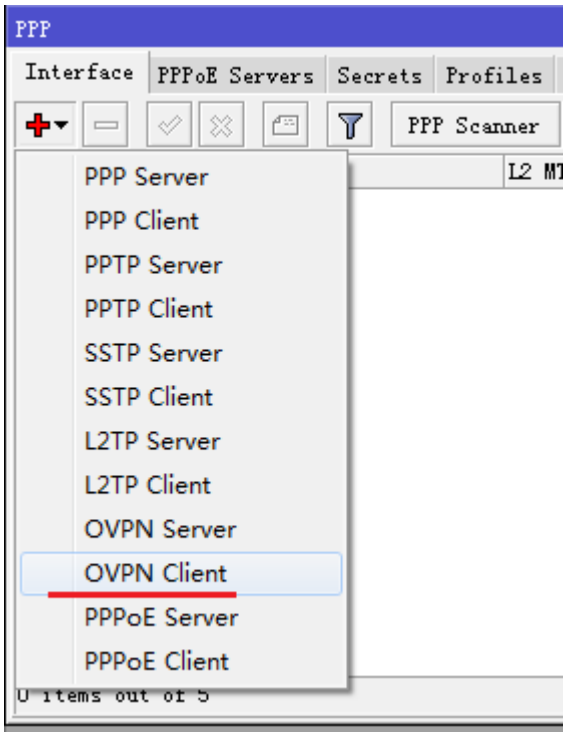
接下来我们需要在另外一台路由器配置 OVPN 客户端，如下面的拓扑图，将 2 台路由建立 OVPN 连接：



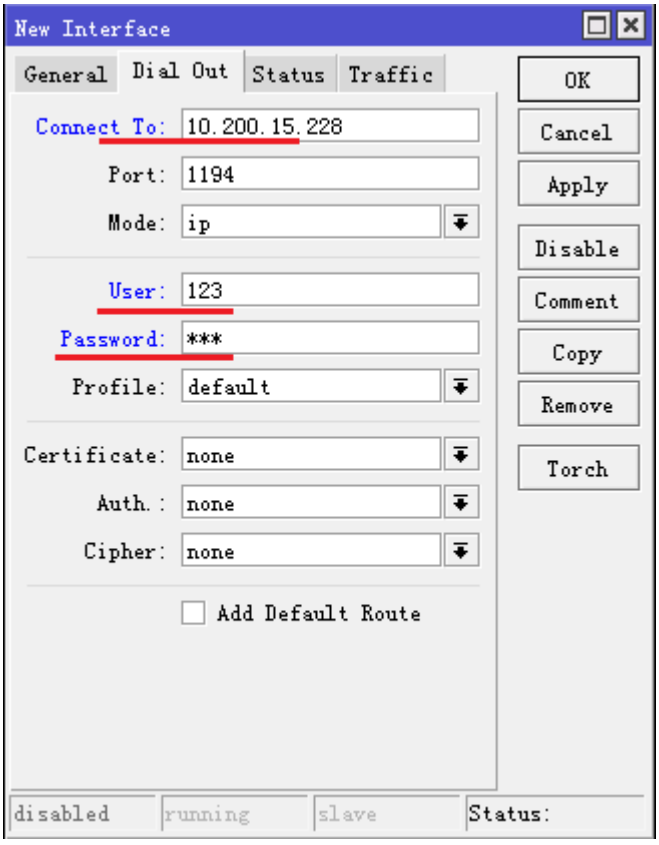
根据以上拓扑图我们假设以下网络环境：

OVPN Server 的 WAN 地址是 10.200.15.228，
OVPN Client 的 WAN 地址是 10.200.15.30

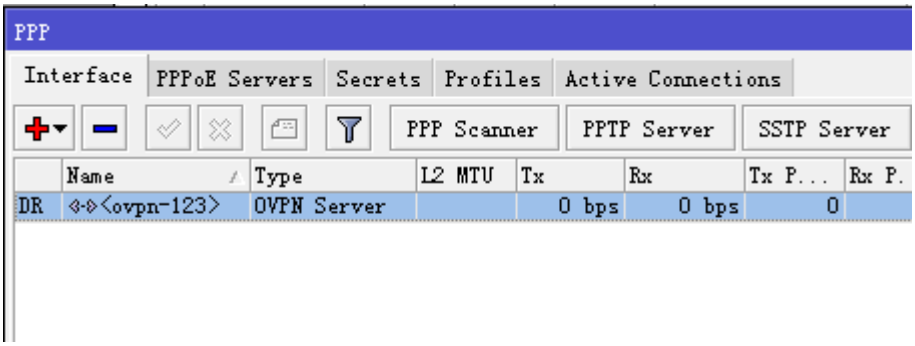
通过这两个 WAN 地址建立 OVPN 的互联，我们在 OVPN Client 建立 ovpn-client 拨号，打开 PPP， 在 interface 里点加号，选择 ovpn-client



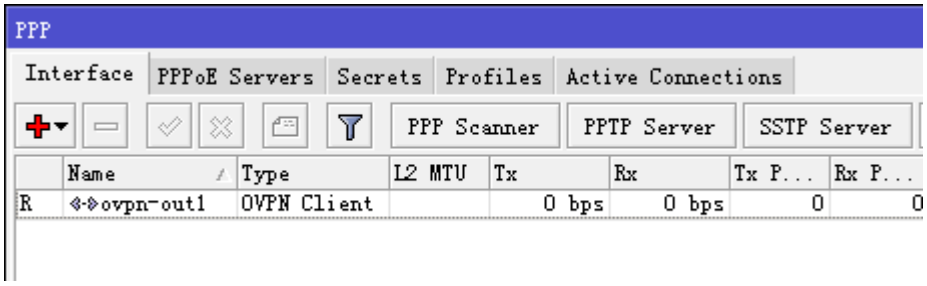
打开后，选择 Dial-out，设置 Connect-to=10.200.15.228，user=123，password=123，其他参数默认



确定后，连接 OVPN 服务器，服务器连接状态如下，显示 DR 前缀

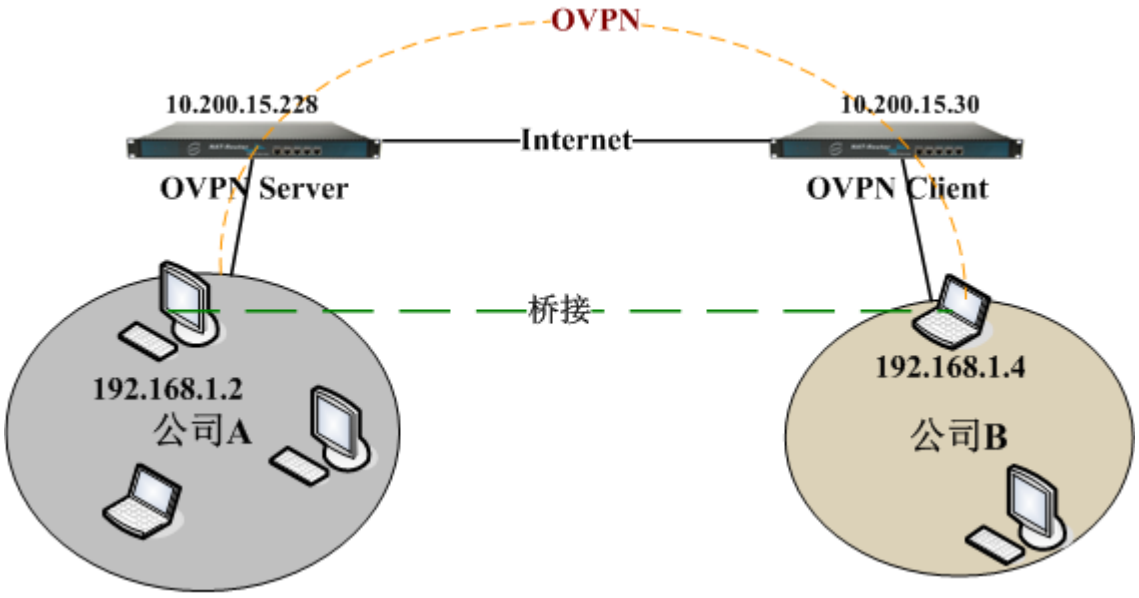


OVPN Client 显示前缀 R，表示连接成功

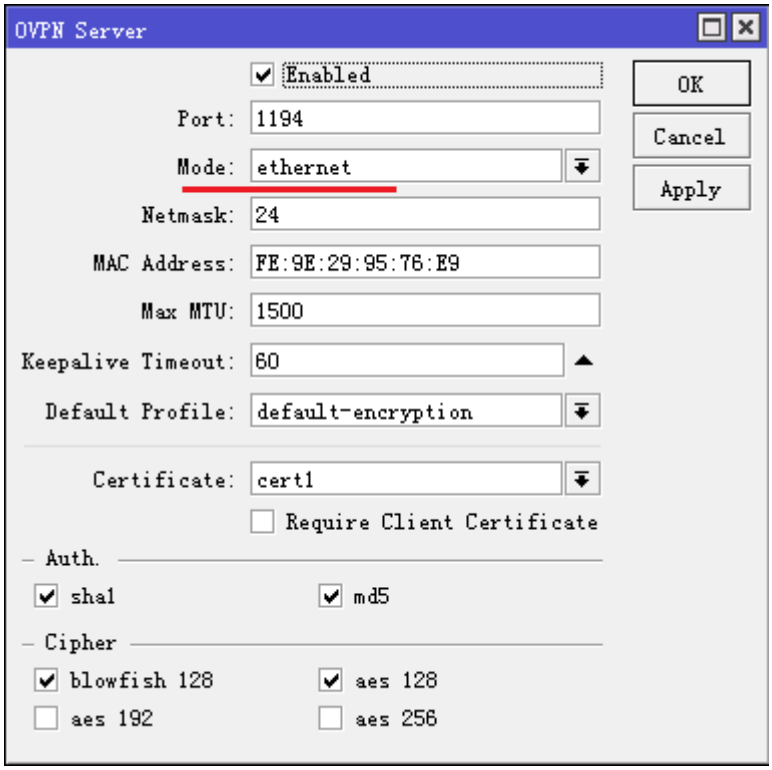


24.5 OVPN bridge 模式

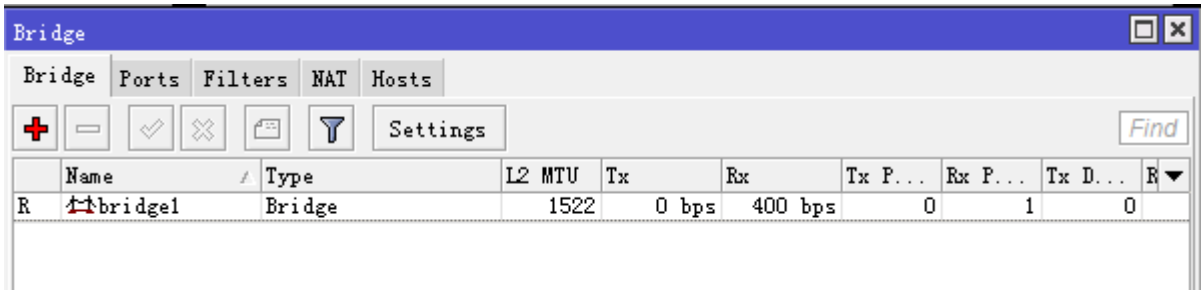
通过 RouterOS 提供的隧道 bridge 功能和 OVPN 的 ethernet 模式实现将两个远程网络建立二层的隧道透传连接，我们依然用之前的网络结构，如下图：



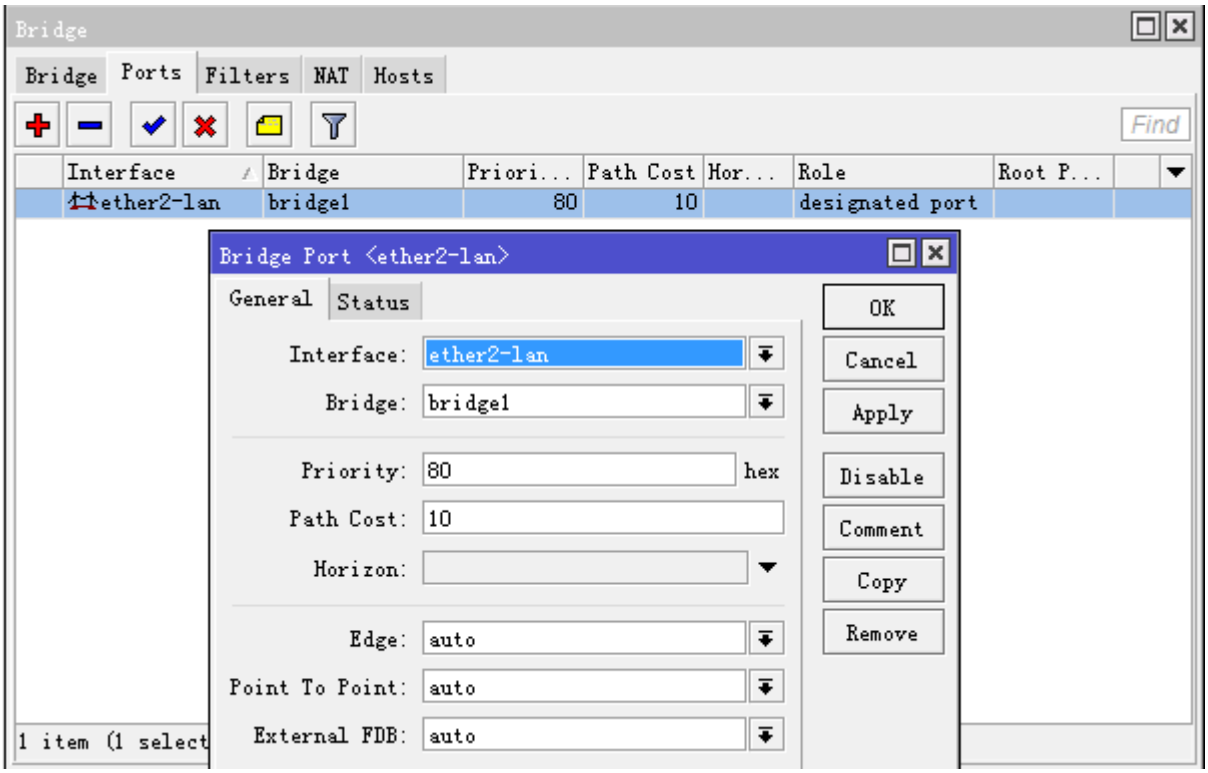
这里我们需要将 OVPN Server 和 OVPN Client 的模式修改为 ethernet



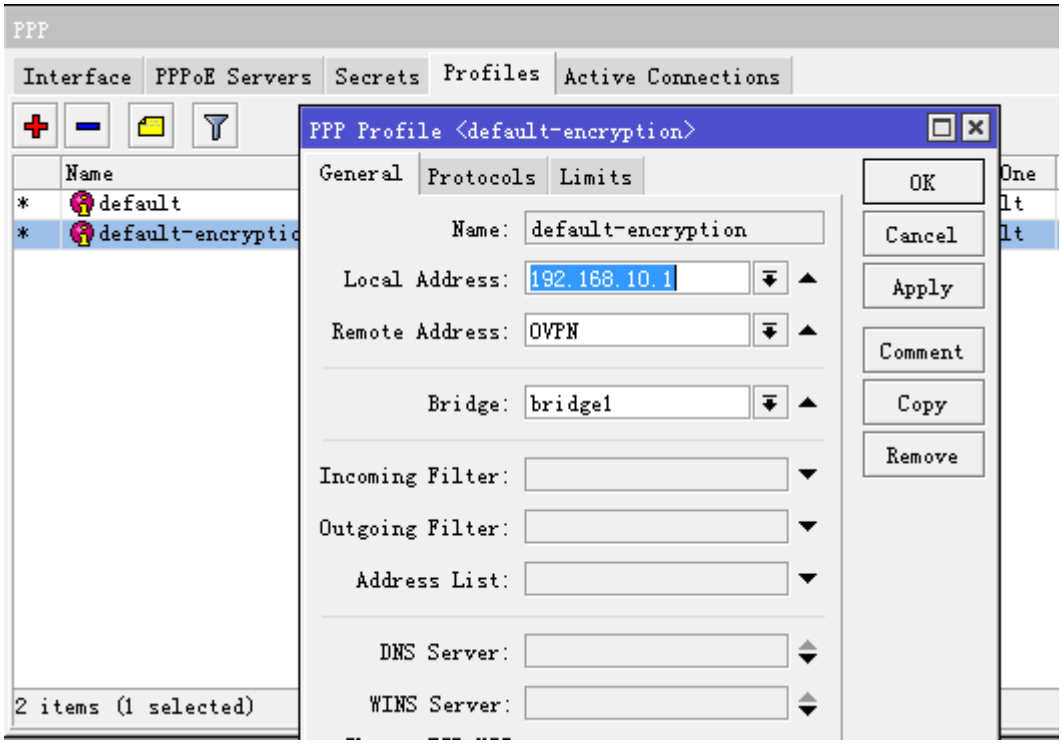
在 bridge 中添加一个桥配置



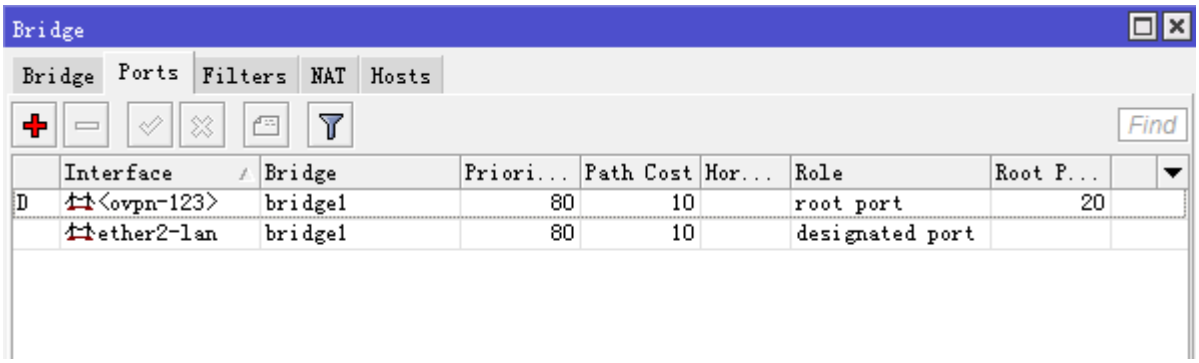
进入 ports 选择，将 ether2-lan 口添加入桥里



回到 ppp profile 里设置 default-encryption 的 bridge 选项设置为刚才我们添加的 bridge1



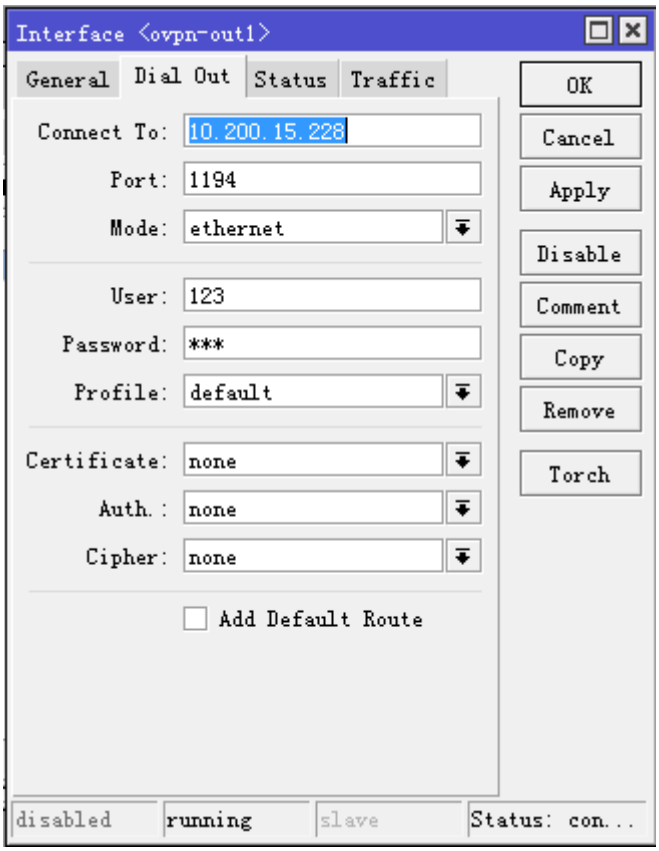
设置完成后，我们建立 OVPN Client 与 OVPN Server 的连接，在 bridge 的 port 里，我们可以看到 123 账号的连接在 port 被自动添加



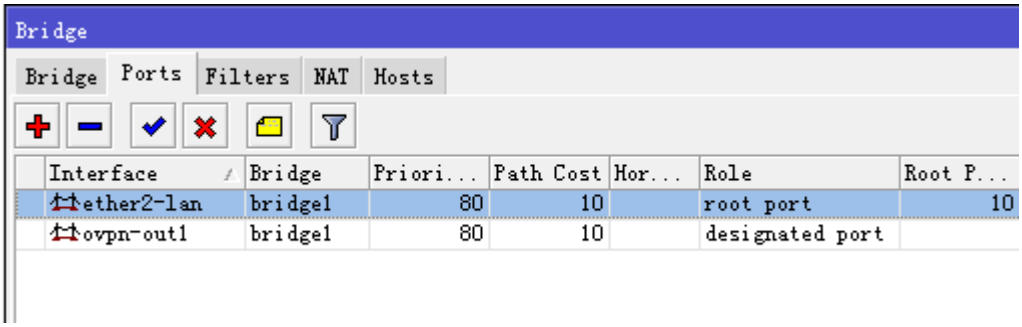
这样 OVPN Server 的 ether2-lan 内网口和 ovpn-123 隧道在同一个桥里

OVPN Client 设置

修改 ovpn-out1 的 mode 为 ethernet 模式



在 OVPN Client 端同样配置桥接，进入 bridge 后添加一个桥取名为 bridge1，并在 port 里将内网网卡 ether2-lan 和， ovpn-out1 接口手动添加到 bridge1 里，如下图



这样的配置，将两个远程的网络桥接在一个广播域内，即大家都在一个局域网内。

注： 在做 nat 伪装规则时，需要避免桥接隧道被转换，需要设置 nat 规则（包括 Server 和 Client 路由器的 nat 规则）

```
/ip firewall nat add out-interface=ether1-wan action=masquerade
```

第二十五章 SSTP 介绍

Secure Socket Tunneling Protocol (SSTP)方式是基于 SSL3.0 通道传输 PPP 隧道，使用 TCP 端口 443 的 SSL 允许 SSTP 通过所有防火墙和代理服务器。

新的 SSTP 协议的支持，并没有完全否决 PPTP 及 L2TP 在微软产品所组成的解决方案中的作用，当企业使用基于 PPTP 和 L2TP 的 VPN 解决方案时，这种协议仍是被常用来解决或是提升企业网络安全性。但两者的数据报通过防火墙、NAT、WEB PROXY 时却都有可能发生一些联机方面的问题。

PPTP 数据报通过防火墙时，防火墙需被设定成同时充许 TCP 连接以及 GRE 封装的数据通过，但大部分 ISP 都会阻止这种封包，从而造成联机的问题;而当你的机器位于 NAT 之后，NAT 亦必需被设定成能转发 GRE 协议封装的数据报。否则就会造成只能建立 PPTP 的 TCP 连接，而无法接收 GRE 协议封装的数据报;WEB PROXY 是不支持 PPTP 协议的。

L2TP OVER IPSEC 的情况和此类似，需要在防火墙上充许 IKE 数据和 ESP 封装的数据同时通过，否则也会出现连接问题。且 WEB PROXY 也是不支持 L2TP OVER IPSEC 协议。

因此 SSTP 在透传方面优于前两种 VPN，新的 VPN 隧道工作在 SSL3.0 通道，因此可以绕过任何的过滤器和代理，因为 SSTP 工作在 TCP 的 443 端口，任何过滤器都会看作正常的传输并通过。

注意：SSTP 的 login、logout 和 SSL 证书处理，各自的任务只能负载到一个 CPU 核心，无论是 RouterOSv6 和 v7 都是同样情况，因此在 RouterOS 大规模部署 SSTP 服务时需要注意单机的 CPU 核心负载问题。

SSTP 客户端

操作路径： /interface sstp-client

add-default-route (*yes* | *no*; 默认: **no**)是否添加 SSTP 远程地址的默认路由
authentication (*mschap2* | *mschap1* | *chap* | *pap*; 默认: **mschap2, mschap1, chap, pap**)启用验证方试
certificate (*字符* | *none*; 默认: **none**) 证书
comment (*字符*; 默认: "")注释
connect-to (*IP:Port*; 默认: **0.0.0.0:443**)远程 SSTP 服务器地址
dial-on-demand (*yes* | *no*; 默认: **no**)根据需求拨号
disabled (*yes* | *no*; 默认: **yes**)是否禁用该接口，默认是被禁用
keepalive-timeout (整型 | *禁用*; 默认: **60**)
max-mru (*整型*; 默认: **1500**)最大接收单位
max-mtu (整型; 默认: **1500**)最大传输单位
mrru (*disabled* | *整型*; 默认: **disabled**)在连接被认可的最大数据报长度。如果一个数据大于隧道的 MTU 值，将会被拆分多个数据报，允许最大长度的 IP 或者以太网数据报发送到隧道
name (*字符*; 默认:)说明接口名称
password (字符; 默认: **""**)用于验证的密码
profile (*name*; Default: **default-encryption**)被使用的 PPP profile
proxy (*IP:Port*; 默认: **0.0.0.0:443**)HTTP 代理服务的地址和端口
user (字符; 默认: "")用于验证的账号名

这个事例示范如何设置 SSTP 客户端，连接服务器 10.1.101.1，用户名"sstp-test"，密码"123"

```
[admin@MikroTik]    /interface sstp-client>add user=sstp-test password=123 \
\... connect-to=10.1.101.1 disabled=no
[admin@MikroTik] /interface sstp-client> print
Flags: X - disabled, R - running
 0    R name="sstp-out1" max-mtu=1500 max-mru=1500 mrru=disabled connect-to=10.1.101.1:443
        user="sstp-test" password="123" proxy=0.0.0.0:443 profile=default
        certificate=none keepalive-timeout=60 add-default-route=no dial-on-demand=no
        authentication=pap,chap,mschap1,mschap2
```

SSTP 服务器

操作路径: /interface sstp-server

这路径显示接口为每个已连接的 SSTP 客户端 ，创建一个接口是为建立到指定服务器的每条隧道，这里在 PPTP 服务器的配置有两种类型动态和静态接口，这点和之前 OVPN 提到两种类型情况一样请参见 25.2 章节。

服务器配置

操作路径: /interface sstp-server server

authentication (*pap* | *chap* | *mschap1* | *mschap2*; Default: **pap,chap,mschap1,mschap2**)	服务器接收的验证模式
certificate (*name* | *none*; 默认: **none**)	SSTP 使用的证书
default-profile (*name*; 默认: **default**)	
force-aes (*yes* | *no*; 默认: **no**)	Force AES encryption (AES256 is supported). If enabled windows clients (supports only RC4) will be unable to connect.
keepalive-timeout (*integer* | *disabled*; Default: **60**)	If server during keepalive period does not receive any packet, it will send keepalive packets every second five times. If the server does not receives response from the client, then disconnect after 5 seconds. Logs will show 5x "LCP missed echo reply" messages and then disconnect.
pfs (*yes* | *no*; Default: **no**)	Enables "Perfect Forward Secrecy" which will make sure that private encryption key is generated for each session. Must be enabled on both server and client to work.
port (*integer*; Default: **443**)	Port for SSTP service to listen on.
tls-version (*any* | *only-1.2*; Default: **any**)	Specifies which TLS versions to allow
verify-client-certificate (*yes* | *no*; Default: **no**)	If set to yes, then server checks whether client's certificate belongs to the same certificate chain.

authentication (*pap* | *chap* | *mschap1* | *mschap2*; 默认: **pap,chap,mschap1,mschap2**) 服务器将使用的验证模式
certificate (*名称*; 默认: **none**)SSTP 将使用的证书的名称。必须使用证书，否则 SSTP 服务器将不能运行
default-profile (*名称*; 默认: **default**)选择 Profile 规则
enabled (*yes* | *no*; 默认: **no**)定义是否启用 SSTP 服务
keepalive-timeout (整型 | *disabled*; 默认: **60**)定义路由发送 Keepalive 数据报时钟周期（以秒为单位），如果没有数据，并且没有在指定的时钟周期响应，这些没有响应的用户将会被注销
max-mru (*整型*; 默认: **1500**)最大接收单位，
max-mtu (*整型*; 默认: **1500**)最大传输单位
mrru (*禁用* | *整型*; 默认: **disabled**)在连接被认可的最大数据报长度。如果一个数据大于隧道的 MTU 值，将会被拆分多个数据报，允许最大长度的 IP 或者以太网数据报发送到隧道
require-client-certificate (*yes* | *no*; 默认: **no**) 如果设置为 yes，这时服务器将检查客户端的证书是否属于与服务器的证书同一证书链

启用 SSTP 服务器可以选择不启用证书或启用证书，根据情况而定，下面是导入证书对 SSTP 服务器配置

```
[admin@MikroTik] /interface sstp-server server> set certificate=server
[admin@MikroTik] /interface sstp-server server> set enabled=yes
[admin@MikroTik] /interface sstp-server server> print
                                enabled: yes
                                   port: 443
                                max-mtu: 1500
                                max-mru: 1500
                                   mrru: disabled
                  keepalive-timeout: 60
                     default-profile: default
                         certificate: server
    require-client-certificate: no
                       authentication: pap,chap,mschap1,mschap2
[admin@MikroTik] /interface sstp-server server>
```

Monitor 命令能查看在客户端和服务器的隧道运行状态：

```
[admin@dzeltenais_burkaans] /interface sstp-server> monitor 0
        status: "connected"
        uptime: 17m47s
    idle-time: 17m47s
          user: "sstp-test"
    caller-id: "10.1.101.18:43886"
           mtu: 1500
```

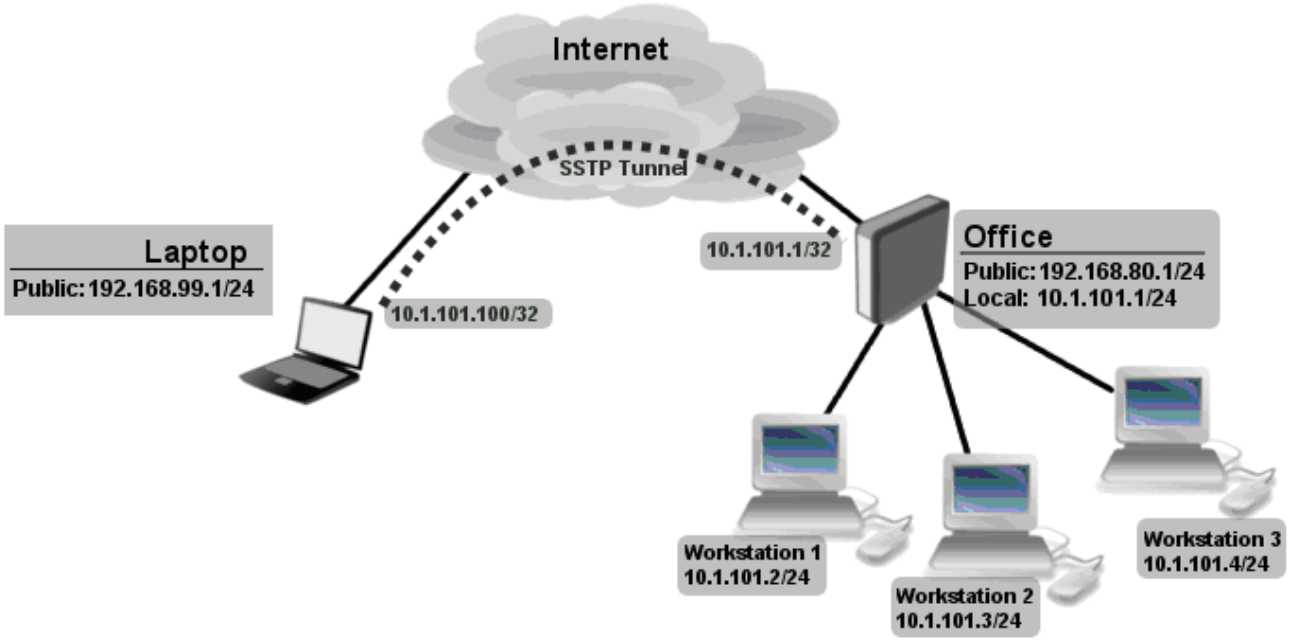
只读属性

status ()当前 SSTP 状态，值显示为“connected”表明隧道连接已经建立
uptime (*时间*)从隧道建立开始经过的时间
idle-time (*时间*)在隧道上最后一次活动经过的时间
user (*字符*)隧道建立的用户名称
mtu (*整型*)使用的 MTU

caller-id (*IP:ID*)连接者的身份，一般以 IP 地址显示

25.1 端到服务器连接

下面事例显示了如何使用 SSTP 加密隧道通过一个计算机连接到远程办公网络，SSTP 服务器分配给计算机一个相同远程办公网络的 IP 地址(不需要通过桥接的 EoIP 隧道)



Office 路由通过 ether1 连接到互联网，内网主机连接到 ether2，笔记本电脑通过互联网连接到 office 路由的公网 IP 地址 (在我们的事例中 IP 地址为 192.168.80.1).

注：在你开始配置 SSTP 前，你需要建立服务器证书，并导入路由器（证书可以通过 windows2008 生成，具体操作可以通过网上查询，证书导入方式和 OVPN 一样）

第一步、创建一个用户

```
[admin@RemoteOffice] ppp secret> add name=Laptop service=sstp password=123
local-address=10.1.101.1 remote-address=10.1.101.100
[admin@RemoteOffice] ppp secret> print detail
Flags: X - disabled
 0    name="Laptop" service=sstp caller-id="" password="123" profile=default
      local-address=10.1.101.1 remote-address=10.1.101.100 routes==""

[admin@RemoteOffice] ppp secret>
```

这里 SSTP 的 local-address 与路由器本地网卡地址相同，并且 remote-address 与本地内网属同一地址段 (10.1.101.0/24).

下一步，启用 SSTP 服务器，并在笔记本电脑建立客户端

```
[admin@RemoteOffice] /interface sstp-server server> set certificate=server
[admin@RemoteOffice] /interface sstp-server server> set enabled=yes
```

```
[admin@RemoteOffice] /interface sstp-server server> print
                                enabled: yes
                                    port: 443
                                max-mtu: 1500
                                max-mru: 1500
                                    mrru: disabled
                keepalive-timeout: 60
                   default-profile: default
                         certificate: server
   require-client-certificate: no
                    authentication: pap,chap,mschap1,mschap2

[admin@RemoteOffice] /interface sstp-server server>
```

SSTP 客户端连接到路由器的公网 IP 地址，这个事例 IP 地址是 192.168.80.1.

注意，根据你的操作系统不同配置你的 SSTP 客户端，当前支持 SSTP 的系统有 windows2008,windows vista 和 vista sp1，其他操作系统不一定支持

检查 SSTP 客户端是否连接

```
[admin@RemoteOffice] /interface sstp-server> print
Flags: X - disabled, D - dynamic, R - running
 #       NAME          USER              MTU            CLIENT-ADDRESS      UPTIME     ENCODING
 0   DR <sstp-... Laptop        1500           10.1.101.18:43886 1h47s

[admin@RemoteOffice] /interface sstp-server>monitor 0
         status: "connected"
        uptime: 1h45s
    idle-time: 1h45s
         user: "Laptop"
    caller-id: "192.168.99.1:43886"
         mtu: 1500
```

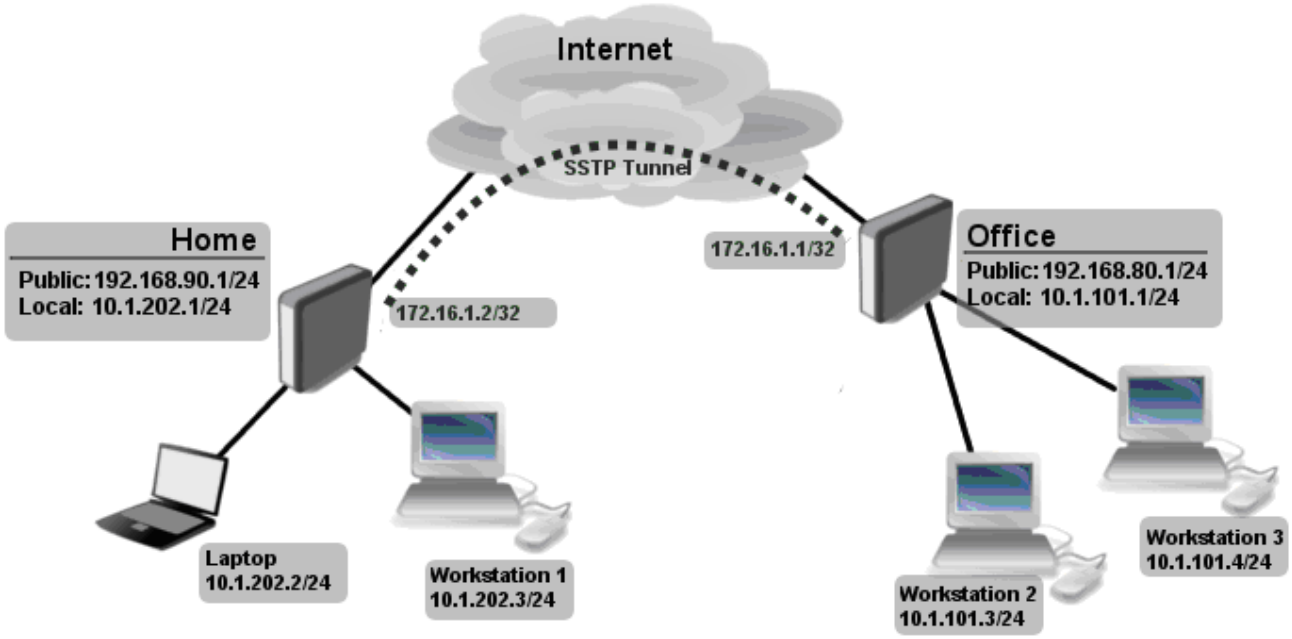
这里（当 SSTP 客户端连接成功），如果你想从笔记本电脑 ping 内网主机，将显示 timeout，因为笔记本电脑将不能从内网主机获取 ARP，解决方法是设置内网网卡的 arp 为 proxy-arp

```
[admin@RemoteOffice] /interface ethernet> set ether2 arp=proxy-arp
[admin@RemoteOffice] /interface ethernet> print
Flags: X - disabled, R - running
  #      NAME                          MTU     MAC-ADDRESS              ARP
  0   R ether1                         1500   00:30:4F:0B:7B:C1         enabled
  1   R ether2                         1500   00:30:4F:06:62:12          proxy-arp
[admin@RemoteOffice] interface ethernet>
```

在 proxy-arp 启用后，将能成功 ping 通过内网主机

25.2 点对点的 SSTP

由于 SSTP 可以修改隧道连接端口，在互联网连接上更加灵活，且 RouterOS 之间的 SSTP 连接可以不要求证书，下面一个事例两个企业内网采用基于 SSTP 隧道连接，考虑下面的网络：



Office 和 Home 路由器连接互联网通过 ether1，内网主机连接到 ether2，两个内网他们不在同一广播域，如果两个内网需要在同样广播域内，你需要使用 BCP，并桥接 SSTP 隧道与本地网卡

首先创建一个用户账号

```
[admin@RemoteOffice] /ppp secret> add name=Home service=sstp password=123
local-address=172.16.1.1 remote-address=172.16.1.2 routes="10.1.202.0/24 172.16.1.2 1"
[admin@RemoteOffice] ppp secret> print detail
Flags: X - disabled
 0    name="Home" service=sstp caller-id="" password="123" profile=default
      local-address=172.16.1.1 remote-address=172.16.1.2 routes=="10.1.101.0/24 172.16.1.1 1"

[admin@RemoteOffice] /ppp secret>
```

注：无论什么时候我们设置 SSTP 客户端添加路由，如果这个选项没有设置，你将需要指定静态路由配置在两个路由器上

接下来启用 SSTP 服务器在 Office 路由器，并在 Home 路由器配置 SSTP 客户端

```
[admin@RemoteOffice] /interface sstp-server server> set certificate=server
[admin@RemoteOffice] /interface sstp-server server> set enabled=yes
[admin@RemoteOffice] /interface sstp-server server> print
                    enabled: yes
                       port: 443
                    max-mtu: 1500
```

```
                    max-mru: 1500
                       mrru: disabled
          keepalive-timeout: 60
            default-profile: default
                certificate: server
require-client-certificate: no
             authentication: pap,chap,mschap1,mschap2
```

在 Home 路由器配置 SSTP 客户端

```
[admin@Home] /interface sstp-client> add user=Home password=123 connect-to=192.168.80.1 disabled=no
[admin@Home] /interface sstp-client> print
Flags: X - disabled, R - running
 0    R name="sstp-out1" max-mtu=1500 max-mru=1500 mrru=disabled connect-to=192.168.80.1:443
        user="Home" password="123" proxy=0.0.0.0:443 profile=default certificate=none
        keepalive-timeout=60 add-default-route=no dial-on-demand=no
        authentication=pap,chap,mschap1,mschap2
[admin@Home] /interface sstp-client>
```

现在需要添加静态路由到 Home 路由器上，用于查找在 Office 路由后的网络

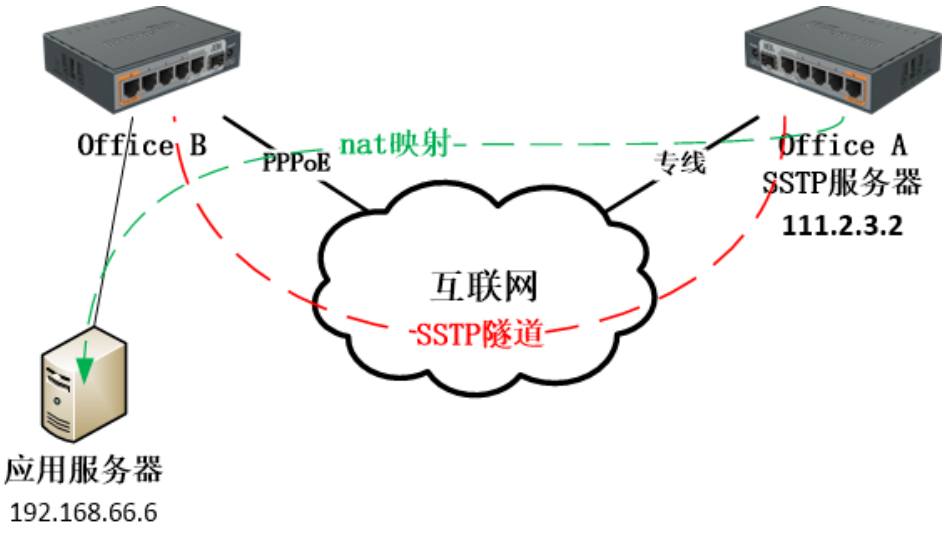
```
[admin@Home] /ip route> add dst-address=10.1.101.0/24 gateway=172.16.1.1
```

在隧道建立后你可以 ping 通远程的网络。

25.3 基于 SSTP 隧道的远程端口映射

当一个网络节点是动态公网 IP，不考虑使用 ddns 的方式，或者获取 IP 是私网地址，可以通过一个固定 IP 节点使用隧道协议中转访问。也可作为隐藏该网络节点 IP 时，通过虚拟隧道作为跳板，访问另一个端节点的内网应用，建立备份通道。

如下：



这里选择 SSTP 隧道协议，因为支持 SSTP 协议，端口可以自定义，连接较为灵活

Office A 路由器配置

首先为 Office A 路由器配置 IP 地址

```
/ip address
add address=192.168.99.1/24 interface=lan disabled=no
add address=111.2.3.2/27 interface=wan disabled=no
```

为 Office A 路由器配置默认网关

```
/ip route
add gateway=111.2.3.1 disabled=no
```

在创建 nat 规则前，在地址列表里面创建 nat 地址列表，用于上网用户的 nat 转换规则使用

```
/ip firewall address-list
add address=192.168.99.0/24 list=nat disabled=no
```

创建内网用户地址列表，用于内网用户 IP 地址段的 nat 转换

```
/ip firewall address-list
add address=192.168.99.0/24 list=nat
```

添加 nat 的伪装规则，并定义内网用户地址列表为 nat

```
/ip firewall nat
add action=masquerade chain=srcnat out-interface=wan src-address-list=nat    disabled=no
```

创建 SSTP 隧道服务，启用 sstp 服务，设置端口为 8433

```
/interface sstp-server server
set enabled=yes port=8433
```

为 Office B 路由器创建账号密码，并分配本地和远端 IP 地址

```
/ppp secret
add name=yus password=123 profile=default-encryption local-address=10.1.1.1    remote-address=10.1.1.2
service=sstp disabled=no
```

Office B 路由器配置

添加内网 ip 地址

```
/ip address
add address=192.168.66.1/24 interface=lan disabled=no
```

创建 PPPoE 拨号

```
/interface pppoe-client
add add-default-route=yes disabled=no interface=wan name=pppoe-out1 password=123 user=pppoe
```

添加 SSTP 隧道客户端规则，连接服务器

```
/interface sstp-client
add connect-to=111.2.3.2:8433 disabled=no name=sstp-office password=123 user=yus
```

此时 SSTP 隧道连接建立完成，进入 sstp-client，使用 monitor 0 查看连接状态

```
[admin@MikroTik] /interface sstp-client> monitor 0
                status: connected
                uptime: 3w6d5m24s
              encoding: AES256-CBC
                   mtu: 1500
     local-address: 10.1.1.2
    remote-address: 10.1.1.1
-- [Q quit|D dump|C-z pause]
```

Office A 的路由和 nat 配置

添加 Office A 的静态路由，从服务端能访问到客户端内网 ip 段

```
/ip route
add dst-address=192.168.66.0/24 gateway=10.1.1.2 disabled=no
```

在 nat 地址列表中添加 10.1.1.2 的客户端地址，做 nat 转换

```
/ip firewall address-list
add address=10.1.1.1 list=nat
```

Office A 路由器做上 sstp-office 接口的伪装，在访问内部应用时伪装为 sstp-office 的接口 IP 地址

```
/ip firewall nat
add action=masquerade chain=srcnat out-interface=sstp-office
```

创建端口映射规则，可以使用不对称端口做映射，访问从 Office A 路由的 9900 端口访问 Office B 内部应用服务器 192.168.66.6 的 80 端口

```
/ip firewall nat
add action=dst-nat chain=dstnat dst-address=111.2.3.2 dst-port=9900 protocol=tcp
to-addresses=192.168.66.6 to-ports=80
```

在这个案例中，如果 Office B 路由器有公网 IP 情况下，本地也可以开启端口映射，这样作为备份端口使用，Office A 路由器通过隧道作为跳板机为主访问节点，实现线路冗余。

第二十六章 IPSec 配置

IPSec（Internet Protocol Security）作为新一代网络安全协议，为网络传输提供了安全保证，使端到端的数据保密成为可能，是互联网上的新一代安全标准。提供包括访问控制、无连接的完整性、数据源认证、抗重放 (replay)保护、保密和有限传输保密性在内的服务，服务基于 IP 层并对 IP 及上层协议进行保护。

IPSec 包含了三个最重要的协议：认证头 AH（Authentication Header），封装安全载荷 ESP（Encapsulating Security Payload），密钥交换协议 IKE（Internet Key Exchange）：

- AH 是一个安全协议头，用于增强 IP 层的安全，提供数据源认证、信息完备校验以及抗重播等功能，从而确保数据包的完整性和真实性，但不提供机密性安全服务。
- ESP 和 AH 同样也是提供 IP 层安全，ESP 协议使用的是加密和校验机制，提供信息源校验、信息完整性、抗重传和机密性等。
- IKE 协议提供密钥协商、建立和维护安全联盟 SA 的服务，为 IPSec 协商生成密钥

IPSec 中 ESP 和 AH 不能同时使用

26.1 IPsec 基本术语：

Peer 对等体：

IPSec 建立 IP 通信的站点被称为 Peer 对等体。对等体配置用于建立 IKE 之间的连接，该连接将用于协商 SA 的密钥和算法

SA 安全联盟

SA（Security Association）安全联盟定义了 IPSec 通信对等体间将使用的密钥和加密算法。SA 是单向的，在两个对等体之间的双向通信，至少需要两个 SA。SA 由一个三元组组成，三元组包括安全参数索引 SPI（Security Parameter Index）、目的 IP 地址和安全协议名（AH 或 ESP）。

协商方式

建立 SA 的方式有以下两种：

手动（manual）：建立 SA 安全联盟比较复杂，安全联盟所需的全部信息都必须手工配置。但优点是可以不依赖 IKE 而单独实现 IPSec 功能，在很多路由器已经淘汰手动配置。

IKE 动态协商（isakmp）：建立 SA 安全联盟更简单，只需 Peer 对等体间配置好 IKE 协商参数，由 IKE 自动协商来创建和维护 SA。

IKE 工作过程

IPSec 保护一个 IP 包之前，先要建立 SA，而 IKE 就是用来建立和管理 SA 的。

IKE 的用途：为 IPSec 协商生成密钥，供 AH/ESP 加解密和验证使用。在 IPSec 通信双方之间，动态地建立安全关联（SA），对 SA 进行管理和维护

IKE 经过两个阶段为 IPSec 进行密钥协商并建立 SA。

第一阶段:通信各方彼此间建立了一个已通过身份验证和安全保护的通道，此阶段的交换建立了一个 ISAKMP 安全联盟，即 ISAKMP SA (也可称为 IKE SA)。 第一阶段交换有两种协商模式:主模式协商和野蛮模式协商。

第二阶段:用已经建立的安全联盟(IKE SA)为 IPSec 协商安全服务，即为 IPSec 协商具体的安全联盟，建立 IPSec SA，产生真正可以用来加密数据流的密钥，IPSec SA 用于最终的 IP 数据安全传送

IPSec 封装模式

隧道模式

在隧道模式下，AH 或 ESP 插在原始 IP 头之前，另外生成一个新 IP 头放到 AH 或 ESP 之前。

传输模式

在传输模式下，AH 或 ESP 被插入到 IP 头之后但在传输层协议之前。

隧道模式生成新的包头安全性比传输模式高，但隧道模式比传输模式占用带宽更多。

IPSec 使用的认证算法和加密算法

认证算法

IPSec 可以使用三种认证算法

- MD5：MD5 通过输入任意长度的信息，产生 128bit 的信息摘要。
- SHA-1：SHA-1 通过输入长度小于 2 的 64 次方比特的消息，产生 160bit 的信息摘要。
- SHA-2：SHA-2 算法相对于 SHA-1 加密数据位数有所上升，安全性能要远远高于

加密算法

加密算法实现主要通过对称密钥系统，它使用相同的密钥对数据进行加密和解密。IPSec 使用以下三种加密算法：

- DES：使用 56bit 的密钥对一个 64bit 的明文块进行加密。
- 3DES：使用三个 56bit 的 DES 密钥（共 168bit 密钥）对明文进行加密。
- AES：使用 128bit、192bit 或 256bit 密钥长度的 AES 算法对明文进行加密。

通信保护协议：

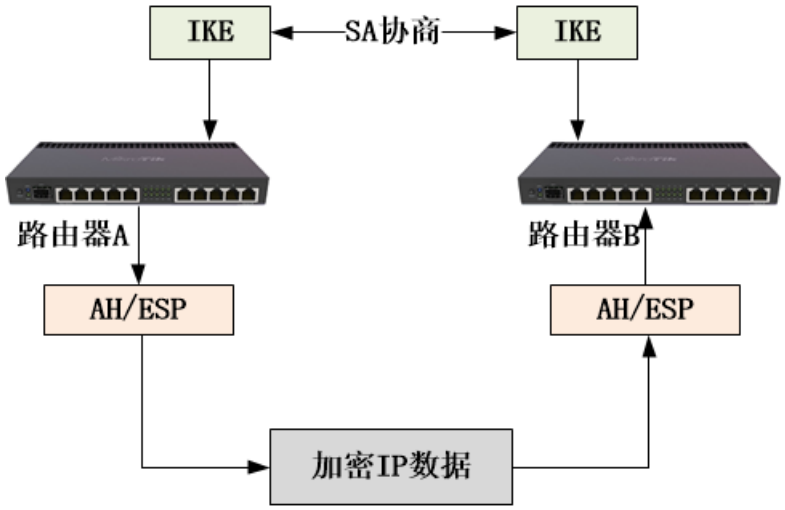
通信保护协议包括：AH 和 ESP 协议：AH 认证头协议主要特点：对数据包头进行校验，不提供保密性，只对整个 IP 数据包提供保护；ESP 除提供 AH 认证头协议的所有功能之外，还有提供数据保密，

安全特性	AH	ESP
数据完整校验	支持	支持（不验证 IP 头）
数据源验证	支持	支持
数据加密解密	不支持	支持
抗重放服务	支持	支持
NAT 穿透	不支持	支持

AH 认证头协议，无法在穿透 NAT，因为 AH 协议会对 IP 包头进行校验，ESP 协议则支持。RouterOS 默认选择的是 ESP 协议（在/ip ipsec policy 路径下， **ipsec-protocols=esp**）

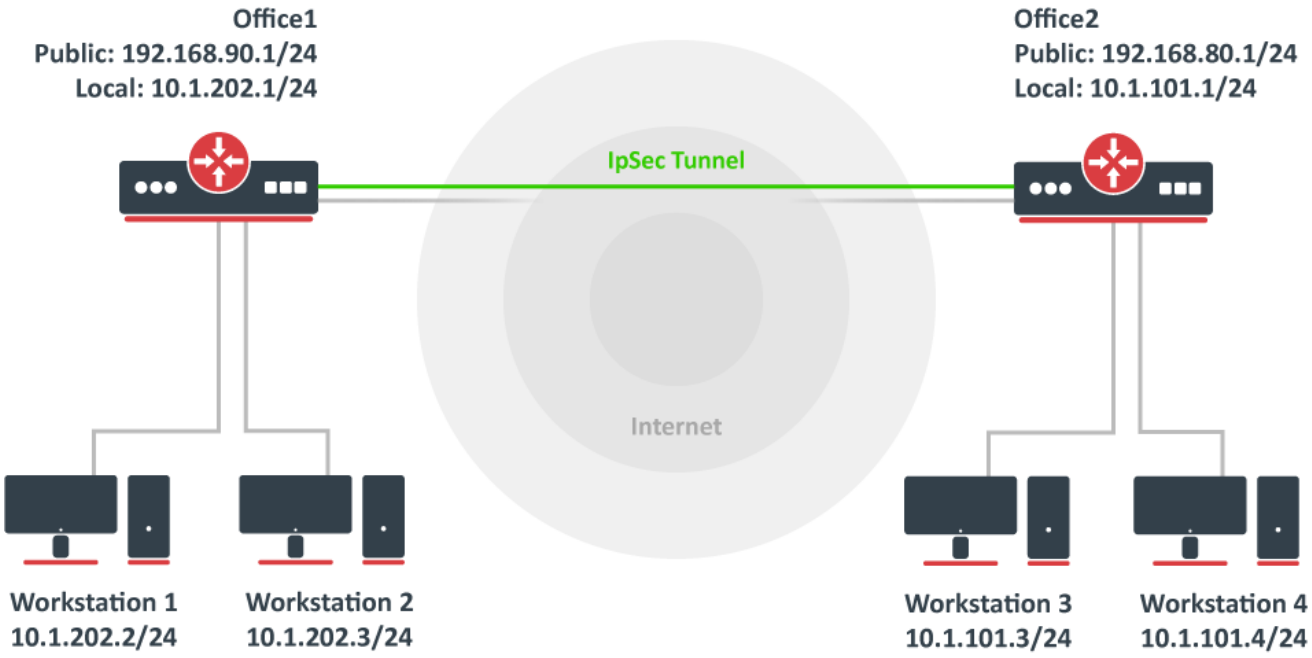
ESP 通常使用 DES、3DES、AES 等加密算法实现数据加密，使用 MD5 或 SHA1 来实现数据完整性认证

IPSec 协议不是一个单独的协议，它给出了应用于 IP 层网络数据安全的一整套体系结构。IPSec 通过认证头 AH 和封装安全载荷 ESP 这两个安全协议来实现 IP 数据报的安全传送。IKE 协议提供密钥协商、建立和维护安全联盟 SA 的服务。



26.2 点对点 IPSec（IKEv1）隧道模式

考虑如下拓扑，两个办公网络接入互联网，办公内网有自己的 IP 段，Office1 为 10.1.202.0/24，Office2 为 10.1.101.0/24，这些主机通过 NAT 转换上网。两个远程办公网络需要通过 IPsec 隧道打通内部网络。



Office 1 配置

首先，为 ESP 创建一个新认证和加密策略，根据需要选择加密参数。通过创建策略组，以便将来调整策略使用，可为每个 Peer 提供可以策略选择项。此参数要求站点之间配置一致，否则两个站点无法连接。

```
/ip ipsec profile
add dh-group=modp2048 enc-algorithm=aes-128 name=ike1-site2
/ip ipsec proposal
add enc-algorithms=aes-128-cbc name=ike1-site2 pfs-group=modp2048
```

继续配置对等体，指定对端办公路由器的地址。IPSec 通过 UDP/500 和 UDP/4500 端口连接，因此要确保两端路由器的防火墙没有限制。为 Peer 对等体创建配置策略。

```
/ip ipsec peer
add address=192.168.80.1/32 name=ike1-site2 profile=ike1-site2
```

下一步是创建一个 identity 身份标识。这里配置基本的预共享密钥安全隧道，确保密钥使用强加密，并之前设置的 Peer 规则。

```
/ip ipsec identity
add peer=ike1-site2 secret=thisisnotasecurepsk
```

如果安全性很重要，可以考虑使用 IKEv2 和另一种 auth-method。

最后，创建一个 policy 策略来控制网络之间的通信，并做相应的加密。

```
/ip ipsec policy
add src-address=10.1.202.0/24 src-port=any dst-address=10.1.101.0/24 dst-port=any tunnel=yes
action=encrypt proposal=ike1-site2 peer=ike1-site2
```

Office2 配置

office2 和 office 1 的配置基本相同，只需要调整 IP 地址配置即可。首先创建新的 Phase 1 和 Phase 2 配置策略：

```
/ip ipsec profile
add dh-group=modp2048 enc-algorithm=aes-128 name=ike1-site1
/ip ipsec proposal
add enc-algorithms=aes-128-cbc name=ike1-site1 pfs-group=modp2048
```

配置 peer 和 identity：

```
/ip ipsec peer
add address=192.168.90.1/32 name=ike1-site1 profile=ike1-site1
/ip ipsec identity
add peer=ike1-site1 secret=thisisnotasecurepsk
```

然后配置 policy：

```
/ip ipsec policy
add src-address=10.1.101.0/24 src-port=any dst-address=10.1.202.0/24 dst-port=any tunnel=yes
action=encrypt proposal=ike1-site1 peer=ike1-site1
```

此时完成以上步骤，隧道已经建立，通过以下命令检查连接状态

```
/ip ipsec
active-peers print
installed-sa print
```

绕过 NAT 和 Fasttrack 规则

此时，如果试图通过 IPsec 隧道发送数据，将无法访问，数据包将丢失。这是因为两端路由器都配置了 NAT 规则(masquerade)，在数据包被加密之前改变源地址。路由器无法加密数据包，因为源地址与策略配置中指定的地址不匹配。

解决办法是在路由器的 IP/Firewall/NAT 添加绕过 nat 规则

Office 1 路由器：

```
/ip firewall nat
add chain=srcnat action=accept place-before=0 src-address=10.1.202.0/24 dst-address=10.1.101.0/24
```

Office 2 路由器：

```
/ip firewall nat
add chain=srcnat action=accept place-before=0 src-address=10.1.101.0/24 dst-address=10.1.202.0/24
```

如果在添加 NAT 绕过规则之前，曾尝试建立 IP 连接，则必须从现有连接会话中清除连接（/ip firewall connection），或重启两台路由器。并且绕过规则必须被放所有 nat 规则顶部，可以通过 move 命令完成，winbox 中可以之间拖动到顶部。

另一个问题是，如果启用了 IP/Fasttrack，数据包就会绕过 IPsec 策略。所以需要在 FastTrack 之前添加接受规则。.

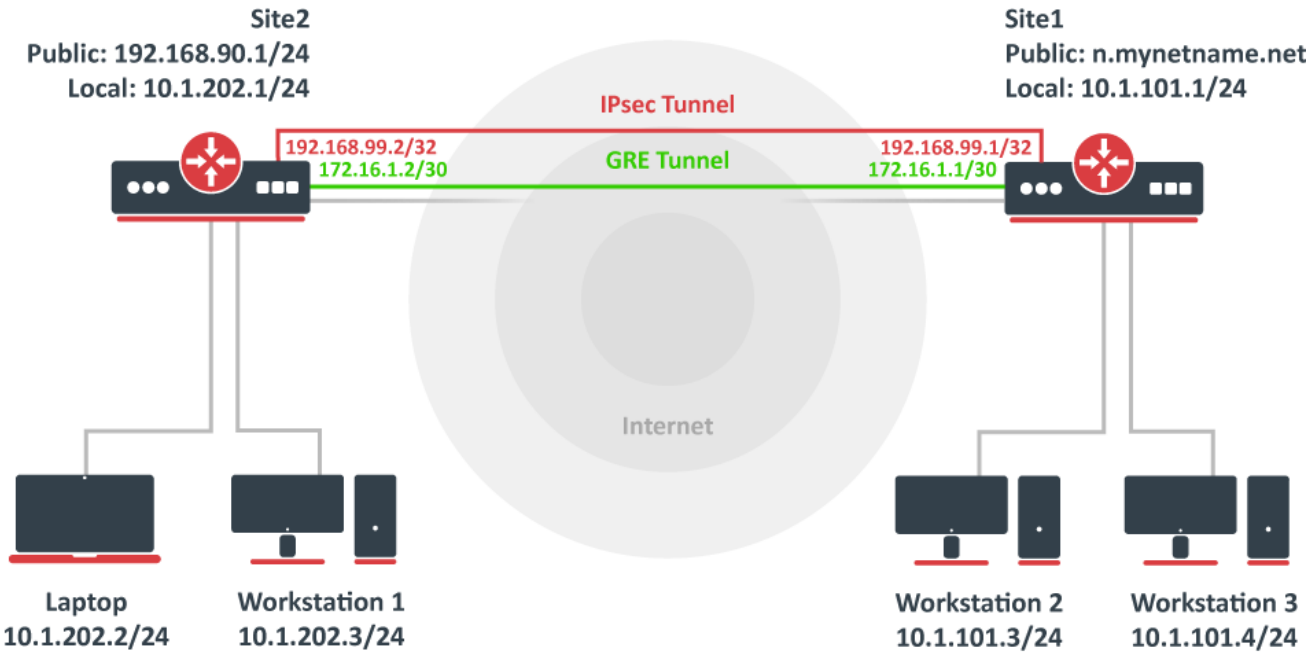
```
/ip firewall filter
add chain=forward action=accept place-before=1
src-address=10.1.101.0/24 dst-address=10.1.202.0/24 connection-state=established,related
add chain=forward action=accept place-before=1
src-address=10.1.202.0/24 dst-address=10.1.101.0/24 connection-state=established,related
```

然而，如果在 IPsec 隧道上有大流量通过，可能会给路由器的 CPU 增加负载。为减小 CPU 负载，可以使用 IP/Firewall/Raw 绕过连接跟踪，这样可以减少 CPU 大约 30%的负载。

```
/ip firewall raw
add action=notrack chain=prerouting src-address=10.1.101.0/24 dst-address=10.1.202.0/24
add action=notrack chain=prerouting src-address=10.1.202.0/24 dst-address=10.1.101.0/24
```

26.3 基于 DDNS 建立 IPSec（IKEv2）隧道

本实例也通过在两台 RouterOS 设备之间创建 IPSec 隧道，然后在通过 IPSec 建立 Gre 隧道，本实例涉及到其中一台路由器没有公网静态 IP 地址，需要使用 DNS 域名建立连接，也就是需要使用 DDNS。为了简单起见，将使用 RouterOS 内置的 DDNS 服务/IP Cloud。同时在本实例中还会应用到环回地址建立 Gre 隧道。



在这个实例中，我们使用了动态 DNS 建立 IPSec 隧道、还有创建环回地址，并通过环回地址创建 Gre 隧道

站点 1 (server)配置

此站点将监听外部连接请求，并作出响应。将使用模式配置为第二个站点提供一个 IP 地址，但是首先，需要使用 bridge 来创建一个环回接口，并分配一个 IP 地址作为环回地址，这个 IP 地址将用于建立 GRE 隧道。

```
/interface bridge
```

```
add name=loopback
/ip address
add address=192.168.99.1 interface=loopback
```

继续 IPsec 配置，使用适合自身所需的加密强度，创建新的阶段 1 和阶段 2 配置。注意，此配置将侦听所有来至外部的 IKEv2 请求，这意味着此配置也能响应其他站点的 IKEv2 请求。

```
/ip ipsec profile
add dh-group=ecp256,modp2048,modp1024 enc-algorithm=aes-256,aes-192,aes-128 name=ike2
/ip ipsec proposal
add auth-algorithms=null enc-algorithms=aes-128-gcm name=ike2-gre pfs-group=none
```

接下来，用 responder=yes 创建一个新的模式规则。这将为其他站点提供一个 IP 配置，以及用于策略生成的环回地址。

```
/ip ipsec mode-config
add address=192.168.99.2 address-prefix-length=32 name=ike2-gre split-include=192.168.99.1/32
system-dns=no
```

通过创建策略组，以便本次或以后的 IPsec 配置调用

```
/ip ipsec policy group
add name=ike2-gre
```

现在可以设置一个新的策略 policy 模板，将用于配匹远端 peer 动态 IP 地址和环回地址

```
/ip ipsec policy
add dst-address=192.168.99.2/32 group=ike2-gre proposal=ike2-gre src-address=192.168.99.1/32
template=yes
```

下一步是创建侦听所有 IKEv2 请求的对等体配置。

```
/ip ipsec peer
add exchange-mode=ike2 name=ike2 passive=yes profile=ike2
```

通过预共享密钥身份验证来设置一个与远端对等体匹配的身份。

```
/ip ipsec identity
add generate-policy=port-strict mode-config=ike2-gre peer=ike2 policy-template-group=ike2-gre secret=test
```

服务器端现在已经配置好并侦听所有 IKEv2 请求。确保防火墙没有阻止 UDP/4500 端口。

最后一步是创建 Gre 隧道接口，当客户端建立 IPsec 连接成功后，Gre 隧道将连接：

```
/interface gre
add local-address=192.168.99.1 name=gre-tunnel1 remote-address=192.168.99.2
```

站点 2 (client)配置

同样为站点 2 创建环回接口和环回地址

```
/interface bridge
add name=loopback
/ip address
add address=192.168.99.2 interface=loopback
```

与服务器配置类似，首先创建新的阶段 1 和阶段 2 配置。由于该站点作为请求端，只要确保它们与服务器端配置的参数一直即可。

```
/ip ipsec profile
add dh-group=ecp256 enc-algorithm=aes-256 name=ike2-gre
/ip ipsec proposal
add auth-algorithms=null enc-algorithms=aes-128-gcm name=ike2-gre pfs-group=none
```

接下来，用 responder=no 创建一个新的模式配置规则。

```
/ip ipsec mode-config
add name=ike2-gre responder=no
```

仍然通过创建一个新的策略组，用于配置调用。

```
/ip ipsec policy group
add name=ike2-gre
```

在客户端站点也创建新的策略模板

```
/ip ipsec policy
add dst-address=192.168.99.1/32 group=ike2-gre proposal=ike2-gre src-address=192.168.99.2/32
template=yes
```

继续进行 Peer 对等配置，由于对端站点是动态 IP，因此在 address 参数下指定对端服务器 DNS 域名。

```
/ip ipsec peer
add address=n.mynetname.net exchange-mode=ike2 name=p1.ez profile=ike2-gre
```

为对端创建一个身份认证。

```
/ip ipsec identity
add generate-policy=port-strict mode-config=ike2-gre peer=p1.ez policy-template-group=ike2-gre secret=test
```

如果配置一切顺利，就会在 policy 下出现新的动态策略。

```
/ip ipsec policy print
Flags: T - template, X - disabled, D - dynamic, I - invalid, A - active, * - default
0 T * group=default src-address=::/0 dst-address=::/0 protocol=all proposal=default template=yes

1 T group=ike2-gre src-address=192.168.99.2/32 dst-address=192.168.99.1/32 protocol=all
proposal=ike2-gre template=yes
```

```
2 DA src-address=192.168.99.2/32 src-port=any dst-address=192.168.99.1/32 dst-port=any protocol=all
action=encrypt level=unique ipsec-protocols=esp
tunnel=yes sa-src-address=172.17.2.1 sa-dst-address=172.17.2.2 proposal=ike2-gre ph2-count=1
```

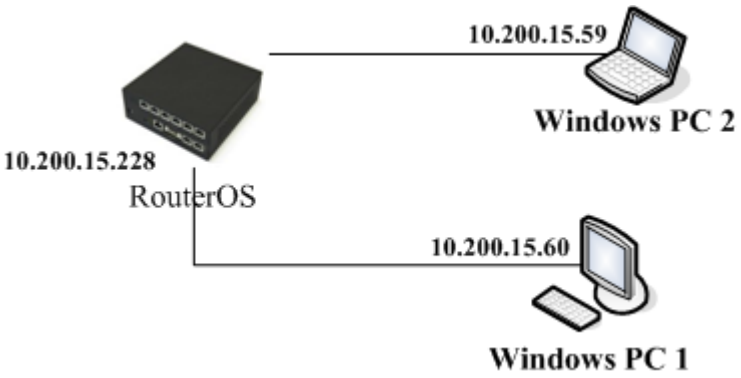
现在在两个站点之间从 192.168.99.2- 192.168.99.1 安全的 IPSec 隧道建立， 然后我们在这两个地址之间创建 Gre 隧道

```
/interface gre
add local-address=192.168.99.2 name=gre-tunnel1 remote-address=192.168.99.1
```

26.4 Windows L2TP/IPsec 连接

Windows 7/10/11 内建了 PPTP 客户端和和 L2TP/IPSec 客户端。PPTP 连接是不需要 IPsec 加密，而 L2TP/IPsec 默认要求建立 IPsec 连接，才能进行 L2TP 的拨号连接，RouterOS L2TP 服务器支持 IPSec 的加密连接，此功能从 v6.16 版本加入到 L2TP Server 配置中。

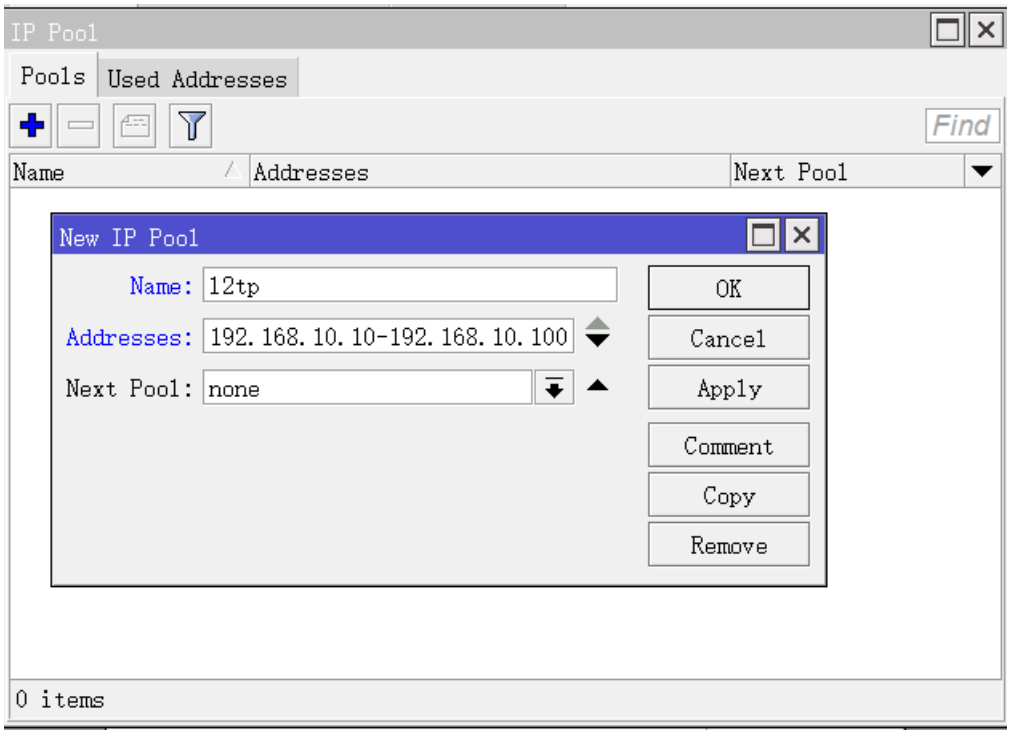
实例的网络结构如下：



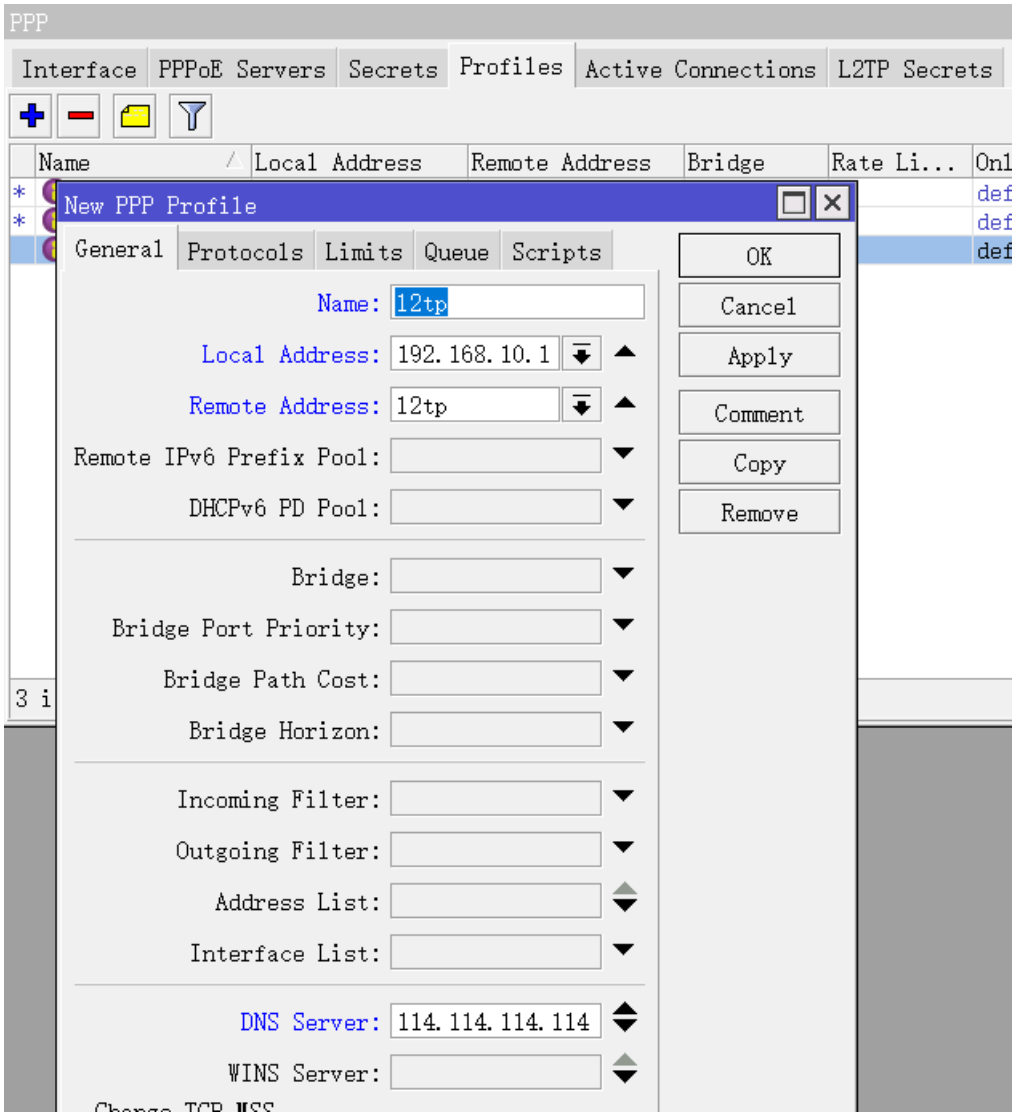
这里 RouterOS 的 IP 地址是 10.200.15.228，两台 PC 的 IP 地址分别是 10.200.15.59,和 10.200.15.60。两台 PC 的 IP 地址必须是固定，以便 IPsec 连接成功。在这个拓扑图里要求所有的地址是能被访问到的，即非 nat 转换的地址（也非 L2TP 隧道分配的 IP 地址）。

RouterOS 配置

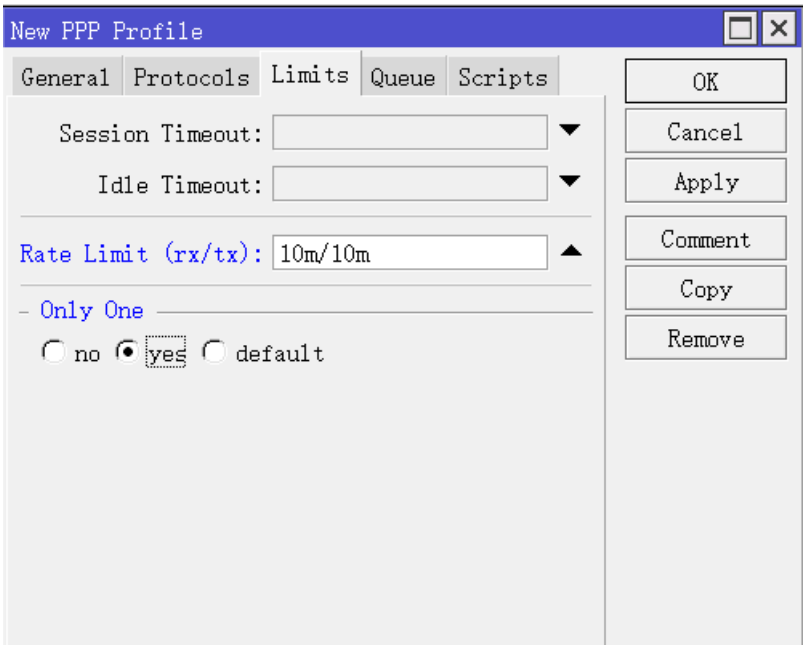
首先进入/ip pool 为 L2TP 拨号用户分配地址池，分配地址范围为 192.168.10.10-192.168.10.100：



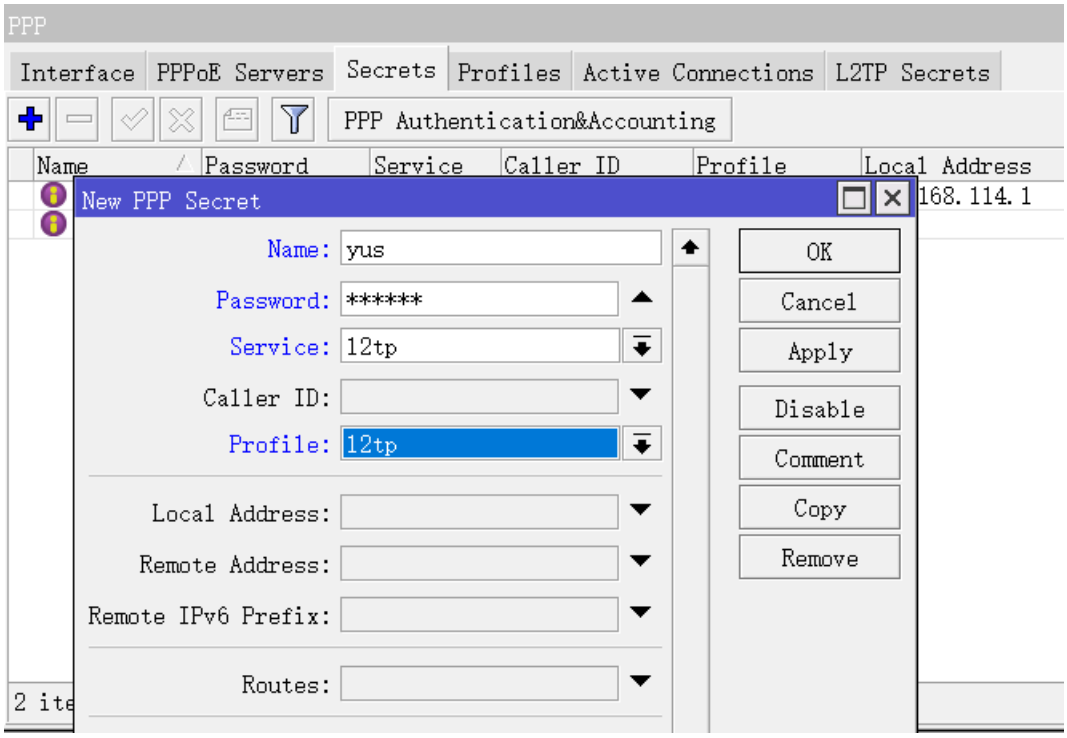
进入/ppp profile 配置创建一个新的 proflie 规则，取名 l2tp，并配置 local-address 为 192.168.10.1，remote address 是之前创建的 l2tp 地址池，用于分配给拨号用户，分配 DNS 为 114.114.114.114。



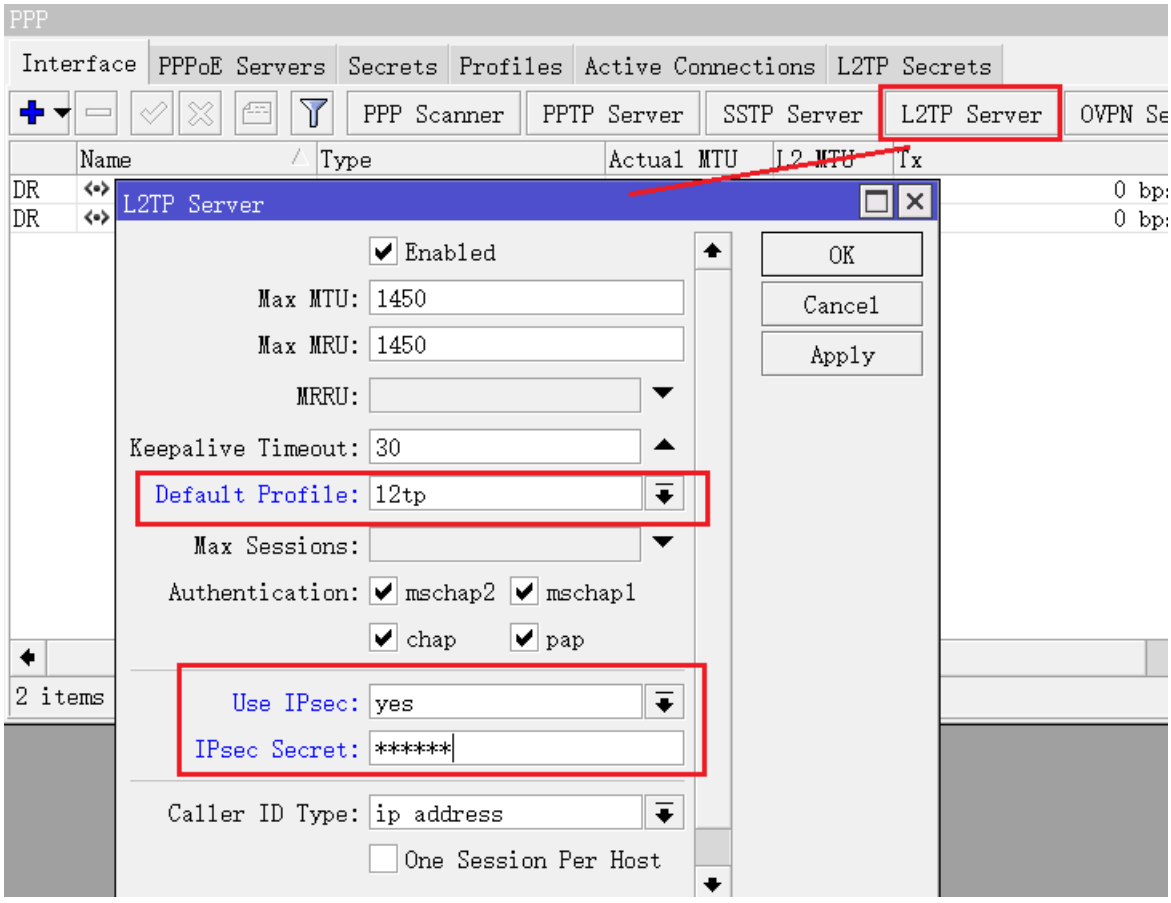
在 limit 选项里的 rate-limit、only one 可以设置，如下面是限制每个用户带宽 10M，only-one 属性是账号登录唯一性，不能多拨。



进入/ppp secret 添加用户账号，配置如下，proflie 定义了之前配置的 l2tp 策略，service 代表该账号只能使用 l2tp 协议登录，其他隧道协议不支持。



最后启用 L2TP Server，default profile 选择 l2tp，下面的 user-ipsec 选择 yes，ipsec-secret 设置密钥，预设值共享密钥为“123456”



以上 RouterOS 的 L2TP server 就配置完成，只后就是 windows 客户端配置

Windows 配置

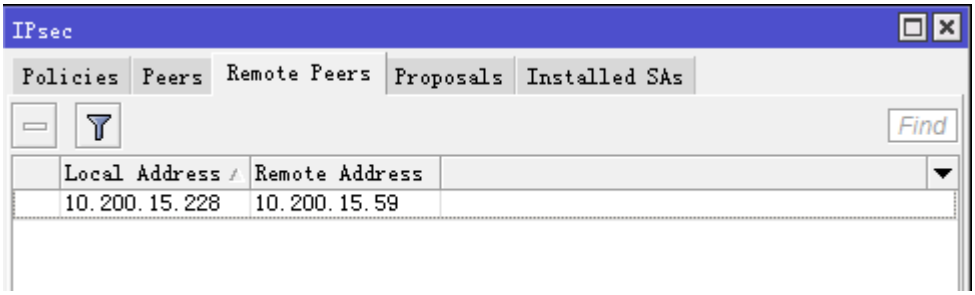
以下是基于 Windows10 配置，打开“网络和 internet”界面，选择添加 VPN 连接，

然后添加如下配置信息：

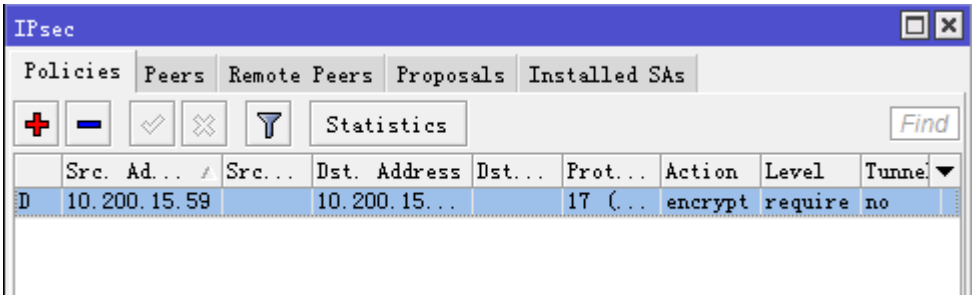
配置完成后点击保存

保存完成后，选择刚新建的 l2tp，点击连接

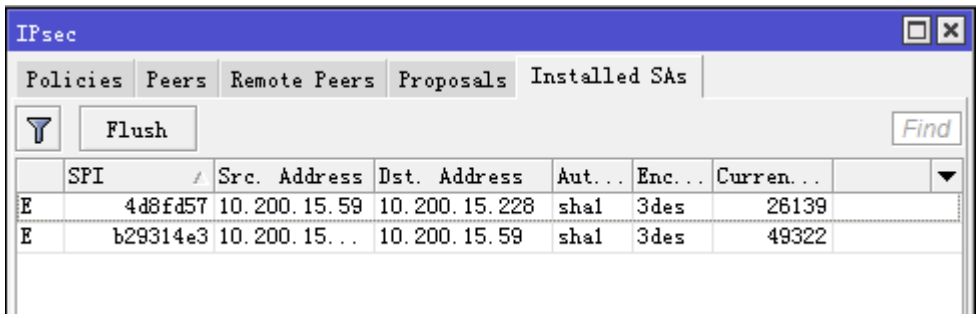
连接完成后，在 remote peers 可以看到连接的 IP 地址



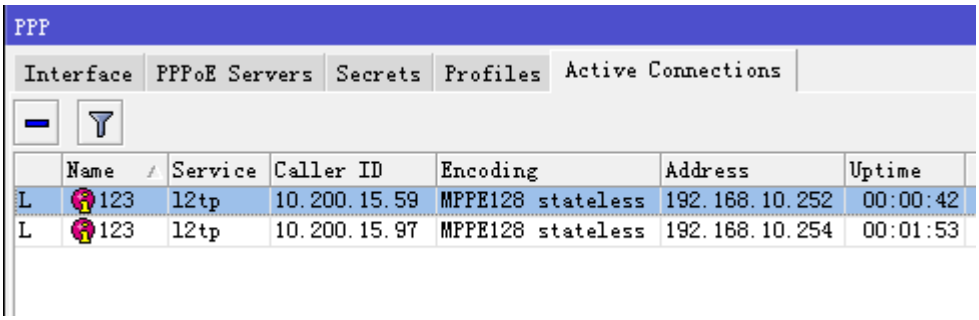
Policies 策略会被自动添加



Installed SAs 状态，注意当你的 L2TP 注销后，可能会出现 Installed SAs 状态没有清楚，第二次回放可能需要使用 Flush 清空状态



PPP 里的 active 状态



第二十七章 EoIP 隧道

EoIP（Ethernet over IP）隧道是一个建立在两个路由器的 IP 传输层之间的以太网隧道协议，是 MikroTik RouterOS 的自由协定。EoIP 接口表现的类似以太网传输，当路由器的桥接功能被启用后，所有的以太网数据流量（所有的以太网协议）将被桥接就如同在两个路由器（启用了桥接功能）之间有物理交换机接口和光纤收发器一样。

27.1 EoIP 介绍

有 EoIP 接口的网络设置：

- 可以在互联网桥接两个远端的局域网络
- 可以实现加密的隧道桥接
- 可以在 802.11b 'ad-hoc'无线网络上桥接局域网

规格

功能包要求：system
等级要求：Level3
操作路径：/interface eoip

属性	描述
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; 默认: **enabled**)	地址解析协议模式： • disabled – 禁用接口的 ARP 功能 • enabled – 启用接口的 ARP 功能 • proxy-arp – 接口将开启 ARP 代理功能 • reply-only – 接口仅对 ARP 列表（/ip arp）的静态条目做响应，动态的 ARP 将不做响应。即对 IP 地址与 MAC 做静态绑定。
clamp-tcp-mss (*yes* | *no*; 默认: **yes**)	
dont-fragment (*inherit* | *no*; 默认: **no**)	
dscp (*integer: 0-63*; 默认: **inherited**)	数据包的 DSCP 值，继承选项，即会从加密数据包中继承 DSCP 值
ipsec-secret (*字符*; 默认:)	当 IPsec 密码设置，路由器会为两端设备创建 IPsec 加密，添加动态的 IPsec 共享的密钥和默认值的策略（默认策略为 phase2 使用 sha1/aes128cbc）
keepalive (*整型[/时间]*,*integer 0..4294967295*; 默认: **10s,10**)	隧道 keepalive 参数设置隧道保持的时间间隔周期，判断远端接到是否连通，通过运行标志"R"显示接口连接状态，如果配置时间周期重试失败，接口运行标志"R"被删除。参数使用以下格式 KeepaliveInterval,KeepaliveRetries，KeepaliveInterval 为重试周期时间

	KeepaliveRetries 为尝试次数 默认 keepalive 为 10 秒，10 次
l2mtu (*整型; 只读*)	二层网络最大传输单元，EoIP 配置无效 N
local-address (*IP*; 默认:)	隧道本地地址，即路由器本地 IP
mac-address (*MAC*; 默认:)	接口的 MAC 地址，根据 IANA 允许使用的免费 MAC 地址范围 **00:00:5E:80:00:00 - 00:00:5E:FF:FF:FF**
mtu (*整型*; 默认: **auto**)	三层网络最大传输单元
name (*字符*; 默认:)	接口名称
remote-address (*IP*; 默认:)	对端 EoIP 路由器的 IP 地址
tunnel-id (*整型: 65536*; 默认:)	隧道 ID 设置，唯一的隧道标识符，必须与对端匹配

EoIP 接口应用在有 IP 层连接，EoIP 通道可以在 IPIP 隧道，PPTP 128bit 加密隧道，PPPoE 连接或任何传输 IP 的连接上运行。具体属性：

- 每个上运行隧道接口可以与一个有相同“隧道 ID”的相应接口配置的远程路由器相连接
- EoIP 接口就类似于 interface 列表下的以太网接口一样
- 这个接口支持以太网接口的所有特征。IP 地址及其他隧道可以在这个接口上运行
- EoIP 协议封装以太网帧在 GRE（IP 协议号 47）数据报中，并把它们发送到 EoIP 隧道的远程端
- EoIP 隧道的最大计数为 65536

注：EoIP 隧道至少有 42 字节的开销（8byte GRE + 14 byte Ethernet + 20 byte IP)

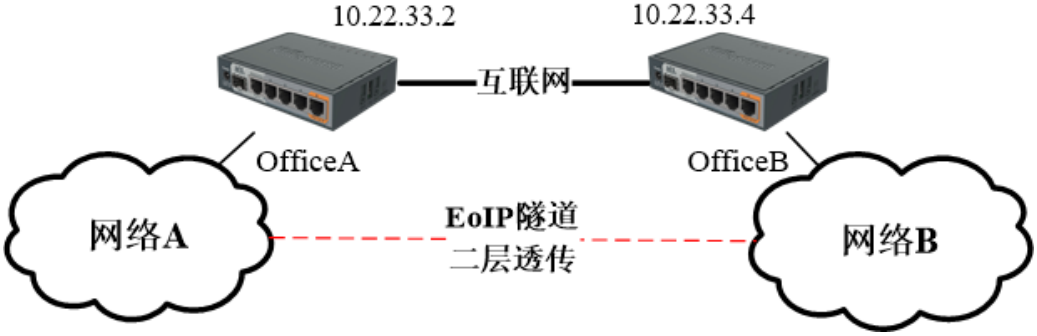
注：tunnel-id 是一种识别隧道的方法。在同一个路由器上不应该有相同 tunnel-id 的隧道。在参与的两个路由器的 tunnel-id 必须是平等的。

mtu 必须设置为 1500 以消除隧道内的数据包分段存储（允许类似以太网络的透明桥接，通过隧道传输全尺寸的以太网帧）。

当桥接 EoIP 隧道时，推荐对每个隧道设置唯一的 MAC 地址以使桥接算法正常工作。对于 EoIP 接口你可以使用从 **00-00-5E-80-00-00** 到 **00-00-5E-FF-FF-FF** 范围的 MAC 地址，IANA 就是为这些情况保留的。

27.2 EoIP 应用实例

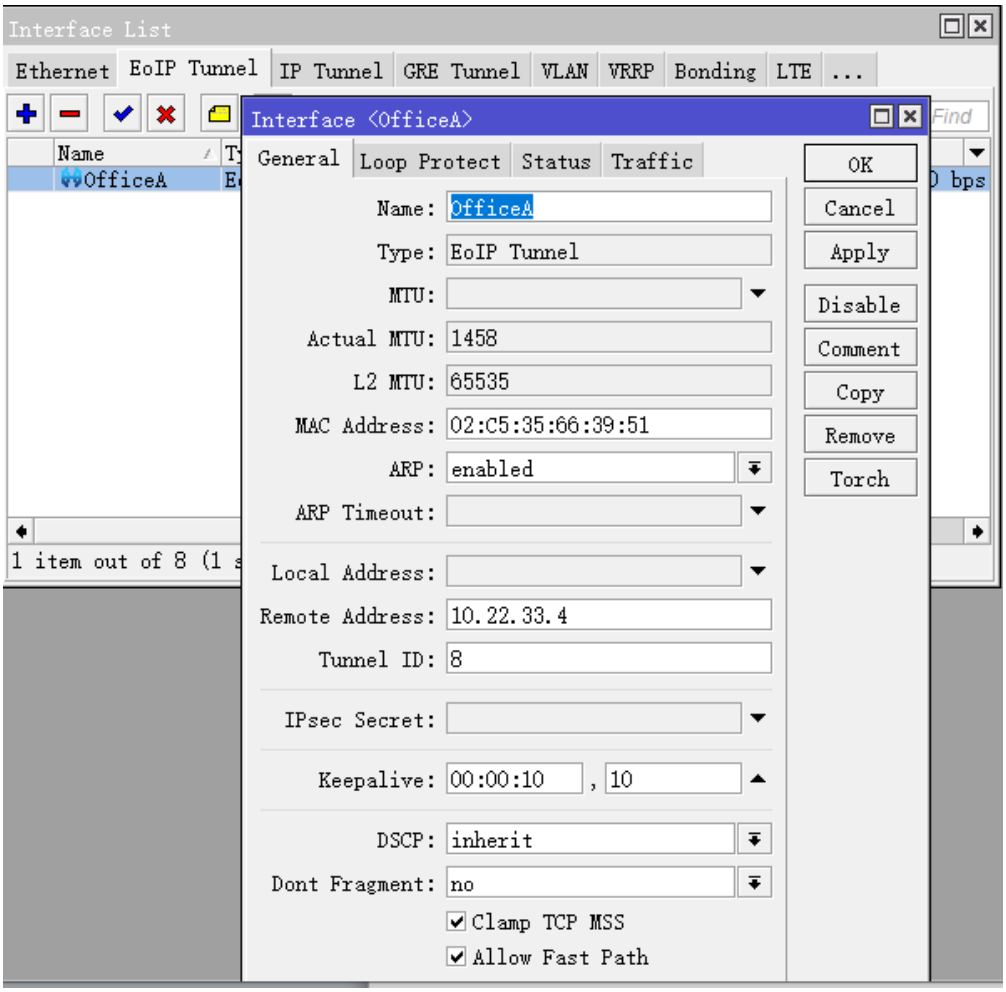
假设有两个异地的办公点 officeA 和 officeB，通过 EoIP 隧道将他们连接起来，并建立 2 层透明桥接，如下图：



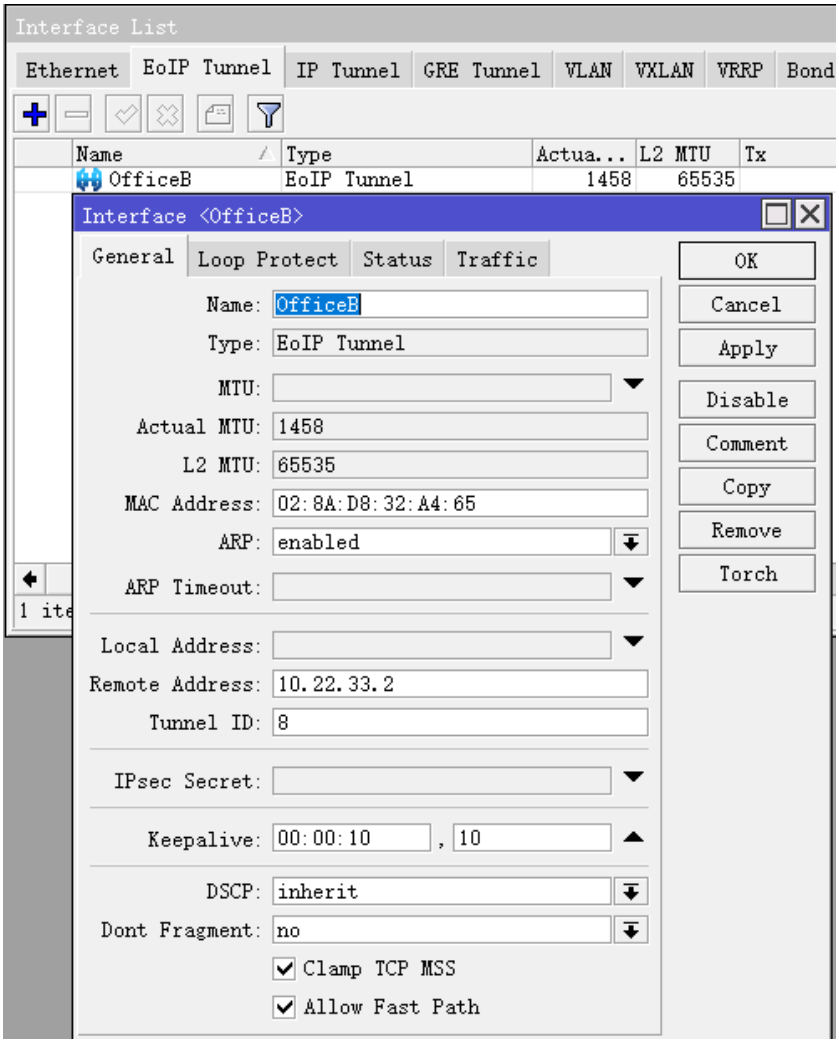
网络参数：

- OfficeA 的公网 IP 地址：10.22.33.2
- OfficeB 的公网 IP 地址：10.22.33.4

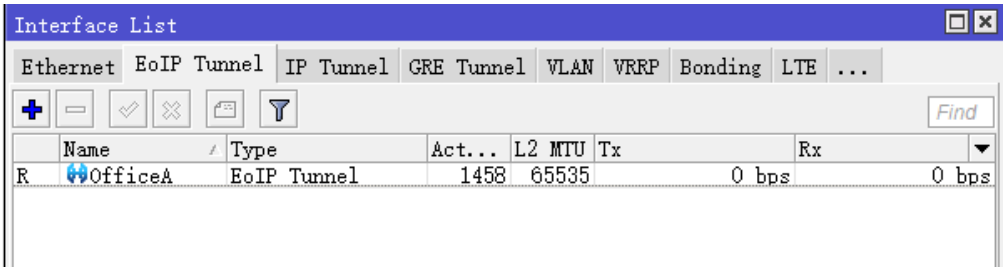
1、这里我们将配置两个 Eoip 隧道的 Tunnel ID 为 8，首先在 Interface 里建立 Eoip 隧道



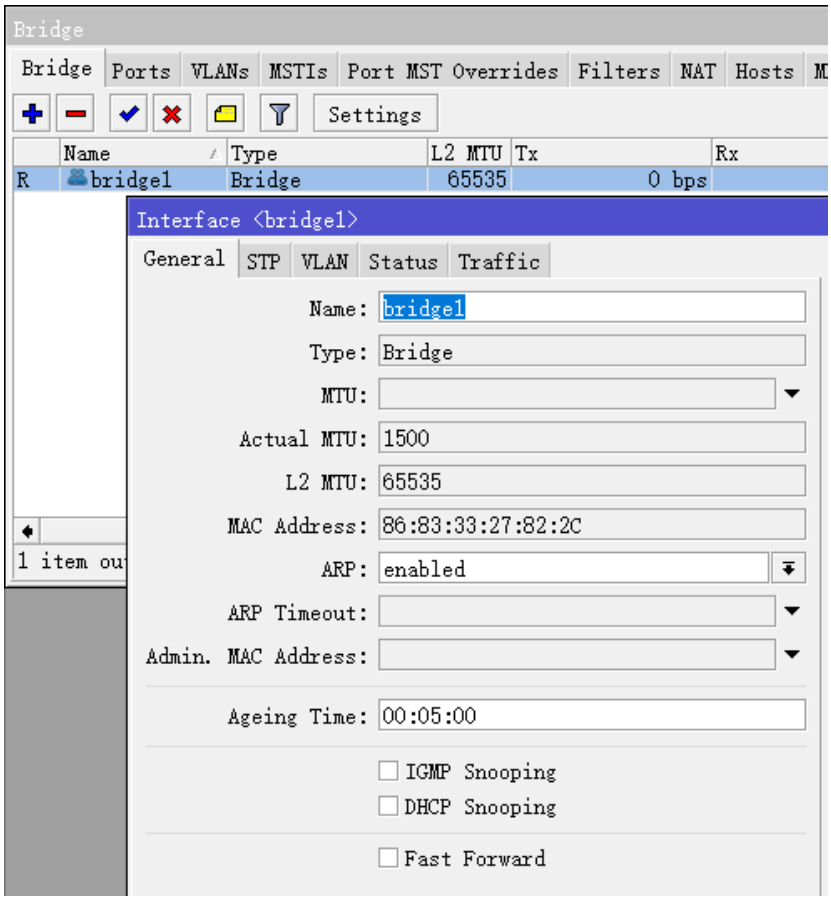
建立 officeB 的 Eoip 隧道：



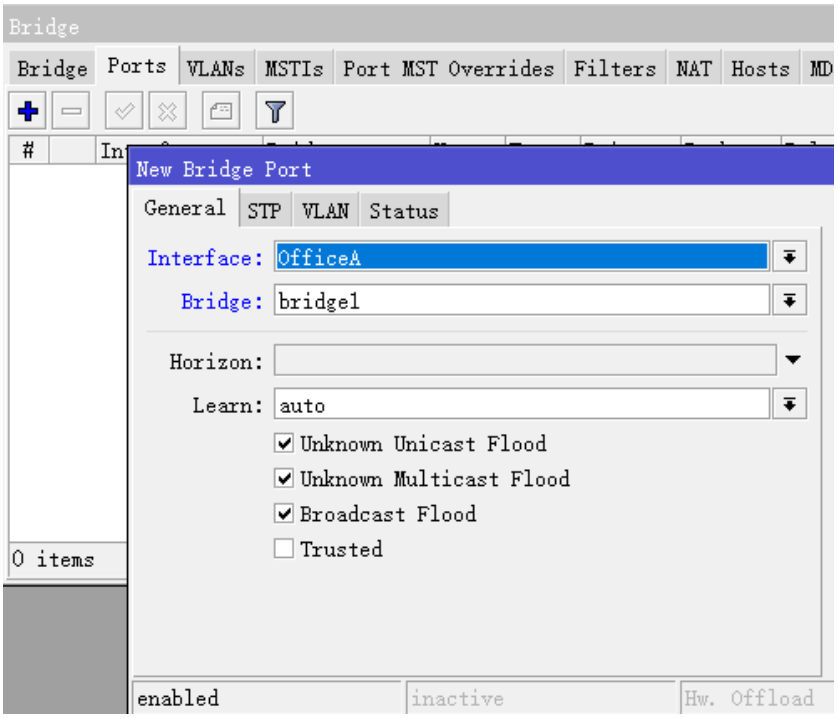
2、接下来，OfficeA 和 OfficeB 的配置基本相同，现在我们来看 officeA 配置，在 interface 中的 Eoip Tunnel 中可以看到 Eoip 隧道连接成功，前缀显示“R”：



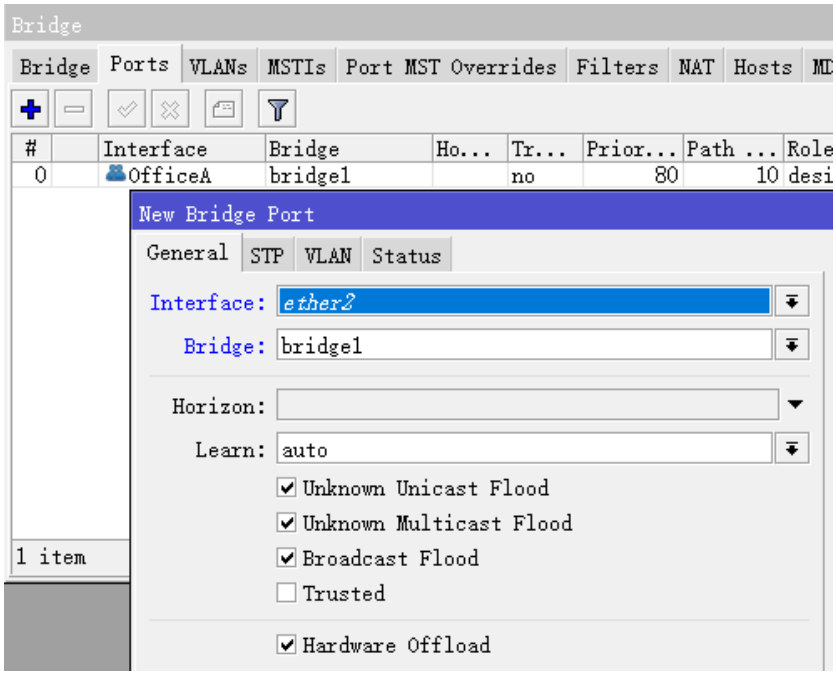
OfficeA 和 OfficeB 路由器，进入 Bridge 菜单都创建一个 bridge 接口，都取名为 bridge1，



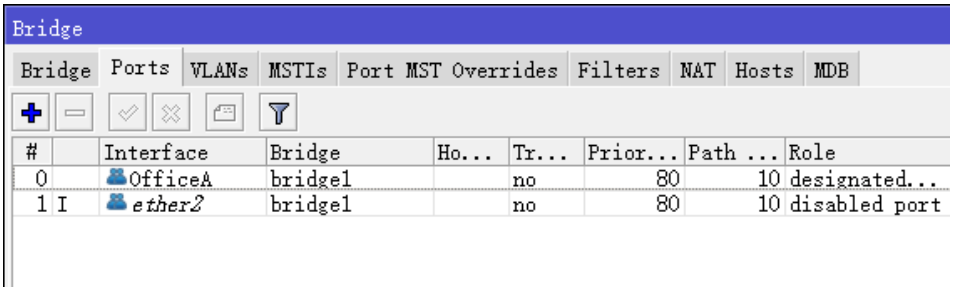
在 OfficeA 路由器，Bridge port 中将 OfficeA 的 EoIP 接口绑定到 Bridge1：



将 ether2 网卡，也绑定到 Bridge1

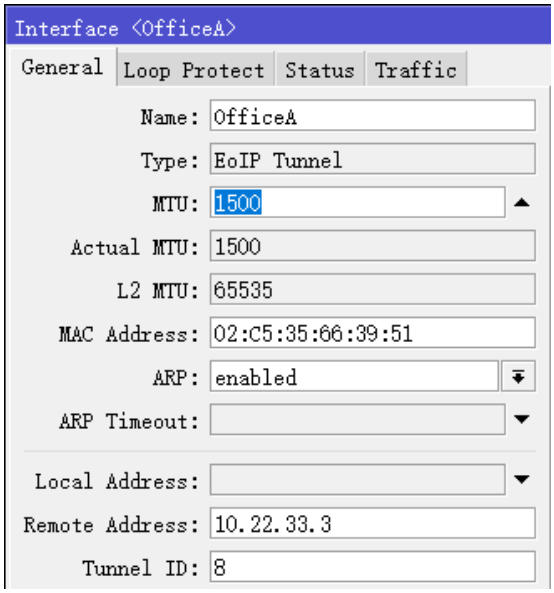


添加完成后如下：



OfficeB 路由器也做同样的操作，将 OfficeB 的 EoIP 接口和 ether2 接口加入到 Bridge1 中。添加完成后，EoIP 将两个远程的局域网打通二层透明传输。

注意，EoIP 的 mtu 值默认为 1458，在实际使用中可能会出现部分网络无法访问，这是因为 mtu 封装导致，解决这个问题通过修改 EoIP 的 MTU 值为 1500 解决，下面是修改 OfficeA 的配置（OfficeB 也做同样的修改）：



注：在做 NAT 规则的时候，特别是伪装，需要指明伪装的端口，如果默认伪装，将会把 EoIP 隧道隐藏，使其二层透穿出现问题，为了避免影响 EOIP 的连接，要选择 out-interface 为 WAN 口。

故障分析

- **路由器可以相互之间 ping 通但 EoIP 隧道依然不能正常工作！**

  检查 EoIP 接口的 MAC 地址——两台通信的路由器它们的 MAC 不应该一样！

关于 Wireless 无线 AP 的 EoIP 隧道配置，请参考《RouterOS 脚本编写与 WLAN 无线教程》的 6.3 章节

第二十八章 GRE 和 IPIP 隧道协议

28.1 GRE 隧道

操作路径: /interface gre

GRE (Generic Routing Encapsulation)一种隧道协议，源于思科。创建一个虚拟点对点连接，能封装各种协议在里面进行通信。

GRE 与 IPIP 和 EoIP 相同都属于无状态隧道连接，即远程隧道中断，所有路由指向该隧道的流量都变成黑洞路由。为了解决这个问题 RouterOS 为 GRE 隧道增加了 keepalive 功能（GRE 隧道增加一个 24 字节的包头，4 字节 gre 头+20 字节 IP 头）

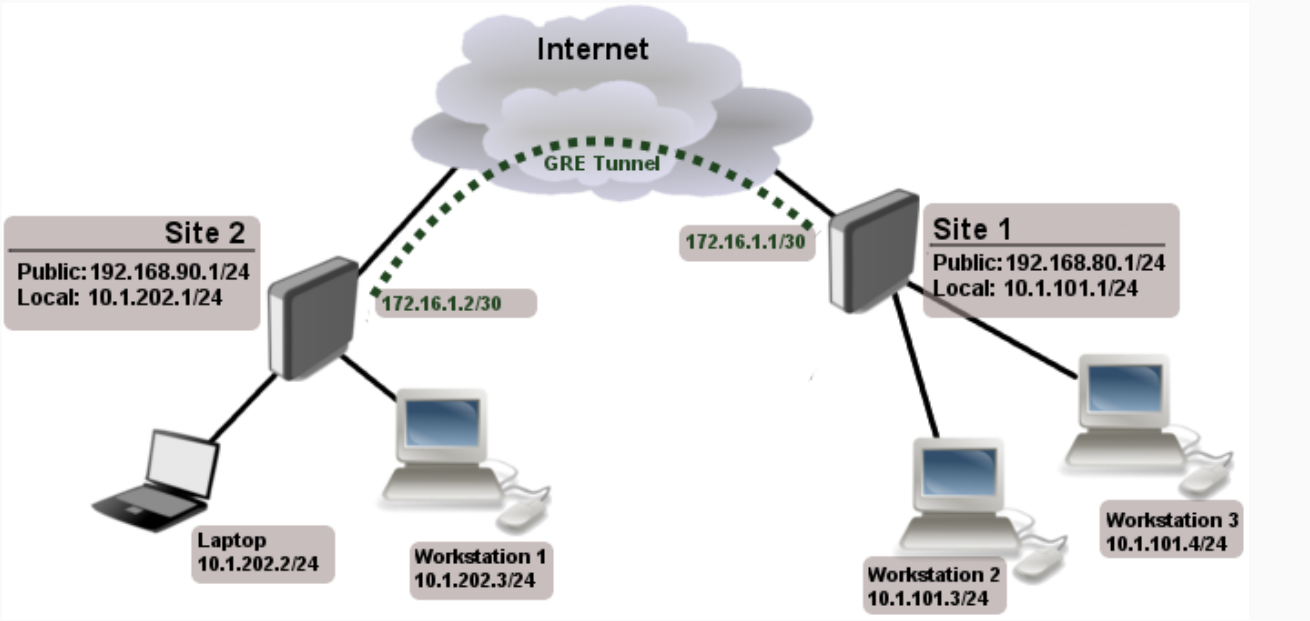
注意：GRE 隧道能转发 IP 和 IPv6 数据报（ethernet 800 和 86dd 类型）

属性

属性	描述
clamp-tcp-mss (*yes* | *no*; 默认: **yes**)	控制是否改变接收到的 TCP SYN 报文的 MSS 大小。启用该功能后，如果当前 MSS 大小超过隧道接口 MTU(考虑到 TCP/IP 开销)，路由器将改变接收到的 TCP SYN 报文的 MSS 大小。接收到的封装报文仍然包含原来的 MSS，只有在解封装后，MSS 才会改变。
dscp (*inherit* | *integer [0-63]*; 默认:)	设置 dscp 值到 IPIP 包头，或从本地 dscp 继承到隧道传输中。
ipsec-secret (*string*; 默认:)	当指定 ipsec-secret 时，路由器将通过动态方式，将 ipsec 对等体添加到 remote-address，使用预共享密钥和缺省值的策略(缺省 phase2 使用 sha1/aes128cbc)。
local-address (*IP*; 默认: 0.0.0.0)	本地路由器的 IP 地址（外网口地址），如果使用默认地址 0.0.0.0，将使用出口 IP 作为 local-address
mtu (*integer*; 默认: **1500**)	三层网络最大传输单元
keepalive (*integer[/time],integer 0..4294967295*; 默认: **10s,10**)	keepalive 参数用于设置隧道运行保持的时间间隔，在此时间间隔内，即使隧道对端发生故障，隧道运行标志"R"仍将保持。如果超过了时间范围，重试失败，则删除接口运行标志"R"。参数填写格式为:"KeepaliveInterval"、"KeepaliveRetries"(其中"KeepaliveInterval"为时间间隔)、"KeepaliveRetries -重试次数"。缺省情况下，keepalive 设置为 10 秒，重试 10 次。
name (*string*; 默认:)	接口名称
remote-address (*IP*; 默认:)	对端 Gre 隧道的 IP 地址

配置事例

下面的事例是通过互联网建立两段的 GRE 隧道连接



上面的网络拓扑有两个办公网站，Site1 本地网络地址 10.1.101.0/24，Site2 本地网络地址 10.1.202.0/24，我们需要通过 GRE 隧道将两个办公的本地网络连接起来

第一步，先创建两个路由器的 GRE 隧道，site 1 配置：

```
/interface gre add name=myGre remote-address=192.168.90.1 local-address=192.168.80.1
```

路由器 sit2 配置：

```
/interface gre add name=myGre remote-address=192.168.80.1 local-address=192.168.90.1
```

注意：在这个事例中 keepalive 参数没有配置，因此隧道接口不管隧道网络是否连接成功，flage 标记仍然是 R（running），因此建议配置中加入 keepalive 值

```
[admin@MikroTik] /interface gre> print
Flags: X - disabled, R - running
 0   R name="myGre" mtu=1476 l2mtu=65535 local-address=192.168.80.1
       remote-address=192.168.90.1 dscp=0
[admin@MikroTik] /interface gre>
```

第二步，现在只需要为隧道接口配置 ip 地址和路由，路由器 site1 配置：

```
/ip address
add address=172.16.1.1/30 interface=myGre

/ip route
add dst-address=10.1.202.0/24 gateway=172.16.1.2
```

路由器 site 2 配置：

```
/ip address
add address=172.16.1.2/30 interface=myGre

/ip route
add dst-address=10.1.101.0/24 gateway=172.16.1.1
```

这样两点之间的三层路由通过 GRE 隧道打通，可以相互访问

28.2 IPIP 隧道

操作路径: /interface ipip

IPIP 隧道一个简单隧道协议，将 IP 数据报封装到 IPIP 隧道连接两端路由器。大多路由都支持该协议包括思科和 Linux。

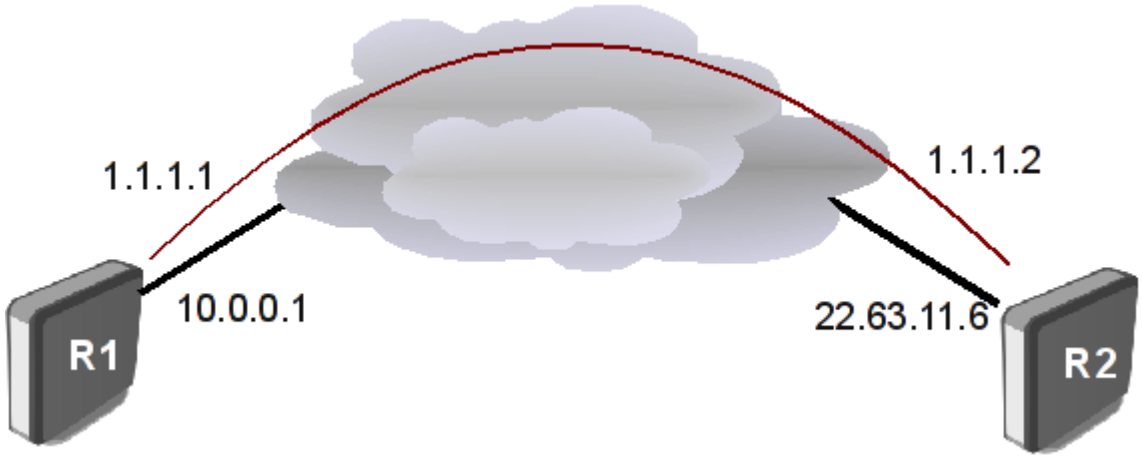
属性

属性	描述
clamp-tcp-mss (*yes* | *no*; 默认: **yes**)	控制是否改变接收到的 TCP SYN 报文的 MSS 大小。启用该功能后，如果当前 MSS 大小超过隧道接口 MTU(考虑到 TCP/IP 开销)，路由器将改变接收到的 TCP SYN 报文的 MSS 大小。接收到的封装报文仍然包含原来的 MSS，只有在解封装后，MSS 才会改变。
dscp (*inherit* | *integer [0-63]*; 默认:)	设置 dscp 值到 IPIP 包头，或从本地 dscp 继承到隧道传输中。
ipsec-secret (*string*; 默认:)	当指定 ipsec-secret 时，路由器将通过动态方式，将 ipsec 对等体添加到 remote-address，使用预共享密钥和缺省值的策略(缺省 phase2 使用 sha1/aes128cbc)。
local-address (*IP*; 默认:)	本地路由器的 IP 地址（外网口地址）
mtu (*integer*; 默认: **1500**)	三层网络最大传输单元
keepalive (*integer[/time],integer 0..4294967295*; 默认: **10s,10**)	keepalive 参数用于设置隧道运行保持的时间间隔，在此时间间隔内，即使隧道对端发生故障，隧道运行标志"R"仍将保持。如果超过了时间范围，重试失败，则删除接口运行标志"R"。参数填写格式为:"KeepaliveInterval"、"KeepaliveRetries"(其中"KeepaliveInterval"为时间间隔)、"KeepaliveRetries -重试次数"。缺省情况下，keepalive 设置为 10 秒，重试 10 次。
name (*string*; 默认:)	接口名称
remote-address (*IP*; 默认:)	对端 IPIP 隧道路由器的 IP 地址

注意：IPIPv6 隧道基于 IPv6 网络，从 v5rc6 版本开始加入，操作路径/interface ipipv6 ，配置属性与 IPv4 版本完全相同

配置事例

下面是一个简单的 IPIP 隧道配置，通过 IPIP 隧道连接互联网的 R1 和 R2 路由



IPIP 配置与 GRE 类似，同样先配置 IPIP 接口和 IP 地址

配置路由器 R1：

```
[admin@MikroTik] interface ipip> add local-address: 10.0.0.1 remote-address: 22.63.11.6
[admin@MikroTik] interface ipip> print
Flags: X - disabled, R - running
  #      NAME                                MTU    LOCAL-ADDRESS     REMOTE-ADDRESS
  0 X    ipip1                               1480   10.0.0.1          22.63.11.6

[admin@MikroTik] interface ipip> en 0
[admin@MikroTik] interface ipip> /ip address add address=1.1.1.1/24 interface=ipip1
```

配置路由器 R2

```
[admin@MikroTik] interface ipip> add local-address=22.63.11.6 remote-address=10.0.0.1
[admin@MikroTik] interface ipip> print
Flags: X - disabled, R - running
  #      NAME                                MTU    LOCAL-ADDRESS     REMOTE-ADDRESS
  0 X    ipip1                               1480   22.63.11.6        10.0.0.1

[admin@MikroTik] interface ipip> enable 0
[admin@MikroTik] interface ipip> /ip address add address=1.1.1.2/24 interface=ipip1
```

现在使用 ping 检查网络是否连接成功

```
[admin@MikroTik] interface ipip> /ping 1.1.1.2
1.1.1.2 64 byte ping: ttl=64 time=24 ms
1.1.1.2 64 byte ping: ttl=64 time=19 ms
```

```
1.1.1.2 64 byte ping: ttl=64 time=20 ms
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 19/21.0/24 ms
[admin@MikroTik] interface ipip>
```

对于 GRE 和 IPIP 隧道，从网络解决方案来说 GRE 的容错性更高，不过他们两都要求使用固定 IP 连接（或使用 DDNS 域名完成动态 IP 解析），因为他们要求对等连接，特别对于互联网连接相对灵活性不如 PPTP、L2TP、OVPN 和 SSTP，后者是服务端与客户端概念，仅要求服务端在互连网即可，具体解决方案由实际环境而定。

第二十九章 OSPF

29.1 OSPF 介绍

OSPF(Open Shortest Path First 开放式最短路径优先)是一个内部网关协议(Interior Gateway Protocol,简称 IGP)，用于在单一自治系统(autonomous system,AS)内决策路由。与 RIP 相比，OSPF 是链路状态路由协议，而 RIP 是距离向量路由协议。

链路是路由器接口的另一种说法，因此 OSPF 也称为接口状态路由协议。OSPF 通过路由器之间通告网络接口的状态来建立链路状态数据库，生成最短路径树，每个 OSPF 路由器使用这些最短路径构造路由表。

OSPF 路由协议是一种典型的链路状态（Link-state）的路由协议，一般用于同一个路由域内。在这里，路由域是指一个自治系统（Autonomous System），即 AS，它是指一组通过统一的路由策略或路由协议互相交换路由信息的网络。在这个 AS 中，所有的 OSPF 路由器都维护一个相同的描述这个 AS 结构的数据库，该数据库中存放的是路由域中相应链路的状态信息，OSPF 路由器正是通过这个数据库计算出其 OSPF 路由表的。

作为一种链路状态的路由协议，OSPF 将链路状态广播数据报 LSA（Link State Advertisement）传送给在某一区域内的所有路由器，这一点与距离向量路由协议不同。运行距离向量路由协议的路由器是将部分或全部的路由表传递给与其相邻的路由器。

OSPF 链路状态技术相对于向量路由协议 RIP 有多个优势：

- 没有跳跃数限制；
- 多播地址被用于发送路由信息更新；
- 更新紧当在网络拓扑变化时被发送；
- 逻辑网络的定义，路由器被分成多个区域；
- 传输和标记扩展路由被注入到 AS。

OSPF 也自身的一些缺点：

- OSPF 需要相当的 CPU 和内存，这是由于 SPF 算法和多路径信息的维护；
- 与 RIP 相比很多复杂的协议需要设置。

MikroTik RouterOS 支持 OSPF version 2 (RFC 2328)和 version 3 (RFC 5340, OSPF for IPv6)。OSPF 可以实现多个区域（Area）管理，默认的核心区域是 area0（0.0.0.0），区域 0 也称为骨干区域（backbone）

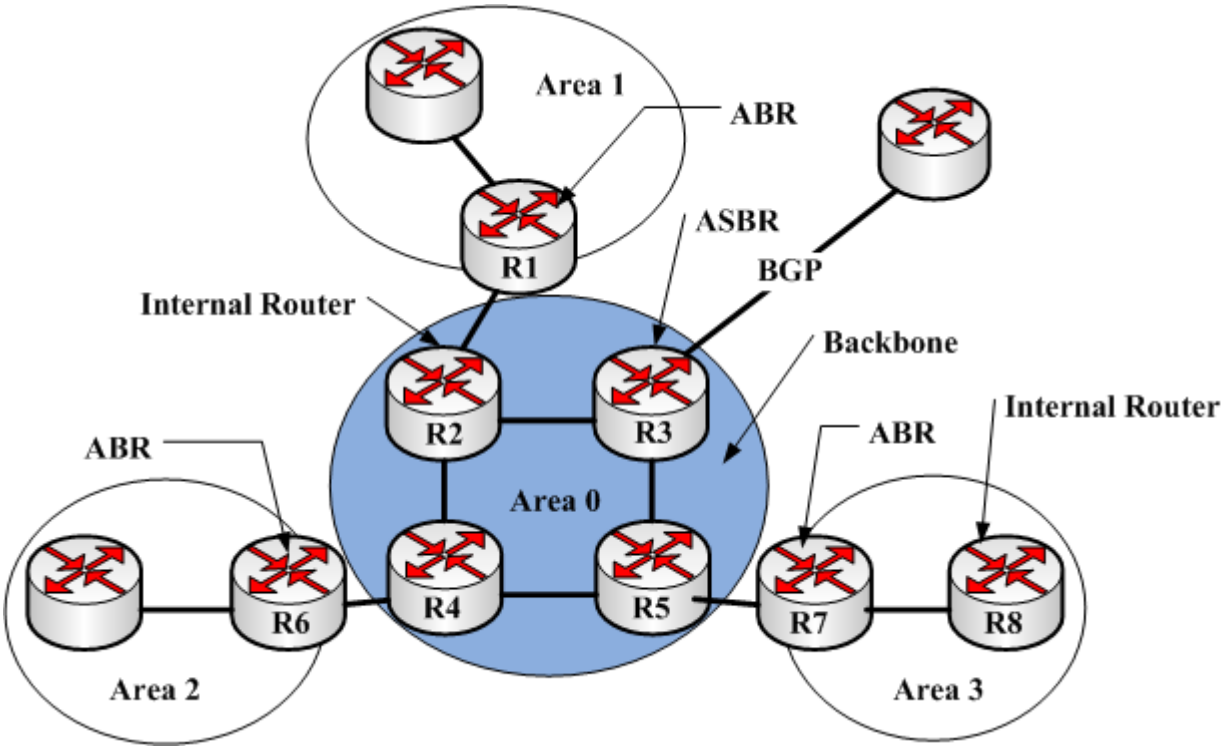
OSPF 术语

相关的 OSPF 运行术语

- Neighbor – 直接连接运行 OSPF 的路由器，相邻且相同区域。邻居间通过 Hello 包找到对方
- Adjacency – 两台 OSPF 路由器能够形成邻居，但并不一定能相互交换 LSA，只要能交换 LSA，关系则称为邻接（Adjacency）。邻居之间只交换 Hello 包，而邻接（Adjacency）之间不仅交换 Hello 包，还要交换 LSA
- Interface – 物理网络接口

- LSA - 链路状态（LSA）就是 OSPF 接口上的描述信息，例如接口上的 IP 地址，子网掩码，网络类型，Cost 值等等，OSPF 路由器之间交换的并不是路由表，而是链路状态（LSA），OSPF 通过获得网络中所有的链路状态信息，从而计算出到达每个目标精确的网络路径。
- DR – 直连路由器，
- BDR – 备份直连路由器
- Area – 区域，用于建立一个分级网络
- ABR – 区域边界路由器，连接多个区域的路由器
- ASBR – 自治系统边界路由器，连接外部路由协议，即位于 OSPF 自主系统和非 OSPF 网络之间
- NBMA - 非广播多路访问网络
- Broadcast – 网络广播
- Point-to-point – 点对点连接，排除需要 DRs 和 DBRs 的网络类型
- Router-ID – OSPF 路由器身份识别的 IP 地址，如果 OSPF 的 Router-ID 没有手动配置，路由器会使用一个以分配的 IP 地址作为 Router-ID
- Link State – 链路状态，定义路由之间接口和邻居路由的关系状态
- Cost – 连接状态协议为每一个连接分配一个值 cost，cost 值计算基于接口的速率，
- Autonomous System – 自治系统，一组路由器使用相同的路由协议交换路由信息

以上 OSPF 术语是理解 OSPF 运行的重要内容，这些将涉及到该章节所有内容。为了更好的理解这些内容，通过以下视图理解什么是 ABR、ASBR、Area 和 Internal Router



从上面的视图可以看到 R1、R6 和 R7 都是 ABR 同时连接了两个区域，R3 则为 ASBR 与外部的路由协议连接，Area0 为核心区域 Backbone，R2 和 R8 路由器所有接口都属于同一区域即是 Internal Router

LSA 类型

OSPF 的 LSA 类型种类繁多，OSPF 又是目前应用最广泛的 IGP 协议，所以不得不对它进行了解，LSA 有以下几种类型：

- **Type1:Router LSA**：每个路由器都将产生 Router LSA，这种 LSA 只在本区域内传播，描述了路由器所有

的链路和接口，状态和开销.

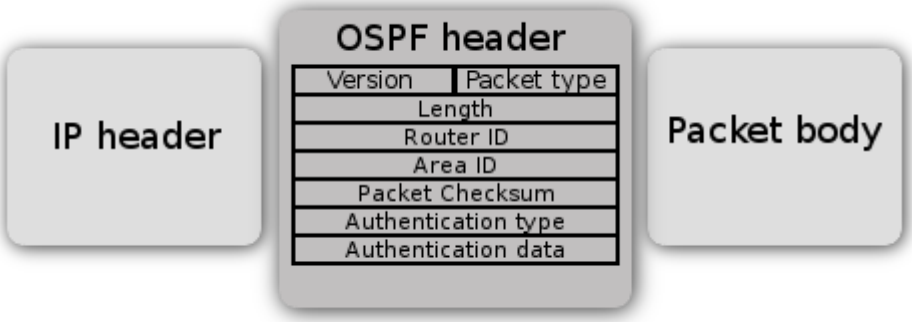
- **Type2:Network LSA**：在每个多路访问网络中，DR 都会产生这种 Network LSA，它只在产生这条 Network LSA 的区域泛洪描述了所有和它相连的路由器（包括 DR 本身）。
- **Type3:Network Summary LSA**：由 ABR 路由器始发，用于通告该区域外部的目的地址.当其他的路由器收到来自 ABR 的 Network Summary LSA 以后，它不会运行 SPF 算法，它只简单的加上到达那个 ABR 的开销和 Network Summary LSA 中包含的开销,通过 ABR，到达目标地址的路由和开销一起被加进路由表里，这种依赖中间路由器来确定到达目标地址的完全路由（full route）实际上是距离向量路由协议的行为。
- **Type4:ASBR Summary LSA:**由 ABR 发出，ASBR 汇总 LSA 除了所通告的目的地是一个 ASBR 而不是一个网络外，其他同 Network Summary LSA.
- **Type5:AS External LSA:**发自 ASBR 路由器，用来通告到达 OSPF 自治系统外部的目的地，或者 OSPF 自治系统那个外部的缺省路由的 LSA.这种 LSA 将在全 AS 内泛洪（4 个特殊区域除外）
- **Type6:Group Membership LSA**
- **Type7:NSSA External LSA:**来自非完全 Stub 区域（not-so-stubby area）内 ASBR 路由器始发的 LSA 通告它只在 NSSA 区域内泛洪，这是与 LSA-Type5 的区别.
- **Type8:External Attributes LSA**
- **Type9:Opaque LSA(link-local scope,)**
- **Type10:Opaque LSA(area-local scope)**
- **Type11:Opaque LSA(AS scope)**

每个路由器的链路状态数据库知道该路由区域有多少路由器，路由器有多少网络接口，路由器之间用什么样的网络连接，每条连接的成本等等，OSPF 网络在完成连接前有以下几个步骤：

- 邻居探测
- 数据库同步
- 路由计算

OSPF 路由通信

OSPF 运行在 IP 网络层（第三层），使用协议编号为 89，通信建立是将对方路由器 IP 地址设置为邻居 IP 地址或 OSPF 组播地址 AllSPF Routers (224.0.0.5) 或 AllDR Routers (224.0.0.6)，每个 OSPF 数据报都包含标准的 24byte 包头



字段	描述

Packet type	这里有几种 OSPF 数据报类型：Hello 数据报，Database Description (DD)数据报，Link state request 数据报，link State Update 数据报和 Link State Acknowledgment 数据报。
Router ID	路由器的一个 IP 地址，需手动配置到路由器
Area ID	允许 OSPF 路由器关联的数据报到指定的 OSPF 区域
Checksum	校验数据报在传输中是否受损，决定是否接收。
Authentication fields	这个字段用于核实路由器接收到的数据报内容没有被修改，且数据报确定来至正确的 Router ID 路由器

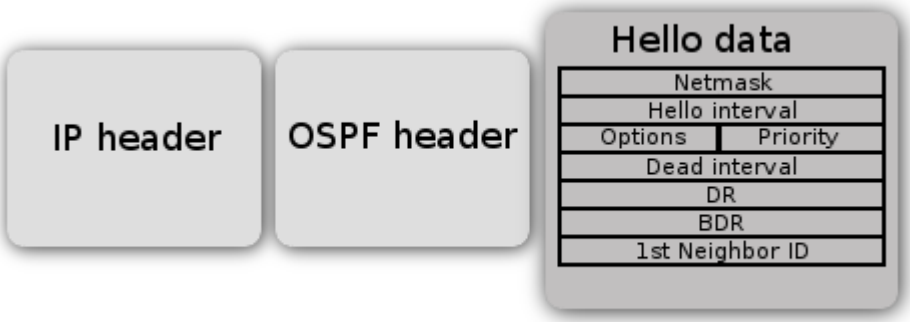
这里有 5 种不同的 OSPF 数据报类型用于确认 LSA 在 OSPF 网络中泛洪。

- **Hello packet** - 用于探测 OSPF 邻居和建立邻接关系
- **Database Description (DD)** - 检查路由器数据库之间同步，当邻接关系建立后交换数据
- **Link-State Request (LSR)** - 用于请求更新邻居的数据库，即向对方请求所需的 LSA，设备只有在 OSPF 邻居双方成功交换 DD 报文后才会向对方发出 LSR 报文
- **Link-State Update (LSU)** - 传递一些指定 LSA 连接状态请求，即向对方发送其所需要的 LSA
- **Link-State Acknowledgment (LSack)** - 用于对收到的 LSA 进行确认

邻居探测

邻居探测通定期从配置接口发送 OSPF Hello 包，Hello 包发送周期为 10 秒，该周期可以通过设置 hello interval 来修改，当路由器学习到一个存在的邻居路由发送的 hello 包，同样会发送一个 Hello 包着为回应。

传输和接收 Hello 包允许路由器对邻居探测做失败处理，如果 Hello 包在死亡周期（Dead interval 为 40 秒），路由器认为连接失败。Hello 协议是为确保相邻路由器之间在 Hello 周期和 Dead 周期内协商一致。如果没有在该时间周期内网完成会，将邻居定义为失效，即 link down。



字段	属性
network mask	发起路由器的接口 IP 地址的子网掩码
hello interval	Hello 包周期，默认 10 秒
options	OSPF 的邻居信息选项
router priority	一个 8bit 值用于帮助选择 DR 和 BDR（无法用于 p2p 连接）

router dead interval	在指定时间周期内未收到邻居 hello 包，便认为连接中断，默认大于 4 次 hello interval 值，即 40 秒
DR	当前 DR 的 router-id
BDR	当前 BDR 的 router-id
Neighbor router IDs	所有邻居的 router-ids 指令清单

注意：Network mask、 Priority、DR 和 BDR 字段被使用只有在邻居连接通过 broadcast 或 r NBMA 网段的情况下

两个路由器无法建立邻居关系可能会有一些原因：

- 路由器之间可能存在两条路径，认为有 Hello 包泛洪攻击
- 接口需要属于相同 area；
- 接口需要属于同一子网内；
- 如果要求验证，路由应该使用相同的验证选项，且密码应该相同；
- Hello 和 Dead 周期应在相同 Hello 数据报中；
- 外部路由和 NSSA 标识应在相同 Hello 数据报中。

数据库同步

链路状态数据库在 OSPF 路由器之间同步非常重要，下面有两类数据库同步：

- 初始数据库同步
- 可靠泛洪.

当两个邻居第一次连接成功，将做初始数据库同步。数据库无法实现同步会导致错误的路由表计算，使路由环路或黑洞路由，邻居之间的第一次连接 OSPF 确认数据库下载，这个步骤称为数据库交互（Database exchange）。

OSPF 路由器发送 LSA 在一系列的 OSPF DD（Database Description）数据报中。路由器每次发送 DD 数据报都需要等前一数据报是否确认达到，确认后在发生。 当所有 DD 数据报被收到，路由器需要更新 LSAs，这时会发送 Link-State Request (LSR) 数据报更新 LSAs，邻居以泛洪的方式通过 Link-state Upade（LSU）数据报响应 LSAs，当所有更新收到，邻居会收到 Link-State Acknowledgment (LSack)数据报，用于 LSAs 的确认完全邻接 Fully adjacent

可靠泛洪是另一种数据库同步模式，即当邻接关系已经建立，OSPF 路由器用于确认其他路由 LSA 是否改变。例如当 OSPF 路由器收到 Link State Update 数据报，会发生一个确认数据报给发生者， 重新打包 LSA 到 LSU 数据报，并发送到所有接口。

路由表计算

当链接状态数据库同步完成，OSPF 路由器就能计算路由表，链路状态数据库描述了路由器与连结之间关联，如果转发等，也包含了每个连结的成本（metric）。 Metric 被用于计算到达目标网络的最短路径

每个路由器都能宣告在自己不同方向的成本值，可以定义非对称的链路（数据报到目标地是一条路径，但响应则是不同路径），非对称路径并不推荐，会使在查找路由问题时变得很困难，成本值在 RouterOS 默认被设置为 10，该值可以修改，例如下面添加 ether2 接口成本值为 100

```
/routing ospf interface add interface=ether2 cost=100
```

对于 Cisco 路由器的接口成本计算是接口带宽的反比，高带宽获得低成本值，如下公式：

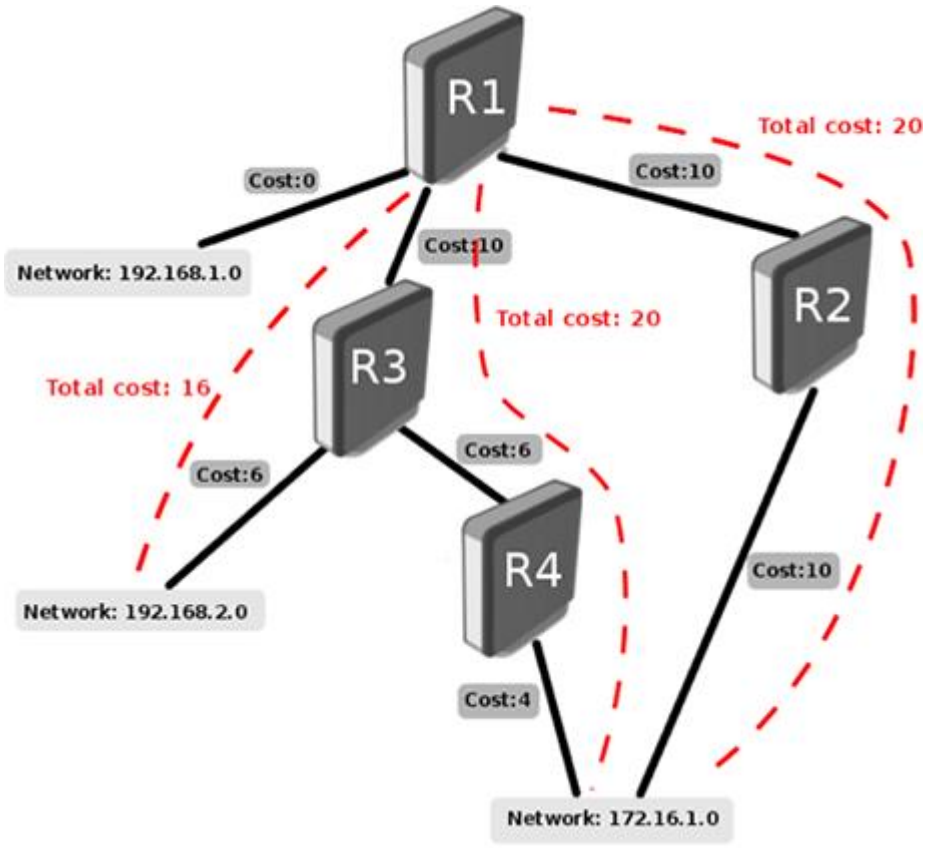
Cost = 100000000/带宽（bps）

OSPF 使用迪杰斯特拉最短路径优先（SPF）算法计算最短路径，该算法将路由器放置在一个树形结构的根，计算到目标累计成本的最短路径，每个路由器计算自己的树形，即使所有路由都使用相同的链路状态数据库。

SPT 计算

假设我们有以下的网络结构，网络包含 4 台路由器，对于 R1 为了建立最短路径树，我们需要将 R1 放到根位置，并计算到每个目标的最短成本

4 台路由器的



如同你从上图看到多条相同成本的最短路径到达 172.16.1.0 网络，这样可以允许负载均衡的方式到达目标，这种被称为 equal-cost multipath (ECMP)。在最短路径树建立后，路由器开始建立相应的路由表

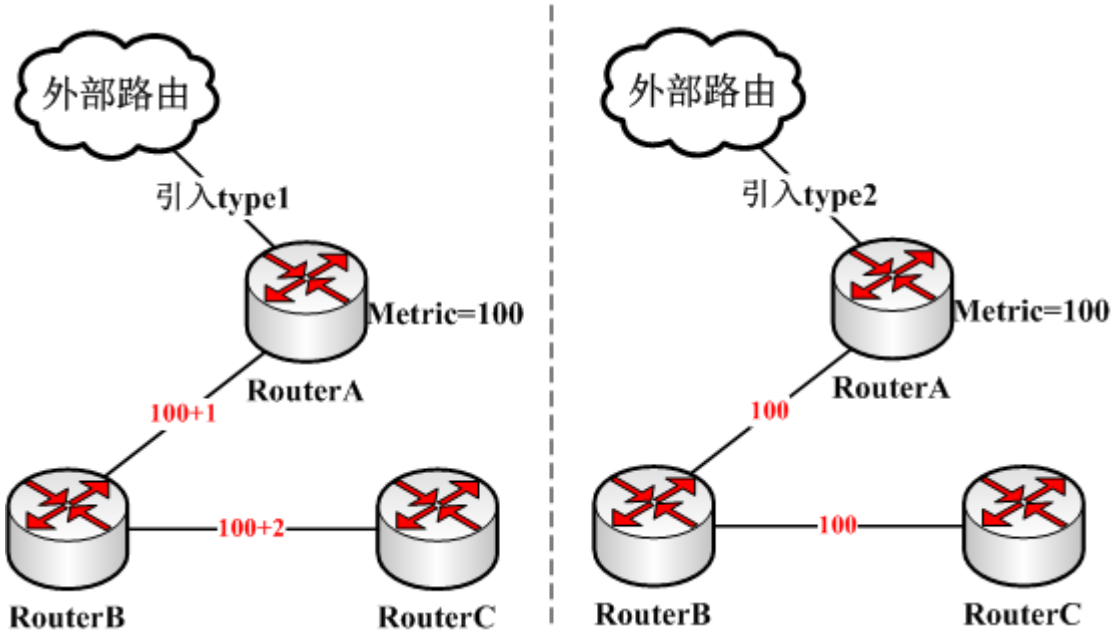
OSPF 路由类型

OSPF 引入外部路由有两种类型 type1 和 type2,这里的。

- **type1** 的计算方法是外部开销（即重发布路由时开销）再加上路由器到 ASBR 的开销，即到达 ASBR 的花费+metric 值
- **type2** 的计算方法是开销=外部开销，即路由器到达外部路由的花费就是 LSA 所携带的 metric 值

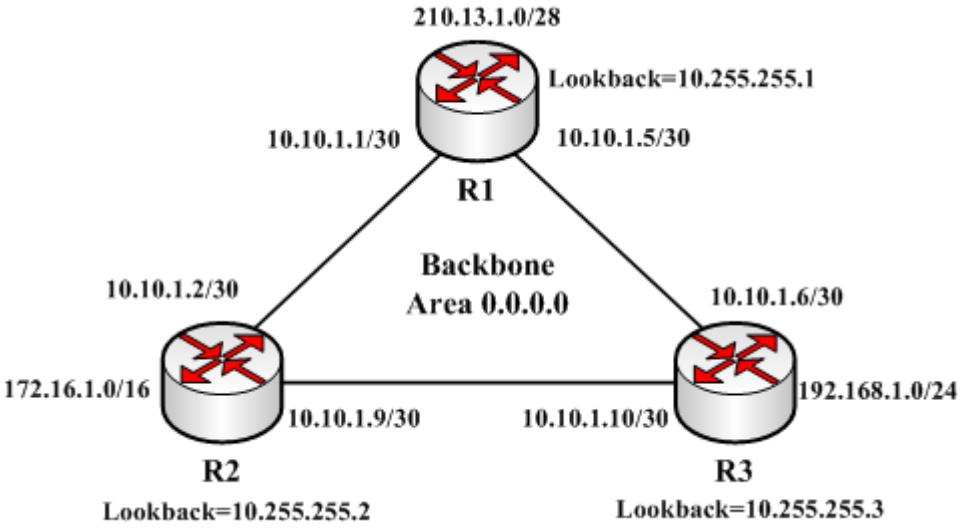
在 OSPF 选路时，去往同一目标网段的 OSPF 路由，开销相同的情况下，type1 的要优于 type2 的开销

例如：有路由器 A、B、C 如下图，RouterA 引入一个外部路由到 ospf，设定 metric 为 100，默认引入的时候是 type2，这样 B 计算出的成本开销就是 100，C 也是 100，如果设置成 type1 的，B 计算的是 100+1=101，C 计算的是 100+2=102。



29.2 基本的 OSPF 配置

下面举例如何配置一个简单的 OSPF 网络，假设以下网络有路由器 R1、R2 和 R3：



根据网络情况，我们需要将 3 台路由器通过 10.10.1.0 划分的 3 个 30 位掩码子网段互连，3 个路由器下分别有 210.13.1.0/28、172.16.1.0/16 和 192.168.1.0/24 本地地址段，需要通过 OSPF 发布并，实现路由互访。

网络配置

首先配置路由器之间互连的 IP 地址，配置如下，路由器 R1：

```
[admin@MikroTikR1]/ip address add address=10.10.1.1/30 interface=ether1
[admin@MikroTikR1]/ip address add address=10.10.1.5/30 interface=ether2
[admin@MikroTikR1]/ip address add address=210.13.1.1/28 interface=ether3
```

路由器 R2

```
[admin@MikroTikR2]/ip address add address=10.10.1.2/30 interface=ether1
[admin@MikroTikR2]/ip address add address=10.10.1.9/30 interface=ether2
[admin@MikroTikR2]/ip address add address=172.16.1.1/16 interface=ether3
```

路由器 R3

```
[admin@MikroTikR3]/ip address add address=10.10.1.6/30 interface=ether1
[admin@MikroTikR3]/ip address add address=10.10.1.10/30 interface=ether2
[admin@MikroTikR3]/ip address add address=192.168.1.1/24 interface=ether3
```

配置 router-id

Instance 作为一个 OSPF 配置实例，在***/routing ospf instance*** 菜单下。对于高级的 OSPF 设置，需要运行多个 OSPF instances，该网络中每台路由器只有一个 instance 配置，我只需要启用默认 instance，在 instace 中配置 router-id

在 R1 中查看默认的 instance，其他两台路由器也是相同:

```
 [admin@MikroTikR1] /routing ospf instance> print
Flags: X - disabled, * - default
 0   * name="default" router-id=0.0.0.0 distribute-default=never
       redistribute-connected=no redistribute-static=no redistribute-rip=no
       redistribute-bgp=no redistribute-other-ospf=no metric-default=1
       metric-connected=20 metric-static=20 metric-rip=20 metric-bgp=auto
       metric-other-ospf=auto in-filter=ospf-in out-filter=ospf-out
```

如同你看到的 router-id 是 0.0.0.0，意思是路由器将使用其中一个路由器 IP 地址作为 router-id。在大多事例中建议设置 loopback（还回接口）IP 地址作为 router-id。Loopback IP 地址是虚拟的，软件地址被用于网络识别，最大的好处是 loopback 地址总是存在且活动的，不会因为物理网卡连接断开而失效。 OSPF 协议将它应用在路由器通信中，通过 router-id 识别路由器。Loopback 接口配置如下：

由于 RouterOS 没有提供单独的 lookback 接口，所以创建一个 bridge 接口替代，并取名为“loopback”，建立 OSPF 的还回地址：

```
[admin@MikroTikR1] /interface bridge> add name=loopback
```

添加还回 ip 地址：

```
[admin@MikroTikR1] > ip address add address=10.255.255.1/32 interface=loopback
```

配置 router-id：

```
[admin@MikroTikR1] /routing ospf instance> set 0 router-id=10.255.255.1
```

配置分别用相同方式配置到 R2 和 R3，R2 是 10.255.255.2/32，R3 是 10.255.255.3/32

Area 配置

将三台路由器的 IP 地址段添加到 OSPF 的 network，通过 area 0 宣告到网络中，在 RouterOS 中 area 0 默认被定义 backbone

```
[admin@ MikroTikR1] /routing ospf area> print
Flags: X - disabled, I - invalid, * - default
 #      NAME                                 AREA-ID           TYPE
 0   * backbone                             0.0.0.0           defaul
[admin@ MikroTikR1] /routing ospf area>
```

R1：

```
[admin@MikroTikR1] /routing ospf network> add network=210.13.1.0/28 area=backbone
[admin@MikroTikR1] /routing ospf network> add network=10.10.1.0/30 area=backbone
[admin@MikroTikR1] /routing ospf network> add network=10.10.1.4/30 area=backbone
```

你可以规划网段，通过设置相应的子网掩码，例如分配 10.10.1.0/30, 10.10.1.4/30, 10.10.1.8/30 的网段，可以规定 OSPF 路由的范围，你也可以这样设置 OSPF 的子网：

```
[admin@MikroTikR1] /routing ospf network> add network=10.10.1.0/'''24''' area=backbone
```

R2:

```
[admin@MikroTikR2] /routing ospf network> add network=172.16.1.0/16 area=backbone
[admin@MikroTikR2] /routing ospf network> add network=10.10.1.0/24 area=backbone
```

R3:

```
[admin@MikroTikR3] /routing ospf network> add network=192.168.1.0/24 area=backbone
[admin@MikroTikR3] /routing ospf network> add network=10.10.1.0/24 area=backbone
```

通过下面操作可以核实 OSPF 操作是否生效：

- 查看 OSPF interface 菜单，确定动态项目已经被创建：

```
[admin@MikroTikR1] /routing ospf interface> print
```

- 检查你的 OSPF neighbors，DR 和 BDR 被选举

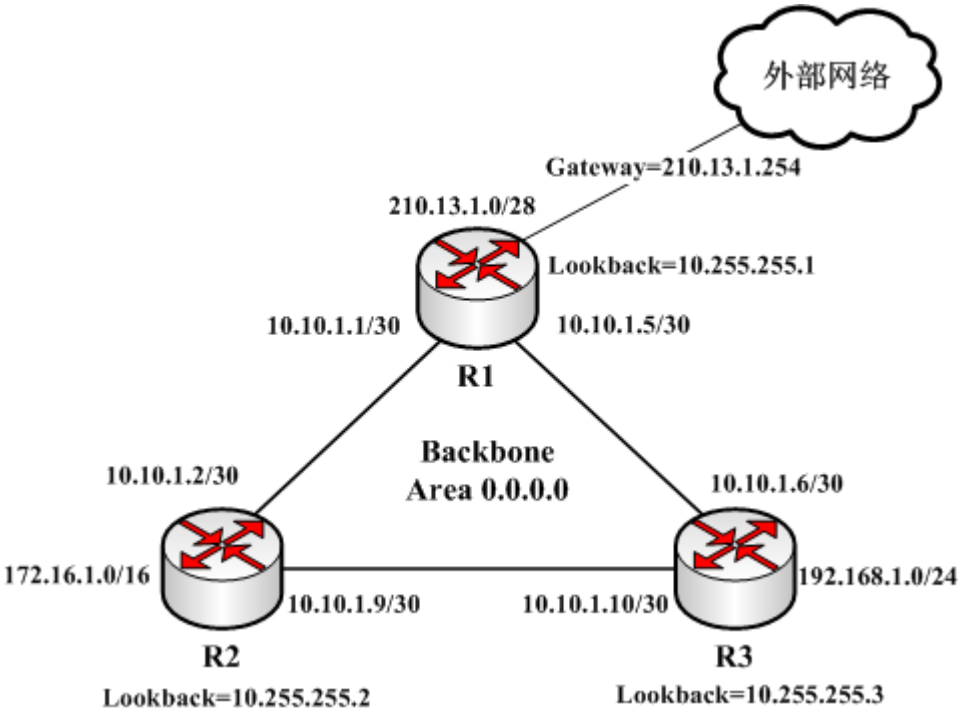
```
[admin@MikroTikR1] /routing ospf neighbor> print
```

- 创建路由器的路由表（确定 OSPF 路由存在，注意看前缀）：

```
[admin@MikroTikR1] > ip route print
```

重分布默认路由

还是按上面的事例，假设我们的 R1 作为该 OSPF 默认网关，并配置一条默认路由 210.13.1.254 与外部网络连接，通过 type2 重分布默认路由到 OSPF 网络中



添加默认路由

```
[admin@MikroTikR1] /ip route > add gateway=210.13.1.254
```

重分布默认路由到 OSPF 中

```
[admin@MikroTikR1] /routing ospf instance> set 0 distribute-default=always-as-type-2
[admin@wireless] /routing ospf instance> print
Flags: X - disabled, * - default
 0   * name="default" router-id=10.255.255.1 distribute-default=always-as-type-2
       redistribute-connected=no redistribute-static=no redistribute-rip=no redistribute-bgp=no
       redistribute-other-ospf=no metric-default=1 metric-connected=20 metric-static=20
       metric-rip=20 metric-bgp=auto metric-other-ospf=auto in-filter=ospf-in out-filter=ospf-out
```

29.3 多 area 配置

Area 0 是所有 OSPF 网络的核心区域，其他区域必须连接到 Area 0（RouterOS 称为 backbone），首先从 OSPF area 0 开始配置，然后是在是其他局域网络。

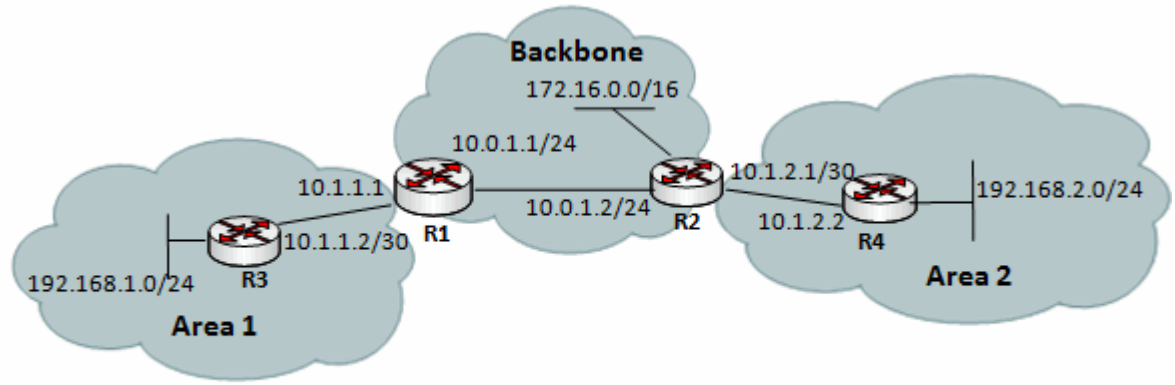


Figure 6.7. Example of multi- area OSPF network

让我们假设 IP 地址已经配置，并且默认的 OSPF instance 已启用

我们需要做下面操作：

- 创建一个 area
- 给附属的 OSPF 网络分配相应的 area

R1 配置：

```
/routing ospf area> add name=area1 area-id=0.0.0.1
/routing ospf area> add network=10.0.1.0/24 area=backbone
/routing ospf area> add network=10.1.1.0/30 area=area1
```

R2 配置：

```
/routing ospf area> add name=area2 area-id=0.0.0.2
/routing ospf area> add network=10.0.1.0/24 area=backbone
/routing ospf area> add network=10.1.2.0/30 area=area2
```

R3 配置：

```
/routing ospf area> add name=area1 area-id=0.0.0.1
/routing ospf area> add network=10.1.1.0/30 area=area1
```

R4 配置：

```
/routing ospf area> add name=area2 area-id=0.0.0.2
/routing ospf area> add network=10.1.2.0/30 area=area2
```

现在你可以检查路由表，通过命令/ip route print

在 R3 的路由表：

```
[admin@R3] > ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 1 ADo  10.0.1.0/24                        10.1.1.1           110
 2 ADC  10.1.1.0/30        10.1.1.2        ether1             110
 3 ADo  10.1.2.0/30                        10.1.1.1           110
 4 ADC  192.168.1.0/24     192.168.1.1     ether2              0
```

如同你看到的，远程网络 172.16.0.0/16 和 192.168.2.0/24 没有在路由表里，因为他们不能被 OSPF 分配，路由再发布功能允许路由协议交换路由信息，例如重新分配静态或者已经连接的 OSPF，在我们的设置里需要重分配已连接的网络，我们需要添加下面配置到路由器 R1，R2 和 R3。

```
[admin@R3] /routing ospf instance> set 0 redistribute-connected=as-type-1
[admin@R3] /routing ospf instance> print
Flags: X - disabled
 0    name="default" router-id=0.0.0.0 distribute-default=never
      <u>redistribute-connected=as-type-1</u> redistribute-static=no
      redistribute-rip=no redistribute-bgp=no redistribute-other-ospf=no
      metric-default=1 metric-connected=20 metric-static=20 metric-rip=20
      metric-bgp=auto metric-other-ospf=auto in-filter=ospf-in
      out-filter=ospf-out
```

现在检查路由器 R3，是否 192.168.2.0/24 和 172.16.0.0/16 被添加到路由表

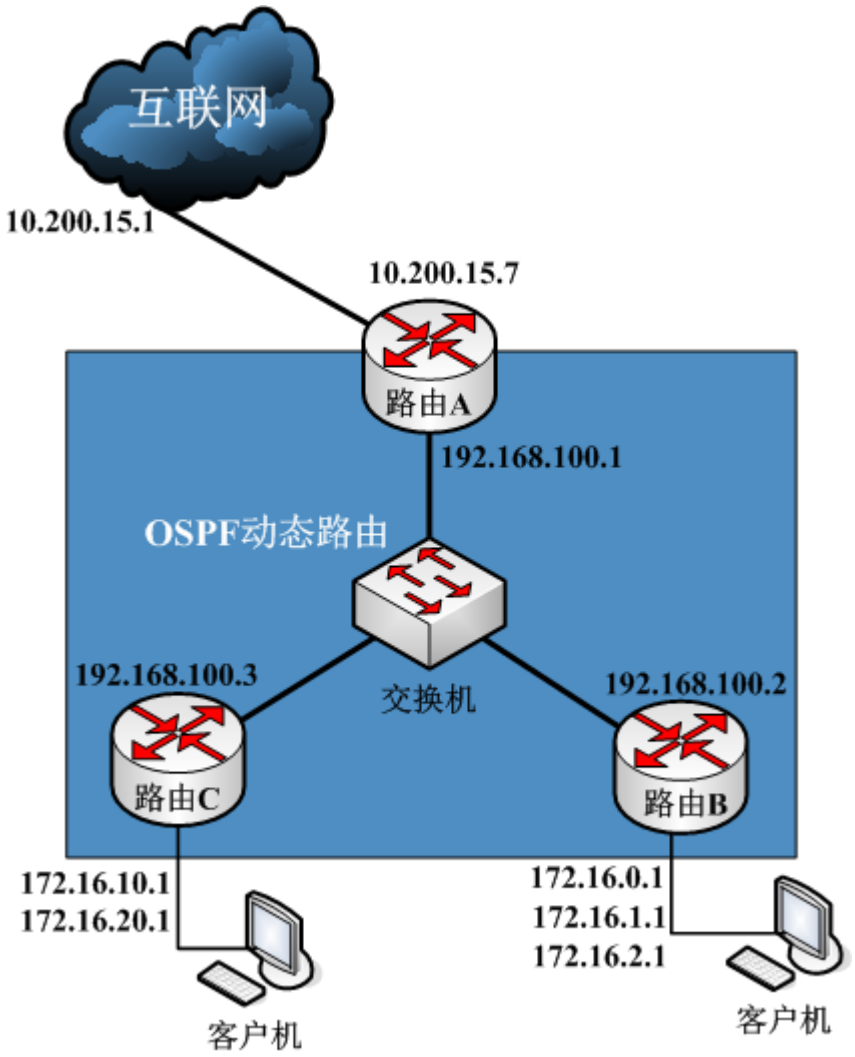
```
[admin@R3] > ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 1 ADo  10.0.1.0/24                        10.1.1.1           110
 2 ADC  10.1.1.0/30        10.1.1.2        ether1             110
 3 ADo  10.1.2.0/30                        10.1.1.1           110
 4 ADo  172.16.0.0/16                      10.1.1.1           110
 5 ADC  192.168.1.0/24     192.168.1.1     ether2              0
 6 ADo  192.168.2.0/24                     10.1.1.1           110
```

29.4 基于 PPPoE 的 OSPF 事例

在较大的内部网络里，各种设备较多，各个区域划分，使得网段不断增加，中小型 ISP 一般方法是采用静态路由，但静态路由配置比较繁琐，需要在每台机器上配置路由表做维护，每台的 IP 地址做了修改，其他几台设

备都要做相应的设置，这样在路由器不断增加的情况下，网络变动造成的调整非常大，通过建立 OSPF 也可以解决路由间备份等问题，采用动态路由变可以很好解决这样繁琐的问题

例如，下面的一个网络，由 ABC 三台路由器组成，



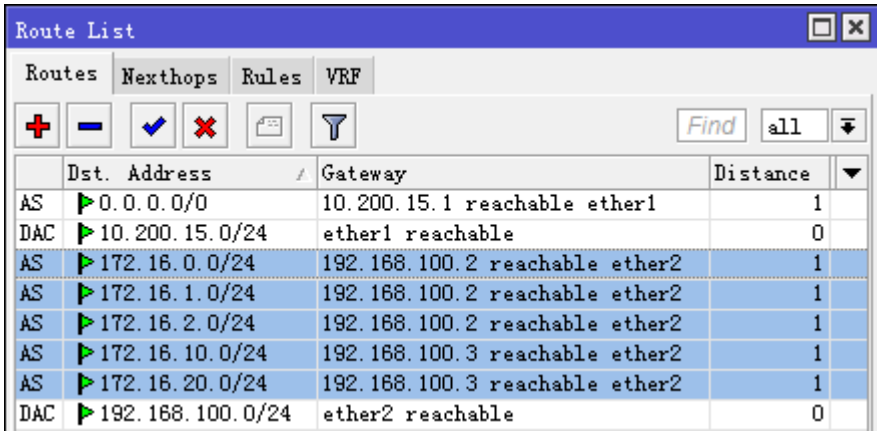
路由器 A 为接入路由器连接互联网，负责互联网接入、NAT 转发和防火墙等，采用静态路由与外网连接。

路由器 B、C 是连接内网用户，做 PPPoE 认证和每个用户的流控。

互联网地址假设为 10.200.15.7，网关为 10.200.15.1，路由器之间互联使用 192.168.100.0/24 的网段，用户网段分别是 172.16.10.1~172.16.20.1 和 172.16.0.1~172.16.2.1，

如果我们采用静态路由方式连接，那么我们需要在每台路由器，都需要在 ip route 里配置到不同网段的路由表，例如：

在路由器 A，我们需要手动添加以下路由：

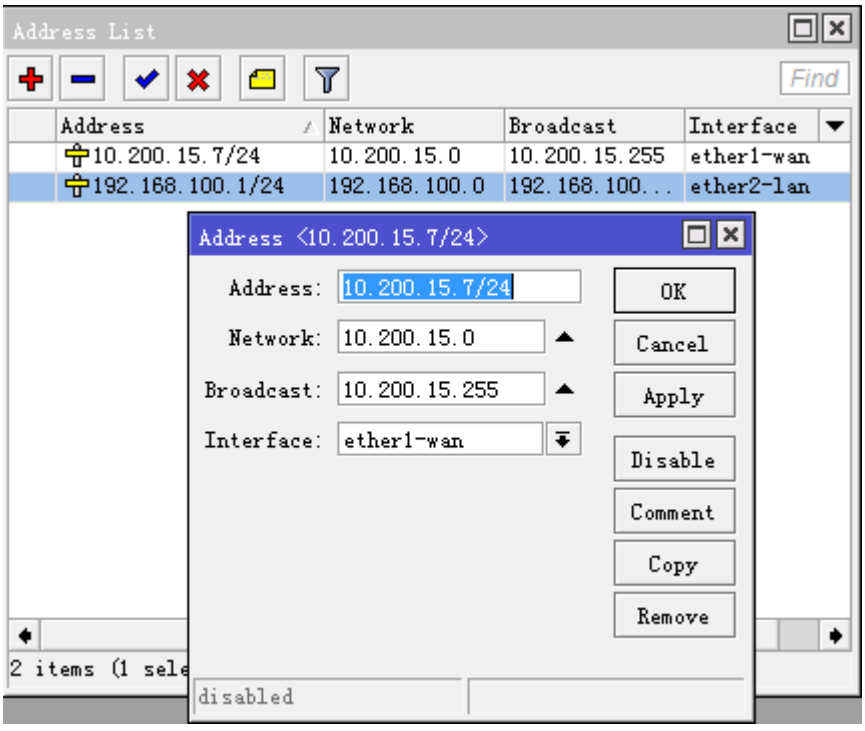


在路由 B 和 C 也要添加相应的路由，如果 IP 地址段有所变动都要修改静态路由表，而且路径也是被限制到一个出口上

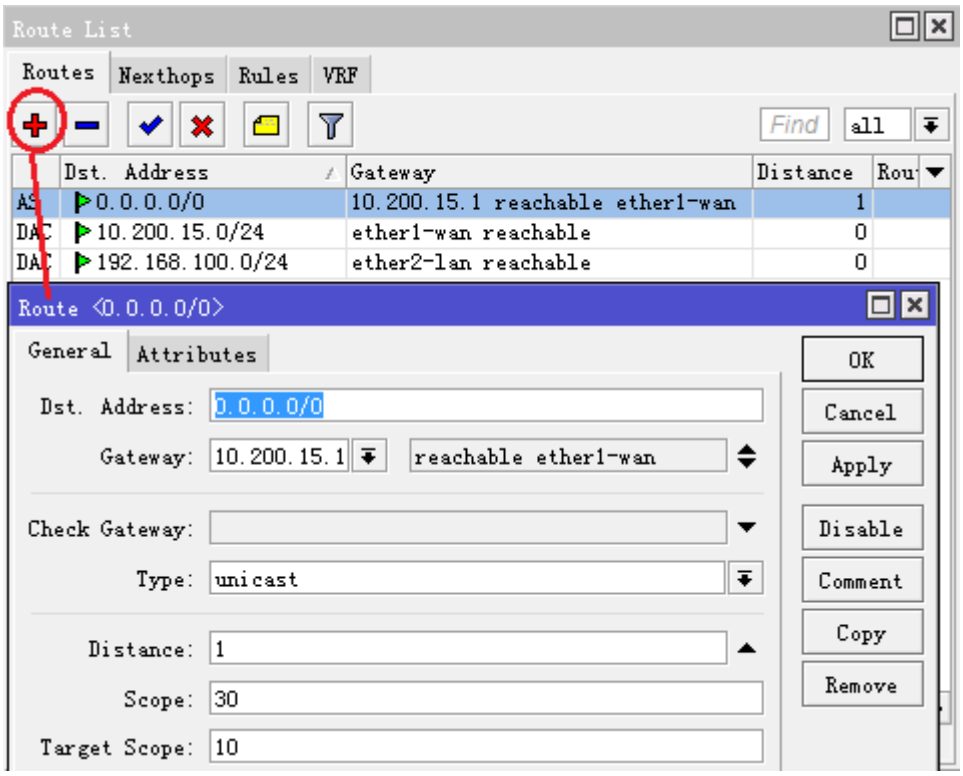
OSPF 配置

下面我们来考虑动态路由的方式，这里我们使用 OSPF，最短路由协议，好处在于每次修改 IP 地址段后，只需要声明一下网段即可，如果有多个网关出口也可以通过 OSPF 动态选择路由出口

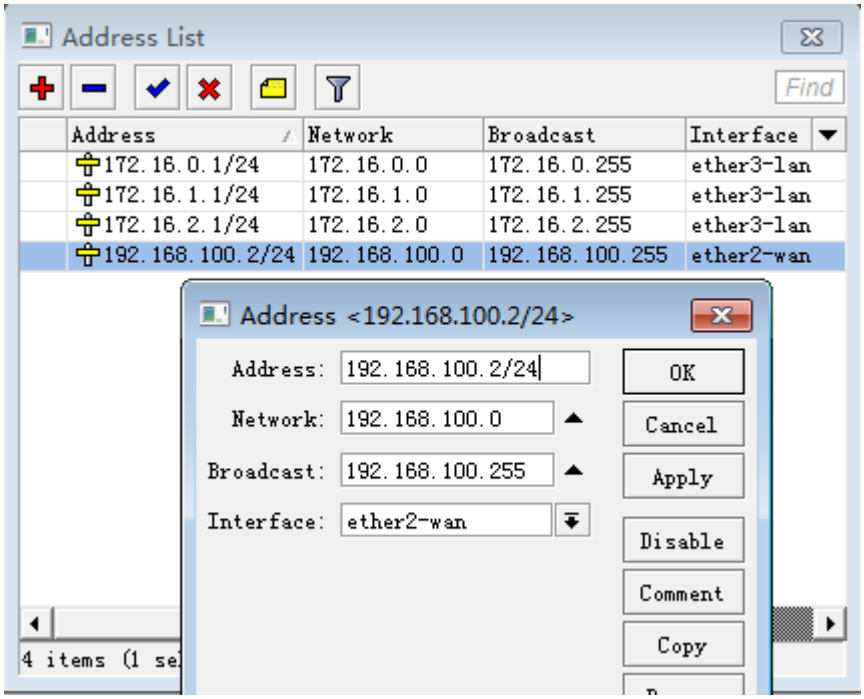
配置首先在路由器 A 配置 IP 地址和互联网的默认网关



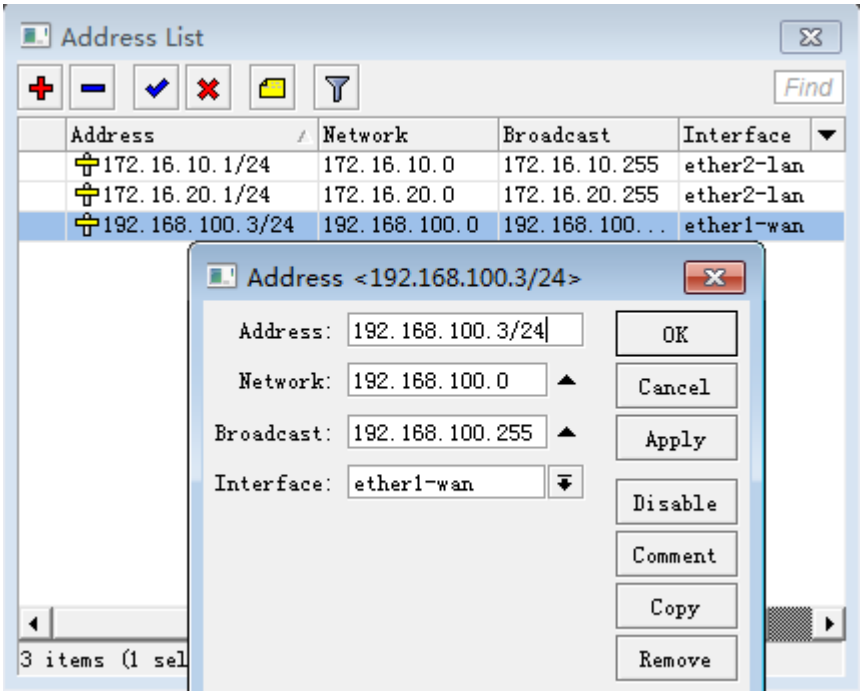
添加路由器 A 的互联网网关



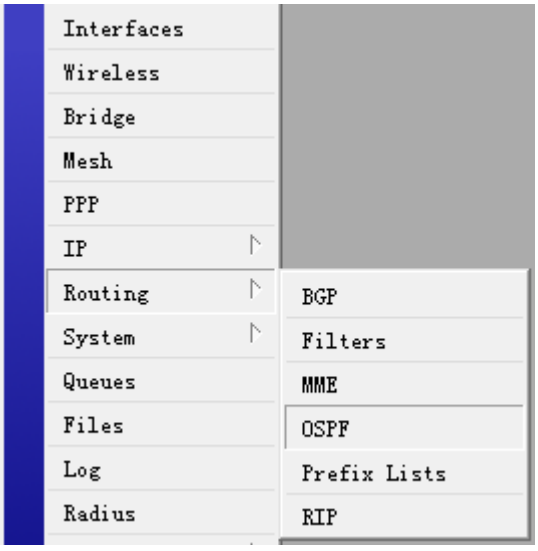
配置路由器 B 的 wan 口 IP 地址



配置路由 C 的 wan 口 IP 地址



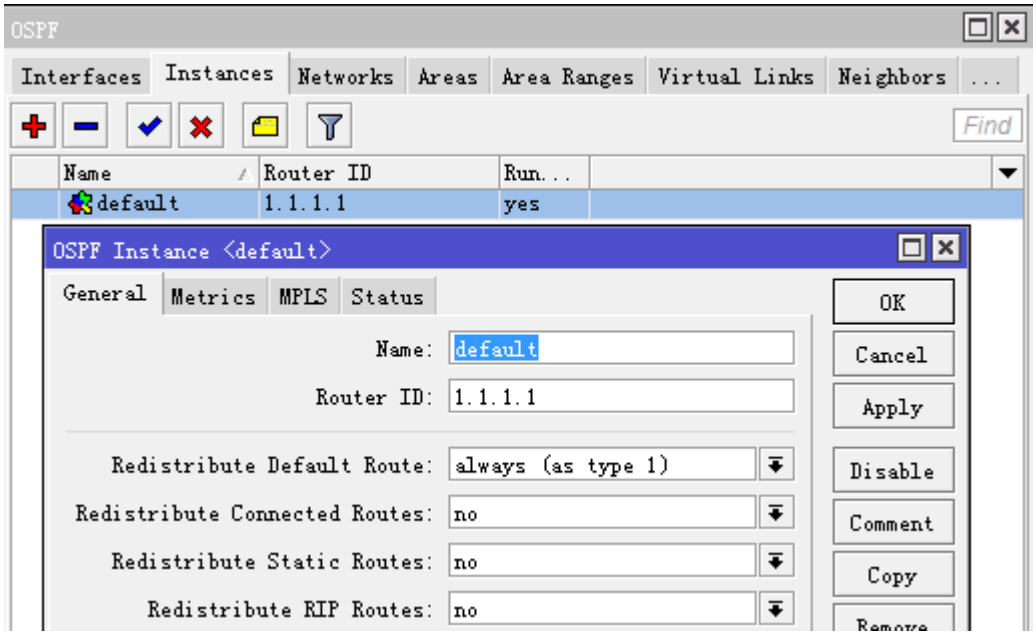
我们将 IP 地址都配置完成，接下来是我们要配置 OSPF 动态路由，我们采用的 RouterOS 版本分别是 5.0rc1 和 3.30，配置 OSPF 需要安装 routing 的功能包，在 winbox 的左边菜单可以看到 Routing 选项，选择 OSPF



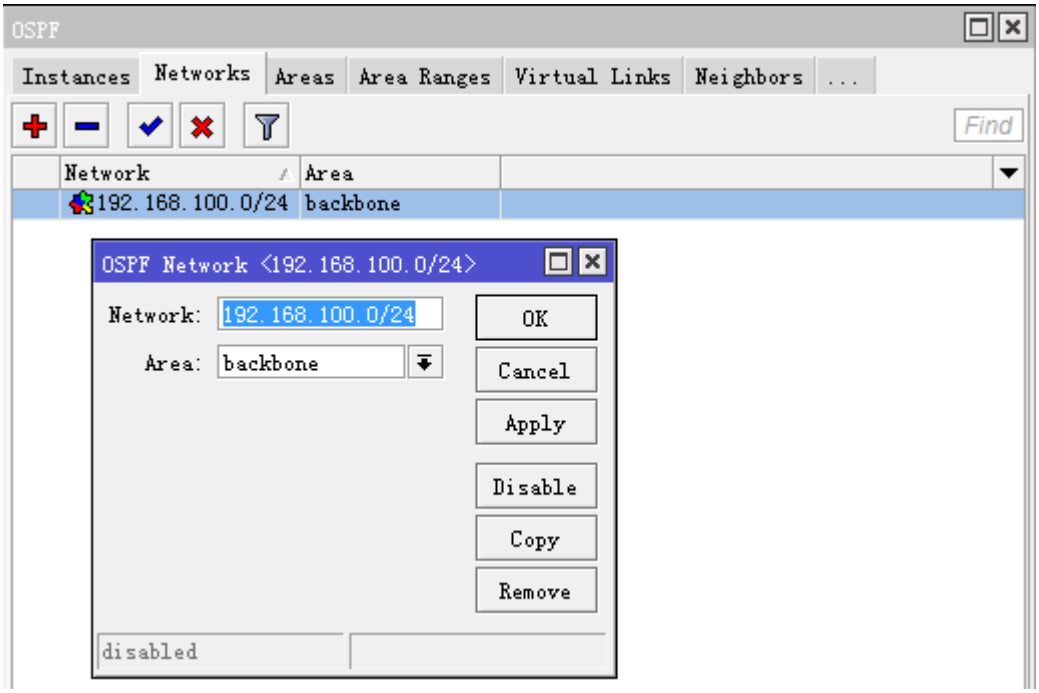
打开 OSPF 后，我们需要在 network 里添加本地路由的网段，申明自己的网络地址范围

路由器 A，

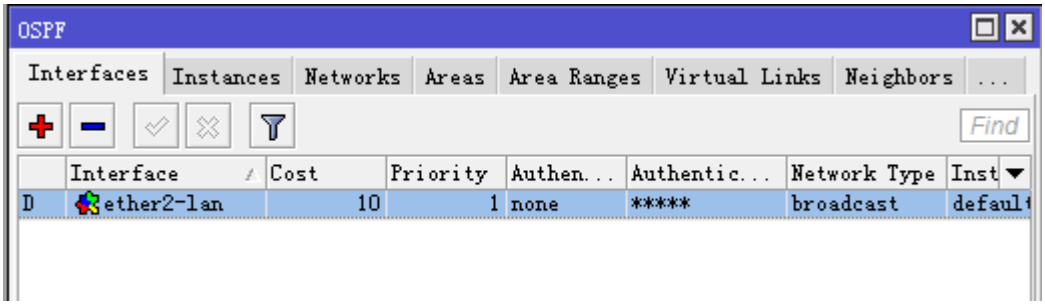
我们可以设置一个路由器的 ID 为 1.1.1.1，如果不设置也可以，OSPF 会自动选择本地接口的 IP 地址为路由器 ID，其他参数默认



添加 network 申明自己的 IP 地址，在路由器 A 只申明 lan 口的 IP 地址，wan 的互联网地址不采用 OSPF

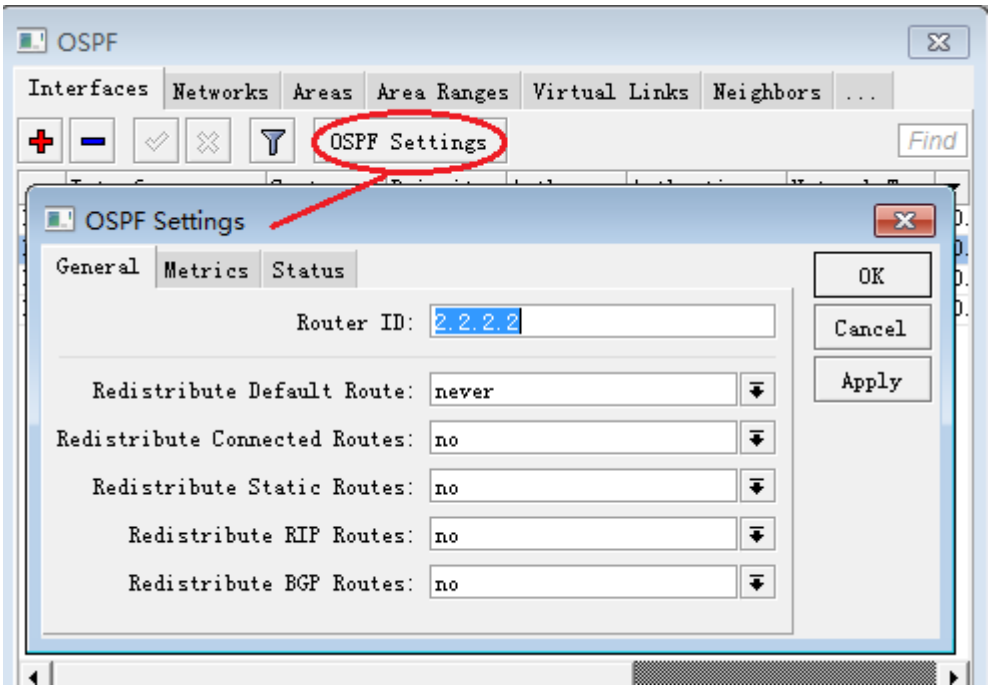


设置完成后，我们可以在 interface 里看到动态添加的接口

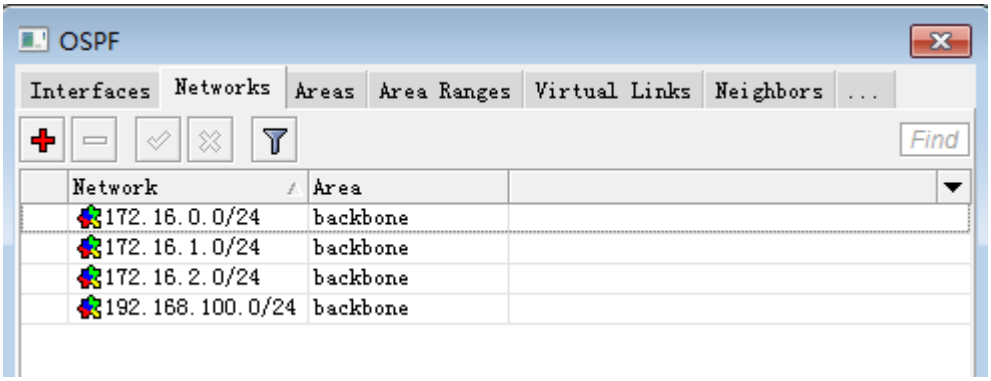


路由器 B

在这里我们也设置路由器的 ID，只是现在我们的版本是 3.30 的，和路由器 A 的 5.0rc1 不同，选择 interface 点 OSPF Settings 设置 ID 为 2.2.2.2

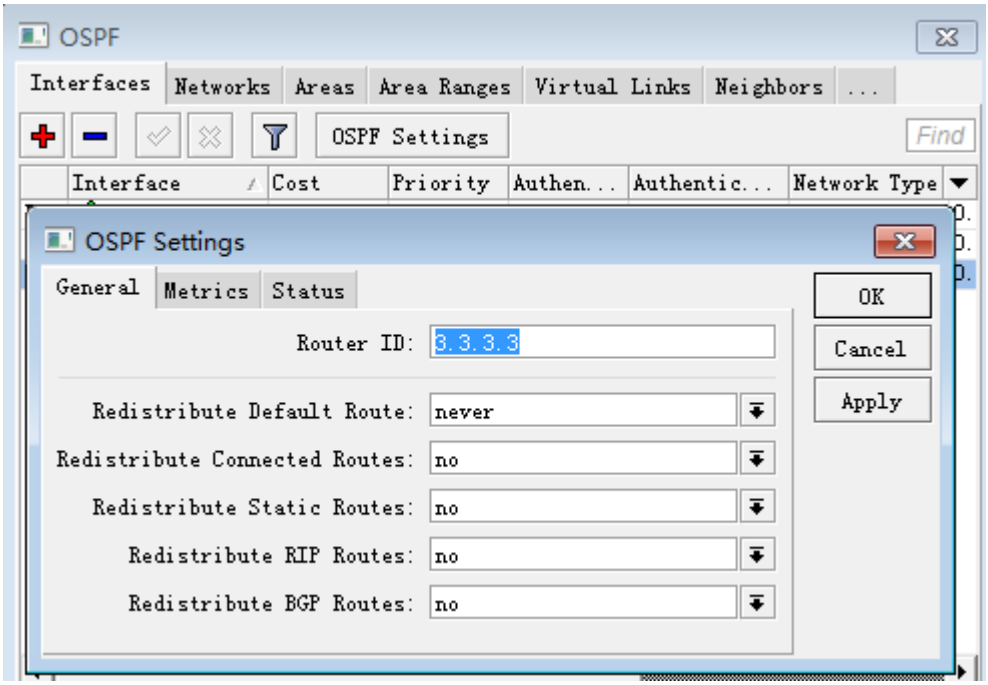


接下来我们同样设置 network 参数，申明自己的网络地址段，路由器 B 包含以下网络

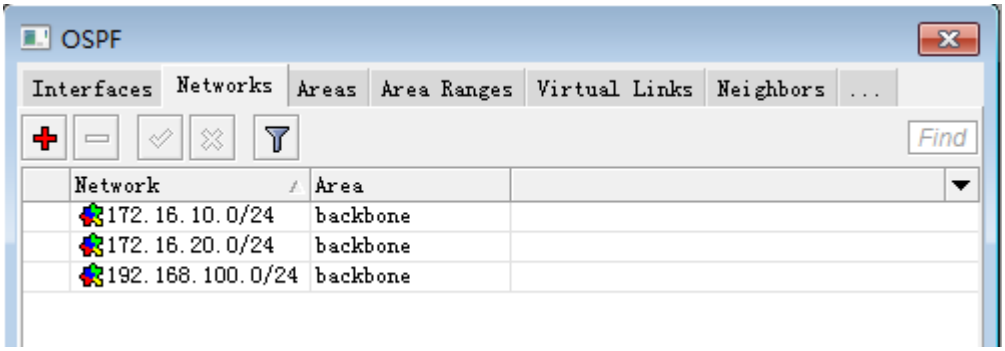


路由器 C

进入 OSPF，添加一个 OSPF 设置

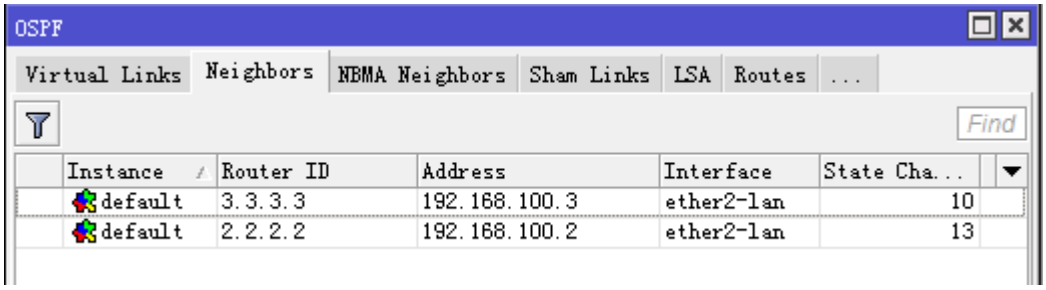


申明自己的 IP 地址段，添加本地网络的 IP 地址，我们都默认采用 Backbone 的域

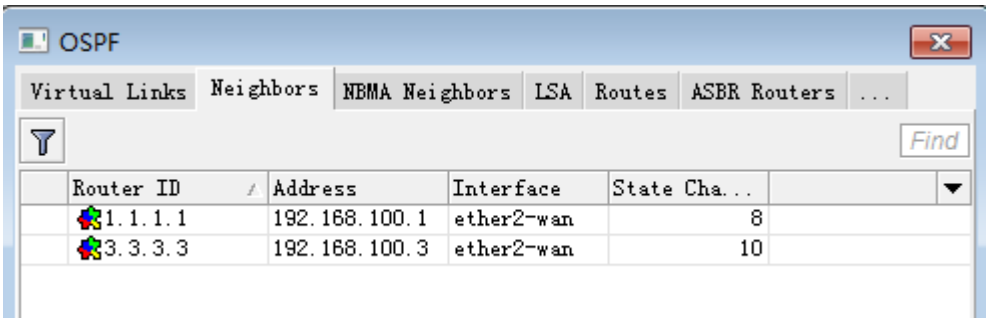


这样 3 台路由器配置完成，如果连接成功后，我们检查下是否建立连接，进入 neighbors 查看是否找到周围的设备

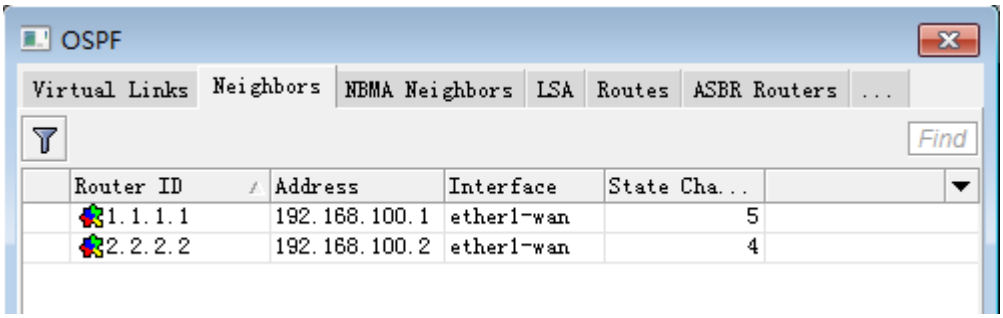
路由器 A，找到了另外两个设备显示如下



路由器 B

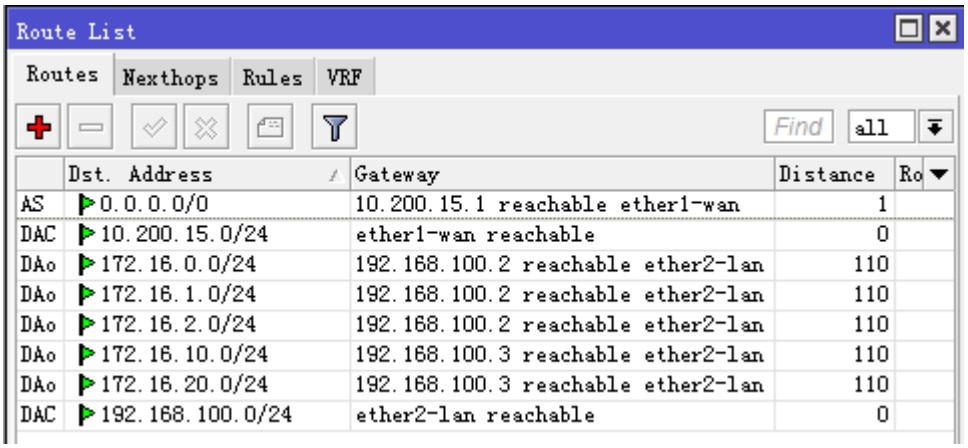


路由器 C

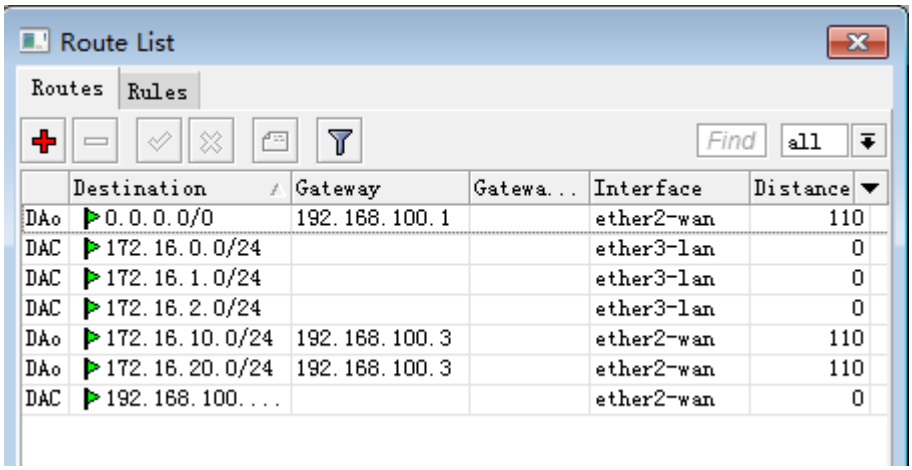


找到相互邻居后，我们可以在 ip route 里看到 OSPF 已经自动生成了路由列表，不需要像静态路由那样每条都手动添加设置

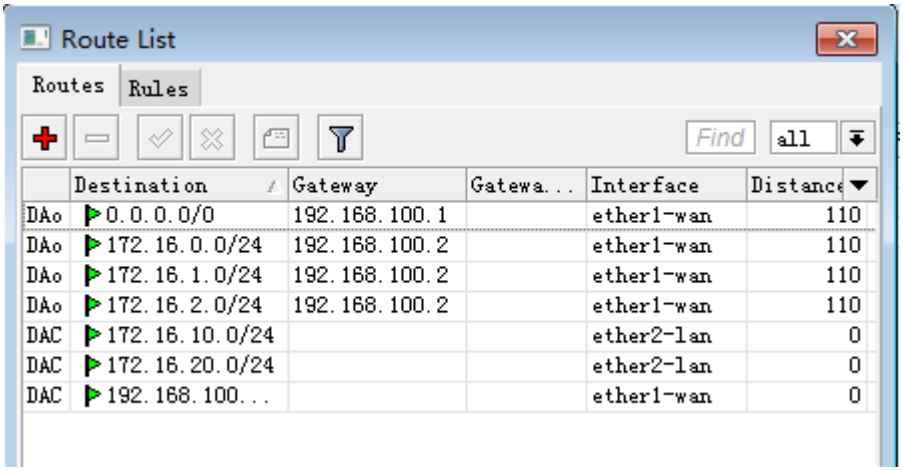
路由器 A 的路由表



路由器 B 路由表



路由器 C 路由表

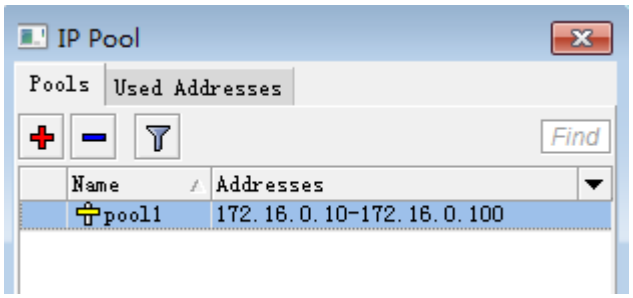


我们检查路由

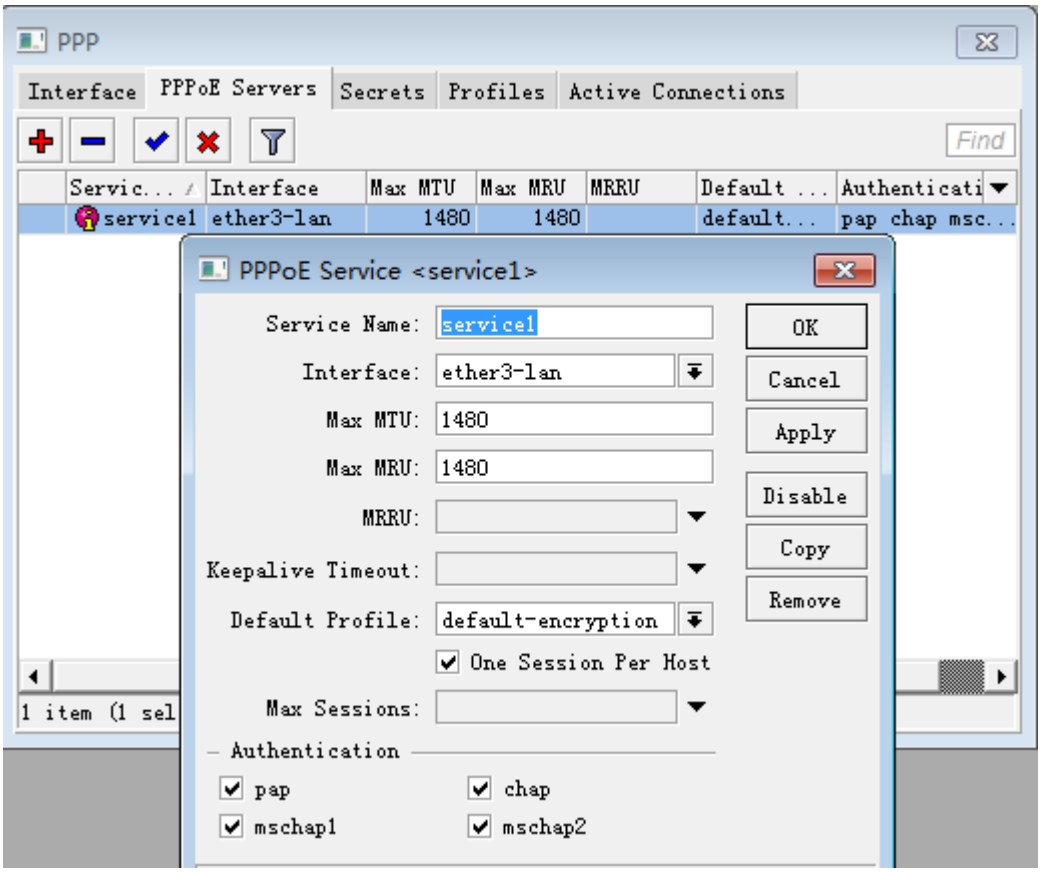
```
[admin@RouterA] > ping 172.16.1.1
HOST                                     SIZE  TTL TIME  STATUS
172.16.1.1                               56     64  1ms
172.16.1.1                               56     64  1ms
172.16.1.1                               56     64  0ms
172.16.1.1                               56     64  1ms
172.16.1.1                               56     64  1ms
172.16.1.1                               56     64  1ms
```

我们可以在路由器 B 建立一个 PPPoE 在内网，用户可以通过 PPPoE 拨号连接上网，

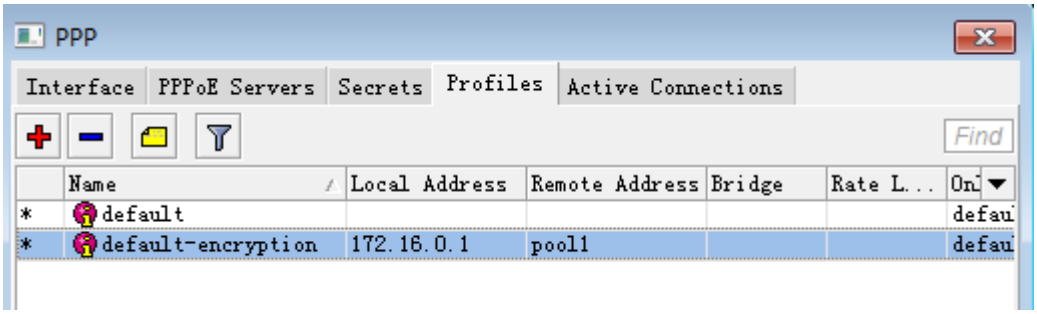
通过 ip pool 分配地址段为

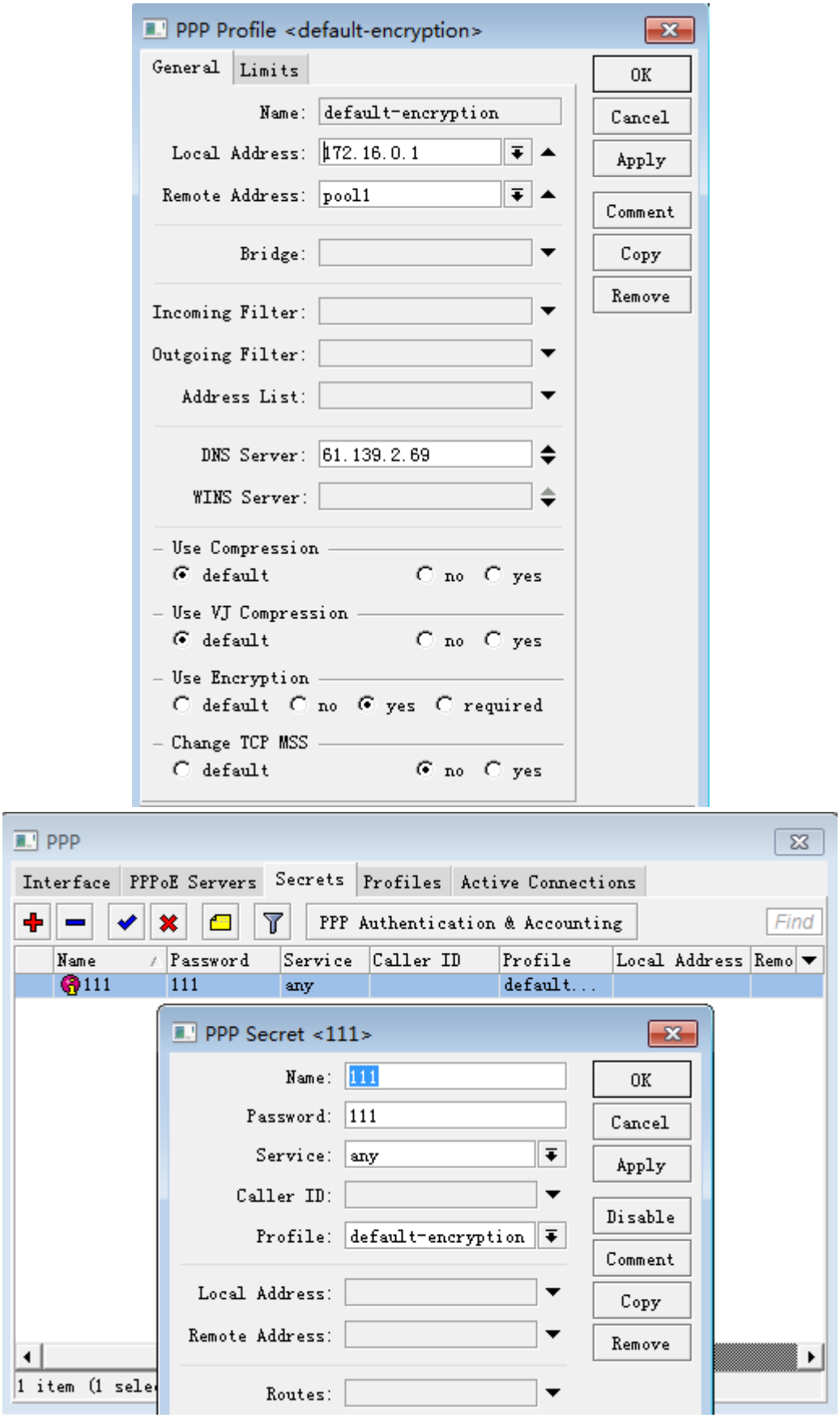


进入 winbox 的 PPP 菜单建立 PPPoE 服务器

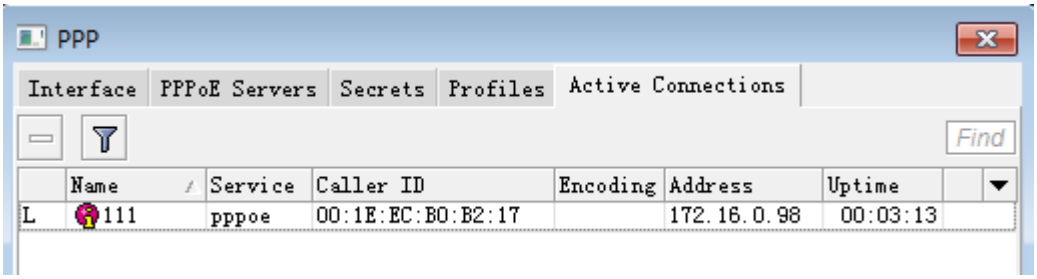


设置 PPPoE 服务器的 Profile 规则，并定义地址池 pool1

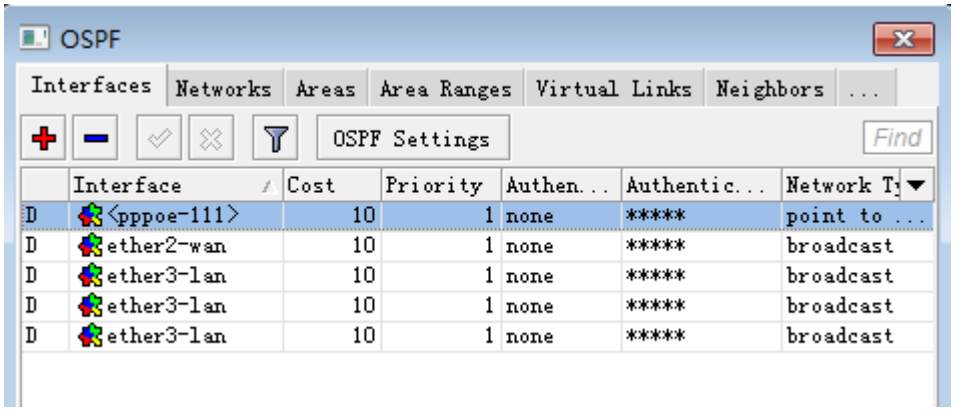




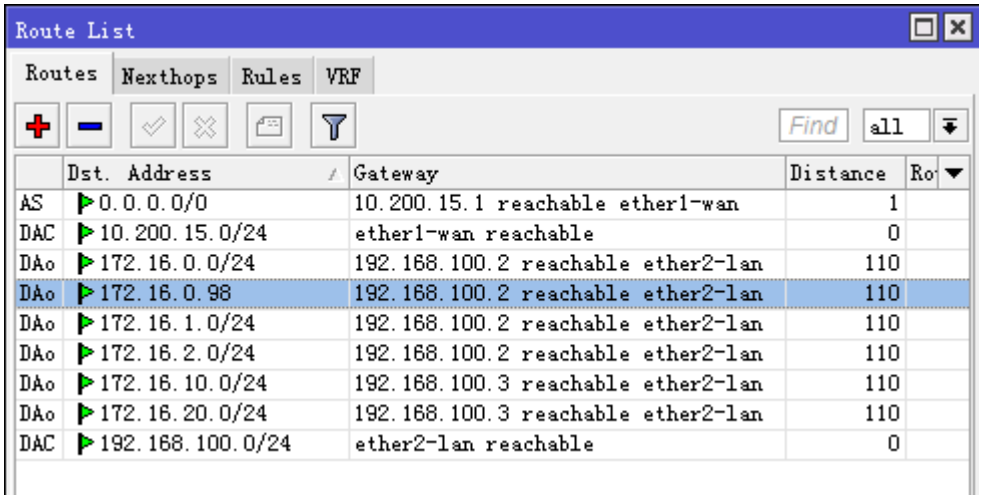
拨号成功后，获得的 IP 地址是 172.16.0.98



我们在路由器 B 的 OSPF 可以看到 pppoe-111 的接口



同样在路由器 A 的路由表里可以找到 172.16.0.98 的



第三十章 BGP

BGP（Border Gateway Protocol）边界网关协议是一种自治系统间的动态路由协议，它的基本功能是在自治系统间自动交换无环路的路由信息，通过交换带有自治系统号序列属性的路径可达信息，来构造自治区域的拓扑图，从而消除路由环路并实施用户配置的路由策略。BGP 是一种 EGP（Exterior Gateway Protocol）协议，而 OSPF、RIP、ISIS 和静态路由等为 IGP（Interior Gateway Protocol）协议，而 BGP 需要通过 IGP 协议来承载。BGP 协议经常用于 ISP 之间的路由交换。最主要功能在于控制路由的传播和选择最好的路由。中国网通 、中国电信、中国铁通和一些大的民营 ISP 和 IDC 运营商都具有 AS 号，全国各大网络运营商多数都是通过 BGP 协议与自身的 AS 号来实现多线互联的。

30.1 BGP 介绍

BGP 使用 TCP 作为其承载协议，端口号 179，提高了协议的可靠性。BGP 不是每次都广播所有的路由信息，而是在初始化全部路由信息后只发送路由增量。这样保证了 BGP 和对端的最小通信量。另外，BGP 通过接收和发送 keep-alive 消息来检测相互之间的 TCP 连接是否正常。

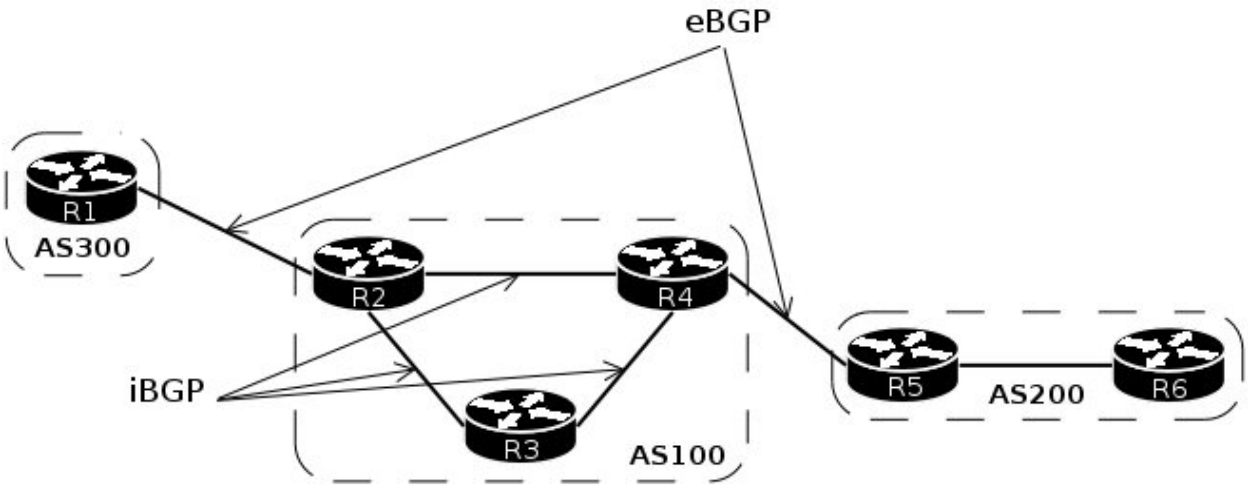
自治系统 AS（Autonomous System）

AS 是指在一个网络管理机构下的拥有相同选路策略的 IP 网络。BGP 网络中的每个 AS 都被分配一个唯一的 AS 号，用于区分不同的 AS。

AS 号码即自治系统号码，是用来标识独立的自治系统的，在同一个自治系统内，使用相同内部路由协议，自治系统间使用外部路由协议（通常是 BGP 协议）。AS 号分为 2 字节 AS 号和 4 字节 AS 号，其中 2 字节 AS 号的范围为 1 至 65535，4 字节 AS 号的范围为 1 至 4294967295。其中 64512 到 65535 为私有 AS 号

BGP 分类

BGP 按照运行方式分为 EBGP（External BGP）和 IBGP（Internal BGP）



EBGP：运行于不同 AS 之间的 BGP 称为 EBGP，IBGP：运行于同一 AS 内部的 BGP 称为 IBGP。

BGP Instance 实例

BGP 配置首先需要建立 BGP 实例，在 RouterOS 中有默认的 BGP 实例（default BGP instance），AS 号需要根据当前网络环境设置，如 AS 号为 65500，我们需要修改默认的 BGP 实例配置，修改操作如下：

```
[admin@MikroTik] > /routing bgp instance set default as=65500 redistribute-static=no
[admin@MikroTik] > /routing bgp instance print Flags: X - disabled
 0    as=65500 router-id=0.0.0.0 redistribute-static=no redistribute-connected=no
      redistribute-rip=no redistribute-ospf=no redistribute-other-bgp=no
      name="default" out-filter=""
[admin@rb11] >
```

BGP 的 Router ID 是一个用于标识 BGP 设备的 32 位的值，通常是 IPv4 地址的形式，在 BGP 会话建立时发送的 Open 报文中携带。对等体之间建立 BGP 会话时，每个 BGP 设备都必须有唯一的 Router ID，否则对等体之间不能建立 BGP 连接。BGP 的 Router ID 在 BGP 网络中必须是唯一的，可以采用手动配置，也可以让 BGP 自己在设备上选取。缺省情况下，BGP 会选择一个路由器的 IP 地址。

BGP Peers

相互交换报文的 BGP 路由器之间互称对等体（Peer），两台 BGP 路由器通过 BGP peers 建立 TCP 连接， 当 TCP 连接建立，路由器之间开始相互交换初始信息，通过 Open 报文交换 BGP 的 route ID、BGP 版本、AS 号和维持时间周期值等，当所有连接和协商完成后 BGP 会话建立，通过 Update 报文交换路由信息。

BGP 的报文

BGP 对等体间通过以下 5 种报文进行交互，其中 Keepalive 报文为周期性发送，其余报文为触发式发送：

- Open 报文：用于建立 BGP 对等体连接。
- Update 报文：用于在对等体之间交换路由信息。
- Notification 报文：用于中断 BGP 连接。
- Keepalive 报文：用于保持 BGP 连接。
- Route-refresh 报文：用于在改变路由策略后请求对等体重新发送路由信息。只有支持路由刷新（Route-refresh）能力的 BGP 设备会发送和回应此报文。

与其他 BGP 路由器建立 TCP 连接，通过如下配置：

```
[eugene@SM_BGP] > /routing bgp peer add remote-address=10.20.1.210 remote-as=65534
[eugene@SM_BGP] > /routing bgp peer print
Flags: X - disabled
 0    instance=default remote-address=10.20.1.210 remote-as=65534 tcp-md5-key=""
      multihop=no route-reflect=no hold-time=3m ttl=3 in-filter=""
      out-filter=""

[eugene@SM_BGP] >
```

通过 print status 命令核实 BGP 连接状态：

```
[eugene@SM_BGP] > /routing bgp peer print status
Flags: X - disabled
 0    instance=default remote-address=10.20.1.210 remote-as=65534 tcp-md5-key=""
      multihop=no route-reflect=no hold-time=3m ttl=3 in-filter=""
      out-filter="" remote-id=10.20.1.210 uptime=1d1h43m16s
      prefix-count=180000 remote-hold-time=3m used-hold-time=3m
      used-keepalive-time=1m refresh-capability=yes state=established
[eugene@SM_BGP] >
```

BGP 在两端对等体连接建立后状态为 **state=established**，

BGP 状态，BGP 对等体的交互过程中存在 6 种状态机：空闲状态（Idle）、连接状态（Connect）、活跃（Active）、Open 报文已发送（OpenSent）、Open 报文已确认（OpenConfirm）和连接已建立（Established）。在 BGP 对等体建立的过程中，通常可见的 3 个状态是：Idle、Active 和 Established。

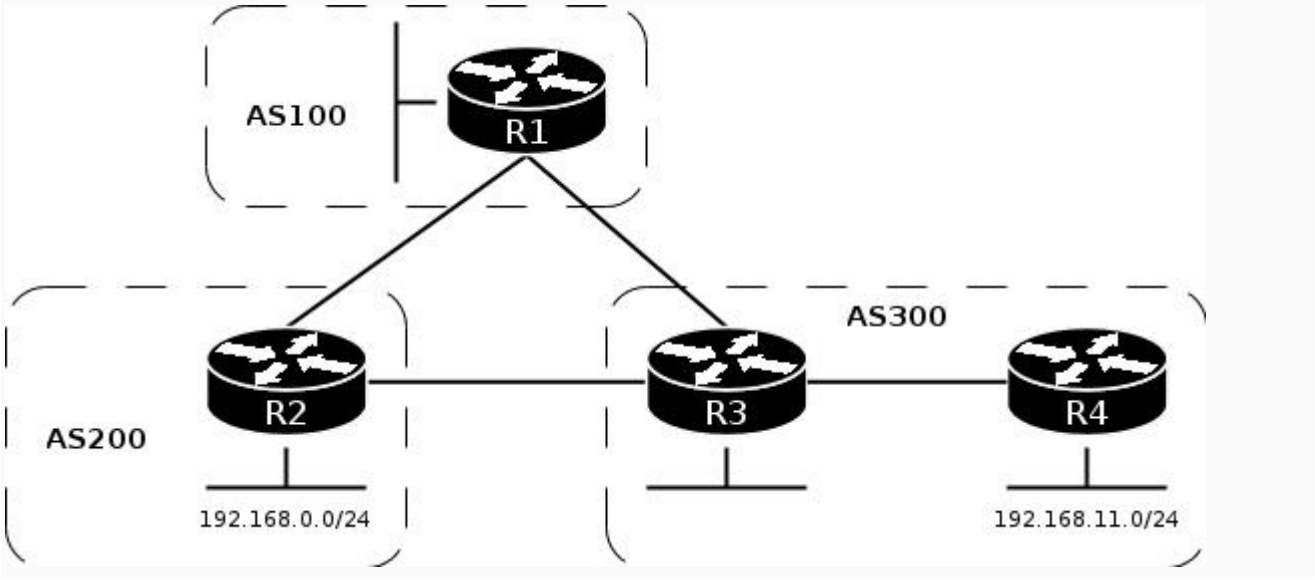
- **Idle**—BGP 进程被启动或被重置，这个状态是等待开始，比如等于指定一个 BGP peer，当收到 TCP 连接请求后，便初始化另外一个事件，当路由器或 peer 重置，都会回到 idle 状态。
- **Connect**—检测到有 peer 要尝试建立 TCP 连接。
- **Active**—尝试和对方 peer 建立 TCP 连接，如有故障，则回到 idle 状态
- **OpenSent**— TCP 连接已经建立，BGP 发送了一个 OPEN 消息给对方 peer，然后切换到 OpenSent 状态，如果失败，则切换到 Active 状态。
- **OpenReceive**— 收到对方 peer 的 OPEN 消息，并等待 keepalive 消息，如果收到 keepalive，则转到 Established 状态，如果收到 notification，则回到 idle 状态，比如错误或配置改变，都会发送 notification 而回到 idle 状态。
- **Established**— 从对端 peer 收到了 keepalive，并开始交换数据，收到 keepalive 后，hold timer 都会被重置，如果收到 notification，就回到 idle 状态。

路由重分布

BGP 预设不会重分布路由，需要通过将相应路由重分布到 BGP 中，各种路由类型设置参数为：redistribute-connected、redistribute-static、redistribute-rip、redistribute-ospf 和 redistribute-other-bgp ，这些参数在 BGP instance 中设置路由过滤

路由过滤

未被过滤掉路由重分布可以导致未知的结果，下面考虑以下的事例，有台路由器 R1、R2 和 R3 建立了 EBGP 关系，R3 和 R4 是 IBGP 网络，当存在一些不合理的配置时，例如 R3 有一条静态路由指到 R2 的 192.168.0.0/24 网络，并重分布静态路由到 BGP 网络，这样 R1 会学到 192.168.0.0/24 来至 R3，这样导致错误的路径。因此我们只希望 R3 发布 R4 到 BGP 网络，R3 通过路由过滤仅重分布 192.168.11.0/24 的网络。



R3 路由器用静态路由重分布到 BGP 网络：

```
/routing bgp instance set default redistribute-static=yes
```

进入 routing filter 添加一个链表 to_R1 过滤向外发布路由，除 prefix=192.168.11.0/24 都拒绝

```
/routing filter add chain=to_R1 prefix=192.168.11.0/24 invert-match=yes action=discard
/routing bgp peer set R1 out-filter=to_R1
```

注意：invert-match　参数是反转匹配，即除了 192.168.11.0/24 其他任何都拒绝

路由过滤通过/routing filter 进行处理。一个路由过滤通过 chain 链表组成一个或多个过滤规则。规则处理是从上往下执行，每一条规则都包含自己的条件按照顺序执行匹配 prefixes 前缀清单。启用路由过滤可以在 BGP peer 的 in-filter 或 out-filter 和 BGP instance 的 out-filter 指定相应的路由过滤链表名称

路由过滤事例

```
[eugene@SM_BGP] routing filter> print chain=Latnet-in
Flags: X - disabled
 0    chain=Latnet-in prefix=10.0.0.0/8 prefix-length=8-32 invert-match=no action=discard

 1    chain=Latnet-in prefix=192.168.0.0/16 invert-match=no action=discard

 2    chain=Latnet-in prefix=169.254.0.0/16 invert-match=no action=discard

 3    chain=Latnet-in prefix=4.23.113.0/24 invert-match=no action=passthrough
      set-bgp-communities=64550:14

 4    chain=Latnet-in prefix=4.36.116.0/23 invert-match=no action=passthrough set-routing-mark="LAN"
set-route-comment="Remote offices"

 5    chain=Latnet-in prefix=8.8.0.0/16 prefix-length=16-32 bgp-communities=2588:800
invert-match=no action=discard
```

```
[eugene@SM_BGP] routing filter>
```

- 规则 0 匹配 prefix 为 10.0.0.0/8，且明确 prefixes 包含 10.0.1.0/24、10.1.23.0/28 等，子网范围在 8-32 的网络，rule #0 matches prefix 10.0.0.0/8 and more specific prefixes like 10.0.1.0/24, 10.1.23.0/28, etc. and discards them (these prefixes are silently dropped from inbound update messages and do not appear in memory)
- 规则 3 设置 BGP 团体属性 prefix 范围 4.23.113.0/24
- 规则 4 有两个执行，rule #4 has two actions. It simultaneously sets routing mark and comment for route to 4.36.116.0/23
- rule #5 discards prefix 8.8.0.0/16 and more specific ones, if they have COMMUNITY attribute of 2588:800

使用以上的过滤，并添加到 Latnet 对等体的 in-filter 参数中：

```
[eugene@SM_BGP] routing bgp peer> set Latnet in-filter=Latnet-in
[eugene@SM_BGP] routing filter> print
Flags: X - disabled
 0    name="C7200" instance=latnet remote-address=10.0.11.202 remote-as=64527 tcp-md5-key=""
nexthop-choice=default multihop=no route-reflect=no hold-time=3m ttl=1 in-filter=""
out-filter=to_C7200

 1    name="Latnet" instance=latnet remote-address=10.0.11.55 remote-as=2588 tcp-md5-key=""
nexthop-choice=default multihop=yes route-reflect=no hold-time=3m ttl=5 in-filter="Latnet-in"
out-filter=to_Latnet

 8    name="gated" instance=latnet remote-address=10.0.11.20 remote-as=64550 tcp-md5-key=""
nexthop-choice=default multihop=no route-reflect=no hold-time=3m ttl=1 in-filter="" out-filter=""

[eugene@SM_BGP] routing bgp peer>
```

BGP Networks

BGP 允许无条件宣告一些网络地址段，通将这些地址添加到/routing bgp networks 列表中。下面通过 network 发布 192.168.0.0/24 到 BGP 网络

```
[eugene@SM_BGP] > /routing bgp network add network=192.168.0.0/24
[eugene@SM_BGP] > /routing bgp network print
Flags: X - disabled
 #    NETWORK
 0    192.168.0.0/24
[eugene@SM_BGP] >
```

RouterOS 的 BGP 静态路由

你可以总是使用静态路由建立一条子网由，RouterOS 在/ip route 目录下提供了相应的 BGP 关系属性设置，静态路由能为 BGP 配置提供更多的属性，例如你添加 10.8.0.0/16 网段，同时可以设置 BGP 本地优先级属性值：

```
/ip route add dst-address=10.8.0.0/16 gateway=10.0.11.1 bgp-local-pref=110
[admin@MikroTik] > /ip ro print
```

```
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf
 #      DST-ADDRESS         PREF-SRC         G GATEWAY          DISTANCE INTERFACE
 0 A S 0.0.0.0/0                          r 10.0.11.1        1         ether1
 1 ADC 10.0.11.0/24       10.0.11.51                           0         ether1
 2 A S 10.8.0.0/16                        r 10.0.11.1        1         ether1
 3 ADC 10.12.0.0/24       10.12.0.2                            0         bonding1
[admin@MikroTik] >
```

BGP Advertisements

RouterOS 提供查看本地路由器发布给对端 BGP peer 的路由条目，通过**/routing bgp advertisements print <peer's address>**命名查看

```
[eugene@SM_BGP] routing bgp advertisements> print 10.0.11.20
 # DST-ADDRESS    NEXTHOP          AS-PATH                      ORIGIN      LOCAL-PREF MED
 0 3.0.0.0/8           159.148.254.250 2588,6747,1299,701,703,80        igp         100
 1 4.0.0.0/8           10.0.11.155      2588,6747{174,1273,1299,2914... igp         100
 2 6.0.0.0/8           10.0.11.155      2588,6747,1299,701,668           igp         100
 3 8.0.0.0/8           159.148.254.250 2588,6747,1299,3356              igp         100
 4 8.0.0.0/9           159.148.254.250 2588,6747,1299,3356              igp         100
 5 8.2.64.0/23         159.148.254.250 2588,6747,1299,3356,16803        igp         100
 6 8.2.144.0/22        159.148.254.250 2588,6747,1299,3356,36394        igp         100
 7 8.3.12.0/24         159.148.254.250 2588,6747,1299,3356,14711        igp         100
 8 8.3.13.0/24         159.148.254.250 2588,6747,1299,3356,26769        igp         100
 9 8.3.15.0/24         159.148.254.250 2588,6747,1299,3356,14711        igp         100
10 8.3.17.0/24         159.148.254.250 2588,6747,1299,25973             igp         100
11 8.3.19.0/24         159.148.254.250 2588,6747,1273,22822,26769       igp         100
12 8.3.37.0/24         159.148.254.250 2588,6747,1299,3356,3356,21640   igp         100
13 8.3.38.0/23         159.148.254.250 2588,6747,1299,3549,16420        igp         100
14 8.3.46.0/24         159.148.254.250 2588,6747,1299,3356,3356,21640   igp         100
15 8.3.208.0/24        159.148.254.250 2588,6747,1299,3549,36431        igp         100
16 8.3.209.0/24        159.148.254.250 2588,6747,1273,22822,26769       igp         100
17 8.3.210.0/24        159.148.254.250 2588,6747,1299,27524             igp         100
18 8.3.216.0/24        159.148.254.250 2588,6747,1299,3356,15170        igp         100
19 8.4.86.0/24         159.148.254.250 2588,6747,1299,3356,14627        igp         100
20 8.4.96.0/20         159.148.254.250 2588,6747,1299,3356,15162        igp         100
21 8.4.113.0/24        159.148.254.250 2588,6747,1299,3356,15162        igp         100
22 8.4.224.0/24        159.148.254.250 2588,6747,1299,3356,13546        igp         100
23 8.5.192.0/22        159.148.254.250 2588,6747,1299,209,13989         igp         100
24 8.6.48.0/21         159.148.254.250 2588,6747,1299,3356,36492        igp         100
25 8.6.89.0/24         159.148.254.250 2588,6747,1299,3356,11734        igp         100
26 8.6.90.0/24         159.148.254.250 2588,6747,1299,3356,16541        igp         100
27 8.6.220.0/22        159.148.254.250 2588,6747,1299,3356,13680        igp         100

[eugene@SM_BGP] routing bgp advertisements>
```

BGP 负载均衡

RouterOS 的 BGP 负载均衡实现不能直接实现，BGP 默认是无法实现一条单独路由的多个下一跳，但可通过下面两种方法实现：

一种方法是在 routing filter 里设置多跳网关，如下：

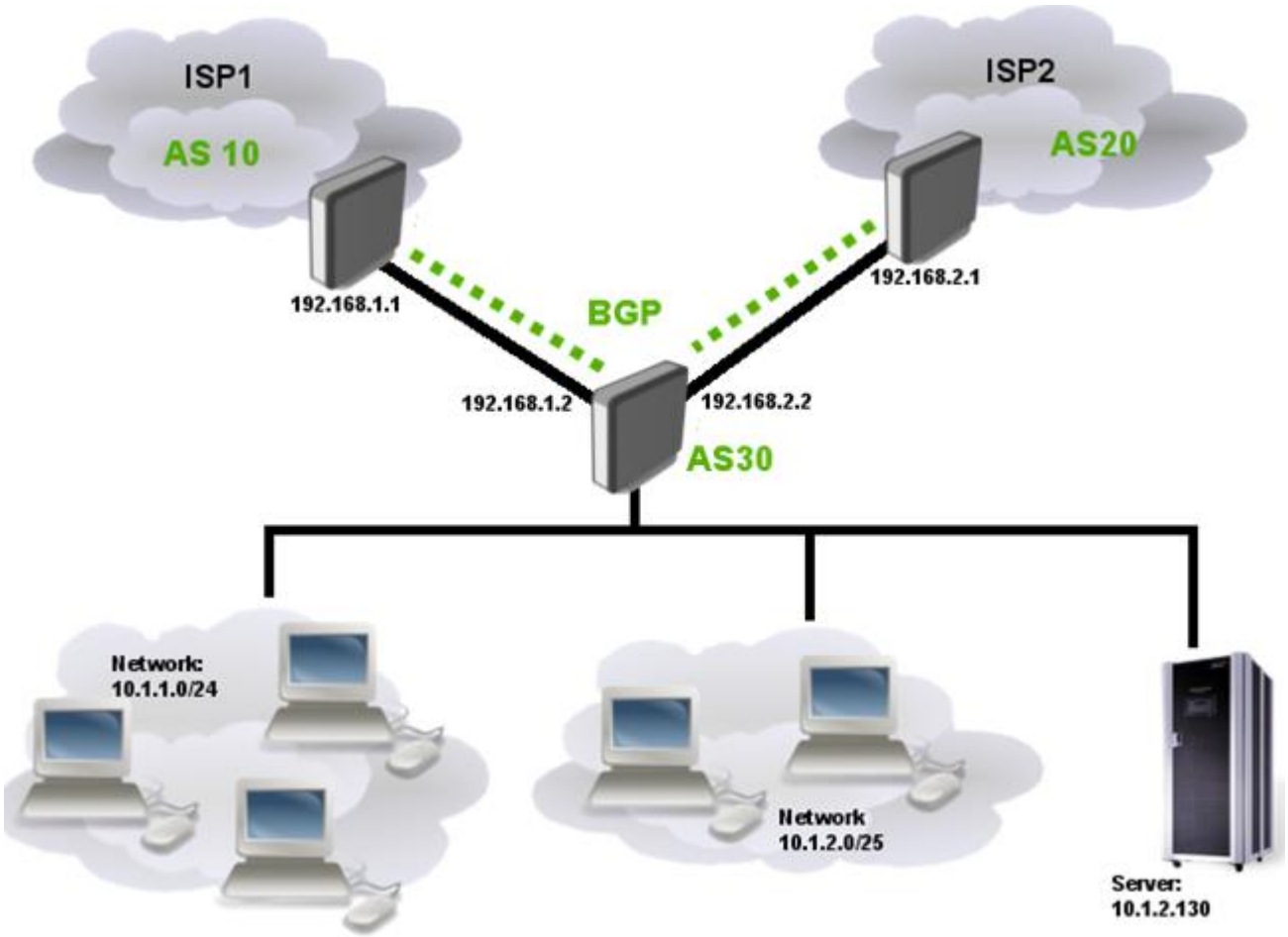
```
routing filter add chain=bgp-in set-in-nexthop=10.0.1.1,10.0.2.1
```

另一种方法，则不能直接使用 BGP next-hop，而是用静态路由或 OSPF 路由的 multiple next-hops，下面是静态路由的配置

```
ip route add dst-address=x.x.x.x/y gateway=10.0.1.1,10.0.2.1
```

30.2 BGP 事例 1

下面是一个多线 BGP 的网络结构，该网络中一台路由器连接 ISP1 和 ISP2，可以实现负载均衡，也可以做链路主备配置。



路由器的本地连接了两个网段 10.1.1.0/24 和 10.1.2.0/24 ，并且通过 AS 30（用私网 IP 建立 BGP 互连）与两个 ISP 互连，在网络中有 10.1.2.130 为服务器，BGP 过程是首先建立连接，将两个本地网段宣告给对方，然后将 ISP1 设为主连接，ISP2 连接为备用

注意:这个事例将不介绍路由器与本地网络和服务器的网络连接配置，主要以 BGP 事例为主。

配置 BGP Peer

这里我们的路由器，已经配置好了与两个 ISP 的网络参数，就直接配置路由器两个 ISP 之间的 BGP peer

```
#set our AS number
/routing bgp instance
set default as=30

#add BGP peers
/routing bgp peer
add name=toISP1 remote-address=192.168.1.1 remote-as=10
add name=toISP2 remote-address=192.168.2.1 remote-as=20
```

如果与对端建立连接，peer 下会显示 E (established)的标记，路由器将接收到两个 ISP 发布的路由

```
[admin@RB1100] /routing bgp peer> print
Flags: X - disabled, E - established
 #    INSTANCE         REMOTE-ADDRESS                    REMOTE-AS
 0 E default          192.168.1.1                       10
 1 E default          192.168.1.2                       20
```

network 宣告和路由过滤

当 peer 连接后，就可以开始宣告我们的网络，并配置路由过滤，路由过滤是对一些不必要的路由进行限制

第一步宣告我们的网络

```
/routing bgp network
add network=10.1.1.0/24 synchronize=no
add network=10.1.2.0/24 synchronize=no
```

第二步指定哪些路由需要过滤

```
/routing bgp peer
set isp1 in-filter=isp1-in out-filter=isp1-out
set isp2 in-filter=isp2-in out-filter=isp2-out
```

in-filter（incoming-filter）负责接收到的路由过滤，out-filter（outgoing-filter）负责发出路由过滤，都需要通过 prefixes 匹配

路径选择

在 filter 的链表中可以指定哪些路由被接受，那么路由被拒绝。BGP 可以通过 as-path 路径长度连接选择优路径(路径越短的 AS-Path 越优先)，这里希望 ISP2 为备份链路，因此将使用到 BGP AS prepend 属性，增加 AS-path 长度。由于我们的网络上 EBGP，对于 MED 而言但不同的 AS 不做比较，修改 BGP AS prepend 也可以称为路径欺骗，这样 BGP 的路径欺骗是 MED 的替代解决方法。RouterOS 通过 set-bgp-prepend，也可以选择

set-bgp-prepend-path 属性，BGP 选路属性还可以使用 bgp-local-pref 属性，bgp-local-pref 默认为 100，数值越高优先级越大。

向 ISP1 发布路由进行过滤，进入 routing filter 下，发布本地的 10.1.1.0/24 和 10.1.2.0/24 网络，并拒绝发布其他路由

```
/routing filter
#accept our networks
add chain=isp1-out prefix=10.1.1.0/24 action=accept
add chain=isp1-out prefix=10.1.2.0/24 action=accept
#discard the rest
add chain=isp1-out action=discard
```

同样想 ISP2 发布路由进行过滤，但这里需要设置 set-bgp-prepend 属性，增加 as-path 长度：

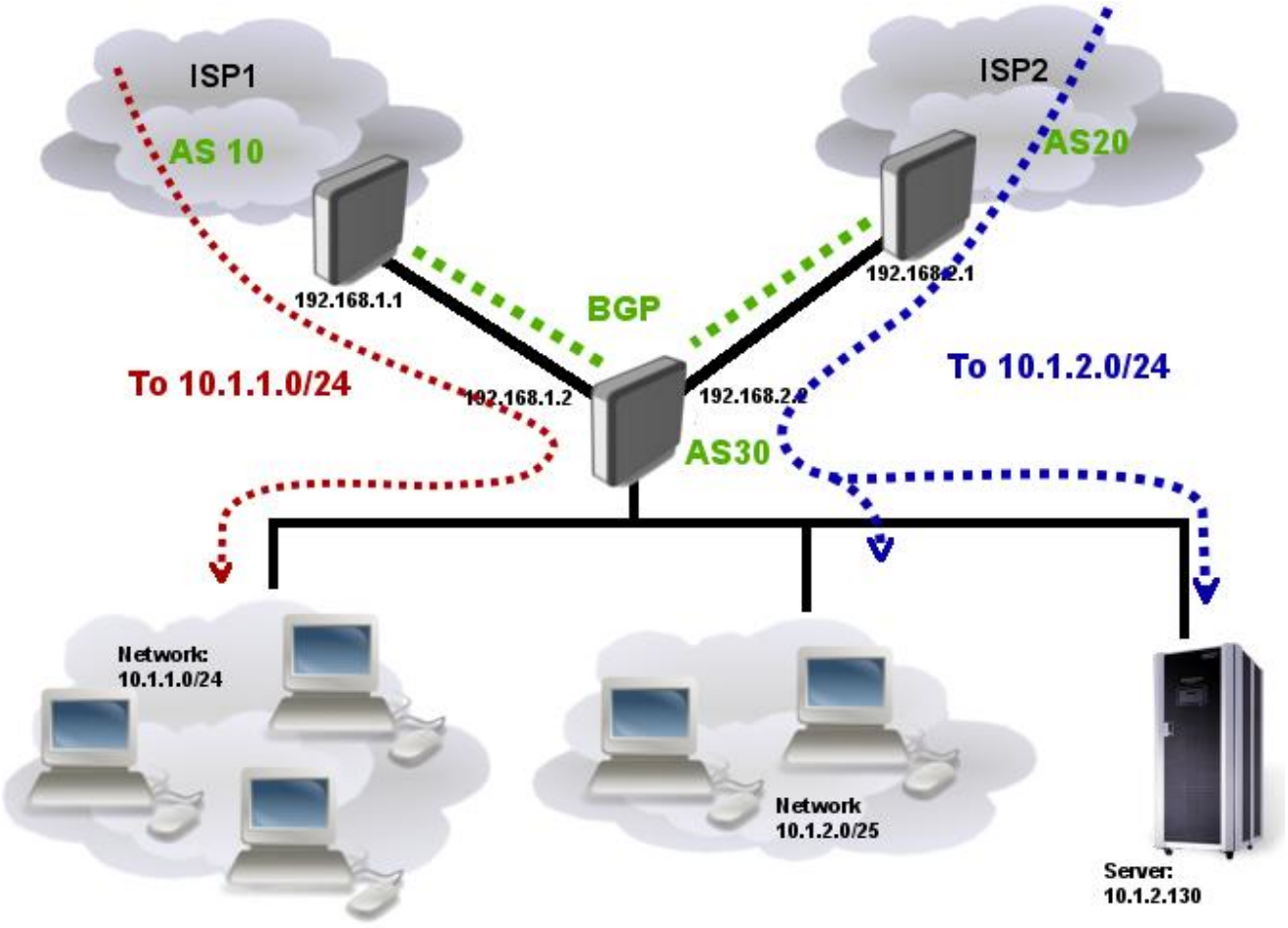
```
/routing filter
#accept our networks and prepend AS path three times
add chain=isp2-out prefix=10.1.1.0/24 action=accept set-bgp-prepend=3
add chain=isp2-out prefix=10.1.2.0/24 action=accept set-bgp-prepend=3
#discard the rest
add chain=isp2-out action=discard
```

由于我们到两个 ISP 是建立默认路由，所以不需要接收，两个 ISP 向我们发布的任何路由，直接配置为拒绝接收，设置两条默认的静态路由，但到 ISP2 的默认路由 distance 设置为 30 作为备份链路，并开启网关 ping 检测

```
/routing filter
add chain=isp1-in action=discard
add chain=isp2-in action=discard
/ip route
add gateway=192.168.1.1 check-gateway=ping
add gateway=192.168.2.1 distance=30 check-gateway=ping
```

负载均衡设置

根据之前的 BGP 网络结构，需要实现到两个 ISP 的路由负载均衡，实现负载均衡有多种方式，下面我们通过设置路由器本地两个段分别走不同的 ISP 路由实现。



将 10.10.1.0/24 走 ISP1，10.10.2.0/24 走 ISP2，当然两个 ISP 直接同时互为备份路由，具体实现配置如下：

向 ISP1 发布路由过滤规则：

```
/routing filter
#accept our networks and prepend second network
add chain=isp1-out prefix=10.1.1.0/24 action=accept
add chain=isp1-out prefix=10.1.2.0/24 action=accept   set-bgp-prepend=3
#discard the rest
add chain=isp1-out action=discard
```

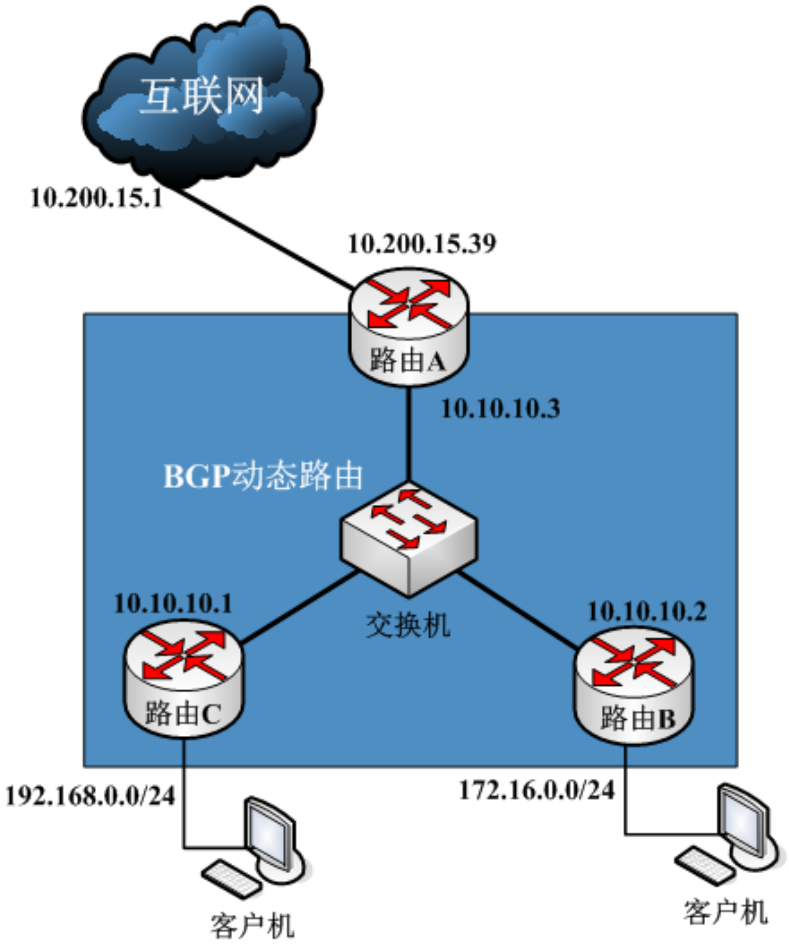
向 ISP2 发布路由过滤规则：

```
/routing filter
#accept our networks and prepend first network
add chain=isp2-out prefix=10.1.1.0/24 action=accept set-bgp-prepend=3
add chain=isp2-out prefix=10.1.2.0/24 action=accept
#discard the rest
add chain=isp2-out action=discard
```

以上配置是 BGP 负载均衡的一种方法。

30.3 BGP 事例 2

由于 BGP 常用于外部网络连接，这里我们仅仅是简单介绍下 BGP 互联的简单操作，我们通过一个事例来实现 3 台 RouterOS 组建一个 BGP 的网络，并通过一个外网接口上网，我们的网络结构如下图：

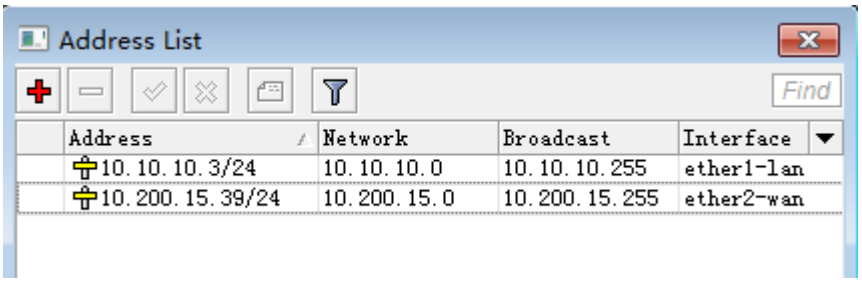


路由器 A 是连接互联网的出口路由器，路由器 B 和路由器 C 分别通过 10.10.10.0/24 与路由器 A 互联，并分别连接了 192.168.0.0/24 和 172.16.0.0/24 的网段

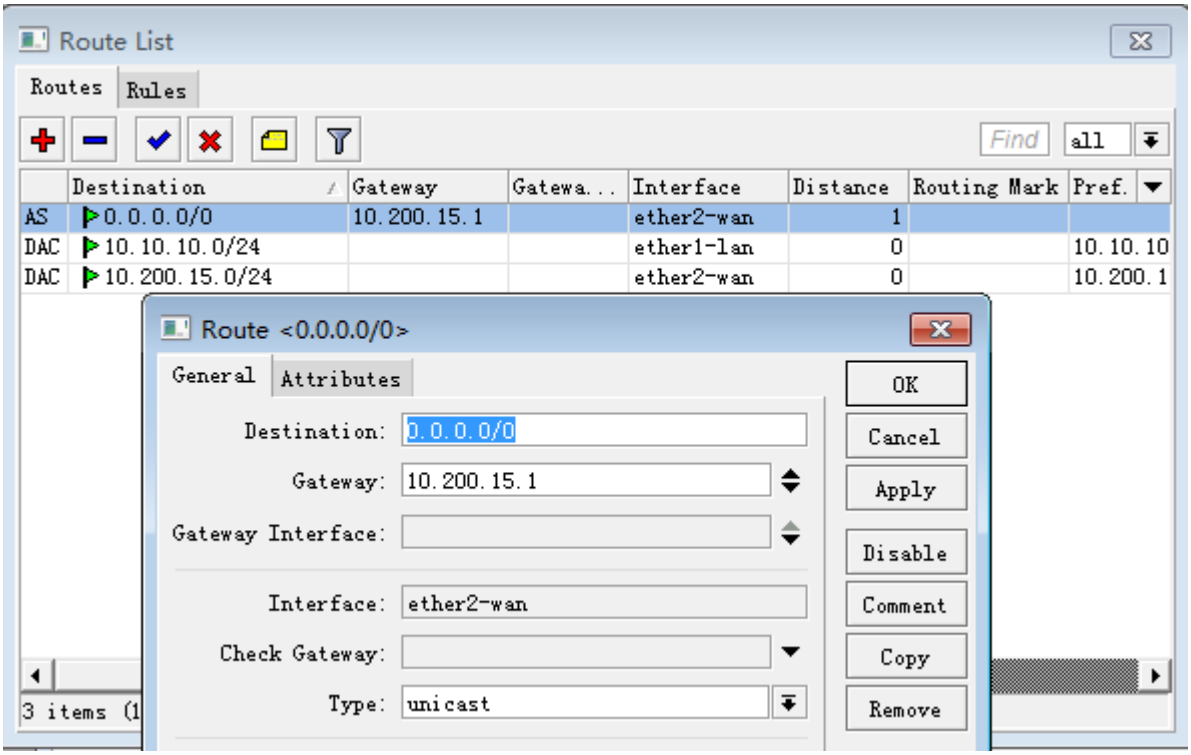
在这里我们使用默认的 AS 编号 65530，BGP 配置，我们首先设置每台路由器的 IP 地址和相关的参数

路由器 A 配置：

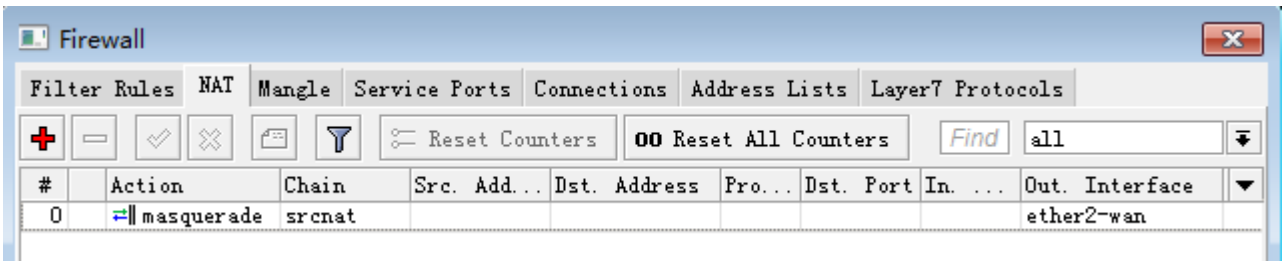
添加 IP 地址到 ip address 中



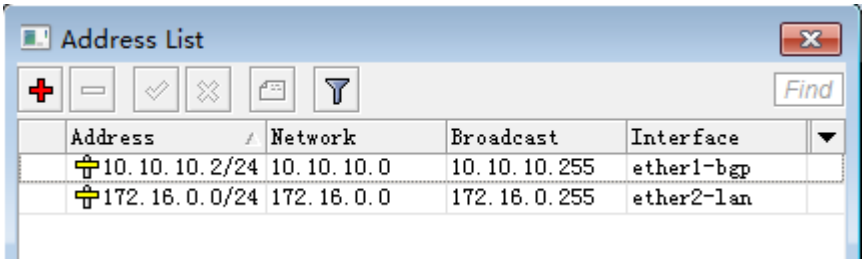
路由器 A 需要连接互联网，所以这里我们设置一个默认静态路由：



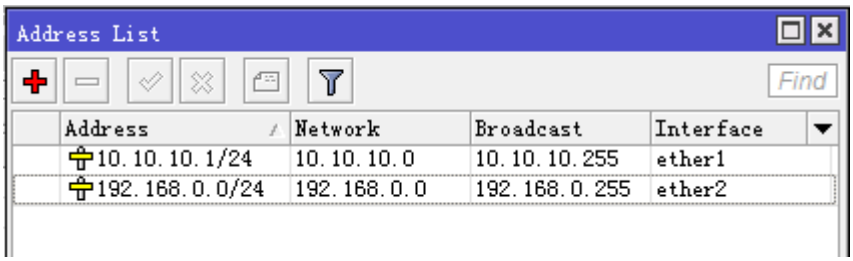
路由器 A 是网关出口，所以我们需要做一个 nat 的转换，进入 ip firewall nat 添加转换规则



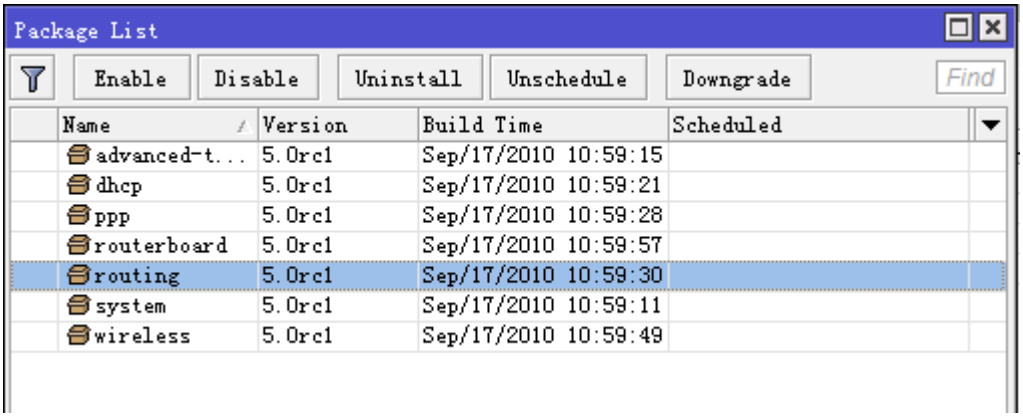
配置路由器 B 的 IP 地址



路由器 C 的 IP 地址



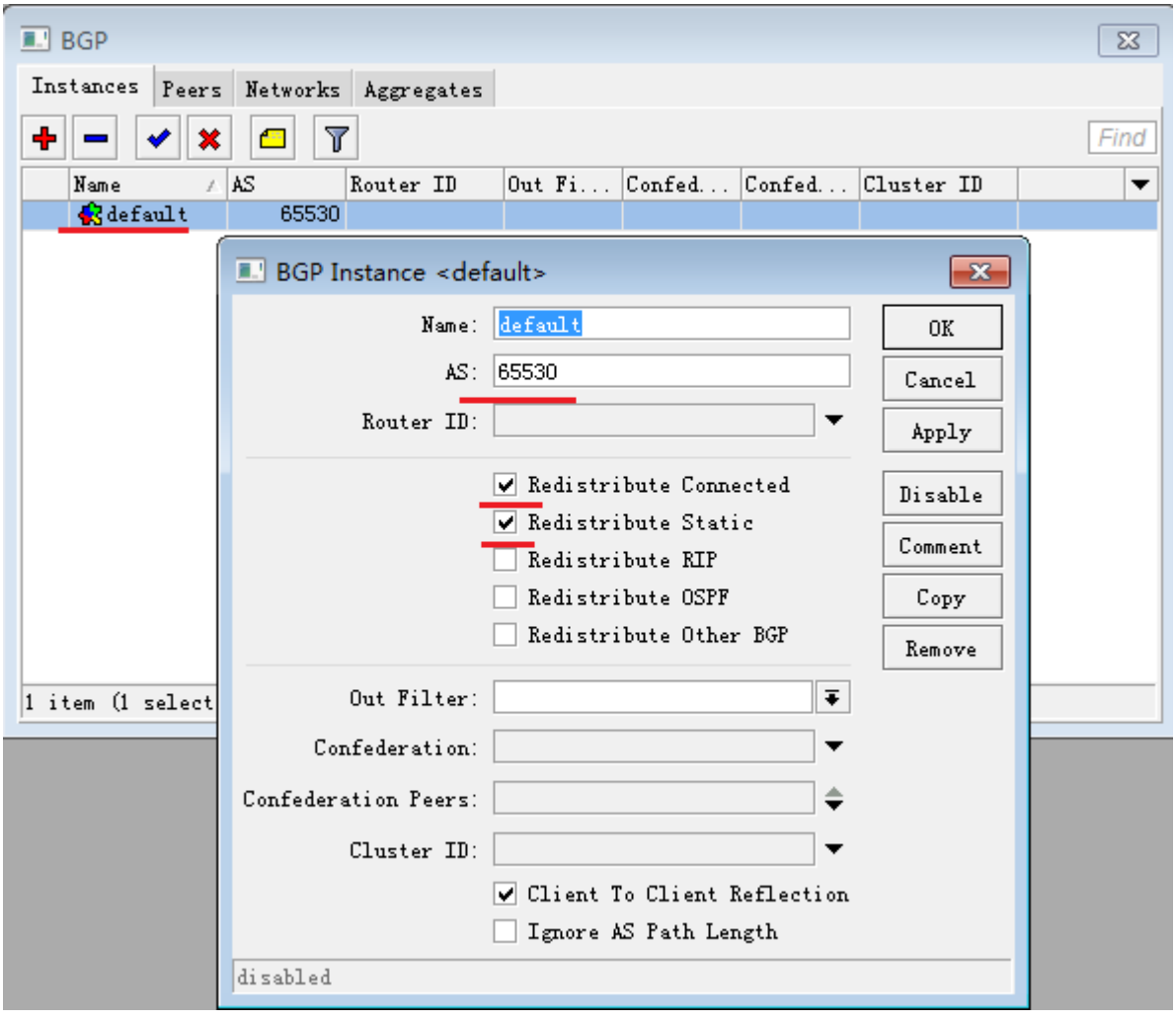
接下来就是配置 BGP 连接，我们首先确定每个路由器都安装了 routing 的功能包，如果没有安装就无法使用动态路由协议，确定是否安装我们可以查看 system package 里的参数：



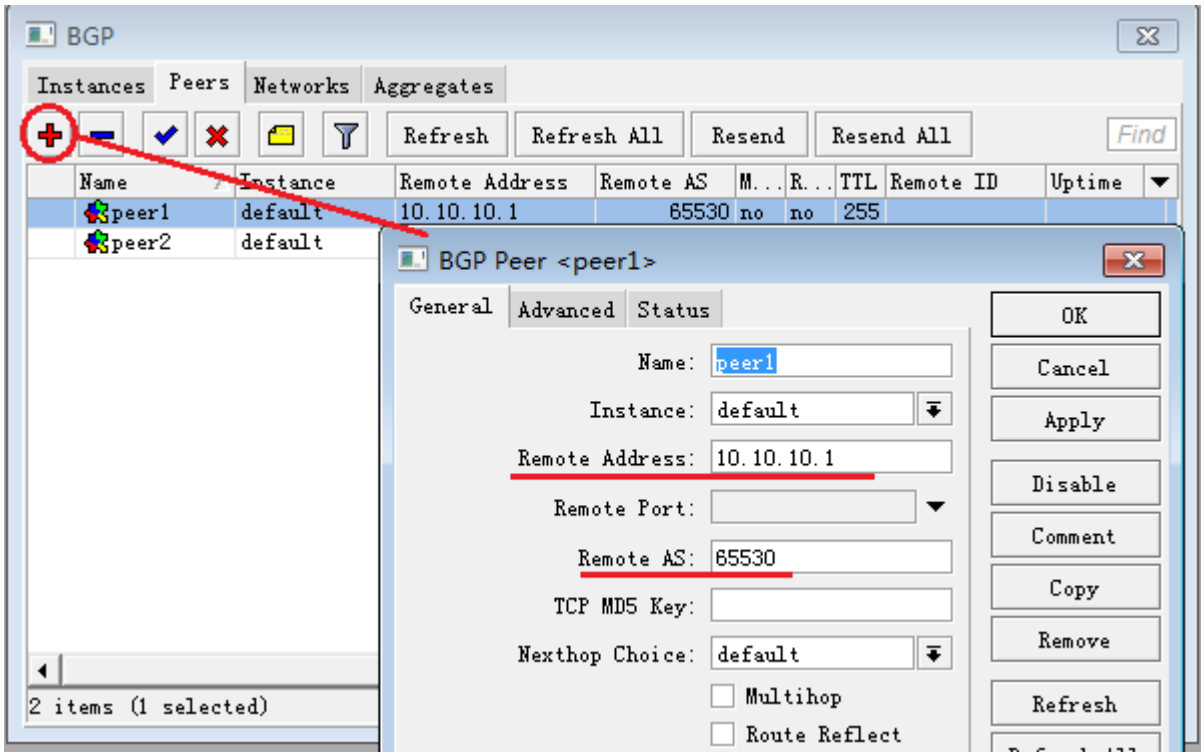
确认安装后，我们可以在 winbox 的左边菜单找到 routing 选项。

配置前我们确定 BGP 的 AS 自治域编号是 65530，即 3 台路由器在一个自治域内。注意这里我们使用了 3 种版本的 RouterOS，分别是 3.30，4.10，5.0rc1，接口有所不同，但操作是一样的。

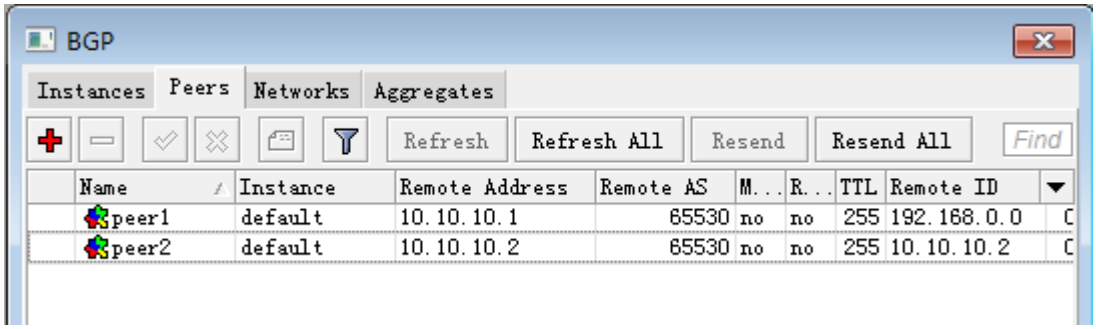
首先配置路由器 A 的 BGP 参数，打开 BGP 后我们选择 Instances，打开 default 选项，我们使用默认的 AS：65530，Router-ID 这里可以不用设置，路由器会自动使用本地连接地址作为 ID，因为路由器 A 是网关出口，所以我们需要分发静态路由给下面的 BGP 路由器在，redistribute-connected 和 redistribute-static 打上勾



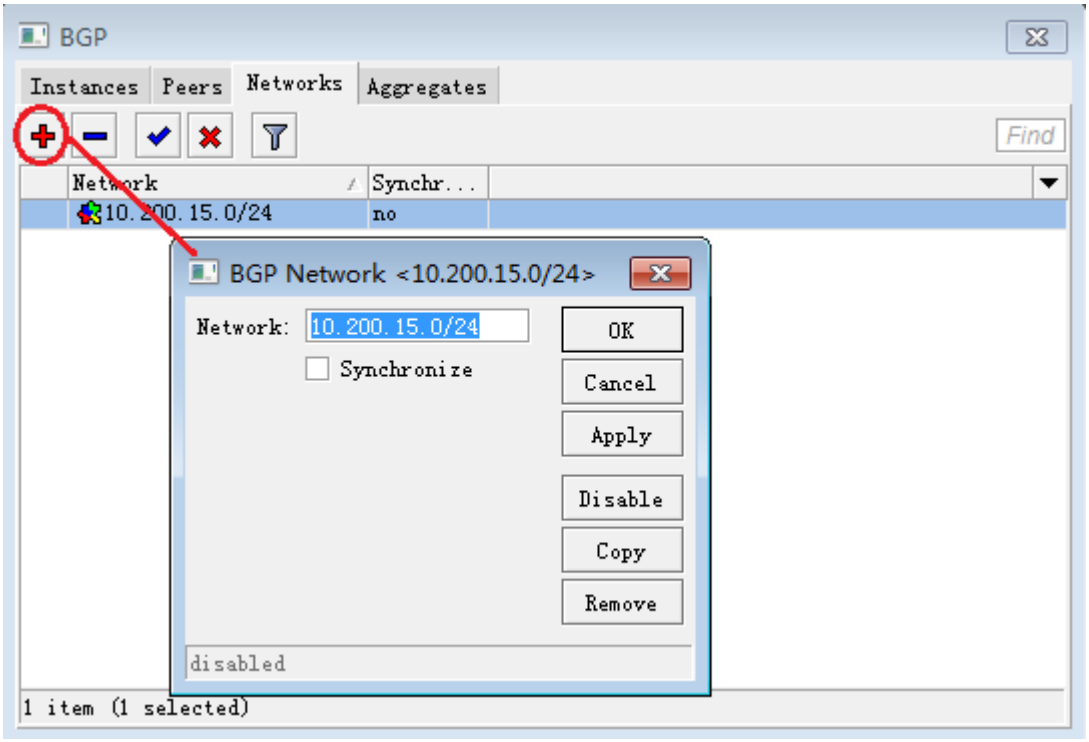
进入 peers 添加对端设备，路由器 A 的 IP 地址是 10.10.10.3，添加路由器 C 和 B 的 IP 地址，分别是 10.10.10.1，并设置 remote-as 参数 65530



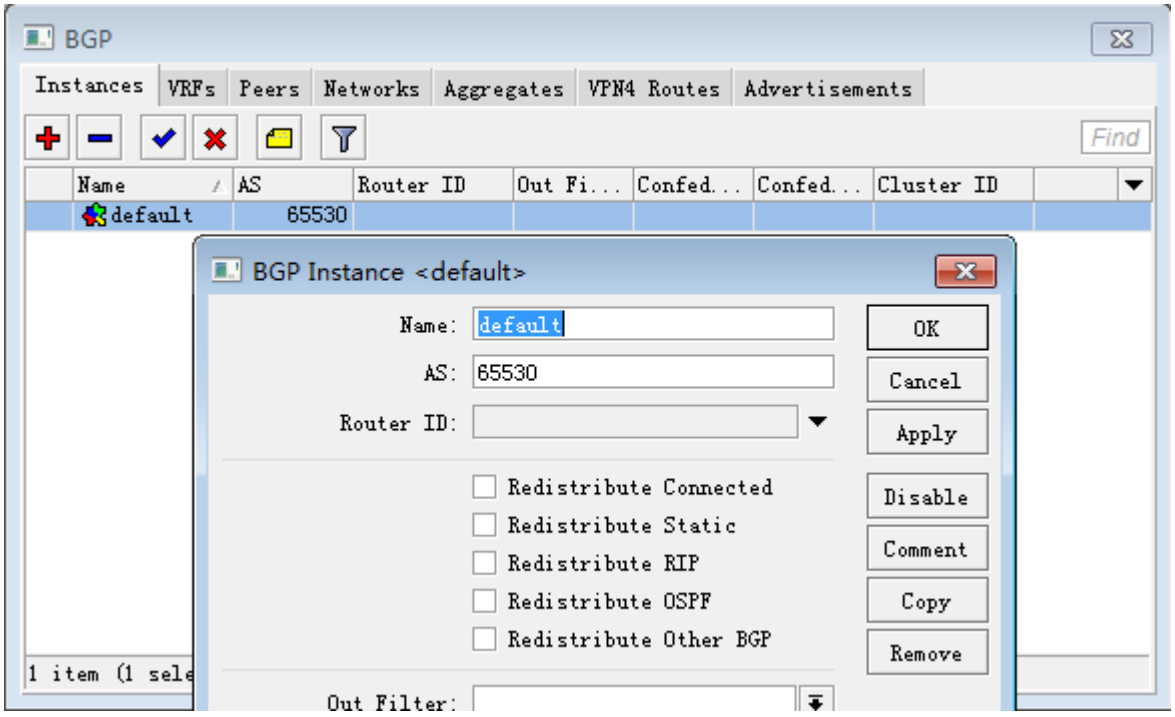
两个对应的路由器添加完成



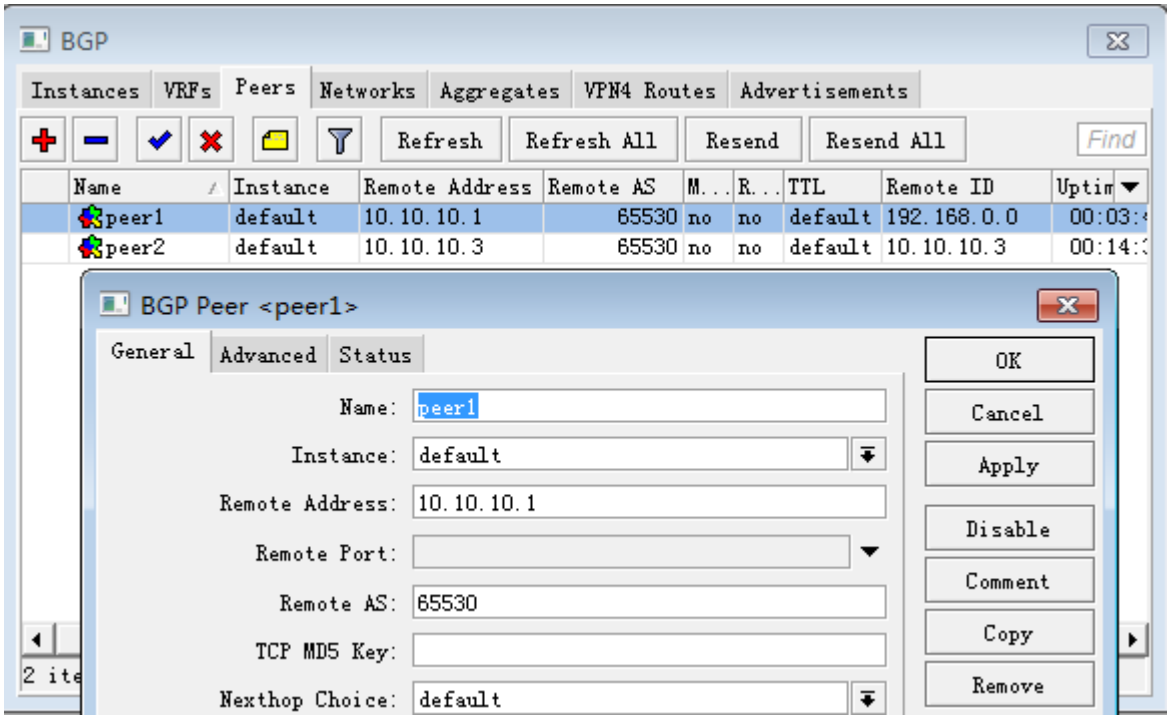
进入 network 申明自己的网段



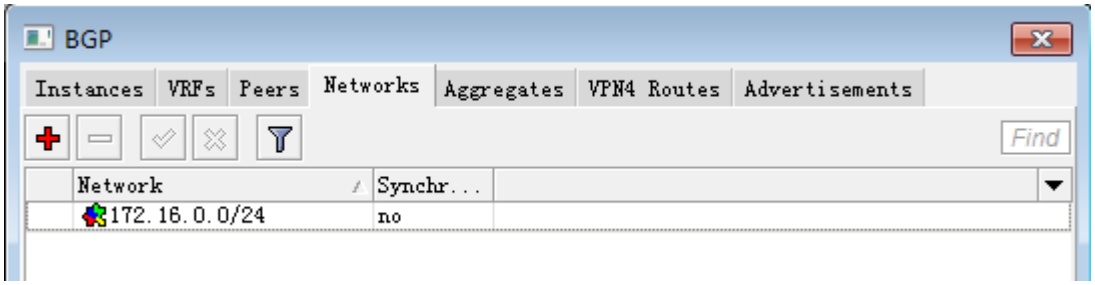
接下来我们配置路由器 B 的 BGP 参数：



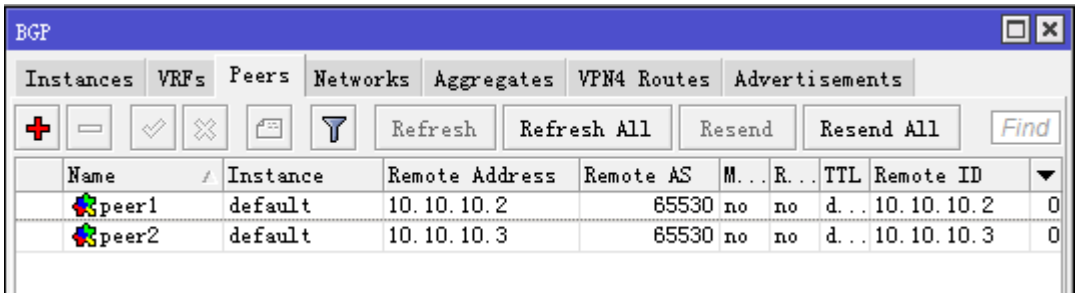
在 peers 的添加 2 个路由器的参数



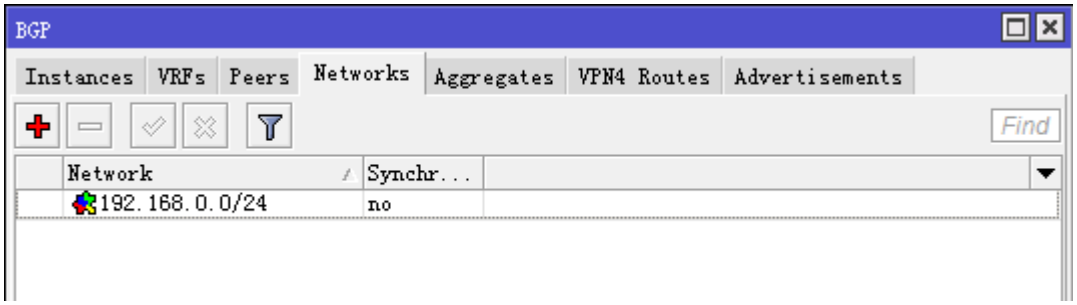
申明自己本地的网络地址段



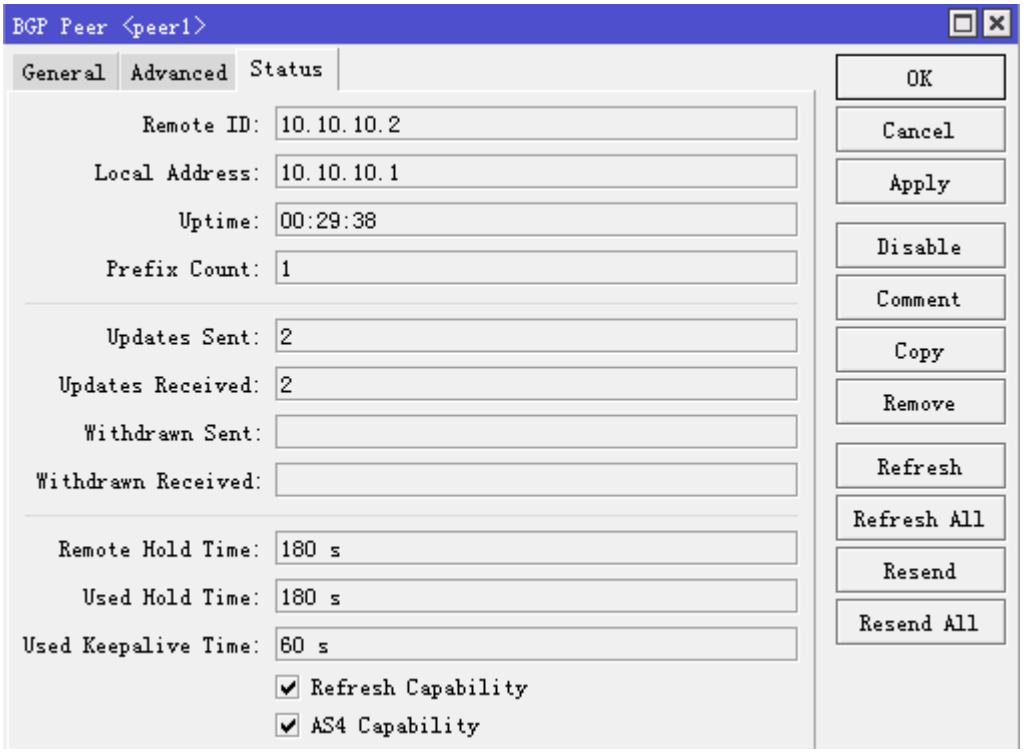
最后是路由器 C 的配置，instances 参数不变，同样在 peer 里，添加相应路由器的参数



申明自己本地的网络地址段

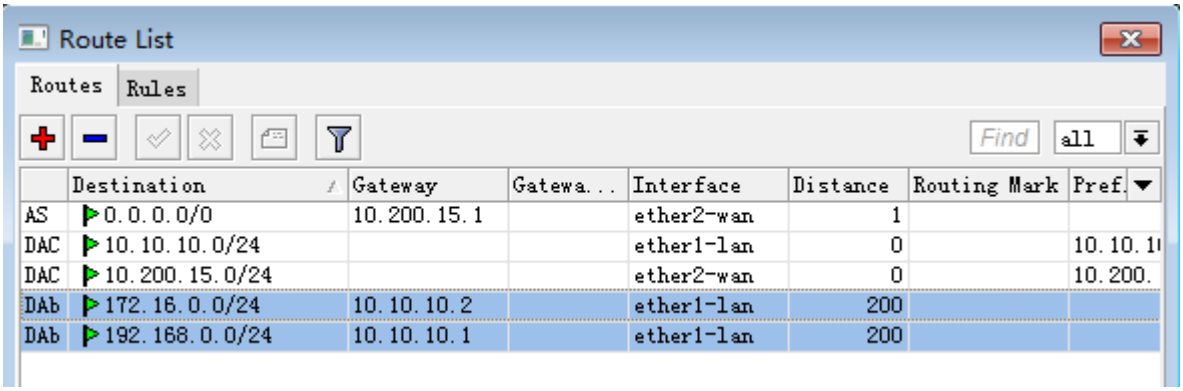


设置完成后，我们在路由器 C 上查看其中一个 BGP 连接状态

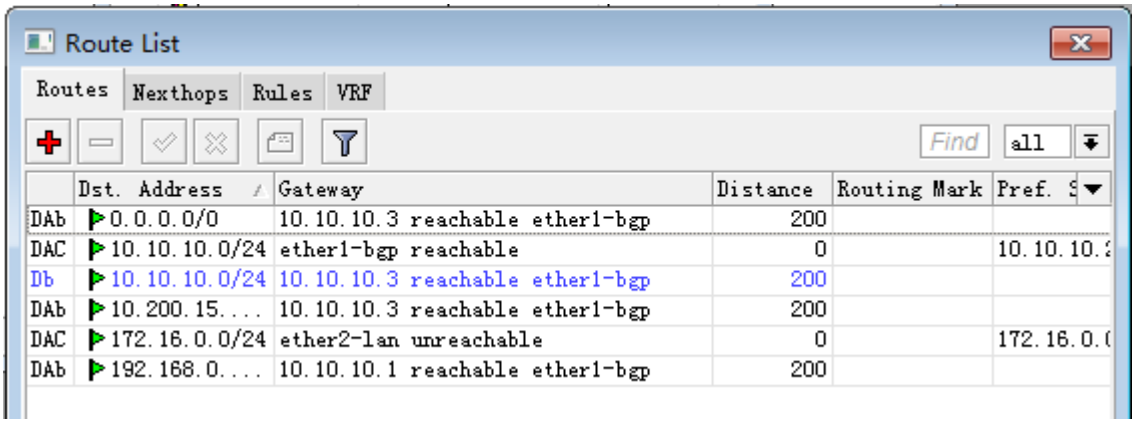


接下来我们可以到每个路由器的 ip route 里查看路由表

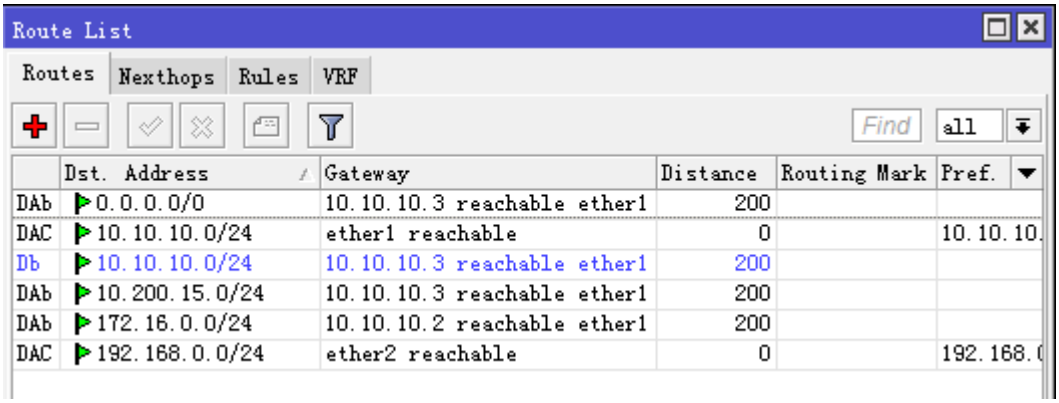
路由器 A：



路由器 B：



路由器 C：



这样一个 BGP 的动态路由建立完成

第三十一章 Proxy 代理

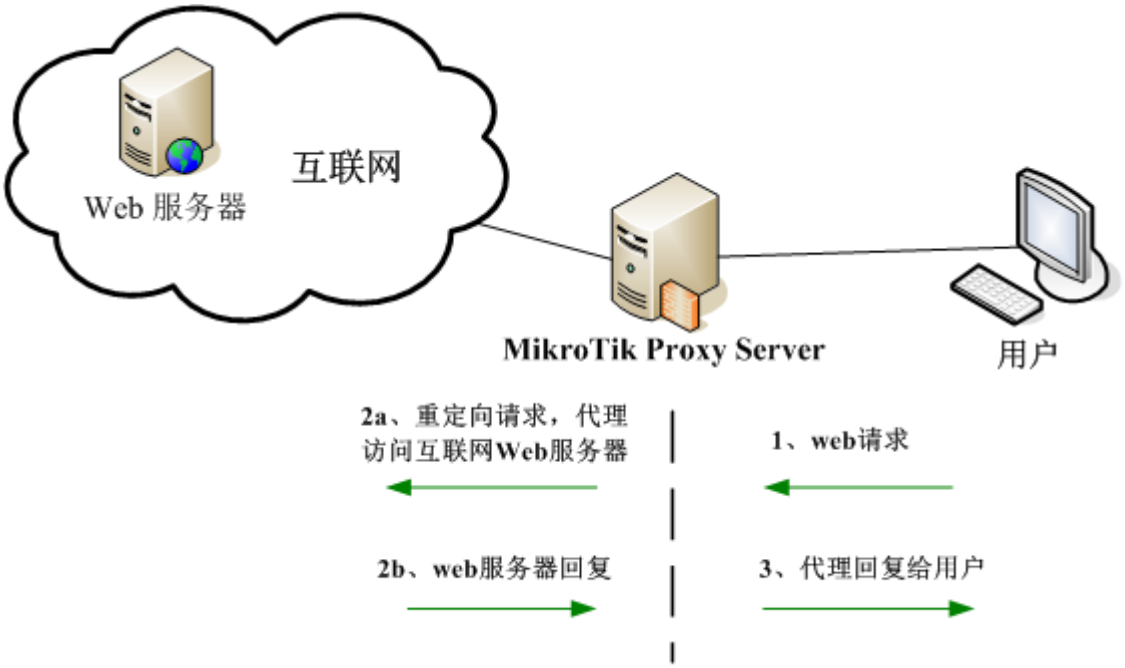
MikroTik RouterOS 可以执行 HTTP 和 HTTP-proxy（FTP 和 HTTP 协议）的代理请求。Proxy Server 具备代理缓存功能，即代理服务器（proxy server）执行对互联网对象请求的缓存，如 HTTP 或 FTP 的部分数据请求，缓存到本地，加速用户访问的速度，MikroTik RouterOS 的代理服务器有以下特点：

- 常规 HTTP 代理
- 透明代理。可以同时透明代理和常规代理
- 源、目的、URL 及请求方法的访问列表
- 缓存访问列表(指定哪些对象需要缓存，哪些不需要)
- 直径访问列表(指定哪些资源应该直接访问，哪些需要通过其他代理服务器)
- 日志功能
- 父级代理服务器支持 – 允许指定其他代理服务器

31.1 Proxy 基本介绍

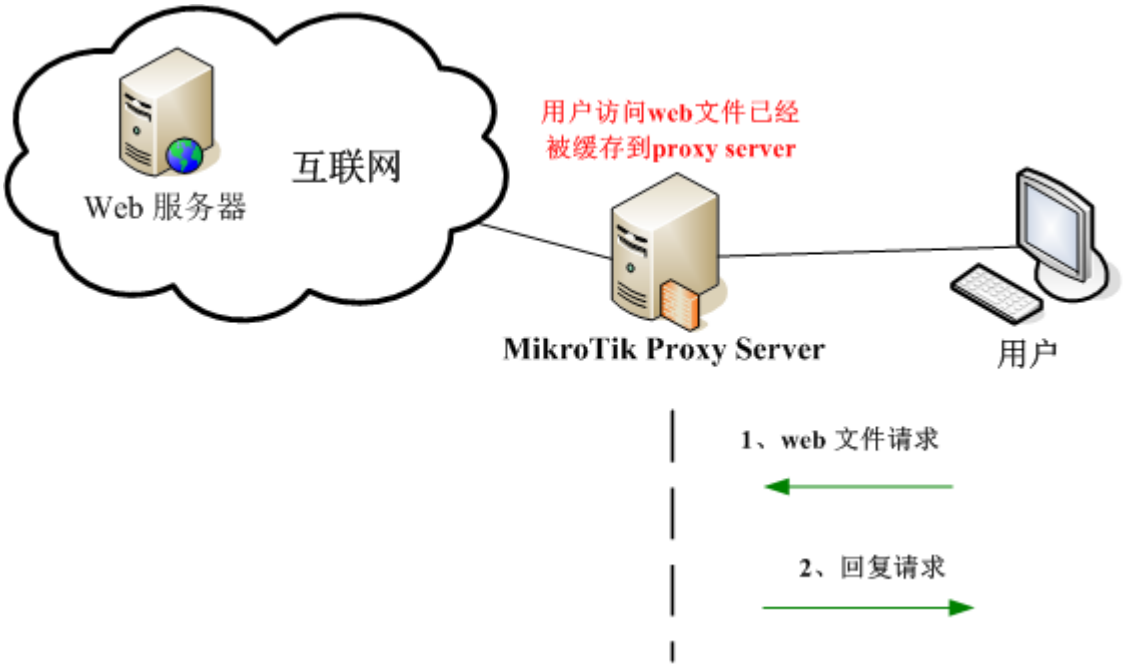
一个 proxy server 通常安装在用户和目标服务器之间,用于代理请求,通过下图可以理解 proxy 是如何工作的：

事例 1，当用户请求 web 服务器，未开启缓存



proxy server 能作为一个 web 代理，监听来至用户的请求，并替用户代理请求 web 服务器。

事例 2，当用户请求 web 服务器档，且 proxy server 的 web 缓存开启：



根据上图，proxy server 作为一个 web 代理服务器同时，可以开启对指定的 web 档缓存，内容保存到 proxy server 本地，当 URL 被第一次请求时，web 内容会被保存，在用户继续访问同一个 URL 请求后，都是由 proxy server 返回给用户，无需再次请求互联网的 web 服务器。即达到了用户访问速度。MikroTik proxy server 只能作为 web 页面小档内容的缓存，无法实现对视频内容缓存，建议使用专用的缓存设备。

通常对 proxy server 的使用基本为以下两点

- **建立 web 缓存**，提升到资源内容的访问速度，较原有互联网访问速度提升（但官方未说明是否能节约互联网流量）
- **建立 HTTP 防火墙**，控制 web 页面访问内容（通常 web 缓存不开启）

MikroTik Proxy server 允许过滤 web 内容，如源/目标地址、TCP 端口、URL、HTTP 请求方法等，这样可以拒绝网站名称，指定 URL 后缀为"*.mp3"文件等

功能包需求：：system
许可等级：Level3
操作路径：/ip proxy（winbox：ip web-proxy）

31.2 Proxy 配置

1、启用常规 proxy 服务

MikroTik RouterOS proxy 配置执行在/ip proxy 菜单，如下启用 proxy 监听端口 8080，设置源地址为 192.168.0.0/24

```
[admin@MikroTik] ip proxy> set enabled=yes port=8080 src-address=192.168.0.0/24

[admin@MikroTik] ip proxy> print
                  enabled: yes
              src-address: 192.168.0.0/24
```

```
                    port: 8080
              parent-proxy: 0.0.0.0:0
               cache-drive: system
       cache-administrator: "admin@mikrotik.com"
       max-disk-cache-size: none
        max-ram-cache-size: 100000KiB
        cache-only-on-disk: yes
  maximal-client-connections: 1000
  maximal-server-connections: 1000
           max-fresh-time: 3d
```

当常规 proxy 服务器配置完成，确保服务器只能是你的客户能使用 proxy，否在将变成开放的 proxy 服务器，常规 proxy 服务器允许客户端使用网页浏览器配置代理访问，不同网页浏览器设置不同，可以百度如何配置网页 proxy。

2、启用透明 proxy 服务

RouterOS 能实现透明代理服务器，无需客户端网页浏览器做任何配置，RouterOS 会将所有 HTTP 请求重定向到本地的 proxy 服务上，这样的处理对于用户是完全透明（用户可能不知道中间存在代理服务器），当开启缓存后，能提升用户访问速度。

设置你的代理端口转移，我们需要进入 dst-nat 配置所有访问 80 端口的数据重定向到本地的 8000 端口：

```
[admin@MikroTik] ip firewall nat> add chain=dstnat protocol=tcp dst-port=80
action=redirect to-ports=8000
[admin@MikroTik] ip firewall nat>
```

你可以设置重定向的 ip 地址范围:

```
[admin@MikroTik] ip firewall nat> add chain=dstnat src-address=192.168.10.0/24
protocol=tcp dst-port=80 action=redirect to-ports=8000
[admin@MikroTik] ip firewall nat>
```

31.3 启用缓存.

在这个实例是基于当你建立了正在运行的 proxy 服务，你需启用缓存功能的介绍。

- **基于 RAM 缓存：**你拥有很好的硬设备，足够的 RAM 用于缓存（RouterOS x86 版本仅支持 2G RAM），如果启用 RAM 缓存，分配 256MB 或更小，并不会对你网络缓存有任何益处，因此不建议使用 RAM 作为缓存。
- **基于 Store 缓存：**利用 U 盘或 SATA 硬盘作为存储介质（说实话由于 RouterOS 系统自身限制无法应用于大规模的缓存，因此只能小范围应用，比起 nginx 和 traffic server 那是差距是非常的大）

开启 proxy 缓存，会加重 RouterOS 的负载，特别是连接会话数和存储 I/O 读写，如果要开启建议做测试或运行前的评估。

注意：以下的配置实例，基于 RouterOS v6.20 后，因此选择相应的版本做操作。

RAM proxy 缓存

即使用内存作为缓存存储介质，相关配置命令：

- max-cache-size：定义最大缓存大小
- max-cache-object-size：定义缓存对象大小
- cache-on-disk：是否缓存到内存

下面是 RAM 缓存的配置命令

```
[admin@MikroTik] /ip proxy> set max-cache-size=unlimited
max-cache-object-size=50000KiB cache-on-disk=no
...
[admin@MikroTik] /ip proxy> print
               enabled: yes
           src-address: ::
                  port: 8080
             anonymous: no
          parent-proxy: 0.0.0.0
     parent-proxy-port: 0
    cache-administrator: webmaster
         max-cache-size: unlimited
  max-cache-object-size: 500000KiB
          cache-on-disk: no
 max-client-connections: 600
 max-server-connections: 600
        max-fresh-time: 3d
  serialize-connections: no
      always-from-cache: no
         cache-hit-dscp: 4
            cache-path: proxy-cache
```

Store proxy 缓存

即使用硬盘或 U 盘作为缓存存储介质，相关配置命令：

- max-cache-size：定义最大缓存大小
- max-cache-object-size：定义缓存对象大小
- cache-on-disk：是否缓存到内存
- cache-path：指定存储路径

这里我们插入 U 盘（具体 U 盘格式化和路径配置，请参见 35.3 章节），并在 file 下查看路径

```
[admin@MikroTik] > file print
 # NAME                                                TYPE
 0 skins                                               directory
 5 usb1/proxy                                           directory
```

```
 6 usb1/proxy/cache                                    web-proxy store
 7 usb1/lost+found                                     directory
```

配置内存缓存命令：

```
[admin@MikroTik] > ip proxy set cache-on-disk=yes cache-path=/usb1/proxy/cache
[admin@MikroTik] > ip proxy print
               enabled: yes
           src-address: ::
                  port: 8080
             anonymous: no
          parent-proxy: 0.0.0.0
     parent-proxy-port: 0
    cache-administrator: webmaster
        max-cache-size: unlimited
  max-cache-object-size: 50000KiB
         cache-on-disk: yes
 max-client-connections: 600
 max-server-connections: 600
        max-fresh-time: 3d
  serialize-connections: no
     always-from-cache: no
        cache-hit-dscp: 4
            cache-path: usb1/proxy/cache
```

查看缓存是否工作

```
[admin@MikroTik] > ip proxy monitor
               status: running
               uptime: 2w20h28m25s
    client-connections: 15
    server-connections: 7
             requests: 79772
                 hits: 30513
           cache-used: 481KiB
       total-ram-used: 1207KiB
 received-from-servers: 4042536KiB
      sent-to-clients: 4399757KiB
  hits-sent-to-clients: 176934KiB
```

属性描述

cache-administrator (*文本*; default: **webmaster**) – 显示在代理错误页面的管理员 e-mail
cache-drive (system | *name*; default: **system**) -指定用于存贮缓存对象的目标磁盘机。你可以使用控制面板完成来查看可用驱动器列表
cache-only-on-disk (yes | no; default: **yes**) -是否在描述磁盘上缓存目录的内存中创建的数据库。这样会减少内存消耗，但会影响速度
enabled (yes | no; default: **no**) - 代理服务器是否启用

max-disk-cache-size (none | unlimited | *整型*: 0..4294967295; default: **none**) -指定最大磁盘缓存大小，以 kb 计算
max-fresh-time (*时间*; default: **3d**) - 存贮缓存对象的最大时间。一个目标的合法时间一般是由对象本身定义的，但以防太长，你可以覆盖最大值
maximal-client-connecions (*整型*; default: **1000**) -客户接受的最大连接数（任何更多的连接都将被拒绝）
maximal-server-connectons (*整型*; default: **1000**) - 到服务器的最大连接数（任何更多来自客户的连接都将被挂起知道一些服务器连接结束）
max-object-size (*整型*; default: **2000KiB**) - 大于指定长度的对象将不会保存在磁盘上。以 kb 计算。如果你想获得一个更高的比特命中率，你应该增加该值（一个 2MiB 对象撞击代表 2048 个 1KiB 撞击）。如果你更想增加速度而不是接生带宽，你应该把这个值设的低一些
max-ram-cache-size (none | unlimited | *整型*: 0..4294967295; default: **none**) -指定最大 RAM 缓存大小，以 kb 计算
parent-proxy (*IP address*:*port*; default: **0.0.0.0:0**) - 把所有请求定向到的 IP 地址及其他 HTTP 代理端口（异常会在"direct access"列表中定义）
0.0.0.0:0 – 没有使用父级代理
port (*port*; default: **8080**) -代理服务器将监听的 TCP 端口。这个会在所有想使用该服务器作为 HTTP 代理的客户上定义。透明（对客户使用零配置）代理设置可以通过使用目的 NAT 特性在 IP 防火墙重定向 HTTP 请求到该端口完成
src-address (*IP address*; default: **0.0.0.0**) - Web 代理将使用这个地址连接父级代理或 web 网站
0.0.0.0 – 合适的 **src-address** 将会自动从路由列表中取出

31.4 access 访问列表

操作路径: ***/ip proxy access***

访问列表像防火墙规则一样采用 FIFO 先进先出算法，规则从顶到底连续处理。第一条匹配的规则指定对连接做何处理。链接将匹配源地址、目标地址、目标端口、URL 链接的子字符串或请求模式

如果链接匹配一条规则，**action** 属性将指定这条规则是执行允许通过或者拒绝。如果没有匹配任何规则，默认将被允许通过

属性描述

action (allow | deny; default: **allow**) -指定通过或拒绝已匹配的包
dst-address (*IP address*/*netmask*) - IP 包的目的地址
dst-host (*wildcard*) - IP 地址或用于连接目标服务器的 DNS 名（这是一个在指定端口与到特定网址路径之前写在他的浏览器的字符串）
dst-port (*port*{1,10}) - 包到达的列表或端口范围
hits (*只读: 整型*) - 被规则修正的请求数
local-port (*port*) -指定包接受的 web 代理端口。这个值应该匹配 web 代理监听的其中一个端口
method (any | connect | delete | get | head | options | post | put | trace) -用于请求的 HTTP 方法（参见本文档最后面的 HTTP 方法部分）
path (*wildcard*) -在目标服务器中的被请求页面名（例如，特定网页的名称或不含它存在的服务器名称的文档）
redirect-to (*文本*) - 以防访问被该规则拒绝，用户应被重定向到这里指定的 URL
src-address (*IP address*/*netmask*) - IP 包的源地址

通过以下事例更好理解访问列表的功能，如禁止访问指定网站.

```
/ip proxy access add dst-host=www.facebook.com action=deny
```

以上配置将禁止访问 http://www.facebook.com 网站，类似的配置也可以仅限制访问源地址，如 192.168.1.0/24 的用户地址段访问 facebook。

```
/ip proxy access add src-address=192.168.1.0/24 dst-host=www.facebook.com action=deny
```

你可以禁止访问包含指定字符的 URL 网站链接，通过“:“匹配包含的字符，如下面：

```
/ip proxy access add dst-host=:mail action=deny
```

该语句将禁止所有包含 mail 的网站连结，如 www.mail.com, www.hotmail.com, mail.yahoo.com

我们也可以禁止下载指定文件类型，如.flv, .avi, .mp4, .mp3, .exe, .dat,

```
/ip proxy access
add path=*.flv action=deny
add path=*.avi action=deny
add path=*.mp4 action=deny
add path=*.mp3 action=deny
add path=*.zip action=deny
add path=*.rar action=deny.
```

注： dst-host 和 **dst-path** 支持通配符匹配，如果仅输入"example"，不会匹配"example.com"，可用通配符'*' (匹配任意字符，如”example*”， '?' 匹配任何一个字符)。也能使用正则表达式，如果使用正则表达式，那么它应该以冒号(':')开始。

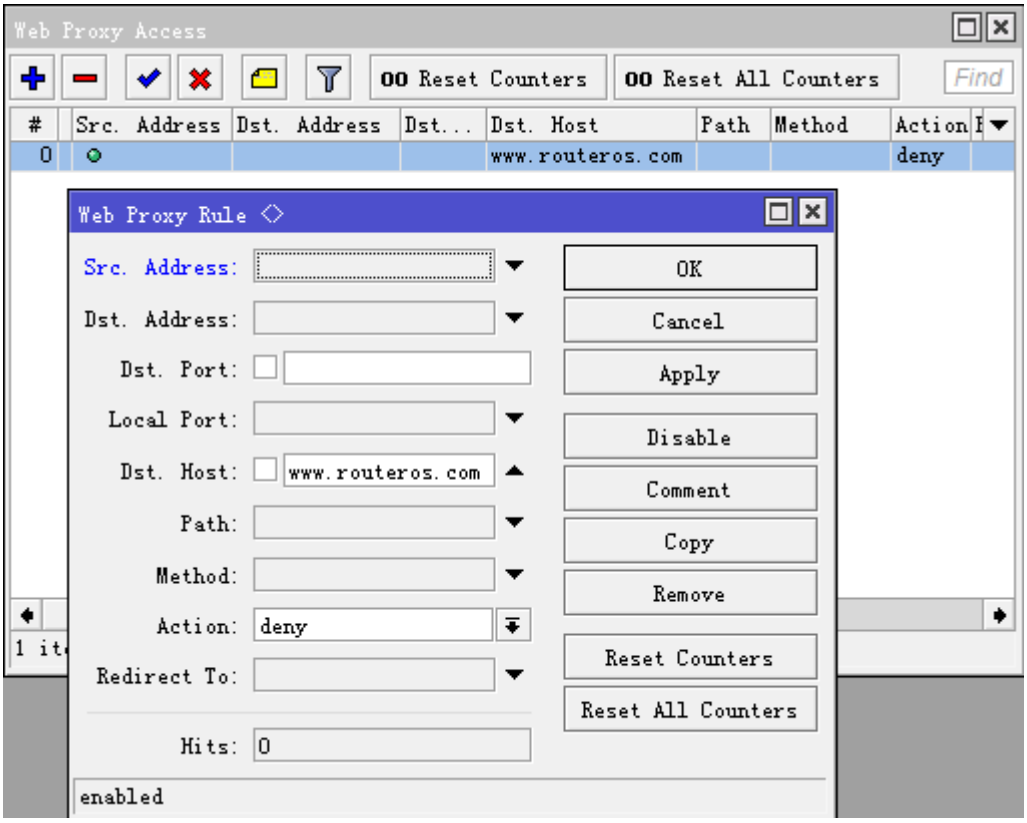
关于使用正则表达式：：

- **** 符号序列是用于在控制面板输入****字符的
- **\.** 样式的意思为只是 **.** (在正则表达式单独的点表示任何符号)
- 显示在给出样式之前任何符号都不允许，我们在样式开始使用**^**符号
- 指定在给出样式之后任何符号都不允许，我们在样式结束的地方使用符号**$**

proxy 访问控制实例

设置禁止访问网站，该设置将禁止访问 http://www.routeros.com

```
/ip proxy access
dst-host=www.routeros.com action=deny
```



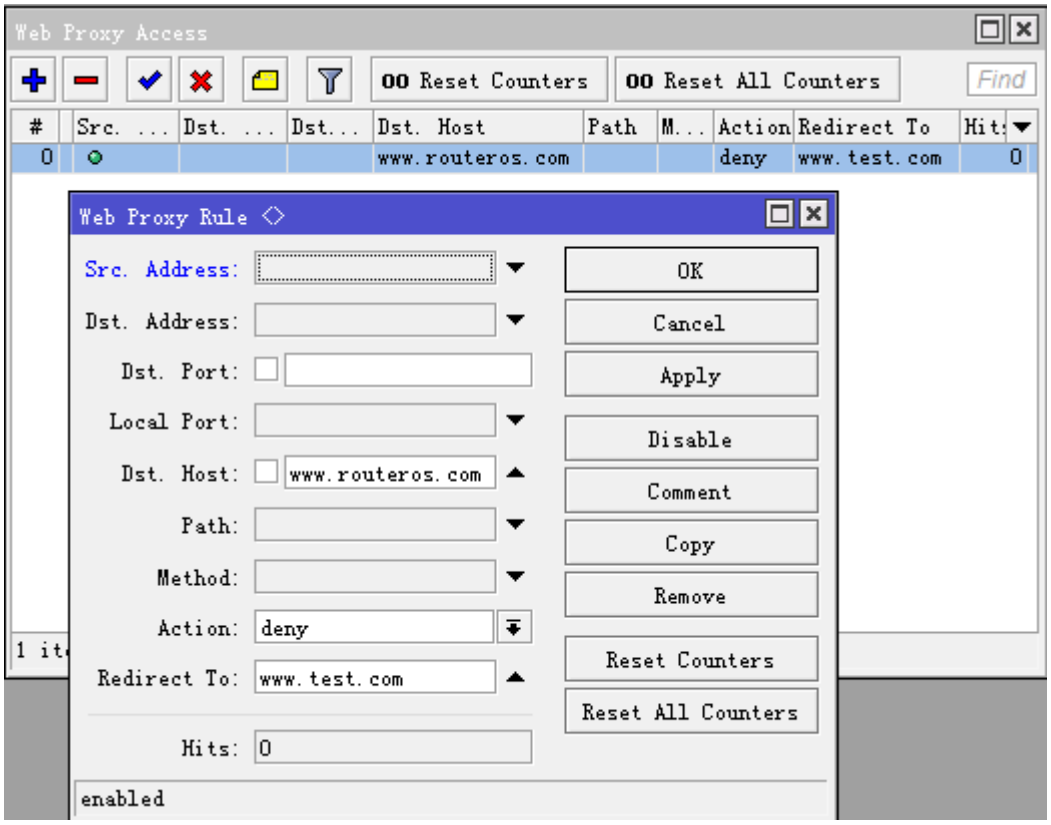
我们可用阻止档如“.mp3, .exe, .dat, .avi”的下载。

```
/ip proxy access
path=*.exe action=deny
path=*.mp3 action=deny
path=*.zip action=deny
path=*.rar action=deny
```

同样我可用阻止所有含“mail”的关键词链接

```
/ip proxy access
dst-host=:mail action=deny
```

也可以通过重定向网站访问到指定页面，如重定向 www.routeros.com 到 www.test.com



31.5 Direct 直接访问列表

操作路径: ***/ip proxy direct***

如果指定了 **parent-proxy** 属性，会告诉代理服务器是否尝试交给父级代理服务器或直接连接被请求的服务器。Direct 访问列表类似于 Access 访问列表，就像前一章节描述的访问列表一样，但不同于 access 访问列表功能的是，直接访问列表默认执行拒绝操作，这种情况发生在没有任何规则匹配或指定的情况下。

属性描述

action (allow | deny; 默认: **allow**) -指定对匹配参数的动作
　　allow -总是直接绕过父级路由器解决匹配的请求
　　deny - 通过父级代理以解决匹配请求。如果没有指定则这个与 **allow** 的效果相同
dst-address (*IP address/netmask*) - IP 包的目的地址
dst-host (*wildcard*) - 用于连接到目标服务器的 IP 地址或 DNS 名（这是在指定特定网页到达的端口与路径之前用户写在他的浏览器中的字符串）
dst-port (*port*{1,10}) -包到达的列表或端口范围
hits (*只读: 整型*) -被规则修正过的请求数
local-port (*port*) -指定包接受的 web 服务器端口。这个值应该与 web 代理监听的其中一个匹配
method (any | connect | delete | get | head | options | post | put | trace) - 用于请求中的 HTTP 方法(参见本文档最后的 HTTP 方法部分)
path (*wildcard*) - 目标服务器中的被请求页面名（例如，特定 web 页面名或不含它存在的服务器名的文档）
src-address (*IP address/netmask*) - IP 包的源地址

31.6 cache 缓存管理

操作路径: ***/ip web-proxy cache***

缓存访问列表指定那些请求（域名、服务器、页面）应该由 web 代理本地缓存，而那些不用缓存。这个列表与 web 代理访问列表完全一样地执行。

属性描述

action (allow | deny; 默认: **allow**) - 指定对已匹配包的动作
  allow – 允许请求缓存对象
  deny – 不允许请求缓存对象
dst-address (*IP 地址*/子网掩码) - IP 包的目的地址
dst-host (*wildcard*) - 用于连接到目标服务器的 IP 地址或 DNS 名（这是在指定特定网页到达的端口与路径之前用户写在他的浏览器中的字符串）
dst-port (端口{1,10}) -包到达的列表或端口范围
hits (*只读: 整型*) - 被规则修正过的请求数
local-port (端口) -指定包接受的 web 服务器端口。这个值应该与 web 代理监听的其中一个匹配
method (any | connect | delete | get | head | options | post | put | trace) - 用于请求中的 HTTP 方法(参见本文档最后的 HTTP 方法部分)
path (*wildcard*) - 目标服务器中的被请求页面名（例如，特定 web 页面名或不含它存在的服务器名的文档）
src-address (*IP 地址*/子网掩码) - IP 包的源地址

31.7 连接列表

操作路径: ***/ip proxy connections***

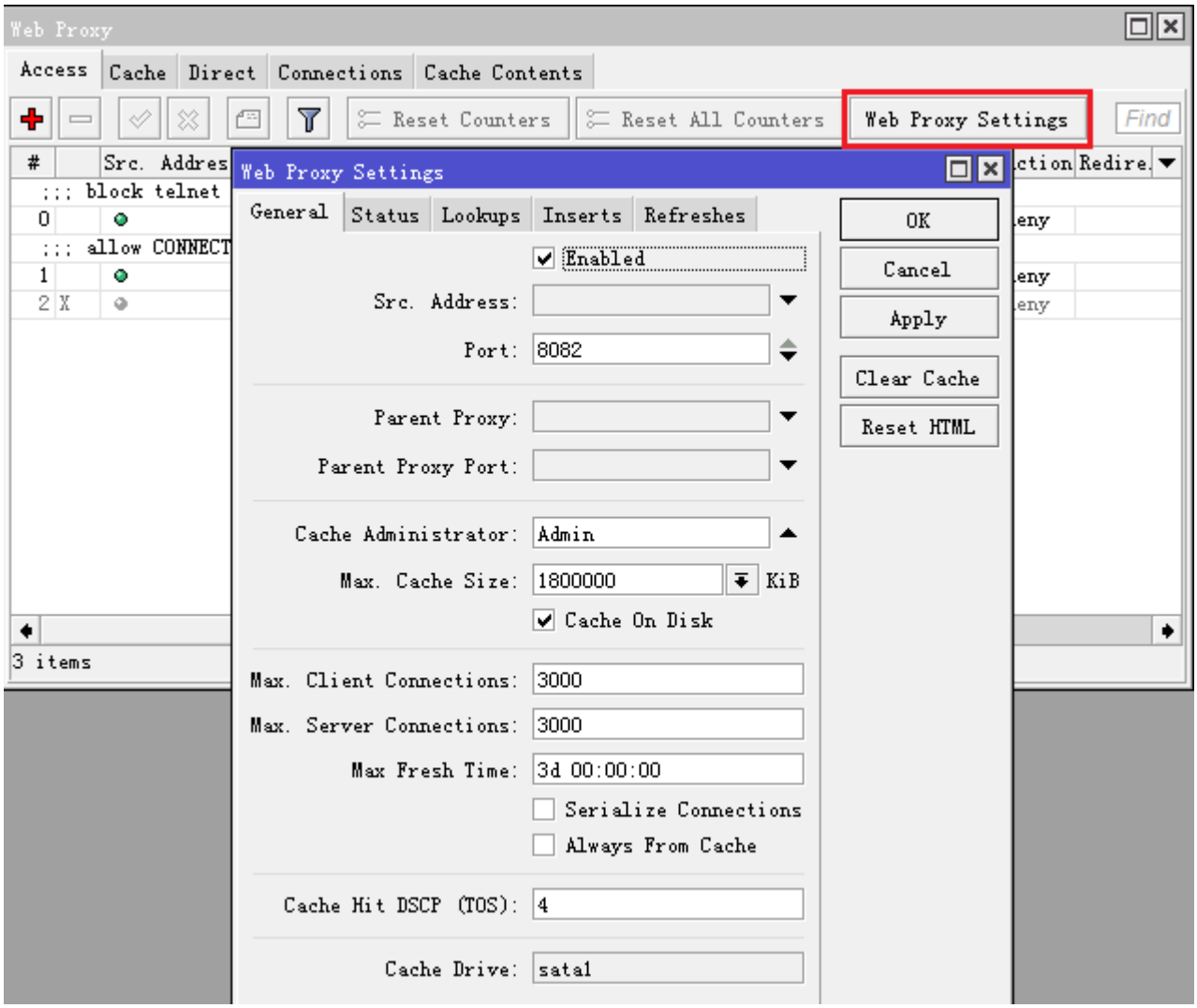
这个目录包含代理存储的当前连接的清单。

属性描述

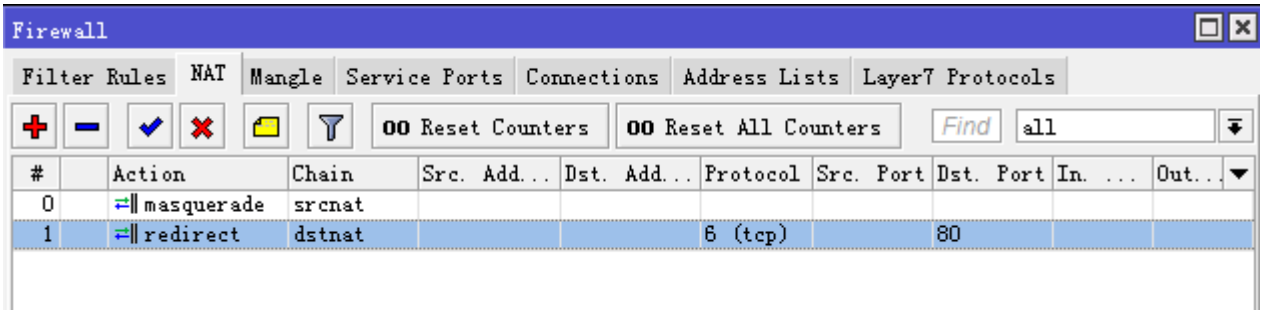
dst-address (*只读: IP 地址*) – 连接的 IP 地址
protocol (*只读: 文本*) – 协定名
rx-bytes (*只读: 整型*) - 客户接收的字节量
src-address (*只读: IP 地址*) – 连接源发站的 IP 地址
state (*只读:* closing | connecting | converting | hotspot | idle | resolving | rx-header | tx-body | tx-eof | tx-header | waiting |) – 打开连接的状态
  closing - 数据传输完成，连接正在最终完成
  connecting - 建立 toe 连接
  hotspot – 检查是否 hotspot 认证允许继续（对 hotspot 代理）
  idle - 闲置状态
  resolving – 解析服务器的 DNS 名
  rx-header – 接受 HTTP 标题
  tx-body – 传输 HTTP 正文给客户
  tx-eof - 写组块端(当传换为分组的回应)
  tx-header – 传输 HTTP 标题给客户
  waiting – 等待来自同等体的传输
tx-bytes (*只读: 整型*) - 由客户发送的字节数

31.8 Web proxy 应用事例

首先我们启用 web-proxy 服务器，这里我们定义代理端口为 TCP/8082 首先配置 web-proxy，配置参数如下：



现在，设置透明传输数据重定向，将所有访问 80 端口的数据重定向到 web-proxy 的 8082 端口上：



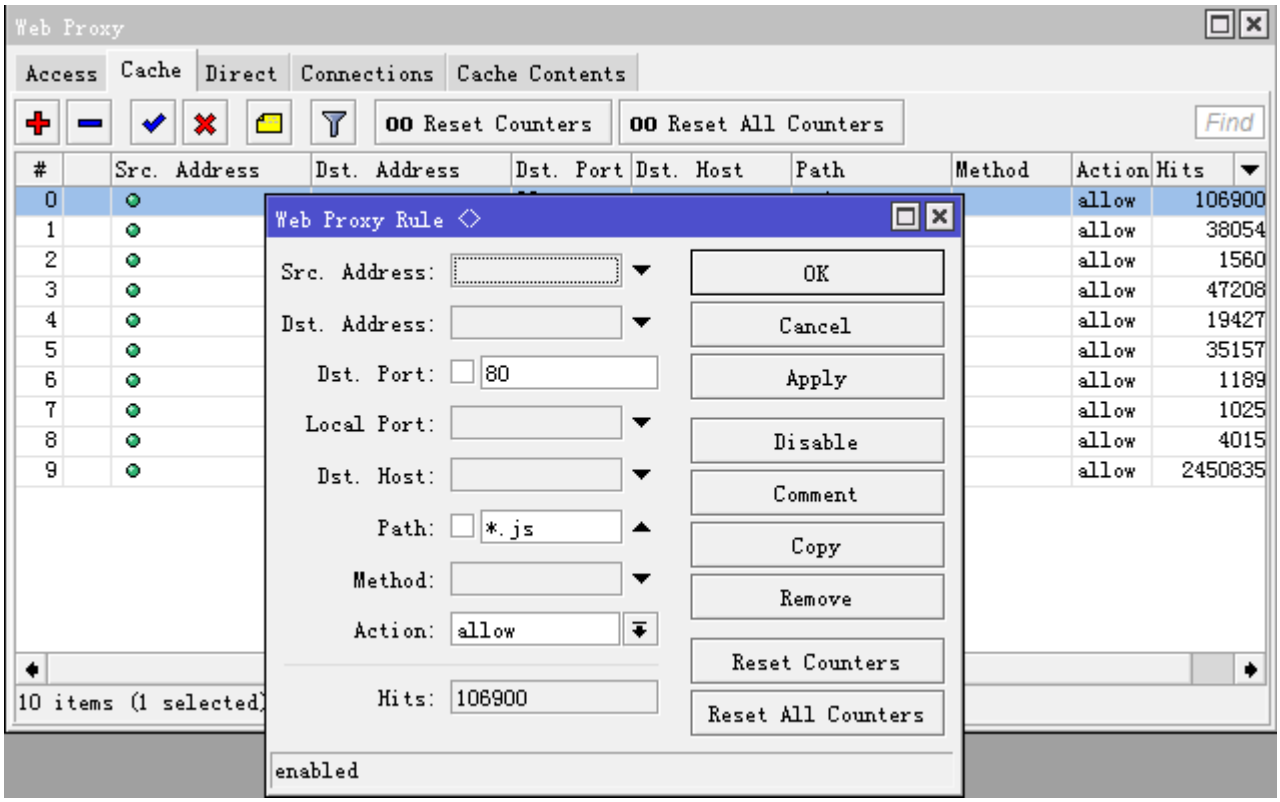
CLI 操作命令

```
/ip firewall nat
chain=dstnat protocol=tcp dst-port=80 action=redirect to-ports=8082
```

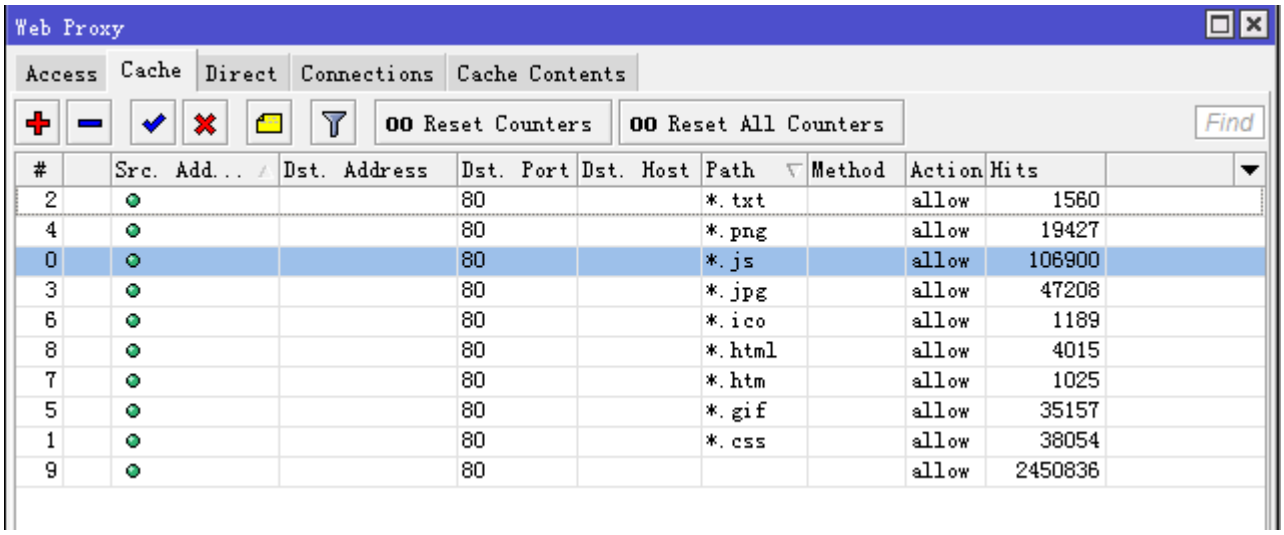
确定你路由器本地的 Proxy 没有打开代理，并禁止外网通过路由器代理上网：

```
/ip firewall filter
chain=input in-interface=<Your WAN Port> src-address=0.0.0.0/0 protocol=tcp dst-port=8082 action=drop
```

缓存我们需要的档例如后缀名为：.js .css .html .jpg…等，这样有利于提高缓存命中率。



我们缓存的列表

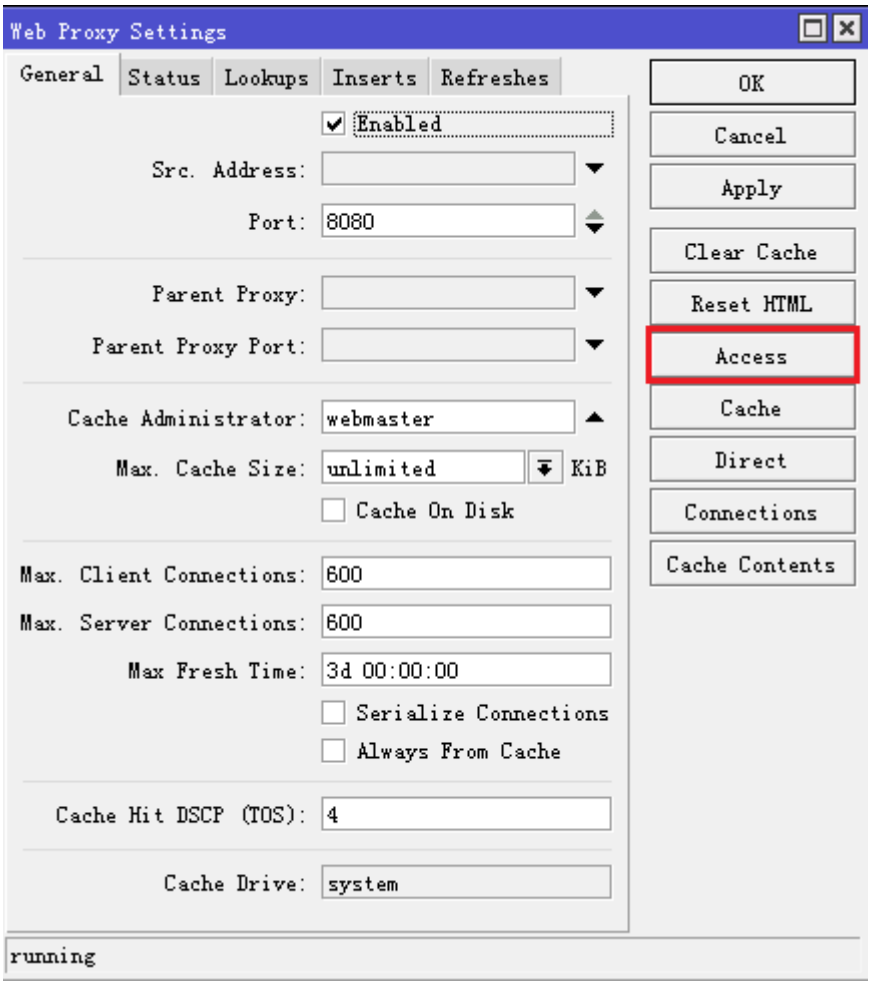


以上为 Cache 的事例，不过建议不要对 html、htm 一类做缓存，因为这一类更新较多，容易造成访问失效，以上操作仅供参考。

31.9 重定向 URL 请求

通过 web proxy 重定向一个用户访问的请求，比如当一个主机打开一个网站时，我们通过 proxy 的重定向功能，将一个网站的图片劫持或显示其他图片或内容，

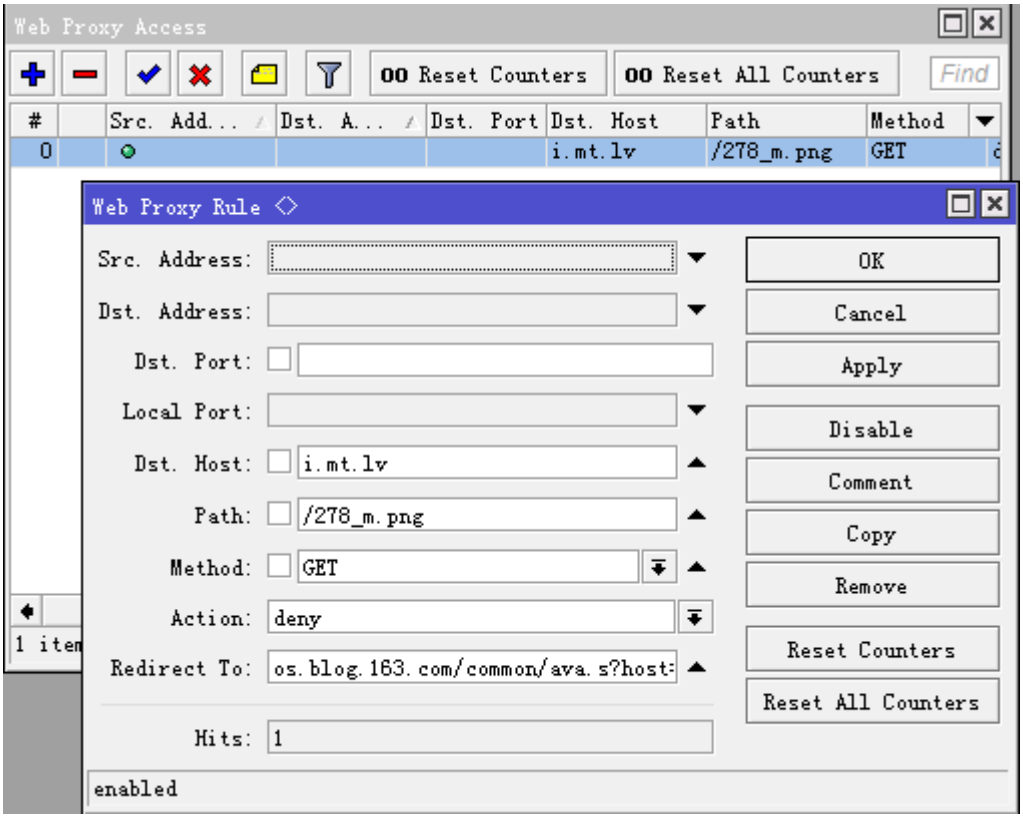
要实现重定向功能，需要使用到 proxy 的 access 功能，通过将访问数据重定向，在这个 proxy 配置里我们没有考虑 cache 设置。



进入 nat，配置 http 80 端口重定向

```
/ip firewall nat
add chain=dstnat action=redirect to-ports=8080 protocol=tcp dst-port=80
```

假设我们需要将”www.mikrotik.com”主页上的 ccr1036 图片更换，他的图片链接是 http://i.mt.lv/278_m.png，我们需要将这张图片重定向，并显示另外一个网站的一张图片”http://os.blog.163.com/common/ava.s?host=athlon_sds&b=2&r=1338650981964”



设置目标主机：Dst-host=i.mt.lv，设置图片路径：path=/278_m.png，设置 http 模式：Method=GET，执行方式：Action=deny，重定向 URL 到以下连结：redirect-to=os.blog.163.com/common/ava.s?host=athlon_sds&b=2&r=1338650981964

设置完成后，最后清空 IE 或者其他浏览器的图片缓存，避免因为浏览器缓存造成测试无果的情况，下面打开 www.mikrotik.com 的情况：

Proxy 这个功能对 CPU 和内存消耗较大，所以需要考虑自己用户和请求量。至于这个功能的用途大家可以自己考虑。

第三十二章 CHR 与虚拟化

关于 RouterOS 的虚拟化可以分为以下情况：

v3.30 前是 Xen 虚拟机，但在 v3.30 后用 KVM 替代了 Xen 虚拟机，RouterOS v4.0 现在支持 2 种不同的虚拟化技术分别是：MetaRouter 和 KVM。而从 v6 开始，MikroTik 开始针对虚拟机平台，推出 CHR，RouterOS 系统直接定制出各种虚拟化平台的镜像文件，也解决了 x86 平台出现不兼容的情况。

CHR

Cloud Hosted Router（CHR）是基于虚拟主机开发的 RouterOS 版本，对 x86 平台的重新定义其使用方式，基于 64bit CPU 构架，能运行在 VMWare, Hyper-V, VirtualBox, Xen。

Metarouter

MetaRouter 是 MikroTik 开发的，当前仅支持 RouterBOARD 400 系列(mips-be)，也只能创建 RouterOS 的虚拟机。 MikroTik 计划添加更多的功能到 MetaRouter 中，因此新的硬件支持将会添加到 MetaRouter 中，甚至会超过 Xen 的功能。

Xen

Xen 是基于 Linux Xen 虚拟机项目，应用于当前的 RouterOS x86 系统（PC），Xen 虚拟机能创建不同的操作系统，但 Xen 已经在 3.0 版本被淘汰，接替他的是 KVM。

KVM

Kernel-based Virtual Machine（KVM）提供虚拟基于 x86 的技术。为 RouterOS 基于 PC 主机提供完善的虚拟技术，KVM 要求支持 CPU 虚拟技术，如 Intel 的 VT-x 或者 AMD-V 等技术。运行要求虚拟机至少分配 16MB 内存，有足够的硬盘空间运行相应的镜像档。镜像档不能在创建后被增加，并且大小只能是你导入创建时的镜像不变。

32.1 CHR(Cloud Hosted Router)

Cloud Hosted Router（CHR）是基于虚拟主机开发的 RouterOS 版本，重新定义了 x86 平台的使用方式，系统基于 64bit 构架，能运行在 VMWare, Hyper-V, VirtualBox, Xen 平台上。虽然 CHR 版本的 RouterOS 没有功能等级限制，但在网络接口速率上做了限制，免费版本是 1Mbit 的网络接口速率，也提供不同速率的网络接口，但需要购买。MikroTik 对 CHR 的特点定义为：

CHR 的特点如下：

- 可做各种功能的实验；
- 可让你在培训课程做 RouterOS 教学演示；
- 没有 24 小时或 Demo 许可限制；
- 发挥 x86 平台虚拟化技术，实现一机多用；
- 支持 64 位构架，性能能进一步释放。

为什么在 CHR 针对几个虚拟平台出现基于 64bit 处理器构架，而传统 RouterOS 在 x86 硬件平台上则没有，仍然采用 32bit 构架，这点应该和硬件平台多样化有关系，涉及到要求兼容驱动过多等因素等关系，毕竟虚拟化平台就那几个硬件驱动优化也不是难事，因此 CHR 的版本出现了 64bit 构架，且 32bit 构架内存仅支持 4G 不到，而 64bit 内存支持方面是客观的。

系统要求

最低配置要求：

- 虚拟主机要求支持 64 位 CPU
- CHR 运行要求提供最低 128 MB RAM
- CHR 存储空间最低要求 128 MB

CHR 已测试平台包括如下：

- 基于 Linux 和 OS X 的 VirtualBox 5
- 基于 OS X 的 VMWare Fusion 7 和 8
- 基于 OS X 的 Qemu 2.4.0.1
- 基于 Windows Server 2012 的 Hyper-V (暂时仅支持第一代 Hyper-V 支持)
- 基于 VMware Esxi v5

安装 CHR

当前 MikroTik 提供 4 个不同的虚拟镜像文件：

- RAW disk image (.img file)
- VMWare disk image (.vmdk file)
- Hyper-V disk image (.vhdx file)
- VirtualBox disk image (.vdi file)

注意：这些是虚拟主机镜像文件，非系统安装镜像，需对应不同虚拟机平台创建虚拟机后导入

CHR 安装步骤

- 第一步： 下载 虚拟镜像文件
- 第二步： 创建虚拟主机
- 第三步： 使用之前下载的镜像文件作为虚拟磁盘驱动
- 第四步： 导入完成后，启动 CHR 虚拟机
- 第五步： 登录 CHR，默认账号 admin，密码为空

下面是官方提供的 4 种虚拟平台的安装操作链接：

- VMWare Fusion / Workstation
- VirtualBox
- Hyper-V
- Amazon Web Services (AWS)

32.2 CHR 许可

CHR 有 4 种许可:

- ***free***
- ***p1*** *perpetual-1* ($45)
- ***p10*** *perpetual-10* ($95)
- ***p-unlimited*** *perpetual-unlimited* ($250)

除了以上 4 种许可，还可以获取 60 天免费试用许可，该许可能用于所有的付费许可等级，申请 60 天免费试用必须登录 www.mikrotik.com，注册 MikroTik 账号。

Perpetual（永久）是无时间限制许可（购买一次，使用永久），一个低版本的 Perpetual ，可以通过购买升级到高级版的 Perpetual ，如 P1 升级到 P10 或者 P-unlimited。

当运行 CHR 主机，通过 MikroTik 的账户访问更新许可，如果 CHR 无法更新许可或者许可失效，将无法升级 RouterOS 最新版本。

许可	接口限制	价格
Free	1Mbit	免费
P1	1Gbit	45 USD
P10	10Gbit	95 USD
P-Unlimited	Unlimited	250 USD

等级许可说明

p1 等级

p1 (perpetual-1) ，无限期使用，每个接口限制为 1Gbps，其他功能没有任何限制。可以通过登录 MikroTik 网站重新申请许可升级为 P10 或 P-unlimited.

p10 等级

p10 (perpetual-10) ，无限期使用，每个接口限制为 10Gbps，其他功能没有任何限制。可以通过登录 MikroTik 网站重新申请许可升级为 P-unlimited.

p-unlimited 等级

p-unlimited (perpetual-unlimited) 无限期使用，没启用任何功能限制

Free（免费）

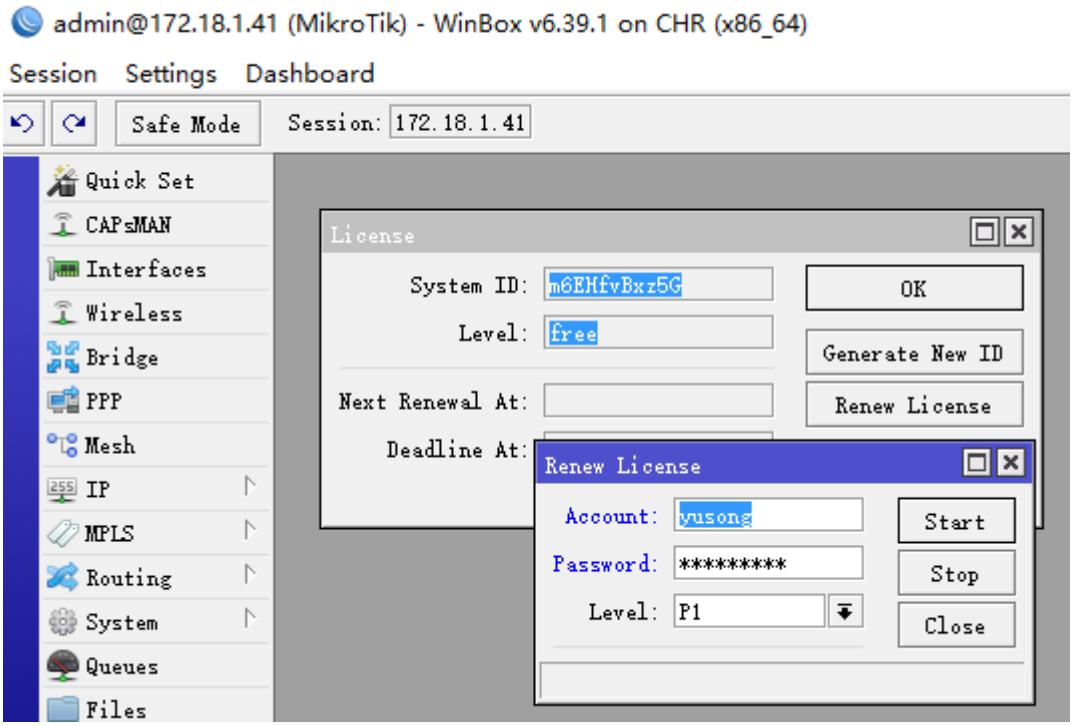
免费许可允许 CHR 无限期使用，但限制所有接口为 1Mbps，其他功能没有任何限制。使用该版本，在下载完成镜像文件，安装后即可使用。

60-day trial（60 天免费试用）

除之前的免费版本外，还允许体验 P1/P10/PU 等许可，时间为 60 天。申请 60 天免费许可，需要有 MikroTik 账号，在 www.mikrotik.com 注册帐号，注册成功后，可以根据 CHR 安装后生成的 ID 进行注册和购买许可。

账号申请后，登录 CHR 的 RouterOS，确保 RouterOS 连接互联网正常（DNS 配置正确），使用命令"**/system license renew**"，根据提示输入账号和密码进行注册。

下面通过 Winbox 方式注册，进入 system license 菜单，点击 Renew License，输入账号密码，和需要申请的等级



申请完成后，我的申请时间为 2017 年 5 月 8 日，下次更新时间（Nex Renewal）2017 年 6 月 7 日，到期时间（Deadline）7 月 7 日

60 天免费许可期限到后，在 License 菜单下的 Limited Upgrades 将会被勾选，即 RouterOS 将无法在升级，如果你计划购买选择的许可，你必须在 60 天内申请购买正式许可。如果 60 天内未购买，期限到后该 CHR 的 System ID 将从你的 MikroTik 账户中删除，该 CHR 的 System ID 将无法使用，必须重新安装一次 CHR，并重新申请购买。

32.3 购买许可

当 CHR 主机安装完成，将获得一个免费的许可，免费许可版本可以通过注册官方账号购买或者升级测试许可

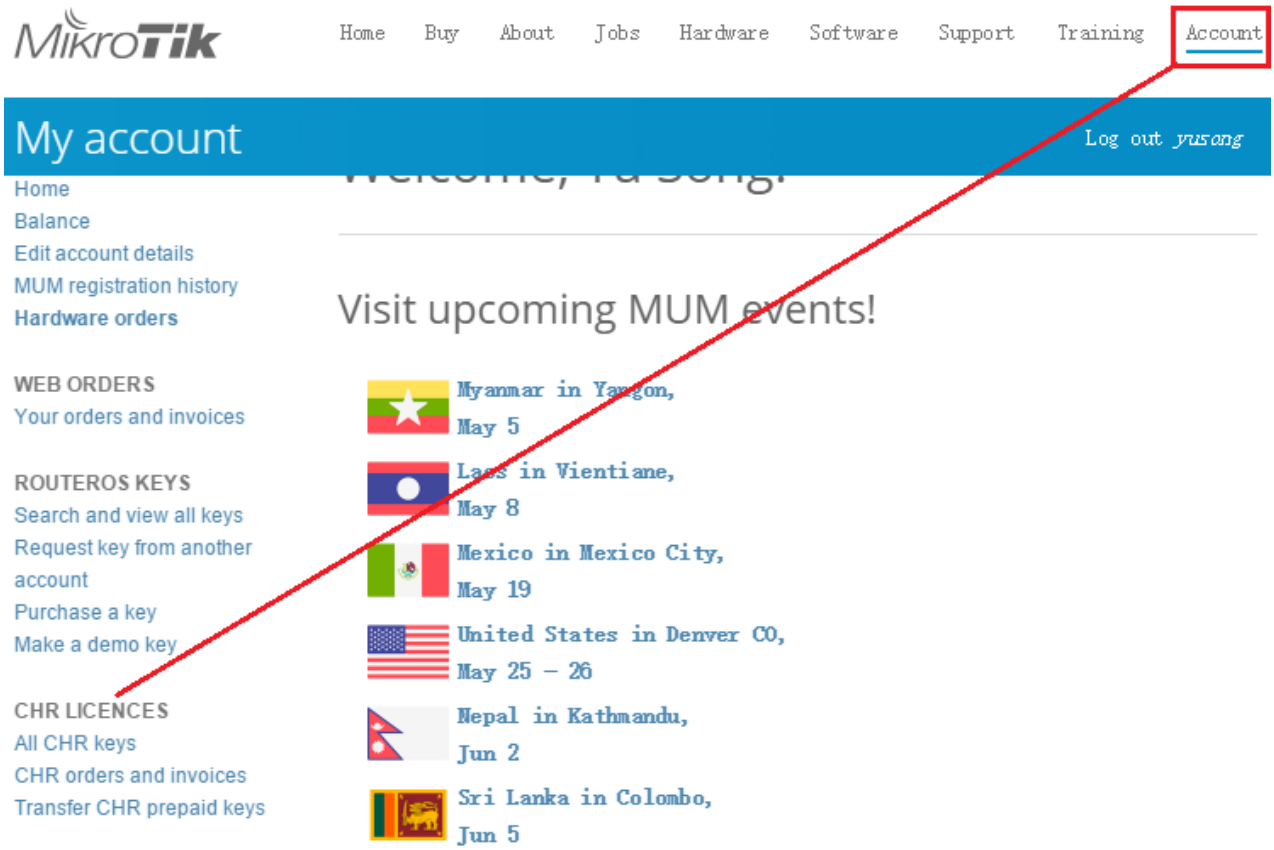
从免费许可升级到 P1 或更高版本

初次从免费试用版升级，首先注册 MikroTik 账号，保证 CHR 的 RouterOS 能连接到互联，然后进入 RouterOS 的 system-License 目录下输入 MikroTik 账号和密码（可以参考之前 60 天免费试用），使用命令行，获取 P1 许可 60 天免费试用，操作如下：

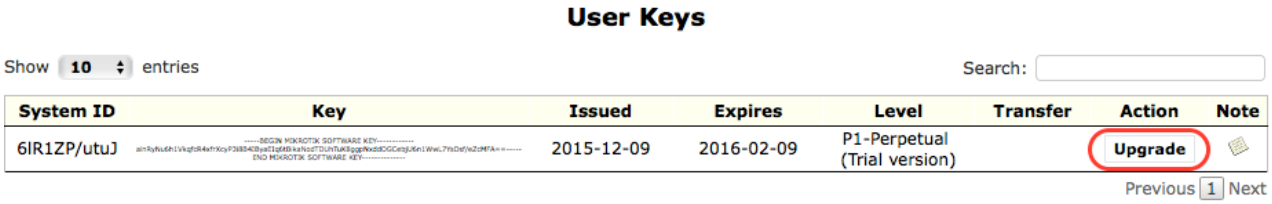
```
[admin@MikroTik] > /system license print
   system-id: 6lR1ZP/utuJ
         level: free

[admin@MikroTik] > /system license renew
account: mymikrotikcomaccount
password: **********************
level: p1
   status: done

[admin@MikroTik] > /system license print
            system-id: 6lR1ZP/utuJ
                  level: p1
   next-renewal-at: jan/10/2016 21:59:59
         deadline-at: feb/09/2016 21:59:59
```

这样通过 renew 命令获取了 P1 许可的 60 天免费使用，同时改 CHR 的系统 ID 将记录到你在 MikroTik 官方账号。

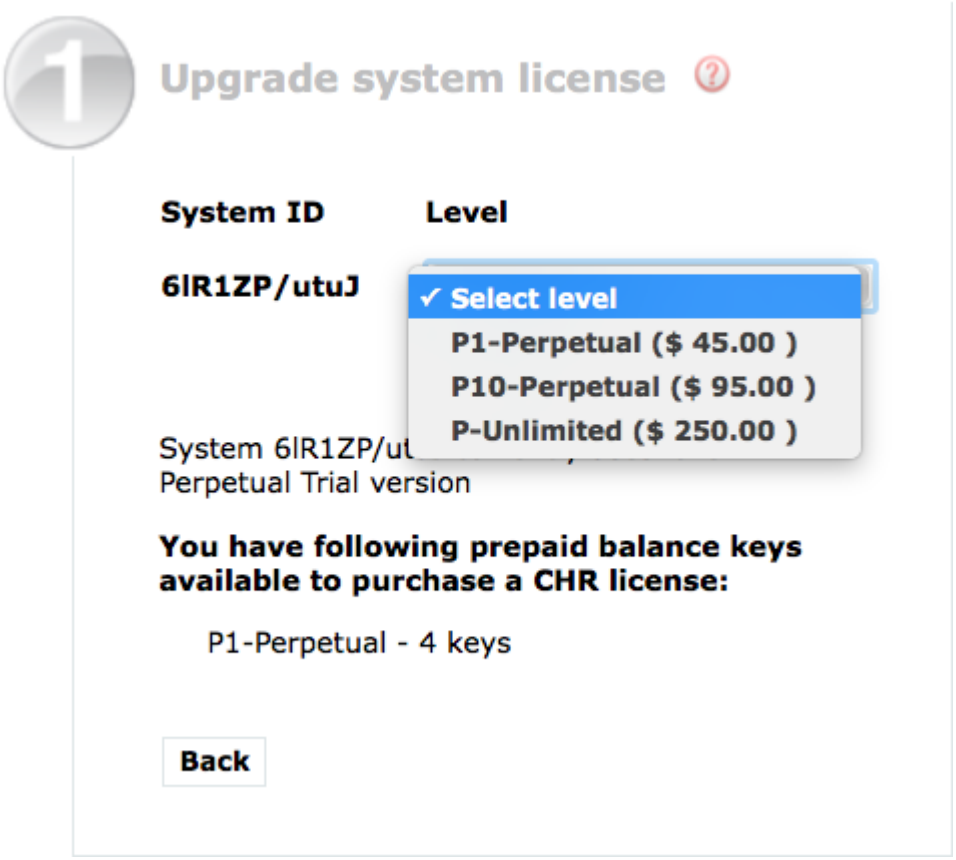
下一步是购买 P1 或者更高版本许可，首先进入 MikroTik.com account server ，登录成功后，选择 CHR LICENSE 下的 'All CHR keys'



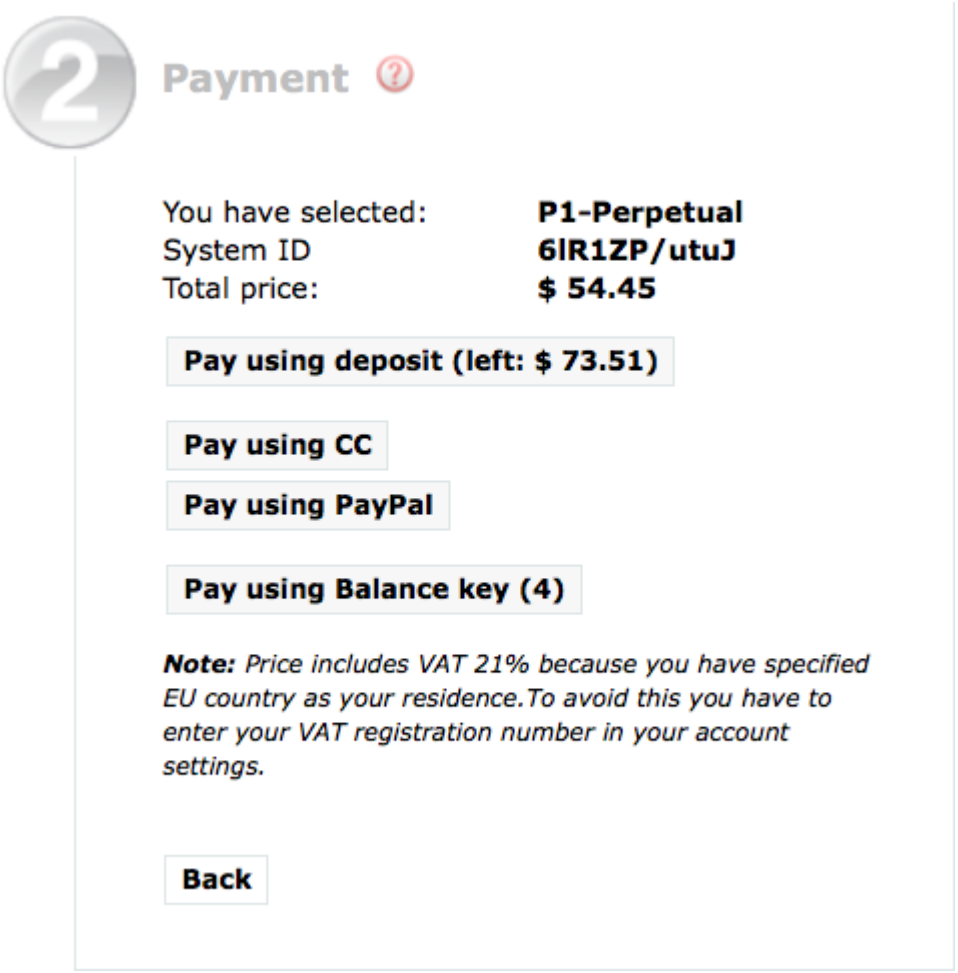
这里将出现你的 CHR 主机和许可信息列表



升级免费试用许可到付费许可，点击'Upgrade'，然后出现选择付费许可的等级和相应价格：


Upgrade system license
System ID
Level
6lR1ZP/utuJ
✓ Select level
P1-Perpetual ($ 45.00)
P10-Perpetual ($ 95.00)
P-Unlimited ($ 250.00)
System 6lR1ZP/ut
Perpetual Trial version
You have following prepaid balance keys available to purchase a CHR license:
P1-Perpetual - 4 keys
Back

选择付费方式：


Payment
You have selected:
P1-Perpetual
System ID
6lR1ZP/utuJ
Total price:
$ 54.45
Pay using deposit (left: $ 73.51)
Pay using CC
Pay using PayPal
Pay using Balance key (4)
Note: Price includes VAT 21% because you have specified EU country as your residence.To avoid this you have to enter your VAT registration number in your account settings.
Back

可以选择账户余额(deposit)，信用卡(CC)，PayPal 或使用帐户预存 key 付款(prepaid)支付。付费完后，新的许可通过网络注册到你的 CHR。

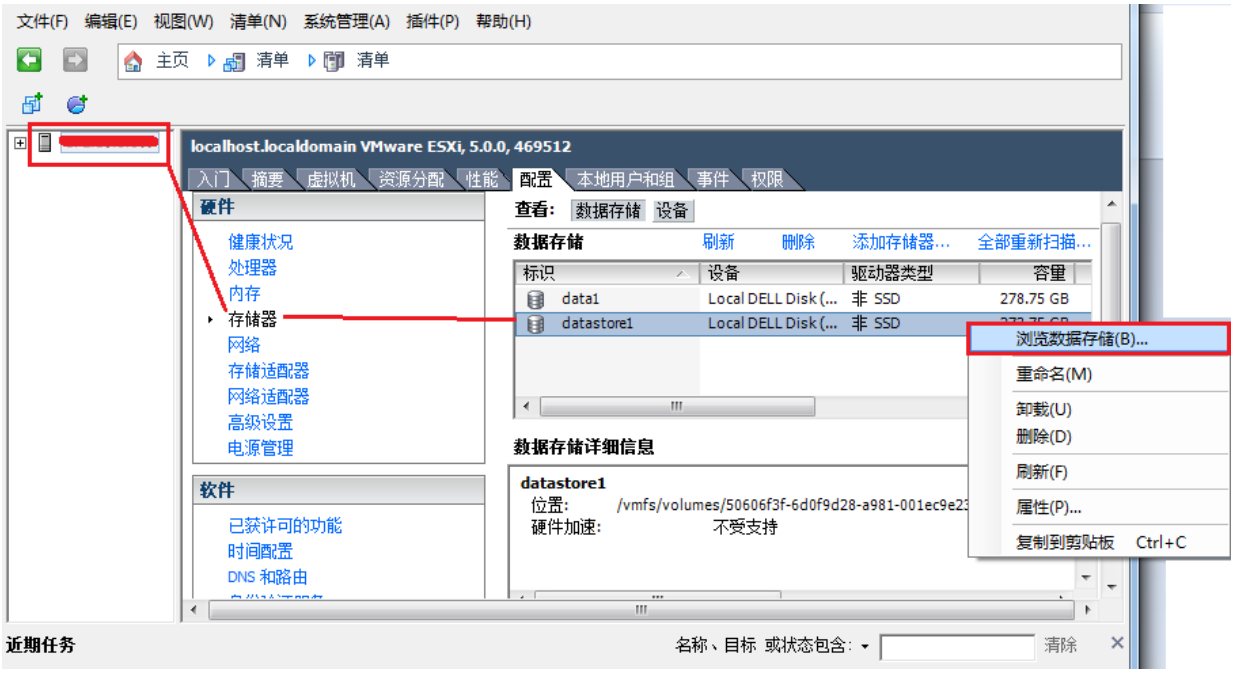
在'/system license' 目录下会显示 RouterOS 许可下一次更新时间（next-renewal-at），即申请 60 天免费许可后的一个月，并会尝试连接 licence.mikrotik.com 在 next-renewal-at 到期日后一小时内不断尝试连接官方许可服务器，如果你购买了付费许可，将会自动更新，当然要求保持网络连接畅通（DNS 必须正确配置）。如果 deadline-at 时间到后，仍未成功连接到服务器，或没有购买付费许可，RouterOS 会考虑 60 天试用许可到期，停止软件许可更新和许可升级，即该 CHR 主机许可生效。但该 CHR 仍然可以继续工作，只是恢复到免费版本，网络接口只有 1Mbps。

32.4 CHR 安装到 vmware esxi 平台

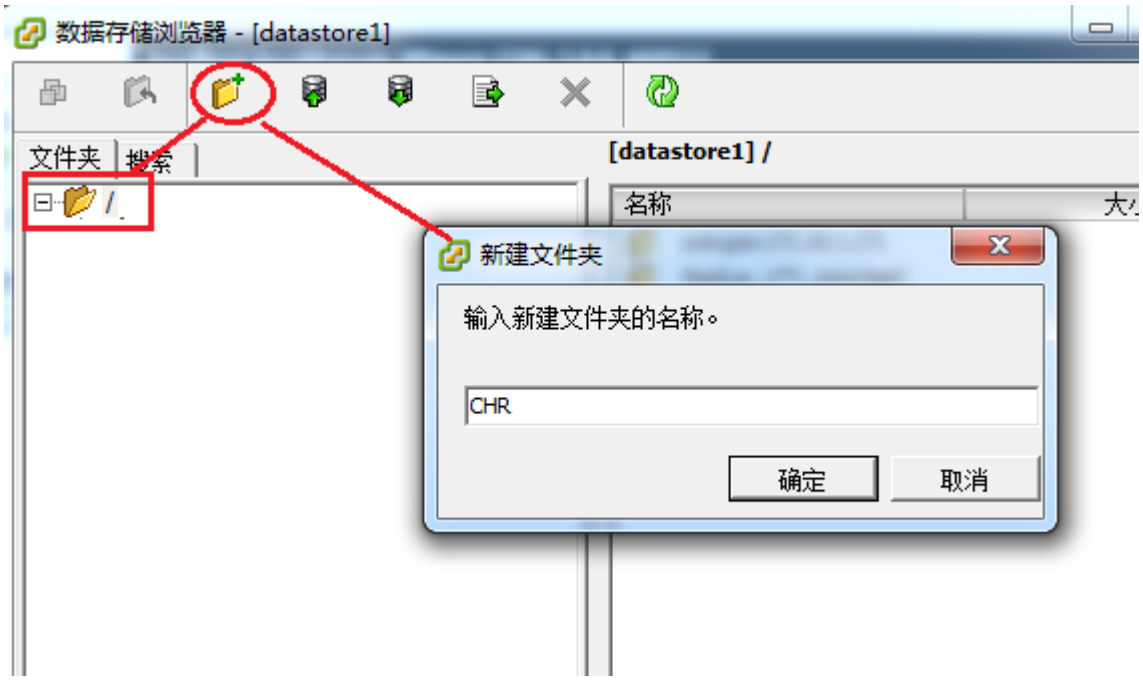
安装 CHR 实例通过 vmware esxi 5.0 作为演示（VM workstation 安装操作类似），首先我们需要从官网下载 wmdk 的镜像档 http://www.mikrotik.com/download/share/chr_6_31rc9.vmdk(此连结仅供参考)，CHR 文件下载请访问 www.mikrotik.com/download

Vmware esxi 6.5 以上版本安装可以参见官方文档

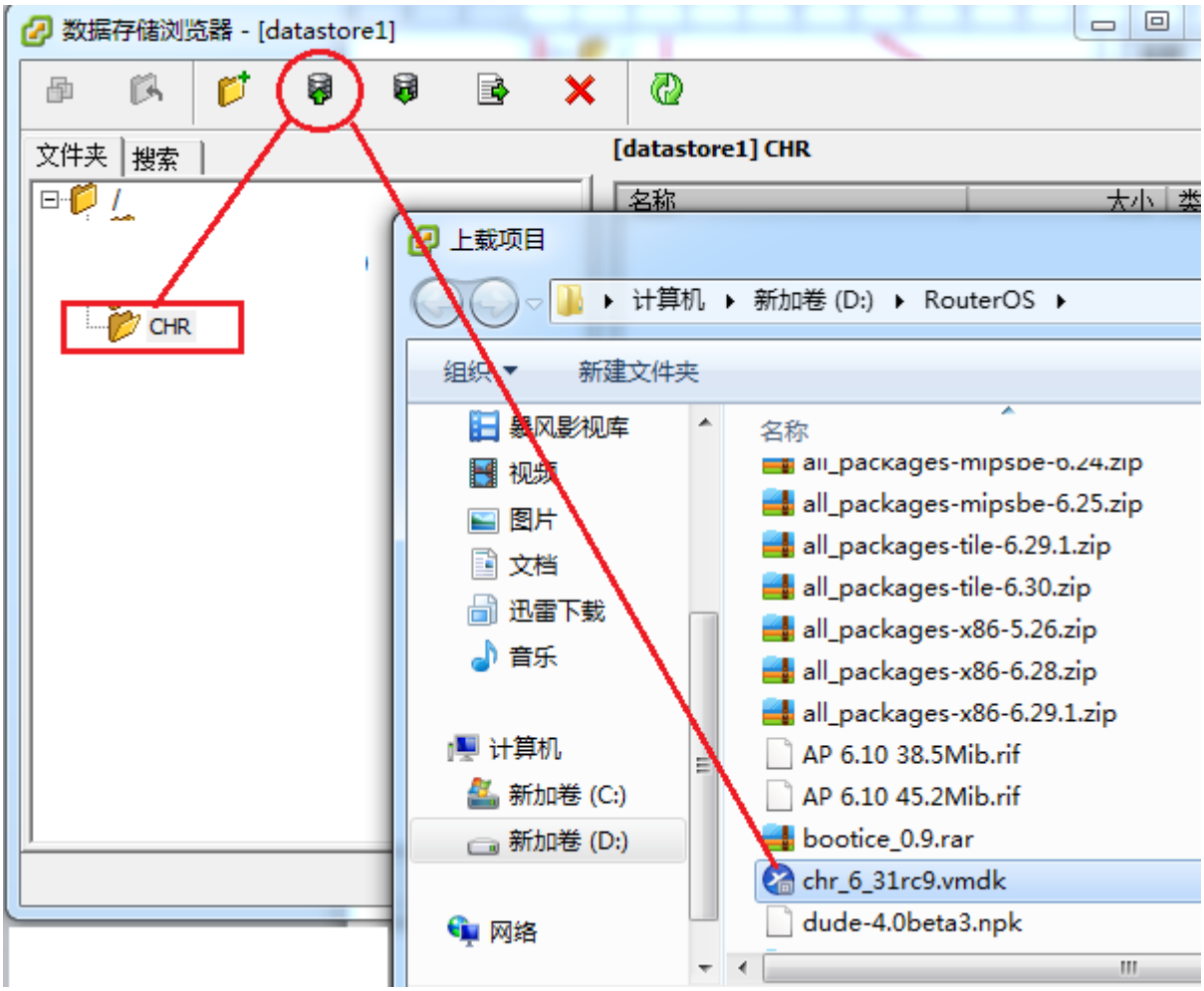
下载完成后，通过 vsphere client 登录 Vmware esxi，选择虚拟主机，并进入配置菜单，打开下面的内存，选择一块 data 数据盘，右键选择“浏览数据存储”



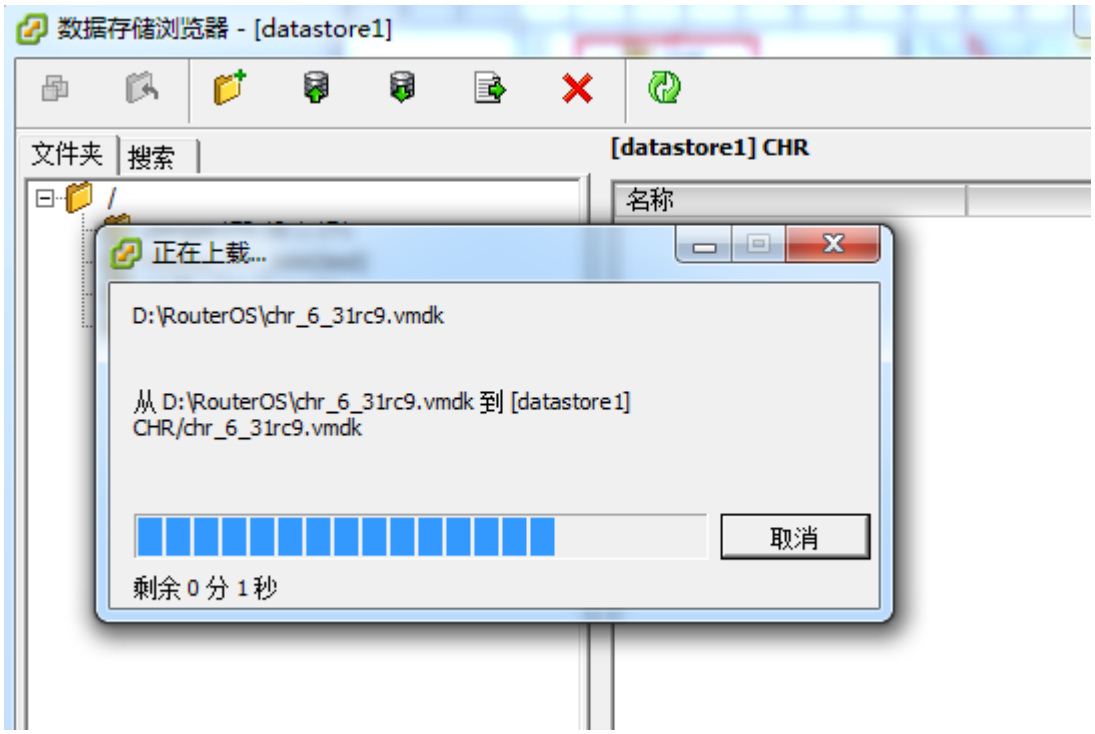
打开后，选择根目录，新建一个文件夹，取名 CHR



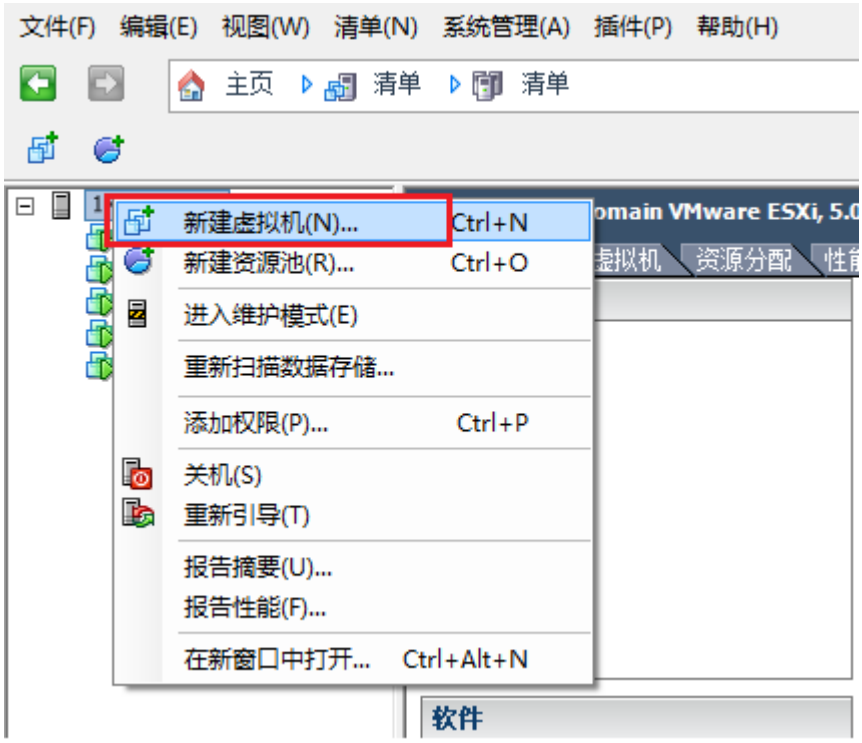
打开 CHR 文件夹，然后上传下载好的 chr_6_31rc9.vmdk 文件



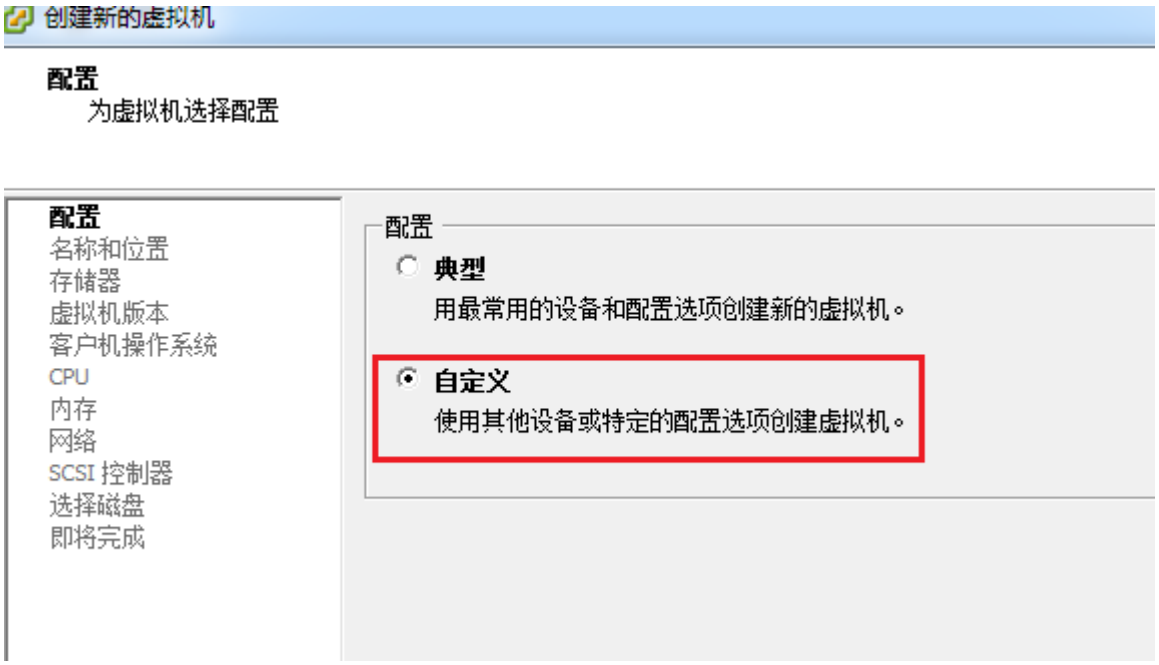
确认开始上传



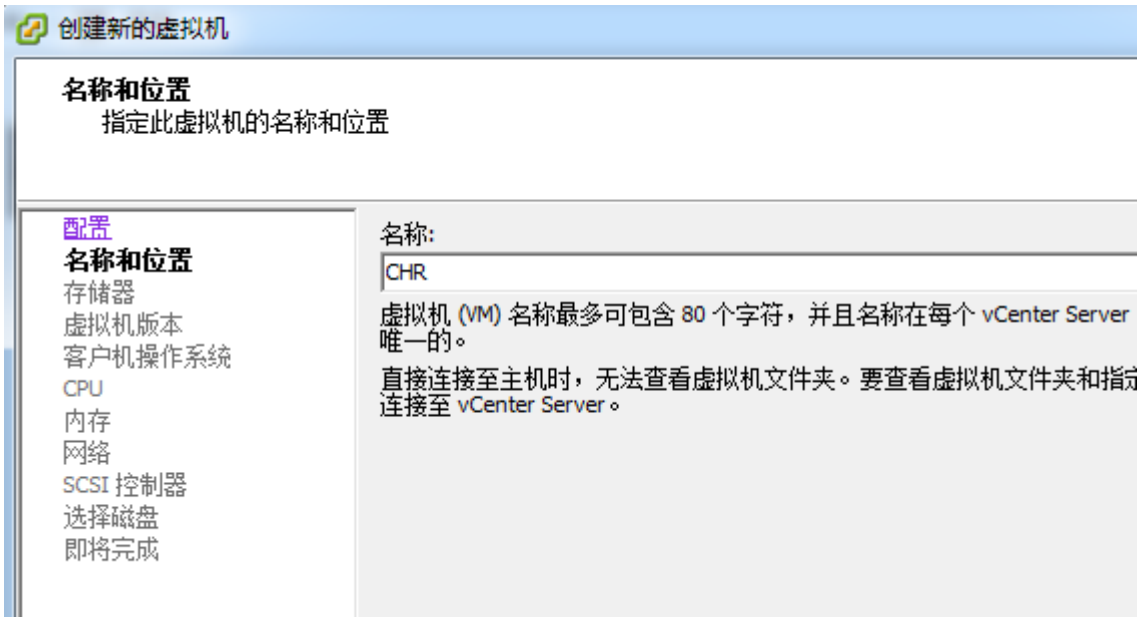
上传完成后，在根菜单选择新建虚拟机：



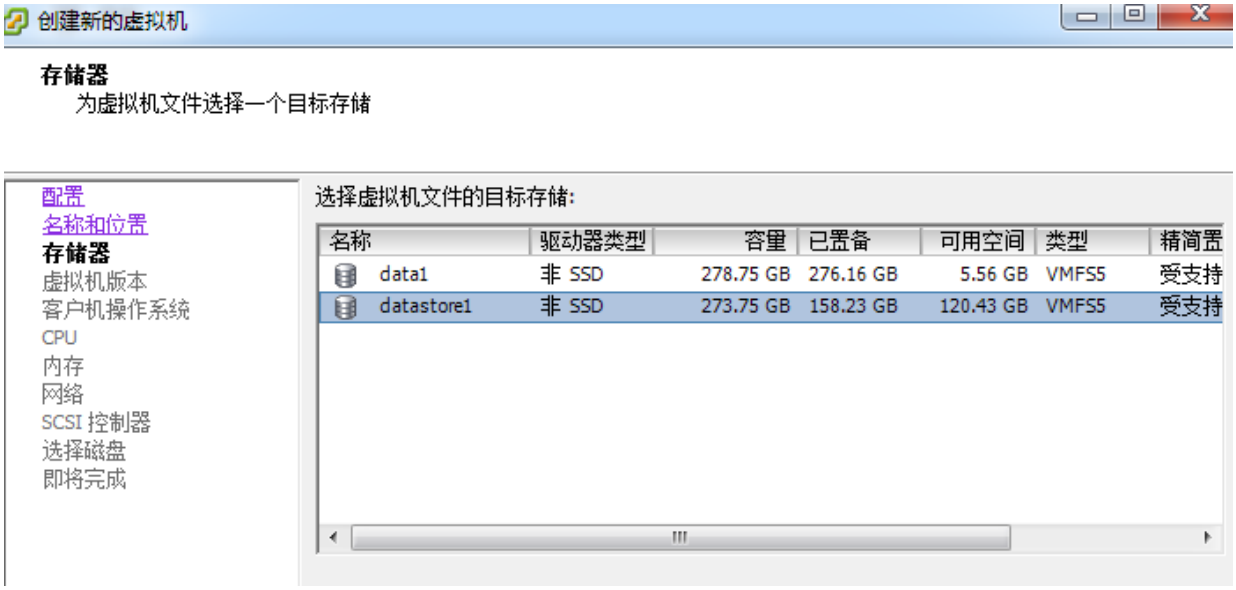
出现对话框后，选择自定义，然后下一步



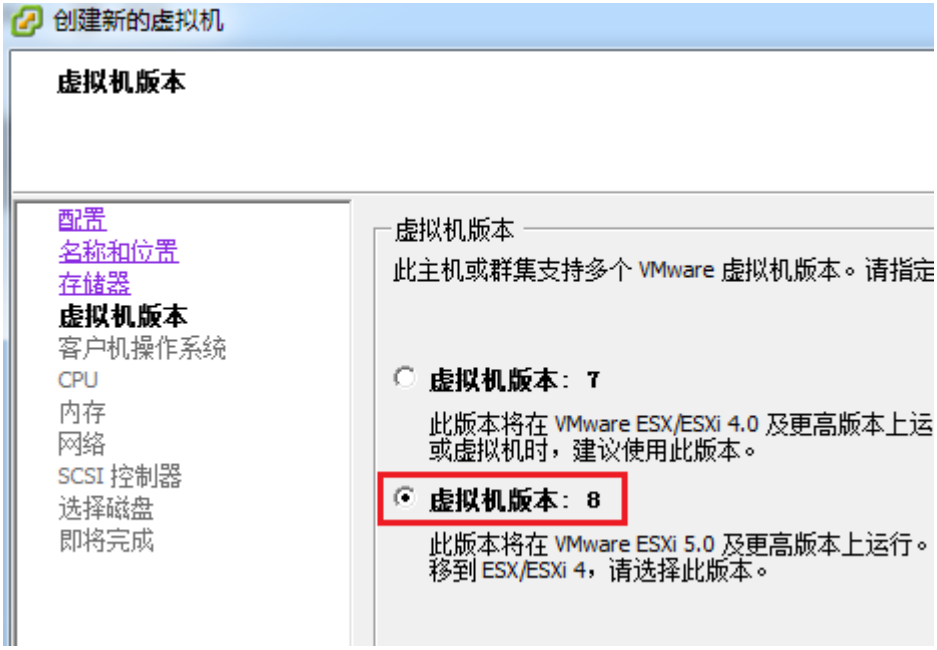
取名可以随意：



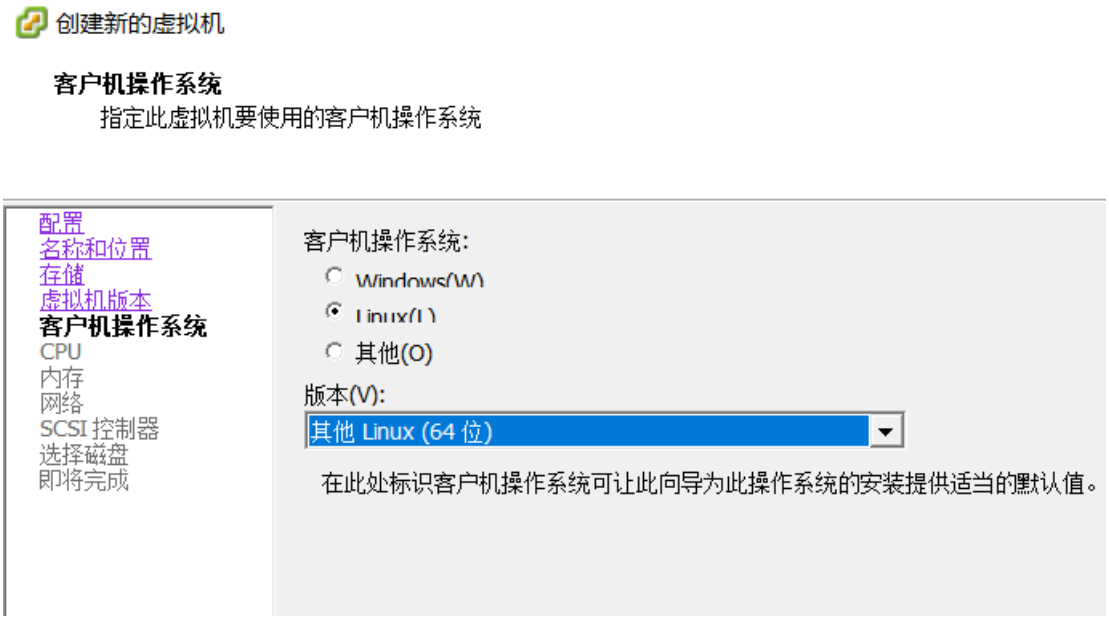
选择存储磁盘



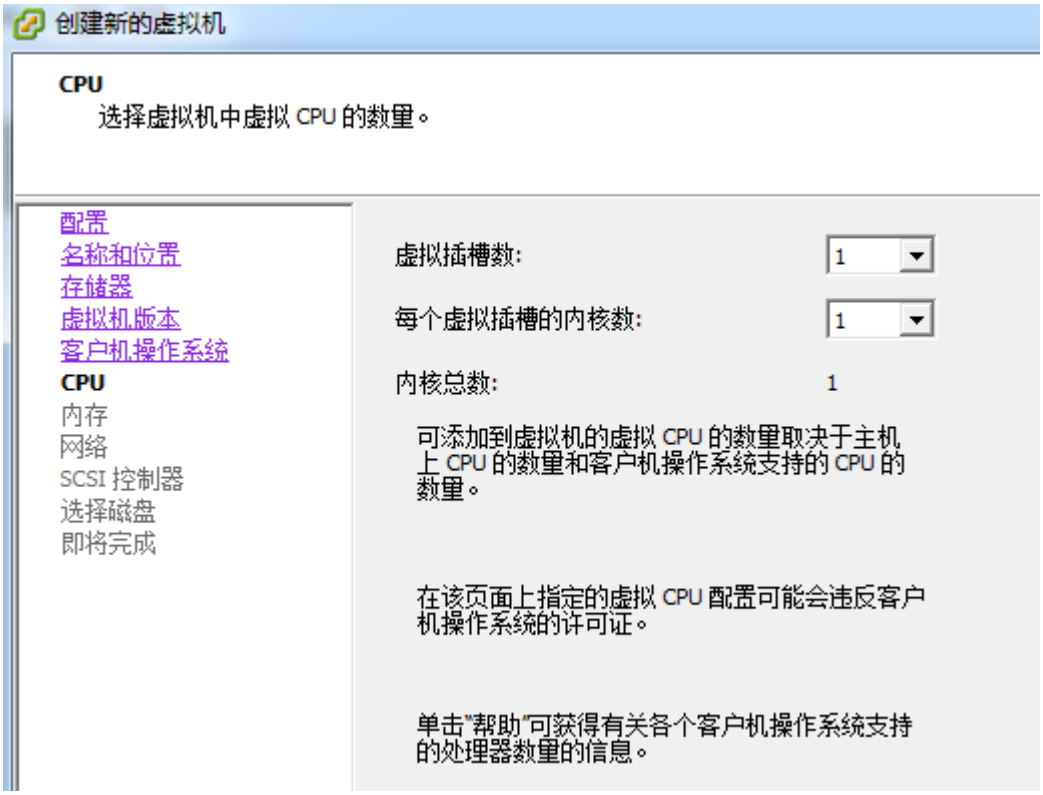
提示选择虚拟机版本，默认就好



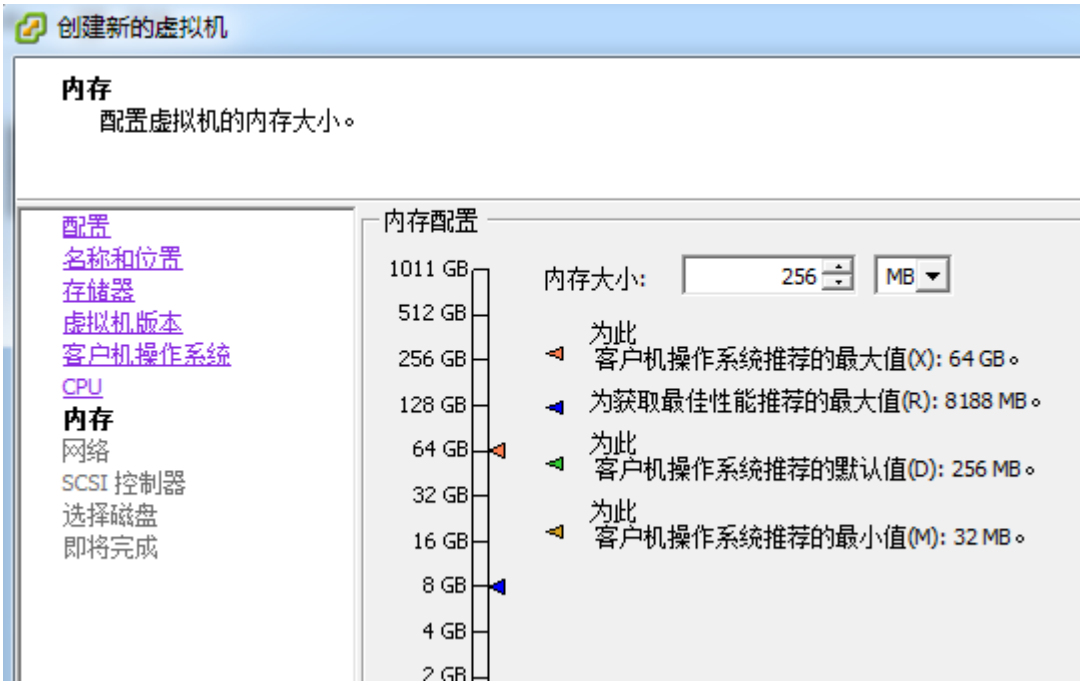
选择操作系统 Linux，版本为：其他 linux（64 位），



选择硬件配置，根据自己需要可以调整 CPU 数量

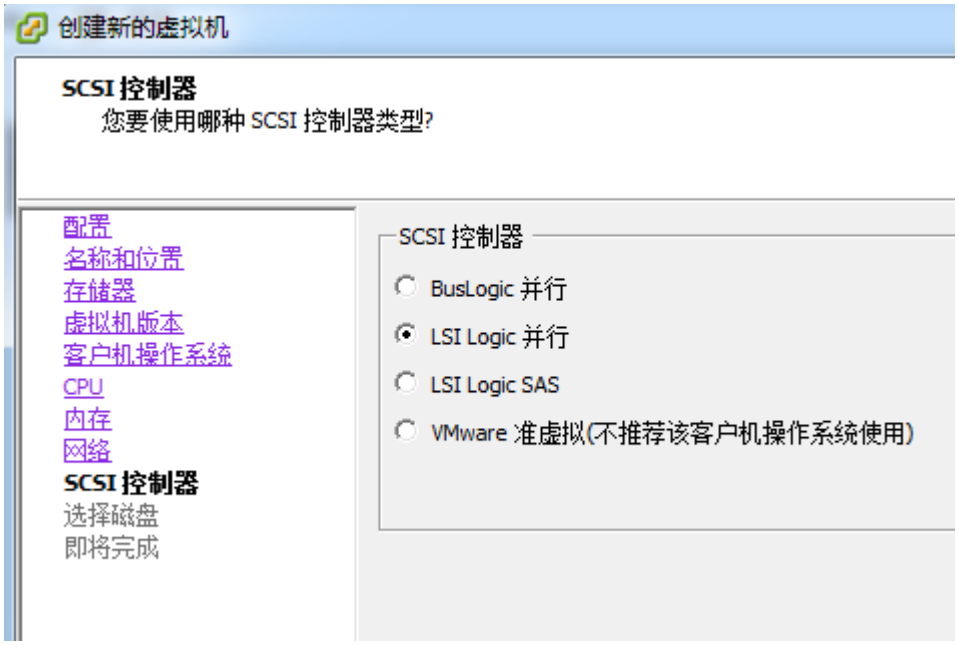


内存大小选择，如果实际应用要求高，可以设置更大内存



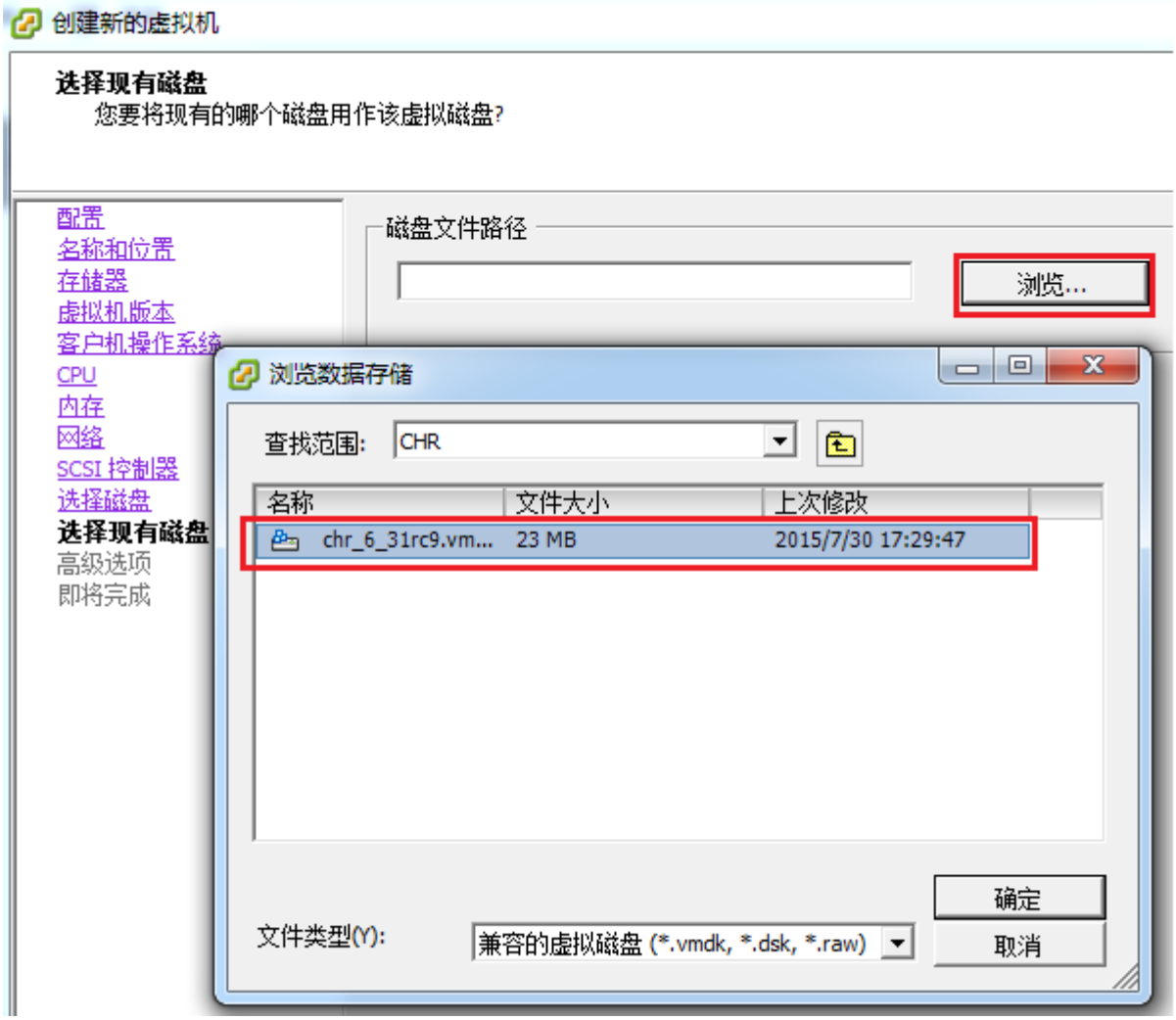
网络接口配置

下面是选择控制器



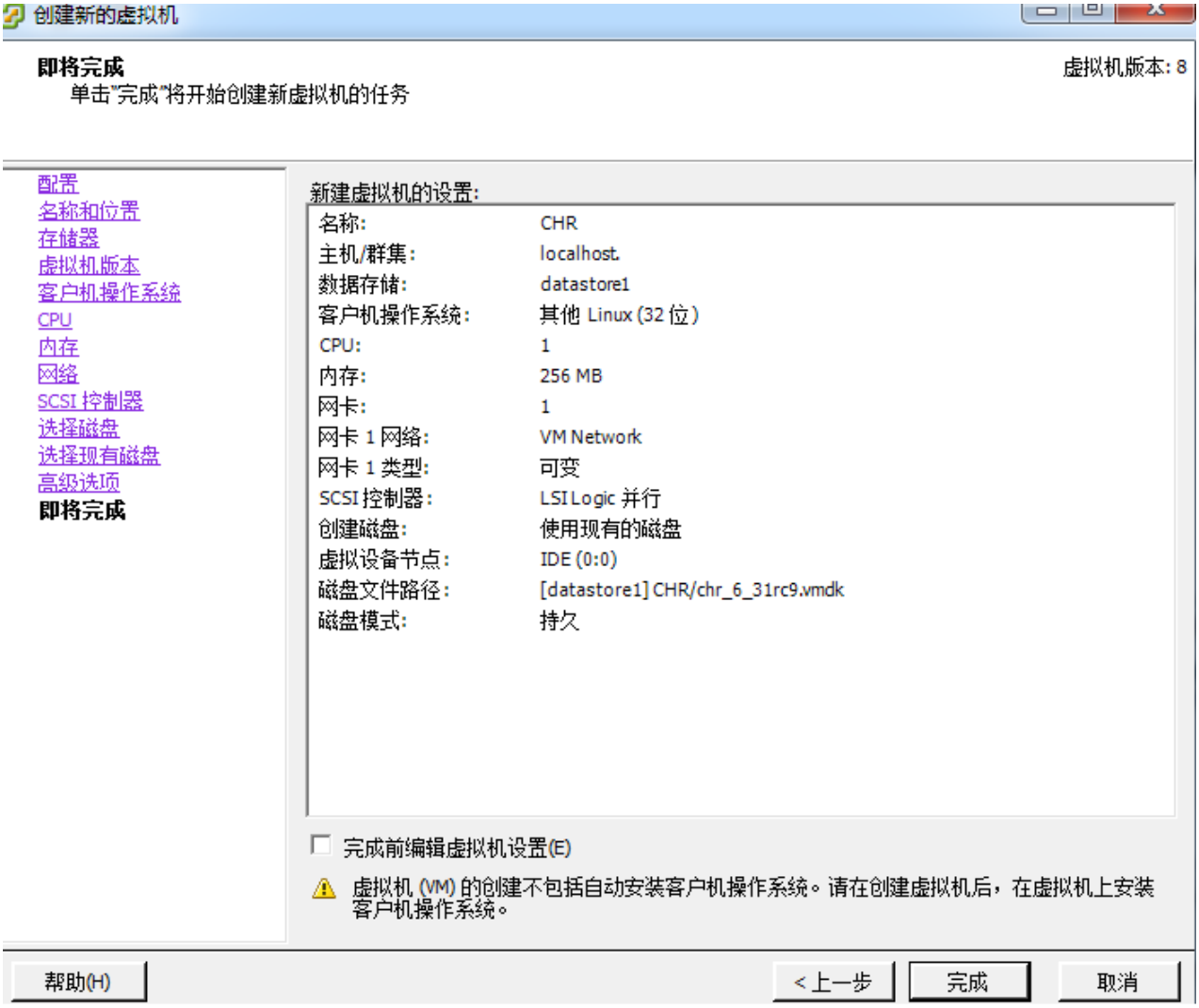
注意在选择磁盘的时候，我们需要选择“使用现有虚拟磁盘“

浏览磁盘文件路径，并加载我们的 CHR 文件

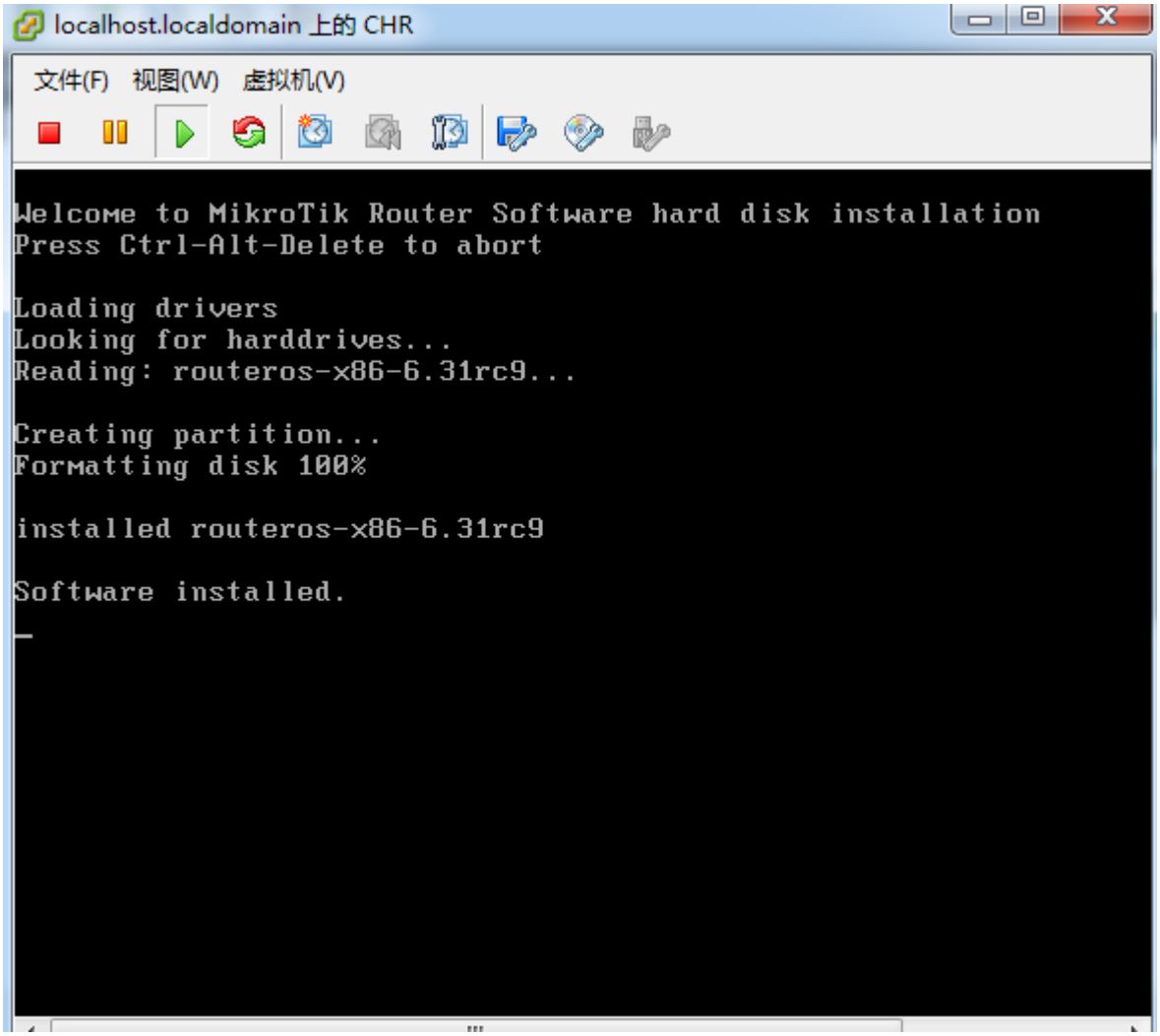


高级选项中，我们选择虚拟设备节点为 IDE(0:0)

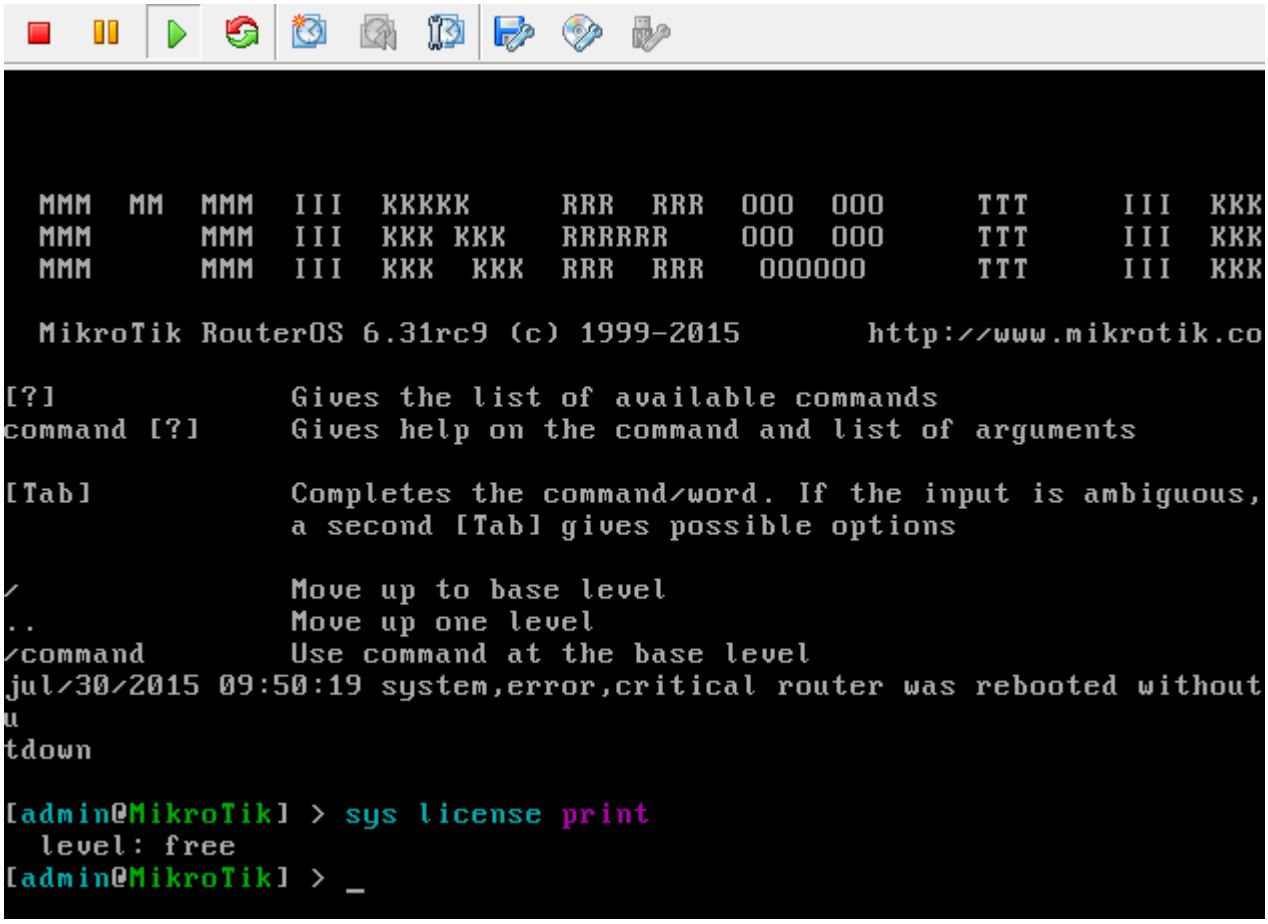
最后是创建完成



我们启动在 VM 的 CHR 虚拟机



启动完成后，进入接口，查看许可为 free



CHR 配置方式和其他平台的 RouterOS 完全一样

32.5 在 VPS 上安装 CHR

在 VPS 创建 CHR，在各种云主机上部署 RouterOS 用途非常广泛。我也是查看了很多网上资料，从中学习和实践，关于网上的一些配置，大多找到的都是获取 Linux 的 IP 地址和网关，通过 RouterOS 脚本写入到镜像，安装完成后能直接连接，但我觉得这部分没有必要，因为 VPS 都使用了 DHCP 服务分配 IP 地址，CHR 的 RouterOS 系统，安装完成后会自动创建 DHCP-client 获取 IP。

查看 VPS 配置

申请相应的 VPS（根据自己需求选择 VPS 节点），硬件配置单核 CPU，1G 内存、20G 以上存储空间足够，安装操作系统选择 CentOS7.x 或 Debian9.x 的 linux 系统都 ok

VPS 我使用 CentOS7 作为演示，通过 SSH 或者控制台检查网络是否正常，确保 DNS 已经配置正确

```
[root@myvps ~]# cat /etc/resolv.conf
; generated by /usr/sbin/dhclient-script
nameserver 108.61.10.10
[root@myvps ~]#
```

可以通过 ping 命令测试 DNS 解析是否正常

```
[root@myvps ~]# ping www.google.com
PING www.google.com (74.125.137.106) 56(84) bytes of data.
64 bytes from 74.125.137.106 (74.125.137.106): icmp_seq=1 ttl=110 time=6.40 ms
64 bytes from 74.125.137.106 (74.125.137.106): icmp_seq=2 ttl=110 time=6.41 ms
64 bytes from 74.125.137.106 (74.125.137.106): icmp_seq=3 ttl=110 time=6.45 ms
64 bytes from 74.125.137.106 (74.125.137.106): icmp_seq=4 ttl=110 time=6.54 ms
^C
— www.google.com ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 6.405/6.454/6.546/0.079 ms
[root@myvps ~]#
```

DNS 解析正常，才能从 mikrotik 下载 CHR 安装文件

查看硬盘名称，通过命令 fdisk -l

```
[root@myvps ~]# fdisk -l
Disk /dev/vda: 26.8 GB, 26843545600 bytes, 52428800 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x0009a079
 Device Boot      Start         End      Blocks   Id  System
/dev/vda1   *        2048    52428766    26213359+  83  Linux
[root@myvps ~]#
```

确认磁盘名称为/dev/vda，此次操作很关键，后续 dd 写入会用到。

使用 dd 命令安装 RouterOS

使用 wget 命令，下载 CHR 镜像文件（Raw disk image），

```
wget https://download.mikrotik.com/routeros/6.47.10/chr-6.47.10.img.zip -O chr.img.zip
```

解压文件：

```
gunzip -c chr.img.zip > chr.img
```

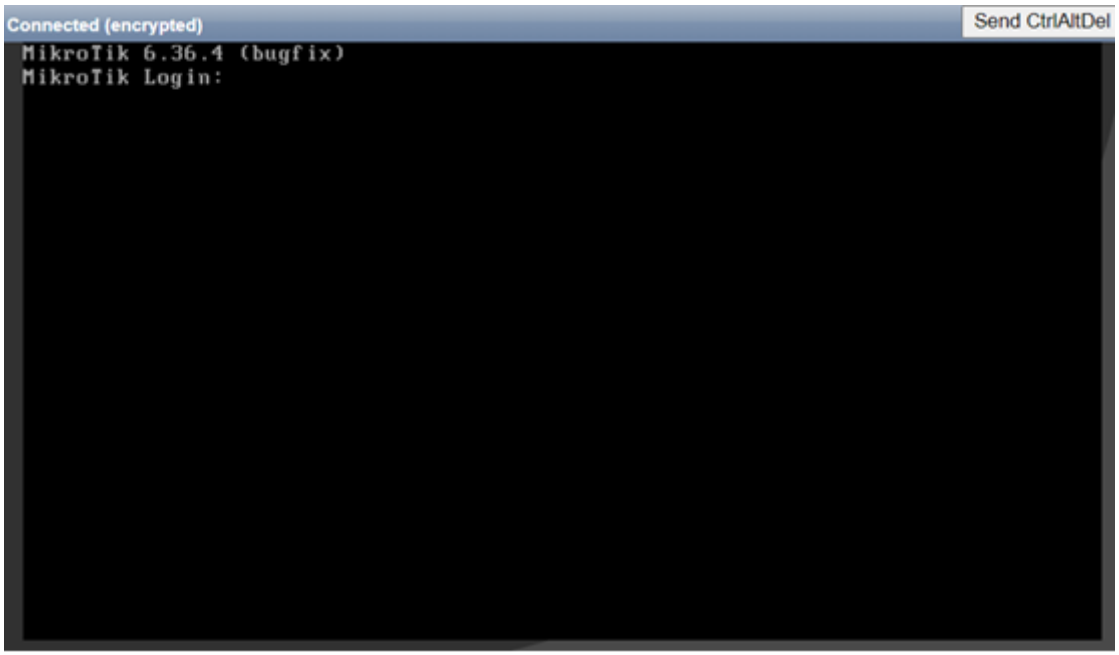
大部分的 VPS 都做了 dhcp 服务器，IP 地址绑定了 VPS。CHR 会创建 dhcp-client 配置，自动获取公网 IP 地址，最后使用 dd 命令将镜像写入 vda：

```
[root@localhost ~]# dd if=chr.img bs=1024 of=/dev/vda && \
[root@localhost ~]# reboot
```

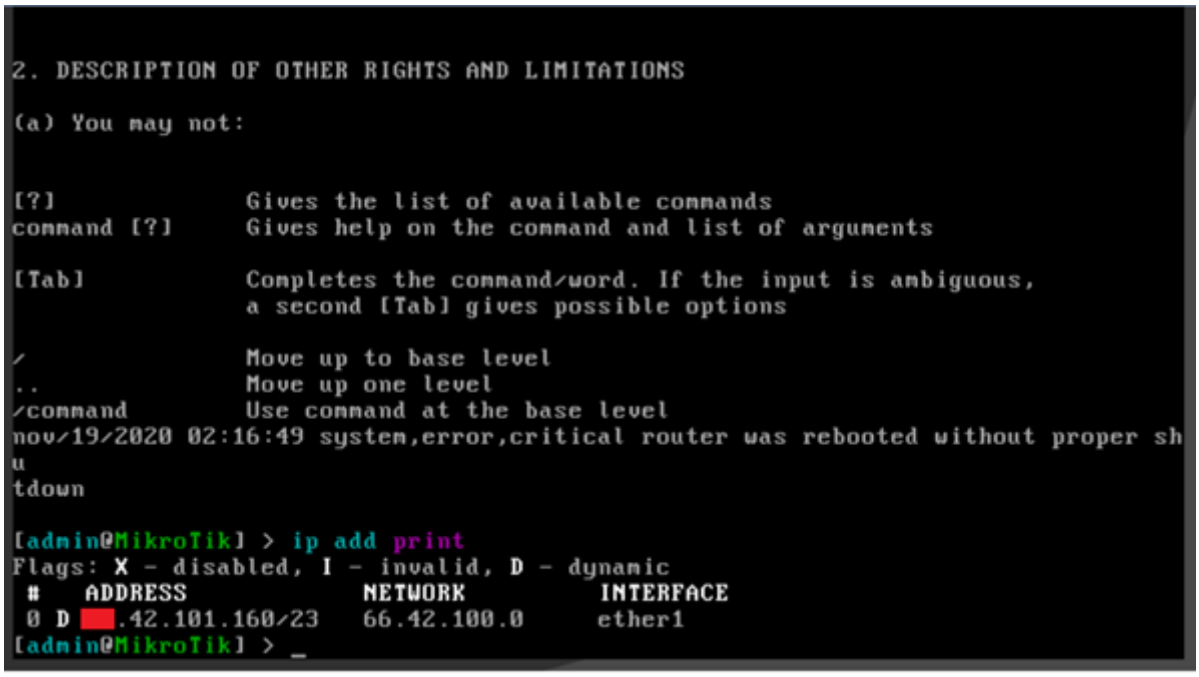
整个安装过程如下：

```
[root@localhost ~]# dd if=chr-6.47.10.img bs=1024 of=/dev/vda
65536+0 records in
65536+0 records out
67108864 bytes (67 MB) copied, 3.59435 s, 18.7 MB/s
[root@localhost ~]# reboot
```

执行到 reboot 后，如果 VPS 不自己重启，在控制台手动重启 VPS，下面是启动完成后的的登录界面，账号为 admin，密码为空



进入控制界面后，通过/ip add print 检查 ip 地址



检查网络没问题后，输入 password 命令修改 admin 密码，黑客对 VPS 扫描很厉害。

VPS 安装完后，很容易被黑，我之前久遇到过一次没有及时修改密码，黑的方式是，添加 RouterOS 计划任务，定时通过 fetch 下载脚本，import 导入脚本，脚本会开启 ip socket 服务，允许指定 IP 连接到 RouterOS 的 socket 接口，然后关闭掉 ip service 下的所有服务端口，使得远程无法管理 RouterOS，还好没有修改 admin 密码。

进入/system package upgrade 下，使用 check-for-updates 命令，检查最新版本

```
[admin@MikroTik] > system package update
[admin@MikroTik] /system package update> check-for-updates
          channel: current
  current-version: 6.36.4
   latest-version: 6.47.7
           status: New version is available

[admin@MikroTik] /system package update>
[admin@MikroTik] /system package update> _
```

使用 download 命令，下载更新 RouterOS 版本

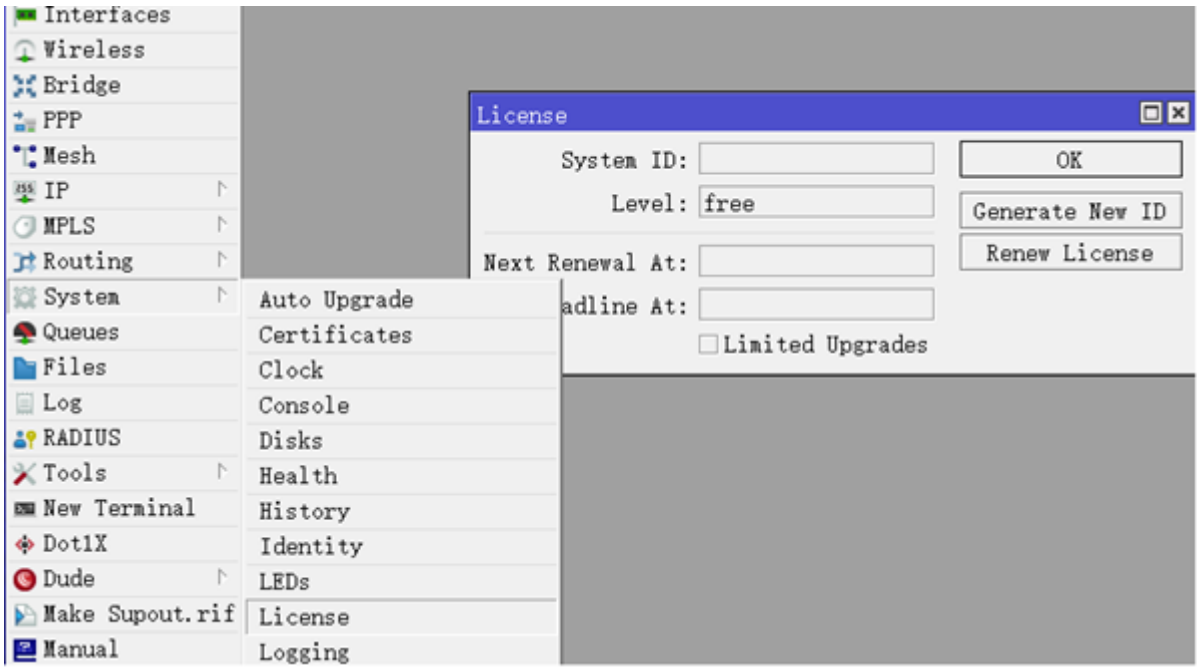
```
[admin@MikroTik] /system package update>
[admin@MikroTik] /system package update> download
          channel: current
  current-version: 6.36.4
   latest-version: 6.47.7
           status: Downloaded, please reboot router to upgrade it

[admin@MikroTik] /system package update>
```

然后使用/system reboot 命令重启

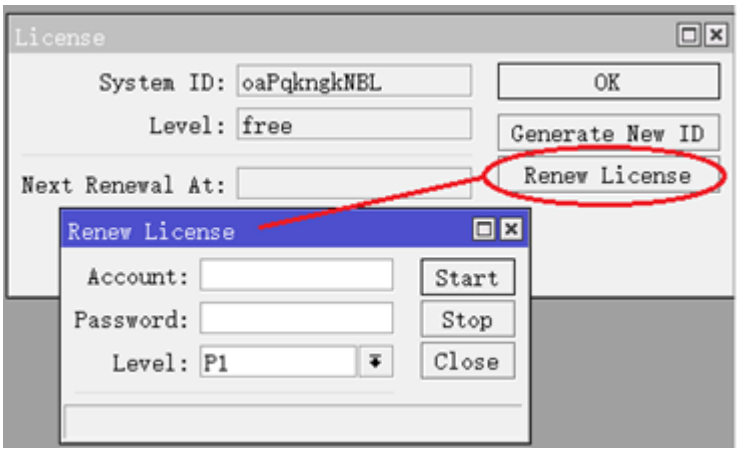
```
[admin@MikroTik] /system package update> /system  reboot
Reboot, yes? [y/N]:
_
```

重启完成后，使用 winbox 登录最新版本的 RouterOS 系统，完成配置，建议在/ip service 菜单下关闭或修改不需要的服务端口。

进入 winbox 配置完成后，可以申请 CHR 的许可，但需要先注册 MikroTik 账号，进入 system license 查看 ID



点击 renew license，输入 MikroTik 注册的账号密码，申请 P1 许可，可以免费使用 60 天



60 天内都可以购买正式版，P1 等级就够用了，价格 45USD，关于 CHR 的许可注册购买流程可以参见 CHR（Cloud Host Router）许可注册购买流程

关于 VPS 隧道的应用，可以参考：RouterOS VPS 隧道用于非 CNLIST 连接

32.6 KVM

Kernel-based Virtual Machine (KVM)是 Linux 下 x86 硬件平台上的全功能虚拟化解决方案，包含一个可加载的内核模块 kvm.ko 提供和虚拟化核心架构和处理器规范模块。该功能被加入到 RouterOS 系统中，能实现多个操作系统在 RouterOS 运行。

对于 RouterOS 的 KVM 功能，通常是运行其他操作系统，实现 RouterOS 无法支持的功能，例如入侵检测(SNORT)、Asterisk、Web 服务器或 Radius 数据库等。还可以用作测试环境，如可以在一台 x86 的 RouterOS 设备上创建与真实网络非常相似的虚拟网络，并设置和测试需要运行的 RouterOS 网络。

基本要求

KVM 需要 Intel VT-x 或 AMD-V CPU 虚拟化支持，请确认在 BIOS 下已经启用，CPU 支持情况请自行查询厂商资料，每个客户机至少需要 16MB RAM 和映像文件所需的足够存储空间。一旦创建了镜像文件，它的大小不能增加。

在 RouterOS KVM 支持需安装并启用 kvm 功能包，使用 KVM 前，最好确保升级到 RouterOS 64bit 版本，因为 v6 版本几乎屏蔽了 64bit，只能运行 2Gbit 内存，避免内存过少无法支持大内存的虚拟机运行。如果你当前的 RouterOS 不支持大于 2Gbit 内存，就需要开启 x86 64bit，开启 64bit 支持需要将 RouterOS 版本调整到 v6.31，因为只有 v6.31 才能开启 64bit 功能，然后在/system resource hardware 开启 x86_64，再升级到最新的 v6 版本。

所有 KVM 相关的配置在/kvm 菜单下操作，KVM 客户端获得的目录配置如下：

- */kvm* – KVM 主配置菜单
- */kvm interface* – KVM 接口配置菜单
- */interface virtual-ethernet* – KVM 接口衔接的虚拟主机

创建 KVM RouterOS 虚拟机

在创建 KVM 虚拟机之前，需准备映像文件。RouterOS 内置了一些命令，可以在无需外部工具的情况下轻松创建和修改 RouterOS 映像。

```
/kvm make-routeros-image file-name=ros1.img file-size=128
```

创建 RouterOS 磁盘映像后，可以使用该镜像，添加一个 KVM 的 RouterOS 虚拟机。

```
/kvm add name=ROS memory=128MiB cpu-count=2 disabled=no disk-images=hda:ros1.img \
    initrd="" kernel="" kernel-cmdline="console=ttyS0"
```

在 initrd 和 kernel 属性为空，这意味着使用了主机 kernel 和 initrd。例如，要添加不支持 SMP 的 guest，我们可以明确 initrd 和 kernel 的设置：

```
/kvm add name=ROS memory=128MiB cpu-count=2 disabled=no disk-images=hda:ros1.img \
    initrd=/boot/initrd.rgz kernel=/boot/vmlinuz kernel-cmdline="console=ttyS0"
```

注意:如果 RouterOS 主机和虚拟机运行的是不同的 RouterOS 版本，initrd 和 kernel 属性为空是比较危险的。如果这些值为空，可能导致不同与主机版本的 RouterOS 会无法运行。

KVM Guest 创建完成后，不会自动启动，需手动启动，命令如下：

```
[admin@proxy] /kvm> start ROS;
[admin@proxy] /kvm> print
Flags: X - disabled
 0    name="ROS" cpu-count=2 memory=128MiB disk-images=hda:ros1.img kernel="/boot/vmlinuz"
       kernel-cmdline="" initrd="/boot/initrd.rgz" vnc-server=0.0.0.0:0 snapshot=no   state=running
[admin@proxy] /kvm>
```

创建接口

通过前面的实例创建了 RouterOS 虚拟机，下面通过命令为虚拟路由器添加一个接口：

```
[admin@proxy] /kvm interface> add virtual-machine=ROS type=dynamic
[admin@proxy] /kvm interface> print
Flags: X - disabled, A - active
 #    VIRTUAL-MACHINE          INTERFACE                TYPE          VM-MAC-ADDRESS
 0    ROS                                               dynamic       02:D9:52:31:11:CC
[admin@proxy] /kvm interface>
```

完成虚拟主机接口添加后，在 RouterOS 主机 virtual-ethernet 下可以看到动态的接口生成：

```
[admin@proxy] /interface virtual-ethernet> print
Flags: D - dynamic, X - disabled, R - running
 #        NAME                                MTU       ARP               MAC-ADDRESS
 0 D R tap1                                   1500      enabled       02:3F:9F:AE:10:34
[admin@proxy] /interface virtual-ethernet>
```

警告：只有在 KVM 虚拟机关闭、停止或禁用时才添加和删除接口。对正在运行的虚拟机进行更改可能会导致主机系统崩溃。

如果创建虚拟接口时不指定 MAC 地址，则自动生成。生成的 MAC 地址格式为 02:XX:XX:XX:XX:XX。对于静态接口，这个地址不会在客户端使用期间发生变化，而动态接口会在每次创建动态接口时发生变化。

连接虚拟机

当前有两种方法连接到 KVM 客户端：

- virtual console;
- vnc.

Console

Console 连接方式：

```
[admin@proxy] /kvm> console ROS
```

进入 ROS 虚拟机后，可以通过相同的 RouterOS 命令查看接口情况：

```
[admin@mr0] > interface print
  Flags: D - dynamic, X - disabled, R - running, S - slave
  #        NAME                                                TYPE                MTU
  0   R    ether1                                              ether               1500
```

如果需要断开虚拟即的 console 控制台，通过 **CTRL + A** ，并输入 **Q** 退回到 RouterOS 主机控制台（如果使用 minicom，连续按下 **CTRL + A** 两次）：

```
[admin@MikroTik] >
```

```
[Q - quit connection]          [B - send break]
[A - send Ctrl-A prefix]     [R - autoconfigure rate]

  Q

  Welcome back!
```

VNC

在连接到 VNC 客户端之前，客户端需要进行一些配置更改，首先查看客户端配置，如果没有开启 VNC，需要关闭在运行的客户端.

```
[admin@proxy] /kvm> print
Flags: X - disabled
 0     name="ROS" cpu-count=2 memory=128MiB disk-images=hda:ros1.img kernel="/boot/vmlinuz"
         kernel-cmdline="" initrd="/boot/initrd.rgz" vnc-server=0.0.0.0:0 snapshot=no state=running
[admin@proxy] /kvm> shut-down 0
```

然后设置 vnc-server 地址

```
[admin@proxy] /kvm> set 0 vnc-server=10.5.100.99:1
[admin@proxy] /kvm> start 0
[admin@proxy] /kvm> print
Flags: X - disabled
 0     name="ROS" cpu-count=2 memory=128MiB disk-images=hda:ros1.img kernel="/boot/vmlinuz"
         kernel-cmdline="" initrd="/boot/initrd.rgz" vnc-server=10.5.100.99:1 snapshot=no
         state=running
[admin@proxy] /kvm>
```

首先确保配置的 VNC 地址在网络中可以访问，然后在地址后配置屏幕编号，连接配置如下：

```
mrz@bumba:/$ vncviewer 10.5.100.99:1
```

配置虚拟网络实例

当前，已经可以看到 RouterOS 主机的 virtual interface 下 tap1 接口，以及在虚拟客户端的 interface 下有 ether1 接口。 所以现在可以配置 IP 地址在两个接口上，建立内外网络。也可以在 RouterOS 主机上创建一个 Bridge 桥接，将虚拟网卡和物理网卡打通，连接到外部的网络。

下面的实例，通过创建 3 台虚拟 RouterOS 路由器，并将 3 台虚拟 RouterOS 连接到一个广播域中.首先创建镜像 ：

```
/kvm
make-routeros-image file-name=R1.img file-size=64
make-routeros-image file-name=R2.img file-size=64
```

```
make-routeros-image file-name=R3.img file-size=64

add name=R1 disk-image=hda:R1.img
add name=R2 disk-image=hda:R2.img
add name=R3 disk-image=hda:R3.img
```

在 RouterOS 主机创建 Bridge 桥接，取名 kvm_bridge，

```
/interface bridge
add name=kvm_bridge
```

创建 3 台虚拟路由器的接口，并自动加入到 kvm_bridge 的桥接：

```
/kvm interface
add virtual-machine=R1 type=dynamic dynamic-bridge=kvm_bridge
add virtual-machine=R2 type=dynamic dynamic-bridge=kvm_bridge
add virtual-machine=R3 type=dynamic dynamic-bridge=kvm_bridge
```

启动 3 台虚拟机，并确认接口创建成功

```
[admin@proxy] /kvm> start [find]

[admin@proxy] > /interface virtual-ethernet print
Flags: D - dynamic, X - disabled, R - running
 #      NAME                                MTU   ARP          MAC-ADDRESS
 0 D R tap2                                1500  enabled      02:20:94:67:D6:D5
 1 D R tap3                                1500  enabled      02:95:EE:EA:43:FF
 2 D R tap4                                1500  enabled      02:05:7E:4B:86:F9
[admin@proxy] > /interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic
 #     INTERFACE                 BRIDGE              PRIORITY PATH-COST  HORIZON
 0   D tap2                      kvm_bridge          0x80     10         none
 1   D tap3                      kvm_bridge          0x80     10         none
 2   D tap4                      kvm_bridge          0x80     10         none
[admin@proxy] >
```

现在通过控制台连接到每一台虚拟机，并设置每台的 IP 在相同网段，首先登录 R1，，配置 IP 地址 192.168.1.1/24：

```
[admin@proxy] > /kvm console R1

[Ctrl-A is the prefix key]

MikroTik 5.0rc8
MikroTik Login: admin
Password:
```

```
[admin@MikroTik] > /ip address add address=192.168.1.1/24 interface=ether1
```

登录 R2，配置 IP 地址 192.168.1.2/24

```
[admin@proxy] > /kvm console R2

[Ctrl-A is the prefix key]

MikroTik 5.0rc8
MikroTik Login: admin
Password:

[admin@MikroTik] > /ip address add address=192.168.1.2/24 interface=ether1
```

R3，配置 IP 地址 192.168.1.3/24

```
[admin@proxy] > /kvm console R1

[Ctrl-A is the prefix key]

MikroTik 5.0rc8
MikroTik Login: admin
Password:

[admin@MikroTik] > /ip address add address=192.168.1.3/24 interface=ether1
[admin@MikroTik] > /ping 192.168.1.1
HOST                                     SIZE  TTL TIME  STATUS
192.168.1.1                              56     64  11ms
192.168.1.1                              56     64  2ms
    sent=2 received=2 packet-loss=0% min-rtt=2ms avg-rtt=6ms max-rtt=11ms

[admin@MikroTik] > /ping 192.168.1.2
HOST                                     SIZE  TTL TIME  STATUS
192.168.1.2                              56     64  12ms
    sent=1 received=1 packet-loss=0% min-rtt=12ms avg-rtt=12ms max-rtt=12ms
```

删除 KVM 虚拟机

KVM 虚拟机配置在 RouterOS 组成包括：kvm, virtual-ethernet, /kvm interface 和镜像文件(/file)。如果镜像文件被删除，但 KVM 中的虚拟机仍然可以运行，直到管理员关闭或禁用虚拟机，此时运行文件将被删除，存储空间才会释放给 RouterOS 主机。

KVM 也可以安装其他系统的镜像文件，包括 CentOS、Ubuntu、Debian 和 Windows 等，上传到/file 目录下，并进行相应的安装，请确保你的存储空间储够大。

32.6 MetaRouter 介绍

MetaRouter 是 RouterOS 从 4.0beta1 和 3.21 版本开始新增加的功能，当前 MetaRouter,只能用于 RB400 系列，用于创建虚拟机，在以后会有更多的硬件平台增加此功能。每一个 Metarouter 是使用设备相同的资源，建立独立的 RouterOS 系统。每一个 Metarouter 至少需要 16M 的 RAM。16M 是绝对最小的值，建议为每一个 Metarouter 使用更大的 RAM。

当前可以创建 8 个 Metarouter 虚拟机，将来新的版本会增加到 16 个。在主设备上，你可以创建 8 个虚拟接口连接到 MetaRouter，唯一可以增加接口的方式只能通过 VLAN。现在 MetaRouter 虚拟机还不能支持外部存储设备。

MetaRouter 功能常用于允许客户或者低特权用户访问自己的“路由”，并根据他们需要自己配置参数，这样不需要另外一个真实的路由器。例如：一个 ISP 能创建一个虚拟路由器，允许特定的用户通过以太接口访问，并定义他们自己的防火墙规则，但只有又不会影响主设备的运行

在/metarouter 目录下给出了以下命令：

- add – 允许你创建一个新的虚拟路由器
- print – 通过列表显示当前所有虚拟路由器
- enable – 启用一个虚拟路由器
- disable – 禁用一个虚拟路由器
- console – 访问一个虚拟路由器的控制台
- interface – 映射相应的网络接口

创建一个 MetaRouter

```
[admin@RB_Meta] /metarouter> add name=mr0 memory-size=32 disk-size=32000    disabled=no
 [admin@RB_Meta] /metarouter> print
 Flags: X - disabled
 #    NAME                    MEMORY-SIZE DISK-SIZE       USED-DISK       STATE
 0    mr0                     16MiB       0kiB            377kiB          running
```

- **name**: 虚拟路由器的名称
- **memory-size**: 分配给虚拟路由器的 RAM 大小
- **disk-size**: HDD 的容量，通过 KB 分配给虚拟路由器 （如果设置为 **0**, 容量默认为动态分配) *
- **used-disk**: 当前使用的硬盘空间 currently used disk space
- **state**: MetaRouter 运行的状态

注：MetaRouter 在使用的动态 HDD 空间时，启用代理功能会占用你所有的 HDD 存储！

默认配置

如果你添加一个新的 MetaRouter 没有指定任何参数，默认会添加动态的 HDD 长度，和 16M 的 RAM：

```
[admin@RB_Meta] /metarouter> add name=mr1
[admin@RB_Meta] /metarouter> print
Flags: X - disabled
 #    NAME                    MEMORY-SIZE DISK-SIZE       USED-DISK       STATE
 1    mr1                     16MiB       0kiB            3kiB            running
```

添加接口

首先需要添加一个新的接口到你的虚拟路由器上，这个操作在 Interface 目录完成，Interface 命令如下面：

```
[admin@MikroTik] /metarouter> interface add
comment      disabled            dynamic-mac-address   type        virtual-machine
copy-from   dynamic-bridge   static-interface        vm-mac-address
```

我们添加一个接口：

```
[admin@MikroTik] /metarouter> interface add virtual-machine=mr1 type=dynamic
```

在物理路由器的 interface 出现一个虚拟接口：

```
[admin@MikroTik] > /interface print
Flags: D - dynamic, X - disabled, R - running, S - slave
#        NAME                                                    TYPE                MTU
8   R   ether9                                                 ether               1500
9   R   test                                                   bridge              1500
10 DR   vif1                                                   vif                 1500
```

连接虚拟机

连接你的虚拟机，使用 console 命令:

```
/metarouter console 0
```

你可以看到你最新添加的虚拟接口：

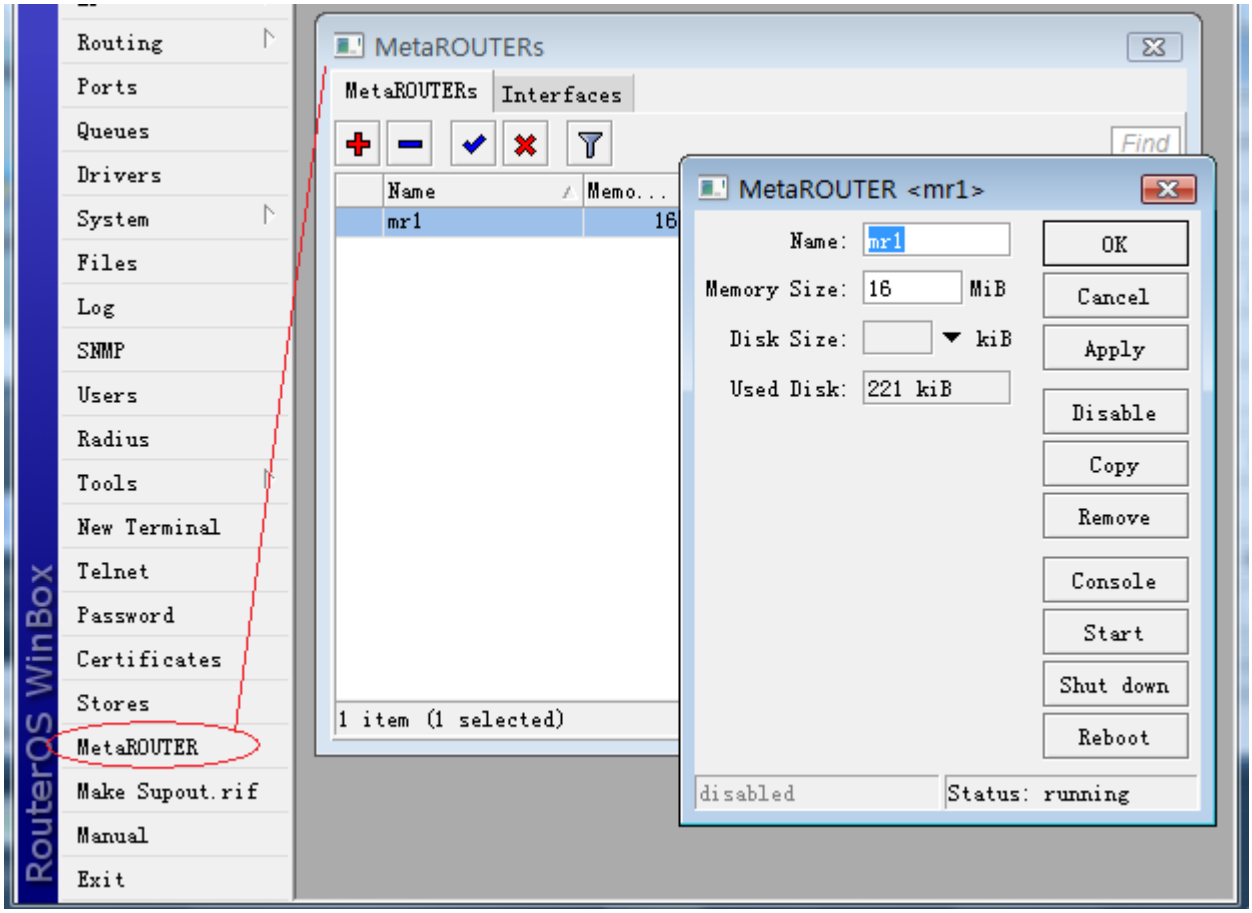
```
[admin@mr0] > interface print
Flags: D - dynamic, X - disabled, R - running, S - slave
#        NAME                                                    TYPE                MTU
0   R   ether1                                                 ether               1500
```

从 MetaRouter 的虚拟机控制台断开，按 CTRL + A 和 Q 退回到物理路由器：

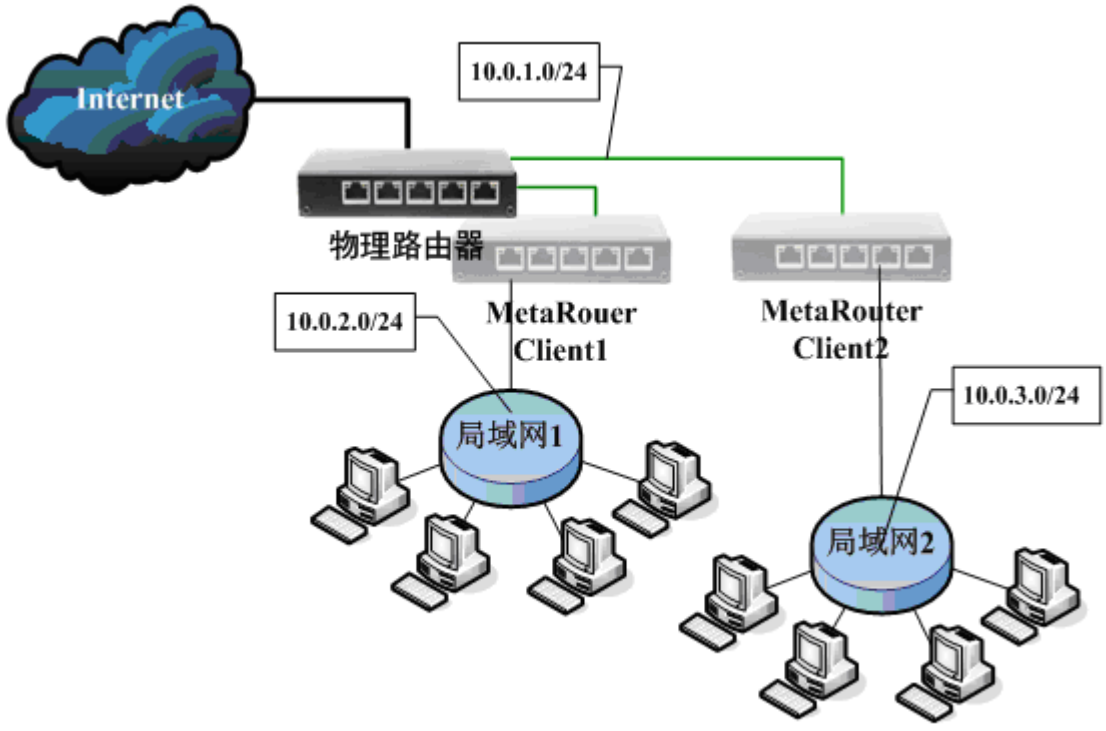
```
[admin@MikroTik] >
[Q - quit connection]        [B - send break]
[A - send Ctrl-A prefix]    [R - autoconfigure rate]

 Q

 Welcome back!
```

MetaRouter 事例

现在你看到之前添加的虚拟接口在物理路由器的 interface 目录中显示为 vif1，当然在 metarouter 的接口中显示为 ether1，你可以在 2 个接口上配置 IP 地址，并连接网络。创建一个 bridge 在允许传输的物理接口和虚拟接口上。下面是 winbox 操作接口

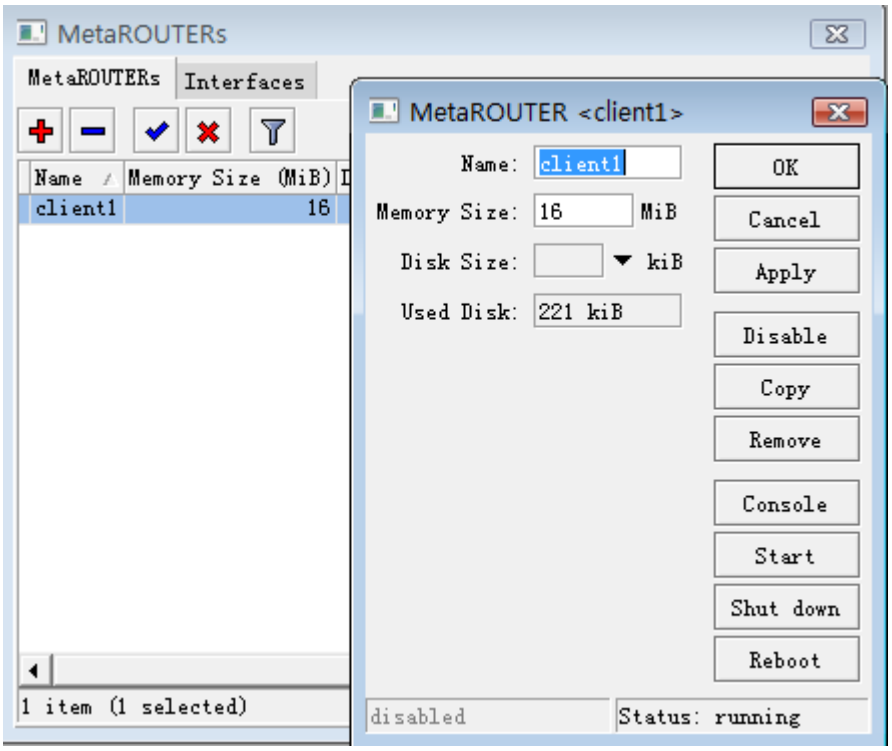


这个事例将介绍如何配置 MetaRouter 功能，为局域网内部独立 RouterOS 虚拟路由，基于 RB450 配置 MetaRouter，我们将建立两个 Client 虚拟路有，并管理两个不同的局域网。目的是让两个个局域网的管理员可以管理自己的路由器（MetaRouter），并根据自己的需要配置他们自己的防火墙、流量控制和 nat 规则，当然他们不能连接物理路由器（没有分配权限）：



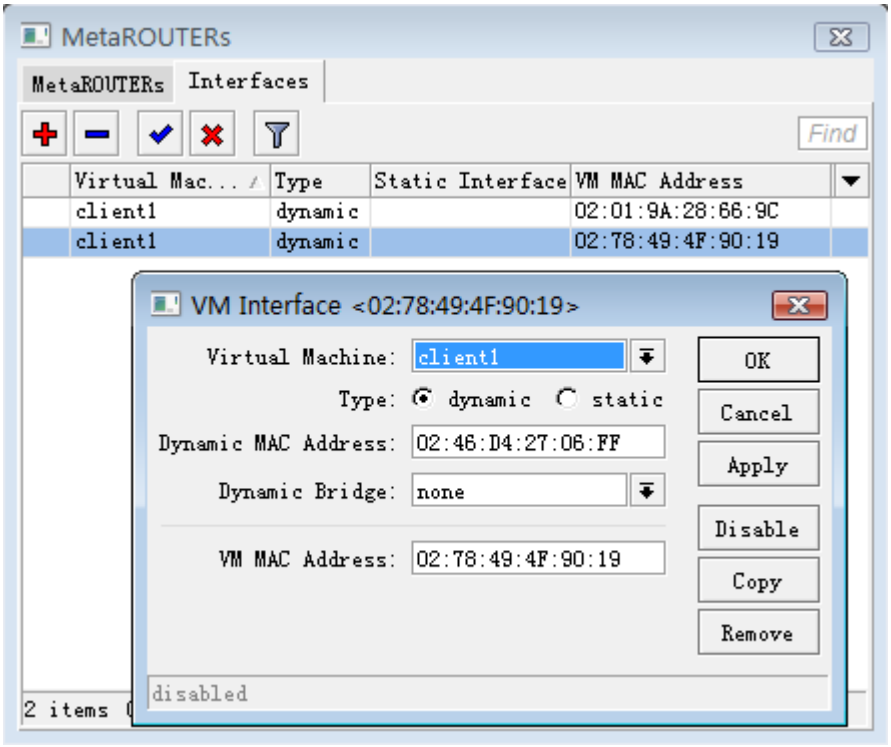
1.给客户添加一个 MetaRouter：

```
[admin@MIKROTIK] /metarouter> add name=client1 memory-size=16
[admin@MIKROTIK] /metarouter> print
Flags: X - disabled
 #    NAME                    MEMORY-SIZE DISK-SIZE       USED-DISK       STATE
 0    client1                 16MiB       0kiB            221kiB          running
[admin@MIKROTIK] /metarouter>
```



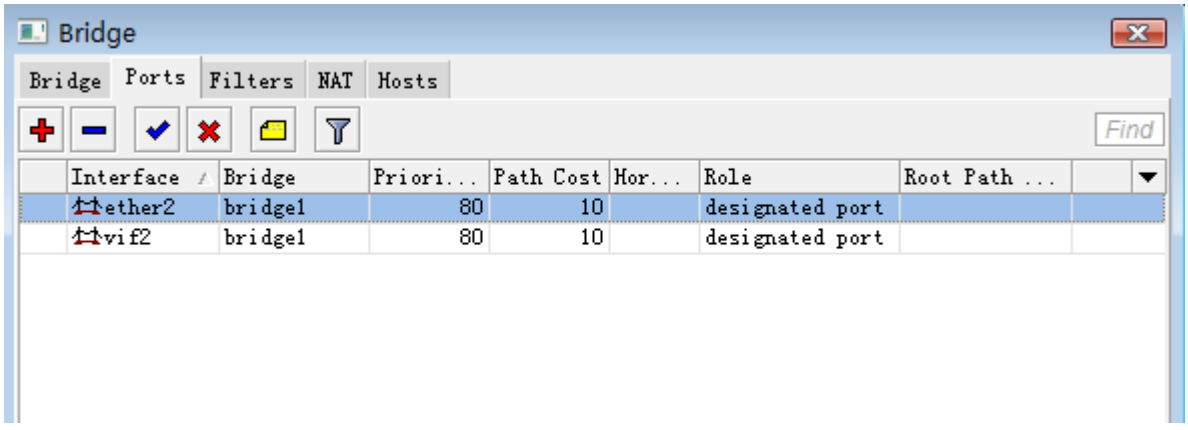
2. 添加 MetaRouter 虚拟机的接口：

```
[admin@MIKROTIK] /metarouter interface> add virtual-machine=client1
[admin@MIKROTIK] /metarouter interface> add virtual-machine=client1
[admin@MIKROTIK] /metarouter interface> print
Flags: X - disabled, A - active
 #    VIRTUAL-MACHINE                    TYPE        VM-MAC-ADDRESS
 0 A client1                             dynamic     02:01:9A:28:66:9C
 1 A client1                             dynamic     02:78:49:4F:90:19
[admin@MIKROTIK] /metarouter interface>
```



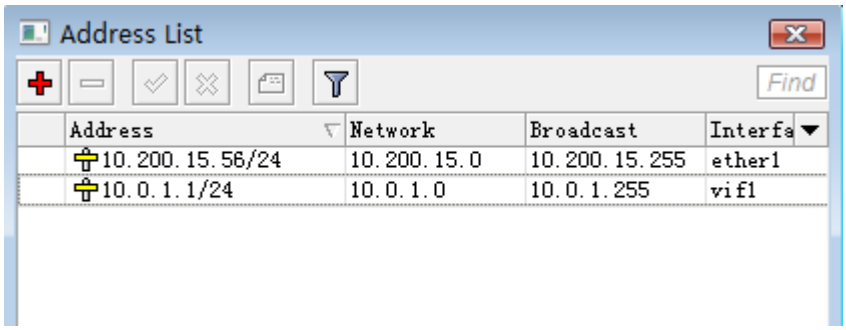
3. 创建一个桥接口，将 MetaRouter 接口与以太网接口桥接，这里我们将 vif2 的虚拟接口放入内网的桥中，用于客户端通过物理接口连接(使用桥接目的是将虚拟路由的内网接口，通过桥接透穿到真实的网络中)：

```
[admin@MIKROTIK] /interface bridge> add
[admin@MIKROTIK] /interface bridge> print
Flags: X - disabled, R - running
 0   R name="bridge1" mtu=1500 arp=enabled mac-address=00:00:00:00:00:00 protocol-mode=none
priority=0x8000 auto-mac=yes admin-mac=00:00:00:00:00:00 max-message-age=20s   forward-delay=15s
transmit-hold-count=6 ageing-time=5m

[admin@MIKROTIK] /interface bridge port> add interface=ether2 bridge=bridge1
[admin@MIKROTIK] /interface bridge port> add interface=vif2 bridge=bridge1
[admin@MIKROTIK] /interface bridge port> print
Flags: X - disabled, I - inactive, D - dynamic
 #     INTERFACE          BRIDGE                PRIORITY   PATH-COST   HORIZON
 0     ether2             bridge1                  0x80          10        none
 1     vif2               bridge1                  0x80          10        none
```



4. 为新的 MetaRouter 接口添加 IP 地址，ether1 作为物理外网连接（假设我们已经配置好物理路由器的网络连接），vif1 用于连接 MetaRouter 主机系统（vif1 可以认为是一个 lan 接口连接）：

```
[admin@MIKROTIK] /ip address> add address=10.0.1.1/24 interface=vif1
[admin@MIKROTIK] /ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #    ADDRESS            NETWORK         BROADCAST       INTERFACE
 0    10.200.15.56/24    10.200.15.0     10.200.15.255   ether1
 1    10.0.1.1/24        10.0.1.0        10.0.1.255      vif1
[admin@MIKROTIK] /ip address>
```



5. 进入 metarouter 控制平台，通过 console 命令：

```
[admin@MIKROTIK] /metarouter> console client1

[Ctrl-A is the prefix key]

Starting...
Starting services...

MikroTik 3.22
MikroTik Login: admin
Password:

[admin@MikroTik] > /sys identity set name=Client1
```

6. 配置 metarouter 的参数，设置以太网接口名称，让客户明白设备的连接情况：

```
 [admin@Client1] /interface ethernet> print
Flags: X - disabled, R - running, S - slave
 #      NAME                MTU   MAC-ADDRESS         ARP
 0 R    ether1              1500  02:49:E8:55:8E:E8   enabled
 1 R    ether2              1500  02:16:16:90:EF:0E   enabled
[admin@Client1] /interface ethernet> set 0 name=wan
[admin@Client1] /interface ethernet> set 1 name=lan
[admin@Client1] /interface ethernet> print
Flags: X - disabled, R - running, S - slave
 #      NAME                MTU   MAC-ADDRESS         ARP
 0 R    wan                 1500  02:49:E8:55:8E:E8   enabled
```

```
 1 R    lan                         1500    02:16:16:90:EF:0E    enabled
[admin@Client1] /interface ethernet>
```

为外网和内网接口配置 IP 地址

```
[admin@Client1] /ip address> add address=10.0.1.2/24 interfae=wan
[admin@Client1] /ip address> add address=10.0.2.1/24 interface=l
[admin@Client1] /ip address> print
Flags: X - disabled, I - invalid, D - dynamic
 #     ADDRESS              NETWORK          BROADCAST        INTERFACE
 0     10.0.1.2/24          10.0.1.0         10.0.1.255       wan
 1     10.0.2.1/24          10.0.2.0         10.0.2.255       lan
```

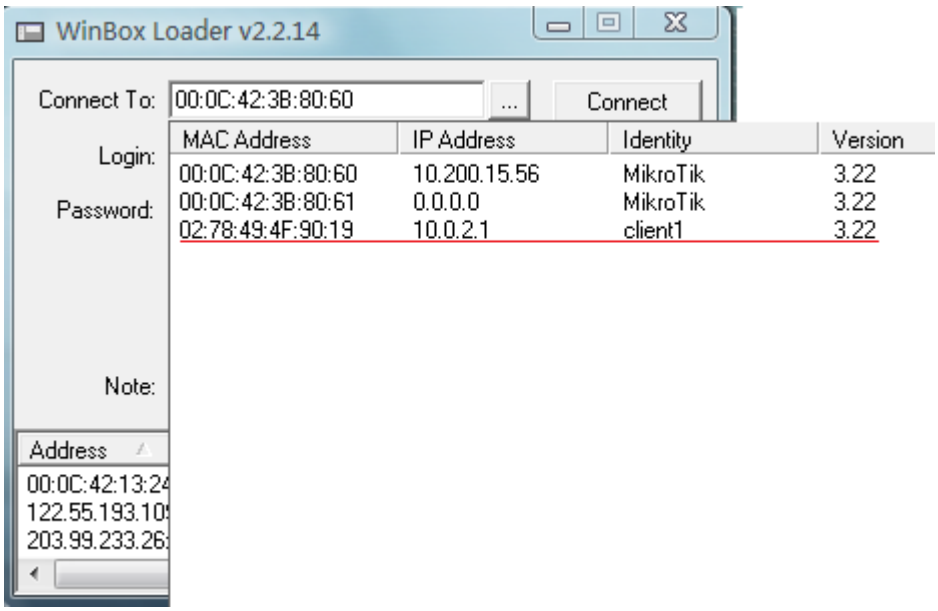
添加默认网关

```
[admin@Client1] /ip route> add gateway=10.0.1.1
[admin@Client1] /ip route> print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS          PREF-SRC      G GATEWAY                DISTANCE   INTERFACE
 0 A S  0.0.0.0/0            r 10.0.1.1                                   1             wan
 1 ADC  10.0.1.0/24            10.0.1.2                                   0               wan
 2 ADC  10.0.2.0/24            10.0.2.1                                   0               lan
[admin@Client1] /ip route>
```

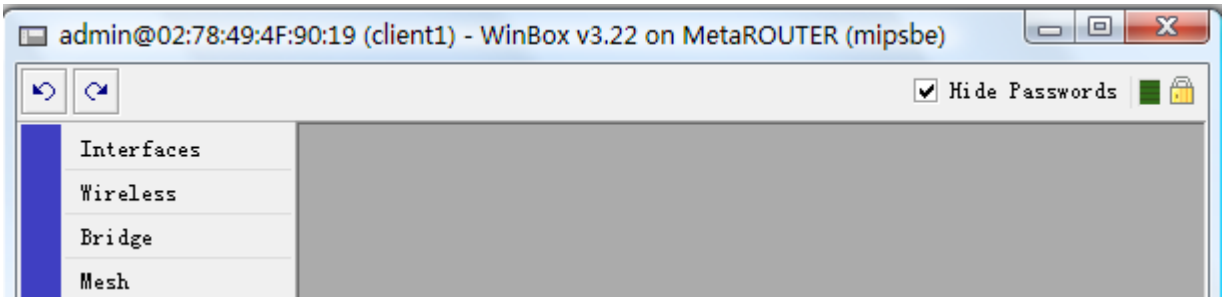
配置 nat 转换

```
[admin@Client1] /ip firewall nat> add action=masquerade out-interface=wan chain=srcnat
```

配置完成后，我们可以通过局域网的 winbox 扫描到配置好的 metarouter

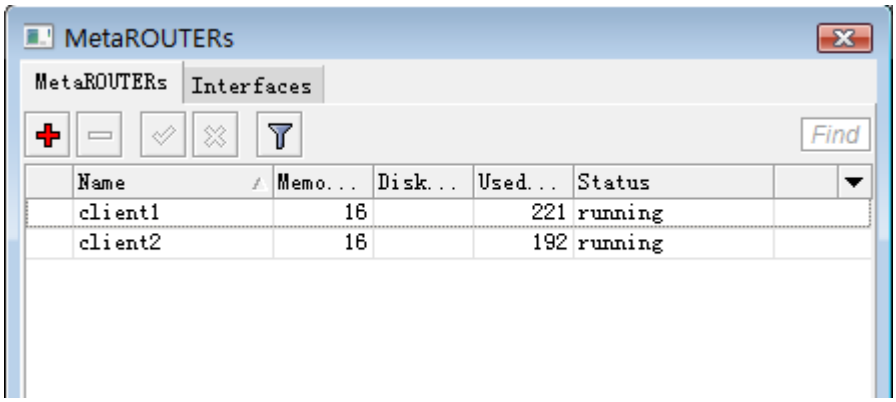


通过连接后，在 winbox 中显示 MetarRouter 信息



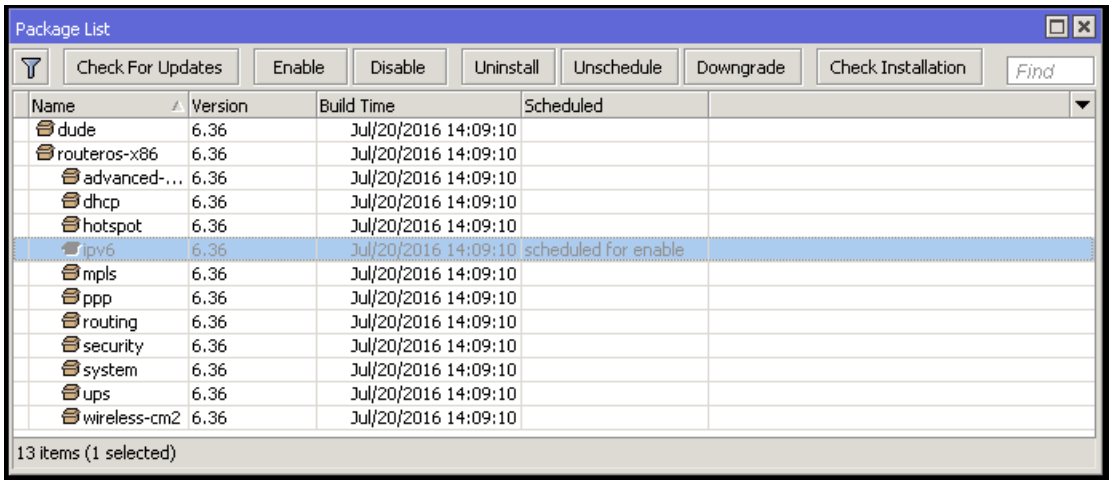
这样我们可以通过局域网内部，连接虚拟的 RouterOS 上网。这样我们可以让局域网 1 的管理进入 client1 的 MetaRotuer 配置自己的网络参数。

我们在用同样的方法建立 10.0.3.0/24 网络的第二个 MetaRouter 在 RouterBOARD 上，只要硬件性能允许，为不同客户提供多个自主路由器管理：



第三十三章节 IPv6

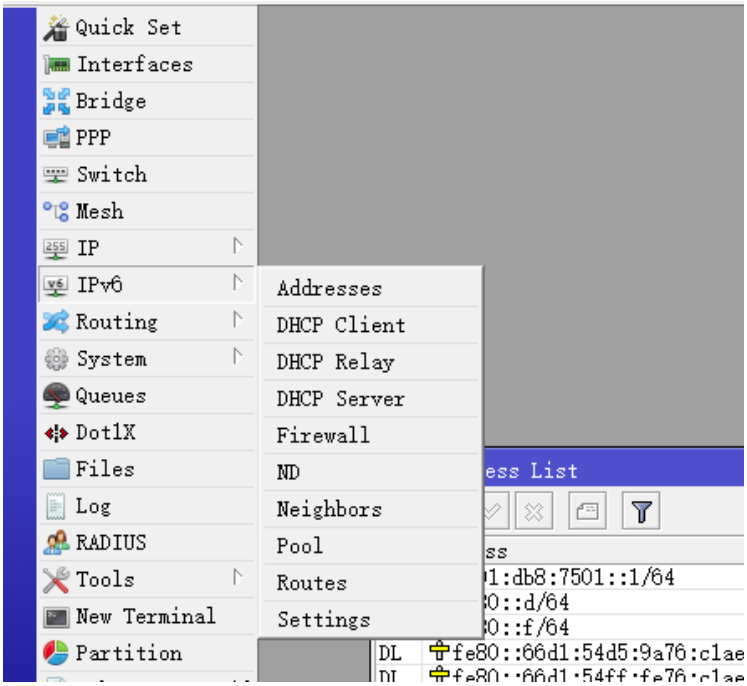
RouterOS 支持 IPv4 和 IPv6 双栈，因此在 RouterOS 能和其他厂商网络设备一样能同时运行 IPv4 和 IPv6 网络。RouterOS 接入 IPv6 前需要查看 RouterOS 是否安装并启用了 IPv6 扩展功能包，进入/system package 查看功能包情况：



默认情况下 RouterOS 的 IPv6 功能包是禁用状态，需要手动点击 enable 启用，并重启路由器。

操作路径： /ipv6 address

如果安装有 IPv6 功能包，可以在 package 菜单下看到 ipv6 的显示，同时也可以在 winbox 左侧菜单看到 IPv6 选项：



IPv6 具有更大的地址空间，IPv4 中规定 IP 地址长度为 32，最大地址个数为 2^32；而 IPv6 中 IP 地址的长度为 128，即最大地址个数为 2^128。与 32 位地址空间相比，其地址空间增加了 2^128-2^32 个。

33.1 地址格式

IPv6 地址结构与 IPv4 地址有所不同，对于 IPv6，是将 128bit 地址分割成 8 个 16bit 区域，每个 16bit 区域被转换成 4 位十六进制和分隔冒号。这样也被称为：colon-hexadecimal（冒分十六进制文本表现形式）.

格式为 X:X:X:X:X:X:X:X，其中每个 X 表示地址中的 16b，在下面的事例中，上方的 IPv6 地址是二进制格式，下方是为十六进制表示

```
二进制
0010000000000001 0000010001110000 0001111100001001 0000000100110001
0000000000000000 0000000000000000 0000000000000000 0000000000001001

十六进制
2001:0470:1f09:0131:0000:0000:0000:0009
```

IPv6 地址书写表示法有以下几种：

1、如果区域有前导 0 是可以省略的，例如：

```
2001:0DB8:0000:0023:0008:0800:200C:417A
2001:DB8:0:23:8:800:200C:417A
```

2、区域出现全 0 时候，采用 0 位压缩表示法：

有时候一个 IPv6 地址中间包含很长的一段 0，可以把连续的一段 0 压缩为"::"。但为保证地址解析的唯一性，地址中"::"只能出现一次，例如：

```
FF01:0:0:0:0:0:0:1101 → FF01::1101
0:0:0:0:0:0:0:1 → ::1
0:0:0:0:0:0:0:0 → ::
```

下面的事例，将 2001:0470:1f09:0131:0000:0000:0000:0009，按照 IPv6 的两种方法依次简化

首先，去掉每个区域领头的 0，如下

```
2001:470:1f09:131:0:0:0:9
```

为了简化编写这些地址，请使用压缩的表现形式，一个连续顺序 0 块由双冒号符号表示(::)。如你看到 IPv6 地址当区域有整段序列为 0，可以将其压缩为“::” 此符号只能出现一次在 IPv6 地址中。

```
2001:470:1f09:131::9
```

IPv6 unicast（单播）地址分为两部分，第一部分包含地址和前缀，第二部分包含接口标识符。 一个简洁方式表示 IPv6 地址/前缀组合如下所示: IPv6 地址/前缀长。

下面是一个地址的示例使用 64 位。

```
3FFE:FFFF:0:CD30:0:0:0:0/64
```

在此示例中的标题由 3FFE:FFFF:0:CD30。 该地址可能还会写入速率压缩的形式

```
3FFE:FFFF:0:CD30::/64。
```

相对于 IPv4 采用小数点子网掩码的表现形有所区别：

```
2001:470:1f09:131::/64
2001:db8:1234::/48
2607:f580::/32
2000::/3
```

33.2 Prefix（前缀）

IPv6 前缀书写格式为“地址/前缀长度”，在 IPv6 中划分网络地址段大小不再称为子网掩码，而称为前缀长度，IPv6 前缀书写格式：

```
2001:470:1f09:131::/64
2001:db8:1234::/48
2607:f580::/32
2000::/3
```

地址类型

下面是几种 IPv6 地址类型：

- Unicast,单播
- Anycast，任播
- Multicast，组播

在 IPv6 网络中没有 Broadcast 地址，与 IPv4 对比 Broadcast 功能完全被 multicast 替代，下面是 IPv6 地址类型、以及与 IPv4 类比。

IPv6 地址类型		IPv6 前缀标识	类似 IPv4
单播地址	Unspecified 未指定地址	::/128	
	Loopback 环回地址	::1/128	127.0.0.1
	Link-local 地址	FE80::/10	169.254.x.x
	Site-local 站点本地地址	FEC0::/10	已弃用
		FC00::/7 其中： -FC00::/8 未定义 -FD00::/8 使用	192.168.x.x 10.x.x.x 172.16.x.x
	全球单播地址	2000::/3	公网 IP
组播地址	Multicast	FF00::/8	224.0.0.0

任播地址	anycast	从单播地址空间中进行分配，使用单播地址格式	

33.3 Interface Identifier（接口标识）

IPv6 地址的最后 64 位是唯一于 IPv6 地址的 64 位前缀的接口标识符。有几种方法可以确定接口标识符：

- EUI-64；
- 随机生成，提供一定匿名功能；
- 手动配置。

EUI-64

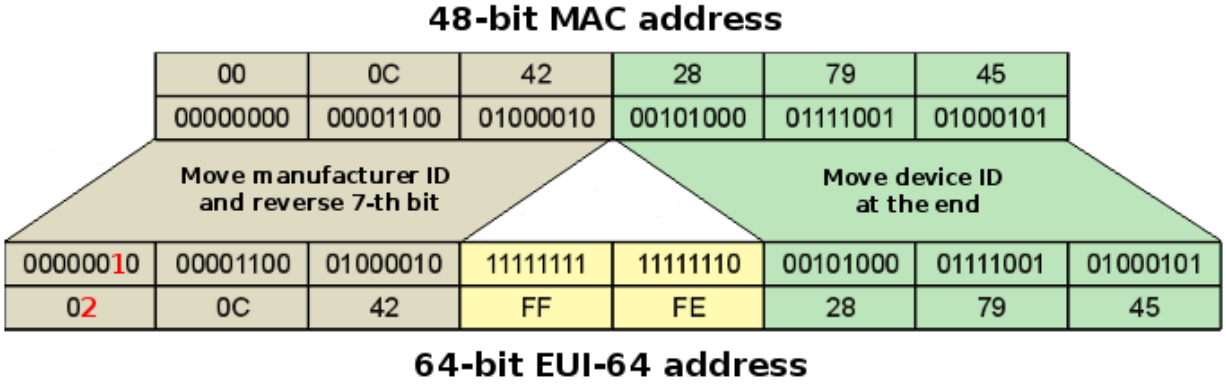
网络适配器的传统接口标识符是 48 位 MAC 地址。这个地址由一个 24 位的制造商 ID 和一个 24 位的设备 ID 组成。

IEEE EUI-64 是一种新的网络接口寻址标准。公司 ID 的长度仍然是 24 位，但是扩展 ID 是 40 位，这为网络适配器创建了更大的地址空间。

通过从网络适配器的 MAC 地址创建一个 EUI-64 地址：

- 0xFFFE 插入到 MAC 地址中，插入位置在制造商 ID 和设备 ID 之间。
- 第一个字节的第 7 位做反转

下面是 MAC 地址 00:0C:42:28:79:45 的事例



根据 EUI-64 规则，我们得到接口身份 ID 为 20C:42FF:FE28:7945，最后本地地址的 IPv6 结果如下：

```
FE80::20C:42FF:FE28:7945/64
```

当 RouterOS 接口上的 eui-64 参数被设置，最后 64bits 会被自动生成，并更新接口标识。在该事例下，最后 bit 需要设置为 0：

```
[admin@MikroTik] > ipv6 address add address=fc00:3::/64 interface=ether3 eui-64=yes
[admin@MikroTik] > ipv6 address print
Flags: X - disabled, I - invalid, D - dynamic, G - global, L - link-local
```

```
 #      ADDRESS                                    INTERFACE                  ADVERTISE
 ...
 5   G fc00:3::20c:42ff:fe1d:3d4/64                ether3                     yes
[admin@MikroTik] > interface ethernet set ether3 mac-address=10:00:00:00:00:01
[admin@MikroTik] > ipv6 address print
Flags: X - disabled, I - invalid, D - dynamic, G - global, L - link-local
 #      ADDRESS                                    INTERFACE                  ADVERTISE
 ...
 5   G fc00:3::1200:ff:fe00:1/64                   ether3                     yes
```

33.4 IPv6 地址分配方式

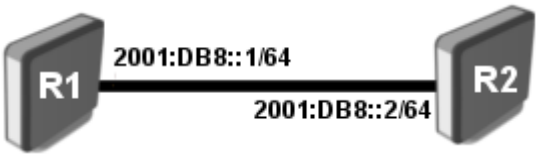
IPv6 协议长达 128bits 的 IPv6 地址，地址范围很大，要求高效合理的地址自动分配和管理，IPv6 地址分配方式除了 link-local（本地链路地址）外，还有以下方式：

- 手动配置 - 手动静态配置 IPv6 地址，及其他网络配置参数。
- 无状态自动地址分配（SLAAC）- 由接口 ID 生成链路本地地址，再根据路由通告报文 RA（Router Advertisement）包含的前缀信息自动配置本机地址。
- 有状态自动地址分配（Stateful）-即 DHCPv6 有状态自动分配，DHCPv6 服务器自动分配 IPv6 地址/前缀及其他网络配置参数（DNS、NIS、SNTP 服务器地址等参数）。

手动静态地址分配

IPv6 地址范围非常大，手动静态地址分配一般是应用于部分专线网络和 IDC 机房分配等，因为这部分业务要求给业务测指定一段固定长度的 IPv6 地址，一般网络分配地址方式不同，可以分为静态的二层接入和三层路由。

下面的实例是静态的二层接入方式，在两台路由器之间手动配置 IPv6 地址，如下图：



R1 配置:

```
/ipv6 address
add address=2001:DB8::1/64 interface=ether1 advertise=no
```

R2 配置:

```
/ipv6 address
add address=2001:DB8::2/64 interface=ether1 advertise=no
```

查看 IPv6 地址配置列表

```
[admin@R1] /ipv6 address> print
```

```
Flags: X - disabled, I - invalid, D - dynamic, G - global, L - link-local
 #      ADDRESS                                    FROM-POOL INTERFACE        ADVERTISE
 0   G 2001:db8::1/64                                    ether1             no
 3 DL fe80::219:d1ff:fe39:3535/64                      ether1             no
```

注意，在添加地址标识出现 G，表示该地址可以通过全球路由，这里也有链路本地地址，并会为每个 IPv6 地址自动创建到接口 I

测试网络是否连通：

```
[admin@R1] /ipv6 address> /ping 2001:DB8::2
HOST                                     SIZE TTL TIME   STATUS
2001:db8::2                              56   64 12ms    echo reply
2001:db8::2                              56   64 0ms     echo reply
    sent=2 received=2 packet-loss=0% min-rtt=0ms avg-rtt=6ms max-rtt=12ms
```

33.5 IPv6 ND

当一个节点需要访问同一二层网络上另一个节点的链路层地址时，就需要进行地址解析。在 IPv4 中使用的是 ARP 协议，在 IPv6 中使用的是 ND(Neighbor discover)协议

RouterOS 使用 Router Advertisement Daemon (RADVD)，通过 RFC 2461 协议，定期地向本地局域网发送路由器通告信息，支持 IPv6 ND（Neighbor Detection）和无状态地址自动配置（stateless address autoconfiguration），对于 IPv6 需用 RADVD，完成无状态地址自动配置功能

Node（节点）是实现 IPv6 的一种网络设备，在 IPv6 网络节点被分为路由器和主机两类：

- **Routers** – 路由器是一个转发 IPv6 数据的节点，不会明确的指向自己
- **Hosts** – 除路由器的任何网络设备。

路由器和主机被严格的分隔开，即在同一时间里路由器不能是主机，主机不能是路由器

StateLess Address AutoConfiguration（无状态地址自动配置）

有两种类型的自动配置：

- **stateless** – 地址配置通过接收路由器宣告信息，该信息包含无状态地址前缀和要求主机不使用有状态地址配置协议。IPv6 的无状态(stateless)自动配置，典型代表是 RADVD。这是 IPv6 协议的一个突出特点：支持网络节点的地址自动配置
- **stateful** – 地址配置通过有状态地址配置协议，即传统的有状态(stateful)，典型代表就是与 IPv4 时代相对应的 DHCPv6，如果 RA 消息不包含地址前缀，则使用有状态协议。

IPv6 一个非常有用的特性，无需使用像 DHCP 这样的有状态配置协议的情况下，进行地址的分配。

具体来说，以 RADVD 为代表的无状态自动配置不需要消耗更多系统资源，也不像传统 DHCP 一样需要一个本地数据库来维护地址分配状态，他只是进行广播前缀地址，客户端收到这种广播后再自己使用 EUI64 算法生成全球唯一的 IPv6 地址，进行自我配置。

因此，RADVD 不能进行 NTP/DNS 等其他传统 DHCP 服务器所能进行的配置。甚至严格的说，它只进行路由广播，地址都是客户端自己根据算法和规范在配置。

DHCPv6 就完全相反，地址池的计算，管理全部是服务器端在做，客户端只是简单的从服务器端取得服务器端已经计算好的地址和其他设置应用到自己身上。

因此，为了兼容 EUI64 算法，RADVD 所能进行管理的地址段要比 DHCPv6 小很多，如果你希望这种无序的自动化管理，并且只是配地址，你可以使用 RADVD，反过来，你如果希望为客户端指定更加详细的 DNS 设定，NTP 设定等等，使用 DHCPv6.

RouterOS 使用 RADVD 定期向同一链接上的所有节点发布关于链接的信息。信息由 ICMPv6"路由器宣告信息"包携带，包含以下字段:

- IPv6 子网前缀
- 默认路由器的 link-local 地址
- 其他相关参数，如 MTU、默认跳数限制和路由器的 lifetime

在主机通过无状态地址自动配置方式时，首先获取到通告信息，IPv6 地址首先从获取的信息中得到子网前缀，然后通过 EUI-64 计算本地身份地址。同时在通告信息中获取默认路由信息。

Neighbors（邻居状态）

Neighbors 菜单为：所有被 IPv6 邻居发现协议，发现的节点列表(邻居缓存表)。

操作路径: /ipv6 neighbor

查看 ipv6 发现邻居情况

```
[admin@test_host] /ipv6 neighbor> print
Flags: R - router
 0     address=ff02::5 interface=main mac-address=33:33:00:00:00:05 status="noarp"

 1     address=ff02::1 interface=main mac-address=33:33:00:00:00:01 status="noarp"

 2 R address=fe80::d7:4cff:fec1:2e32 interface=main mac-address=00:0C:42:28:79:45 status="stale"
```

只读属性

属性	描述
address (*ipv6 地址*)	节点的链路地址。
inteface (*字符*)	节点在那个接口被探测到
mac-address (*字符*)	节点探测的 Mac 地址.
router (*yes* | *no*)	探测节点是否为路由器
status (*noarp* | *incomplete* | *stale* | *reachable* | *delay* | *probe*)	发现条目的缓存状态:

	• **noarp** – 也称为 empty 状态，表示节点上没有相关邻居。 • **incomplete** – 正在解析地址，但邻居链路层地址尚未确定。 • **reachable** – 已知可达的邻居，地址解析成功（10 秒以前） • **stale** – 无法确定邻居是否可达，但探测数据已经发出，不再验证该节点是否可达。 • **delay** - 无法确定邻居是否可达，探测被延迟一段时间，以便给上层协议提供一个机会来提供可达性确认。 • **probe** - 无法确定邻居是否可达，发送单播向邻居探测来验证可达性

各状态间的相互转换如下：

1. 在 noarp（empty）状态时，如果有报文要发送给相邻节点，则在本地邻居缓存表建立该相邻节点，并将该表项置于 INCOMPLETE 状态，同时向相邻节点以组播方式发送 NS 报文。
2. 节点收到邻居回应的单播 NA 报文后，将处于 INCOMPLETE 状态的邻居缓存表项转化为 REACHABLE 状态。如果地址解析失败（发出的组播 NS 超时），则删除该表项。
3. 处于 REACHABLE 状态的表项，如果在 REACHABLE_TIME 时间内没有收到关于该邻居的“可达性证实信息”，则进入 STALE 状态。此外，如果该节点收到邻居节点发出的非 S 置位 NA 报文，并且链路层地址有变化，相关表项会进入 STALE 状态。
4. 处于 STALE 状态的表项，当有报文发往该邻居时，这个报文会利用缓存的链路层地址进行封装，并使该表项进入 DELAY 状态，等待收到“可达性证实信息”。
5. 进入 DELAY 状态后，如果 DELAY_FIRST_PROBE_TIME 时间之内还未收到关于该邻居的“可达性证实信息”，则该表项进入 PROBE 状态。
6. 在 PROBE 状态时，节点会周期性地用 NS 报文来探测邻居的可达性，探测最大时间间隔为 RETRANS_TIMER，在最多尝试 MAX_ UNICAST_SOLICIT 次后，如果仍未收到邻居回应的 NA 报文，则认为该邻居已不可达，该表项将被删除。

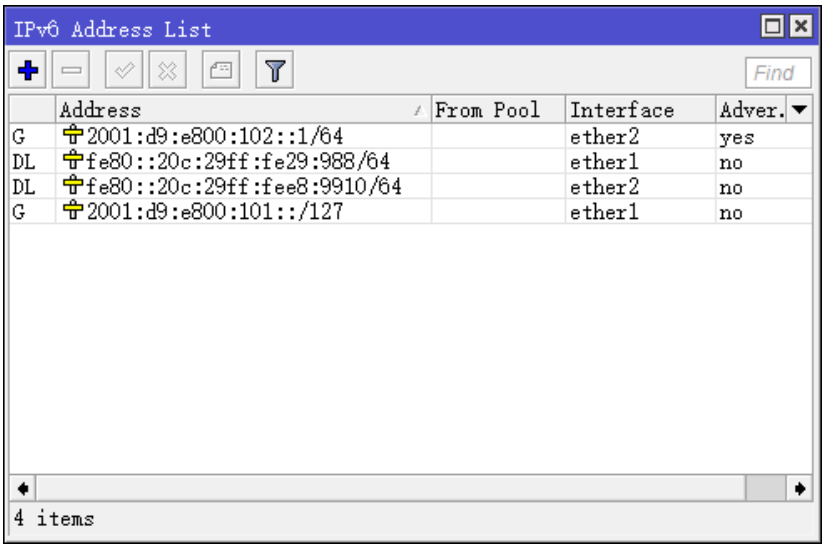
IPv6 无状态地址自动配置实例

IPv6 业务地址运营商通过三层互联，将前缀 64 的地址指定到用户的 RouterOS，RouterOS 通过无状态地址自动配置方式将业务 IPv6 地址分配给终端使用者，如事例中的一台 CentOS 7 服务器。

三层互联地址（运营商）：2001:d9:e800:101::1/127，本地：2001:d9:e800:101::/127

业务地址：2001:d9:e800:102::/64

首先，进入 winbox 的 ipv6 下 address 菜单，添加 IPv6 地址



互联地址配置到 ether1 接口，业务 IP 地址配置到 ether2 接口

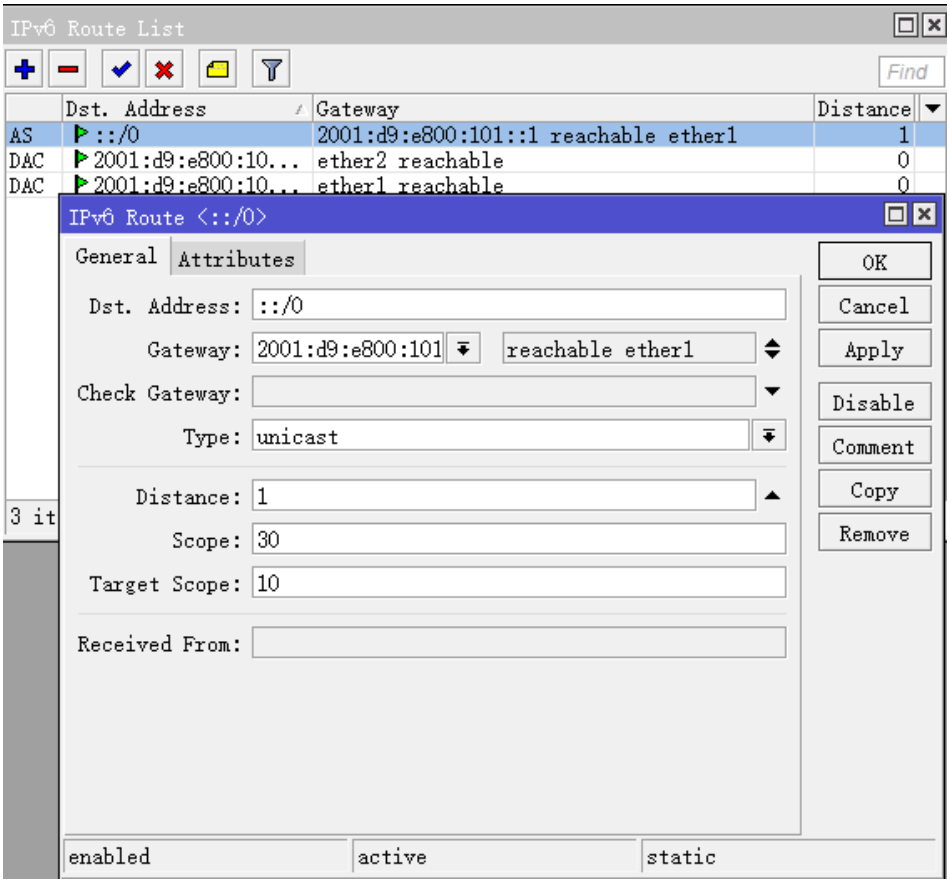
配置脚本：

```
/ipv6 address
add address=2001:d9:e800:101::/127 advertise=no interface=ether1
add address=2001:d9:e800:102::1 interface=ether2 advertise=yes
```

提示：如果运营商分配/127 的互联 IP，当前 RouterOS IPv6 不支持/127 的前缀，只能使用/126 解决！

注意在 ether2 接口上配置 advertise=yes，通过启用宣告功能，开启无状态自动配置

然后进入 ipv6 - route 添加默认路由



配置脚本

```
/ipv6 route
add distance=1 gateway=2001:d9:e800:101::1
```

接下来配置配置 CentOS 7 服务器，首先查看网卡配置，默认情况下，CentOS 7 已经默认设置了 IPv6 的相关参数

```
[root@localhost ~]# cat /etc/sysconfig/network-scripts/ifcfg-ens160
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens160
UUID=ef040552-9e68-4a4a-9be3-fd9bfbe93c7d
DEVICE=ens160
ONBOOT=yes
```

```
IPADDR=192.168.20.2
NETMASK=255.255.255.0
GATEWAY=192.168.20.1
```

现在通过命令 ip -6 addr 查看获取的 IPv6 地址

```
[root@localhost ~]# ip -6 addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 state UNKNOWN qlen 1000
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000
    inet6 2001:d9:e800:102:beb8:3427:cbb7:2ba/64 scope global noprefixroute dynamic
       valid_lft 2591763sec preferred_lft 604563sec
    inet6 fe80::2c1d:2fbb:990e:3845/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

可以看到 ens160 网卡，已经获取的 IPv6 地址

由于 ND 下默认配置，DNS 宣告是关闭的，因此我们要进入 ND 菜单下，将 Advertise DNS 开启，下发 DNS 给 CentOS 7，通过 ND 宣告 DNS，

```
[admin@MikroTik] /ipv6 nd> set advertise-dns=yes
```

在宣告前我们要在 RouterOS 配置 IPv6 网络的 DNS ，ip dns 菜单下支持 v4 和 v6 的 DNS

```
[admin@MikroTik] > ip dns set server=2001:d9:e800:101::
```

33.6 DHCPv6-Server

无状态地址配置（ND）方式中并不记录所连接的 IPv6 主机的具体地址信息，可管理性差。DHCPv6 属于一种有状态地址自动配置协议，DHCPv6 技术则能有效的对地址进行管理。

与其他 IPv6 地址分配方式（手动静态配置、通过路由器无状态自动配置等）相比，DHCPv6 的优点是， DHCPv6 方式不仅可以记录为 IPv6 主机分配的地址，还可以为特定的 IPv6 主机分配特定的地址，以便于网络管理。DHCPv6 支持为网络设备分配 IPv6 前缀，便于全网络的自动配置和网络层次性管理。

DHCPv6 协议架构中，主要包括以下三种角色：

- DHCPv6 Client：DHCPv6 客户端，通过与 DHCPv6 服务器进行交互，获取 IPv6 地址/前缀和网络配置信息，完成自身的地址配置功能。
- DHCPv6 Relay：DHCPv6 中继，负责跨广播域转发来自客户端方向或服务器方向的 DHCPv6 报文，协助 DHCPv6 客户端和 DHCPv6 服务器完成地址配置功能。如果服务器和客户端不在同一广播域范围内，则需要通过 DHCPv6 中继来转发报文，这样可以避免在每个广播域范围内都部署 DHCPv6 服务器，便于进行集中管理。
- DHCPv6 Server：DHCPv6 服务器，负责处理来自客户端或中继代理的地址分配、地址续租、地址释放等请求，为客户端分配 IPv6 地址/前缀和其他网络配置信息。

组播地址

DHCPv6 是一种运行在客户端和服务器之间的协议，与 IPv4 中的 DHCP 一样，所有的协议报文都是基于 UDP 的。但由于在 IPv6 中没有广播报文，因此 DHCPv6 使用组播报文，客户端也无需配置服务器的 IPv6 地址

在 DHCPv6 协议中，客户端不用配置 DHCPv6 Server 的 IPv6 地址，而是发送目的地址为组播地址的 Solicit 报文来定位 DHCPv6 服务器。客户端发送广播报文来定位服务器。为避免广播风暴，在 IPv6 中，已经没有了广播类型的报文，而是采用组播报文。DHCPv6 用到的组播地址有两个：

- FF02::1:2（All DHCP Relay Agents and Servers）：所有 DHCPv6 服务器和中继代理的组播地址，这个地址是链路范围的，用于客户端和相邻的服务器及中继代理之间通信。所有 DHCPv6 服务器和中继代理都是该组的成员。
- FF05::1:3（All DHCP Servers）：所有 DHCPv6 服务器组播地址，这个地址是站点范围的，用于中继和服务器之间的通信，站点内的所有 DHCPv6 服务器都是此组的成员。

DHCPv6 报文承载在 UDPv6 上，客户端侦听的 UDP 目标端口号是 546，服务器、中继侦听的 UDP 端口号是 547。

DUID

DUID 是 DHCP 唯一标识符（DHCP Unique Identifier），每个服务器或客户端有且只有一个唯一标识符，服务器使用 DUID 来识别不同的客户端，客户端则使用 DUID 来识别服务器，DHCPv4 通过 MAC 地址识别主机，DHCPv6 则通过 DUID。

客户端和服务器 DUID 的内容分别通过 DHCPv6 报文中的 Client Identifier 和 Server Identifier 选项来携带。两种选项的格式一样，通过 option-code 字段的取值来区分是 Client Identifier 还是 Server Identifier 选项。

身份联盟（IA）

身份联盟 IA（Identity Association）是使得服务器和客户端能够识别、分组和管理一系列相关 IPv6 地址的结构。每个 IA 包括一个 IAID 和相关联的配置信息。

客户端必须为它的每一个要通过服务器获取 IPv6 地址的接口关联至少一个 IA。客户端用给接口关联的 IA 来从服务器获取配置信息。每个 IA 必须明确关联到一个接口。

IA 的身份由 IAID 唯一确定，同一个客户端的 IAID 不能出现重复。IAID 不会因为设备的重启等因素发生丢失或改变。IA 中的每个地址都有首选生存期和有效生存期。一个接口至少关联一个 IA，一个 IA 可以包含一个或多个地址信息。

DUID 用于客户端和服务器唯一标识，只有 IAID 会在不同的客户端之间变化对应他们的分配接口

服务器中的客户端绑定不再显示 MAC 地址，而是使用 DUID(十六进制)和 IAID。更新后 MAC 地址会自动转换为 DUID。

操作路径: /ipv6 dhcp-server

注意：RouterOS DHCPv6 Server 只能授权 IPv6 前缀（DHCPv6-PD），而不支持 address 分配。

DHCPv6 基本配置

从运营商获取到了 2001:123:123::/56 的地址长度，在路由器启用 DHCPv6-PD 服务器，首先需要在/ipv6 pool 下创建地址池，如下：

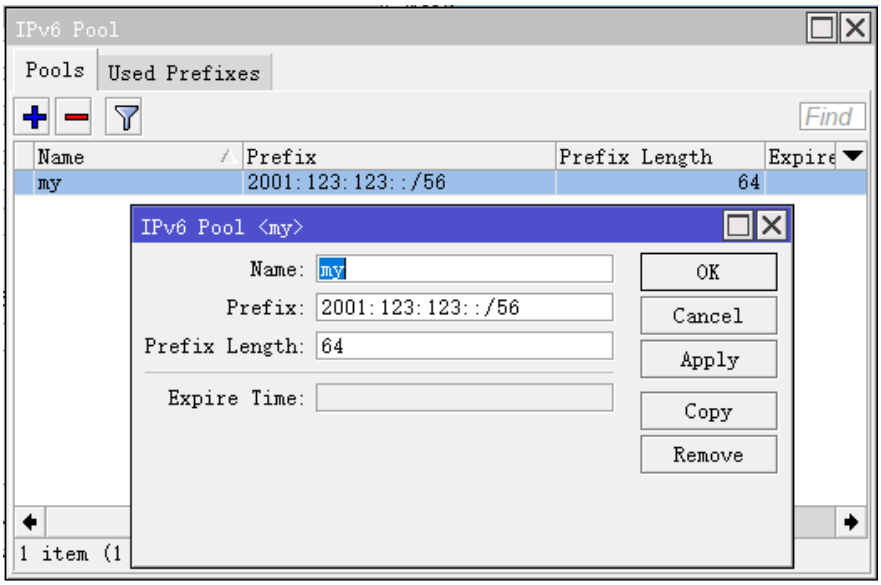
```
/ipv6 pool add name=my prefix=2001:123:123::/56    prefix-length=64
```

注意，prefix-length 是 64bits，即客户端会从/56 的地址池中获取/64 的前缀

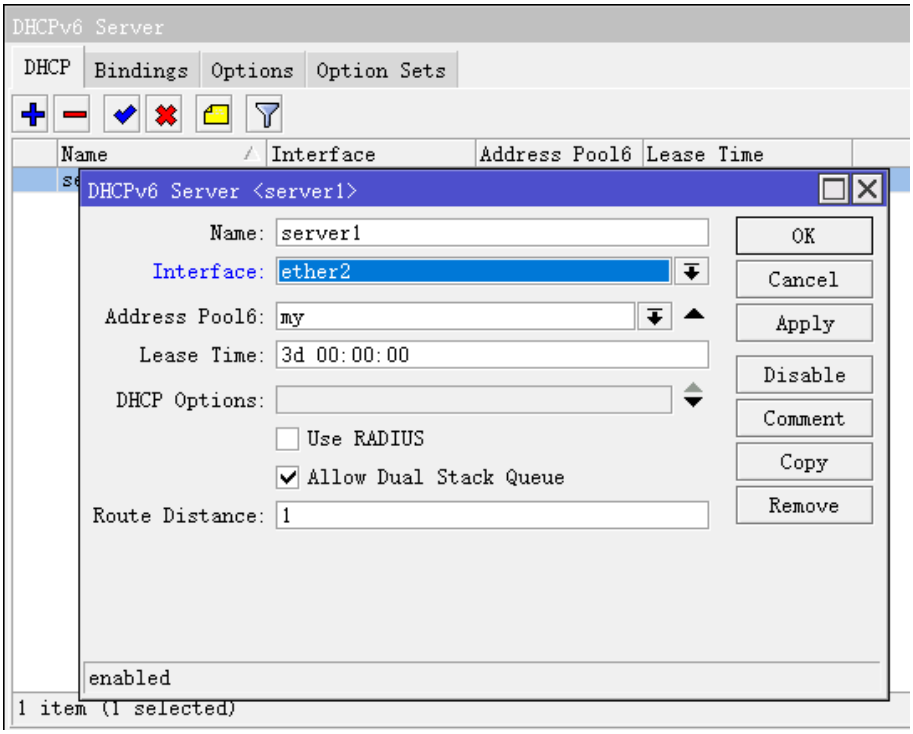
进入 ipv6 dhcp-server 启用 DHCPv6 服务

```
/ipv6 dhcp-server add name=server1 address-pool=my interface=ether2
```

DHCPv6 配置 Winbox 配置



创建 DHCPv6 Server



Bindings

操作路径：/ipv6 dhcp-server binding

DHCPv6 通过 DUID 来识别客户端用户 ID，类似于 DHCPv4 的 MAC，当客户端获取 IPv6 前缀后，会动态添加到 Binding 中，也可以手动静态添加 。

在 binding 菜单下，查看动态分配的前缀/62 的条目

```
[admin@RB493G] /ipv6 dhcp-server binding> print detail
Flags: X - disabled, D - dynamic
 0 D address=2a02:610:7501:ff00::/62 duid="1605fcb400241d1781f7" iaid=0
       server=local-dhcp life-time=3d status=bound expires-after=2d23h40m10s
       last-seen=19m50s

 1 D address=2a02:610:7501:ff04::/62 duid="0019d1393535" iaid=2
       server=local-dhcp life-time=3d status=bound expires-after=2d23h43m47s
       last-seen=16m13s
```

Rate limiting（限速）

在 DHCPv6 binding 下，可以设置指定 IPv6 地址的带宽限速，需要将动态条目变为静态后，设置 rate-limit 参数，会动态添加 simple queue 规则

警告：当需要对 IP/IPv6 地址进行限速时，需要确保 FastTracked 功能没有开启，否则限速会失效

配置限速前，需要将指定的 DHCPv6 binding 条目变为静态，否则无法设置动态条目的限速规则：

```
[admin@MikroTik] > /ipv6 dhcp-server binding print
```

```
Flags: X - disabled, D - dynamic
 #    ADDRESS                                DUID                SERVER                STATUS
 0 D fdb4:4de7:a3f8:418c::/66      0x6c3b6b7c413e          DHCPv6_Server        bound
```

通过 make-static 命令将条目“0”变为静态

```
[admin@MikroTik] > /ipv6 dhcp-server binding make-static 0

[admin@MikroTik] > /ipv6 dhcp-server binding print
Flags: X - disabled, D - dynamic
 #    ADDRESS                              DUID                  SERVER                STATUS
 0    fdb4:4de7:a3f8:418c::/66    0x6c3b6b7c413e              DHCPv6_Server        bound
```

变为静态后，可以设置条目“0”的 rate-limit 参数：：

```
[admin@MikroTik] > /ipv6 dhcp-server binding set 0 rate-limit=10M/10M
```

设置完成后，将在 queue simple 菜单下看到该段地址的动态限速规则

```
[admin@MikroTik] > /queue simple print
Flags: X - disabled, I - invalid, D - dynamic
 0   D name="dhcp<6c3b6b7c413e fdb4:4de7:a3f8:418c::/66>" target=fdb4:4de7:a3f8:418c::/66 parent=none
packet-marks="" priority=8/8 queue=default-small/default-small limit-at=10M/10M max-limit=10M/10M
burst-limit=0/0 burst-threshold=0/0   burst-time=0s/0s bucket-size=0.1/0.1
```

当 **allow-dual-stack-queue** 属性启用，假如同时配置了 DHCPv4 和 DHCPv6 时，将同时执行 IPv4 和 IPv6 的双栈限速策略在一条 simple queue 规则中

如下所示，**allow-dual-stack-queue** 启用，可以看到在一条 simple queue 里同时创建 IPv4 和 IPv6 双栈地址：

```
[admin@MikroTik] > /queue simple print
Flags: X - disabled, I - invalid, D - dynamic
 0   D name="dhcp-ds<6C:3B:6B:7C:41:3E>" target=192.168.1.200/32,fdb4:4de7:a3f8:418c::/66 parent=none
packet-marks="" priority=8/8 queue=default-small/default-small limit-at=10M/10M max-limit=10M/10M
burst-limit=0/0 burst-threshold=0/0 burst-time=0s/0s bucket-size=0.1/0.1
```

33.7 DHCPv6-Client

RouterOS 中的 DHCPv6-client 可以作为 DHCPv6-client 和 DHCP-PD client，因此，它可以从 DHCP-PD 服务器获取前缀，也可以从 DHCPv6 服务器获取 DHCPv6 有状态地址.

操作路径: /ipv6 dhcp-client

快速设置

下面是一个简单的实例，展示如何启用 DHCP-client 如何接收 IPv6 前缀，并添加到地址池中

```
/ipv6 dhcp-client
add request=prefix pool-name=test-ipv6 pool-prefix-length=64 interface=ether13
```

当接收到前缀，使用 print detail 命令，详细显示客户端状态

```
[admin@x86-test] /ipv6 dhcp-client> print detail
Flags: D - dynamic, X - disabled, I - invalid
 0      interface=bypass pool-name="test-ipv6" pool-prefix-length=64 status=bound
        prefix=2001:db8:7501:ff04::/62   expires-after=2d23h11m53s request=prefix
```

服务器分配前缀为 2a02:610:7501:ff04::/62 ，并添加到 ipv6 pool 中

```
[admin@MikroTik] /ipv6 pool> print
Flags: D - dynamic
 #    NAME                                PREFIX                              REQUEST
PREFIX-LENGTH
 0 D test-ipv6                            2001:db8:7501:ff04::/62             prefix
                                  64
```

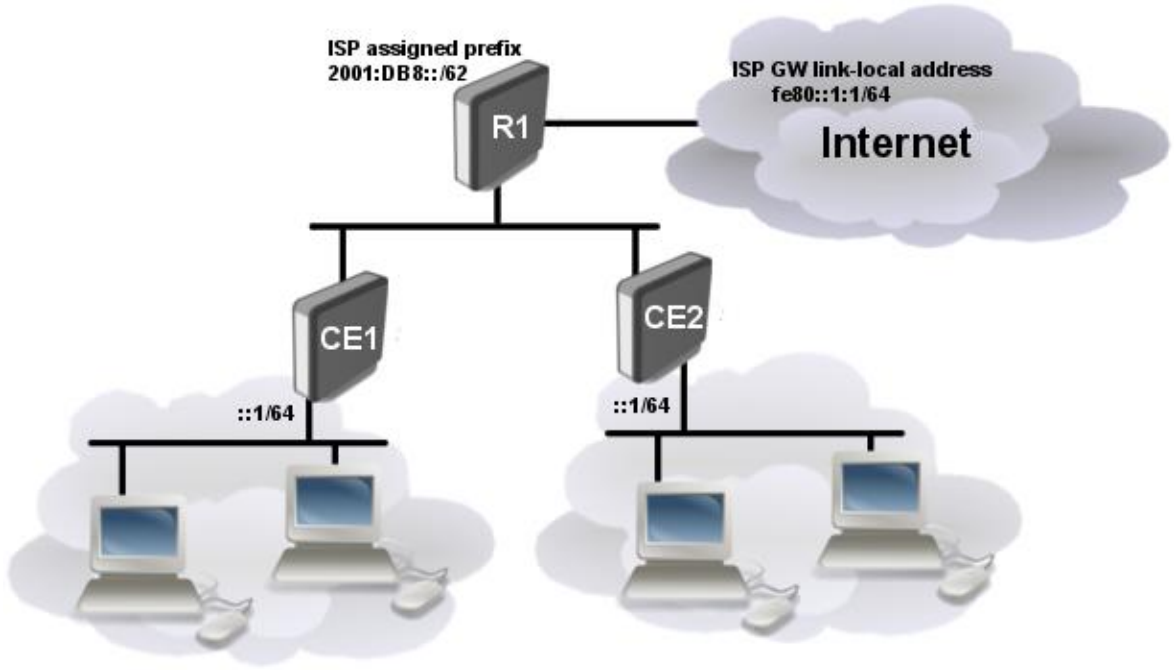
这样 IPv6 地址获取

属性

属性	描述
add-default-route (*yes* | *no*; 默认: **no**)	是否在客户端连接成功后，添加 IPv6 默认路由
comment (*字符*;)	客户端注释标签
disabled (*yes* | *no*; 默认: **no**)	
interface (*字符*; 默认:)	DHCPv6 在那个网络接口运行.
pool-name (*字符*; 默认:)	IPv6 pool 名称，用于接收 IPv6 前缀
pool-prefix-length (*字符*; 默认:)	前缀长度参数，为 IPv6 Pool 设置，添加接收到的 IPv6 前缀。前缀长度必须大于接收到的前缀长度。
prefix-hint (*字符*; 默认:)	包含一个首选前缀长度
request (*prefix, address*; 默认:)	选择 DHCPv6 请求方式，是获取 address、IPv6 前缀或两者同时获取。
script (*字符*; 默认:)	在 DHCP-client 状态改变时，运行脚本，可以获得以下脚本： • pd-valid – 当客户端获取前缀; • pd-prefix –在任何情况下客户端获得的前缀; • na-valid – 当客户端获取地址; • na-address - 在任何情况下客户端获得的地址
use-peer-dns (*yes* | *no*; 默认: **yes**)	是否接受 IPv6 DHCP 服务器分配的 DNS 设置

DHCPv6 Server 与 Client 配置实例

通下面的实例一个完整网络的实例，介绍 DHCPv6 Server 与 Client 的配置：



- ISP 通过静态路由方式，将前缀 2001:DB8::/62 分配到路由器 R1
- 路由器 R1 运行 DHCPv6 服务器分配/64 前缀长度给客户端路由器 CE1 和 CE2
- 在路由器 CE1 和 CE2 运行 DHCP 客户端，并接收来至路由器 R1 分配的前缀/64 地址
- 客户端路由器使用本地接口的 RA 接收前缀，

R1 路由器配置

```
/ipv6 route
add gateway=fe80::1:1%to-ISP

/ipv6 pool
add name=myPool prefix=2001:db8::/62 prefix-length=64

/ipv6 dhcp-server
add address-pool=myPool disabled=no interface=to-CE-routers lease-time=3m name=server1
```

CE1 路由器配置

```
/ipv6 dhcp-client
add interface=to-R1 request=prefix pool-name=my-ipv6

/ipv6 address
add address=::1/64 from-pool=my-ipv6 interface=to-clients advertise=yes
```

CE2 路由器配置

```
/ipv6 dhcp-client
add interface=to-R1 request=prefix pool-name=my-ipv6

/ipv6 address
add address=::1/64 from-pool=my-ipv6 interface=to-clients advertise=yes
```

查看状态，当配置完成，核实每个 CE 路由器接收到自己的 IPv6 地址前缀

在 DHCPv6 服务器端：

```
[admin@R1] /ipv6 dhcp-server binding> print
Flags: X - disabled, D - dynamic
 #     ADDRESS                                    DUID                IAID SERVER
STATUS
 1 D 2001:db8:1::/64                          0019d1393536        566 server1        bound
 2 D 2001:db8:2::/64                          0019d1393535        565 server1        bound
```

客户端情况：

```
[admin@CE1] /ipv6 dhcp-client> print
Flags: D - dynamic, X - disabled, I - invalid
 #      INTERFACE              STATUS          REQUEST          PREFIX
 0      to-R1                  bound           prefix          2001:db8:1::/64

[admin@CE1] /ipv6 dhcp-client> /ipv6 pool print
Flags: D - dynamic
 #     NAME                    PREFIX                                PREFIX-LENGTH
 0 D my-ipv6                  2001:db8:1::/64                                  64
```

还可以看到在 ipv6 address 下的地址，是从前缀池自动添加的

```
[admin@CE1] /ipv6 address> print
Flags: X - disabled, I - invalid, D - dynamic, G - global, L - link-local
 #      ADDRESS                                    FROM-POOL INTERFACE       ADVERTISE
 0   G 2001:db8:1::1/64                                    to-clients          yes
..
```

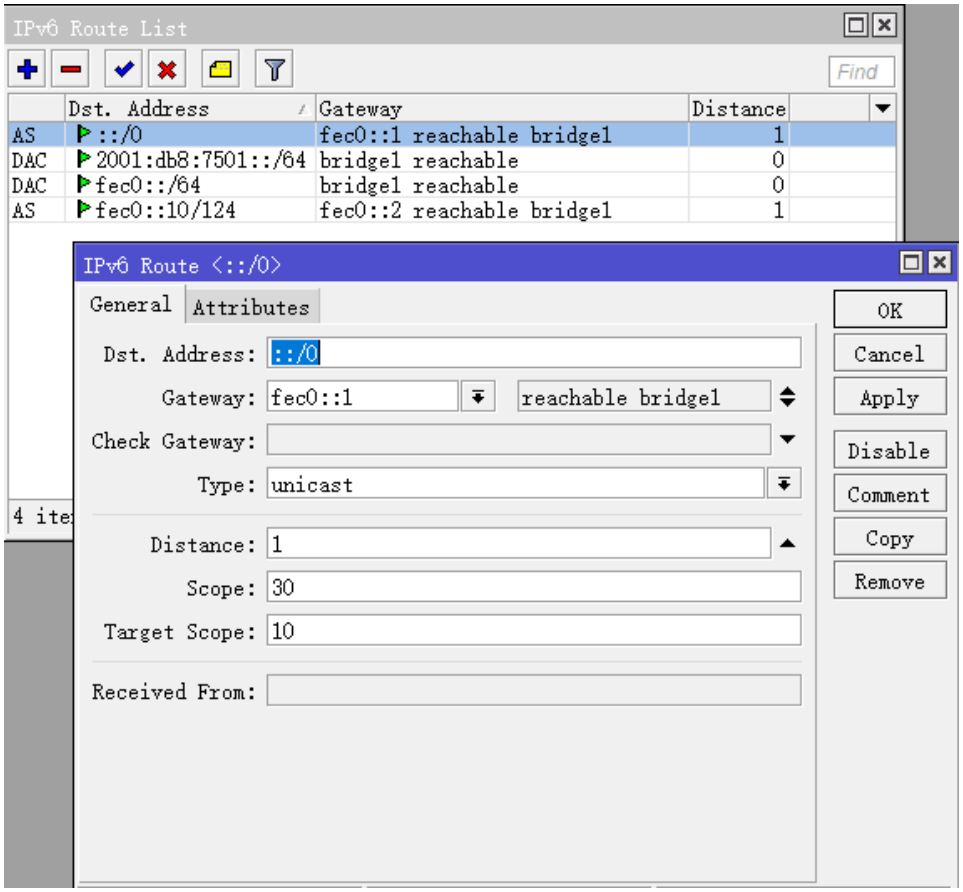
并在地址池使用情况看到地址已被分配

```
[admin@CE1] /ipv6 pool used> print
POOL            PREFIX                                    OWNER           INFO
my-ipv6        2001:db8:1::/64
```

33.8 Route（IPv6 路由）

操作路径：/ipv6 route

IPv6 路由配置和 IPv4 一样，通过添加默认网关指定默认路由，都是在 route 菜单下



对于静态路由，基本配置与 IPv4 相同，下面是一个简单的 ipv6 添加目标静态路由的配置实例：

```
[admin@MikroTik] > ipv6 route add dst-address=2001::/16 gateway=fc00:dead:beef::2
[admin@MikroTik] > ipv6 route print detail
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, o - ospf, b - bgp, U - unreachable
 0 A S    dst-address=2001::/16 gateway=fc00:dead:beef::2 reachable ether1 distance=1 scope=30
target-scope=10
```

ipv4 和 ipv6 最显著的区别是，如果接口被指定，链路本地地址也可以用作路由下一跳网关：

```
[admin@MikroTik] > ipv6 route add dst-address=2002::/16 gateway=fe80::21a:4dff:fe56:1f4d%ether1
[admin@MikroTik] > ipv6 route print detail
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, o - ospf, b - bgp, U - unreachable
 ...
 1 A S    dst-address=2002::/16
          gateway=fe80::21a:4dff:fe56:1f4d%ether1 reachable distance=1
          scope=30 target-scope=10
```

另一个小的区别是，没有黑洞或禁止路由，只有 unreachable（无法到达）。

下面的事例，是达到相同的目标地址，通过不同的接口，在一条路由规则配置多条网关：

```
[admin@MikroTik] > ipv6 address print
```

```
Flags: X - disabled, I - invalid, D - dynamic, G - global, L - link-local
 #      ADDRESS                                     INTERFACE            ADVERTISE
 0   G fc00:1::1/64                                 ether1               no
 1   G fc00:2::1/64                                 ether2               no
[admin@MikroTik] > ipv6 route add dst-address=2001::/16 gateway=fc00:1::2,fc00:2::2
[admin@MikroTik] > ipv6 route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, o - ospf, b - bgp, U - unreachable
 #         DST-ADDRESS                 GATEWAY                              DISTANCE
 0 A S   2001::/16                   fc00:2::2 reachable ether1,   1
                                      fc00:1::2 reachable ether2
```

RouterOS 通过一条规则完成多网关配置，对于 linux 内核而言，并非在同一条规则完成，而是通过多条规则实现，当我们查看 linux 内核路由表时，看到如下：

```
# ip -6 route
2001::/16 via fc00:2::2 dev eth1   proto static   metric 1024   mtu 1500 advmss 1440 metric10 4294967295
2001::/16 via fc00:1::2 dev eth0   proto static   metric 1024   mtu 1500 advmss 1440 metric10 4294967295
...
```

33.9 IPv6 防火墙

IPv6 地址可用地址范围已经非常大，全球 IPv6 地址分配远远大于 IPv4，由于 IPv4 不够用，国内运营商给家庭宽带大部分分配的是私网 IP 地址，只有申请才能获取公网 IPv4 地址，而且家庭内网的设备只能通过 IPv4 私网地址通过 NAT 转换上网，由于路由器的 NAT 功能隐藏了内网的设备，避免暴露在公网上，也起到了一定的安全作用。IPv6 地址的庞大地址池，所以不再会有私网 IP 地址的说法，并且能为家庭每一个电器设备分配一个 IPv6 公网地址，也就是你的家庭网络所有 IPv6 地址都是公网，但这也带来了安全问题。对于 IPv6 安全策略是非常重要的。

下面简单介绍下，IPv6 防火墙的安全策略

IPv6 路由器本地防火墙策略

WAN 接口为路由器接入 IPv6 的互联网端口，具体配置时需要根据你路由器接口名称修改，基本的安全策略思路如下：

1. 接受 established/related 的数据包;
2. 接受 ICMPv6 数据
3. 接受 UDP 路由跟踪协议端口 33434-33534
4. 接受 DHCPv6 客户端的 PD 授权端口 UDP 546
5. 丢弃来至互联网接口的 link-local 地址（本地链路地址）;
6. 接受 link-local 地址，接受组播地址，接受自己的 IPv6 地址访问路由器;
7. 丢弃 drop anything else;

配置如下：

```
/ipv6 firewall filter
add action=accept chain=input comment="allow established and related" connection-state=established,related

add chain=input action=accept protocol=icmpv6 comment="accept ICMPv6"

add chain=input action=accept protocol=udp port=33434-33534 comment="defconf: accept UDP traceroute"

add chain=input action=accept protocol=udp dst-port=546 src-address=fe80::/16 comment="accept
DHCPv6-Client prefix delegation."

add action=drop chain=input in-interface= WAN log=yes log-prefix=dropLL_from_public src-address=fe80::/16

add action=accept chain=input comment="allow allowed addresses" src-address-list=allowed

add action=drop chain=input

/ipv6 firewall address-list
add address=fe80::/16 list=allowed
add address=your_ipv6 address    list=allowed
add address=ff02::/16 comment=multicast list=allowed
```

IPv6 用户侧防火墙策略

当从运营商分配到 IPv6 地址后，并通过路由器分配到内网主机 IPv6 地址，内网的主机将获取公网 IPv6 地址，这样带来一个安全问题，即全球互联网都可以访问到你的主机，而不是像 IPv4 通过路由器的 nat 转换后到互联网，nat 可以隐藏私网 IPv4 地址，隐藏通过配置相应的防火墙保护路由器后的主机非常重要，大致配置方式如下：

1. 接受 established/related 数据包;
2. 丢弃非法数据包，并记录到日志中;
3. 接受除了 WAN（外网接口）ICMPv6 数据包；
4. 接受从客户端到互联网的连接;
5. 丢弃其他所有数据。

防火墙 1，4，5 规则策略组合非常重要，允许从内网到外网的访问，但外网向内网访问被拒绝，保证网络内部 IPv6 网络的安全性，避免别外部恶意访问，配置如下：

```
/ipv6 firewall filter
add action=accept chain=forward comment=established,related connection-state=established,related

add action=drop chain=forward comment=invalid connection-state=invalid log=yes log-prefix=ipv6,invalid

add action=accept chain=forward comment=icmpv6 in-interface=! WAN protocol=icmpv6

add action=accept chain=forward comment="local network" in-interface=!WAN src-address-list=allowed

add action=drop chain=forward log-prefix=IPV6
```

如果需要指定外部 IPv6 地址访问，需在最后一条规则 drop 所有数据前面加上目标 IPv6 地址段 2001:db8:1::/64

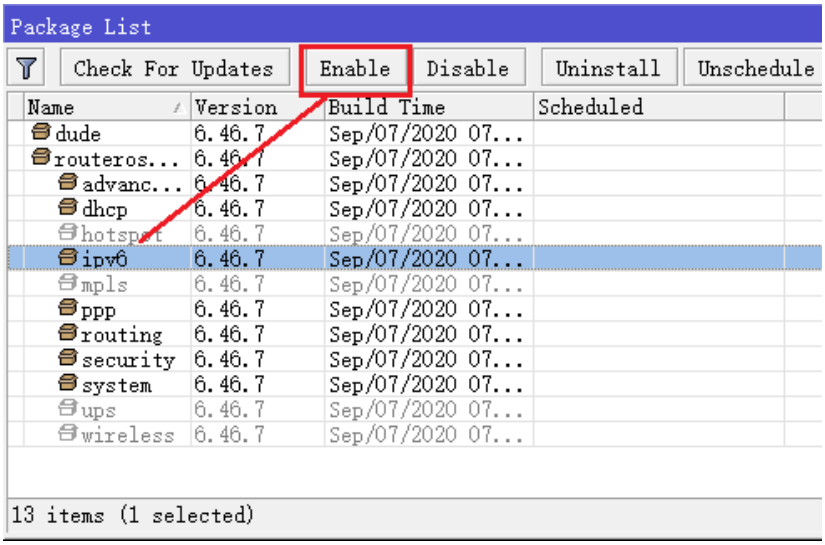
```
[admin@MikroTik] /ipv6 firewall filter> print
Flags: X - disabled, I - invalid, D - dynamic
 0      ;;; allow established and related
        chain=input action=accept connection-state=established,related log=no log-prefix=""

 1      ;;; accept ICMPv6
        chain=input action=accept protocol=icmpv6 log=no log-prefix=""

 2      ;;; defconf: accept UDP traceroute
        chain=input action=accept protocol=udp port=33434-33534 log=no log-prefix=""

 3      ;;; accept DHCPv6-Client prefix delegation.
        chain=input action=accept protocol=udp src-address=fe80::/16 dst-port=546 log=no log-prefix=""

 4      chain=input action=drop src-address=fe80::/16 in-interface=pppoe-out2 log=yes
log-prefix="dropLL_from_public"

 5      ;;; allow allowed addresses
        chain=input action=accept src-address-list=allowed log=no log-prefix=""

 6      chain=input action=drop log=no log-prefix=""

 7      ;;; established,related
        chain=forward action=accept connection-state=established,related log=no log-prefix=""

 8      ;;; invalid
        chain=forward action=drop connection-state=invalid log=yes log-prefix="ipv6,invalid"

 9      ;;; icmpv6
        chain=forward action=accept protocol=icmpv6 in-interface=!WAN log=no log-prefix=""

10      ;;; local network
        chain=forward action=accept in-interface=!WAN src-address-list=allowed log=no log-prefix=""

11      chain=forward action=drop log=no log-prefix="IPV6"

[admin@MikroTik] /ipv6 firewall filter> add action=accept chain=forward dst-address=2001:db8:1::1/64
[admin@MikroTik] /ipv6 firewall filter> move 12 11
```

33.10 IPv6 应用实例

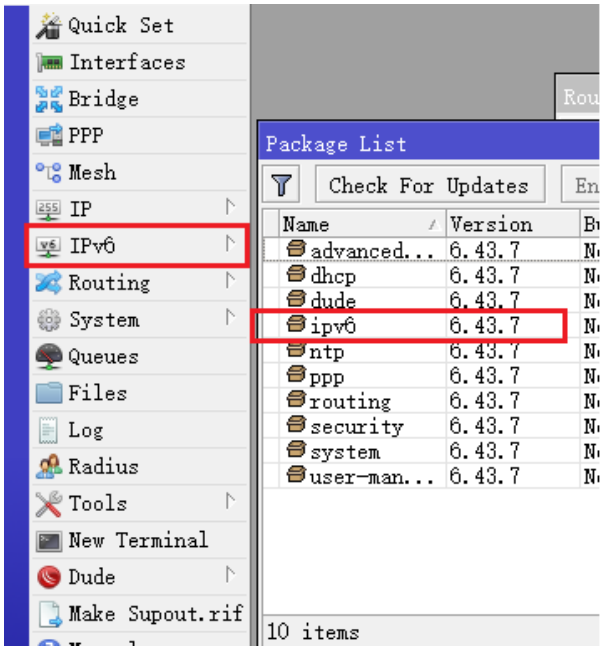
家宽获取并分配 IPv6 地址

当前三大运营商已经在给家庭宽带分配 IPv6 地址，如何通过 RouterOS 路由器获取并分配给内网的主机使用，这里简单说明下

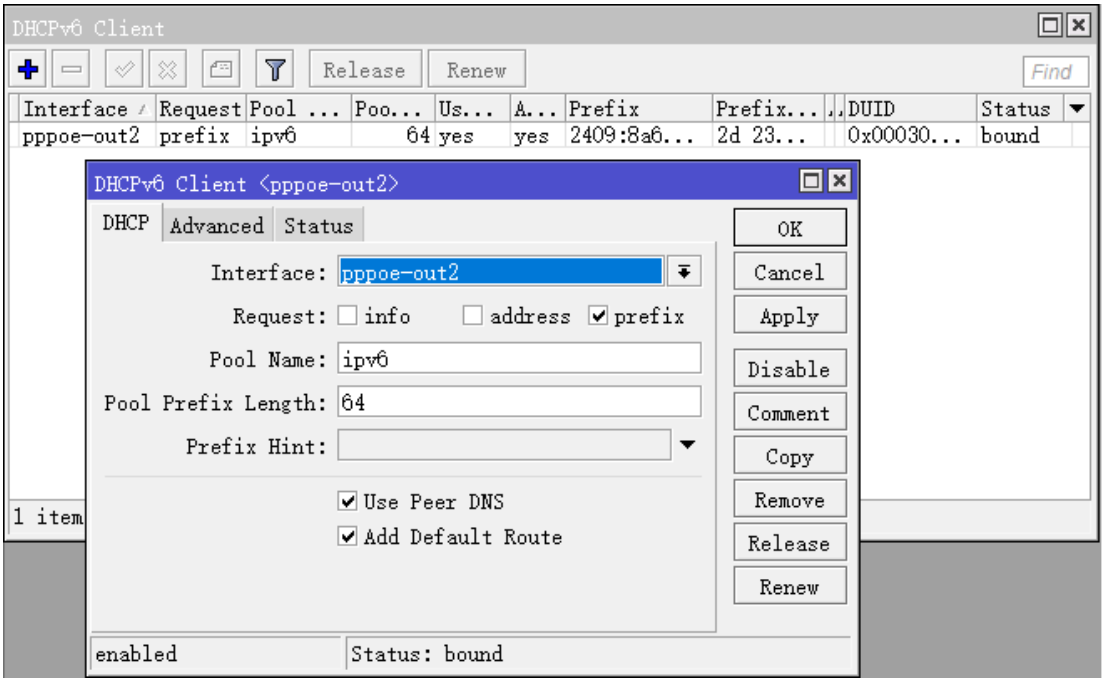
首先需要确认你的 RouterOS 已经安装并启用了 IPv6 功能包，MikroTik 硬件的 RouterOS 都带有 IPv6 功能包，但默认情况下是 disabled 状态，需要在 system package 里面选择 enable 启用，并重启路由器



启用完成后，在 winbox 左侧菜单会出现 IPv6 选项



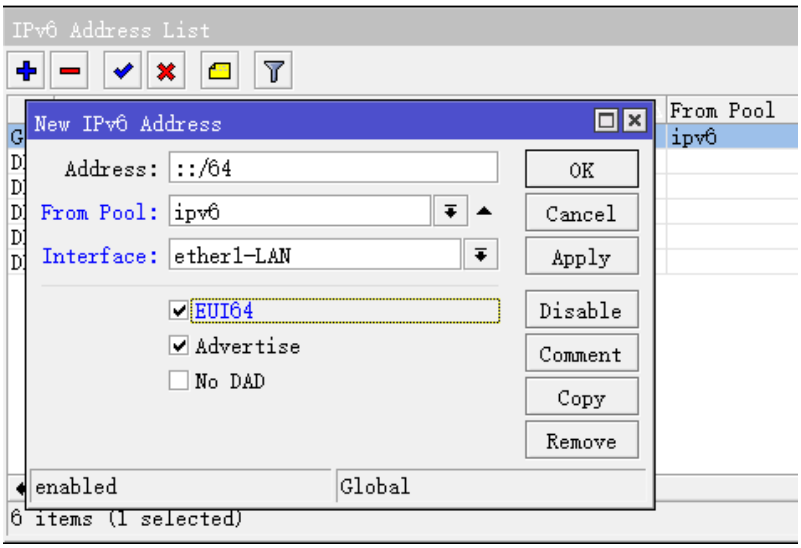
下面我们以家庭宽带是路由器拨号为例（非光猫拨号），RouterOS 已经通过 PPPoE 完成拨号上网，进入的 IPv6->DHCPv6-client 菜单，添加 pppoe-out2 的拨号接口，请求方式选择 prefix，将获取的 IPv6 地址，添加到“ipv6“地址池中，地址池前缀长度为 64（Pool Prefix Length），添加运营商的 DNS（Use Peer DNS）和默认路由（Add Default Route）：



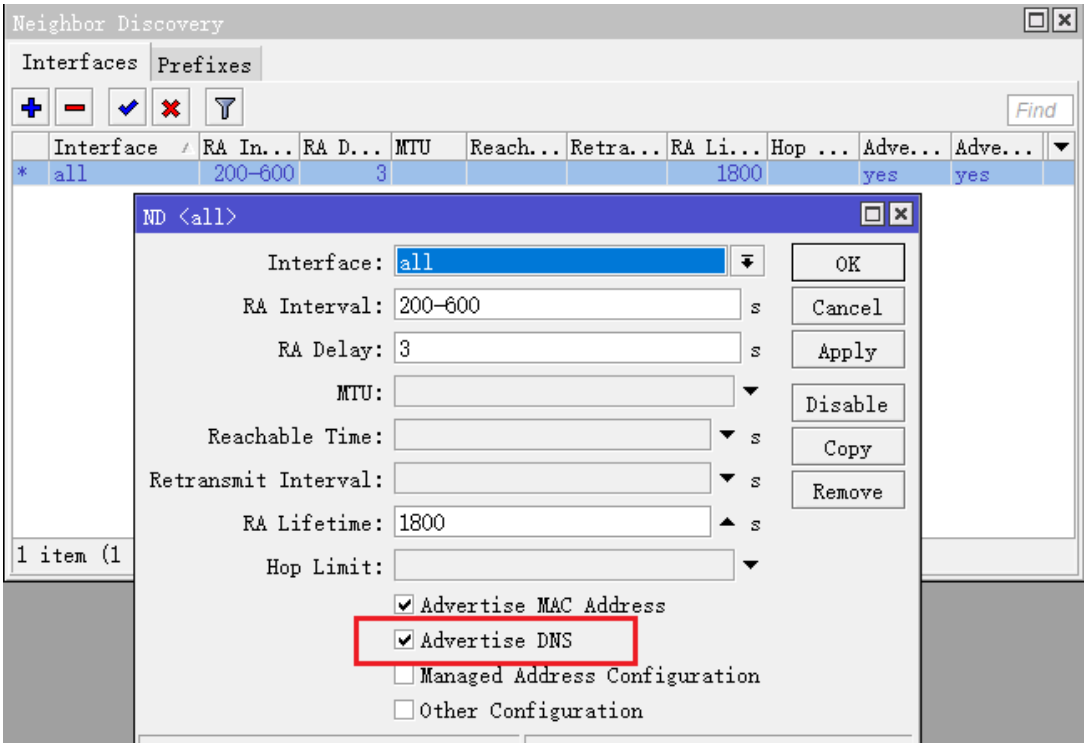
添加完成后，当 Status：bound，说明 IPv6 获取成功，如果没有则说明运营商未分配 IPv6。

下面是为内网分配 IP 地址，采用 ND 的无状态地址分配方式为内网主机分配 IPv6 地址，

进入 IPv6->address 菜单，添加一个地址项，Address 参数默认即可，From pool 为刚才的 ipv6 地址池，interface=ether1-LAN 内网口，启用 EUI64（IPv6 地址生成通过网卡 MAC 地址），当然也可以不选择 EUI64，根据具体需求。



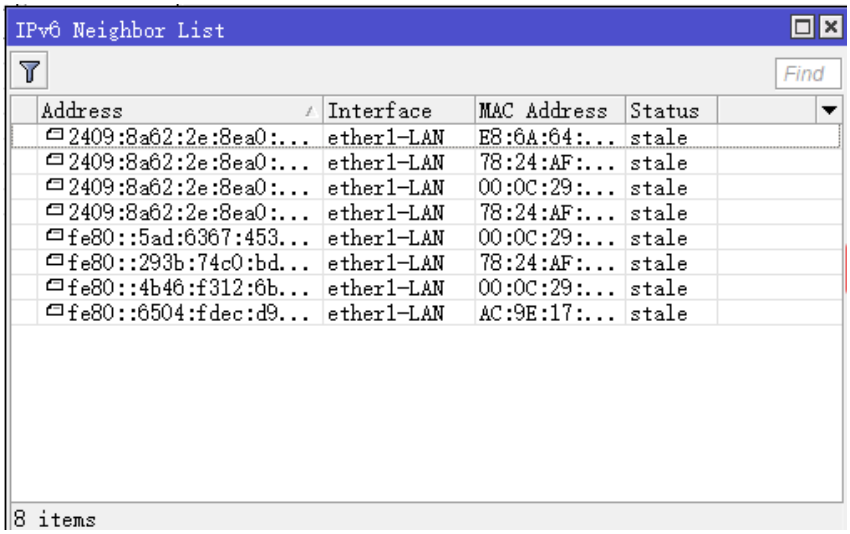
ND 分配方式，确保 IPv6->ND 菜单下的 Advertise DNS 勾选，用于 ND 为主机的 DNS 分配



剩下的就是确保 windows 电脑的 internet 协议版本 6（TCP/IPv6）已经勾选

然后查看 windows 电脑获取的 IPv6 地址信息

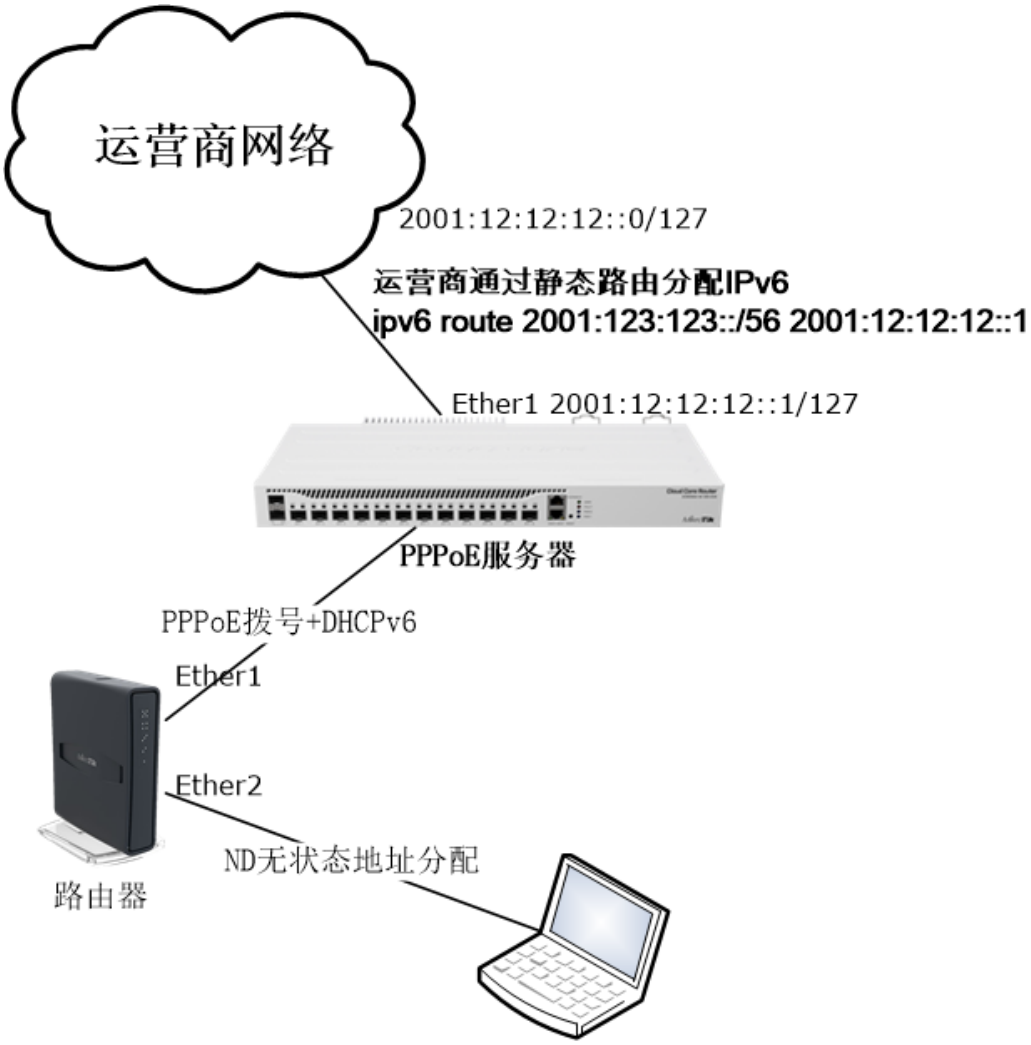
在 IPv6 Neighbor 列表查看当前分配的地址的情况



实现 PPPoE 服务器 IPv4 和 IPv6 双栈网络配置

三大运营当前为家庭宽带分配 IPv4 和 IPv6 双栈方式是 PPPoE+DHCPv6，如果要自己网络建立这套 PPPoE+DHCPv6 运营系统在 RouterOS 也可以实现，即为客户建立 PPPoE 认证服务，同时通过 PPPoE 隧道分配 IPv4 地址上网，又通过 DHCPv6 为当前客户分配 IPv6 地址，建立 IPv4 和 IPv6 双栈网络。

IPv4 网络接入从运营商方式这里忽略，主要讲解 IPv6 网络部署，从运营商通过静态路由方式获取 IPv6 地址：2001:123:123::/56，需要分配给网内 PPPoE 拨号客户，网络拓扑如下：



PPPoE 服务器配置：

IPv4 与运营商网络配置省略，主要介绍配置和运营商的 IPv6 互联网，采用的是静态地址分配，需取消 advertise 宣告

```
/ipv6 address
add address=2001:12:12:12::1/127 interface=ether1 advertise=no disabled=no
```

提示：如果运营商分配/127 的互联 IP，当前 RouterOS IPv6 不支持/127 的前缀，只能使用/126 解决！

添加 IPv6 默认路由

```
/ipv6 route
add gateway=2001:12:12:12::0 disabled=no
```

创建分配给客户的 IPv4 地址池

```
/ip pool
add name=pppoe ranges=192.168.9.2-192.168.9.10 disabled=no
```

创建 IPv6 地址池

```
/ipv6 pool
add name=my prefix=2001:123:123::/56 prefix-length=64 disabled=no
```

在 PPP profile 菜单下，新建一个 pppoe 策略，设置 dhcpv6-pd-pool 地址池为 my，其他 IPv4 参数如下：

```
/ppp profile
add dhcpv6-pd-pool=my dns-server=114.114.114.114 local-address=192.168.9.1 name=pppoe
remote-address=pppoe use-encryption=yes disabled=no
```

在 ip dns 下设置 IPv6 DNS 服务器地址

```
/ip dns
set servers=2400:3200::1
```

在 ppp secret 菜单下创建客户账号，账号 yus，密码 yus，并选择 pppoe 的 profile 策略

```
/ppp secret
add name=yus password=yus profile=pppoe disabled=no
```

配置并启用 PPPoE 服务器到 bridge 接口

```
/interface pppoe-server server
add default-profile=pppoe disabled=no interface=bridge service-name=service1
```

客户路由器配置

在 interface 新建一个 pppoe-client 拨号接口，设置如下：

```
/interface pppoe-client
add add-default-route=yes disabled=no interface=ether3 name=pppoe-out1 user=yus password=yus
```

在 ipv6 dhcp-client 菜单下 DHCP 客户端添加获取 IPv6 地址，并设置地址池前缀分配长度为 64

```
/ipv6 dhcp-client
add add-default-route=yes interface=pppoe-out1 pool-name=my pool-prefix-length=64 request=prefix
disabled=no
```

进入 ipv6 address 菜单下，创建通过 ND 无状态地址分配，配置如下：

```
/ipv6 address
add address=::1/64 from-pool=my interface=ether2 advertise=yes disabled=no
```

这样一个完整的 IPv4 和 IPv6 双栈网络配置完成。

注意：运营商分配的是/56 长度，下发给客户端路由器的是/64 长度（2^64 个地址，约 16 亿亿个 IPv6 地址），一共可以分配 2^8 个客户端，也就是 256 个，因为要使用 ND 无状态地址分配，只能匹配前缀是 64，如果你能从运营商分配到/48 的 IPv6 前缀，就有 2^16 地址，这个实例仅供参考。

33.11 IPv6 Setting

操作路径: /ipv6 settings

IPv6 Settings 菜单下，允许配置一些相关的 IPv6 内核参数

警告: 如果更改了设置，新配置不会立刻将 SLAAC 地址分配给路由器，而旧配置仍然生效，因此需重启路由器，或者等到 SLAAC 地址超时

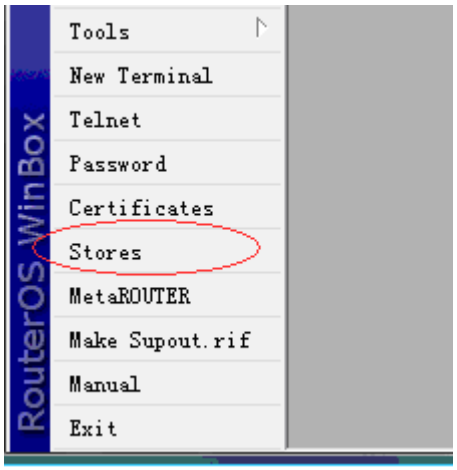
属性描述

属性	描述
forward (*yes* | *no*; 默认: **yes**)	启用或禁用接口间的数据包转发
accept-redirects (*no*| *yes-if-forwarding-disabled*; 默认 :**yes-if-forwarding-disabled**)	是否接受 ICMP 重定向消息。通常应在主机上启用，而在路由器上禁用该参数
accept-router-advertisements (*no* | *yes* | *yes-if-forwarding-disabled*; 默认: **yes-if-forwarding-disabled**)	接受路由器广播(RA)消息。如果启用，路由器获得地址方式，将使用无状态地址配置

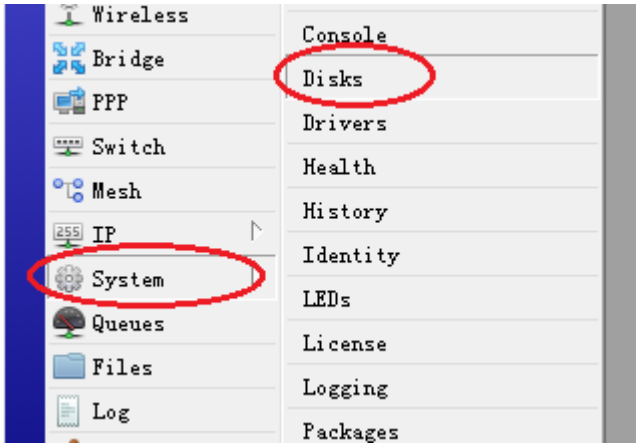
第三十四章 RouterOS Stores/disk 功能

RouterOS 在 3.15 后增加了 stores 存储功能，支持各种本地系统存储和外部设备存储，主要应用于 Web-proxy、User-Manager 和 The Dude 数据存储，在 3.23 后由于 RouterOS 支持 log 日志的本地存储，所以 Store 的应用有所增加，在 6.20 后 stores 菜单被 disk 替换，在 winbox 被归属于 system 目录下，在 terminal 仍然在根目录下。

除了 RouterOS 使用本地系统盘存储外，我们可以在 PC 或者 RouterBOARD 上增加各种存储设备，比如 RouterBOARD 可以选择 CF/MircoSD 方式存储，而 PC 可以选择增加硬盘、U 盘等方式。我们进入 winbox 后可以选择 store 目录，进入存储管理，



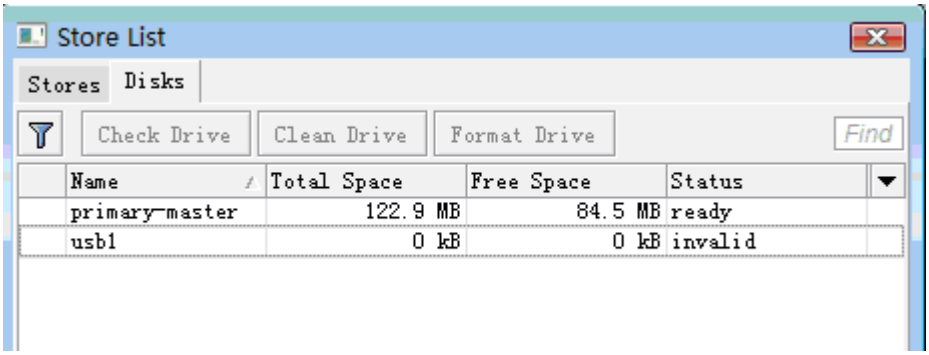
6.20 以前



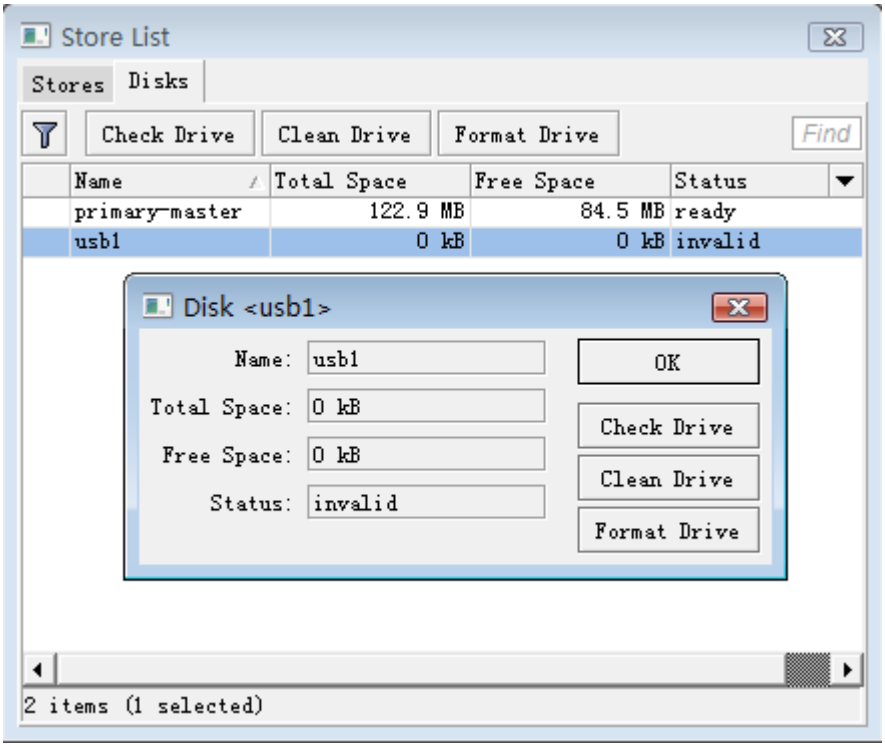
6.20 以后

34.1 RouterOS 使用 U 盘扩展存储

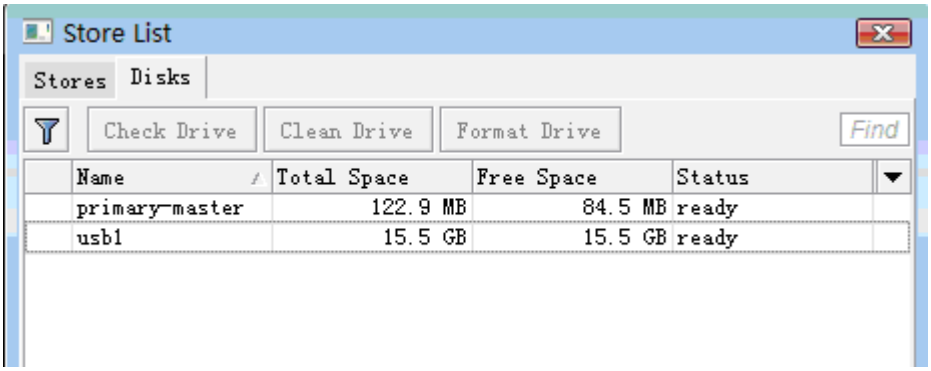
这里我们通过使用 U 盘来演示，在 PC 上增加外部存储的操作，我们将 1 个 16G 的 U 盘，插入 RouterOS PC 的 USB 接口，这个时候，我们可以在 Store 的 Disk 目录中找到 usb1 的硬盘信息：



当前状态为 invalid，即无法识别，因为 RouterOS 的硬盘分区和我们常用的 U 盘分区不同，所以我们需要选择 usb1，对 U 盘做格式化操作，选择 Format Driver 的选项



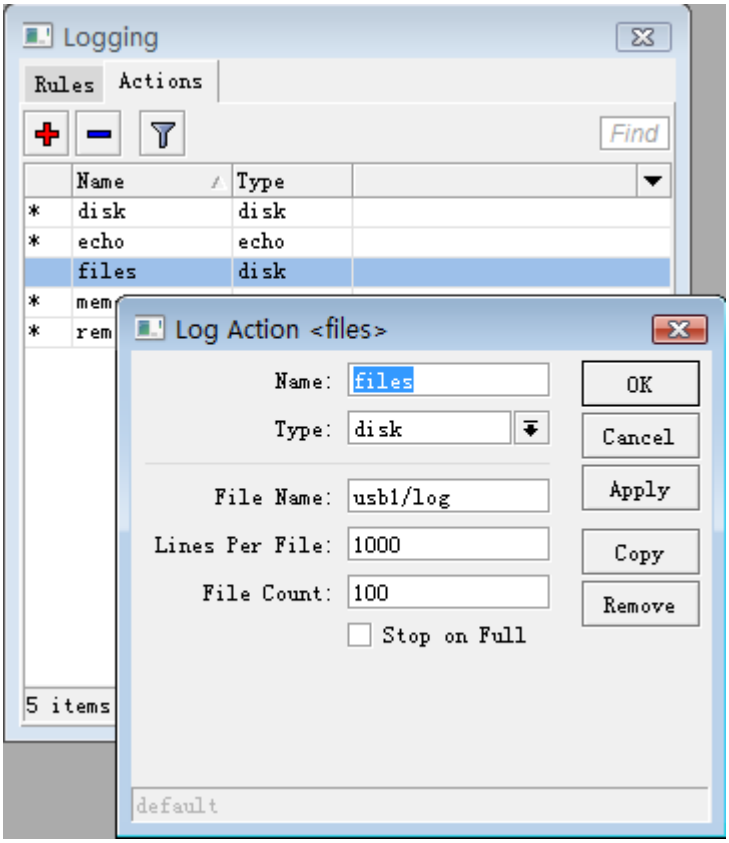
在格式化完成后，我们可以看到当前的 usb1 状态为 ready，能够正常识别到容量和空闲存储空间：



34.2 存储 log 日志信息

在 RouterOS3.23 后增加了可以将 log 日志存储到 RouterOS 上的存储设备里，由于本地系统存储空间有限，我们可以通过外部存储的 U 盘扩展，这里我们通过我们可以使用日志记录

首先我进入 system logging 配置 Action，并新建立一个 files 规则，并定义存储方式 type 为 disk：

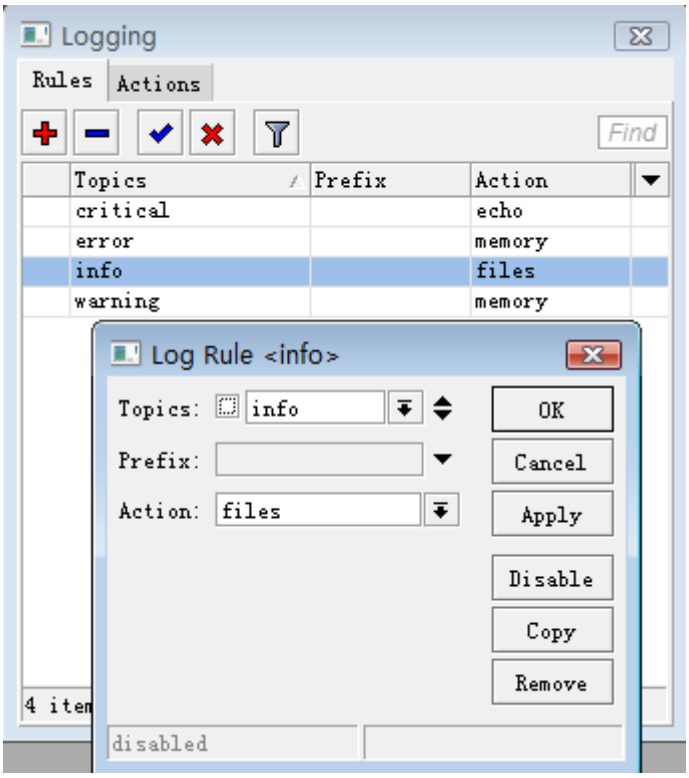


Disk 类型几个参数如下：

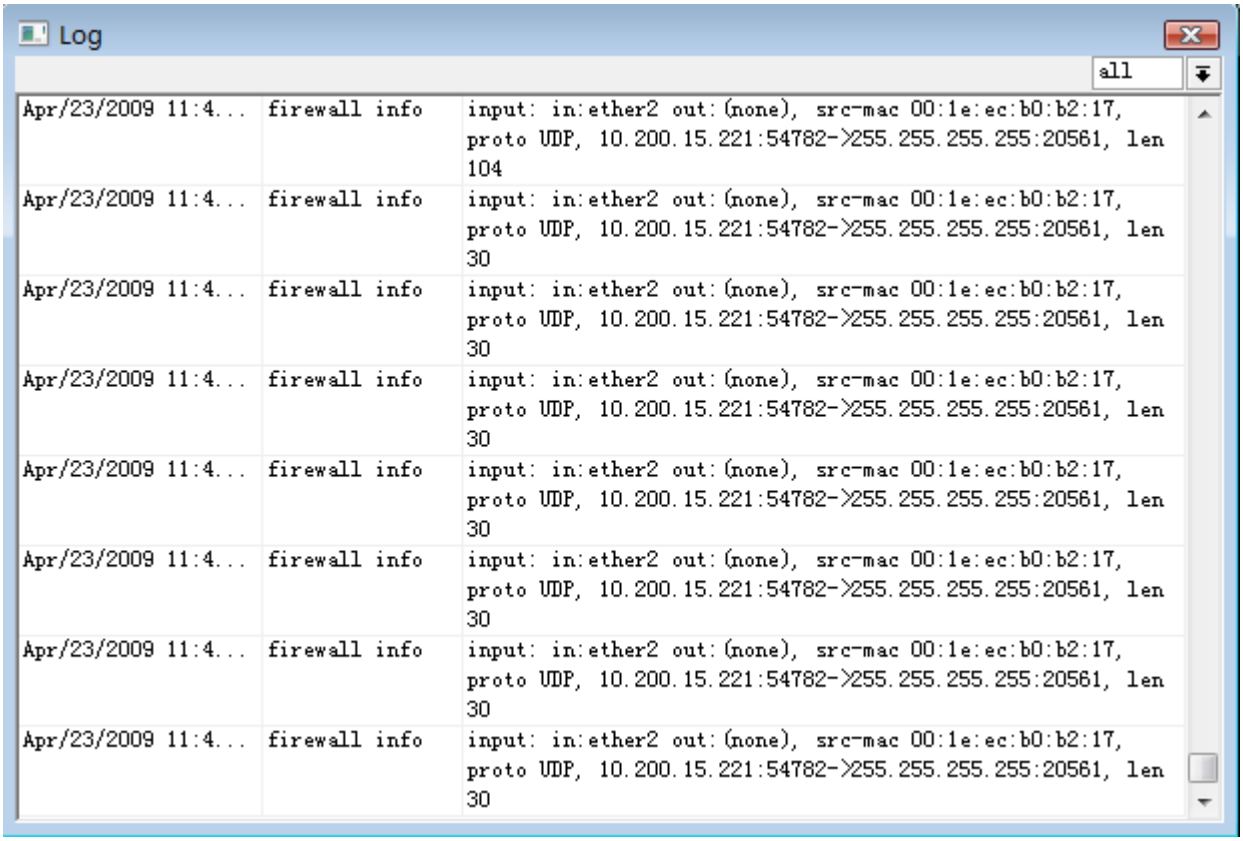
- **Type：**log 日志记录方式，这里我们选择 disk
- **File Name：**文件存储的路径，如果是 usb1 的 U 盘，我给的路径是 usb1/log
- **Lines Per File：**每个文件记录多少条信息
- **File Count：**log 日志一共建立多少个文件，如果日志记录超出文件数量，将会从 log0 号从头开始记录并覆盖原来的文件
- **Stop on Full：**当 log 建立的文件到达后，停止向文件写入 log 日志

注：文件名建立的原则，即**<filename>.0.txt**，**<filename>.1.txt**，**<filename>.n.txt** 的顺序建立，文件的大小可以自行定义。

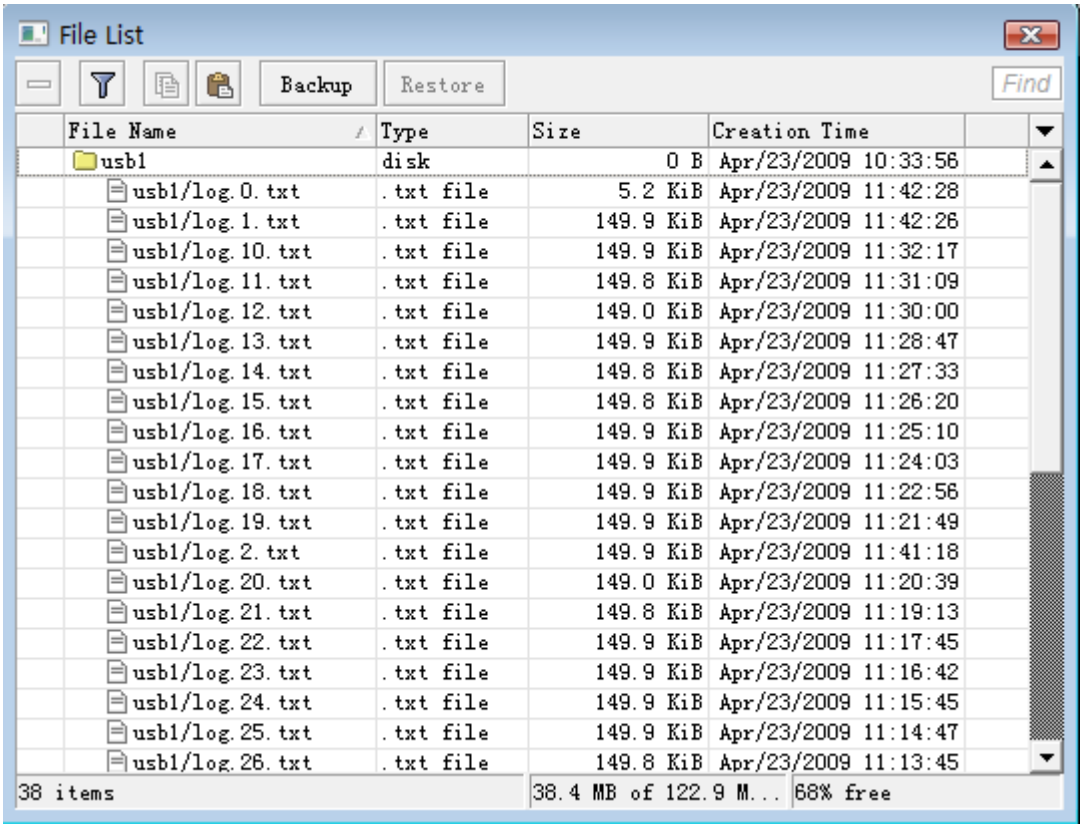
设置 logging 的 info（信息记录）为 files 操作，即记录到 usb1 中



我们可以看到在 log 中记录的防火墙信息



在 file list 中可以看到 usb1 中建立的 log 文件，文件以 txt 文本文件存储

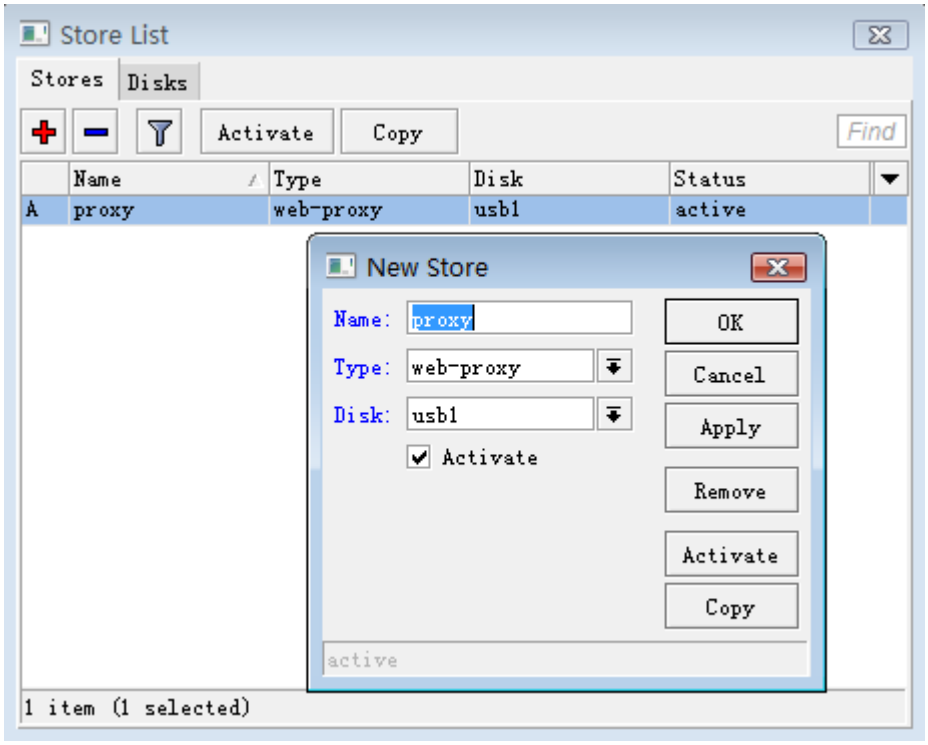


注：当在记录大量的日志信息，使用本地存储设备写入数据时，会出现 CPU 占用较大的情况，需要注意合理分配你的系统资源，建议尽量做远程日志记录的存储。

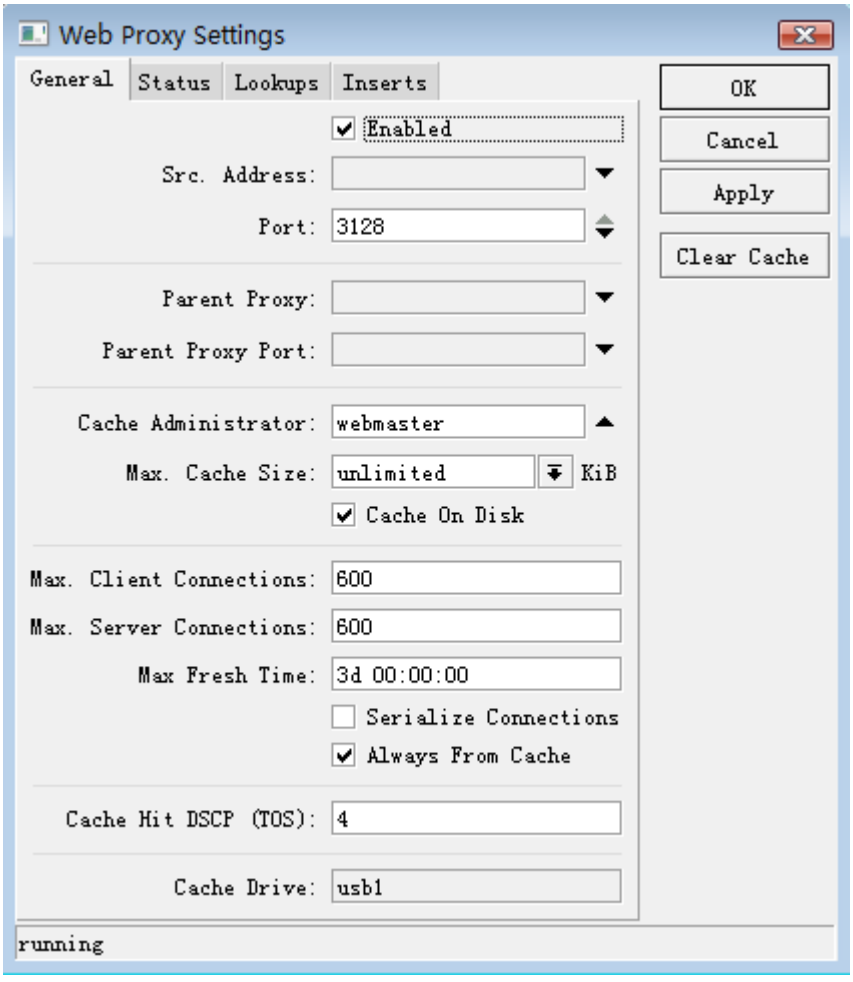
34.3 Web-Proxy 使用 U 盘存储

一些特殊网络环境，可能会用到 Web 缓存功能，需要将访问过的静态页面缓存到硬盘中，做二次访问。由于系统盘空间有限，我们可以使用 U 盘做为网页数据的外部存储。

下面在 Store 目录下添加一个名 Proxy 的规则，选择类型为 web-proxy，指定硬盘为 usb1，但 web-proxy 建议不要使用 U 盘存储，尽量使用 SSD 存储。

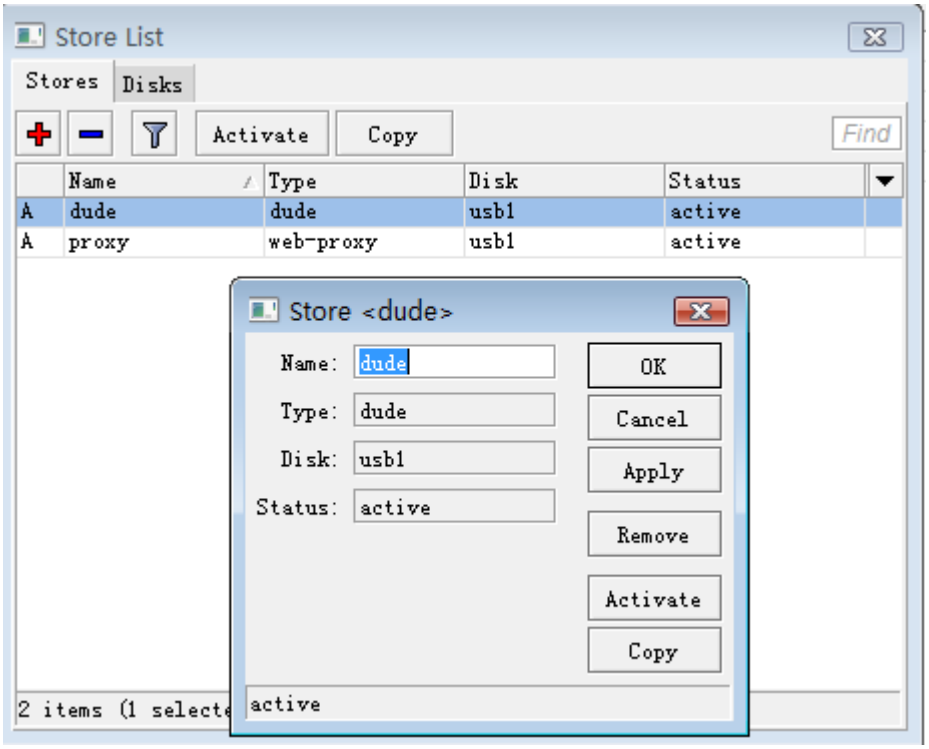


设置 Web-Proxy 的配置，这里可以看到 Cache Drive 会根据 Store 的配置，调用 usb1 的外部存储



34.4 Store/disk 的其他应用

Store 可以建立 The Dude 网络管理器的拓扑结构图的数据存储，如下图



Store 功能也可以用户存储 User-Manager 的数据库存储，能建立 User-Manager 的数据库。

第三十五章 UPNP

MikroTik RouterOS 支持即插即用（Universal Plug and Play），通用即插即用 (UPnP) 是一种用于 PC 计算机、智能设备和其他网络设备的常见对等网络连接的体系结构，即建立透明的点对点网络连接。

BitComet 官方网站对 UPnP 的解释：UPnP 通用即插即用，是一组协议的统称，不能简单理解为 UPnP 等于自动端口映射。在 BitComet 下载中，UPnP 包含了两层意思：

1. 对于一台内网计算机，BitComet 的 UPnP 功能可以使网关或路由器的 NAT 模块做自动端口映射，将 BitComet 监听的端口从网关或路由器映射到内网计算机上。
2. 网关或路由器的网络防火墙模块开始对 Internet 上其他计算机开放这个端口。

UPnP 在一个网络下，通过配置启用数据连接在两个设备，UPnP 不需要依靠特定的硬件支持，如果一个设备能动态的加入网络（DHCP 和 DNS 服务器正确配置），且能正常接入网络。UPnP 能实现简单的 NAT 穿透方案，能让客户端从 NAT 后获得双向东 P2P 网络。

操作路径: /ip upnp

在 RouterOS 配置中，有两种接口类型用于 UPnP：internal (内部接口，用于本地客户端连接) 和 external (外部接口，用于互联网连接)。通常一台路由器可能仅有一个外部接口和一个公网 IP 地址，且可能有多个内部接口，通过 src-nat 翻译内部 IP 上网。

属性描述

属性	描述
allow-disable-external-interface (*yes* | *no* ; 默认: **yes**)	是否允许用户禁用路由器的外部接口，该功能(对于用户在不需要任何验证程序的情况下，能关闭路由器的外部接口)是要求的标准，但事实上在 UPnP 开发中并未达到预期，原本设计为家庭用户建立自己本地网络，因此你可以选择禁用
enabled (*yes* | *no* ; 默认: **no**)	启用 UPnP 服务
show-dummy-rule (*yes* | *no* ; Default: **yes**)	启用一个工作区为作为损坏设施，处理不存在的 UPnP 规则，例如突然出现的错误信息

警告: 如果你没有禁用 allow-disable-external-interface, 任何本地用户能从本地网络在没有验证程序下禁用路由器的外部接口

35.1 UPNP 接口

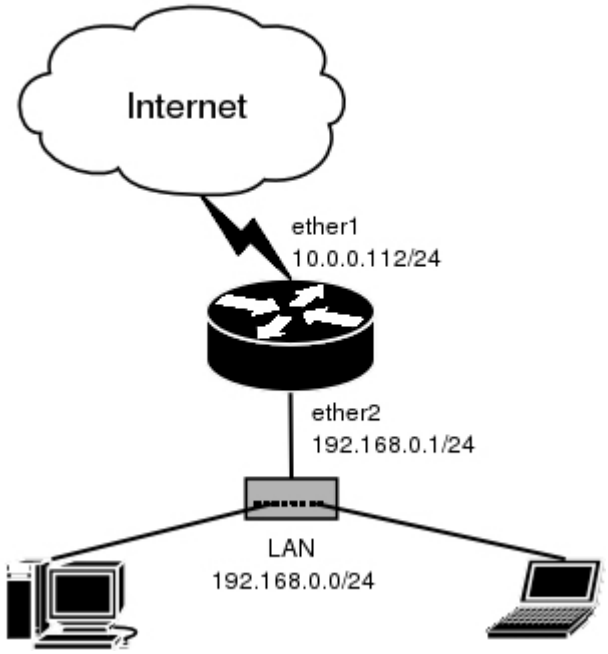
UPNP 接口用于定义路由器的公网和内网接口

操作路径: /ip upnp interfaces

属性	描述
interface (*字符*; 默认:)	选择运行 UPnP 的接口名称
type (*external* | *internal*; 默认: **no**)	uPnP 接口类型： • external – 分配路由器的公网接口和相应 IP 地址 • internal – 分配路由器的本地网络的接口
forced-external-ip (*Ip*; 默认:)	当外部接口被指定，且有多个 IP 地址，允许指定那一个公网 IP 被使用

35.2 配置事例

参考下面的一个简单的网络，外网接口是 ether1（IP 地址 10.0.0.112/24），内网接口是 ether2（IP 地址 192.168.0.0/24）



在正确配置网络后，确定 src-nat 伪装策略已经配置：

```
[admin@MikroTik] ip upnp> /ip firewall src-nat print
Flags: X - disabled, I - invalid, D - dynamic
  0    chain=srcnat action=masquerade out-interface=ether1
[admin@MikroTik] ip upnp>
```

进入 ip upnp 启用 UPnP 服务：

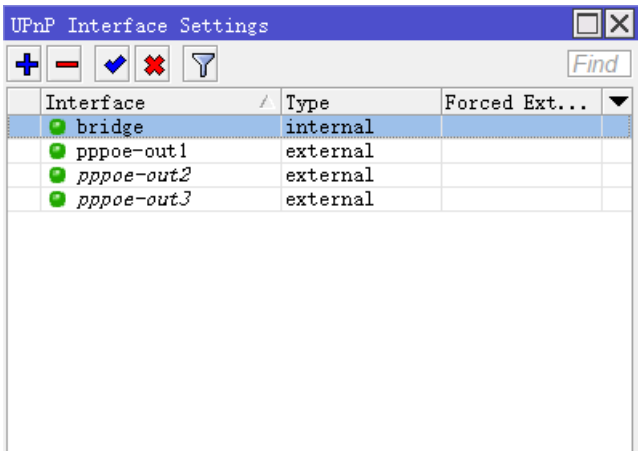
```
[admin@MikroTik] ip upnp> set enable=yes
[admin@MikroTik] ip upnp> print
```

```
                                    enabled: yes
      allow-disable-external-interface: yes
                                show-dummy-rule: yes
[admin@MikroTik] ip upnp>
```

配置相应的接口，设置外部接口和内部接口，ether1 为外部接口，ether2 为内部接口

```
[admin@MikroTik] ip upnp interfaces> add interface=ether1 type=external
[admin@MikroTik] ip upnp interfaces> add interface=ether2 type=internal
[admin@MikroTik] ip upnp interfaces> print
Flags: X - disabled
  #     INTERFACE TYPE
  0 X ether1       external
  1 X ether2       internal

[admin@MikroTik] ip upnp interfaces> enable 0,1
[admin@MikroTik] ip upnp interfaces>
```

Winbox 操作，如果你是光猫拨号的 pppoe-out1 接口，就选择 external 外部 interface 为 pppoe-out1，如下：



第三十六章 RouterOS 工具

36.1 Netwatch 监控

Netwatch 工具通过 ping 监控网络中的主机，并能通过状态的改变产生定义的事件。

规格

需要功能包: ***advanced-tools***
等级: *Level1*
操作路径: ***/tool netwatch***

Netwatch 监控的是在网络上的主机状态。 通过在列表中指定 IP 地址，并发送间隔的 ICMP 的 ping 探测和执行控制脚本。在主机状态改变时根据 netwatch 的情况下命令。

属性描述

属性	描述
down-script (*字符*; 默认:)	当一个被监控主机状态变为 down 时，执行一次脚本
host (*IP*; 默认: **0.0.0.0**)	通过 ping 监控主机的 IP
interval (*时间*; 默认: **1m**)	监控主机的时间间隔周期，默认为 1 分钟，如果减少时间周期，会增加系统消耗和流量，需根据设备和网络情况调整.
timeout (*时间*; 默认: **1s**)	Ping Timeout 时间，默认大于 1000ms 就被视为 down 状态，也可以根据需要调整
up-script (*string*; Default:)	当一个被监控主机状态变为 up 时，执行一次脚本

警告:从 RouterOS v6.42 Netwatch 被限制为 read,write,test,reboot 脚本策略，如果登录账号脚本权限不够，这时脚本无法执行。如果脚本的策略大于 read,write,test,reboot，脚本也不会被执行，请确保你的脚本不会超过上述策略。

如果权限问题，可在/system scripts 菜单下禁用 RouterOS 脚本的权限检查，当 Netwatch 没有足够的权限执行脚本时，可使用这种方法，但会降低整体安全性，因此建议为脚本分配适当的权限，提高安全性。

事例

这个事例将运行脚本 gw_1 或 gw_2 根据网关的状态来修改默认网关：

```
[admin@MikroTik] system script> add name=gw_1 source="/ip route set [find dst-address 0.0.0.0]
gateway=10.0.0.1"
[admin@MikroTik] system script> add name=gw_2 source="/ip route set [find dst-address 0.0.0.0]
gateway=10.0.0.217"
[admin@MikroTik] system script> /tool netwatch
```

```
[admin@MikroTik] tool netwatch> add host=10.0.0.217 interval=10s timeout=800ms up-script=gw_2
down-script=gw_1
[admin@MikroTik] tool netwatch> print
Flags: X - disabled
  #    HOST            TIMEOUT          INTERVAL         STATUS
  0    10.0.0.217      997ms            10s              up
[admin@MikroTik] tool netwatch> print detail
Flags: X - disabled
  0    host=10.0.0.217 timeout=997ms interval=10s since=feb/27/2003 14:01:03
       status=up up-script=gw_2 down-script=gw_1

[admin@MikroTik] tool netwatch>
```

让我们来看上面的例子，如果当前默认网关变为无法到达，就改变默认路由。有两个脚本，当主机状态改变为 **up** 脚本"gw_2"执行一次。在这个事例中，相当于进入控制台执行下面的命令：

```
[admin@MikroTik] > /ip route set [/ip route find dst 0.0.0.0] gateway 10.0.0.217
```

/ip route find dst 0.0.0.0 命令是返回在路由表中 **dst-address** 值为 **0.0.0.0** 的参数，通常这种值确认为默认路由条目。

当主机状态改变为 **down** 脚本"gw_1"执行一次。如下面：

```
[admin@MikroTik] > /ip route set [/ip route find dst 0.0.0.0] gateway 10.0.0.1
```

如果 10.0.0.217 地址无法到达，改变默认网关。

下面是另一个事例，无论什么时候 10.0.0.215 主机断线，发送 e-mail 通知到你指定的邮箱：

```
[admin@MikroTik] system script> add name=e-down source="/tool e-mail send    from="rieks@mt.lv"
server="159.148.147.198" body="Router down" subject="Router at second floor is down" to="rieks@latnet.lv" "
[admin@MikroTik]
[admin@MikroTik] system script> add name=e-up source="/tool e-mail send from="rieks@mt.lv"
server="159.148.147.198" body="Router up" subject="Router at second floor is up" to="rieks@latnet.lv" "
[admin@MikroTik] system script>
[admin@MikroTik] system script> /tool netwatch
[admin@MikroTik] system netwatch> add host=10.0.0.215 timeout=999ms \
\... interval=20s up-script=e-up down-script=e-down
[admin@MikroTik] tool netwatch> print detail
Flags: X - disabled
  0    host=10.0.0.215 timeout=998ms interval=20s since=feb/27/2003 14:15:36
       status=up up-script=e-up down-script=e-down

[admin@MikroTik] tool netwatch>
```

关于脚本编写可以参考**《RouterOS 脚本编写与 WLAN 无线教程》**

36.2 绘图显示（Graphing）

Graphing 是 RouterOS 内置的一个简单图形化监管工具，用于 RouterOS 在一段时期内的接口流量，系统状态等情况，类似于 cacti 和 zabbix 一类的图像记录工具。

Graphing 只能粗略的记录流量情况，而无法进行有效的管理和数据提取。可以参考 cacti 的搭建，通过 snmp 对 RouterOS 设备的流量、系统资源和硬件信息进行采集和图形化管，内容如下

树莓派（RaspberryPi 3/4）基于 CentOS 搭建 Cacti1.2.16
Cacti1.2.16 安装向导设置
Cacti 1.2.16 添加 MikroTik RB 设备状态和流量图

需要功能包: ***system***,
等级需要: *Level1*
操作路径: ***/tool graphing***

Graphing 工具可以显示的图形为：

- Routerboard 健康状态（电压和温度)
- 资源使用（CPU，内存和硬盘使用 Disk usage)
- 通过 Interfaces 的传输情况
- simple queues 中的传输情况

Graphing 由两部分构成- 第一部分是收集数据信息，另一部分在一个 Web page 中显示数据访问图形的地址为 **http://[Router_IP_address]/graphs/** 或是通过浏览 RouterOS 的默认网页进入。

在路由器中数据收集每间隔 5 分钟，但保存到系统驱动中是每隔一个 **store-every** 时间，当重起路由后，显示的信息在重起前为最后一次存储到磁盘中的数据。

RouterOS 每一个项目产生四种图示 generates four graphics for each item:

- "Daily" Graph (5 Minute Average)
- "Weekly" Graph (30 Minute Average)
- "Monthly" Graph (2 Hour Average)
- "Yearly" Graph (1 Day Average)

从一个网络去访问每个图形，可以通过 **allow-address** 指定这个网络的访问项目。

操作路径: /tool graphing

属性描述

store-every (5min | hour | 24hours; 默认: **5min**) – 多长时间将信息存储到系统驱动上。

存储信息到系统驱动上为每小时：

```
/tool graphing set store-every=hour
[admin@MikroTik] tool graphing> print
```

```
    store-every: hour
[admin@MikroTik] tool graphing>
```

健康情况

操作路径: ***/tool graphing health***

这个子项目提供关于 RouterBoard 的电压和温度的信息，但你必须安装 **routerboard** 功能包和使用 RouterBoard：

属性描述

allow-address (*IP 地址*/子网; 默认: **0.0.0.0/0**) –设置允许访问图形的 IP 地址段，默认通过打开 **http://[Router_IP_address]/graphs/**
store-on-disk (yes | no; 默认: **yes**) – 是否将信息存储到系统存储上，如果选择为'no'，这些信息将存储到 RAM 中，重启后回丢失

接口流量图

操作路径: ***/tool graphing interface***

显示有多少流量传输在一段时期内通过了一个 interface

属性描述

allow-address (*IP 地址*/子网; 默认: **0.0.0.0/0**) –设置允许访问图形的 IP 地址段，默认通过打开 **http://[Router_IP_address]/graphs/**
interface (名称; 默认: **all**) – interface 的名称
store-on-disk (yes | no; 默认: **yes**) -是否将信息存储到系统存储上，如果选择为'no'，这些信息将存储到 RAM 中，重启后回丢失

仅 **192.168.0.0/24** 的网段监视通过 **ether1** 的传输情况，并将信息写入到磁盘中:

```
[admin@MikroTik] tool graphing interface> add interface=ether1
allow-address=192.168.0.0/24 store-on-disk=yes
[admin@MikroTik] tool graphing interface> print
Flags: X - disabled
 #    INTERFACE ALLOW-ADDRESS          STORE-ON-DISK
 0    ether1      192.168.0.0/24       yes
[admin@MikroTik] tool graphing interface>
```

流控图显示

操作路径: ***/tool graphing queue***

在这个子选项中可以指定一个队列**/queue simple** 到图形显示中去。

属性描述

allow-address (*IP 地址*/子网; 默认: **0.0.0.0/0**) -设置允许访问图形的 IP 地址段，默认通过打开 **http://[Router_IP_address]/graphs/**

allow-target (yes | no; 默认: **yes**) – 允许在**/queue simple target-address** 中那些 IP 段访问 graphing web
simple-queue (*名称*; 默认: **all**) – 要监测 的 simple queue 名称
store-on-disk (yes | no; 默认: **yes**) - 否将信息存储到系统存储上，如果选择为'no'，这些信息将存储到 RAM 中，重启后回丢失

添加一个 simple queue 到图形列表，simple-queue 名称为 **queue1**，限制访问网段，并存储相关信息到磁盘中:

```
[admin@MikroTik] tool graphing queue> add simple-queue=queue1 allow-address=192.168.0.0/24
store-on-disk=yes
```

资源图表

操作路径: ***/tool graphing resource***

提供路由器在一段时期内资源使用情况:

- CPU usage
- Memory usage
- Disk usage

属性描述

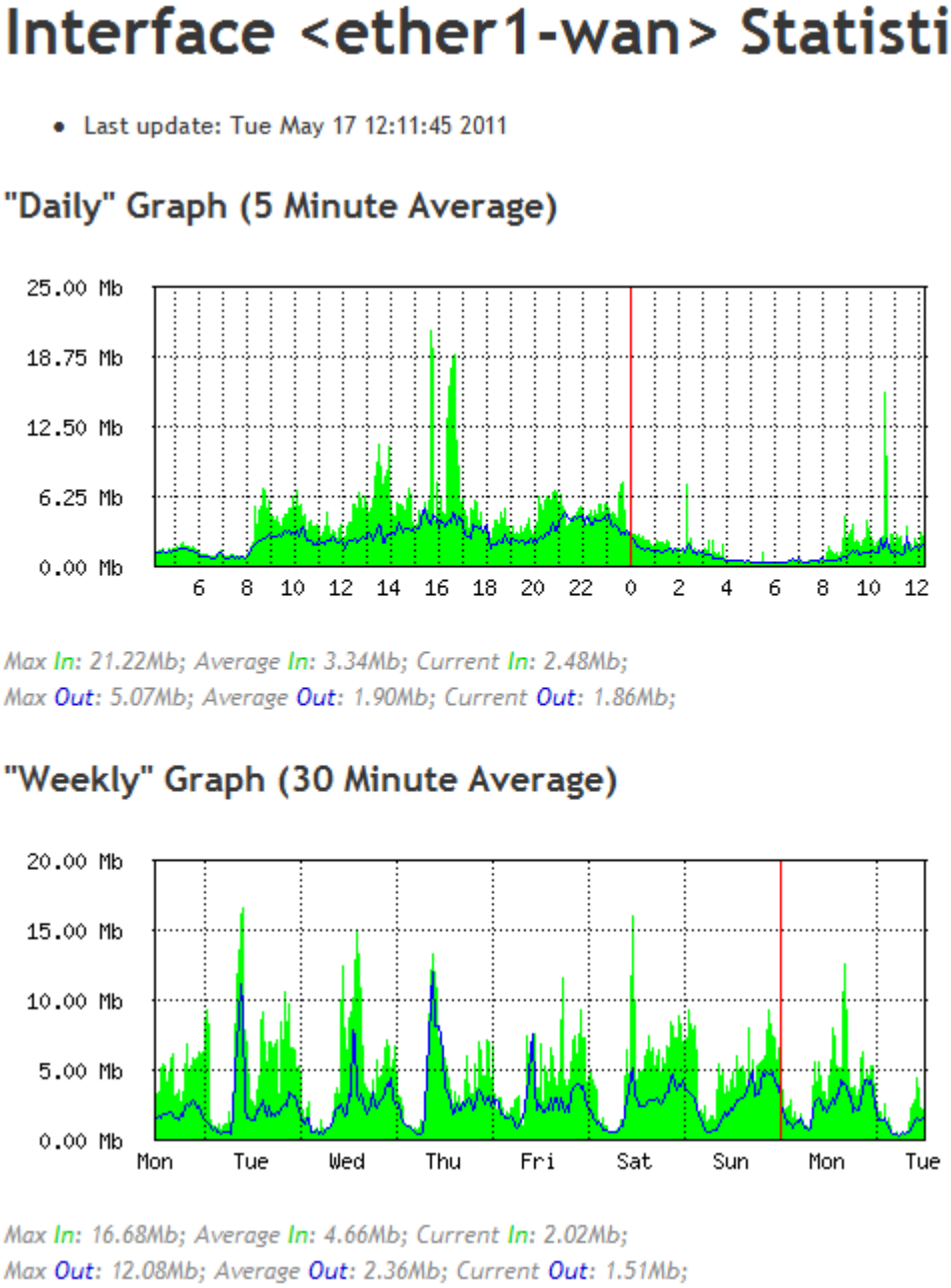
allow-address (*IP 地址*/子网; 默认: **0.0.0.0/0**) -设置允许访问图形的 IP 地址段，默认通过打开 **http://[Router_IP_address]/graphs/**
store-on-disk (yes | no; 默认: **yes**) -否将信息存储到系统存储上，如果选择为'no'，这些信息将存储到 RAM 中，重启后回丢失

添加允许监视者的 IP 地址段为 **192.168.0.0/24** :

```
[admin@MikroTik] tool graphing resource> add allow-address=192.168.0.0/24 store-on-disk=yes
[admin@MikroTik] tool graphing resource> print
Flags: X - disabled
 #    ALLOW-ADDRESS          STORE-ON-DISK
 0    192.168.0.0/24         yes
[admin@MikroTik] tool graphing resource>
```

我们可以通过 IP 地址登录 RouterOS 的 web 页面选择 Graphs 进入图形页面

我们可以选择一张网卡，查看流量统计



36.3 Bandwidth-test 带宽测试

带宽测试用于监测远程 MikroTik 路由器的吞吐量（有线或无线），从而去发现网络传输瓶颈。

协议属性

TCP 测试使用 TCP 协议标准，根据 TCP 算法得出有多少包延迟，被丢弃和其他 TCP 算法特性。关于内部速度设定和状态分析请查看 TCP 协议。吞吐量的统计是用来计算整个 TCP 数据流的大小。TCP 内部连接的大小和使用没有包含在吞吐量的统计中。因此当在测算吞吐量时，这个统计并不像 UDP 协议一样可靠。

UDP 测试发送的数据报的数量是接收方当前所收到包的数量的 110%或更多。要得到连接的最大吞吐量，数据报要设置最大 MTU 为 1500 字节。这并不是 UDP 协议标准所要求的。 通过这样设置，便可以得到近似最大吞吐量。

注：Bandwidth Test 会使用所有可获得的带宽(by default)，并做可能冲击网络的使用性。

Bandwidth Test 比较占用资源。如果需要测试路由器的真实吞吐量，你应该运行 bandwidth test 通过所测路由器。这样做你需要三台路由器相连：Bandwidth 服务器，测试路由器（Testing Router）和 Bandwidth 客户端：

注：如果用 UDP 协议，那么 Bandwidth Test 所测的数据是 IP header+UDP header+UDP。如果用 TCP 协议，那么 Bandwidth Test 所测的数据仅为 TCP 数据。（不包含 TCP 数据报头和 IP 数据报头）。

Server 配置

操作路径：***/tool bandwidth-server***

属性描述

allocate-udp-ports-from – 分配 UDP 端口
authenticate (yes | no; 默认: **yes**) – 通信要求验证客户端（通过账号和密码）
enable (yes | no; 默认: **no**) – 为客户端启用连接
max-sessions – bandwidth-test 最大的客户端连接数

Bandwidth 服务器：

```
[admin@MikroTik] tool bandwidth-server> print
                      enabled: yes
                 authenticate: yes
  allocate-udp-ports-from: 2000
                 max-sessions: 10
[admin@MikroTik] tool>
```

查看会话连接

```
[admin@MikroTik] tool> bandwidth-server session print
  # CLIENT              PROTOCOL DIRECTION USER
  0 35.35.35.1          udp      send      admin
  1 25.25.25.1          udp      send      admin
  2 36.36.36.1          udp      send      admin
[admin@MikroTik] tool>
```

开启没有客户端的 **bandwidth-test** 服务器

```
[admin@MikroTik] tool bandwidth-server> set enabled=yes authenticate=no
[admin@MikroTik] tool bandwidth-server> print
                      enabled: yes
```

```
                authenticate: no
  allocate-udp-ports-from: 2000
                max-sessions: 10
[admin@MikroTik] tool>
```

Client 配置

操作路径: ***/tool bandwidth-test***

属性描述

(*IP address*) - 目标主机 IP 地址
assume-lost-time (*时间*; 默认: **0s**) – 设定如果 Bandwidth Server 无响应多久后丢弃连接
direction (receive / transmit / both; 默认: **receive**) - 测试方式
do (名称 | string; 默认: **""**) - 脚本源代码
duration (*时间*; 默认: **0s**) - 测试时长
0s – 测试时间没有被限制
interval (*时间*: 20ms..5s; 默认: **1s**) – 报告间隔时间（秒钟计算）
local-tx-speed (*整型*; 默认: **0**) – 本地发送最大速率(bits per second)
0 – 没有速率限制
local-udp-tx-size (*整型*: 40..64000) – 本地 UDP 发送最大数据报
password (*文本*; 默认: **""**) – 测试的密码
protocol (udp | tcp; 默认: **udp**) – 使用的网络协议
random-data (yes | no; 默认: **no**) – 如果随即数据设置为 yes，Bandwidth 测试数据报的有效载荷，将有不可随机数据流，使连接利用数据压缩，将不会扭曲结果(如果较低性能的 CPU，**random-data** 应设置为 **no**)
remote-tx-speed (*整型*; 默认: **0**) – 远程接收测试的最大速率(bits per second)
0 – 没有速率限制
remote-udp-tx-size (*整型*: 40..64000) – 远程 UDP 发送最大数据报
user (*名称*; 默认: **""**) - 远程用户名

在 10.0.0.211 主机上运行 15 秒发送和接收 **1000**-byte UDP 数据报的带宽测试，用户名为 admin.

```
[admin@MikroTik] tool> bandwidth-test 10.0.0.211 duration=15s direction=both
\... size=1000 protocol=udp user=admin
                     status: done testing
                     duration: 15s
                  tx-current: 3.62Mbps
    tx-10-second-average: 3.87Mbps
        tx-total-average: 3.53Mbps
                  rx-current: 3.33Mbps
    rx-10-second-average: 3.68Mbps
        rx-total-average: 3.49Mbps
[admin@MikroTik] tool>
```

36.4 Torch (实时通信监听)

实时通信监听被称为 torch 它是用于监视正在运行的一个接口的网络通信流量情况，你可以监听通过该接口的协议、网络端口、源和目的地址，并可以通过分类监听。该功能有助于管理对接口网络情况的判断和指定 IP、协议和端口的分析。

操作路径: ***/tool torch***

属性描述

(*名称*) – 用于监视的接口名
dst-address (*IP address/netmask*) – 目的地址和子网掩码是用来通信，任意的目的地址是: 0.0.0.0/0 .
freeze-frame-interval (*时间*) - 屏幕输出暂停的立即时间
port (名称 | 整型) – 端口的名
protocol (any | any-ip | ddp | egp | encap | ggp | gre | hmp | icmp | idpr-cmtp | igmp | ipencap | ipip | ipsec-ah | ipsec-esp | iso-tp4 | ospf | pup | rdp | rspf | st | tcp | udp | vmtp | xns-idp | xtp) – 协议名
  any - 任何以太网和网络协议
  any-ip – 任何网络协议
src-address (*IP address/netmask*) – 源地址和子网掩码是用来进行通信，所有源地址是: 0.0.0.0/0

注：如果规定了一个特殊的端口,仅有 tcp 和 udp 协议将被过滤, 这就是说协议包含 any any-ip tcp udp.除了上行和下行, 你已经用命令指定输出(例如,你将得到协议镞仅是以防万一如果协议被明确指出).

下面的例子是利用 telnet 协议监视通过 **ether1** 接口的通信情况:

```
[admin@MikroTik] tool> torch ether1 port=telnet
 SRC-PORT                    DST-PORT                    TX          RX
 1439                        23 (telnet)                 1.7kbps     368bps

[admin@MikroTik] tool>
```

IP 协议通过 ether1 接口所显示的情况：

```
[admin@MikroTik] tool> torch ether1 protocol=any-ip
 PRO.. TX          RX
 tcp    1.06kbps   608bps
 udp    896bps     3.7kbps
 icmp   480bps     480bps
 ospf   0bps       192bps

[admin@MikroTik] tool>
```

IP 协议作用于 10.0.0.144/32 这台主机连接 ether1 接口所显示的情况：

```
[admin@MikroTik] tool> torch ether1 src-address=10.0.0.144/32 protocol=any
 PRO.. SRC-ADDRESS      TX          RX
 tcp    10.0.0.144      1.01kbps    608bps
 icmp   10.0.0.144      480bps      480bps

[admin@MikroTik] tool>
```

tcp/udp 协议通过 ether1 接口所显示的情况：

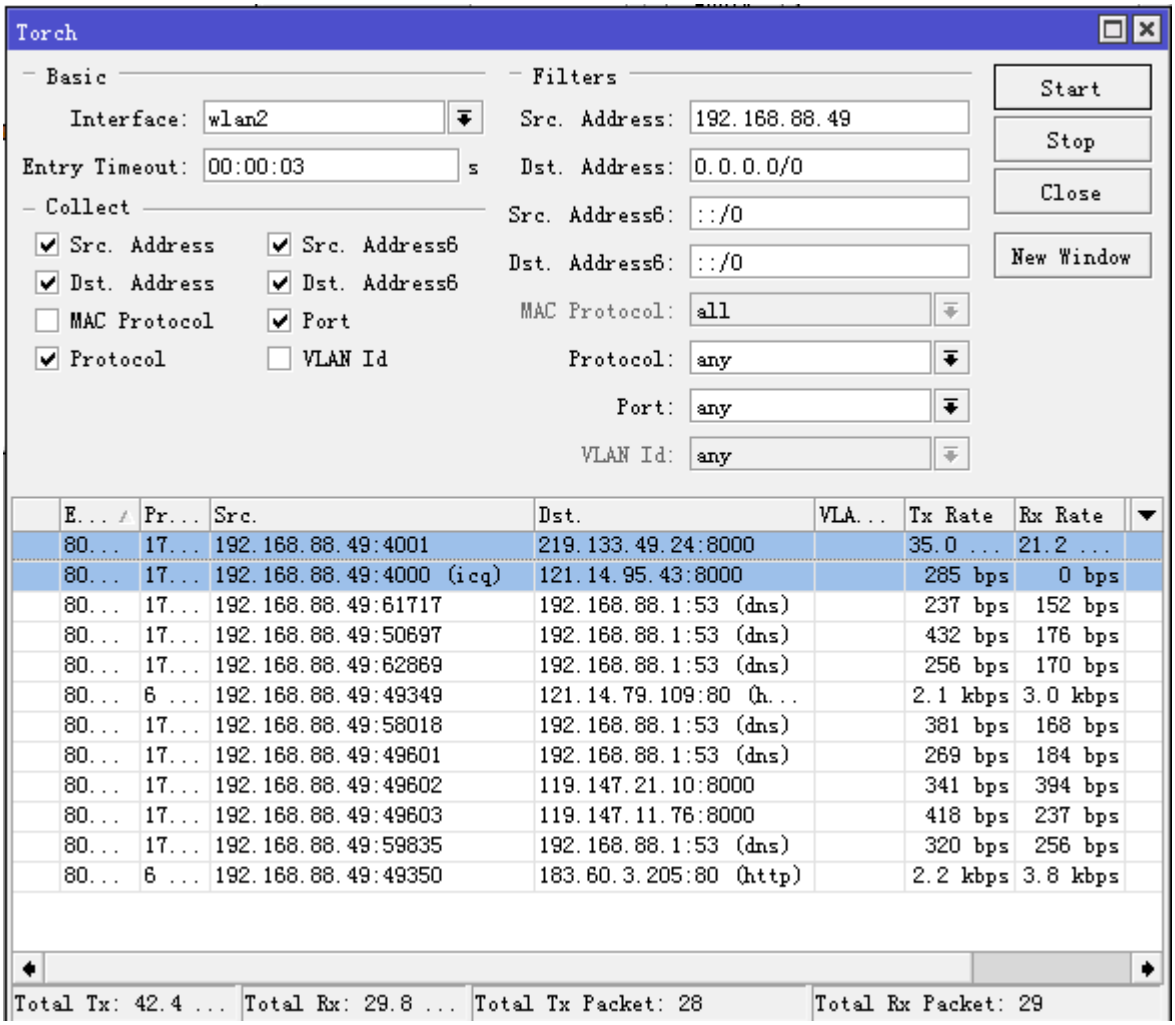
```
[admin@MikroTik] tool> torch ether1 protocol=any-ip port=any
 PRO.. SRC-PORT                DST-PORT                TX          RX
 tcp     3430                  22 (ssh)                1.06kbps    608bps
 udp     2812                  1813 (RADIUS-acct)      512bps      2.11kbps
 tcp     1059                  139 (netbios-ssn)       248bps      360bps
[admin@MikroTik] tool>
```

查看 QQ 连接服务器

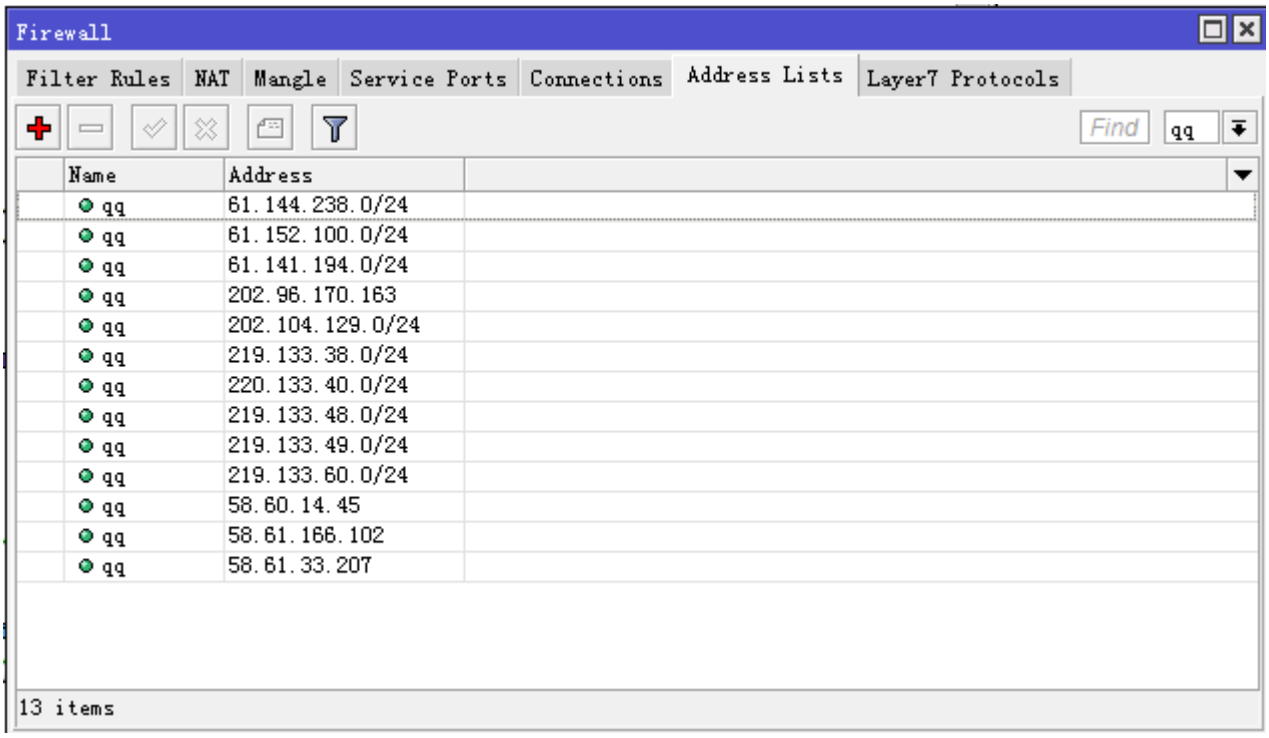
如果想知道 QQ 连接的服务器 IP 地址，可以通过 torch 来一一排查，如果是想控制 QQ 的连接访问，普通端口禁用没有意义，一般是封 IP 或使用 L7 协议。

首先我需要通过 RouterOS 的工具查找每次 QQ 连接服务器的 IP 地址，在这里我们使用的是 RouterOS 自带的 torch 工具，Torch 是用于监测相应网卡的数据连接状态、协议、端口和流量的工具，在/tool 中可以找到 Torch。

我们需要监测 interface 为内网网卡，QQ 连接通常使用 8000 端口连接服务器，我们通过查看 dst-port 为 8000 的连接，当 8000 端口无法连接时，QQ 会使用 80 或者 443 端口，如下图：



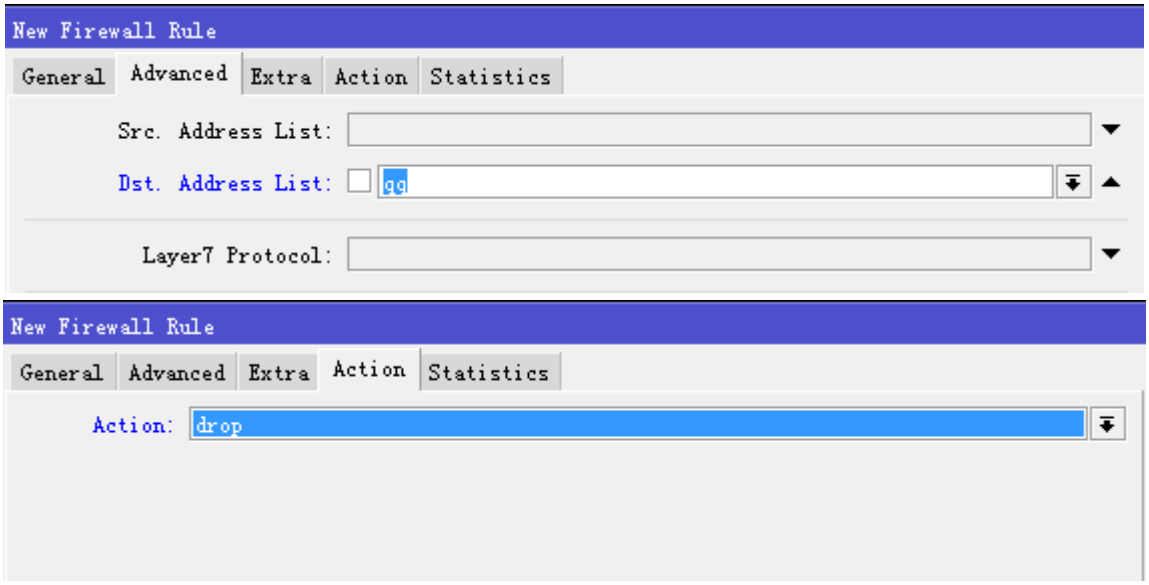
当看到 8000 端口的连接的 dst-address，可以判断为 QQ 的服务器，我通过 ip firewall address-list 填写 QQ 服务器的IP地址，比如219.133.49.24是QQ服务器地址，而添加到address-list名字取为qq，填写address：219.133.49.0/24，用于规则调用：



当在添加完 QQ 服务器 IP 地址后，在 ip firewall filter 的 forward 链表添加过滤规则：

```
/ip firewall filter add chain=forward dst-address-list=qq action=drop
```

Winbox 操作如下：



通过 torch 得到 QQ 服务器的 IP 地址，需要反复测试，直到 QQ 不能连接到服务器为止。

注： 在 v5.0 后，支持同时打开多个 Torch 工具

36.5 E-mail 发送工具

E-mail 工具非常有用，允许从路由发送 e-mail，这个工具被用于备份配置和导出网络管理信息，Email 工具用于纯验证和 TLS 加密，其他模式不支持

操作路径: /tool e-mail

属性

这个子菜单下可以设置 SMTP 服务器

from (*字符*; 默认: **<>**)显示接收者的名称或者 email 地址。
password (*字符*; 默认: **""**)SMTP 服务器要求验证的密码
server (*IP:Port*; 默认: **0.0.0.0:25**)SMTP 服务器的 IP 地址和端口
username (*字符*; 默认: **""**)SMTP 服务器要求使用的用户名。

Email 发送使用一些命令/tool e-mail send

发送命令采用下面参数:

body (*字符*; 默认:)邮件的实际信息
file (*字符*; 默认:) Email 的附件文件名
from (*字符*; 默认:)名称或者 e-mail 地址，.
password (*字符*; 默认:)密码用于 SMTP 服务的验证
server (*IP:Port*; 默认:) IP 地址和 SMTP 服务器端口
subject (*字符*; 默认:)邮件标题.
tls (*yes|no*; 默认: **yes**)是否使用 TLS 加密
to (*字符*; 默认:)目的 e-mail 地址
user (*字符*; 默认:)用于 SMTP 验证的用户名

基本事例

这个事例将说明如何导出配置，并每隔 24 小时发送 e-mail

1. 配置 SMTP 服务器

```
[admin@MikroTik] /tool e-mail> set server=10.1.1.1:25 from="router@mydomain.com"
```

2. 添加新的脚本，并取名称“export-send”

```
/export file=export
/tool e-mail send to="config@mydomain.com" subject="$[/system identity get name]    export) \
body="$[/system clock get date] configuration file" file=export.rsc
```

3.添加计划任务（scheduler），并设置运行 export-send 脚本

```
/system scheduler
add on-event="export-send" start-time=00:00:00 interval=24h
```

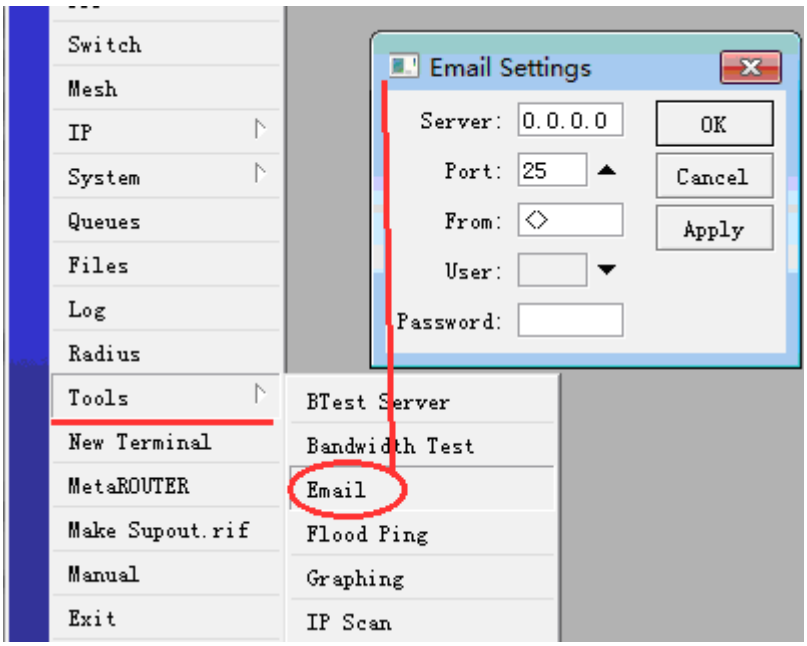
查看下面的接口列表：

```
[admin@MikroTik] interface> print
Flags: X - disabled, D - dynamic, R - running
 #     NAME                TYPE            RX-RATE     TX-RATE     MTU
 0   R ether1              ether                 0           0     1500
 1   R bridge1             bridge                0           0     1500
 2   R ether2              ether                 0           0     1500
 3   R wlan1               wlan                  0           0     1500
[admin@MikroTik] interface>
```

进入/interface bridge 桥接配置，添加一个桥：

```
[admin@MikroTik] /interface bridge> add
[admin@MikroTik] /interface bridge> prin
Flags: X - disabled, R - running
 0   R name="bridge1" mtu=1500 arp=enabled mac-address=00:00:00:00:00:00
       protocol-mode=none priority=0x8000 auto-mac=yes
       admin-mac=00:00:00:00:00:00 max-message-age=20s forward-delay=15s
       transmit-hold-count=6 ageing-time=5m
[admin@MikroTik] /interface bridge>
```

E-mail 在 winbox 下的操作路径：



36.6 Fetch 文件拷贝

Fetch 是 RouterOS 中的一个控制台工具。可以实现 HTTP、FTP 或 SFTP(v6.45 开始支持 SFTP)将文件从 RouterOS 设备复制到其他网络设备，还可以用于发送 POST/GET 请求，并将各种类型的数据发送到远程服务器。并支持 HTTPS 协议;默认情况下不进行证书检查，但是将 check-certificate 设置为 yes 将启用本地证书存储的信任验证。

操作路径: /tool fetch

属性

属性	描述
address (*字符*; 默认:)	拷贝文件设备的 IP 地址
ascii (*yes* / *no*; 默认: **no**)	
dst-path (*字符*; 默认:)	目标文件名称和路径
host (*字符*; Default:)	域名或者虚拟主机域名(如果你使用网站拷贝文件), 例如 `address=wiki.mikrotik.com host=forum.mikrotik.com` 在这个事例里, 域名会被解析为 IP 地址
keep-result (*yes* / *no*; 默认: **yes**)	如果 yes, 创建一个输入档。
mode (*ftp*/*http*/*tftp*; 默认: **http**)	选择连接协议- http, ftp 或者 tftp.
password (*字符*; 默认: **anonymous**)	密码, 需要登陆设备的验证信息
port (*整型*; 默认:)	端口。
src-path (*字符*; 默认:)	你需要拷贝远程文件的标题
url (*字符*; 默认:)	URL 指向文件, 也能用 IP 地址和路径代理
user (*字符*; 默认: **anonymous**)	用户名, 登陆远程设备的用户名。

下面的事例,显示了如何通过 FTP 从 192.168.88.2 的路由器上拷贝文件"conf.rsc",并保存 文件名为"123.rsc"。还需要使用用户名和密码登陆到路由器上

```
[admin@mt-test] /tool> fetch address=192.168.88.2 src-path=conf.rsc \
user=admin mode=ftp password=123 dst-path=123.rsc port=21 \
host="" keep-result=yes
```

另外一个实例显示了 url 的用法, 这种方式采用 http 模式

```
[admin@test_host] /> /tool fetch url="http://www.mikrotik.com/img/netaddresses2.pdf" mode=http
  status: finished

[admin@test_host] /> /file print
 # NAME                  TYPE              SIZE              CREATION-TIME
 ...
 5 netaddresses2.pdf      .pdf file         11547             jun/01/2010 11:59:51
```

发送信息到远端主机

可以使用 fetch 发送 HTTP POST 请求到远端的 HTTP 服务器,下面一个事例是发送一个地理坐标信息给 php 页面:

```
/tool fetch http-method=post http-content-type="application/json"
http-data="{\"lat\":\"56.12\",\"lon\":\"25.12\"}" url="http://testserver.lv/index.php"
```

当然也可以通过 RouterOS 脚本来完成 GPS 信息获取，并发送到远端服务器.

返回结果赋值给一个变量

从 RouterOS v6.43 开始，fetch 支持将返回结果赋值给变量，例如，可以通过获取 HTTP 值，返回结果触发一个脚本执行，下面是使用脚本获取一个 PHP 页面值，如果值为“0”，将触发禁用 ether2 网卡的操作，这个操作可以配合计划任务周期执行，查看值是否改变。

```
{
    :local result [/tool fetch url=http://10.0.0.1/disable_ether2.php as-value output=user];
    :if ($result->"status" = "finished") do={
        :if ($result->"data" = "0") do={
            /interface ethernet set ether2 disabled=yes;
        } else={
            /interface ethernet set ether2 disabled=no;
        }
    }
}
```

SFTP

从 6.45beta50 开始，fetch 开始支持 SFTP (SSH File Transfer Protocol) ，配置实例如下

```
[admin@MikroTik] > /tool fetch url="sftp://10.155.126.200/home/x86/Desktop/50MB.zip" user=x86
password=root dst-path=disk1
       status: downloading
  downloaded: 1048KiB
        total: 51200KiB
     duration: 6s
-- [Q quit|D dump|C-z pause]
```

36.7 Packet Sniffer

Packet sniffer 工具能抓取和分析进入、通过和从路由器发出的数据报（除了通过 Switch 芯片的数据传输，RouterBOARD 的 Switch 设置）

操作路径: **/tool sniffer**
需要功能包: **system**

抓包功能有以下属性：

属性	描述
file-limit (*整型 10..1000000000*; 默认: 10)	抓取数据的存储文件大小，以 KB 为单位，Sniffer 在抓取到限制值后将停止
file-name (*字符*; 默认: *""*)	文件名，Sniffer 数据报将存储在这个档中
filter-address1 (*IP 地址/子网:port*; 默	抓取过滤的第一组 IP 地址

认：0.0.0.0/0：0-65535）	
filter-address2 （*IP address/netmask:port*；Default：0.0.0.0/0:0-65535）	抓取过滤的第二组 IP 地址
filter-protocol（*all-frames* / *ip-only* / *mac-only-no-ip*；默认：**ip-only**）	过滤指定协议 • **ip-only** - 仅抓取 IP 数据包 • **all-frames** - 抓取所有数据报 • **mac-only-no-ip** - 抓取非 IP 数据
filter-stream（*yes* / *no*；默认：no）	Sniffed packets that are devised for sniffer server are ignored
interface（*all* / *ether1* / *...*；默认：all）	网卡接口选择
memory-limit（*整型 10..4294967295*；默认：10）	抓包的内存限制范围，单位 KB，达到值后 sniffer 将停止
memory-scroll（*yes* / *no*；默认：no）	
only-headers（*yes* / *no*；默认：no）	保持数据报的头部到内存，而非整个数据报
running（*只读*）	如果 Sniffer 被启动，这时值会是 yes，否则是 no
streaming-enabled（*yes* / *no*；默认：no）	定义是否将抓取到的数据报发到指定的 Sniffer 服务器
streaming-server（*ip address*；默认：）	Sniffer 服务器 IP 地址

filter-address1 和 **filter-address2 被用于指定 2 个参与连接的主机** (例如：他们将匹配从一个源地址到另外一个匹配的目标地址的数据报)。这个属性仅能在 **filter-protocol** 是 **ip-only**.

在以下的事例，**streaming-server** 将会被添加，数据流将启用，**file-name** 被设置为“test”， packet sniffer 通过 Start 命令执行，并在一段时间后通过 stop 停止：

```
[admin@MikroTik] tool sniffer> set streaming-server=192.168.0.240 \
\... streaming-enabled=yes file-name=test
[admin@MikroTik] tool sniffer> print
                  interface: all
              only-headers: no
              memory-limit: 10
                  file-name: "test"
                file-limit: 10
      streaming-enabled: yes
       streaming-server: 192.168.0.240
           filter-stream: yes
        filter-protocol: ip-only
        filter-address1: 0.0.0.0/0:0-65535
        filter-address2: 0.0.0.0/0:0-65535
                running: no
[admin@MikroTik] tool sniffer> start
```

```
[admin@MikroTik] tool sniffer> stop
```

运行 Packet Sniffer

这个命令被用于控制 packet sniffer 的运行时间。**start** 命令被用于启动和重置抓包， **stop** 命令则是停止抓包、通过 **save** 命令指定 file-name 保存当前的抓包信息

在下面的事例中，packet sniffer 将启动被在一段时间后停止：

```
[admin@MikroTik] tool sniffer> start
[admin@MikroTik] tool sniffer> stop
```

下面是通过 save 命令保存到指定的文件中：

```
[admin@MikroTik] tool sniffer> save file-name=test
[admin@MikroTik] tool sniffer> /file print
  # NAME                                TYPE          SIZE        CREATION-TIME
  0 test                                unknown       1350        apr/07/2003 16:01:52
[admin@MikroTik] tool sniffer>
```

抓取的数据包

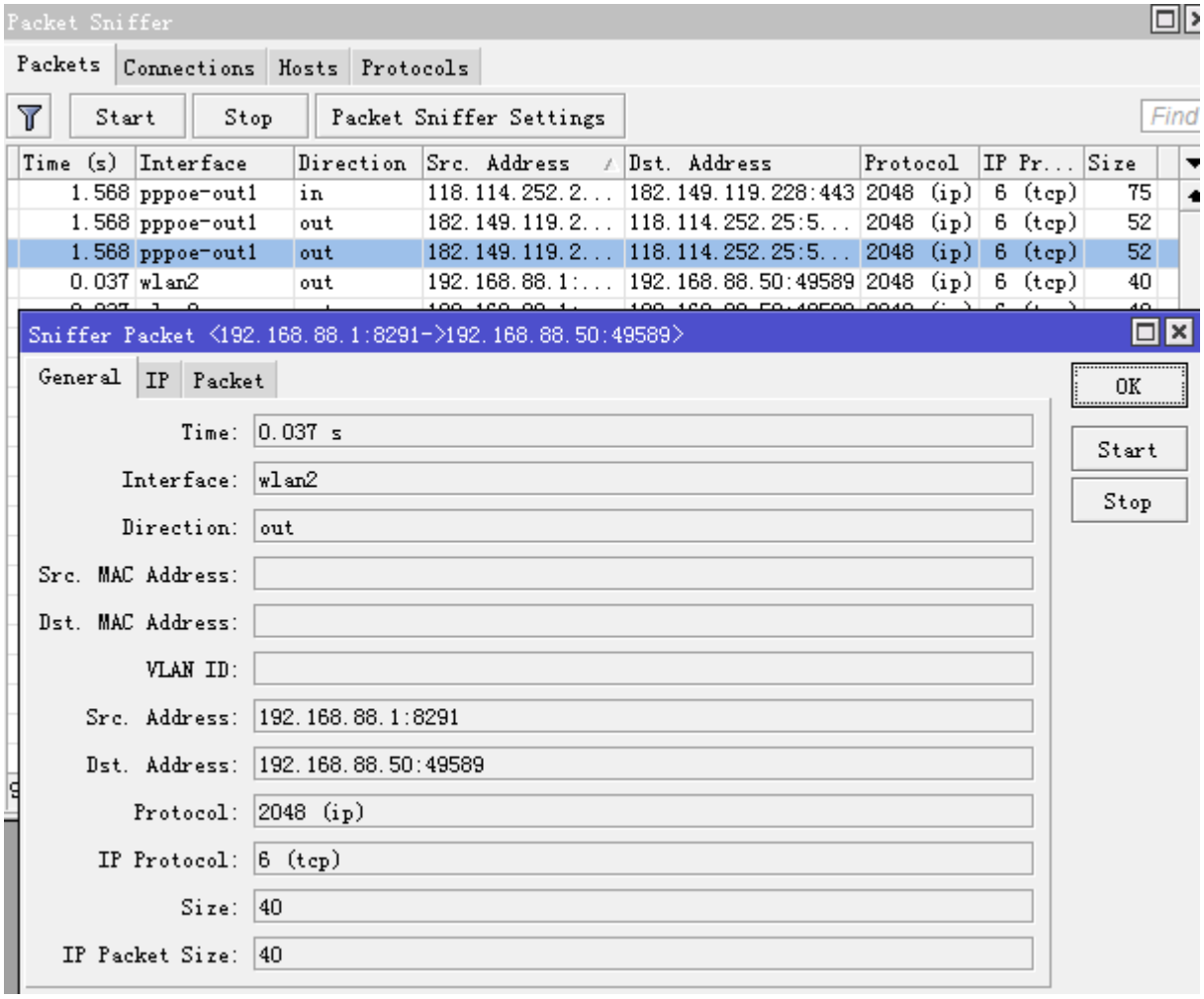
```
子菜单: /tool sniffer packet
```

这个子菜单允许查看抓到数据报的清单

属性	描述
data (*只读: text*)	表明数据报含的数据报
direction (*只读: in | out*)	数据报的方向，进入(**in**) 或者离开(**out**)
dscp (*只读: 整型*)	IP DSCP 值
dst-address (*只读: IP 地址*)	目标 IP 地址
fragment-offset (*只读: 整型*)	IP 分段偏移
identification (*只读: 整型*)	IP 识别
interface (*只读: 名称*)	抓到数据报所在的网卡接口名称
ip-header-size (*只读: 整型*)	IP 数据包头长度
ip-packet-size (*只读: 整型*)	IP 数据包长度
ip-protocol (*只读: ddp | egp | encap | ggp | gre | hmp | icmp | icmpv6 | dpr-cmt | igmp | ip | ipencap | ipip | ipsec-ah | ipsec-esp | iso-tp4 | ospf | pim | pup | rdp | rspft | st | tcp | udp | vmtp | vrrp | xns-idp | xtp*)	IP 协议的名称
protocol (*只读: ip | arp | rarp | ipx | ipv6*)	以太网协议名称

size (*只读: 整型*)	数据包长度
src-address (*只读: IP 地址*)	源 IP 地址
src-mac (*只读: MAC 地址*)	源 MAC 地址 Source MAC address
data (*只读: 字符*)	IP 数据
tcp-flags (*只读: ack | cwr | ece | fin | psh | rst | syn | urg*)	TCP 标记
time (*只读: time*)	数据报达到的时间
ttl (*只读: 整型*)	TTL 值
vlan-id (*只读: 整型*)	数据包的 VLAN-ID
vlan-priority (*只读: 整型*)	数据包的 VLAN-Priority

Winbox 下的显示



Packet Sniffer 协议

子菜单: /tool sniffer protocol

在这个子菜单下，你所有能被抓取的协议

属性	描述
bytes (*只读: 整型*)	数据字节
ip-protocol (*只读: ddp | egp | encap | ggp | gre | hmp | icmp | icmpv6 | dpr-cmt | igmp | ip | ipencap | ipip | ipsec-ah | ipsec-esp | iso-tp4 | ospf | pim | pup | rdp | rspft | st | tcp | udp | vmtp | vrrp | xns-idp | xtp*)	IP 协议
packets (*只读: 整型*)	数据包 The number of packets
port (*只读: 整型*)	TCP/UDP 的端口
protocol (*只读: ip | arp | rarp | ipx | ipv6*)	协议的名称或者编号
share (*只读: 百分百*)	指定以字节的传输类型与所有传输的比较,

如以下事例

```
[admin@MikroTik] tool sniffer protocol> print
  # PROTOCOL IP-PR... PORT            PACKETS     BYTES     SHARE
  0 ip                                 77          4592      100 %
  1 ip          tcp                    74          4328      94.25 %
  2 ip          gre                    3           264       5.74 %
  3 ip          tcp         22 (ssh)   49          3220      70.12 %
  4 ip          tcp         23 (telnet) 25         1108      24.12 %
[admin@MikroTik] tool sniffer protocol>
```

Packet Sniffer 主机

子菜单: /tool sniffer host

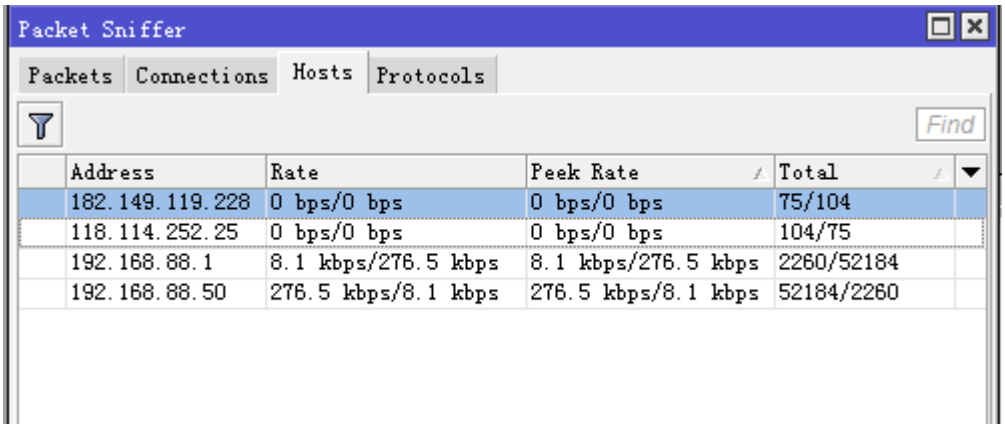
该子菜单显示了参与连接的主机列表

Property	Description
address (*只读: IP 地址*)	主机的 IP 地址
peek-rate (*只读: 整型/整型*)	最大的数据接收和发送速率
rate (*只读: 整型/整型*)	当前速率
total (*只读: 整型/整型*)	所有数据报的速率

在下面可以查看的主机列表

```
[admin@MikroTik] tool sniffer host> print
  # ADDRESS         RATE          PEEK-RATE             TOTAL
  0 10.0.0.4        0bps/0bps     704bps/0bps           264/0
  1 10.0.0.144      0bps/0bps     6.24kbps/12.2kbps     1092/2128
```

```
 2 10.0.0.181      0bps/0bps      12.2kbps/6.24kbps     2994/1598
 3 10.0.0.241      0bps/0bps      1.31kbps/4.85kbps     242/866
[admin@MikroTik] tool sniffer host>
```



Packet Sniffer 连接

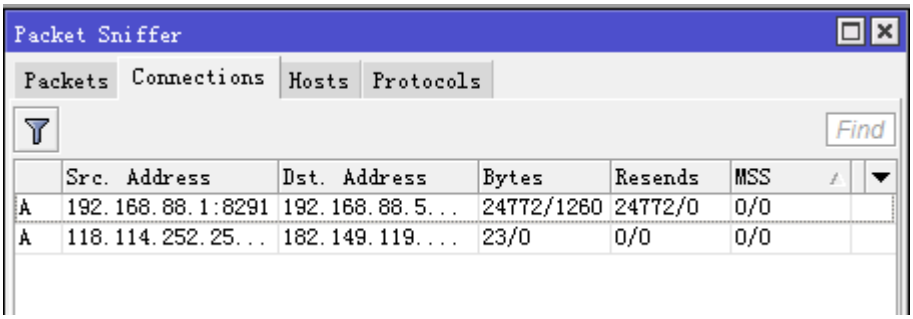
子菜单: /tool sniffer connection

你可以得到在整个抓包周期里连接列表

Property	Description
active (*只读: yes | no*)	连接是否活动
bytes (*只读: 整型/整型*)	当前连接字节
dst-address (*只读: IP address:port*)	目标地址
mss (*只读: 整型/整型*)	最大分段长度
resends (*只读: 整型/整型*)	在当前连接下重新发送的数量
src-address (*只读: IP address:port*)	源地址

这里显示了得到的连接列表

```
[admin@MikroTik] tool sniffer connection> print
Flags: A - active
  #    SRC-ADDRESS          DST-ADDRESS                BYTES      RESENDS    MSS
  0 A 10.0.0.241:1839       10.0.0.181:23 (telnet)      6/42       60/0       0/0
  1 A 10.0.0.144:2265       10.0.0.181:22 (ssh)       504/252     504/0       0/0
[admin@MikroTik] tool sniffer connection>
```



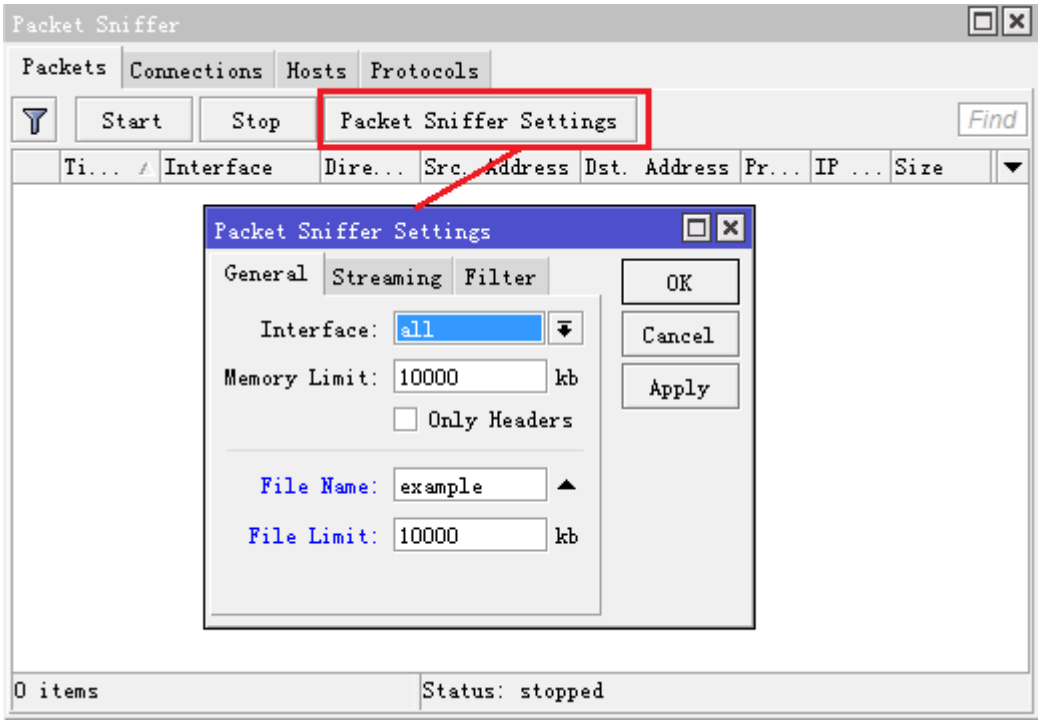
下载抓包结果

我们可以把抓包的结果保存到档中，便于更详细的分析。

属性	描述
file-name (*字符*; 默认: **""**)	抓包存储的文件名

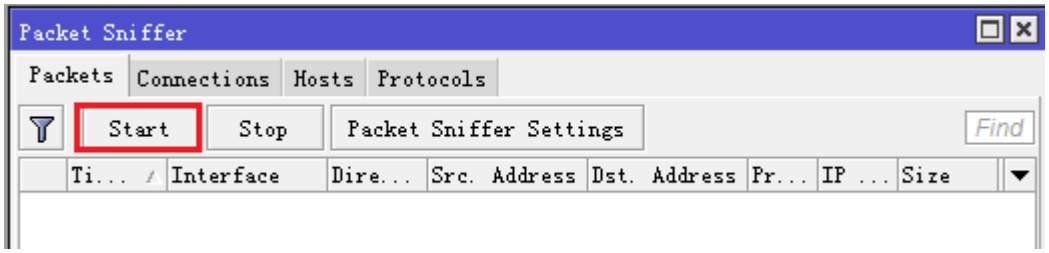
保存抓包的档设置

```
[admin@MikroTik] /tool sniffer set file-name=example file-limit=10000
```



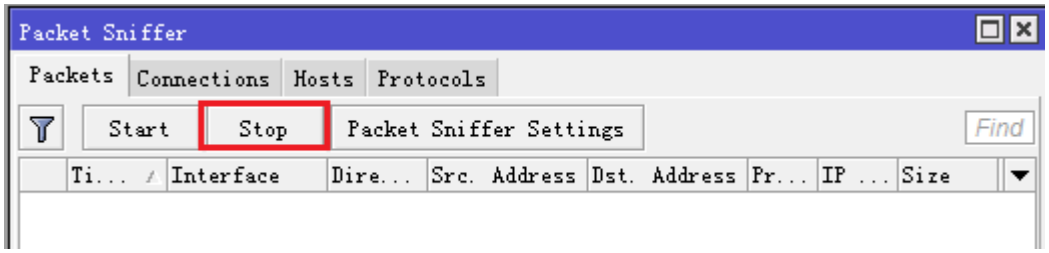
启动 sniffer

```
[admin@MikroTik] /tool sniffer start
```



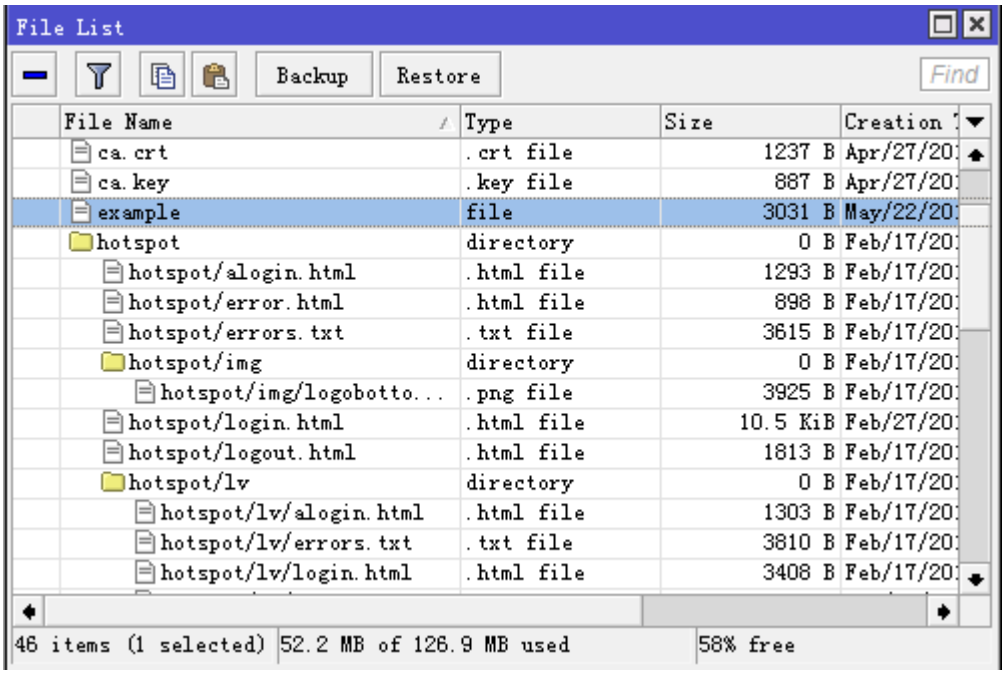
在分析一段时候后，停止抓包,

```
[admin@MikroTik] /tool sniffer stop
```



抓包结果可以通过 File 里下载，你可以选择 winbox 访问，通过拖放的方式下载到你的计算机上，也可以通过 ftp 访问下载：

```
[admin@MikroTik] /file print
 # NAME                TYPE              SIZE                CREATION-TIME
 0 example             file              44092               jan/02/2010 01:11:59
```



当你获取抓包档后，可以通过各种分析工具进行查看，例如 wireshark，这个档可以被 wireshark 直接打开。

36.8 SMB

从 RouterOS v5.12 开始支持 SMB（Server Message Block）协议，SMB 通信协议是微软（Microsoft）和英特尔(Intel)在 1987 年制定的协议，主要是作为 Microsoft 网络的通讯协议。

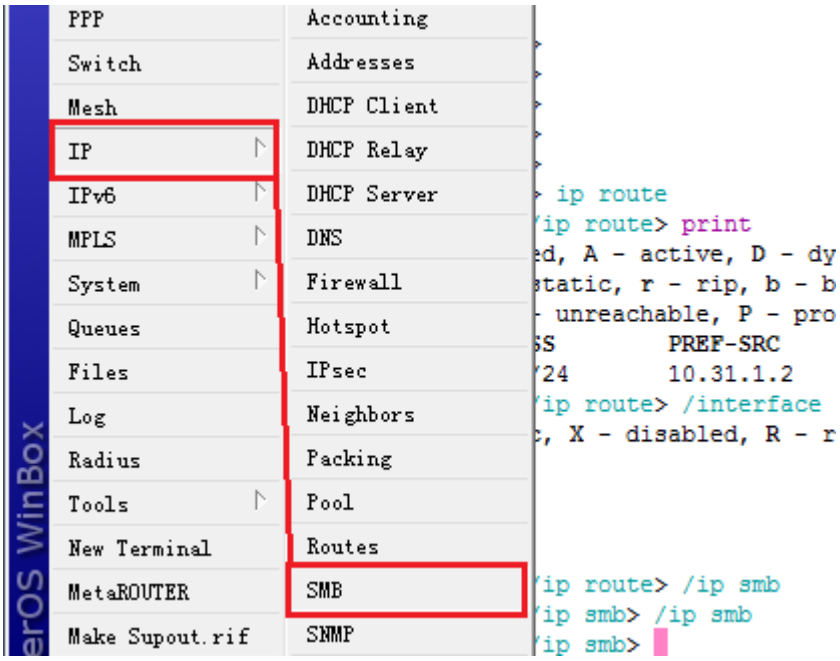
SMB 是在会话层（session layer）和表示层（presentation layer）以及小部分应用层（application layer）的协议。SMB 使用了 NetBIOS 的应用程序编程接口 （Application Program Interface，简称 API）。另外，它是一个开放性的协议，允许了协议扩展——使得它变得更大而且复杂；SMB 协议是基于 TCP－NETBIOS 下的，一般端口使用为 139，445。

操作路径：**/ip smb**

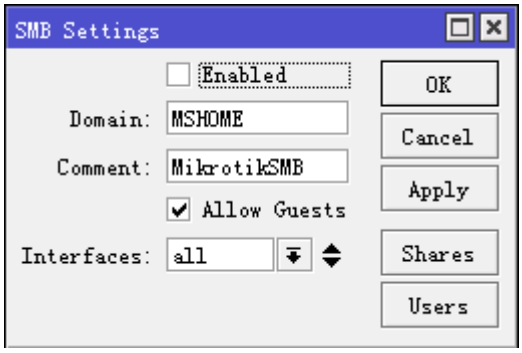
开启 SMB 协议，进入 ip smb 下可进行配置，兼容所有的 windows 系统，该功能更适合于企业办公环境，能通过本地或者远程共享档，也方便个人的档共享。那下面是 RouterOS 的 SMB 操作。

注意：RouterOS 只支持 SMB v1.0 和 v2.002

首先 5.12 版本在 ip 目录下增加了 SMB 选项，如下图：



Winbox 下的 SMB 设置接口，通过 enabled 开关 SMB 功能。



在命令行进入 ip smb，可以看到设置参数，通 set 命令启用 SMB 协议

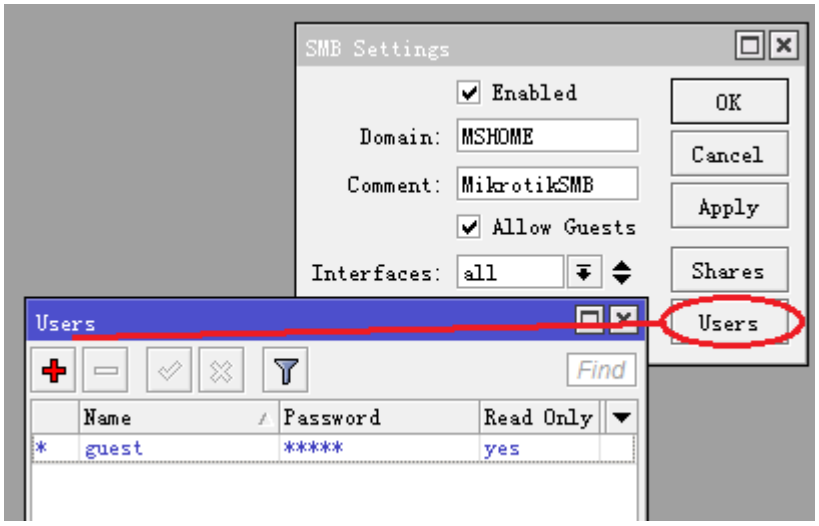
```
[admin@MikroTik] /ip smb> print
```

```
            enabled: no
             domain: MSHOME
          comment: MikrotikSMB
      allow-guests: yes
         interfaces: all
[admin@MikroTik] /ip smb> set enabled=yes
```

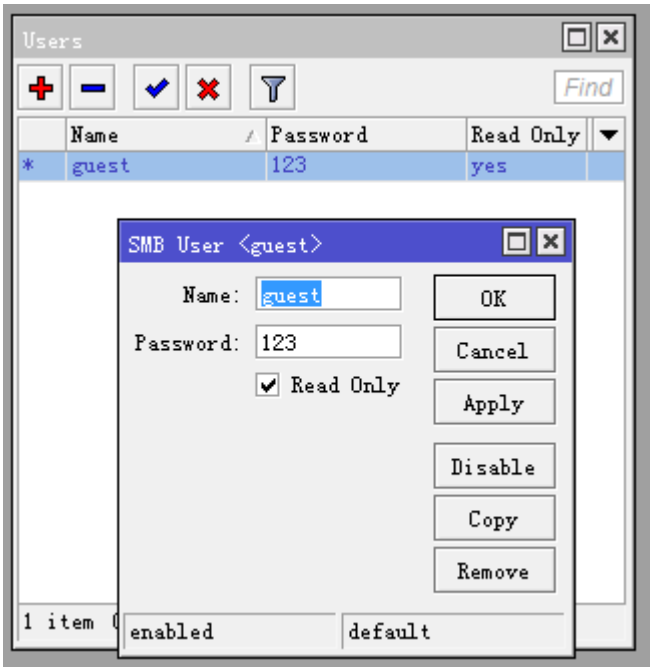
启用 SMB 协议后，以下参数：

- Domain - 默认的域为 MSHOME；
- Comment - 注释为 MikrotikSMB；
- Allow-guests - 允许 guests 访问，SMB 配置默认情况下 guest 账号的密码为空，默认 guest 就可以访问共享档；
- Interface - 选择允许或者限制某些网络接口访问，默认为所有网络接口。

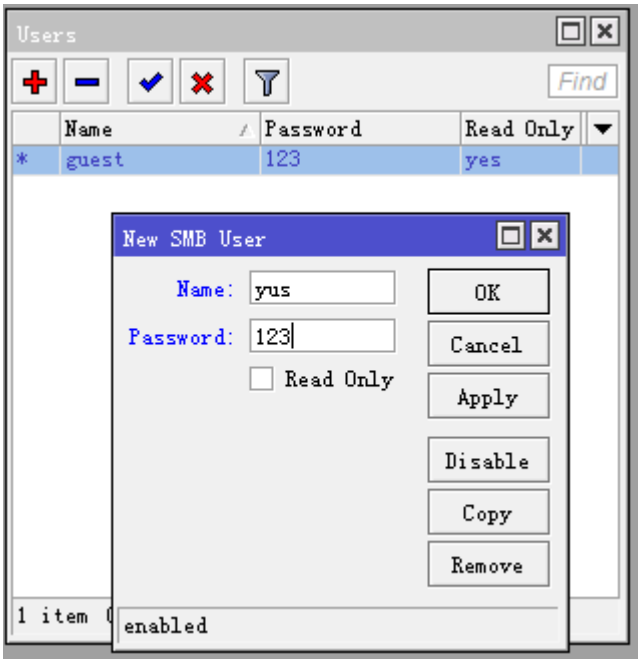
MikrotikSMB 允许设置登陆的账号和密码，预设帐号是 guest，没有密码，read-only=yes 即 guest 账号登陆后，对共享目录下的文件只有只读权限



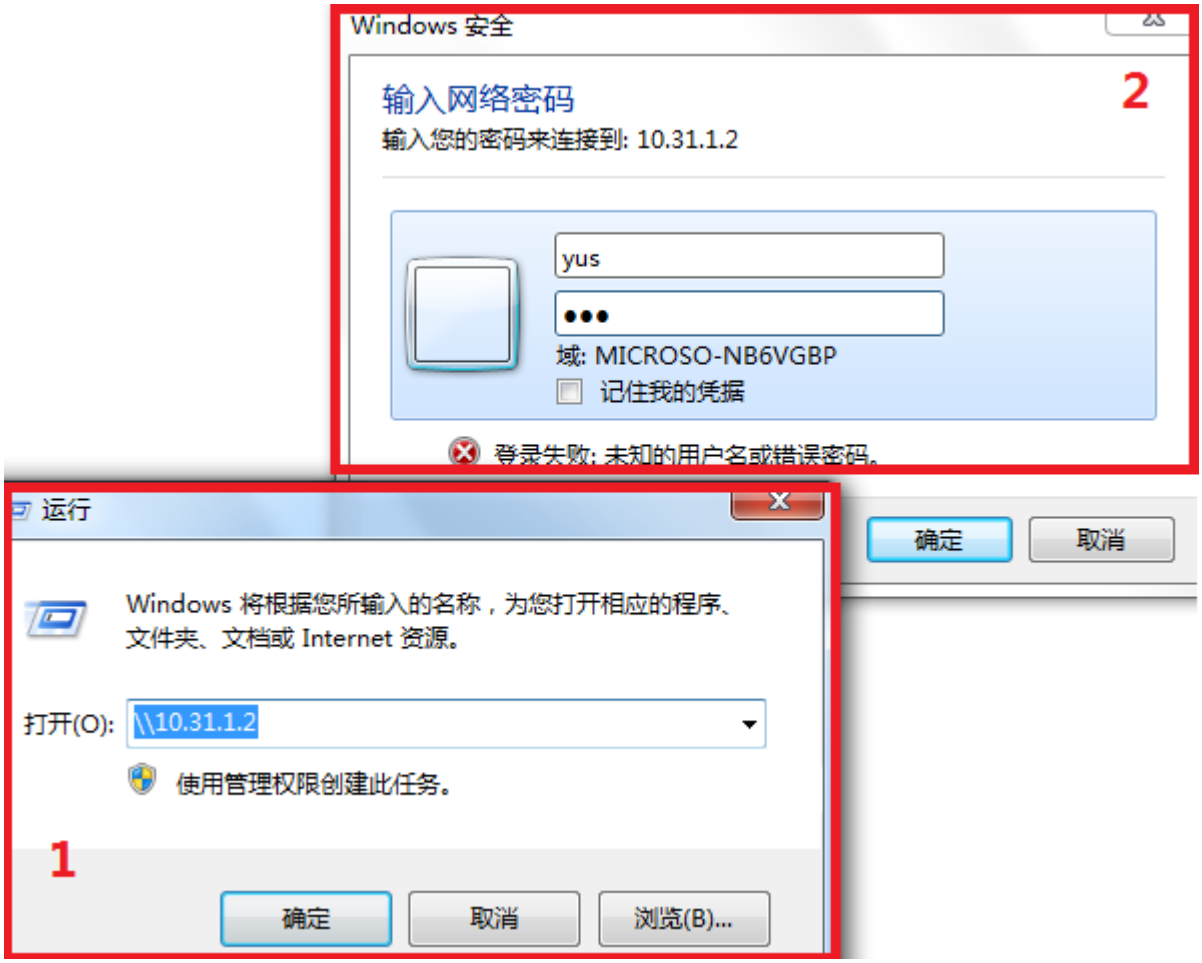
当然我们也可以设置 guest 的密码，或者新建一个用户账号和控制读写权限，下面是给 guest 账号配置一个密码 123。



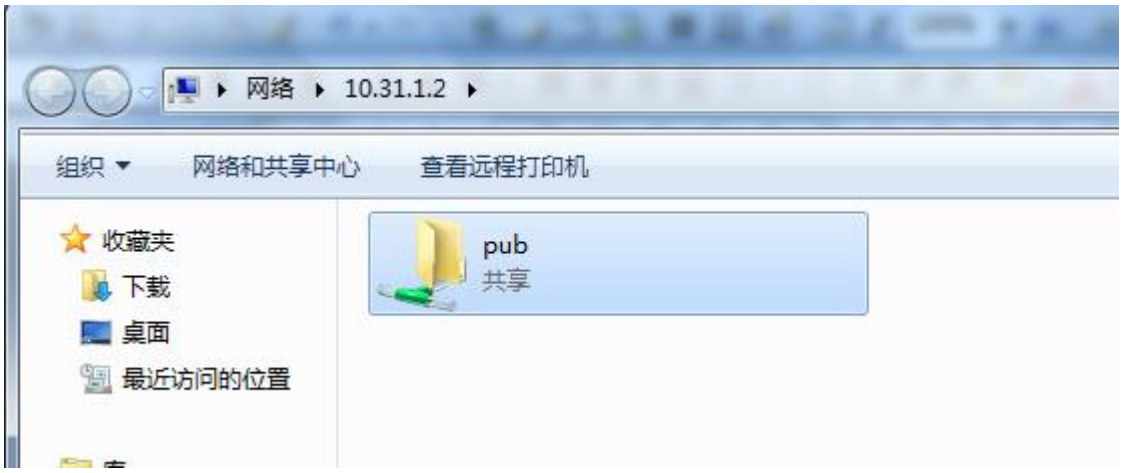
这里我也可以新建一个 yus 账号，密码 123，read-only=no，启用读写权限：



登陆 RouterOS 的档共享，从 windows 打开运行…输入\\10.31.1.2（RouterOS 连接的 IP 地址），确定后，可以看到 windows 提示的要去输入账号和密码

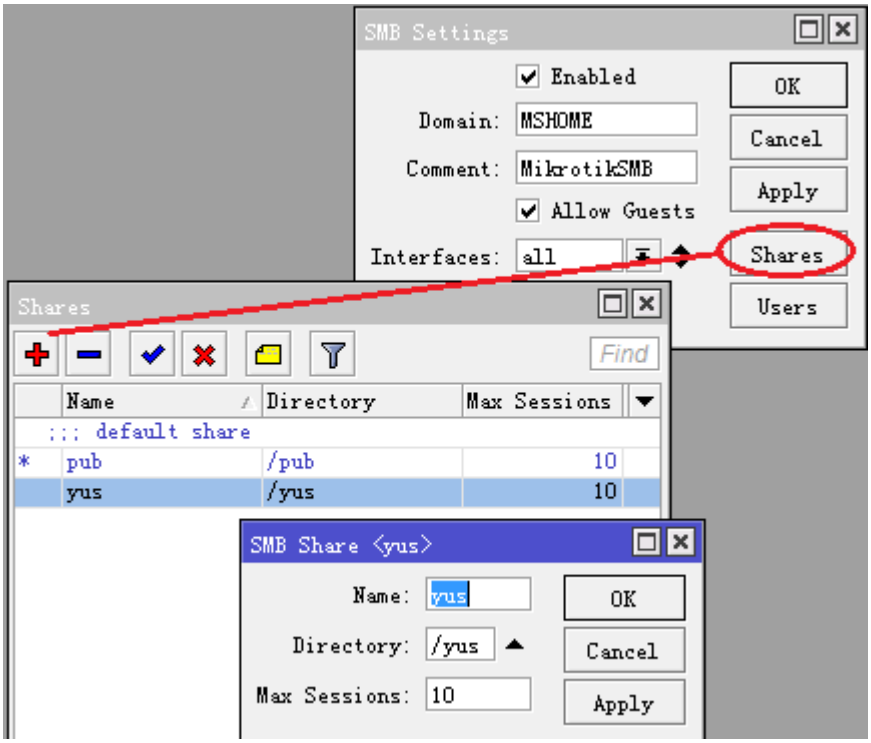


登陆后，可以看到以下的目录

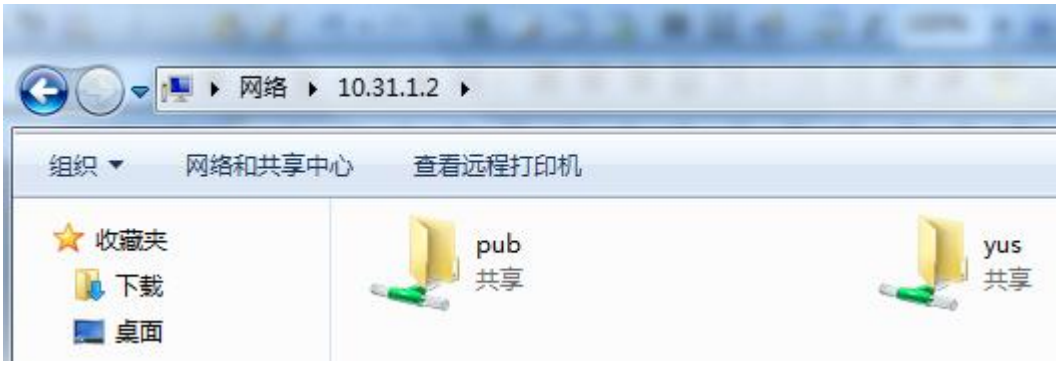


剩下的操作就不用我多说了吧，复制剪切粘贴和 windows 操作一样了，当然你也可以在 RouterOS 的 SMB 中分配各种账号给相关用户，共享各种数据！

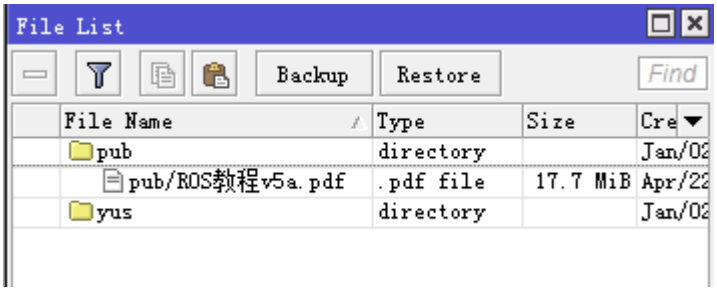
也可以为自己新建一个目录，但 SMB 还不具备不同目录下的权限管理：



我们可以看到 windows 访问目录下，增加了 yus 目录



我们可以在 RouterOS 目录下，找到对应的文件目录和内容



注意：如果你修改了 RouterOS 中 SMB 相关的配置，如账号密码、用户权限、目录等后，你的 Windows 系统在没有重启的情况下是会出现无法访问的问题，因为开机第一次访问，windows 会保存相关参数，如果服务端修改了相关参数会无法访问的可能，如果出现 windows 无法的问题，一般选择重启计算机，如果不想重启可以打开“运行”，输入“services.msc”，重启“workstation”服务试试看。

还需要注意：拷贝和读写文件，对 RouterOS 的磁盘和 CPU 消耗比较大，大档和频繁读写操作时需要特别考虑稳定性。

36.9 Profiler 工具

Profiler 工具是查看 RouterOS 中各种进程占用 CPU 的情况，有助于鉴别进程占用 CPU 资源情况。

操作路径: ***/tool profile***

如下进行操作：

```
[admin@MikroTik] > /tool profile
NAME                 USAGE
sstp                 9%
ppp                  0.5%
ethernet             0%
queue-mgmt           0%
console              0.5%
dns                  0%
winbox               0%
logging              0%
management           1.5%
ospf                 0%
idle                 87.5%
profiling            0.5%
queuing              0%
routing              0%
bridging             0%
unclassified         0.5%
```

idle – 空闲 CPU 率，即 100%-idle=当前 CPU 使用率

unclassified – 其他未分类进程

多 CPU 平台查看

多 CPU 平台，允许查看每个 CPU 的使用情况，例如下面查看系统中第二个 CPU 的资源占用情况

```
[admin@MikroTik] > /tool profile cpu=2
NAME                 CPU          USAGE
ethernet             1            0%
kvm                  1            2.5%
management           1            0.5%
idle                 1            96.5%
profiling            1            0%
unclassified         1            0.5%
```

"**cpu**" 属性允许选择第几个 CPU，或其他两个属性 **all** 和 **total**

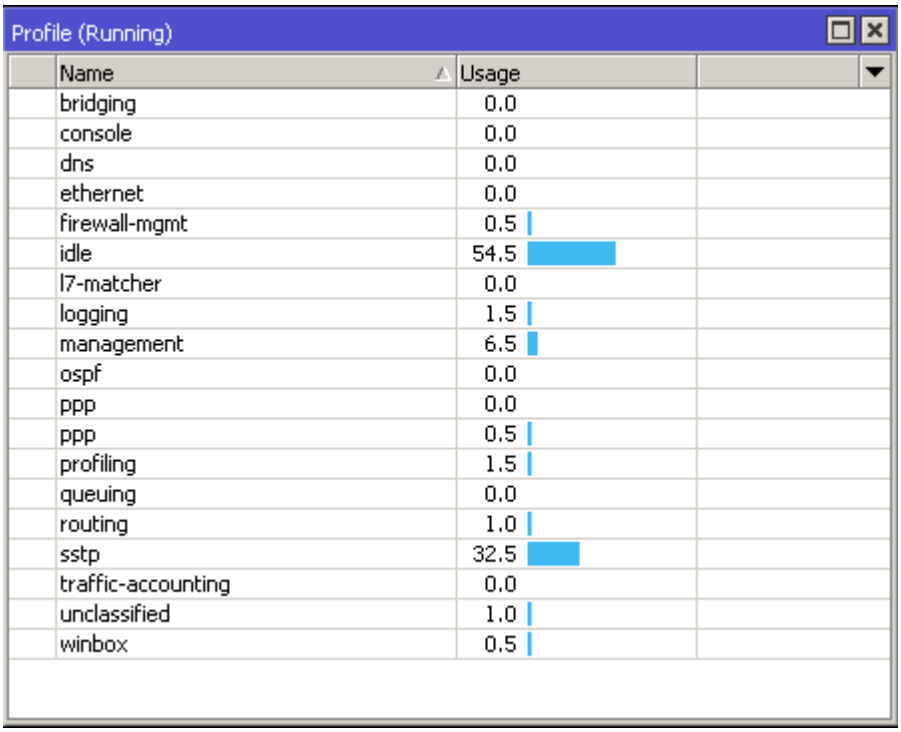
- **total** – 是设定显示所有 CPU 核心相加的平均进程使用情况

- **all** – 是设置显示每个可工作 CPU 的进程使用情况

例如在双核 CPU 系统下的操作：

```
[admin@x86-test] > /tool profile cpu=all
NAME                 CPU            USAGE
ethernet              1              0%
kvm                   0              0%
kvm                   1              4.5%
management            0              0%
management            1              0.5%
idle                  0              100%
idle                  1              93%
profiling             0              0%
profiling             1              2%

[admin@x86-test] > /tool profile cpu=total
NAME                  CPU           USAGE
ethernet              all           0%
console               all           0%
kvm                   all           2.7%
management            all           0%
idle                  all           97.2%
profiling             all           0%
bridging              all           0%
```

该工具在 winbox 的使用情况：



36.10 Traffic Generator（流量发生器）

Traffic Generator（流量发生器）工具允许测试评估 DUT（Device Under Test，被测设备）或 SUT（System Under Test，被测系统）的性能

该工具能在指定端口产生并发送 RAW 数据包。能收集延迟参数、tx/rx 速率、统计丢包率和发现无序的数据包。Traffic Generator 使用方式与 bandwidth test 工具类似，但与 bandwidth test 不同是 Traffic Generator 不需要客户端支持，是一套独立的数据包流量生成器，只能做 Tx 方向的数据包发送，如果要双向测试，需要两端设备同时进行发送测试。

操作路径： /tool traffic-generator

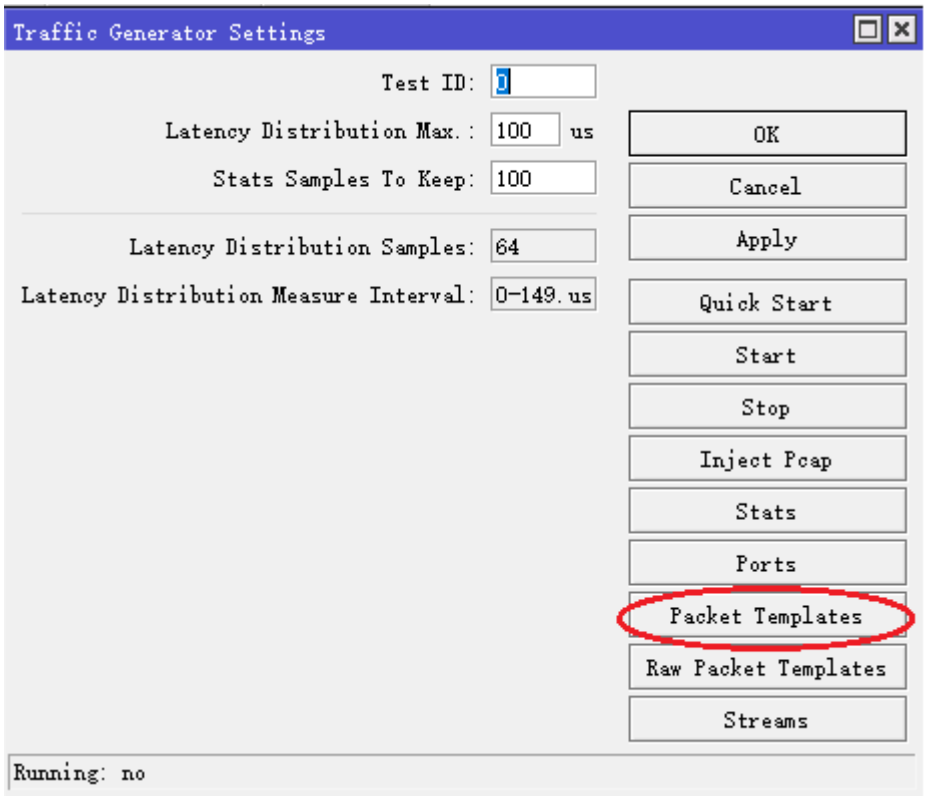
该菜单下可以设置常规的流量发射器的属性和包含用于快速开始和停止的命令

命令介绍

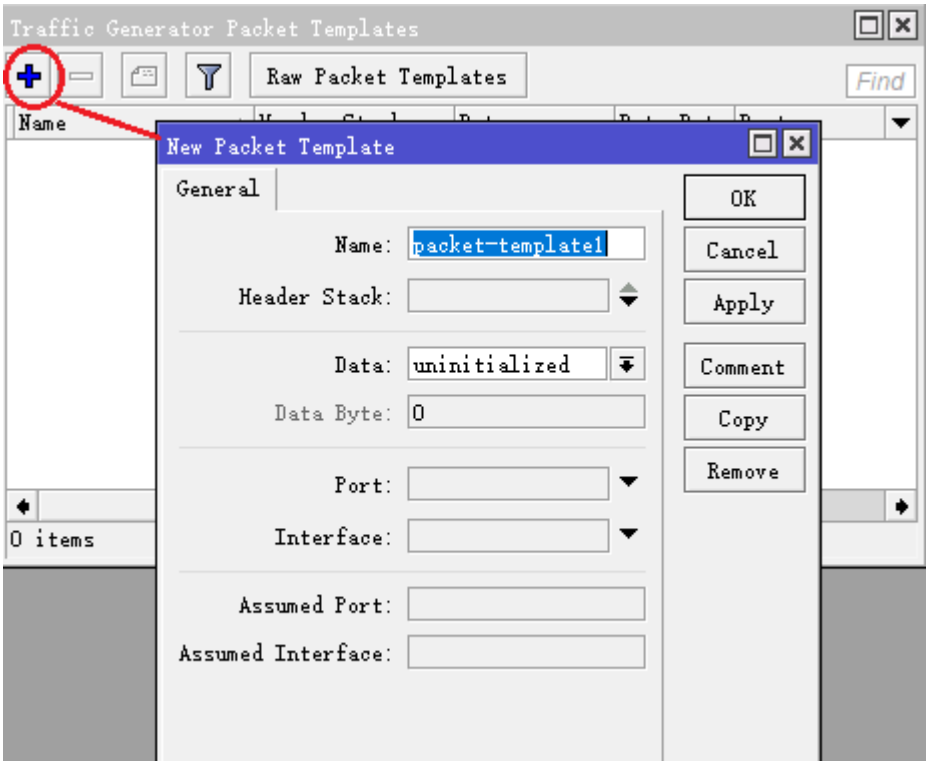
属性	描述
quick ()	该命令允许快速启动数据包发射器，并在终端控制台输出运行状态。此命令可以接受多个参数，覆盖设置包括 packet template 和 stream 等. • tx-template -数据包模板（packet templates）产生流量设置（最大 16 个模板） • duration – 测试时间长度 • entries-to-show – 在终端控制台显示多少行状态 • freeze-frame-interval – 终端控制台更新周期
start ()	该命令启动流量发生器工具，该命令只接受 **test-id** 参数。
stop ()	该命令停止运行中的流量发生器
inject ()	注入 RAW 数据到接口。
inject-pcap ()	直接从 pcap 文件注入 RAW 数据

基于 Winbox 的简单使用方法

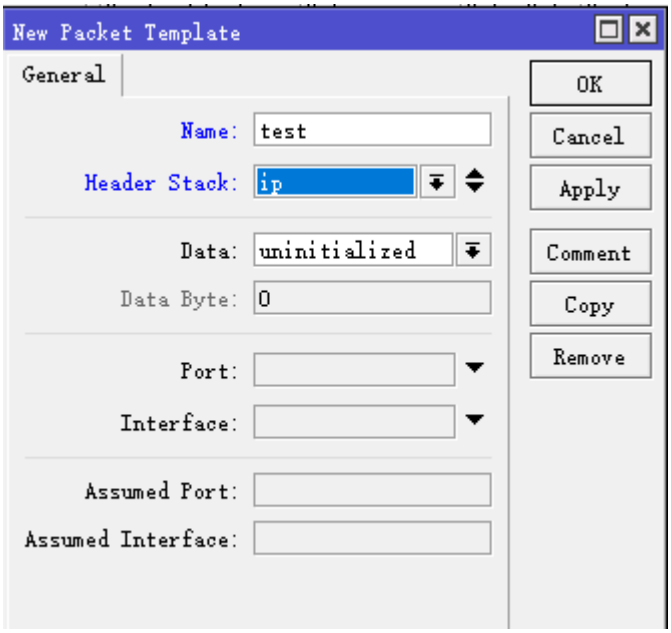
打开 Traffic Generator 后，我们首先要选择 Packet Templates 建立数据包模板



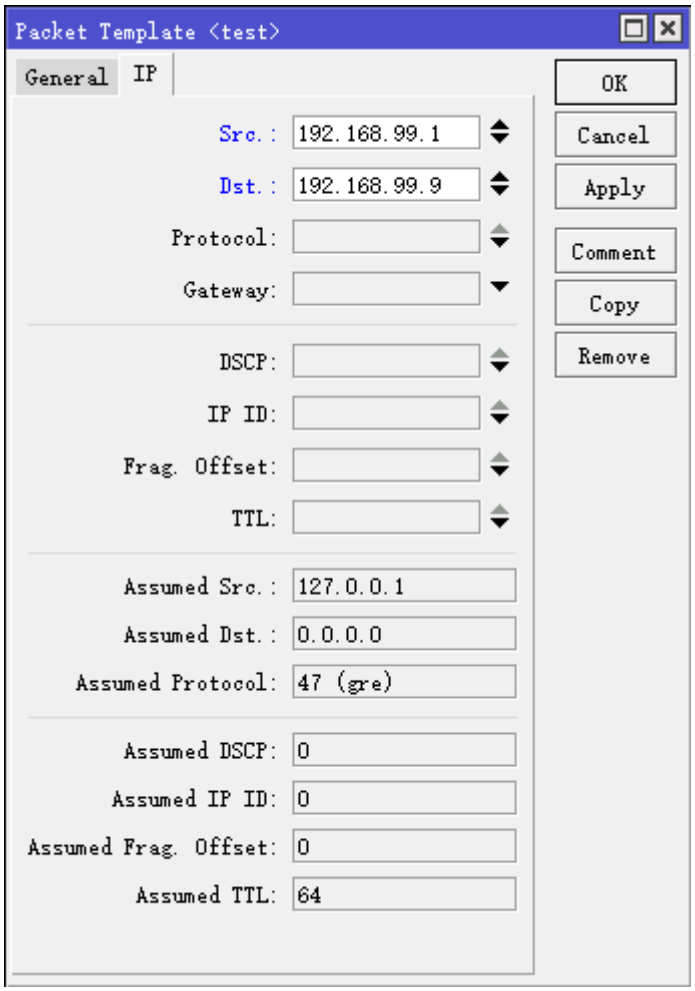
进入 Packet Templates 后，新建一个数据包模板规则



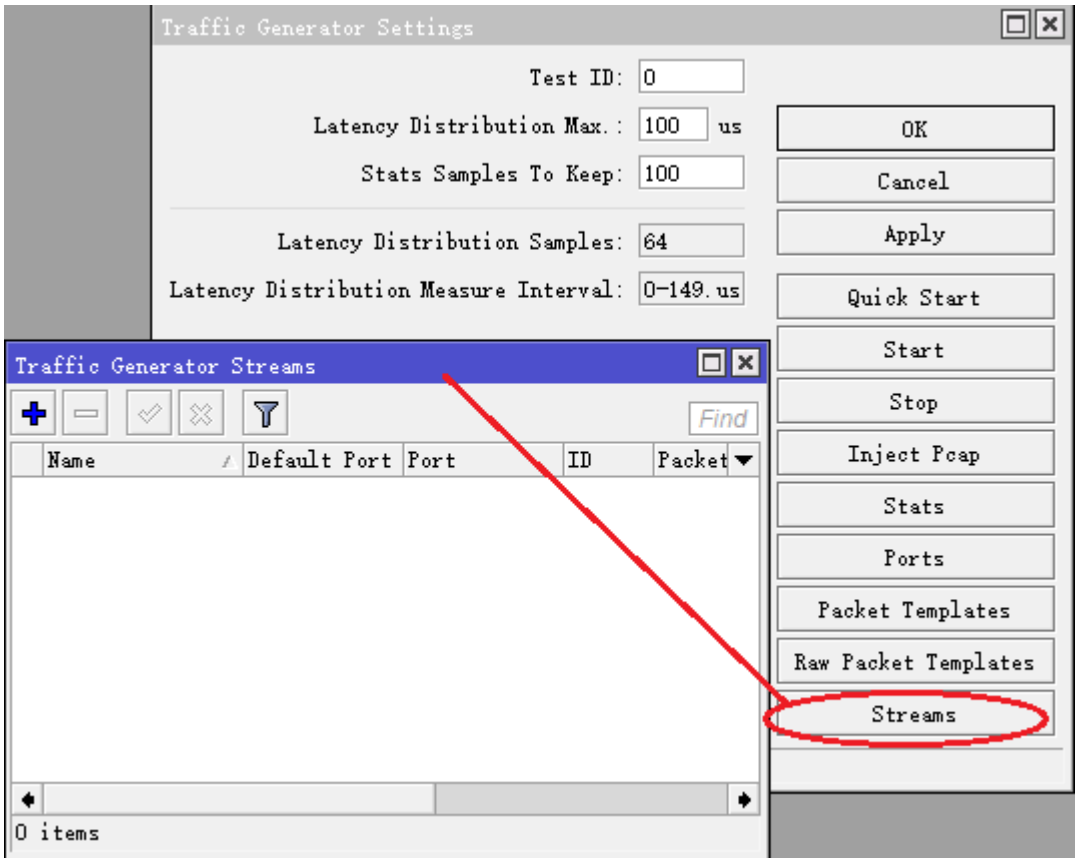
设置 name 为 test，即模板名称，设置 header 为 IP 数据包



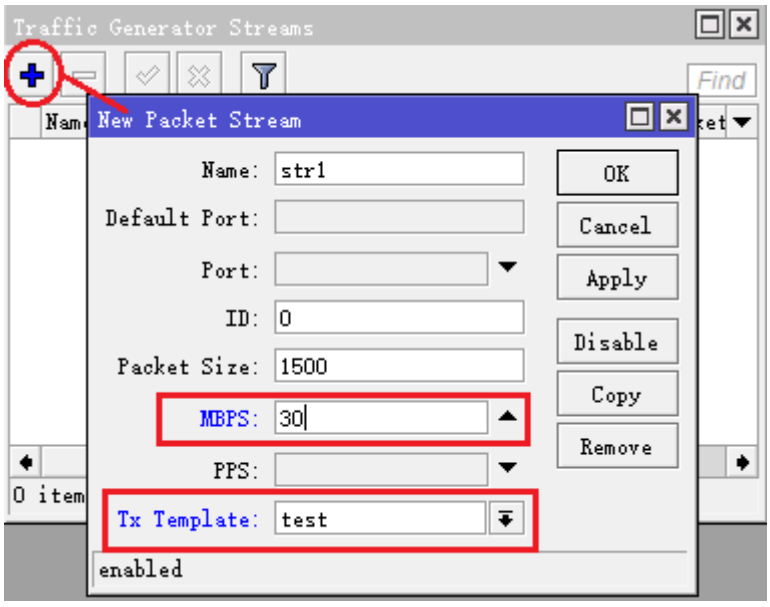
点击 ok，后重新进入 test 规则，配置 IP 数据包的假设参数，设置该数据包规则的源 IP 地上是本地路由器 IP 地址 192.168.99.1，目标 IP 地址 192.168.99.9，也就是要测试的目标设备。



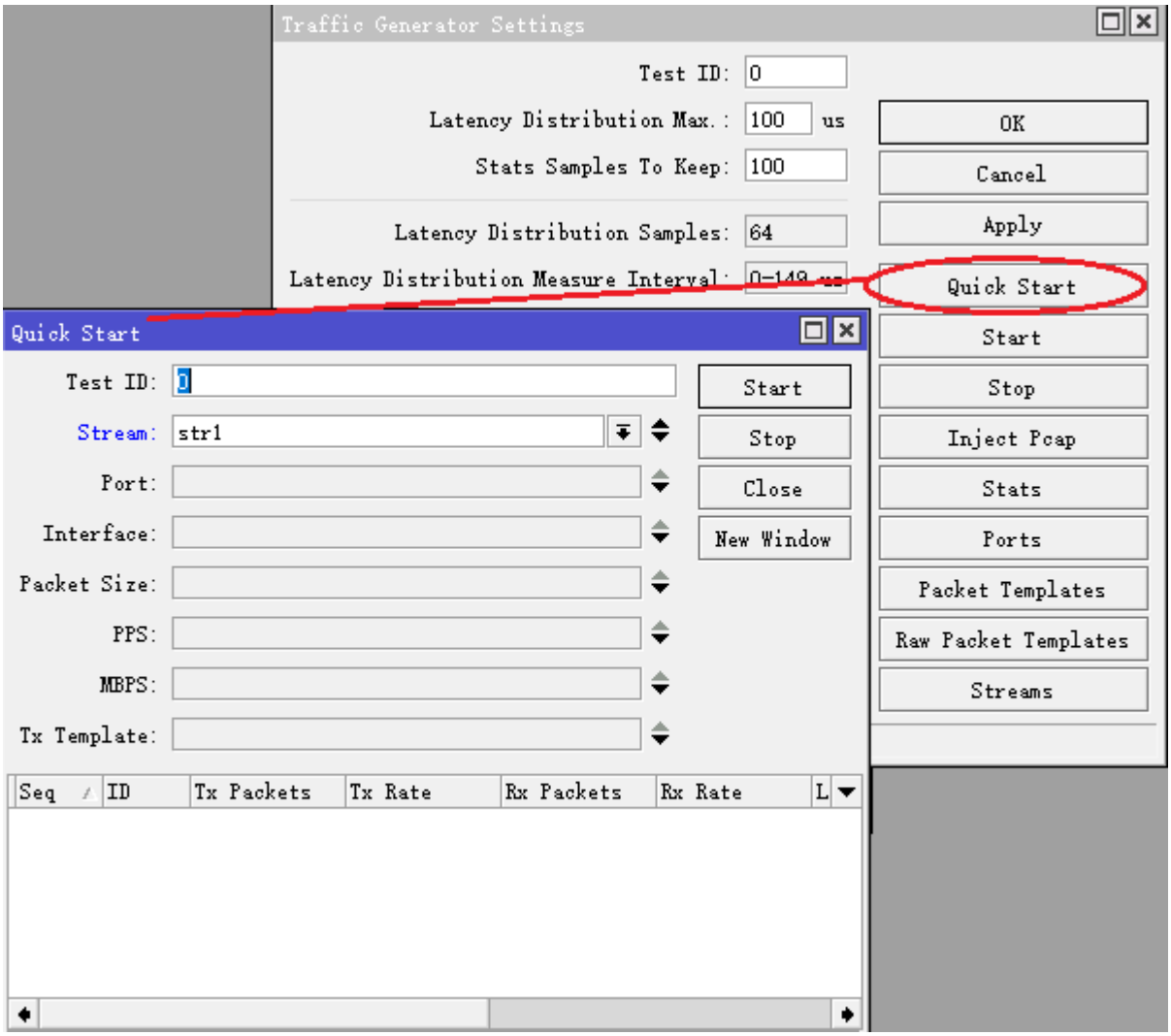
设置完成后点击 ok，然后进入 stream 菜单，配置数据流模板



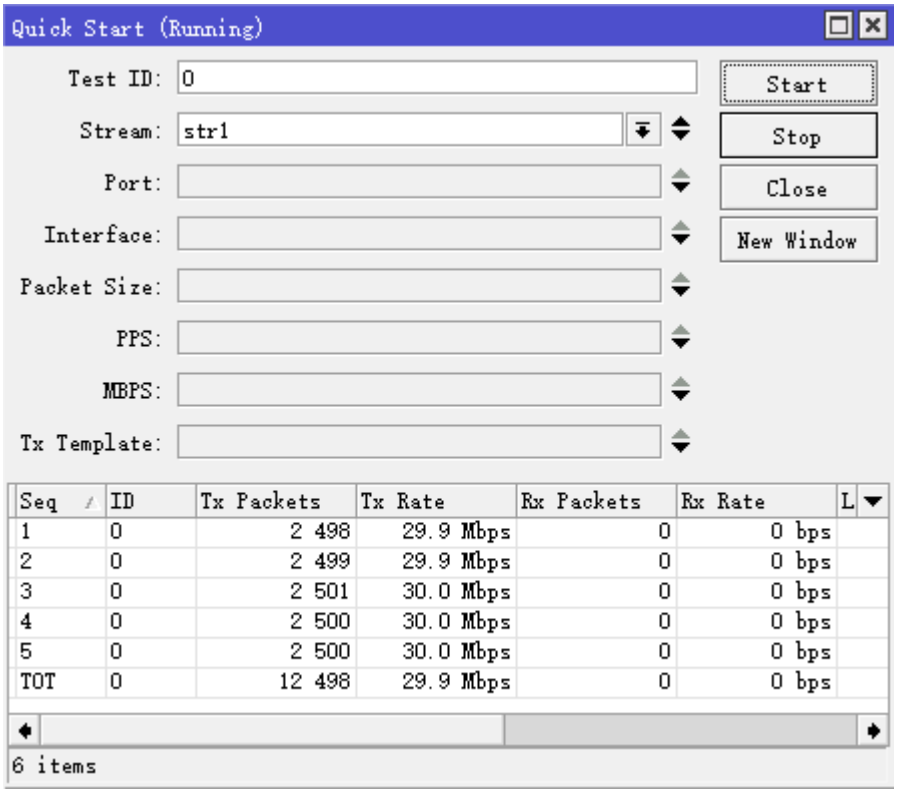
新建一个 stream 流量规则，设置 Tx Templates 为 test，MBPS 设置 30Mbps，为测试带宽速率



创建 stream 模板后，回到 Traffic Generator 菜单，选择 Quick start，设置 stream 为之前创建的 str1



选择完成后，点击 start 开始测试



36.11 Speed test

Speed Test 是一个易用的测试工具，主要对 ping 延迟测试，以及 MikroTik 设备之间的 TCP 和 UDP 吞吐量测试

"speed-test" 命令是基于 Ping Tool 和 Bandwidth Test，因此为保证测试 Bandwidth test server 需要开启

操作路径: /tool speed-test

属性	描述
address (*IP 地址*; 默认: **""**)	主机 IP 地址
connection-count (*整型*; 默认: **20**)	如何设备超过 20 个核心，将使用核心计数参数
password (字符; 默认:)	对端设备登密码
test-duration (*整型*; 默认: **10**)	测试周期 (默认 5 项测试为 10 秒，每项测试间隔 1 秒进行，总时长为 55 秒
user (*字符*; 默认: **admin**)	对端设备管理账号，默认设置 admin

事例

Bandwidth 和 speed 工具都需要通过 MikroTik 之间的设备测试，这个测试并不能完全模拟实际应用，只是一个产生流量的模拟环境，了解网络和设备的基本情况作为参考。

下面是一个简单的测试，设备 A(192.168.88.1)到设备 B (192.168.88.2)，默认账号 admin，没有密码

```
[admin@MikroTik]] > /tool speed-test address=192.168.88.1
                    status: done
         time-remaining: 0s
      ping-min-avg-max: 541us / 609us / 3.35ms
   jitter-min-avg-max: 0s / 76us / 2.76ms
                  loss: 0% (0/100)
          tcp-download: 921Mbps local-cpu-load:30%
            tcp-upload: 920Mbps local-cpu-load:30% remote-cpu-load:25%
          udp-download: 917Mbps local-cpu-load:6% remote-cpu-load:21%
            udp-upload: 916Mbps local-cpu-load:20% remote-cpu-load:6%
```

如果测试中任何设备的 CPU 达到 100%，将会发出警告信息：

```
[admin@MikroTik]] > /tool speed-test address=192.168.88.1
                    ;;; results can be limited by cpu, note that traffic generation/termination
                        performance might not be representative of forwarding performance
                    status: done
         time-remaining: 0s
      ping-min-avg-max: 541us / 609us / 3.35ms
   jitter-min-avg-max: 0s / 76us / 2.76ms
                  loss: 0% (0/100)
```

```
tcp-download: 721Mbps local-cpu-load:78%
   tcp-upload: 820Mbps local-cpu-load:100% remote-cpu-load:84%
udp-download: 906Mbps local-cpu-load:10% remote-cpu-load:54%
   udp-upload: 895Mbps local-cpu-load:55% remote-cpu-load:12%
```

"test-duration" 参数可以修改 5 项的测试周期，分别如下：

- Ping 测试 50ms 延迟
- TCP 接收
- TCP 发送
- UDP 接收
- UDP 发送

第三十七章 SOCKS 代理

SOCKS 是基于应用层协议穿透防火墙的代理服务器，其作用是在防火墙阻止了一些应用端口，也能通过 SOCKS 代理穿透连接网络。

客户端通过应用层软件连接到 SOCKS 代理服务器，SOCKS 服务器会查看访问列表（access list）是否允许客户端访问远端的应用资源。如果允许，SOCKS 服务器将数据包传递给应用服务器，并创建一个应用服务器与客户端的连接。

RouterOS 支持中的 SOCKSv4 和 v5（v6.47 开始支持 SOCKS5），默认端口为 TCP/1080。

37.1 SOCKS Server

RouterOS v6.47 支持 SOCKSv4/v5 代理服务器，允许其他客户端通过 v4/v5 版本连接到 RouterOS 的代理服务器，v5 版本支持账号验证和无账号验证两种模式。

属性描述

- **connection-idle-timeout** (时间; 默认: 2m) – 连接空闲超时时间，即连接空闲后多长时间关闭
- **enabled** (yes | no; 默认: no) – 是否启用 SOCKS 代理
- **max-connections** (整型: 1..500; 默认: 200) – 同时建立最大连接数
- **port** (整型: 1..65535; 默认: 1080) – SOCKS 服务器监听连接的 TCP 端口
- **version**（4|5;）- SOCKS 服务器版本
- **auth-method**（none | password）- 选择验证模式，password 模式要求用户和密码验证

Access List（访问列表）

操作路径: /ip socks access

在 SOCKS 访问列表中，你能添加访问策略控制访问 SOCKS 服务器。SOCKS 访问列表类似于防火墙列表策略。

属性描述

- action (allow | deny; 默认: allow) – 选择此规则的执行方式。allow – 匹配这个规则将允许数据包转发，并进一步处理；deny - 匹配这个规则将拒绝数据访问
- dst-address (IP address/netmask) – 目标 IP 地址（对端服务器）
- dst-port (port) – 目标 TCP 端口
- src-address (IP address/netmask) – 源 IP 地址（客户端）
- src-port (port) – 源 TCP 端口

Active Connections（活动连接）

操作路径: /ip socks connections

活动连接列表显示了所有已经建立 TCP 连接，用于 SOCKS 代理服务器连接维护

属性描述

- dst-address (只读: IP 地址) – 目标 IP 地址
- rx (只读: 整型) – 接收字节
- src-address (只读: IP 地址) – 源 IP 地址
- tx (只读: 整型) – 发送字节
- type (只读: in | out | unknown) – 连接类型，in – 向内连接；out – 向外连接；unknown – 连接正被发起

查看当前 TCP 连接列表

```
[admin@MikroTik] ip socks connections> print
 # SRC-ADDRESS                DST-ADDRESS                TX         RX
 0 192.168.0.2:3242           159.148.147.196:80         4847       2880
 1 192.168.0.2:3243           159.148.147.196:80         3408       2127
 2 192.168.0.2:3246           159.148.95.16:80           10172      25207
 3 192.168.0.2:3248           194.8.18.26:80             474        1629
 4 192.168.0.2:3249           159.148.95.16:80           6477       18695
 5 192.168.0.2:3250           159.148.95.16:80           4137       27568
 6 192.168.0.2:3251           159.148.95.16:80           1712       14296
 7 192.168.0.2:3258           80.91.34.241:80            314        208
 8 192.168.0.2:3259           80.91.34.241:80            934        524
 9 192.168.0.2:3260           80.91.34.241:80            930        524
10 192.168.0.2:3261           80.91.34.241:80            312        158
11 192.168.0.2:3262           80.91.34.241:80            312        158
[admin@MikroTik] ip socks connections>
```

User（账号）

操作路径：/ip socks users

为用户创建账号和密码，以及配置相关的参数

- name（字符）- 用户账号
- only-one （yes | no） – 账号是否唯一
- password （字符） - 账号密码
- rate-limit （字符）- 限制账号带宽

37.2 应用实例

FTP 服务通过 SOCKS 代理服务器

假设一个内网 192.168.0.0/24，并做了 nat 伪装（masqueraded），RouterOS 网络接口配置一个公网 IP 10.1.0.104/24 和内网网关 192.168.0.1/24，互联网中有一台 FTP 服务器 IP 地址是 10.5.8.8。我们希望让内网客户端 IP 地址 192.168.0.2 访问这个 FTP 服务器

首先，确认 nat 伪装规则已经配置

```
[admin@MikroTik] ip firewall nat> print
Flags: X - disabled, I - invalid, D - dynamic
 0    chain=srcnat action=masquerade src-address=192.168.0.0/24
[admin@MikroTik] ip firewall nat>
```

然后确认访问公网的 FTP 服务已经被防火墙拒绝，即访问 FTP 服务是不能直接通过正常应用协议访问：

```
[admin@MikroTik] ip firewall filter> print
Flags: X - disabled, I - invalid, D - dynamic
 0    chain=forward action=drop src-address=192.168.0.0/24 dst-port=21 protocol=tcp
[admin@MikroTik] ip firewall filter>
```

这里就需要使用 SOCKS 代理服务器去访问，首先启用 SOCKS 服务器：

```
[admin@MikroTik] ip socks> set enabled=yes
[admin@MikroTik] ip socks> print
                   enabled: yes
                      port: 1080
    connection-idle-timeout: 2m
           max-connections: 200
[admin@MikroTik] ip socks>
```

接下来配置访问列表，添加客户端 IP 地址 192.168.0.2/32 到 SOCKS 访问列表中，允许数据传输从 FTP 服务器到客户端(且允许目标端口从 1024 到 65535 的所有 IP 地址)，然后拒绝所有：

```
[admin@MikroTik] ip socks access> add src-address=192.168.0.2 dst-port=21 action=allow
[admin@MikroTik] ip socks access> add dst-port=1024-65535 action=allow
[admin@MikroTik] ip socks access> add action=deny
[admin@MikroTik] ip socks access> print
Flags: X - disabled
 0    src-address=192.168.0.2 dst-port=21 action=allow
 1    dst-port=1024-65535 action=allow
 2    action=deny
[admin@MikroTik] ip socks access>
```

这样，SOCKS 服务器已经配置完成，查看 Connctions 中断数据传输和连接情况：

```
[admin@MikroTik] ip socks connections> print
 # SRC-ADDRESS                DST-ADDRESS                TX          RX
 0 192.168.0.2:1238           10.5.8.8:21                1163        4625
 1 192.168.0.2:1258           10.5.8.8:3423              0           3231744
[admin@MikroTik] ip socks connections>
```

注意：为了让客户端正确连接 SOCKS 代理服务器，必须在客户端应用程序上配置 SOCKS 的 IP 地址和端口，在这个实例中 IP 地址设置为 192.168.0.1，TCP 端口为 1080。

Firefox 如何连接 socket5 代理

Firefox 只能通过非账号的方式连接 SOCKS 服务器，假设内网接口地址 172.18.10.1/24，并做了 nat 伪装（masqueraded），RouterOS 上网通过 PPPoE 拨号

因此 RouterOS，只需要启用 SOCKS 服务，并选择 version 版本为 5，命令配置如下：

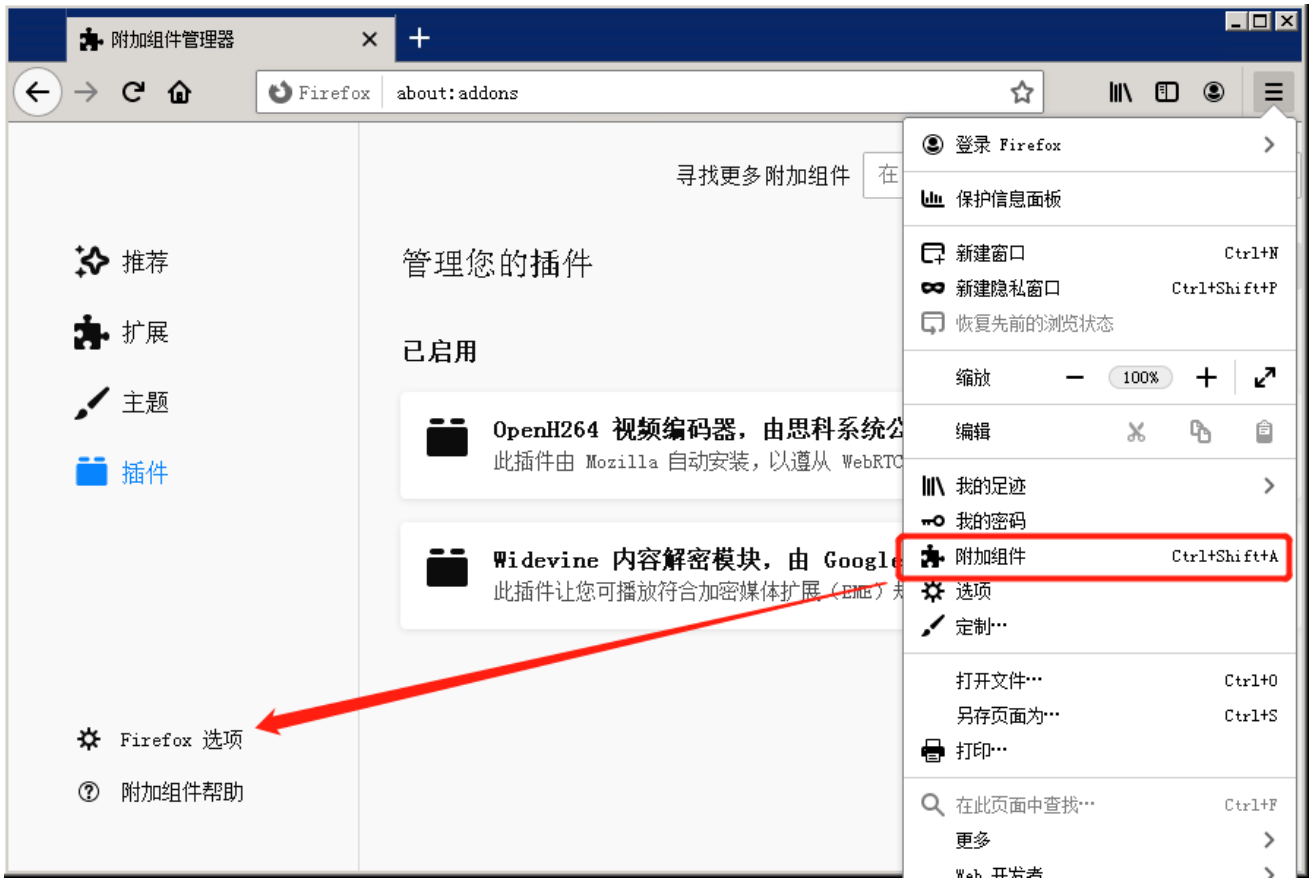
```
/ip socks
set enabled=yes version=5
```

RouterOS 的 srcnat 配置指定上网接口为 pppoe-out1，如果有其他出口，请选择指定的接口名称

```
[admin@MikroTik] ip firewall nat> print
Flags: X - disabled, I - invalid, D - dynamic
 0    chain=srcnat action=masquerade out-interface=pppoe-out1
[admin@MikroTik] ip firewall nat>
```

Firefox 配置如下：

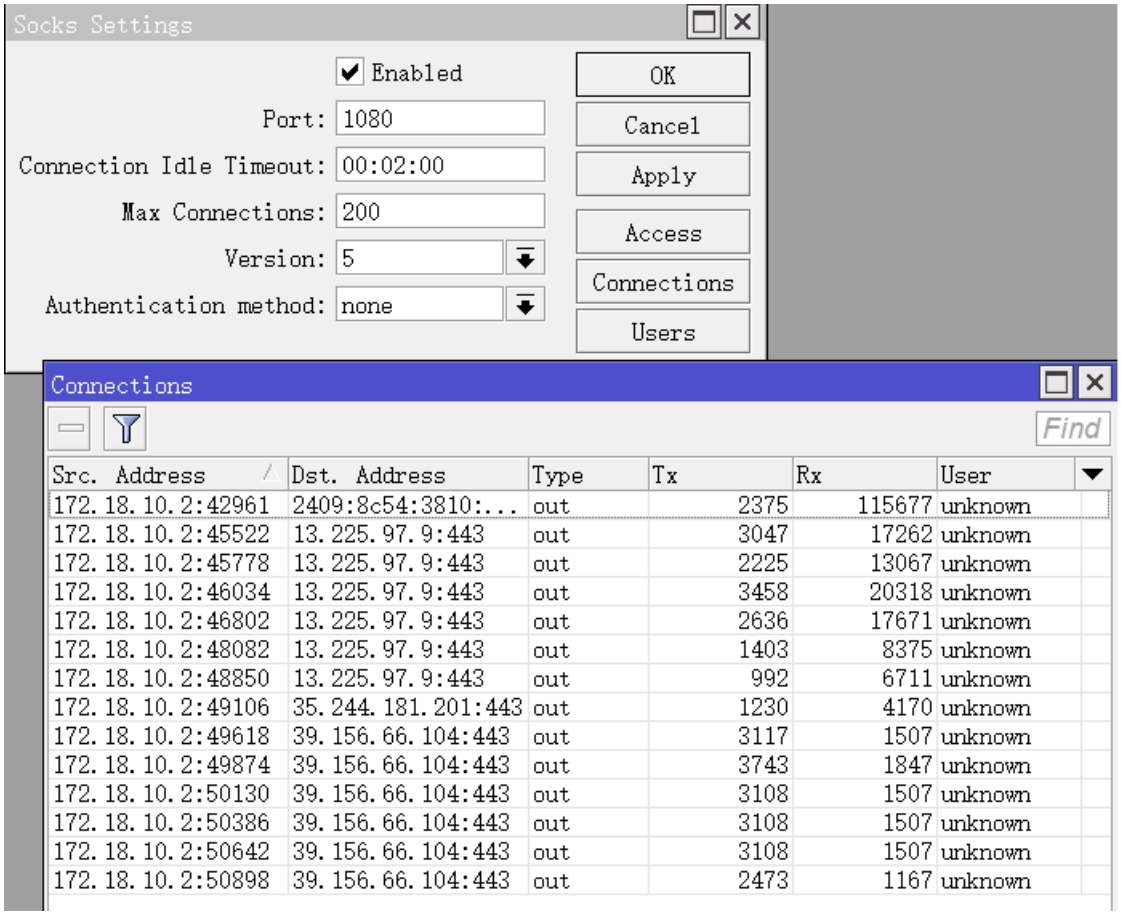
打开功能菜单，选择附加组件



移动到网络设置，点击设置

选择手动代理配置，firefox 仅支持无账号连接，选择服务器 IP 和 SOCKS 版本即可

SOCKSv5 连接成功后，可以在 connection 查看连接的会话状态：



由于 firefox 不支持账号登录，所以 user 栏显示为 unknow

第三十八章 User Manager v4

RouterOS 集成的 User Manager 是一套 RADIUS 系统，只不过是 MikroTik 专为 RouterOS 开发的 RADIUS，即只能与 RouterOS 进行对接，不支持其他系统，与 RADIUS 相同使用 UDP/1812 为认证授权端口，UDP/1813 为计费端口，支持 RADIUS 控制的 Incomming 端口默认为 3799，User Manager 主要应用于：

- Hotspot 用户管理；
- PPP（PPTP、L2TP、OVPN、SSTP/PPPoE）用户管理；
- DHCP 用户管理；
- WLAN 无线用户管理；
- RouterOS 登录账号管理

从 2.9.25 版本开始 RouterOS 集成了 User Manager 功能，但在 RouterOS 4.0 开始重新对 User Manager 进行了优化，特别在策略配置上变动较大，后面的版本在此基础上不断改进和完善。

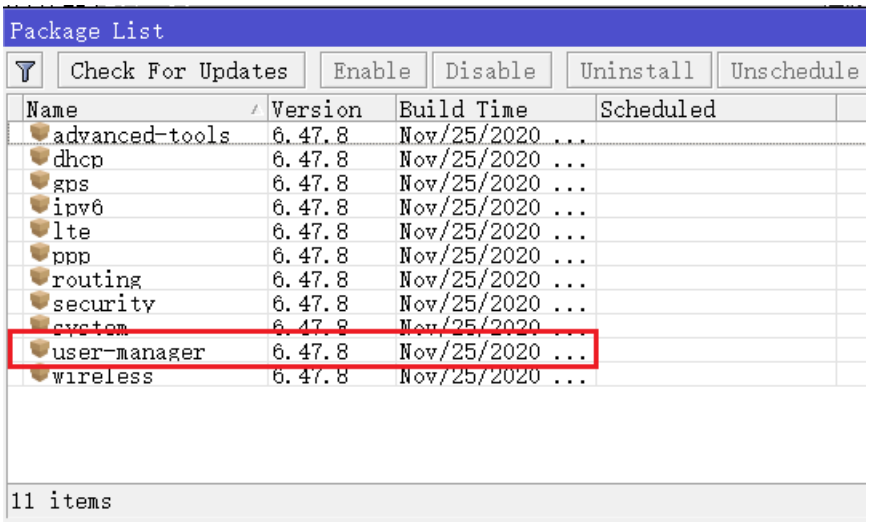
User Manager 操作主要通过 Web 接口进行管理对接 RouterOS，添加、删除和查询用户信息，同样也可以使用 CLI 操作，并且可以进行备份和导入。User Manager 通过安装包形式嵌入到 RouterOS 上，即 User Manager 版本与 RouterOS 版本一致。User Manager 和 RouterOS 能工作在 x86、MIPS、PowerPC 和 TILE 平台，路由器至少需要 32MB 内存和 2MB 的 HDD 空间，HDD 用于数据和日志信息存储，RouterOS 可以使用 USB 存储或者挂载硬盘（具体关于存储请参见 RouterOS Store 章节）。

在 RouterOS v4.0 修改为在线用户许可方式：

- Level3 – 10 在线用户
- Level4 – 20 在线用户
- Level5 – 50 在线用户
- Level6 – 无限制用户

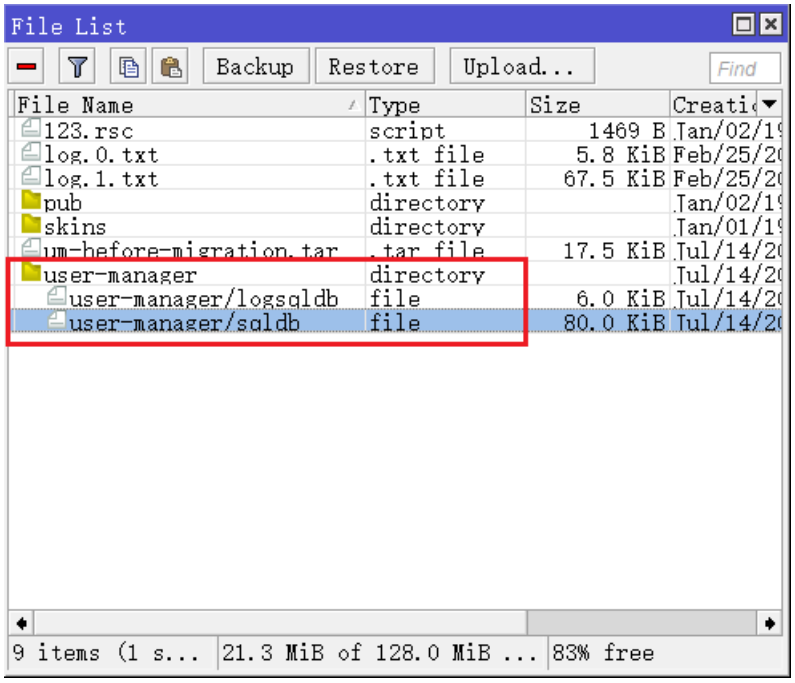
38.1 User Manage 基本配置

User Manager 是集成在 RouterOS 的一个功能，即是一个功能包，使用前首先需要确认 RouterOS 的 system package 里有 user manager 功能包，如下图



如果没有安装功能包，请下载一个与当前 RouterOS 相同版本的 user-manager.nkp，上传到 files 文件根目录下，通过命令重启，RouterOS 会自动安装 user manager 功能包。

安装完成后，会在 File 文件下看到创建的 user-manager 目录，以及 log 和数据的 sql 文件：



创建 Customer 账号

Customer 账号，即 User Manager 的客服账号，用于为客户服务管理的账号，有别于 RouterOS 的 admin 账号，是独立存在于 User Manager 的管理账号。Customer 可以配置对接的 RouterOS 路由器、用户账号管理和 User Manager 的其他设置。User Manager 允许创建多个 Customer 账号，分配不同的管理权限。

注意：User manager v3 有一个默认的 Customer 账号 admin（在 v3 Customers 被称为 Subscriber），密码为空，当 user manager 功能包安装后，第一次通过浏览器访问，可以使用 admin 和空密码登录。（User Manager v3 版本是独立的版本号，该版本主要应用在 RouterOSv2.9.x 和 v3.x 版本，因此请不要将 RouterOS 版本与 User Manager 版本混淆）

随着 RouterOS 升级到 v4.x 版本，User Manager 也升级到 v4，在 v4 版本前 Customers 被称为 subscribers，在 v4 版本安装后，第一个 Customers 账号，需要初始化，不在像 v3 版本默认已经添加"admin"，因此必须通过 RouterOS 终端控制台添加，user-manager 的命令行操作路径在**/tool user-manager**

创建 v3 或 v4/v5 的 Customer 创建进入 **/tool user-manager customer** 目录，如下面创建 admin 账号：

```
[admin@MikroTik] /tool user-manager customer> add login="admin" password="PASSWORD"
permissions=owner
```

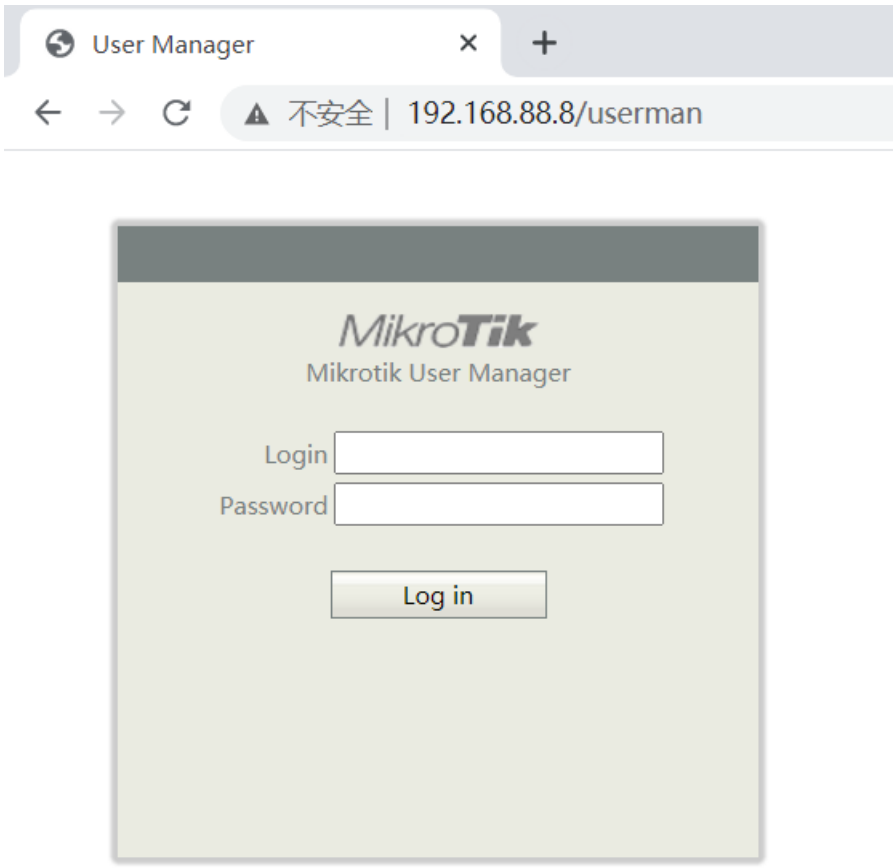
可以使用下面的命令修改“admin”的密码

```
[admin@ MikroTik] /tool user-manager customer set admin password=PASSWORD
```

通过 **print** 命令查看添加或修改的信息。

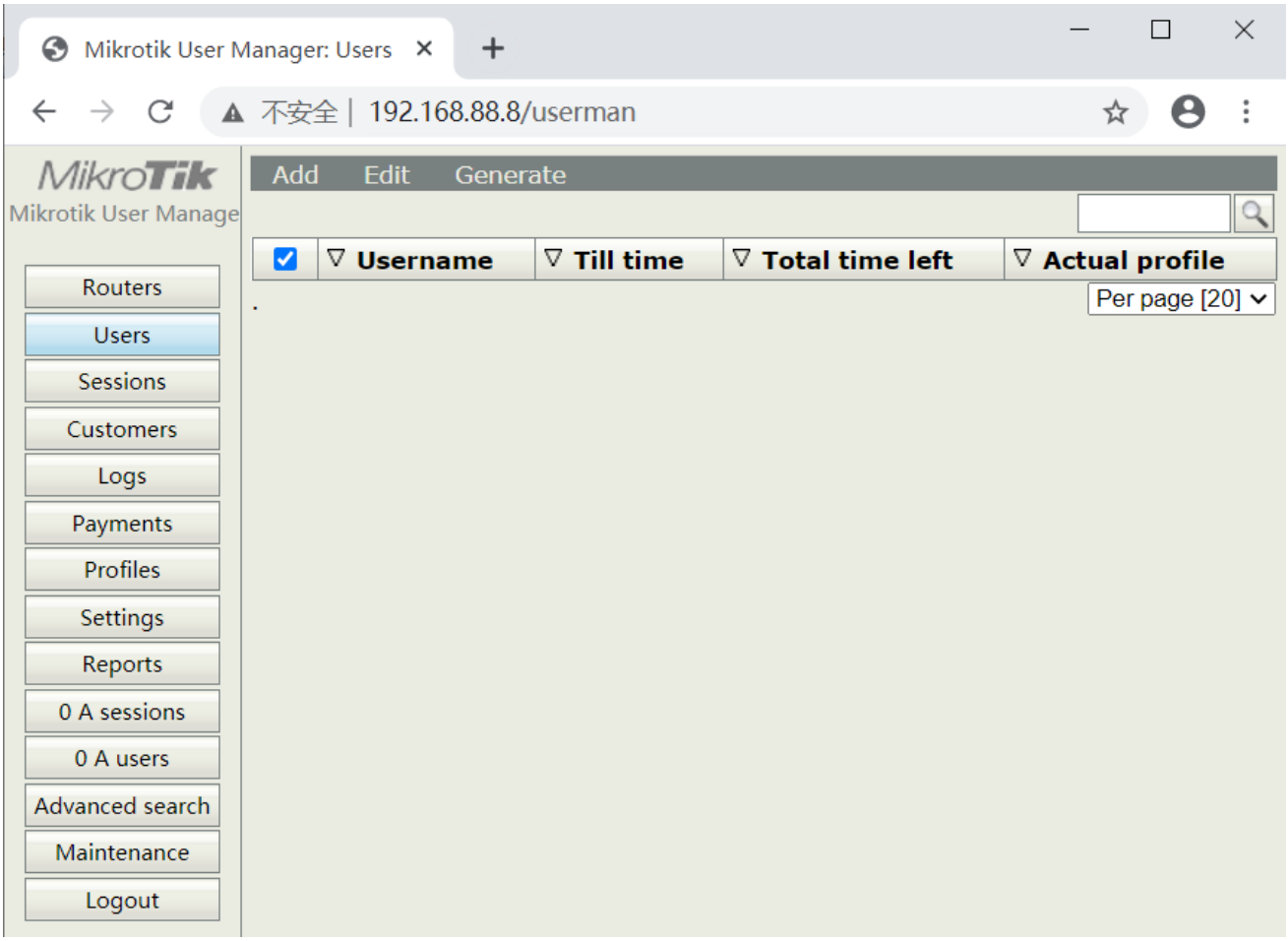
```
[admin@ MikroTik] /tool user-manager customer> print
 Flags: X - disabled
  0    login="admin" password="adminpassword" backup-allowed=yes currency="USD"
       time-zone=-00:00 permissions=owner signup-allowed=no paypal-allowed=no
       paypal-secure-response=no paypal-accept-pending=no
```

在功能包和 Customers 的账号确认后，即可通过 web 页面登录到 User Manager，登录默认是 80 端口（可以在 ip service 里修改 www 的访问端口），访问路径 http://路由器 IP/userman，如下图（建议使用 Chrome 浏览器）：

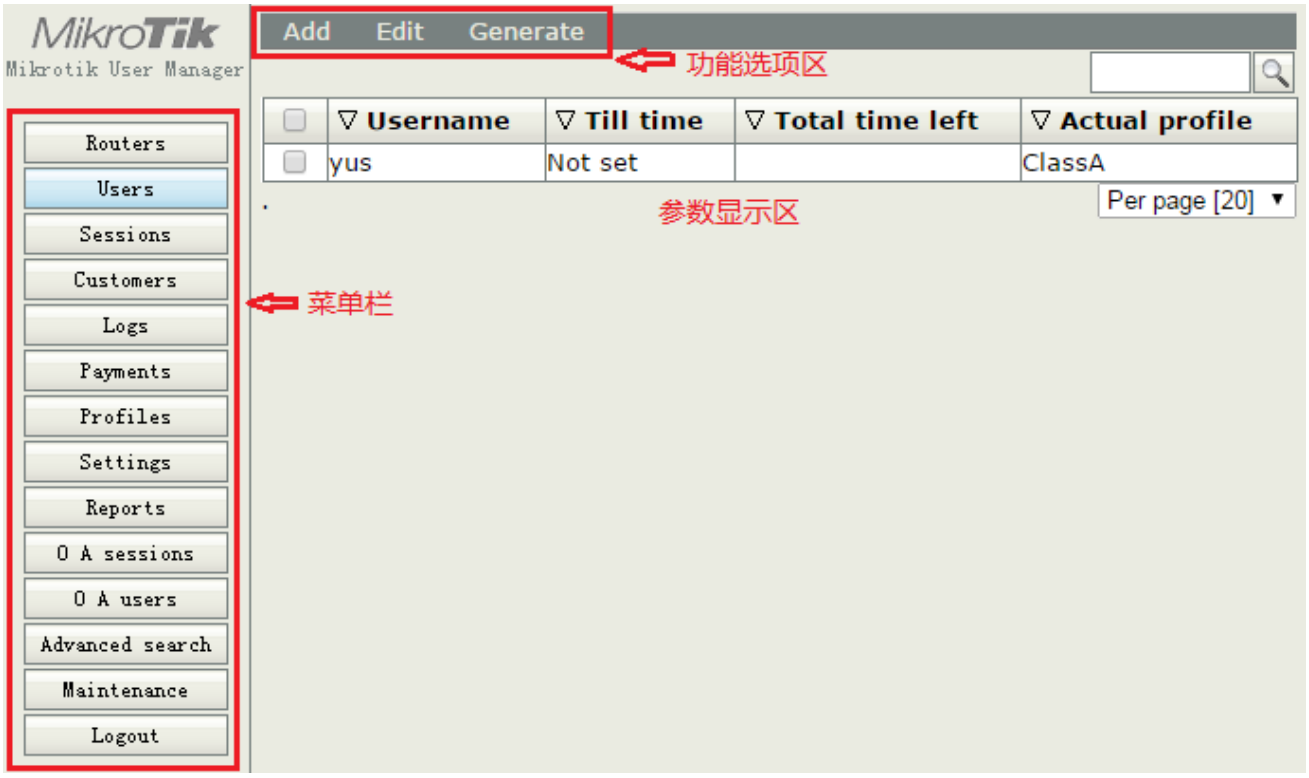


操作接口

User Manager 登录名是在/tool user-manager customer 下设置内容，如账号为 admin，密码为 PASSWARD，登录后接口如下：



User Manager 操作区域划分



菜单栏：

- Routers：连接到 User Manager 的路由器 IP 地址和 Secret 等参数
- Users：用户帐户信息添加、编辑、删除等
- A Sessions：用户认证会话信息
- Customers：添加顾客管理系统，可以建立多个用户管理系统
- Logs：认证日志
- Payments：用户付款记录信息
- Profiles：用户计费策略
- Settings：User manager 网页设置参数、语言设置和网银付款接口设置
- Reports：汇出 User Manager 统计报告
- A users：在线用户
- Advanced search：高级搜索
- Maintenance：数据维护
- Logout：退出

功能选项区：

- Add：添加一个新的规则
- Edit：对选中的规则进行编辑，包括禁用、启用、删除、修改和复位记录数据信息
- Generate：生成备份文件

语言设置

User Manager 默认是英文接口，MikroTik 提供了语言设置功能，提供了默认的模板方便管理者修改语言环境，模板链接：http://wiki.mikrotik.com/images/5/59/En_EN_def.txt

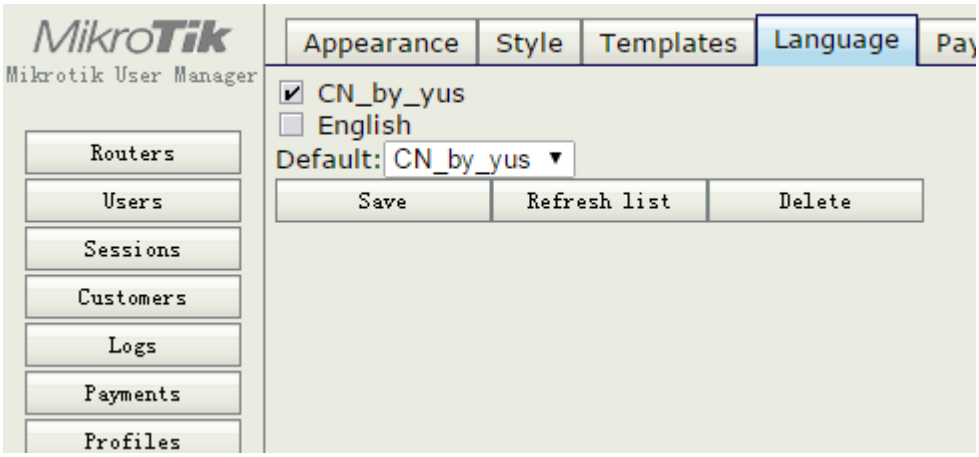
下载模板后，可以通过文本编辑，编辑请使用 UTF-8 编码，如通过文本打开后，编辑里面的内容，msgid 为字符 ID 默认代表对应的字段名，而 msgstr 就是可以修改的语言字符，可以根据这些内容修改。

```
文件(F)  编辑(E)  格式(O)  查看(V)  帮助(H)
msgid ""
msgstr ""
"Project-Id-Version: \n"
"POT-Creation-Date: \n"
"PO-Revision-Date: 2010-02-17 14:42+0300\n"
"Last-Translator: Mikrotik <noreply@mikrotik.com>\n"
"Language-Team: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=utf-8\n"
"Content-Transfer-Encoding: 8bit\n"
"X-Poedit-Language: CN_by_yus\n"

msgid "104bit wep"
msgstr "104-bit WEP"

msgid "40bit wep"
msgstr "40-bit WEP"

msgid "a sessions"
msgstr "当前会话"

msgid "a users"
msgstr "当前在线"
```

在文本中的 X-Poedit-Language 是设置文本在 User Manager language 菜单下的显示名称，默认是：English\n，这里我修改为"X-Poedit-Language: CN_by_yus\n"

编辑完成后，保存档后缀名为“.lng”，文件取名随意，然后上传到 RouterOS 的 files 的根目录下。

上传完成后，如果你正在使用 User Manager 页面，请重新登录一次，然后再进入页面 setting 里的 language 选项，可以看到上传的文件，显示名称为你所设置 X-Poedit-Language 属性名称。多个语言文件可同时存在路由器上，管理员可以根据自己需要选择语言环境。

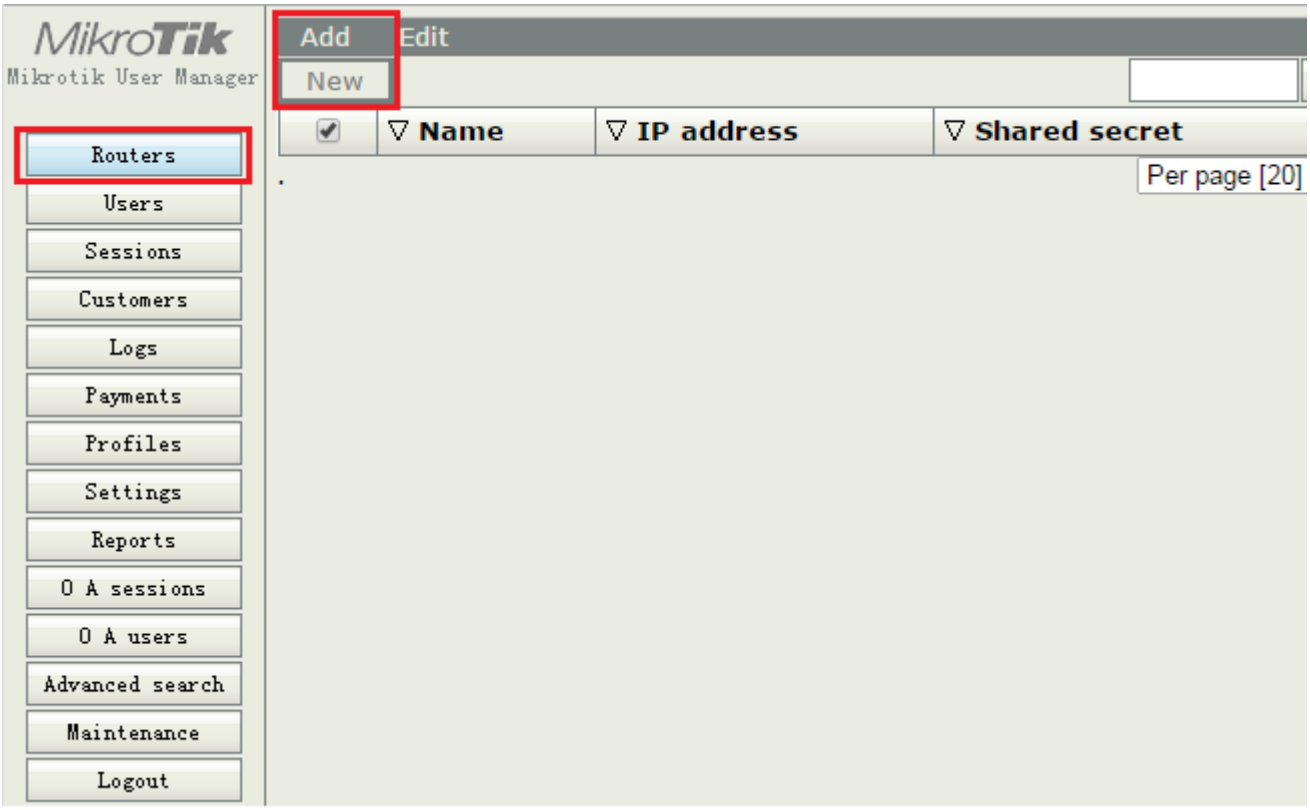


后期的版本在菜单栏的下面新增了语言选项，直接提供语言环境切换。

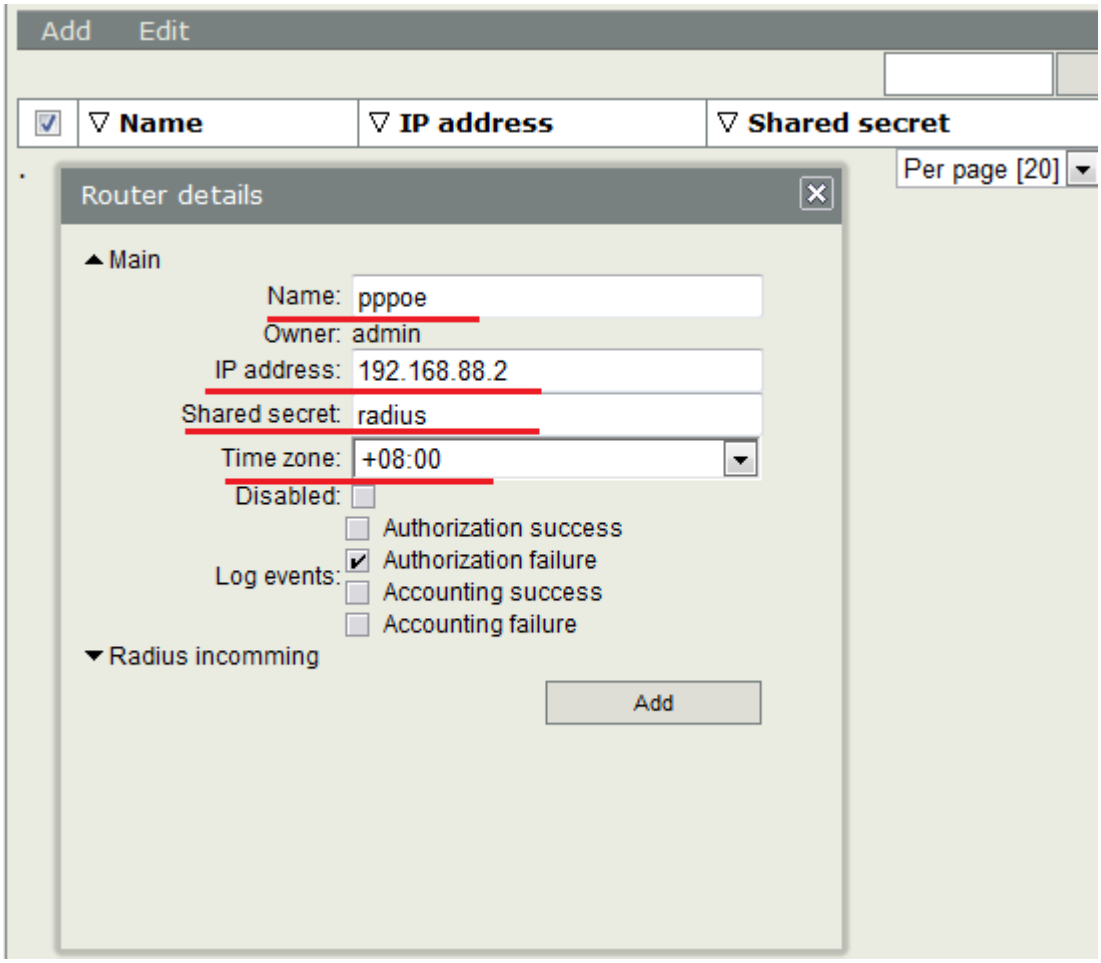
38.2 对接 RouterOS 配置事例

配置 User Manager 与其通信的认证路由器，我们可以在 Routers 菜单下添加需要与 User Manager 通信的认证设备，如 PPP 和 Hotspot 要求用户认证的设备。可以 Add 添加一个新的设备，也可以同时添加多个认证设备（不同等级有所限制）

创建路由器连接，打开 User Manager 页面后，选择 Routers 项，新增 RouterOS 连接

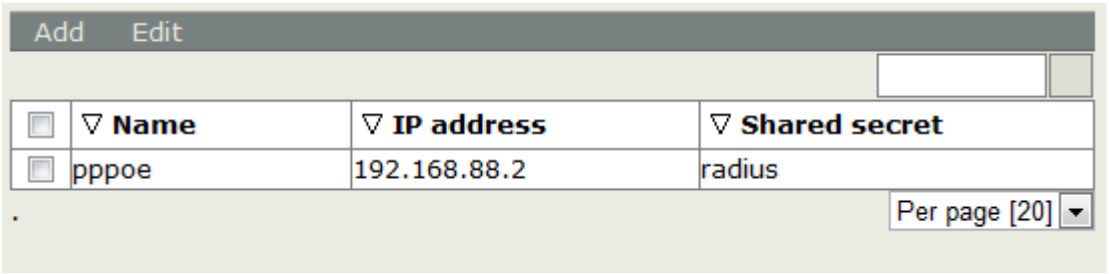


选择 Add->New 后，出现下面的选项，将新增 RouterOS 取名叫 pppoe，与该 RouterOS 对接 IP 地址是 192.168.88.2，Shared Secret（密钥）是 RADIUS，Time zone 选择+08:00，中国在+8 区



Routers 必须配置的参数

- Name：定义一个规则名称
- IP address：设置对端认证设备的 IP 地址
- Shared secret：相互通信的密钥

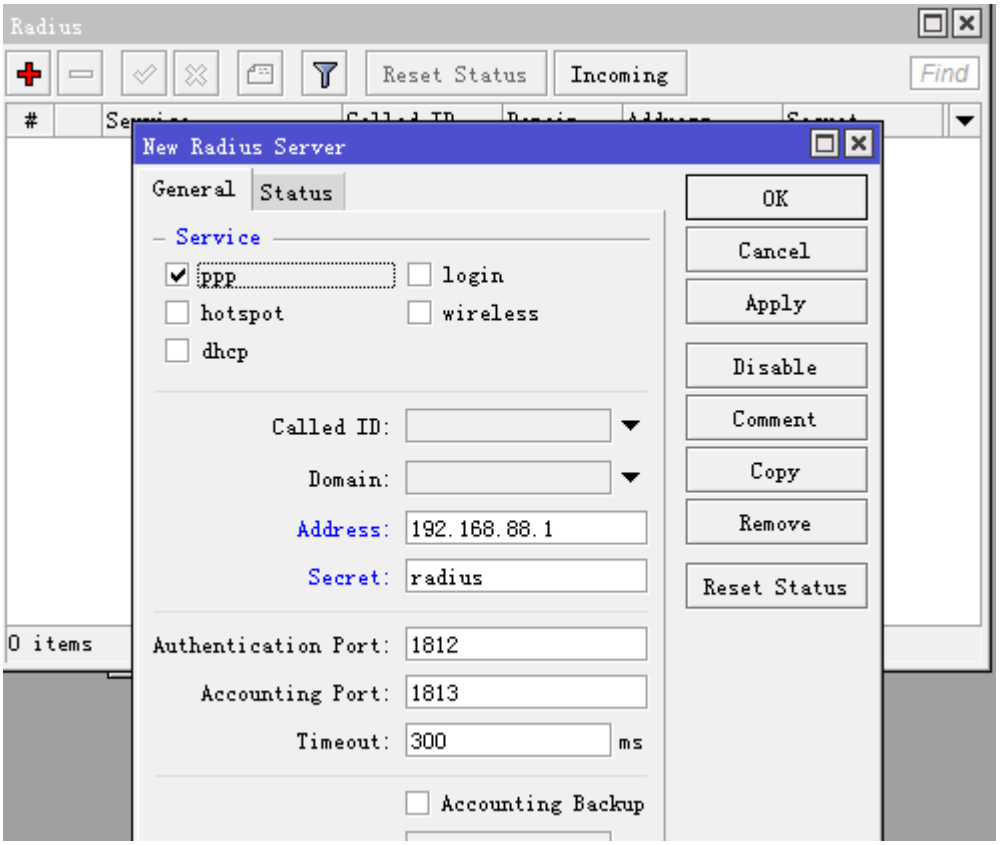


添加完成后，我们的 User manager 就与 192.168.88.2 的 RouterOS 认证设备建立连接，当然这台 RouterOS 也要在 RADIUS 里配置与该 User Manager 的对接参数。

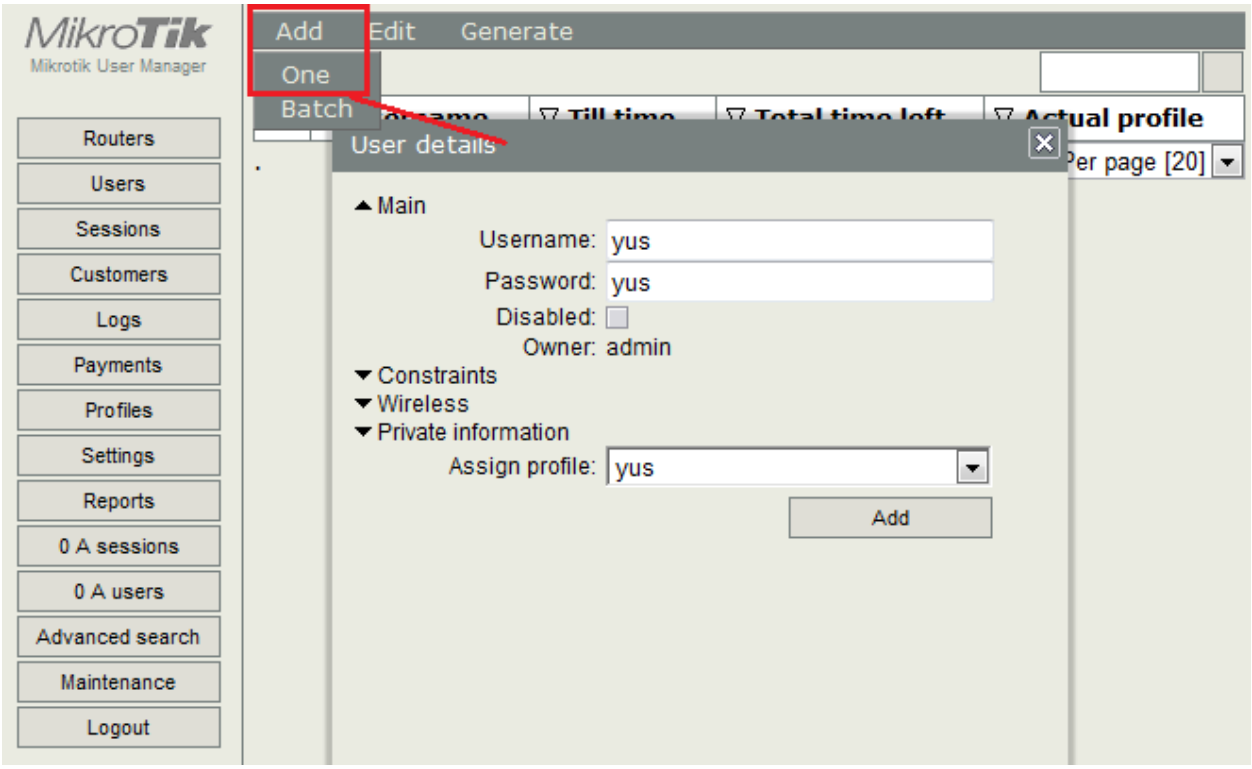
命令行操作

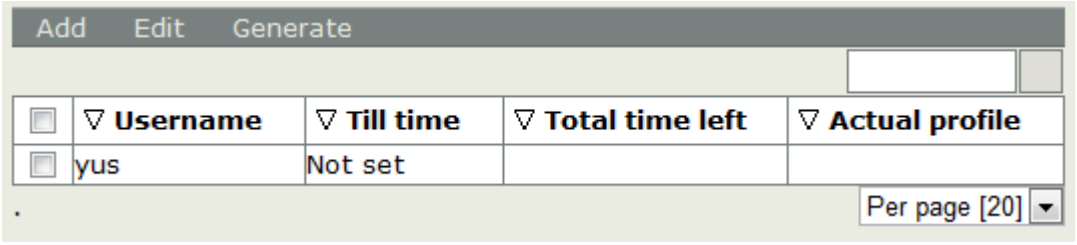
```
[admin@MikroTik] /tool user-manager router> add customer=admin ip-address=192.168.88.2
name=pppoe shared-secret=RADIUS
```

我们进入 RouterOS 的 RADIUS 菜单下，添加一个对应的 ppp 配置如下


Radius
Reset Status
Incoming
Find
New Radius Server
General
Status
Service
ppp
login
hotspot
wireless
dhcp
Called ID:
Domain:
Address: 192.168.88.1
Secret: radius
Authentication Port: 1812
Accounting Port: 1813
Timeout: 300 ms
Accounting Backup
OK
Cancel
Apply
Disable
Comment
Copy
Remove
Reset Status
0 items

添加一个用户账号


MikroTik
Mikrotik User Manager
Routers
Users
Sessions
Customers
Logs
Payments
Profiles
Settings
Reports
0 A sessions
0 A users
Advanced search
Maintenance
Logout
Add
Edit
Generate
One
Batch
Actual profile
Per page [20]
User details
Main
Username: yus
Password: yus
Disabled:
Owner: admin
Constraints
Wireless
Private information
Assign profile: yus
Add

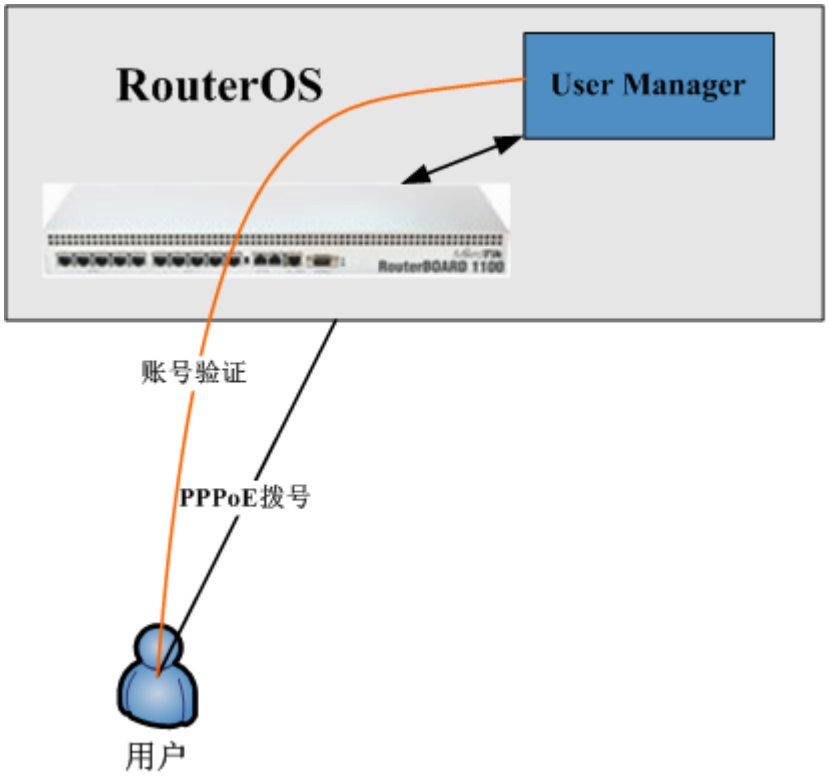


通过命令行操作

```
[admin@MikroTik] /tool user-manager user> add customer=admin name=yus password=yus
shared-users=1
```

38.3 PPPoE 认证的事例操作

下面我们介绍下 User Manager 与 PPPoE 的对接，我们在同一台 RouterOS 上建立 PPPoE 和 User Manager，即我们不在从/ppp secret 里建立为 PPPoE 用户拨号的账号，而是在 User Manager 里建立用户账号，

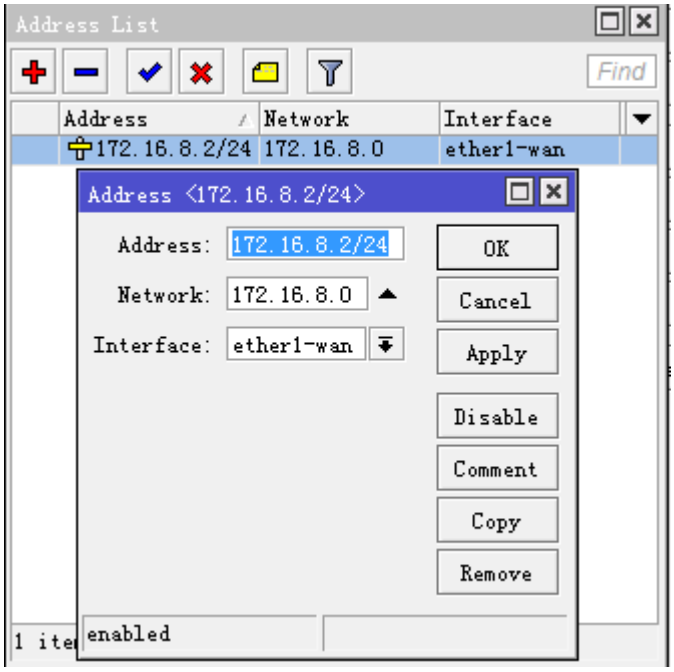


RouterOS 的配置如下：

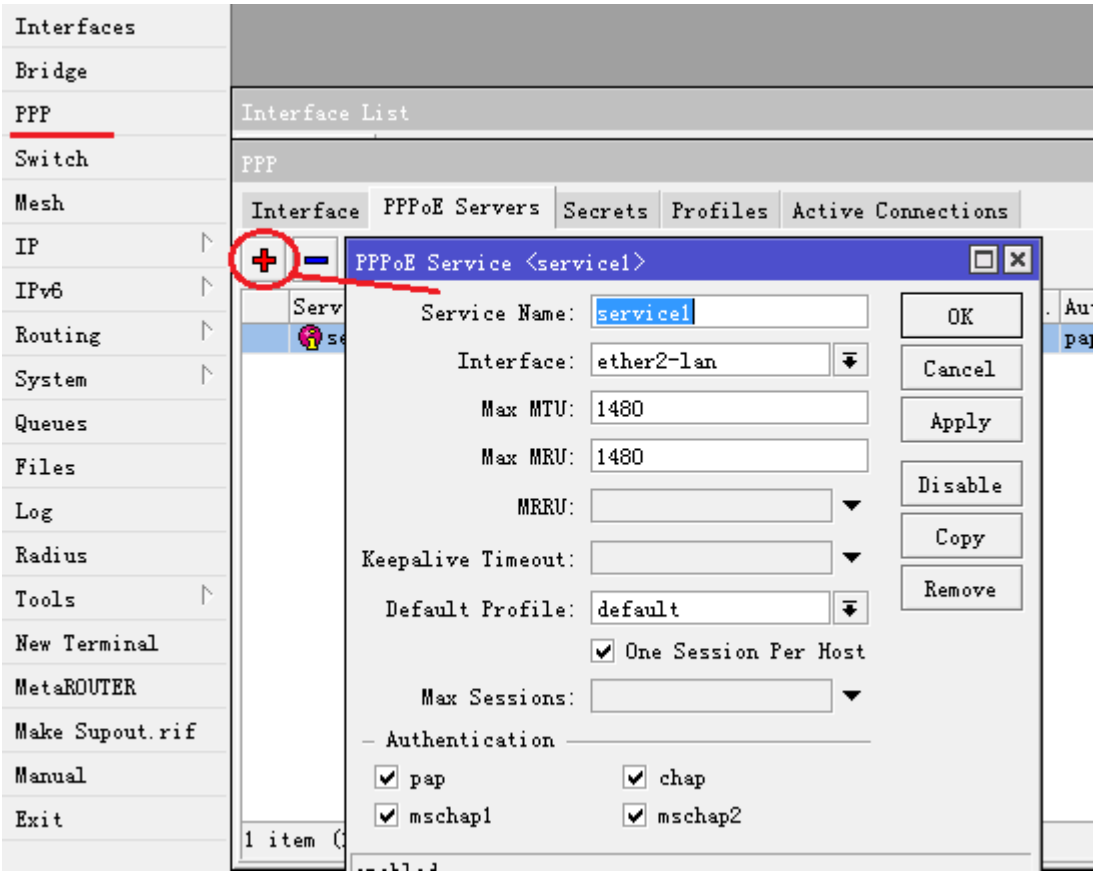
- WAN 口 IP 地址为 172.16.8.2，网关是 172.16.8.1
- LAN 口采用 PPPoE 验证，可以不需要设置 IP 地址
- 启用 User Manager，并与本地的 RouterOS 连接，用户账号通过 User Manager 分配

第一步：配置 RouterOS 的 PPPoE 认证

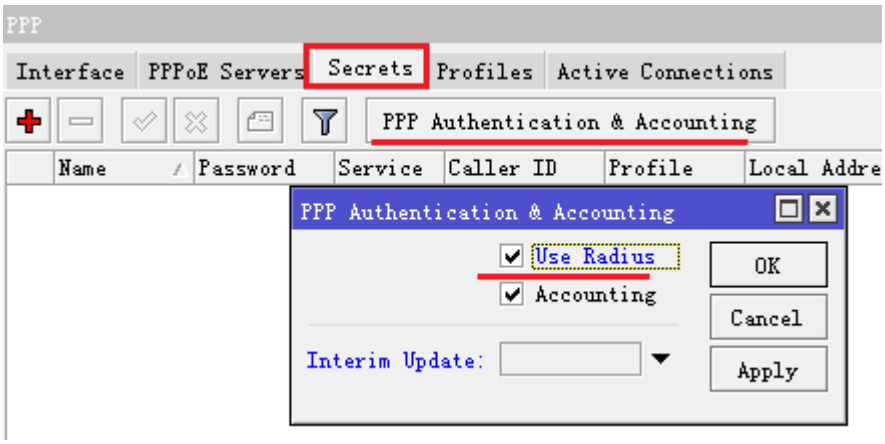
具体的 PPPoE 可以参考 PPPoE 一章，这里我们作简单的关于与 User Manager 配置的介绍，我们先进入 ip address 添加 WAN 口 IP 地址



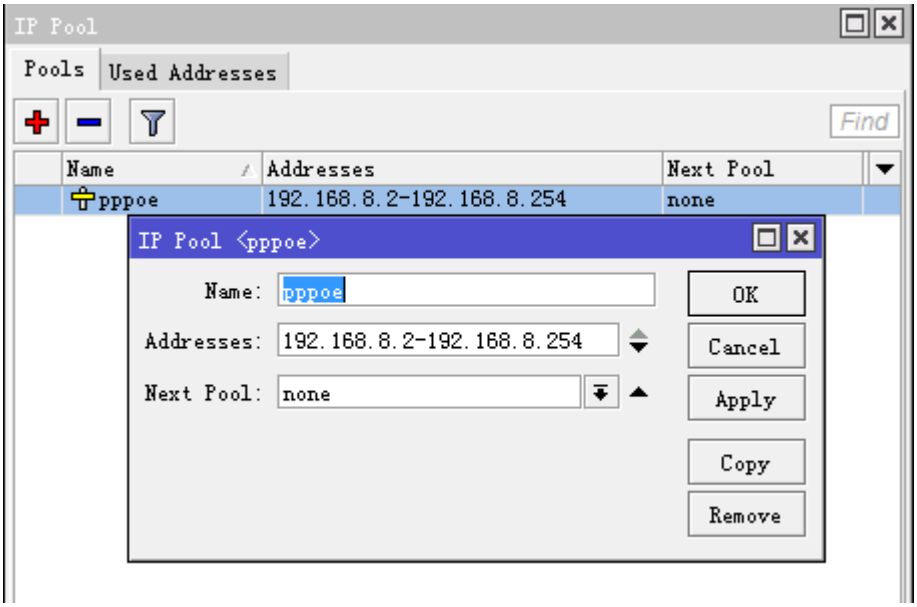
我们配置 PPPoE 服务器，进入 ppp 选择 PPPoE Service，添加并启用一个 interface=ether2-lan 的 PPPoE 服务器



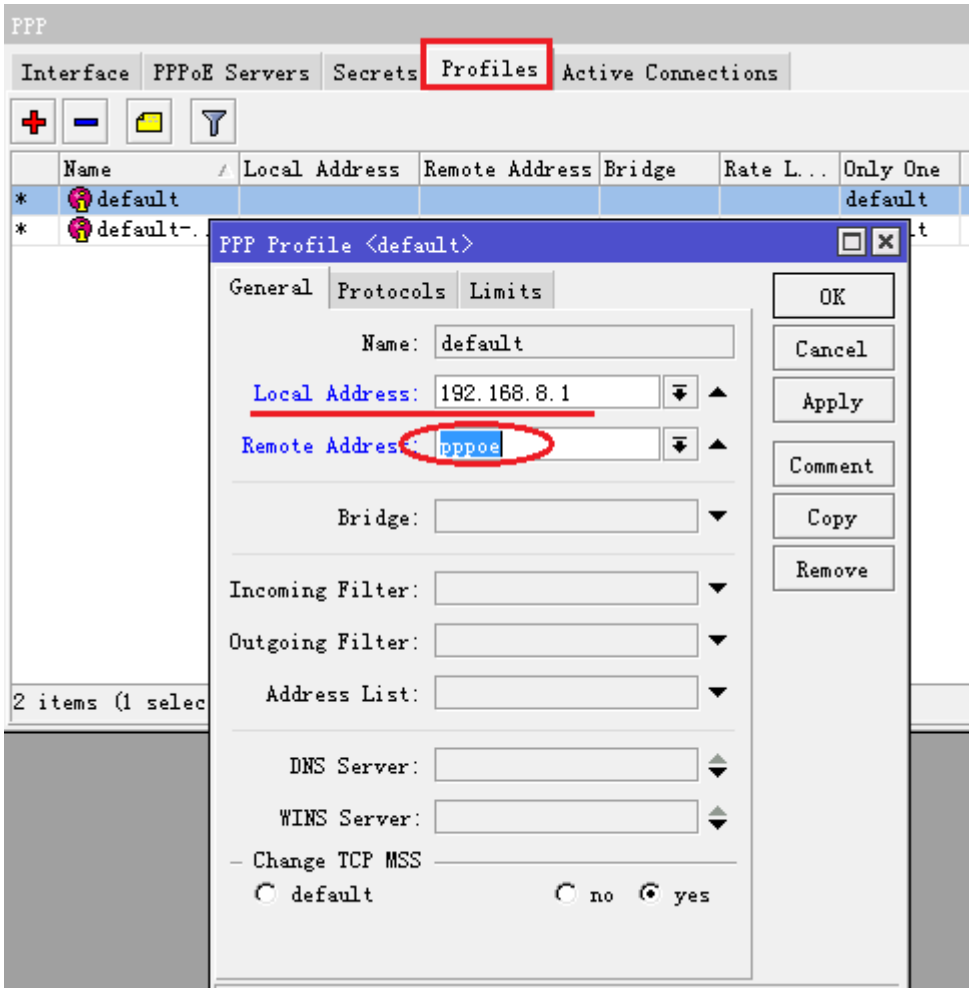
启用 PPPoE 服务器后，我们需要进入 Secrets 里将 PPP Authentication & Accounting 设置里的 Use RADIUS 选择上



我们需要指定分配给用户的 IP 地址池，我们在 ip pool 里定义地址池，我们给客户端分配的地址范围是 192.168.8.2-192.168.8.254，取名 pppoe



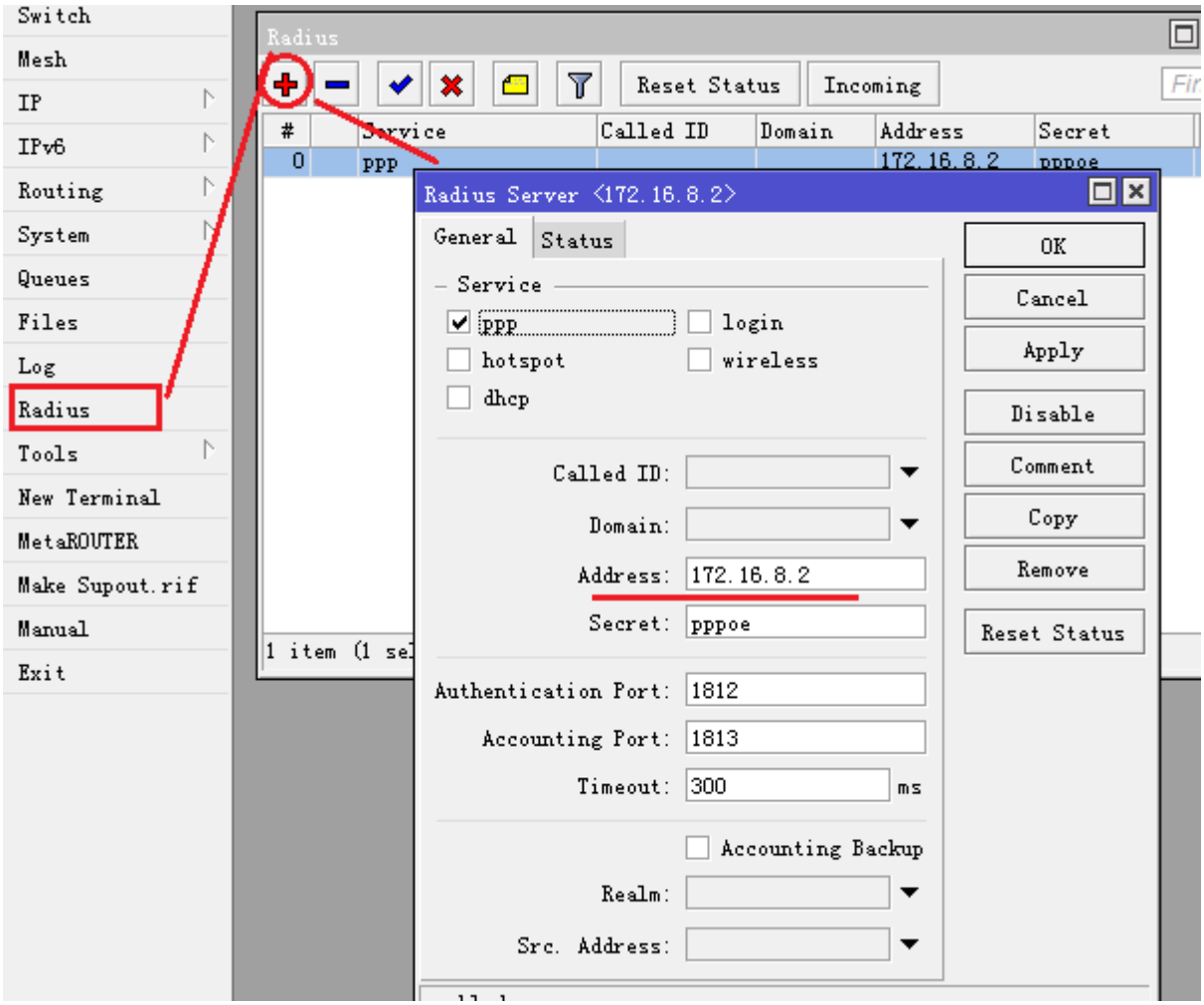
Pool 定义完成后，我们在 ppp profile 里 default 设置用户地址分配，我们设置客户端的网关是 192.168.8.1，



注：这里的 remote-address 我们可以选择设置，因为我们启用了 User Manager，用户的 IP 地址可以由 User Manager 来分配，在之后的内容会提到如何操作

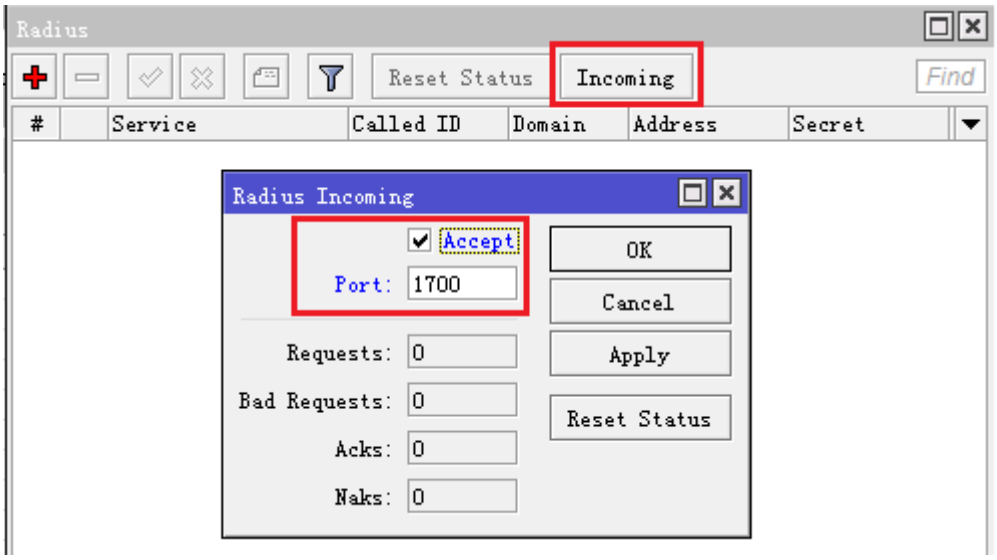
以上基本完成 PPPoE 服务器的配置，当然你需要设置其他的路由网关和 nat 转换，这里不再做介绍

第二步、配置 RouterOS 本机的 RADIUS 连接参数，设置 service 为 ppp，address=172.16.8.2，secret=pppoe



由于是 User Manager 与 RouterOS 在同一台设备上，我们使用 RouterOS 的 WAN 地址作为 RADIUS 的连接地址，我们也可以使用回还地址 127.0.0.1

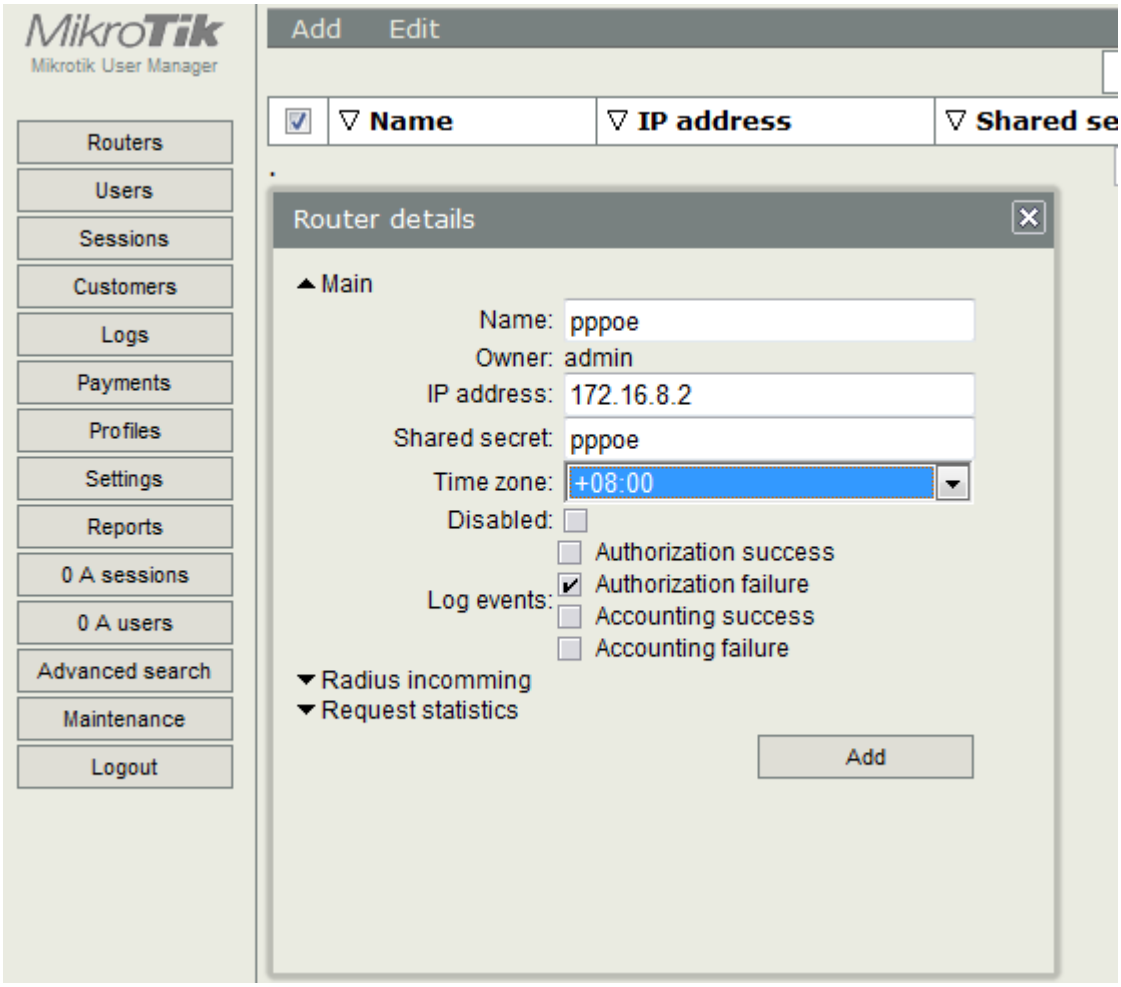
配置 incoming 参数，默认端口是 1700，incoming 参数是由 RADIUS 向 RouterOS 发送指令请求，例如在 User Manager 中剔除在线用户



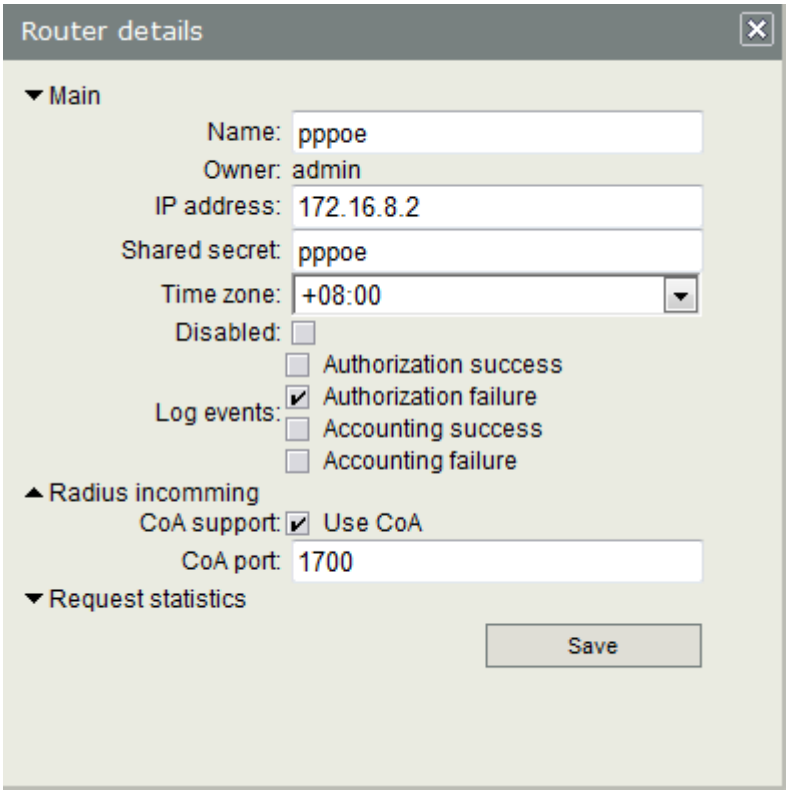
第三步，User Manager 配置

之前我们已经配置了 RouterOS 的 PPPoE 和 RADIUS 参数，现在我们配置 User Manager 的参数，让 PPPoE 用户验证和 User Manager 建立完整的通信。

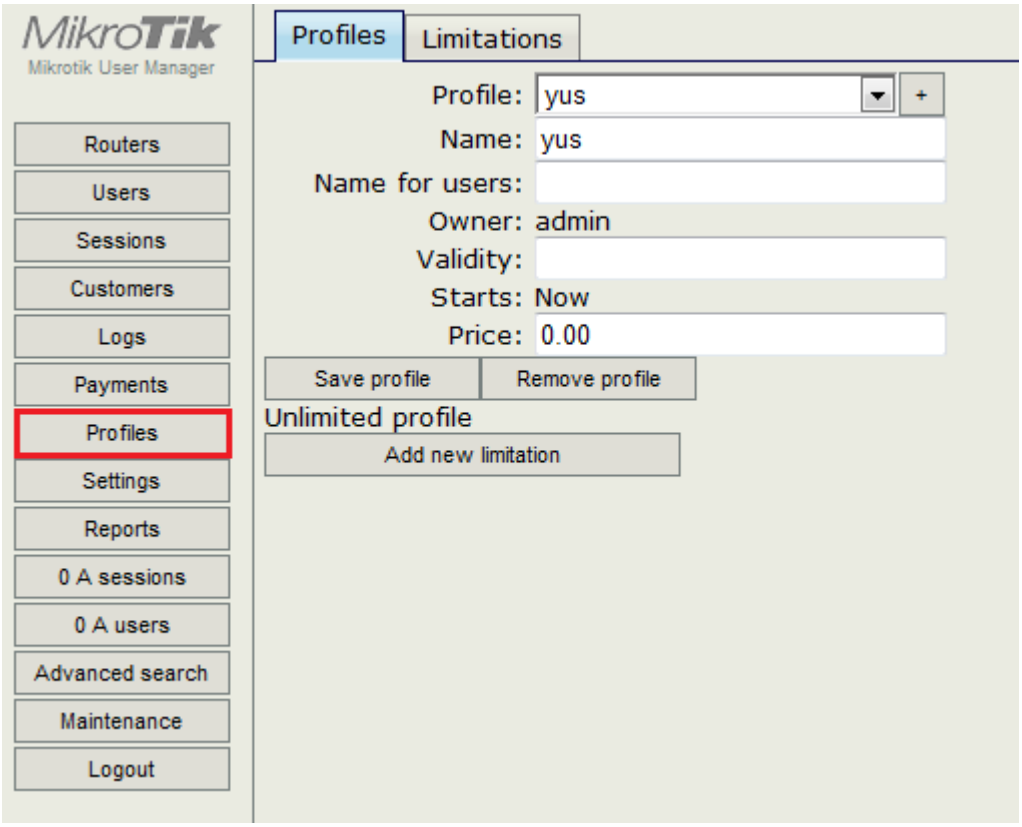
我们先进入 Router 里添加参数，IP address 同样是本地的 WAN 口 172.16.8.2，Shared secret 同样是 pppoe



在 RADIUS incomming 里开启端口

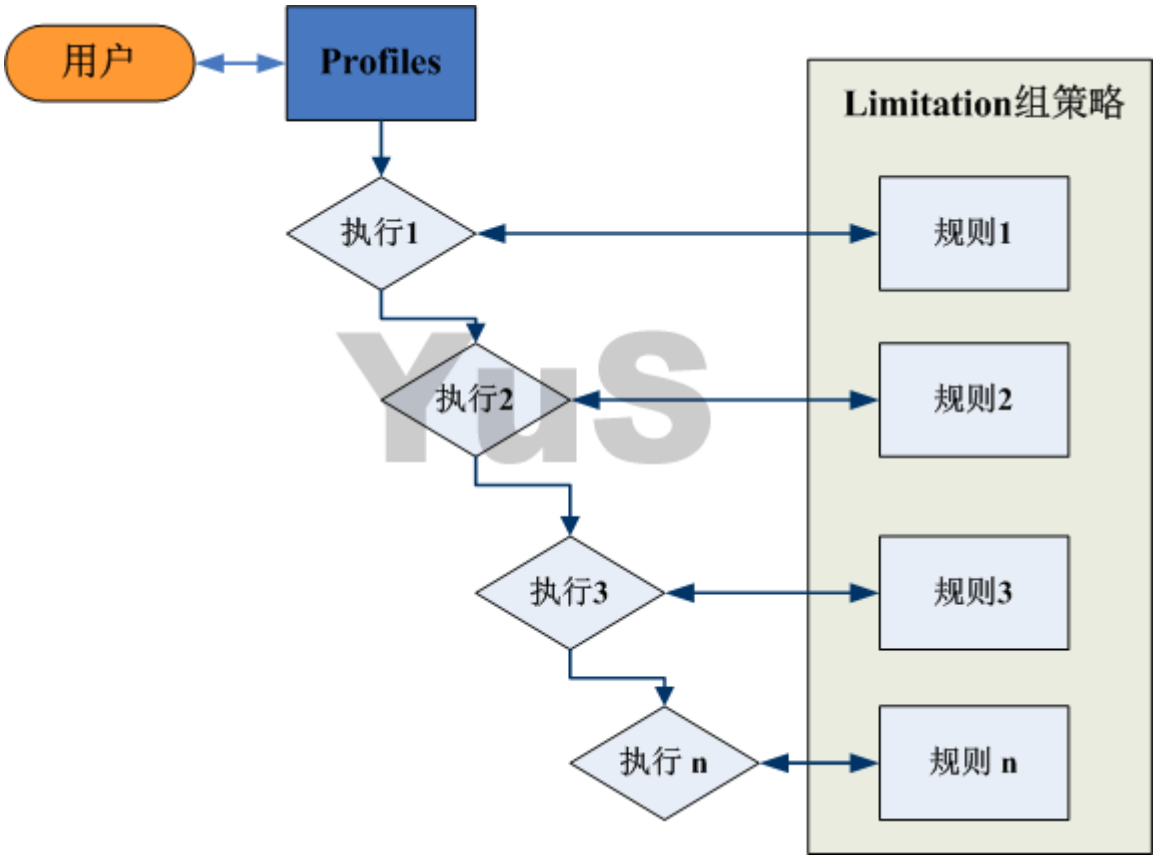


添加完成 Router 参数后，基本完成了 PPPoE 验证与 User Manager 的对接，接下来就是添加用户账号，在添加用户账号前我们需要设置对应 proflies 规则



Profiles 主要定义用户的组策略，包括使用时长，计费价格、流控策略和用户周策略

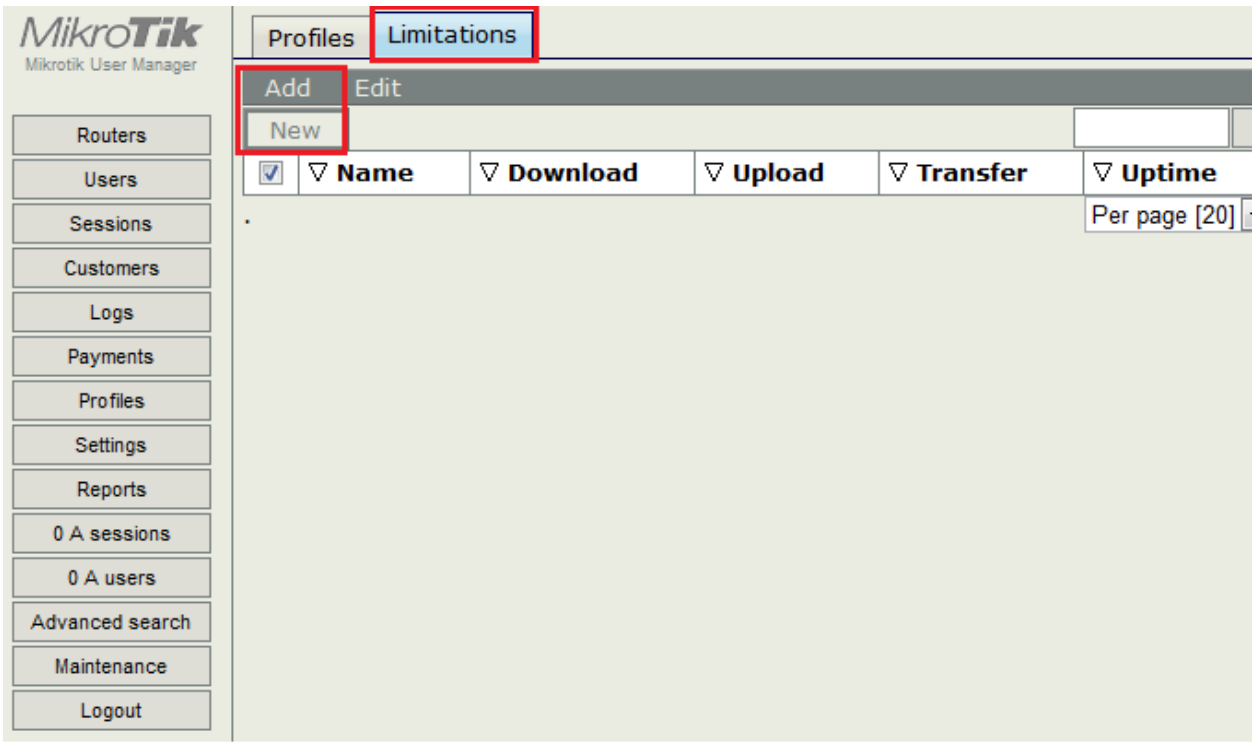
新的 User Manager 与以往的不同，引入了 Profiles 策略组，Profiles 组策略包括 Limitation 限制策略，我们可以建立多条 Limitation 策略，这些策略可以由 Profiles 调用，并设置他们的执行周期，即可以设置周一到周日我们可以选择执行那一条 Limitation 策略，实现不同时段的服务，如下图



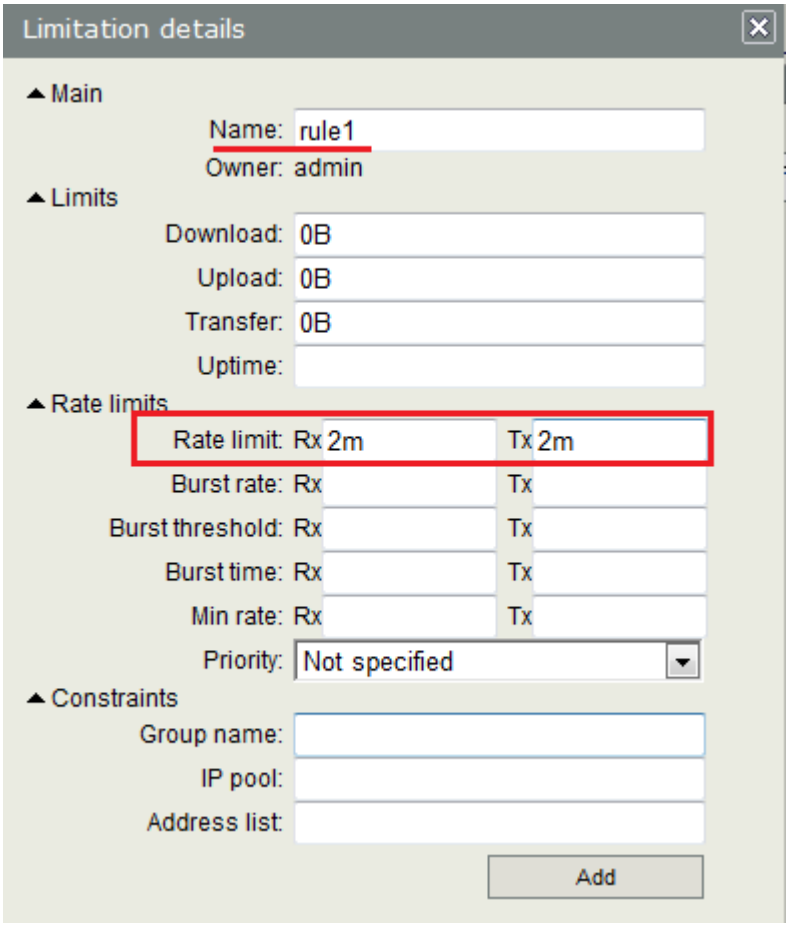
Limitation 的规则在添加后，并没有实际作用，只有在被 profiles 选择后才被调用，同时 User 里的用户账号也调用对应的 Profiles 策略，即一环扣一环的策略方式（其实在后面开发的 NAT RADIUS 也是如此，引入功能和策略上还要比这个复杂点，可惜没有找到好归属）

我们举一个例，星期一到星期五，我们给用户 2M 带宽，星期六和星期天带宽是 3M，Profile 策略名为 YUS，分配给用户是 yus，有效期 30 天（30d）

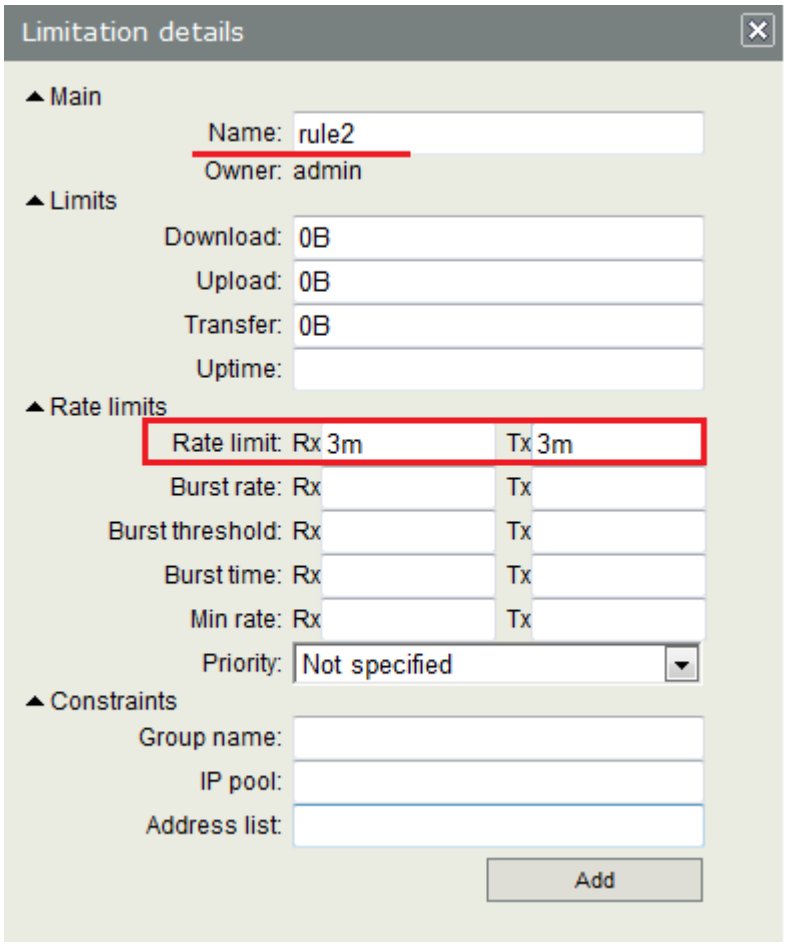
我们首先要在 limitation 里添加 2 个规则，定义星期一到星期五带宽 2M 为 rule1，星期六到星期天 3M 为 rule2，进入 Profiles 的 limitations

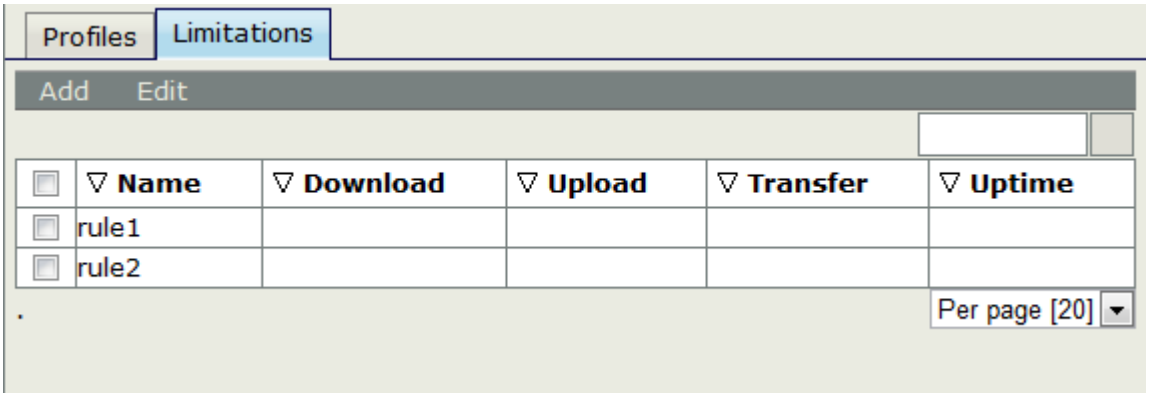


我们添加第一条规则，取名 rule1



添加第二条 3M 的规则，取名 rule2





在 limitation 规则里有几个参数，这些参数有助于对用户的管理

Limits 栏是定义用户使用的流量和时间限制，非我们认为的带宽速率控制，内容包含如下

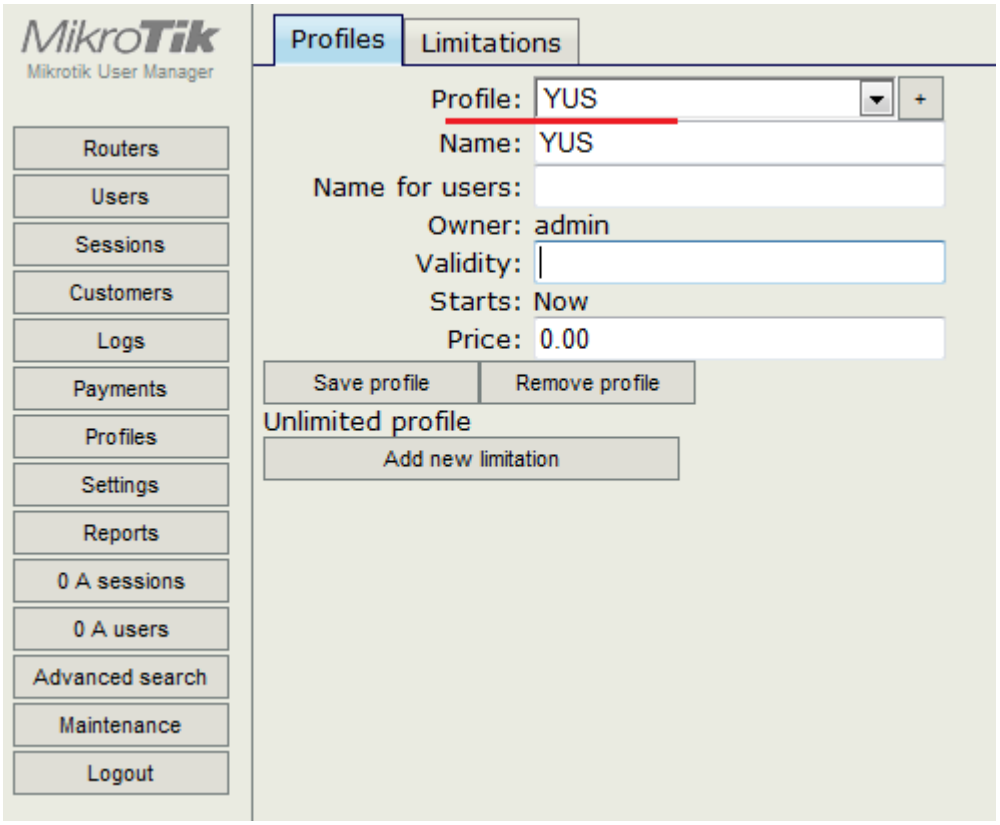
- Download 用户下载流量限制
- Upload：用户上传流量限制
- Transfer：用户总流量限制
- Uptime：用户上线时间限制

Rate Limits 栏，包括了带宽控制，我们一般设置 Rate Limit 值，这个与 queue 里的 Max-limit 相同，mini limit 与 limit-at 相同，其他参数是 Burst 值，具体参数请参考 Queue 章节

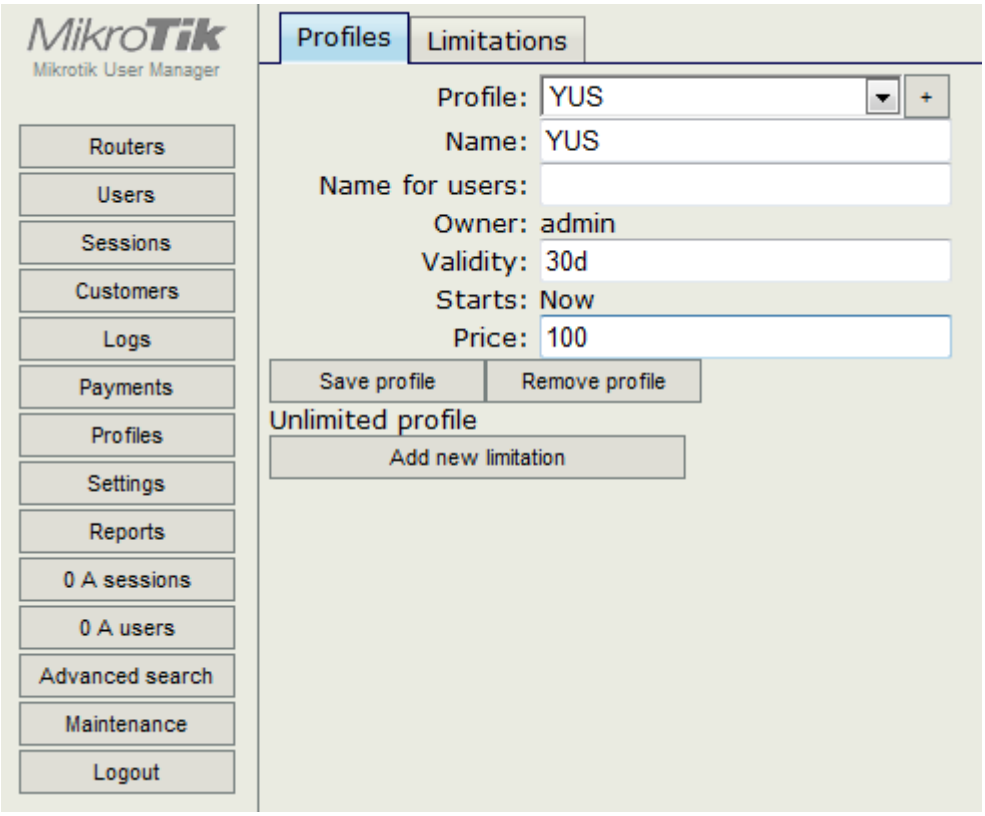
Constraints 栏，是约束限制，包括了 Hotspot 的组策略，IP Pool 地址池和 address-list，这些参数都是可以从 RouterOS 里直接调用

- Group：对应的是 Hotspot 的 Profiles，直接输入对应的 Hotspot Profiles 名称
- IP Pool：对应 RouterOS 的 ip pool 值
- Address-list：可以自定义地址列表，当定义后，在线用户会自动添加到对应 RouterOS 的 ip firewall address-list 中

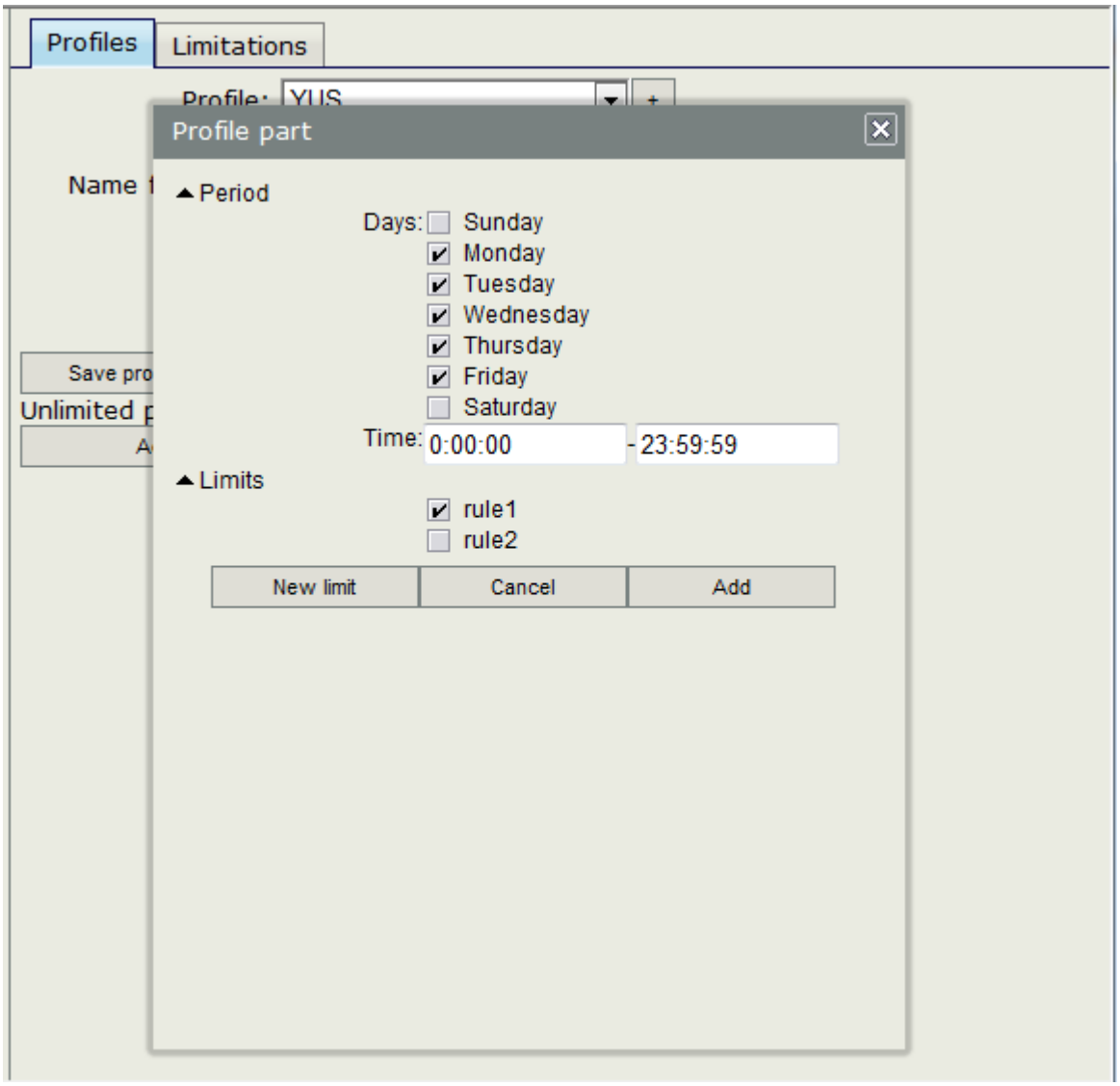
我回到 profile 页面，并添加一个 YUS 的 Profile 规则



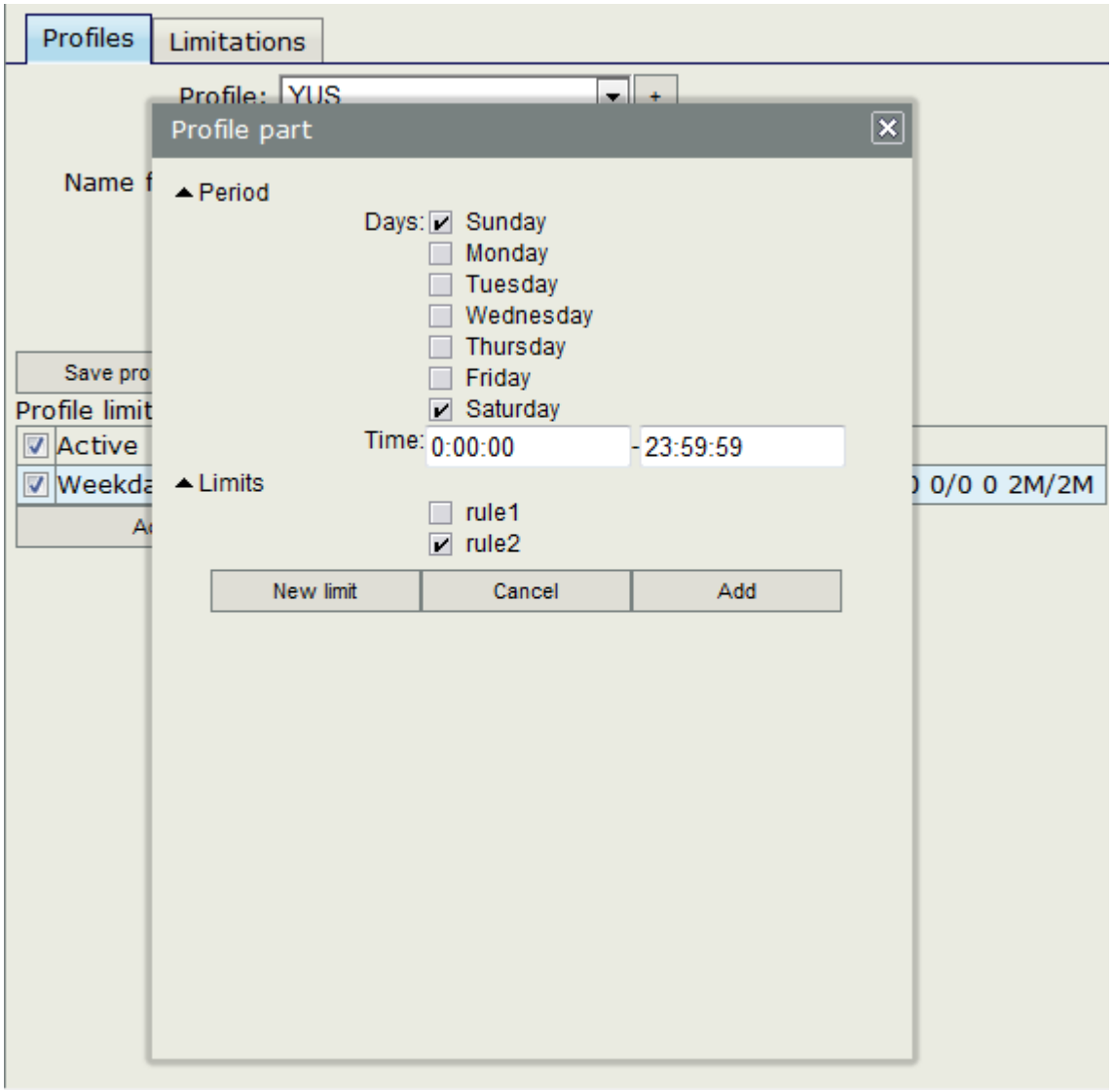
Validity 是有效期，我们设置 30d（30 天），Price：当然是给用户充值的金额，我们可以设置 100 元



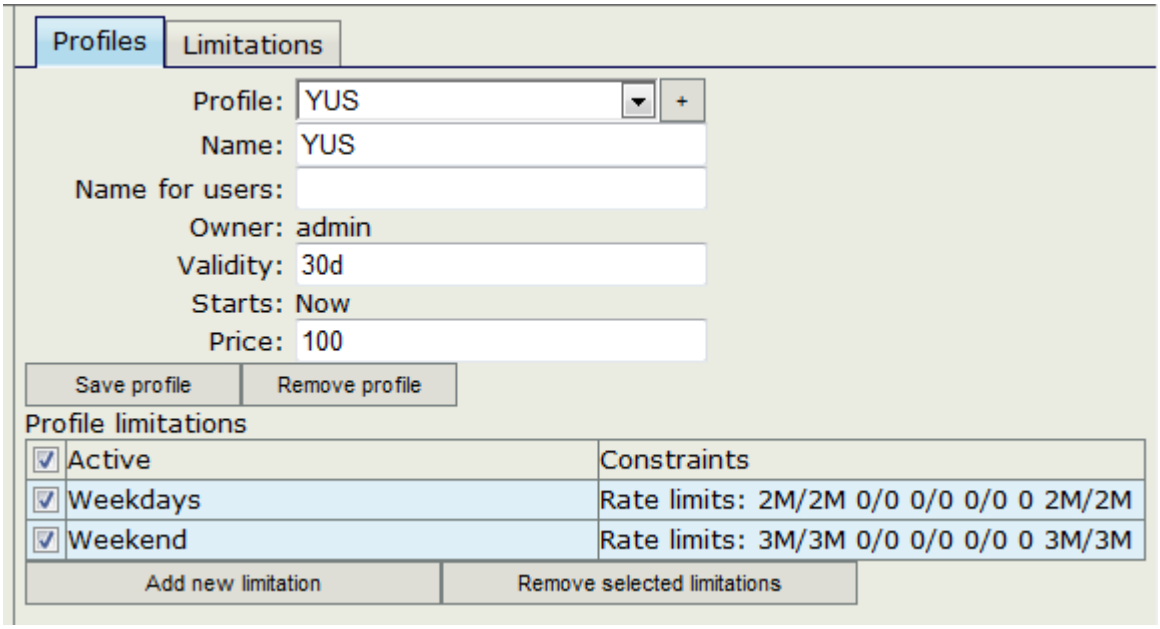
当设置完成后，我们在下面的 add new limitation 添加我们的组规则，首先我们添加星期一到星期五的策略，选择 rule1



接下来定义星期六和星期天的规则

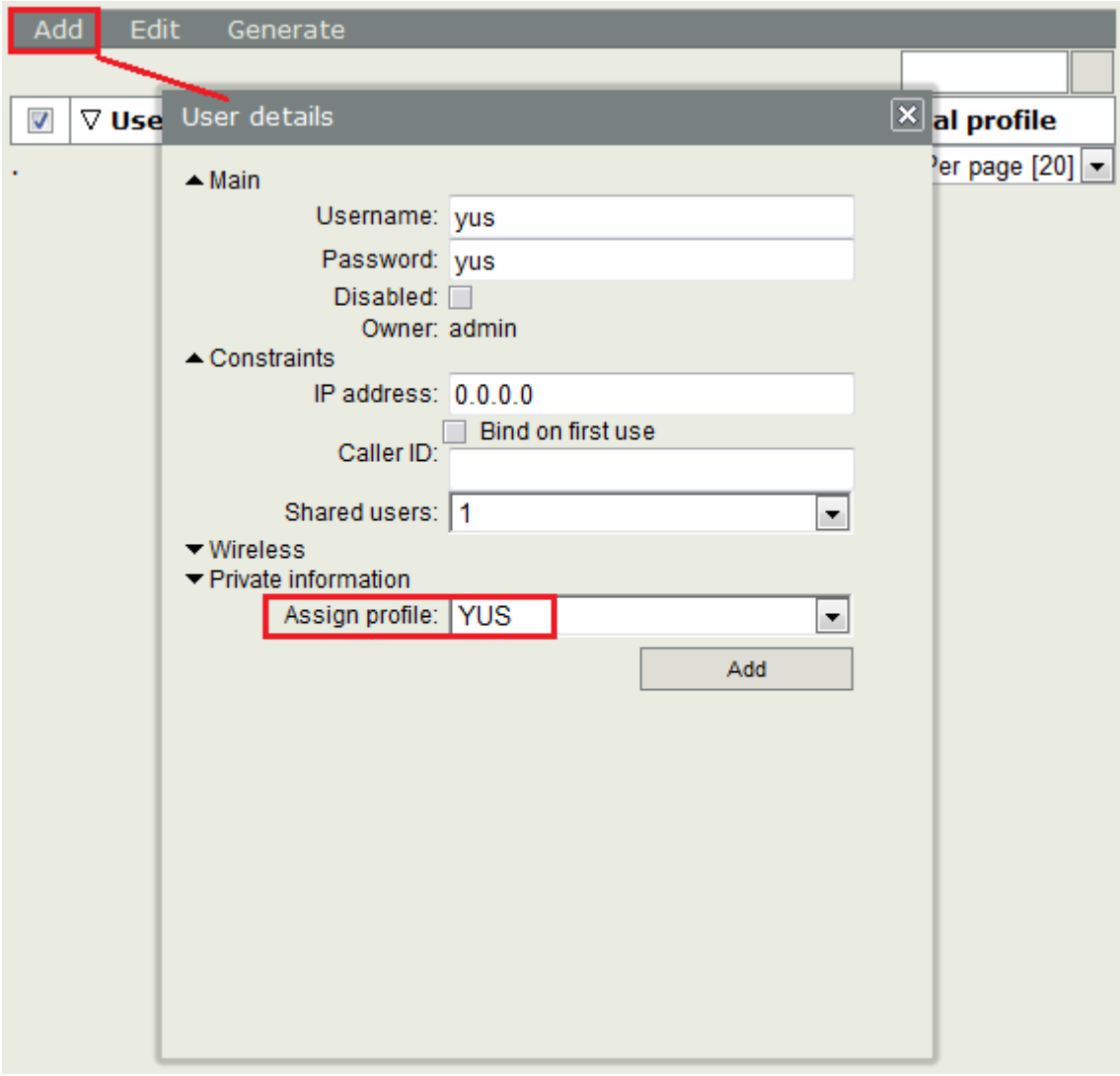


定义完成后，如下所示



注：你可以设置更灵活的时间策略，比如从一天的早上 10:00 到晚上 23:59 带宽是 2M，00:01 到 9:59 带宽是 3M

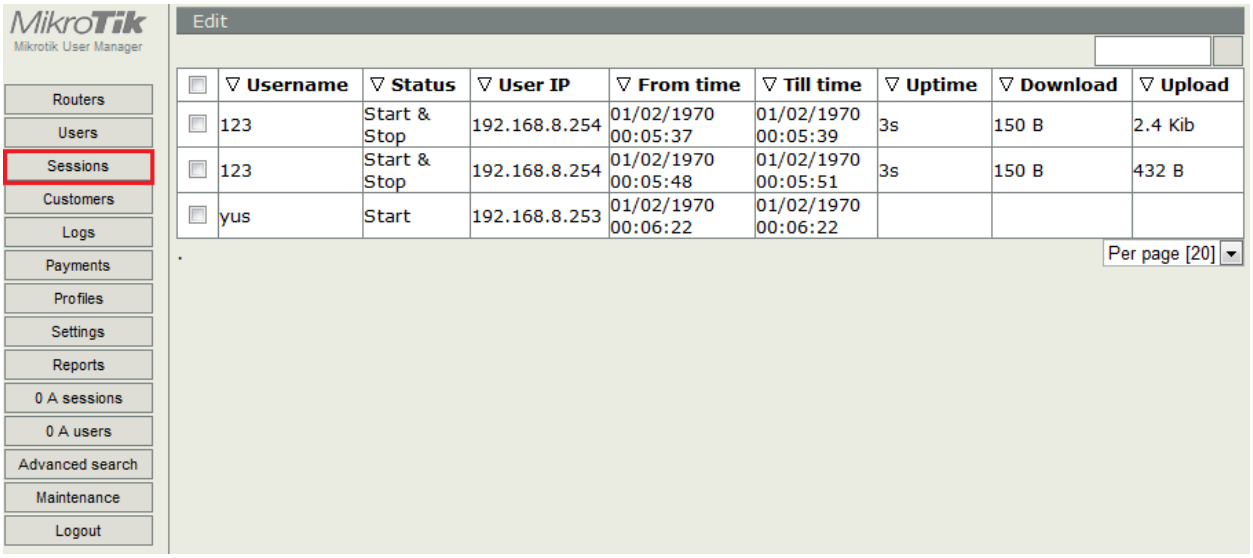
Profile 定义完成后，我们就在 Users 里添加我们需要新增的账号，取名 yus，并选择 Profile 为 YUS



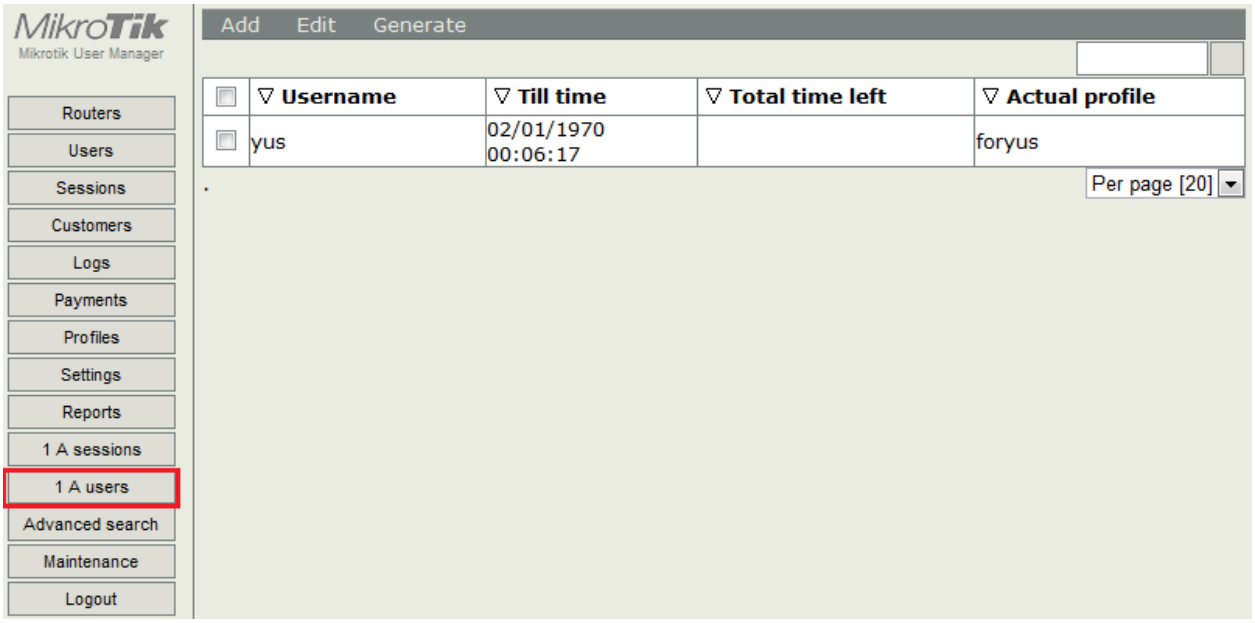
在 User 添加规则时，我们可以看到 constraints 有 3 个参数，他们可以根据你的需要定义

- IP address：为该用户手动分配 ip 地址
- Bind on first use：当用户第一次使用时就绑定他的 IP 或 mac 地址到 Caller ID 里（PPTP、L2TP 等绑定 IP，PPPoE 用户绑定 MAC 地址）
- Called ID：绑定用户的 IP 或者 MAC 地址
- Shared users：该账号共享使用的用户数

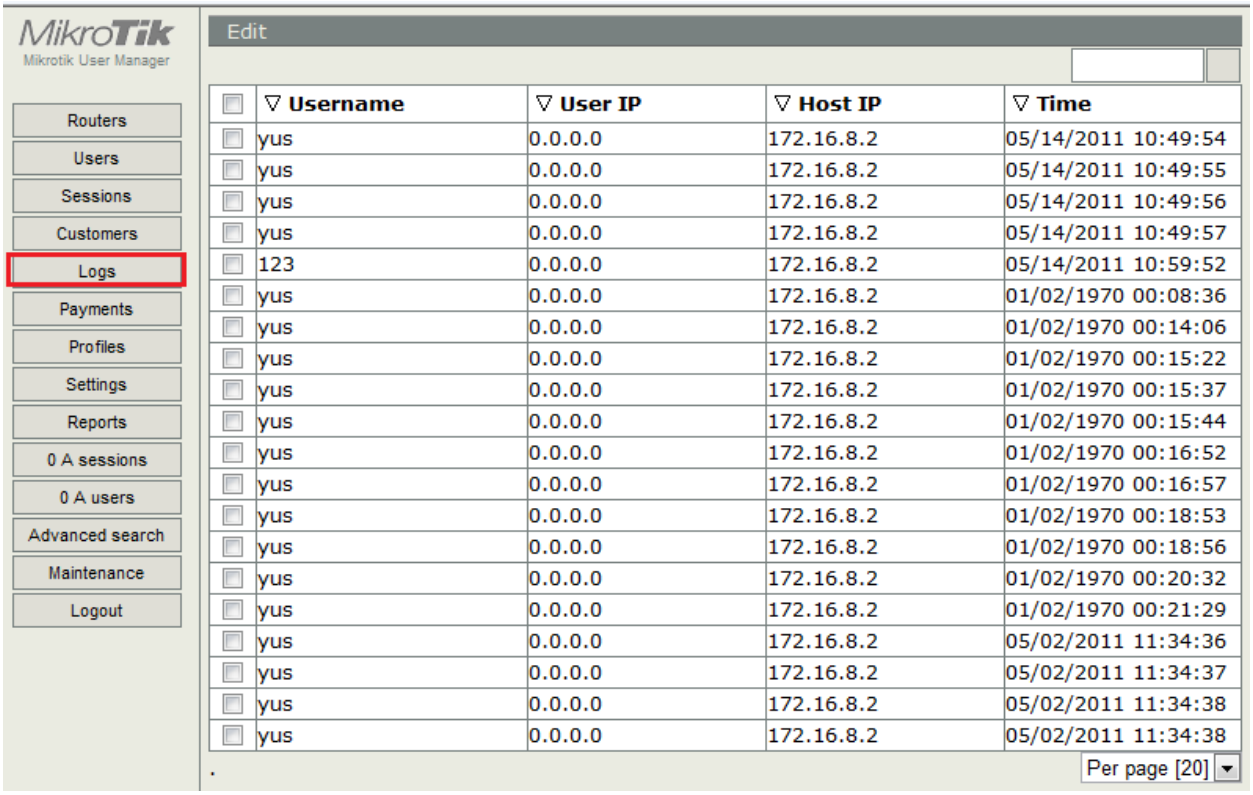
这样从设置 User Manager 连接到添加策略和用户账号基本完成，之后可以使用 yus 账号登录。我们可以在 Sessions 菜单里查看账号登录情况



在 A users 里可以看到当前在线用户

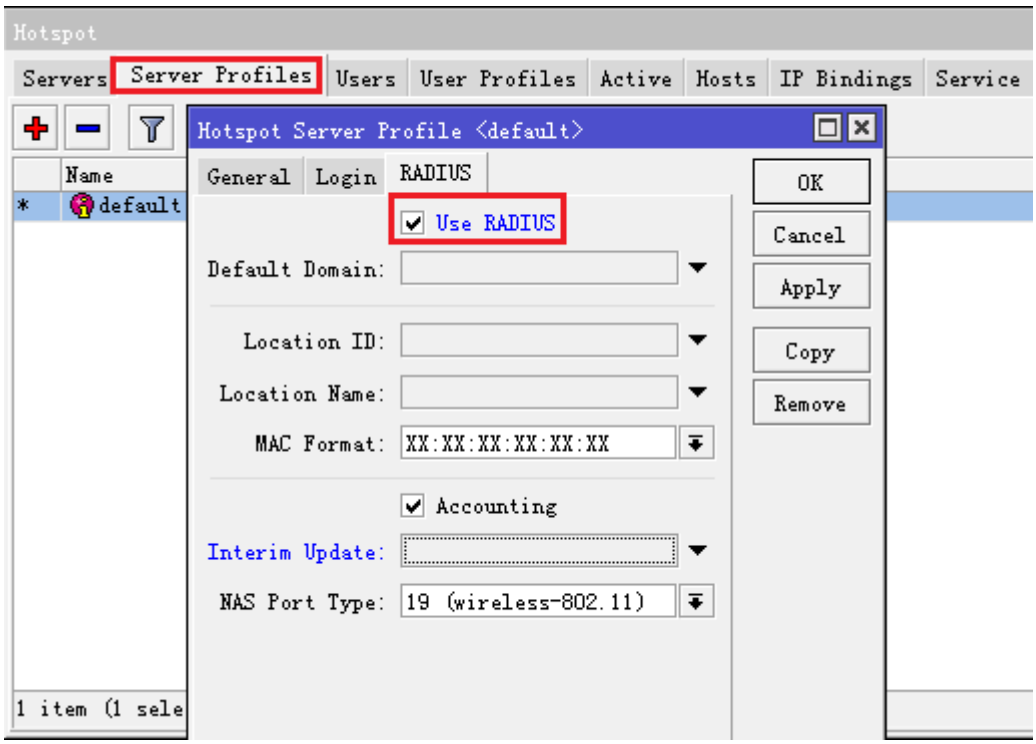


我们可以在 log 中查看用户登录日志

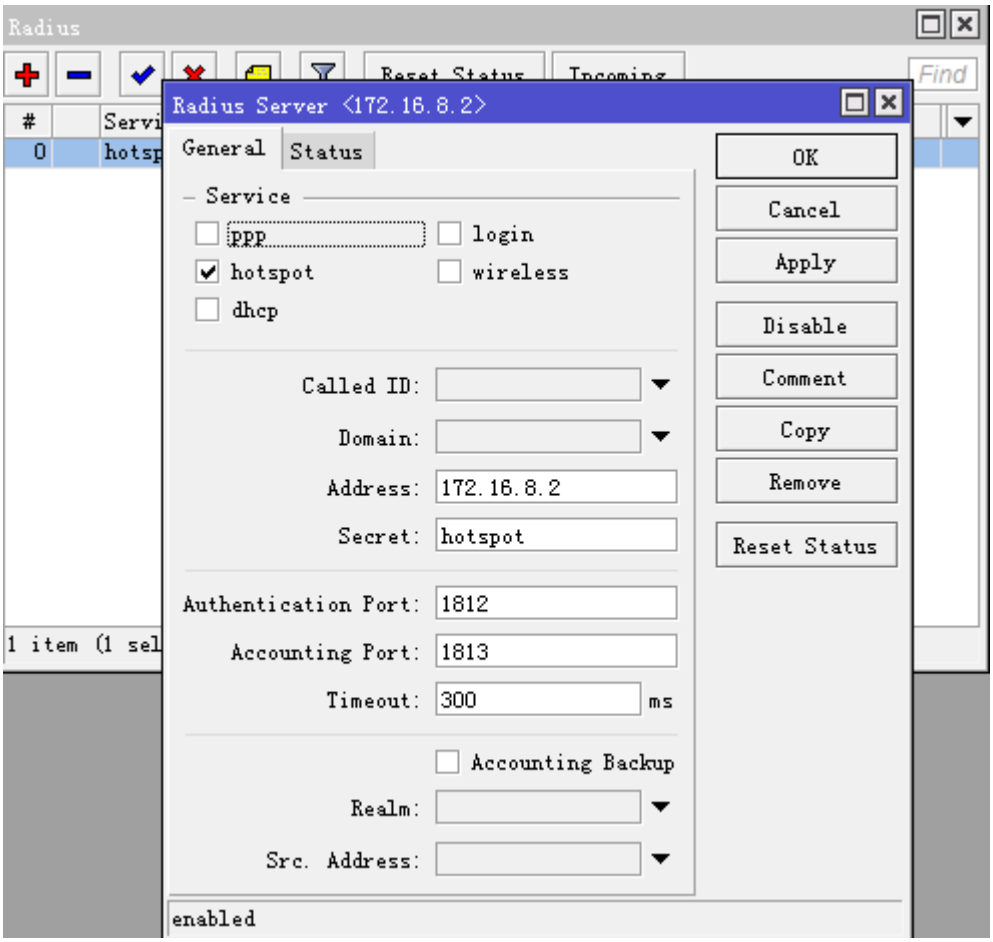


38.4 Hotspot 认证 User Manager 连接

这里我们补充下 Hotspot 认证，我们需要进入 hotspot 的 Server Profiles 设置 RADIUS 连接，启用 RADIUS 参数



RouterOS 的 RADIUS 配置参数，我们选择 hotspot，并设置 Secret 为 hotspot



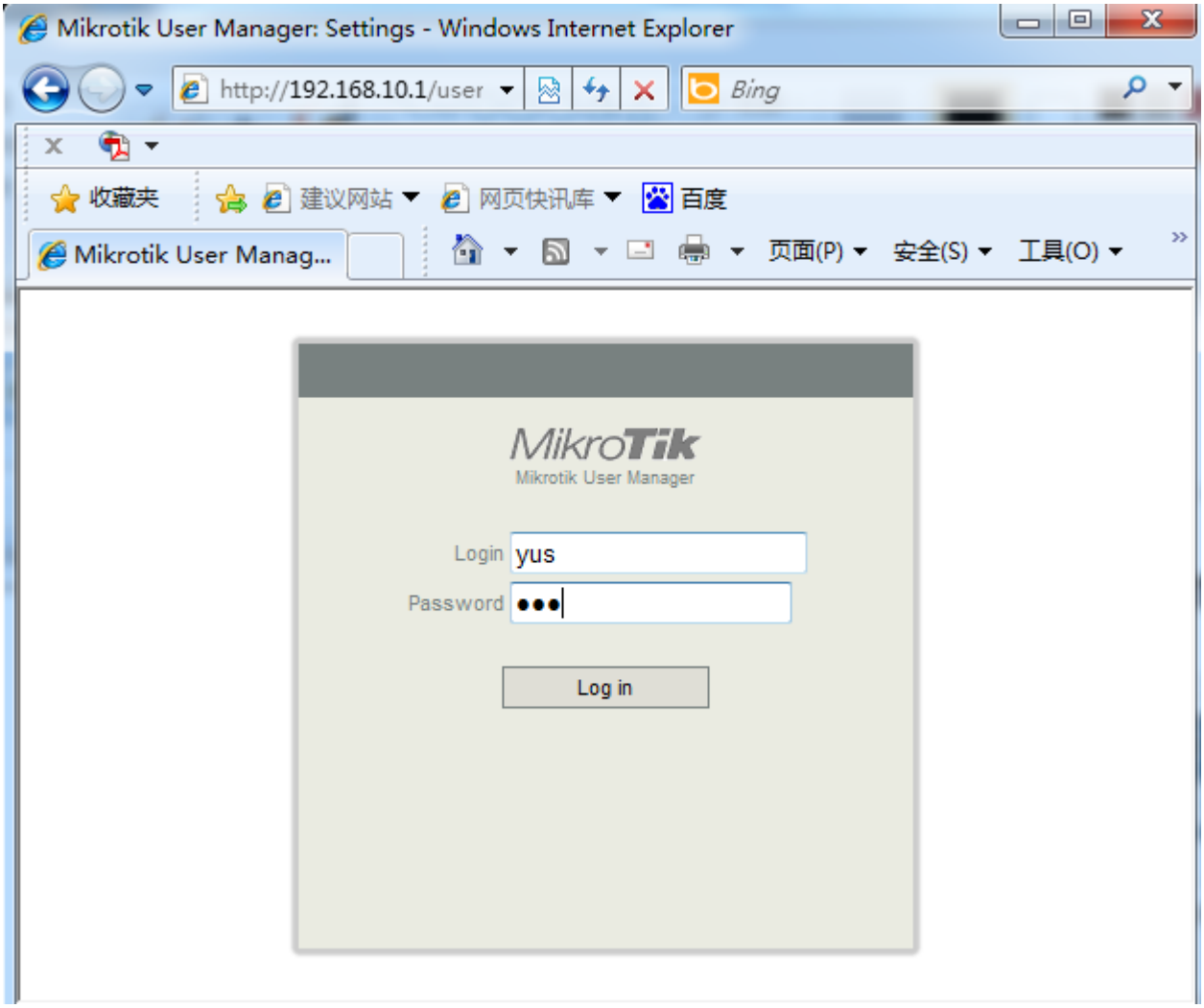
其他的 User Manager 参数相同，仅需要修改 Secret 参数。

38.5 用户自助服务

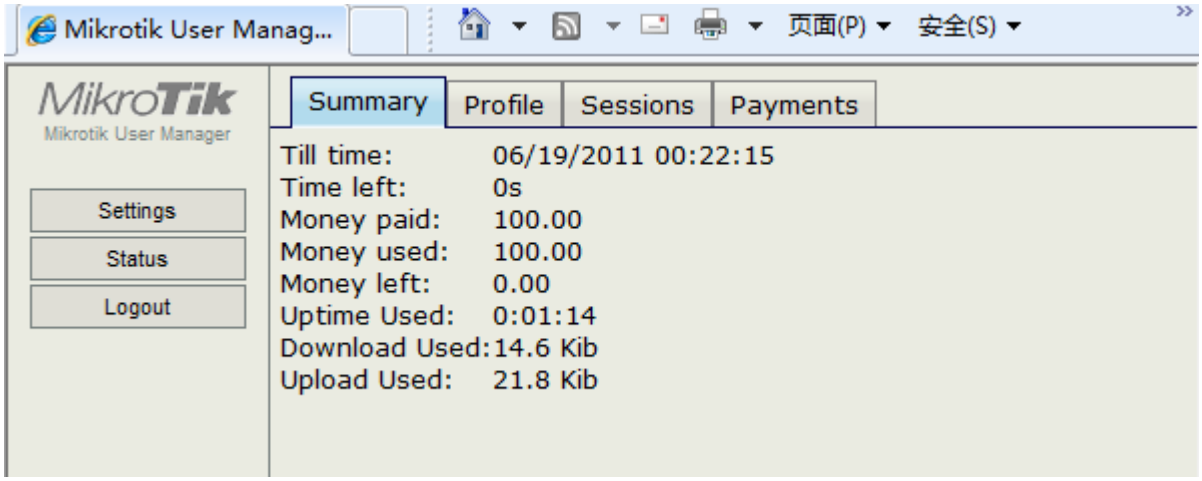
User Manager 支持用户通过网页访问，并让用户登陆到自己的账号进行自助查看和设置，用户通过 http://router_IP/user 页面下设置，你可以为用户自助访问页面做一个静态的域名，方便用户能直接访问。

例如我们的 User Manager 的 IP 地址是 192.168.10.1，我们在浏览器里输入 http://192.168.10.1/user 进入登录页面

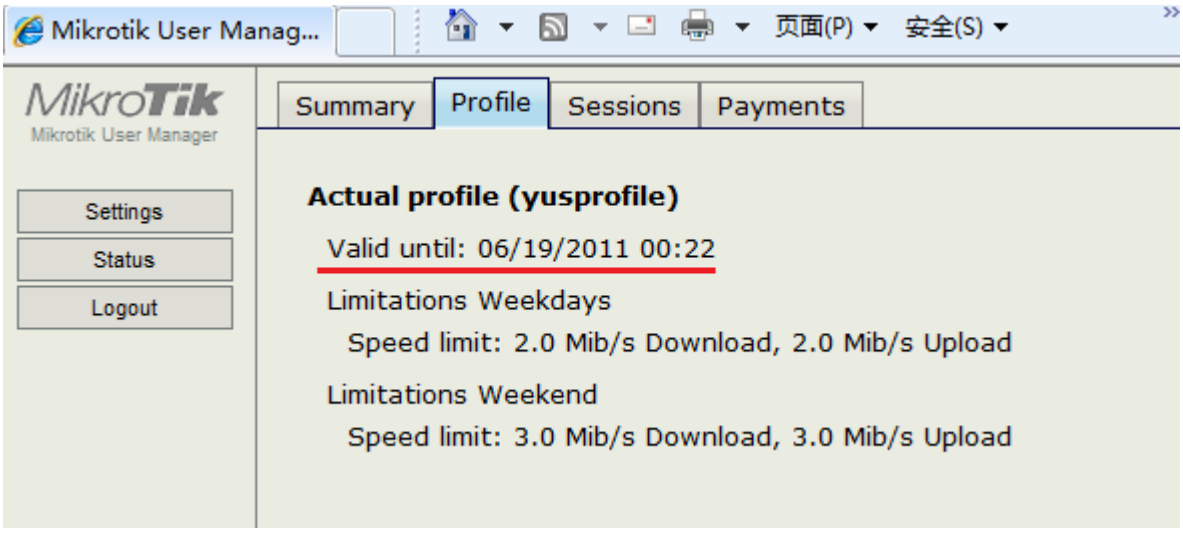
在 user 页面，我们使用的是用户自己的账号，如账号 yus，密码是 yus



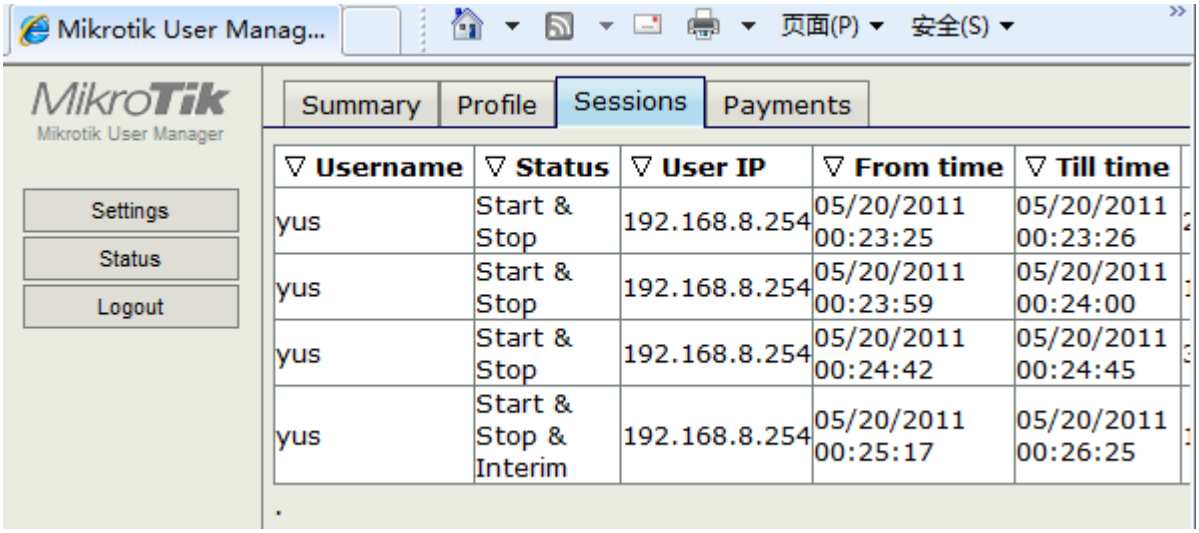
我们登录后，首先看到 Summary 页面，即用户基本使用情况



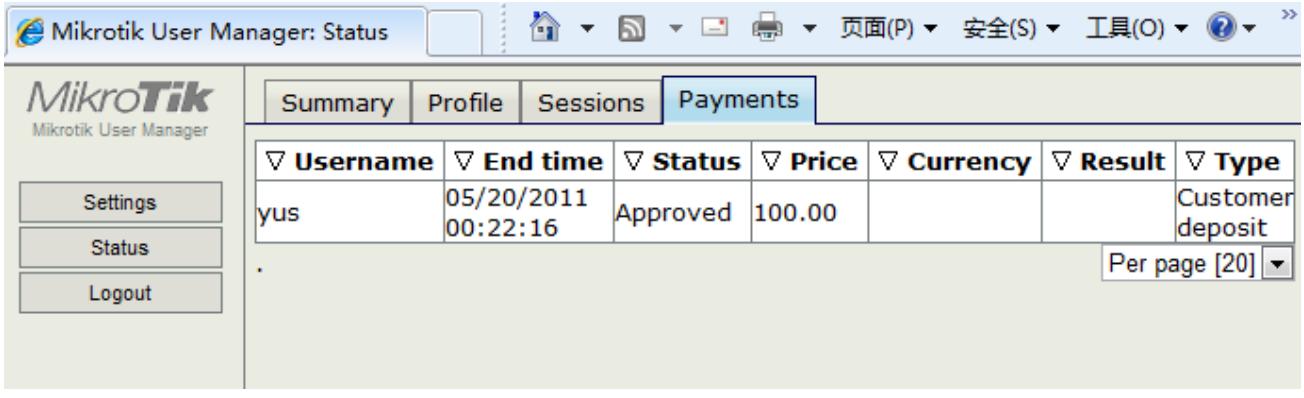
点击 Profile 标签，可以查看策略组的情况，如 Valid until 有效期和流量策略情况



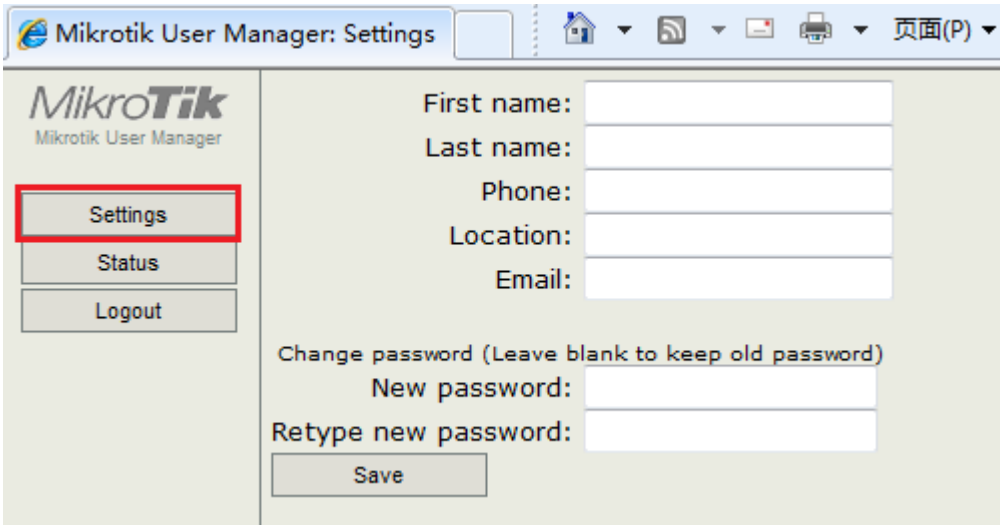
在 Sessions 里用户可以查看自己登录情况，如状态、使用的 IP 地址，登录时间和结束时间



在 Payments 里用户可以查看付款情况



在 settings 里，用户可以修改自己的私人信息，并可以为自己修改密码：



38.6 User Manager 数据导出与导入

User Manager 数据备份，不能通过 web 页面完成，需要进入命令行，通过命令汇出和导入，操作如下：

导出备份数据，并取名为 userman

```
[admin@MikroTik] /tool user-manager> database save name=userman
```

导出后，数据保存在 RouterOS 的 file 目录下可以找到，并看到后缀为.umb，类型是 userman backup

```
[admin@MikroTik] /tool user-manager> /file print
 # NAME                  TYPE                  SIZE CREATION-TIME
 0 skins                 directory                  jan/01/1970 08:00:05
1 web-proxy1             directory                  nov/19/2016 14:54:59
 2 web-proxy1/logsqldb   file                6.0KiB nov/19/2016 14:55:05
 3 log.0.txt             .txt file          18.7KiB feb/09/2017 09:00:01
4 userman.umb            userman backup     14.5KiB feb/09/2017 09:04:52
```

将 userman.umb 导入：

```
[admin@MikroTik] /tool user-manager> database load name=userman.umb
```

第三十九章 Scheduler（计划任务）

Scheduler 计划任务，通过设置定时或周期安排执行相应的脚本操作，计划任务能有效完成预期的任务，说的高级点就是基于 Scheduler 和 Script 两个功能，实现自己编写自己的脚本，实现智能路由器。关于脚本编写可以参考《RouterOS 脚本编写与 WLAN 无线教程》

规格

功能包需求: ***system***
等级需求: *Level1*
操作路径: ***/system scheduler***

39.1 计划任务介绍

计划任务列表通过调用脚本，并触发脚本执行，在指定的时间或者是在指定的时间周期执行任务，在计划任务调用脚本，可以设置脚本名称或执行写入脚本。

属性描述

interval (*时间*; 默认: **0s**) - 脚本执行的间隔周期时间，脚本将在指定的时间周期内反复执行。
name (*名称*) – 任务名称
on-event (*名称*) – 脚本执行名。通过调用**/system script** 里的脚本规则名称，也可以直接写入脚本。
run-count (*只读: 整型*) – 监视脚本使用数，这个计数器记录当每个脚本执行一次，计数器便增加 1
start-date (*日期*) – 开始脚本执行的日期
start-time (*时间*) – 开始脚本执行的时间
startup – 默认在系统启动 3 秒后执行脚本. 计划任务选项里对 start-time 设置了 startup，则在系统启动完成后 3 秒运行。

Run-count 记录了计划任务的执行次数，当路由器重启后，计数器会重置，如果有复杂的脚本执行模式，通常可能会涉及到计划多个脚本，在多个计划任务中切换，执行一个时禁用另外一个。

注意:如果计划任务项目的启动时间设置为 startup，它执行是在控制台启动后的 3 秒时间。这意味着所有具有 startup 和执行时间间隔为 0 的脚本，将在每次路由器启动完成时执行一次。如果间隔被设置为 0 以外的值，计划任务将不会在启动时运行。

39.2 计划任务事例

通过下面的简单事例，介绍下计划任务在 RouterOS 是如何操作执行的。

事例 1：我们添加一个任务执行系统日志记录测试，并间隔 1 小时执行一次，在 log 日志中显示“log test”，on-event 我可以直接写入 RouterOS 脚本

```
[admin@MikroTik] system script> add name=logtest on-event=:log info “test”
start-time=startup interval=1h
[admin@MikroTik] /system scheduler> print detail
```

```
Flags: X - disabled
 0    name="logtest" start-time=startup interval=1h on-event=:log info "log test"
owner="admin" policy=ftp,reboot,read,write,policy,test,winbox,password,sniff,sensitive,
        api run-count=5 next-run=09:05:19
```

在 winbox 中，将脚本直接配置到 Schedule 的 On-event 下：

事例 2：添加 2 个脚本改变带宽设置队列规则"ABC"，每天上午 9 点限制为 10Mb/s，下午 5 点限制为 20Mb/s。这个队列的规则。

先在 queue simple 添加一条对内网 ip 段 192.168.0.0/24 的流控规则(6.0 版本前配置流控目标 IP 为 target-addresses，6.0 开始为 target)：

```
[admin@MikroTik] /queue simple> add name=ABC target=192.168.0.0/24 max-limit=10M/10M
[admin@MikroTik] /queue simple> print value-list
              name: queue1
              target: 192.168.0.0/24
              parent: none
      packet-marks:
              priority: 8/8
               queue: default-small/default-small
              limit-at: 0/0
            max-limit: 10M/10M
```

```
          burst-limit: 0/0
    burst-threshold: 0/0
          burst-time: 0s/0s
        bucket-size: 0.1/0.1
```

然后配置脚本，并添加计划任务(**注：**在 2.9 之前定义脚本查找字符串是不需要加双引号的，但在 3.0 后中需要注明字符串，需加上双引号，如查找流控规则名”ABC”)，进入 script 脚本编辑器，添加两台脚本规则 start_limit 和 stop_limit

```
[admin@MikroTik] queue simple> /system script
[admin@MikroTik] system script> add name=start_limit source={/queue simple set [find
name=”ABC”] max-limit=10M/10M }
[admin@MikroTik] system script> add name=stop_limit source={/queue simple set [find
name=”ABC”] max-limit=20M/20M }
[admin@MikroTik] system script> print
  0 name="start_limit" source="/queue simple set [find name=”ABC”] max-limit=10M/10M "
    owner=admin run-count=0

  1 name="stop_limit" source="/queue simple set [find name=”ABC”] max-limit=20M/20M "
    owner=admin run-count=0
```

进入计划任务下使用 on-event 调用脚本编辑器中的两条脚本，可以直接在 on-event 中填写脚本名称

```
[admin@MikroTik] system script> .. scheduler
[admin@MikroTik] system scheduler> add interval=24h name="set-10M" \
\... start-time=9:00:00 on-event=start_limit
[admin@MikroTik] system scheduler> add interval=24h name="set-20M" \
\... start-time=17:00:00 on-event=stop_limit
[admin@MikroTik] system scheduler> print
Flags: X - disabled
  #    NAME       ON-EVENT   START-DATE   START-TIME INTERVAL            RUN-COUNT
  0    set-10M    start... oct/30/2008 09:00:00    1d                  0
  1    set-20M   stop_... oct/30/2008 17:00:00    1d                  0
[admin@MikroTik] system scheduler>
```

事例 3：下面是通过电子邮件发送每周备份路由器配置的脚本：

```
[admin@MikroTik] system script> add name=e-backup source={/system backup
save name=email; /tool e-mail send to="root@host.com" subject=([/system
{... identity get name] . " Backup") file=email.backup}
[admin@MikroTik] system script> print
  0 name="e-backup" source="/system backup save name=ema... owner=admin
    run-count=0

[admin@MikroTik] system script> .. scheduler
[admin@MikroTik] system scheduler> add interval=7d name="email-backup" \
\... on-event=e-backup
[admin@MikroTik] system scheduler> print
```

```
Flags: X - disabled
  #    NAME       ON-EVENT START-DATE   START-TIME INTERVAL             RUN-COUNT
  0    email-... e-backup oct/30/2008 15:19:28    7d                    1
[admin@MikroTik] system scheduler>
```

用 RouterOS 电子邮件发送，需要对你接收邮箱设置，即开启接收邮箱的 SMTP 服务，操作路径**/tool e-mail**
例如（**注：**建议是自己的 SMTP 服务器，一些正规网站的邮件服务器可能会将发送信息屏蔽）：

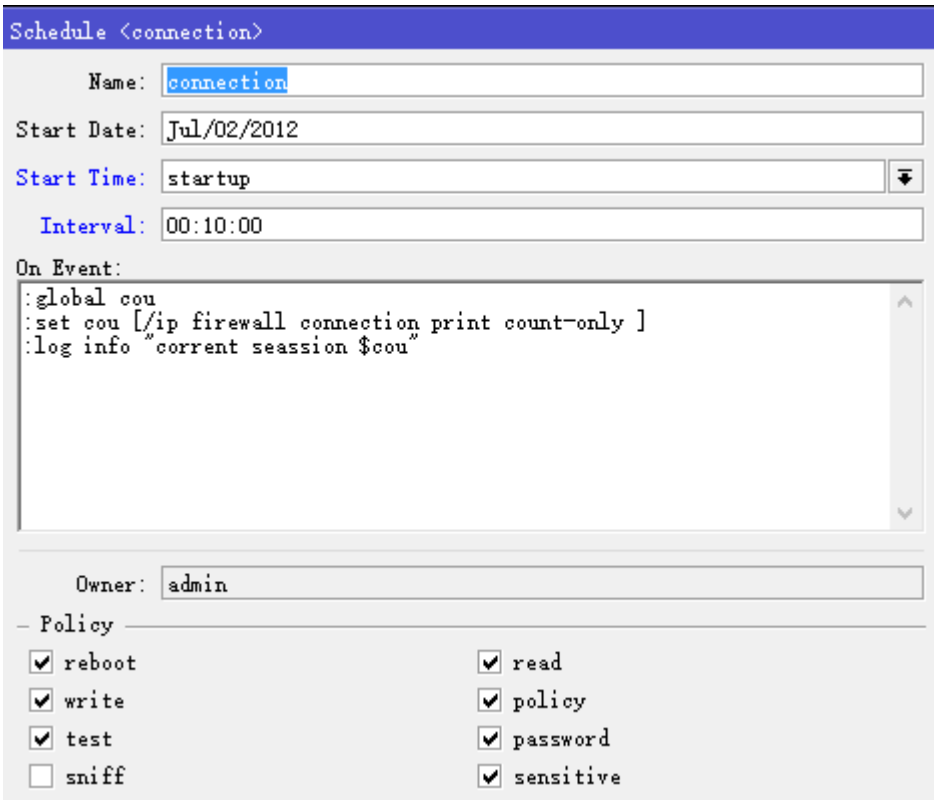
```
[admin@MikroTik] tool e-mail> set server=159.148.147.198 from=SysAdmin@host.com
[admin@MikroTik] tool e-mail> print
    server: 159.148.147.198
      from: SysAdmin@host.com
[admin@MikroTik] tool e-mail>
```

事例 4：下面的例子每 10 分钟统计一次 firewall connection 的会话数，并记录在 log info 里:

首先我们编写脚本，获取会话数量，并通过:log info 命令记录当前会话数

```
:global cou
:set cou [/ip firewall connection print count-only ]
:log info "corrent seassion $cou"
```

进入 schedule 新建一个计划任务取名 connection，设置 interval 为 10 分钟，On-event 加入脚本



以上的计划任务都涉及了 RouterOS 的 Script 脚本编辑，所以 RouterOS 要完成指定的计划任务，会编写脚本是必须的。

Changelog

6.7.1e 修改

9.3 章节，增加使用 address-list 创建黑白名单事例
9.1 章节，更正 simple queue 的错误的内容，并更新内容

6.7e 修改

7.1/7.2 章节，更新 DHCP-client/server 配置内容
7.5 章节，新增 DHCP server 的 vendor classes 厂商 ID 分类
7.6 章节，新增 rate limit 流量控制
4.6 章节，更新 GPS 部分内容
9.1/9.2 章节，更新 firewall filter 内容
2.18 章节，新增 neighbor discovery 邻居发现协议
11.1 章节，更新 fastpath 内容
12.10 章节，更新 PCQ 动态流控实例
8.8 章节，新增 RouterOS 搭建私有 DDNS 服务
8.9 章节，新增 DNS cache 捕获域名 IP 地址
12.9 章节，新增 Queue tree HTB 配置实例
2.2 章节，更新 RouterOS 备份与复位管理

6.6e 修改

17.5 章节，调整到 6.14 章节，更名为单网卡多次拨号 PPPoE 配置
3.3 章节，更新 Thoughput 吞吐量内容
12.5 章节，删除 simple queue 在 v6 改进中遗留的错误说明
3.10 章节，删除关于 interface 下的 master port 的配置内容（已经取消功能，避免配置理解错误）
3.10 章节，更新 switch 配置介绍，新增 port、port isolation 和 cpu flow control 内容
6.13 章节，新增 PCC 在 IDC 应用
32.5 章节，新增在 VPS 上安装 CHR
6.15 章节，新增策略路由导致外网 nat 端口映射内网访问失效
26.1 章节，新增 IPSec 基本术语
26.2/26.3/26.4 章节，更新相应的 IPsec 实例

6.5e 修改

更新基础知识内容（附七层模型视频链接）
37 章节，新增 SOCKS
1.3 章节，附 Winbox 视频介绍
1.7 章节，附 hAP ac2 单线接入上网配置视频
6.13 章节，更新 PCC 原理内容
3.6 章节，新增 reset 和 Mode 按钮（附视频链接）
6.1 章节，新增 RIB 和 FIB
5.3 章节，新增 local arp proxy 介绍

21.7 章节，更新并重新命名高并发 PPPoE 认证服务集群
14.3 章节，新增 Bridge VLAN（access/trunk）内容
2.8 章节，更新 RouterOS 升级和降级内容
14 章节：调整小节顺序
14 章节：附交换机原理与 RouterOS Bridge 二层交换机配置视频
14.8 章节：更新 L3-hardware-offloading
32.5 章节：更新 KVM 介绍和配置

6.4e 修改

3.6 章节，更新固件升级内容
9.2 章节，新增过滤指定域名
9.5 章节，新增两个 IP 网络单向访问控制
12.10 章节，更新 PCQ 介绍
33.6 章节，新增 DHCPv6 server
33.9 章节，新增 IPv6 防火墙
33.10 章节，新增 IPv6 应用实例
33.4 章节，更新 IPv6 地址分配方式
2.11 章节，修改标题为管理服务端口（ip service），并更新内容
2.13 章节，更新日志内容
2.17 章节，新增 File 目录介绍
10.2 章节，更新 srcnat 介绍不清晰内容
10.3 章节，新增正确配置 nat 规则接口参数，并修正非对称端口映射的错误配置
1.1 章节，netinstall 内容更新
1.2 章节，更新 winbox 和 webfig 介绍
第三十四章节，调整到第三章节，总章节为三十八章
第三十二章节，更名为 CHR（Cloud Host Router），将 KVM、Xen 和 Metarouter 调整在该章的 32.5 章节其他虚拟化介绍
第十五章节，更名为 interface vlan（三层）
15.4 章节，部分 QinQ 内容调整到 14.13 章节
6.1&6.2 章节，更新 RIB 和 FIB 内容

6.3.14e 修改

11.3 章节，更新 fasttrack
3.13 章节，更新 cloud 内容
8.5 章节，新增 DoH 内容
8.7 章节，新增 DNS forward 转发内容
14.7 章节，v7 L3 hardware-offloading 说明
6.12 章节，更新返程路由内容
27 章节，更新 EoIP 内容
15.4 章节，更新 QinQ 内容
14 章节，调整 bridge vlan 内容

6.3.13e 修改

21 章节，更新 hotspot 部分内容
21.13 章节，增加 https 认证登录方式
23.3 章节，修正 PPTP 应用事例内容错误
34.6 章节，增加 IPv6 无状态地址配置事例
16.5 章节，增加 RouterOS 与 CentOS7 配置 bond
17 章，更新 VRRP 内容
38.6 章节，更新 fetch 内容
14.11 章节，增加 fast forward 介绍
14.12 章节，新增（R）STP 介绍
4.3 章节，更新部分内容
3.10 章节，合并到 3.9 章节
13 章节和 18 章节整合，将 Nth 内容和 mangle 整合为 13 章节，一共 39 章

6.3.12e 修改

4.4 章节，补充 interface list 介绍
37.11 章节，增加 speed test 介绍
6.11 章节，更新和修改 PCC 内容
1.3 章节，更新调整 winbox 介绍
4.6 章节，增加 GPS 章节介绍
37. 章节，更新 netwatch 介绍
10.3 章节，删除错误 nat 映射实例，动态 ip 需指定 dst-address，不能只设置 in-intarface。
26.3 章节，增加基于 SSTP 隧道的远程端口映射
34 章节，增加 IPv6 章节

6.3.11e 修改

12.6 章节，RouterOS v6 queue 更新
3.14 章节，更新 cloud 内容
3.12 章节，新增 CRS bonding 和端口隔离介绍
第三章节，更名为 MikroTik RB/CCR/CRS 介绍
1.1 章节，调整和更新 RouterOS 安装介绍
1.1 章节，新增自定义 RouterOS 默认启动配置脚本
37.6 章节，新增两个 feth 事例
6.10 章节，补充一处 nat 配置遗漏
二十三章节，更新 PPTP 介绍，并修改部分错误
14.9 章节，新增不同 MikroTik 硬件的二层 VLAN 配置
2.2 章节，新增云备份功能介绍

6.3.9e 修改

1.5 章节，添加基本操作命令内容
2.1 章节，更新 RouterOS 账号管理
2.5 章节，更新 Resource 内容
3.5 章节，增加 RB 复位键功能说明
6.1 章节，增加路由距离介绍
9.2 章节，新增 filter 基本配置事例
9.2 章节，新增规则条件执行

14.4 章节更新，创建二层透明流控过滤
将 15 章节，VLAN 中 PPPoE 服务调整到 22.8 章节
37.10 章节，增加 Traffic Generator 介绍

6.3.8e 修改:

14.1 章节 更新 bridge settings 介绍
增加 14.7 章节 bridge Hardwae offloading（6.41 变动）
增加 14.8 章节 Bridge VLAN 过滤（6.41 变动）
增加 14.9 章节 IGMP Snooping（6.41 变动）
4.6 章节，增加 LTE Passthrough（透传）
10.1 章节，更新 nat 内容和属性介绍
9.1 章节，增加 firewall filter 属性介绍
8 章节，更新 DNS 章节内容，添加 DNS 的正则表达介绍
14.3 章节，删除关于 auto isolate 隔离错误理解

6.3.7e 修改:

6.5 章节，修改并更新寄偶源地址策略路由标记
12.4 章节，增加两个 simple queue 注意事项
3.8 章节，更新 RouterBOARD switch 介绍
12.2 章节，增加 interface queue 内容
12.3 章节，更新 queue type 介绍内容
11.3 章节，更新 fasttrack 内容
1.1 章节,删除 netinstall tile 版本的内容(在 ccr 系列刚推出时 RouterOS v6.0rc 版本的 netinstall 有单独做 tile 平台，现在已经不再区分）
15 章节，更新 vlan 介绍
6 章，调整相关路由章节顺序，更新 RouterOS 策略路由小节内容
6.6 章节，增加 RouterOS 源地址策略路由

6.3.6e 修改

6.12 章节，IP 地址错误修正
6.11 章节，多线路端口映射内容修正
33.5 章节，更新 CHR 介绍
4.5 章节，增加 LTE 客户端介绍
4.6 章节，增加 PoE-out 介绍
10.3 章节，调整非对称端口映射内容

6.3.5e 修改

16 章节，更新 bonding 模式介绍和 link-monitoring 介绍
16.3 章节，修改 bonding 的 balance-rr 华为交换机配置错误
38.6 章节，增加 user mananger 数据导出与导入

9.2 章节，增加 ICMP limit 属性在 v6.34 变动后的配置
22.3 章节，增加 PPPoE scanner 介绍
22.4 章节，更新 PPPoE server 章节
6.12 章节，增加 PPPoE 走下行，商务专线走上行
14.6 章节，增加 Bridge nat 修改 VRRP 接口 MAC
11.2 章节，更新关于 CCR 系列 fastpath 说明

6.3.4e 修改

12.8 章节，增加 queue tree v6 实例
2.8 章节，调整和更新 RouterOS 升级操作
39 章节，修改和调整计划任务内容
36 章节，新增 UPNP 内容，将其他章节后移，当前总共 39 章节
3G 接口扩展设置调整到 4.4 章节
6.9 章节，修改 PPTP 借线内容，添加静态和策略路由实例
9.7 章节，增加 QQ L7 代码控制介绍
3.3 章节，补充吞吐量性能介绍
32 章节，更新 proxy 章节内容
12.9 章节，修正 PCQ 一个实例介绍错误
PDF 版本解除复制锁定，可以对文本进行复制

6.3.3e 修改

3.9 章节，增加 RB switch 配置介绍
3.10 章节，增加 RB 802.1Q trunk 配置
5.4 章节，增加 arp-timeout 介绍
9.3 章节，增加 address-list 添加域名规则
9.4 章节，增加 interface list 配置介绍
10.2 章节，增加 pcc src-nat 负载均衡
10.8 章节，增加 RAW 介绍
10.6 章节，新增一个参考实例

6.3.2e 修改

15.4 章节，修改 CCR1072 表内的配置错误
2.13 章节，调整 log 日志管理内容
36.10 章节，增加 profiler 工具介绍
1.8 章节，增加基本网络故障排查命令与方法
2.8 章节，增加在线升级介绍
增加 33.5 章节，CHR 介绍
6.10 章节，调整 PCC 负载均衡内容
修正了大量错字和错句，感谢网友认真阅读，通过邮件指明大量错字，错句。

6.3.1e 修改

2.9 章节，补充说明重启时间保存操作
16.2 章节，修改设备 mode 笔误
10.2 章节，新增 same nat 规则说明
9.2 章节，调整和补充自定义链表介绍
9.3 章节，增加 address-list 介绍
9.4 章节，增加内网多 IP 段访问控制
15.4 章节，更新基于 VLAN 的 PPPoE 认证内容
12 章节，对 simple queue v6 的错误解释做调整
12.5 章节，调整 RouterOS v6.0 queue 变动的内容

6.3e 修改

修正 36 章后下一章为 38 章，缺 37 章，当前总章节为 38 章
37 章节, 修正和补充 user manager 内容
6.7 章节, 修正 80 端口策略路由最后注意说明表述不清楚
11.3 章节，增加 fasttrack 内容
20.2 章节，删除 RADIUS 合作模式错误实例
24.1 章节，更新 L2TP 配置内容
2.2 章节，增加 6.13 备份加密介绍和复位命令介绍
3.7 章节，增加 routerboard setting 介绍
6.11 章节，增加多线路返程路由介绍
17.3 章节，增加说明 vrrp 多 pppoe 拨号弊端

6.2.4e 修改

1.3 章节 增加 winbox v3 介绍
3.12 章节 增加 6.27 cloud 服务设置变动
5.3 章节 补充和修改 APR 内容
25 章节 补充 OVPN 介绍和证书创建
27.1 章节 ipsec 配置修正

6.2.3e 修改

12.9 章节 修改 PCQ 实例的上下行写反错误，并调整该章节的内容
20.1 章节 RADIUS 属性未翻译修正

6.2.2e 修改

12.8 章节 简单的 Queue Tree 实例 limit-at 脚本错误修正
3.13 章节 flashfig 文字错误
9.2 章节 文本过滤域名错误
6.6 章节 线路判断内容调整

9.4 章节，qq 限制图文不符
DoS 防御丢失 9.8
加入 2.15 SNMP 章节

6.2.1e 修改：

12.4 章节 修改笔误为：
注意：v6.0 后 FIFO 算法被取消，因此对于无 FIFO 算法的 v6 版本，move 调整队列规则优先级已无任何意义
12.8 章节 limit-at 脚本没有写数值
7.4 章节 补充 DHCP-relay 丢失部分内容
7.2 章节 增加 Lease 租约

参考文献：

http://wiki.mikrotik.com

https://help.mikrotik.com

http://www.routerboard.com

https://ros.tw/wp/

相关网络资料