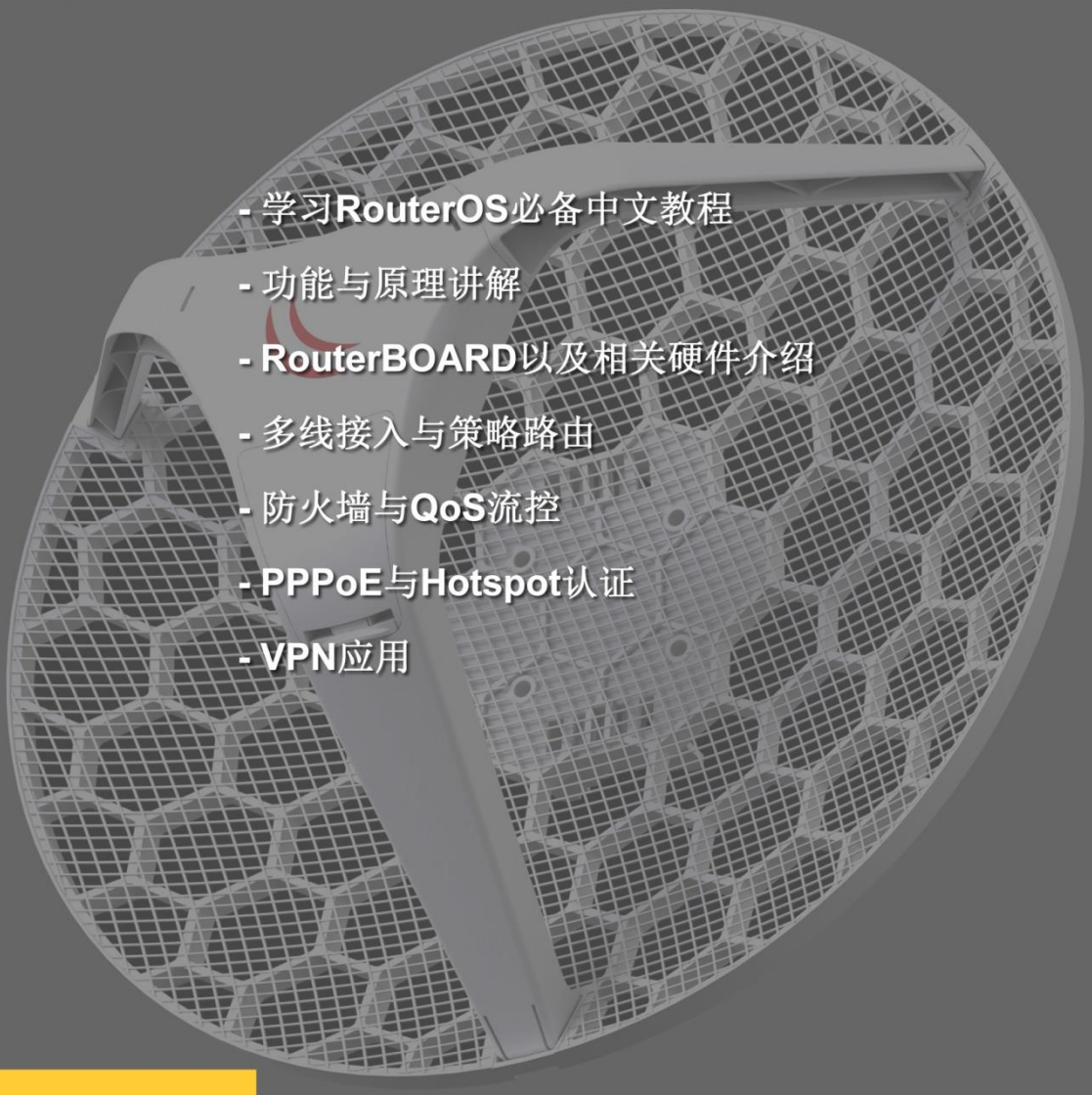
MikroTik

RouterOS入门到精通 v6.3.3e



- 学习RouterOS必备中文教程
- 功能与原理讲解
- RouterBOARD以及相关硬件介绍
- 多线接入与策略路由
- 防火墙与QoS流控
- PPPoE与Hotspot认证
- VPN应用

编：余松

www.irouteros.com

前 言

Mikrotik 公司 1996 年成立于拉脱维亚，RouterOS 是 1997 年研发的专业路由系统，经历了多年的发展，已经发展到 6.0 版本。最初 MikroTik 开发 RouterOS 目的是解决无线局域网传输问题（WLAN），后来通过不断扩展功能实现了多种功能集于一身路由操作系统。在国内早期最大的使用人群是网吧、小区宽带和企业网络管理，这个和国外的情况有点差别，在国外 RouterOS 不仅用于解决路由管理，大多应用在 WLAN 的覆盖和传输，RouterOS 在基于 802.11abgn/ac 协议上的高带宽传输有自己的明显优势，特别是他独有的 Nstrem 和 Nv2 协议，近段时间国内的 WLAN 市场发展非常迅猛，对基于 RouterOS 开发的 RouterBOARD 产品需求也在不断增长。其实当前的 RouterOS 已经具备 1Gbps 网络环境的完整解决方案。通过脚本可以让 RouterOS 完成二次功能开发和应用，在设计上具备一定的开放性。

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资料整理后编辑而成，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，在此基础上还有另外一套教程：《RouterOS 无线教程》和 MikroTik 配套的监控软件《The Dude 中文实用教程》。

从 RouterOS 6.0 版本开始，MikroTik 对 RouterOS 做大的改动，从内核优化、Queue 大改动、新的 Tilte 硬件构架等，核心改动较之前变化很大，性能提升明显。但对于 x86 平台的用户感觉没有 RouterBOARD 那么明显，因为后期的 RouterBOARD 加入了 FastPath 功能，能通过硬件快速转发数据包，这个是 x86 平台无法做到的，但从总体上内核的优化和 Queue 性能的改进也能带来不小的提升。

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，希望初步涉及网络的朋友能了解下，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内容，初级的网络工程师如 CCNA 或 HCNA 等，这些资料可以去看看和学习。

版本: **v6.3.3 e-book**
适用: **RouterOS v5.x，6.x**
作者: 余松
Web: **www.irouteros.com**
E-mail: **athlon_sds@163.com**
Blog **http://blog.163.com/athlon_sds**
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**
应用说明 **13**
基础知识 **18**
OSI 七层参考模型 18
二层 数据链路层 18
三层 网络层 19
四层 传输层 22
RouterOS 基本 FAQ 22
第一章 RouterOS 基本操作 **24**
1.1 RouterOS 安装介绍 24
CD 光盘安装 24
USB 安装 29
NetInstall 安装 31
1.2 RouterOS 登录方式 44
方式 1 Console 连接 45
方式 2 Winbox 46
方式 3 显示器+键盘 47
方法 4 MAC 层访问 (Telnet 与 Winbox) 48
MAC WinBox Server 48
MAC 登陆列表 49
MAC telnet 访问客户端 49
1.3 Winbox 操作 50
Winbox v3 版本 53
工作区域和子窗口 56
子窗口菜单栏 56
快速查询 57
过滤筛选 58
自定义显示列表 59
详细模式 60
拖&放 60
流量监测 61
项目复制 62
Winbox 目录转移设置 65
1.4 Webfig 操作 65
连接到 RouterOS 66
操作界面介绍 68
列表项目配置 69
Files 文件操作 72
1.5 CLI（command Line interface）命令行操作 73
命令帮助 73
指令执行概述 76
Tab 快速输入 77
基本操作命令 77
快速设置 Setup 79
1.6 安全模式 80

1.7 RouterOS 简单网络配置事例 82
第一步：网络接口配置 82
第二步：添加 IP 地址 83
第三步：添加默认网关 84
第四步，NAT 地址转换 85
第五步，DNS 配置 86
端口映射 87
主机带宽控制 88
ADSL/PPPoE 拨号配置 89
1.8 基本网络故障排查命令与方法 90
第二章 System 系统管理 95
2.1 RouterOS 账号管理 95
账号分组权限 95
新建账号 97
2.2 RouterOS 备份与复位管理 97
系统备份与恢复 98
导出指令（Export） 99
导入指令（Improt） 100
系统复位 101
2.3 系统重启与关机 102
2.4 RouterOS 主机名 102
2.5 系统资源管理 103
IRQ 配置管理 105
USB 端口信息 107
PCI 信息 107
RouterOS x86 平台多 CPU 设置 108
2.6 Watchdog 监测 109
2.7 RouterOS 功能包（Packages） 110
查看功能包 112
功能包操作事例 112
2.8 升级和降级 RouterOS 113
RouterOS 升级包区别 113
RouterOS 升级操作 114
在线升级 116
降级选项 117
2.9 SNTP client 117
2.10 telnet 和 ssh 客户端 119
telnet 客户端 119
SSH 客户端 119
2.11 RouterOS 常用协议与端口 120
修改 service 端口 120
常用服务列表 121
2.12 Note 笔记 122
2.13 Log 日志管理 123
Log（日志） 124
Logging 日志管理 124
Log 管理执行方式 126
使用 The Dude 管理器记录系统日志 127

2.14 Supout.rif 技术支持文件 130
2.15 SNMP 131
启用 SNMP 132
Community 团体名 132
SNMP 配置实例 133
SNMP write 134
第三章 MikroTik RouterBOARD 介绍 135
3.1 RouterBOARD 的发展 135
3.2 RouterBOARD 产品命名与标识 138
RouterBOARD 基本命名规则 138
集成无线网卡命名 139
无线网卡接口类型命名 139
外壳类型命名 139
Cloud Core Router 命名 140
Cloud Router Switch 命名 140
3.3 RouterBOARD Throughput（吞吐量） 140
3.4 RouterBOARD 的型号与分析 143
3.5 RouterBOARD 复位 144
3.6 升级 RouterBOARD 固件 147
3.7 RouterBOARD Settings 148
3.8 RouterBOARD Switch 介绍 150
3.9 RouterBOARD Switch 配置 151
基本原理 152
Host Table(主机列表) 154
Vlan Table(VLAN 列表) 155
Rule Table（策略列表） 156
交换配置实例 156
3.10 基于 Atheros 交换芯片的 802.1Q VLAN 配置 158
配置 VLAN 管理 IP 160
3.11 RouterBOARD 端口镜像 161
端口镜像配置 161
Rule 指定镜像内容 162
3.12 CRS 二层交换介绍 165
术语和解释 165
端口交换（Port Switching） 166
CRS 交换配置 169
3.13 RouterBOARD LCD 触摸屏 172
LCD 触摸屏校准 173
LCD 截屏 173
LCD PIN 密码 174
LCD 流量图 174
3.14 Cloud 175
Cloud 配置： 176
Cloud 在 v6.27 调整 177
3.15 Flashfig 177
Flashfig 配置实例 178
第四章 接口配置（Interface） 184
4.1 Interface 基本操作 184

4.2 流量监视 185
4.3 以太网接口（Ethernet） 186
以太网接口配置 186
接口状态监测命令 188
线路故障探测 188
第五章 IP 配置与 ARP 190
5.1 IP 地址 190
5.2 ARP 地址解析协议 191
5.3 ARP 模式 191
Enabled 192
Disabled 192
Reply-only 192
Proxy-arp 193
5.4 ARP 绑定 196
ARP 双向绑定事例 197
ARP Timeout 设定 197
第六章 路由设置（Route） 199
6.1 RouterOS 路由介绍 199
RIB 路由信息库（路由表） 199
默认路由（Default route） 200
直连路由（Connected routes） 200
等价多路径路由(ECMP) 201
网络接口作为路由网关 201
6.2 RouterOS 路由分类 201
静态路由 202
ECMP 等价多路径路由 203
策略路由 204
6.3 ISP 目标地址路由 204
6.4 网关断线处理 206
6.5 源地址策略路由奇偶标记 207
6.6 光纤和 ADSL 静态路由 211
6.7 HTTP 端口的策略路由 213
6.8 PPTP 借线路由操作 215
配置 PPTP-Server 216
配置 PPTP-Client 218
路由配置 219
6.9 RouterOS 策略路由 219
6.10 RouterOS 负载均衡 222
PCC 负载均衡 222
六线的 PCC 负载均衡 231
PCC 实现比例权重路由 232
6.11 多线接入的返程路由 234
返程路由原理 234
返程路由的多线路端口映射 236
第七章 DHCP 操作配置 239
7.1 DHCP-Client 设置 239
7.2 DHCP-Server 设置 240
DHCP Lease 租约 243

7.3 DHCP Options 245
Classless static Route 245
Option-set 246
7.3 Alerts 警报 247
7.4 DHCP-Relay 配置 247
第八章 DNS 251
8.1 DNS 配置 251
8.2 内部 DNS 域名解析 252
刷新 DNS 缓存 252
8.3 DNS 劫持 253
第九章 防火墙过滤（Firewall Filte） 255
9.1 Firewall 过滤 255
防火墙过滤方式 257
自定义链表 258
防火墙过滤流程 258
9.2 防火墙 filter 事例 259
Input 事例 259
Forward 事例 260
Jump 规则的使用 262
基于协议的自定义链表 265
ICMP 类型：代码值 266
防火墙 action 命令说明 266
源 IP 地址与目标 IP 地址 267
文本过滤 269
9.3 Address-list 269
v6.36 允许 address-list 添加域名 271
9.4 Interface list 273
9.5 内网多 IP 段访问控制 273
9.6 P2P 协议过滤 275
9.7 RouterOS L7 协议 275
9.8 使用 wireshark 分析网页视频 278
9.9 DMZ 配置事例 282
9.10 RouterOS v6 ip settings 283
9.11 DoS 防御 284
诊断 SYN 攻击 285
保护路由器 285
第十章 网络地址翻译 nat 287
10.1 nat 介绍 287
10.2 src-nat 291
Same nat 291
PCC src-nat 负载均衡 292
10.3 dst-nat 293
端口映射配置 294
dst-nat 数据转移 294
dst-nat 数据重定向 294
DNS 重定向 294
1:1 nat 实例 294
非对称端口映射 295

10.4 Stats 295
10.5 Connection 连接状态 296
10.6 Tracking 连接跟踪 297
10.7 高性能 nat 实践 299
10.8 Raw 301
第十一章 RouterOS Fastpath **303**
11.1 RouterBOARD FastPath 支持列表 304
11.2 CCR 系列的 Fastpath 305
11.3 IPv4 FastTrack 307
第十二章 带宽控制（Queue） **310**
12.1 流控原理 310
12.2 队列类型（Queue Type） 312
PFIFO 及 BFIFO 312
SFQ 312
PCQ 313
RED 314
Bursts 314
12.3 Burst 突发值 315
Burst 原理 315
Burst 事例 315
12.4 Simple Queue 简单队列 322
应用举例 323
12.5 RouterOS v6.0 Queue 变动 325
V6.0 前 325
V6.0 后 326
V6.0 配置变动 328
V6.0 Queue 事例 329
环境 1：Queue tree 设置 2M，simple queue 设置 6M 329
环境 2：Queue tree 设置 6M，simple queue 设置 2M 330
12.6 HTB 等级令牌桶介绍 331
双重限制 331
优先级 332
事例 1：普通事例 332
事例 2：max-limit 事例 333
事例 3: inner 队列 limit-at 334
事例 4：leaf 队列的 Limit-at 335
12.7 v6.0 Simple Queue 等级优先 337
12.8 Queue tree 队列树 340
简单的 Queue Tree 实例 341
12.9 PCQ 配置 343
分类器 344
简单的 PCQ 实例 346
12.10 HTB 与 PCQ 流量控制 349
12.11、网吧的 PCQ 与 HTB 353
HTB 游戏优先 354
12.12 Connection Rate 流量控制 355
12.13 L7 流控 373
12.14 PPP Profile 动态 QoS 376

第十三章 Mangle 分类标记 380
13.1 Mangle 介绍 380
13.2 Mangle 应用 381
Peer-to-Peer 传输标记 381
Mangle 限制 2 级代理 382
第十四章 Bridge 网桥 383
14.1 Bridge 配置 383
Bridge setting 385
Bridge Port 386
Bridge monitor 387
Bridge host 388
14.2 Bridge 防火墙 388
Bridge Filter 391
Bridge nat 392
14.3 Bridge 实现二层端口隔离 393
14.4 如何建立一个透明传输整形器 396
14.5 通过 Bridge Filter 控制 MAC 地址 397
过滤源和目标 MAC 地址 398
过滤指定厂商的 MAC 地址 400
第十五章 VLAN 403
15.1 802.1Q 协议 403
Q-in-Q 404
15.2 简单 VLAN 事例 405
15.3 Trunk 连接 407
15.4 基于 VLAN 的 PPPoE 认证 408
QinQ 添加脚本 410
第十六章 Bonding 412
16.1 Bonding 基本操作 412
16.2 RouterOS 与交换机做 Bond 配置 414
方法一、balance-rr 414
方法二、balance-alb 415
16.3 官方 EoIP 隧道的 Bonding 415
第十七章 虚拟路由冗余协议(VRRP) 419
17.1 VRRP 路由 419
17.2 简单的 VRRP 事例 420
17.3 VRRP 多 PPPoE 拨号接入配置 423
第十八章 RouterOS Nth 430
18.1 Nth 原理介绍 430
18.2 Passthrough 对 Nth 的控制 430
18.3 Nth 在负载均衡的应用 432
18.4 Nth 在端口映射的应用 434
第十九章 RouterOS 数据流(Packet Flow) 437
19.1 IP 数据流流程图 437
二层网络 437
三层网络 437
19.2 RouterOS 6.0 对 Queue 调整后的数据流 439
19.3 功能模块与结构 440
判断处理 440

19.4 功能处理流程 441
桥接设置 use-ip-firewall=yes 441
路由 – 从 Ethernet 到 Ethernet 接口 441
路由 - 从一个桥接口道不同的桥接口 441
IPsec 加密 442
IPsec 解密 442
第二十章 RADIUS 444
20.1 RADIUS 客户端 444
RADIUS 连接终止 447
20.2 RouterOS 连接 RADIUS 备份 447
RADIUS 备份模式 447
第二十一章 HotSpot 热点认证网关 449
21.1 HotSpot 介绍 449
21.2 HotSpot Setup 向导配置 451
21.3 HotSpot 接口设置 453
21.4 HotSpot profile 策略 453
21.5 Walled Garden 内院 455
IP 方式 Walled Garden 457
21.6 Hotspot binding 459
21.7 Hotspot host 列表 460
21.8 HotSpot Users 管理 462
21.9 Hotspot active 在线管理 465
21.10 Hotspot 配置事例 466
21.11 Hotspot 即插即用功能 475
21.12 HotSpot 防火墙部分 477
第二十二章 PPPoE 配置 481
22.1 PPPoE Client 设置 481
监视 PPPoE 客户端 482
22.2 ADSL 拨号上网事例 483
22.3 PPPoE Server 设置 484
22.4 基于 802.11g 无线网络的 PPPoE 服务 485
22.5 Winbox 配置 PPPoE 服务 487
22.5 大型 PPPoE 服务应用 492
第二十三章 PPTP 495
23.1 PPTP 客户设置 496
23.2 PPTP 服务器设置 497
23.3 PPTP 应用实例 497
Router to Router 安全隧道实例 497
通过 PPTP 隧道连接终端客户 503
Windows 的 PPTP 设置 505
第二十四章 PPTP 与 L2TP 服务 507
24.1 同时建立 PPTP 和 L2TP 服务器 507
L2TP 的 Windows 注册表修改 510
24.2 VPN 的几种应用方式 511
24.3 BCP 协议 513
第二十五章 Open VPN 520
25.1 OVPN Client 520
25.2 OVPN Server 521

OVPN 服务器配置 521
25.3 使用 certificate 创建 OVPN 证书 522
25.4 OVPN 配置 527
25.5 OVPN bridge 模式 534
第二十六章 SSTP 介绍 539
26.1 端到服务器连接 541
26.2 点对点的 SSTP 543
第二十七章 IPSec 配置 545
27.1 IPSec 点对点配置实例 545
27.2 Windows L2TP/IPsec 连接 553
IPSec 配置 554
RouterOS 配置 555
Windows 配置 557
v6.16 后简化 L2TP/IPsec 配置 561
第二十八章 EoIP 隧道 562
28.1 EoIP 配置 562
28.2 EoIP 应用实例 563
第二十九章 GRE 和 IPIP 隧道协议 567
29.1 GRE 隧道 567
配置事例 567
29.2 IPIP 隧道 569
配置事例 569
第三十章 OSPF 571
30.1 OSPF 介绍 571
OSPF 术语 571
OSPF 路由通信 573
邻居探测 574
数据库同步 575
路由表计算 575
OSPF 路由类型 576
30.2 基本的 OSPF 配置 577
网络配置 578
配置 router-id 578
Area 配置 579
重分布默认路由 580
30.3 多 area 配置 581
30.4 基于 PPPoE 的 OSPF 事例 582
第三十一章 BGP 595
31.1 BGP 介绍 595
自治系统 AS（Autonomous System） 595
BGP 分类 595
BGP Instance 实例 596
BGP Peers 596
路由重分布 597
路由过滤 597
RouterOS 的 BGP 静态路由 599
BGP Advertisements 600
BGP 负载均衡 600

31.2 BGP 事例 1 601
配置 BGP Peer 602
network 宣告和路由过滤 602
路径选择 602
负载均衡设置 603
31.3 BGP 事例 2 605
第三十二章 Proxy 代理 613
32.1 access 访问列表 615
32.2 Direct 直接访问列表 616
32.3 cache 缓存管理 617
32.4 连接列表 617
32.5 Web 代理应用事例 618
32.6 重定向 URL 请求 621
第三十三章 虚拟化技术 625
33.1 虚拟化技术应用 625
33.2 KVM 介绍 626
AMD 支持情况 626
Intel 支持情况 626
33.3 MetaRouter 介绍 627
33.4 MetaRouter 事例 629
33.5 CHR(Cloud Hosted Router) 635
Cloud Host Router 安装到 VM EX 平台 636
第三十四章 IP Accounting 访问日志记录 648
34.1 IP 访问记录 648
IP 访问快照 648
34.2 Web 获取 IP 访问信息 649
第三十五章 RouterOS Stores/disk 功能 651
35.1 RouterOS 使用 U 盘扩展存储 651
35.2 存储 log 日志信息 652
35.3 Web-Proxy 使用 U 盘存储 655
35.4 Store/disk 的其他应用 656
第三十六章 RouterOS 工具 658
36.1 Netwatch 监控 658
36.2 绘图显示（Graphing） 659
健康情况 660
接口流量图 660
流控图显示 661
资源图表 661
36.3 Bandwidth-text 带宽测试 663
36.4 Torch (即时通信监听) 665
36.5 E-mail 发送工具 669
36.6 Feth 文件拷贝 670
36.7 3G 模块设置 671
联通 3G 拨号测试 672
电信 CDMA 上网卡配置 674
36.8 Packet Sniffer 677
36.9 MikroTik SMB 684
36.10 Profiler 工具 689

多 CPU 系统下查看 690
第三十七章 User Manager v4 692
37.1 User Manage 基本配置 692
创建 Customer 账号 693
操作界面 694
语言设置 696
37.2 对接 RouterOS 配置事例 697
37.3 PPPoE 认证的事例操作 701
37.4 Hotspot 认证 User Manager 连接 717
37.5 用户自助服务 718
第三十八章 Scheduler（计划任务） 722
38.1 计划任务介绍 722
38.2 计划任务事例 722
Changelog 726
参考文献： 728

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 v6.x 版本，使用 Linux 3.3.5 的内核，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：

等级 / 功能	Level 0	Level3	Level 4	Level 5	Level 6
升级	无法升级				
无线 AP	24 小时	不支持	支持	支持	支持
无线桥接和客户端	24 小时	支持	支持	支持	支持
RIP，OSPF，BGP 协议	24 小时	支持	支持	支持	支持
EoIP 隧道在线用户	24 小时	1 条	无限制	无限制	无限制
PPTP 隧道在线用户	24 小时	1 条	200	无限制	无限制
PPPoE 隧道在线用户	24 小时	1 条	200	500	无限制
L2TP 隧道在线用户	24 小时	1 条	200	无限制	无限制
OVPN 隧道在线用户	24 小时	1 条	200	无限制	无限制
SSTP 隧道在线用户	24 小时	1 条	200	无限制	无限制
Hotspot 认证在线用户	24 小时	1 条	200	500	无限制
VLAN	24 小时	1 条	无限制	无限制	无限制
P2P 防火墙规则	24 小时	1 条	无限制	无限制	无限制
NAT 规则	24 小时	无限制	无限制	无限制	无限制
RADIUS 客户端	24 小时	支持	支持	支持	支持
Queue 流量控制规则	24 小时	无限制	无限制	无限制	无限制
Web 代理	24 小时	支持	支持	支持	支持
User Manager 在线用户	24 小时	10	20	50	无限制

x86 平台

- AMD、Intel、VIA 和其他兼容的 x86 平台
- SMP – RouterOS 3.0 后兼容多核心 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；
- RouterOS 5.0 以前都采用 Linux v2.6 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。

X86 兼容硬件支持，参考 http://wiki.mikrotik.com/wiki/Supported_Hardware

MIPS 平台

- **支持系统** – 4kc MIPS　RouterBOARD 500 (532, 512 和 511)与 RouterBOARD 100 (133、133c、150、192)
- **支持系统** – 24kc MIPS RouterBOARD 400(411/411A/411AH、433/433AH/433UAH、450/450G、493/493AH)

- **支持系统** – 24kc MIPS RouterBOARD 700(711、711A、750/750G、750UP、751 系列，SXT、Groove、OmniTik)
- **支持系统** – 24kc MIPS hAP 系列
- **支持系统** –74kc MIPS RouterBOARD2011 系列
- **支持系统** –74kc MIPS RouterBOARD900 系列(911、951、912)
- **支持系统** –74kc MIPS CCR 系列

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，其他都是 32bit 系统

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 盘写入安装文件，实现 U 盘安装
- CD 镜像文件安装

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 Binary configuration backup saving and loading
- 通过 Exprot 和 import 可生成读写的文本格式

Firewall

- 状态过滤功能 Statefull filtering
- 源和目标 NAT
- NAT 助手(h323, pptp, quake3, sip, ftp, irc, tftp)
- 内部连接状态，路由和数据包标记
- 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

- …

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 无线 AP 和客户端
- 支持 IEEE802.11n
- 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 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
- 用户账户记录
- 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 数据包转发不在经过 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
- 数据包 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 数据包封装协议
- 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：

- 完全一至的用户接口
- 运行时配置和监控
- 支持多个连接访问
- 用户策略配置
- 恢复和撤销历史记录，undo/redo 操作
- 安全模式操作
- Scripts 能事先安排执行时间和执行内容，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 显示卡进行控制
- **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 服务默认运行在 TCP 端口 23
- **SSH** - SSH (安全 shell) 服务默认运行在 TCP 端口 22
- **MAC Telnet** - MikroTik MAC Telnet 协议是默认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 界面管理，管理界面类似 Winbox，无需安装任何客户端。

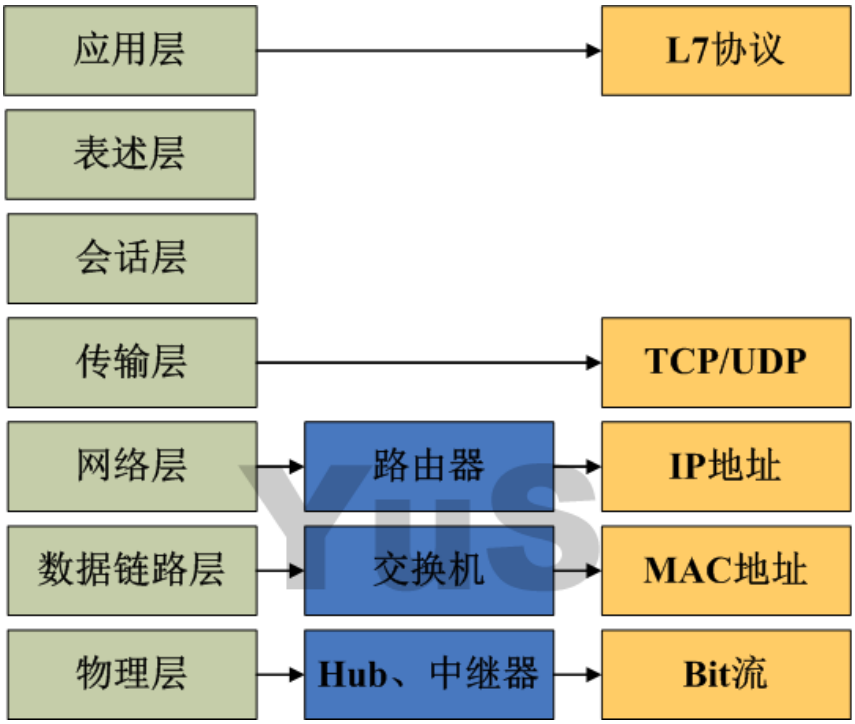
基础知识

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资料，例如 CCNA 和 HCNA 这些入门网络工程师的资料。

- **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

- 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交换机和网桥设备，通过 MAC 地址通信传输；
- 运行在网络层的有路由器，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 数据包发送到指定目的地；
- 应用层，主要是 http、DNS、FTP、P2P 等应用协议等；

二层 数据链路层

二层交换机属数据链路层设备，可以识别数据包中的 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：

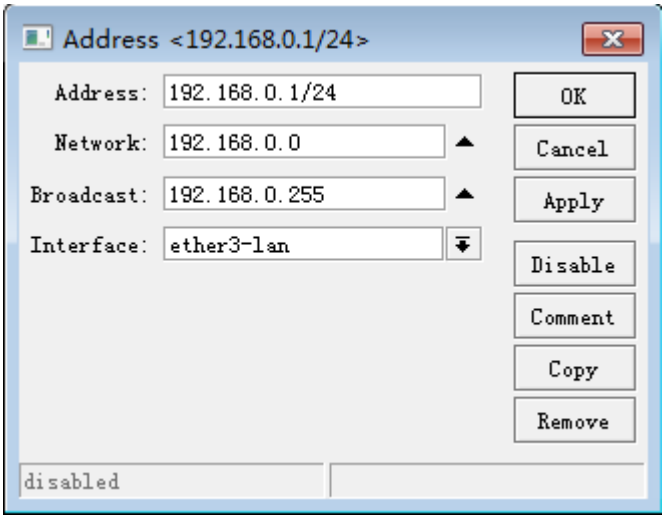
三层 网络层

- 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

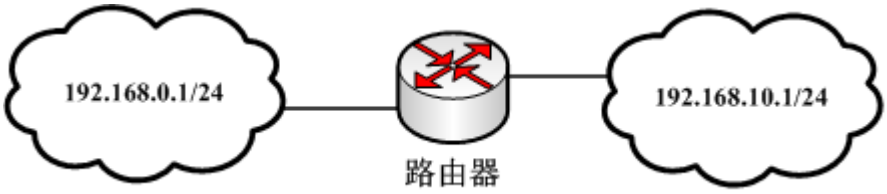
- 255.255.255.0 如用二级制表示每个 255 都是 2 的 8 次方（0-255）256

- 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

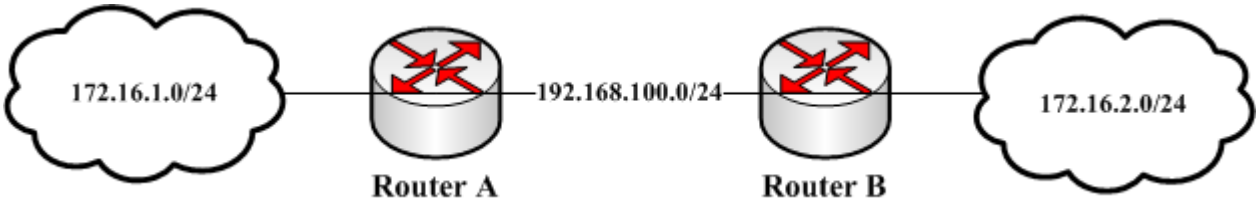


路由器

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 数据包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 数据包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”（即激活状态）。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	掩码	网关	标志	接口
73.0.0.0	255.0.0.0	201.66.37.254	AG	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	掩码	网关	标志	接口
91.32.74.21	255.255.255.255	201.66.37.254	AS	eth0

重叠路由

假设在路由表中有下列重叠项：

目的	掩码	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叠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）：

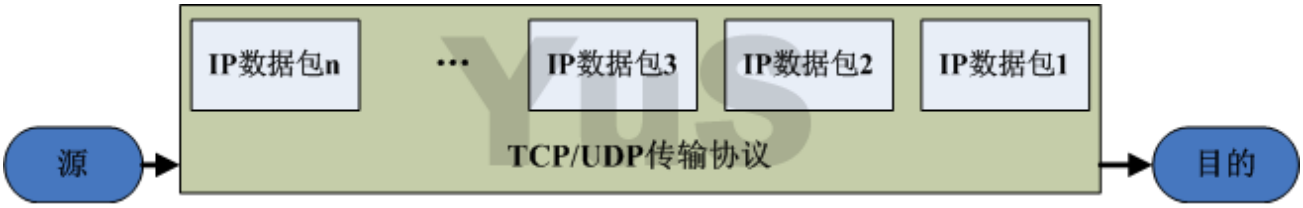
接口	IP 地址	子网掩码
eth0	201.66.37.1	255.255.255.0
eth1	201.66.37.2	255.255.255.0

对于重叠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 数据包由源发向目标。

- 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 核心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；
- 兼容当前的所有 x86 平台的硬件，如 Intel 和 AMD 等，支持自助开发平台 RouterBOARD、CRS 和 CCR 系列。

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，并可以通过安装文件建立自己的 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。多数路由器的处理器多在 533MHz-800MHz。

5、MikroTik RouterOS 是否支持多 CPU 处理？

早期的版本不支持多 CPU 平台，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。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。

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，SCSI 和 SAS 不支持。

7、软件路由相对于 Cisco 路由器如何？

可用作到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。

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，但主要是 PC 的 x86 构架和嵌入式 MikroTik RouterBOARD 构架，其他平台暂不支持。X86 包括现在的 Intel、AMD 和 VIA 等处理器，而 MikroTik RouterBOARD 则包括 MIPS、PPC、Atheros 和 Tilera 平台。

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、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 镜像文件小于 30M，所以建议安装到 SSD 或 FLASH 硬盘上，如果没有特殊需求，建议系统硬盘不超过 32G。

2、使用 U 盘安装，仅支持 X86 平台，RouterOS 限 3.0 以上版本

3、使用 netinstall 网络安装程序，主要用于 RouterBOARD、CRS 或 CCR 平台，也适用于支持 PXE 的 x86 平台。

CD 光盘安装

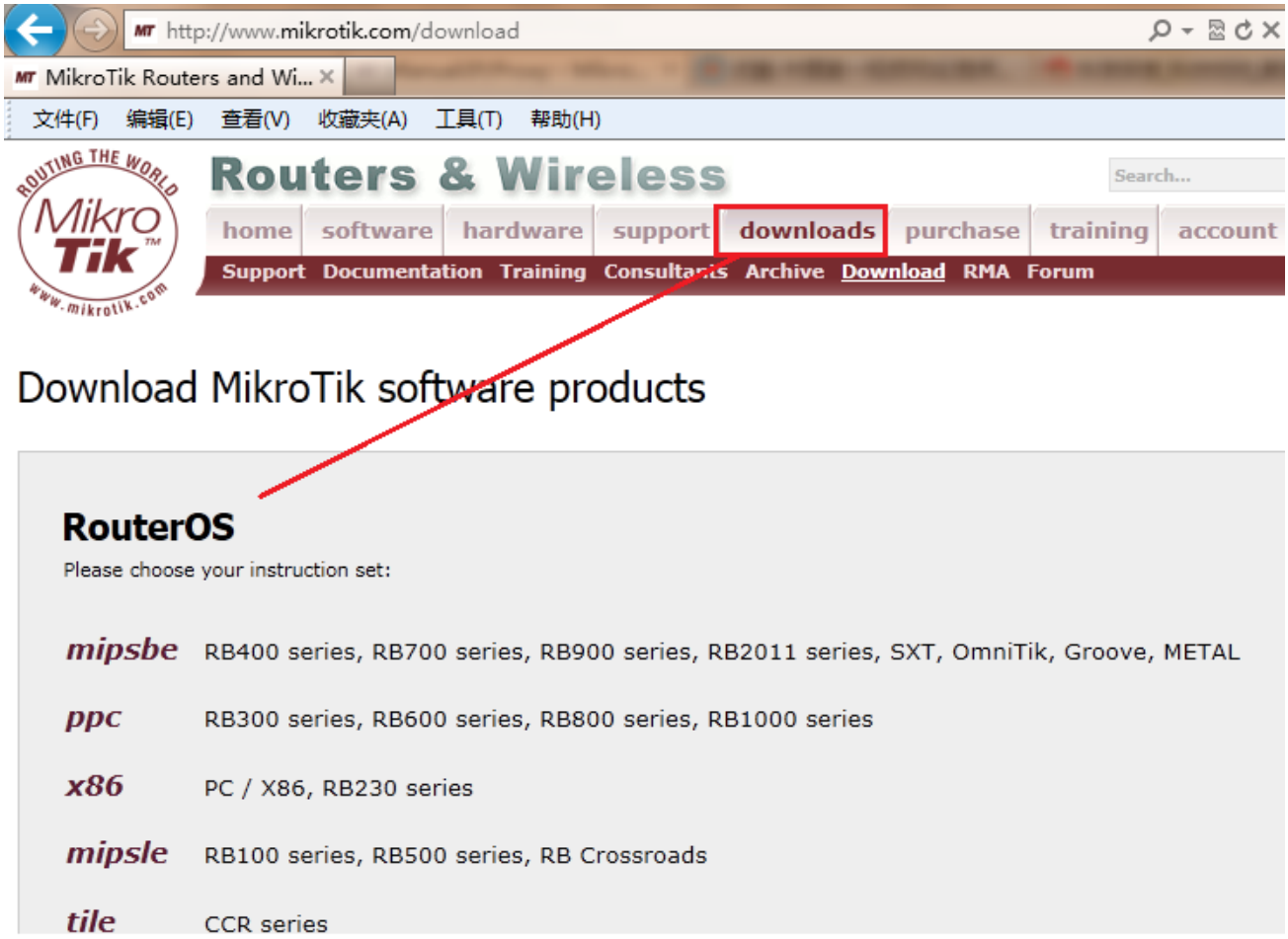
CD 安装是基于 PC 的 x86 硬件最常见的安装方式，通过 CD 将 MikroTik RouterOS 安装到基于 x86 平台的 PC 硬件上

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

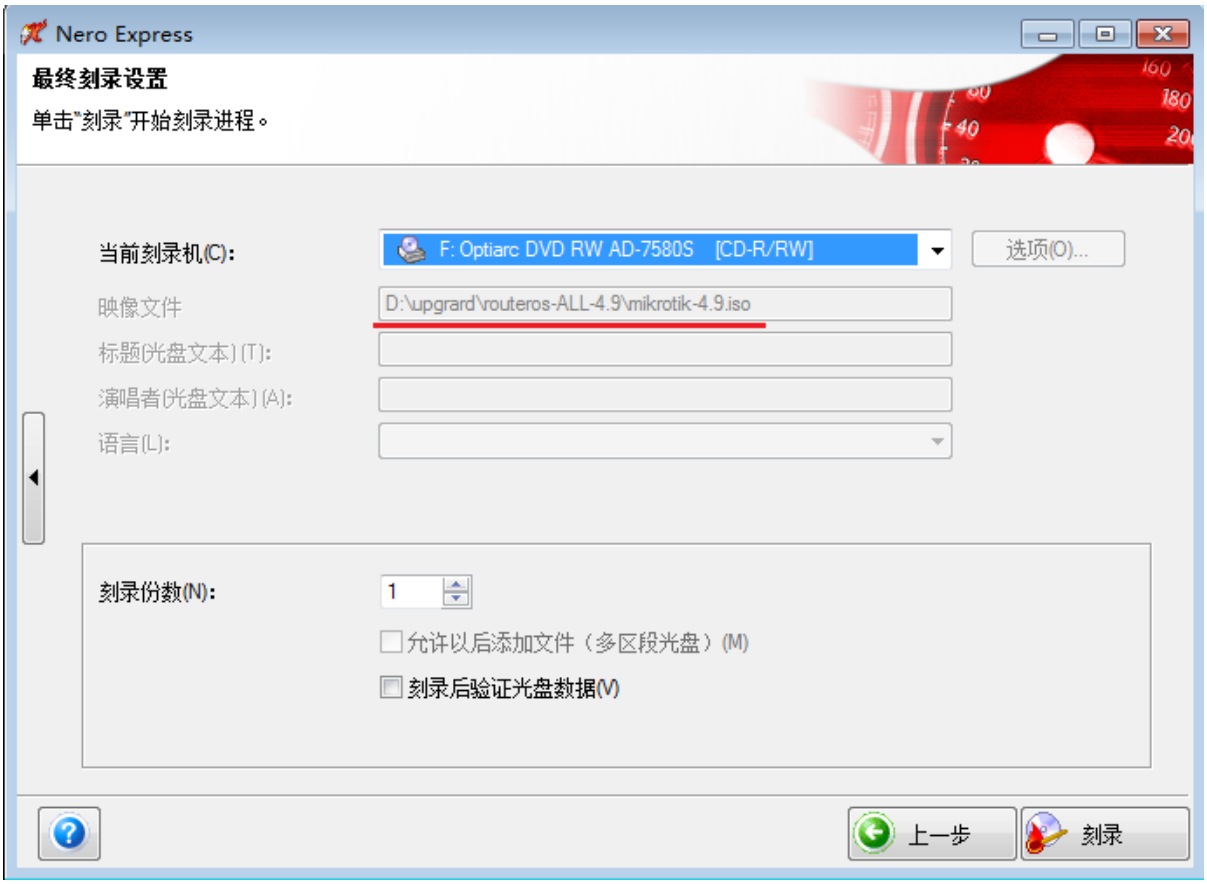
准备 MikroTik RouterOS 安装

1. 下载 MikroTik 的镜像文件下载页面,

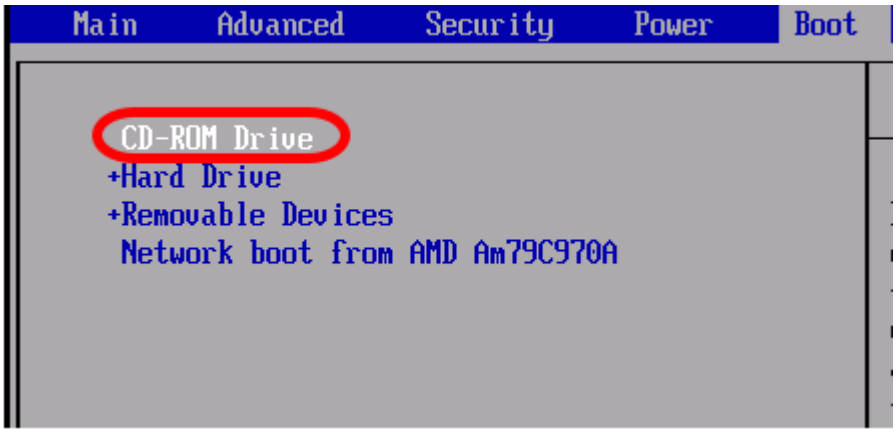


选择 x86 下的 CD Image

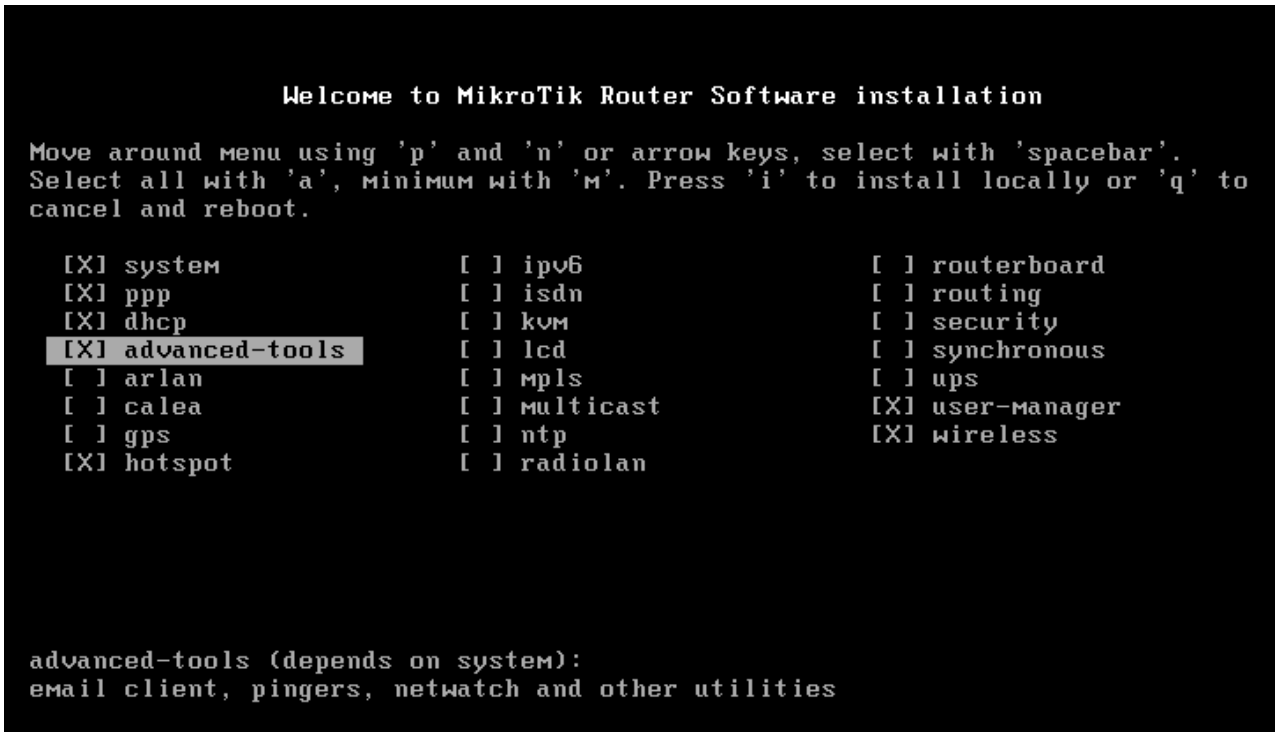
2.把 ISO 镜像下载到硬盘后，通过刻录光驱和刻录程序，将镜像刻录为 CD/DVD



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：

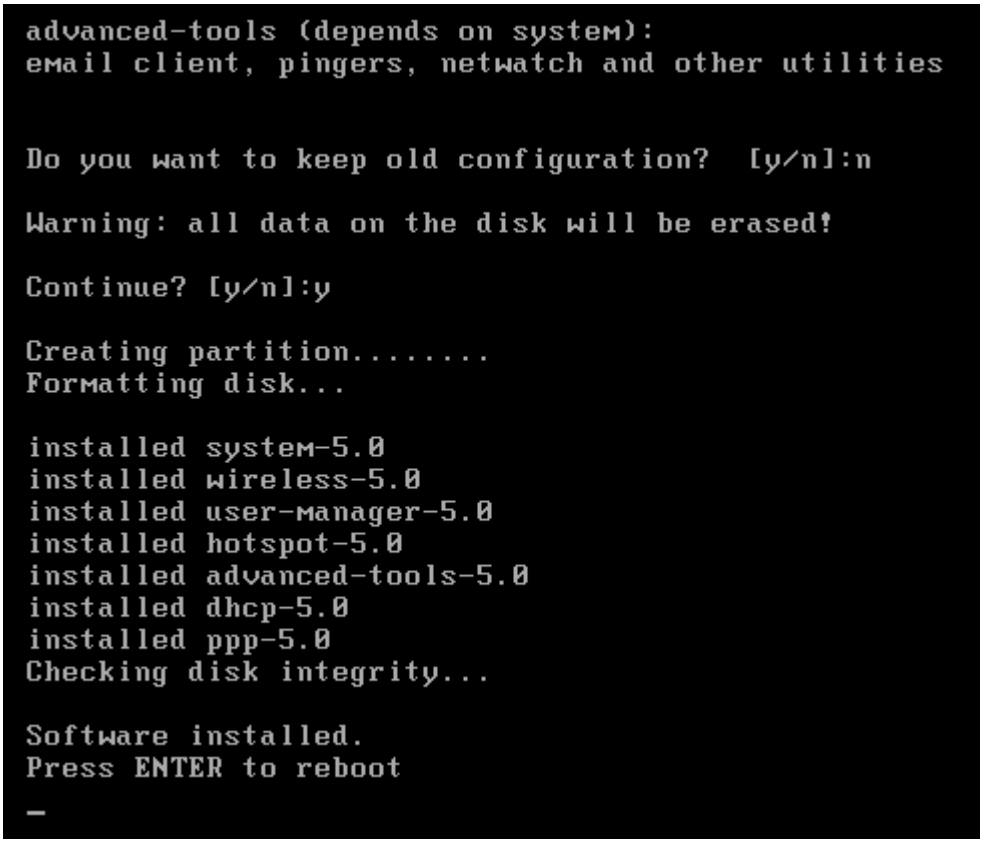


4. PC 将从 RouterOS CD/DVD 光盘引导启动，安装并选择相应的功能包

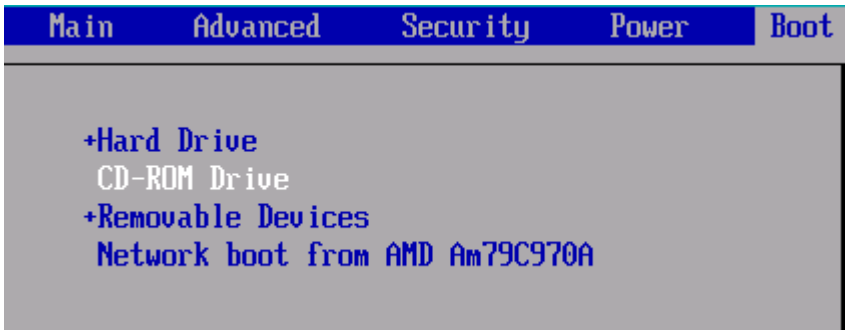


5. 选择你需要安装的功能包，使用“空格”选择功能包，“a”可以选择所有功能包，或者“m”选择最小安装，按“i”开始安装 RouterOS，如果你之前在同一台 PC 上安装过 RouterOS，你可以复位和保存设置，提示如下“**Do you want to keep old configuration?**”如果选择“**n**”复位设置，选择“**y**”保存之前配置

6. 安装完成后，将提示你重启



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



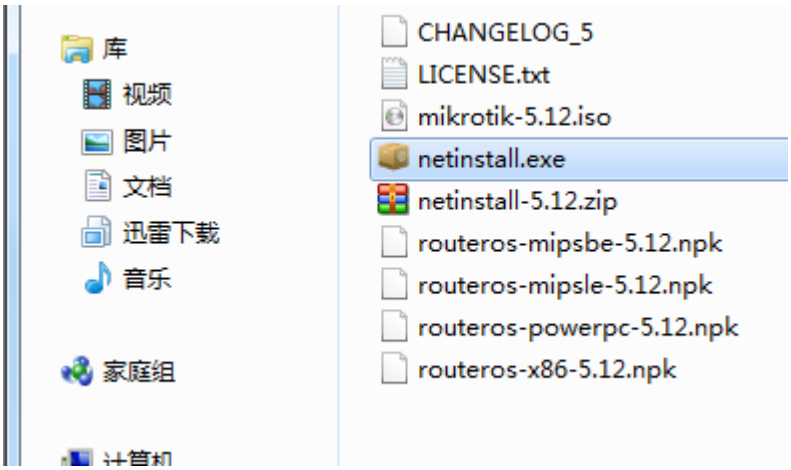
8. 启动后 RouterOS 提示登录信息，登录默认账号是 admin，没有密码

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。

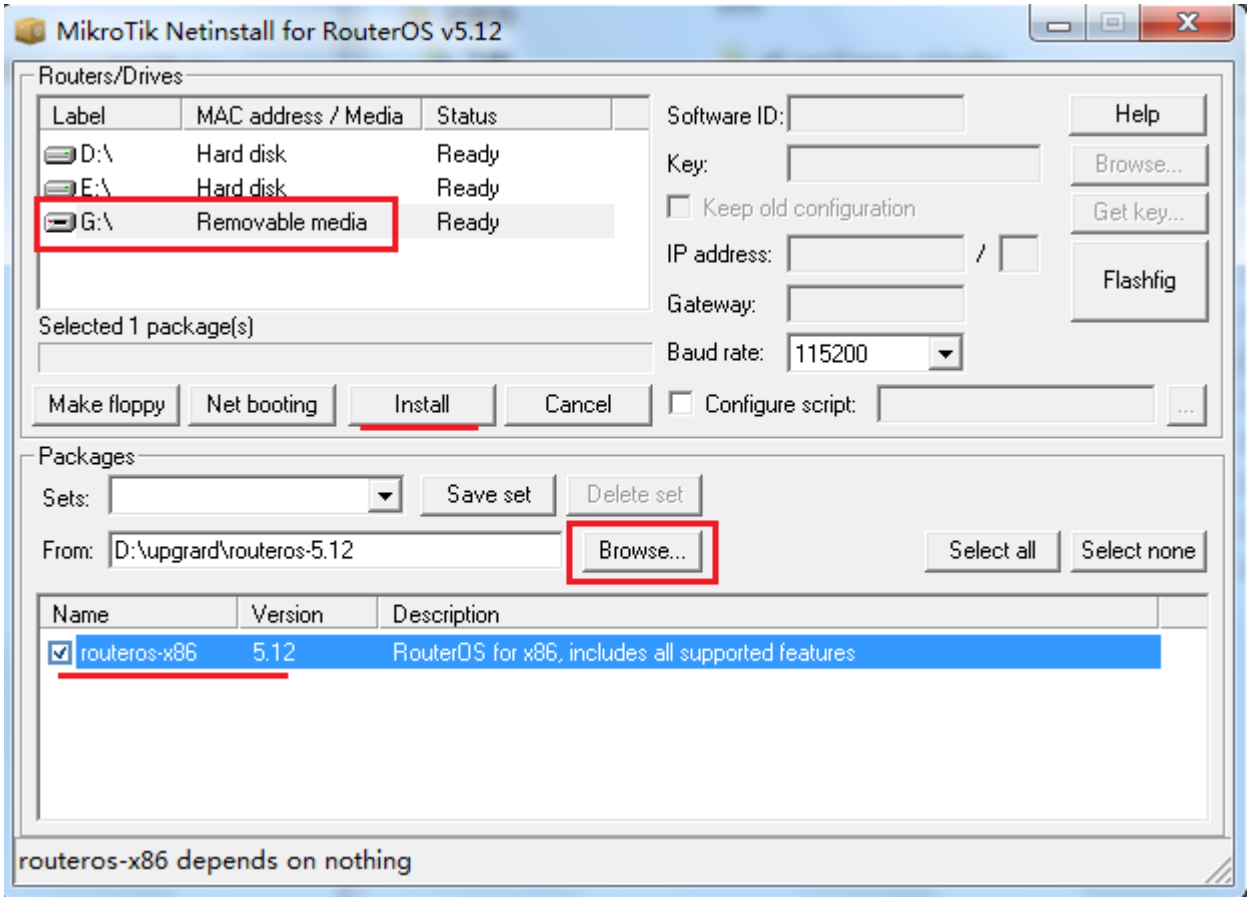
```
  MMM      MMM       KKK                          TTTTTTTTTTT      KKK
  MMMM    MMMM       KKK                          TTTTTTTTTTT      KKK
  MMM MMMM MMM  III  KKK  KKK  RRRRRR     OOOOOO      TTT     III  KKK  KKK
  MMM  MM  MMM  III  KKKKK     RRR  RRR  OOO  OOO     TTT     III  KKKKK
  MMM      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 5.0 (c) 1999-2011       http://www.mikrotik.com/

UPGRADE NOW FOR FULL SUPPORT
----------------------------
FULL SUPPORT benefits:
- receive technical support
- one year feature support
- one year online upgrades
    (avoid re-installation and re-configuring your router)
To upgrade, register your license "software ID"
 on our account server www.mikrotik.com

Current installation "software ID": ZYBI-N3R2

Please press "Enter" to continue!
```

USB 安装

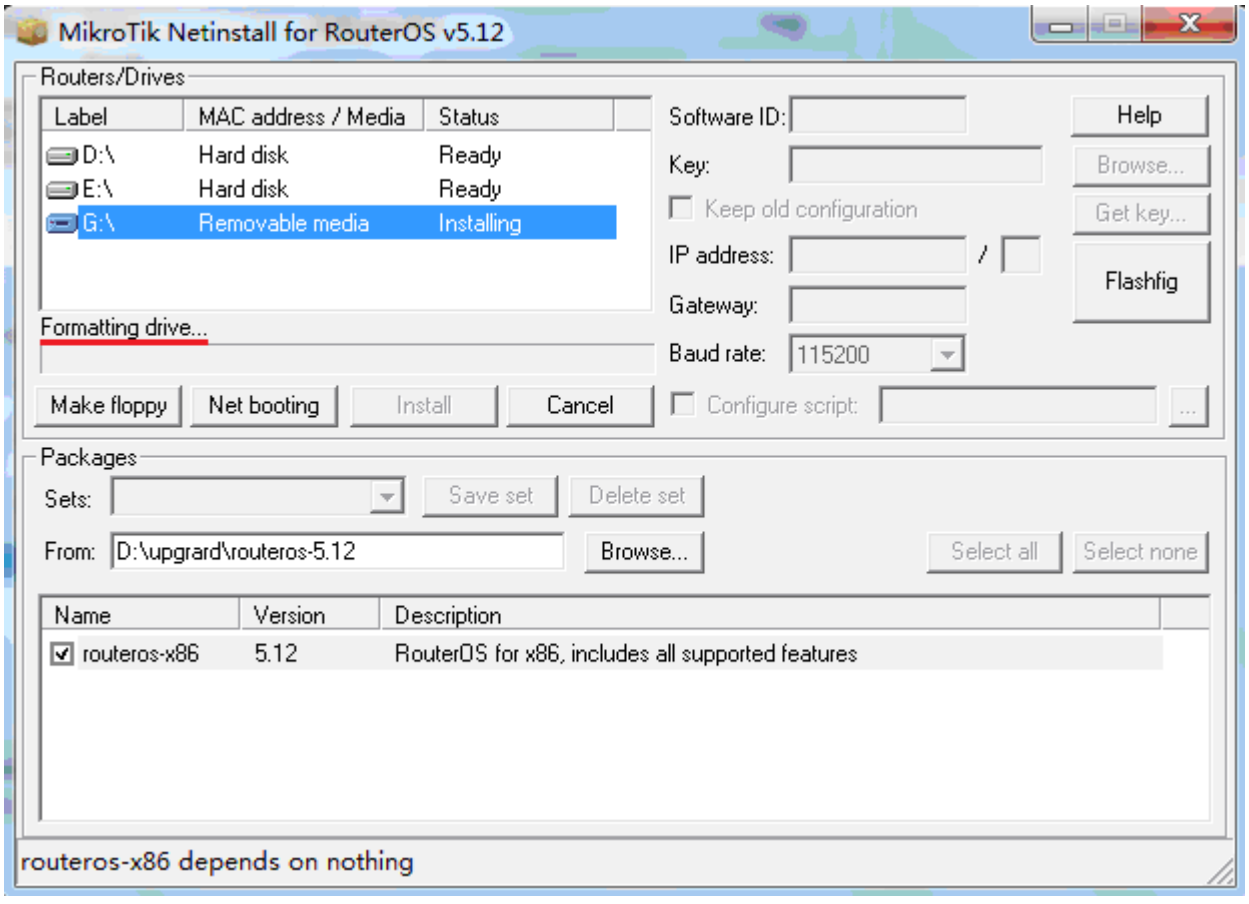
1、到 www.mikrotik.com 下载 netinstall 软件，并找到 netinstall.exe 软件



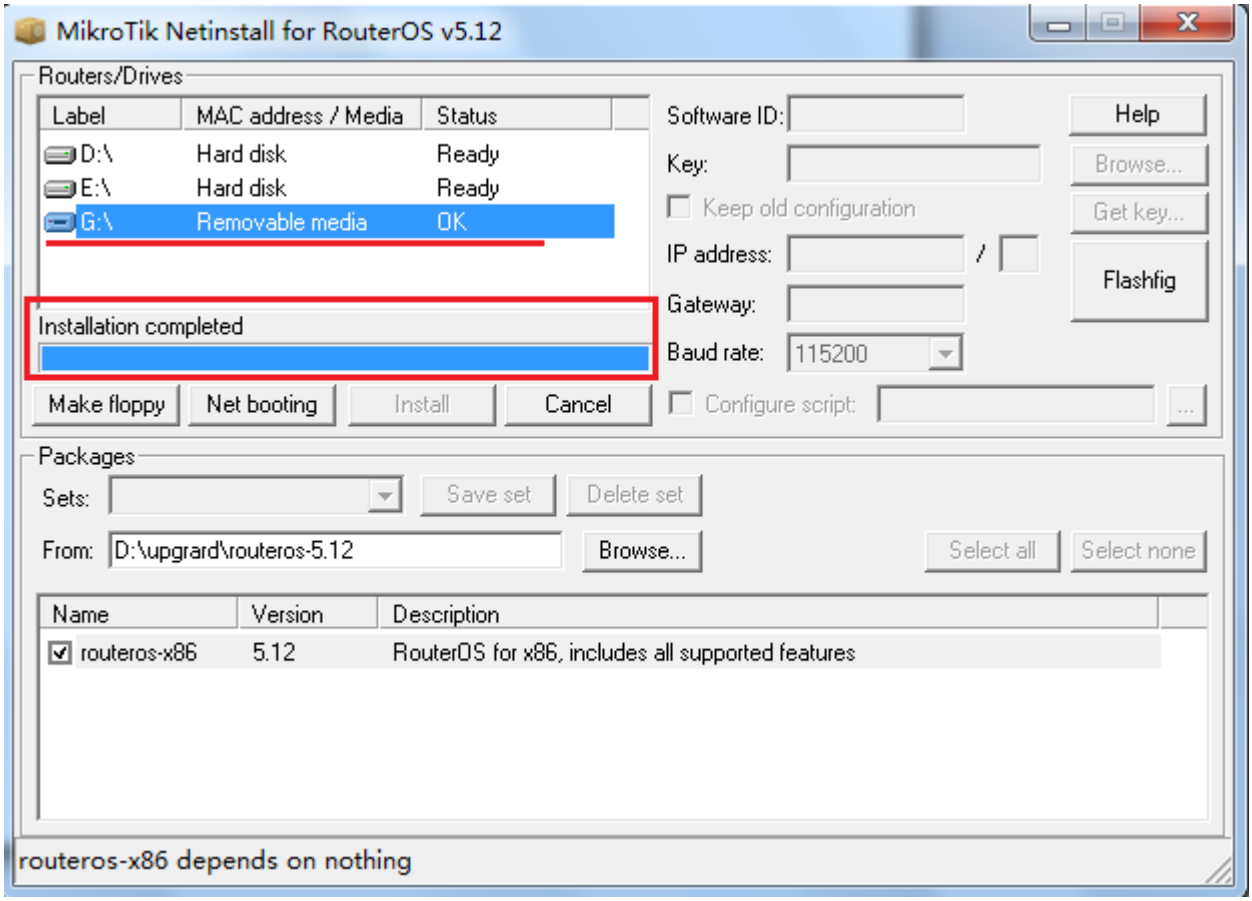
2、U 盘安装需要使用 3.0 以上版本 netinstall 软件，将 U 盘插入一台 Windows 电脑的 USB 接口后，选择 routeros-x86 安装包启动 netinstall 软件，并找到 Removable media 设备，即移动存储的 U 盘。选择 Browse 选择安装包的路径，找到路径后，软件会自动筛选相应的功能包，U 盘使用的是 routeros-x86 功能包，然后点击 install 安装系统。



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，并设置电脑的 BIOS 通过 USB 引导启动，启动后可以看到系统正在安装。

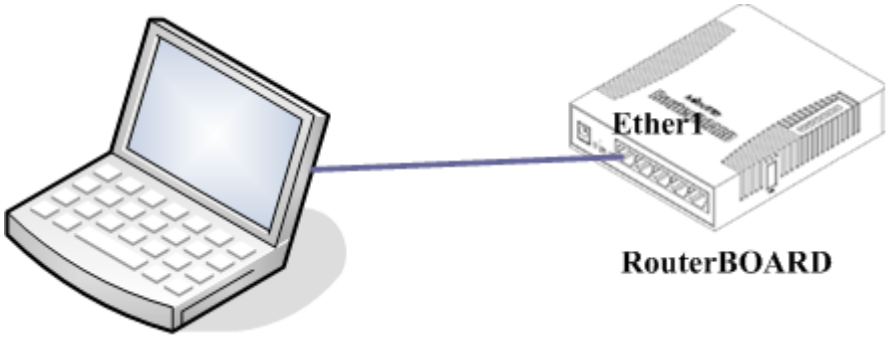
NetInstall 安装

Netinstall 还支持网络安装 RouterOS，这种方式应用于 RouterBOARD 的 RouterOS 系统安装和复位，以及支持 PXE 网卡引导的 PC，这里我们主要讲 RouterBOARD 的操作，PXE 的 PC 主机，可以在 BIOS 中配置引导，操作类似 RouterBOARD 安装。

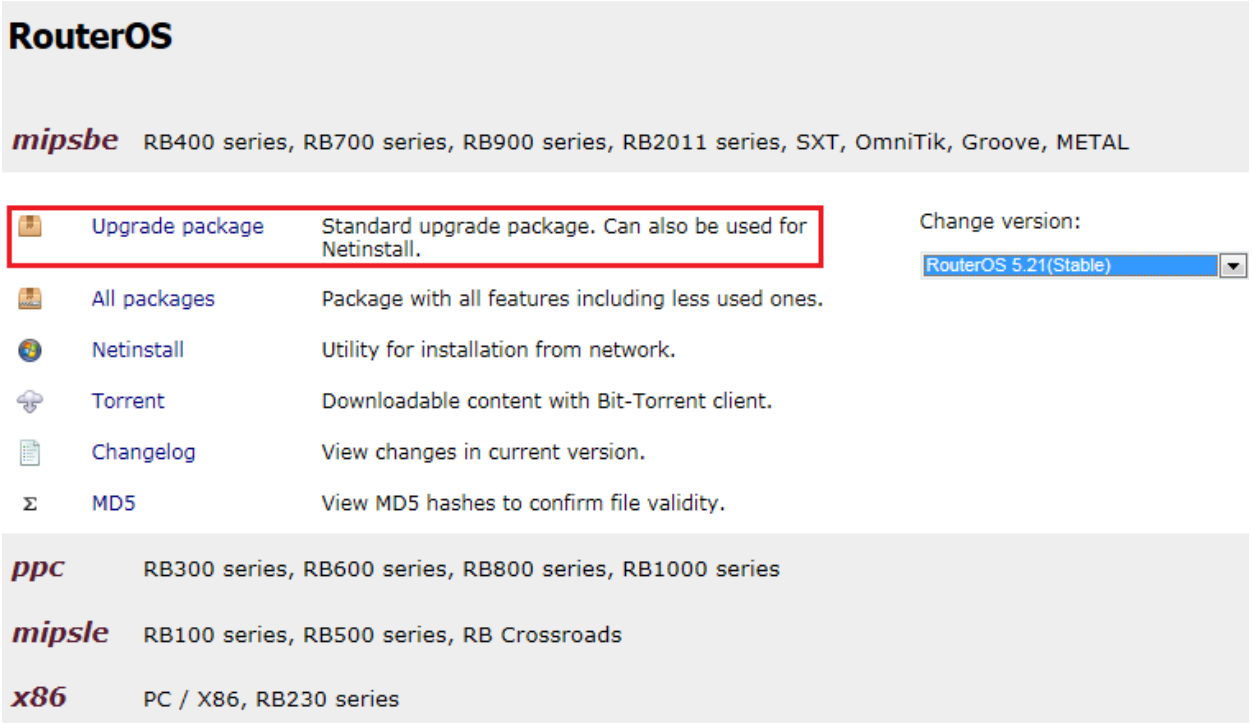
1、安装 RouterBOARD

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。

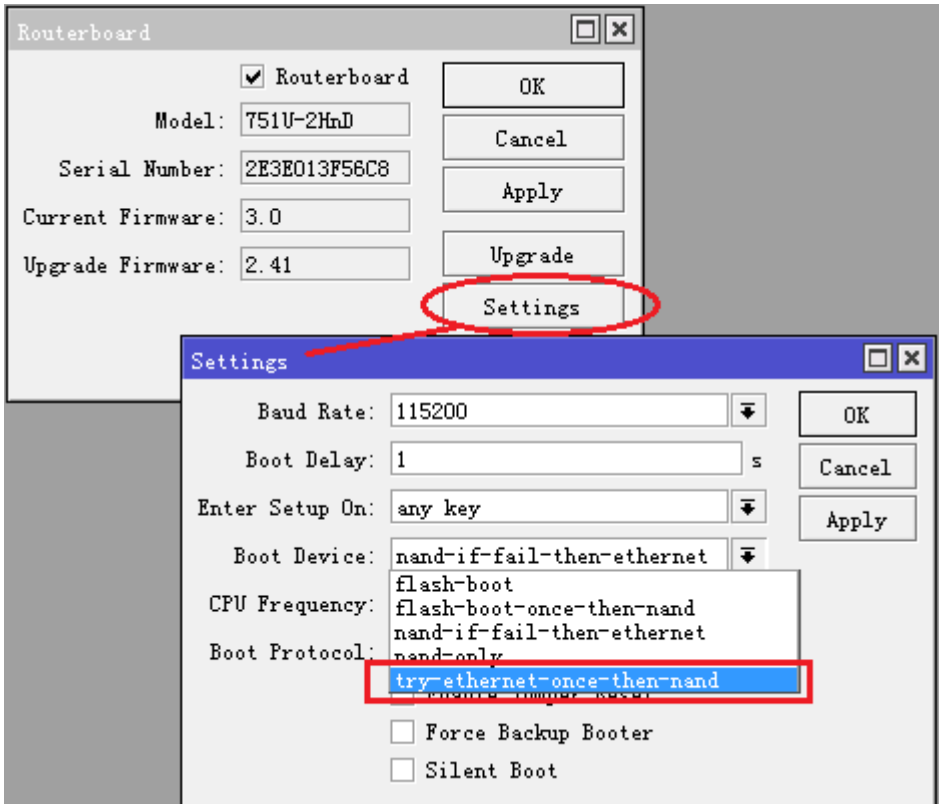
Netinstall 早期需要通过 console 口连接进入 RouterBOARD 的“BIOS”修改引导方式为 ethernet（以太网引导），即通过使用 netinstall 工具进行网络安装，现在 RouterBOARD 提供了新的方式可以在没有 console 口的情况安装，一种是 winbox 设置引导，一种是长按固件复位按钮引导。



首先我们需要准备 netinstall 工具和 RouterOS 安装包，这些工具和软件我们可以从 http://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 upgrade package。



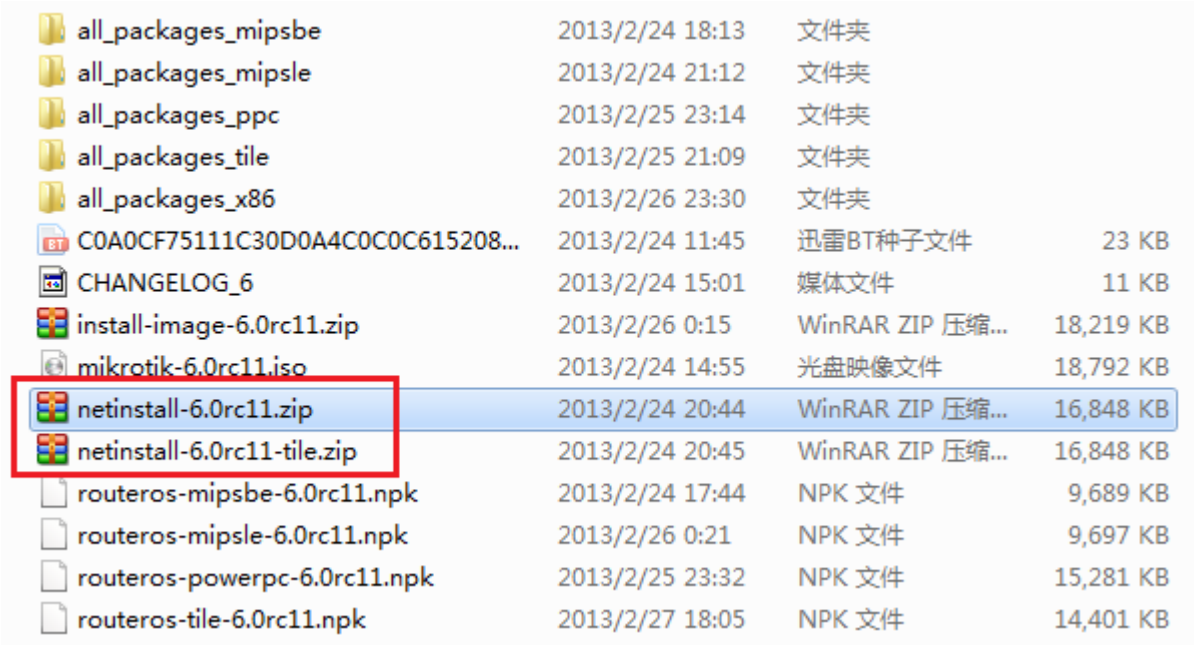
准备好工具和软件后，现在需要电脑和 RB 设备通过网线连接，将网线接在 RB 设备的 Ether1 上，然后进入 winbox 选择 system->routerboard->settings 修改重启后的引导方式。如下图：



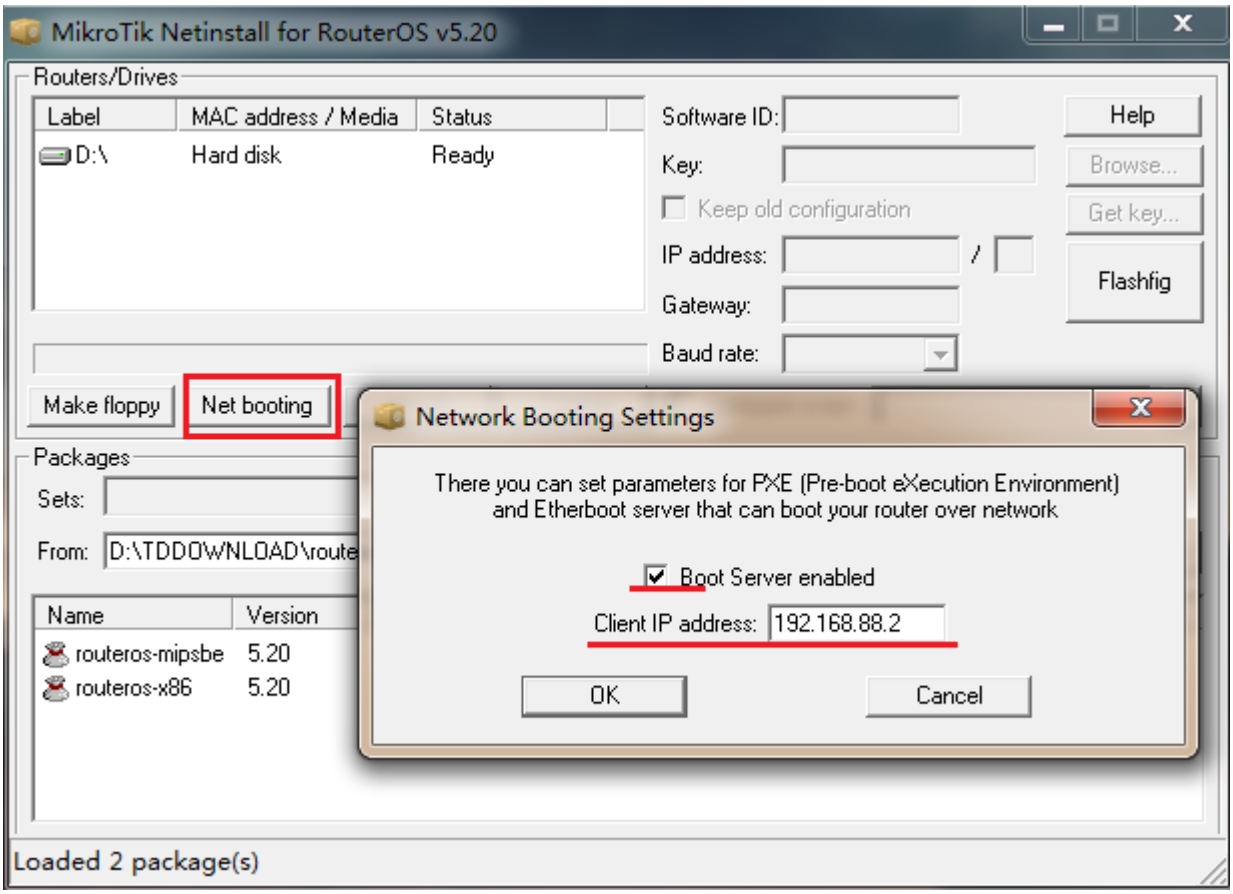
这里我们选择 try-ethernet-once-then-nand，即通过 ethernet 引导一次，如果失败就选择 nand 引导

以上配置完成后，启动 netinstall，注意如果你是 windows vista 或 7、8 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。

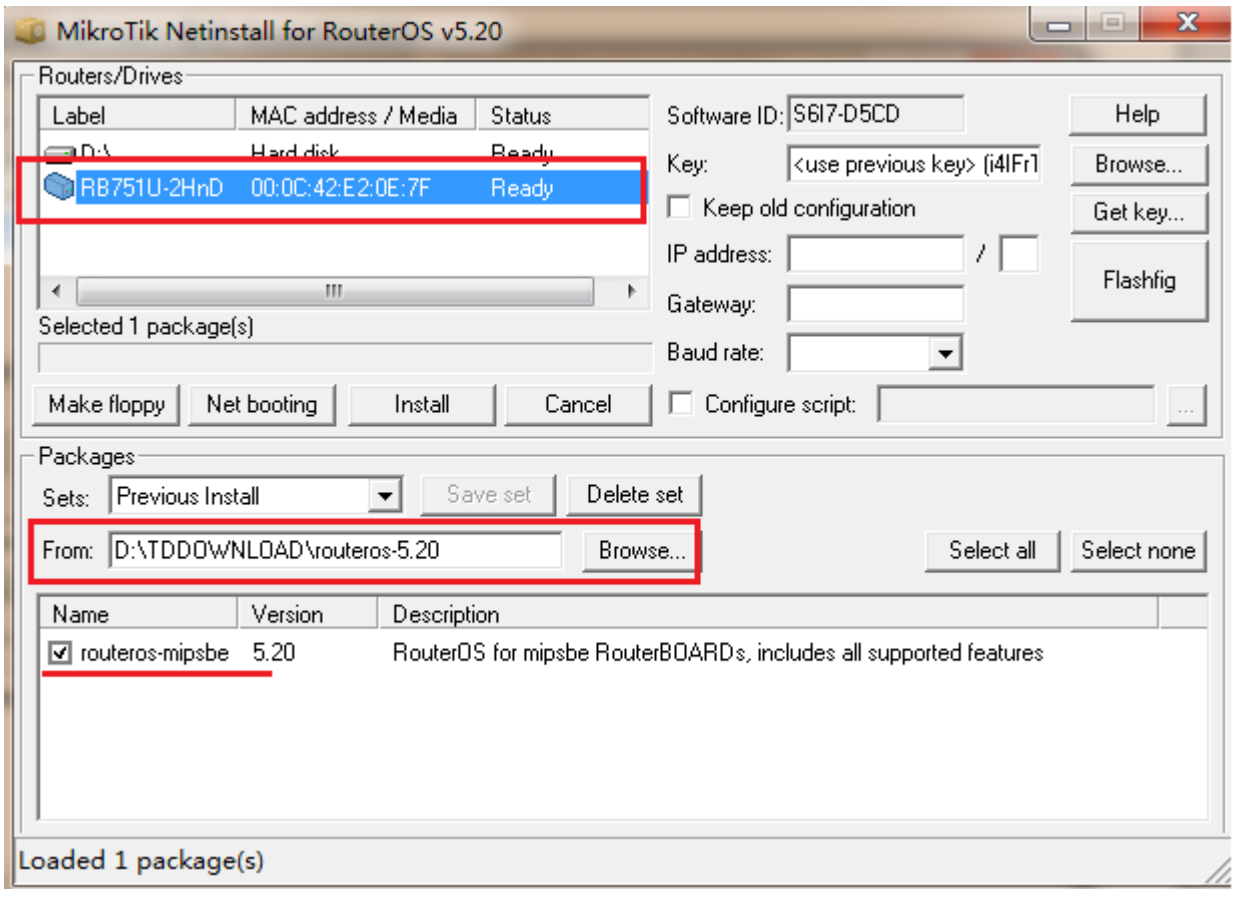
注意 6.0 版本后，netinstall 软件分被划分了两个版本一个是新的 tile 硬件（CCR 系列设备）和普通版本，即除了 tile 硬件以外的 RB 和 PC。



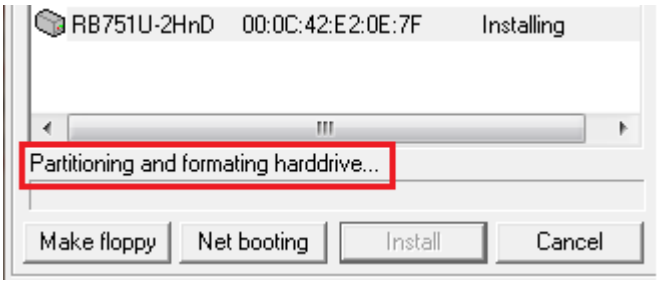
启动软件后，我们首先配置 Net booting，选择 Boot Server Enabled，设置网络连接 ip 地址，这个地址一定要和你的电脑的网卡在同一网络段中，如我的电脑是 192.168.88.3/24，配置给 Netinstall 分配给 RB 的地址是 192.168.88.2



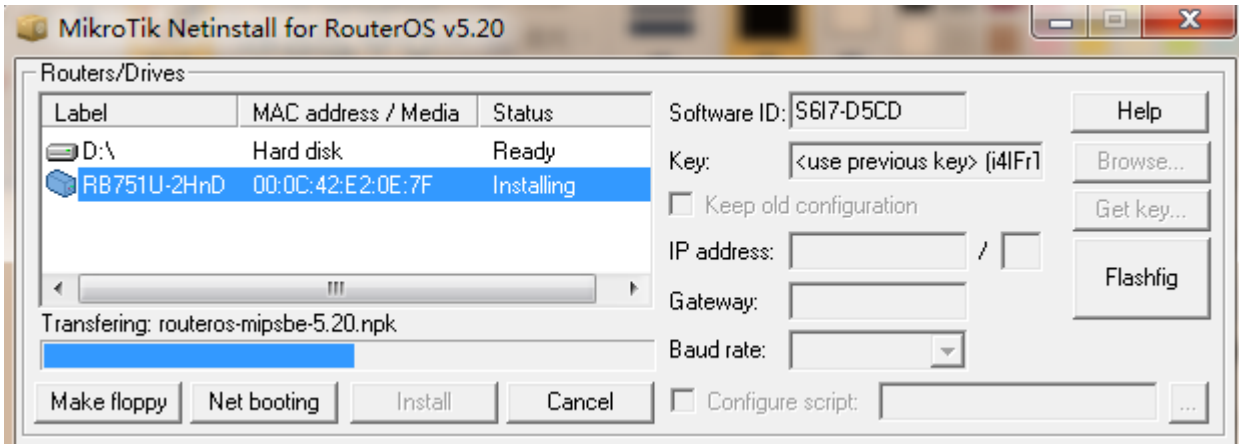
设置好以上参数，就可以通过命令重启 RouterBOARD，重启后 RouterBOARD 会自动搜索以太网引导服务器，找到后会在 netinstall 中显示设备型号和 mac 地址，显示 Ready 状态，你指定好安装包路径后，会自动选择设备的安装包，如下图



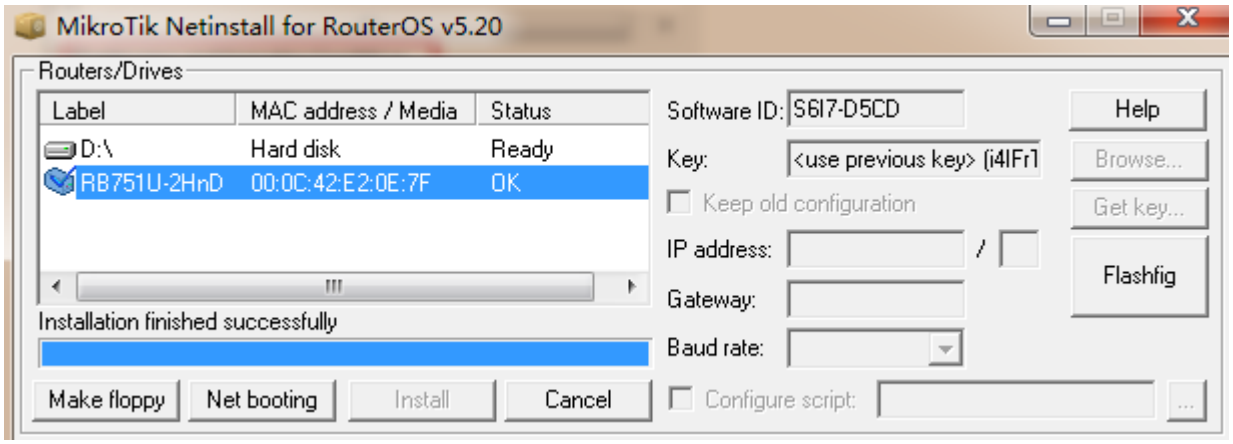
之后，我们点击 install 按钮，首先 netinstall 会对 RouterBOARD 的存储进行分区和格式化



之后传输安装包，进行安装：



安装完成后显示 installation finished successfully



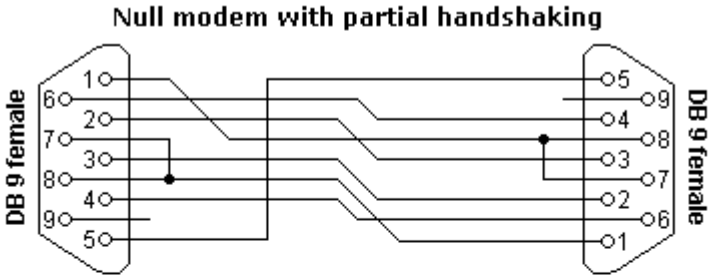
退出 netinstall，RouterBOARD 设备重启，这样复位工作就完成。

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：

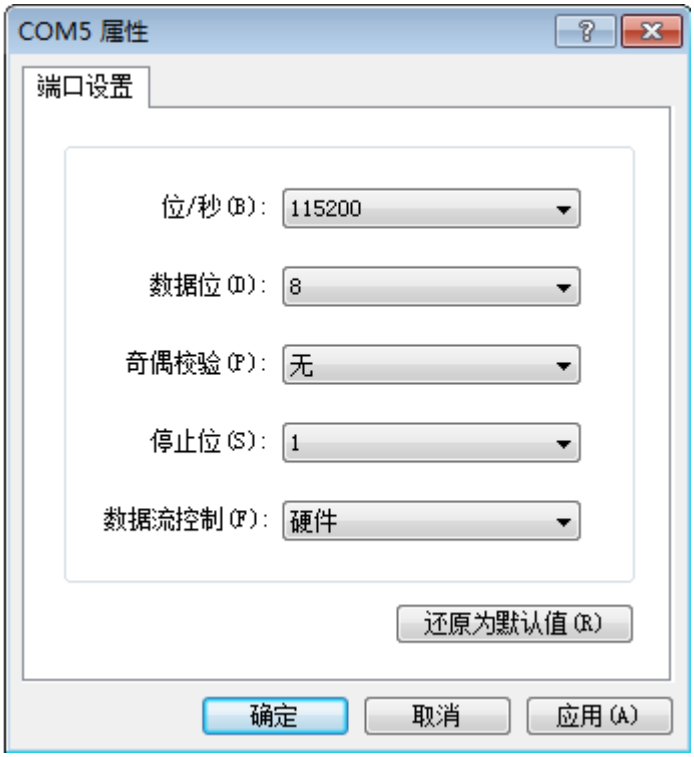


Connector 1	Connector 2	Function
1	7 + 8	RTS_2 → CTS_2 + CD_1
2	3	Rx ← Tx
3	2	Tx → Rx
4	6	DTR → DSR
5	5	Signal ground
6	4	DSR ← DTR
7 + 8	1	RTS_1 → CTS_1 + CD_2

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。

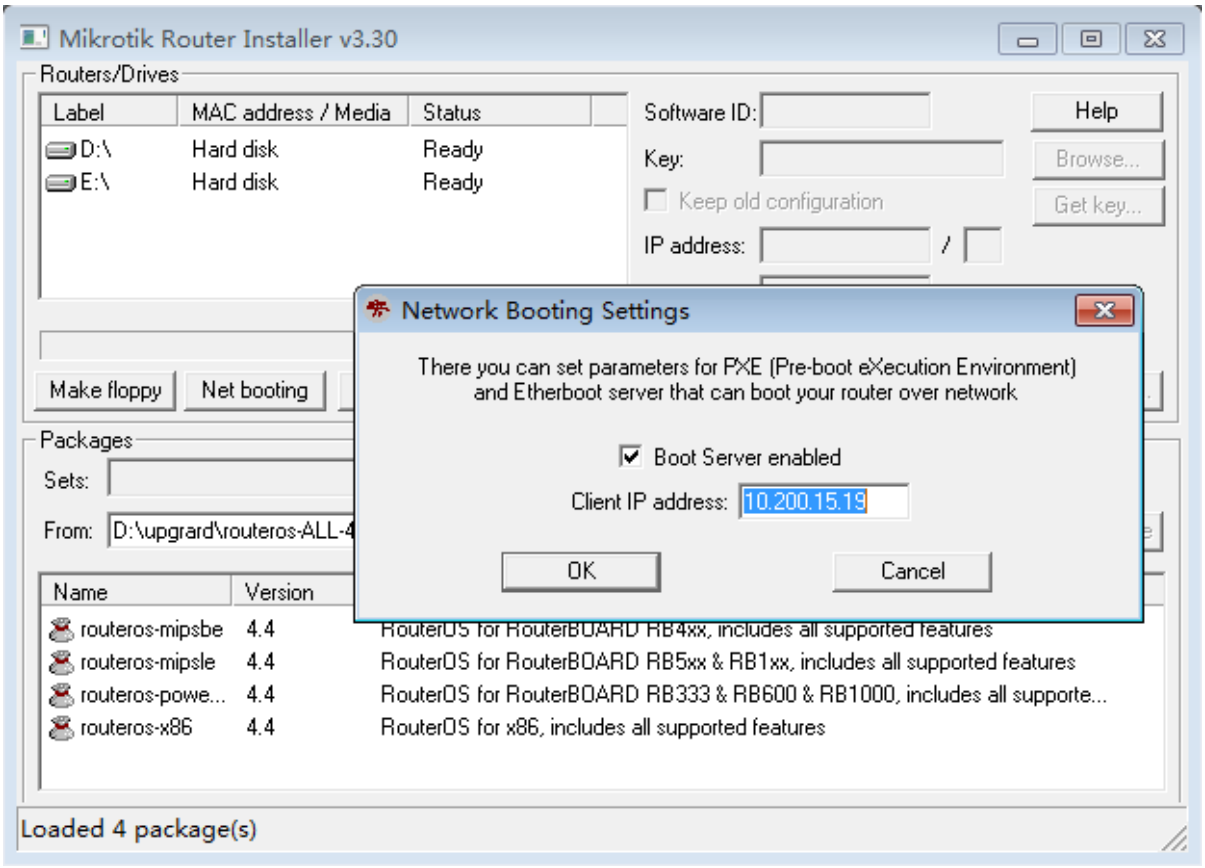
串口控制（管理端）功能允许通过一个串行接口访问路由器的串口终端控台一个特殊的串行接口线通过工作站或者便携式电脑的串口（COM）连接到路由器的串口。在 windows 电脑上常用的串口连接程序是超级终端（HyperTerminal）。

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列表，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:
```

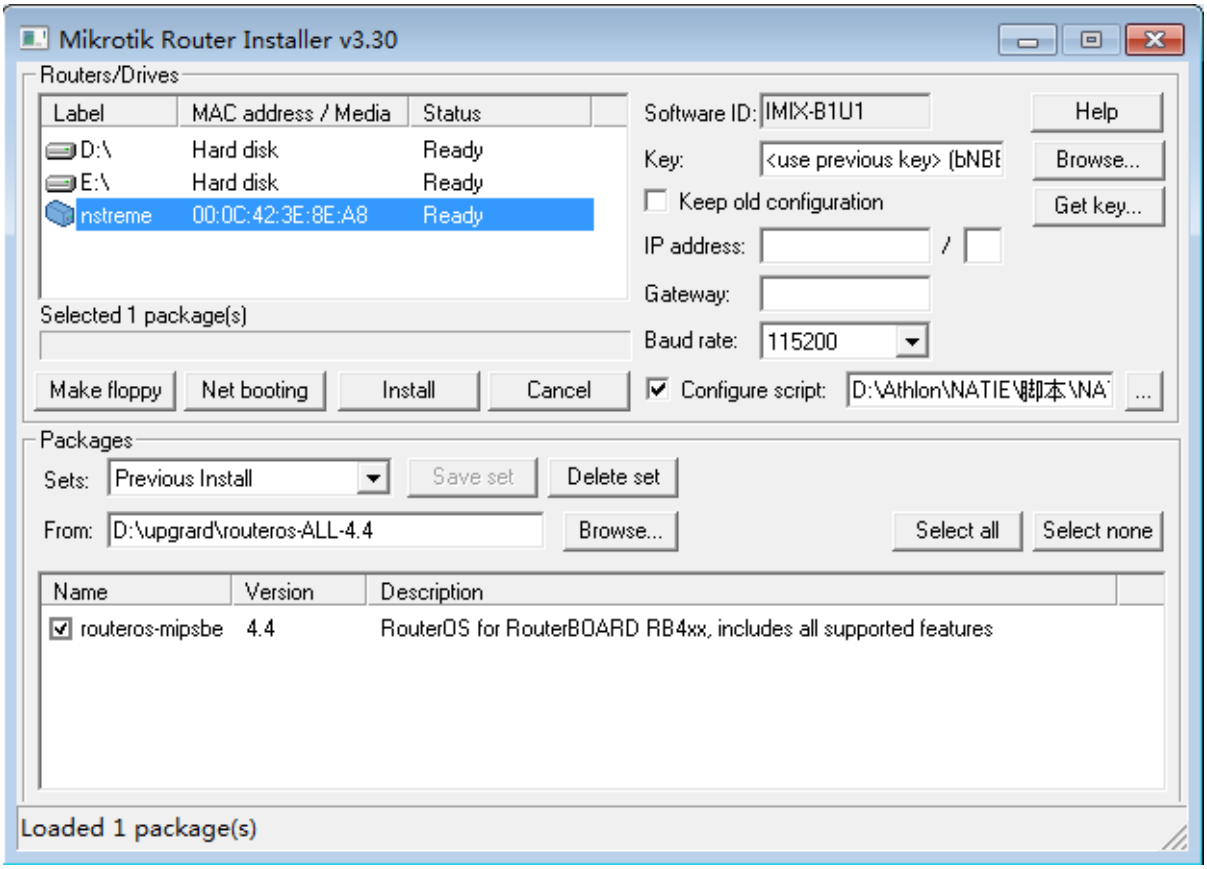
按“e”键，是选择从以太网卡引导 RouterBoard：

```
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。

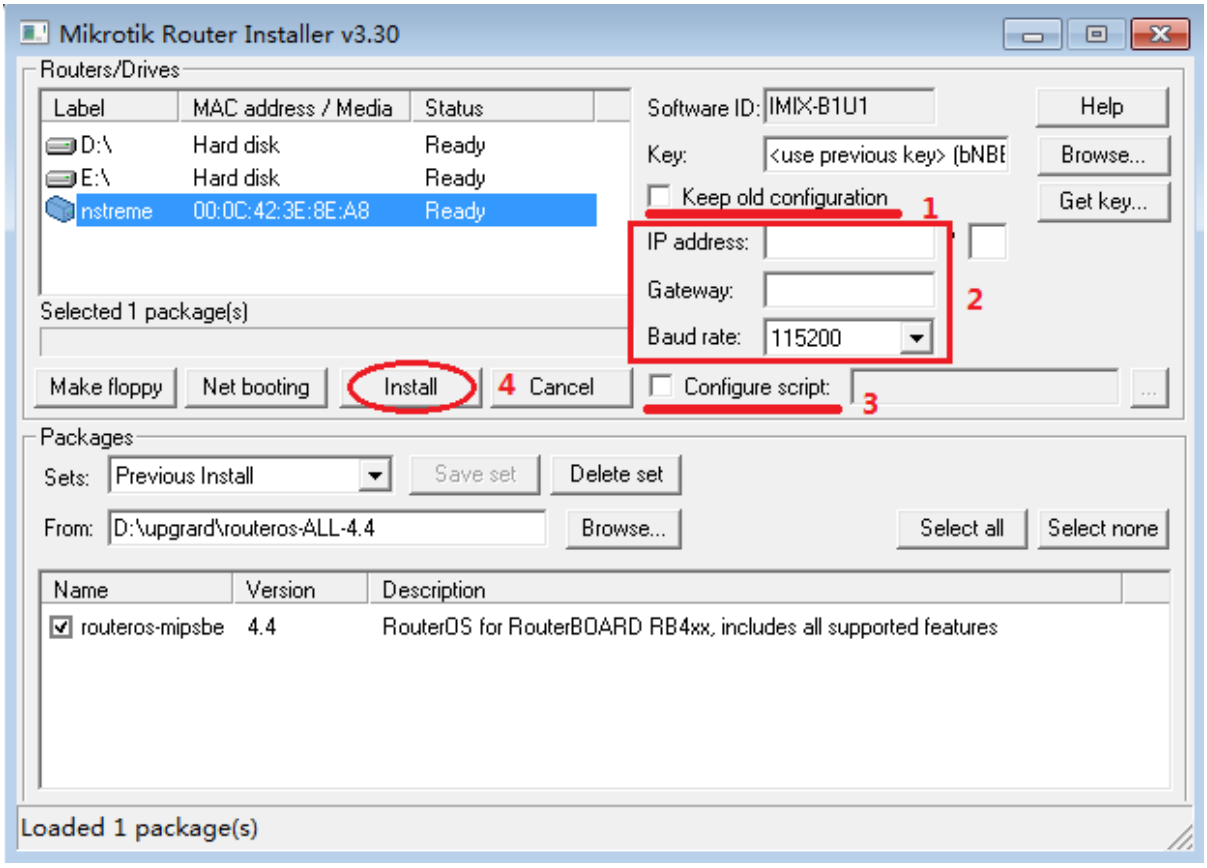


这时 Netinstall 会自动识别 RouterBOARD 的型号，并在指定的目录下查找对应的功能包，Netinstall 自动为 RB450G 查找到了合适的 RB4xx 的安装包，你也可以手动指定安装功能包路径。

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预设置参数：



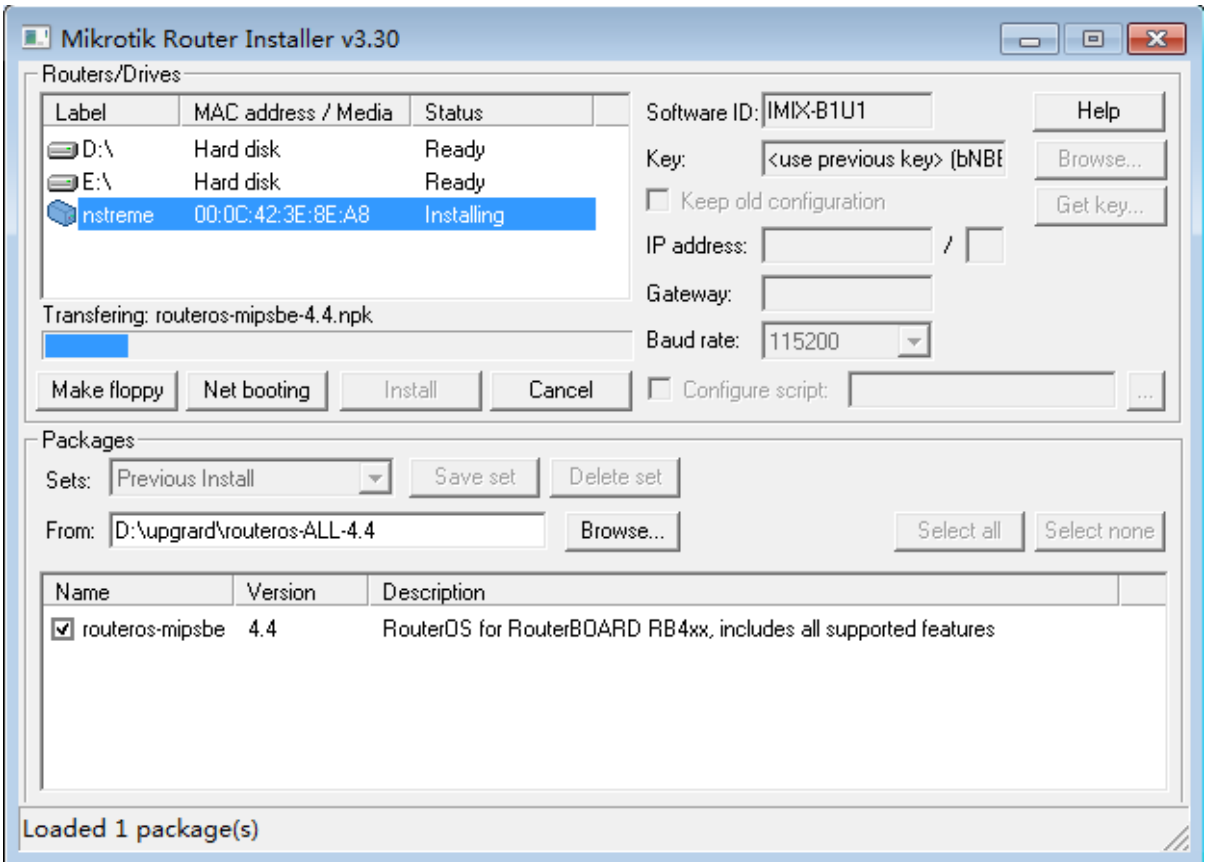
1、Keep old configuration　保留原来的配置不变
2、配置 ip 地址和网关，并设置传输速率采用 115200
3、配置 RouterBOARD 的脚本
4、开始安装

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。

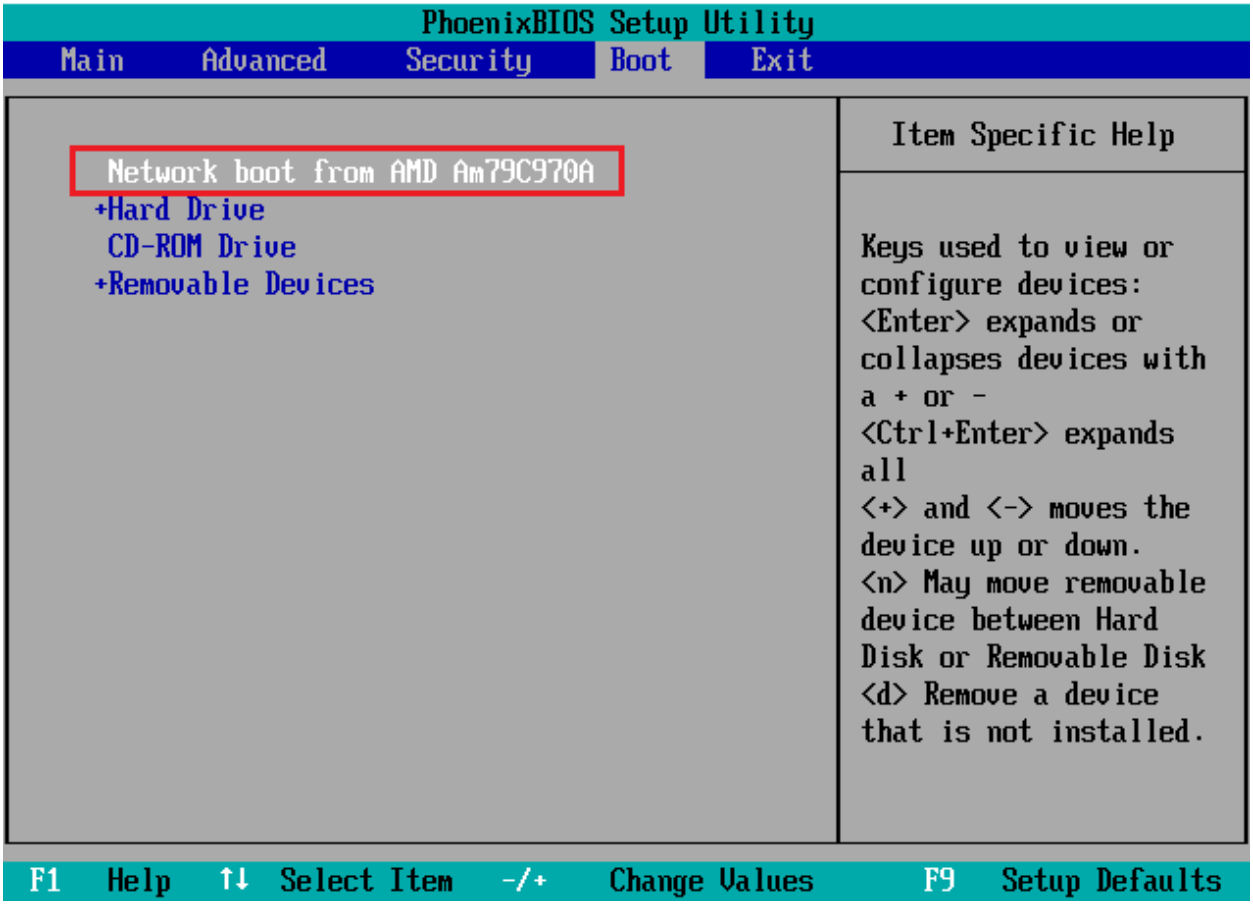
```
Select boot device:
 * e - boot over Ethernet
   n - boot from NAND, if fail then Ethernet
   1 - boot Ethernet once, then NAND
   o - boot from NAND only
   b - boot chosen device
your choice: n - boot from NAND, if fail then Ethernet
```

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。

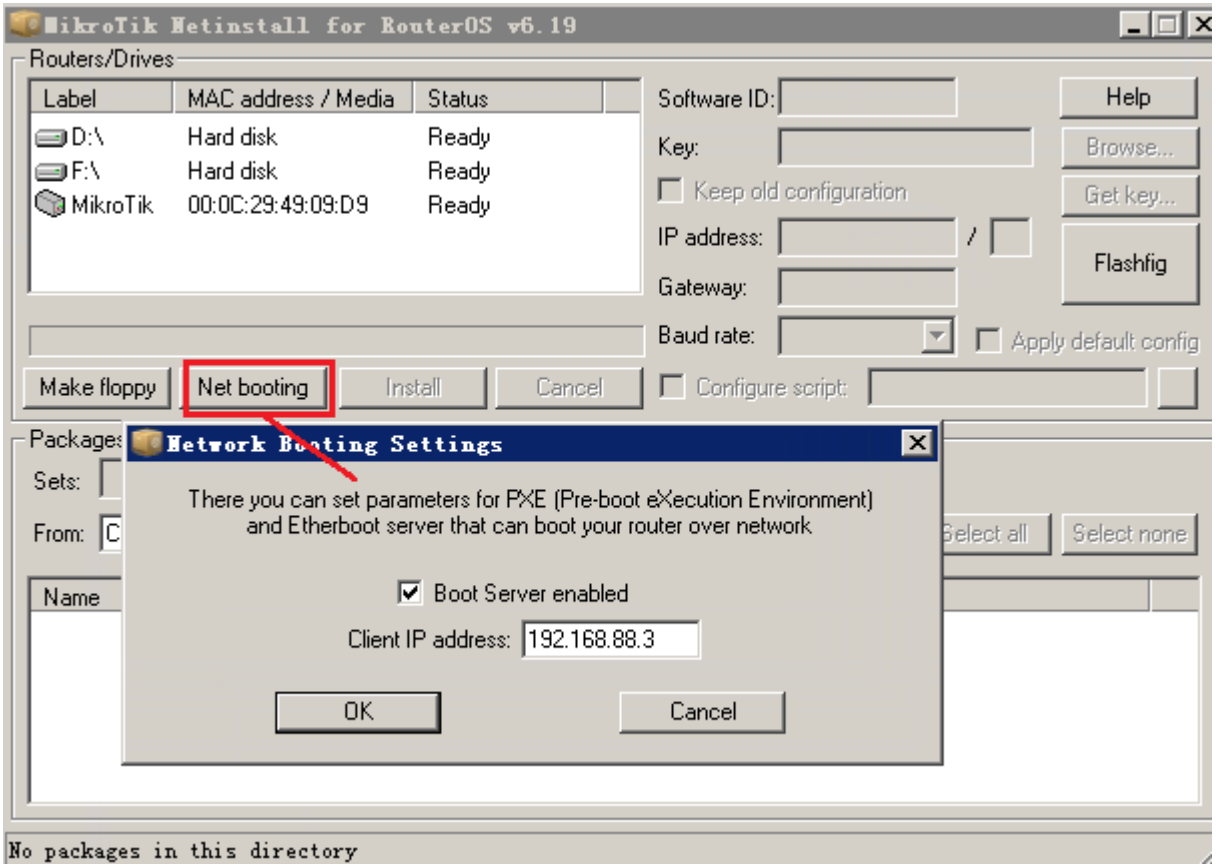
3、PXE 安装 RouterOS 到 x86

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 SATA 接口，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 界面不同），以下界面供参考。



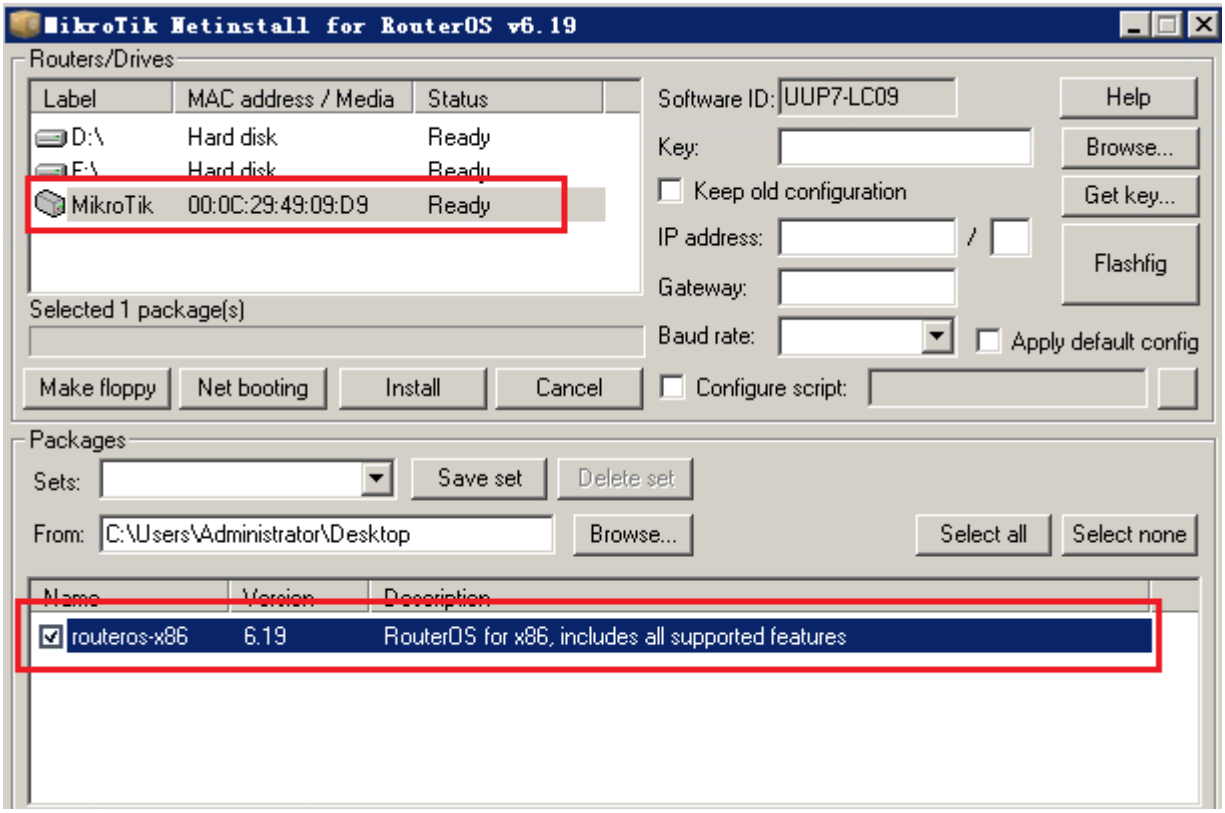
将安装 windows 的笔记本电脑或 PC 主机与安装 RouterOS 连接，并在 windows 系统上准备 netinstall 软件，打开后将 netbooting 设置与本地 ip 地址同一段，windows 电脑的 ip 是 192.168.88.2，分配给被安装主机 ip 是 192.168.88.3。准备好 x86 的安装包 routeros-x86-6.19.npk（可以从 download.mikrotik.com 下载，注意下载的是 for Netinstall），放在与 netinstall 相同目录。



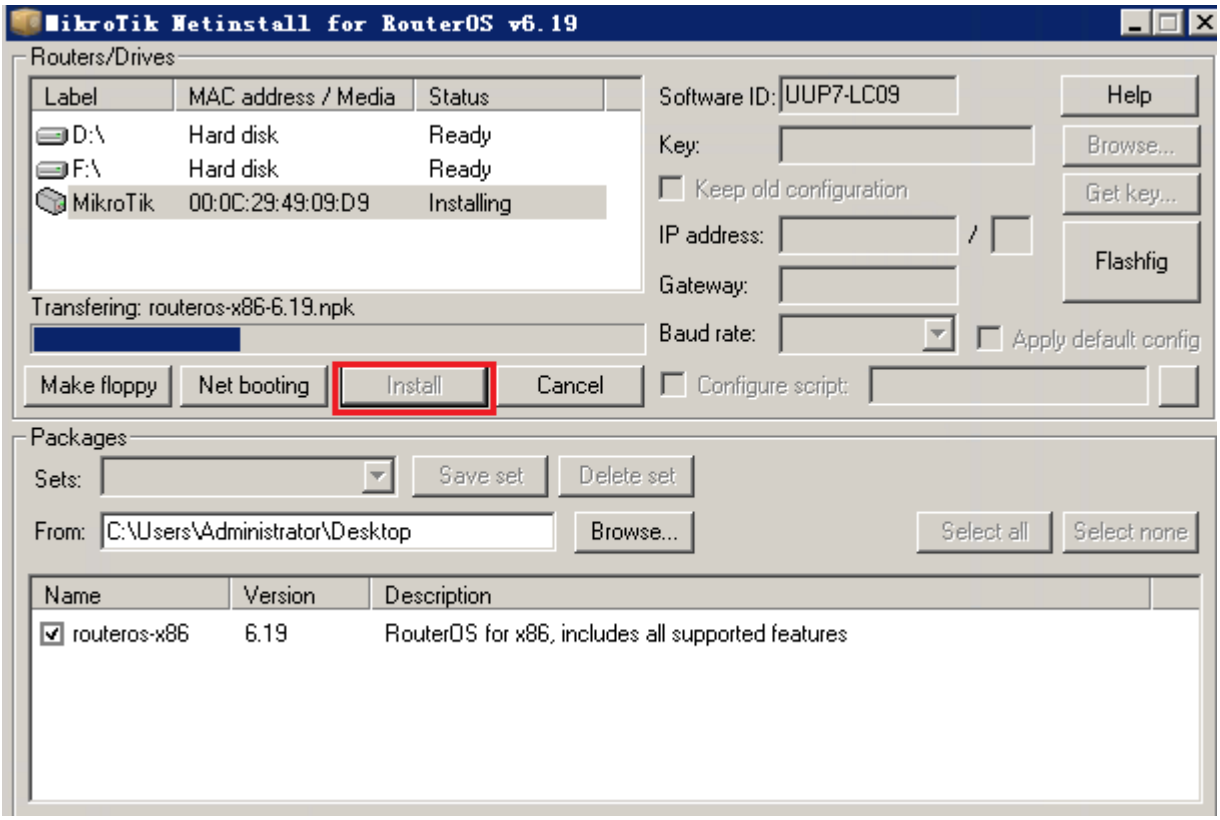
BIOS 和 netinstall 都设置好后，启动主机从网卡的 PXE 引导，如果引导成功进入下面的界面：

```
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 出现被安装主机的 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，找到后选择 6.19 的安装包：



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



1.2 RouterOS 登录方式

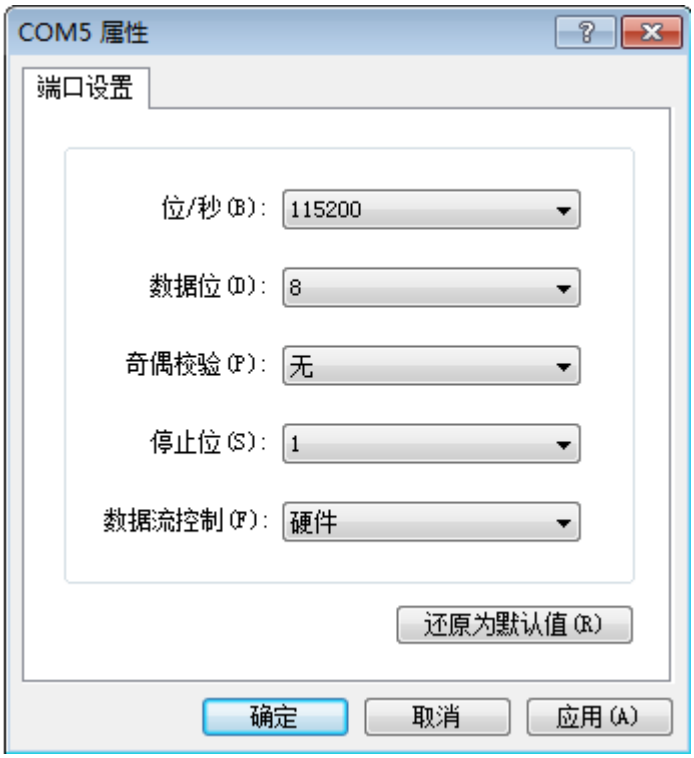
在安装完成 RouterOS 后，准备第一次登陆 RouterOS，RouterOS 初始账号是“**admin**”，密码是“**空**”下面介绍如何通过哪些方式可以连接到 RouterOS。

方式 1 Console 连接

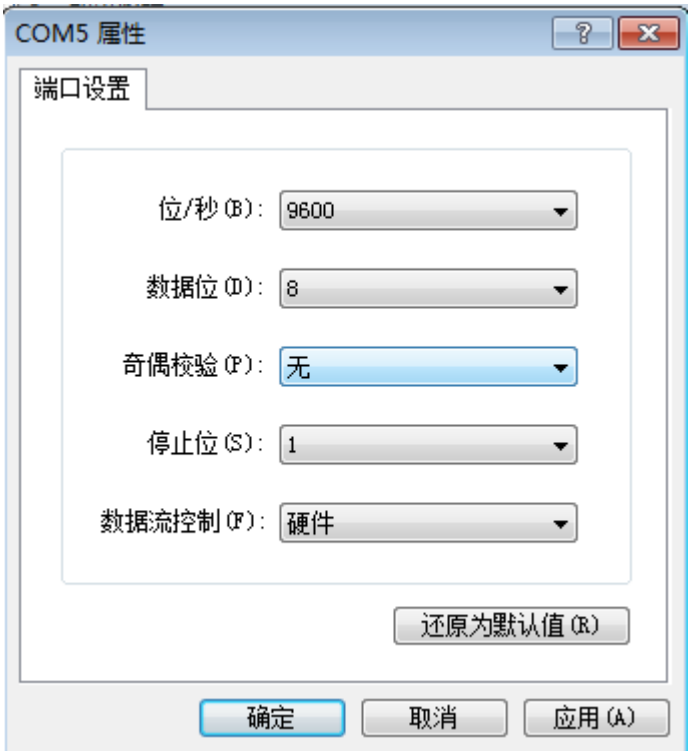
如果你的设备是 RouterBOARD，没有显示器接口连接功能，你必须找到一条 Console 线（普通的 Console 线），或者采用方式 2。

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：

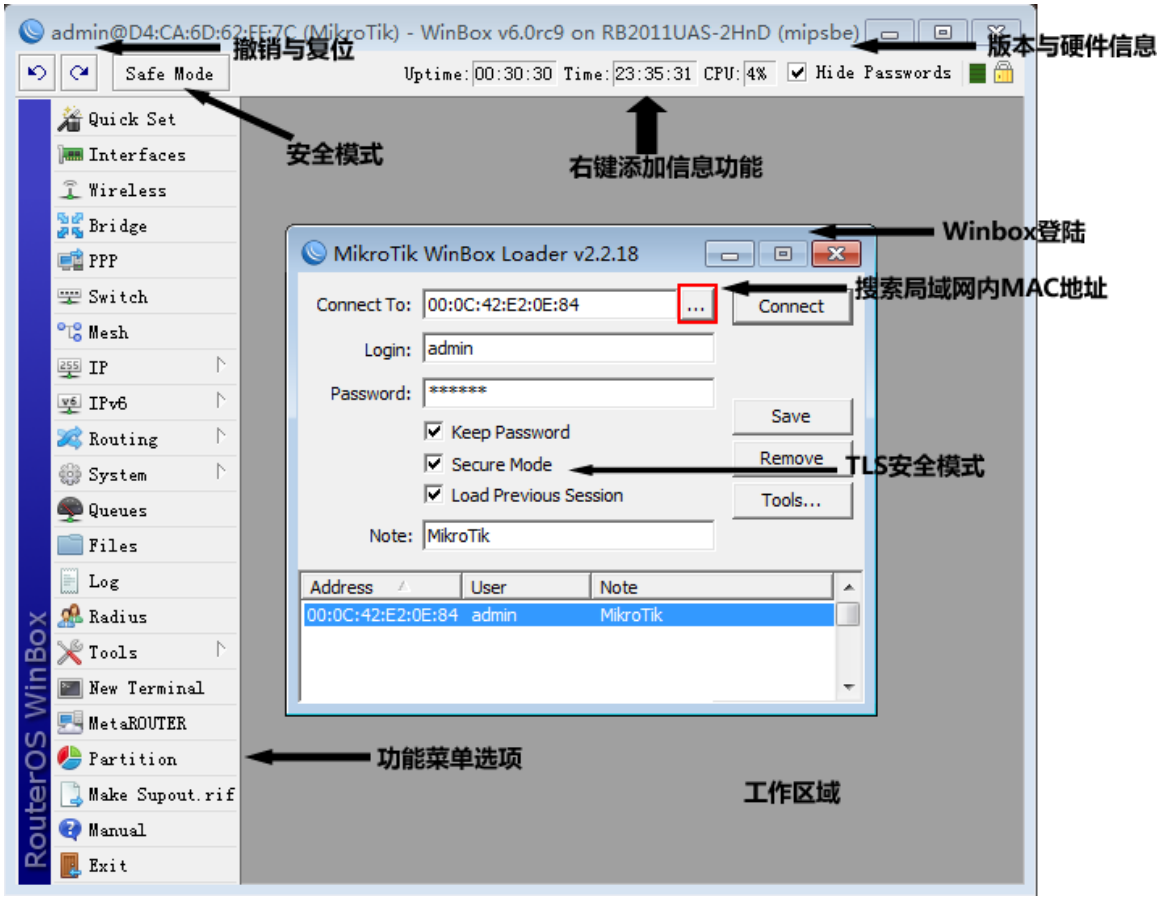
DB9f	功能	DB9f	DB25f
1+4+6	CD+DTR+DSR	1+4+6	6+8+20
2	RxD	3	2
3	xD	2	3
5	GND	5	7
7+8	RTS+CTS	7+8	4+5

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

你可以下载 winbox 应用程序，通过网上搜索可以找到，或者链接 WinBox。下载完成后你要确定你的电脑与路由器已通过以太网线连接，或者他们两个连接在同一局域网内的交换机上。

运行 winbox，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



这个方法适用于任何 RouterOS 设备，注意你的 PC 网卡的 MTU 值必须是 1500

方式 3 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 4 MAC 层访问（Telnet 与 Winbox)

通过 MAC 地址进行连接是用来访问没有设置 IP 地址的 RouterOS 路由设备．这种连接类似于 IP 地址连接．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。Disabled (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** interface 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

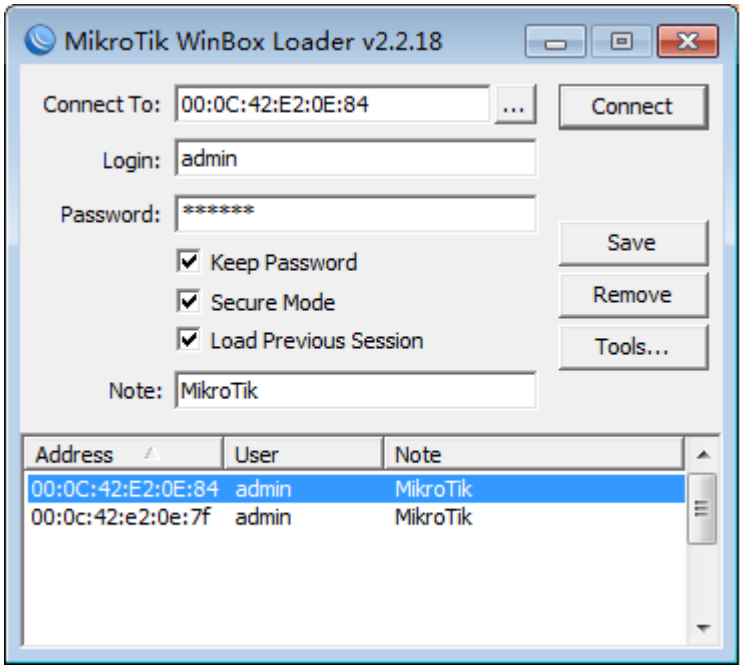
通过 MAC 地址登陆同一局域网的 RouterOS：

```
[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 内能通过远程配置各种参数，包括 **Telnet**, **SSH**, **WinBox** 和 **Webbox**。在这里我们将着重介绍怎样使用 **Winbox**，Winbox 是 MikroTik RouterOS 的远程图形管理接口（GUI）：



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，进行图形界面操作。

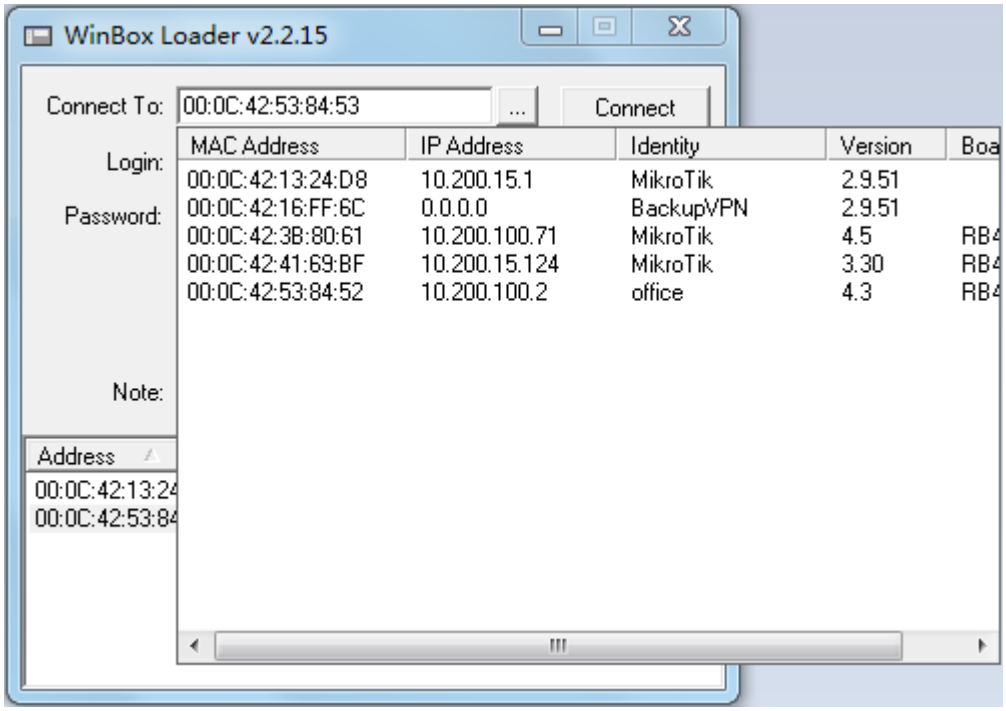
注：在 winbox2.2.12 后增加了可选择的 MAC 登陆或者 IP 登陆的功能，同时需要提醒大家当打开迅雷下载软件后，会占用 MAC 扫描的 UDP 端口，所以建议使用 winbox 的 MAC 登陆时，关闭掉迅雷下载。

通过连接到 RouterOS 路由器的 HTTP（TCP 80 端口）欢迎界面下载 Winbox.exe 可执行文件，也可以登陆 www.mikrotik.com 网站去下载 ，下载后保存在你的电脑中，之后直接在你 Windows 电脑上运行 Winbox.exe 软件，无需安装。

下面是对相应的功能键做介绍:

-

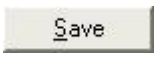
  搜索和显示 MNDP (MikroTik Neighbor Discovery Protocol) 或 CDP (Cisco Discovery Protocol) 设备。可以通过该功能键搜索同一子网内 MikroTik 和 Cisco 设备。并能通过 MAC 地址登陆到 MikroTik RouterOS 进行操作。



  注：在 winbox2.2.12 后的版本增加了 MAC 地址和 IP 地址选择功能，可根据搜索内容选择使用 MAC 地址连接或是 IP 地址连接。

-

  通过指定的 IP 地址（默认端口为 80，不许特别指定，如果你修改了端口需要对具体访问端口做自定）或 MAC 地址（如果路由器在同一子网内）登陆路由器。

- 

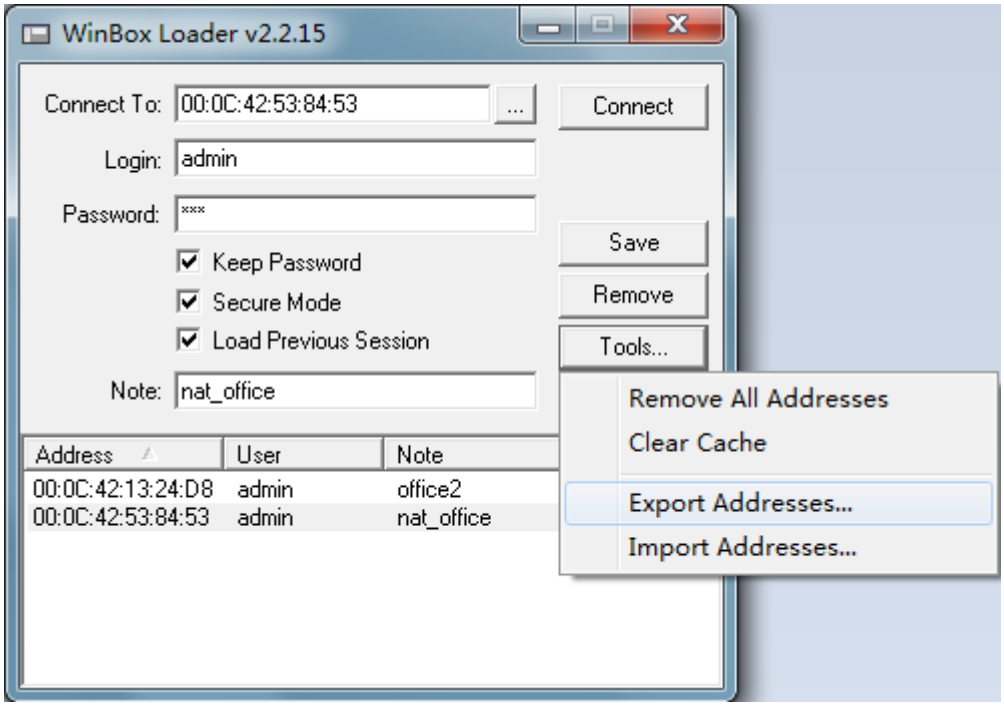
  保存当前连接列表（当需要运行它们时，只需双击）

-

  删除从列表中选择的项目

- Tools...

  删除所有列表中的项目，清除在本地的缓存，从 wbx 文件导入地址或导出为 wbx 文件



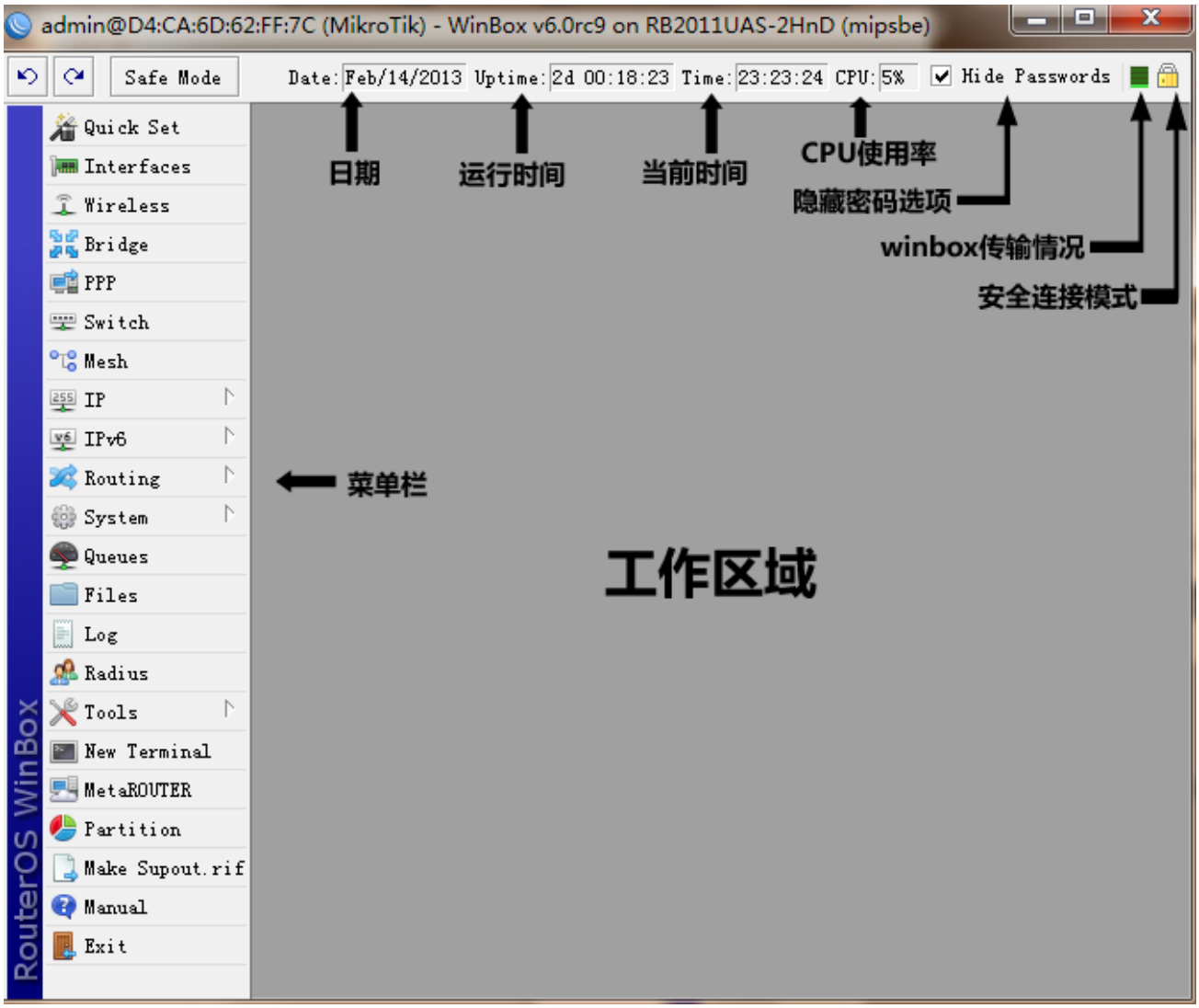
- Secure Mode（安全模式）：提供保密并在 winbox 和 RouterOS 之间使用 TLS（Transport Layer Security）协议
- Keep Password（保存密码）：保存密码到本地磁盘的文本文件中

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台同样的任务。

接口概述

Winbox 接口被设计为直观的界面，接口包括以下：

- 工具栏在顶部用户可以添加一些工具，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：

```
[用户名]@[Router's IP 或者 MAC] ( [RouterID] ) - Winbox [RouterOS 版本] on [RB 型号] ([平台])
```

从截图上我们能看到用户通过账号 **admin** 登陆，连接 IP 地址 **10.1.101.18**，路由器的身份 ID 是 **MikroTik**，当前安装 RouterOS 版本是 **v5.0beta1**，RouterBOARD 型号 **RB800**，平台是 **PowerPC**

在左边工具栏有 **undo** 和 **redo** 按钮，快速撤销和恢复操作

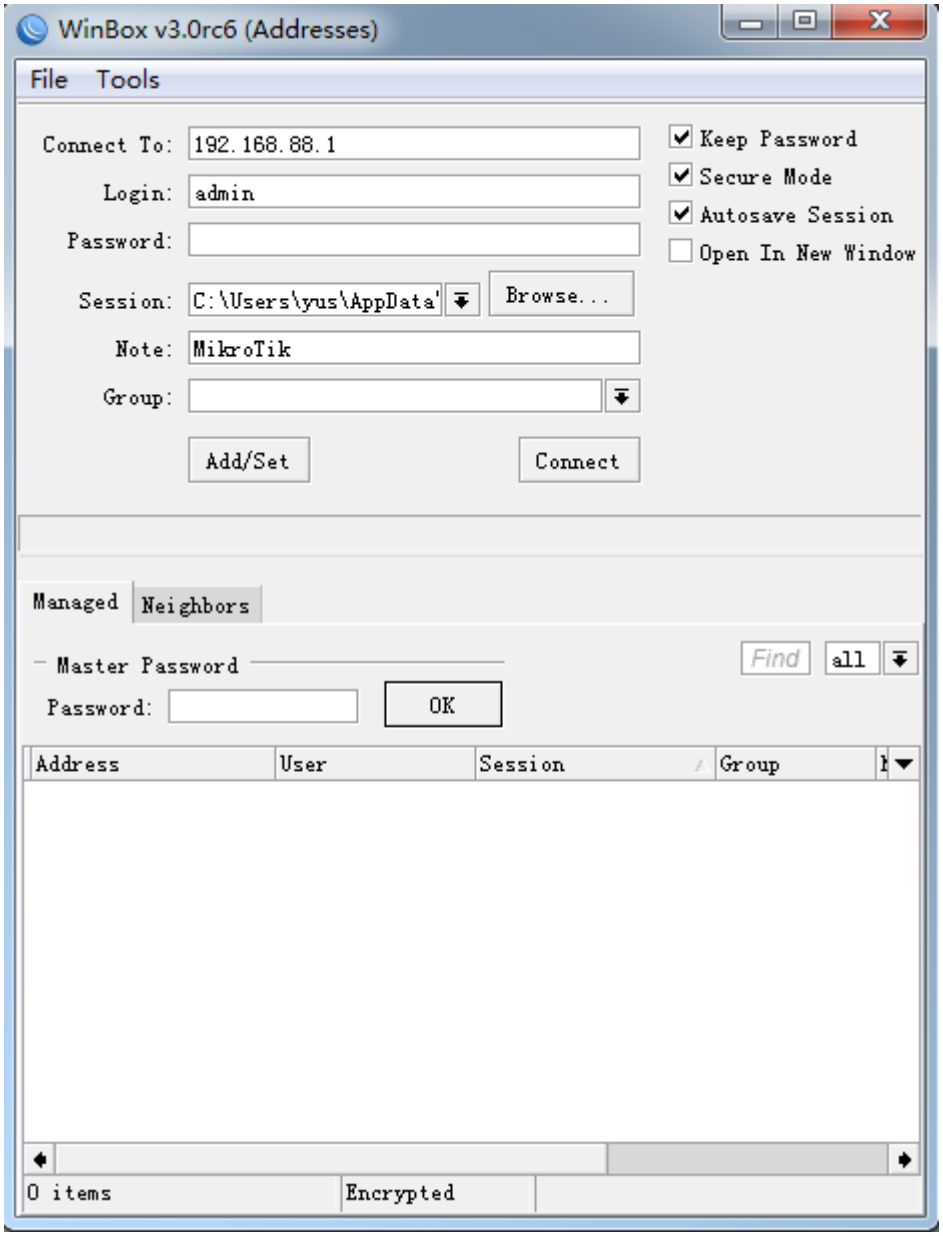
右边工具栏：

- Winbox 传输指示显示一个绿色栏
- 指示 winbox 连接使用 TLS 加密
- 复选框 **Hide password**.这个复选框替换所有敏感信息为“*”（如：各种密码，PPP secret Passwords）

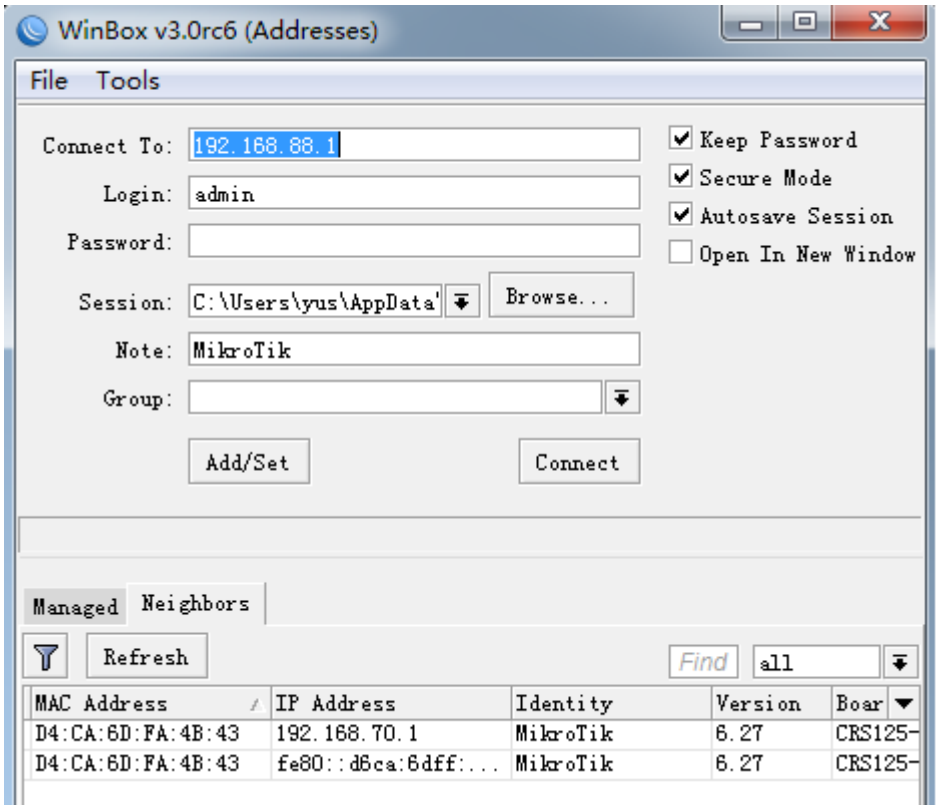
Winbox v3 版本

MikroTik 对 winbox 版本也在不断更新，winbox v3 做出了相应的改动，主要集中在安全和使用功能上，新的 winbox 3 如下特点：

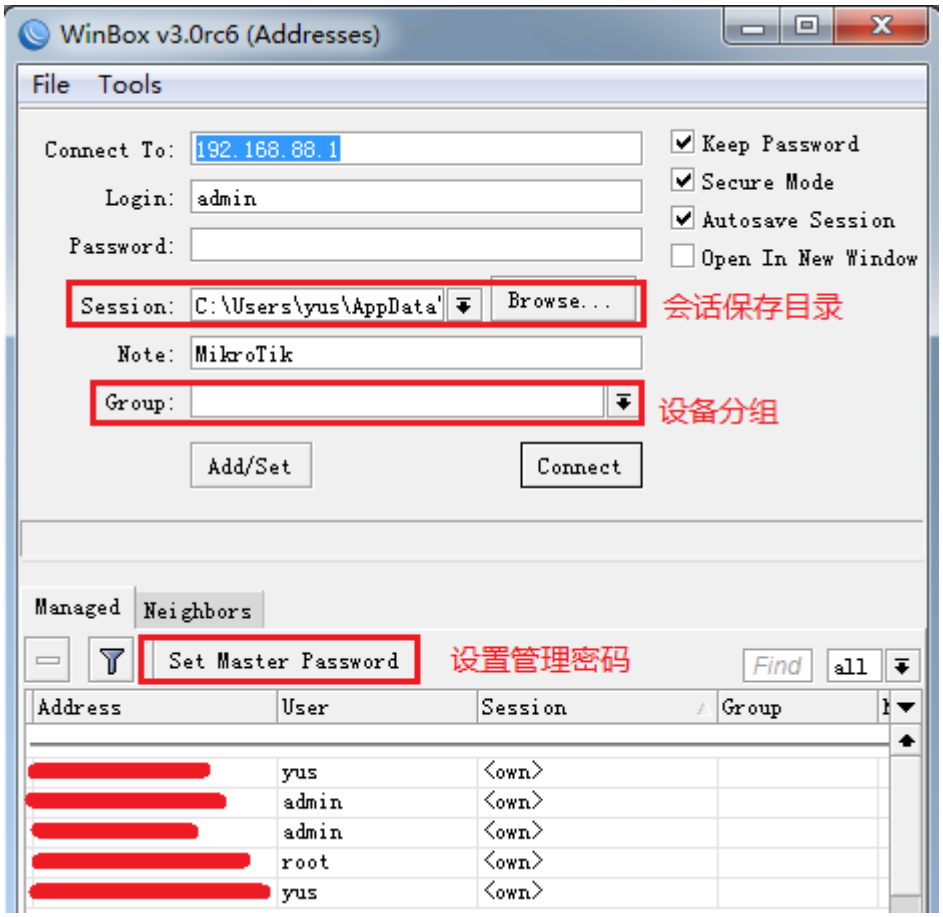
- 增加了组功能，可对各类 RouterOS 设备分组管理，增大了设备保存列表条目
- 在邻居发现功能中能对 IP 地址、MAC、RouterOS 版本分类
- 能过滤搜索，具备邻居（Neighbors）发现功能
- Winbox 能自动升级新版本
- 新增了当连接丢失，提供重新连接功能，并能保存连接会话
- 增加了设备管理密码验证功能



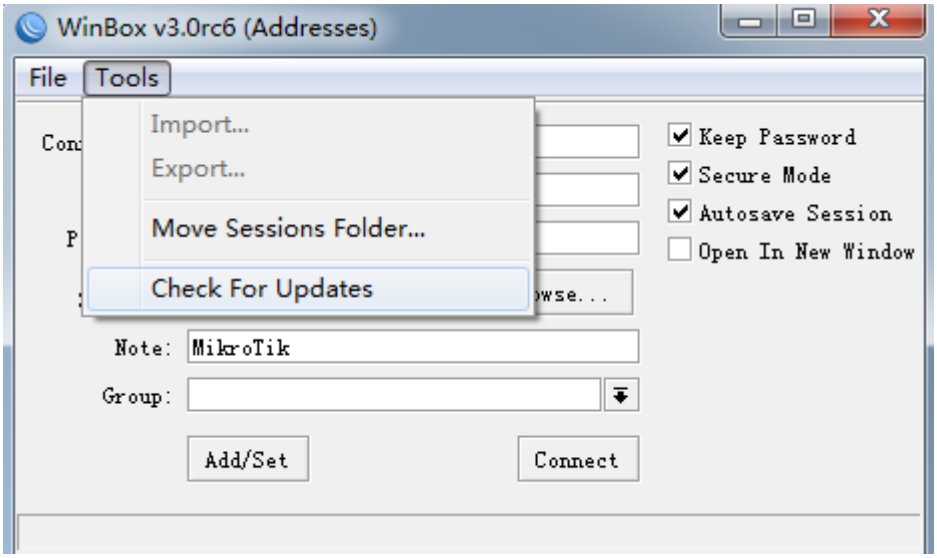
在 winbox v3 版本将 按钮键取消，代替这个功能的是 Neighbors 邻居发现列表，当点击开 Neighbors 列表后，会自动搜索当前局域网内的 MikroTik 设备，如下图：



增加了连接 Session 保存目录和 Group 设备分组选项，并添加了设备分组管理后的密码验证管理

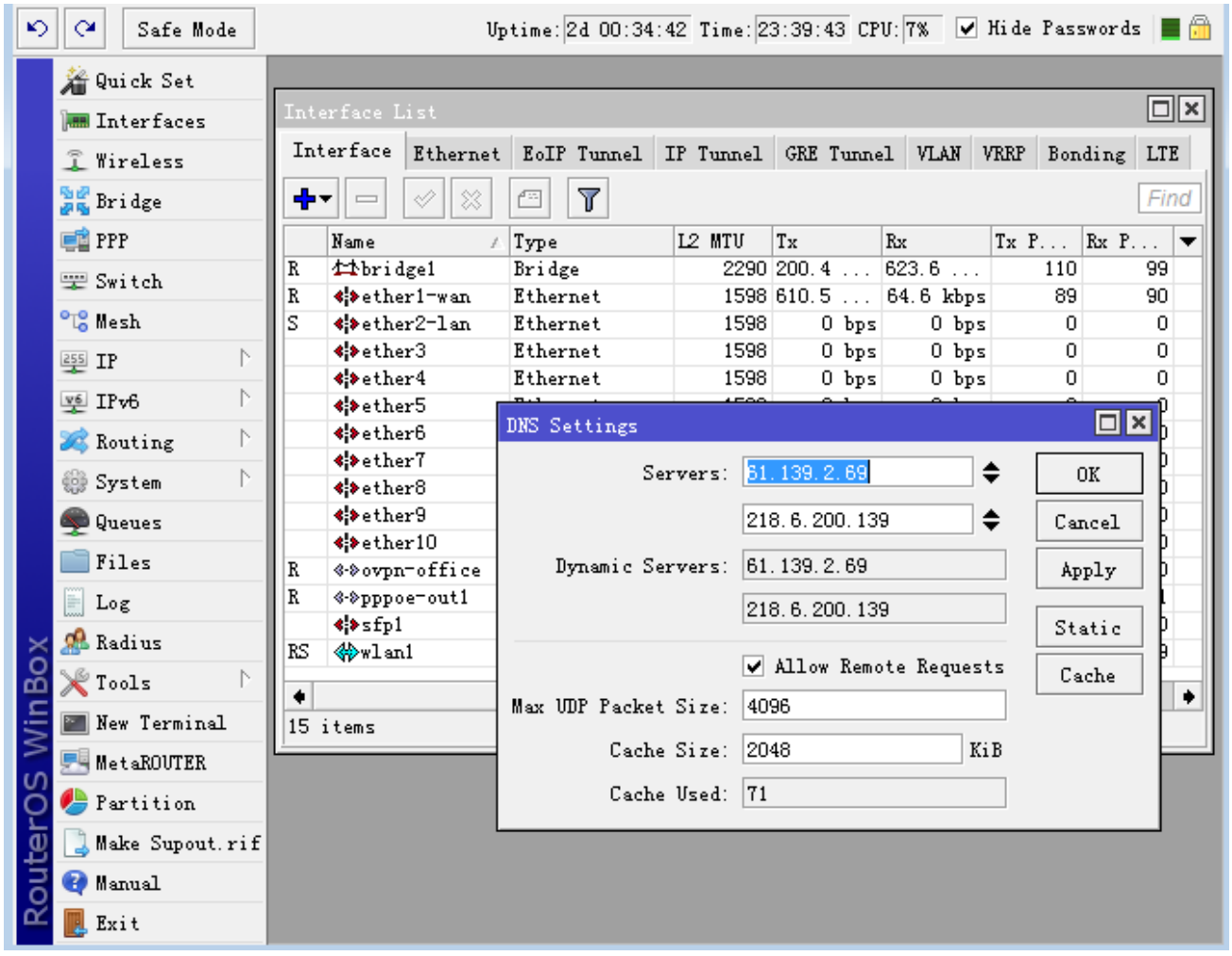


打开 Tools 菜单，可以找到 Check For Updates 选项，可以通过连接 MikroTik 服务器，检查版本更新情况：



工作区域和子窗口

Winbox 采用 MDI 接口，即所有菜单配置（子窗口）被从属于主窗口下（父窗口）,子窗口不能被拖出工作区域



子窗口菜单栏

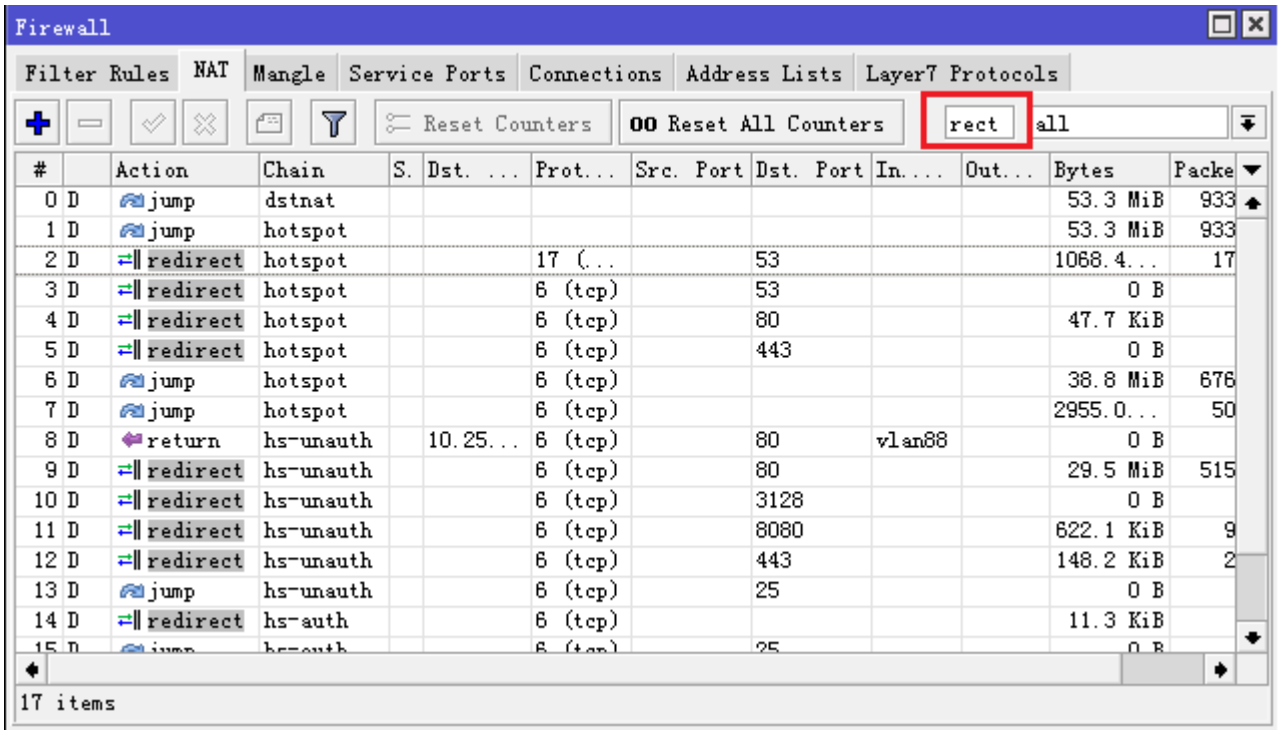
每个子窗口有自己的工具栏，大多窗口都有相同的工具栏按钮：

图标	功能		图标	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			查询关键字
	启用一个项目			撤销操作
	禁用一条项目			恢复操作

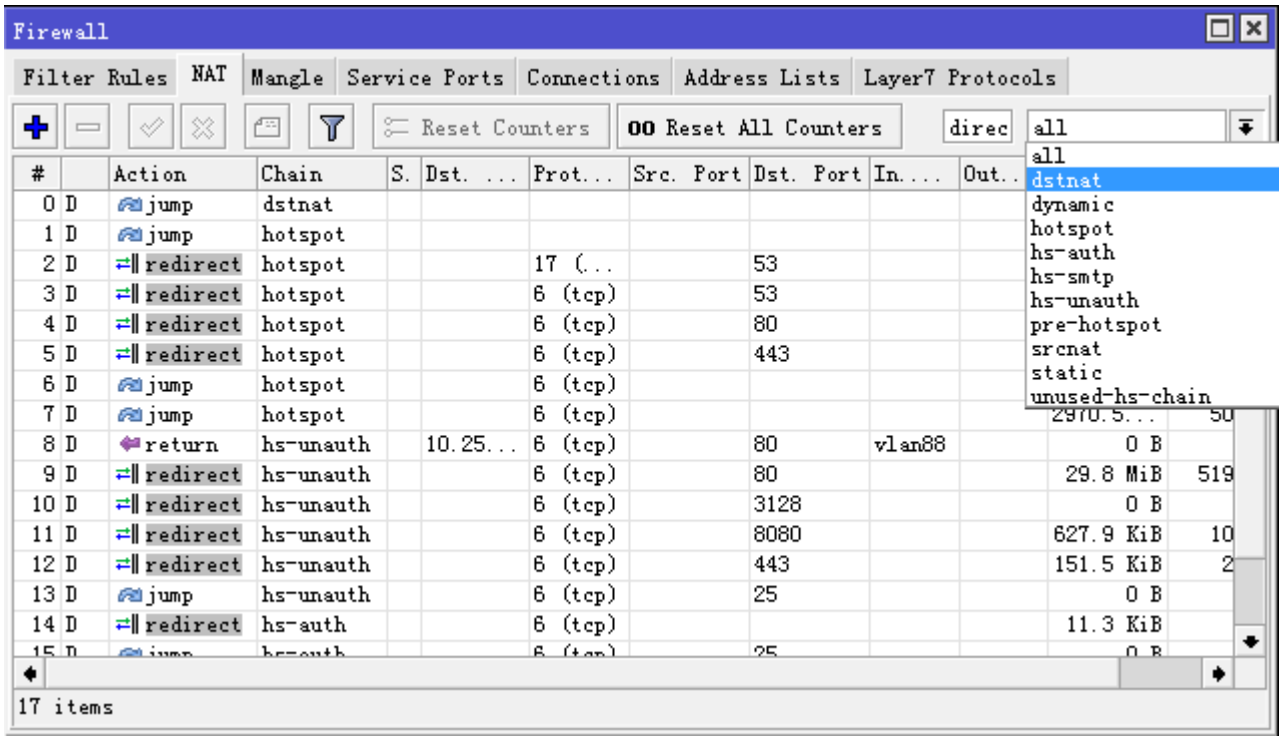
Winbox 管理菜单栏：

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



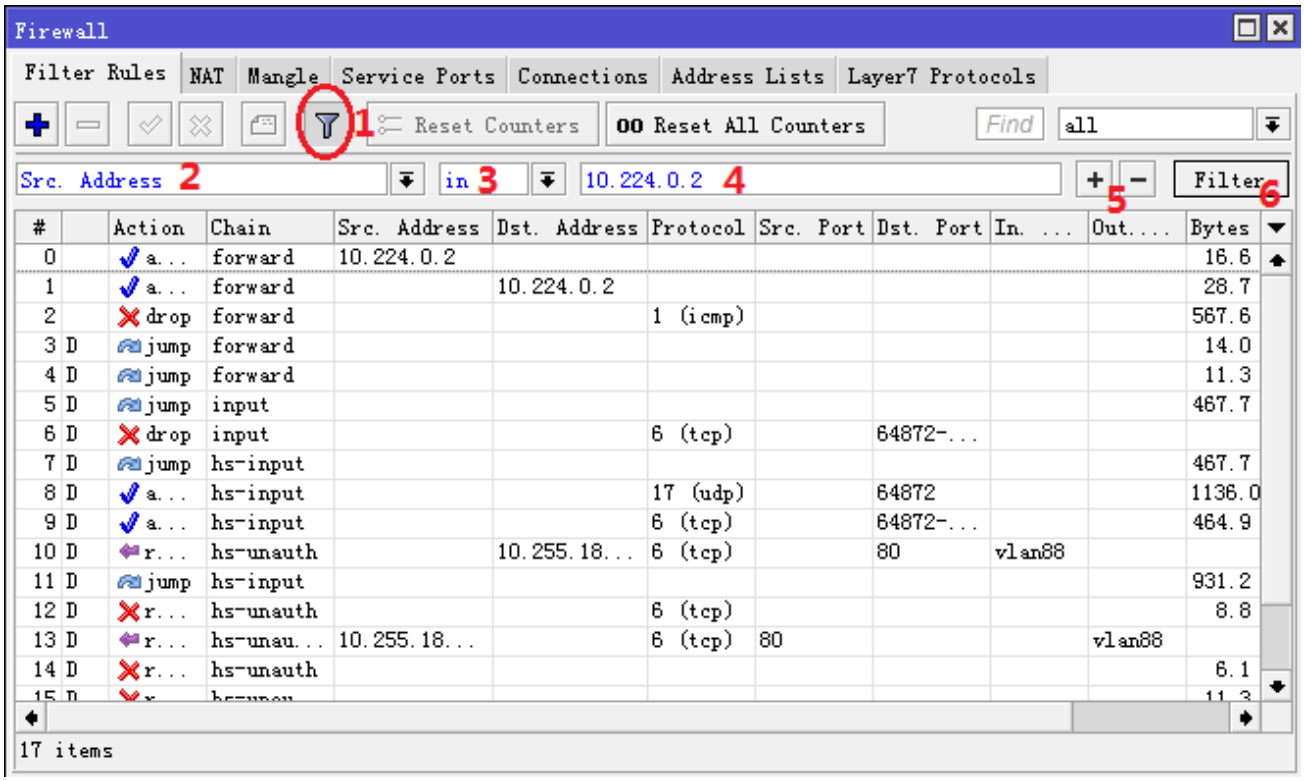
注： 上面是在 nat 中的快速查询框，注意的右侧由于个下拉框，这个是可以调出配置的路由表，例如： dstnat 被选中，这时仅 dstnat 的列表中查询



类似的下拉框在 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



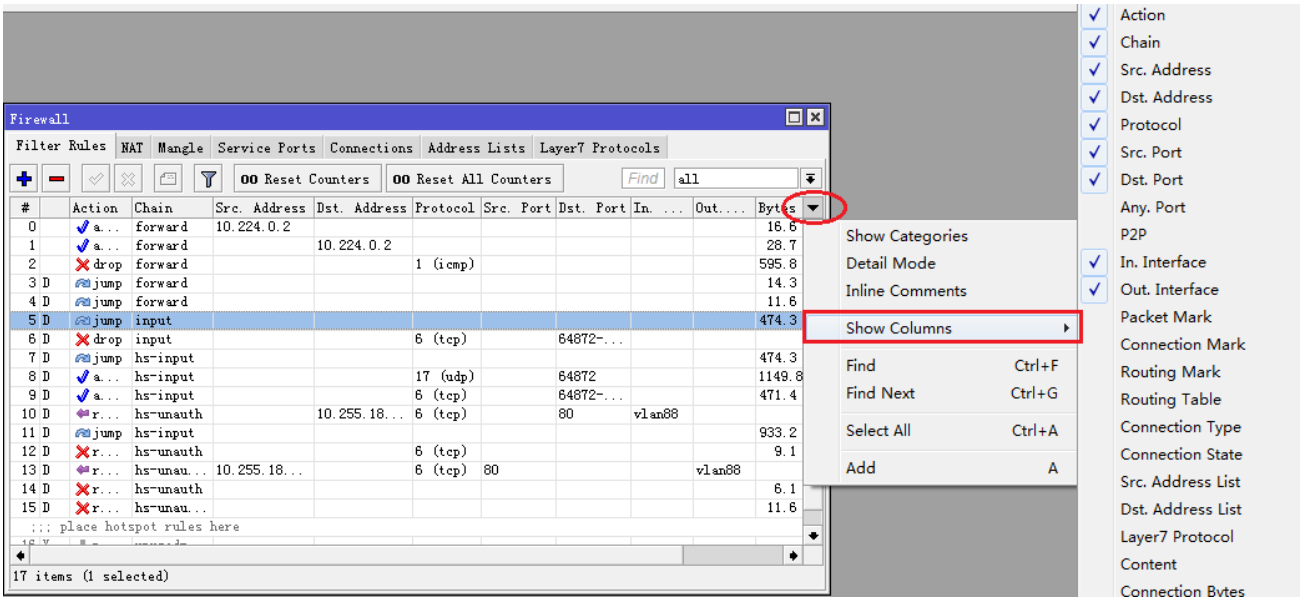
显示如何快速过滤路由表中地址范围是 10.0.0.0/8

1. 按 Filter 按钮
2. 从第一下拉框选择 **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

如同你从截图上看到筛选出路由表里范围为 10.0.0.0/8 的目标地址

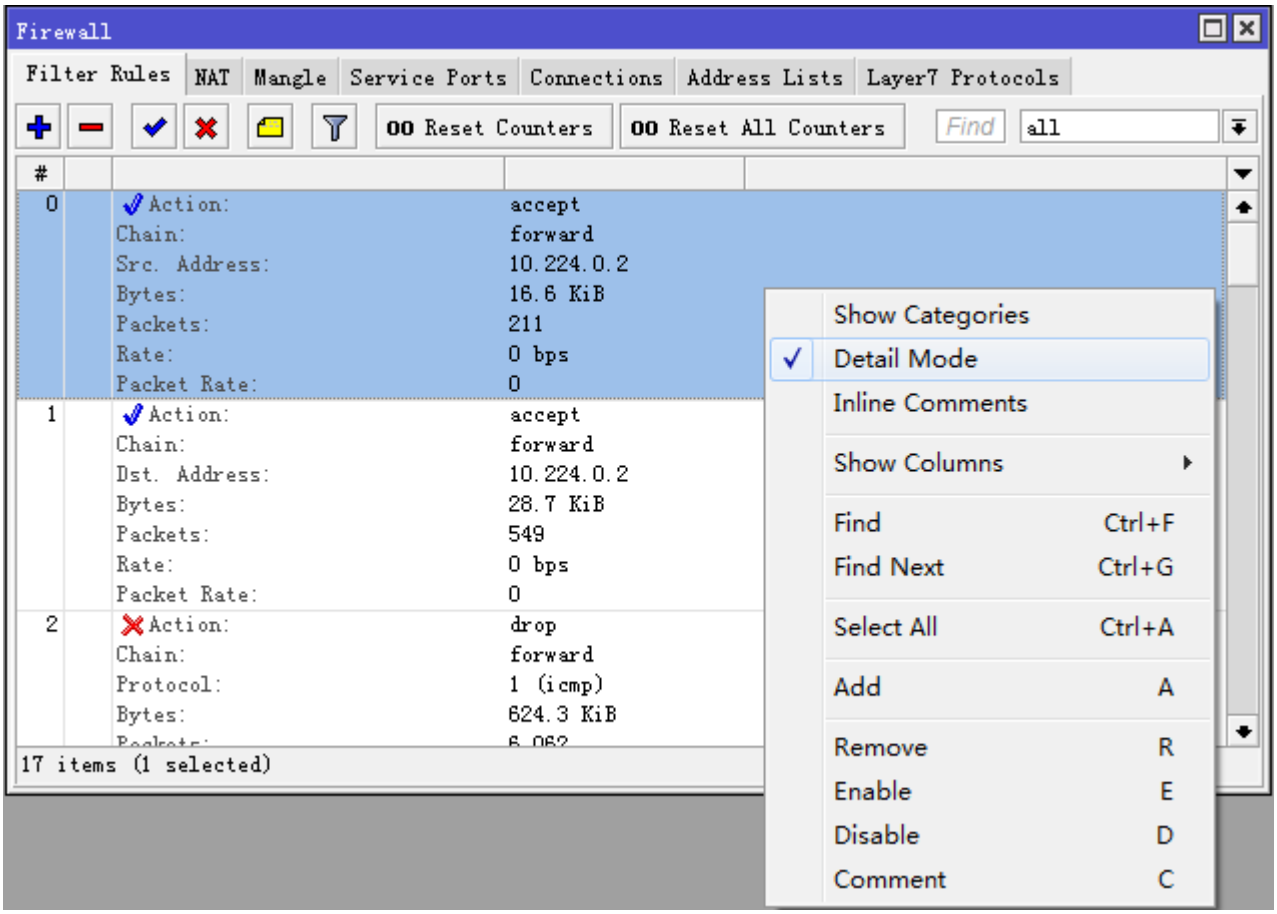
自定义显示列表

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



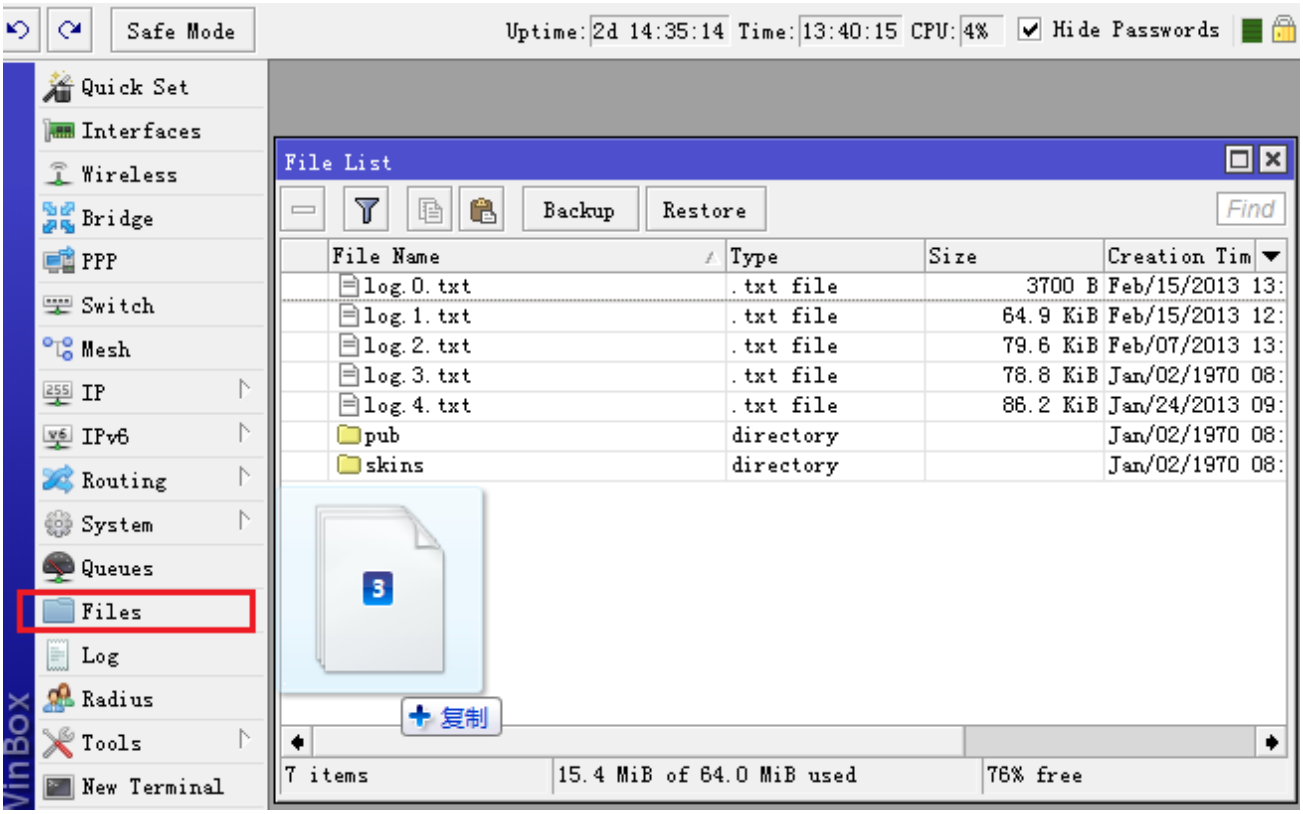
详细模式

在 winbox 中我们可以启用详细模式，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

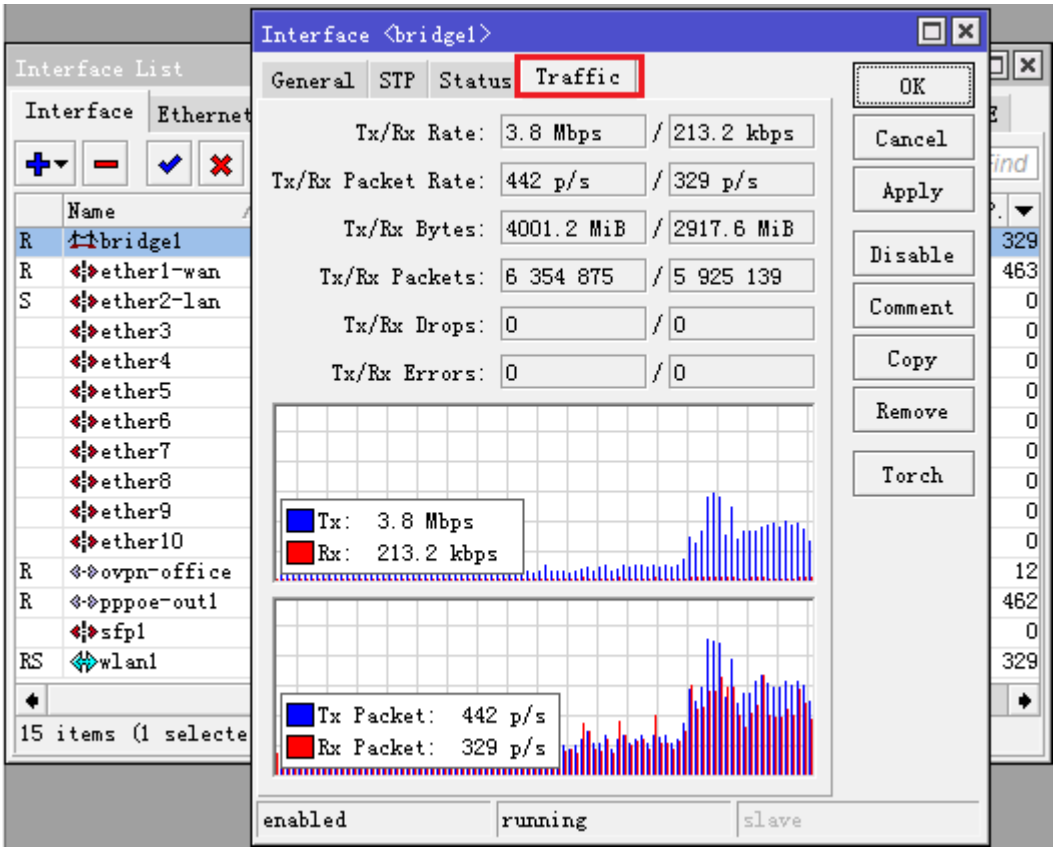
Winbox 可以上传和下载文件，我们可以使用 winbox 拖&放功能实现，我们可以从 windows 电脑上将文件拖&放到 File 列表下，在 5.0 后可以随意拖放到 winbox 中的任意位置，即可上传文件。也可以从 file 列表下拖出文件。



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。

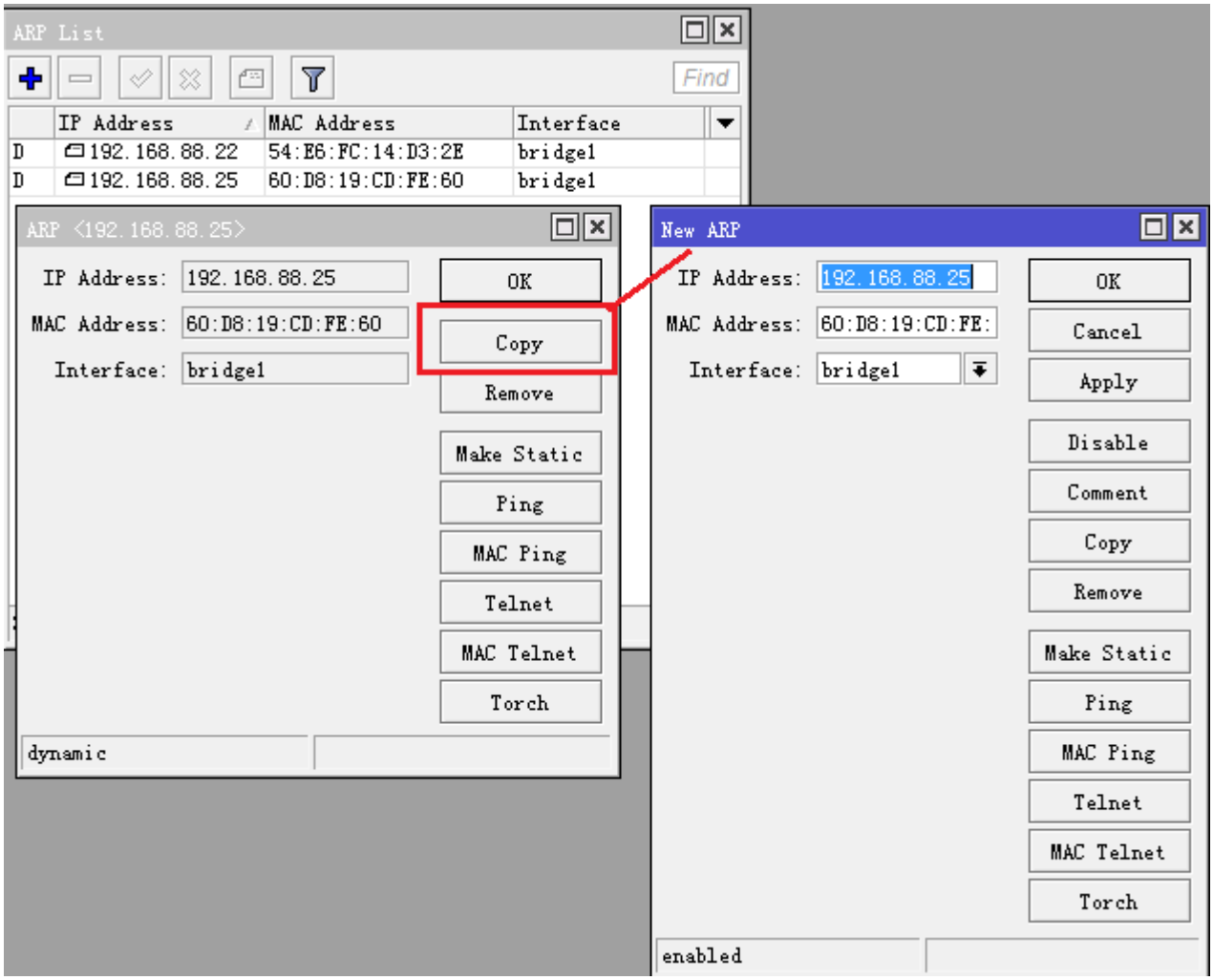
流量监测

Winbox 能使用一个工具流量监测每个网口，以及 Queue 和 Firewall 每个规则的实时流量监测截图下显示了以太网口的流量监测图

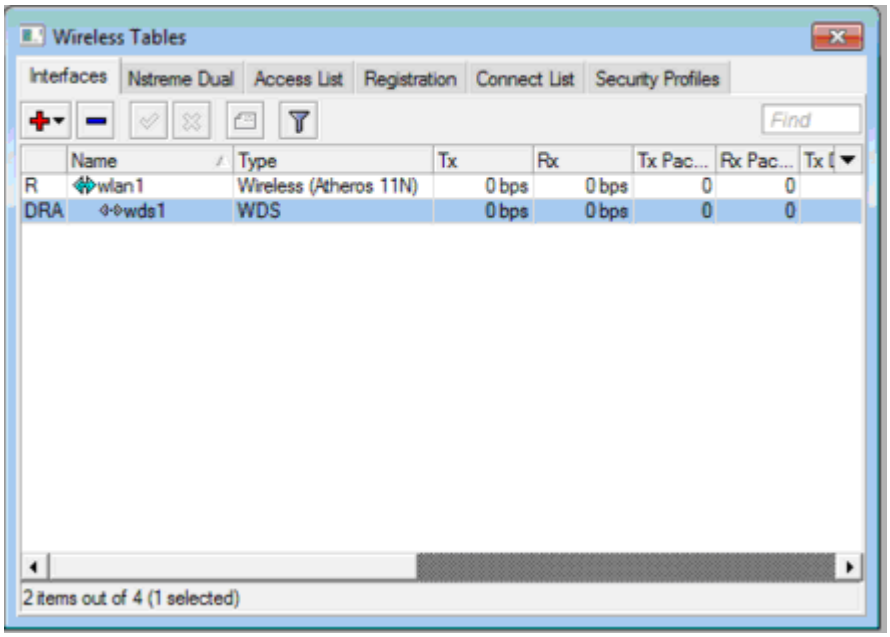


项目复制

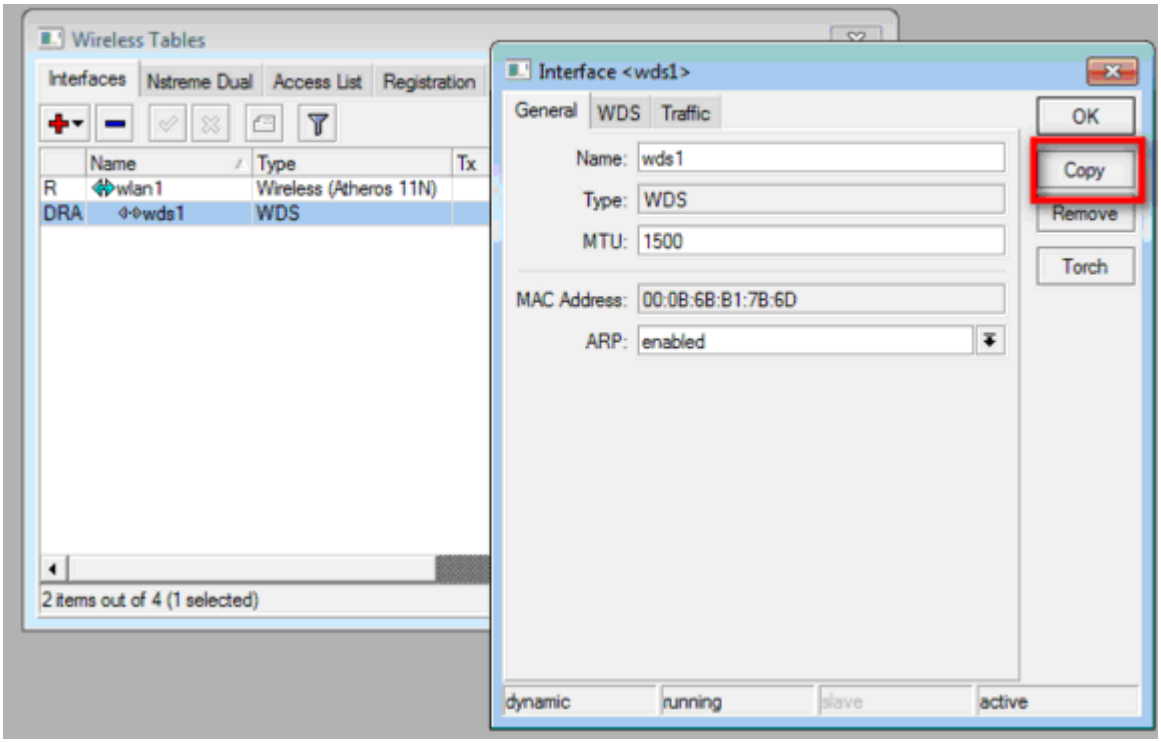
这个显示了如何简单的在 winbox 复制一个项目，如下图的 ARP，我们可以通过 copy 命令复制，当然在 ARP 条目选项中，提供了 Make static 功能，可以直接变为静态 ARP。



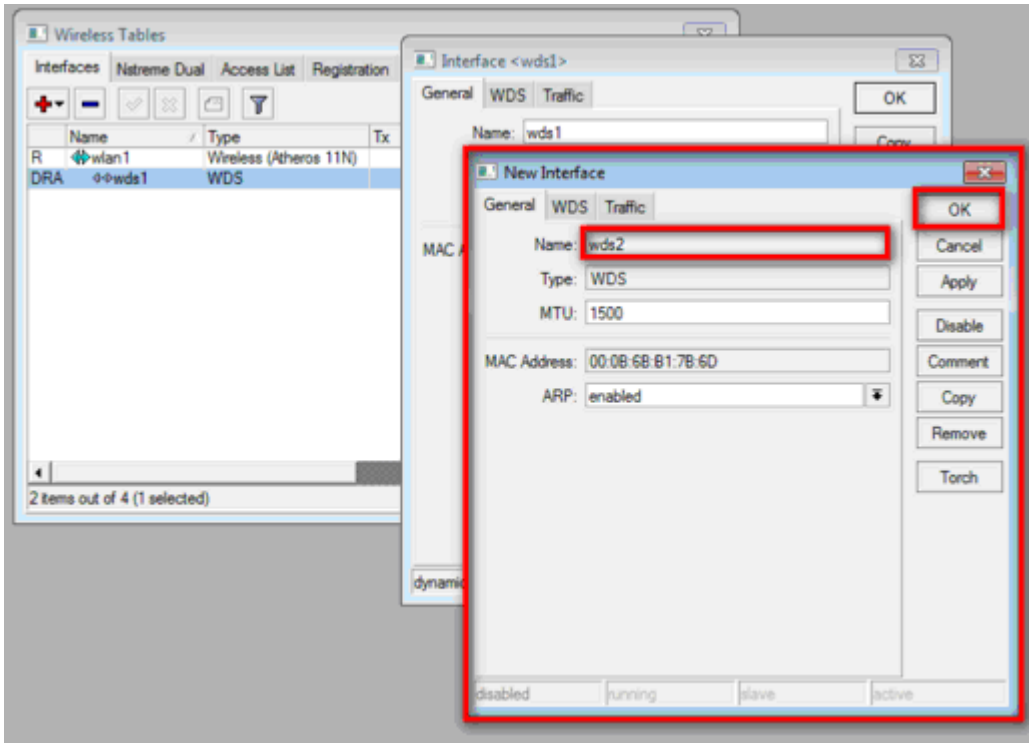
我们也可以通过 COPY 按钮将动态的 WDS 接口变为一个静态接口，这个截图显示了 WDS 的字符前缀状态，如同你看到的 DRA，D 代表动态，R 代表运行，A 代表活动



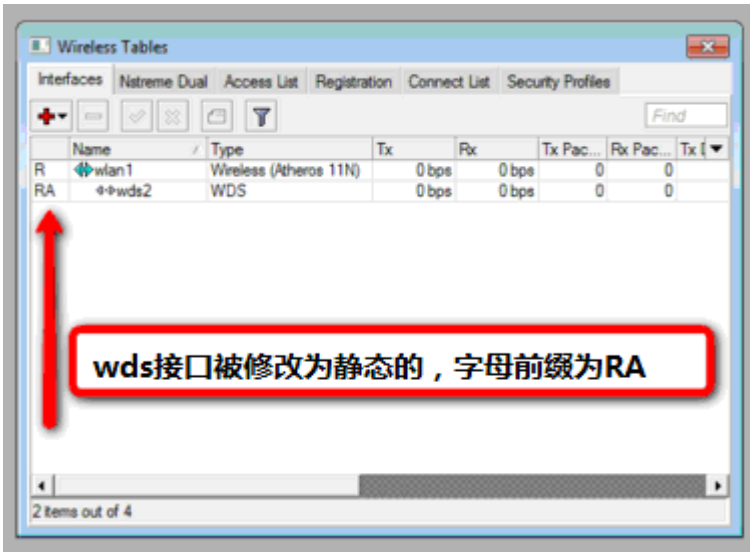
双击 wds1 接口，并点 COPY 按钮



一个新的对话框出现，一个新的 Name 将被自动创建，在这里为 WDS2



你看到新的接口状态已经修改



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?**

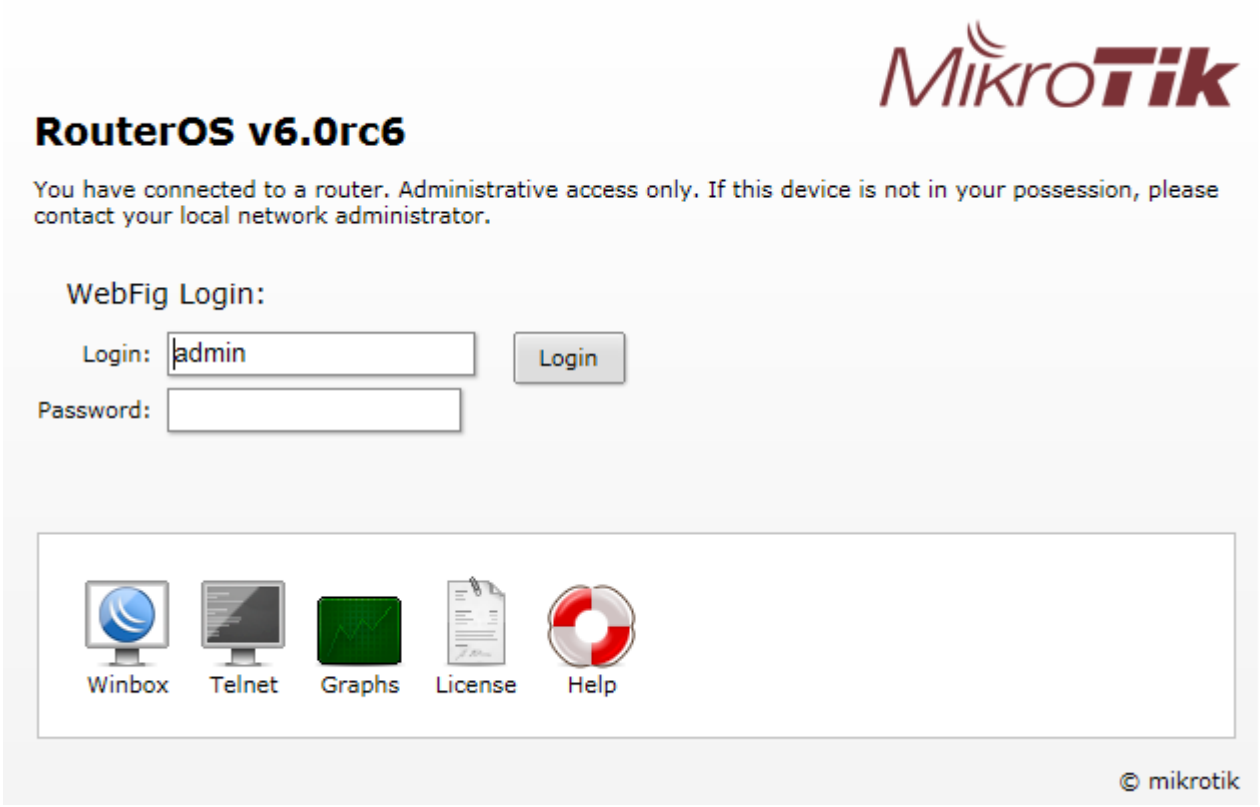
  能，使用 Wine 图形接口，可以运行 Winbox 并连接到 RouterOS。

- **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

一般 RouterOS 在 v4.0 后基本上 ether1 接口默认 IP 地址是 192.168.88.1，也可当你配置好 RouterOS 的 IP 地址后，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。

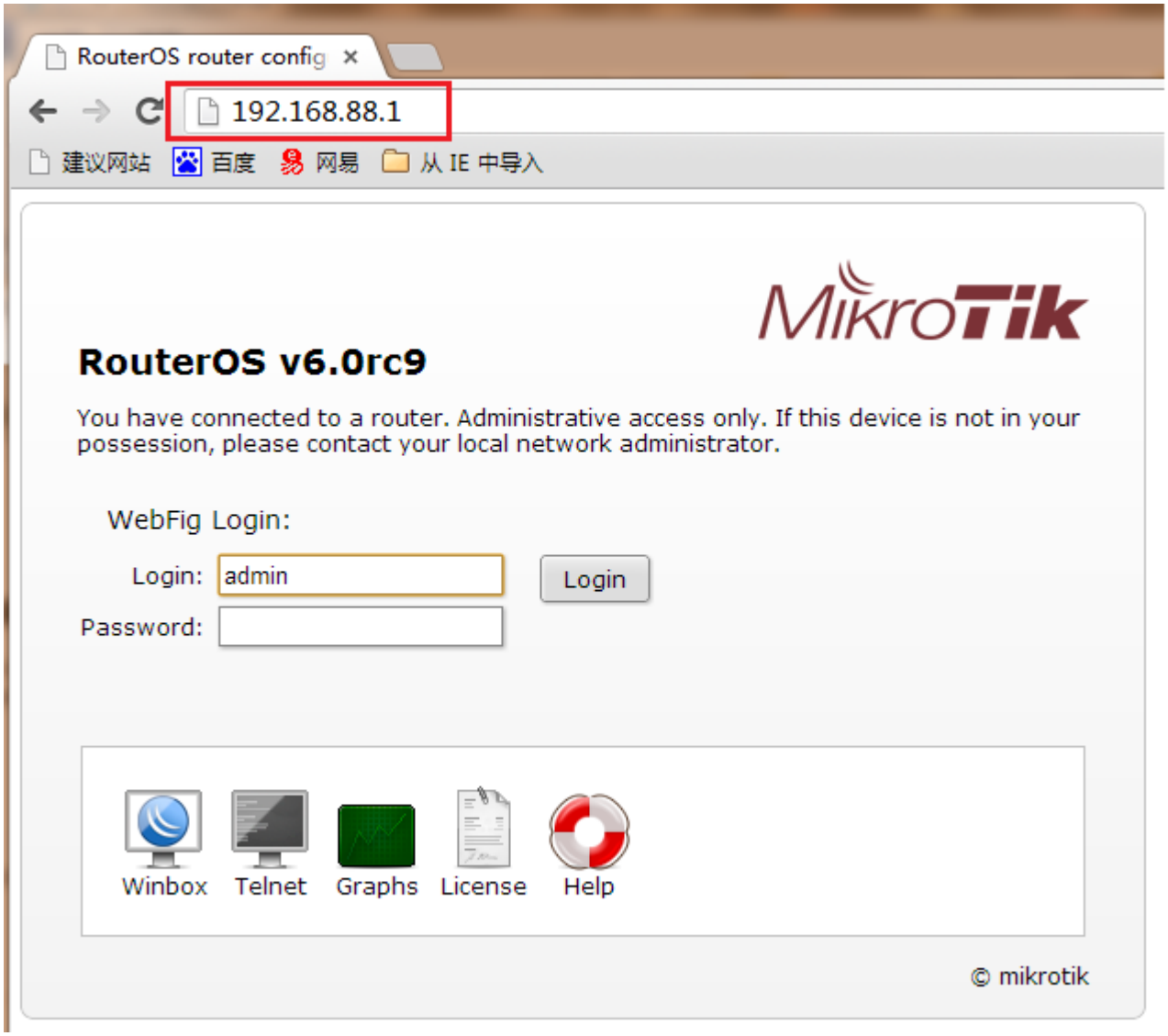
WebFig 是 RouterOS 基于 Web 浏览器的配置工具。能直接管理路由器，不需要增加任何管理软件，通过 Web 浏览器配置，Webfig 是一个独立的平台，可以通过移动设备访问，被用于路由器的直接管理配置。

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，

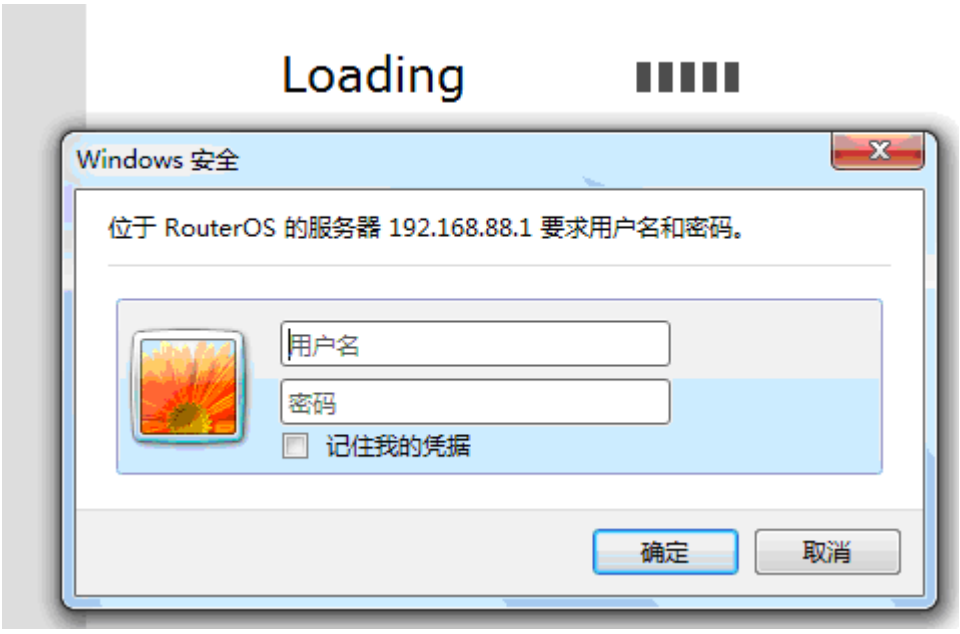
注：因为 webfig 是基于 IP 层访问，如果你需要修改连接路由器的 IP 地址，请谨慎操作；当配置二层的 bridge 或者 ip firewall filter 的 input 链表需要考虑到是否影响 IP 访问

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 的图标启动，如下图。



在点击 webfig 的图标后，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



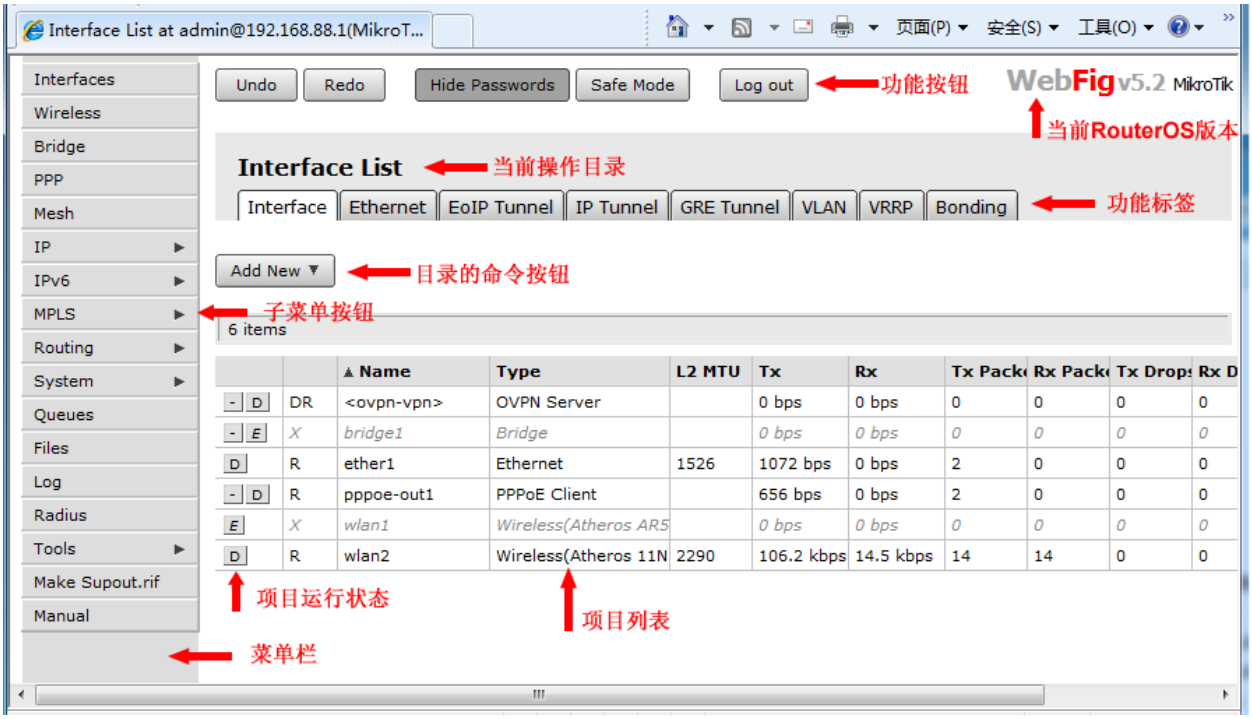
输入正确的用户名和密码即可登录到 RouterOS 的 webfig 配置

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

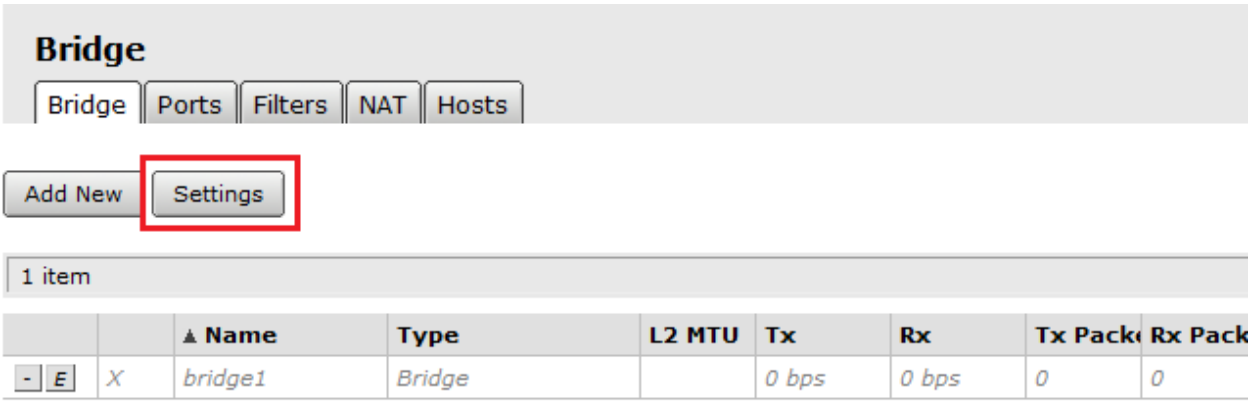
WebFig 界面被设计为近似于 Winbox，几乎同样的布局，菜单栏在左侧，undo 和 redo 在顶部，工作区域在中间部分



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标签，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

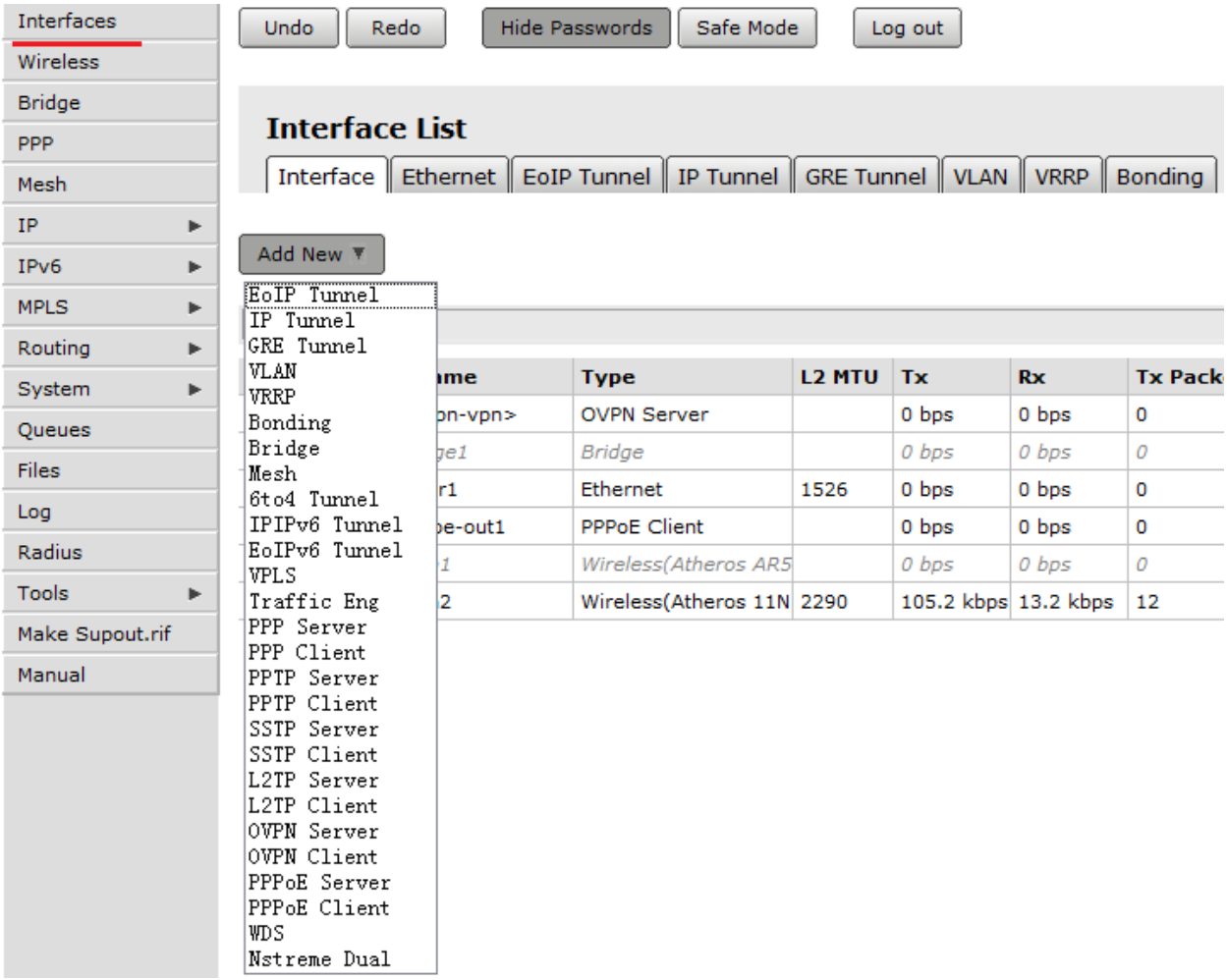


在下面的项目列表里，第一栏我们可以看到相应的字符按钮，他们的代表内容如下：

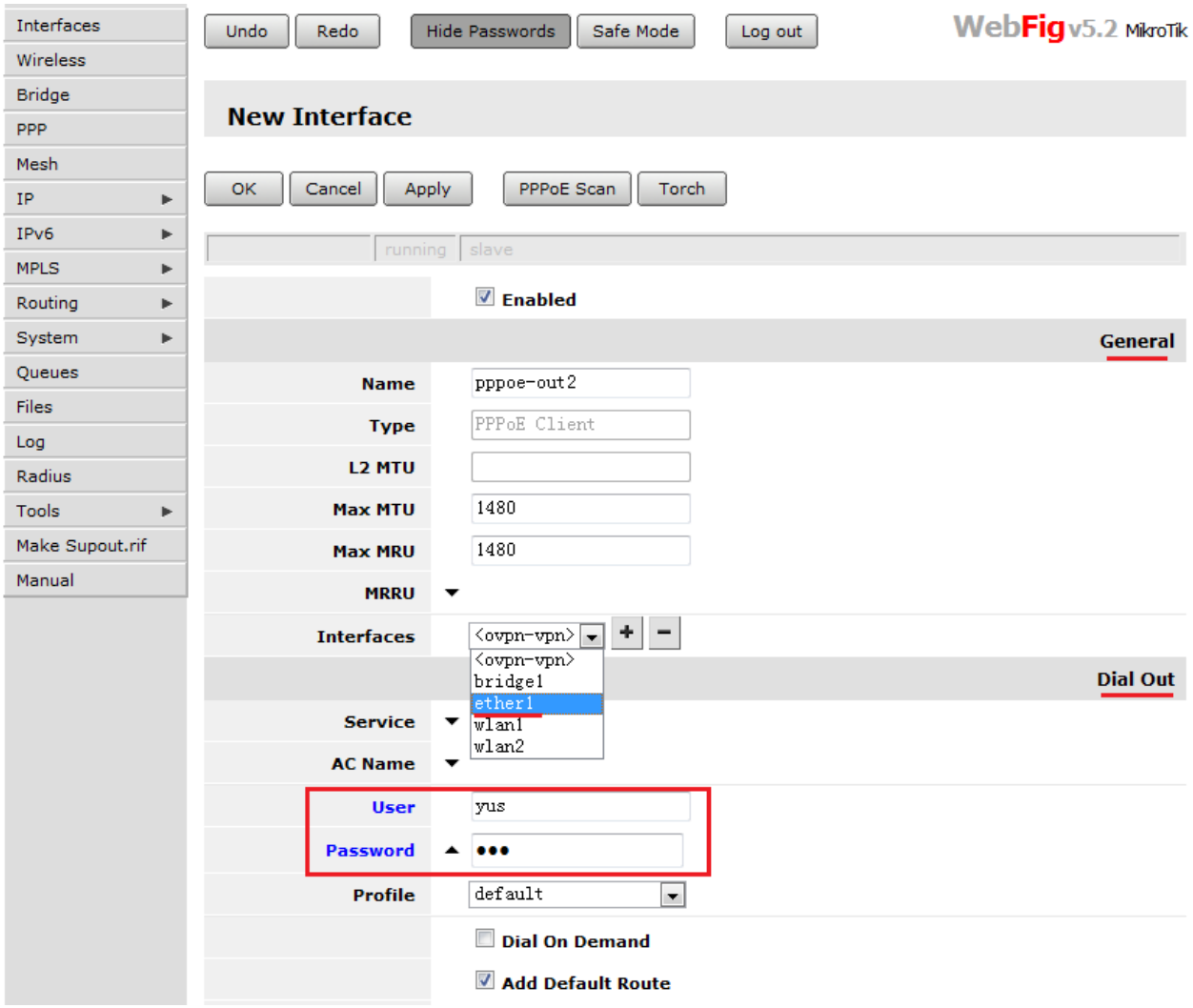
- E - 当前项目启用状态
- D - 当前项目禁用状态
- - - 删除当前项目

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

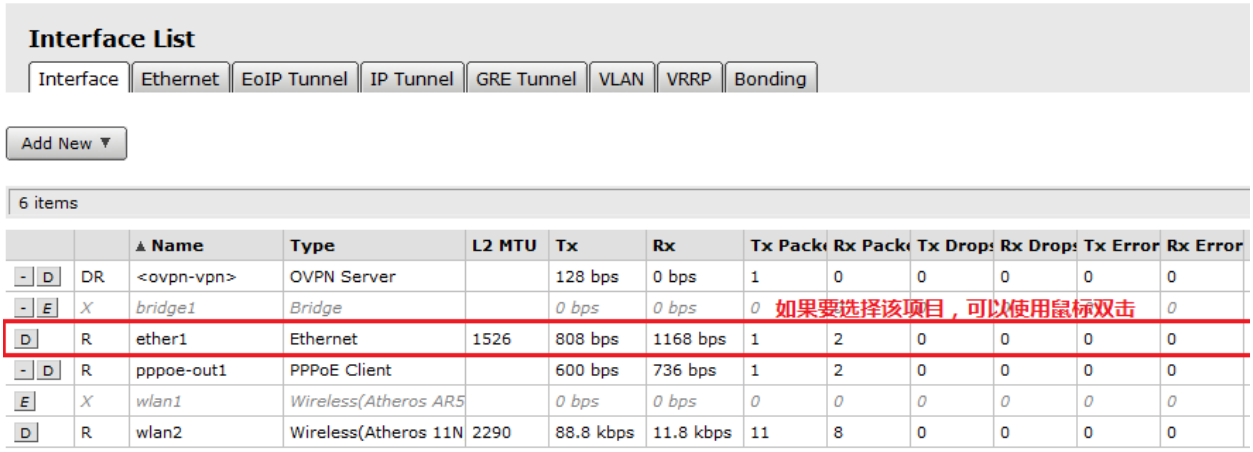
我们可以看到 pppoe-client 里后面两项 PPPoE Scan 和 Torch 按钮

- *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

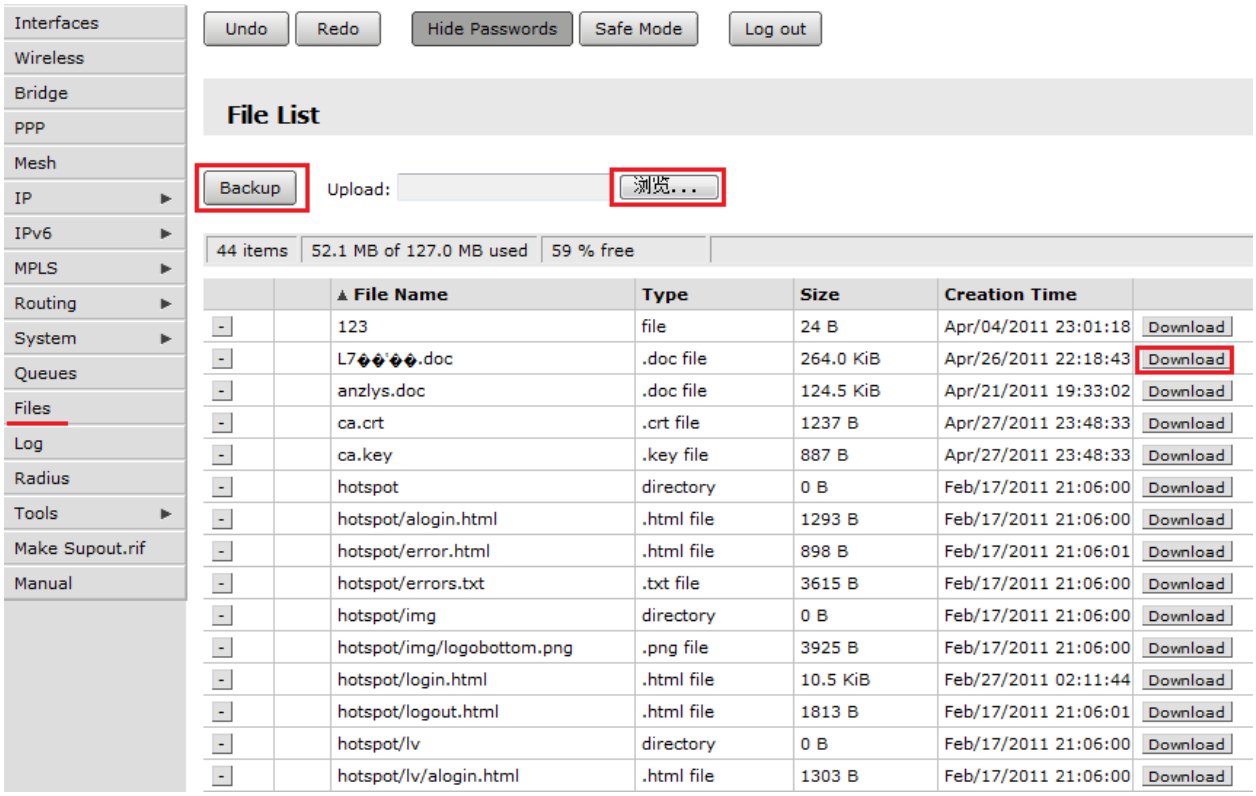
下面的图显示了 ether1 的状态，link ok 表示以太网卡网线已经接入，并且有信号。

当我们需要修改和编辑 interface 列表中的一个网卡时，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文件，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文件，在列表中有每个文件后面都有一个 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。我们通过 CLI 可以在 Console 设置和管理 RouterOS。

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] >
```

命令帮助

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。

```
[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,并返回控制台命令
password – 修改当前管理帐号密码
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 – 推出控制台
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 – 终端平台命令操作界面
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 – 撤消命令
user – 管理员帐号管理
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 层目录

```
 [admin@MikroTik] ip>

 .. – 回到根目录
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
the-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
web-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] >                    | 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>                  |
```

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

指令执行概述

Command	指令
command [Enter]	执行指令
[?]	显示该目录中的所有指令列表
command [?]	显示指令的帮助和变量列表
command argument [?]	显示指令的变量帮助
[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
/	移动到根目录
/command	执行根目录中的指令
..	移动到上一级目录
""	指定一个空字符串

在配置 IP 地址中，配置'address'和'netmask'参数时，在许多事例中你可以将 IP 地址和子网掩码一起定义，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：

```
/ip address add address 10.0.0.1/24 interface ether1
```

```
/ip address add address 10.0.0.1 netmask 255.255.255.0 interface ether1
```

Tab 快速输入

在 console 控制台有两种方式帮助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字符，console 会自动查找到对应的命令和参数，例如一下操作：

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

基本操作命令

接口管理（Interface Management）

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>
```

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=qq

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

 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=qq

 1 X chain=forward action=drop dst-address-list=qq
[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
```

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
- 载入接口驱动
- 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 地址和子网淹码：

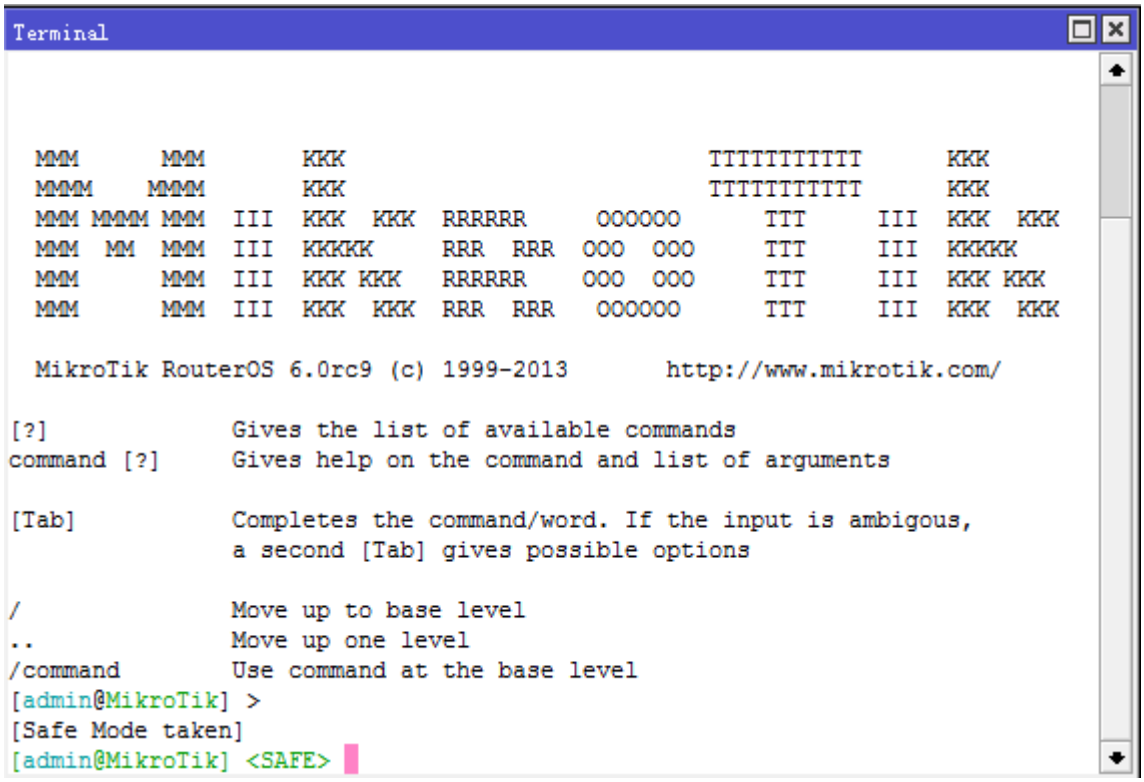
```
your choice: a
enable interface:
ether1   ether2   wlan1
enable interface: ether1
ip address/netmask: 10.1.0.66/24
#Enabling interface
/interface enable ether1
#Adding IP address
/ip address add address=10.1.0.66/24 interface=ether1 comment="added by setup"
 + a - add ip address
 * g - setup default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

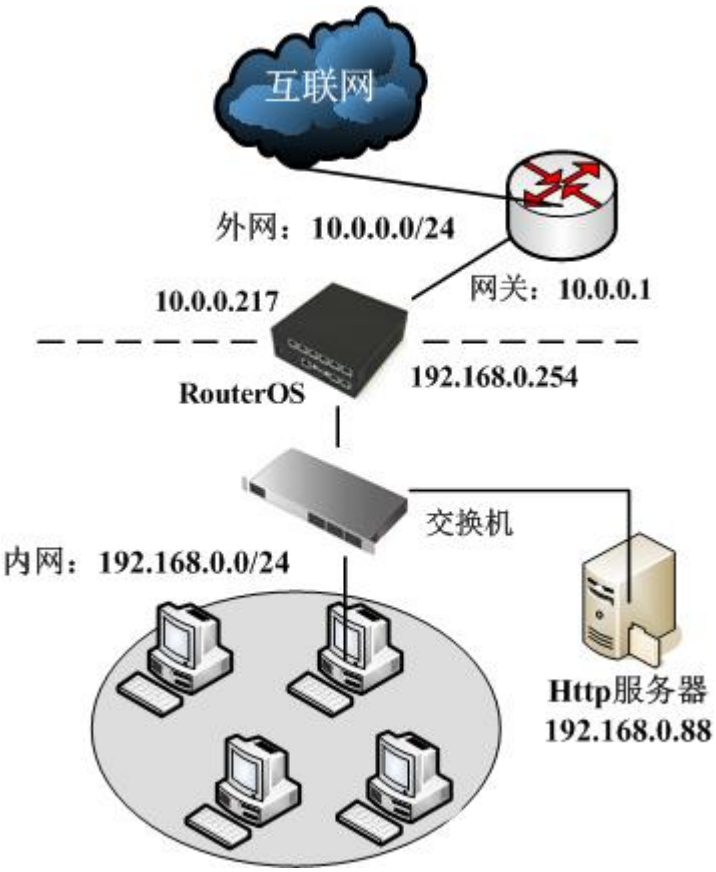
```
[admin@MikroTik] ip route>
[Safe Mode taken]

[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 连接或者 winbox 连接中断，这时连接超时 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：



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：

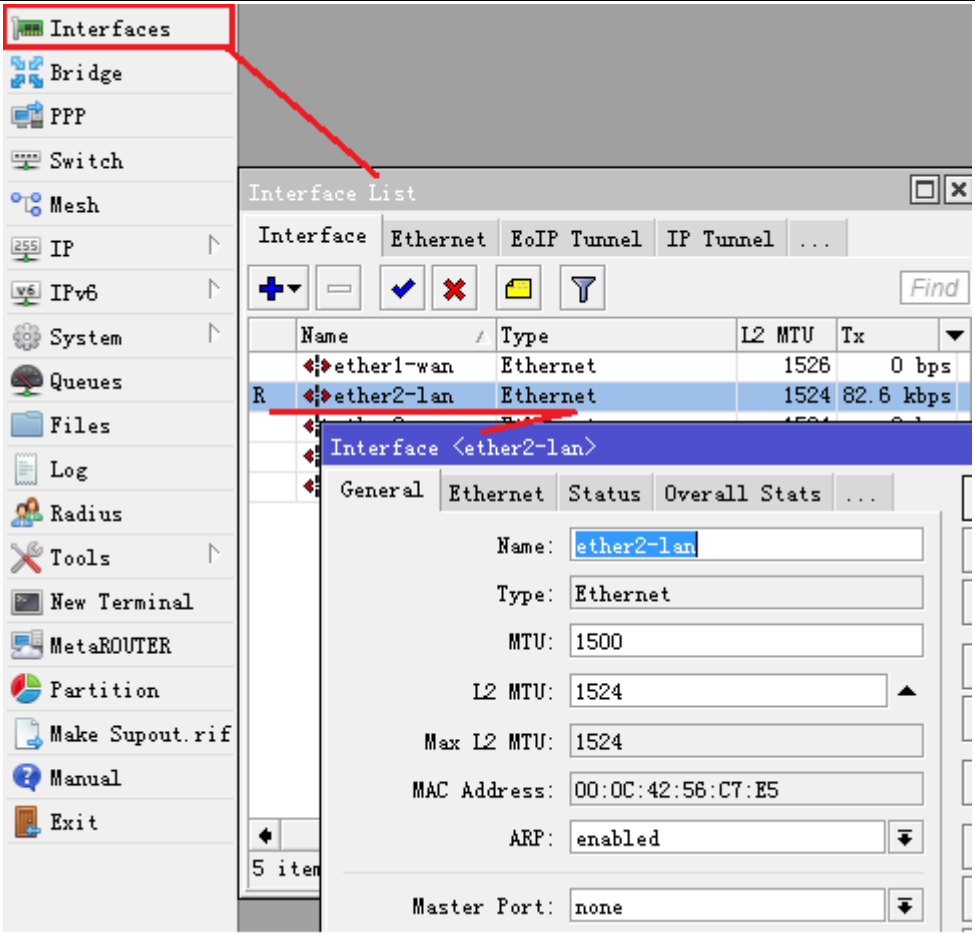
- LAN 内网使用地址为：192.168.0.0 子网淹码 24-bit（255.255.255.0）。路由器的地址在这个网络中为 192.168.0.254
- 接入 ISP 的外网 WAN 为 10.0.0.0 子网淹码 24-bit（255.255.255.0）。路由器的地址是在网络中为 10.0.0.217
- 外网 DNS 为 61.139.2.69，202.98.68.96

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

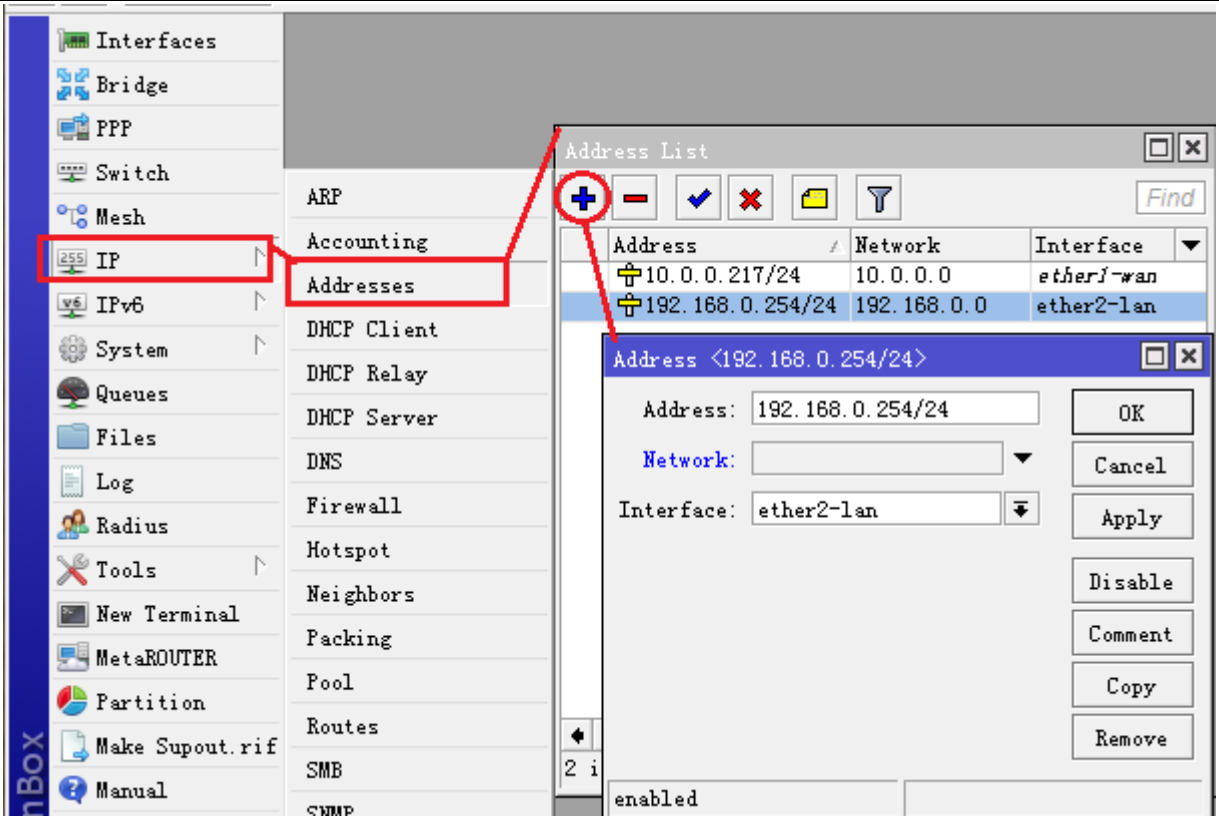
在/interfaces 列表中修改 ether1 为 ether1-wan，定义为外网接口；修改 ether2 为 ether2-lan 定义为内网接口，如图：



同样将 ether2 修改为 ether2-lan，指定内网接口：

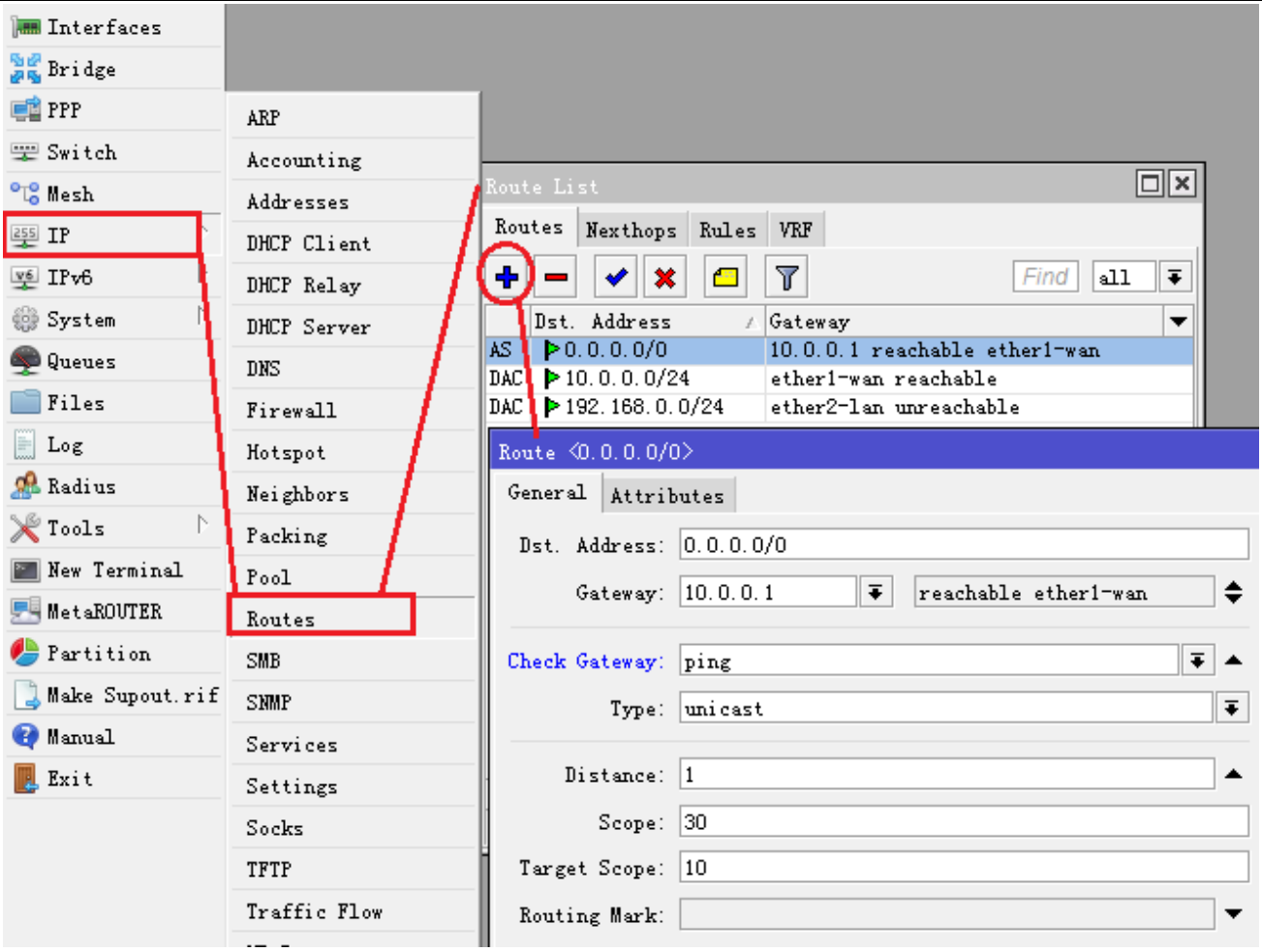
第二步：添加 IP 地址

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：



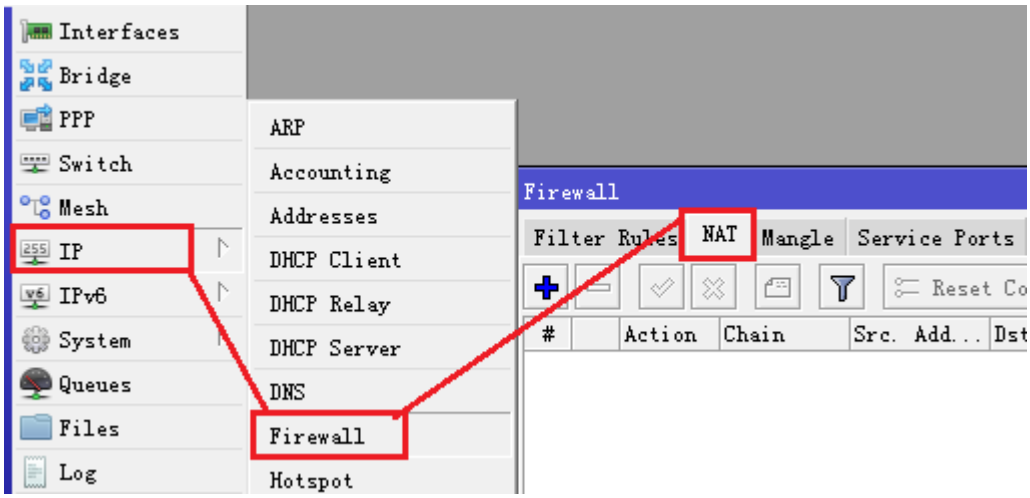
第三步：添加默认网关

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：

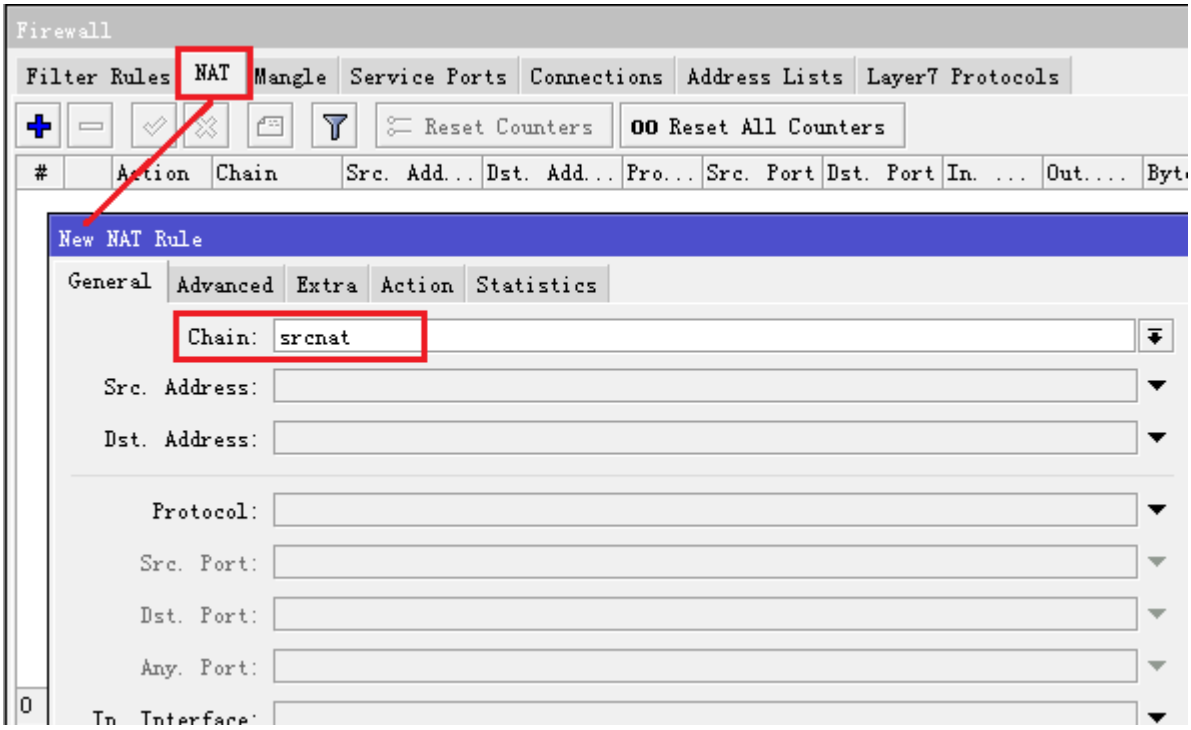


第四步，NAT 地址转换

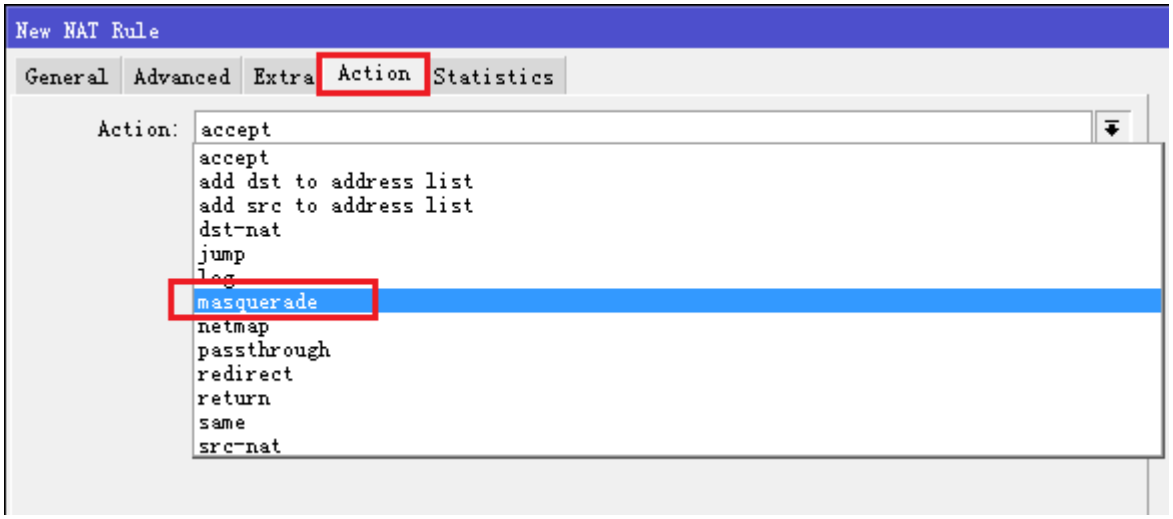
在/ip firewall nat 里点击“+”添加伪装规则：



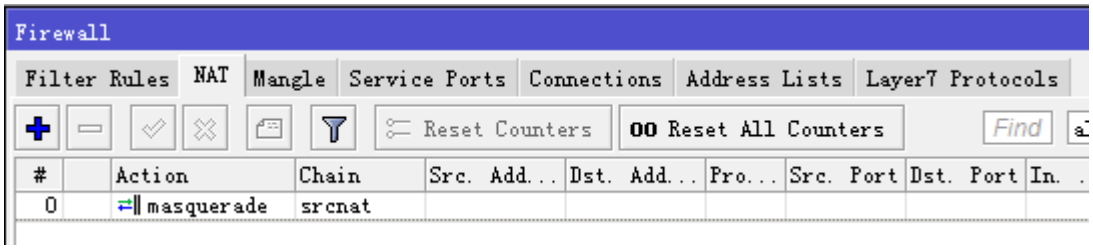
在 NAT 里添加新的规则，在 chain 里选择 srcnat 链表：



在选择 action 里的 action=masquerade 规则：

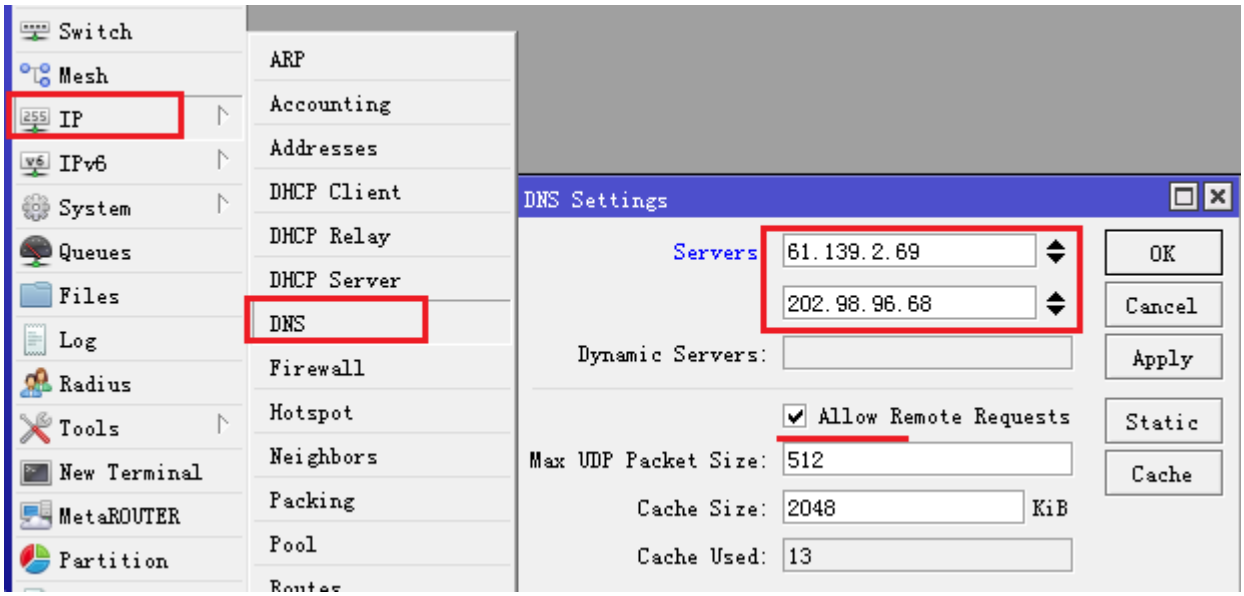


添加完成后：



第五步，DNS 配置

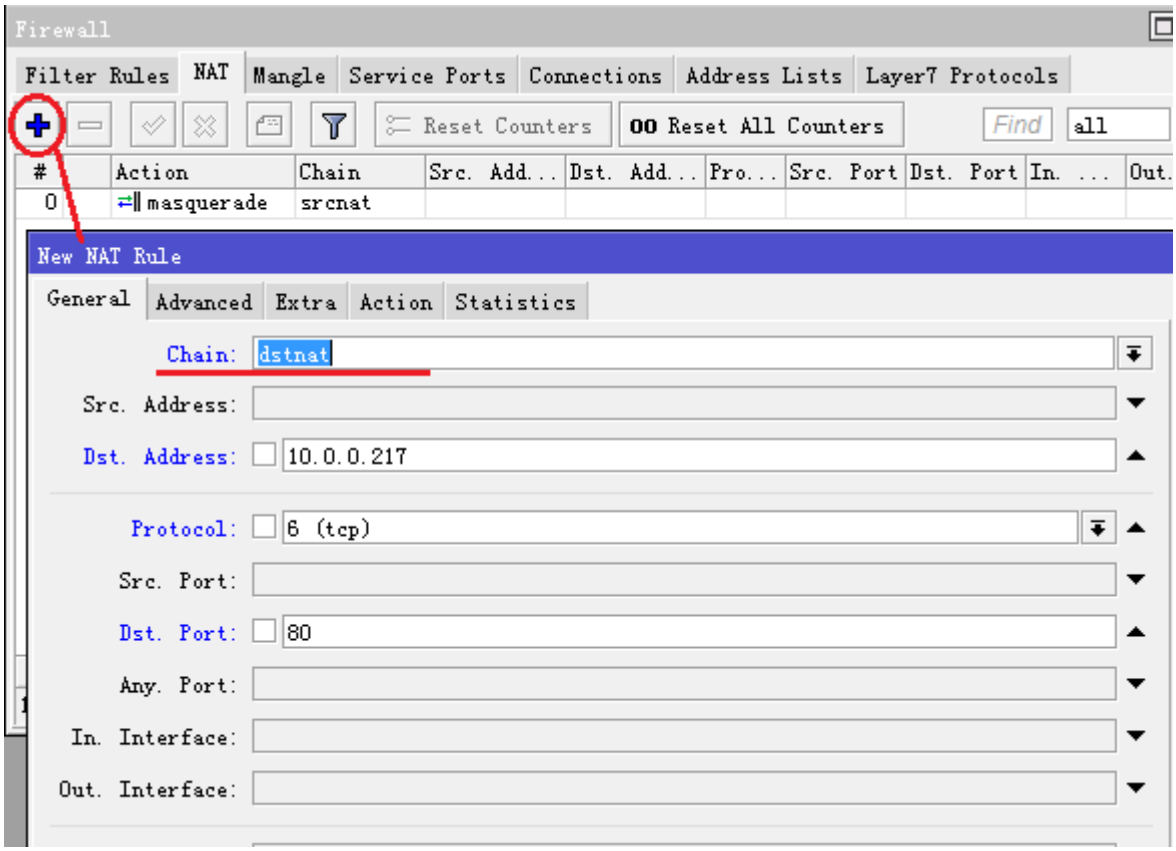
在/ip dns 的 settings 中添加多个 DNS 服务器地址，根据需要启用 DNS 缓存（allow remote requests），并能通过 Cache size 修改 DNS 缓存大小：



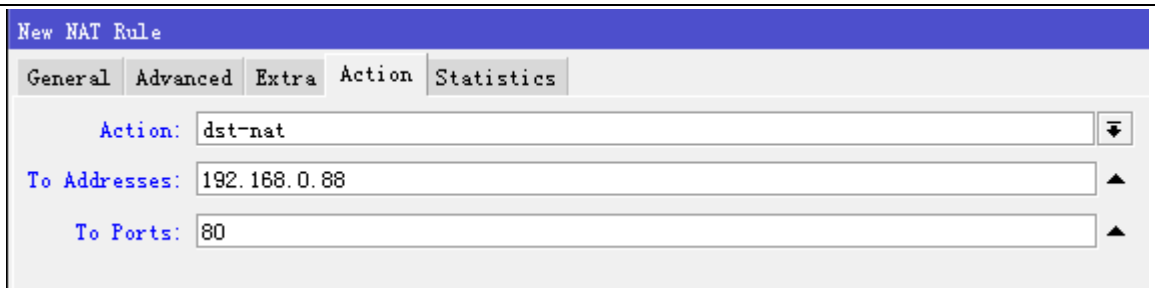
到此，上述的单线上网事例就已经配置完成！

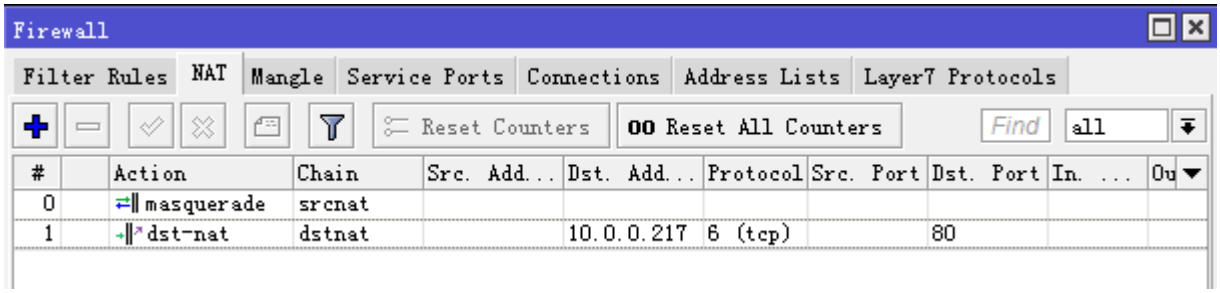
端口映射

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



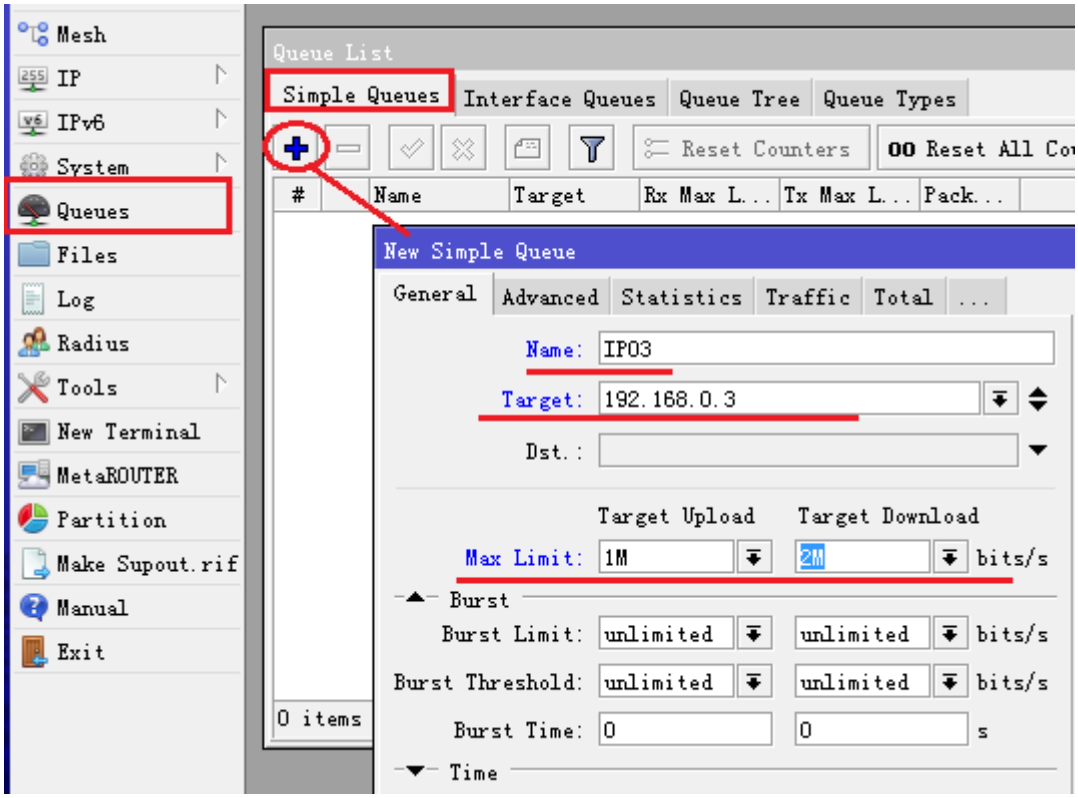
在 action 选择 dst-nat 操作，to-address 设置内网 http 服务器 IP 地址，和端口 80

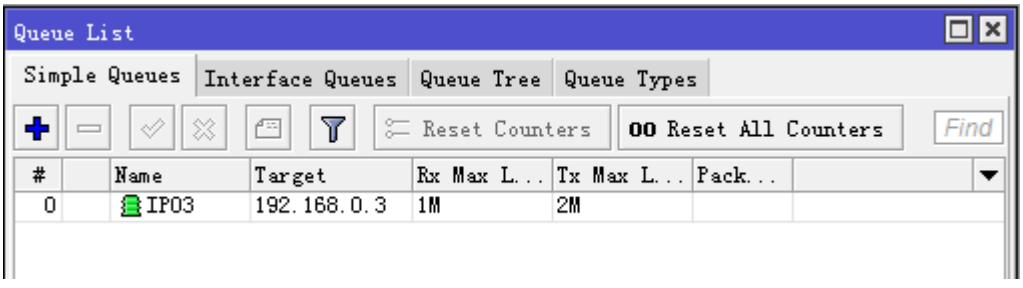




主机带宽控制

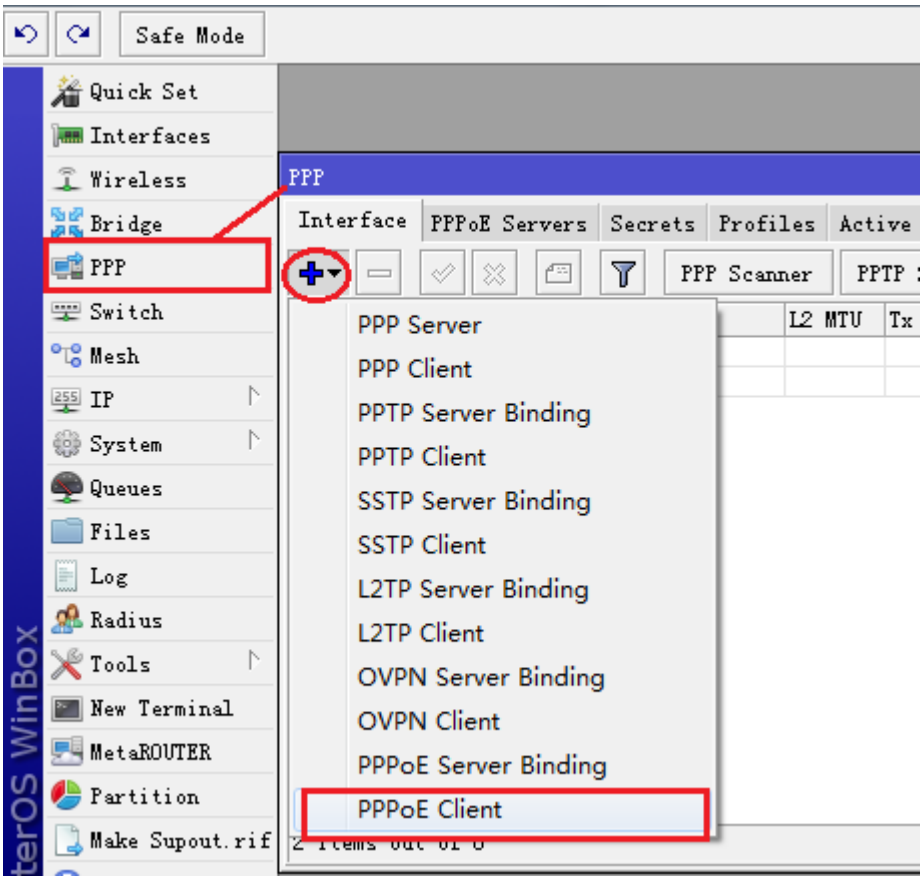
进入 Queue 添加带宽控制规则，选择 simple queue，添加主机 IP 是 192.168.0.3，并取名为 IP03，设置带宽为上行(upload)1m，下行(download)2m



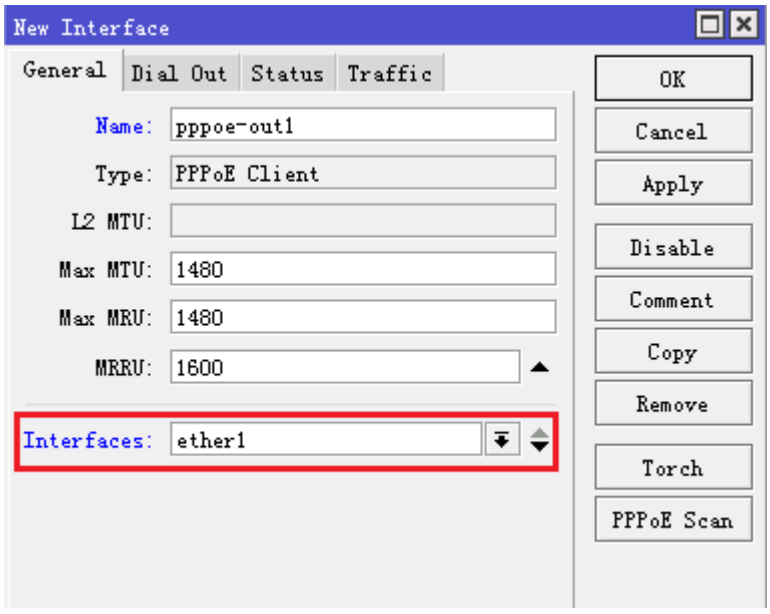


ADSL/PPPoE 拨号配置

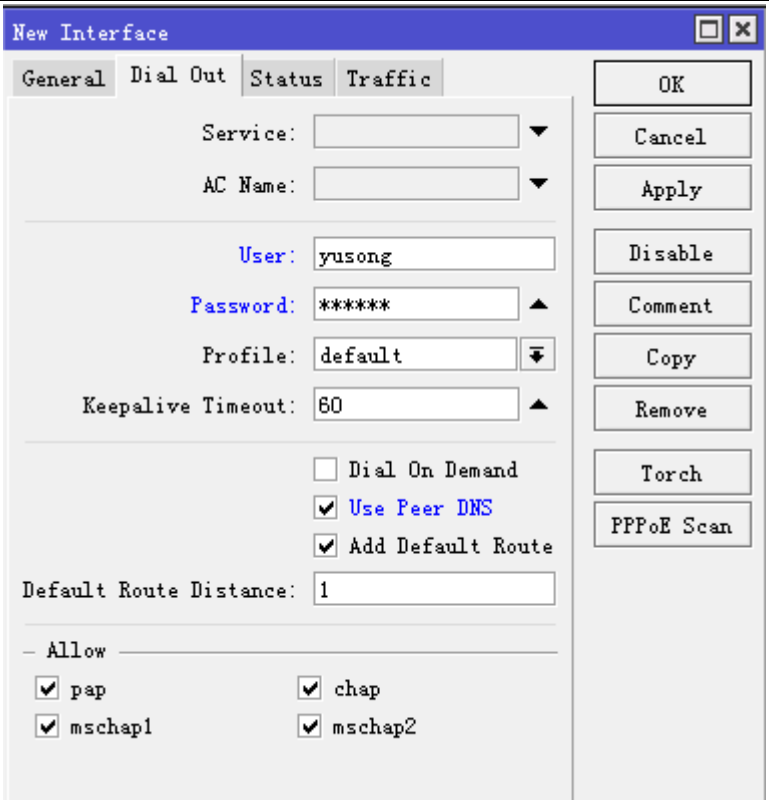
ADSL 拨号需要建立 PPPoE-client，进入 ppp 里点加号创建一个 PPPoE-client：



PPPoE-client 创建后取名 pppoe-out1，选择拨号的 interface 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

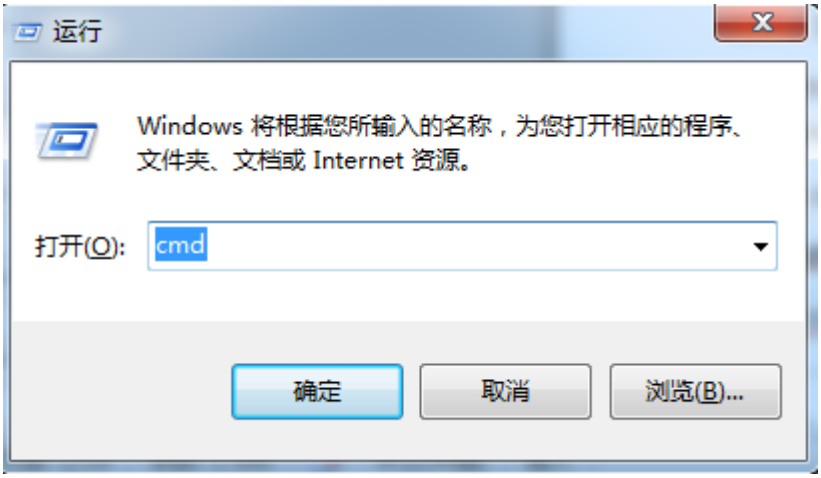


网线连接后 pppoe-out1 会自动拨号，连接成功后 pppoe-out1 会显示 R，代表拨号成功，可以进入/ip address 下查看自动获取的 ip 地址，网关也会自动添加到/ip route 中，会自己配置 DNS 到/ip dns 下

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”



下面我们举例一下常用的 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 回应。 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紧张 ping，也可以使用 arp -a 命令查看对方主机的 mac 被学习到，来判断是否在线。也可以通过 arp 来判断是否局域网内存在 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，路由是否安装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 数据包生存周期，避免在互联网中产生永久环路。IP 数据包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 数据包并向 IP 包的发送者发送 ICMP time exceeded 消息

通过该命令可以看到数据包是如何通过网络，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 (即时通信监听)

即时通信监听被称为 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 36.4 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 36.10 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。

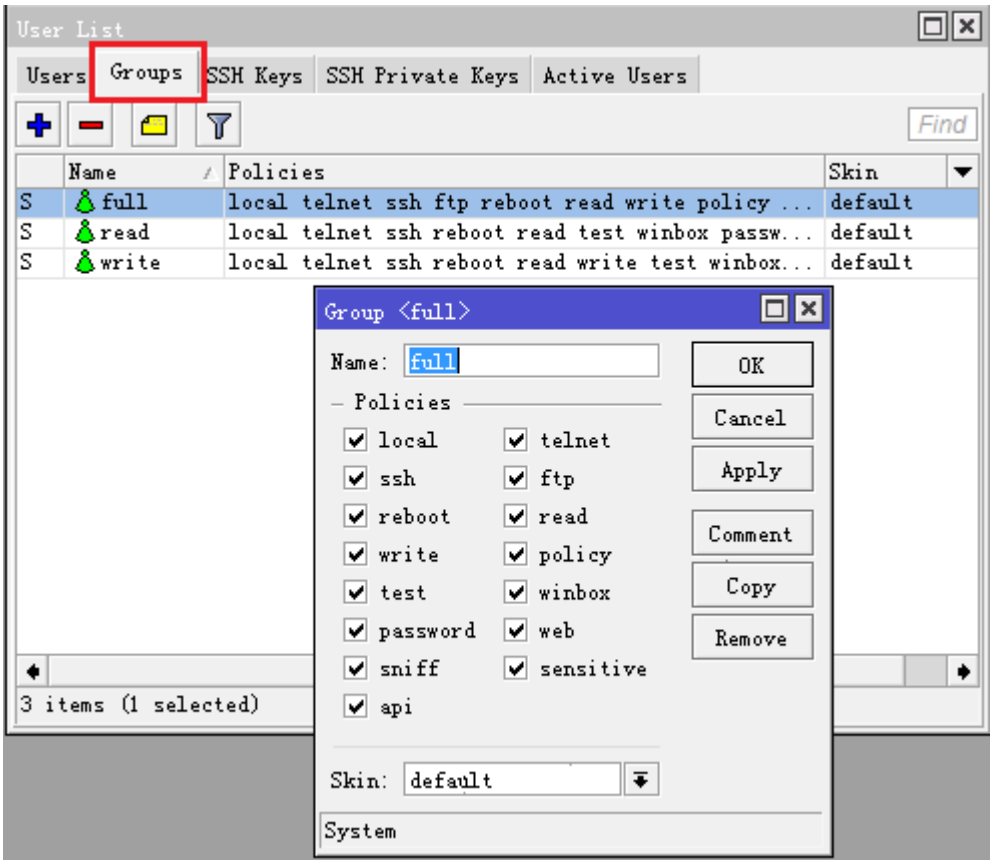
```
[admin@MikroTik] > user
[admin@MikroTik] /user> print
Flags: X - disabled
 #    NAME                GROUP               ADDRESS             LAST-LOGGE
 0    ;;; system default user
      admin               full                                    may/02/201
[admin@MikroTik] /user> set admin password=123456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

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] >
```

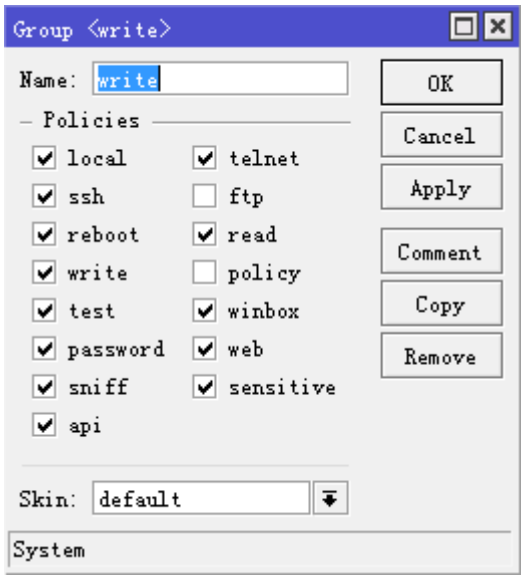
账号分组权限

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，进入 winbox 的 system—user 菜单下的 groups：

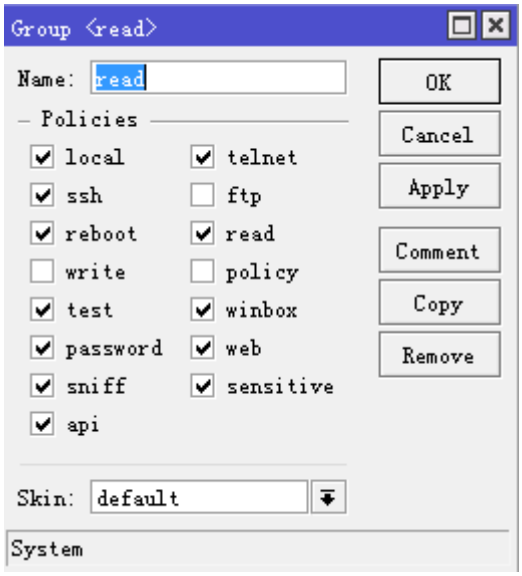


在每个组里我们可以定义他们的权限策略，包括是否有权使用 local、telnet、ssh、ftp、reboot、read、write、policy、winbox、web…

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。

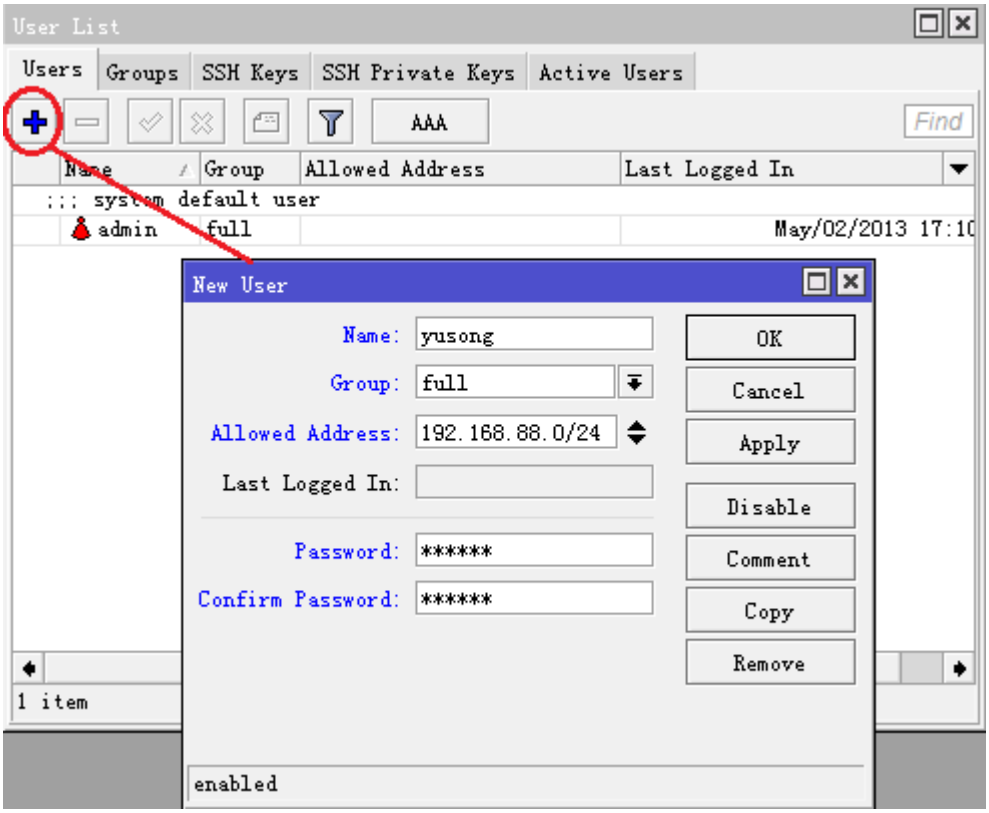


Read 权限除了不能上传文件和修改密码外，不能写入任何的配置，即只能登陆设备查看



新建账号

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 可以通过 backup 下的 save 命令将系统备份为二进制文件，采用 FTP 访问或在 winbox 中的 file 列表中下载备份文件，并可以通过备份文件恢复路由器设置。

RouterOS 通过 export 命令导出配置文件，可生成文本文件（可编辑脚本），同样使用 FTP 或通过在 winbox 中的 file 中下载文件，导入配置则将脚本文本文件导入路由器。

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。

注：为了保证备份不会失败，请在将备份的文件恢复到同样的软件版本和同样的硬件配置上去.

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** 指令载入配置，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] – 载入备份文件的配置
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：

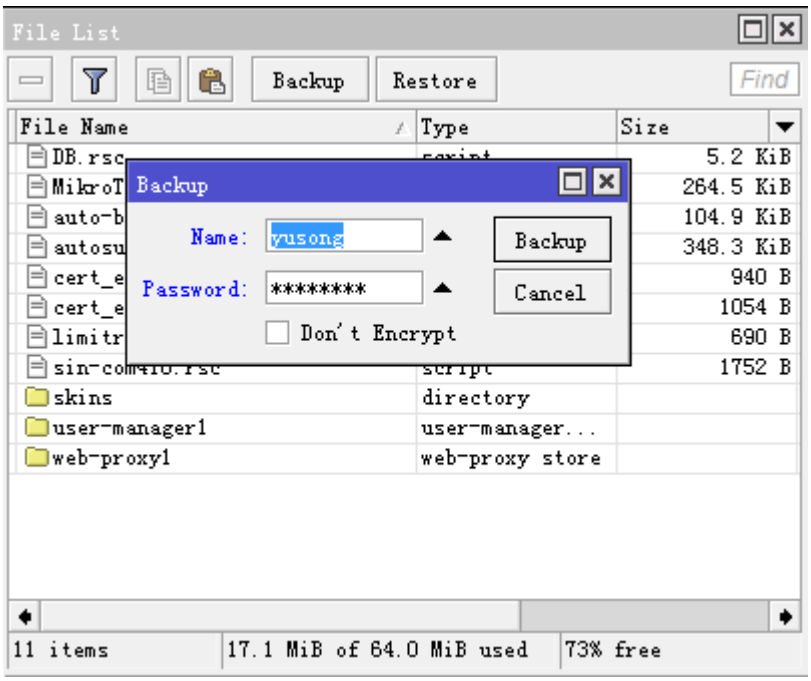
```
[admin@MikroTik] > file print
  # NAME                            TYPE          SIZE        CREATION-TIME
  0 test.backup                     backup        12567       aug/12/2002 21:07:50
[admin@MikroTik] >
```

导入备份文件 **test**:

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

```
...
```

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



导出指令（Export）

指令名称：***export***

Export 指令用于导出脚本配置信息，这个命令可以在任何目录下执行。**export** 同样也可以通过 **file** 属性指定保存的文件名，可用 FTP 或者进入 winbox 下载。export 导出的文件是明文，且是标准的 RouterOS 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] – 保存的文件名称。

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>
```

在路由器中查看导出的文件：

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

RouterOS 5.12 新增功能 export compact 命令，该命令简化了导出的参数，仅导出修改的配置，系统默认配置参数将不再被一并导出。有助于对比路由器的配置和方便导入其他路由器。

```
[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] >
```

6.0 版本后对 export 做了新的调整

export compact 命令被取消，当使用 export 命令时，直接导出基本配置，系统默认配置参数将不再被一并导出。如果需要导出详细的配置可以使用 verbose

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

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。

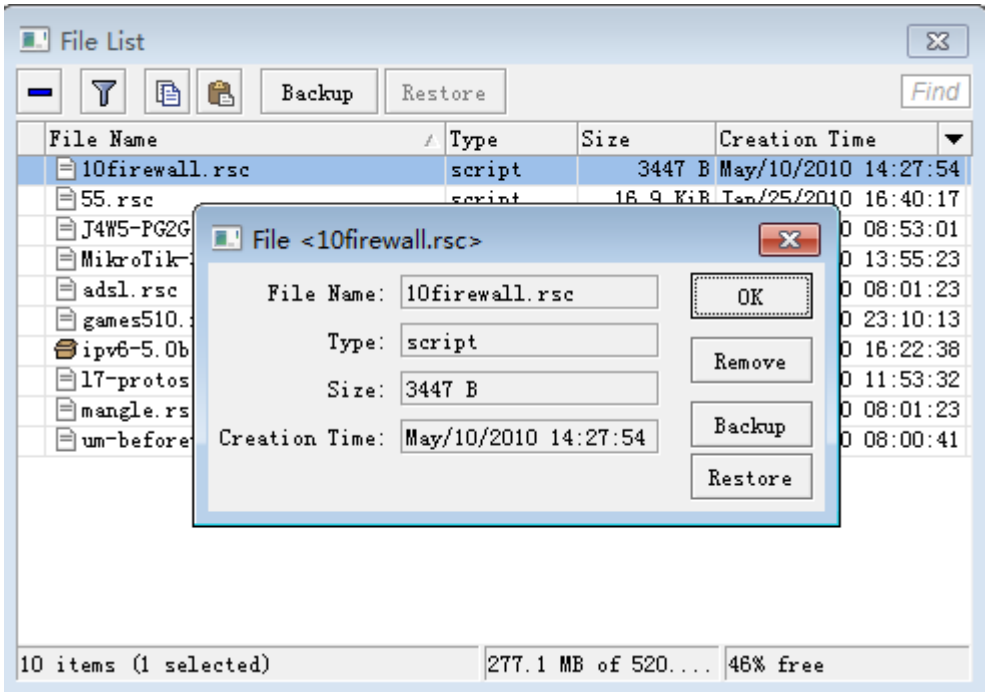
指令描述：

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] >
```

Winbox 里可以查看生产的.rsc 的文件：



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）IP 地址和其他配置将会被抹去，在 **reset** 指令执行后路由器将会重启。RouterOS v3.x 后版本，在复位后 ether1 接口会配置默认 IP 地址 192.168.88.1/24

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 文件后，复位会导入指定的 RouterOS 脚本配置文件。

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，并停止和卸载系统文件，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，并停止和卸载系统文件，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] >
```

设置路由器身份名：

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

2.5 系统资源管理

操作路径：***/system resource***

查看系统资源可以了解 RouterOS 的运行情况

注：通过 monitor 命令实时的 CPU 占用率、内存和硬盘等使用情况。

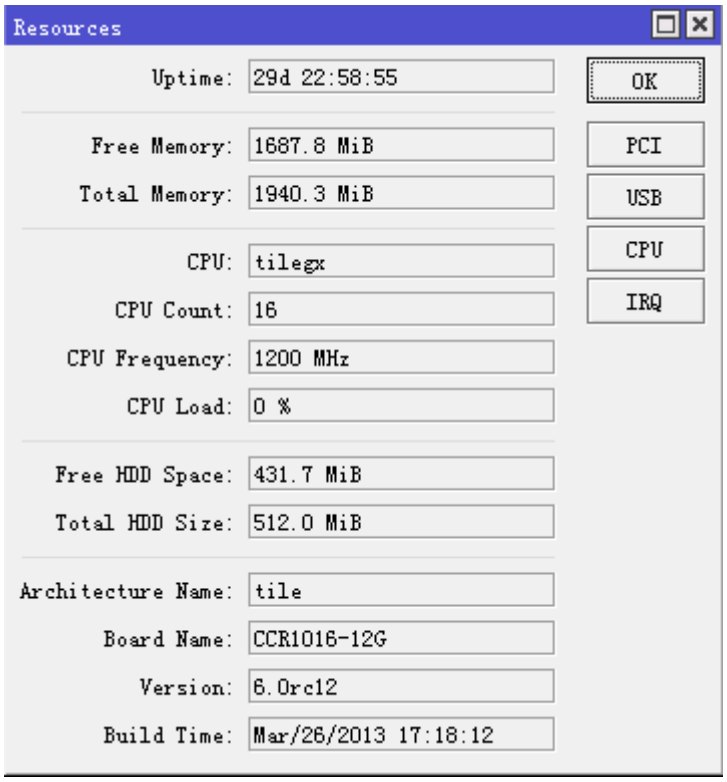
查看基本的系统资源情况：

```
[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
```

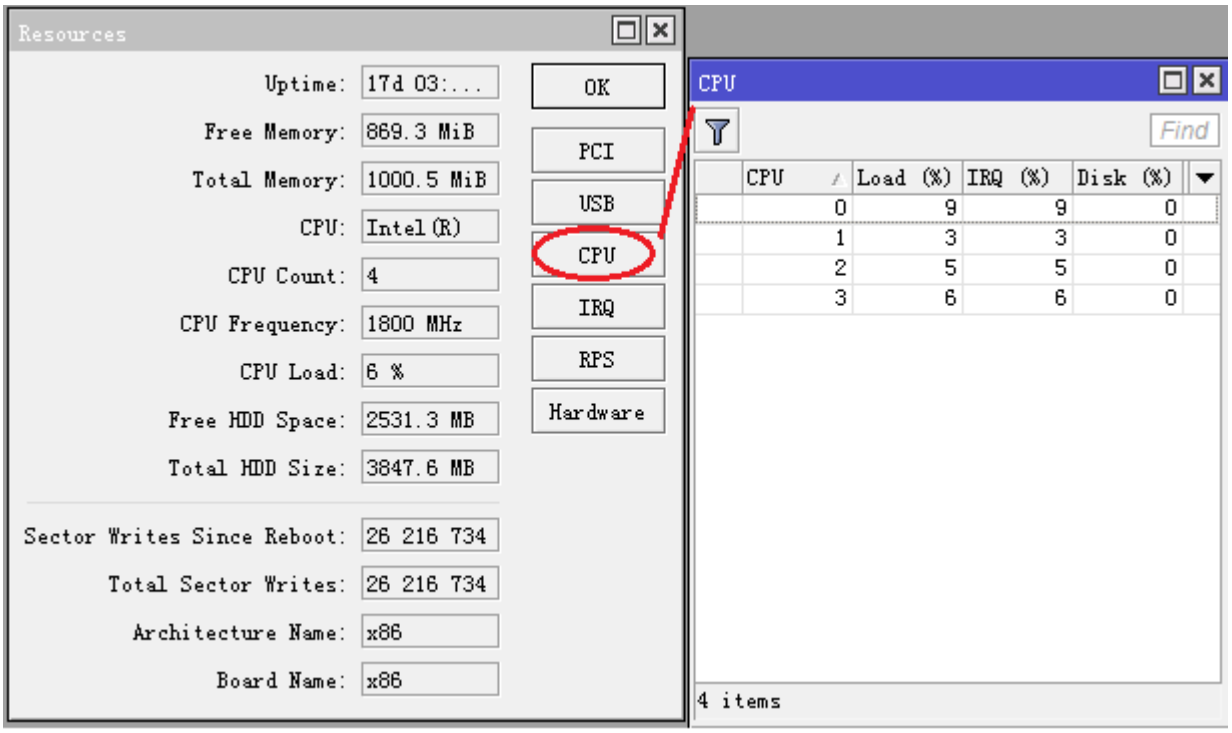
实时查看系统 CPU 和空闲内存使用情况：

```
[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
```

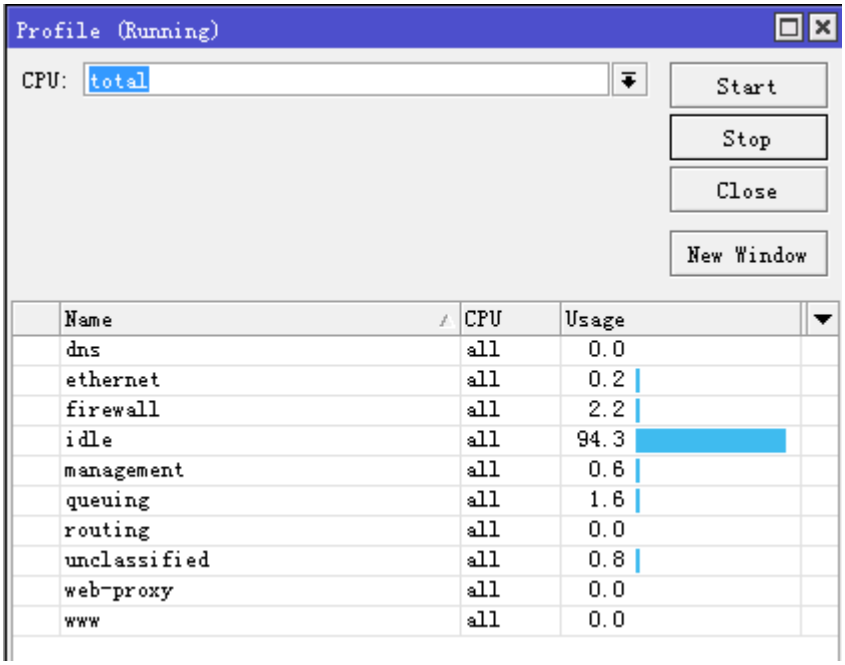
使用 winbox 查看：



RouterOS 5.0 后对多 CPU 进行了优化，可以查看每个 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

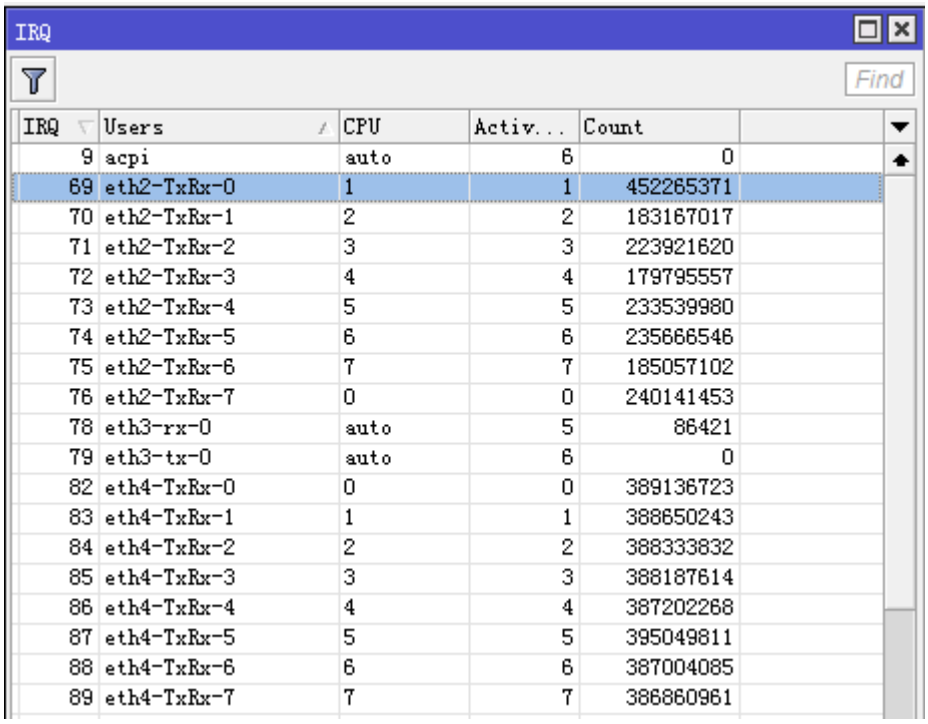
命令路径: ***/system resource irq print***

IRQ “中断”简单的说就是，每个硬件设备（如：硬盘、网卡、USB 设备等）都需要和 CPU 通信，让 CPU 响应这些硬件设备的请求，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,硬件设备主动打扰 CPU 的现象就可称为硬件中断,就像你正在工作的时候收到 QQ 消息一样,一次 QQ 信息，你就会查看，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。

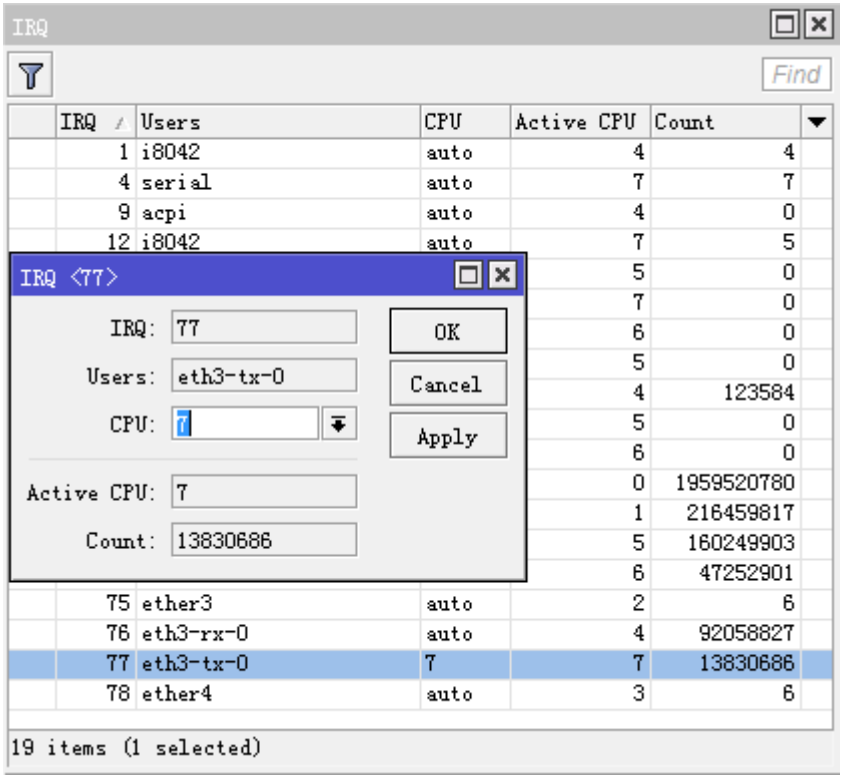


IRQ	Users	CPU	Activ...	Count
9	acpi	auto	6	0
69	eth2-TxRx-0	1	1	452265371
70	eth2-TxRx-1	2	2	183167017
71	eth2-TxRx-2	3	3	223921620
72	eth2-TxRx-3	4	4	179795557
73	eth2-TxRx-4	5	5	233539980
74	eth2-TxRx-5	6	6	235666546
75	eth2-TxRx-6	7	7	185057102
76	eth2-TxRx-7	0	0	240141453
78	eth3-rx-0	auto	5	86421
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 给对应的硬件设备，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>
```

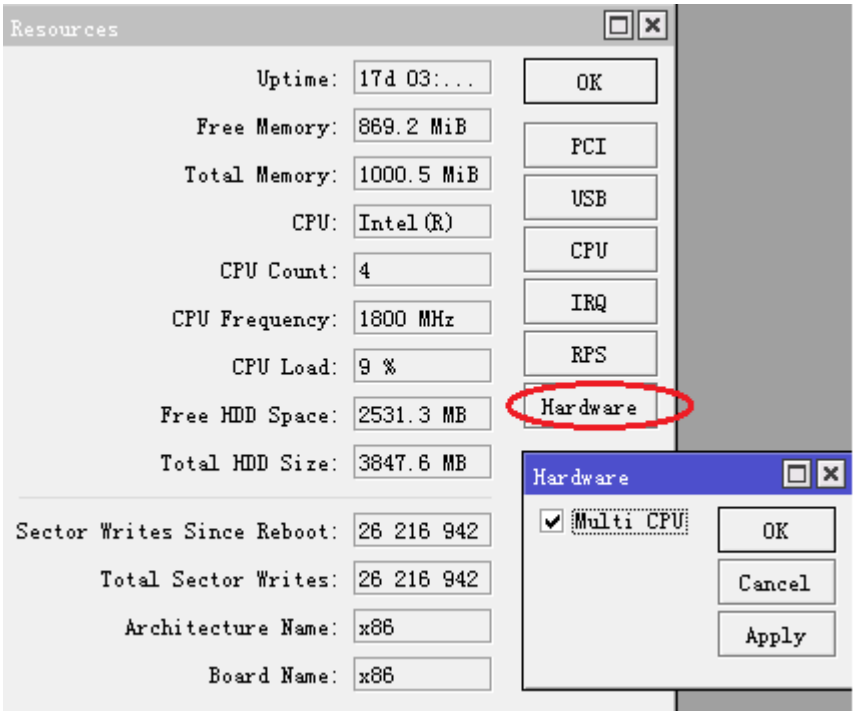
RouterOS x86 平台多 CPU 设置

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锁死，一旦发生这样的情况将发出重启指令。软件计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。

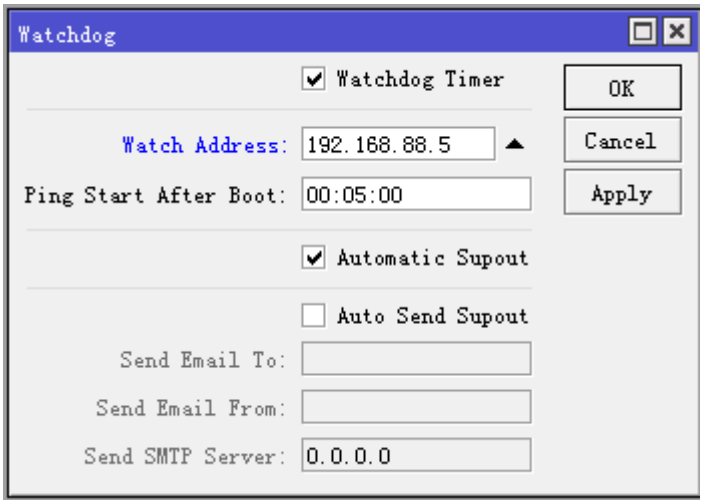
属性描述

auto-send-supout (yes | no; 默认: **no**) – 技术支持文件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，如果新的技术支持文件产生将命名为"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: **""**) – 接收技术支持文件的邮件地址。
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 技术支持文件，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 分钟后没有回应，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 和 bt 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,	支持全球定位系统设备

x86, tile)	
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)	多协议标签交换（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)

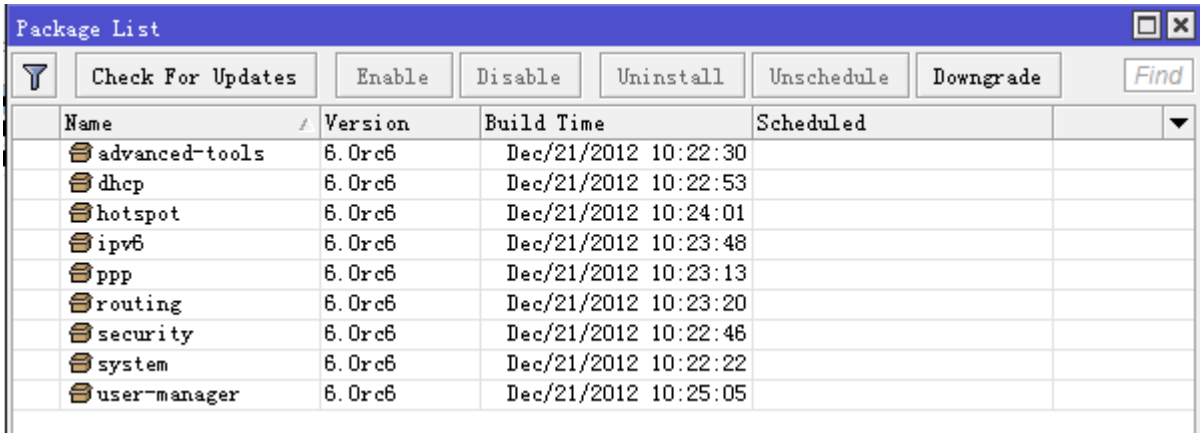
查看功能包

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文件。

RouterOS 升级包区别

当通过 BT 下载完 RouterOS 软件后，如“Routeros-ALL-6.0rc6”里面有多个文件，每个文件对应不同的硬件做升级和降级设置，BT 包里有四个文件名（2015 年下半年官方已经取消了 BT 下载，仅提供主升级包和扩展升级包下载）：

- **all_packages_mipsbe** - 对应所有 Atheros 芯片的 RB400、700、900、2011 系列产品和 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 设备，如 hAP lite 和 hAP ac 等
- **all_packages-arm** – 对应 ARM 芯片的 RB3011 设备

其他文件：

mikrotik-x.x.iso 光盘镜像文件，用于 x86 平台安装。

如果是 2.9 版本的 BT 文件区分如下：

- **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）

RouterOS 升级操作

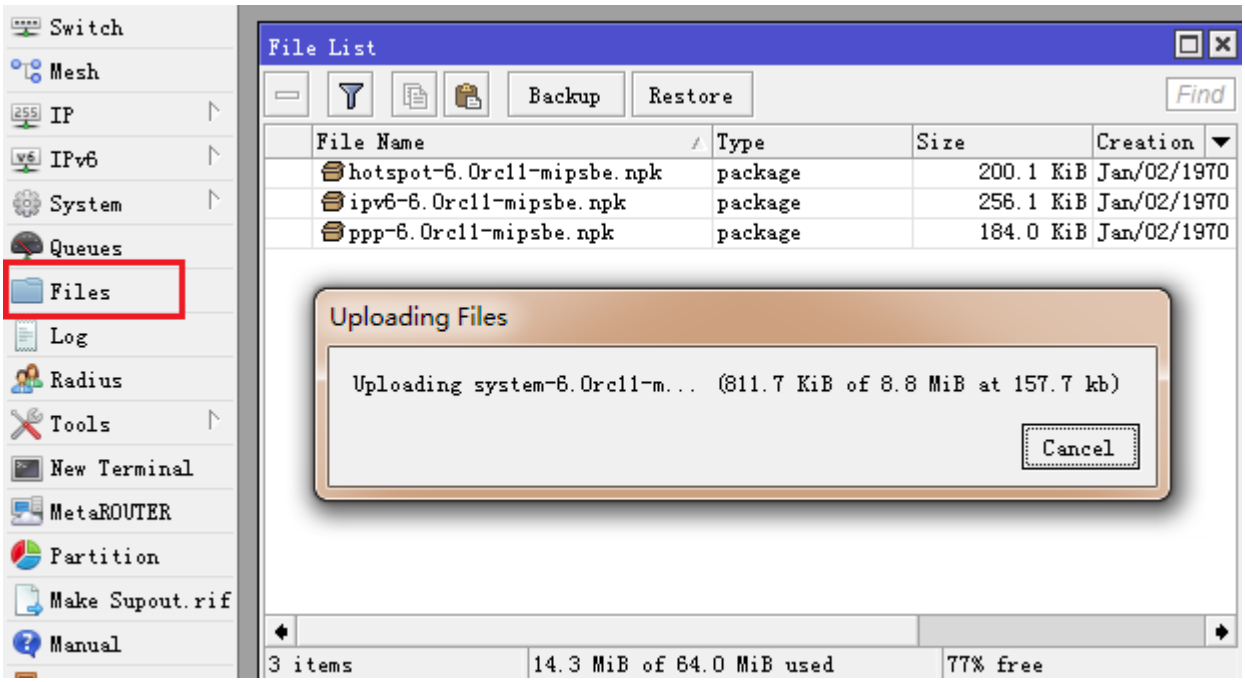
根据你使用 RouterOS 的情况不同，选择上传的升级包文件（注：system-x.x.x.npk 的升级包是必须要，否则无法升级）。如何来确定你当前使用的功能包，可以通过在 system package>的目录中查询对照如下图：

```
Terminal
[?]             Gives the list of available commands
command [?]     Gives help on the command and list of arguments

[Tab]           Completes the command/word. If the input is ambigous,
                a second [Tab] gives possible options

/               Move up to base level
..              Move up one level
/command        Use command at the base level
[admin@MikroTik] > sys
[admin@MikroTik] /system> package
[admin@MikroTik] /system package> print
Flags: X - disabled
 #    NAME                      VERSION                    SCHEDULED
 0    system                    6.0rc6
 1    routing                   6.0rc6
 2    security                  6.0rc6
 3    ppp                       6.0rc6
 4    user-manager              6.0rc6
 5    dhcp                      6.0rc6
 6    advanced-tools            6.0rc6
 7    hotspot                   6.0rc6
 8    ipv6                      6.0rc6
[admin@MikroTik] /system package>
```

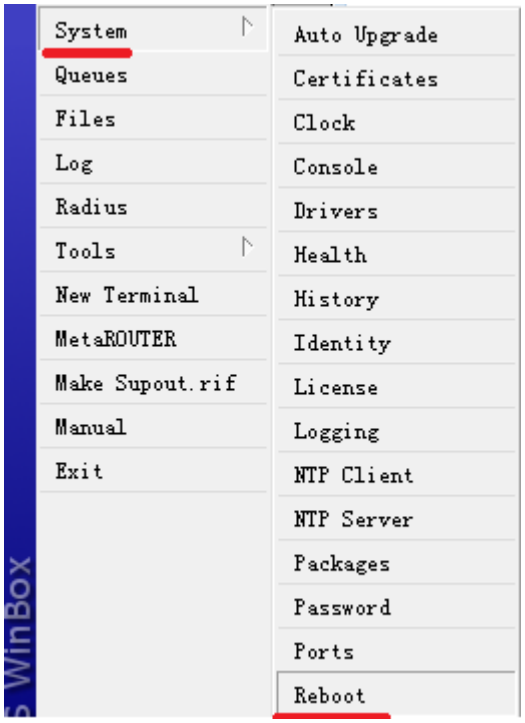
建议根据自己的需求安装或升级功能包(无线选择 wireless、PPPoE 认证选择 PPP 等等)，过多的安装功能会影响路由器的性能

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，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。

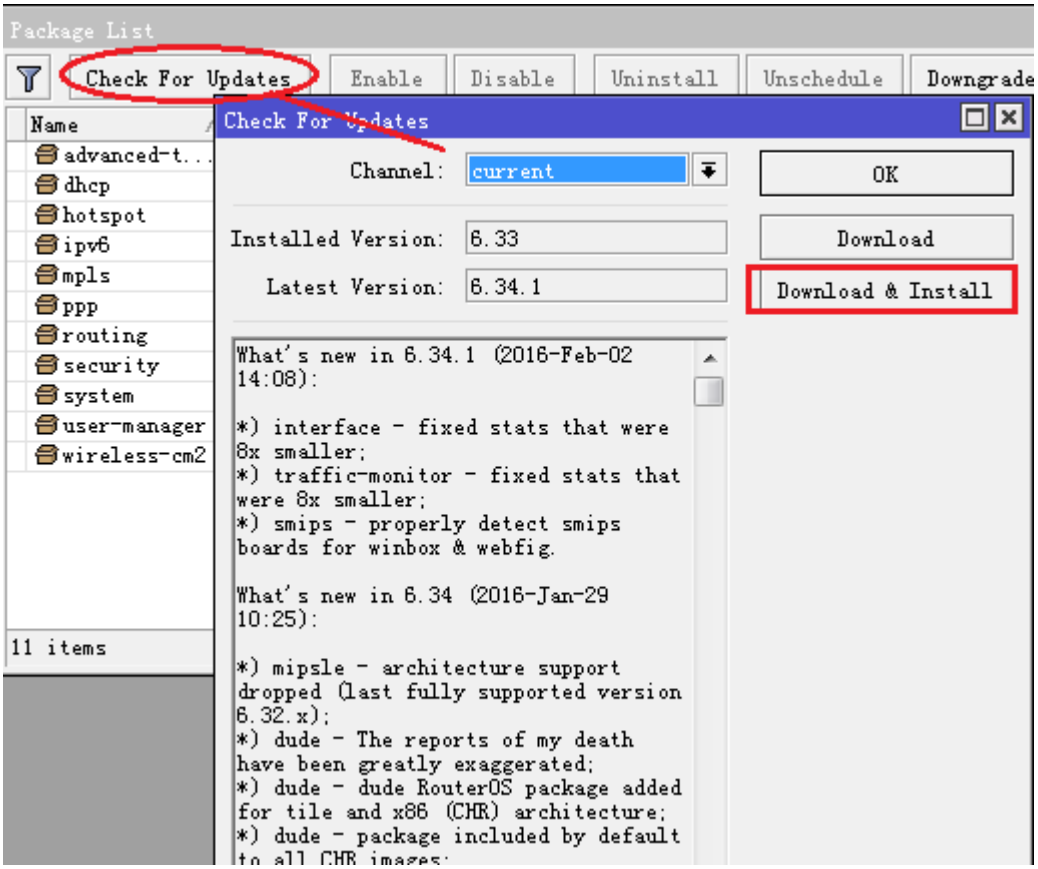
在线升级

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、稳定版本和 bug 修复版本三类。

- Bugfix only – bug 修正版本，即对前一个版本的 RouterOS 做 bug 修复
- Current – 当前版本，即当前发行的相对稳定版本
- Release candidate – 候选开发版本，即正在开发的版本，主要提供测试。

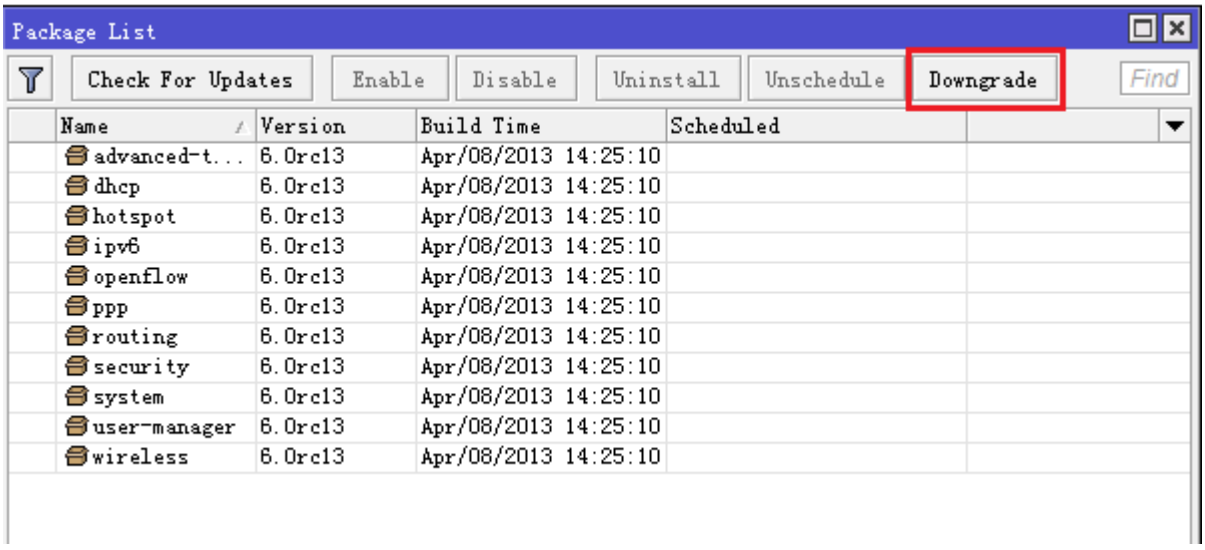
因此在你选择版本升级时，一定要考虑使用哪一个版本，并看清楚官方的 changelog 信息，是否有你需要修复 bug 的地方或者你感兴趣的功能等。

下面是 RouterOS 进入 system -> package 目录下选择 check for updates 的升级界面，当你选择好你需要升级的版本，并点击 Download&Install，RouterOS 会自动连接远端的 MikroTik 升级服务器。



降级选项

在 system package 中可以看的右上角有一个 Downgrade 的命令，这个将高版本降级到低版本的选项（需要同样将低版本的功能包上传到 RouterOS 的 FTP 的 files 中，或者通过 winbox 将低版本的功能包拖到 files 里）。



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。

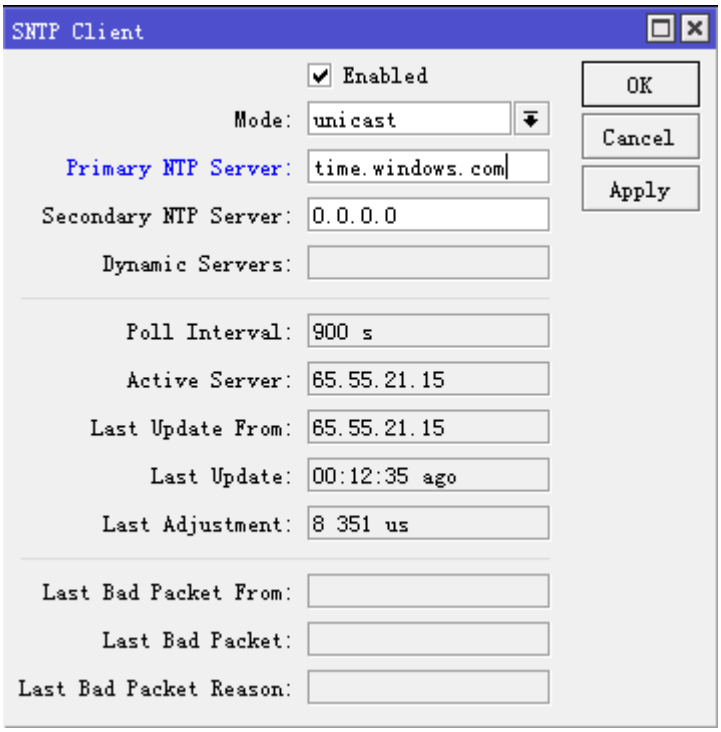
操作路径 **/system ntp client**

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。

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

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

SSH 提供了将命令远程发送的功能，通过""在登录方式加入执行命令

```
/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 RouterOS 常用协议与端口

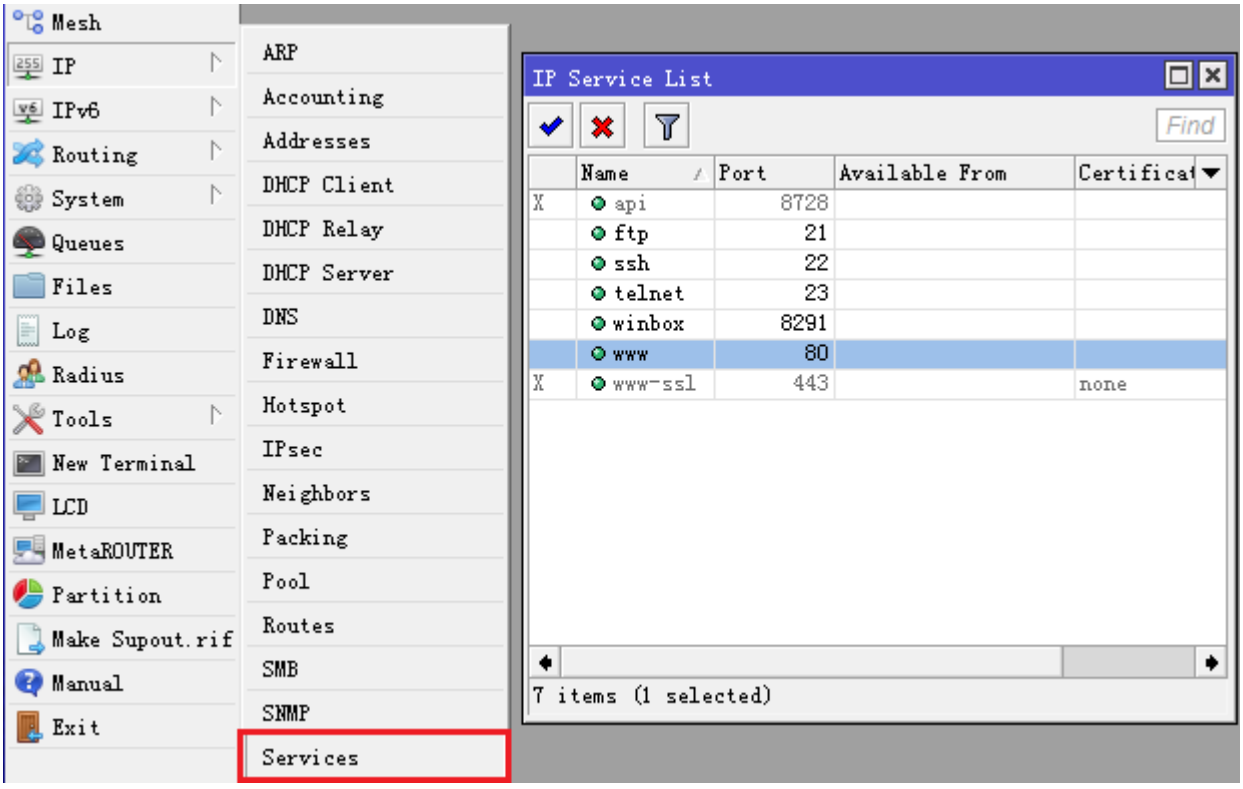
MikroTik RouterOS 提供的接口服务，如常用的 telnet、SSH、WWW、FTP 和 winbox 等，出于安全这些接口允许禁用或修改端口和限制访问 IP 地址。

操作路径: ***/ip service***

属性描述

name - 服务名称
port (*整型*: 1..65535) - 监听的端口
address (*IP 地址 掩码*; 默认: **0.0.0.0/0**) – 限制访问服务的 IP 地址
certificate (*名称*; 默认: **none**) - （对于不需要认证的服务缺省）特定服务所使用的认证名称

通过 winbox 进入 ip service



修改 service 端口

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>
```

常用服务列表

以下是常用的协议和端口的列表，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：

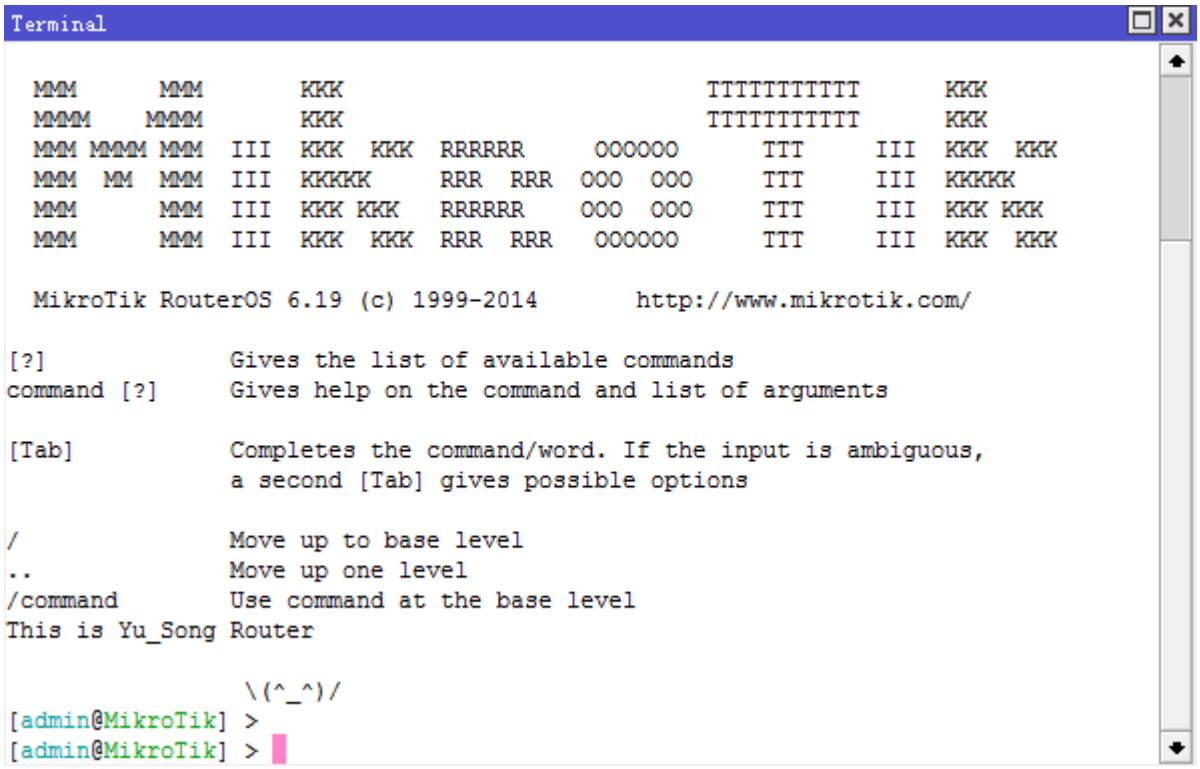
```
This is Yu_Song's Router

                \(^_^)/




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）、撤消（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

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，不同的系统事件和状态信息都能被 RouterOS 的 log 记录下，日志能被存储到本地路由器的内存或者文件中。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，将日志存储到远端的服务器。

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

查看本地日志，如下面可以查看到系统 info 类型的日志，如 admin 从那个 IP，通过什么方式登出设备，admin 修改了什么规则等：

```
[admin@MikroTik] /log> print
15:22:52 system,info device changed by admin
16:16:29 system,info,account user admin logged out from 10.13.13.14 via winbox
16:16:29 system,info,account user admin logged out from 10.13.13.14 via telnet
16:17:16 system,info filter rule added by admin
16:17:34 system,info mangle rule added by admin
16:17:52 system,info simple queue removed by admin
16:18:15 system,info OSPFv2 network added by admin
```

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

action (*名称*; 默认: **memory**) – 用户可选择在**/system logging action** 指定操作的类型
prefix (*文本*) – 本地日志前缀
topics (info | critical | firewall | keepalive | packet | read | timer | write | ddns | hotspot | l2tp | ppp | route | update | account | debug | ike | manager | pppoe | script | warning | async | dhcp | notification | pptp | state | watchdog | bgp | error | ipsec | RADIUS | system | web-proxy | calc | event | isdn | ospf | raw | telephony | wireless | e-mail | gsm | mme | ntp | open | ovpn | pim | radvd | rip | sertcp | ups; 默认: **info**) – 指定日志组或者日志信息类型。

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>
```

为保证日志内被存储下来，并导出用于分析，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，该文件可以通过 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台的 (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** – 日志显示在控制台屏幕上
- **email** – 日志通过 email 发送
- **memory** – 日志被存储到本地内存
- **remote** – 日志发送到远端服务主机

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：

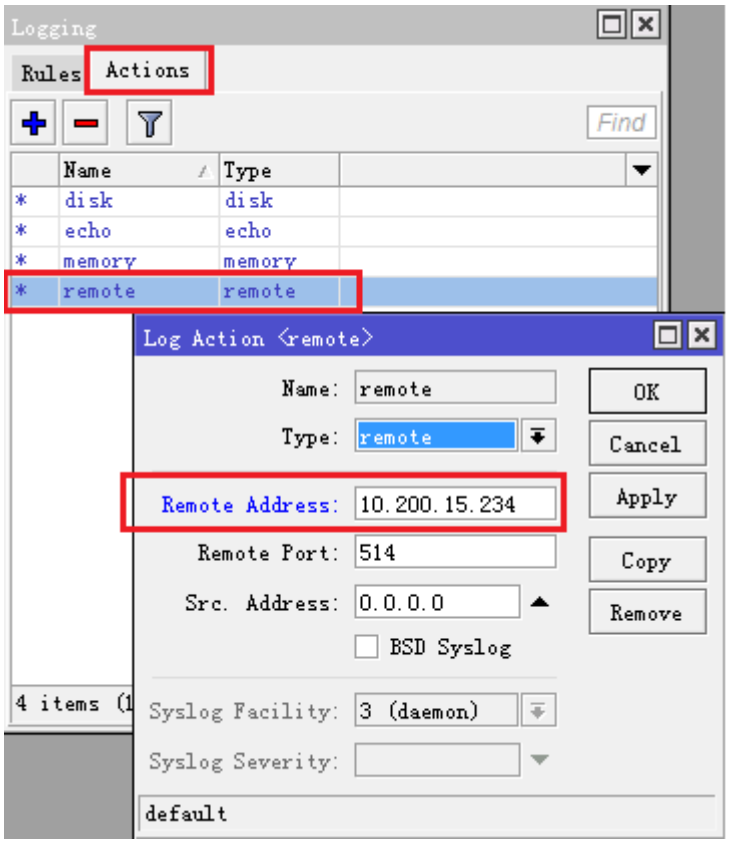
```
[admin@MikroTik] /ip firewall filter>   add chain=forward protocol=tcp dst-port=80
 action=log log-prefix=80port
```

```
[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

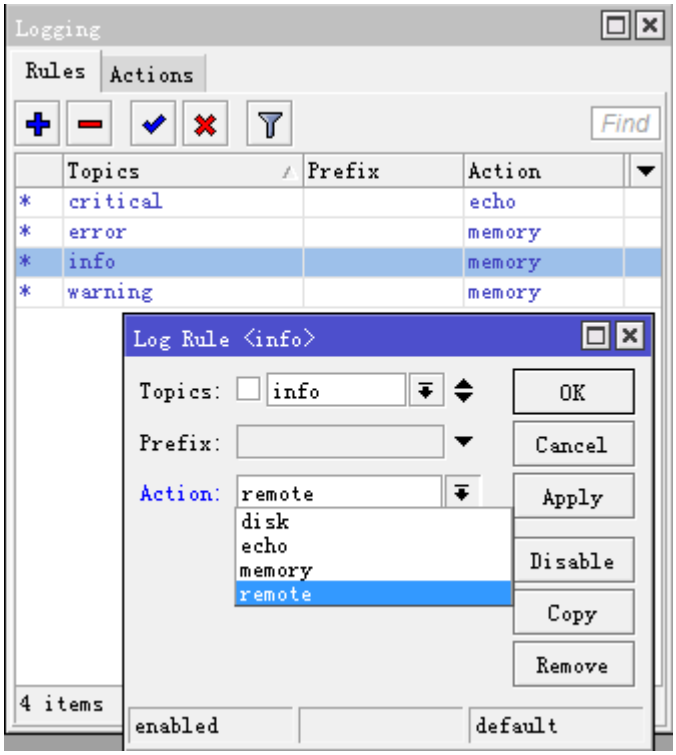
在 MikroTik 提供的 syslog 软件，用于记录 RouterOS 的系统日志信息，但这个软件只能记录 1000 条，不能做长时间记录和定期存储。在新版本的 The Dude 网络管理软件中增加了系统日志记录和下载的功能，这个我们可以通过 The Dude 管理器对我们需要的 RouterOS 日志信息进行记录和管理。

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：

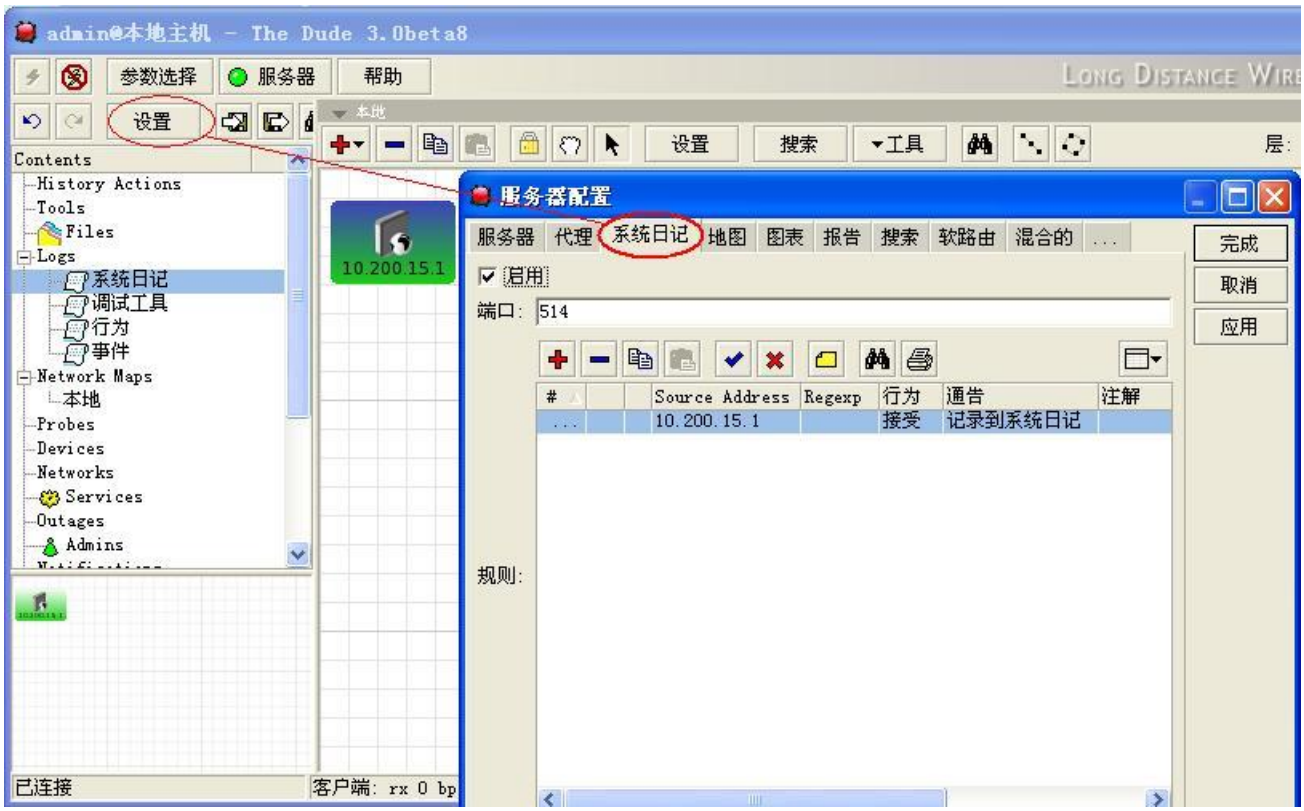


进入 system logging 后配置 actions 里的 remote 参数，将 Remote Address 配置为指定的系统信息接受的 The Dude 服务器 IP 地址。

然后将需要记录的日志信息，设置为 remote：

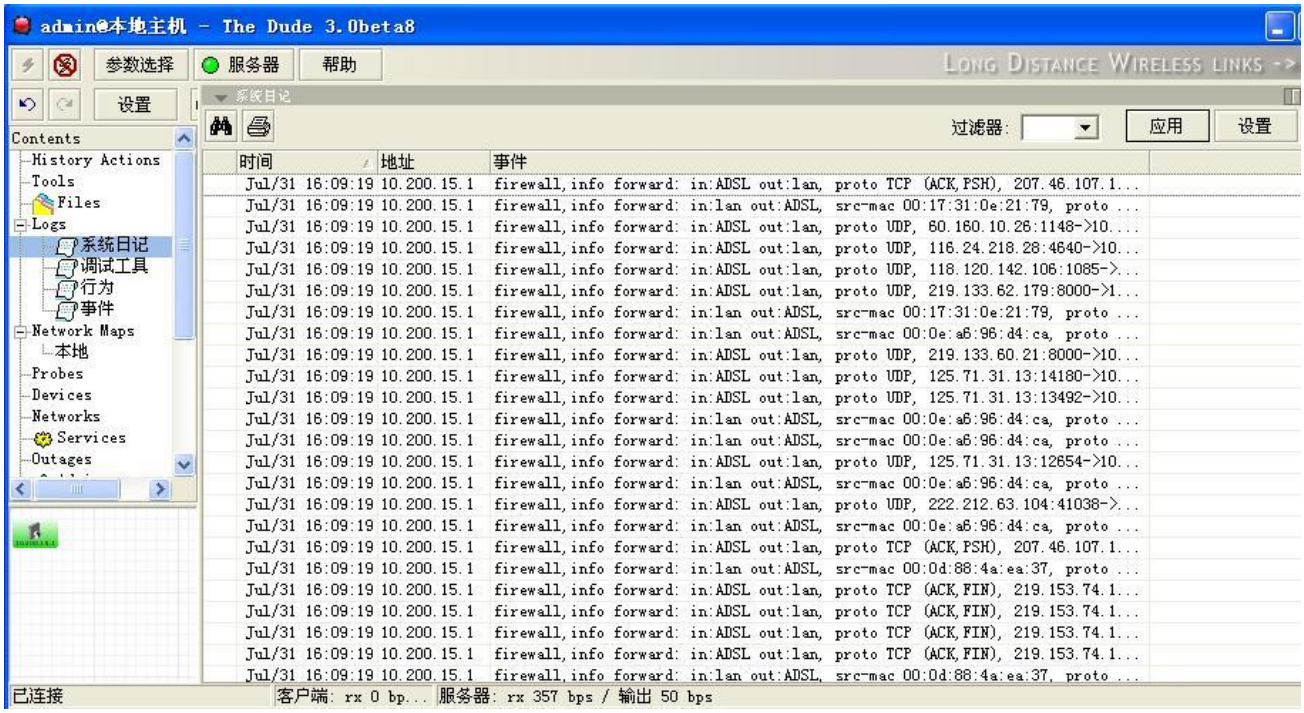


下面是在 10.200.15.234 的 The Dude 网络管理器上配置系统日志信息，进入“设置”，选择“系统日记”，并配置相应的端口和 IP 地址。



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。

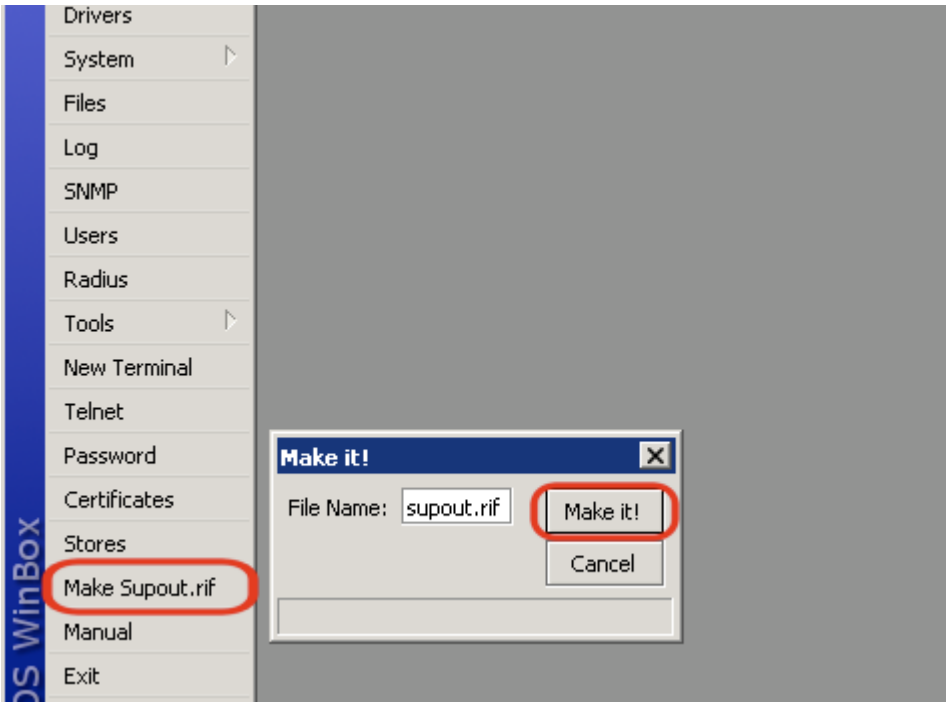
通过 The Dude 网络管理软件，可以方便的记录每台 RouterOS 的日志信息和情况，同时达到监控的目的。关于 The Dude 网络监控软件的功能介绍可以参考《The Dude 中文实用教程 13e》

2.14 Supout.rif 技术支持文件

技术支持文件被用于调试 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文件中（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

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



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

Console 操作

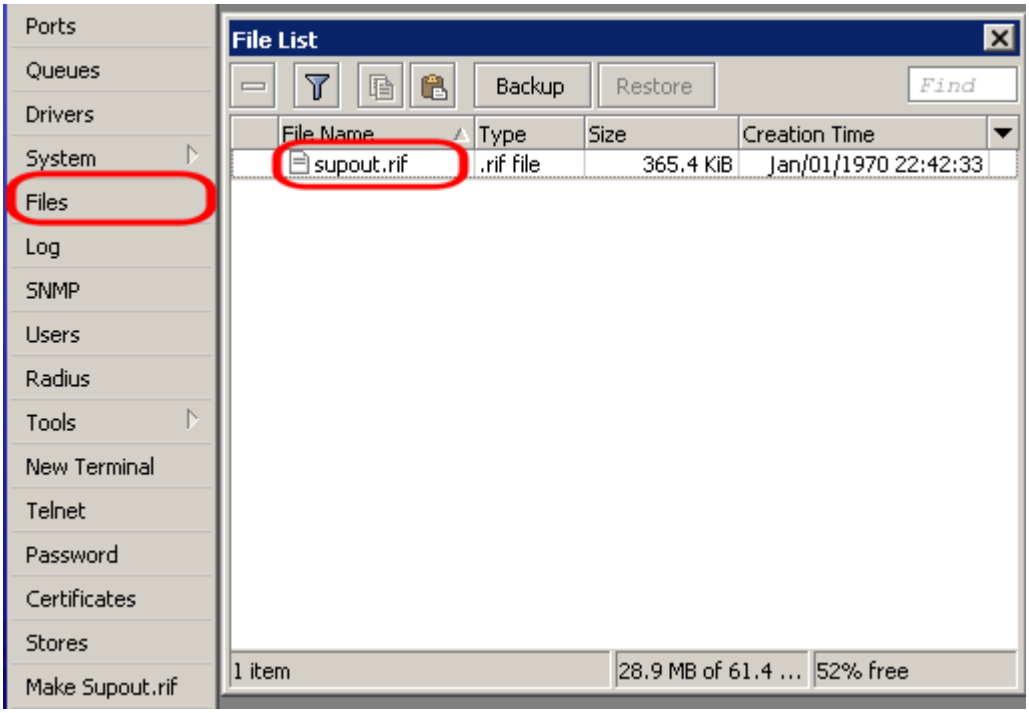
生成 suppout.rif 在 console 下运行命令

```
[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**.

下载技术支持文件

技术支持文件能通过 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

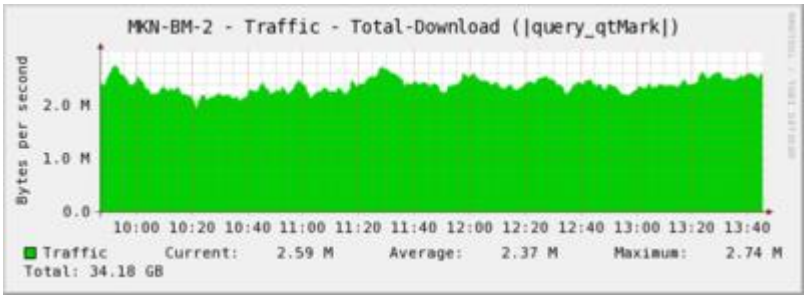


找到文件后，通过 e-mail 的附件，并说明问题发送给 MikroTik 技术支持(support@mikrotik.com)

2.15 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 模板，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

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
```

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符和限制访问 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，注意 winbox 路径是 ip-> snmp

```
[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>
```

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>
```

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
```

第三章 MikroTik RouterBOARD 介绍

RouterBOARD 是 MikroTik 基于 MIPS、PPC 和 Tilera 等平台开发的硬件，采用的是基于 Linux 内核开发的 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 使用的是 JunOS，基于 FreeBSD 开发。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，只是当系统软件开放了后系统就会贬值，现在 Google 的 Android 也是基于 Linux 开发的，但 Android 已经的普及已经改变了智能手机行业。

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

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，但非 SoC 平台，真正的 SoC 平台是从 2006 年的 RB112 开始，RB 已经走过 5 年的时间，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。已经 RB1100、RB1100AH 和双核的 RB1100AH×2，还增加了一款万兆网卡的 RB1200，以及支持 SFP 光模块的 RB 和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，RB700 系列开始逐步淘汰，RB900 系列接替，并在 2014 年下半年推出 802.11ac 产品

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	10×10G	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×	无	802.11bgn

		100M		
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，因此此表仅供参考，具体型号和参数请连接 www.routerboard.com

RB 淘汰和停产的情况

- RB100 系列-淘汰
- RB200 系列-淘汰
- RB/CRD-淘汰
- RB300 系列-淘汰
- RB400 系列：RB411-停产，RB411A-停产，RB411UAHR-停产、RB411R-停产
- RB500 系列-淘汰
- RB600 系列-淘汰
- RB1000 系列：RB1000 –淘汰

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><通道数量>

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

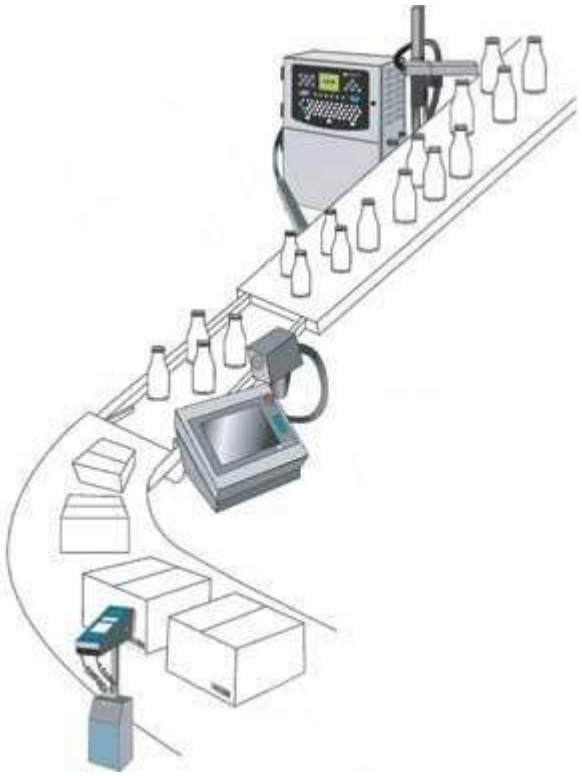
- -<n>G – 千兆以太网端口数量
- -<n>S – SFP 端口数量
- -<n>S+ - SFP+端口数量

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

3.3 RouterBOARD Throughput（吞吐量）

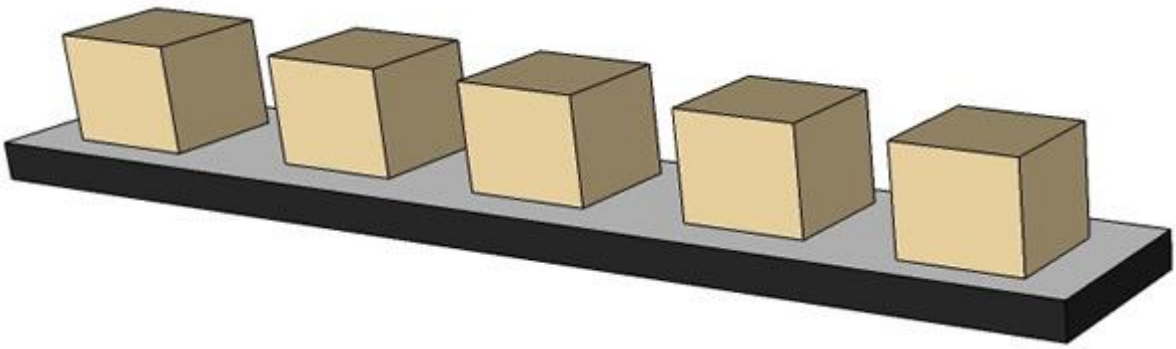
路由器 Throughput，一般是以太网到另一以太网的转发性能，数据流出或流入需要路由器处理，这代表了路由器性能。RouterBOARD 仅给出了桥接和路由下开启和关闭防火墙的结果，并没有给出真正的 nat 测试结果，这些仅作参考。

首先我们要了解什么是 throughput，我们知道一般数据包处理都要经过 CPU，而 CPU 的处理能力和速度直接决定转发数据包的能力，处理数据包就像流水线一样，对每个包进行拆包、分析、重新封装和转发

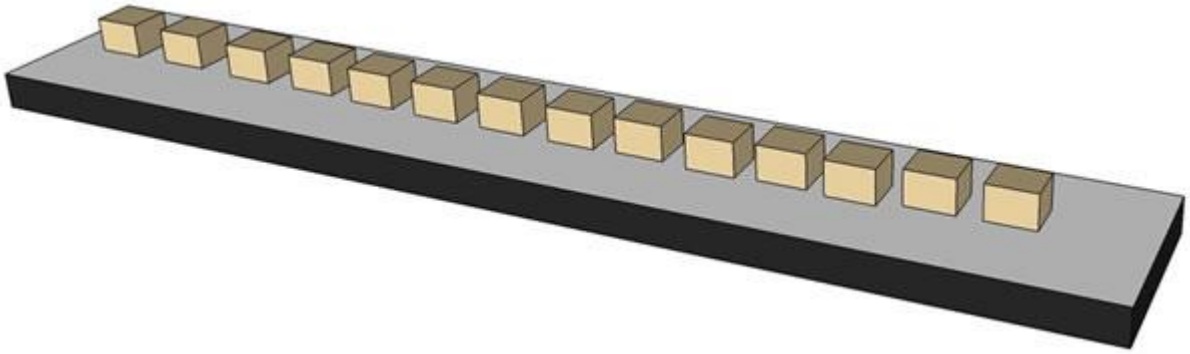


数据包的大小也决定了处理速度和吞吐量、如 128Byte 包每秒能处理 10000 个，并不能做到 64 Byte 包每秒处理 20000 个，而是只比 10000 个略多一点点，比如 10100 个。如他的路由器在处理最大的 1518Byte 包时每秒 8000 个，根据理论计算处理 1518Byte 包 100M 线速的极限值是 8127 个，所以折算出来的 Throughput 就是 100M*8000/8127=98.44M，有些厂家就很自豪地宣布，我的路由器 Throughput 高达 98.44M，殊不知，原来这个路由器在处理最小的 64Byte 包时每秒是 11000 个，根据理论计算处理 64Byte 包 100M 线速的极限值是 148810 个，所以折算出来的 Throughput 只有 100M*11000/148810=7.39M，两者相差 13 倍多。

为什么数据包大小能影响路由器的吞吐量能力，我们举一个例子，假设一个员工在流水线上检验每个产品的包装是否合格，第一天流水线上是 4 个产品装成 1 个大箱子的包装，他一分钟能检查 20 个大箱包装，即一分钟 80 个产品的包装就检验完成，但事实上他只检查了 20 个大箱子的包装，并非 80 个产品的每个包装。



结果第二天工厂要求每个产品的包装都要检查，但他每分钟仍然只能检查 20 个产品的包装，反而比昨天检验 80 个产品产品少了 4 倍，和昨天的 4 个产品一个大包装完全不一样，导致今天只有 20 个产品，是因为包装方式完全不一样。如果要让这个员工检验每个产品的包装提高到 80 个，只能是提升员工的检验水平，要让员工提高到每分钟 80 个产品的检验，肯定这个员工肯定要疯掉。



这个道理和我们的路由器处理大包和小包的道理是一样的，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。我们衡量路由器的吞吐量是以 64byte 的小包，在每秒钟转发了多少个包，单位是 pps（per packet seconds）

参照思科官方 Cisco 3745 两个百兆以太网端口在 64 字节时达到 225018pps 的转发速率，即 225kpps。RB1100AH 在 1333MHz 的频率下路由模式 262kpps，桥接模式下 400kpps

RouterBOARD 测试结果又以下几种：

- 64byte 包吞吐量反应了 CPU 的性能
- 1500byte 包吞吐量反应了内存性能
- 512byte 包反应了 CPU 和内存整体性能

RouterBOARD 最大以太网吞吐量，对照表可以在下面的链接找到：
http://www.routerboard.com/pdf/routerboard_performance_tests.pdf

所有测试是通过以下方式完成：

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

根据官方提供的参数提供一些的性能图像分析，图 1 是早期 RouterBOARD 在 64byte 数据包，使用桥接模式下的对比图

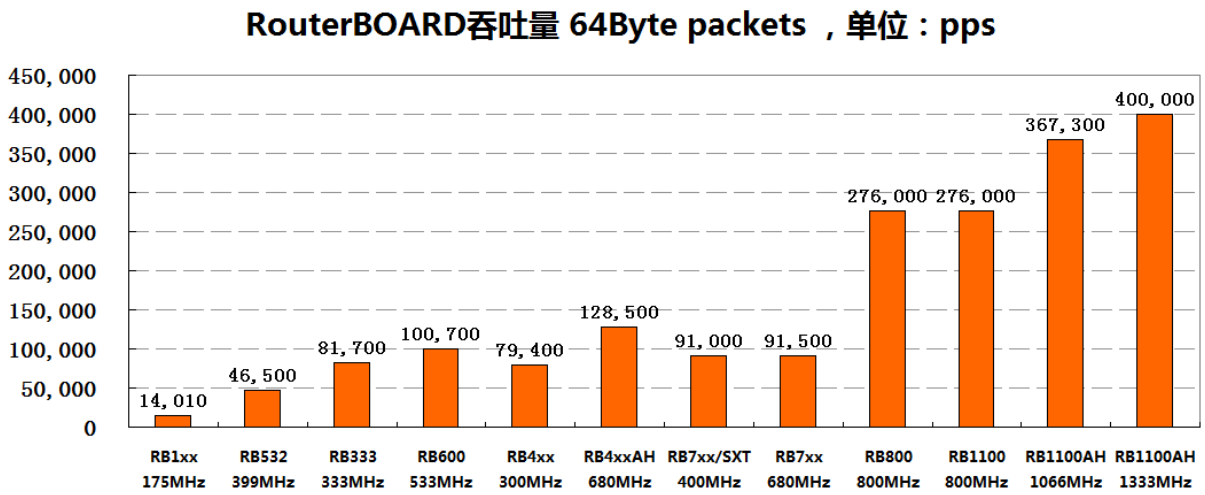


图 1

下面图 2 则是近年来新的 RB 和 CCR 等产品的对比，这里是采用 RouterOS 6.0 版本，支持 Fast path 功能，所以相同产品性能上有较大提升，可以从图 1 的 RB4xxAH（680MHz）做对比几代产品的差别，注意单位是 **kpps**

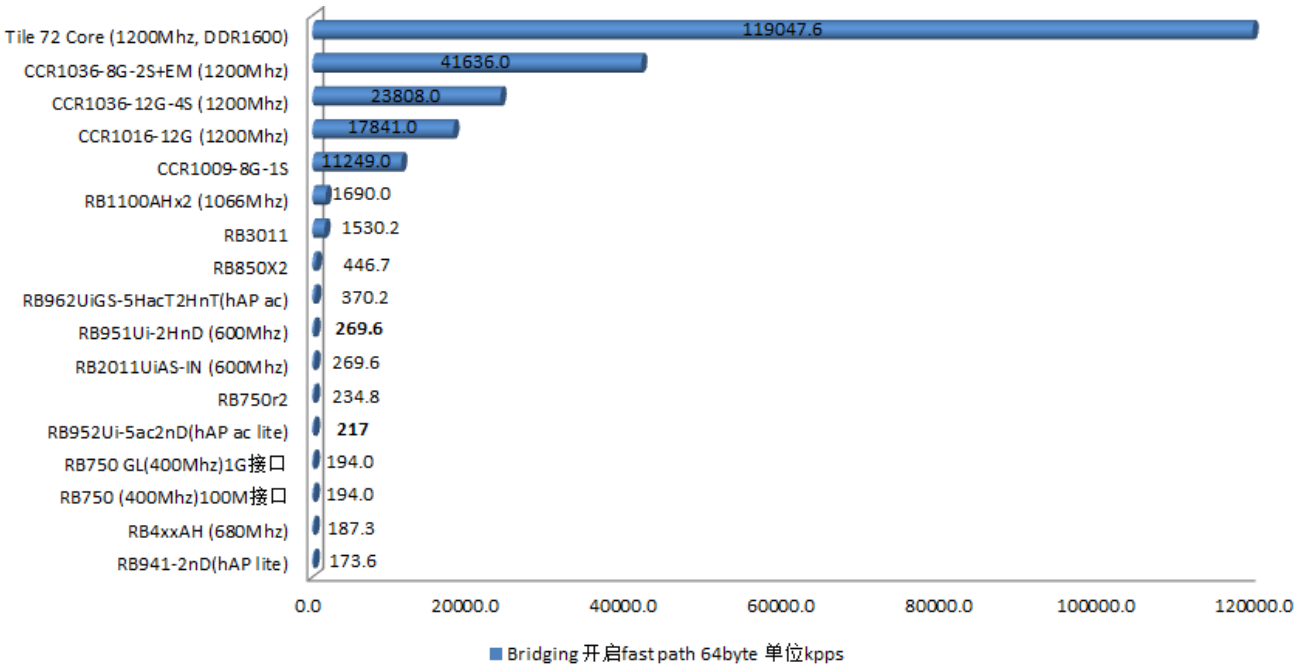


图 2

从上图可以看到 CCR 系列的性能与 RB1100AH×2 差距是非常明显的，由于构架的不同带来数十倍的提升，以上数据供大家参考。

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 台电脑的网络环境，RB450G 表现就比较好，如果不建议 CPU 较高，可以支持 180 台电脑的网络，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。

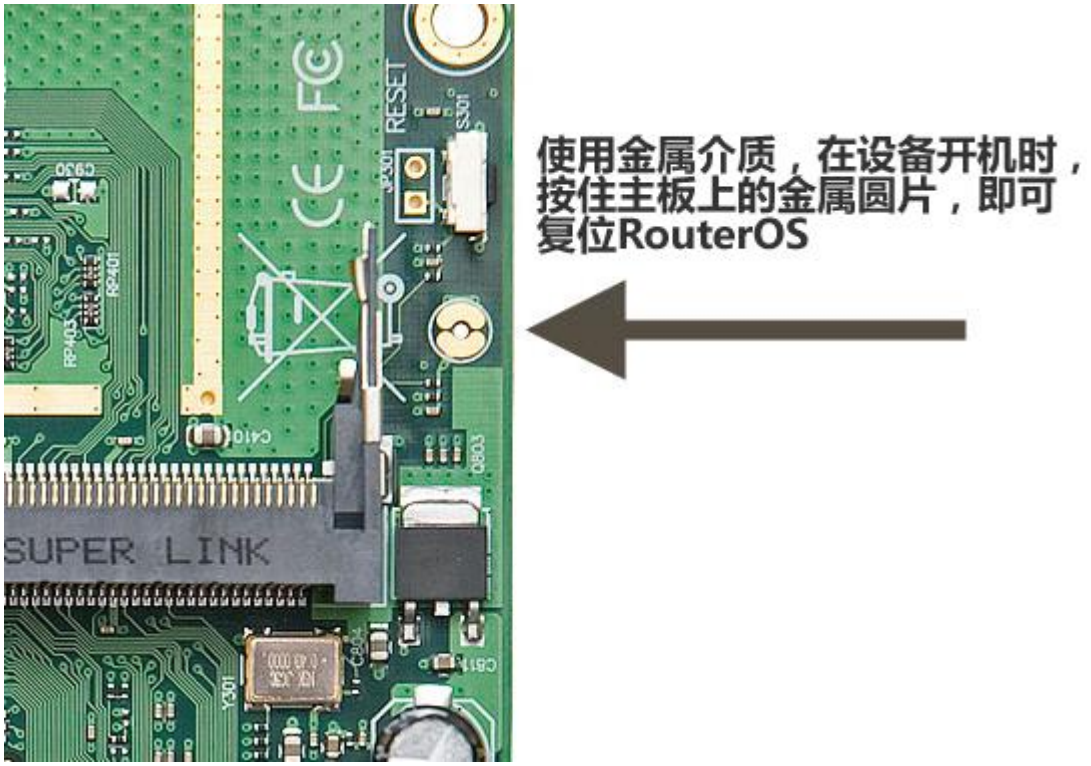
RB1200 的 CPU 和 RB1100AH 相同，只是采用了 10 个万兆网卡的解决方案，看似网卡吞吐量大了，但设备性能没有任何提升

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。

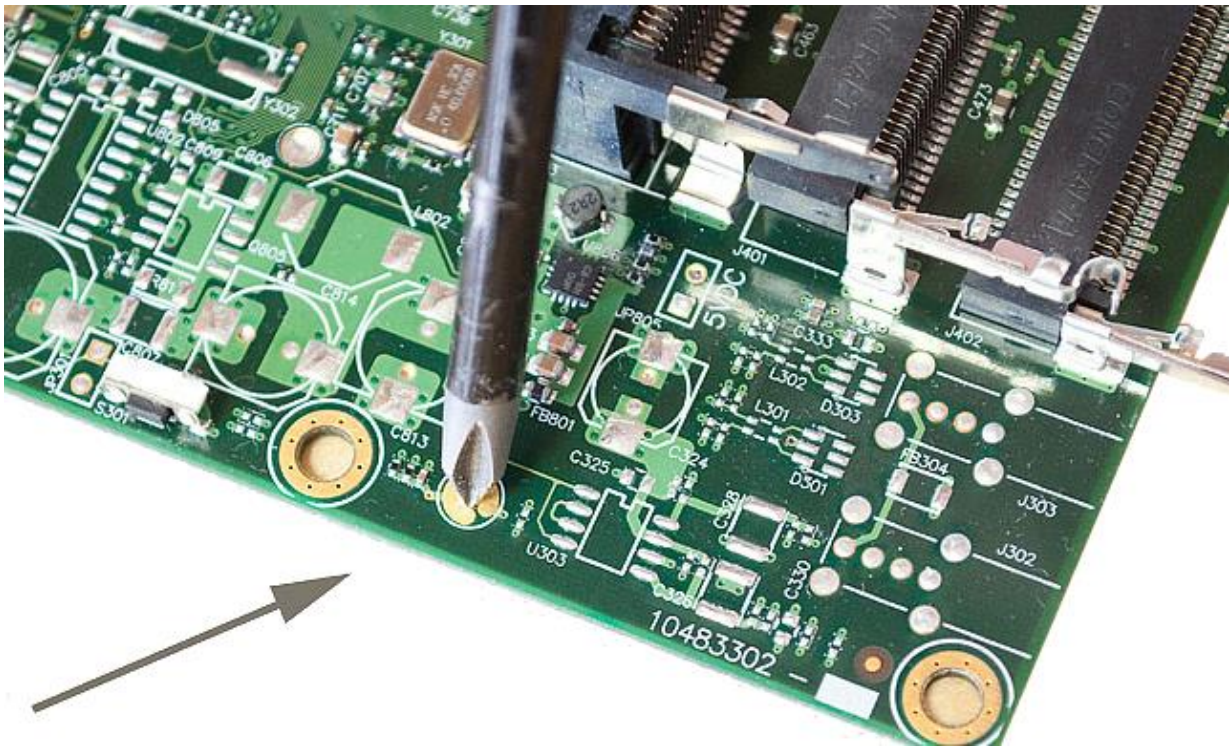
3.5 RouterBOARD 复位

RouterBOARD 是硬件化的 RouterOS，当 RouterBOARD 的 RouterOS 密码丢失或配置错误后，一般只能通过 Netinstall 重装 RouterOS 系统或者使用 RouterBOARD 的复位跳针（或复位孔）。你只需要在设备启动时短路接口，直到启动完成。

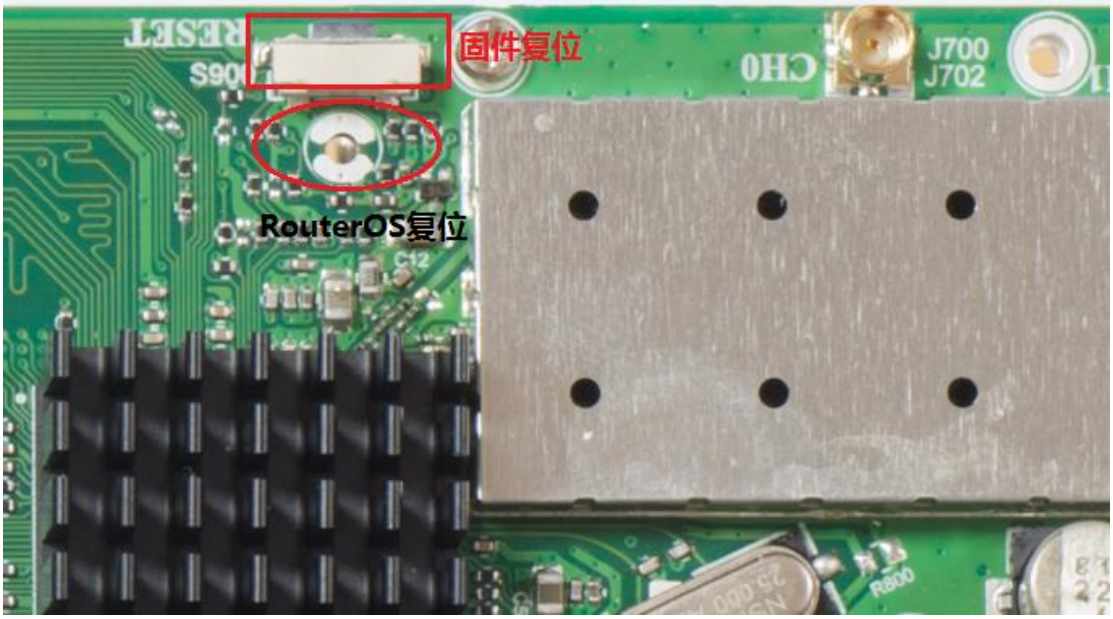
如下图是 RB411 主板的金属复位片。**注意：**旁边白色的按钮，则是固件复位按钮，而非 RouterOS 的复位按钮。



下面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。

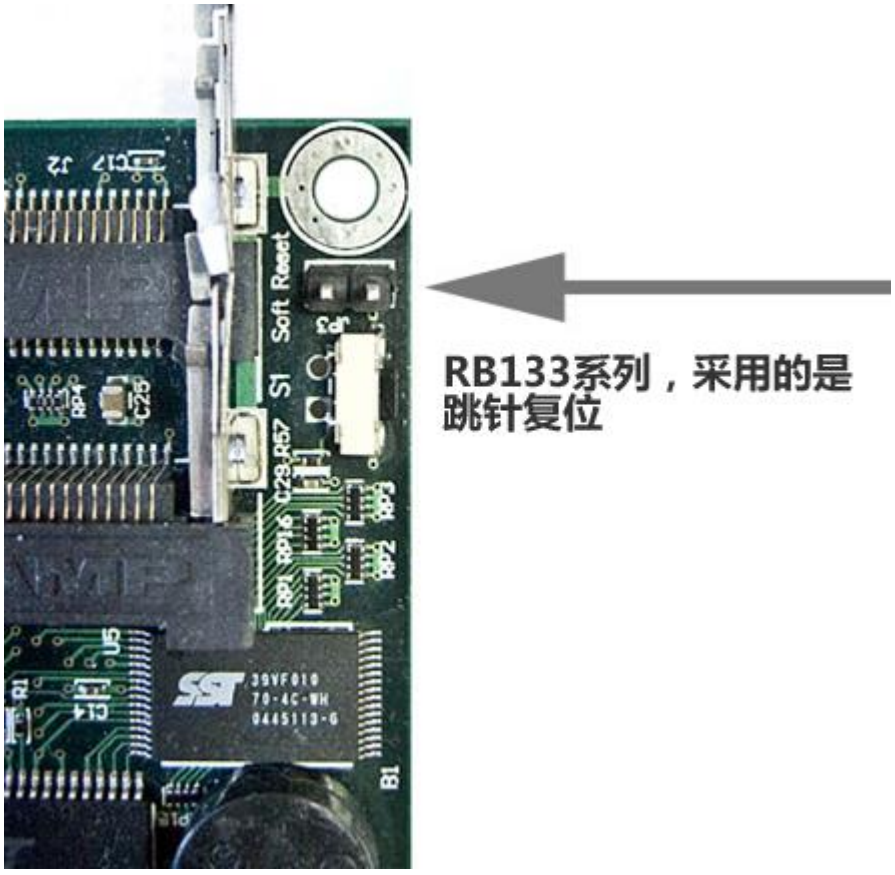


下面是 RB2011USA-2HnD 的复位按钮，靠外侧的是固件复位按钮（启动时长按固件复位按钮，可以通过 ethernet 方式引导），后面圆形的是 RouterOS 复位接口（在启动前短路这个复位接口，直到启动完成，RouterOS 软件就会复位）。



早期型号复位

下面的图片是 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

如上图，RB751U 的固件复位按钮接口（不同 RB 设备的固件复位按钮不同），启动时长按，直到 netinstall 找到并显示设备信息，便可以进行安装复位，操作和之前一样。

3.6 升级 RouterBOARD 固件

RouterBOARD 产品启动 BOIS 程序都有更新，对 RouterBOARD 系列的启动引导和硬件兼容性进行修正和更新。RouterBOARD 固件后缀名为.fwf，每个系列的 RouterBOARD 所使用的固件都不相同，固件下载地址可以到 www.routerboard.com，如下面的列表：

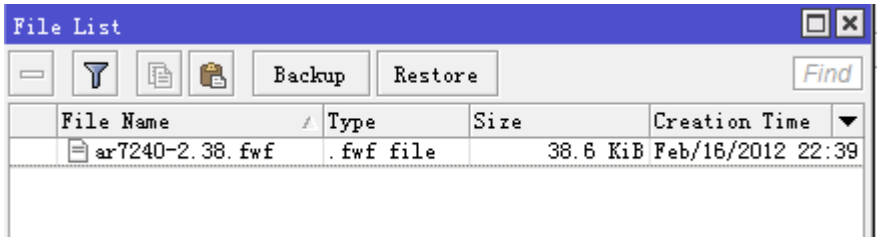
RB 型号	固件前缀
CCR1016，CCR1036	-
RB1000，RB1100，RB1100AH，RB1100AHx2，RB1200	mpc8548
RB2011 系列	ar9344
RB900 系列	ar9344
RB800	mpc8343
RB600	mpc8343
RB333	mpc8323
RB400 系列(411/A/AH、433/AH、433AH、450/G、493/AH ）：	ar7100
RB700 系列（750、750G、711、751）包括 SXT，Groove	ar7100
RB532	rc32434
RB100 系列（112、133/C、150、192）	adm5120

每隔一段时间，RouterBOARD 的固件都会更新一次，所以通过在 RouterOS 中操作更新最新的 RouterBOARD 固件，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 file list），当前的 RouterBOARD 是 RB450，所以这里我们上传的 ar7240 的固件文件如下图：



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，才可以更新。

注：你也可以通过网络更新固件，要求 RouterBOARD 能连接到网络，会自动检测最新的固件，如果有新固件发布，也可以直接通过 upgrade 升级，RouterBOARD 也可以通过网络升级下载升级固件。

3.7 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 载入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

3.8 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：

功能	QCA8337	Atheros8316	Atheros8327	Atheros8227	Atheros7240	ICPlus175D	Other
端口交换	支持	支持	支持	支持	支持	支持	支持
端口镜像	支持	支持	支持	支持	支持	支持	无
主机数	2048 条	2048 条	2048 条	1024 条	2048 条	无	无
Vlan 数	4096 条	4096 条	4096 条	4096 条	16 条	无	无
规则数	92 条	32 条	92 条	无	无	无	无

QCA8337 包含：

- **RB3011**（ether1-ether5，ether6-ether10）
- **RB750Gr2 (hEX)**（ether1-ether5）
- **RB962UiGS-5HacT2HnT(hAP ac)**（ether1-ether5）

Atheros8316 包含：

- **RB493G**(ether1+ether6～ether9, ether2～ether5),
- **RB1200**(ether1～ether5),
- **RB450G**(所有接口，ether1 可选项),
- **RB435G**(所有接口，ether1 可选项),
- **RB750G** (ether1～ether5)
- **RB1100**(ether1～ether5, ether6～ether10).

Atheros8327 包含：

- **RB2011 系列**(ether1～ether5+sfp1)
- **RB750GL**(ether1～ether5)
- **RB751G-2HnD**(ether1～ether5)
- **RB951G-2HnD**(ether1～ether5)
- **RB1100AH**(ether1～ether5, ether6～ether10)
- **RB1100AHx2**(ether1～ether5, ether6～ether10).

Atheros8227 包含：

- **RB2011 系列**(ether6～ether10).

Atheros7240 包含：

- **RB750**(ether2～ether5),
- **RB750UP**(ether2～ether5),
- **RB751U-2HnD**(ether2～ether5)
- **RB951-2n**(ether2～ether5)

ICPlus175D 包含：

- 后期 **RB450**(ether2～ether5)
- 后期 **RB433 系列**(ether2～ether3).

ICPlus175C

- 早期 **RB450**(ether2～ether5)
- 早期 **RB433 系列**(ether2～ether3).

ICPlus178C 包含

- **RB493 系列**(ether2～ether9)

命令行下的操作路径/interface ethernet switch ，该菜单下列出了系统中所有的交换芯片：

```
[admin@MikroTik] /interface ethernet switch> print
Flags: I - invalid
 #    NAME      TYPE              MIRROR-SOURCE    MIRROR-TARGET
 0    switch1   Atheros-8316 ether2               none
```

根据交换芯片类型，获得各自的配置功能和参数

3.9 RouterBOARD Switch 配置

随着 RouterOS 4.0 发布后，RouterBOARD 系列路由产品开始支持以太网口的硬件交换，如 RB450、RB750、RB433、RB493 和 RB1100 等路由器，他们拥有 3-11 个以太网口，他们的都可以配置 switch 硬件交换口，即数据通过二层转发，不在经过 RouterOS 路由软件处理，完全和交换机转发相同。Switch 功能在 3.0 以上的软件版本被支持。

基本原理

交换功能让交换组内地端口实现线速转发，就像以太网交换机一样。配置这个功能要求把所需端口指定到同一个“master port”，一个 master 端口将把流量传递给 RouterOS，RouterOS 通过连接 master 端口与所有交换组接口通信。

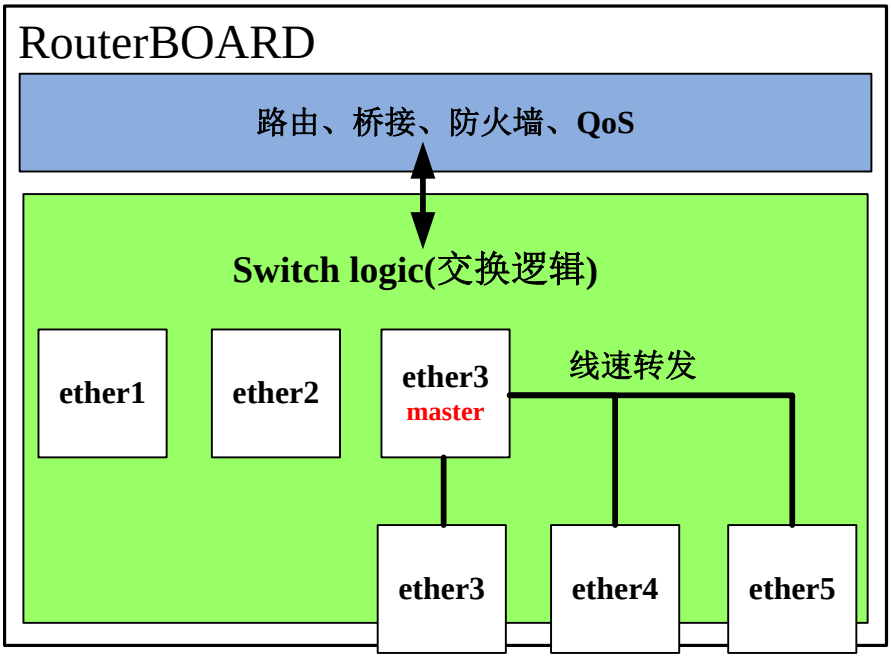
例如，下面是一个 5 给以太网接口的 RouterBOARD

```
[admin@MikroTik] > interface ethernet print
Flags: X - disabled, R - running, S - slave
 #     NAME        MTU   MAC-ADDRESS          ARP          MASTER-PORT        SWITCH
 0 R   ether1      1500  00:0C:42:3E:5D:BB enabled
 1     ether2      1500  00:0C:42:3E:5D:BC enabled      none               switch1
 2     ether3      1500  00:0C:42:3E:5D:BD enabled      none               switch1
 3     ether4      1500  00:0C:42:3E:5D:BE enabled      none               switch1
 4 R   ether5      1500  00:0C:42:3E:5D:BF enabled      none               switch1
```

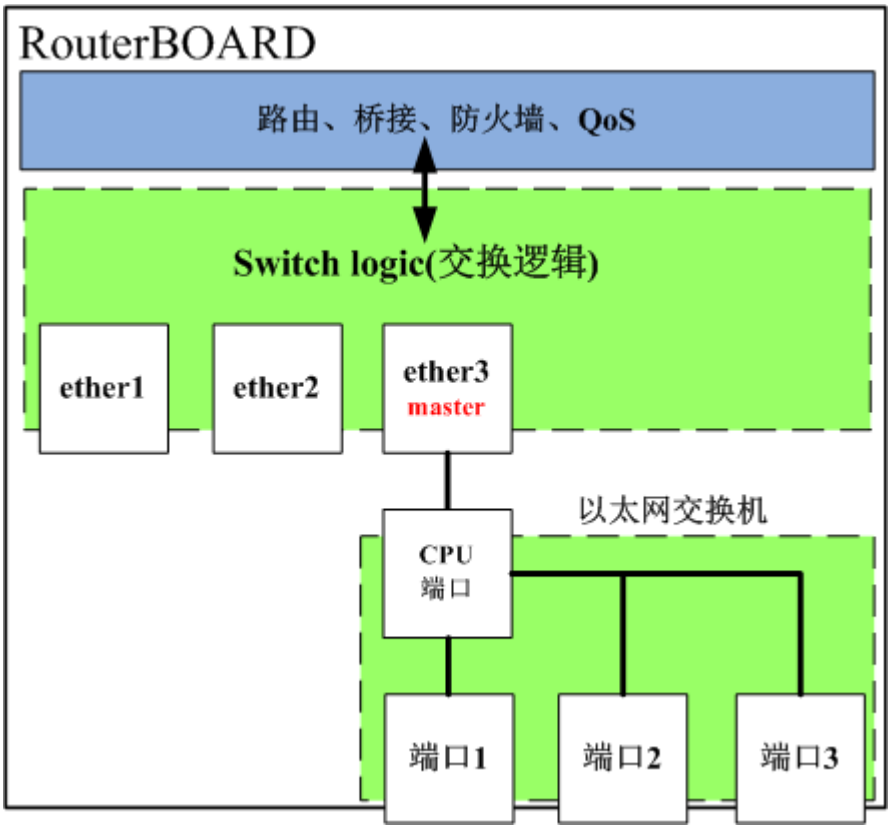
我们配置 3 个交换端口，包含 ether3、ether4 和 ether5:

```
[admin@MikroTik] /interface ethernet> set ether4,ether5 master-port=ether3
[admin@MikroTik] /interface ethernet> print
Flags: X - disabled, R - running, S - slave
 #     NAME        MTU   MAC-ADDRESS          ARP          MASTER-PORT        SWITCH
 0 R   ether1      1500  00:0C:42:3E:5D:BB enabled
 1     ether2      1500  00:0C:42:3E:5D:BC enabled      none               switch1
 2 R   ether3      1500  00:0C:42:3E:5D:BD enabled      none               switch1
 3   S ether4      1500  00:0C:42:3E:5D:BE enabled      ether3             switch1
 4 RS  ether5      1500  00:0C:42:3E:5D:BF enabled      ether3             switch1
```

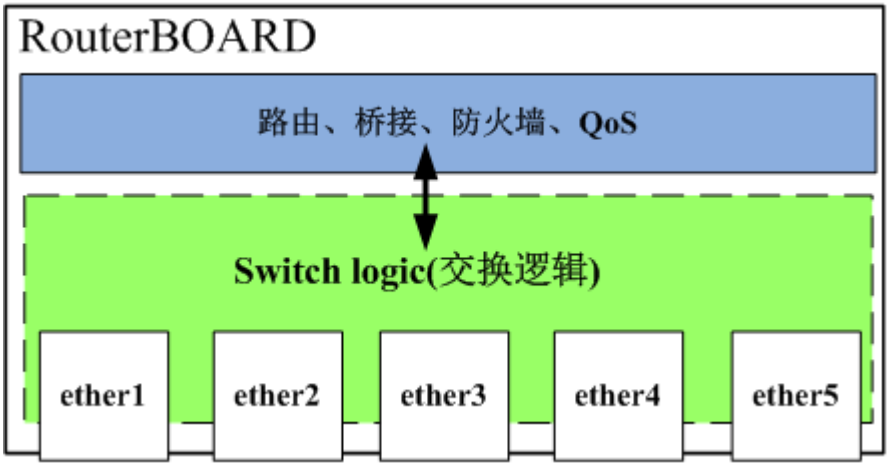
ether3 现在是这个组的 master 端口。注意：之前 RouterOS 会将相关数据传给 ether4 和 ether5 接口，但现在 ether3 被标记为 master 端口，所有到 ethter4 和 ether5 的数据都会传给 ether3。



事实上这个配置类似于 RouterBOARD 有 3 个以太网接口，通过 ether3 连接到交换芯片，这样在这个配置中出现了 4 个端口，可以理解为，并不是 ether3 作为 master，而是 ether3 连接到了交换芯片端口，将从属的接口组成了一个以太网交换机：



下面这台 RouterBOARD 是 5 端口，看看 5 端口的交换情况：

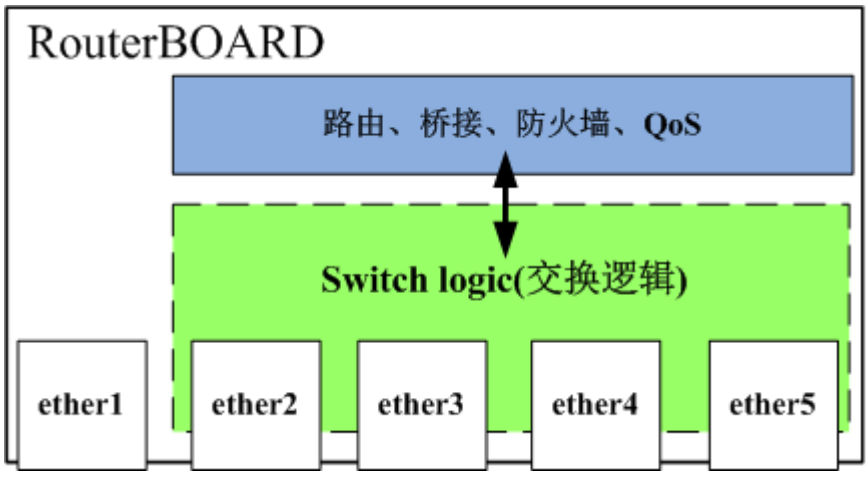


这里能看到，当一个端口接收到一个数据包，首先会传递给交换逻辑处理。交换逻辑决定数据包去哪一个端口。传递数据包是向 RouterOS，同样也可以被称为传递给交换芯片（cpu port）。即交换转发数据包给 cpu port，数据包开始被 RouterOS 某个接口处理进入的数据。当数据包不必通过交换逻辑发送到 cpu port 处理，也就不占用任何的 CPU 时钟周期，这样所有帧转发实现线速转发。

以 RB450G 为例，ether1 有一个功能允许添加或删除默认交换逻辑组。默认 ether1 被包含着交换逻辑组中。这个配置可以被修改，进入/interface ethernet switch，配置命令 set switch1 switch-all-ports=no

- **switch-all-ports**=yes/no

"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(主机列表)

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。每条动态条目设置 5 分钟的超时。

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，请确认端口是否指定了“master-port”，当动态 MAC 地址存在于列表中，你可以将存在于列表中的 MAC 添加为静态条目 ，当添加静态条目后，可以获得更多的功能属性控制数据。

- copy-to-cpu=yes/no – 数据被克隆，并发送到 CPU 端口
- redirect-to-cpu=yes/no – 数据包被重定向到 CPU 端口
- mirror=yes/no – 数据被克隆，并发送到镜像端口
- drop=yes/no – 丢弃来自某个端口的数据包 MAC 地址

Vlan Table(VLAN 列表)

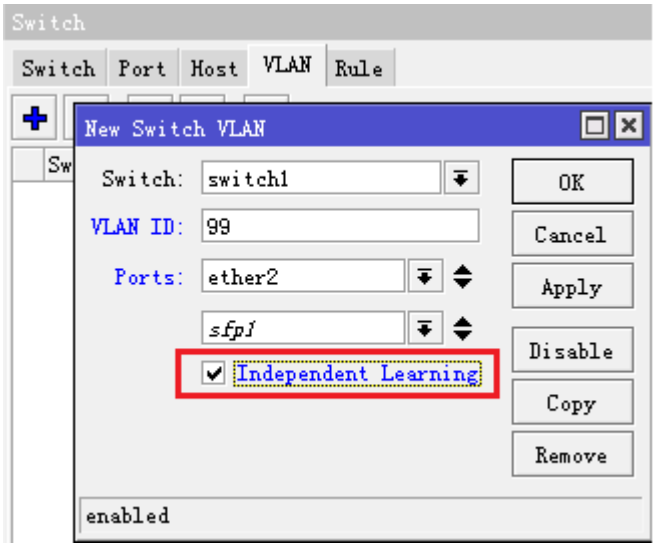
Vlan 列表为带有 802.1q 的数据包指定转发规则。这些规则优先级高于交换组配置的’master-port’属性。列表包含了相应的 vlan 标记 id 到交换组的一个或多个端口上，因此带有 vlan 标记的包，会根据相应的设置被调度到指定到交换组的一个或多个端口，即通过精确的逻辑控制带有 vlan 标记的包通过 vlan 模式进行端口间的调度。

修改每个交换组端口的 vlan 参数通过 **/interface ethernet switch port**

Vlan 模式可以获得一些参数：

- **disabled** – 忽略 VLAN 表，视作数据包没有做 vlan 标记。
- **fallback** – 默认模式，处理数据包时，VLAN 标签将不会出现在 VLAN 表中，视作数据包没有 VLAN 标签。数据包的 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。drop packets with vlan tag that is not present in vlan table. Packets with vlan tags that are present in vlan table, but incoming port does not match any port in vlan table entry does not get dropped.
- **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。

基于 Vlan 标签 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 标签模式在出端口(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：

- **leave-as-is –** 数据包在出端口保持不变
- **always-strip –** 如果 VLAN 包头在数据包出现，会被删除掉，即取消标签（access）
- **add-if-missing –** 如果 VLAN 包头没有出现，会添加到数据包中，即打上标签（trunk）

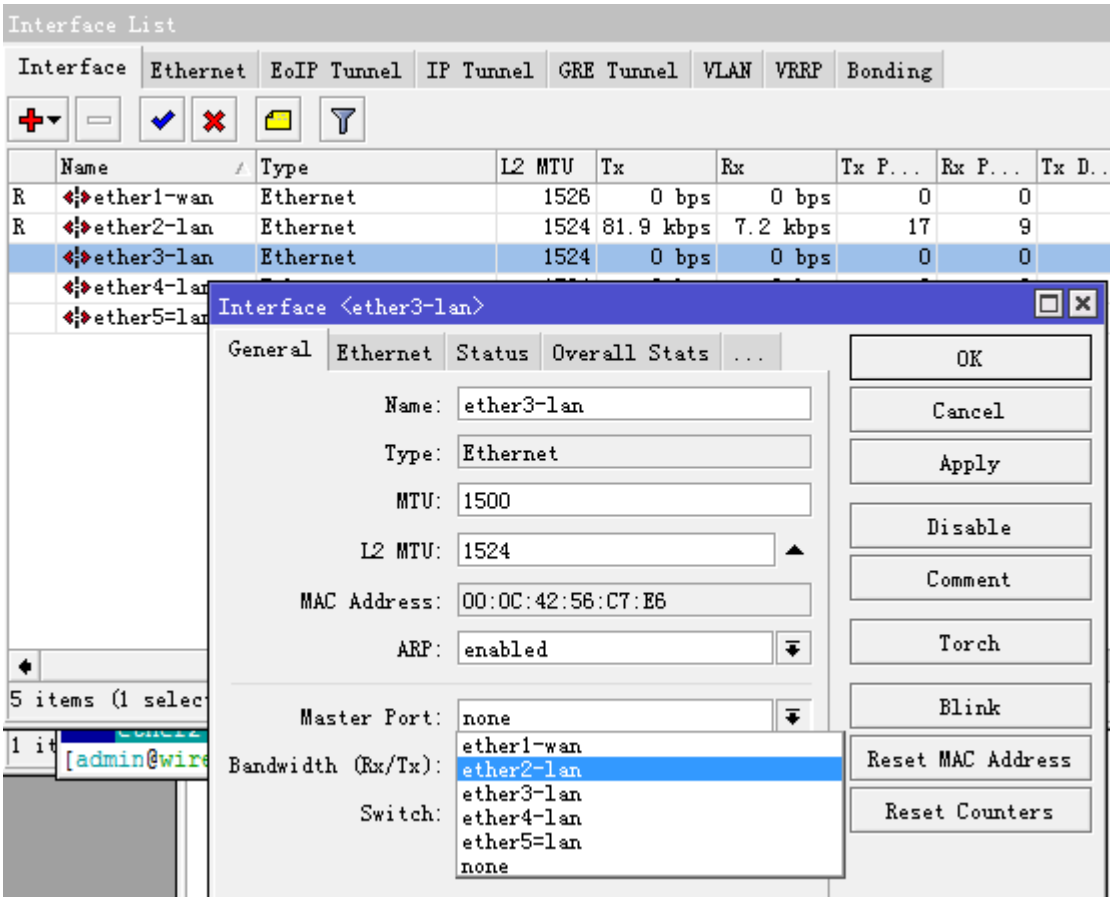
Rule Table（策略列表）

Rule 表是非常强大的工具，允许全线速处理二层、三层、四层的过滤、转发和 VLAN 标记，每条规则包含一个条件部分和执行部分

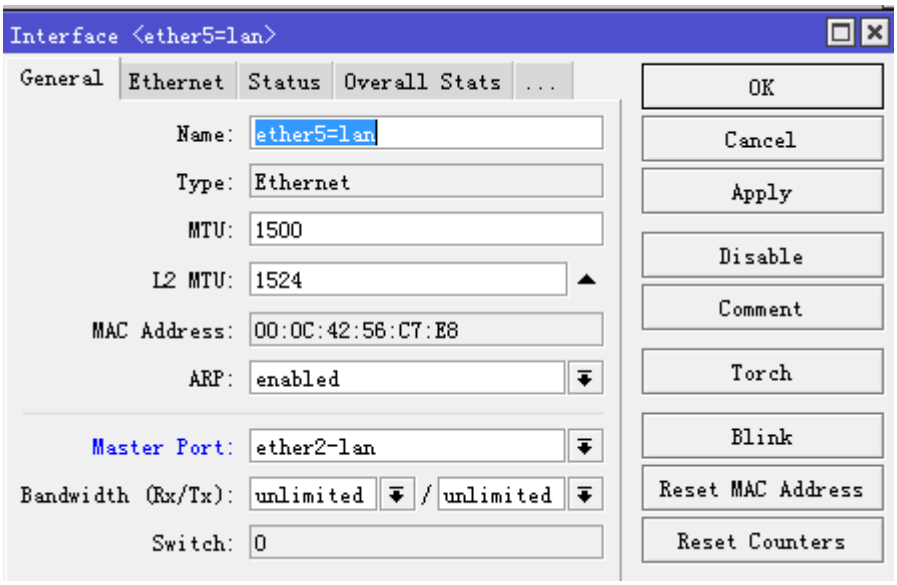
IPv4 和 IPv6 指定条件时不能同时出现在相同规则中。策略列表的策略类似于/ip firewall filter 一样的顺序执行。由于策略表受到交换芯片硬件性能的限制，规则数量根据条件的不同，如 MAC 层、IPv4、IPv6 和四层传输等受到限制。例如 Atheros8316 交换芯片支持 8-32 条可用规则，而 Atheros8327/QCA8337 交换芯片支持 24-96 条，你可以通过/interface Ethernet switch rule print 命令查看最后规则是否出现”invalid”提示，表示规则不能加入交换芯片中

交换配置实例

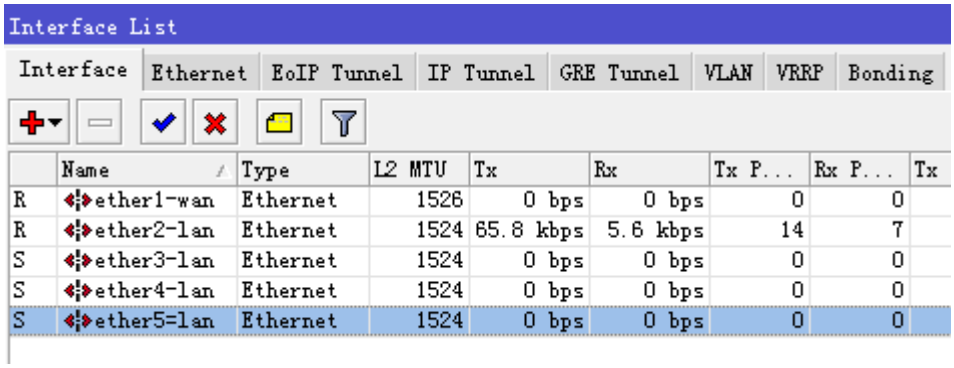
下面我们用 RB750 为例，RB750 一共有 5 个以太网口，分别为 ether1 到 ether5，我们将 ether1 设置为 wan 口，ether2 到 ether3 为 lan 口，我们将 ether2 到 ether3 设置 switch 交换口。设置硬件交换，需要将 1 个网口设置为主端口（Master port），其他口为从端口（Slave port），我们已 ether2-lan 为 Master，其他网口为从端口。我们就只需要配置 ether3 到 ether5 的参数，配置 ether2-lan 接口：



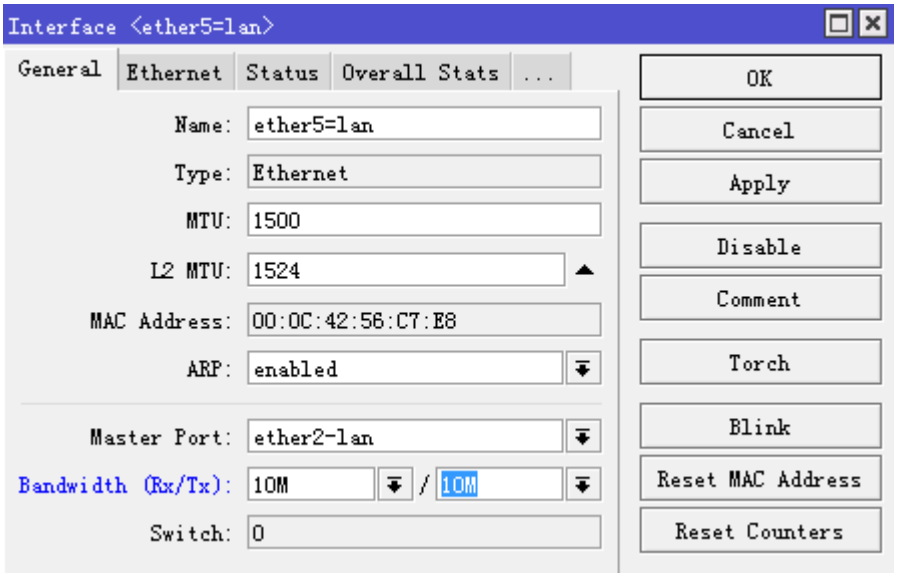
配置 ether5 接口



这样 ether2、ether3 和 ether4 和 ether5 设置为 switch 交换口，4 个口可以多到数据的硬件转发，他们在 interface 前缀都有一个“S”，如下图



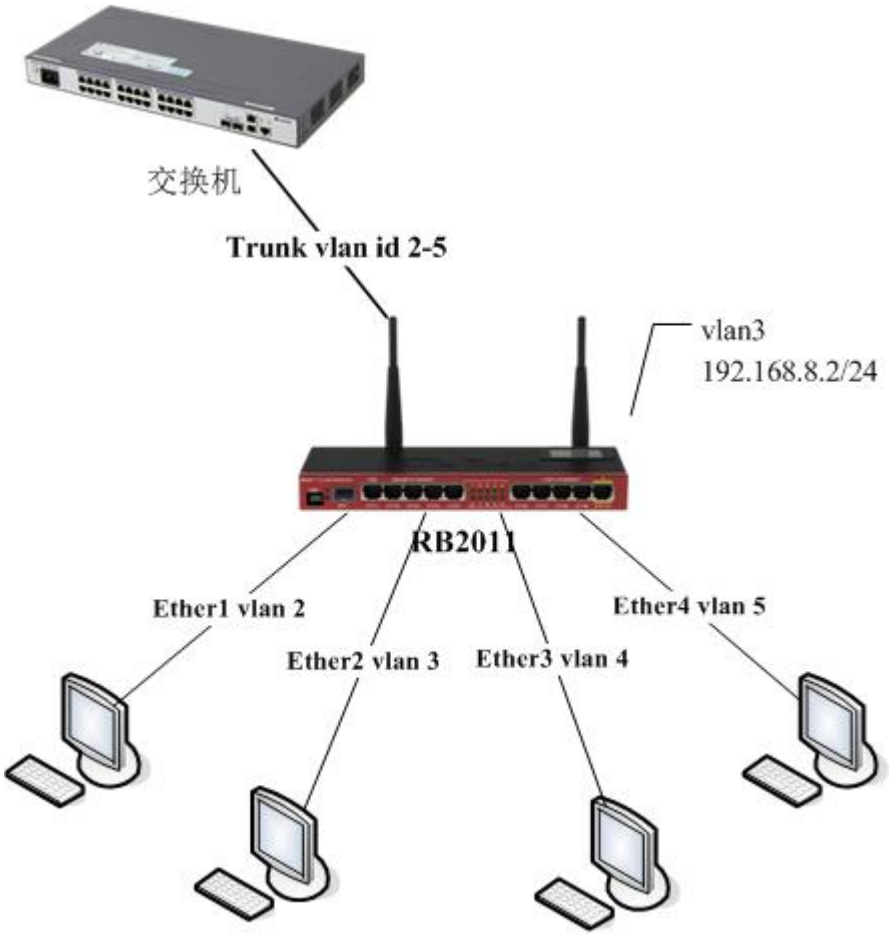
同时可以通过 interface 中的 Bandwidth 设置每个端口的带宽，如我们可以将 ether5-lan 设置硬件转发带宽为 10Mbps



3.10 基于 Atheros 交换芯片的 802.1Q VLAN 配置

该配置是基于 RouterOS v6 版本讲解，RouterBOARD 系列（非 CRS 系列）采用 Atheros 交换芯片可以实现 802.1Q 的 VLAN 配置，该功能支持采用 QCA8337, AR8316, AR8327, AR8227 和 AR7240 交换芯片。这样的配置与 RouterOS interface 下的 Bridge 和 VLAN 配置有别，基于 Atheros 交换芯片是将二层协议交给交换芯片处理，而基于 interface 下的 bridge 和 VLAN 配置则是将二层协议交由路由器的 CPU 处理，因此两者在效率和性能上是有差别。一个不占用 CPU 资源，一个消耗 CPU 资源。

下面我们以 RB2011Ui 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



首先将各个接口配置到交换组，即设置 sfp1 为 master-port

```
/interface ethernet
set name=ether1 master-port=sfp1
set name=ether2 master-port=sfp1
set name=ether3 master-port=sfp1
set name=ether4 master-port=sfp1
```

设置 VLAN 模式到各个接口，"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（打上标签），其他 ether1-ether4 为 access 口（去标签）

```
/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
```

添加 VLAN 列表条目，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 independent-learning=no ports=sfp1,ether1 switch=switch1 vlan-id=2 independent-learning=yes
add independent-learning=no ports=sfp1,ether2 switch=switch1 vlan-id=3 independent-learning=yes
add independent-learning=no ports=sfp1,ether3 switch=switch1 vlan-id=4 independent-learning=yes
add independent-learning=no 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 的对应序列号为#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界面添加。

```
[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
```

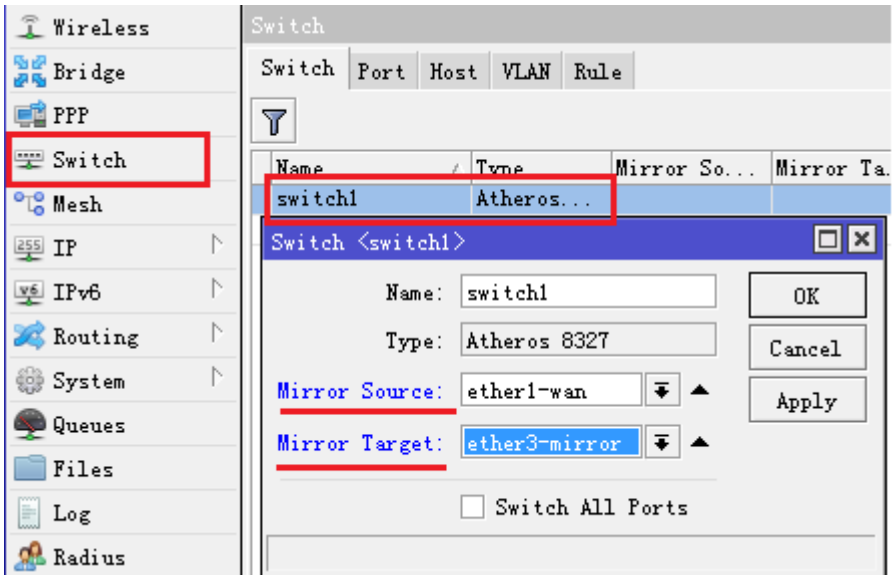
3.11 RouterBOARD 端口镜像

RB400、RB700、RB900 和 RB2011 系列在支持端口镜像功能，该功能在 3.26 后被引入，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），并发送一个复制数据到某一个端口（mirror-target）。

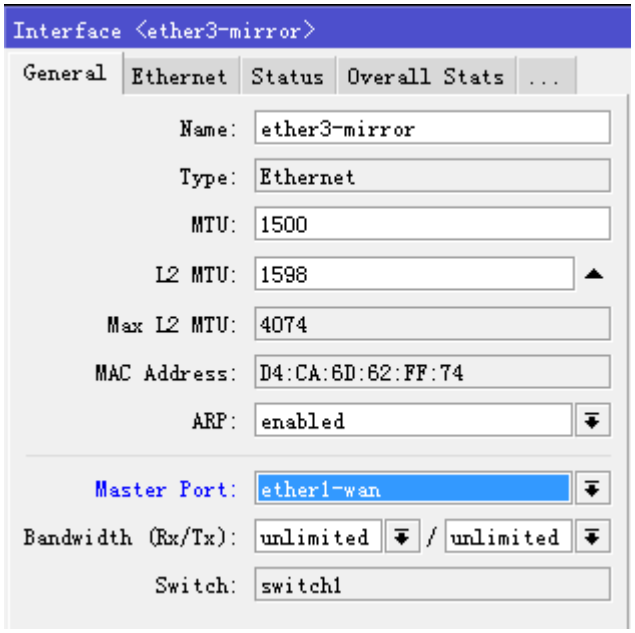
这个功能可以轻松的配置一台网络监测设备接收所有指定端口的传输流量。注意：mirror-source 和 mirror-target 端口必须属于同一交换配置下。（查看那个端口属于哪个交换配置在“/interface Ethernet switch port”目录下），同样 mirror-target 能设定特殊的“CPU”值，意思是“sniffed”数据包被发送至 CPU 端口。

端口镜像配置

例如:下面的配置，我们将 ether1-wan 接口的所有数据镜像到 ether3-mirror 接口上， mirror source 为镜像源接口，mirror target 为镜像目标接口



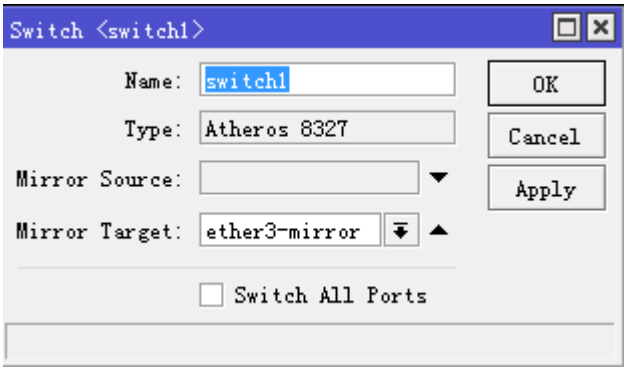
注意：ether1 和 ether3 要实现镜像，必须配置到一组交换上，即 ether3 的 Master Port 是 ether1



以上配置为全镜像配置，即 ether1 的所有数据都镜像到 ether3，但 RouterOS 为我们提供了镜像规则，指定哪些 MAC、IP 或协议镜像到指定端口。

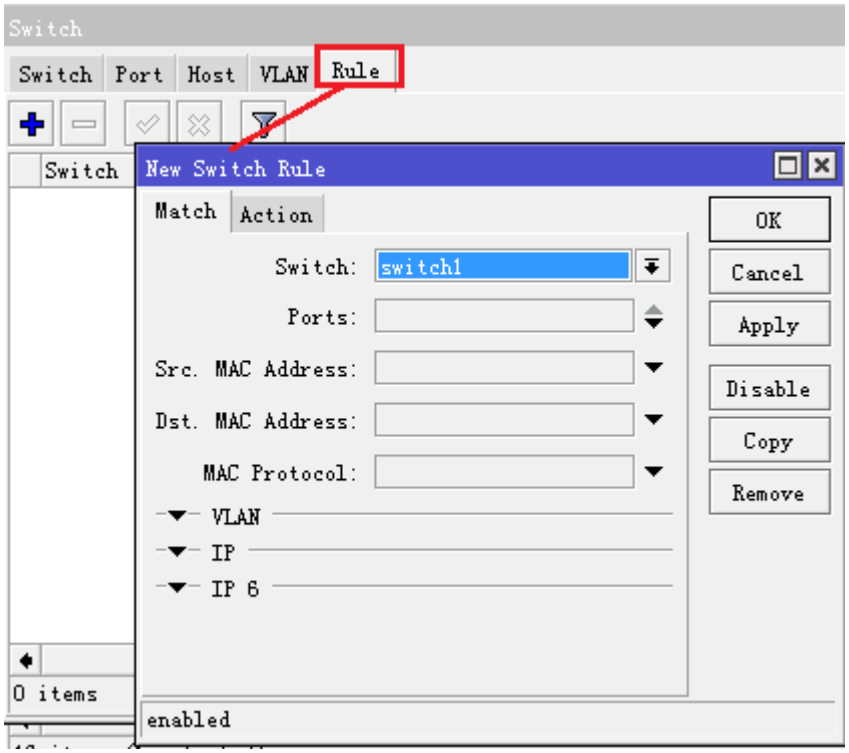
Rule 指定镜像内容

一般我们理解的镜像数据都是全镜像，但如果具备 Switch 功能的 RouterBOARD 可以指定需要镜像的内容到，指定接口，这里选择 rule 表。但在定义之前，修改之前的镜像配置，即只指定 Mirror Target 接口为 ether3-mirror，源接口不再指定，我们只需要在规则中配置。



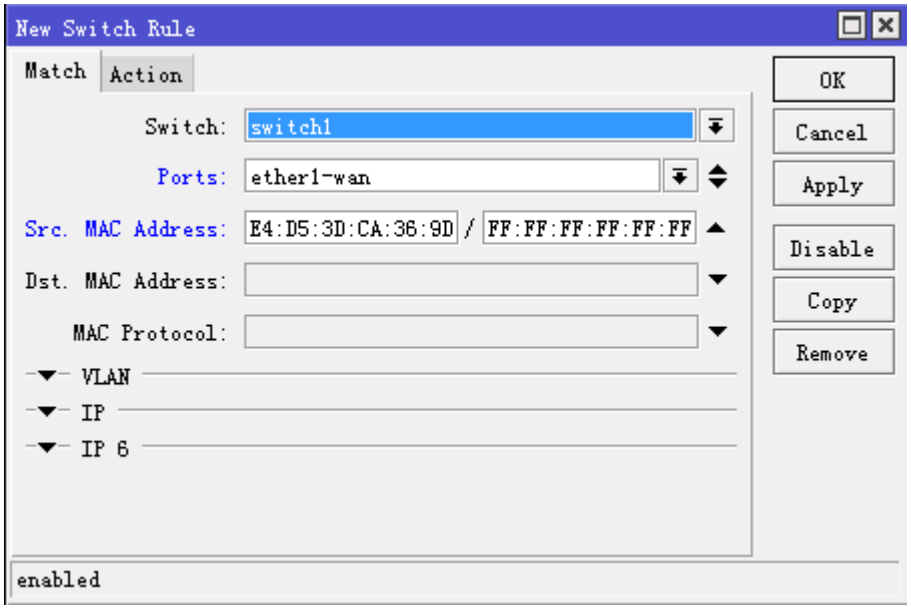
1、镜像 MAC

在 switch 菜单下，我们可以看到 rule 规则菜单，选择后点击添加

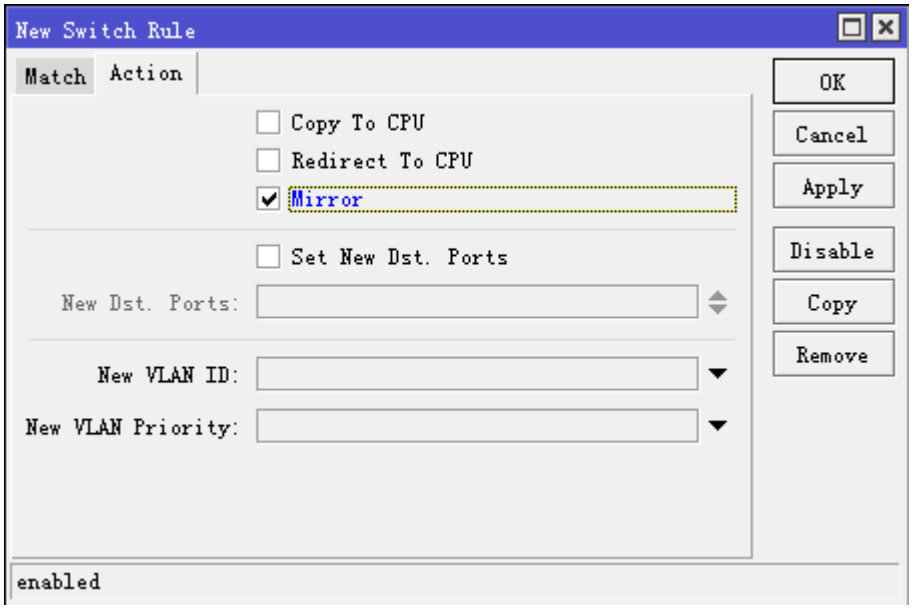


在规则中，我们可以选择接口、MAC、MAC 协议、VLAN、IP 和 IPv6 等。

下面我们将 ether1 接口上源 MAC 地址为 E4:D5:3D:CA:36:9D，镜像到 ether3 上



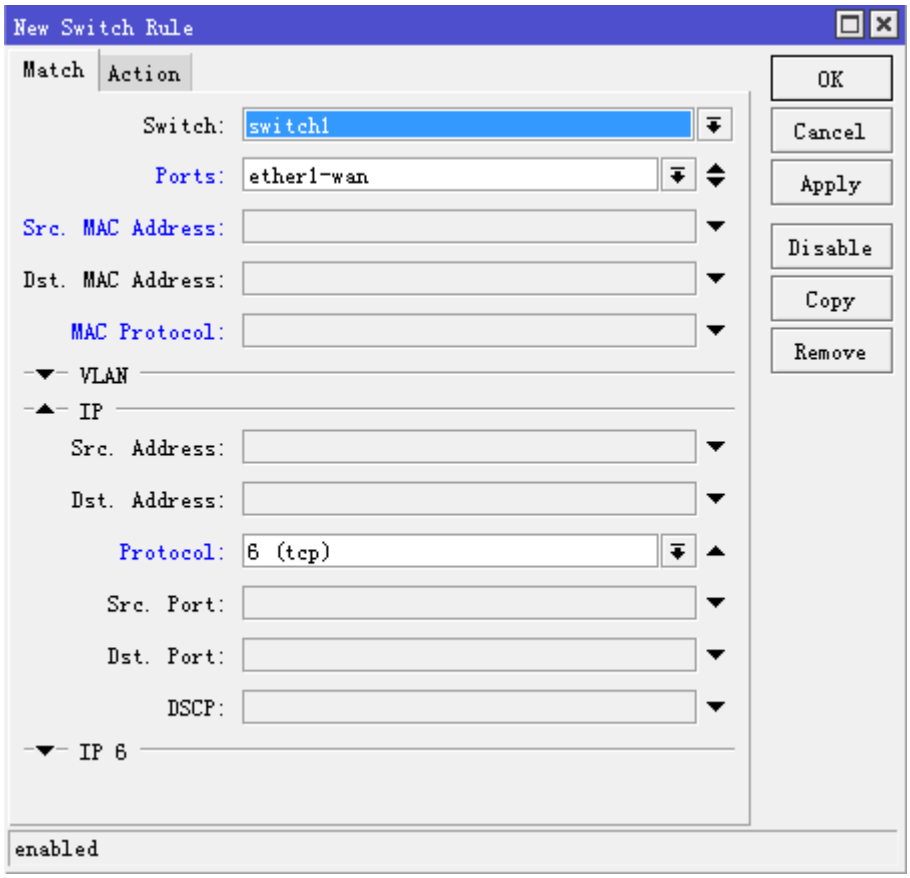
选择 action 菜单下的 Mirror，即镜像到之前我们定义的 Mirror Target 接口 ether3 上



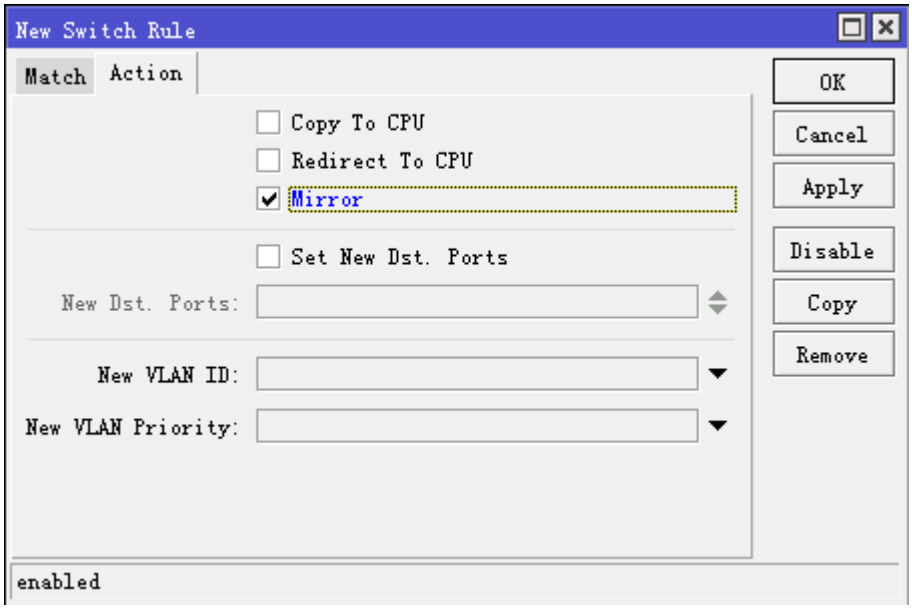
这样源 MAC 为 E4:D5:3D:CA:36:9D 的所有数据被镜像到 ether3 接口，剩下的就是你通过网络设备接收分析数据

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 二层交换介绍

Cloud Router Switch 系列是采用 MIPS CPU，并集成交换芯片，CRS 交换机能应用到各种以太网络，包括 VLAN、二层网络管理和无线/有线统一数据包处理等

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。

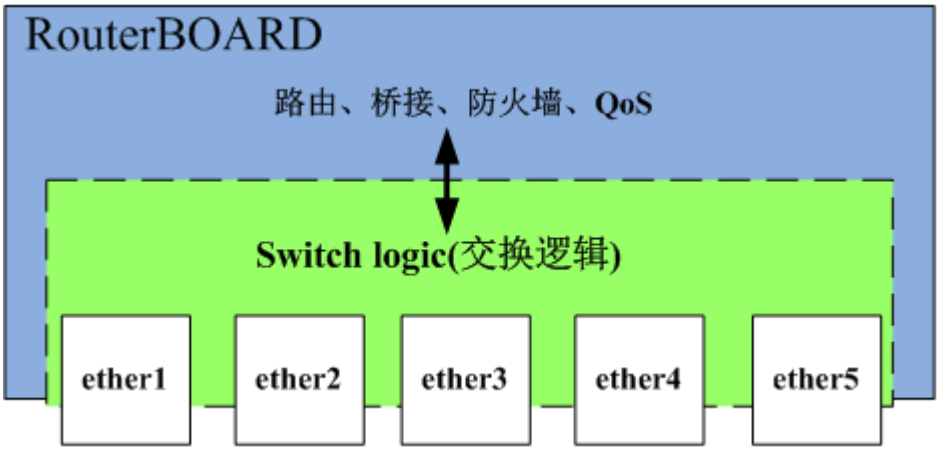
Traffic Types	字母缩写	优先级	协议举例	业务特征
Network Control	NC	7	BGP, PIM, SNMP	网络维护与管理报文的可靠传输，要求低丢包率
InternetWork	IC	6	STP, OSPF,	大型网络中区分于普通流量的网

Control			RIP	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

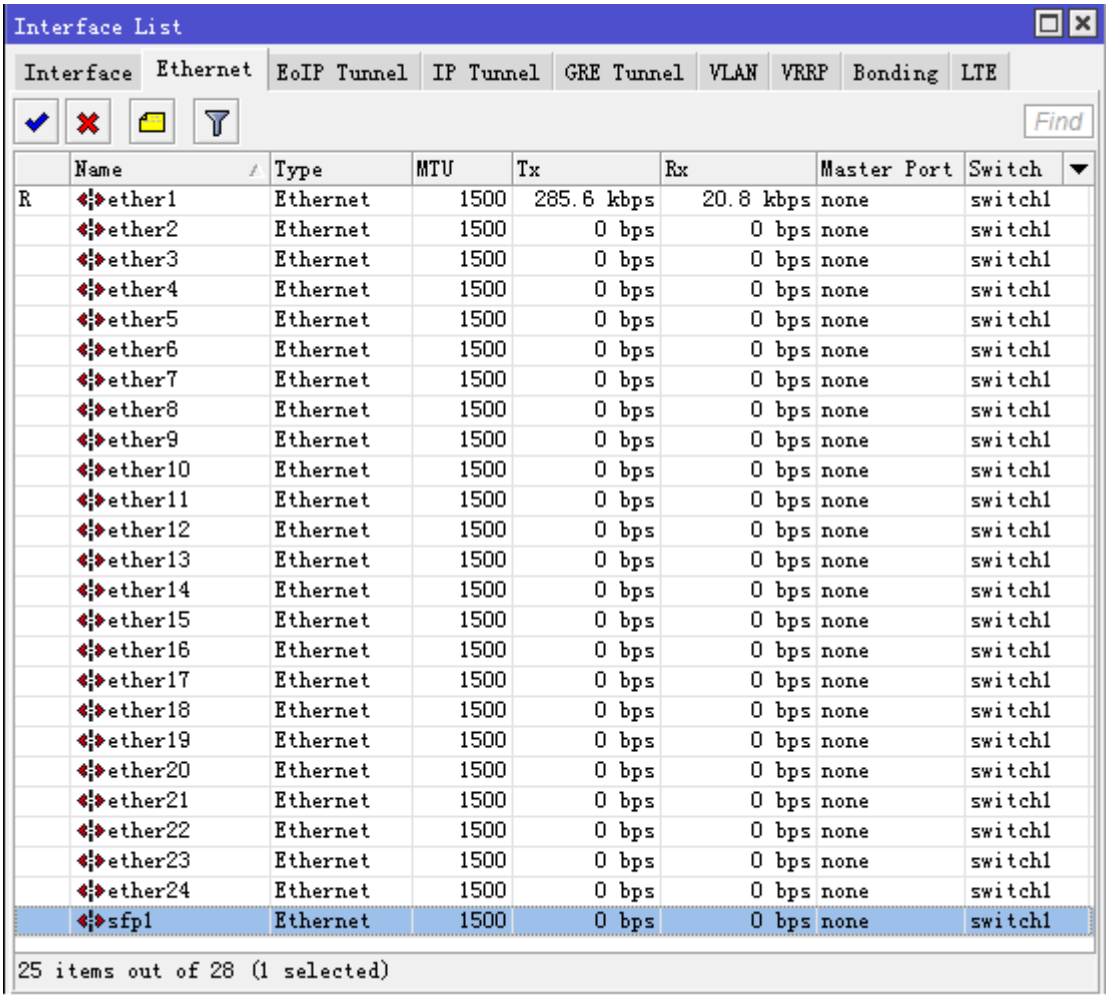
下面是 RouterBOARD 5 口交换芯片 Here is a general diagram of RouterBoard with a five port switch chip:



这里能看到，当一个端口接收到一个数据包，首先会传递给交换逻辑处理。交换逻辑决定数据包去哪一个端口。传递数据包是向 RouterOS，同样也可以被称为传递给交换芯片（cpu port）。即交换转发数据包给 cpu port，数据包开始被 RouterOS 某个接口处理进入的数据。当数据包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

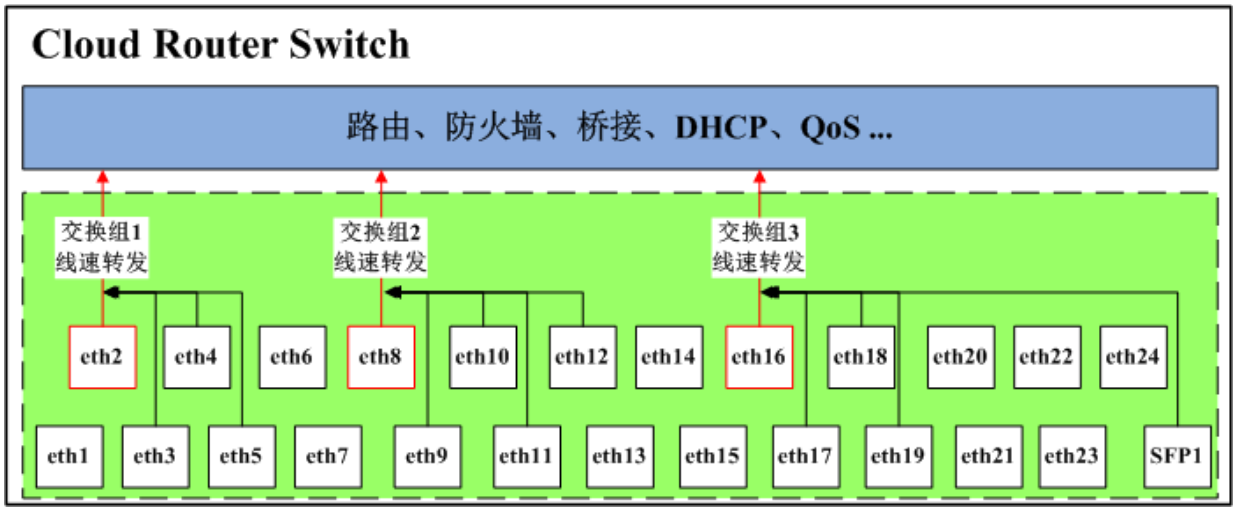
	Name	Type	MTU	Tx	Rx	Master Port	Switch
R	ether1	Ethernet	1500	268.1 kbps	14.2 kbps	none	switch1
	ether2	Ethernet	1500	0 bps	0 bps	none	switch1
S	ether3	Ethernet	1500	0 bps	0 bps	ether2	switch1
S	ether4	Ethernet	1500	0 bps	0 bps	ether2	switch1
S	ether5	Ethernet	1500	0 bps	0 bps	ether2	switch1
	ether6	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)

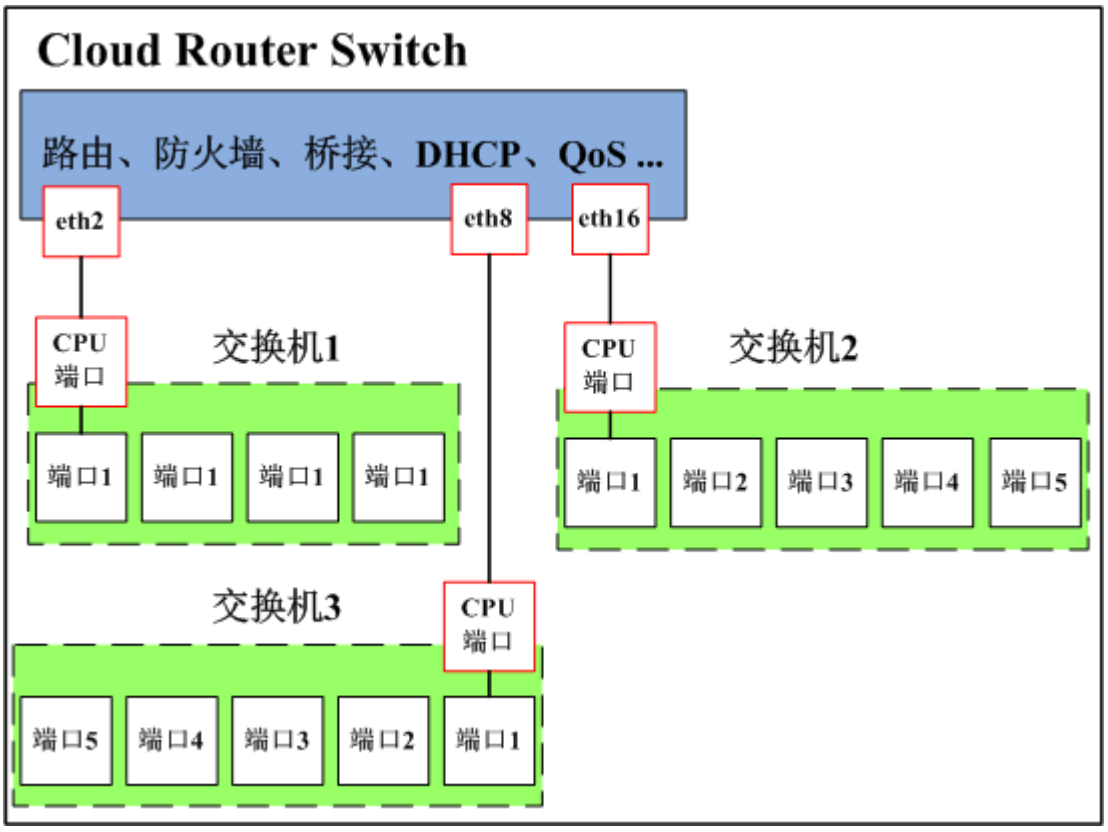
- 交换组 1：ether2-ether5
- 交换组 2：ether8-ether12
- 交换组 3：ether16-ether19，sfp1

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：

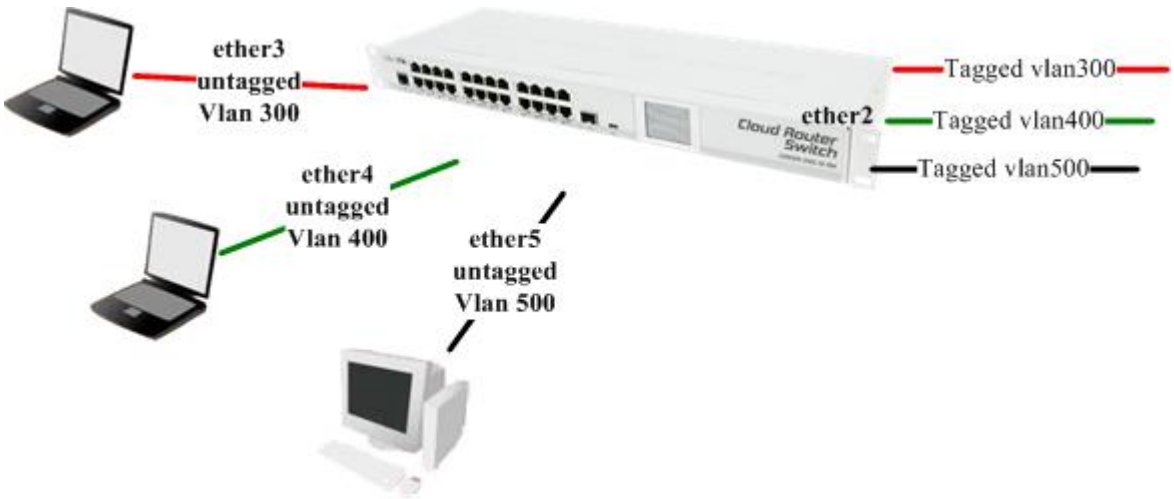


CRS 交换配置

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

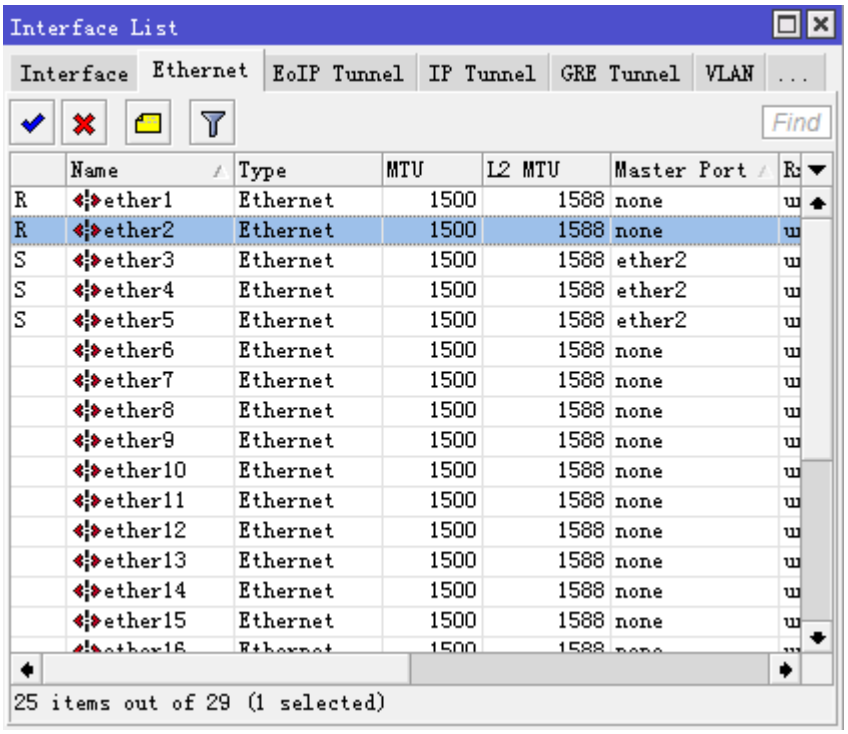
基于端口 VLAN



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

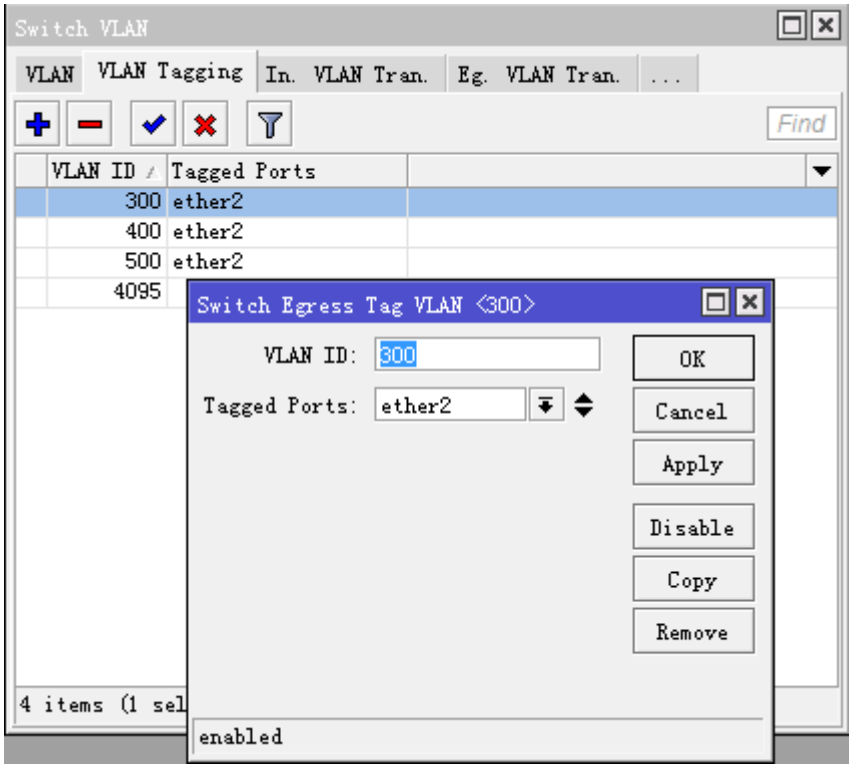



```
/interface ethernet
set ether3 master-port=ether2
set ether4 master-port=ether2
set ether5 master-port=ether2
```

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
```

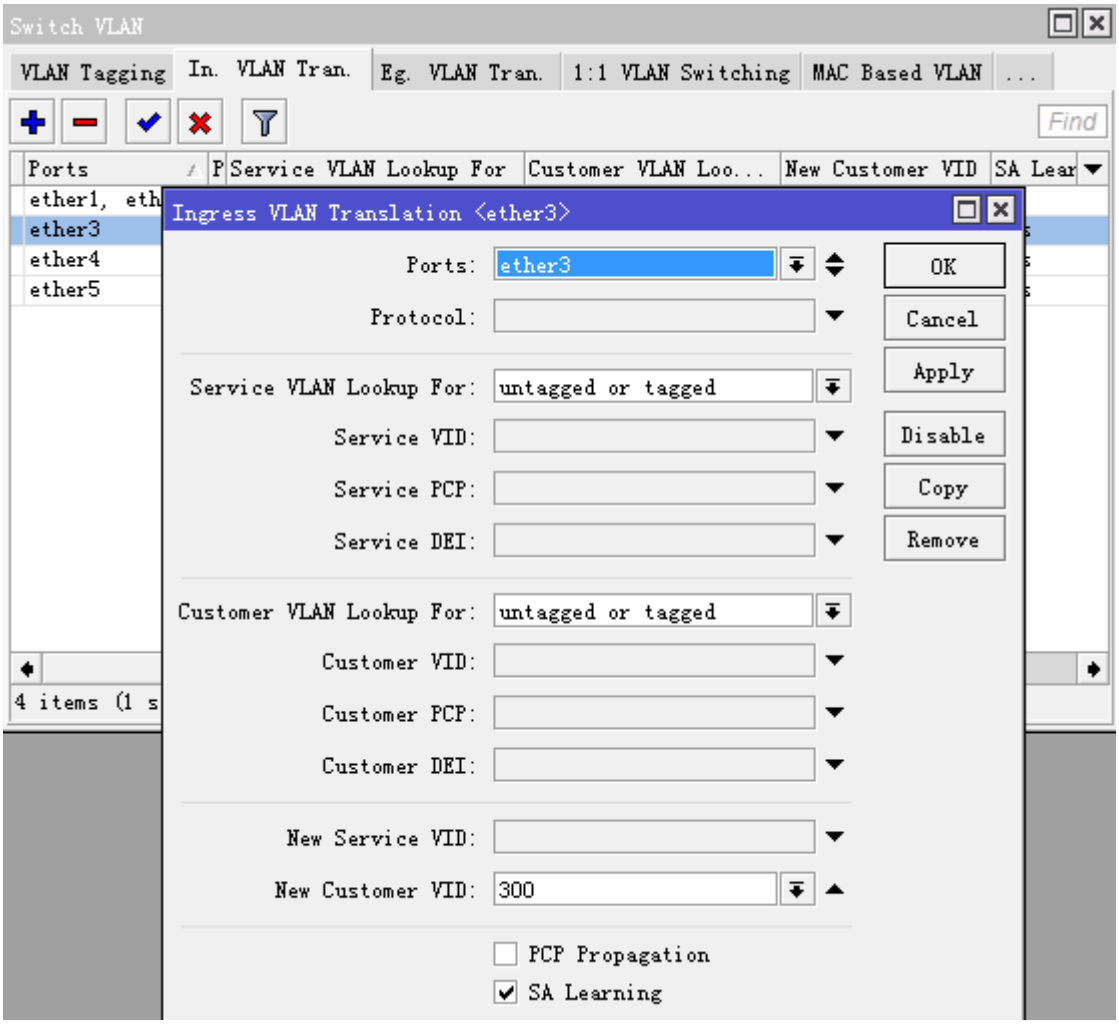
Winbox 中操作路径名有所不通，是在 switch vlan 中的 vlan tagging 下



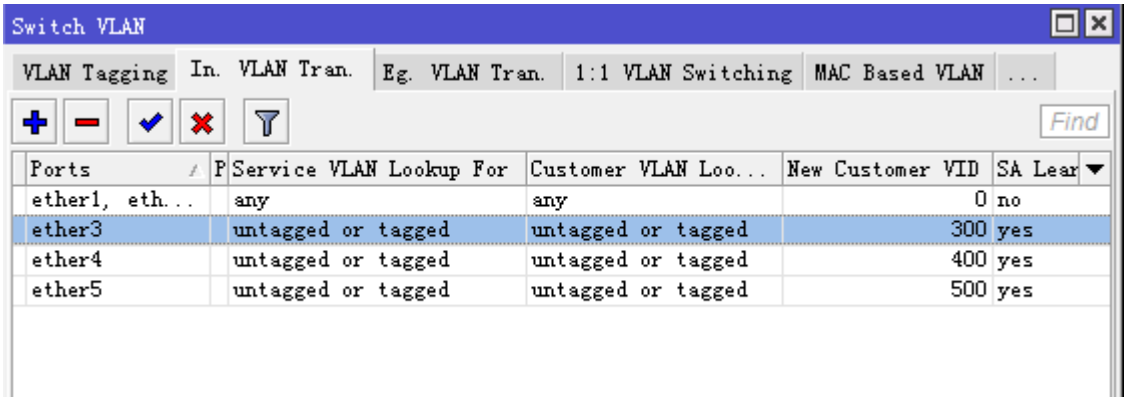
步骤 3、下面是添加 ether3、4、5 为 acess 口（取消标签），将对应 vlan 配置到相应端口

```
/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）。

3.13 RouterBOARD LCD 触摸屏

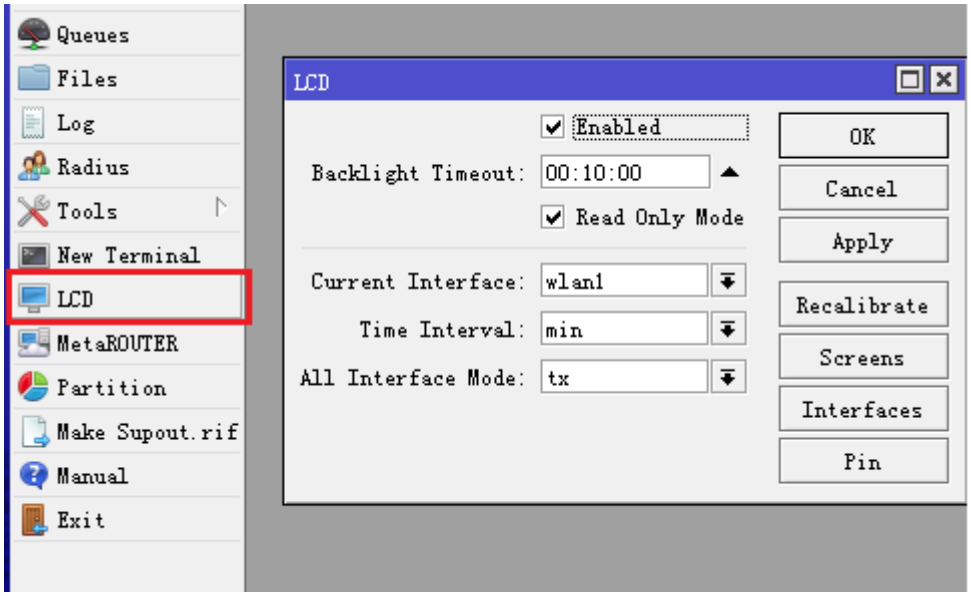
RouterBOARD 2011U 系列和 CCR 系列室内机型都集成了一块 LCD 触摸屏，方便快速查看设备状态和简单配置情况。触摸屏需要按压屏幕表面才能反馈操作。因此较轻或者快速短暂按压可能不能有反馈（这触摸屏看来

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

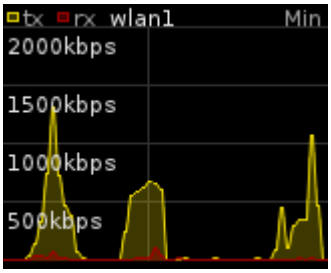
如下面图，我们可以通过触摸屏重启和复位配置，当做这些操作时我们就需要输入 PIN 密码，已确认管理能有权利执行。

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

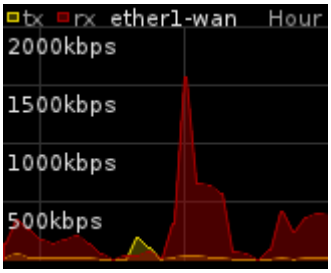
LCD 流量图

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

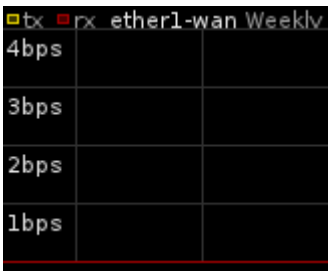


- **Hour** – 以小时显示流量。单位为每 5 分钟。垂直分割线为 1 小时，垂直分割线时间区域分三段，表示为从右到左为前 60 钟，中间 60 分钟，后 15 分钟



- **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 RouterBOARD 设备提供动态域名解析服务。即你的 RB 设备将自动获得一个域名，能为你经常变更的 IP 提供动态域名解析，让你知道如何连接自己的 RB 路由器。当然你在 nat 内的 RouterBOARD 肯定无法享受到 cloud 服务。

当前 Cloud 仅提供两种服务：

- ddns (提供 IPv4 外部地址的 DNS 域名解析，IPv6 不支持)
- 近似时间同步(当 NTP 无法工作时，实现时间同步，精确到秒，根据 UDP 数据包延迟)

注：事实上 DDNS 服务提供是由 MikroTik 的 cloud 服务器实现，所以用户必须调整防火墙策略保证能正常连接到 cloud 服务器

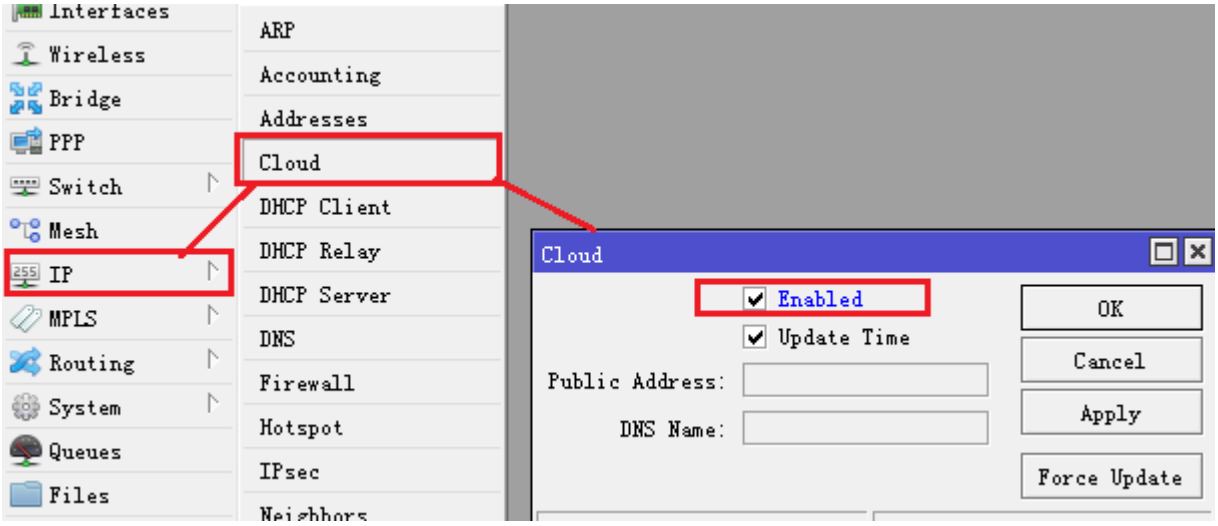
警告：如果路由有多个 IP 地址或网关，精确的 IP 地址更新可能会出现错误!解决方式是找到 DDNS 域名主机，将主机路由指向指定的网关。

工作方式：

- 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
- Cloud 时间更新：路由器重启后和每个 ddns 更新周期(也工作在当路由器网关 IP 变更或强制 ddns 更新命令）

Cloud 配置：

进入 winbox 后，打开 ip cloud，点击 Enabled，就开启了 cloud 服务，当然你要保证 RB 路由器网络连接正常，且防火墙没有对 cloud 服务器做限制



启用后自动得到显示如下：

```
[admin@MikroTik] /ip cloud> set enabled=yes
[admin@MikroTik] /ip cloud> print
          enabled: yes
    update-time: yes
         status: updated
       dns-name: 123456789abc.sn.mynetname.net
 public-address: 159.148.147.211
[admin@MikroTik] /ip cloud>
```

dns-name 是 cloud 服务器分给你的 DNS 域名，如上面的"123456789abc.sn.mynetname.net"，public-address 就是当前你的 RB 路由器，如上面的"159.148.147.211"，你可以用这个域名通过互联网连接你的路由器。

你可以通过 nslookup 命令解析：

```
C:\Users\yus>nslookup 123456789abc.sn.mynetname.net
服务器:  UnKnown
Address:  192.168.88.1

非权威应答:
名称:    123456789abc.sn.mynetname.net
Address:  159.148.147.211
```

Cloud 在 v6.27 调整

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;

例如 6.27 后的操作如下：

```
[admin@MikroTik] /ip cloud> set ddns-enabled=yes
[admin@MikroTik] /ip cloud> print
     ddns-enabled: yes
      update-time: yes
  public-address: 23.15.191.175
         dns-name: 496e02f8a689.sn.mynetname.net
           status: updated
          warning: DDNS server received request from IP 23.15.191.175 but your
                   local IP was 10.199.100.233; DDNS service might not work.
[admin@MikroTik] /ip cloud>
```

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

IP cloud 功能估计只能应用到 Tel 和 Un 的 PPPoE 拨号，因为只有他们才分配公网 IP，我想这个功能直接秒杀那些所谓的智能路由器，估计他们还没有想到为用户的路由器建立动态域名服务。

注意：在实际使用过程中，Cloud 功能出现了本地路由器能正确识别公网 ip，但服务器的域名解析返回的 IP 却不对的情况，该功能在部分 RouterBOARD 设备出现，所以使用第三方 DDNS 服务仍然有必要作为备选。

3.15 Flashfig

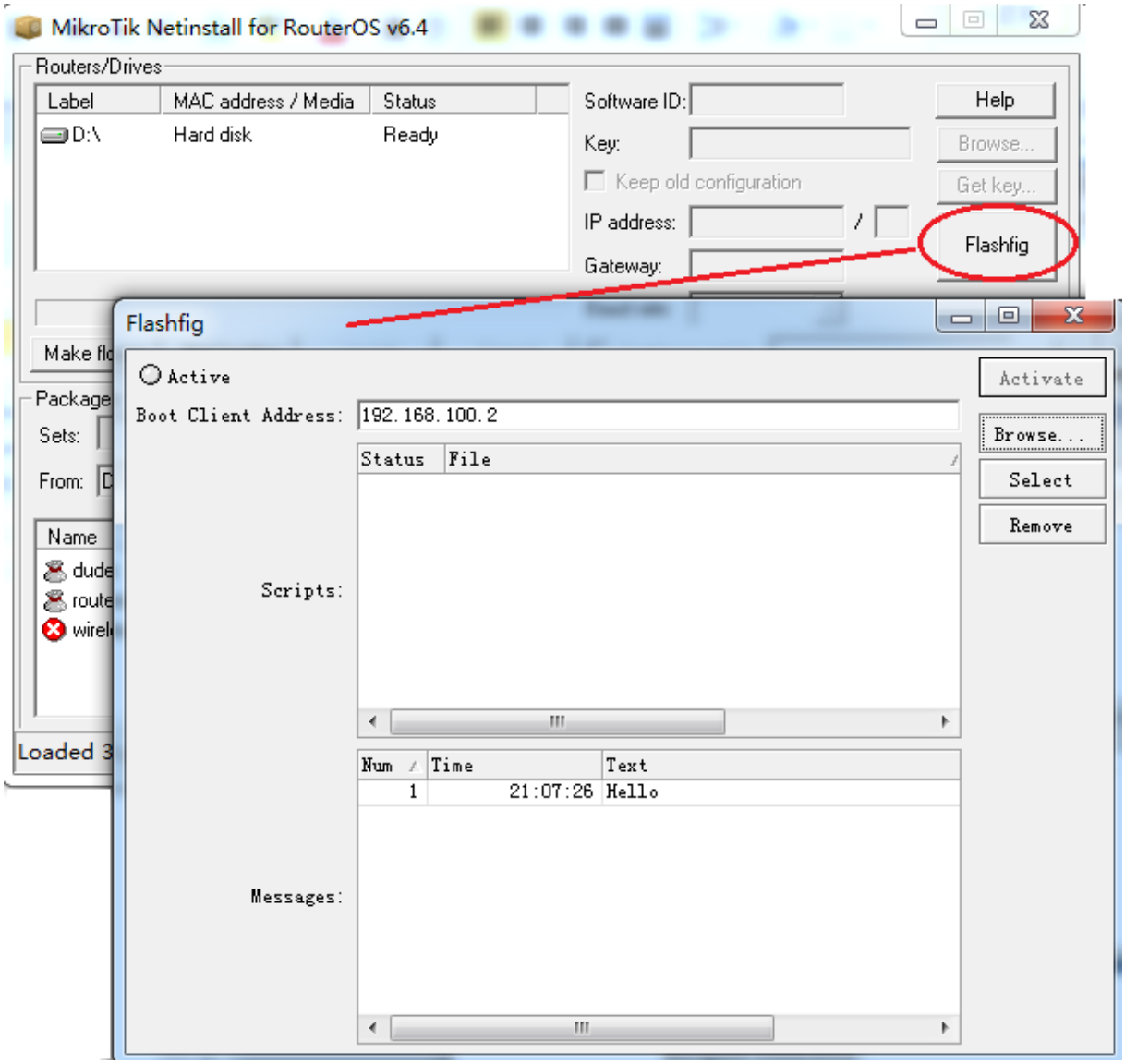
Flashfig 是一个应用于大量 RouterBOARD 的初始配置程序，主要被用于 MikroTik 分销商，ISP 或需要短时间对大量 MikroTik 路由进行配置的公司。

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 电脑。Flashfig 应用程序集成在 Netinstall 软件中。

Flashfig 支持所有的 RouterBOARD，运行 Flashfig 软件的电脑和 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 电脑必须具备以下条件 ：

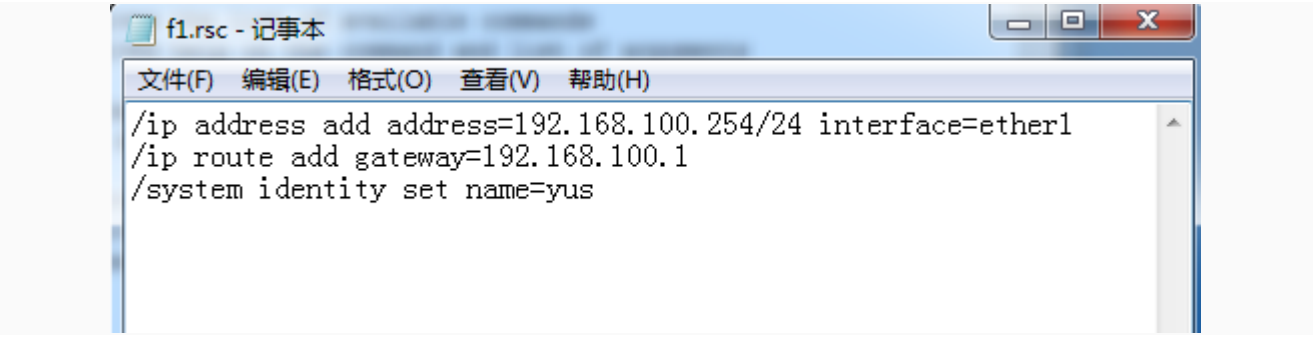
- 以太网接口

- .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

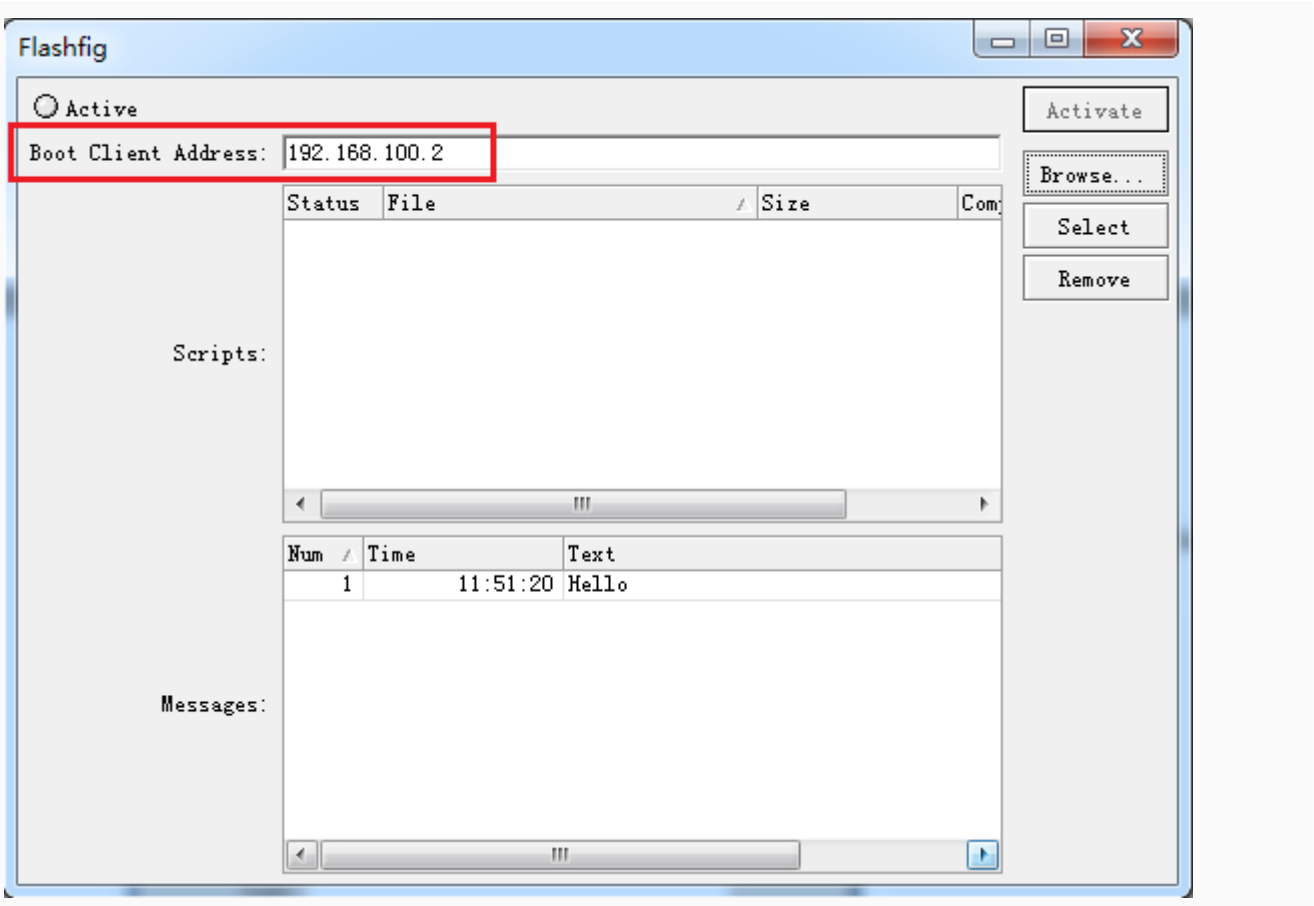
Netinstall 配置

- 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。

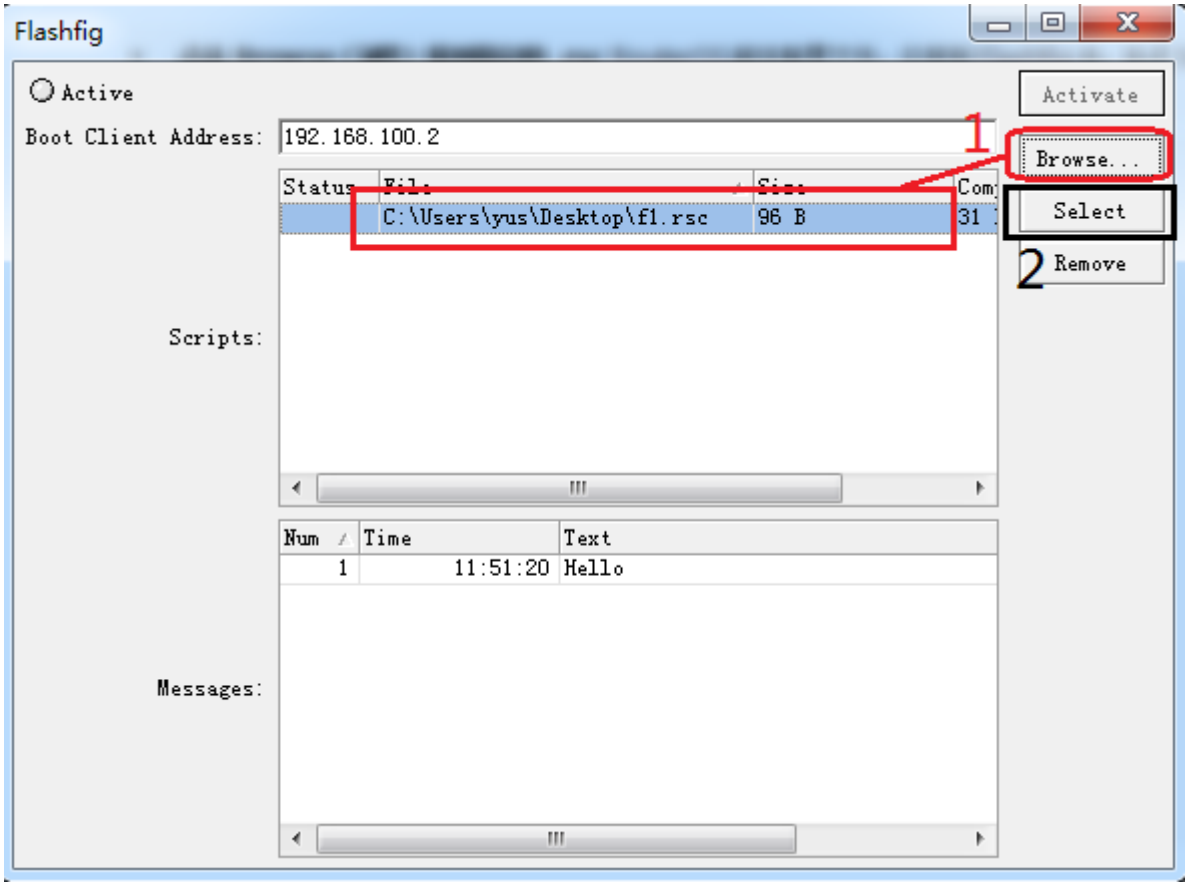


上面是编辑的 f1.rsc 脚本，编辑好后保存。

分别引导客户端的 IP 地址，这个 IP 地址必须和 windows 电脑在相同子网中，这里我们设置 windows 电脑 IP 为 192.168.100.99，分配给 RouterBOARD 引导地址为 192.168.100.2

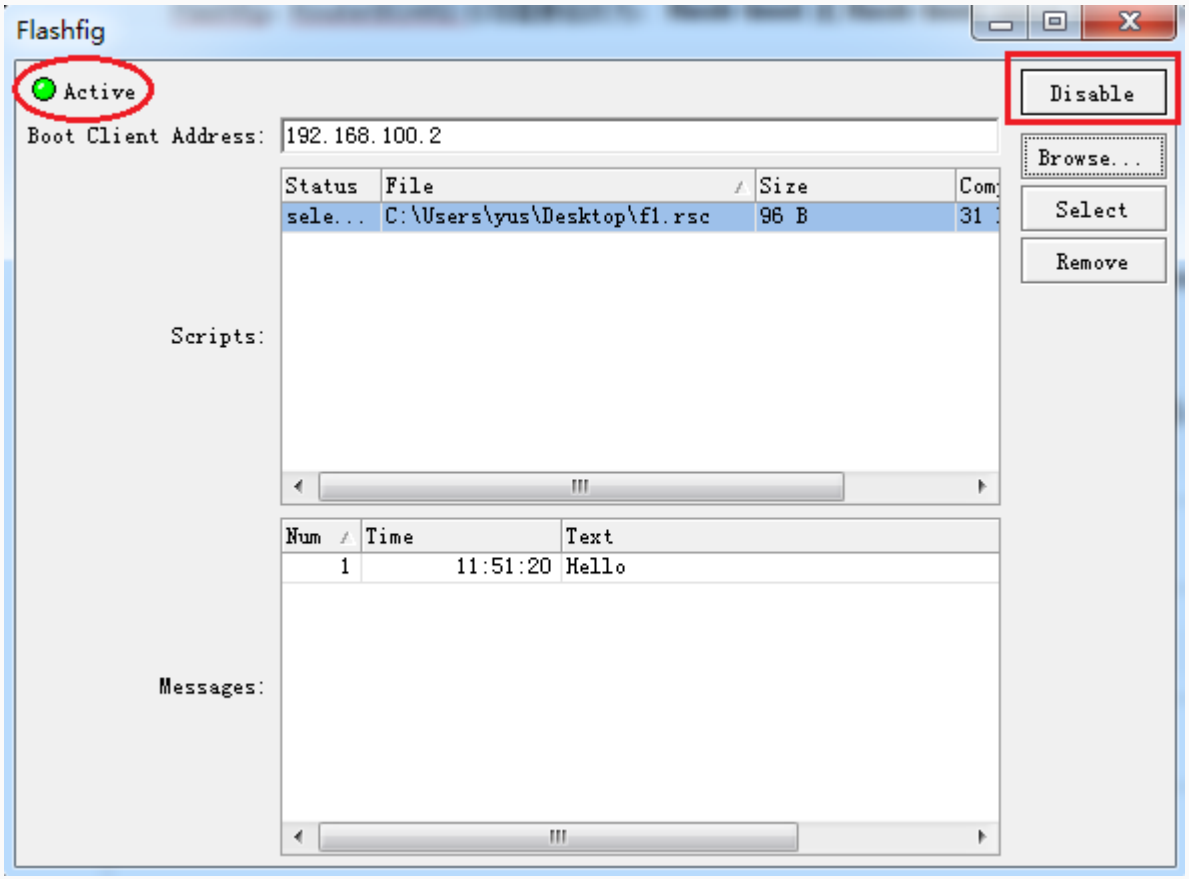


点击 Browse（浏览）找到指定的 .rsc RouterOS 脚本配置文件，应用到 Flashfig 中，并点 Select 选择急脚本文件指定生效。



注意：选择脚本文件不能使用中文路径。

点击 Activate 激活 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。

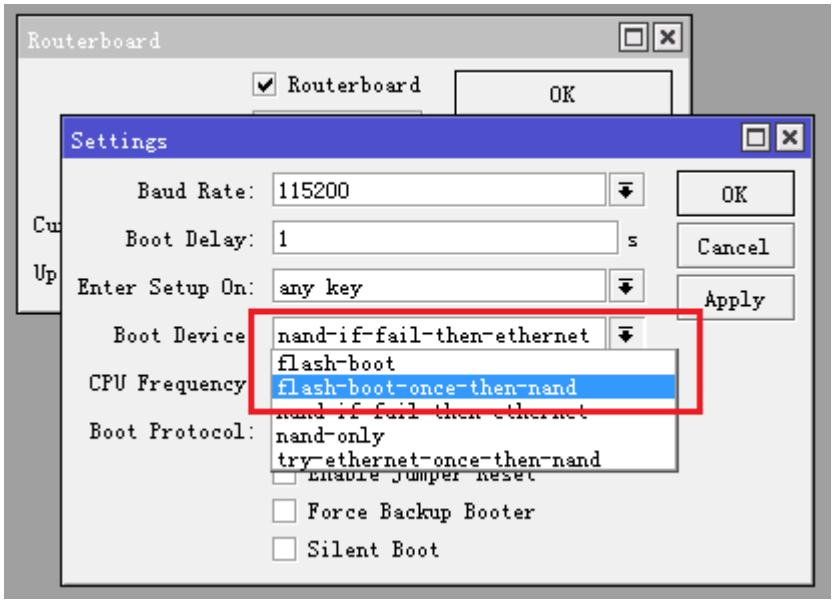
如果你的 RouterBOARD 的 Flashfig 没有启动，可以通过 Winbox 或 Console 连接修改引导，CLI 配置如下

```
/system routerboard settings set boot-device=flash-boot
```

也可以选择：

```
/system routerboard settigs set boot-device=flash-boot-once-then-nand
```

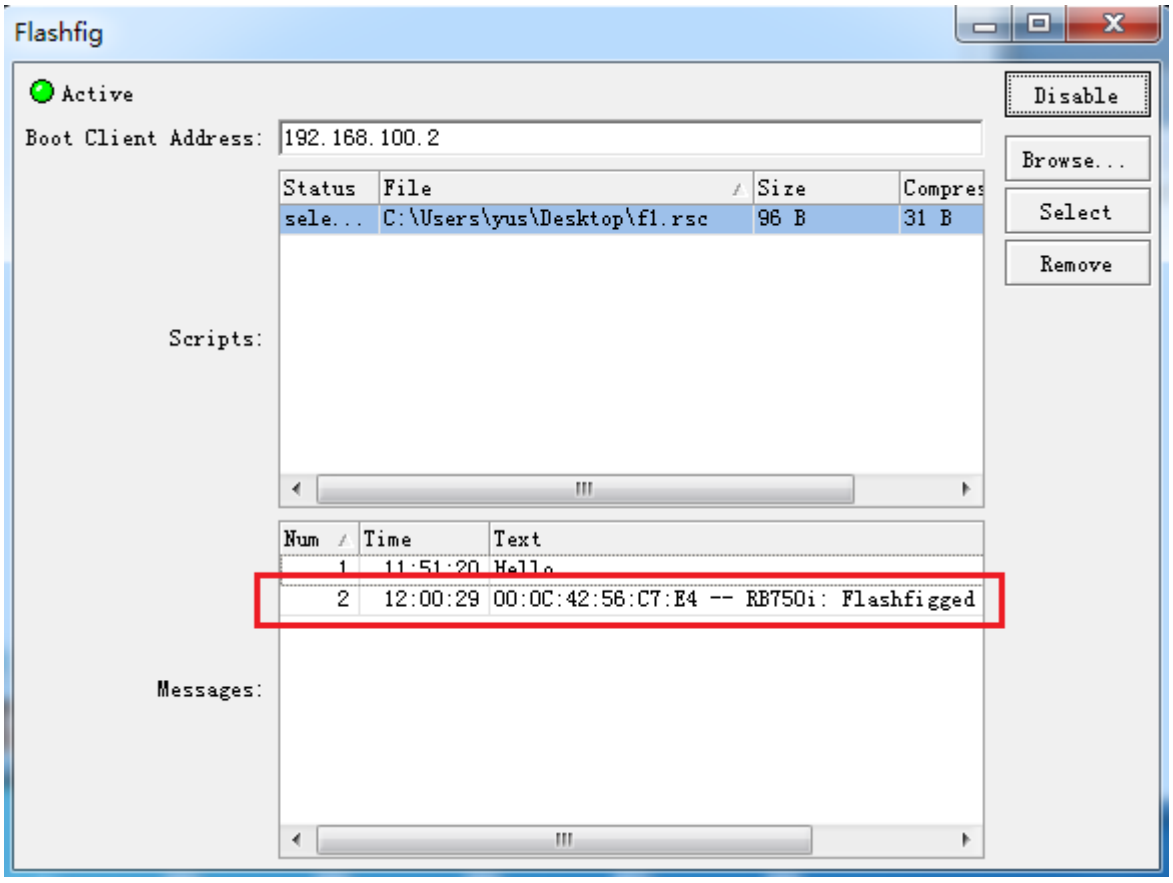
在 Winbox 下进入/system roeuterboard 配置：



- 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 电脑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

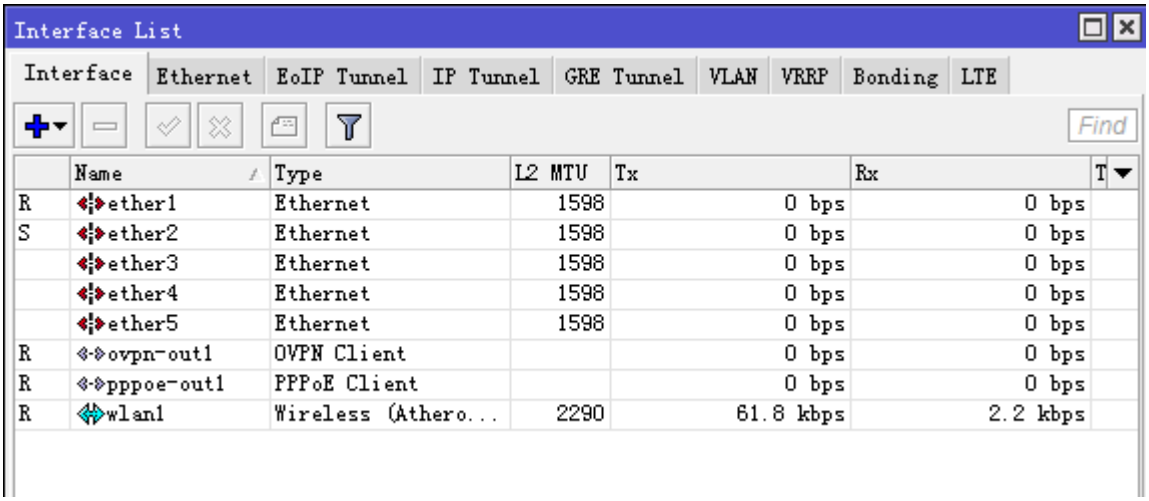


在 Messages 中提示，”RBxxx Flashfigged”，且 RouterBOARD 发出声音或 LED 闪烁信号，就可以将 RouterBOARD 断电重启。

第四章 接口配置（Interface）

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 支持各种网络接口卡，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。



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)
rx-rate (*整型*; 默认: **0**) – 最大数据接收率
0 - no limits
tx-rate (*整型*; 默认: **0**) – 最大数据发送率
0 - no limits

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 流量监视

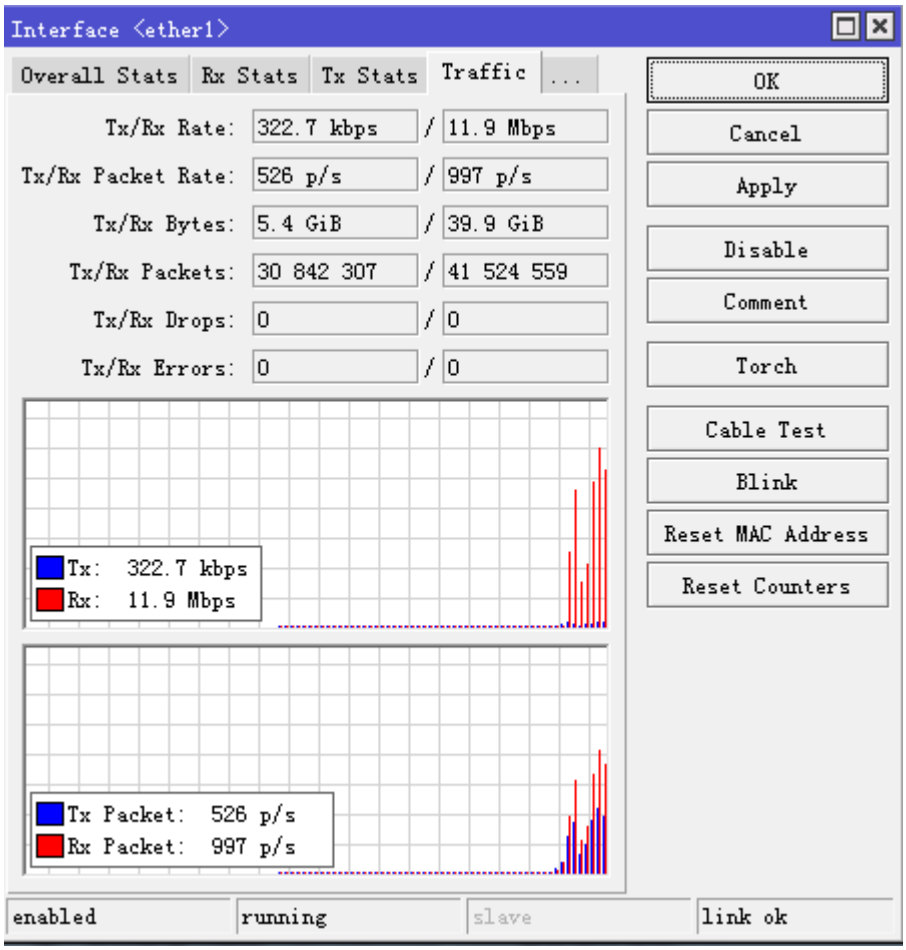
指令名称：***/interface monitor-traffic***

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）

对于 x86 平台的 RouterOS 支持各种以太网卡，对于各种型号的以太网卡支持，MikroTik 没有给出明确的支持列表，也只能通过登陆 wiki.mikrotik.com 查找相关资料。

功能包：***system***
等级：*Level1*
操作路径：***/interface ethernet***

以太网接口配置

操作路径：***/interface ethernet***

属性描述

arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; 默认: **enabled**) - 地址解析协议的模式

auto-negotiation (*yes* | *no*; 默认: **yes**) – 当被启用，以太网接口将自动协商最大的性能获得最好的连接

注: Auto-negotiation 在两个终端必须同时禁用，否则以太网将不能正常工作

注 2: Gigabit（千兆）传输不能工作在 auto-negotiation 被禁用状态。

bandwidth(*整型/整型,*默认: **unlimited/unlimited**) – 设置最大的接口 Rx/Tx 带宽限速, Tx 支持所有采用 Atheros 交换芯片的 RouterBOARD，RX 仅支持 AR8327 交换芯片的 RouterBOARD

cable-setting (*default* | *short* | *standard*; 默认: **default**) – 修改网线连接长度 (仅适用于 NS DP83815/6 网卡)

disable-running-check (*yes* | *no*; 默认: **yes**) – 路由器网卡自动探测网络设备功能，如果这个参数设置为“no”，路由器网卡将禁用监测功能

full-duplex (*yes* | *no*; 默认: **yes**) – 定义数据传输是否同时在两个方向上，即全双工。

l2mtu (*整型*; 默认:) – 二层最大传输单位

mac-address (*MAC*; 默认:) – 一个网卡的介质访问控制地址

master-port (*name* | *none*; 默认: **none**) – 设置交换组的主接口

mdix-enable (*yes* | *no*; 默认:) – 是否在该接口上开启 MDI/X 自动线序纠正功能

mtu (*integer*; 默认: **1500**) – 网卡最大传输单位

name (*string*; 默认:) – 定义以太网卡名称

speed (*10Mbps* | *100Mbps* | *1Gbps*| *10Gbps*; 默认: **最大带宽**) – 设置网卡的数据传输速度，默认情况下根据网卡支持最大传输单位

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

接口状态监测命令

指令名称：***/interface ethernet monitor***

属性描述

status (link-ok | no-link | unknown) – 以太网卡接口的状态，包括：
link-ok – 网卡以连接到网络
no-link – 网卡没有连接到网络
unknown – 网卡未确认
rate (10 Mbps | 100 Mbps | 1000 Mbps) – 实际的连接速率
auto-negotiation (done | incomplete) – 相邻连接的状态判断。
done – 判断完成
incomplete – 判断失败
full-duplex (yes | no) – 是否为全双工数据传输

例如：通过 Monitor 命令可以查看现在以太网卡的连接状态，link-ok 为以太网络连接正常：

```
[admin@MikroTik] interface ethernet> monitor ether1,ether2
                 status: link-ok     link-ok
    auto-negotiation: done       done
                rate: 100Mbps 100Mbps
          full-duplex:  yes          yes
```

修改以太网卡 mac 地址：

```
[admin@MikroTik] interface ethernet>set 0 mac-address=00:0C:42:03:11:0A
```

Interface 中的其他功能，将会在后面准备介绍

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。

第五章 IP 配置与 ARP

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，网段到远端地址结束。

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 列表，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

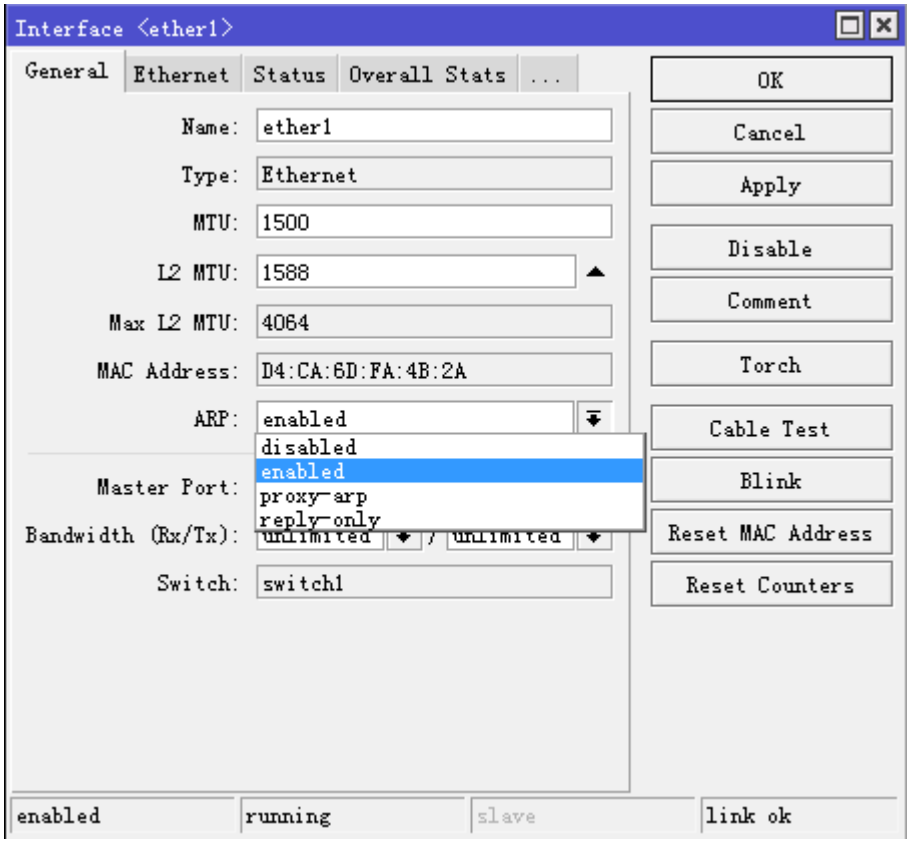
注：RouterOS 最大的 ARP 的条目数为 8192，所以在同一广播域内主机数量被限制在 8192 台。

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 请求将不被路由器回应，因此必须添加静态的 ARP 条目。首先需要在/ip arp 列表添加对端 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 电脑中：

```
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 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>
```

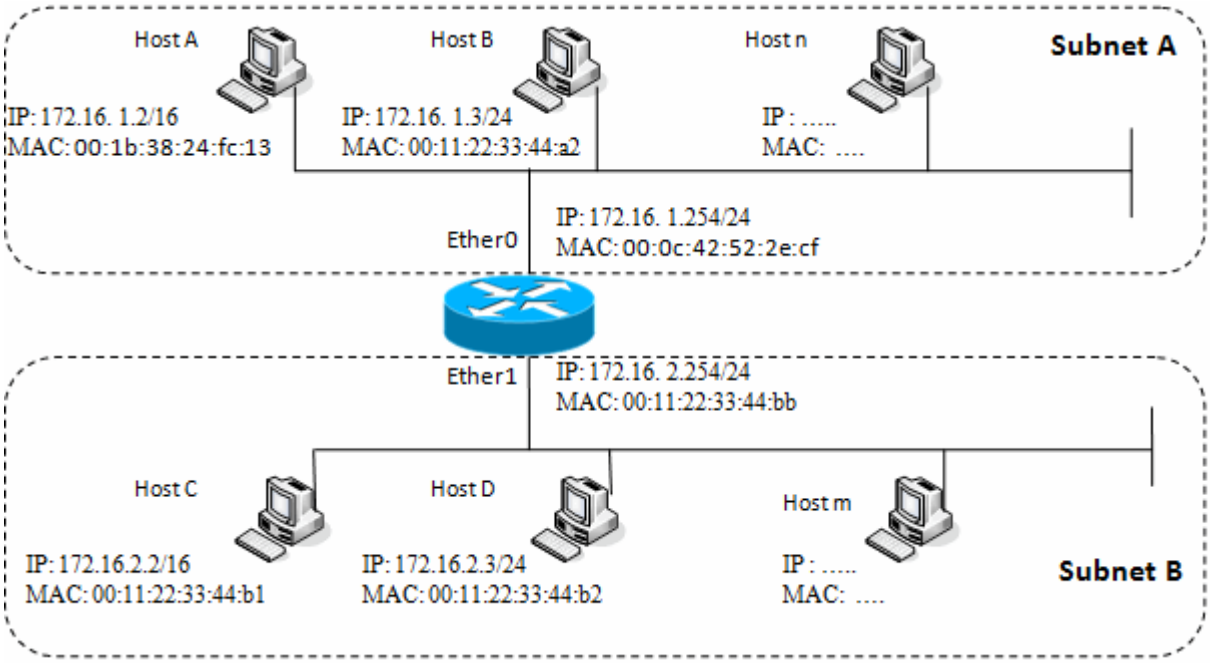
如果在一个接口上使用静态 ARP 记录会使网络更安全，即我们常说的路由器 ARP 绑定，除了配置静态 ARP 条目，还必须将该接口上的 arp 设置为 'reply-only'，相关操作在下面的**/interface** 目录中，如果非静态的 ARP 条目，将无法与路由器进行通信：

```
[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 ARP 代理，将实现 ARP 透明代理的功能，即将直接连接两个相同子网，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 地址回应该请求，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 想发送数据包到 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 在同一子网段是直连，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包

该信息的数据包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

数据包详细内容

```
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 请求数据包将 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

数据包详细内容

```
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 发送回应数据包，在回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)的数据包都直接发送到路由 (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：

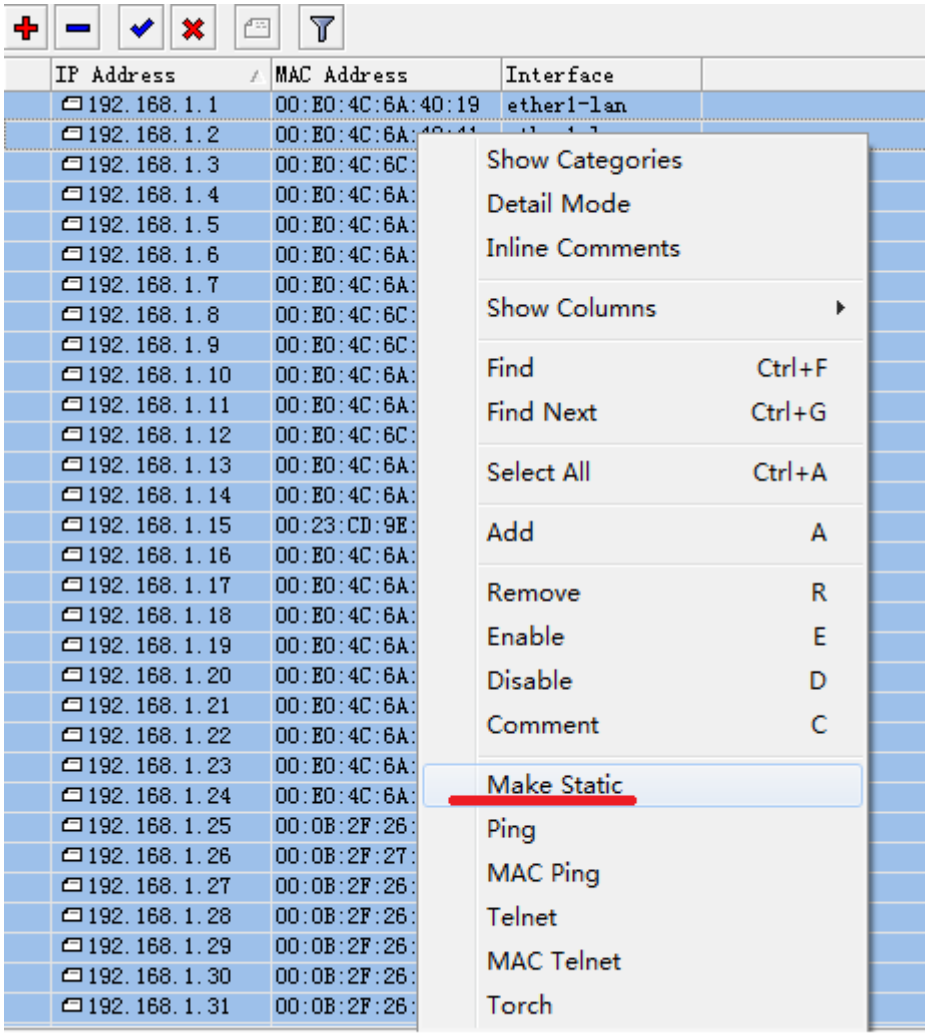
```
[admin@MikroTik] ip arp> /interface ethernet print
Flags: X - disabled, R - running, S – slave
#      NAME          MTU       MAC-ADDRESS        ARP
0   R ether1         1500   00:30:4F:0B:7B:C1     enabled
1   R ether2         1500   00:30:4F:06:62:12     enabled
[admin@MikroTik] /interface ethernet
[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>
```

5.4 ARP 绑定

虽然主机在 IP 网络中是通过 IP 地址通话，但实际上硬件地址（MAC 地址）被用于主机到其他主机的数据传输。地址解析协议 Address resolution protocol (ARP) 是提供硬件地址与 IP 地址之间的解析。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。路由器通过 ARP 列表的记录来回应各个主机的数据。我们也可通过静态的 ARP 记录，要求路由器只对静态的 ARP 做回应。这样就可以避免出现如有用户擅自修改 IP 地址或者通过 ARP 病毒影响路由路由器工作。如通过下面的设置：

1. 在 WinBox 中将动态 ARP 设置为静态的 ARP 条目。

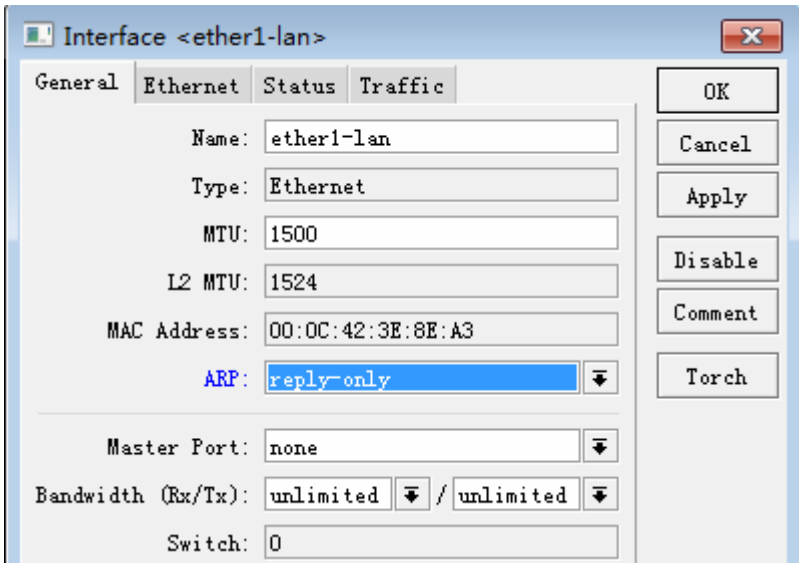


或者通过命令操作：

```
[admin@MikroTik] ip arp> add address=192.168.1.248 interface=ether1-lan
mac-address=00:21:00:56:00:12
```

同样的我们可以将所有的动态 ARP 条目修改为静态。

2. 设置 ether1-lan interface 仅回应静态 ARP 的请求，arp=reply-only：



命令操作如下：

```
[admin@RB230] > interface ethernet set ether2 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 请求将不被路由器回应，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 电脑做对路由器绑定的设置

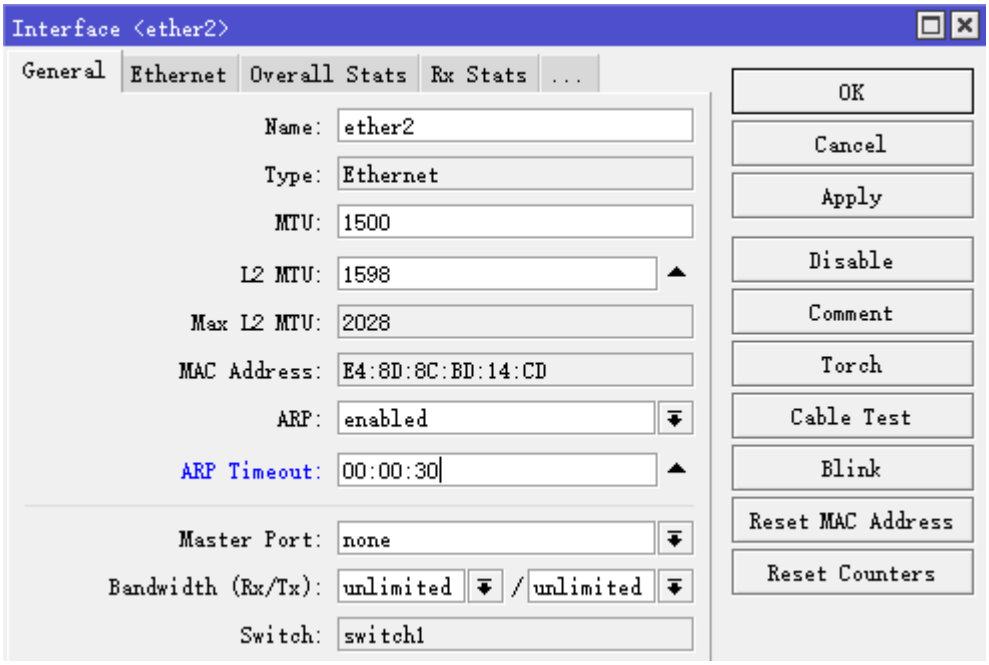
```
C:\> arp -s 10.5.8.254   00-aa-00-62-c6-09
```

也可以编辑 windows 的批处理文件（.dat）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：


Interface <ether2>
General
Ethernet
Overall Stats
Rx Stats
...
Name: ether2
Type: Ethernet
MTU: 1500
L2 MTU: 1598
Max L2 MTU: 2028
MAC Address: E4:8D:8C:BD:14:CD
ARP: enabled
ARP Timeout: 00:00:30
Master Port: none
Bandwidth (Rx/Tx): unlimited / unlimited
Switch: switch1
OK
Cancel
Apply
Disable
Comment
Torch
Cable Test
Blink
Reset MAC Address
Reset Counters

第六章 路由设置（Route）

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***

6.1 RouterOS 路由介绍

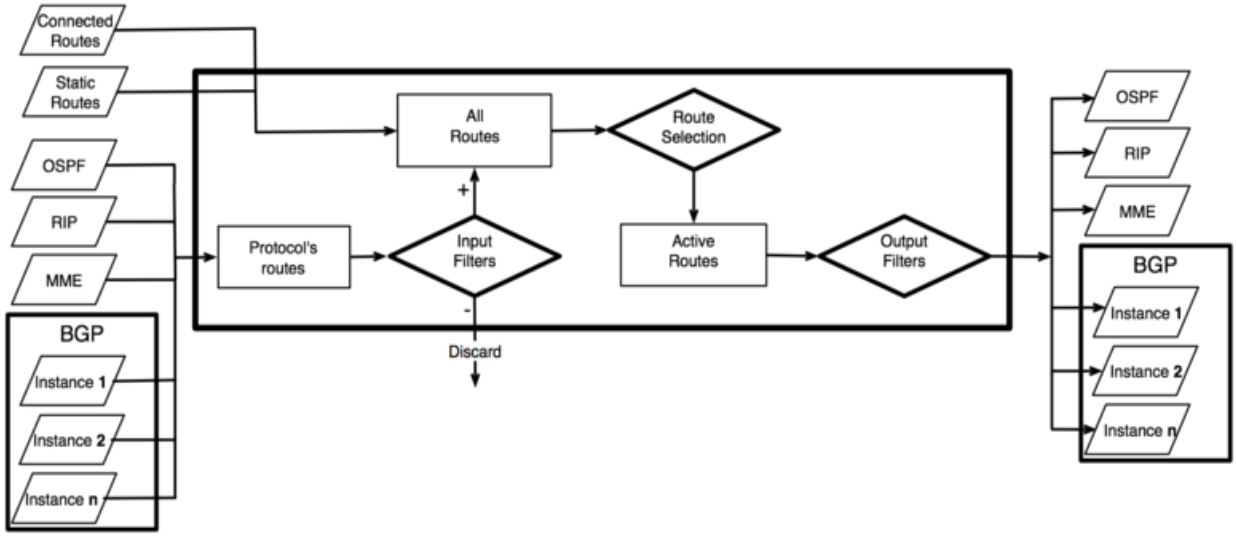
RouterOS 支持各种路由规则和策略，在实际网络环境中可选择适合自己网络的路由规则，如 Tel 和 Un 多线接入、目标和源地址策略路由、基于端口的策略路由和多线路绑定的负载均衡、已经负载均衡演变的权重路由，下面是 RouterOS 支持的几种方式的路由：

- **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 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**

路由器是连接多个相互独立子网，保持路由信息连续的设备，对于路由器来说需要通过以下相关的功能保持路由正常工作：

- 每一种路由协议（除 BGP）有一份内部路由表，即按照路由协议制定路由选择路径，BGP 没有内部路由表的，它从所有周边设备存储的 RIB 获取路由表。
- 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 路由信息库（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，计算每条目标信息项最佳路径，建立和更新转发信息库，在不同路由协议间分配路由。

通常路由转发通过目标地址决定，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。这样的操作查找匹配给定地址的最具体的路线被称为路由表查询（routing table lookup）。

如果路由表包含多条路由到同一目标地址（dst-address，前面所提到的精确子网匹配），有且仅有一条会被转发，这条路由会映射到 FIB 里，并标记为 active（激活）

当转发决策使用附加参数，如源地址路由匹配，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 中标记数据包）。所有路由都会首先到默认的 main 路由表，然后通过 routing-mark 分配到具体的路由表中。

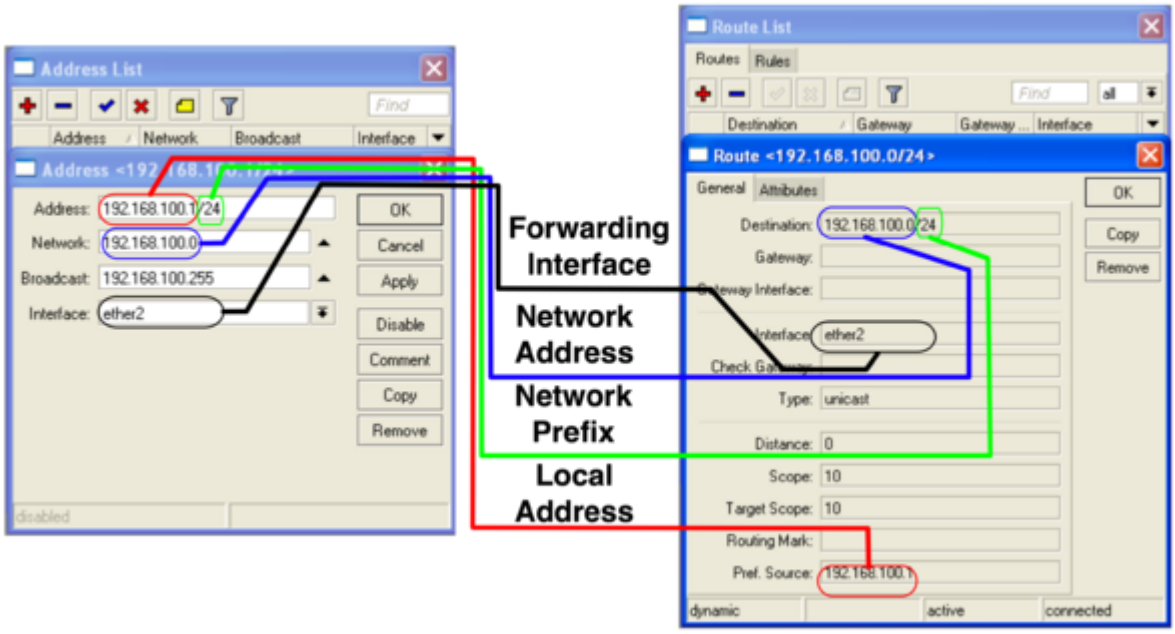
默认路由（Default route）

即目标路由（dst-address）为 *0.0.0.0/0* ，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，这时所有在该路由表中查询路由绝不会丢失路由，因为默认路由给出了默认出口的路径选择。

直连路由（Connected routes）

直连路由是创建在路由器已经启用的网络接口上配置的 IP 网络路由，这样的地址根据子网范围会自动生成直连路由网络，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 192.168.88.1/24，那 192.168.88.0～255 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。



每个直连路由都有一个对应的 IP 地址属性参数，如下面：

- 直连路由中的 **dst-address** 等同于 ip 地址属性中的 **network**。
- 直连路由中的 dst-address 子网部分，等同于 IP 地址属性中的子网
- 直连路由中的 pref-src 等同于主机的 ip 地址
- 直连路由中的 interface 与实际 IP 配置的 interface 等同

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，但如果通过配置，如实现负载均衡，即 ECMP (Equal cost multi-path)路由，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。

ECMP 能做到是因为转发决策被缓存的结果，转发数据包有相同的源地址、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）。

网络接口作为路由网关

网关可以使用网络接口来代替下一跳的网关 IP 地址，接口路由有下面的具体属性：

- 不同于直连路由，路由的接口做下一跳，但不能被用于下一跳到查询。
- 可以分配多个接口到 gateway 作为网关，也可以创建 ECMP 路由，但不能用于直连路由的网关值

6.2 RouterOS 路由分类

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 数据包指定到默认网关的路由，这需要手动指定默认的网关。

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）

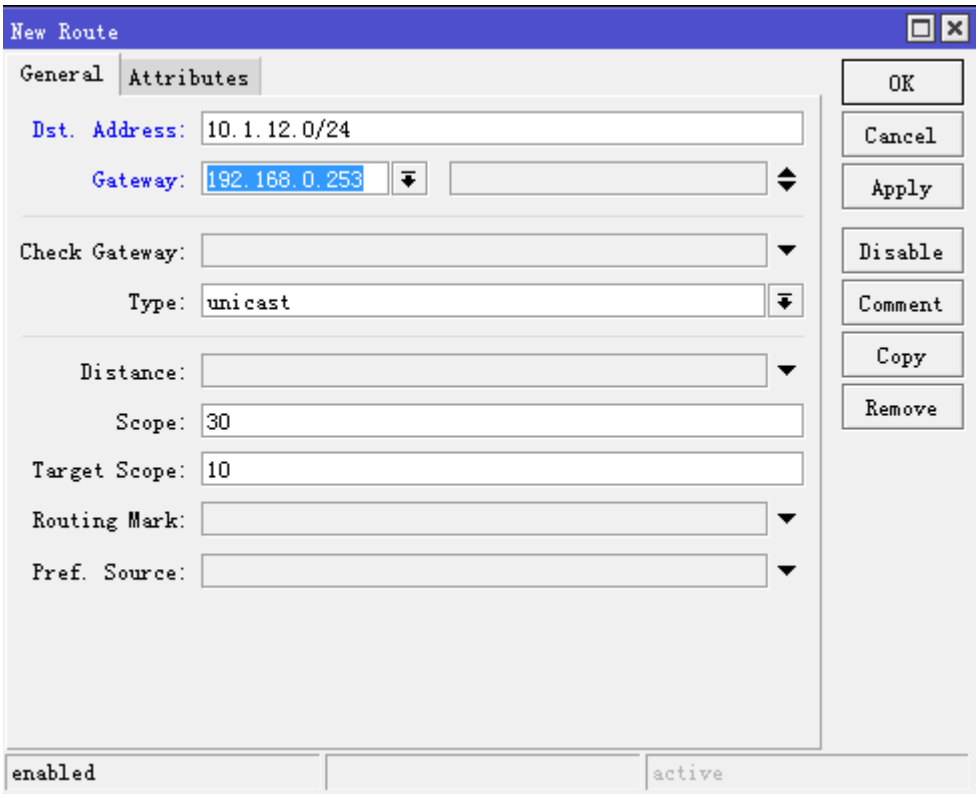
静态路由

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

当接入网络使用同一类型网络，又是多线路接入时，可以采用 ECMP 负载均衡。最基本的负载均衡“Equal-Cost Multi-Path Routing”即等价多路径路由，但这种方式在普通的路由交换中最常见，是非 nat 静态路由的一种常见路由负载均衡方式。

注意：Equal-Cost Multi-Path Routing（等价多路径路由）在做 nat 的负载均衡缺点是每 10 分钟会重新均衡线路，这样 session 将会被指定到其他网关，出现访问的源地址改变，目标地址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。

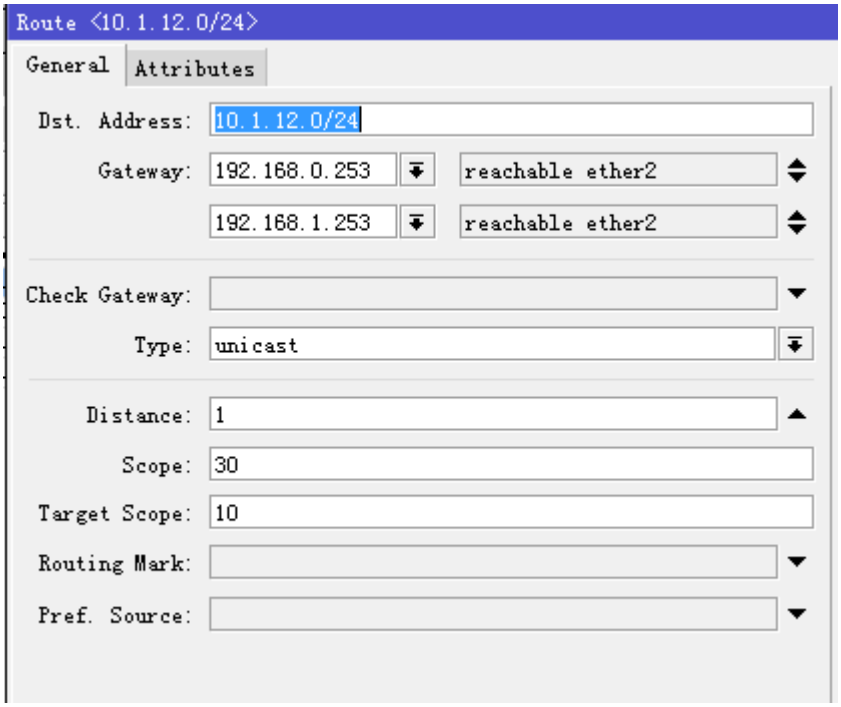
- Equal-Cost Multi-Path Routing 的操作，通过在 ip route 添加多网关的静态路由（格式如：**gateway=x.x.x.x,y.y.y.y**）路由协议会建立动态的多路路由。

等价路由在路由设备中是最常见的配置，但这个并不能应用到做 nat 设备的路由器上，因为源地址信息以及被隐藏。

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 RouterOS 最灵活的路由配置，需要涉及到 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 做负载均衡也属于策略路由种类，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。

6.3 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。

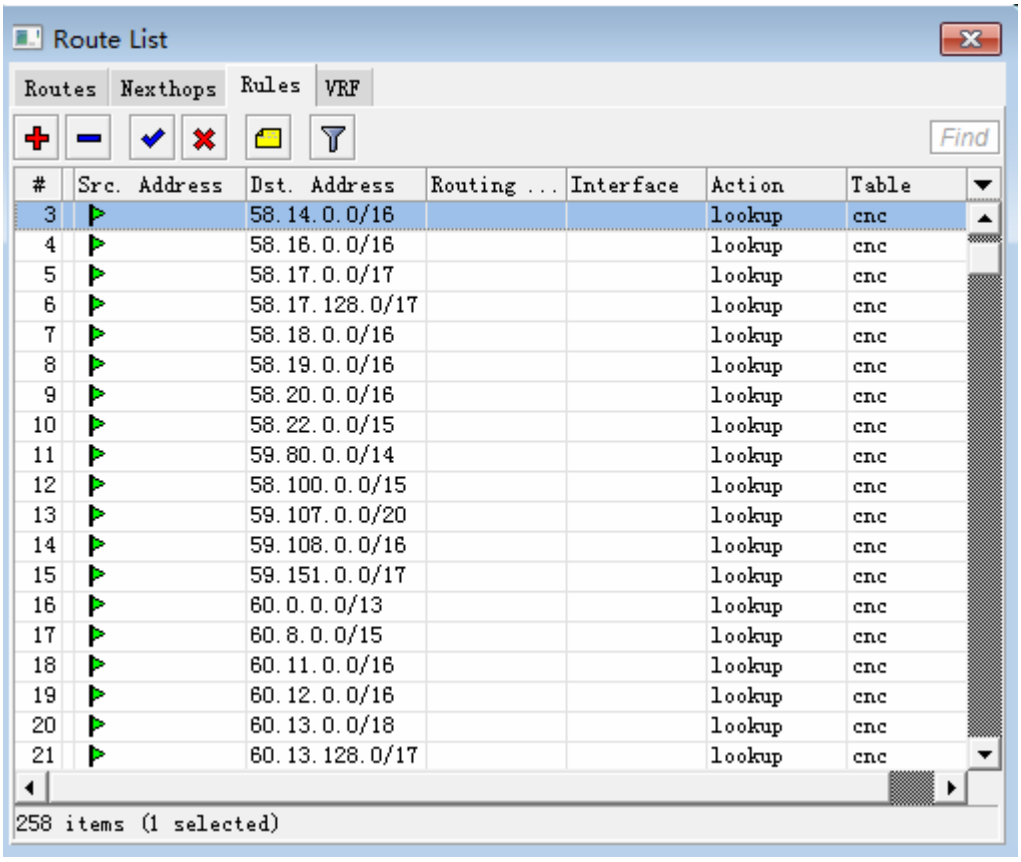
上传 Tel 或 Un 路由脚本后，在根目录下使用 import 命令导入：

```
[admin@MikroTik] > import cnc1.rsc
```

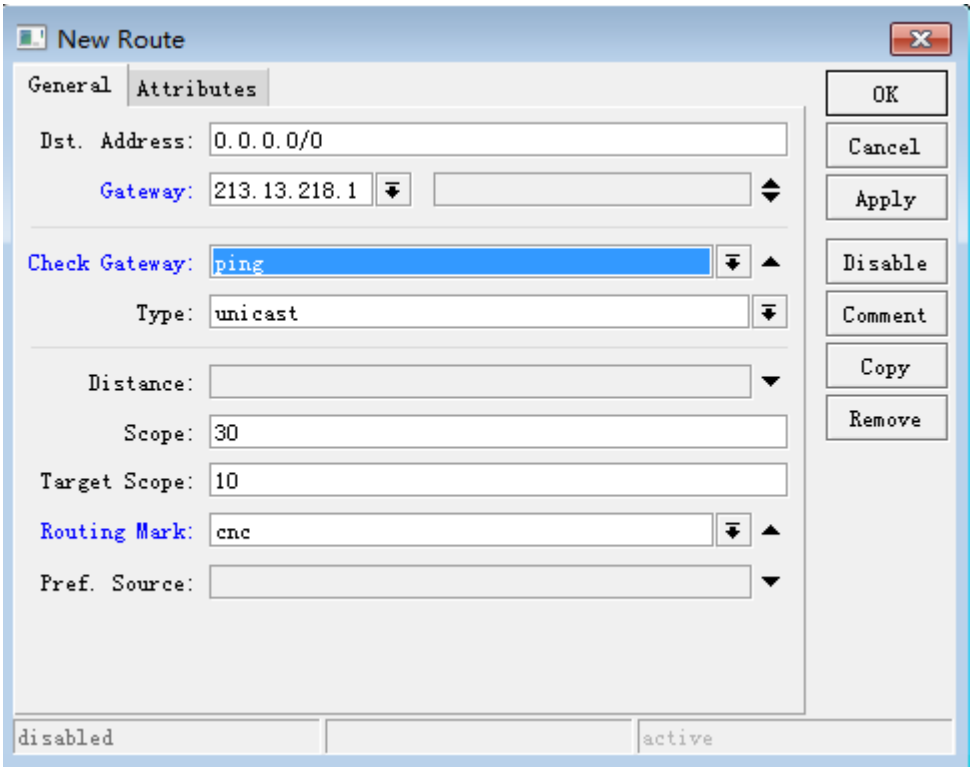
设置路由规则时命令如下：

```
/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：



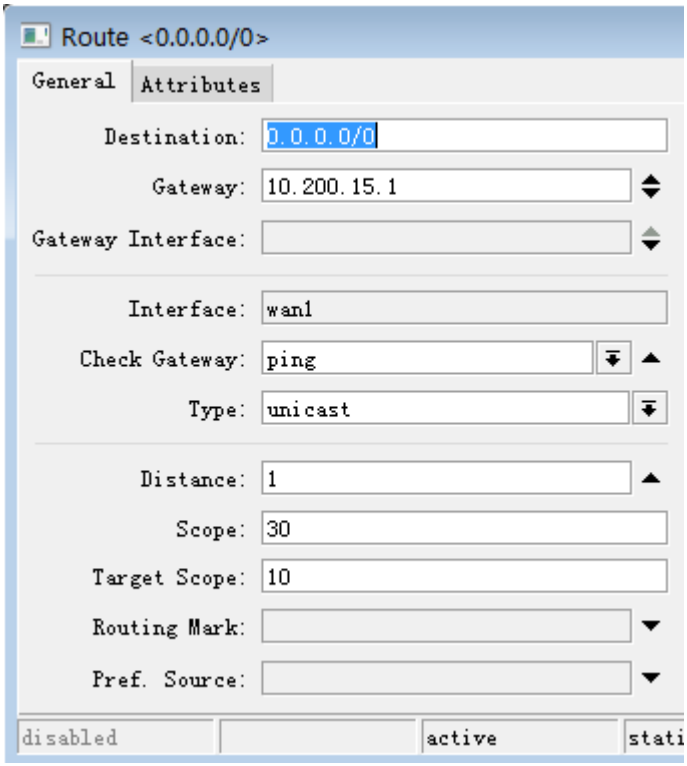
之后我们在 ip route 中 routing-mark 选择对应的 cnc 路由表



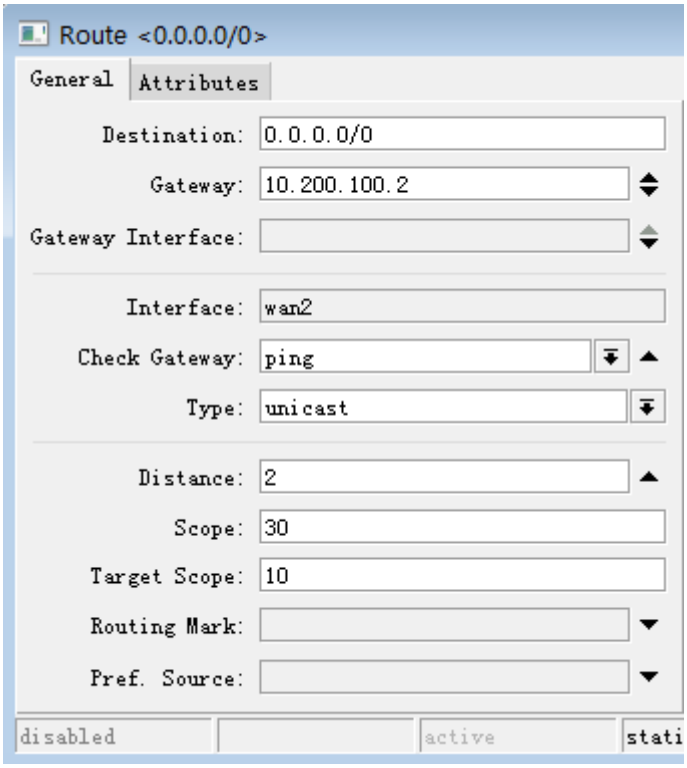
6.4 网关断线处理

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。

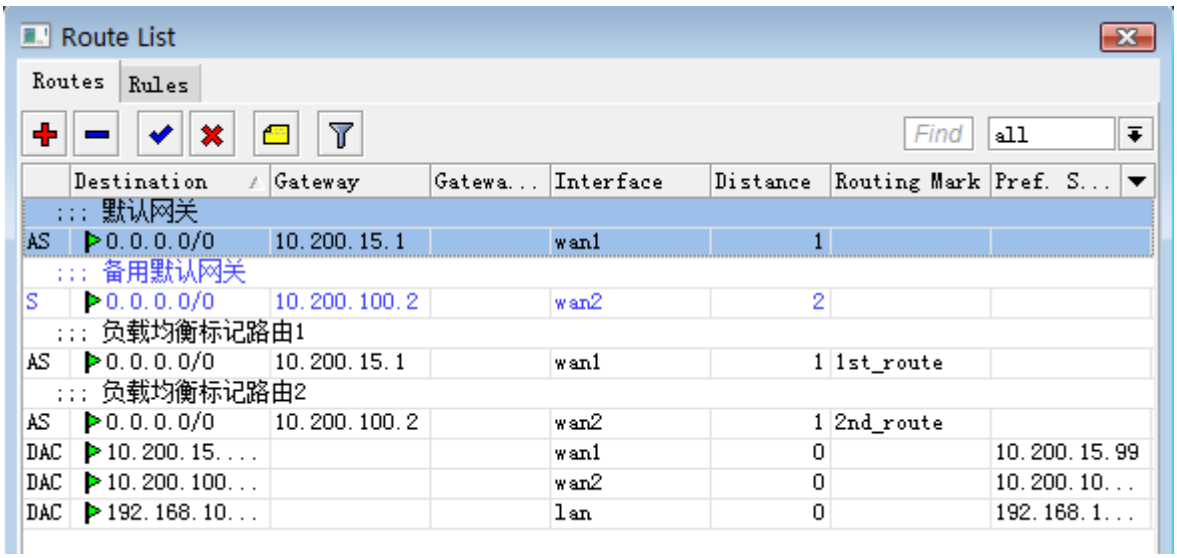
如下图：默认网关的 **distance** 设置为 1，并设置 check-gateway=ping，通过 ping 监测网关状态：



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：



6.5 源地址策略路由奇偶标记

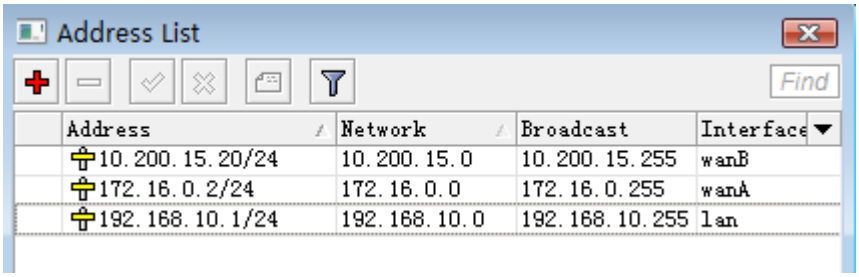
在双线策略路由器情况下，早期的策略路由是通过标记一段地址走一条线路，如我们标记192.168.10.2-192.168.10.127 走线路 A，剩下的 IP 地址则走线路 B，这样的策略路由在一定的情况下出现效率不高的问题，如当用户 IP 地址是顺序增加，但没有到 127 的时候。线路 B 就不会起到流量分配的作用。

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。

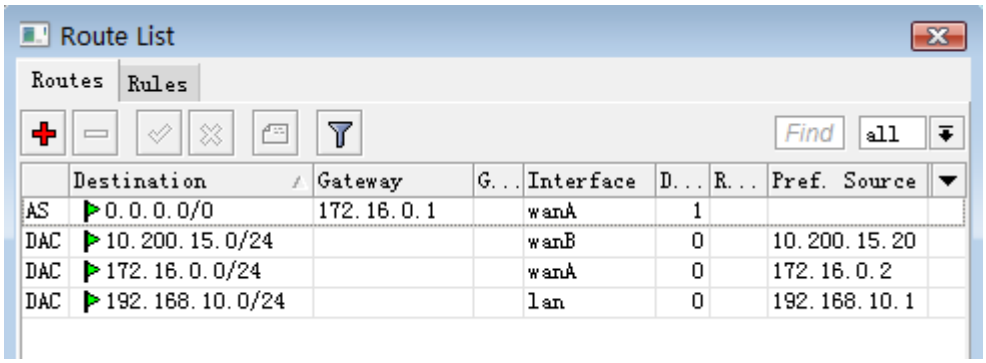
操作步骤如下：

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 标记数据包；
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 A 和 B，A 的 IP 地址是 172.16.0.2，网关是 172.16.0.1；B 的 IP 地址是：10.200.15.20，网关是 10.200.15.1。

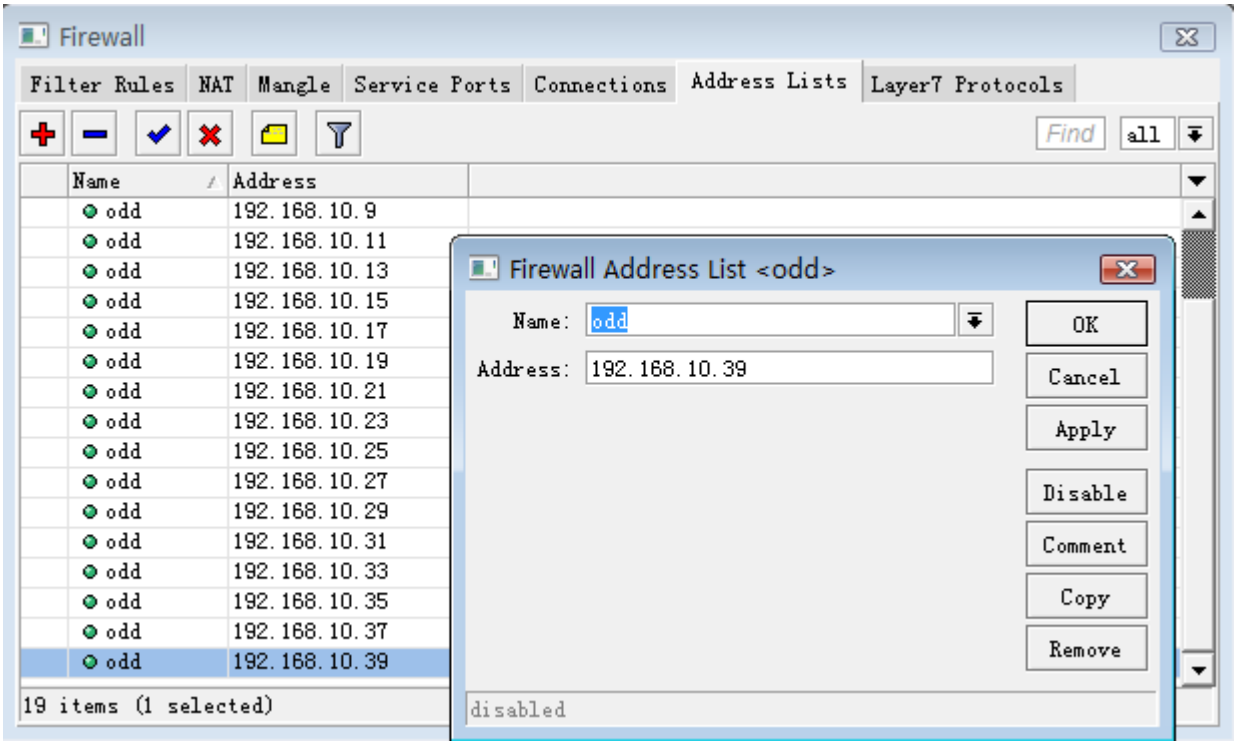


在 ip route 中配置以线路 A 的网关 172.16.0.1 为默认路由：



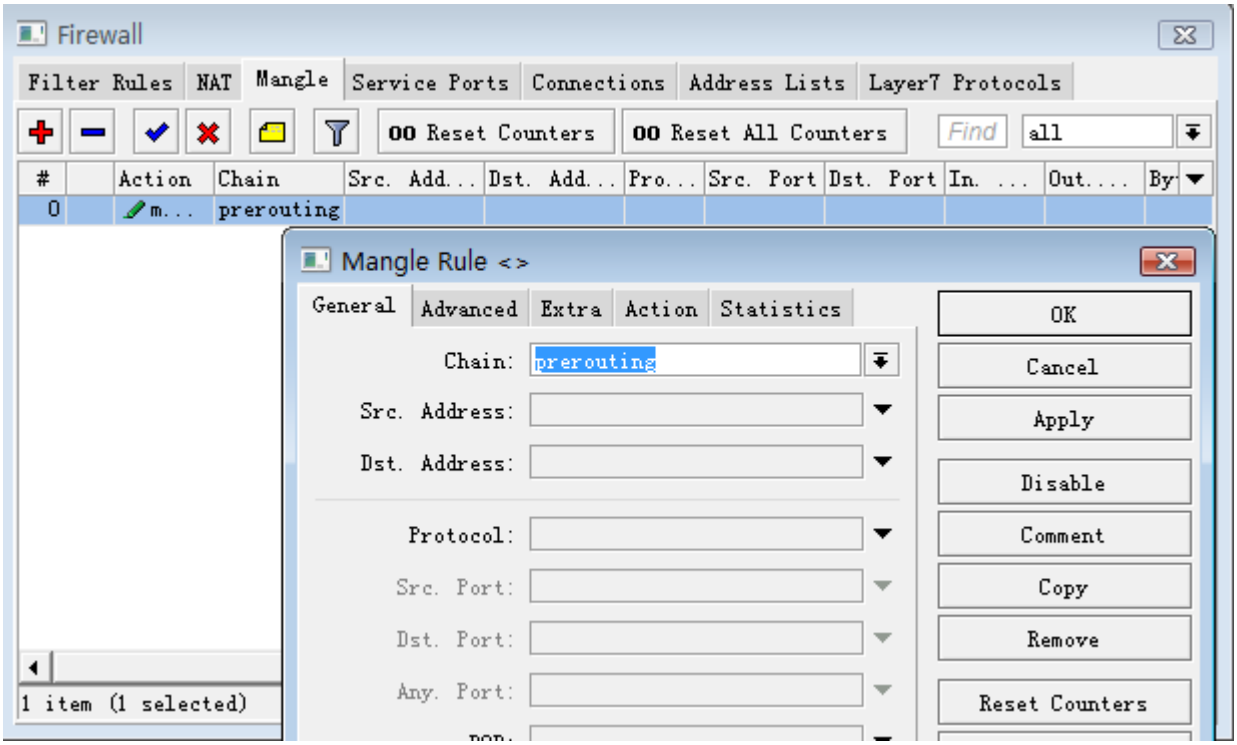
步骤 2：配置好 IP 地址和路由后，接下在 ip firewall address-list 中添加奇数的地址列表，因为是双线路由，我们只需要配置一条奇数的列表，而偶数的列表可以不用配置，因为奇数被标记后，剩下的就为偶数地址。

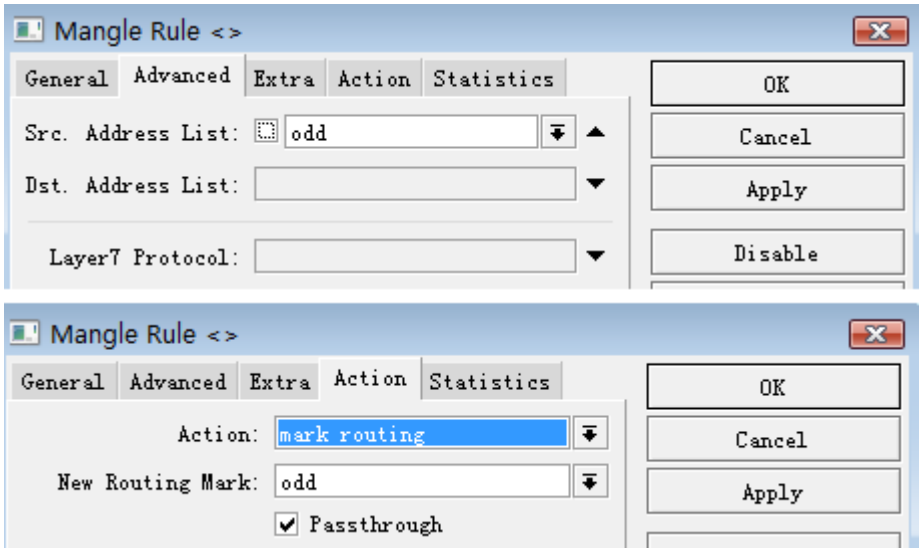
我们将奇数列表地址取名为 odd，并向地址列表里面添加你网络内所有的奇数的 IP 地址：



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 chain=prerouting：

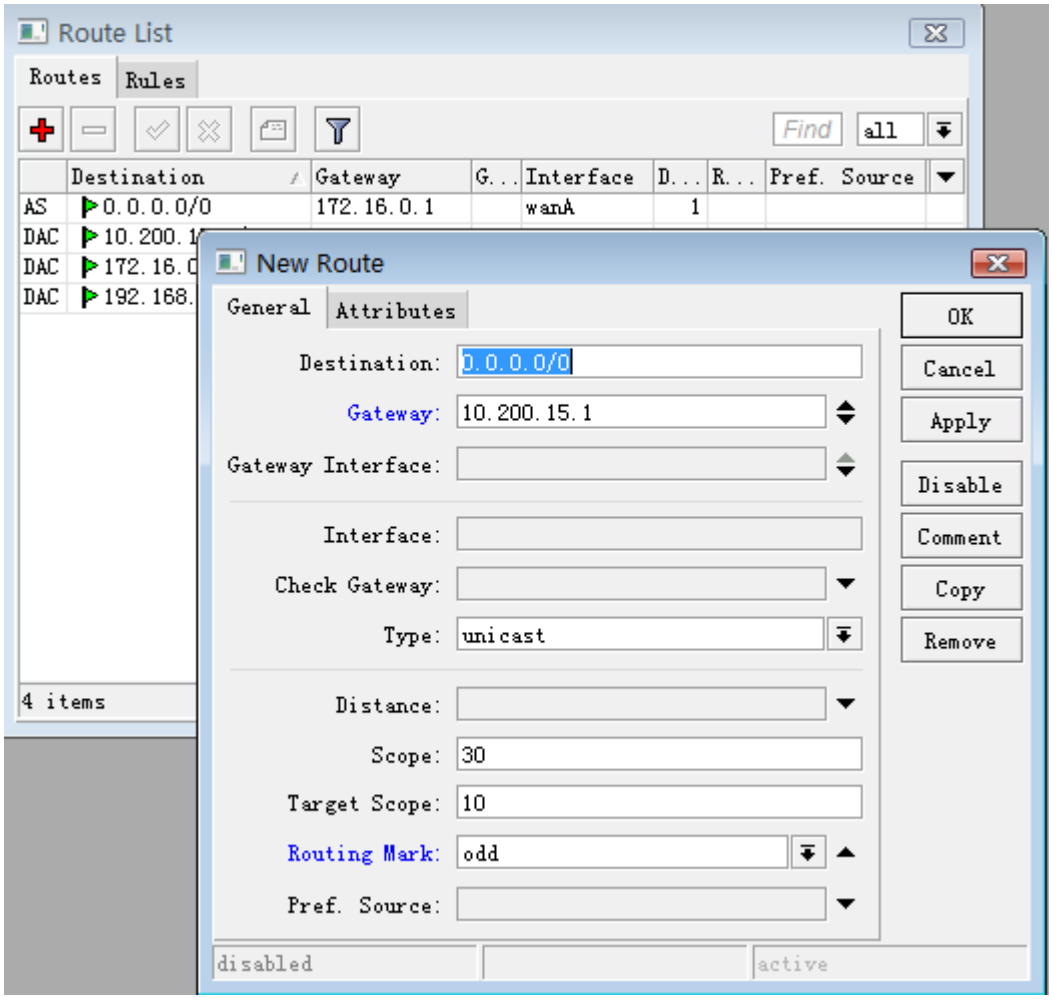
```
[admin@MIKROTIK] /ip firewall mangle> add chain=prerouting action=mark-routing new
-routing-mark=odd src-address-list=odd
[admin@MIKROTIK] /ip firewall mangle> print
Flags: X - disabled, I - invalid, D - dynamic
 0   chain=prerouting action=mark-routing new-routing-mark=odd passthrough=yes
     src-address-list=odd
```



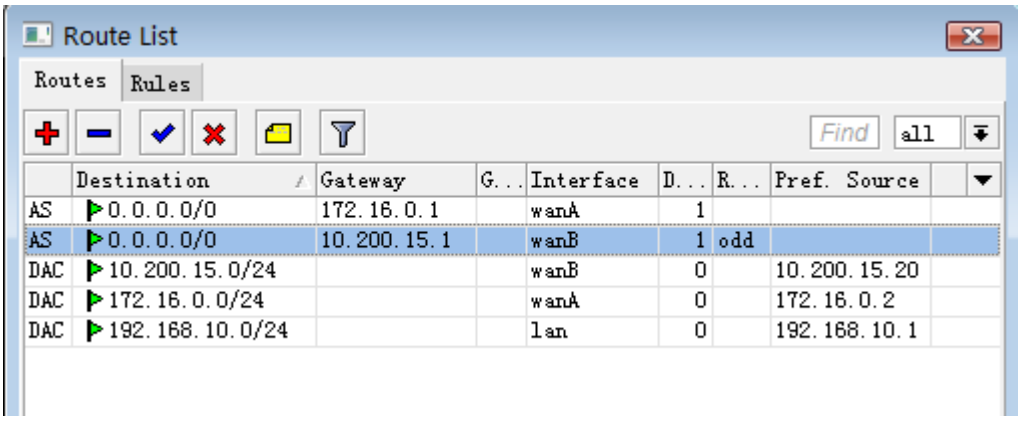


因为只标记了奇数的 ip 地址，剩下的便是偶数的，所以不用再做配置。

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 B 的网关即可，操作如下配置地址如下：



配置只需要添加 gateway=10.200.15.1 和 routing-mark=odd，点确认：



奇数的 IP 地址便从 B 线路的 10.200.15.1 的网关出去，剩下的偶数 IP 地址从默认的线路 A 的 172.16.0.1 的网关出去。

6.6 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 的文件上传到路由器的 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”的关键字替换为 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，

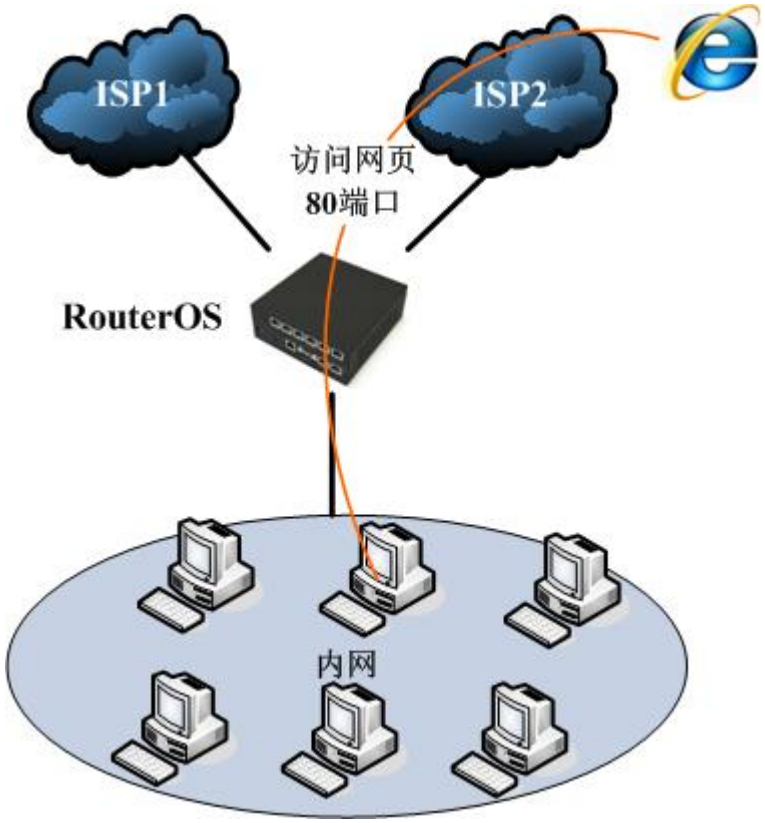
需要禁用掉 Tel 网关的静态路由策略，让所有的数据走默认的 Un 线路，Tel 网关为：222.212.48.1。脚本设置如下

```
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。

6.7 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：

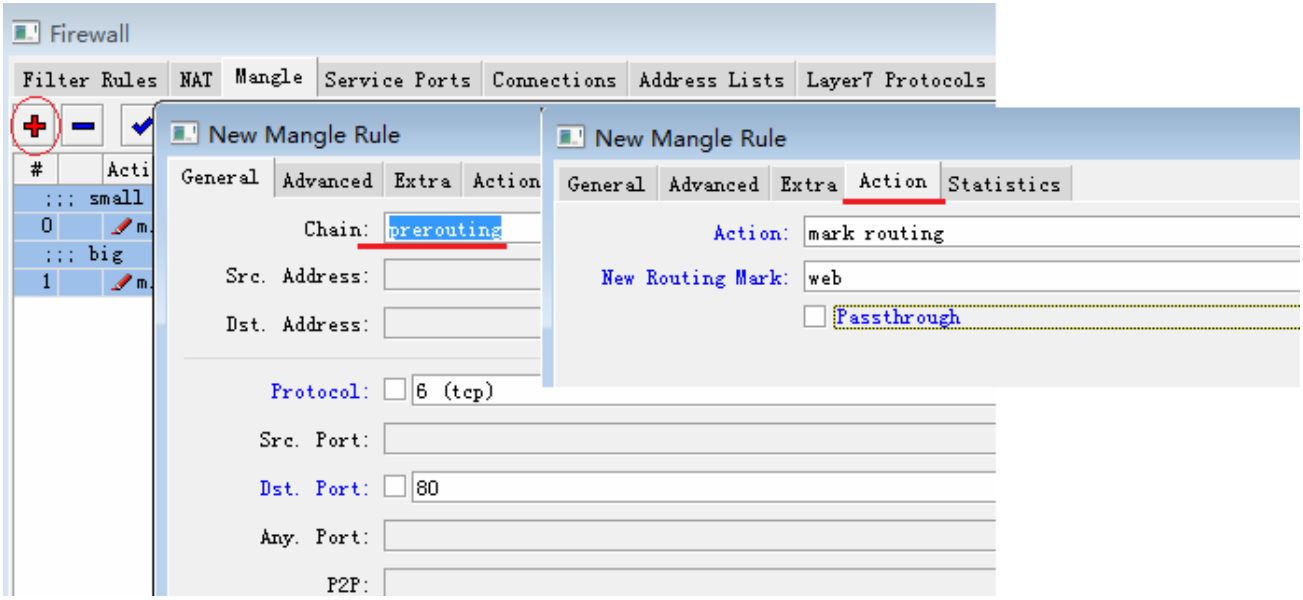


网络情况：

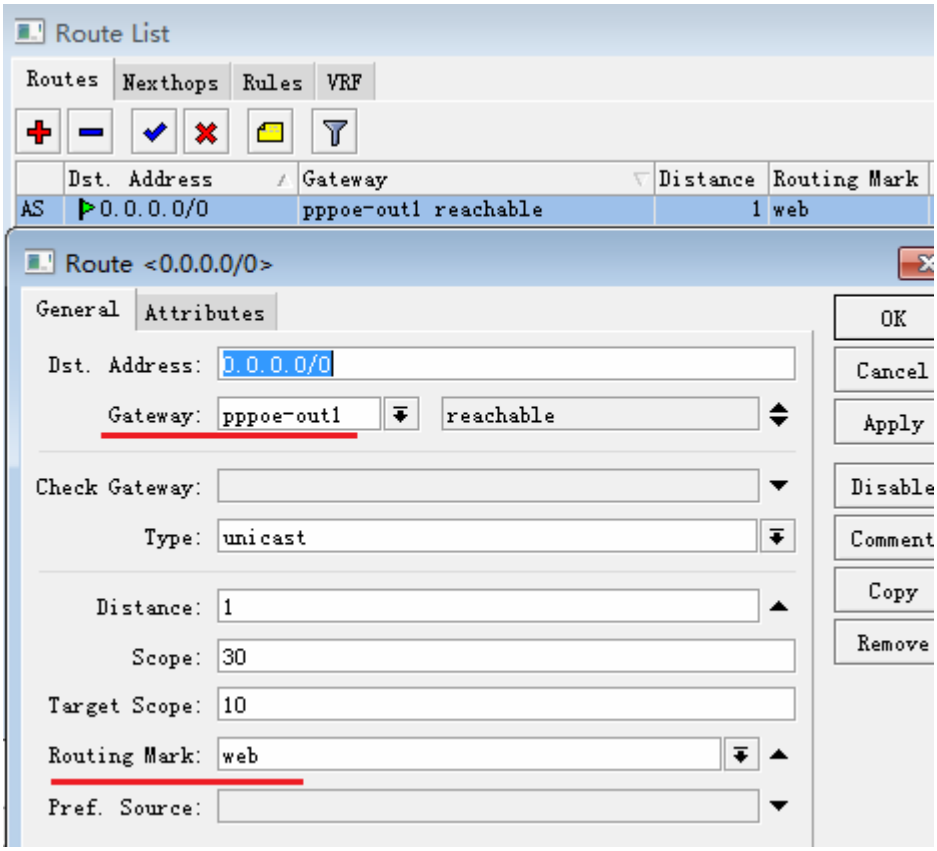
我们有两个 ISP 接入的线路，一个是 ISP1 通过光纤接入，另外一个 ISP2 的 PPPoE 拨号的 ADSL 连接，我们需要通过将访问网页的数据都转移到 PPPoE 拨号上，其他的的数据默认走 ISP1 的光纤（注意，转移 80 端口网页到 ADSL 上，也要注意 DNS 的配置，因为 DNS 解析关系到网站关联的 IP 地址，最好设置 ADSL 的分配的 DNS）。

实际配置

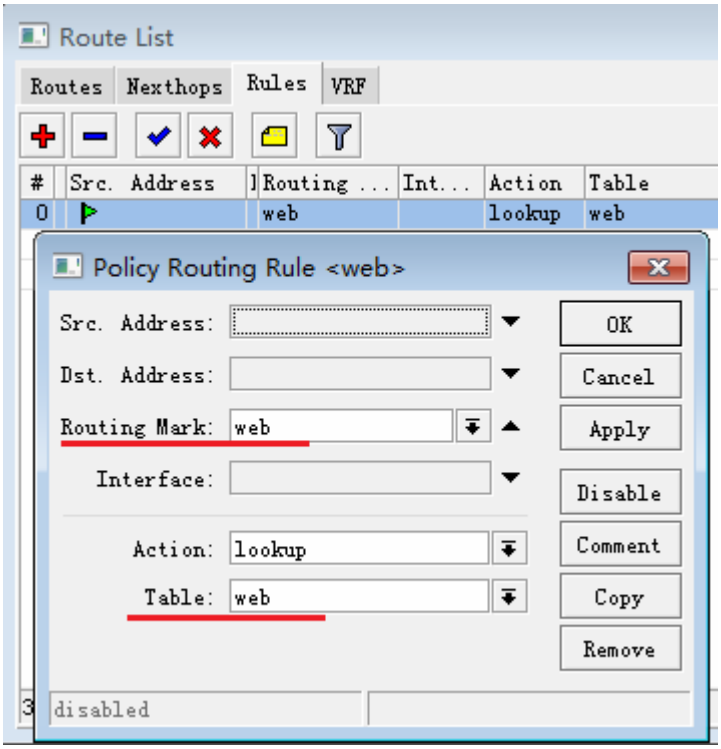
现在我们定义访问网页的端口，访问网页的端口是 TCP 80 端口，我们进入/ip firewall mangle 中做数据标记，从标记中提取路由标记，命名为“web”，因为我们在前面的连接标记中做过了 passthrough 的设置，在这里就不用在重复设置。



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：



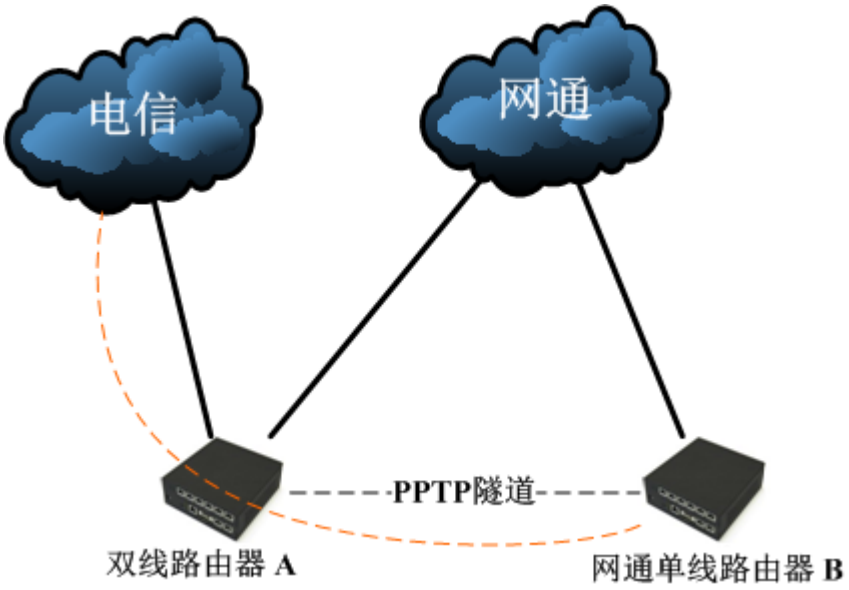
在 ip route rules 定义的 web 标记在 web 路由表中去查找路由。

注：当配置电信或联通的双线，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，需要注意一下几点：

1. 如果你设置 TCP/80 的网页走电信线路，请将用户请求的 DNS 设置为的电信 DNS，因为 DNS 用电信会解析返回的 IP 是电信地址，如果用联通 DNS 则会导致解析 IP 地址是联通，但却走到电信线路。如果 TCP/80 网页指定联通则设置为联通 DNS。
2. 确认 DNS 的 IP 地址是否走到正确的电信或者联通线路。

6.8 PPTP 借线路由操作

假设一个双线路由器有 Tel 和 Un 两条线路，并做了以 Un 为主，Tel 为静态路由策略设置。而另一个接入点的单线 Un 路由器接入了 Un 的线路，并且想通过 PPTP 隧道的方式借用双线路由器的 Tel 线路，现在看下面的图例

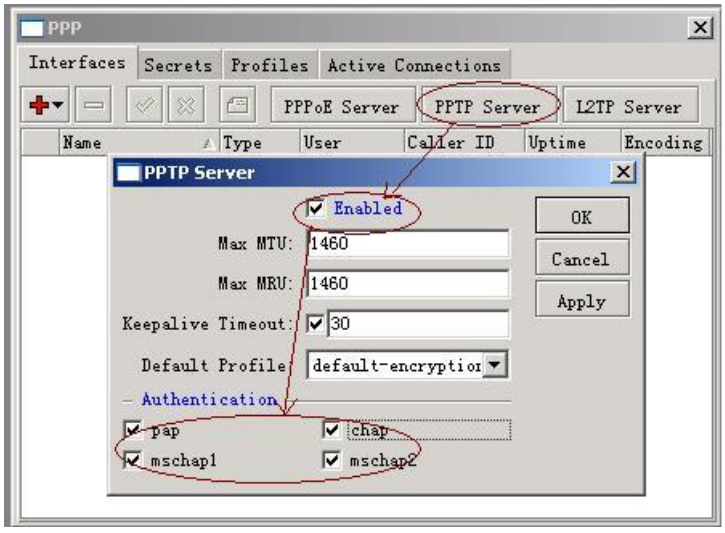


根据上面的案例，接入点 A 和 B 他们都是共同使用了 Un 的线路，这里 Un 两个点之间的延迟小于 10ms，网络延迟小才能保证足够的网速给 B 做 Tel 的访问。首先建立从接入点 B 到 A 的 PPTP 隧道，我们在接入点 A 设置 PPTP 服务器，在接入点 B 设置客户端。这里接入点 A 的 Un IP 地址为 202.112.12.10，BUn 地址为 202.112.12.12。

注：以下方案配置实例基于 2.9 界面，配置参数完全通用于所有 RouterOS 版本

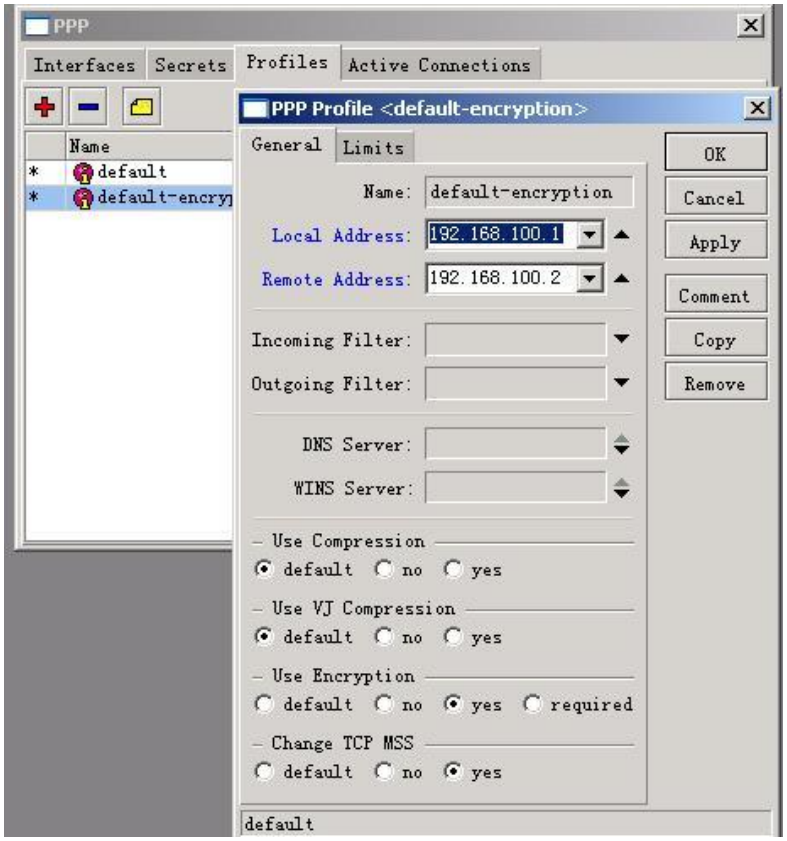
配置 PPTP-Server

在接入点 A 启用 PPTP-Server，并设置密码传输的加密类型：



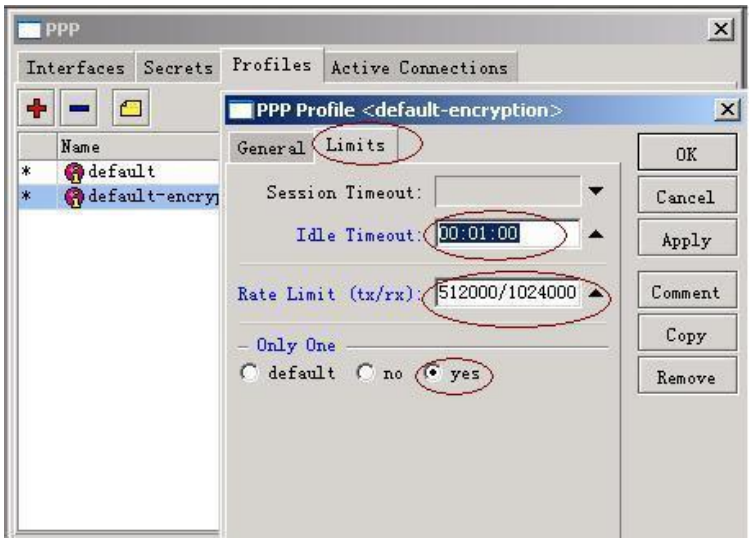
在这里 Default-Profile 我们采用 default-encryption，同样你也可以在 PPTP-Server 的 profiles 中创建自己的规则。Keepalive-Timeout 是 PPTP-Server 主动使用 ICMP 协议探测客户端是否在线，如果客户端使用了防火墙或禁止 ICMP 探测，那无法探测到客户端，Server 就会主动断开该客户端的连接，这个设置需要用户自己根据网络情况判断。

设置 Profile 定义客户和主机的访问地址：



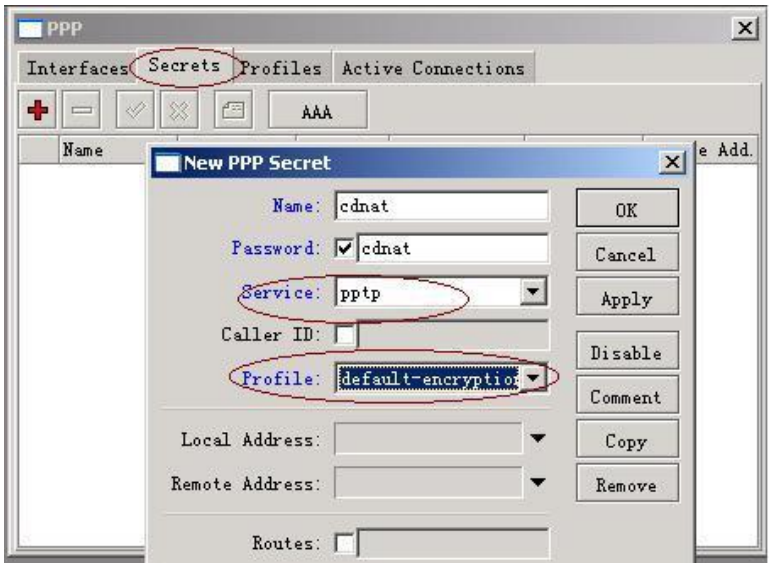
在这里我们给 PPTP-Server 分配的 IP 地址为 192.168.100.1(local-address) ，给客户端分配的地址为 192.168.100.2(remote-address) 。分配 IP 地址也可以通过账号设置 Secrets 进行，在这里我们只有一个客户端所有可以直接通过 profile 中的规则设置，如果有多个客户端也可以通过/ip pool 中的地址池做 DHCP 的分配。

配置 limit 参数：



在 limit 参数中，我们可以看到 idle-timeout，这个是客户端在没有流量超过 1 分钟后，就断开客户端。Rate-limit 是对该类用户的流量控制这里设置的上行为 512K，下行 1M 的带宽。最后是 only-one 该账户是否为唯一，这里设置为 yes。

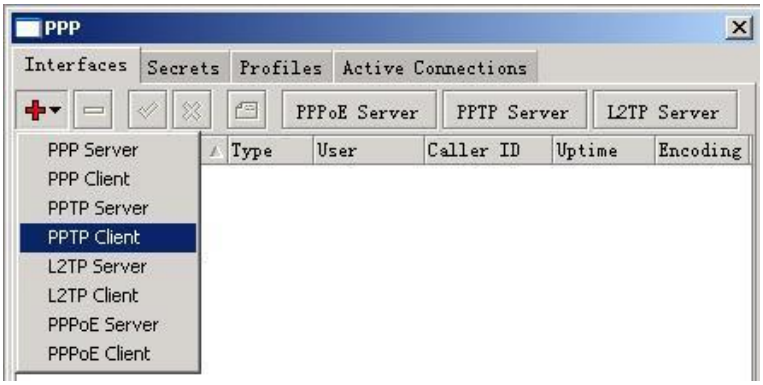
设置客户端的账号密码：



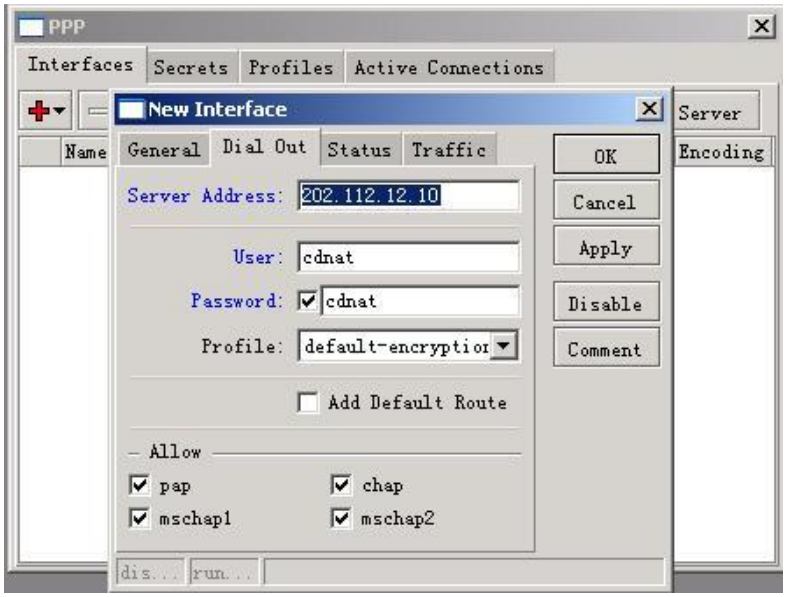
进入 secret 设置账号和密码以及相关信息，设置好 name 和 password 后，选择 service 服务类型为 pptp，profile 规则为 default-encryption。这样 PPTP-Server 就已经设置完成。

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 选项添加 PPTP-Client：



进入 dial-out 设置 PPTP 拨号信息，在 server-address 的地址为 202.112.12.10 级接入点 A 的 Un 地址：

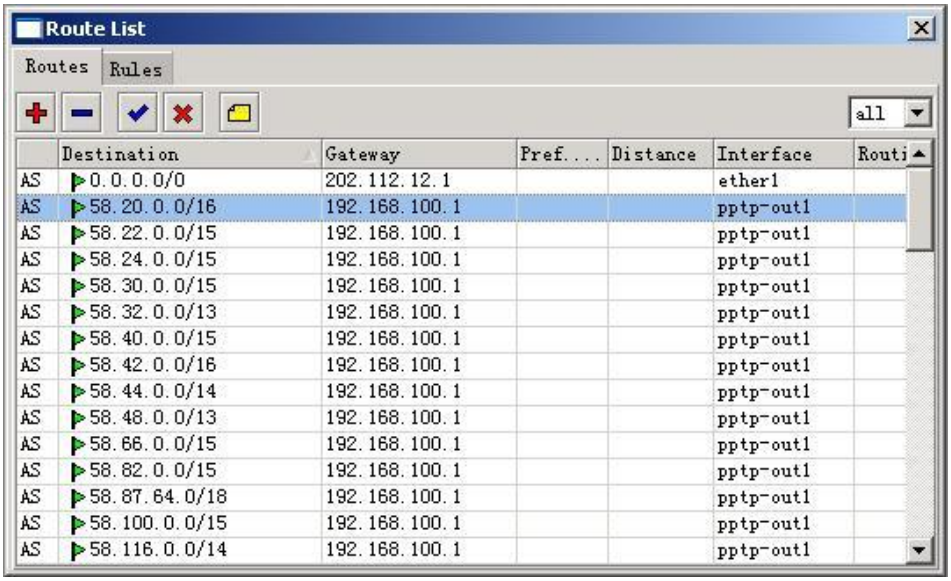


设置账号和密码分别为 MikroTik，设置完成后，便可以与接入点 A 的 PPTP-Server 连接。

路由配置

在这里接点 A 和 B 都做了 IP 地址的 NAT 转换，且接点 A 已经做了 Tel 的静态路由规则，即 A 点可以实现访问 Un 和 Tel 的分流，在 A 点不需要在做任何设置。B 点就需要指定通过 AB 两点间的 PPTP 隧道到 Tel 的线路，他指定的网关为 A 点的 PPTP 的 IP 地址（192.168.100.1）

设置 Tel 访问的网关：



通过编辑 Tel 的路由脚本，并导入路由表中，则实现了通过 PPTP 隧道使用 A 接入点的 Tel 线路，完成了借线功能。

6.9 RouterOS 策略路由

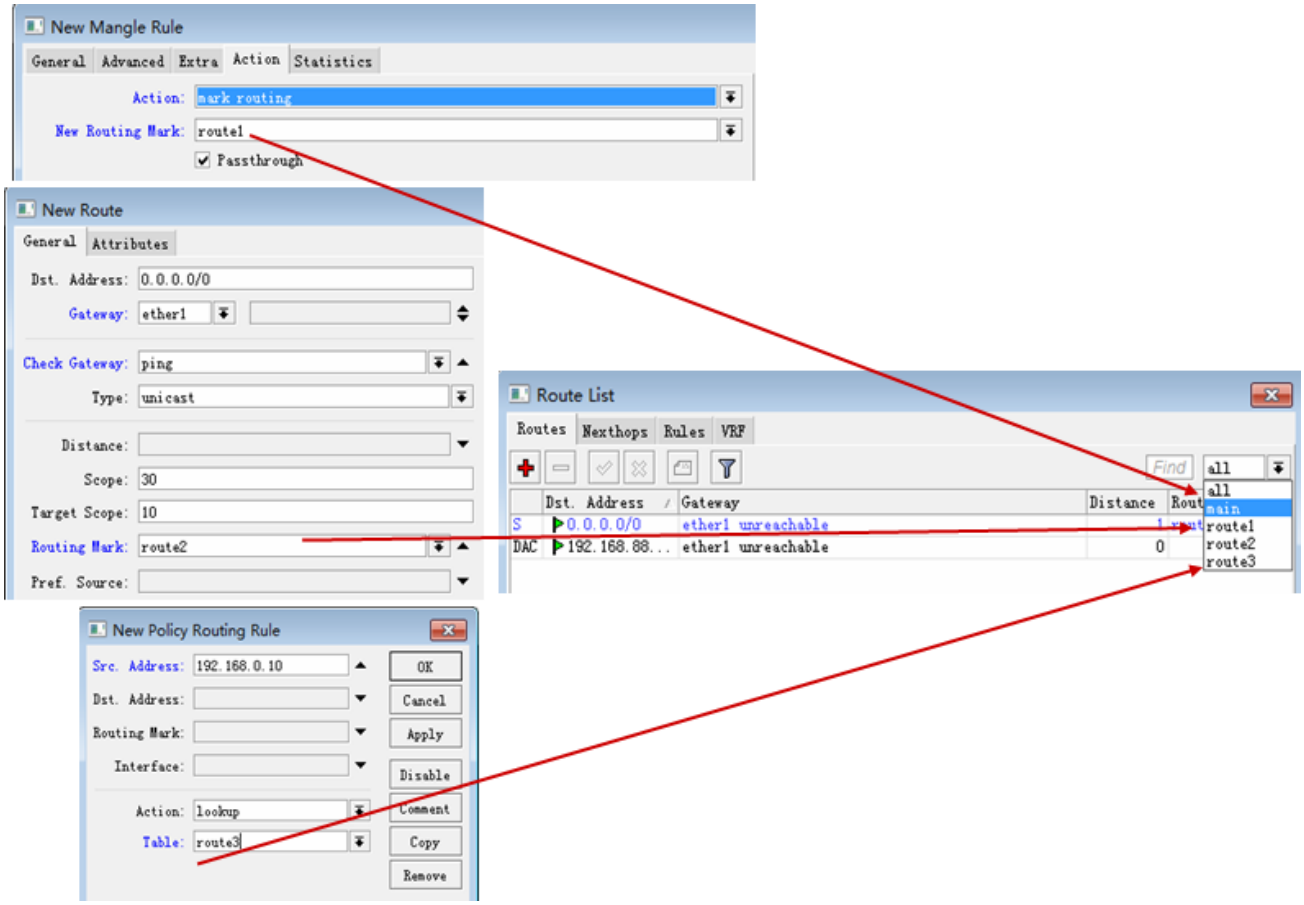
RouterOS 能维护多个独立的路由表，能灵活的分配策略路由规则，我们可以在多个菜单下通过命令标记路由与定义路由策略表，策略路由可以让我们有选择性的调度 IP 数据包到不同的网关出口

- /ip firewall mangle mark-routing （支持源目标和端口路由）
- /ip route routing-mark （支持源目标路由）
- /ip route rule table （支持源目标路由）

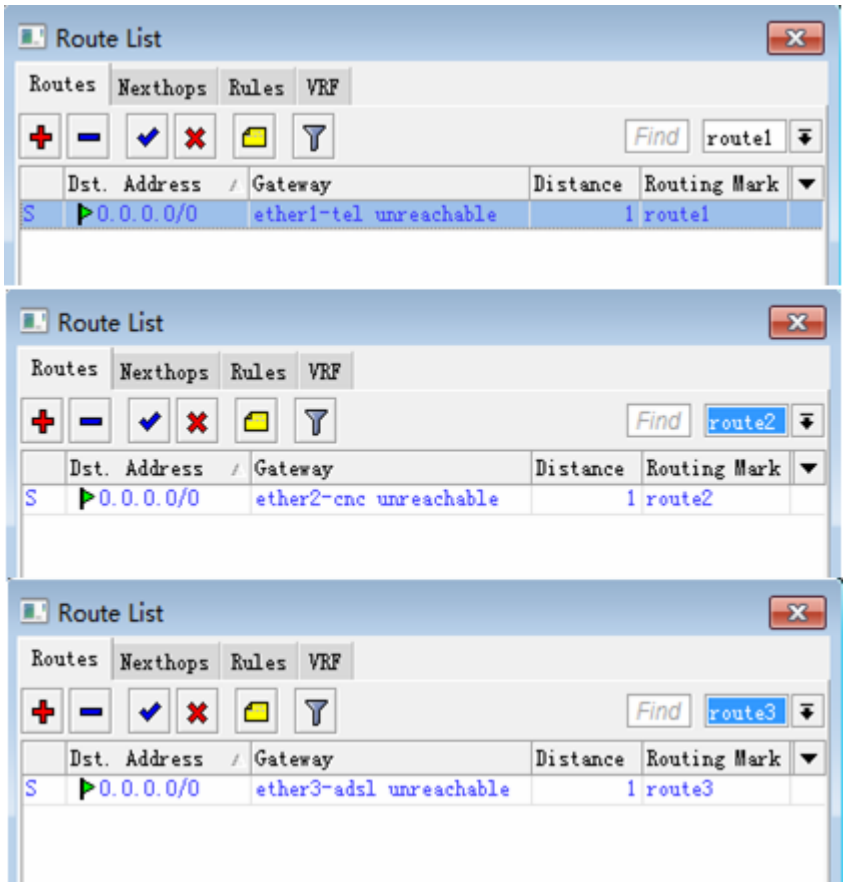
他们之间关系是平等的：

mark-routing = routing-mark = table

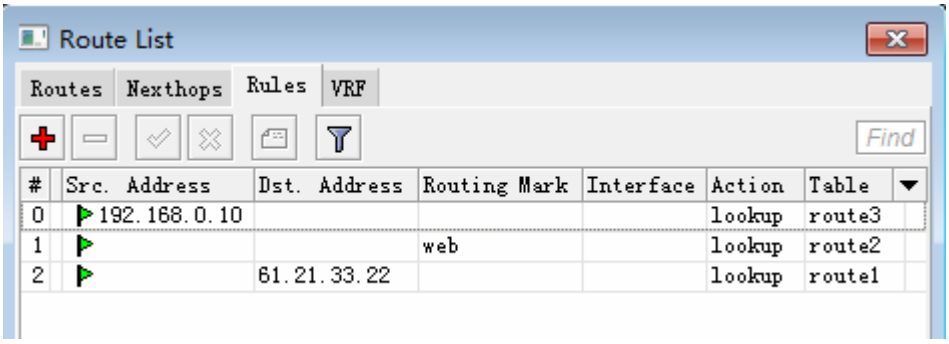
从下面的的操作图中可以看到，在 ip firewall mangle、routing-mark 和 table 中的建立的路由表，可以在 ip route 的右侧列表找到对应的路由表



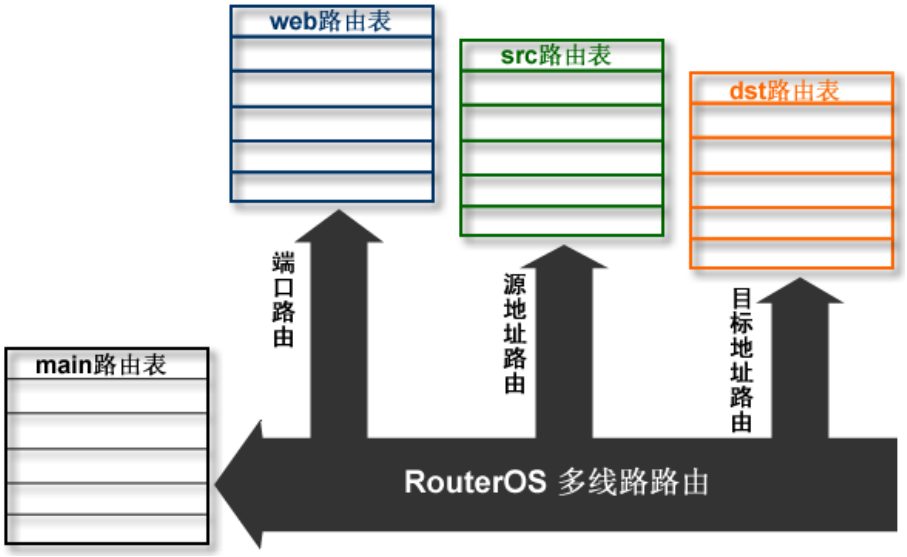
我们选择这些定好的路由表 route1、route2 和 route3 等，给他们定义各自的路由和网关，但他们并不是缺省的默认网关，而是建立在路由标记下的网关



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：



当他们被定义后都会在 ip route 中新建路由表，如图：



如果建立了多个路由表，RouterOS 会首先处理新建的路由表，最后剩下的数据到 Main 表，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。

6.10 RouterOS 负载均衡

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，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，PCC是最新的负载均衡功能，其简单、有效、易扩展，且没有严重的副作用，通过PCC可以更好的实现路由权重调配。

PCC 是 RouterOS 最新，最主要的负载均衡策略：

- 使用哈希算法先对基于源地址/端口、目标地址/端口或不同组合的分类；
- 再使用 mangle 分类数据包和路由标记；
- 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 数据包是将数据封装在内，connections 用于连接运输数据包，如同数据包是火车的每节车厢，connections 则是铁轨，Connections 可以选择不同的连接方式（TCP/UDP）到达目的地。

注：关于 Nth 负载均衡在后面的章节会单独介绍。

PCC 负载均衡

根据官方在 wiki 页面解释“PCC 从一个 IP 数据包头选取字段，并将这些字段转换为 32-bit 值用于哈希算法。假如一个数据包头获取指定的字段，给转换后取得的值指定一个分母，并计算得到一个余数，然后会去与一个指

定的余数对比，如果相等该数据包会被命中。数据包头可以选择的字段如 src-address, dst-address, src-port, dst-port 等”下面是 RouterOS 完整的可用字段：

- both-addresses
- both-ports
- dst-address-and-port
- src-address|src-port
- both-addresses-and-ports
- dst-address|dst-port
- src-address-and-port

这里需要理解一些 PCC 的相关条款

IP 数据包的包头包含了许多字段，其中有两个字段分别是 IP 的源和目标地址，TCP 和 UDP 数据包，在 TCP 和 UDP 包头中包含有源端口和目标端口。

首先 PCC 应用会将指定的字段发送给哈希函数，例如提供 IP 地址和端口，将 IP 地址和端口的每位作为十进制，取其整数，例如源 IP 地址是 1.1.1.1，源 TCP 端口是 10000，目标 IP 地址是 2.2.2.2 和目标端口 80，当我们将这些数字相加 1+1+1+1+10000+2+2+2+2+80 = 10092，根据官方的所诉，10092 最后一位数字是 2，因此哈希输出为 2，这组 IP 和端口组合的哈希值每次输出都是 2。这里最重要的是相同 IP 端口组合的哈希结果是一样的，我们利用该属性来作为 PCC 负载均衡算法的基础。

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=src-address-and-port:3/0

/ip firewall mangle add chain=prerouting action=mark-connection \
  new-connection-mark=2nd_conn per-connection-classifier=src-address-and-port:3/1

/ip firewall mangle add chain=prerouting action=mark-connection \
  new-connection-mark=3rd_conn per-connection-classifier=src-address-and-port:3/2
```

我们依次看看三条规则

- 第一条规则表示：哈希函数的输出值是取源 IP 地址和端口（src-address-and-port），得到的哈希值除以 3，余数为 0，执行的操作标记为 1st_conn。
- 第二条同理，哈希值除以 3 ，余数为 1，并标记为 2nd_conn，
- 第三条规则，哈希值除以 3，余数为 2，执行标记为 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，我们可以取关键字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，例如

- 3%3=0，3 除以 3 余数为 0，3 能被 3 整除。
- 4%3=1，4 除以 3 余数为 1，4 不能被 3 整除，余 1。
- 2%3=2，2 除以 3 余数为 2，2 不能被 3 整除，余数 2

- 6%3=0，6 除以 3 余数为 0，6 能被 3 整除。

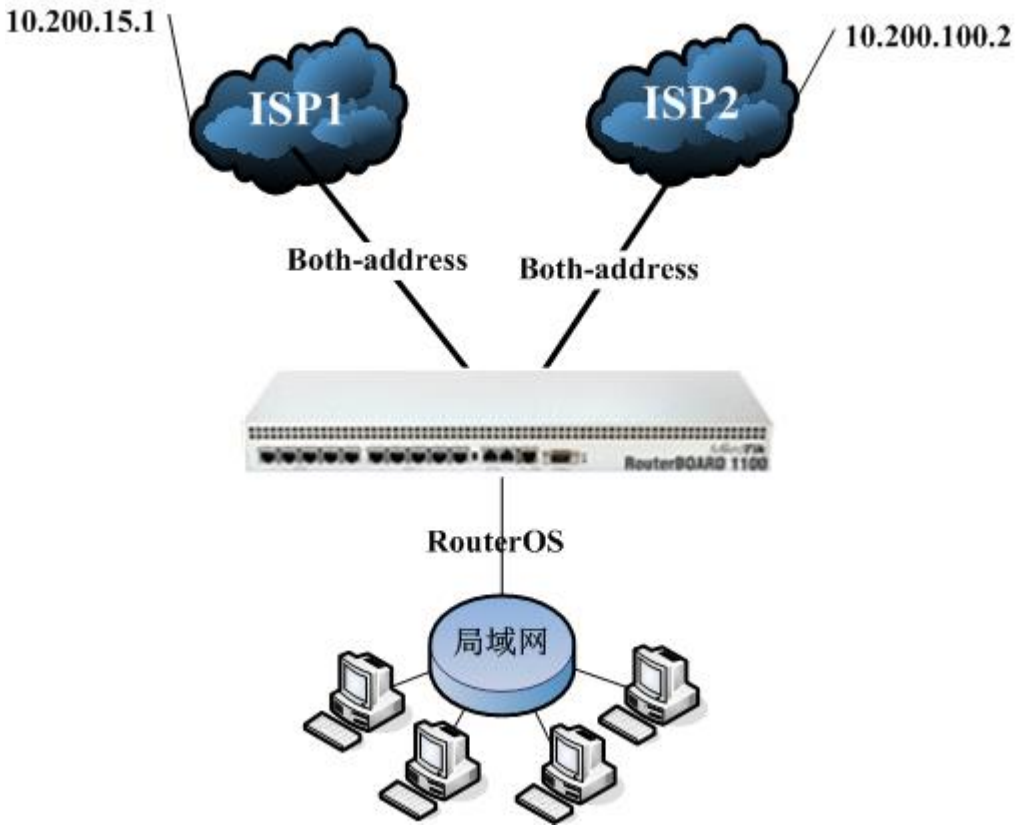
PCC 虽然能用于负载均衡，但 PCC 自身是无法直接用于路由，或 routing marks，PCC 仅仅只能用于数据包匹配，不能直接用于数据包标记。

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

一、 双向地址负载均衡

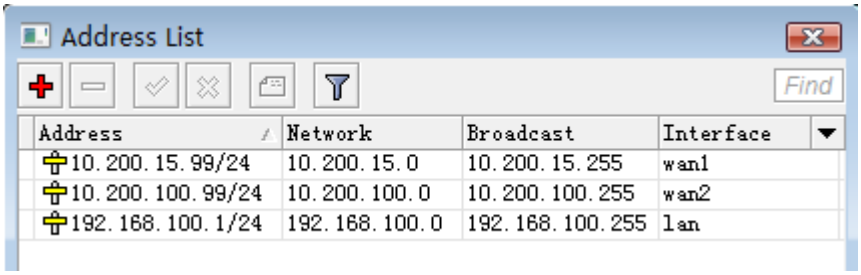
通分组源地址和源端口实现负载平衡，这里我们建立 2 个 WAN 出口分别是 wan1 和 wan2，网络环境如下：



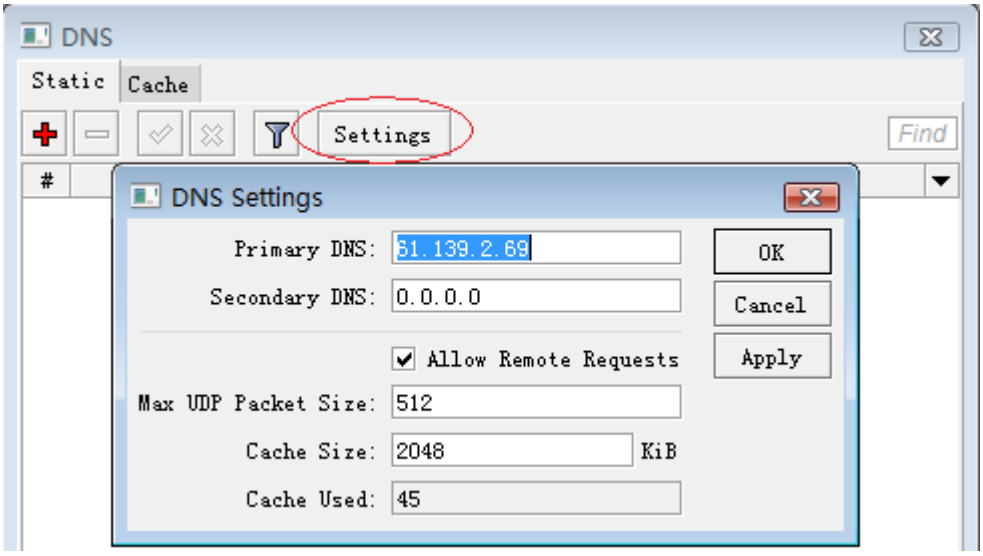
- ISP1 地址 10.200.15.99/24，网关：10.200.15.1；
- ISP2 地址 10.200.100.99/24，网关：10.200.100.2；
- 内网 IP 地址 192.168.100.1/24；
- 启用 DNS 缓存功能，用 192.168.100.1 作内网 DNS 解析；

基本配置

首先进入 ip address 配置 IP 地址：



在 ip dns setting 中配置好 DNS 缓存，DNS 为：61.139.2.69

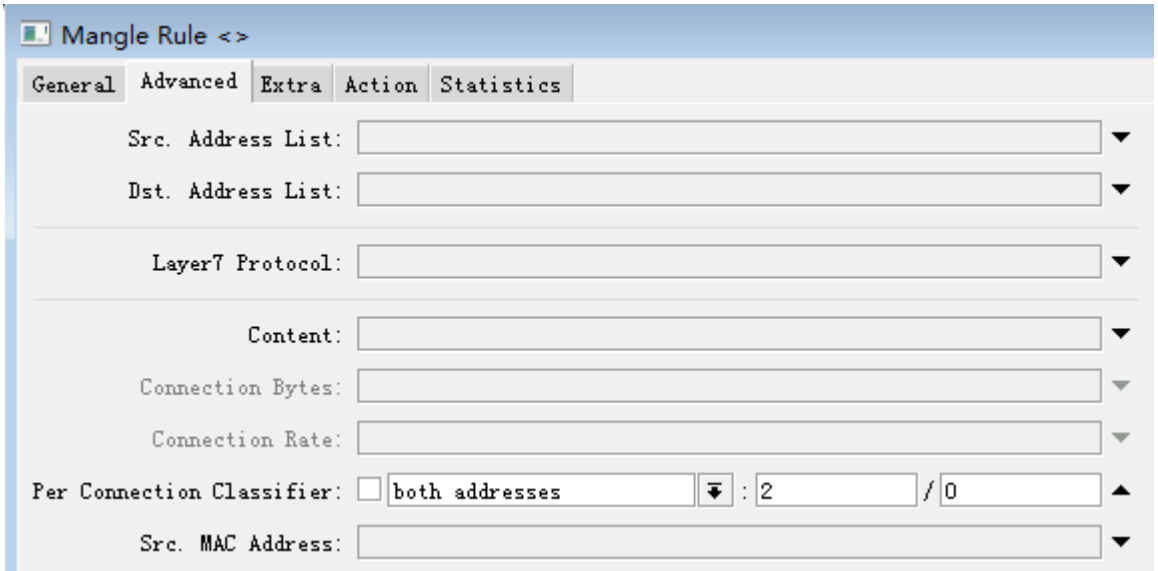


Mangle 标记配置

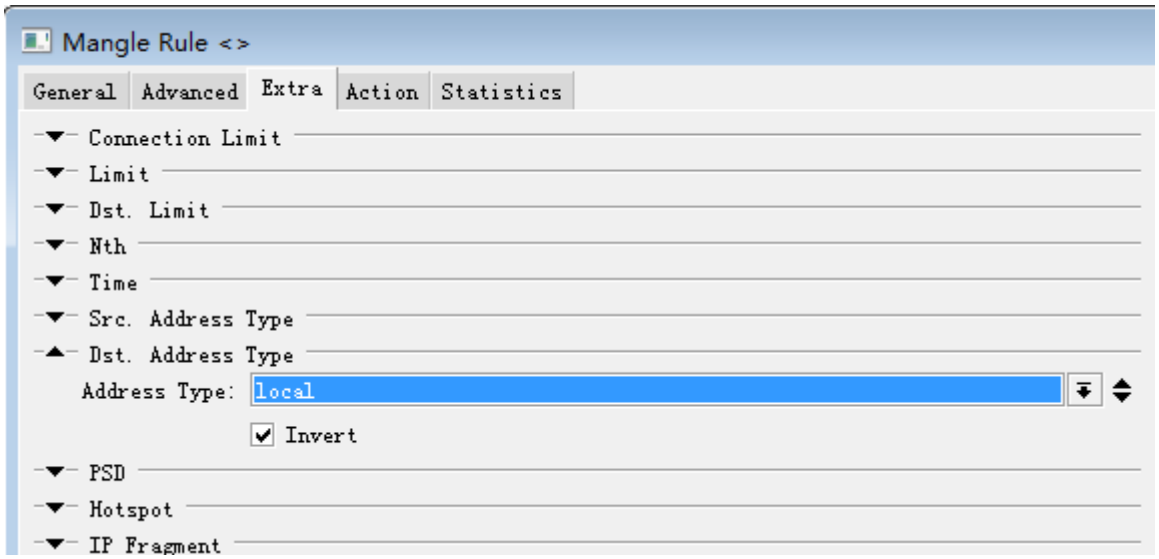
接下来我们进入 ip firewall mangle 标记连接和路由，我们使用 per-connection-classifier 双向地址进行分类做连接分类标记。

首先我们需要将进入路由的的连接进行标记

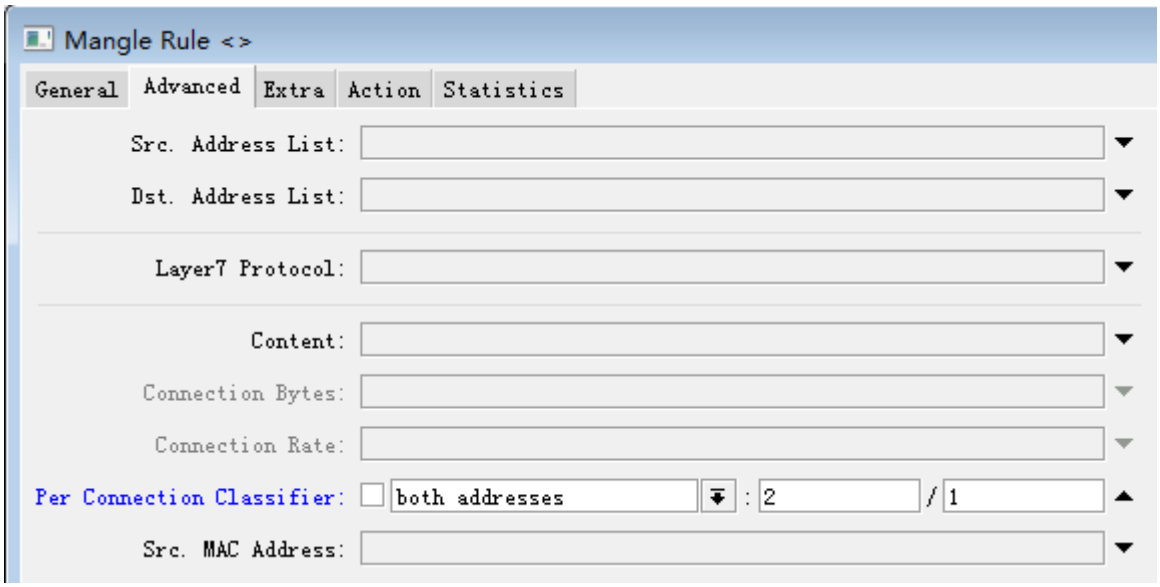
如下图，我们进入一条 mangle 规则，中的 advanced 标签内容可以看到 per-connection-classifier 分类器，选择 both-addresses 的分类：



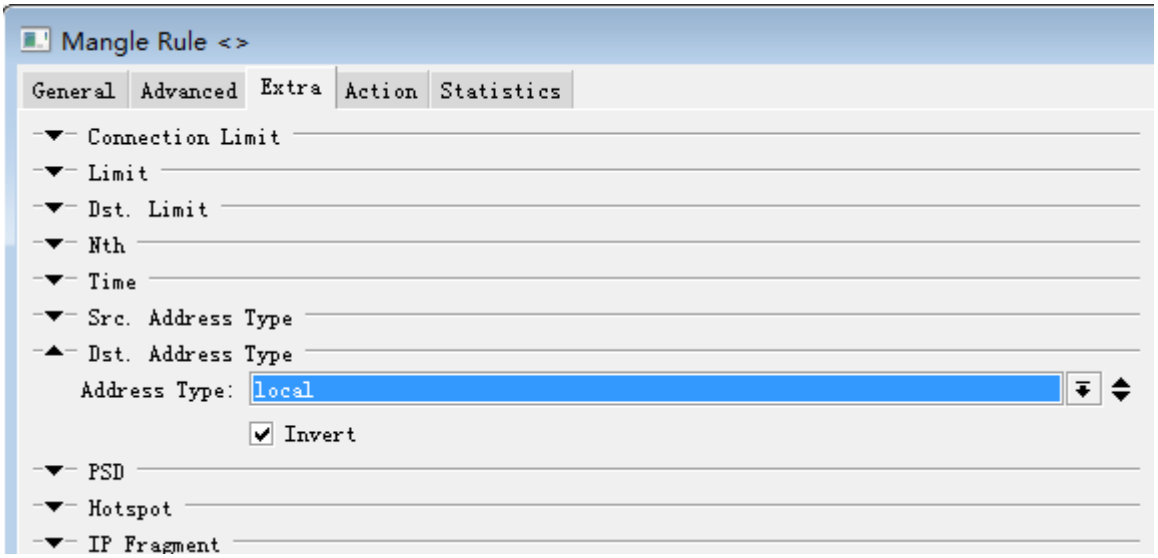
然后选择 dst-address-type=!local，即除了目标地址类型排除本地地址：



注：2 条线的分类代码定义是第一条线为 2/0，第二条为 2/1



同样选择一下地址类型：



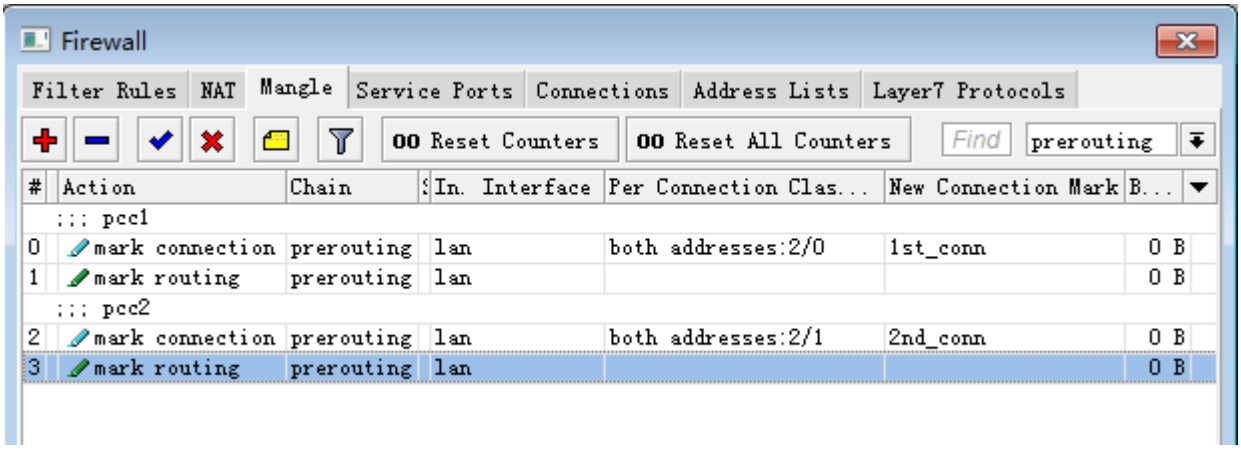
下面命令是提取走第一条线路的连接标记取名为 **1st_conn**，并从连接里提取路由标记名位 **1st_route**，设置：per-connection-classifier=both-addresses:2/0，设置 in-interface=lan

```
/ip firewall mangle
add action=mark-connection chain=prerouting comment="" disabled=no \
    in-interface=lan new-connection-mark=1st_conn passthrough=yes \
    per-connection-classifier=both-addresses:2/0
add action=mark-routing chain=prerouting comment="" connection-mark=1st_conn \
    disabled=no in-interface=lan new-routing-mark=1st_route passthrough=yes
```

提取走第二条线路的连接标记取名为 **2nd_conn**，并从连接里提取路由标记名位 **2nd_route**，设置：per-connection-classifier=both-addresses:2/1，设置 in-interface=lan:

```
/ip firewall mangle
add action=mark-connection chain=prerouting comment="" disabled=no \
    in-interface=lan new-connection-mark=2nd_conn passthrough=yes \
    per-connection-classifier=both-addresses:2/1
add action=mark-routing chain=prerouting comment="" connection-mark=2nd_conn \
    disabled=no in-interface=lan new-routing-mark=2nd_route passthrough=yes
```

在 winbox 的 mangle 中设置完成后如下：

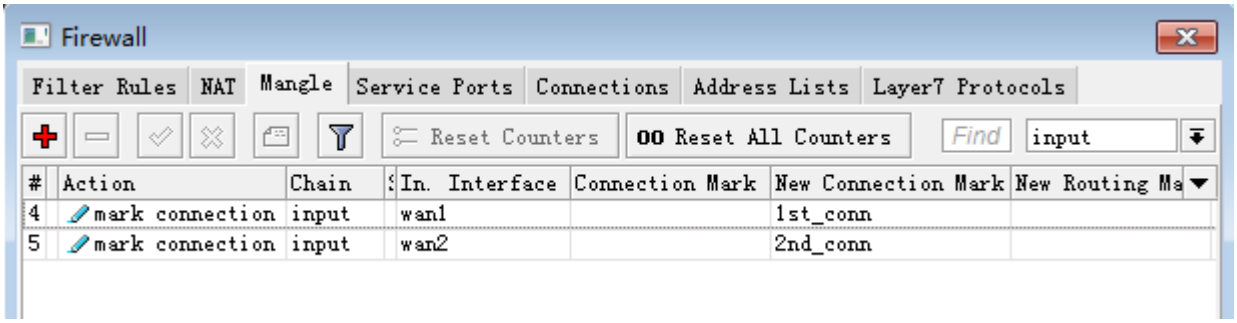


回程路由设置

定义数据包从那个接口进入，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=1st_conn
add chain=input in-interface=wan2 action=mark-connection new-connection-mark=2nd_conn
```

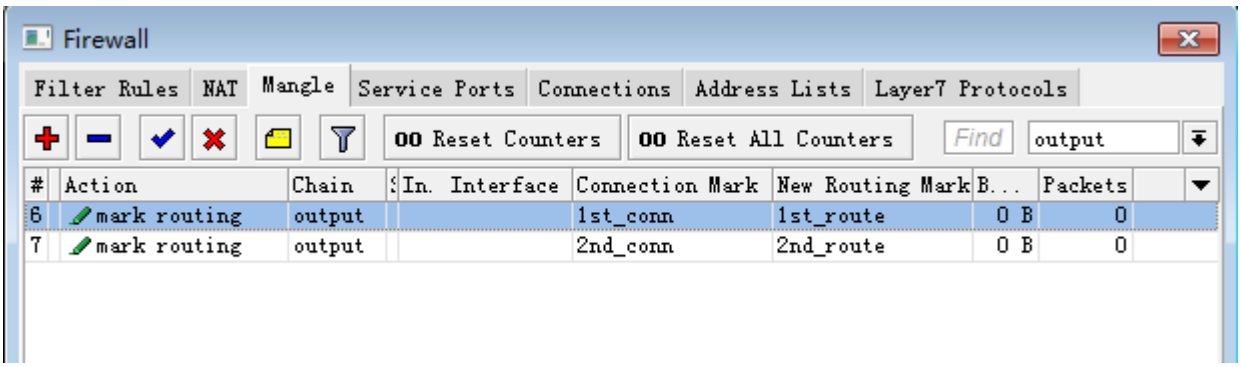
winbox 设置



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：

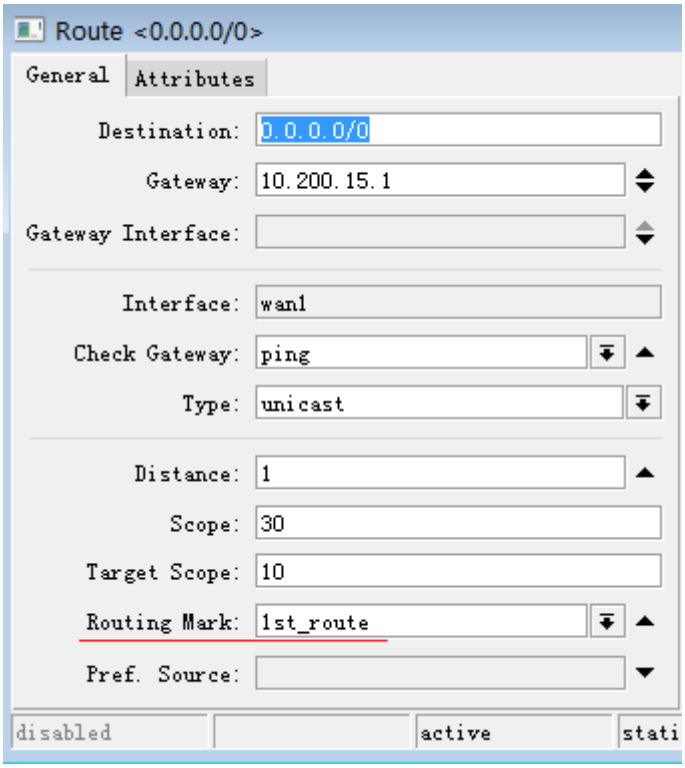
```
add chain=output connection-mark=1st_conn action=mark-routing new-routing-mark=1st_route
add chain=output connection-mark=2nd_conn action=mark-routing new-routing-mark=2nd_route
```

winbox 设置

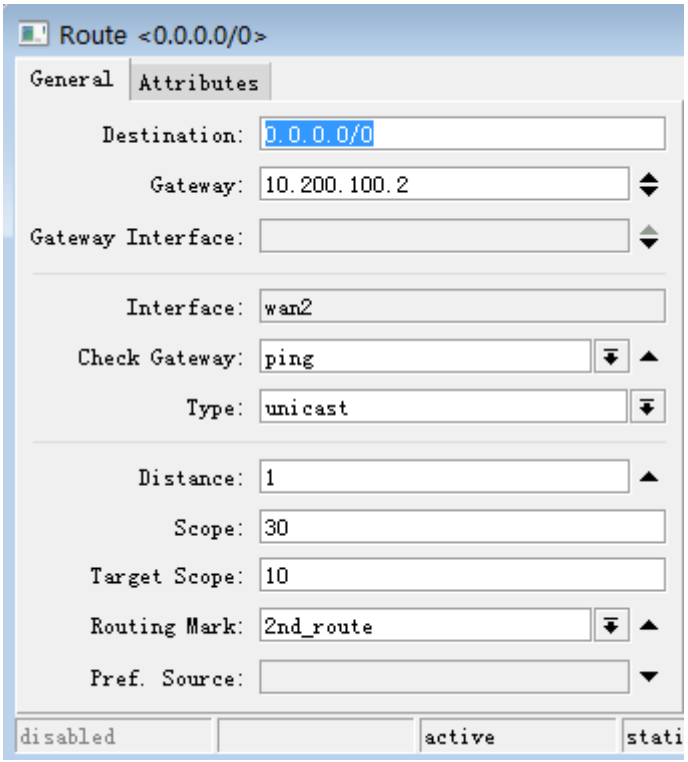


路由配置

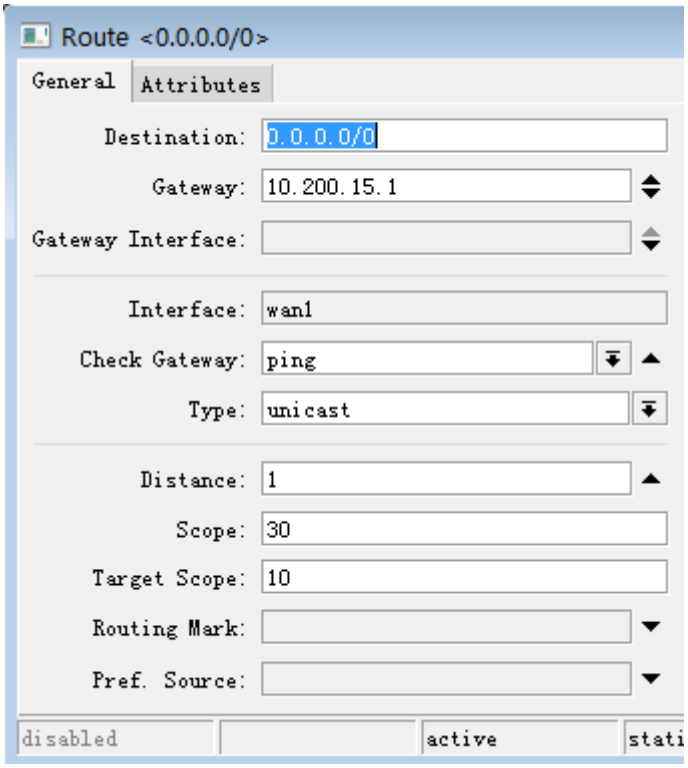
配置完标记路由后，我们进入 ip route 配置路由，首先设置负载均衡的标记路由，首先设置第一条线路的路由标记，设置 routing-mark=1st_route：



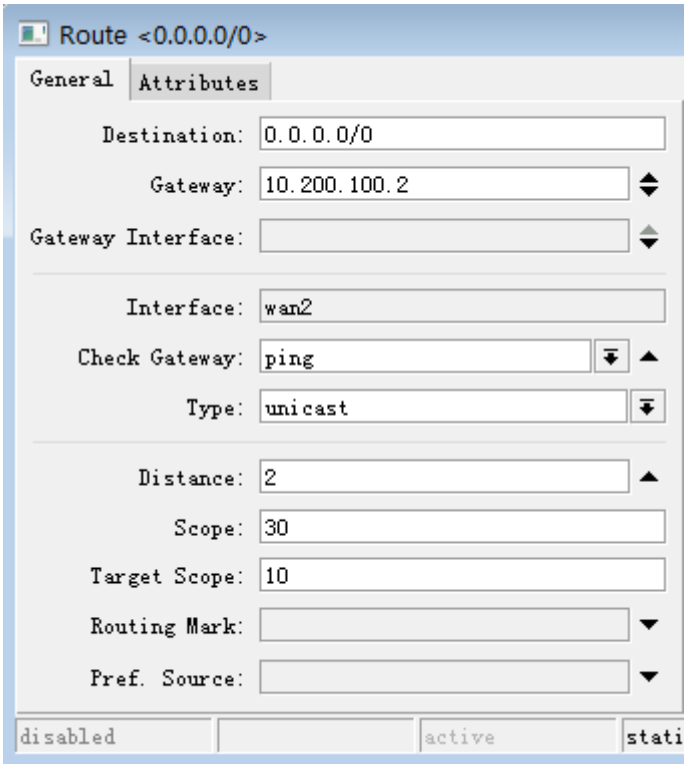
设置第二条线路的路由标记，设置 routing-mark=2nd_route：



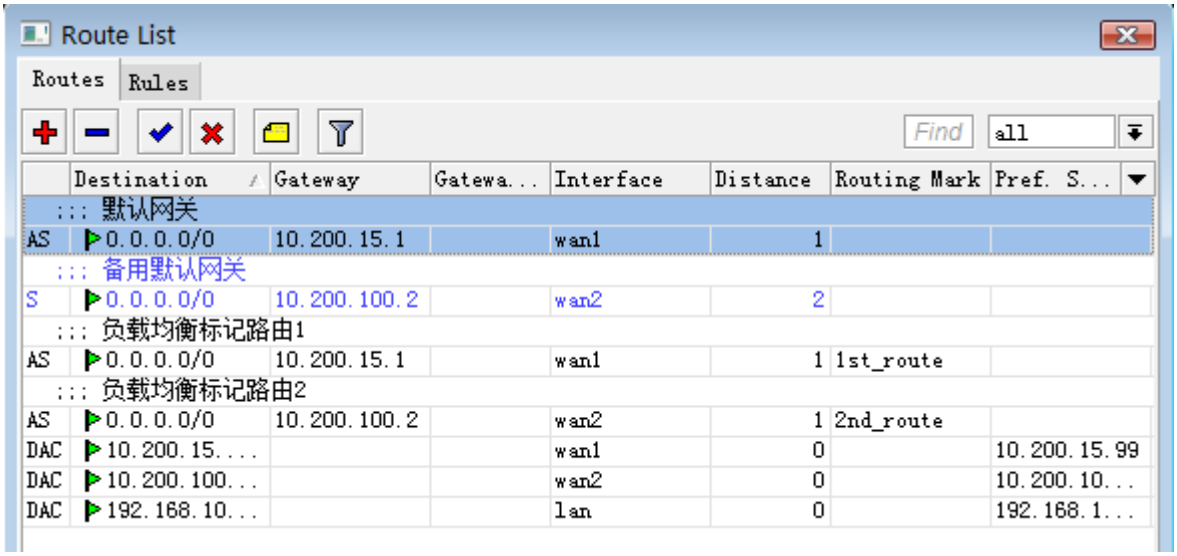
配置默认网关和备份网关，默认网关的 **distance** 设置为 1，并设置 check-gateway=ping，通过 ping 监测网关状态：



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：



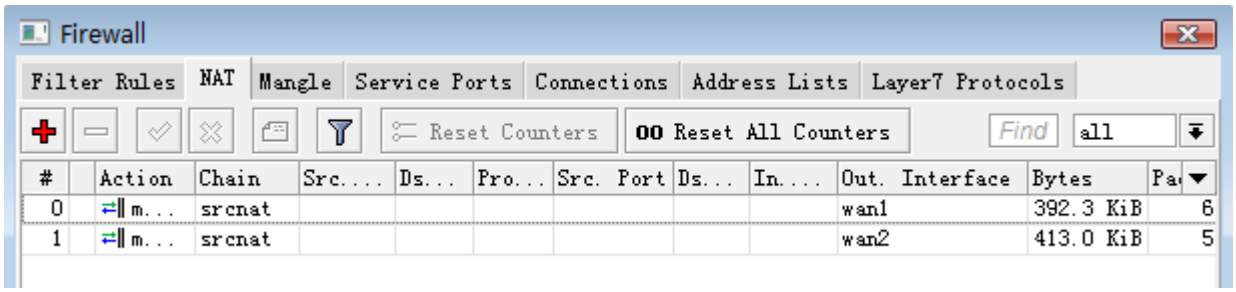
配置完成后的路由标如下图：



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：

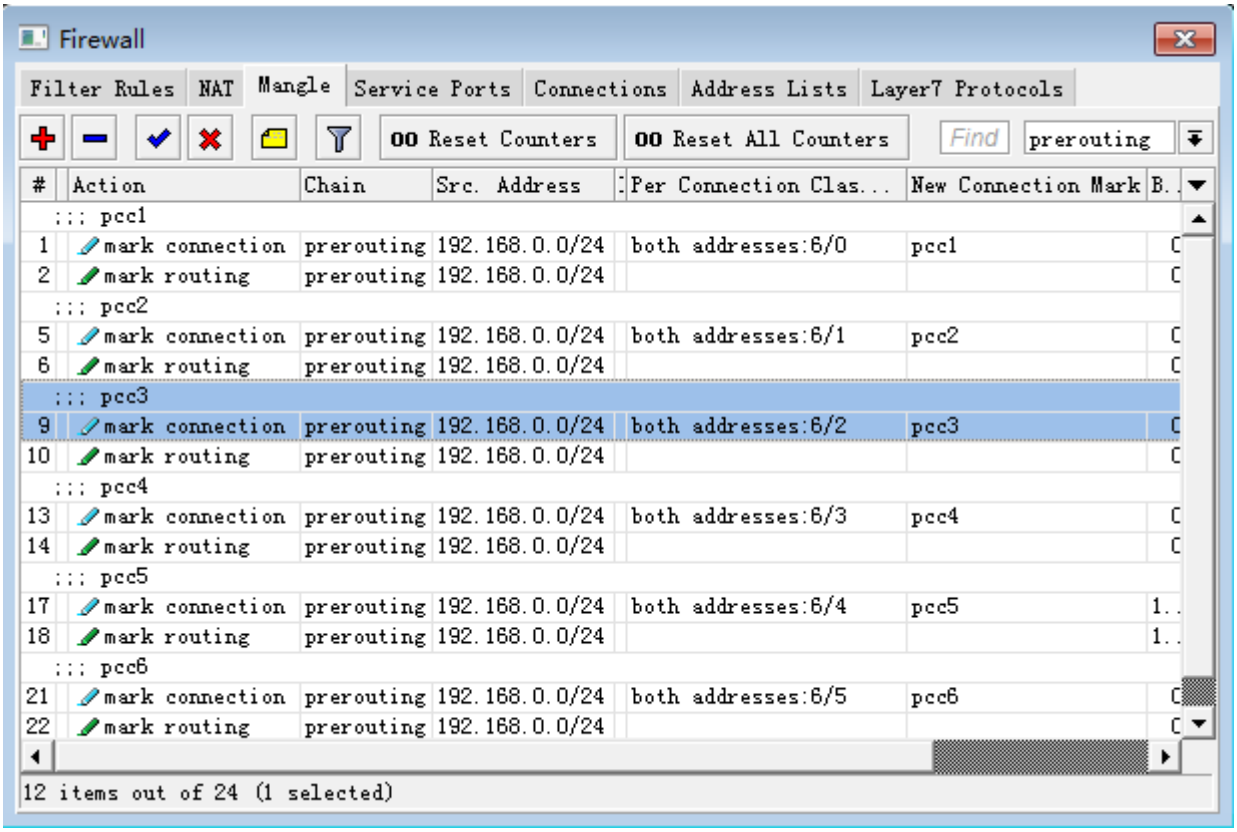
```
/ip firewall nat
add action=masquerade chain=srcnat out-interface=wan1
add action=masquerade chain=srcnat out-interface=wan2
```



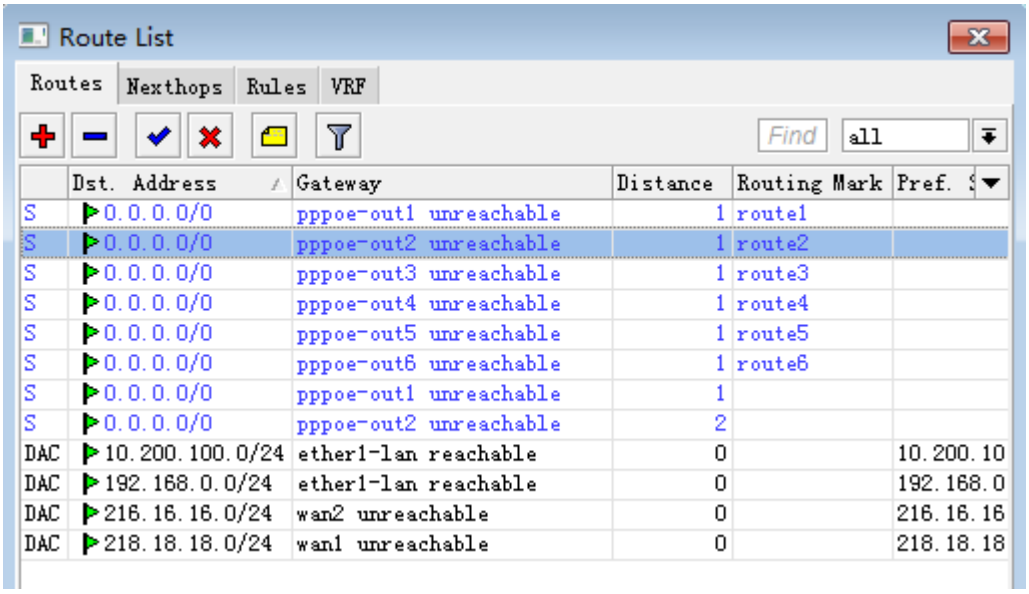
六线的 PCC 负载均衡

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 实现比例权重路由

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，想做策略将两条线路实现权重的路由策略，我们可以通过 PCC 来实现。

平常我们都是用 PCC 做多条相同带宽出口的负载均衡，而这次我则是通过他的分类原理实现比例权重的路由策略，当然 Nth 也可以实现，但 Nth 不如 PCC 稳定好用。

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（8/33：25/33），那就是要按照 1:3 的比例分配路由,我的策略是将 PCC 策略看成 4 份,然后路由指定按照 1:3 的路由规则分配。

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这里仅通过 CLI 命令讲解。

```
/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

其实权重的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=8M
add chain=srcnat action=masquerade out-interface=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。

6.11 多线接入的返程路由

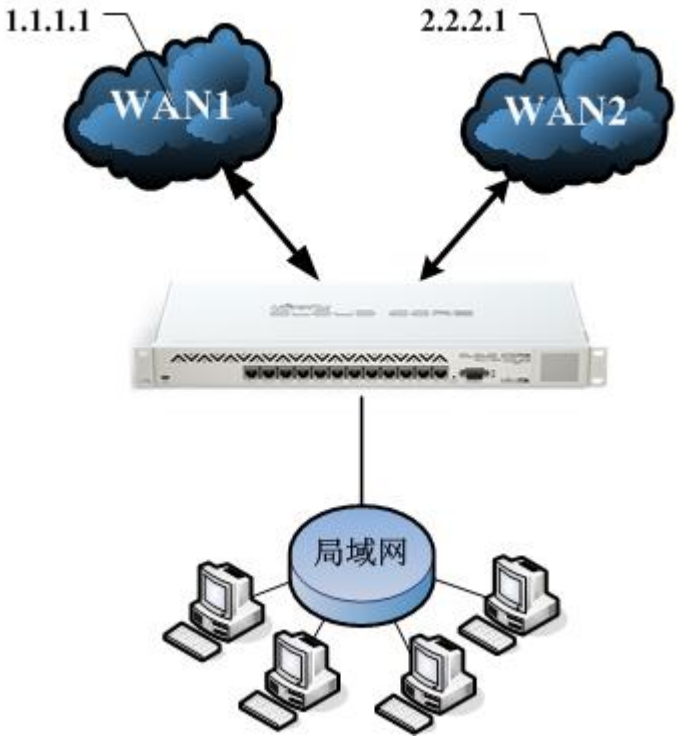
当做 RouterOS 的 PCC 多线路负载均衡的时候，我们会发现一组规则，在 ip firewall mangle 的 input 和 output 链表中出现的，这条规则用于非默认路由的返程，即从那个三层口到路由器，就从那个三层接口返回。通常路由器默认路由只有一条（除 ECMP 外），返程路由可以保证数据不会被拒绝。

默认路由（Default Route）：即网关设备的默认路由出口，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。

返程路由原理

下面具体点，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
```

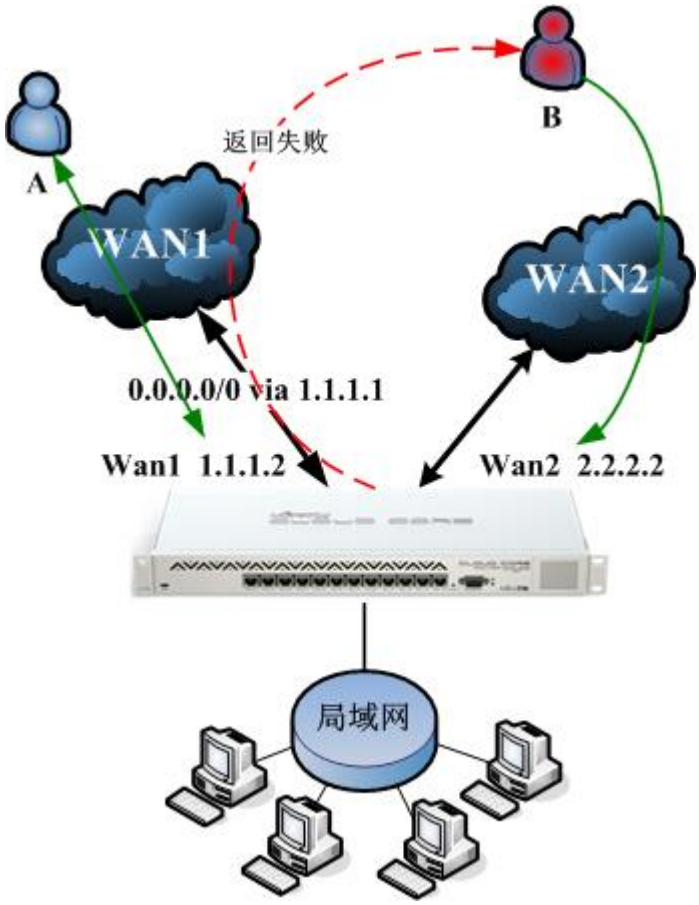
注意正常情况下，我们会配置一条默认路由，例如：

```
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
```

路由配置完成了，现在假设没有 input 和 output 的两组规则，会发生什么事情，参考下图：



上面的拓扑图有两个外部用户 A 和 B 需要访问路由器（该实例都以访问路由器自身为例，即对于路由器在 RouterOS 的处理链表为 input 和 output），路由器上双线接入，由于路由器默认网关只有一个，因此会出现 A 和 B 用户访问不同线路产生的结果不同。

当外部网络 A 访问 1.1.1.2 的时候，1.1.1.2 回复外部请求，整个 RouterOS 路由器的默认网关是 1.1.1.1，理所当然的会走 wan1 的 1.1.1.1 出去，从原路进来，从原路返回。

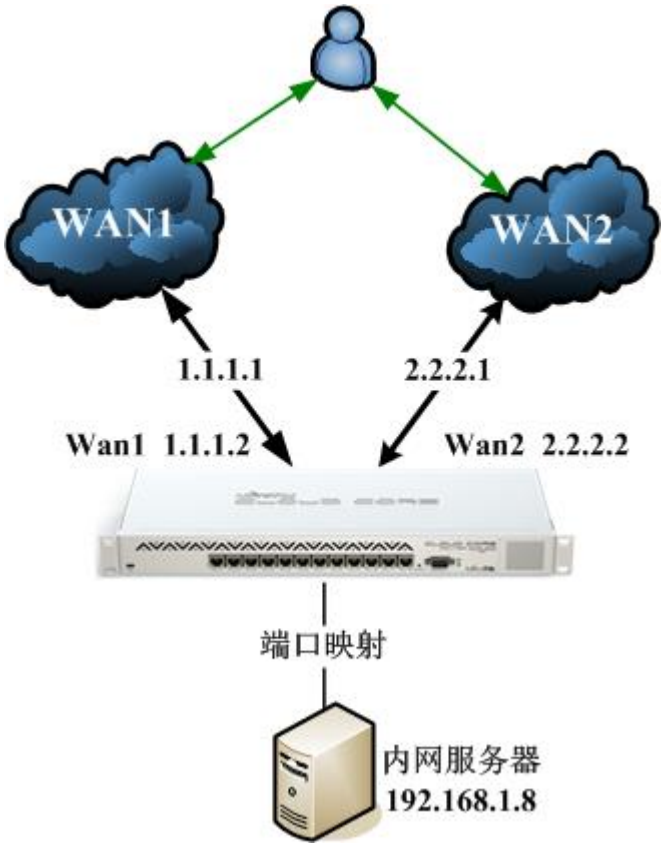
但如果外部网络 B 访问的是 2.2.2.2，那 2.2.2.2 也会回复外部请求，正常的情况下应该是走 2.2.2.1 回复，但路由器自身的默认路由是 1.1.1.1，返回到外部 B 不会走 2.2.2.1 网关回去，而是走的默认网关 1.1.1.1，而我们路由器都是做了 nat 的转换，即 B 请求 2.2.2.2，回复的却是 1.1.1.2，B 网络就认为是非法数据，数据包被丢弃掉。

因此在配置了 input 和 output 规则后，所有访问外网接口的 IP 就按照访问的原路有返回

返程路由的多线路端口映射

这里我们有一个延伸出来的应用，如端口映射，在多线路下要实现内部服务器端口映射同时被各个出口线路访问，也受到返程路由的影响，即当有电信和联通出口时，我们可以做两条端口映射，一条电信映射一条联通映射。

因此我们把 PCC 的返程路由配置引入多线路的端口映射应用中，



配置中同样是在 mangle 中做以下标记

```
/ 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

再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
```

剩下的就是配置端口映射到内网服务器上

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=80 protocol=tcp action=dst-nat
to-addresses=192.168.1.8   to-ports=80

add chain=dstnat dst-address=2.2.2.2 dst-port=80 protocol=tcp action=dst-nat
to-addresses=192.168.1.8 to-ports=80
```

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 服务器询问一下 option 代码：

- option 1 - SUBNET_MASK
- option 3 - GATEWAY_LIST
- option 6 - TAG_DNS_LIST
- option 33 - STATIC_ROUTE
- option 42 - NTP_LIST
- option 121 - CLASSLESS_ROUTE

对于有特殊 option 需求的环境，请参考以上请求代码

属性描述

add-default-route (yes | no; 默认: **yes**) – 是否自动添加指定的 DHCP 服务器的默认路由
client-id (*文本*) – 与 administraor 或 ISP 相符合的参数
enabled (yes | no; 默认: **no**) – 是否启用 DHCP 客户端
host-name (*文本*) – 客户端的主机名
interface (*名称*; 默认: **(unknown)**) – 任何以太网 interface (这包括 wireless 和 EoIP 隧道)
use-peer-dns (yes | no; 默认: **yes**) – 是否使用对端 DHCP 服务器的 DNS 的设置(将会添加到/ip dns 中)
default-route-distance (integer:0..255; 默认:) – 自动添加默认路由的路由距离，这个功能要求 add-default-route（添加默认路由）设置为 yes 才会生效
status (*bound* | *error* | *rebinding...* | *requesting...* | *searching...* | *stopped*) – 显示 DHCP-Client 的状态

命令描述

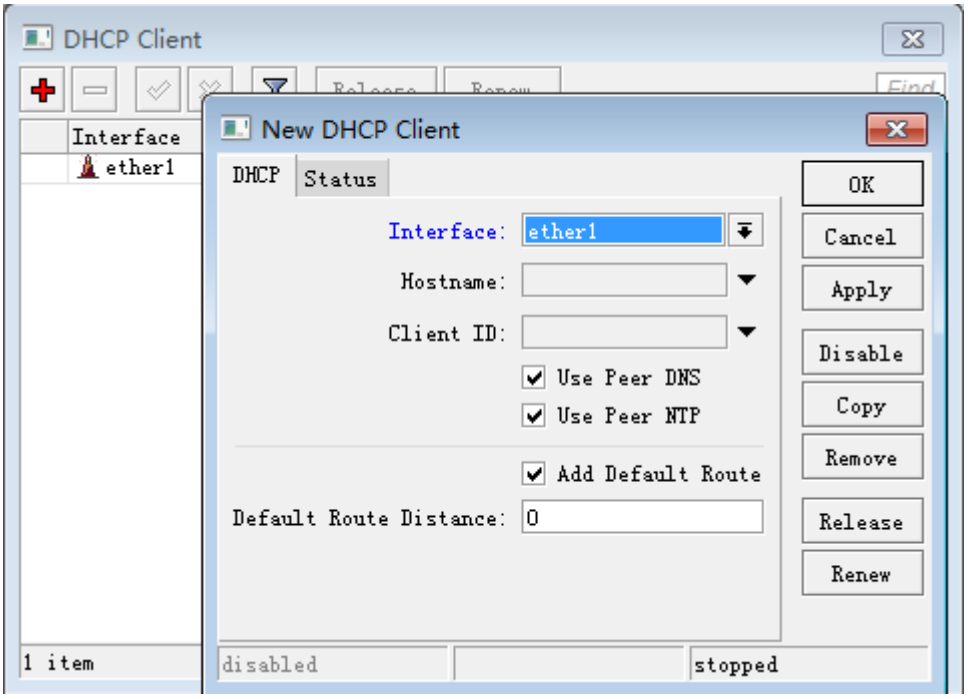
renew（id） – 更新当前的租约，如果更新操作没有成功，客户端将试着初始化租约。
release (id) – 释放当前 DHCP 绑定，并重启 DHCP 客户端

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：



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/掩码*; 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

事例：配置 DHCP 服务器在 **ether1** interface 上，并分配给 10.0.0.2 到 10.0.0.254 的网络地址段，设置网关为 **10.0.0.1**，DNS 服务器为 **159.148.60.2**，租约时间为 3 天：

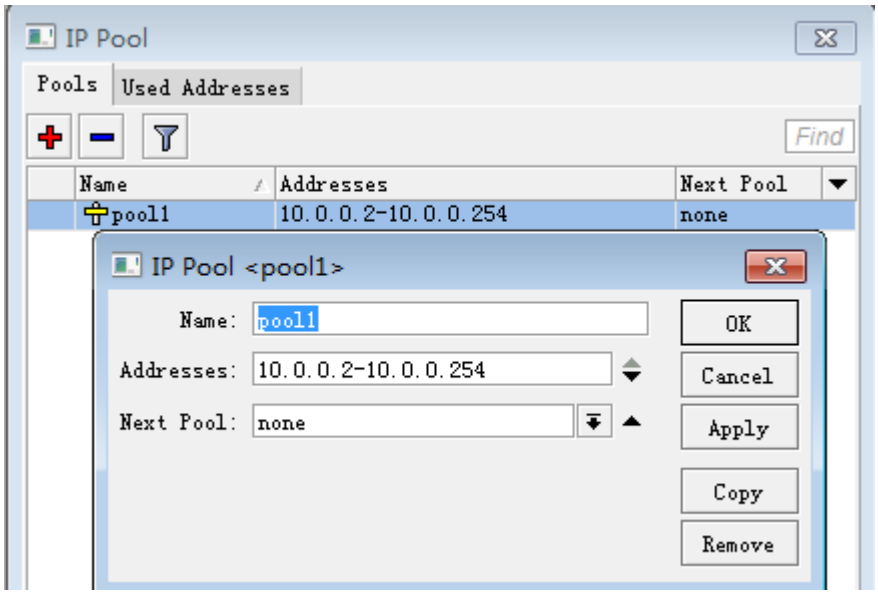
```
[admin@MikroTik] ip dhcp-server> setup
```

```
选择 DHCP 服务器运行的网络接口

dhcp server interface: ether1
选择 DHCP 网络地址段

dhcp address space: 10.0.0.0/24
设置网关地址

gateway for dhcp network: 10.0.0.1
选择 IP 地址池给 DHCP 服务器

addresses to give out: 10.0.0.2-10.0.0.254
设置 DNS 服务器

dns servers: 159.148.60.2
设置租约时间

lease time: 3d
[admin@MikroTik] ip dhcp-server>
```

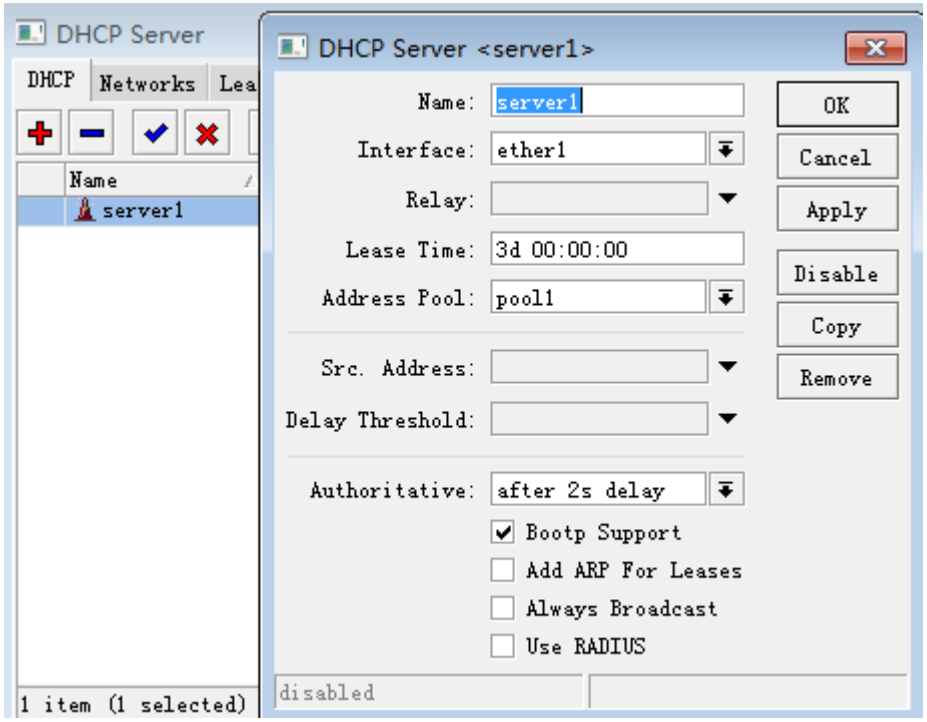
上面向导中设置的内容，通过命令查看如下：

```
[admin@MikroTik] ip dhcp-server> print
Flags: X - disabled, I - invalid
 #   NAME           INTERFACE RELAY          ADDRESS-POOL LEASE-TIME ADD-ARP
 0   dhcp1          ether1    0.0.0.0        dhcp_pool1   3d         no
[admin@MikroTik] ip dhcp-server> network print
 # ADDRESS           GATEWAY        DNS-SERVER     WINS-SERVER     DOMAIN
 0 10.0.0.0/24       10.0.0.1       159.148.60.2
[admin@MikroTik] ip dhcp-server> /ip pool print
 # NAME                                  RANGES
 0 dhcp_pool1                            10.0.0.2-10.0.0.254
[admin@MikroTik] ip dhcp-server>
```

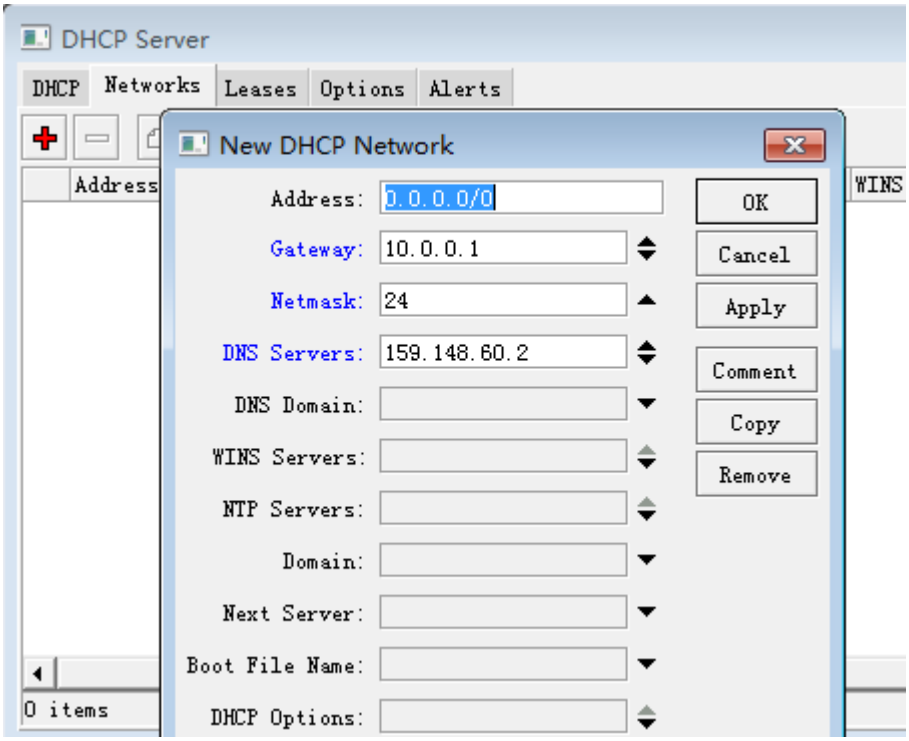
Winbox 操作：添加 DHCP 服务，首先进入/ip pool 中分配地址池范围，注意添加时要排除网关地址：



进入/ip dhcp-server 添加 DHCP 服务接口到 ether1 和对应的地址池



在/ip dhcp-server network 中添加分配的网关和 DNS 参数：



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：

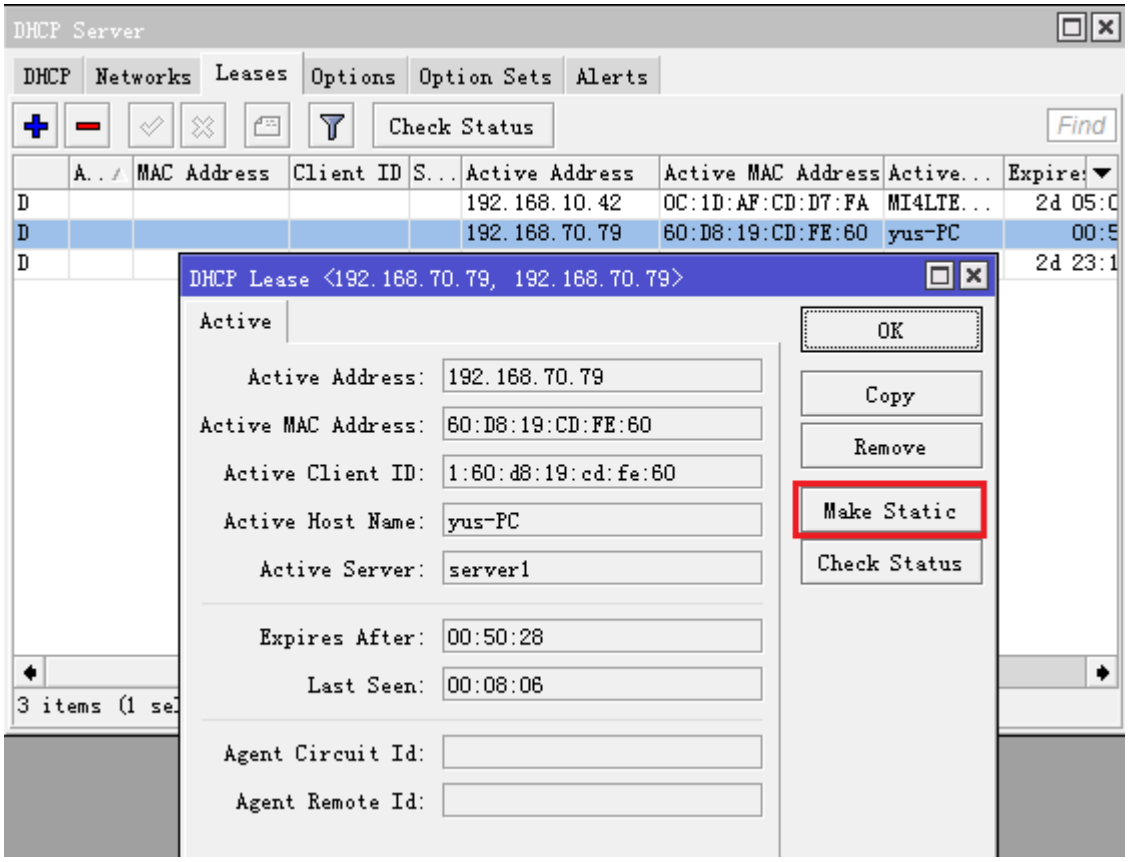
```
[admin@MikroTik] /ip dhcp-server lease> print
Flags: X - disabled, R - radius, D - dynamic, B - blocked
 #   ADDRESS                 MAC-ADDRESS       HOST-NAME    SERVER    RATE-LIMIT    STATUS
 0 D 192.168.10.42           0C:1D:AF:CD:D7:FA MI4LTE-bu... server2                 bound
 1 D 192.168.70.79           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

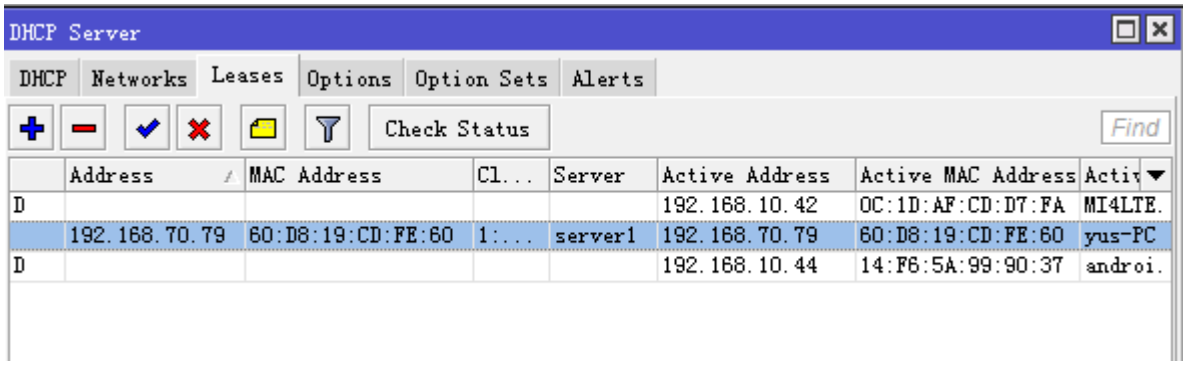
下面通过 winbox 操作来演示如何通过 DHCP Lease 将一个客户端 MAC 和 IP 做静态绑定，假设我们有一台主机 192.168.70.79 和 MAC 地址为 60:D8:19:CD:FE:60，要求做静态绑定

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

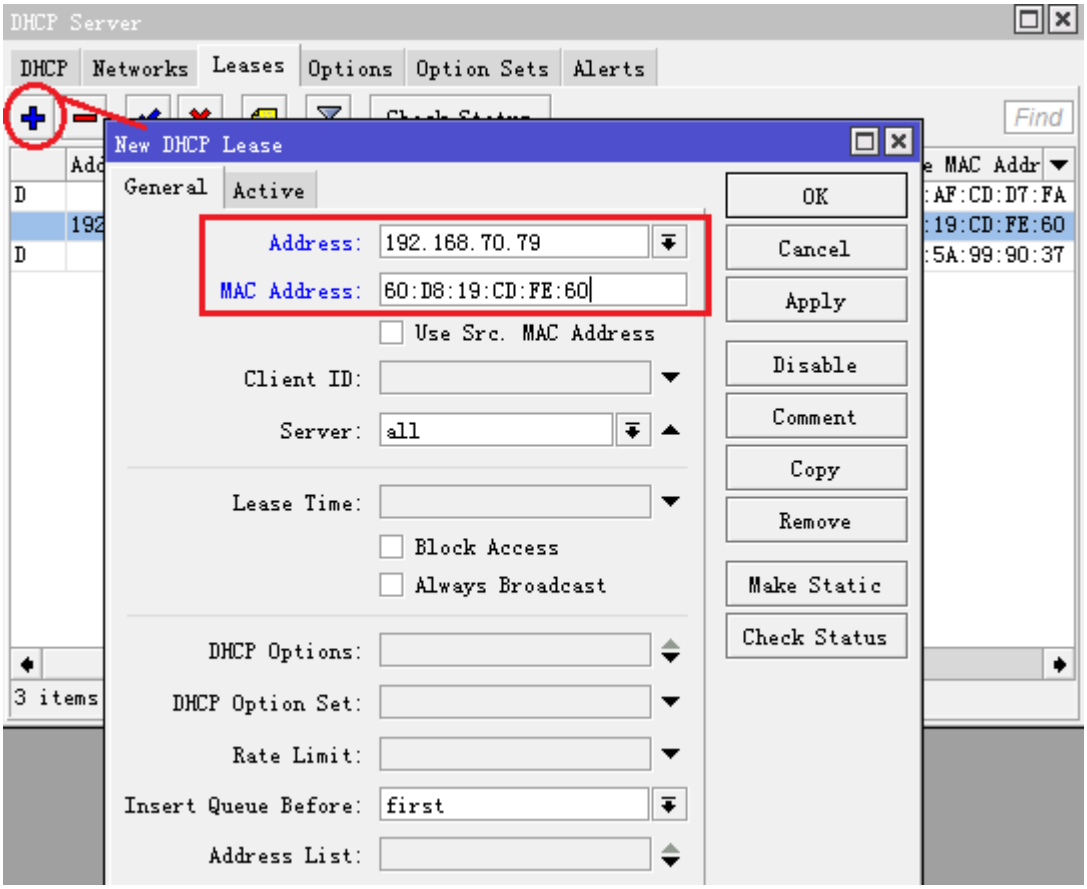


配置后可以看到租约静态绑定的效果：



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**	**61**	**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 回应信息，“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 服务器回应信息是否有效，如果一个回应来至未知服务器，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>
```

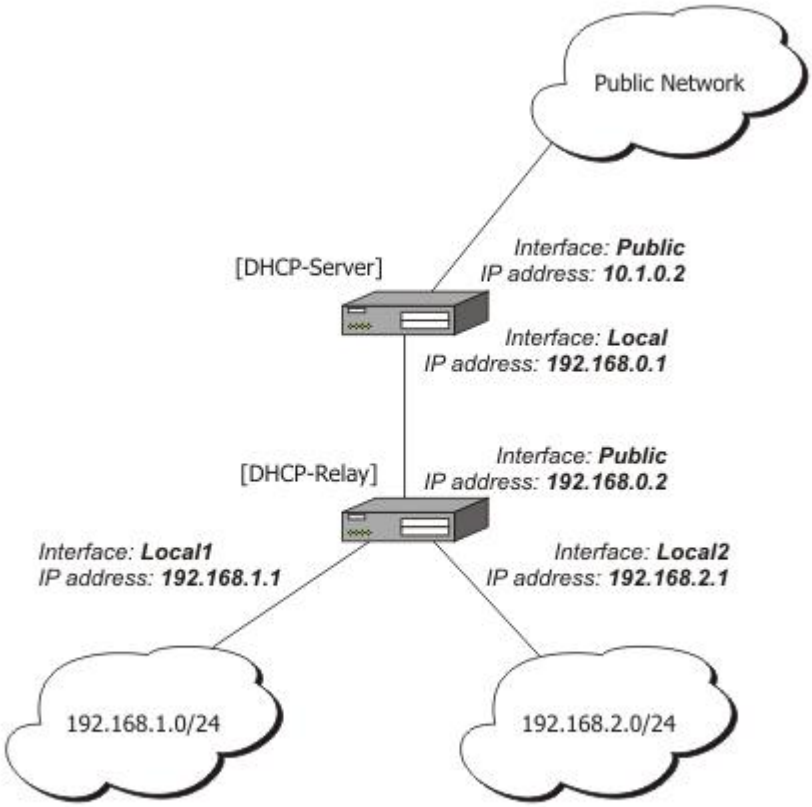
当系统发现流氓 DHCP 服务器发出报警，可以通过执行自定义脚本完成相应操作，可以设置 on-alert 设置脚本发送邮件提醒管理员等

DHCP 回应单播方式，“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。

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>
```

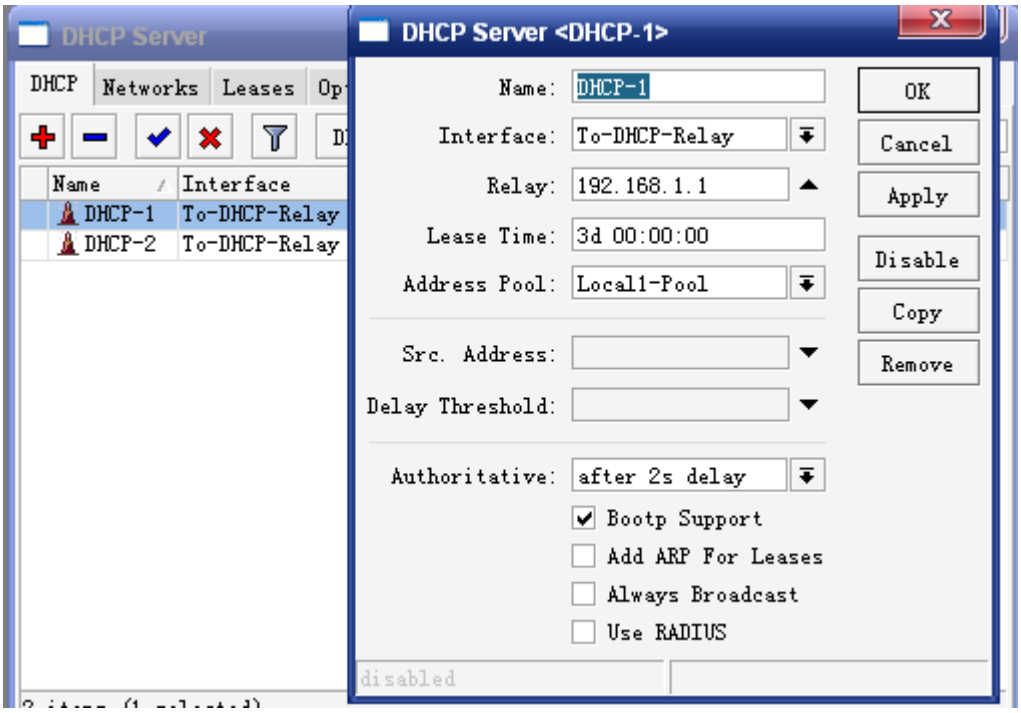
设置两个 DHCP 服务器在 **DHCP-Server** 的路由器上，这里分别添加两个 IP 地址段 **192.168.1.0/24** 和 **192.168.2.0/24**，如下：

```
/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>
```

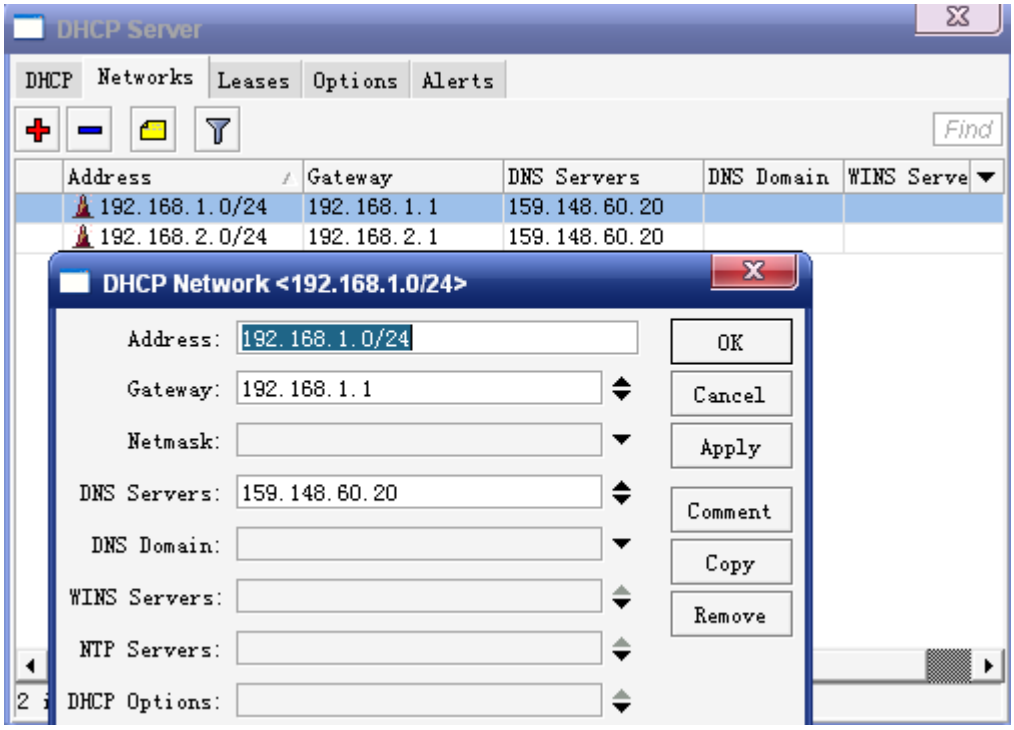
创建 DHCP 服务：

```
/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>
```

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

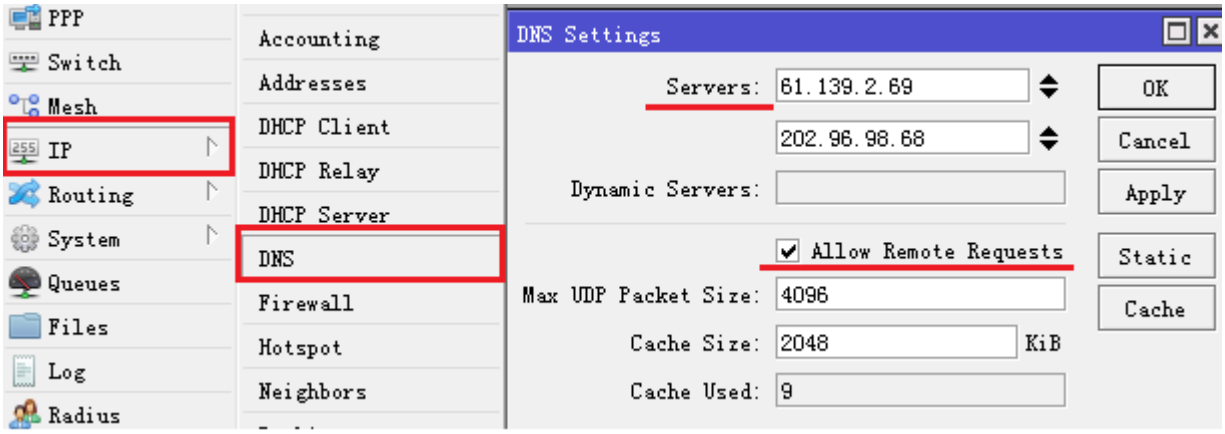
allow-remote-requests (yes | no) – 是否允许远程网络的请求
servers (*IP 地址*; 默认：**0.0.0.0**) – DNS 服务器
cache-size (整型：512..10240; 默认：**2048 kB**) – 指定 DNS 缓存的长度单位为 KB
cache-max-ttl (时间; 默认：**7d**) – 指定缓存记录的最大存活周期
cache-used (只读*:整型*) – 显示当前使用的缓存大小 KB

注：如果/ip dhcp-client 和 PPPOE-client 属性下的 use-peer-dns 设置为 yes，这时/ip dns 下的 servers 将会改变，并修改 DHCP 服务的 DNS 设置。

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：



注：这里的 allow remote requests 为启用 DNS 缓存功能，cashe size 设定缓存存储大小

缓存状态

DNS 缓存是使用最小的 DNS 请求时间连接到外部的 DNS 服务器，这相当于一个简单的本地 DNS 服务。

操作路径: ***/ip dns cache***
name (只读*: 名称*) – 主机的 DNS 名称
address (只读*: IP 地址*) – 主机 IP 地址
ttl (时间) – 剩余的存活周期

可以通过 flush Cache 命令清空当前缓存

8.2 内部 DNS 域名解析

操作路径: ***/ip dns static***

MikroTik RouterOS 在 DNS 缓存中嵌入了 DNS 服务器的一些功能，如通过使用路由器的 DNS，指定解析域名的 IP 地址，即指定外部或内部域名在本地解析。

属性描述

name (文本) – 设置对应的外部和内部的域名。
address (*IP 地址*) – 分配指定域名的 IP 地址

事例：为 **www.example.com** 域名添加静态 DNS，IP 地址是 **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
 # NAME                                          ADDRESS          TTL
 0 aaa.aaa.a                                     123.123.123.123 1d
 1 www.example.com                                 10.0.0.1         1d
[admin@MikroTik] ip dns static>
```

刷新 DNS 缓存

操作指令: ***/ip dns cache flush***

flush – 清除内部 DNS 的缓存 clears internal DNS cache

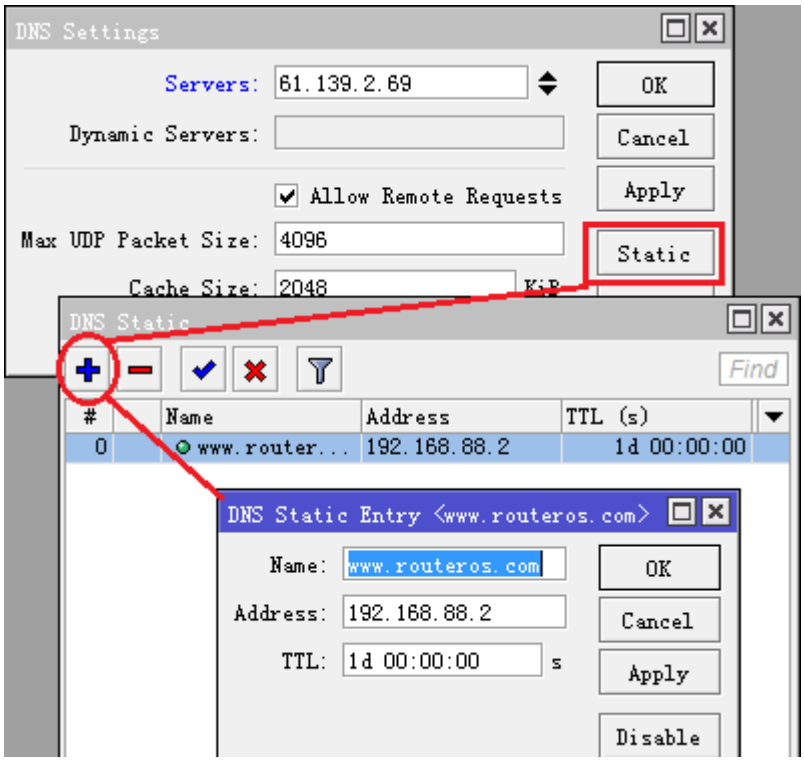
```
[admin@MikroTik] ip dns> cache flush
[admin@MikroTik] ip dns> print
              servers: 61.139.2.69,218.6.200.139
  allow-remote-requests: yes
    max-udp-packet-size: 512
             cache-size: 2048KiB
          cache-max-ttl: 1w
             cache-used: 4KiB
[admin@MikroTik] ip dns>
```

8.3 DNS 劫持

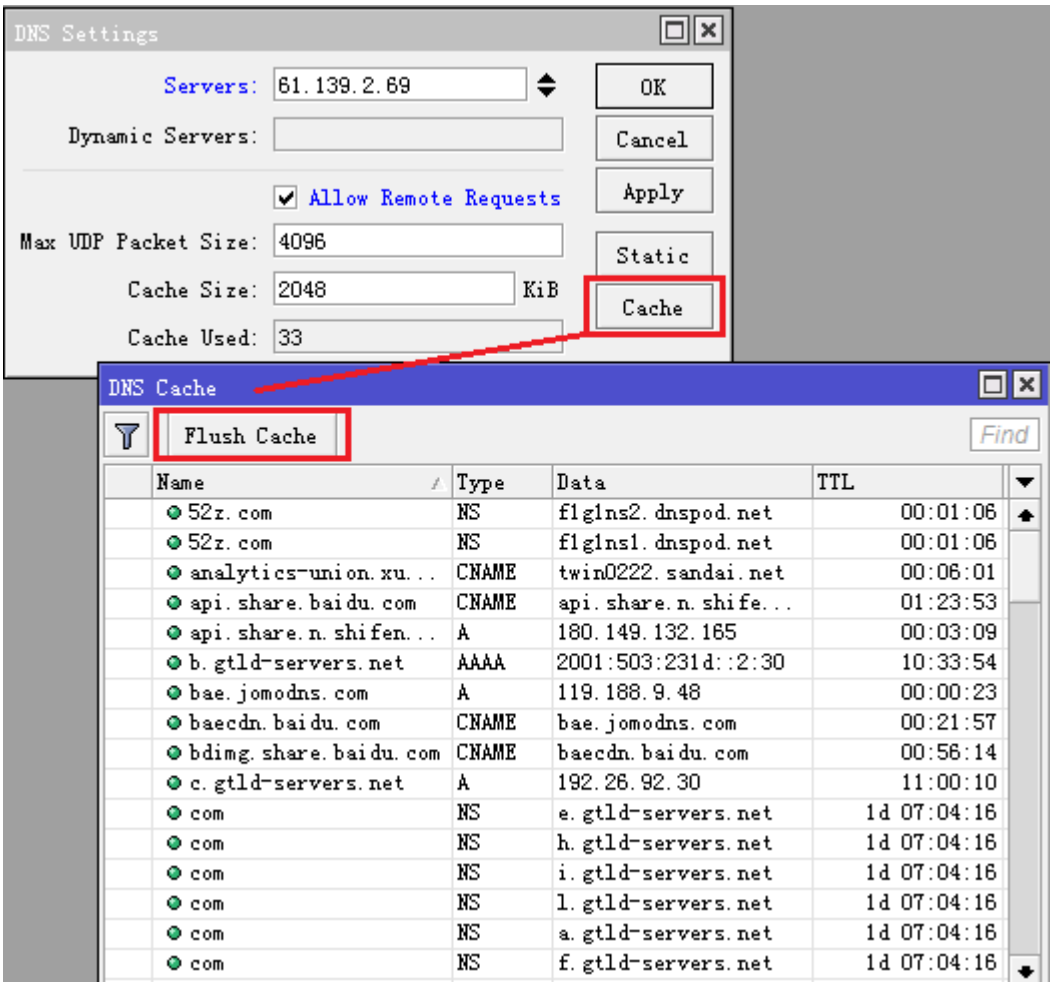
DNS 劫持是当用户请求某一网站时，将网站的 DNS 解析强制到指定的主机，即等同于 A 记录。注意：要实现 RouterOS 路由网关的域名 A 记录查询，必须满足：

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：



第九章 防火墙过滤（Firewall Filte）

在 RouterOS 通过 ip firewall filter 能对 IP 数据包、P2P 协议、源和目标 IP、端口、IP 协议、协议（ICMP、TCP、MSS 等）、网络接口、对内部的数据包和连接作标记、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，RouterO 的防火墙规则从数据包来源方向上分类：分为 input、foreward 和 output 三种链表（chain）过滤，不管是二层或者三层过滤上都包含这三个链表。RouterOS 的防火墙包括了对 address-list 和 L7-protocol 等调用。

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 的数据包全部丢弃掉：

```
/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 过滤

操作路径: ***/ip firewall filter***

网络防火墙始终保持对那些有威胁敏感的数据进入内部网络中，无论怎样网络都是连接在一起的，总是会有某些从外闯入你的 LAN，窃取资料和破坏内部网络，同时也根据网络管理员的要去配置 ACL，适当的配置防火墙可以有效的保护网络。

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
- 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 的数据包从那里到达或通过那里里去
  - 内部数据流与连接标记
  - ToS (DSCP)

- 数据包内容
- Connection-rate 连接速率
- PCC 分离器
- 数据包大小
- 包到达时间

基本过滤规则

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 数据包，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，需要在三条默认的链中对 **action=jump** 策略中相关的 **jump-target** 进行配置。

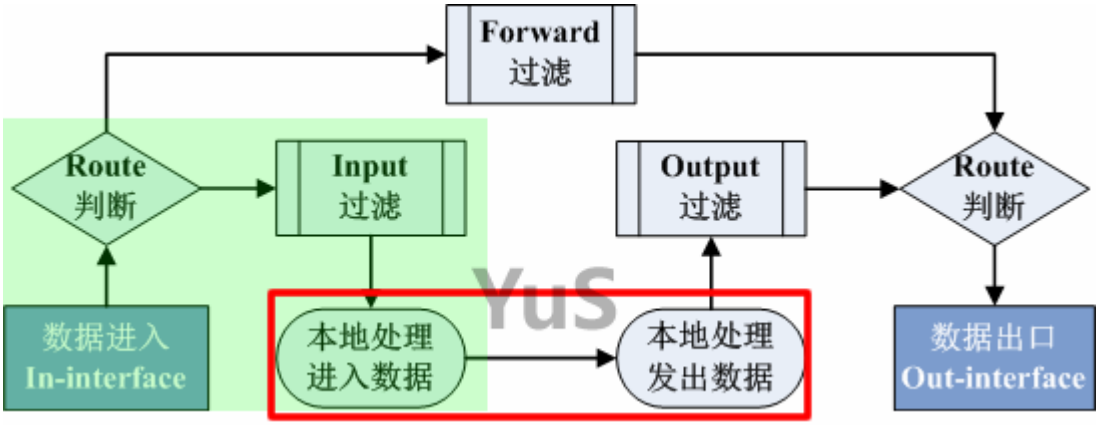
过滤链

下面是三条预先设置好了的 chains，他们是不被能删除的：

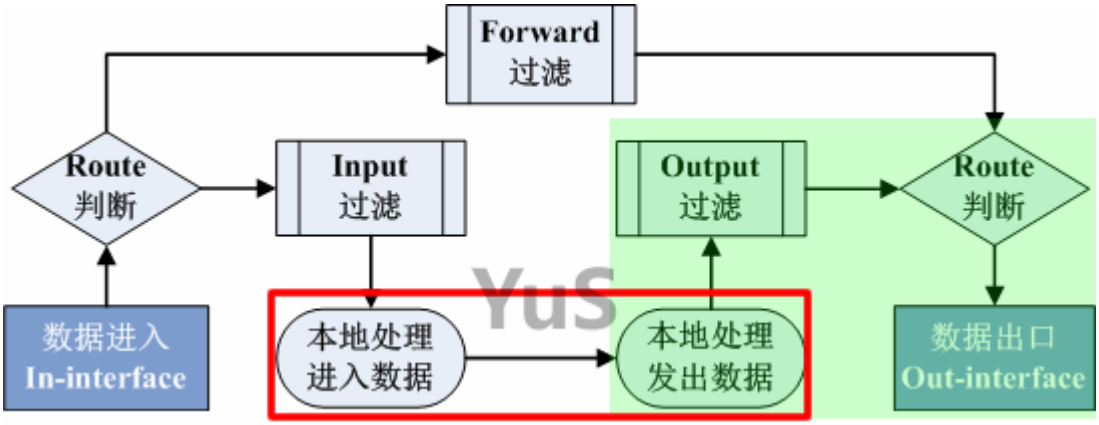
- **input** – 用于处理进入路由器的数据包，即数据包目标 IP 地址是到达路由器一个接口的 IP 地址，经过路由器的数据包不会在 input-chains 处理。
- **forward** – 用于处理通过路由器的数据包
- **output** – 用于处理源于路由器并从其中一个接口出去的数据包。

他们具体的区别如下：

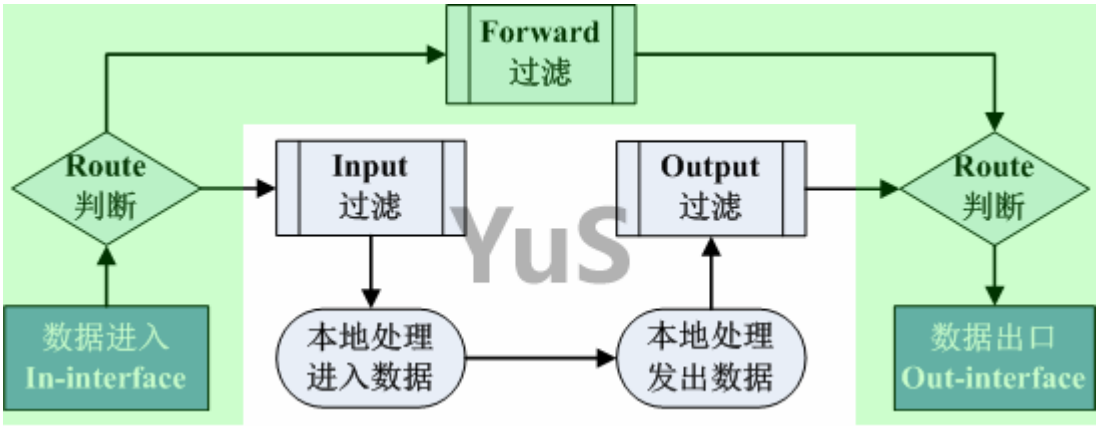
IP 数据包进入 input 链表的数据工作流程，阴影部分代表经过的处理部件：



IP 数据包进入 output 链表的流程，阴影部分代表经过的处理部件：



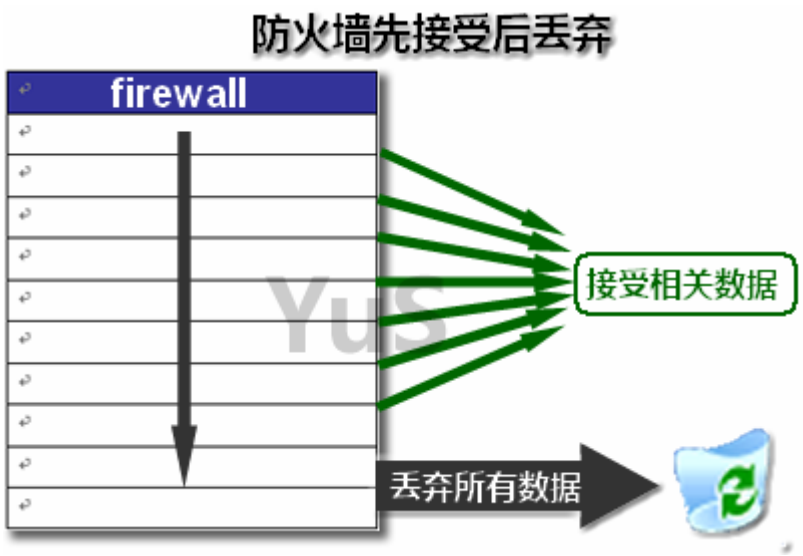
IP 数据进入 forward 链表的流程，阴影部分代表经过的处理部件

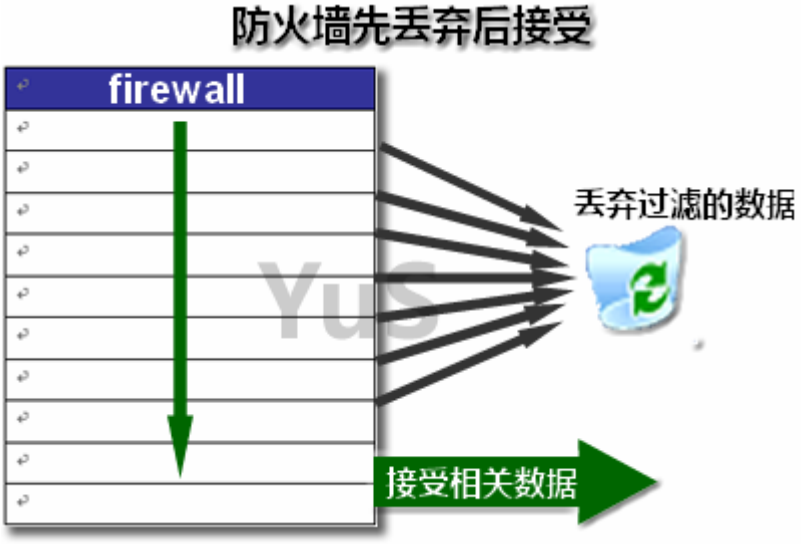


当处理一条 chain（数据链），策略是从 chain 列表的顶部从上而下执行的。如果一个数据包满足策略的条件，这时会执行该操作。

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：

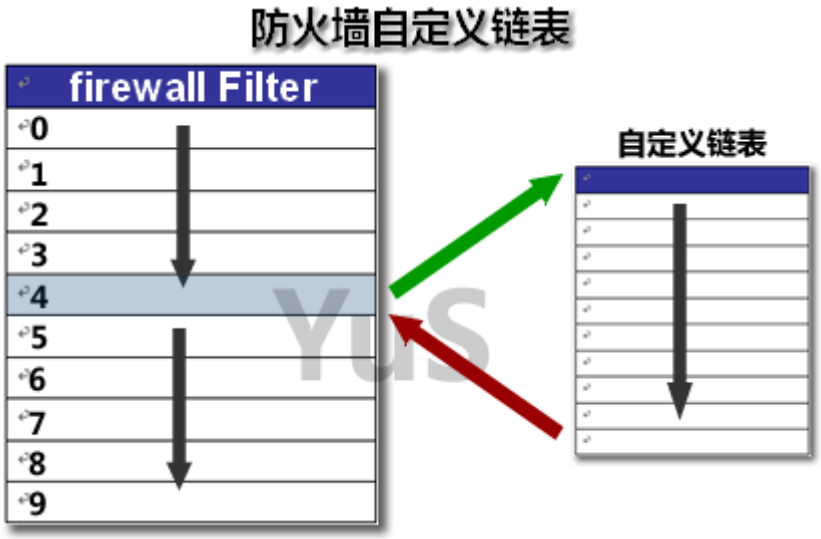




以上两种方式通过图可以很好的理解，在整个 filter 中，当规则为空时，默认是接受所有数据通过，我们通常情况下最多的方式是加入拒绝那些数据通过，即先丢弃后接受。

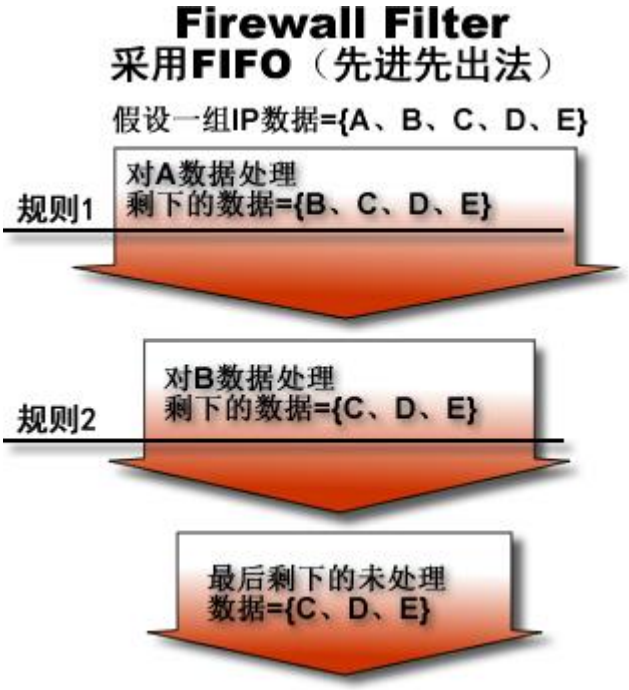
自定义链表

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。



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），这个和早期的 simple Queue 一样（不过 v6.0 后 simple Queue 不再遵循 FIFO 原则），理解如下图



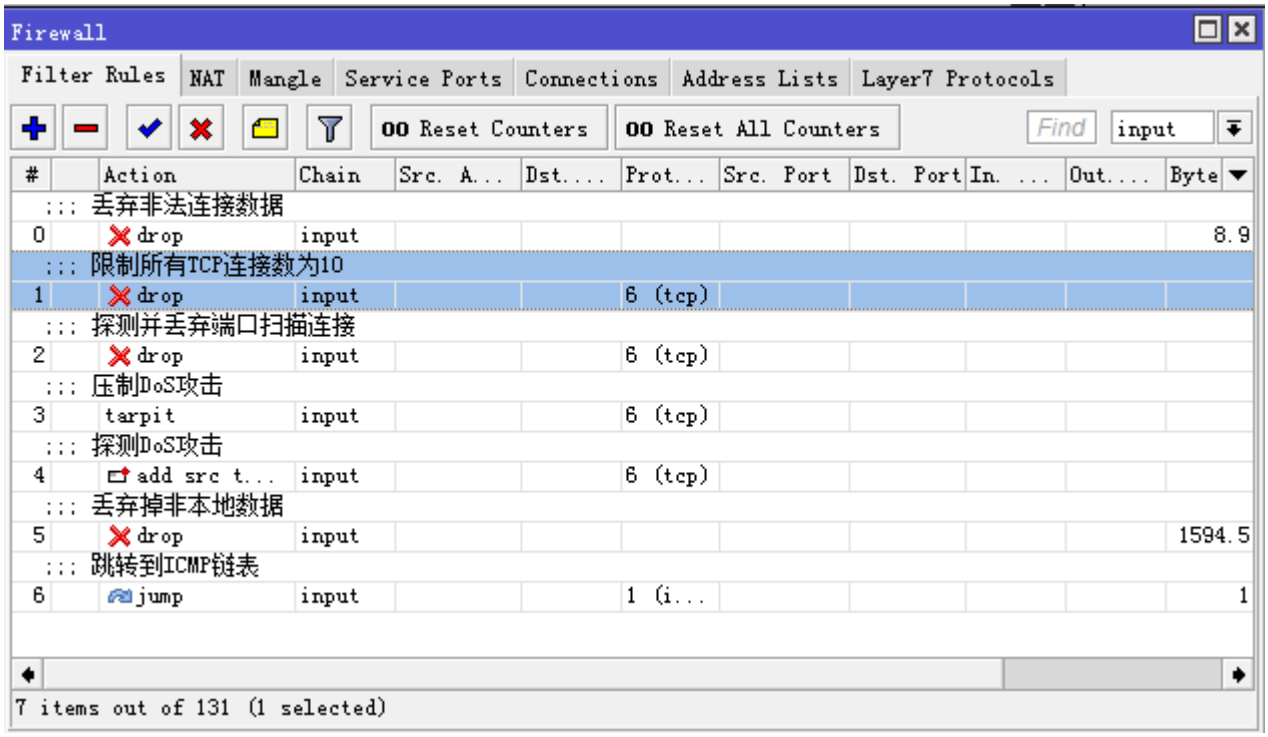
假设我们有 A、B、C、D、E 等 5 个数据，当第一条规则处理了 A 数据，继续向下传递时，第二条规则中不会在出现 A 数据，第二条规则处理 B 数据，当然在后面的规则也不会出现 B 数据。整个流程先处理先通过，规则越靠前，就越优先处理，即从上往下的一个执行过程。

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。

9.2 防火墙 filter 事例

Input 事例

我先从 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，下面是一个对路由器保护的策略配置



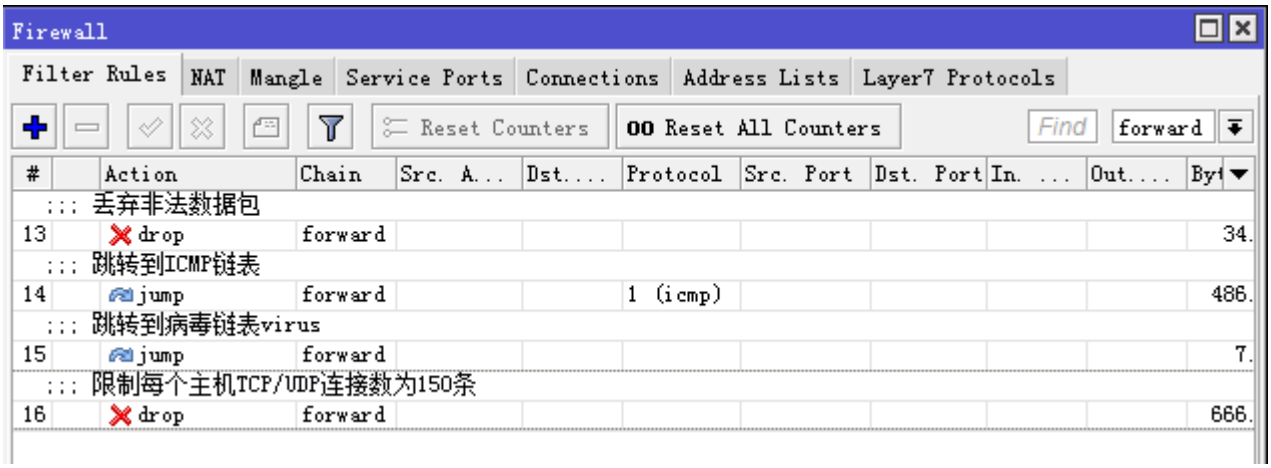
从 input 链表的第一条开始执行，这里一共有 7 条规则，配置命令如下：

```
[admin@Mikrotik] /ip firewall filter> print
Flags: X - disabled, I - invalid, D - dynamic
 0    ;;;丢弃非法连接数据
      chain=input action=drop connection-state=invalid
 1    ;;;限制所有 TCP 连接数为 10
      chain=input action=drop protocol=tcp connection-limit=20,0
 2    ;;;探测并丢弃端口扫描连接
      chain=input action=drop protocol=tcp psd=21,3s,3,1
 3    ;;;压制 DoS 攻击
      chain=input action=tarpit protocol=tcp src-address-list=black_list connection-limit=3,32
 4    ;;;探测 DoS 攻击
      chain=input action=add-src-to-address-list protocol=tcp address-list=black_list
address-list-timeout=1d connection-limit=10,32
 5    ;;;丢弃非本地数据
      chain=input action=drop dst-address-type=!local
 6    ;;;跳转到 ICMP 链表
      chain=input action=jump jump-target=ICMP protocol=icmp
```

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。

Forward 事例

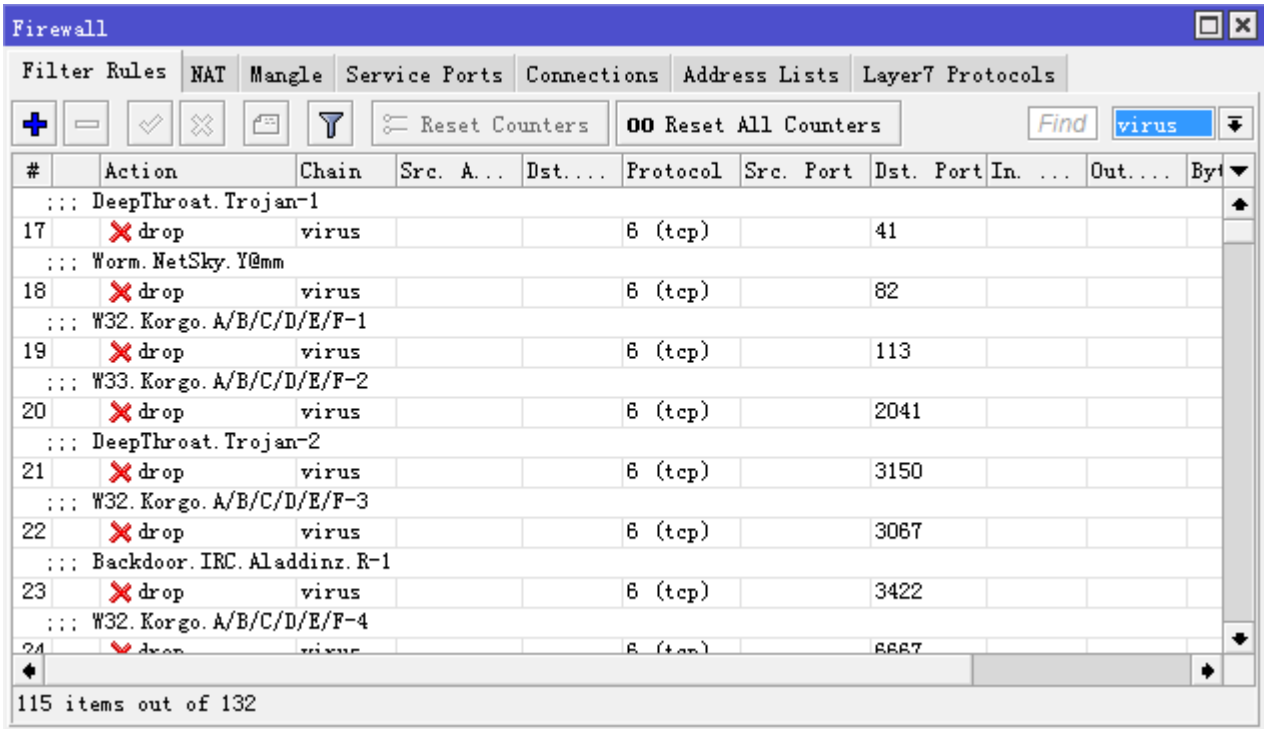
Input 是对进入路由器方向数据处理，即 input 的作用更多的是在为保护路由器做配置，而 forward 链表，则是在对由外向内或由内向外的一种过滤方式



forward 链表，一共有 4 条规则，包括两个跳转到自定义链表 ICMP 和 virus 链表：

```
13   ;;;丢弃非法数据包
     chain=forward action=drop connection-state=invalid
14   ;;;跳转到 ICMP 链表
     chain=forward action=jump jump-target=ICMP protocol=icmp
15   ;;;跳转到病毒链表 virus
     chain=forward action=jump jump-target=virus
16   ;;;限制每个主机 TCP/UDP 连接数为 150 条
     chain=forward action=drop connection-limit=150,32
```

forward 规则仍然包含了丢弃非法数据包和 ICMP，也是我们常见的基本配置，在后面我们增加了一个自定义的病毒过滤链表 virus，在这个表里面包含了常见的或及时发现的一些入侵端口或应用协议



我们列举几个简单的配置实例：

```
17   ;;; DeepThroat.Trojan-1
     chain=virus action=drop protocol=tcp dst-port=41
18   ;;; Worm.NetSky.Y@mm
     chain=virus action=drop protocol=tcp dst-port=82
19   ;;; W32.Korgo.A/B/C/D/E/F-1
     chain=virus action=drop protocol=tcp dst-port=113
20   ;;; W33.Korgo.A/B/C/D/E/F-2
     chain=virus action=drop protocol=tcp dst-port=2041
21   ;;; DeepThroat.Trojan-2
     chain=virus action=drop protocol=tcp dst-port=3150
```

最后一条规则是“限制每个主机 TCP/UDP 连接数为 150 条”，这里我们可以和前面的 input 规则定义的“限制所有 TCP 连接数为 10”，限制连接数，设置的是 connection-limit 这个参数，我们可以对比下这两条规则的配置。

限制所有 TCP 连接数为 10：

```
chain=input action=drop protocol=tcp connection-limit=20,0
```

限制每个主机 TCP/UDP 连接数为 150 条：

```
chain=forward action=drop connection-limit=150,32
```

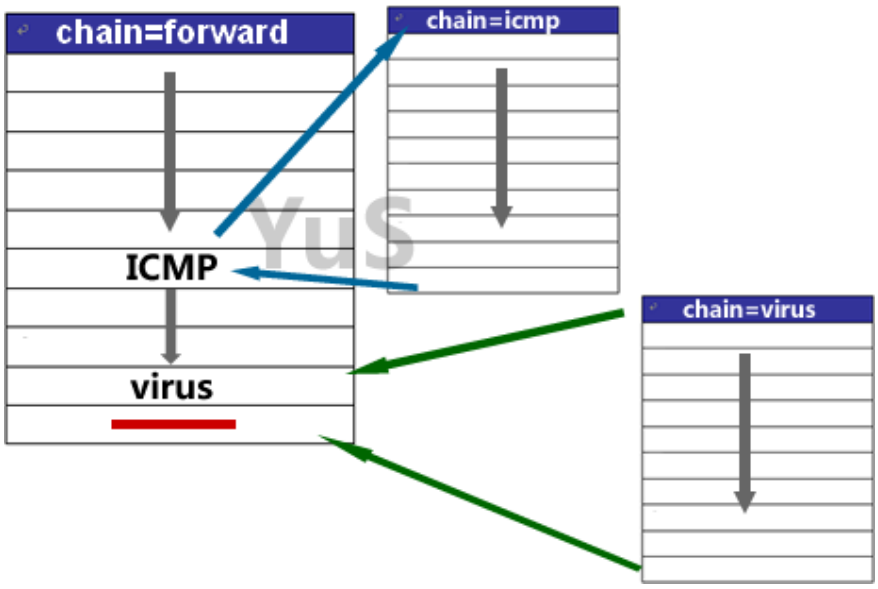
从这两条规则对比出，限制所有主机连接数 connection-limit=20,0，而限制每个主机 connection-limit=150,32，即每个主机用 32 表示，所有主机用 0 表示。

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。

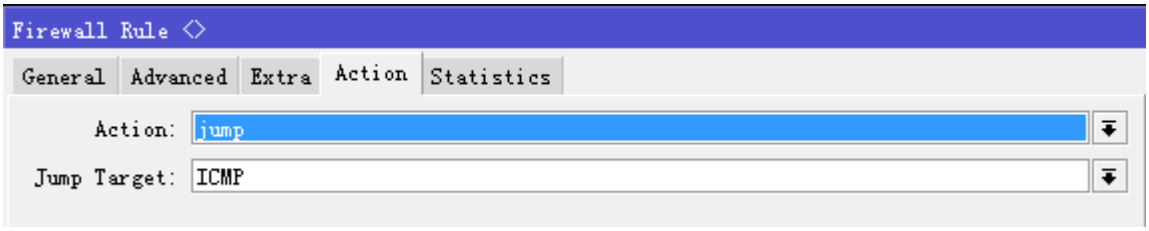
Output 是对由路由器主动发出的数据进行过滤，这类数据需要限制的很少，所以在此就不具体讲解，操作与 input 较类似，读者可以自行实验。

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：

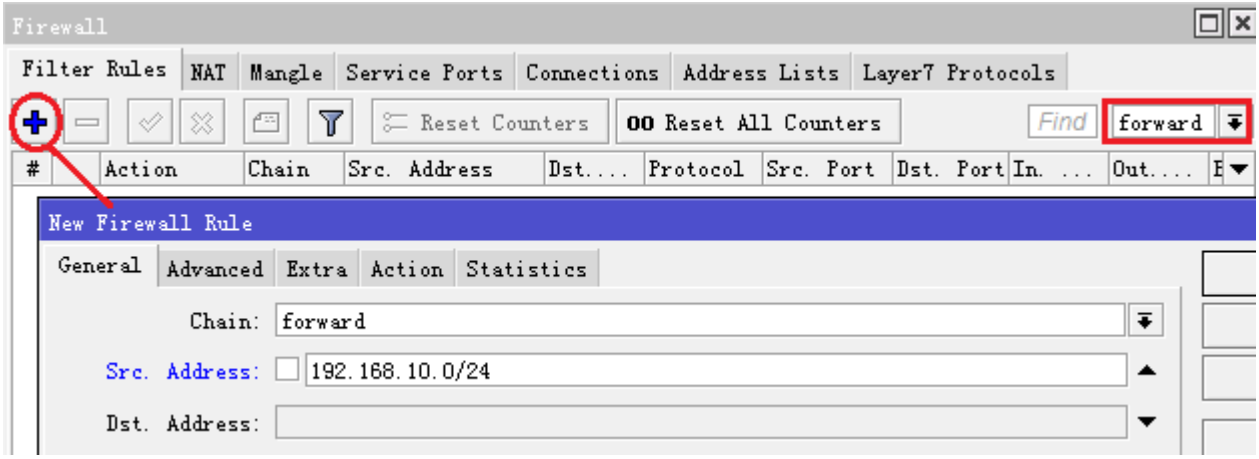


在 winbox 中 jump 的设置

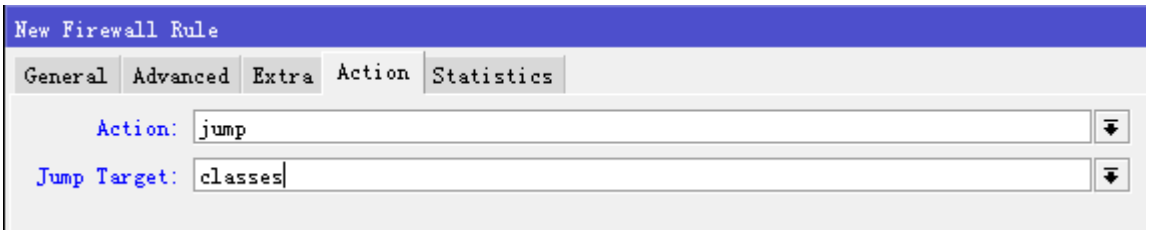


虽然我们可以一次将所有规则在 forward 或 input 等链表中配置，但对于分类管理和查找非常不便，Jump 操作让我们可以将各类过滤规则分类，如我们企业网络，可以将员工 IP 地址段和经理 IP 地址段区分开；在校园网络可以将学生 IP 段和教师 IP 段分开；在 ISP 网络中可以将应用数据进行特点分类等等...

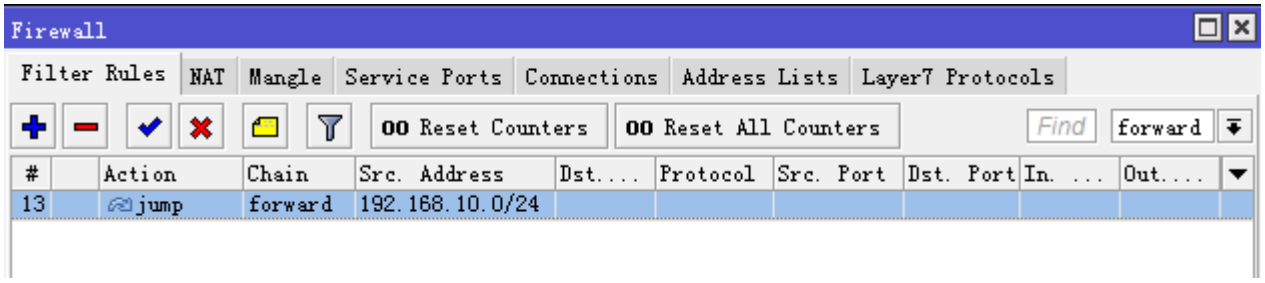
例如我们定义一个 classes 分类的过滤链表，下面是通过 winbox 配置，我们对学生的 ip 地址段 192.168.10.0/24 进行分类跳转：



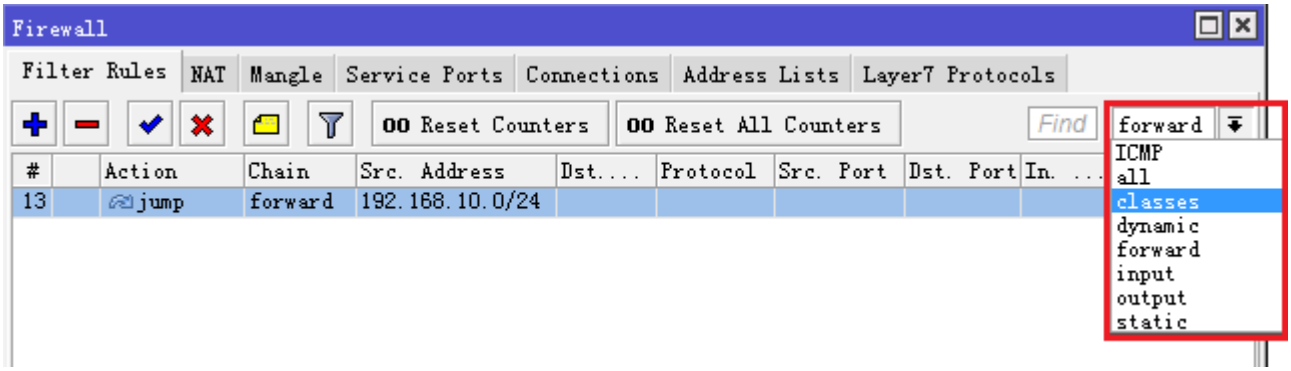
设置 action 为 jump，jump-target 用于指定链表，也可以用于创建一个新链表，这里取名 classes



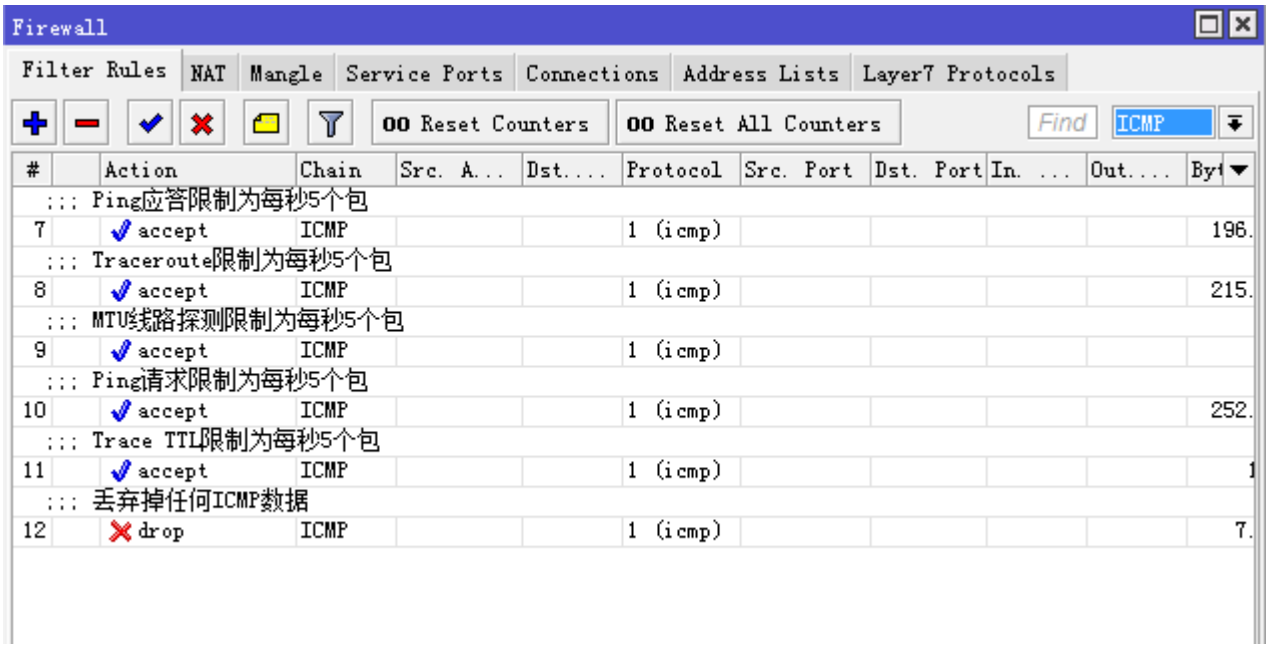
这样我们可以进入 classes 里设置所需的规则，由于 jump 规则已经选择了 192.168.10.0/24 的用户地址，在 classes 链表中，我们无需设置 src-address 的 ip 地址，简化了配置。



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：

```
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
```

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** – 回应答复

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 回应请求向外发送和回应答复进入
- 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

防火墙 action 命令说明

- Accept – 接受数据包，例如接受允许数据通过；
- Add-dst-to-address-list – 根据规则条件，将 IP 数据包的目标地址 IP 添加到指定 address-list；

- Add-src-to-address-list – 根据规则条件， 将 IP 数据包的源地址 IP 添加到指定的 address-list；
- Drop – 丢弃掉数据包（不会发送 ICMP 拒绝信息）；
- 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 – 拒绝数据包，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。

源 IP 地址与目标 IP 地址

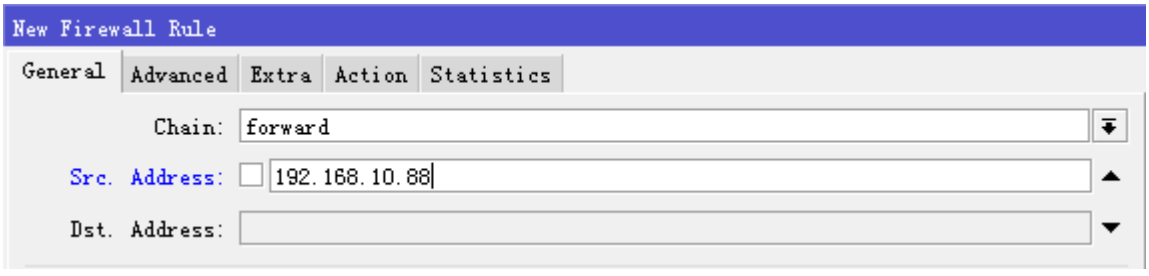
通常我们拒绝一个 IP 访问或者端口访问，仅仅过滤到他源地址/端口或目标地址/端口即可，因为单向通行已经被阻断，但如果我们是先接受后丢弃的方式，就会涉及到如何判断源地址/端口和目标地址/端口，与他们在 ip firewall filter 的链表，我们先看看下面的图：

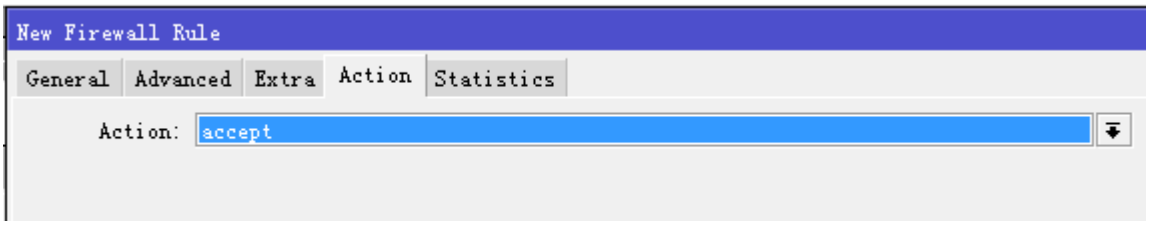
我们从该图上可以看到，内网主机 192.168.10.88 与 web 服务器 218.88.88.88 通信的情况，内网主机 192.168.10.88 向路由器请求连接，不同情况下源目标 IP 地址的转变。

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 服务器收到后会回应主机 192.168.10.88，这时 web 服务器回应发送数据变成了 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。

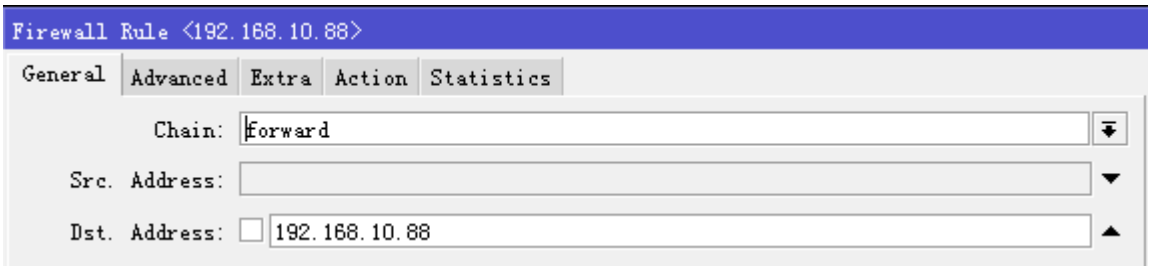
如果我们在配置先接受后丢弃方式时，我们需要允许 192.168.10.88 访问路由器，其他数据都拒绝，按照之前的双向通信的原则，我们需要配置两条 accept 规则，一条 drop 规则。

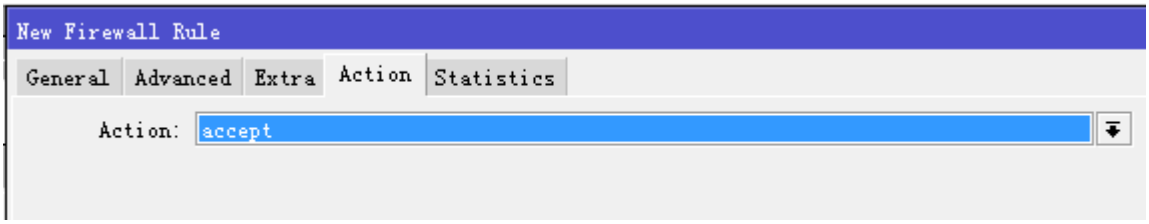
第一条接受 src-address=192.168.10.88



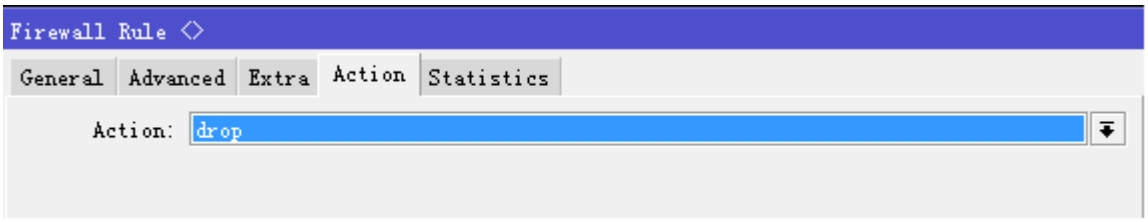


第二条规则，接受 dst-address=192.168.10.88

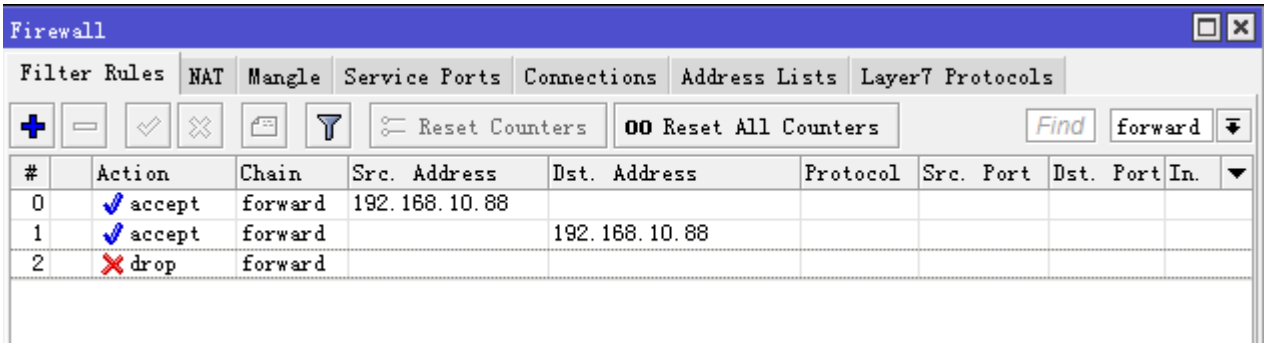




第三条规则，丢弃所有的数据，这里我们直接配置一条 action=drop 规则



规则如下：



脚本配置如下：

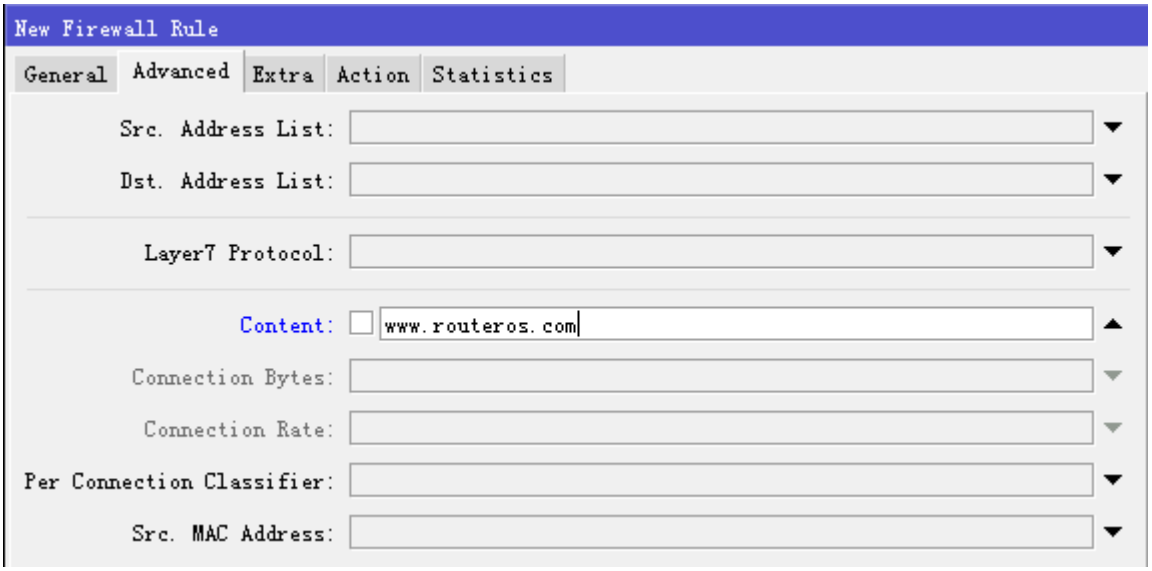
```
/ip firewall filter
add chain=forward src-address=192.168.10.88
```

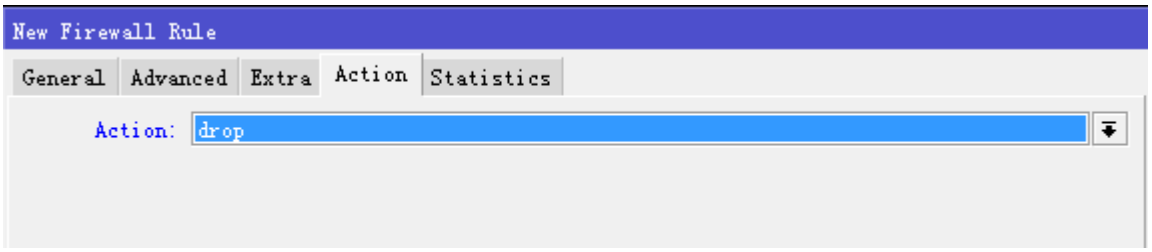
```
add chain=forward dst-address=192.168.10.88
add action=drop chain=forward
```

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字。

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

9.3 Address-list

操作路径: /ip firewall address-list

RouterOS 在防火墙中提供了地址列表功能，允许管理员创建 IP 地址列表组（address-list），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。

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

属性	描述
address (*IP 地址/掩码 | IP-IP*; 默认:)	该属性可以设置一个独立的 IP 地址/掩码或 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：

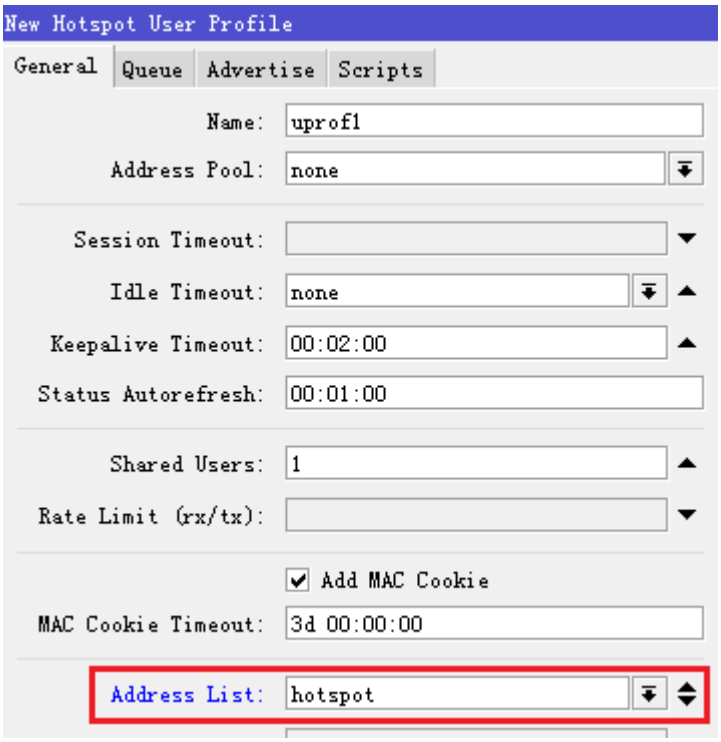
```
/ip firewall address-list add list=drop_traffic address=192.0.34.166/32
/ip firewall address-list print
Flags: X - disabled, D - dynamic
 #    LIST           ADDRESS
 0    drop_traffic 192.0.34.166

/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。

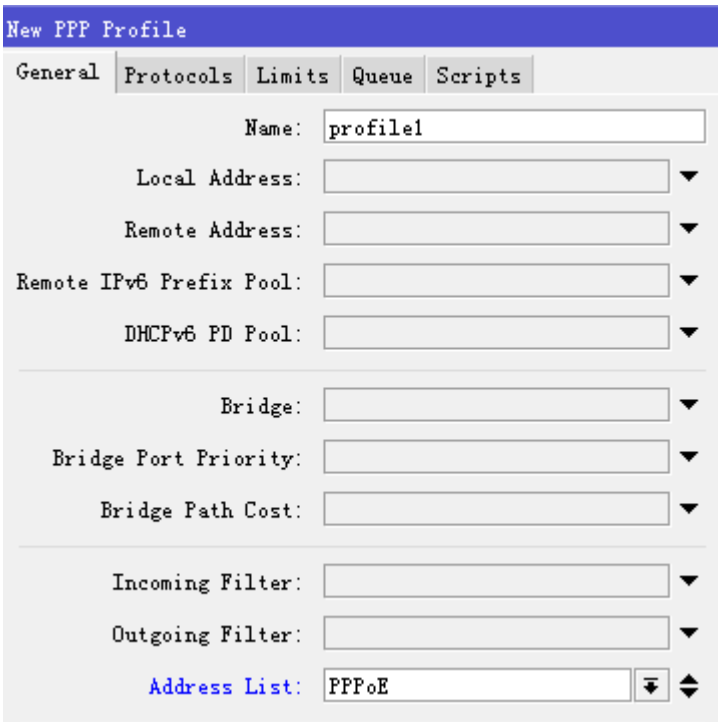
```
/ip firewall address-list print
Flags: X - disabled, D - dynamic
 #    LIST           ADDRESS
 0    drop_traffic 192.0.34.166
 1 D drop_traffic 1.1.1.1
 2 D drop_traffic 10.5.11.8
```

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。

v6.36 允许 address-list 添加域名

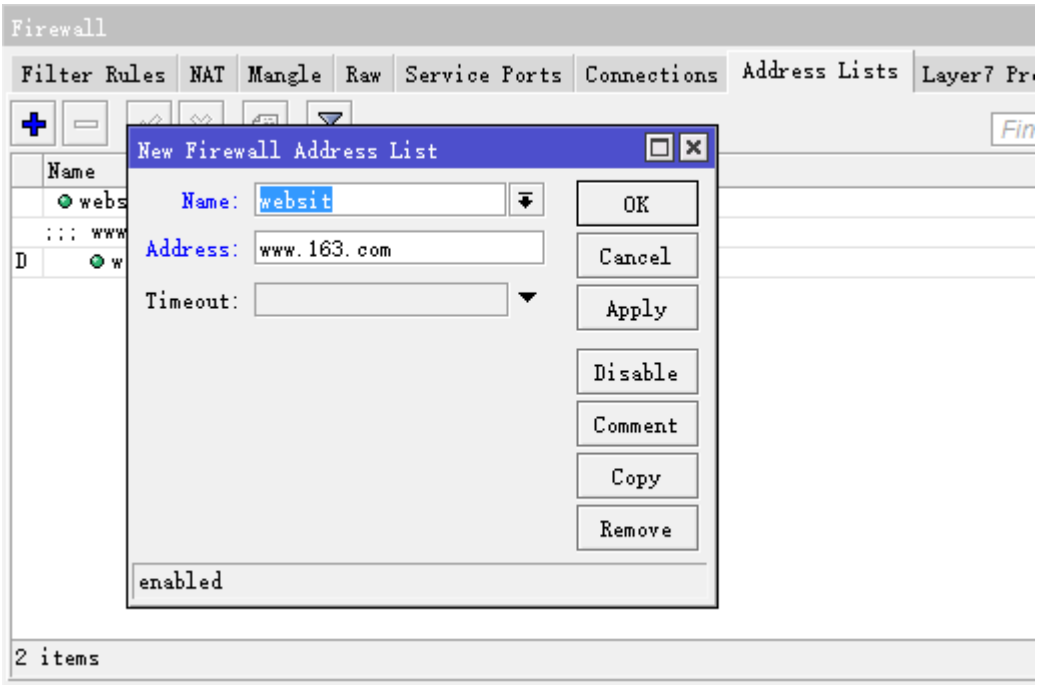
Address-list 从 RouterOS v6.36 后， 允许 address 属性支持域名，即方便我们对域名站点的 IP 控制，下面我们通过一个实例为具体配置情况

下面我们创建一个 websit 的列表，并添加 address=www.mikrotik.com

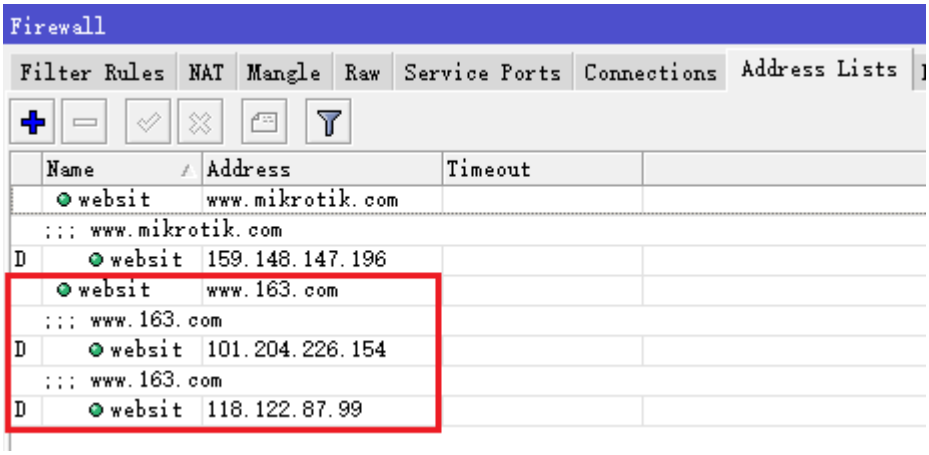
```
[admin@MikroTik] /ip firewall address-list> add list=websit address=www.mikrotik.com
[admin@MikroTik-J] /ip firewall address-list> print
Flags: X - disabled, D - dynamic
 #    LIST                              ADDRESS                                 TIMEOUT
 0    websit                            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

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。

操作路径：/interface list

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

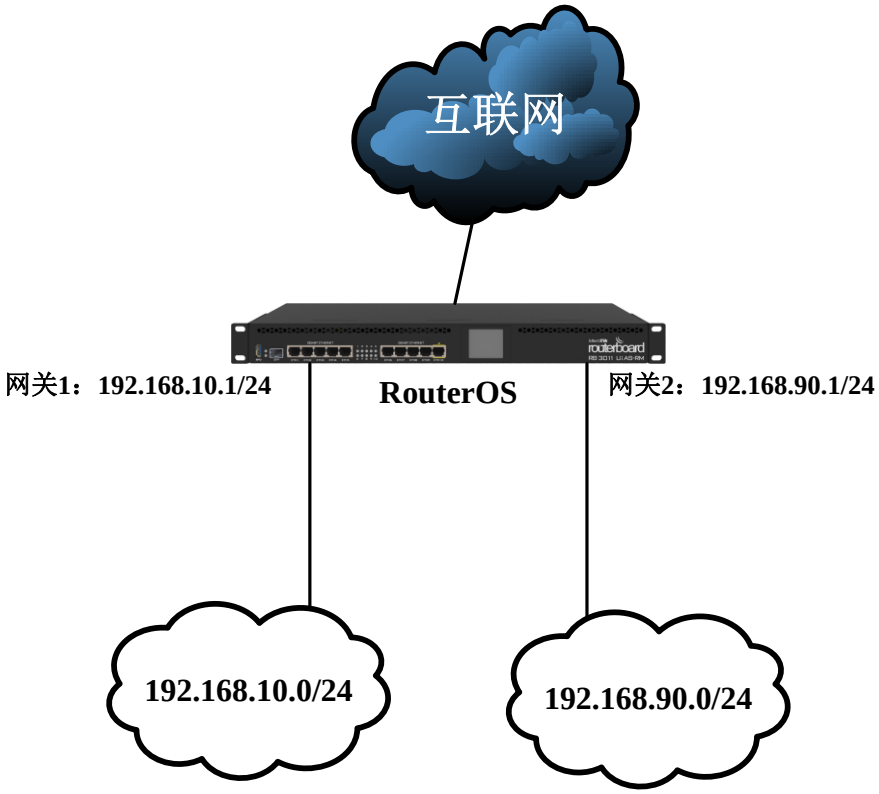
```
[admin@MikroTik-J] /interface list member> print
Flags: X - disabled, D - dynamic
 #    LIST                                INTERFACE
 0    lan                                 ether4
 1    lan                                 ether5
```

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=""
```

9.6 P2P 协议过滤

Peer-to-peer 协议即我们所说的用于主机间点对点传输 P2P。这个技术有许多优秀的应用如 Skype，但 P2P 技术也应用到了下载和视频等应用上，这些应用中网络中影响到 http 和 e-mail 的正常使用。RouterOS 能识别小部分 P2P 协议的连接，并能通过 QOS 进行过滤，丢弃所有的 P2P 协议（**RouterOS 自带的 P2P 识别在实际应用中仅供参考**）：

```
[admin@MikroTik] /ip firewall filter> add chain=forward p2p=all-p2p action=drop
[admin@MikroTik] /ip firewall filter> print chain=forward
Flags: X - disabled, I - invalid, D - dynamic
0    chain=forward action=drop p2p=all-p2p
```

能探测到该协议的列表：

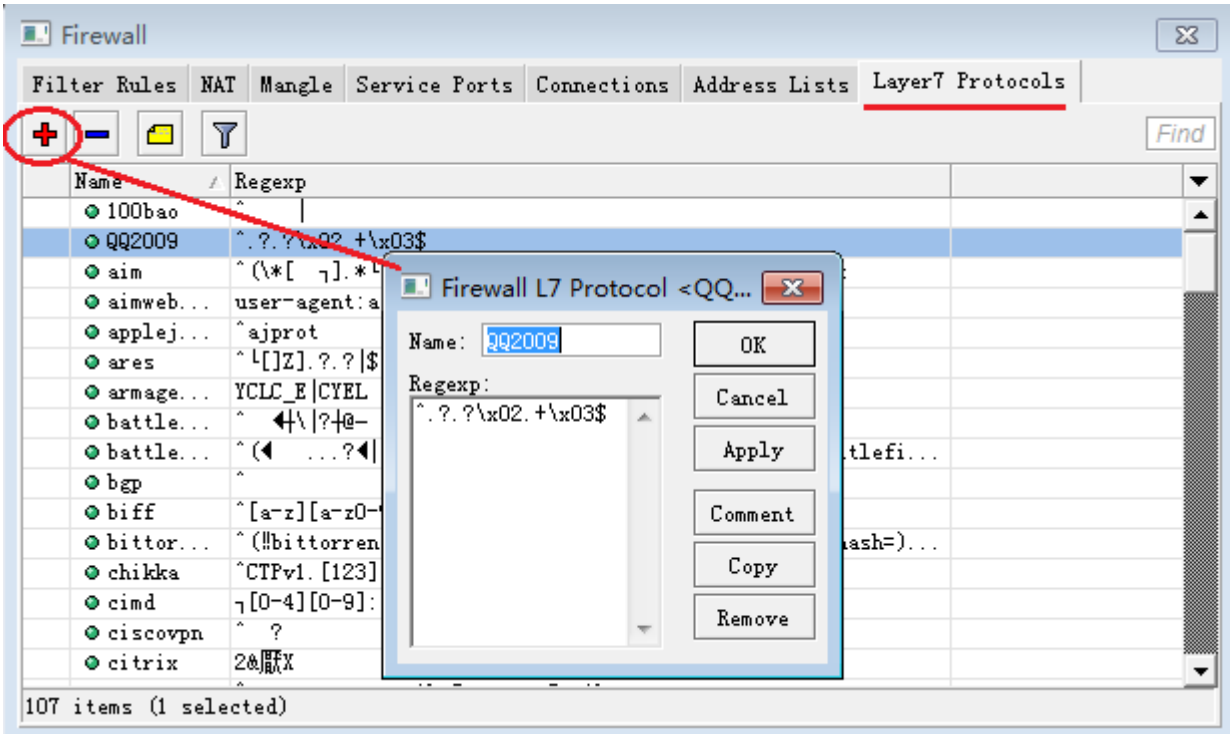
- **Fasttrack** (Kazaa, KazaaLite, Diet Kazaa, Grokster, iMesh, giFT, Poisoned, mlMac)
- **Gnutella** (Shareaza, XoLoX, , Gnucleus, BearShare, LimeWire (java), Morpheus, Phex, Swapper, Gtk-Gnutella (linux), Mutella (linux), Qtella (linux), MLDonkey, Acquisition (Mac OS), Poisoned, Swapper, Shareaza, XoloX, mlMac)
- **Gnutella2** (Shareaza, MLDonkey, Gnucleus, Morpheus, Adagio, mlMac)
- **DirectConnect** (DirectConnect (AKA DC++), MLDonkey, NeoModus Direct Connect, BCDC++, CZDC++)
- **eDonkey** (eDonkey2000, eMule, xMule (linux), Shareaza, MLDonkey, mlMac, Overnet)
- **Soulseek** (Soulseek, MLDonkey)
- **BitTorrent** (BitTorrent, BitTorrent++, uTorrent, Shareaza, MLDonkey, ABC, Azureus, BitAnarch, SimpleBT, BitTorrent.Net, mlMac)
- **Blubster** (Blubster, Piolet)
- **WPNP** (WinMX)
- **Warez** (Warez, Ares; starting from 2.8.18) – 该协议能被丢弃掉（drop），但不能被限制速度

9.7 RouterOS L7 协议

RouterOS V3.0 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、QQ、MSN、魔兽世界……网络程序做限制和过滤。

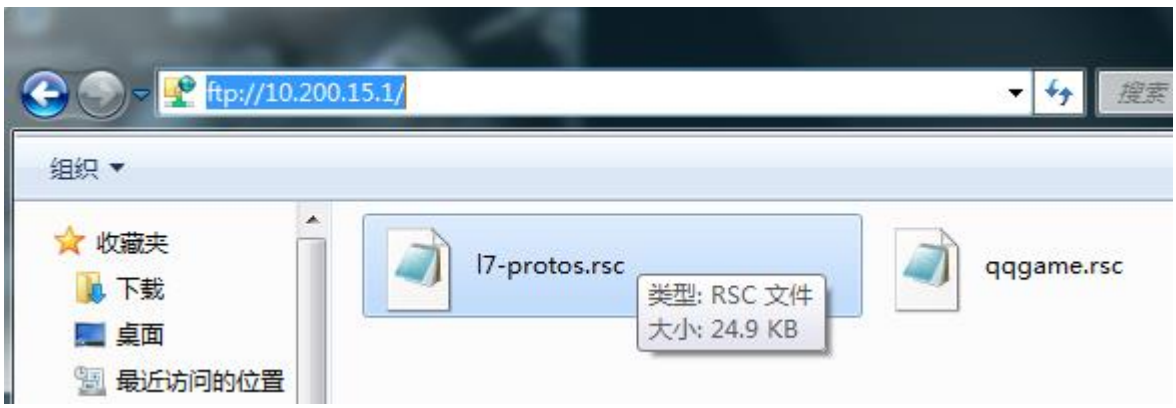
在防火墙 Layer7-protocol 目录下，你可以添加正则表达式字符串协议，定义他们的名称，并在防火墙 filter 目录下丢弃他们的数据。这个功能将不会查看单独的数据包，会查看整个数据的相关连接，收集数据直到第 10 个数据包或者 2kb 数据，来执行第一次操作

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资料了解。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。



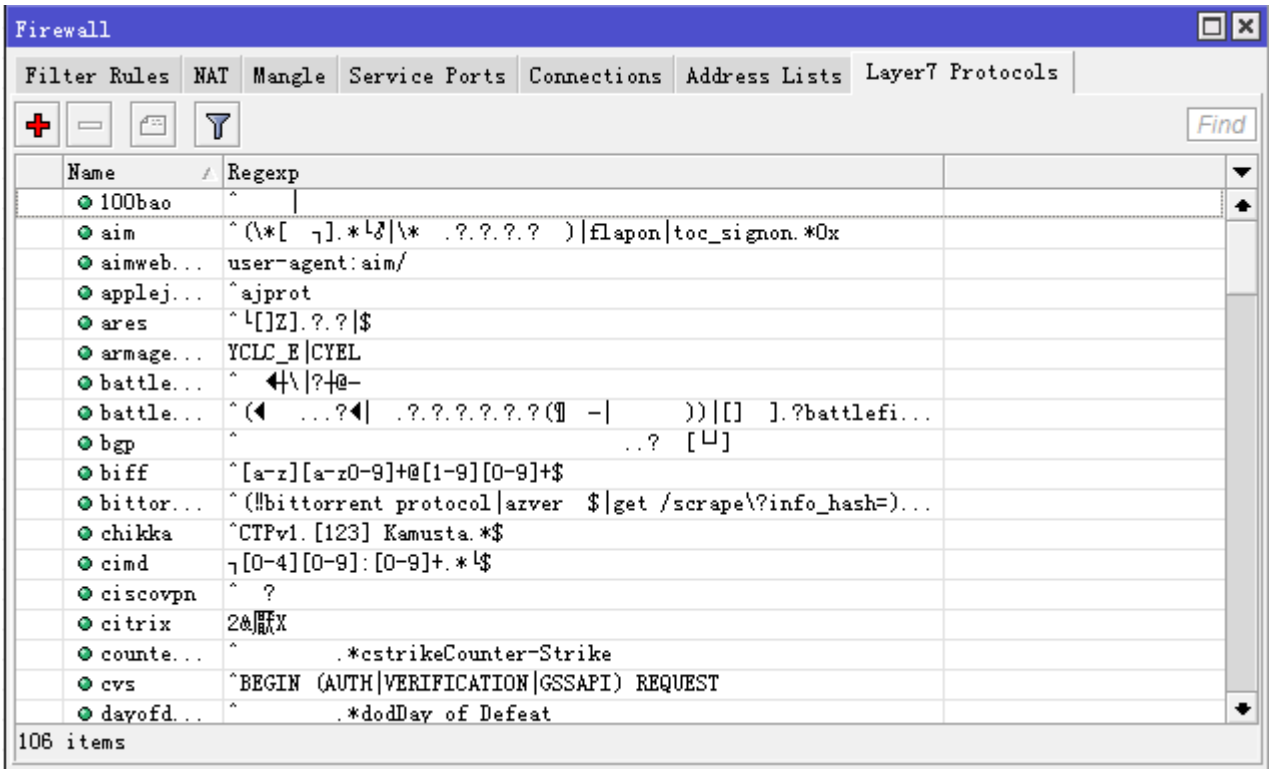
之后我们在命令行(Terminal)中导入 7 层协议脚本，用 import l7-protos.rsc 命令来导入脚本

```
[admin@MikroTik] > import l7-protos.rsc
Opening script file l7-protos.rsc
Script file loaded and executed successfully
```

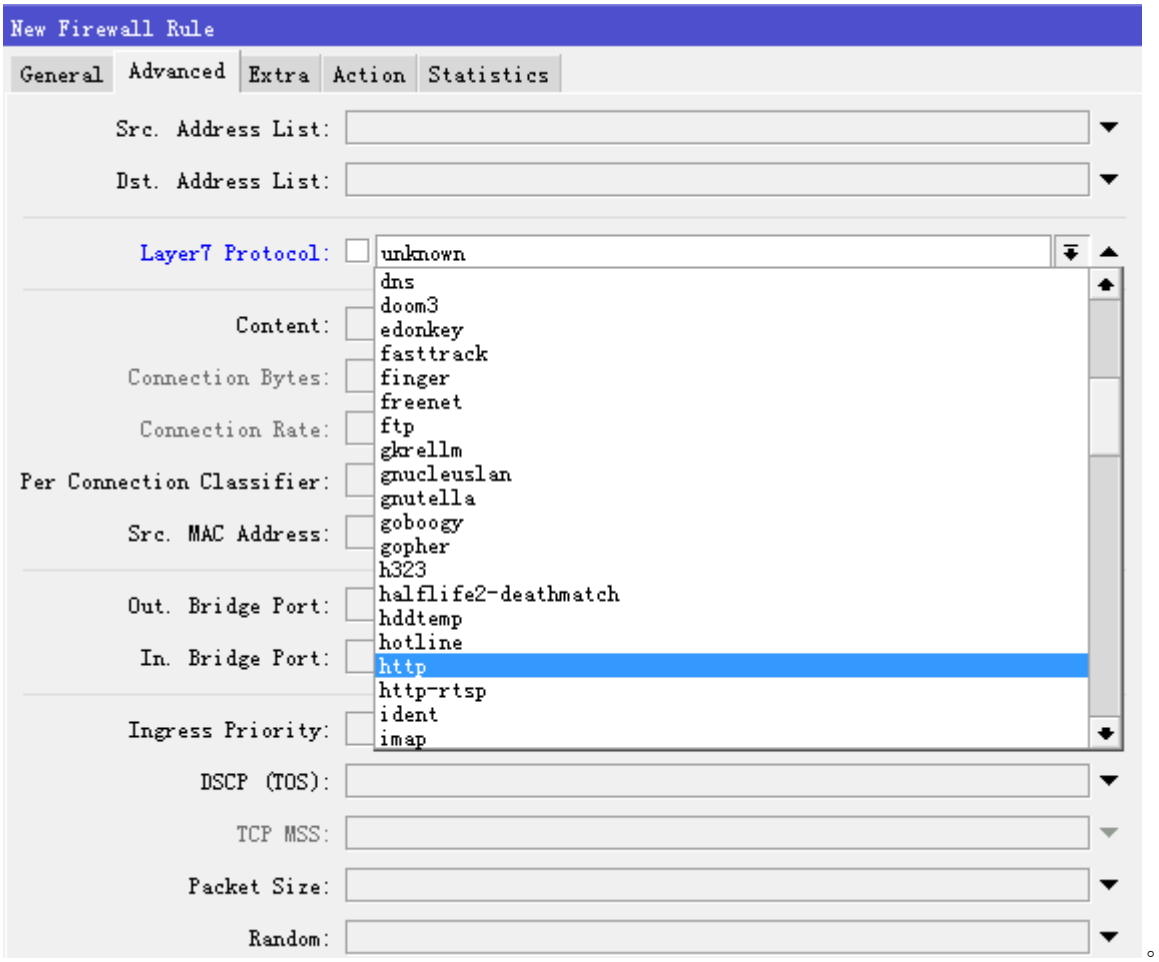
```
[admin@MikroTik] >
```

当系统提示 Script file loaded and executed successfully，说明脚本成功导入。

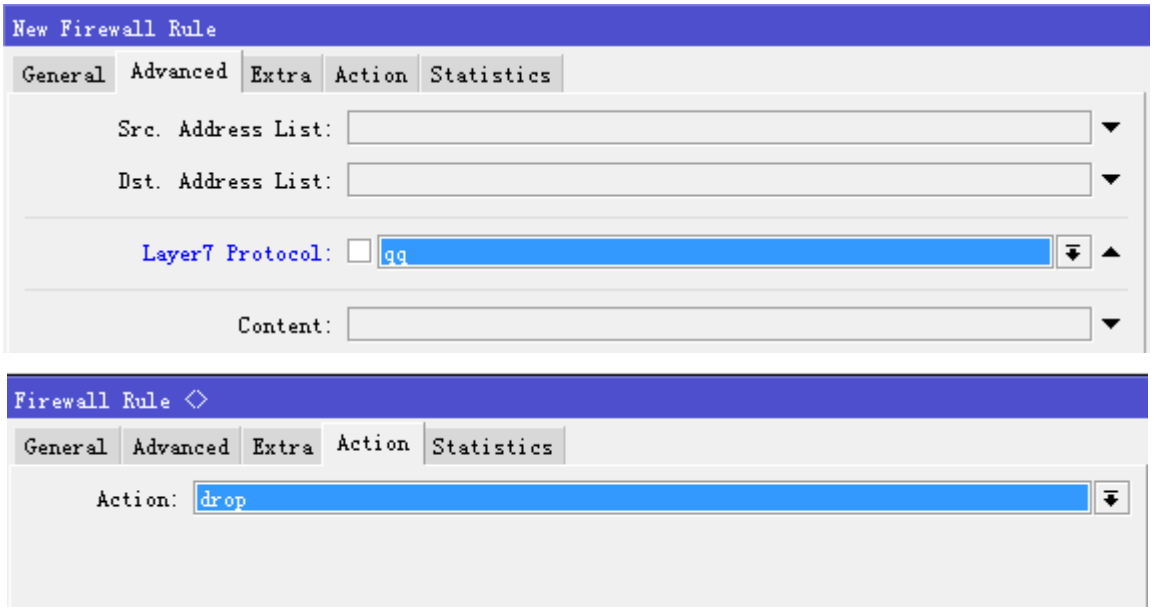
导入脚本后，我们可以在 Layer7 Protocols 中看到



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



在这里我们通过禁止登陆 QQ 为例，在这里我们禁止所有用户无法登陆 QQ。添加一条规则后，进入 Advanced 中的 Layer7 Protocols 选项选择 qq，然后在 Action 中设置为 drop 丢弃。注意：L7 禁止 QQ 的规则设置好后，需要重启才能生效。



其他的操作也同以上设置类似，如果需要对 IP 地址或者 IP 段控制可以通过 src-address 或者 dst-address 进行设置。

9.8 使用 wireshark 分析网页视频

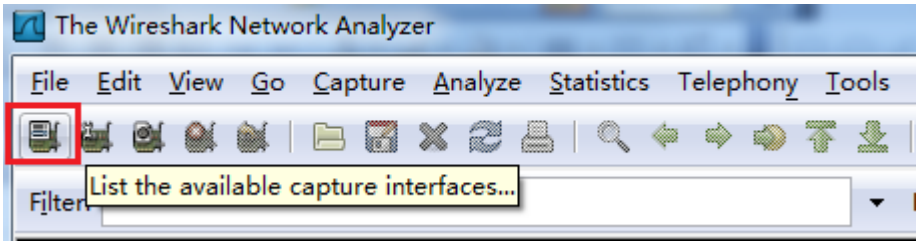
Wireshark 是非常著名的数据分析软件，我们首先需要下载 wireshark 并安装，具体 wireshark 的安装和详细操作请 baidu 或 google，下面简单介绍下 wireshark 如何抓包分析网页视频特征，并写出 L7 的代码。

注意：改分析内容仅供参考，可能会涉及后期相关内容变更。

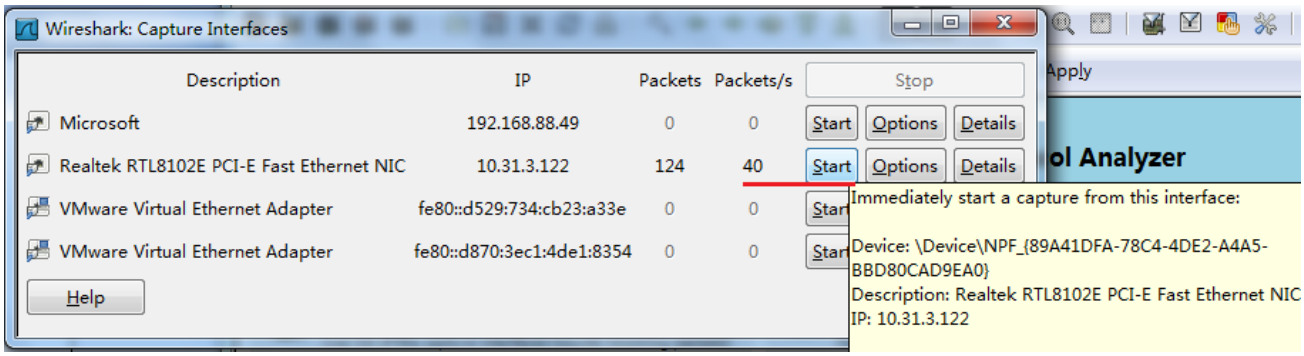
操作流程：

1、 启动 wireshark，并找到需要抓取数据包的网卡，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链接，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



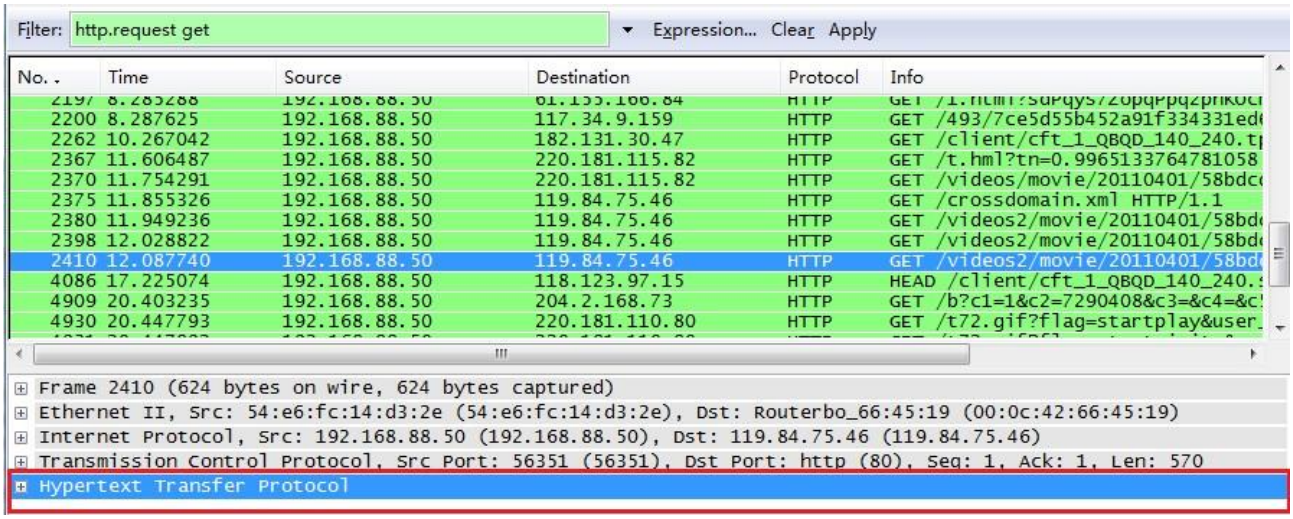
通过列表可以选择我们需要抓取数据包的网卡



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，

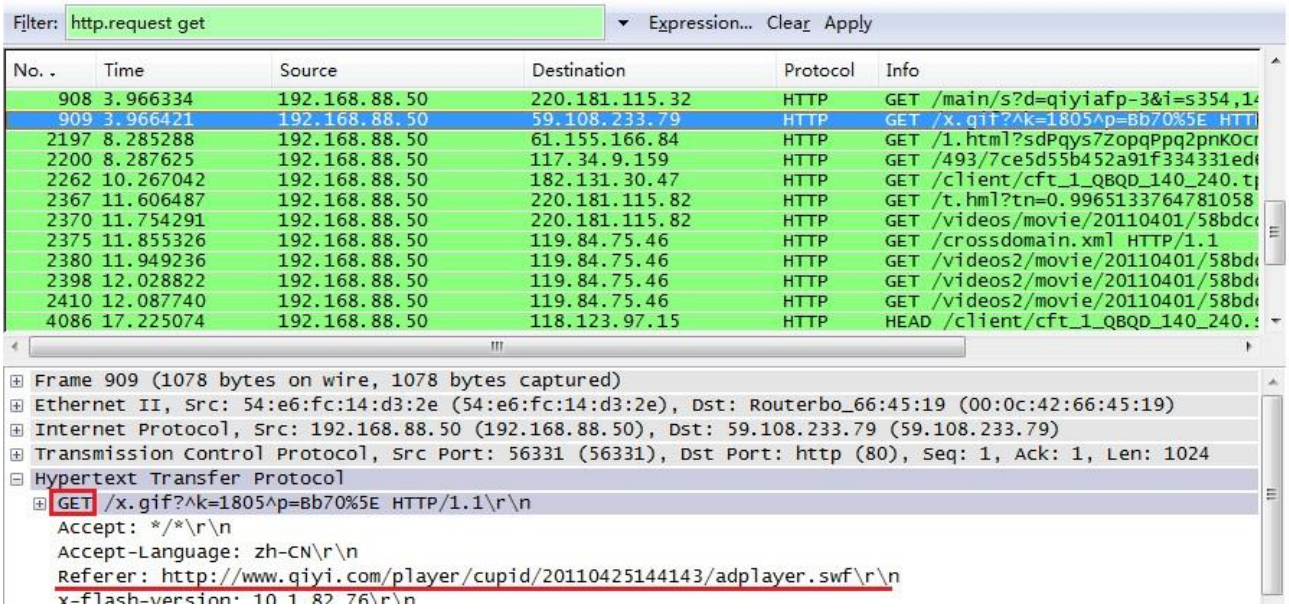
第三步、先点击 start 按钮开始抓取，然后点击播放网页视频

第四步、视频播放到一段时间后，我们可以停止抓取数据包，分析数据包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

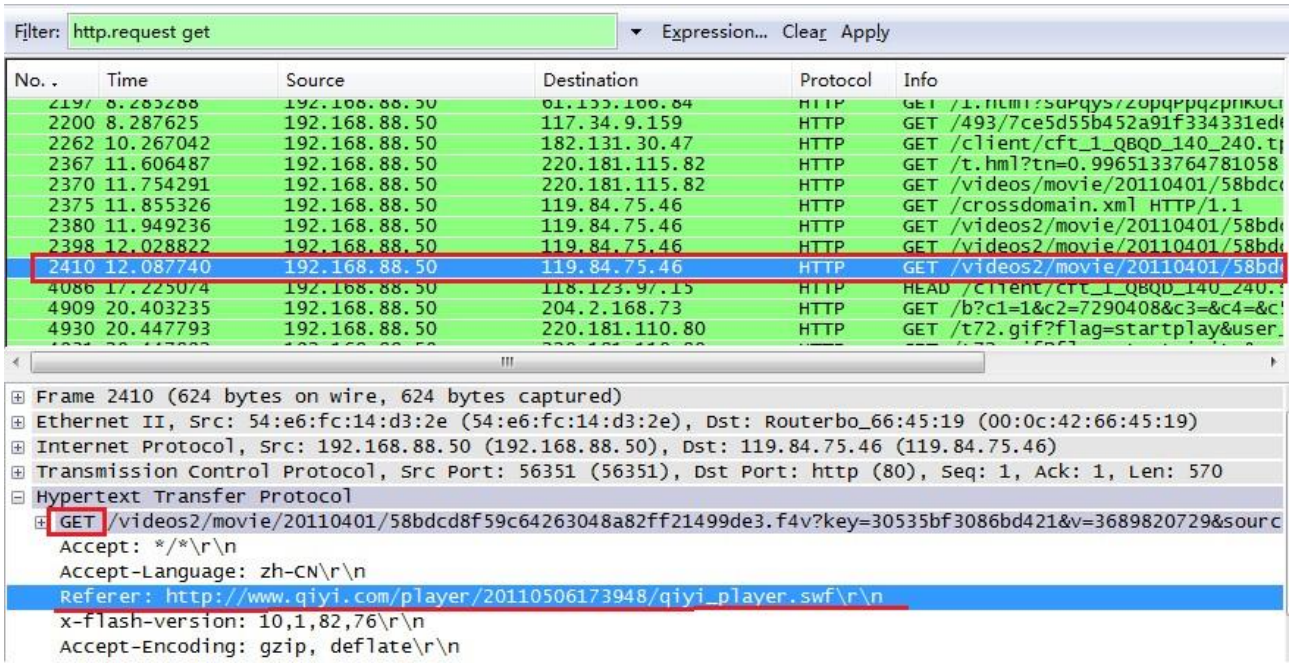
当然我们要找的是视频文件，如.swf 结尾的内容（有些可能是 mp4 之类），其他 jpg、js、png 这些都不用考虑。上图，我们找到了一个相关匹配的 GET 值，我们看到主机向 119.84.75.46 的服务器获取一个.swf 的 flash 视频文件，



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字，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字有

```
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
```

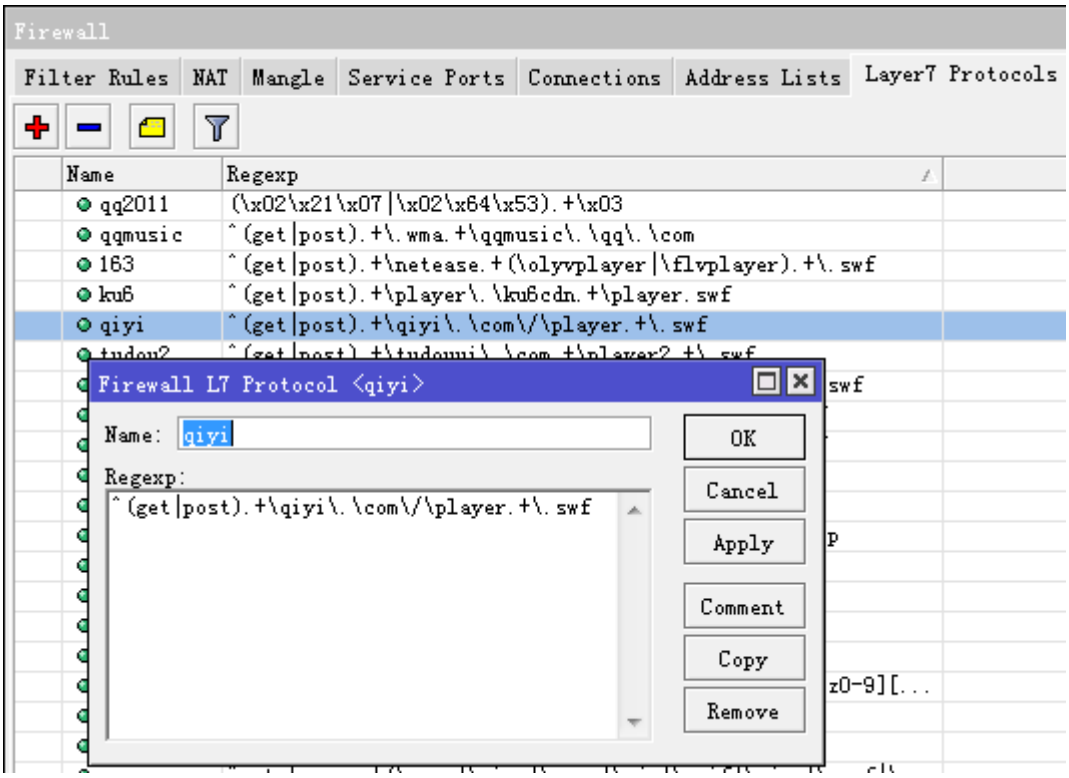
代码分析：

```
^ ：                标示内容的起始
(get|post) ：  中间的“|”标示 get 或者 post
.+ ：               中间任意字符
\ ：                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

下面是几个视频和网页的 L7 的正则表达式（该表达式仅供参考）

Sina 视频	^(get|post).+\you\.\video\.\sina\.\com\.\cn.+\.swf
Tudou 视频 1	^(get|post).+\tudouui\.\com.+\tudouvideoplayer.+\.swf
Tudou 视频 2	^(get|post).+\tudouui\.\com.+\player2.+\.swf
163 视频 1	^(get|post).+\ws\.\126\.\net\/\movieplayer.+\-.swf
163 视频 2	^(get|post).+\netease.+(\olyvplayer|\flvplayer).+\.swf
ku6 视频	^(get|post).+\player\.\ku6cdn.+\player.swf
qqmusic	^(get|post).+\.wma.+\qqmusic\.\qq\.\com
qqzone	^get.+qzone.+(\.css|\.ico|\.png|\.js|\.gif|\.jpg|\.swf|\.htm|\.html)
奇艺视频	^(get|post).+\qiyi\.\com\/\player.+\.swf

9.9 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 10.1.0.1/24，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>
```

9.10 RouterOS v6 ip settings

操作路径: /ip settings

RouterOS v6 新增了 ip 层的设置菜单，**ip settings** 允许设置几个 IP 协议相关的核心参数，由于 RouterOS 内核基于 Linux，所以这些参数基本来至于 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：	是否接受 ICMP 重定向信息，通常情况下在主机端启用，而路由

no)	器禁用
accept-source-route (*yes* | *no*; 默认: **no**)	是否接受数据包的 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
ip-forwarding (*yes* | *no*; 默认: **yes**)	启用或禁止 IP 路由数据包在接口之间转发。通常情况下开启该功能。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，即是否打开反向路径过滤功能，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，严格的逆向路径。每一个进入到数据包都会测试反向的 FIB，如果接口没有最优的逆向路径检查将失败。默认情况下失败的数据包被丢弃。 • loose – Loose 模式被定义在 RFC3704as ，宽松的逆向路径。每一个进入到数据包都会测试反向的 FIB，如果源地址是不可到达的数据包通过任何接口检查将会失败。 当前建议推荐启用 strict 模式阻止基于 IP 欺骗的 DDos 攻击，如果启用了非对称的路由或其他复杂的路由策略，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。
arp-timeout （时间秒; 默认:30s）	修改 arp 协议超时时间，默认是 30 秒

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 文件对比。

9.11 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%，路由器响应变得非常缓慢，甚至无响应。由于攻击者发送了大量的 syn 数据包，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包造成 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，但我们之间建设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 和其他数据包处理规则
- 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）

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
```

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 列表，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 攻击者数据包（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 网络地址翻译 nat

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 ADSL 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。

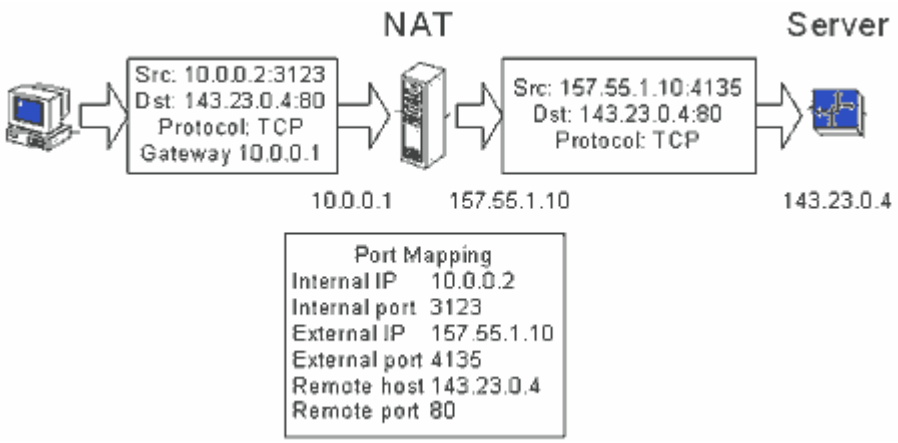
规格说明

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(已翻译)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。nat 网关的作用就是在数据包进/出局域网时重写 IP 地址的作用。

输出数据包转换的示例：



网络地址翻译包括两种类型:

- 源网络地址翻译或者 srcnat。这种类型的网络地址翻译工作在从一个 natted 网络产生的数据包上。nat 路由器在 IP 包通过它的时候用一个新的公网 IP 地址代替了其私有源地址。相反的操作适用于响应包从相反方向通过路由器时。
- 目标网络地址翻译或者 dstnat。 这种类型的网络地址翻译工作在到达一个 natted 网络的数据包上。它通常用于使一个私有网络上的主机能够被因特网访问。dstnat 路由器在 IP 包通过该路由器到达私有网络时替换了 IP 包的目标 IP 地址。

nat 缺点

在一个使用了网络地址翻译的路由器背后的主机并不拥有真实的端对端的连接。因此一些因特网协议就不在有网络地址翻译的情况下工作。一些来私有用网络外部或者无连接协议如 UDP 协议且需要 TCP 连接初始化的服务将被打断。此外，一些协议内在与 NAT 不兼容，一个鲜明的事例就是 IPsec 中的 AH 协议。

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）

当数据包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 请求被重定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

action (accept | add-dst-to-address-list | add-src-to-address-list | dst-nat | jump | log | masquerade | netmap | passthrough | redirect | return | same | src-nat; 默认: **accept**) -如果数据包与规则匹配 action 将启用

- **accept** - 接收数据包。不进行任何动作。例如：数据包通过而且没有其他任何适用于它的规则
- **add-dst-to-address-list** - 向 **address-list** 参数指定的地址表中添加 IP 包的目的地址
- **add-src-to-address-list** - 向 **address-list** 参数指定的地址表中添加 IP 包的源地址
- **dst-nat** - 用 **to-addresses** 及 **to-ports** 参数指定的变量取代 IP 包的目的地址
- **jump** - 跳转到由 **jump-target** 参数指定的链
- **log** - action 的每个匹配都将对系统日志添加一条消息
- **masquerade** - 以一个路由策略自动分配的 IP 地址取代 IP 包的源地址
- **netmap** - 创造一个 IP 地址从一端到另一端的静态 1：1 映像。通常用于分配公用 IP 地址到专用内网的主机上
- **passthrough** - 忽略次条规则并转到下一个规则
- **redirect** - 把 IP 包的目的地址替换成一个路由器的本地地址
- **return** -返回到跳转发生的链
- **same** - 从允许范围内分配给特定客户每个连接相同的源/目的 IP 地址。这种情况通常用于来自期望相同客户的相同客户地址对多重连接的服务。
- **src-nat** - 把 IP 包的源地址替换成由 **to-addresses** 和 **to-ports** 参数指定的值

address-list (*名称*) -指定地址列表的名称以收集使用了 **action=add-dst-to-address-list** 或 **action=add-src-to-address-list** 动作规则的 IP 地址。

address-list-timeout (*时间*; 默认: **00:00:00**) - 在 address-list 参数指定的地址列表删除地址之后的时间间隔。与 **add-dst-to-address-list** 或 **add-src-to-address-list** 动作一起起使用

- **00:00:00** - 从地址列表中永久删除

chain (dstnat | srcnat | *name*) – 选择或者定义一个规则的链。由于不同的数据流通过不同的链，所以为新规则选择正确的链必须很小心。如果输入一个与默认链名（srcnat 和 dstnat）不匹配，那么会生产一个新的链。

dstnat - 在这个链中的规则会在路由前被应用。代替 IP 包目的地址的规则应放在这里。

srcnat - 在这个链中的规则会在路由后被应用。代替 IP 包源地址的规则应放在这里。

comment (*文本*) - 对规则的描述性注解。一条注解能被用于从脚本中识别规则。

connection-bytes (*整型-整型*) - 当且仅当一定给定量字节从特定连接传输时与数据包进行匹配。

0 - 代表无穷大。例如：**connection-bytes=2000000-0** 如果大于 2MB 数据从相关连接传输就与规则匹配。

connection-limit (*整型*, 子网掩码) - 限制每个地址或地址群的连接限度。

connection-mark (*名称*) - 与通过 mangle 机制标记的特定连接数据包进行匹配

connection-type (ftp | gre | h323 | irc | mms | pptp | quake3 | tftp) - 与基于连接跟踪助手信息的相关连接的包进行匹配。相关连接助手必须在**/ip firewall service-port** 下启用

content (*文本*) - 文本数据包必须按顺序排列以与匹配规则

dst-address (*IP 地址*/掩码 | *IP address-IP address*) - 指定 IP 包的目的地址范围

address/netmask – 对合法网络地址的换算，例如：1.1.1.1/24 被转换为 1.1.1.0/24

dst-address-list (*名称*) - 在用户自定义的地址列表中匹配数据包的目的地址

dst-address-type (unicast | local | broadcast | multicast) - 在 IP 包的目的地址类型中匹配其中之一

unicast - 用于点对点传输的 IP 地址。这种情况仅限于一个发送者和一个接受者

local -与分配到路由器接口的地址匹配

broadcast - 这个 IP 包从 IP 子网的一个点到其他所有点发送信号

multicast - 这种类型的 IP 地址负责从一个或多个点到其他一系列点的传输

dst-limit (*整型*/*时间*{0,1},*整型*, dst-address | dst-port | src-address{+},*time*{0,1}) - 在每个目的 IP 或者每个目的端口库上限制每秒数据包绿（pps）。与 **limit** 匹配相反，每个目的 IP 地址/目的端口都有自己的限度。其选项如下（按出现次序）：

Count - 最大平均包率。以 pps 衡量，除非跟随在 **Time** 选项之后。

Time - 指定包率衡量的时间间隔

Burst - 以成组方式匹配的包数量

Mode - 包率限制分类方式

Expire - 指定已记录的 IP 地址/端口将被删除的过期时间，时间间隔。

dst-port (*整型*: 0..65535-*整型*: 0..65535{*}) - 目的端口数或范围

hotspot (*多选项:* from-client | auth | local-dst) - 从各种不同的 Hot-Spot 中匹配从客户获得的包。所有值都可以被取消。

from-client - 如果一个包来自于 HotSpot 客户则为真

auth - 如果一个包来自验证用户则为真

local-dst - 如果一个包拥有本地目的 IP 地址则为真

icmp-options (*整型*:*整型*) - 与 ICMP 的 Type:Code 域匹配

in-interface (*name*) - interface the packet has entered the router through

ipv4-options (any | loose-source-routing | no-record-route | no-router-alert | no-source-routing | no-timestamp | none | record-route | router-alert | strict-source-routing | timestamp) - 与 ipv4 标题选项匹配

any - 与 ipv4 选项中至少一个匹配

loose-source-routing - 与发射源路由选项的包进行匹配。次选项一般用于路由基于源提供信息的因特网数据报

no-record-route - 以无记录路由选项匹配包。次选项一般用于路由基于源提供信息的因特网数据报

no-router-alert - 以无路由警报选项匹配包

no-source-routing - 以无源路由选项匹配包

no-timestamp - 以无时间印章选项匹配包

record-route - 以记录路由选项匹配包

router-alert - 以路由警报选项匹配包
strict-source-routing - 以严密的源路由选项匹配包
timestamp - 以时间印章选项匹配包
jump-target (dstnat | srcnat*name*) - 将要跳转的目标链名称，如果使用了动作 **action=jump**
limit (*整型*/*时间*{0，1}，*整型*) - 按给定限度限制包匹配率。对于减少日志信息数量有用
Count - 最大平均包率。以 pps 衡量，除非跟随在 **Time** 选项之后。
Time - 指定包率衡量的时间间隔
Burst - 以成组方式匹配的包数量
log-prefix (*文本*) - 所有写入日志的信息都包含次中指定的前缀。与 **action=log** 一起使用。
nth (*整型*, *整型*: 0..15，*整型*{0,1}) - 与特定的由规则获取的第 N 个包匹配。16 个可用计数器之一可被用来计算包数
Every - 匹配每第 **Every+1** 个包。例如：如果 **Every=1** 那么规则匹配每第二个包
Counter - 指定要使用的计数器。
Packet - 以给定包的数量进行匹配。显然地，这个值必须在 **0** 和 **Every** 之间。如果这个选项用于一个给定的计数器，那么在这个选项里必须至少有 **Every+1** 个规则，以包含所有在 **0** 和 **Every** 之间的值
out-interface (*name*) - 离开路由器的包的接口
packet-size (*整型*: 0..65535-*整型*: 0..65535{0,1}) - 按字节匹配指定大小或大小范围的包
Min - 指定大小范围或独立的值的下限
Max - 指定大小范围的上限
phys-in-interface (*name*) -与添加到一个桥设备的桥端口物理输入设备匹配。仅在数据包从桥到达并通过路由器时有用
phys-out-interface (*name*) - 与添加到一个桥设备的桥端口物理输出设备匹配。仅在数据包从桥离开路由器时有用
protocol (ddp | egp | encap | ggp | gre | hmp | icmp | idrp-cmtp | igmp | ipencap | ipip | ipsec-ah | ipsec-esp | iso-tp4 | ospf | pup | rdp | rspf | st | tcp | udp | vmtp | xns-idp | xtp | *整型*) - 与由协议名称或编号指定的特定 IP 协议匹配。如果你想指定端口就应该进行这个配置。
psd (*整型*, *时间*, *整型*, *整型*) - 试图探测 TCP 及 UDP 扫描。建议对高号码端口分配低权重以减少被误判的频率，例如来自被动模式的 FTP 迁移
WeightThreshold - 来自不同主机且被作为端口扫描序列的带有不同目的端口的最新的 TCP/UDP 包的总权重值
DelayThreshold - 来自同意主机且被当作可能端口扫描子序列带有不同目的端口的包延迟
LowPortWeight - 特权目的端口（<=1024）的数据包权重值
HighPortWeight –非特权目的端口（<=1024）的数据包权重值
random (*整型*) - 以给定概率随机匹配包
routing-mark (*name*) - 对 mangle 标记的特定路由的包进行匹配
same-not-by-dst (yes | no) - 当选择要与 **action=same** 规则匹配的包的新源 IP 地址时指定是否对目的 IP 地址进行计数
src-address (*IP 地址*/子网掩码 | *IP 地址* - *IP 地址*) - 指定源 IP 包产生的地址范围。
src-address-list (*name*) - 与用户定义的地址列表中的数据包源地址匹配
src-address-type (unicast | local | broadcast | multicast) - 与 IP 包的源地址类型中的一个匹配
unicast - 用于点对点传输的 IP 地址。这种情况仅限于一个发送者和一个接受者
local -与分配到路由器接口的地址匹配
broadcast - 这个 IP 包从 IP 子网的一个点到其他所有点发送信号
multicast - 这种类型的 IP 地址负责从一个或多个点到其他一系列点的传输
src-mac-address (*MAC address*) - 源 MAC 地址
src-port (*整型*: 0..65535-*整型*: 0..65535{*}) - 源端口数或范围
tcp-mss (*整型*: 0..65535) - 与 IP 包的 TCP MSS 值匹配

time (*时间-时间*, sat | fri | thu | wed | tue | mon | sun{+}) - 允许产生基于数据包到达时间和日期的过滤器，或者对于本地产生的数据包的离开时间和日期
to-addresses (*IP address-IP address*{0,1}; default: **0.0.0.0**) -取代初始 IP 包地址的地址或地址范围
to-ports (*整型*: 0..65535-*整型*: 0..65535{0,1}) - 取代初始 IP 包端口的端口或端口范围
tos (max-reliability | max-throughput | min-cost | min-delay | normal) - 对 IP 头服务类型（ToS）域的值指定一个匹配
max-reliability – 最大的可靠性 (ToS=4)
max-throughput – 最大的吞吐量 (ToS=8)
min-cost – 最低的成本代价(ToS=2)
min-delay – 最小的延迟 (ToS=16)
normal – 普通服务 (ToS=0)

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，你应该使用 MikroTik 路由器的源网络地址翻译特性。当数据包通过路由器时，伪装将把从 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。为了使用伪装，必须向 nat 配置中添加一个带有“隐藏”动作的的源网络地址翻译规则：

```
/ip firewall nat add chain=srcnat action=masquerade 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 10.5.8.109 作为源地址，1024 作为源端口。因特网将不可能访问本地地址。如果你允许对本地网络服务器访问，你应该使用目的网络地址翻译（nat）。

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，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

```
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，但无法解决不了解 IP 地址的情况，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
```

在 RouterOS 5.0 后，可以实现通过公网接口的方式配置端口映射，即 dst-address 参数可以不用写明 IP，仅指明公网网卡的进入接口即可，例如上面的规则我们也可以写为：

```
/ip firewall nat add action=dst-nat chain=dstnat in-interface=Public \
    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=dst-nat 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

如果你想从公用 IP 子网 11.11.11.1/32 访问本地的 2.2.2.2/32，你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=2.2.2.2

/ip firewall nat add chain=srcnat src-address=2.2.2.2 \
        action=netmap to-addresses=11.11.11.1
```

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对称。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：

- 可以对原始映射端口的重复映射
- 保证访问安全性，对方不容易知道内部映射端口是多少
- 有利于端口映射到多次和灵活处理

使用非对称的端口映射前提要保证，采用默认的 masquerade 伪装规则，即对内外网络进行伪装。

```
/ip firewall nat add chain=srcnat action=masquerade
```

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
```

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。

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** 意思即数据包是已知连接的一部分，**new** 意思为数据包开启了一个新连接，**related** 意为数据包开始了一个新连接，但与一个已存在连接想联系，如 FTP 数据传输或 ICMP 错误信息，**invalid** 意为数据包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包在 **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。

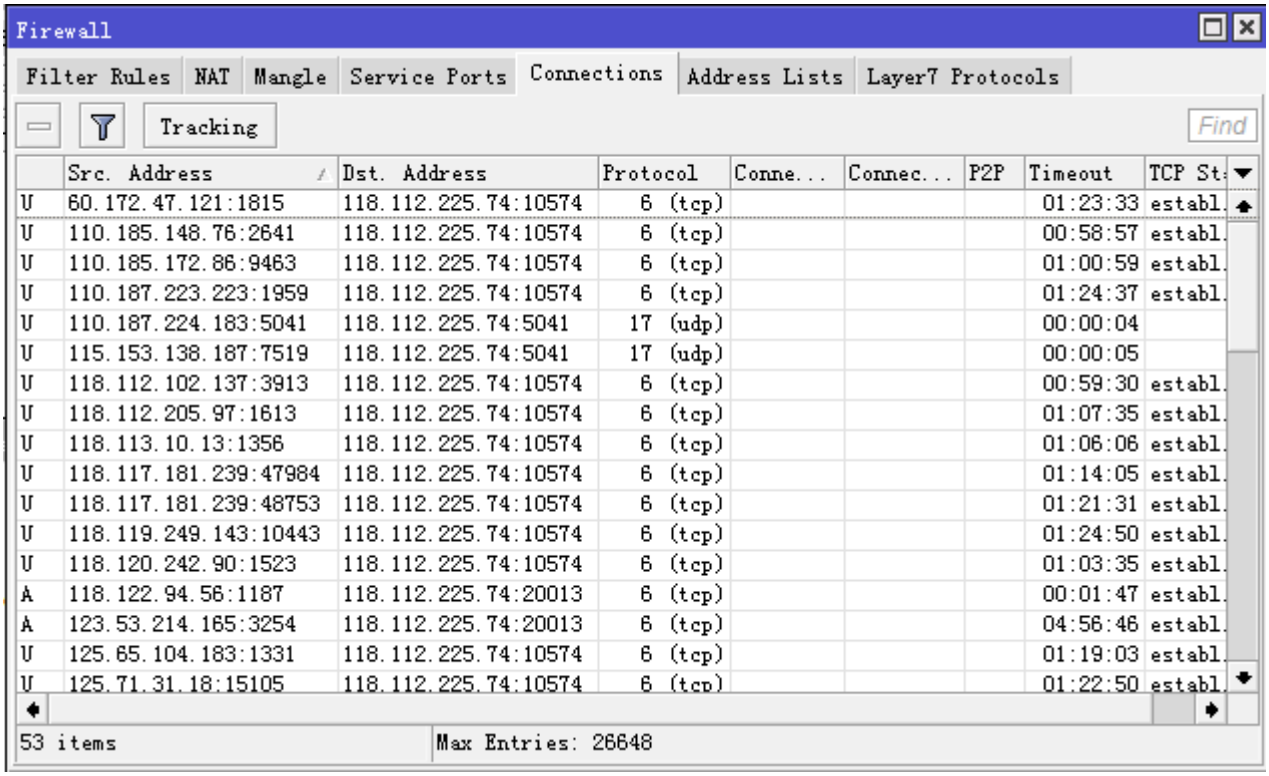
属性描述

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 协议名和序号
p2p (*只读: 文本*) – P2P 协议
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 以使发送器能识别回应并能够用适当的 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

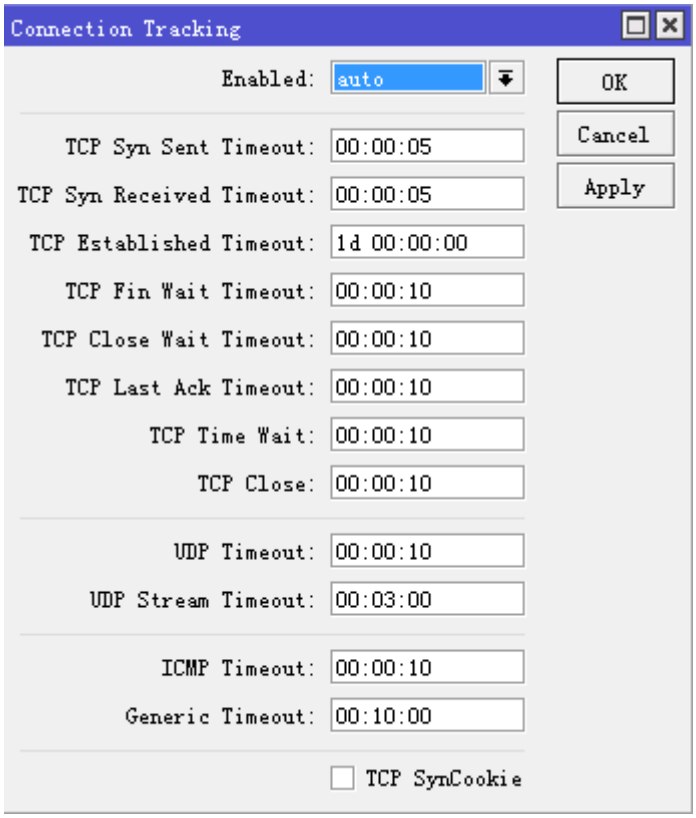
unreplied (*只读:* true | false) - 显示请求是否未被回应



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 连接的转换状态跟踪，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列表中删除。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列表中的连接数量。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包率，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。

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，不做会话的链接跟踪意味着数据包不需要发给 CPU 处理，而直接通过交换芯片完成，后面 Fastpath 会讲到。

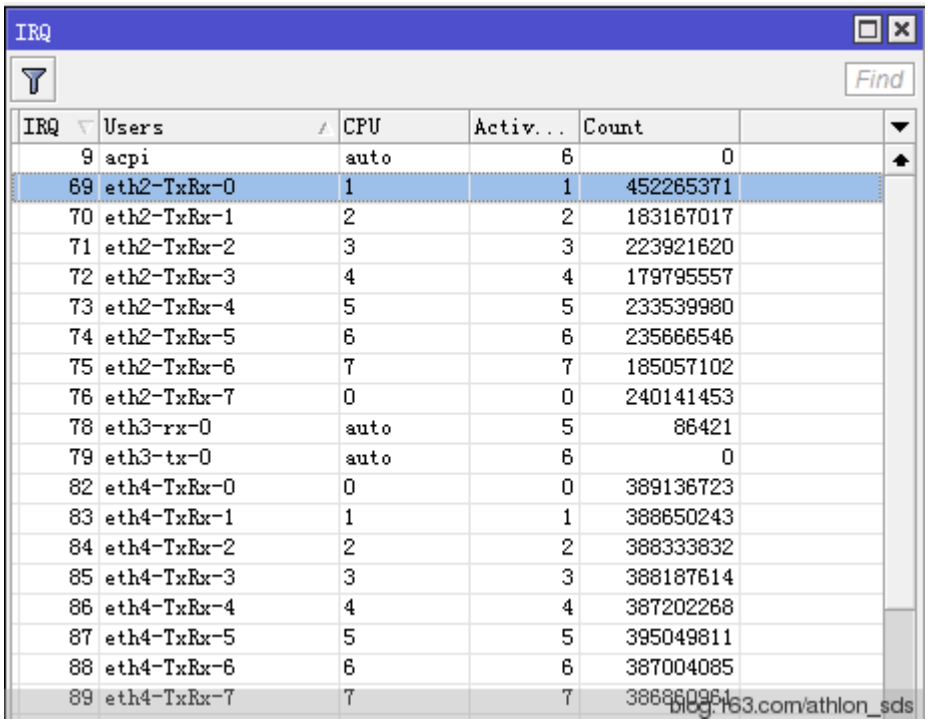
10.7 高性能 nat 实践

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：

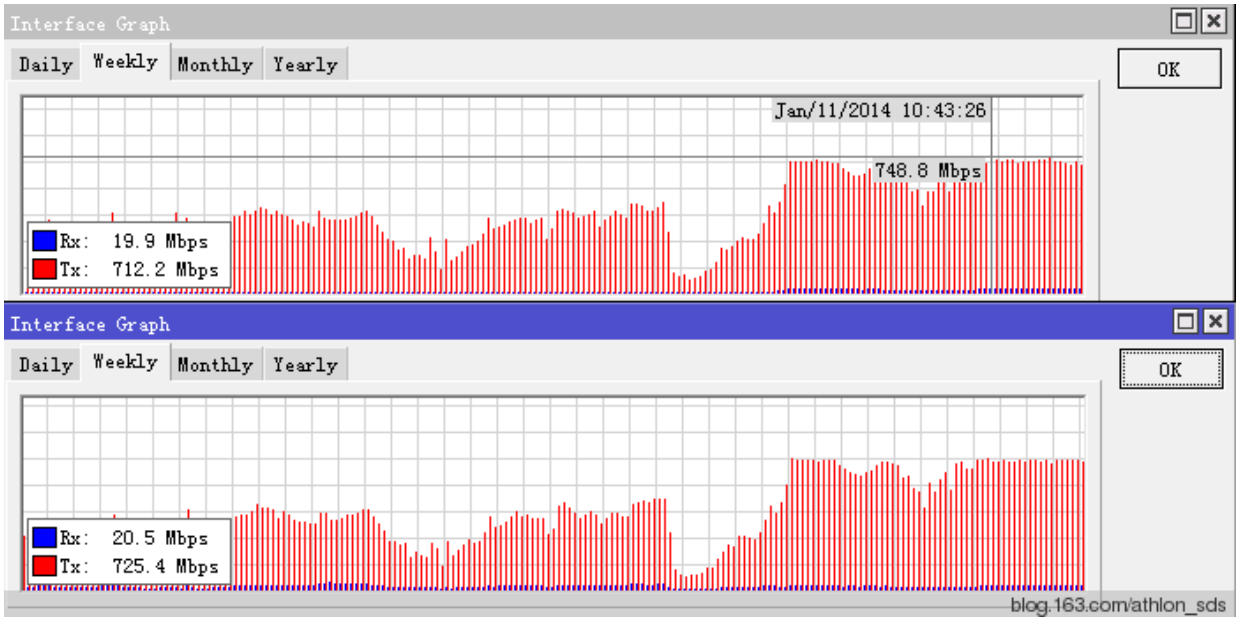
- Intel Xen 5606 × 2 （未打开超线程，一共使用 8 核）+ Intel 芯片组（别人的主机，主板只知道是 ASUS，型号没有记）
- 内存： DDR3 ECC 内存 2G ×2 （双通道需要配置 2 根 2G 内存，当然 RouterOS 只能识别 2G）
- 硬盘：1G Flash
- 网卡：Intel 82580 4 口网卡，每个网口中断 8
- RouterOS v6.6，功能包：仅安装 system、ppp 和 advanced-tools

使用中，手动调整了 IRQ，指定网卡每个终端负载到相应的 CPU：

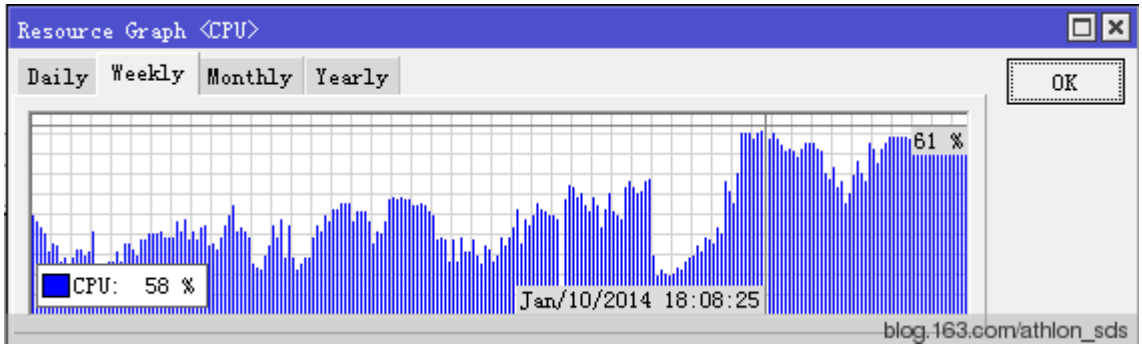


该配置主要用于跑 nat，其配置均省略，但出现过配置 simple queue 做整体流控后自动重启现象，后取消流控配置后，运行均无出现死机情况，这点估计是 RouterOS 在 Queue 还存在 bug。

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：



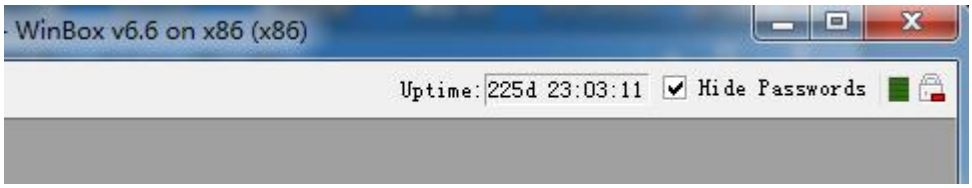
CPU 负载情况：



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：



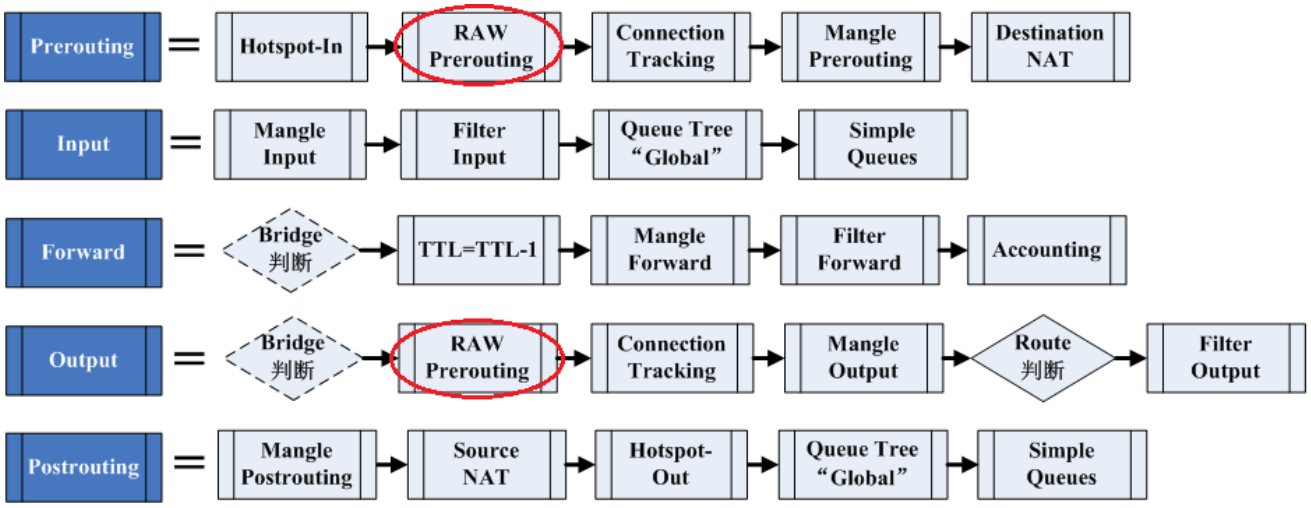
在此案例中网卡的中断数量对 CPU 均衡起到较大作用，所以在针对大流量的网络处理上，配合多 CPU 必须选择中断数高的网卡，以上内容供大家参考！

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 列表不能匹配链接状态的跟踪，例如 connection-state 和 L7 协议。

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，不予响应，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包
- **output** – 用于处理数据包源于路由器，并通过其中一个接口离开，即由路由器自身发出的数据包

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 做链接跟踪，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

第十一章 RouterOS Fastpath

RouterOS 基于 RouterBOARD 硬件支持新的功能 Fast path，该功能允许数据包转发不在经过 Linux 内核处理，直接由芯片处理后转发，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。

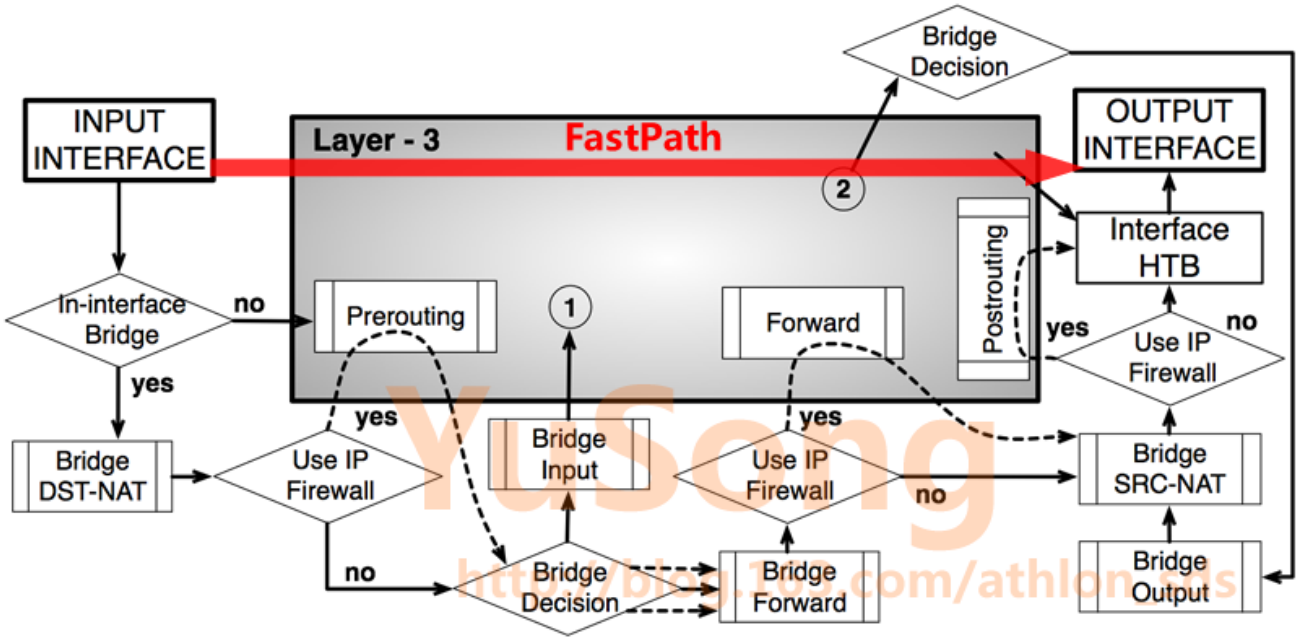
由于 RouterOS v6 开启 Fastpath 有部分功能限制，但当所有条件都满足时，Fastpath 会自动工作。当你没有防火墙规则（在将来的版本还会支持多种防火墙规则），如 nat 等，connection tracking 将自动关闭，Fastpath 自动启动生效，这样将使得 RouterOS 的吞吐量倍增。因此在新的版本中加入了 connection tracking 默认设置为 auto。

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 v6.8 版本：

- Firewall 规则不能做配置；
- ~~Traffic-flow 需关闭（/ip traffic-flow enabled=no），~~ 6.33 取消该限制
- Queue Simple 和 trees 的 parent=global 不能配置；
- 源接口不能设置为 bridge 接口或 bonding slave 接口；
- 目标接口 queue 设置为 only-hw-queue，并且 queue tree 条目的 parent 不能设置“dst-interface”
- 不能将接口设置 mesh 和 metarouter；
- sniffer, torch 和 traffic generator 不能运行；
- connection tracking 不能开启；
- ip accounting 功能必须关闭 (/ip accounting enabled=no);
- VRFs 不能设置 (/ip route vrf 规则为空);
- Hotspot 不能启用(/ip hostspot 不能添加任何服务接口)；
- IpSec 策略不能配置 (ROS v6.8)；
- ~~mac-ping,mac-telnet 或 mac-winbox 会话连接未启用；~~ 6.33 取消该限制
- /tool mac-scan 未被使用;
- /tool ip-scan 未被使用;
- /tool wol 未被使用;
- /interface mesh traceroute 未被使用;

“/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。

注意：当前 VLAN、PPP 和 Bonding 接口是无法支持 FastPath。（从 6.30 开始 Fastpath 支持 VLAN 和 VRRP 接口，同时支持 Bonding 接口的 rx 方向）



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 RouterBOARD FastPath 支持列表

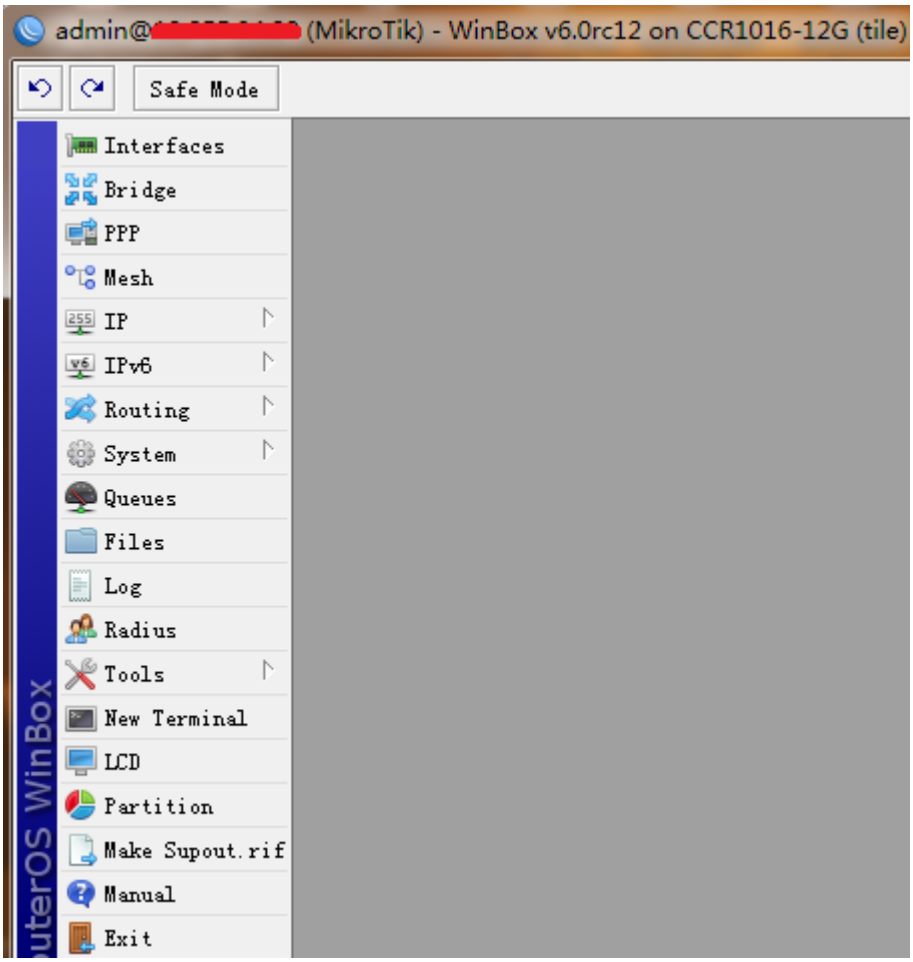
FastPath 功能仅支持部分网卡的快速转发，在下面的表中，列出了支持的 RouterBOARD 设备和其所属的网卡

RouterBoard	网络接口
RB3xx 系列	ether1,2
RB6xx 系列	ether1,2
RB7xx 系列	所有以太网卡
RB8xx 系列	ether1,2

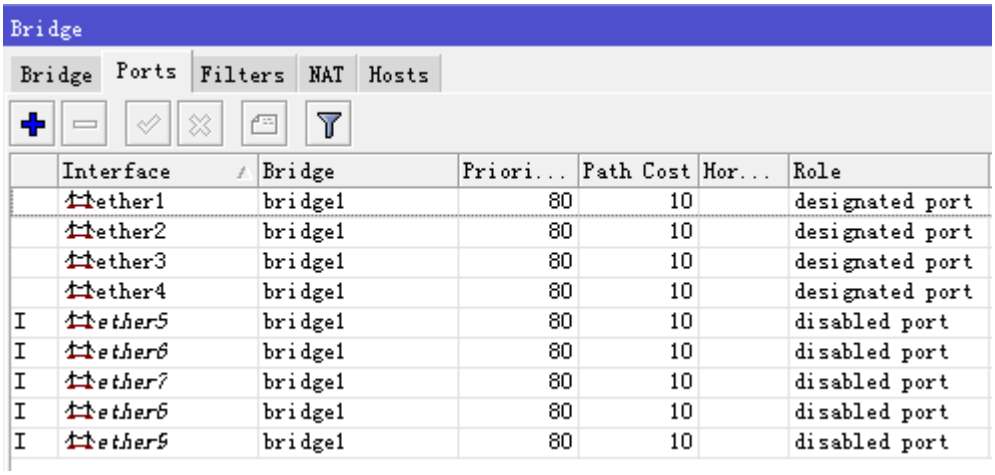
RB9xx 系列	所有以太网卡
RB1000	所有以太网卡
RB1100 系列	ether1-10,11
RB2011 系列	所有以太网卡和 SFP 接口
CCR 系列	所有以太网卡和 SFP 接口

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 16、36 和 72 核三个版本，但在该系列机型上你不会找到 switch 功能菜单，这点让我感到奇怪，对此我进行了一点点分析，在 RouterOS 的 tile 版本中没有找到相关 switch 的单独功能包，即说明 CCR 系列没有开设 Switch 功能

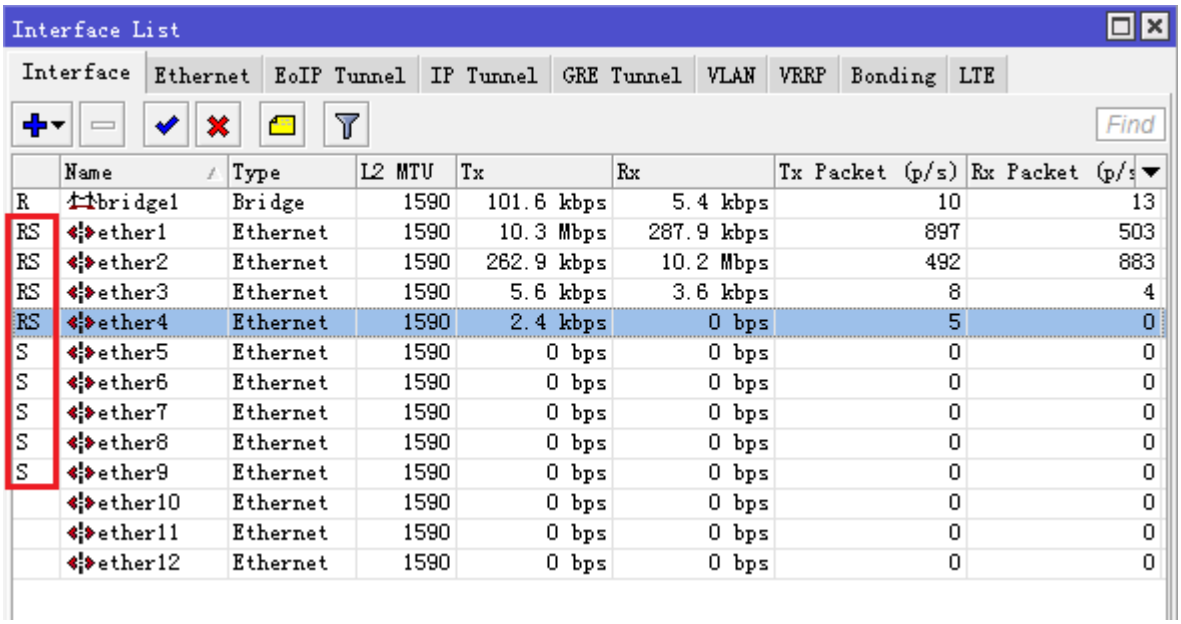


但当我们将 Bridge 桥接功能开启后，并在 bridge port 中将指定的网卡加入 bridge1 后，会发现在 interface 中网卡前缀自动加上了 S 标记，即 Slave，与其他 RouterBOARD 配置 switch 功能一样的前缀标示



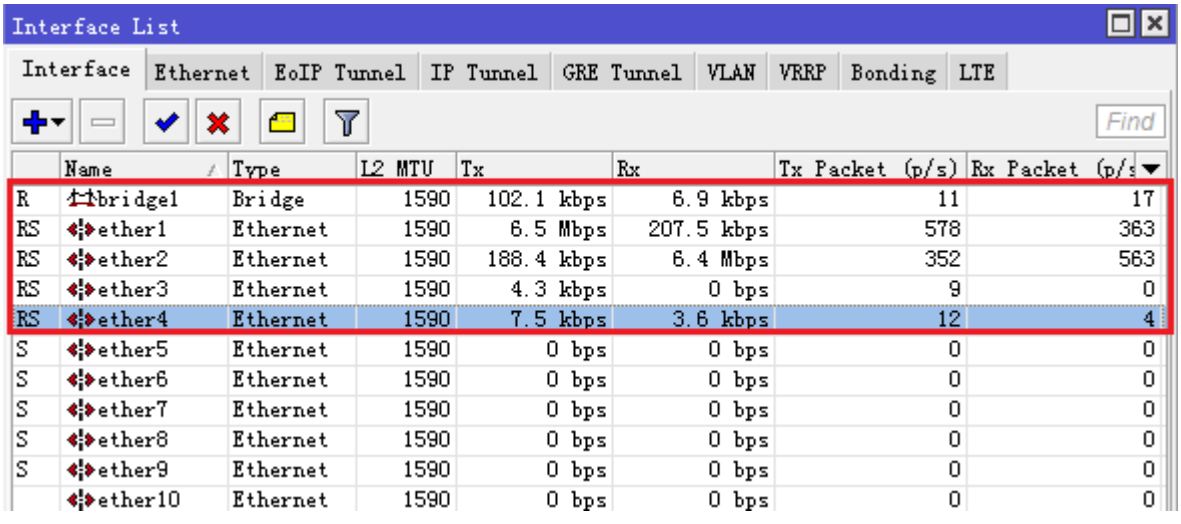
	Interface	Bridge	Priori...	Path Cost	Hor...	Role
	ether1	bridge1	80	10		designated port
	ether2	bridge1	80	10		designated port
	ether3	bridge1	80	10		designated port
	ether4	bridge1	80	10		designated port
I	ether5	bridge1	80	10		disabled port
I	ether6	bridge1	80	10		disabled port
I	ether7	bridge1	80	10		disabled port
I	ether8	bridge1	80	10		disabled port
I	ether9	bridge1	80	10		disabled port

Interface 中看到 Slave 的前缀



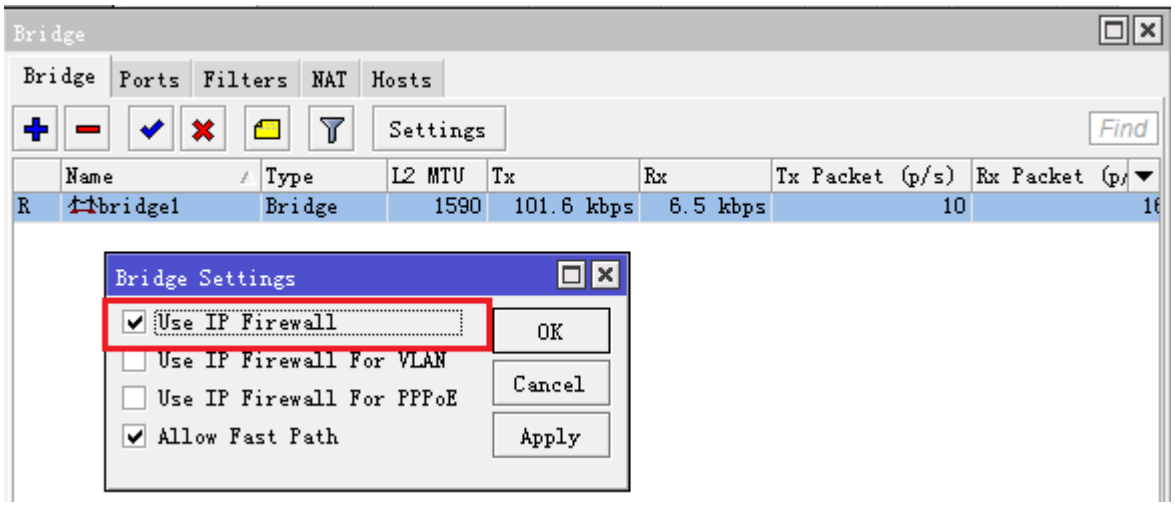
	Name	Type	L2 MTU	Tx	Rx	Tx Packet (p/s)	Rx Packet (p/s
R	bridge1	Bridge	1590	101.6 kbps	5.4 kbps	10	13
RS	ether1	Ethernet	1590	10.3 Mbps	287.9 kbps	897	503
RS	ether2	Ethernet	1590	262.9 kbps	10.2 Mbps	492	883
RS	ether3	Ethernet	1590	5.6 kbps	3.6 kbps	8	4
RS	ether4	Ethernet	1590	2.4 kbps	0 bps	5	0
S	ether5	Ethernet	1590	0 bps	0 bps	0	0
S	ether6	Ethernet	1590	0 bps	0 bps	0	0
S	ether7	Ethernet	1590	0 bps	0 bps	0	0
S	ether8	Ethernet	1590	0 bps	0 bps	0	0
S	ether9	Ethernet	1590	0 bps	0 bps	0	0
	ether10	Ethernet	1590	0 bps	0 bps	0	0
	ether11	Ethernet	1590	0 bps	0 bps	0	0
	ether12	Ethernet	1590	0 bps	0 bps	0	0

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。



	Name	Type	L2 MTU	Tx	Rx	Tx Packet (p/s)	Rx Packet (p/s
R	bridge1	Bridge	1590	102.1 kbps	6.9 kbps	11	17
RS	ether1	Ethernet	1590	6.5 Mbps	207.5 kbps	578	363
RS	ether2	Ethernet	1590	188.4 kbps	6.4 Mbps	352	563
RS	ether3	Ethernet	1590	4.3 kbps	0 bps	9	0
RS	ether4	Ethernet	1590	7.5 kbps	3.6 kbps	12	4
S	ether5	Ethernet	1590	0 bps	0 bps	0	0
S	ether6	Ethernet	1590	0 bps	0 bps	0	0
S	ether7	Ethernet	1590	0 bps	0 bps	0	0
S	ether8	Ethernet	1590	0 bps	0 bps	0	0
S	ether9	Ethernet	1590	0 bps	0 bps	0	0
	ether10	Ethernet	1590	0 bps	0 bps	0	0

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，将引入新的功能 FastTrack，根据官方解释可以使你的路由器 Firewall/NAT 性能提升 5 倍，官方介绍 FastTrack= FastPath + Conntrack，主要特点为：

1. FastTrack 加速指定的连接跟踪条目的数据包处理
2. FastTrack 具备 Full NAT 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，并允许数据包被定义为 FastTrack
7. 并非所有的数据包都能被 FastTrack 处理，有一些数据包将仍然使用普通的 Conntrack（连接跟踪）

IPv4 FastTrack 管理器会自动标记会话连接，用 firewall 的 action=fasttrack-connection 标记连接为 fasttrack。 当前仅支持 TCP 和 UDP 连接，IPv4 FastTrack 管理器支持包括 SNAT 和 DNAT

注意，fasttrack 数据包将直接绕过 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；
- /tool mac-scan 未被使用；
- /tool ip-scan 未被使用；
- /tool wol 未被使用；
- /interface mesh traceroute 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）

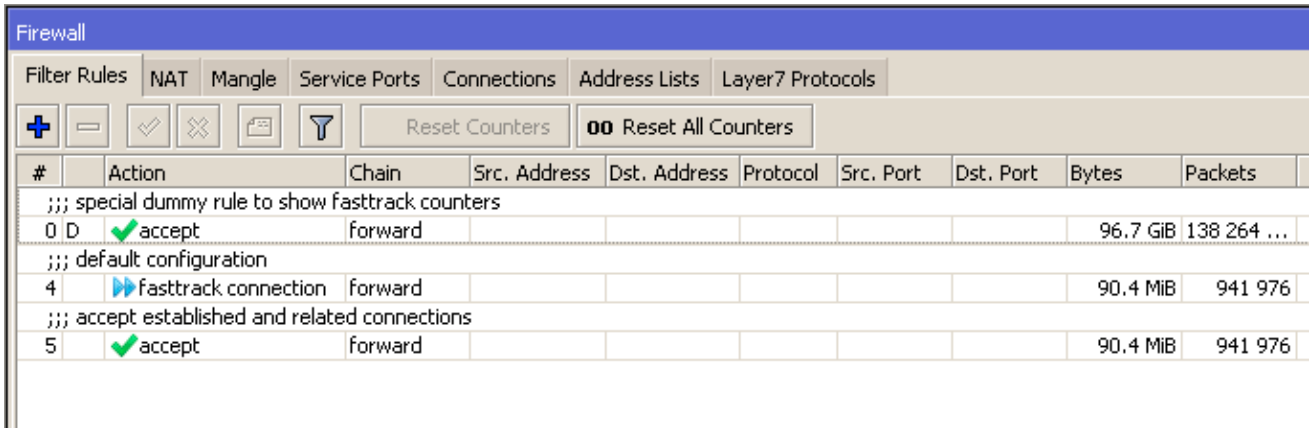
Fasttrack 支持的硬件设备列表.

RouterBoard	接口
RB6xx 系列	ether1,2
RB7xx 系列	所有接口
RB800	ether1,2
RB9xx 系列	所有接口
RB1000	所有接口
RB1100 系列	ether1-11
RB2011 系列	所有接口
RB3011 系列	所有接口
CRS 系列	所有接口
CCR 系列	所有接口

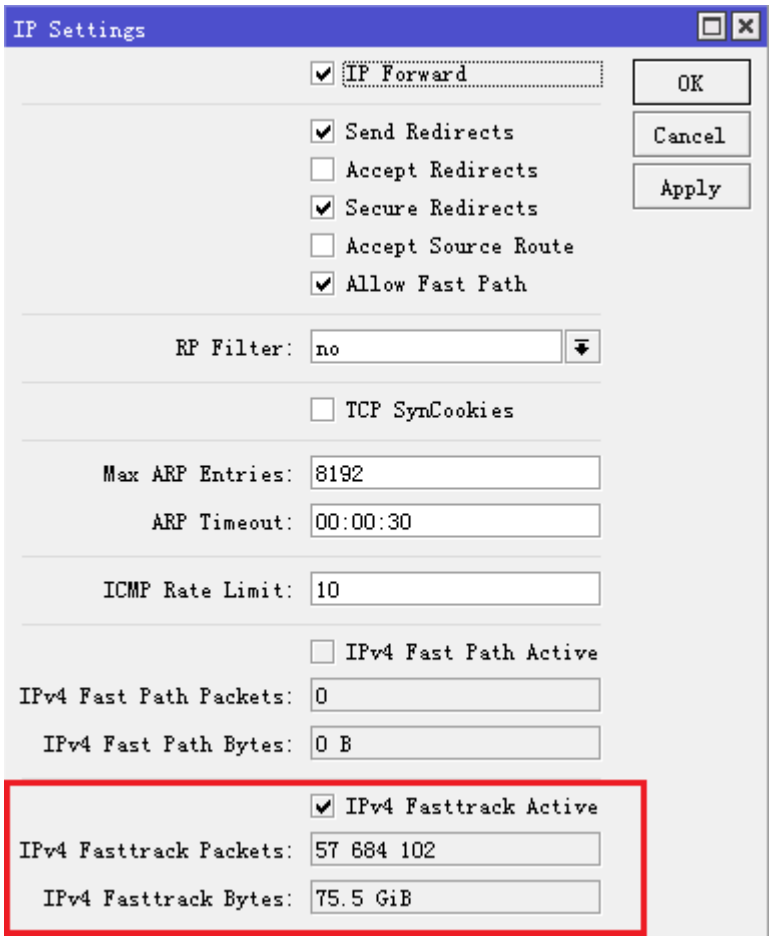
启用 FastTrack，进入 ip firewall filter 配置如下，并将该规则放在规则顶端

```
/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：



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。

第十二章 带宽控制（Queue）

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品质的服务相关。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包上(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)。

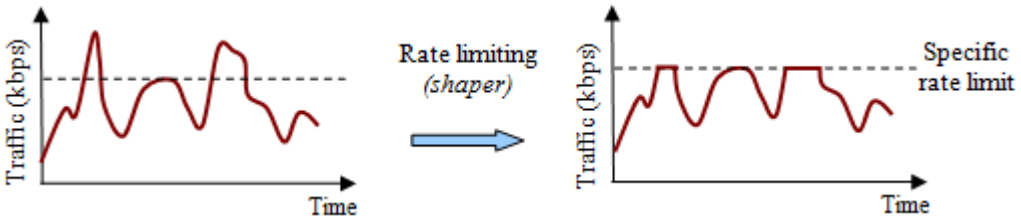
12.1 流控原理

Queue 流控用于对网络接口数据流发送和接收数据进行控制。传输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。

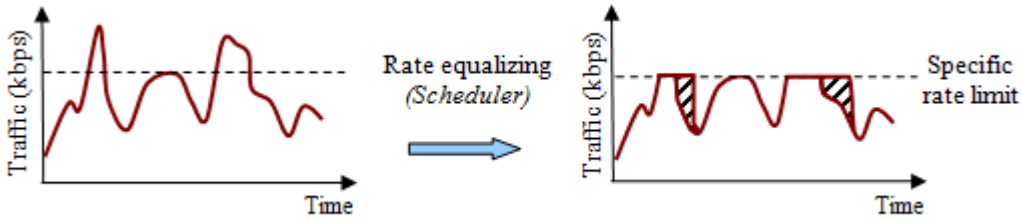
流控执行两种方式：

- 丢弃所有超出的流量限制的数据包– **rate limiting** (丢弃或整形流量)，当 queue-size=0 100%流量被限制
- 延迟发送超出指定流量限制加入到队列中的数据– **rate equalizing** (计划任务)，当 queue-size=无限制（unlimited）100%比例均衡发送

下面的视图让你进一步理解 **rate limiting** 和 **rate equalizing** 的区别：

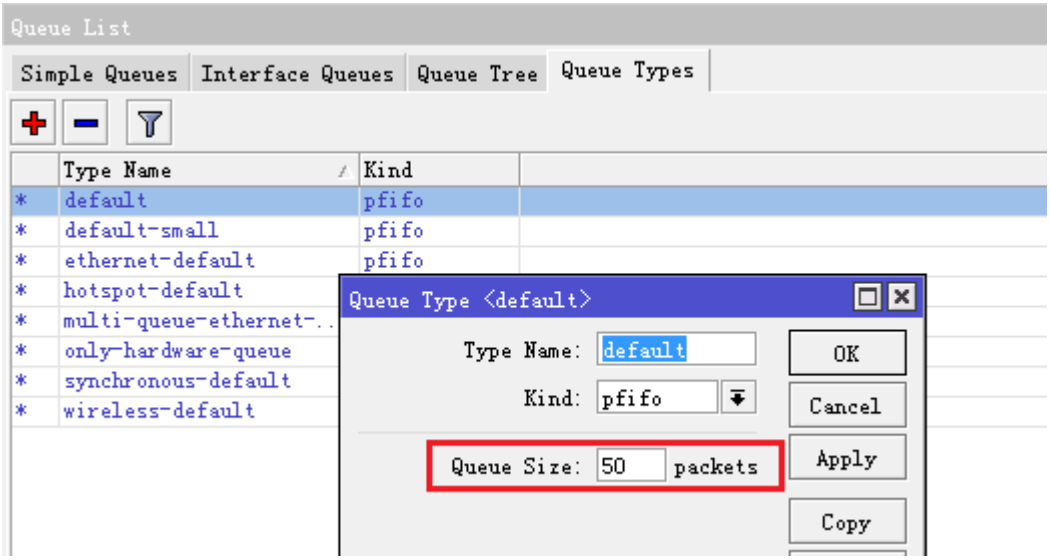


上图显示了所有传输流量超出了指定带宽的那部分被直接丢弃掉。



上图显示了当传输流量超出了指定带宽的那部分，将进入队列容器（**queue-size**）并延迟发送。注意：数据包被延迟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包，数据包同样会被丢弃。

在 RouterOS 队列容器可以通过/queue type 指定，每种类型的 queue type 有不同的队列长度大小，可以指定数据包和字节（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 决定数据包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)。请注意在数据包过滤前与 **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 标记数据包中调用

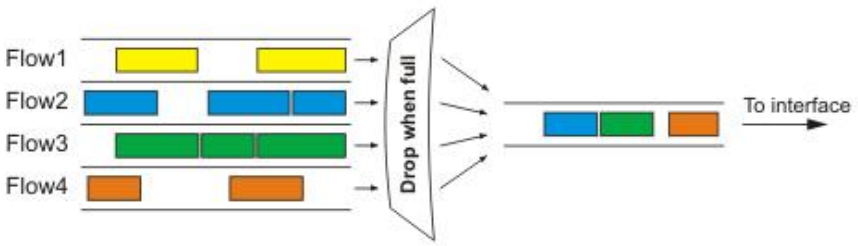
12.2 队列类型（Queue Type）

操作路径: ***/queue type***

在这个子目录你可以创建自己的客户队列类型。之后，将可以在**/queue tree**， **/queue simple** 或 **/queue interface** 使用了

PFIFO 及 BFIFO

这些队列规则是基于先进先出算法的(FIFO：First-In First-Out)。PFIFO 和 BFIFO 的区别在于一个是以数据包为单位衡量的，而另一个是以字节为单位。其中只有一个叫做 **pfifo-limit** (**bfifo-limit**)的参数，它是用来定义一个 FIFO 队列可以容纳多少数据的。每一个不能排队（如果队列满了）的包都要被丢弃，队列长度过大会增加执行时间。

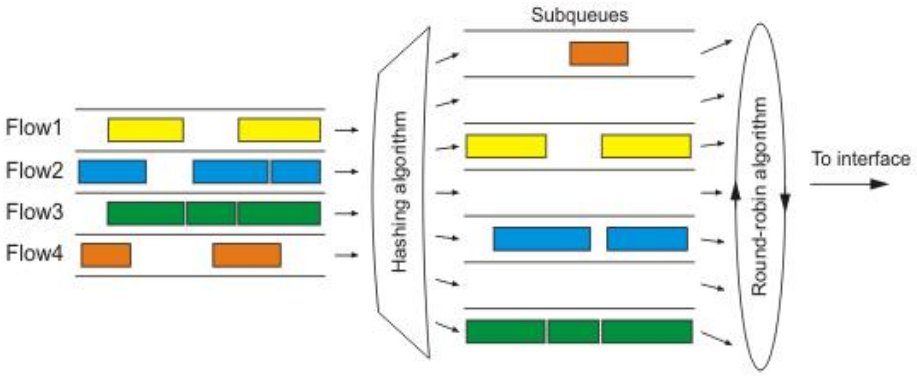


如果你的连接不拥塞的话，建议使用 FIFO 队列规则。

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。

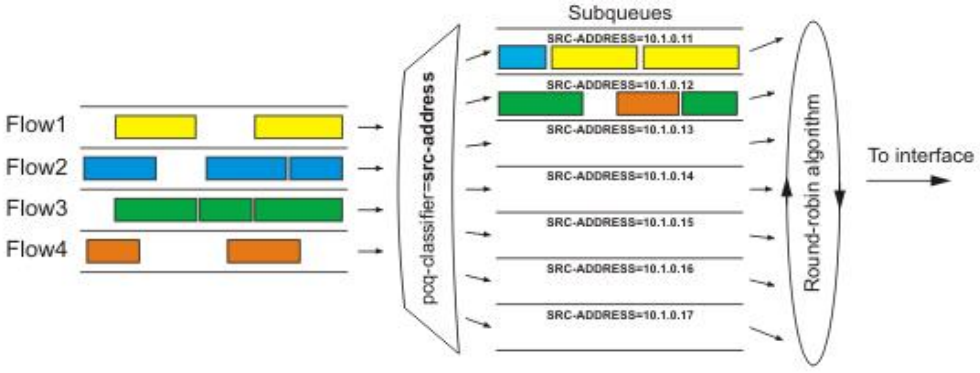


整个 SFQ 队列可以容纳 128 个数据包并且对这些包有 1024 个子队列可用。对拥挤的连接使用 SFQ 可以保证一些连接不至于空等待（starve）。

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。它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** 大小的数据包。PCQ 队列的总大小不能大于 **pcq-total-limit** 包。

以下实例说明了 PCQ 对数据包的用法，以它们的源地址分类。

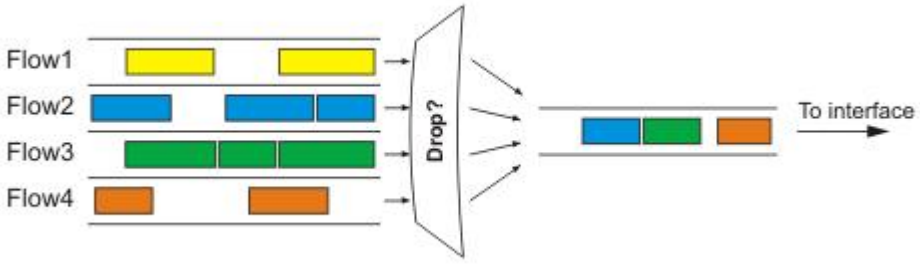


如果你以 **src-address** 对包分类那么所有带有不同源 IP 地址的包将被集合在不同的子队列中。现在你可以使用 **pcq-rate** 参数对每一个子队列进行限制或均衡。或许最重要的部分是决定我们到底应该把这个队列附属到哪个接口上。如果我们把它依附在本地接口上，那么所有来自公网接口的数据流都将以 src-address（很可能这不是我们想要的）地址分组；相反地如果我们把它依附到公共接口，所有来自我们客户的数据都会以 src-address 分组——于是我们可以很容易的限制或者均衡客户的上载。

用 **pcq-classifier** 分类后为了在子队列中均衡速率，设置 **pcq-rate** 为 **0** 几乎不用管理，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** 的数据包的情况。



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

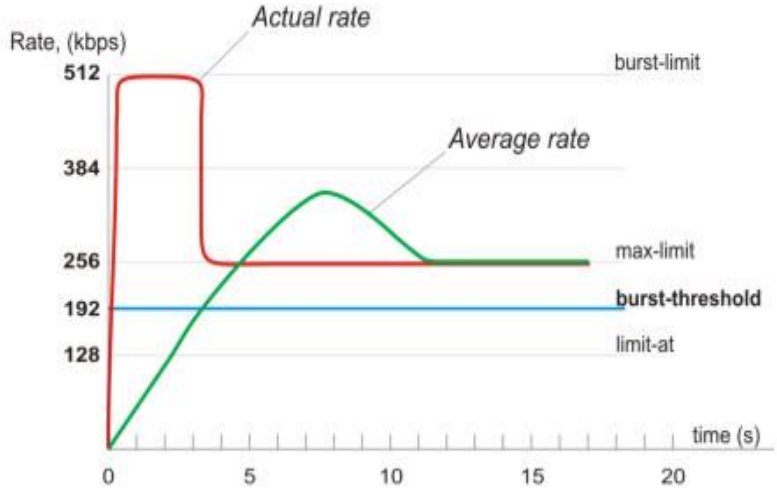
bfifo-limit (整数; 默认: **15000**) - BFIFO 队列可以容纳的最大字节数
kind (bfifo | pcq | pfifo | red | sfq) – 选择队列控制类型
- ***bfifo*** - *字节先进先出*
- ***pcq*** – *每次连接队列*
- ***pfifo*** – *数据包先进先出*
- ***red*** – *随机早先探测*
- ***sfq*** - *随机公平队列*

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。数据包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，数据包标记才有可能
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 下载一个文件，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)。

12.3 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-提么时间内大于或者等于 burst-threshlod

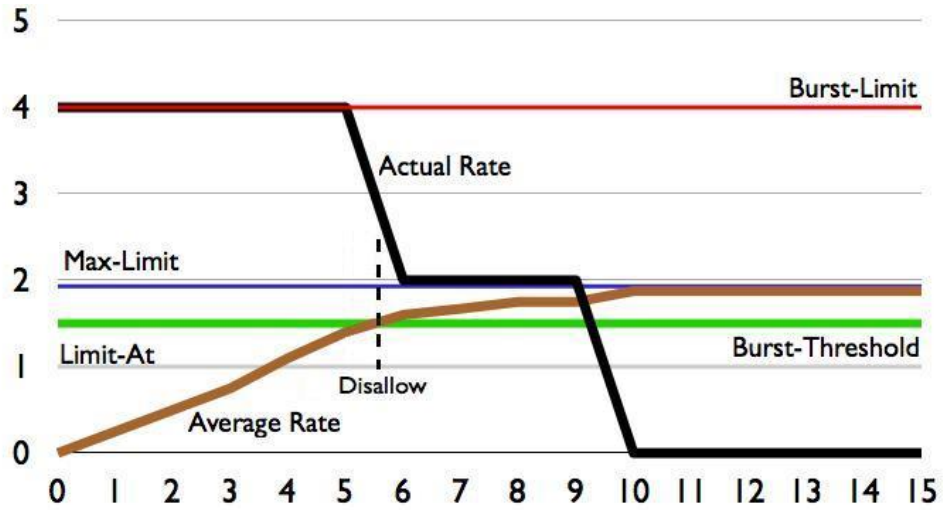
Burst 原理很简单，如果 burst 被允许 max-limit 被 burst-limit 代替，当 burst 被禁止 max-limit 恢复不变

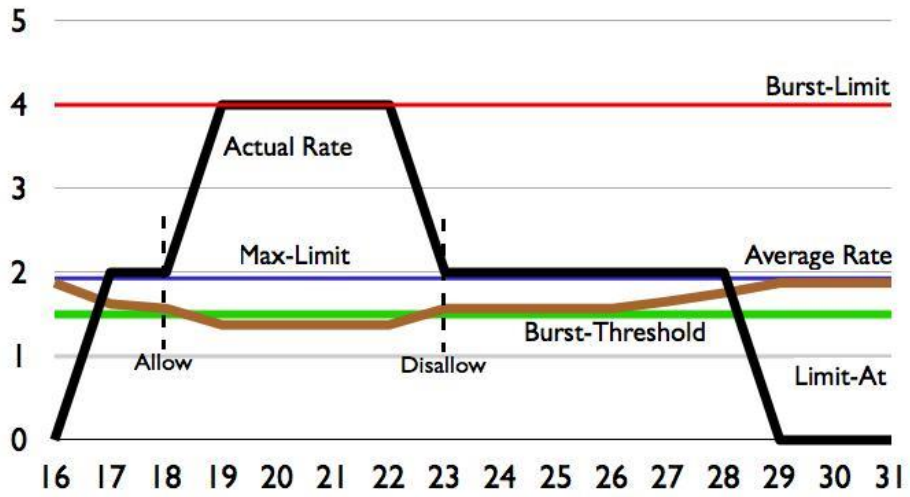
1. **burst-limit** (整型)：能被 burst 允许达到的最大上传和下载数据
2. **burst-time** (时间)：一段时间，单位秒，用于平均速率的计算 (并非实际的 burst 时间长度)
3. **burst-threshold** (整型)：这个参数是通过计算后比对，并开关 burst 功能
4. **average-rate** (隐含只读参数)：路由器计算平均速率根据 **burst-time** 划分为 16 份，每份都会计算出一个平均速率进行比对
5. **actual-rate** (隐含只读参数)：队列的实际传输带宽

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。

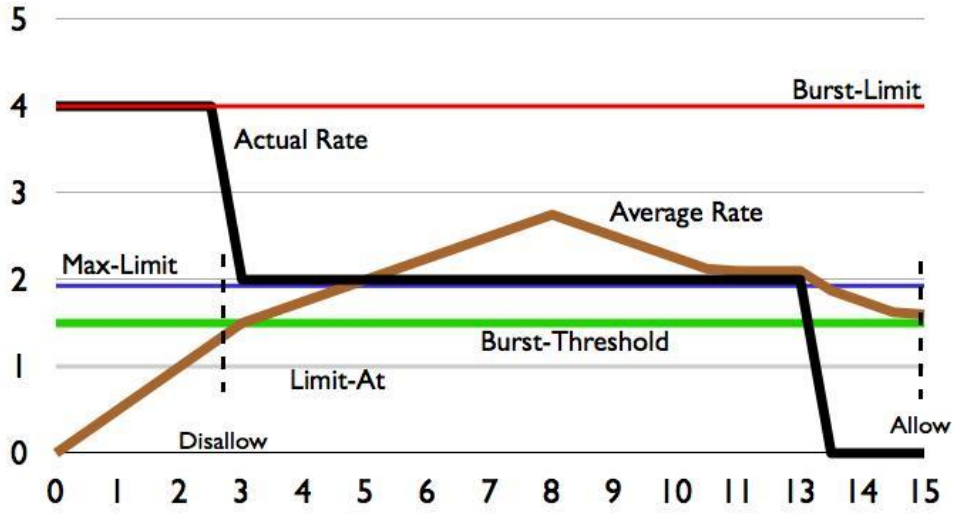
每个 Average rate（平均速率）是根据 burst time 的 1/16，因此这个事例是 1 秒钟 计算一次平均速率

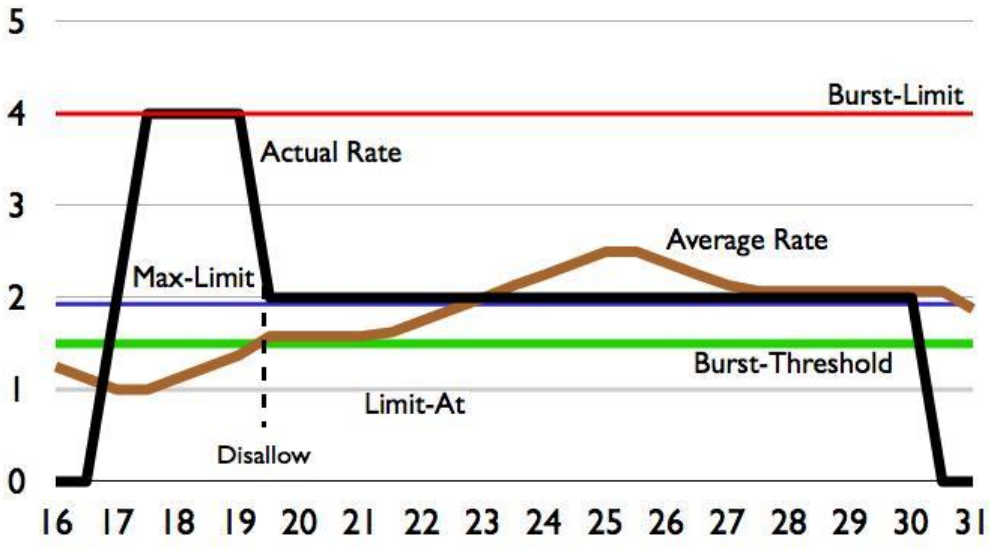
时间	average-rate	burst	实际速率
0	(0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0)/16=0Kbps	average-rate < burst-threshold → Burst 开启	4Mbps
1	(0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+4)/16=250Kbps	average-rate < burst-threshold → Burst 开启	4Mbps

2	(0+0+0+0+0+0+0+0+0+0+0+0+0+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+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+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+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+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+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 → Burst not allowed	2Mbps (1Mb per 0,5sek)

5.5	(0+0+0+0+0+2+2+2+2+2+2+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) /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) /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) /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) /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) /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) /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) /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) /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) /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) /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) /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) /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) /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) /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) /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+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+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+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+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+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+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+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+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+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+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+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+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+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+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+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+1+0)/8=1875Kbps	average-rate > burst-threshold → Burst not allowed	0Mbps (0Mb per 0,5sek)

12.4 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。

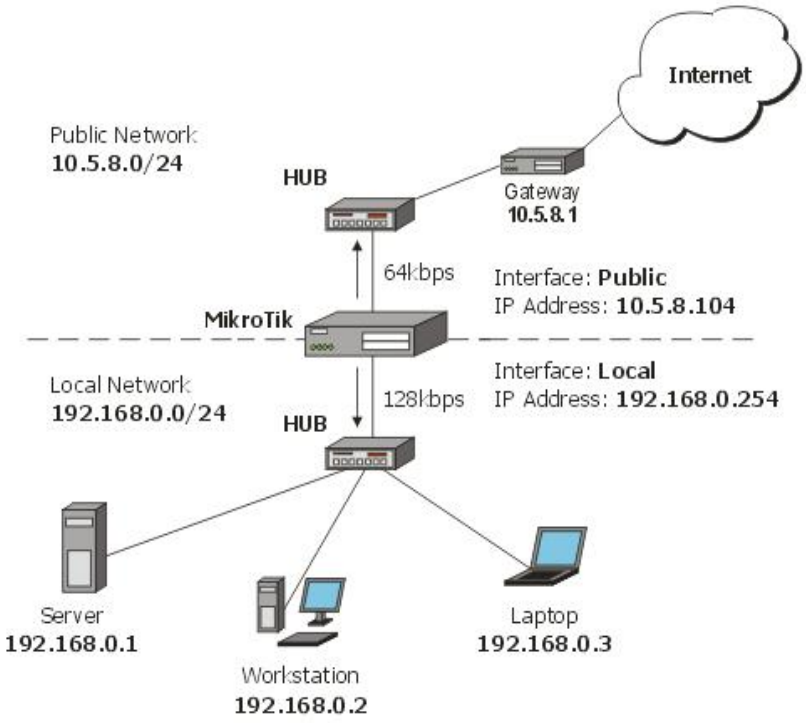
Simple queues 是有序对队列即 FIFO，每个数据包都必须经过每一个队列处理，直到最后一条队列规则，即如果有 1000 条队列，匹配的队列规则是排列在第 1000 条，那么数据包过经过前面 999 条后，才能到达该规则。因此 Simple queue 在出现大量队列规则后，处理效率会降低，再加上 v6 版本前端数据流通过路由器被 queue 重复处理，造成性能消耗更大（参考 12.5 章节）。

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）

应用举例

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
```

我们使用 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，用于 queue simple 中 FIFO 的优先顺序（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）

```
[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>
```

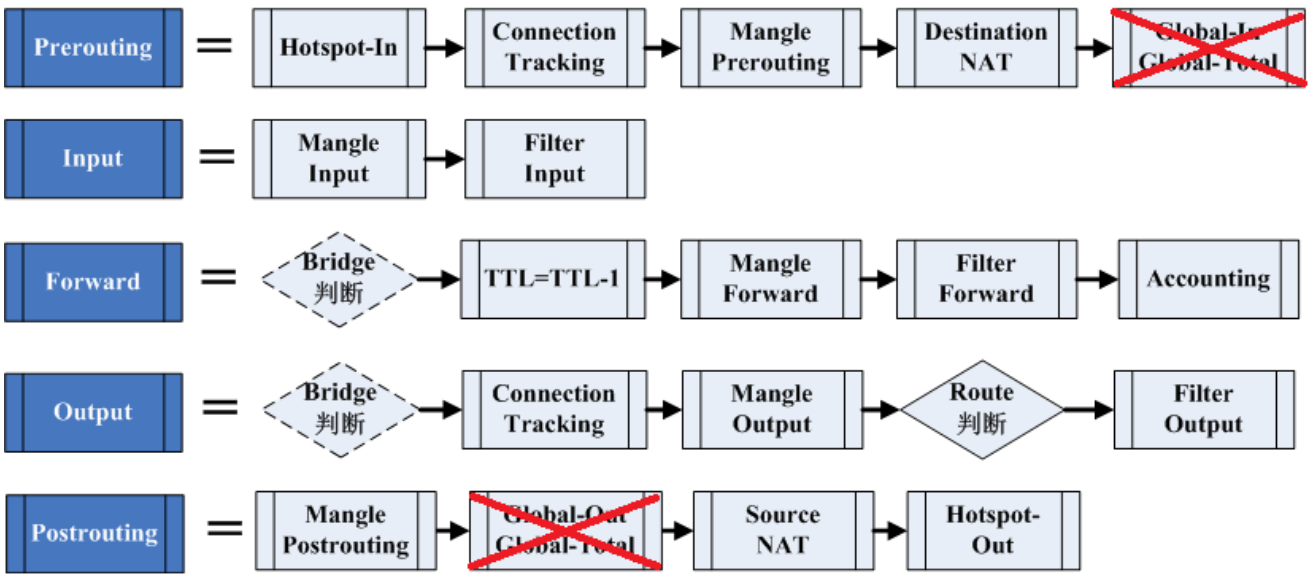
12.5 RouterOS v6.0 Queue 变动

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。

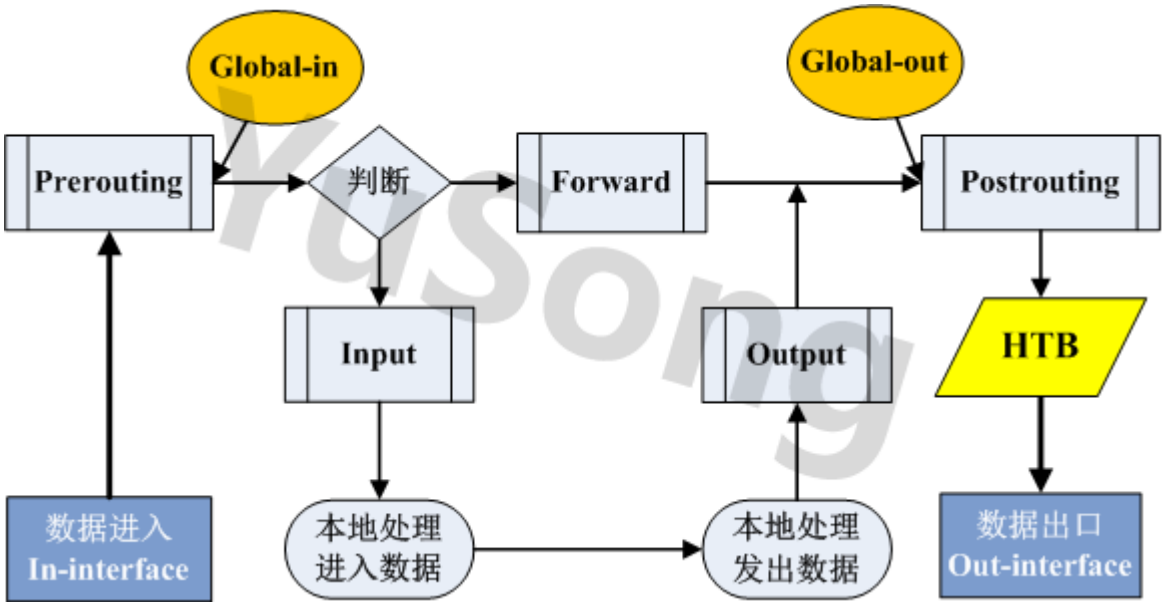
首先我们来看看 IP 数据流，Queue tree 和 simple queue 会出现在 Input 和 Postrouting 两个链表中，且 Queue tree 会首先获得 IP 数据流，处理后再专递给 Simple Queue，但他们之间互相独立没有联系，即两个独立的功能模块。

V6.0 前

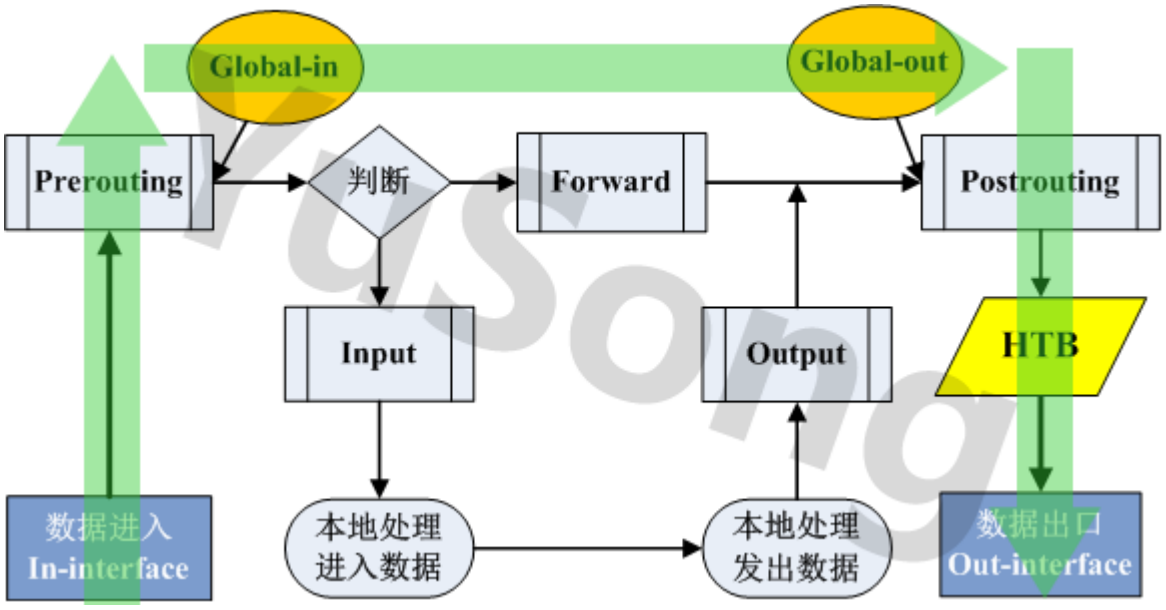
我们看看在 v6.0 之前的链表结构图，可以看到 global-in 属于 prerouting 链表，global-out 属于 postrouting，但 v6.0 重新设计后被取消。



再看看，下面的v6.0前的数据流程图，global-in包含在prerouing，但global-in会处理通过路由（forward）和进入路由器（input）的数据，没有明确队列如何处理进入路由器（input）的数据，global-out会再一次处理通过路由器（forward）的数据，queue队列会重复处理2次通过路由器的数据。

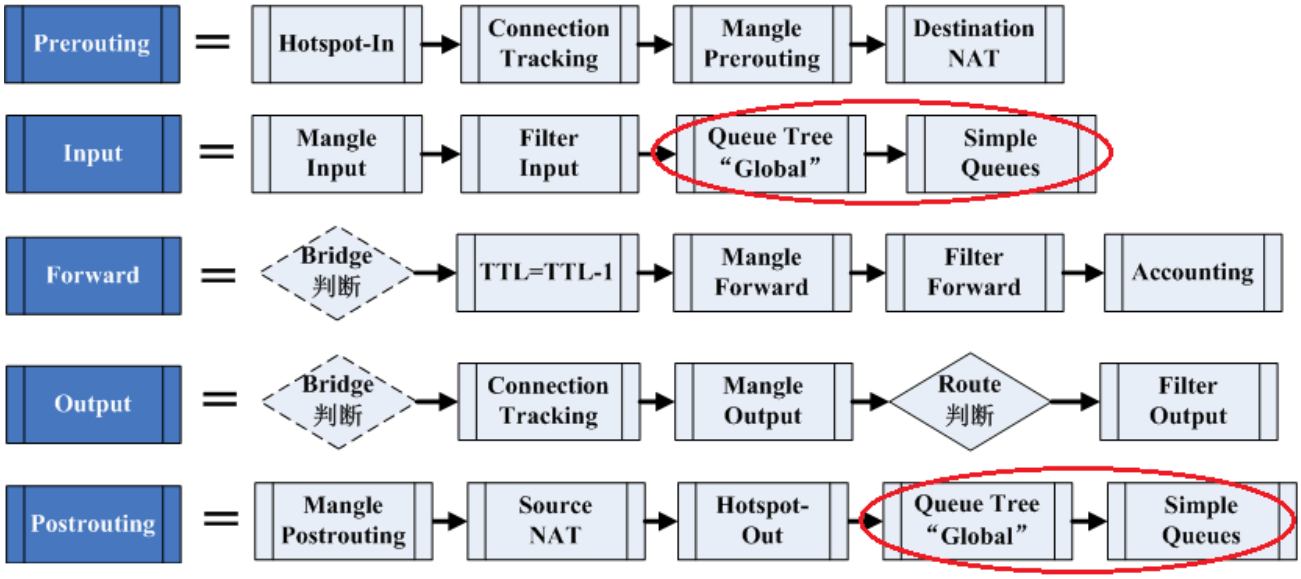


如下图，我们可以看到，当数据流通过路由器的 in-interface 进入后，在从 out-interface 转发出去，会被Global-in 和 Global-out 先后处理两次，造成 CPU 过多的开销。



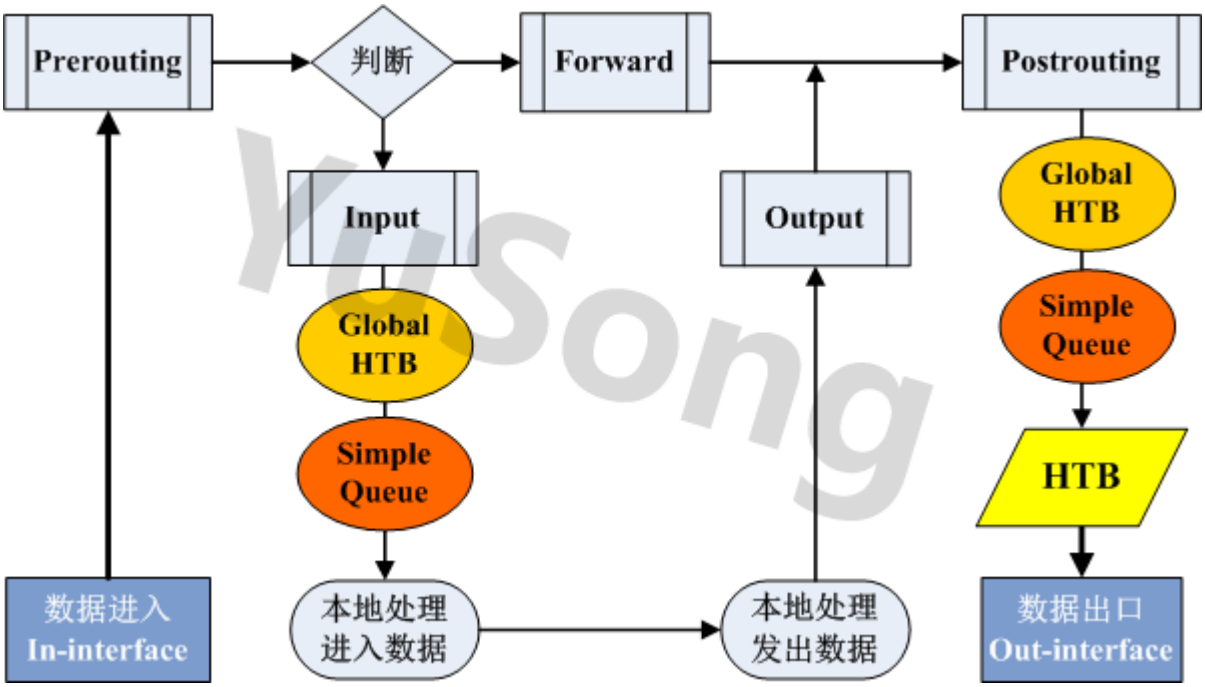
V6.0 后

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：



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，这样能给你建立双重 QoS 策略。

同时如下图所见，通过路由器的流量不会再被 queue 重复处理，queue 明确了进入路由器的数据，数据先进入 queue tree（Global HTB），然后进入 simple queue，这样同一个数据包会 global HTB 和 simple queue 捕获（在 v6.0 版本前他们是共享 HTB）



现在所有通过路由器的 queue 流量，都在 src-nat 之后， PCQ 策略已经得到更新，PCQ 策略在 nat 后更明确，从 connection tracking 匹配。

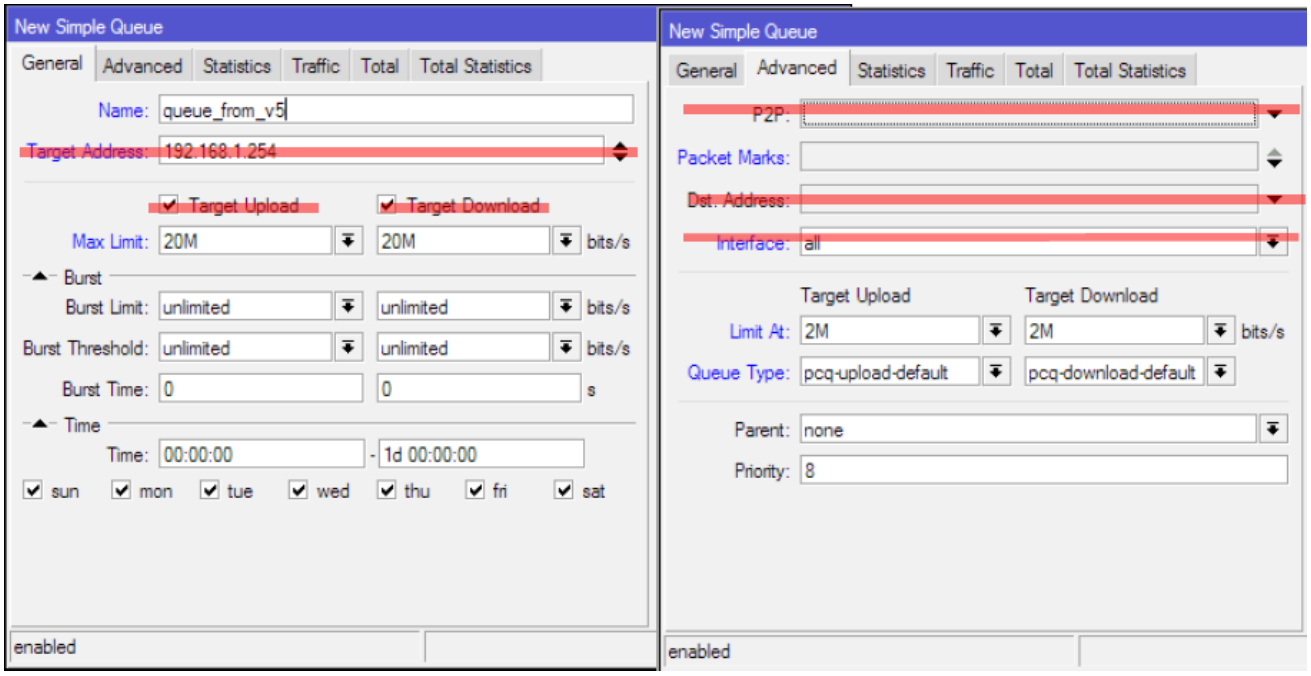
可以总结以下两点：

- **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 结构，父级和子级，这样优先级将有助于分布父级流量，类似于 queue tree，仍然采用 FIFO 顺序执行的优先级处理。**

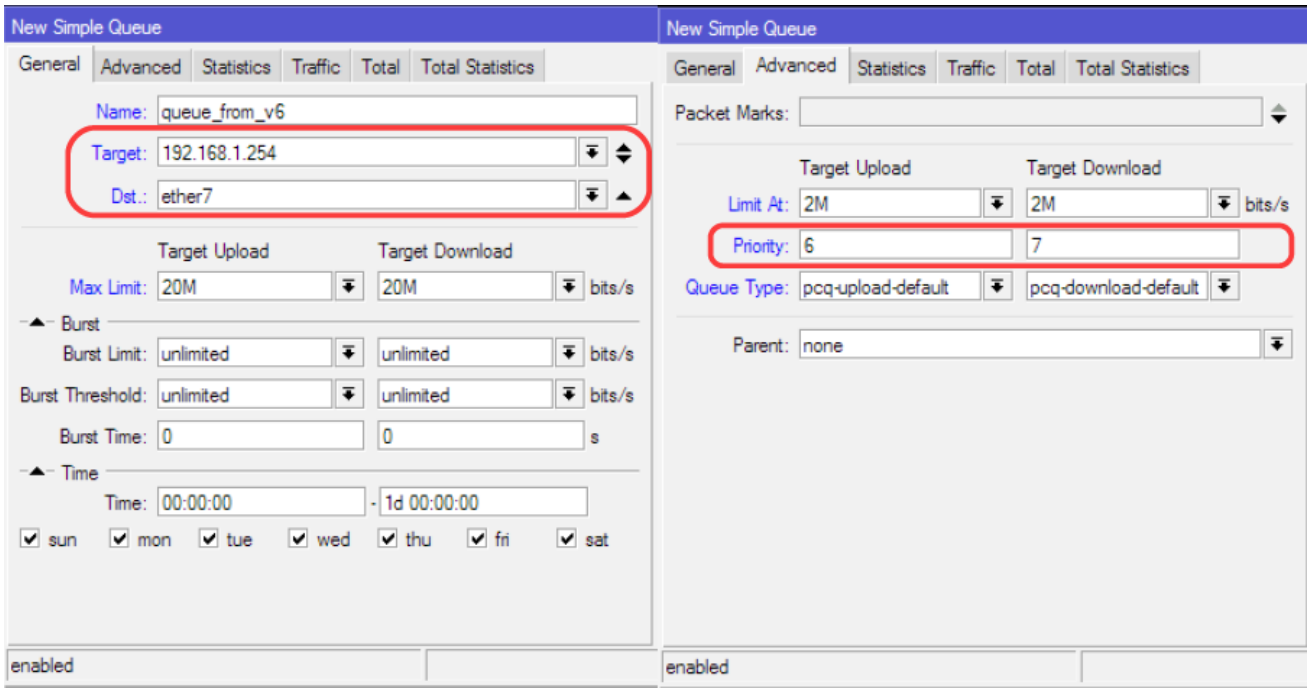
- **虽然 Queue tree 和 simple queue 被分离，但数据流 flow 首先经过 Queue tree 被处理，接下来是 simple queue。**

V6.0 配置变动

下面对比 6.0 之前和 6.0 之后的 simple queue 配置界面，下面是 v5.0 的界面与调整内容：



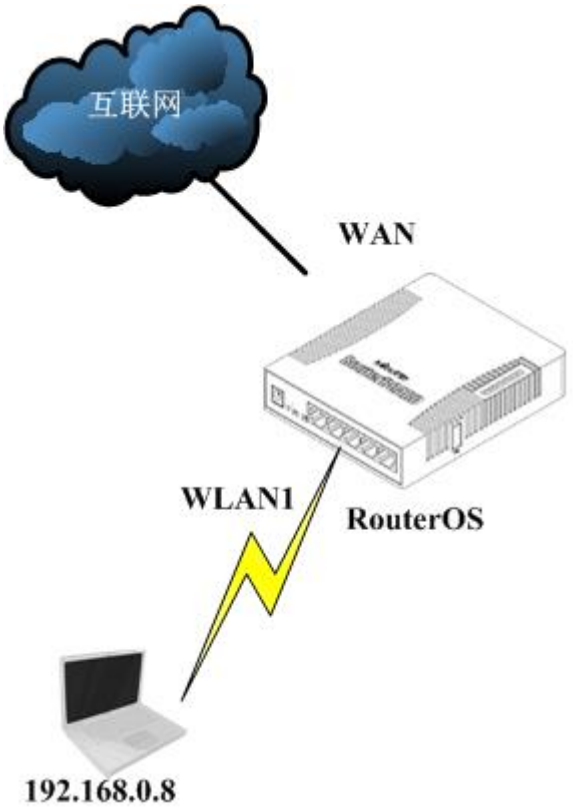
RouterOS 6.0 版本的 simple queue 界面，General 界面 6.0 把 dst-address 从 advanced 栏放到了 general 下(dst)，原有 interface 属性合并到 target 里



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包，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
```

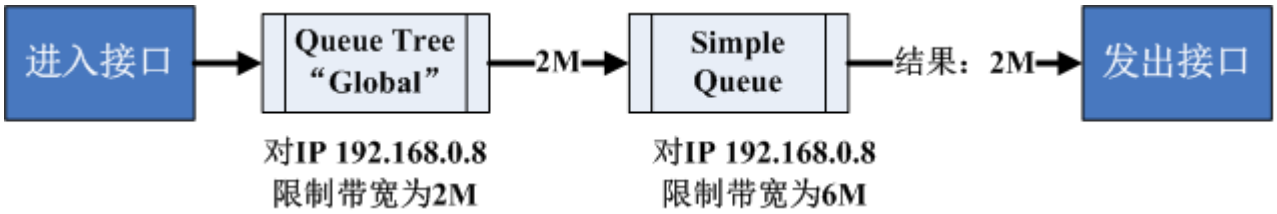
然后进入 queue tree 中添加一条流控规则，parent=wlan1，下载数据流向 wlan1 网卡，并设置为 2M

```
/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

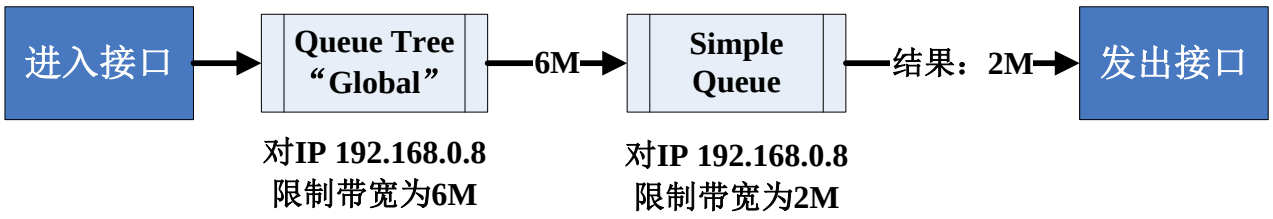
我们将两个规则的带宽调整下，即 Queue Tree 设置 6M

```
/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=6M/6M 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。

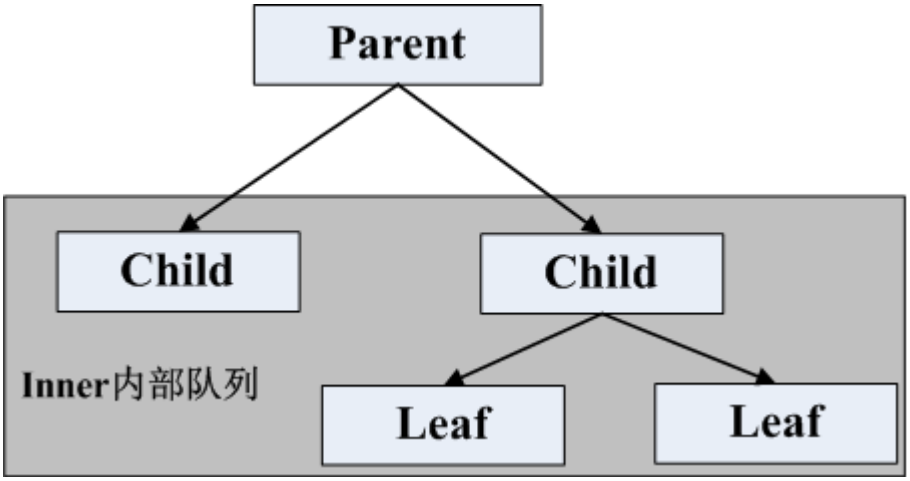
12.6 HTB 等级令牌桶介绍

HTB (Hierarchical Token Bucket)算法的流量管理功能，可有效提高带宽利用率和限制各种网络流量等。对于正常上网的内网主机，系统将允许它偶然突破最大限速；相反，对于长期下载的内网主机，系统将会减小它的带宽，使其对其他主机的影响降到最低。

支持根据 IP 地址、协议、端口等信息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。

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

每个队列在 HTB 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) ，那就是为什么，确保最佳的使用双重限制功能，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)

如果父级与主父级可以设置为 CIR(parent)=MIR(parent)

- 任何子级的最大速率必须小于或者等于父级的最大速率

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:

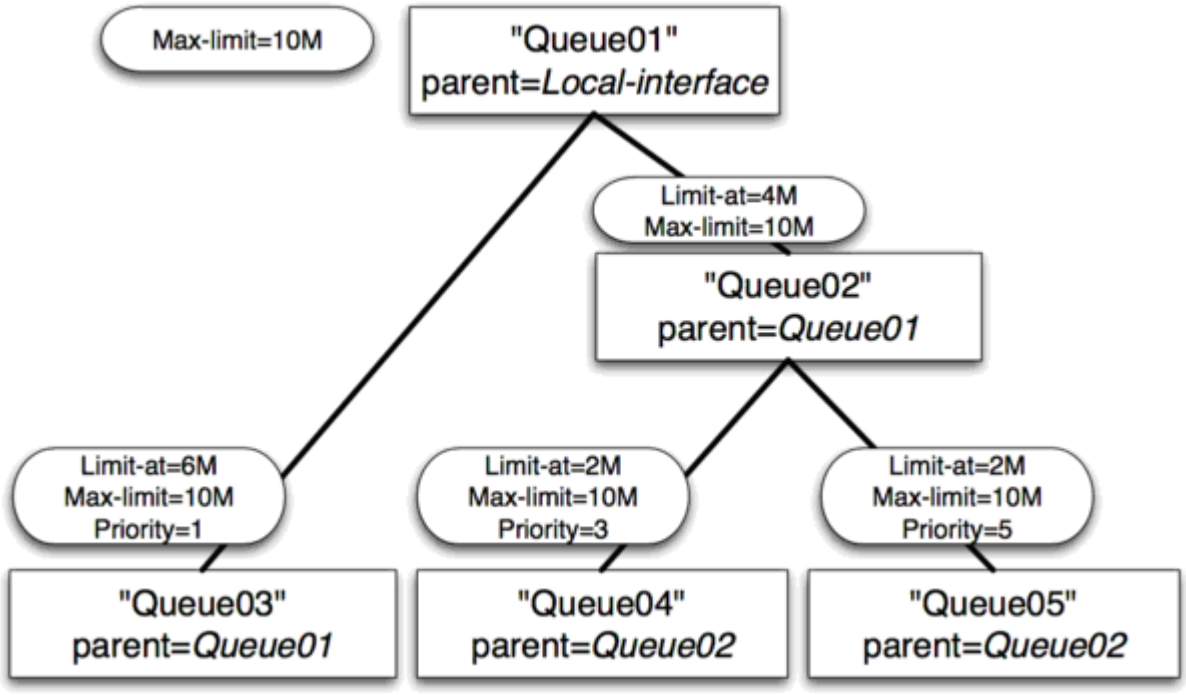
- 对于 **leaf** 叶队列的优先级对于自己 **inner**（内部队列）没有任何意义，即 **inner** 内部队列与其所属的 **leaf**（叶队列）的优先级是不可比较。
- 如果 **max-limit** 被设定（非 0）

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

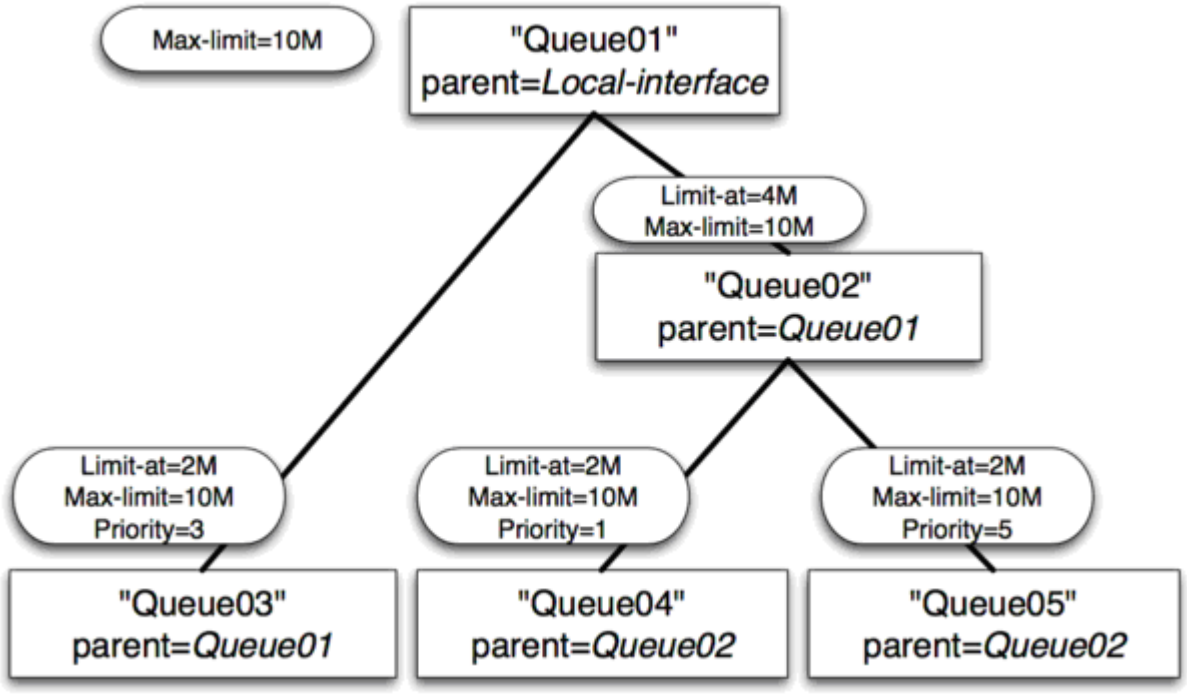


- **Queue01 limit-at=0Mbps max-limit=10Mbps**
- **Queue02 limit-at=4Mbps max-limit=10Mbps**
- **Queue03 limit-at=6Mbps max-limit=10Mbps priority=1**
- **Queue04 limit-at=2Mbps max-limit=10Mbps priority=3**
- **Queue05 limit-at=2Mbps max-limit=10Mbps priority=5**

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

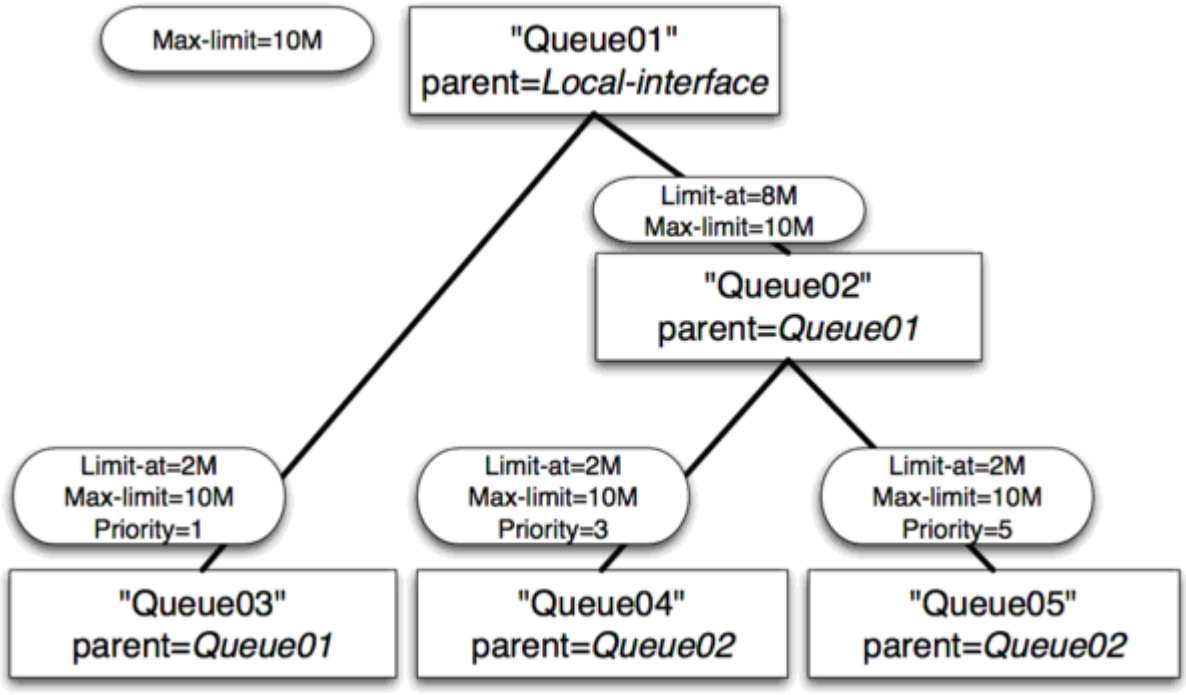


- **Queue01 limit-at=0Mbps max-limit=10Mbps**
- **Queue02 limit-at=4Mbps max-limit=10Mbps**
- **Queue03 limit-at=2Mbps max-limit=10Mbps priority=3**
- **Queue04 limit-at=2Mbps max-limit=10Mbps priority=1**
- **Queue05 limit-at=2Mbps max-limit=10Mbps priority=5**

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



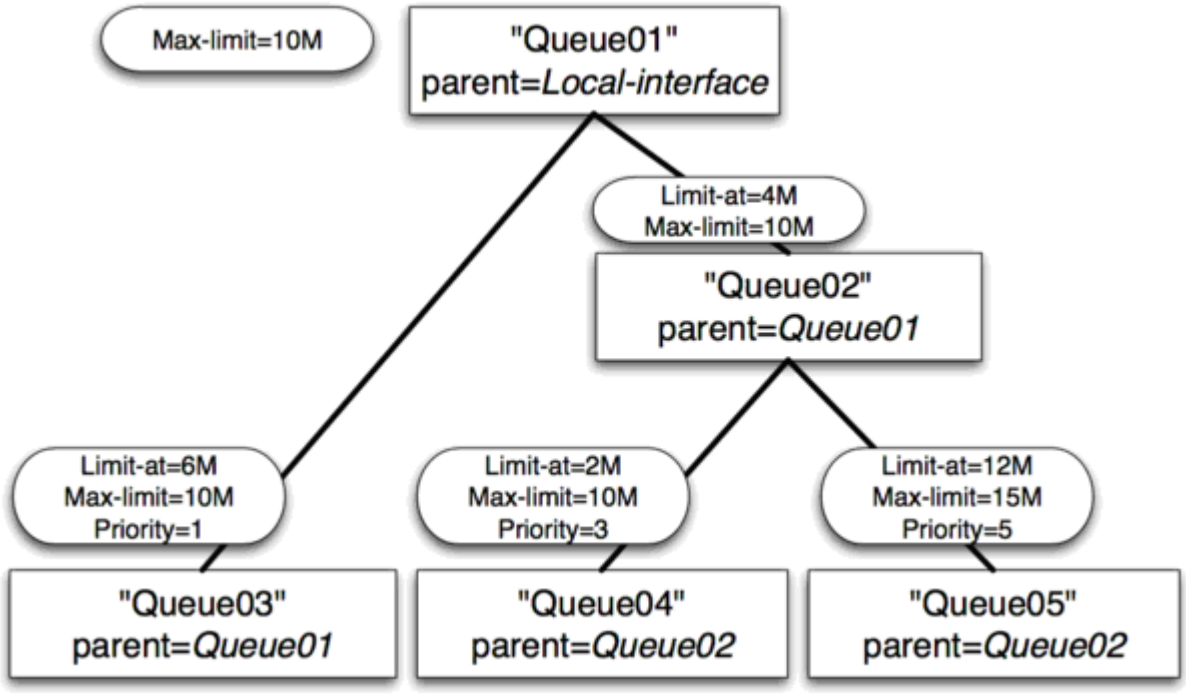
- **Queue01 limit-at=0Mbps max-limit=10Mbps**
- **Queue02 limit-at=8Mbps max-limit=10Mbps**
- **Queue03 limit-at=2Mbps max-limit=10Mbps priority=1**
- **Queue04 limit-at=2Mbps max-limit=10Mbps priority=3**
- **Queue05 limit-at=2Mbps max-limit=10Mbps priority=5**

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

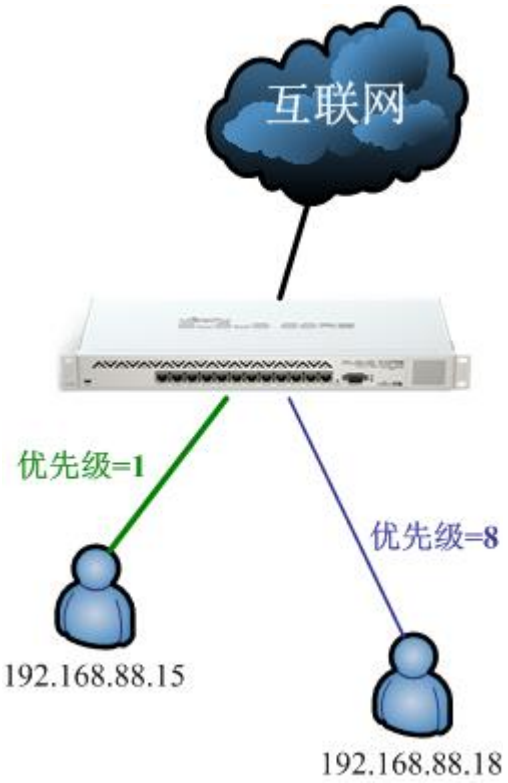
当添加一个简单队列时，将产生 3 个 HTB 类(in global-in， global-total and global-out)，但在接口队列中不添加任何类。

当数据包通过路由器时，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，如果数据包是从路由器发出的，它们将穿过 global-total，global-out 及 interface 队列。

12.7 v6.0 Simple Queue 等级优先

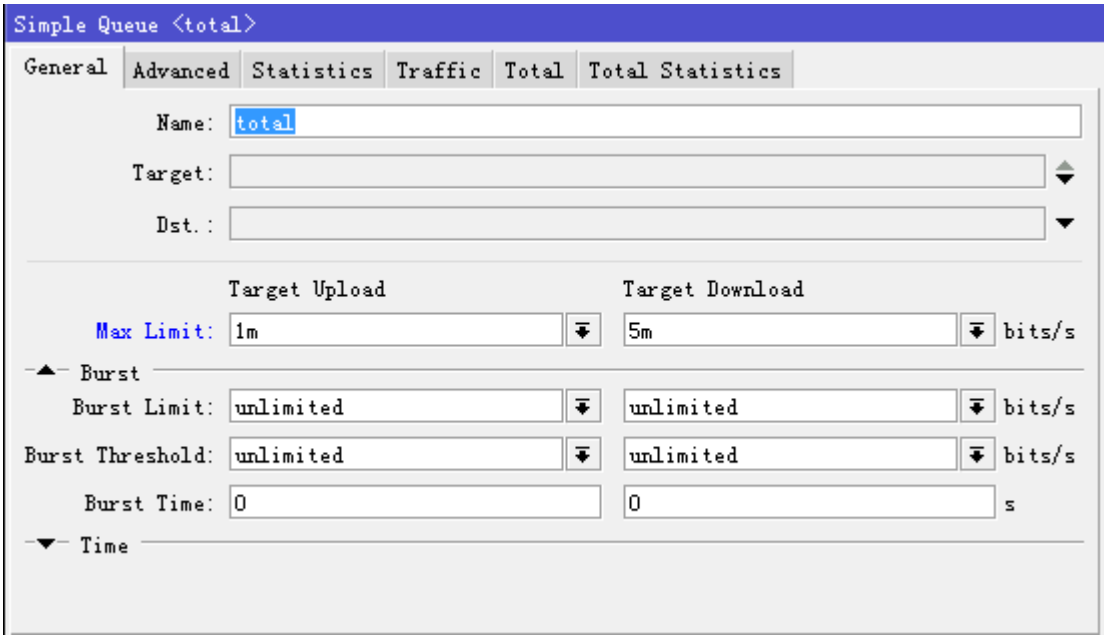
下面是一个 queue simple 下的等级优先流控规则实例，出口带宽是 5M，我们需要为内网主机设置优先级流控，如果在过去我们需要使用到 queue tree 的 HTB 功能，即需要通过 mangle 标记 ip 数据包进行操作，而现在我们可以不通过 mangle 来完成，直接在 simple queue 中操作。

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。



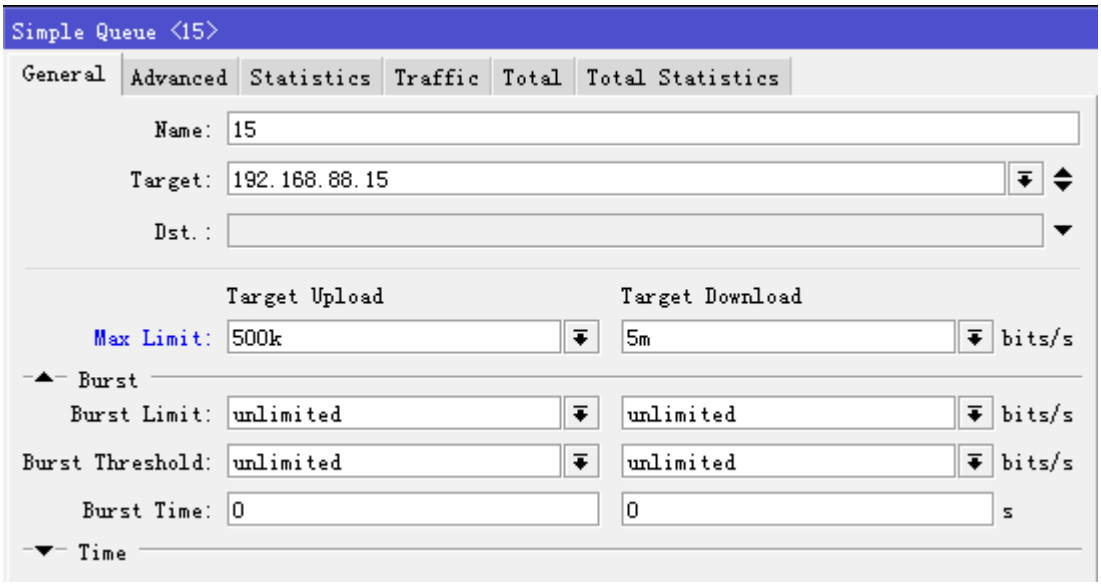
注意：simple queue 的等级优先规则同样需要按照 HTB 的原则进行配置

首先我们进入 simple queue 中，添加一条父级总带宽规则 total



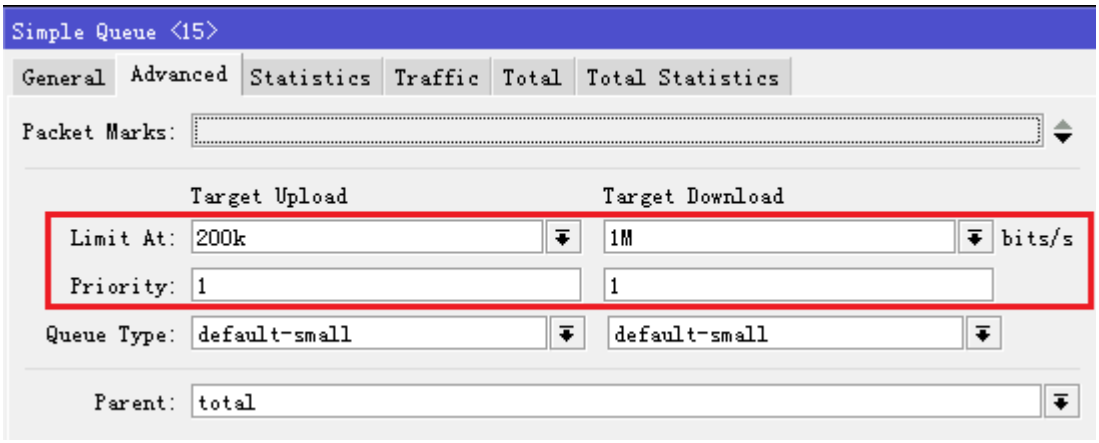
Max-limit 设置整个网络上行带宽为 1m，下行带宽为 5m

配置第二条规则取名 15，定义主机 192.168.88.15 的带宽规则：

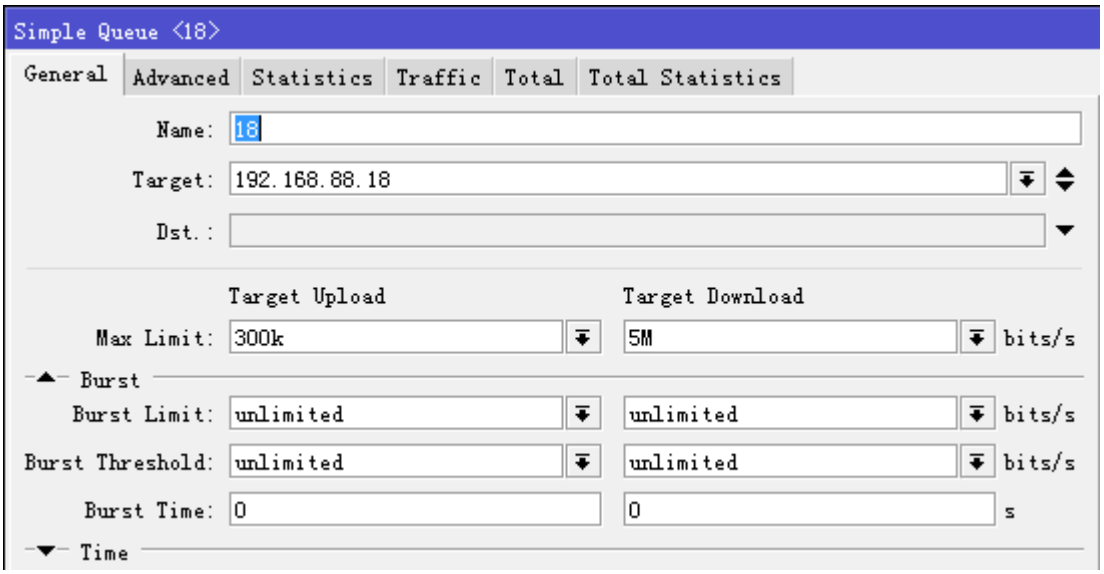


Max-limit 设置目标主机 192.168.88.15 上行带宽为 500k，下行带宽为 5m

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

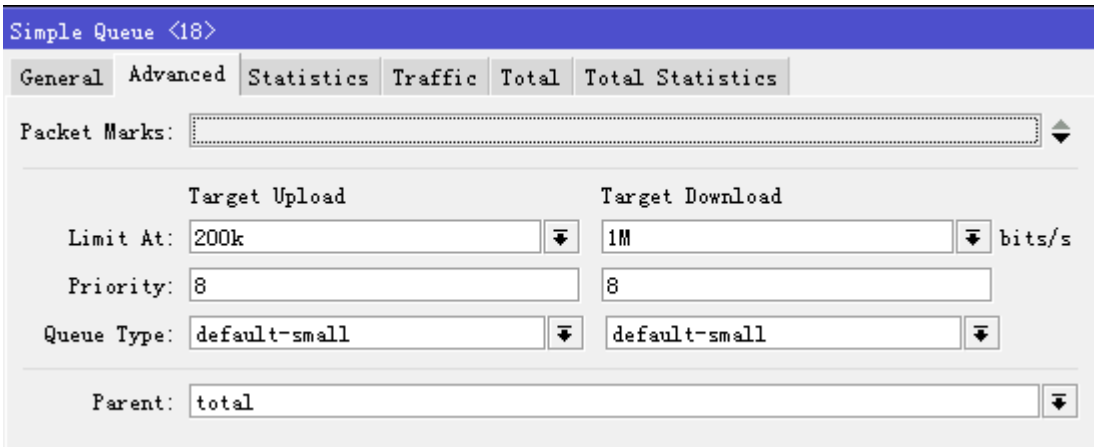


配置第三条规则取名 18，定义主机 192.168.88.18 的带宽规则：

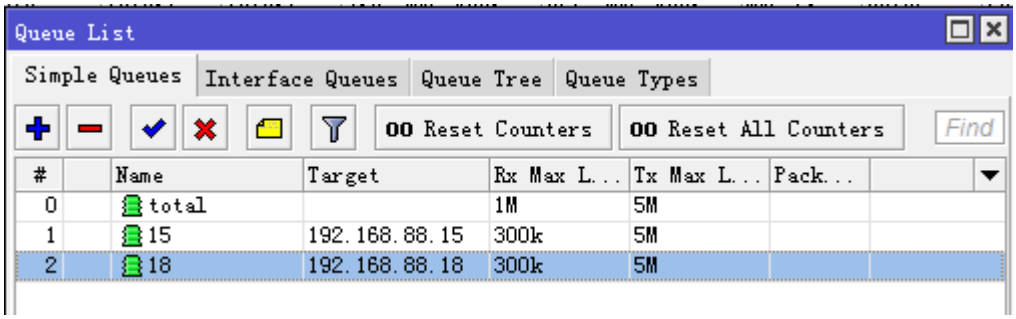


Max-limit 设置目标主机 192.168.88.18 上行带宽为 500k，下行带宽为 5m，与之前没有区别

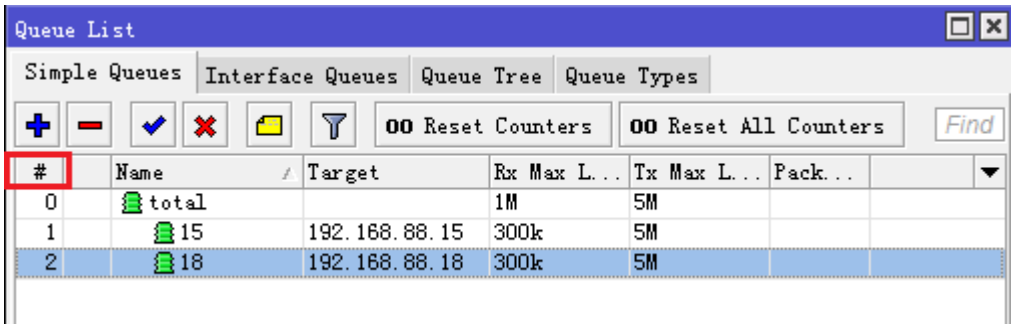
Advanced 栏中 limit-at 也相同，只是 priority 优先级设置为 8 最低，parent 父级是 total



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。

12.8 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** 下标记数据包流进行匹配，建立流控队列。

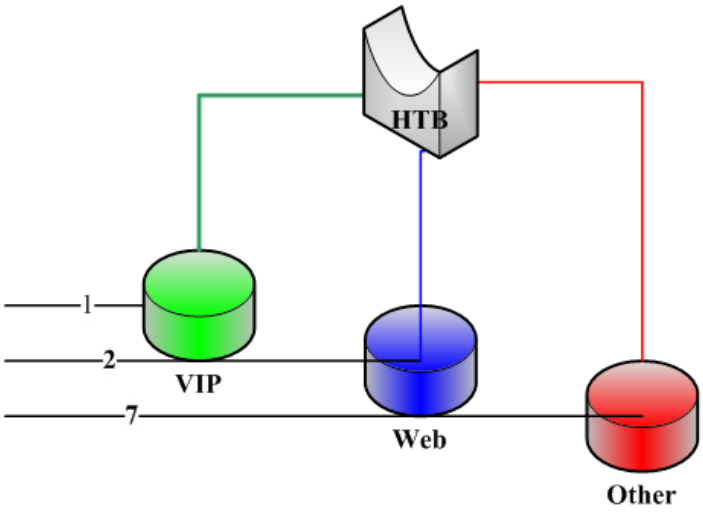
属性描述

burst-limit (*整数*) - 当脉冲串激活时可以达到的最大数据率
burst-threshold (*整数*) - 用于计算是否允许脉冲。如果上一次脉冲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** 下标记的数据包流。 当前队列参数仅应用于用这个数据流标记标识了的数据包。
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包并采用 **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
```

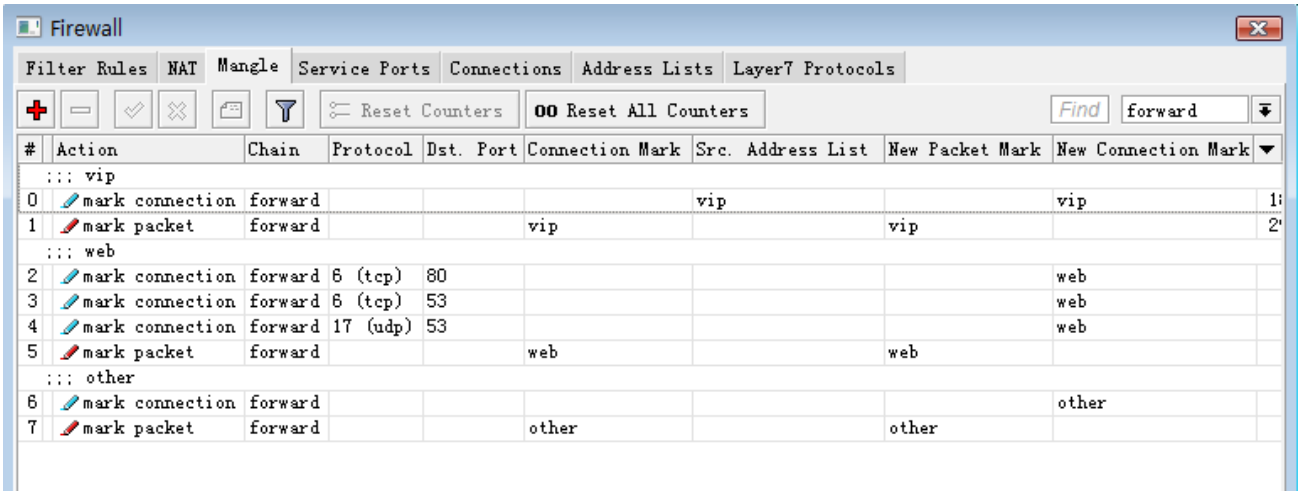
跟着定义 web 数据，这里我们需要针对访问网页的 tcp/80 端口和域名解析的 DNS 端口 tcp/53 和 udp/53 端口标记：

```
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 的数据包，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	

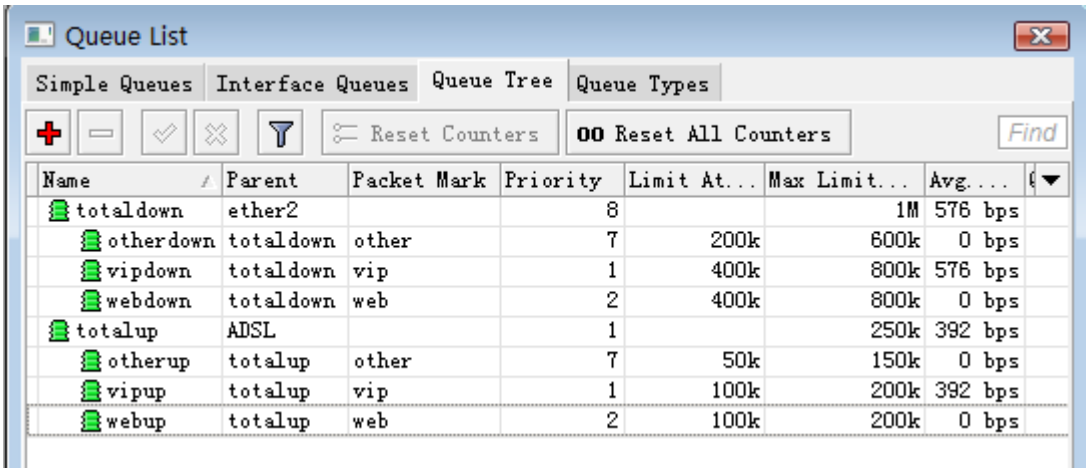
标记数据完成后，我们进入 queue tree 中，对数据进行优先级的配置，ADSL 总带宽为 1Mbps 下行，250kps 的上行，给三类数据带宽分配如下

- **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
```

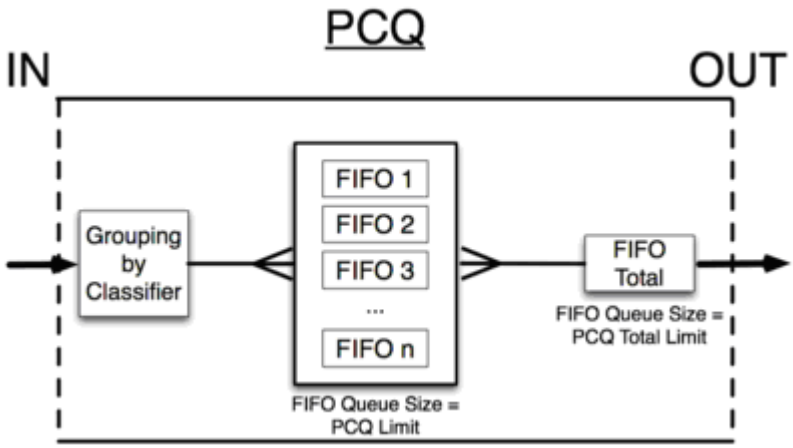


Name	Parent	Packet Mark	Priority	Limit At...	Max Limit...	Avg....
totaldown	ether2		8		1M	576 bps
otherdown	totaldown	other	7	200k	600k	0 bps
vipdown	totaldown	vip	1	400k	800k	576 bps
webdown	totaldown	web	2	400k	800k	0 bps
totalup	ADSL		1		250k	392 bps
otherup	totalup	other	7	50k	150k	0 bps
vipup	totalup	vip	1	100k	200k	392 bps
webup	totalup	web	2	100k	200k	0 bps

12.9 PCQ 配置

PCQ 算法比较简单，首先利用分类器从相应数据流中区分一个子数据流，然后在每一个子数据流上建立独立的 FIFO 队列长度和限制，再归类所有的子数据流在一起，并应用全局 FIFO 队列长度和限制。以下是 PCQ 参数:

- **pcq-classifier** (dst-address | dst-port | src-address | src-port; 默认: "")：选择子数据流分类类型。
- **pcq-rate (数字)：**每个子数据流可获得的最大数据带宽。
- **pcq-limit (数字)：**在数据包中一个子数据流的队列长度
- **pcq-total-limit (数字)：**全局 FIFO 队列的队列长度

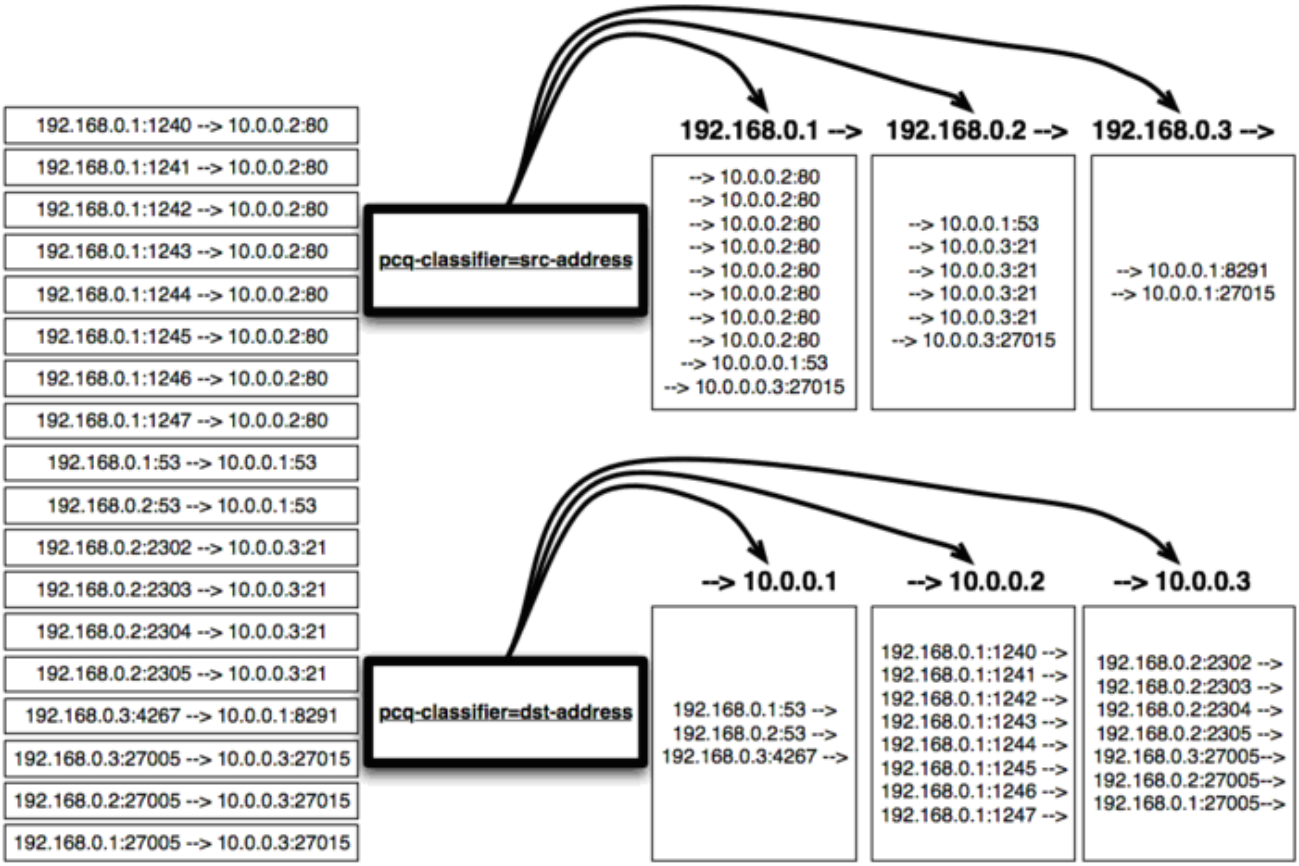


因此，当有 100 个队列需要限制 1000kbps 下载时，我们可以使用 1 个 PCQ 队列和该 PCQ 队列包含的 100 子数据流队列。

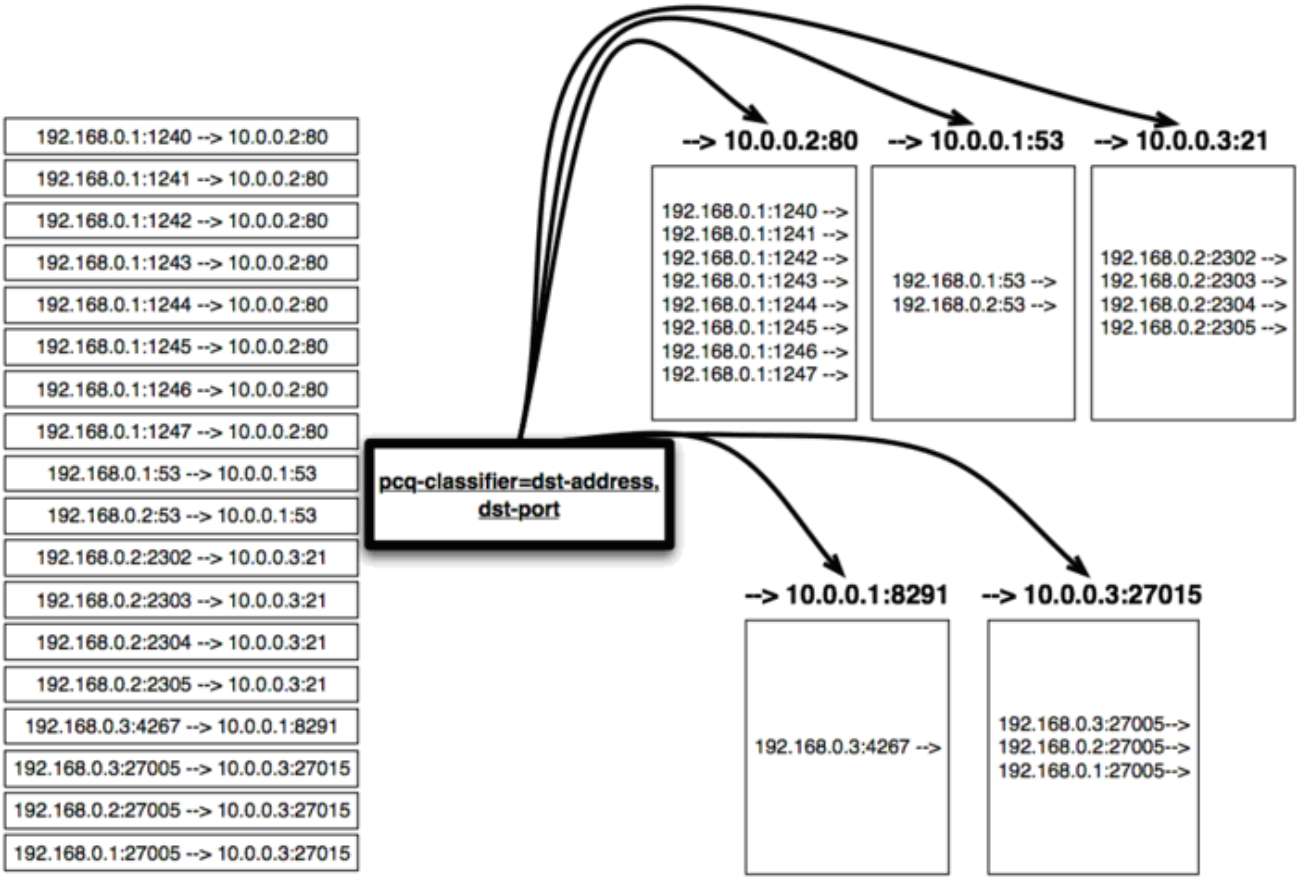
分类器

为更好的理解分类器，我用一组 IP 地址和端口到对应的地址和端口数据流的实例，这时我们将选择一种分类器，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。

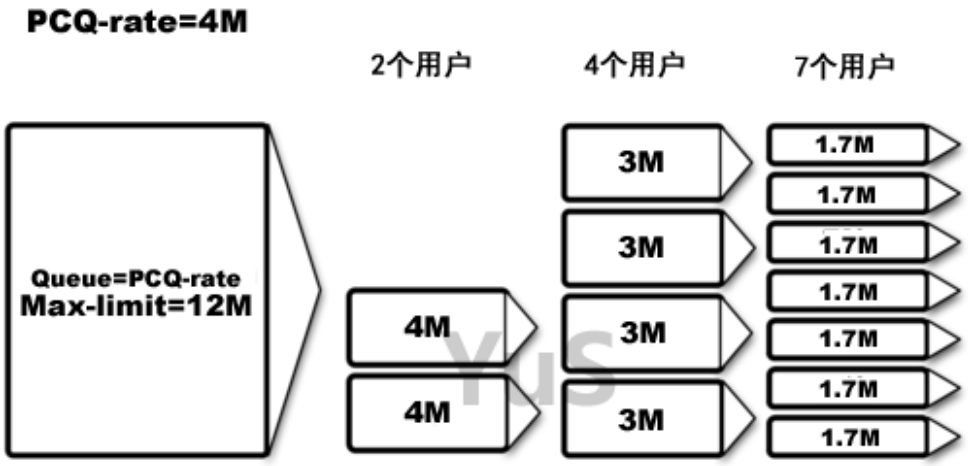
PCQ 的目标和源地址分类原理图：

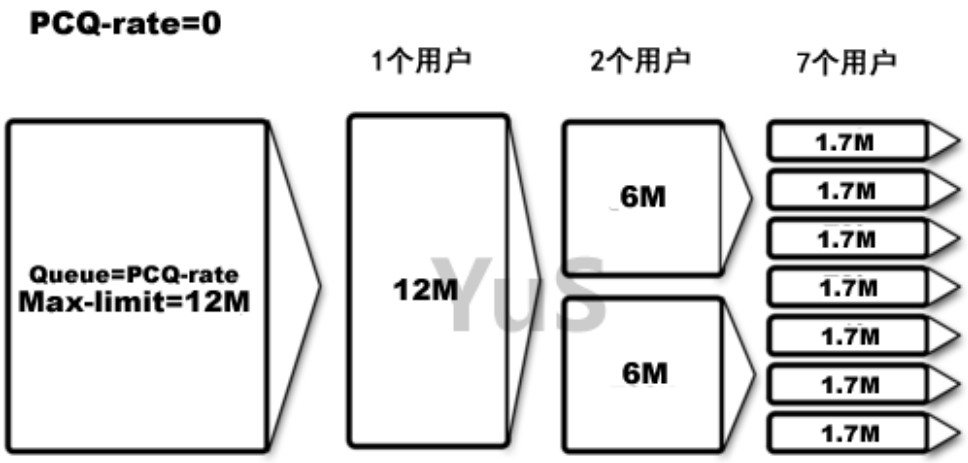


端口分类



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的时候，会造成在上网空闲时候带宽的浪费，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：



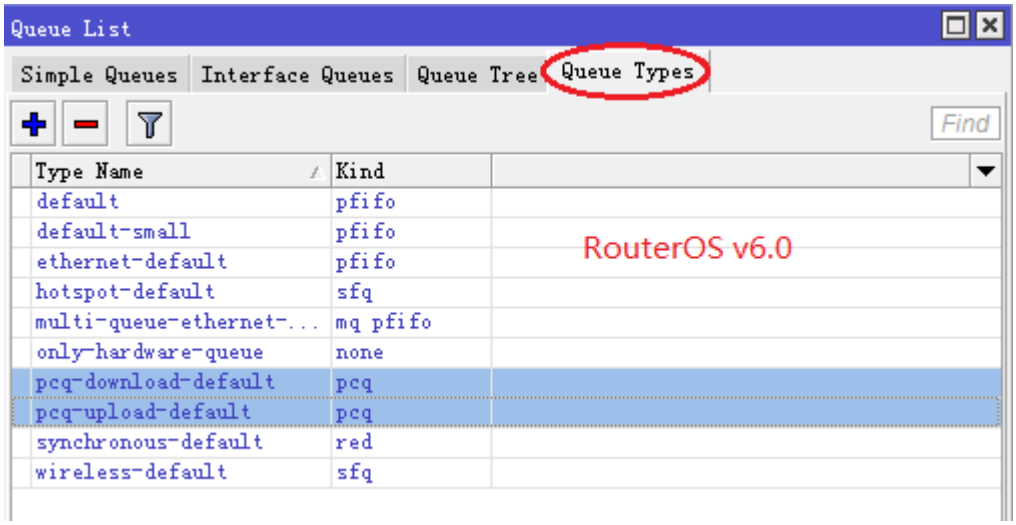


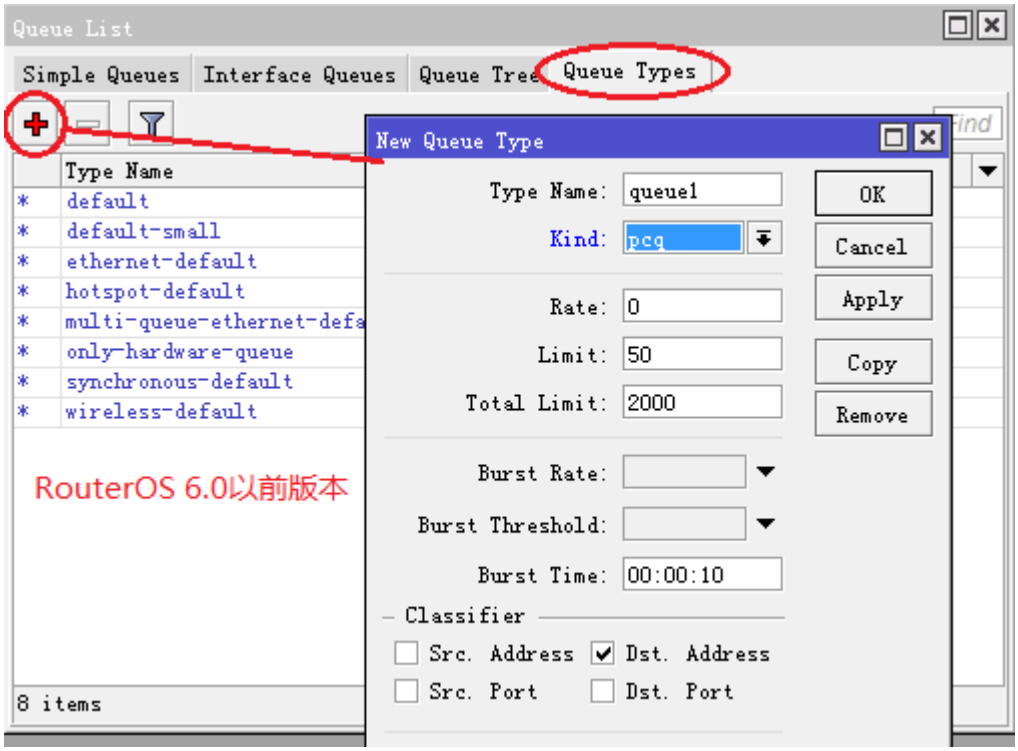
通过上图，我们可以看到当 PCQ 的速率设定为 128k 的时候，平均每个用户将会得到同样的带宽 128k，当上网高峰期的时候 PCQ 才会做二次流量分配，如果 PCQ 的速率在开始就设定为 0k，这样在一个用户的时候就可以得到全部带宽，之后是 2 个用户平均分配，依次类推，但最后带宽会控制在 73k 的范围内，控制最小使用带宽，保证用户正常使用。

简单的 PCQ 实例

通过下面的实例来看看 PCQ 如何配置，假设我们有一个网段为 192.168.10.0/24，需要通过 PCQ 流量控制，估计有 100 个用户在线，每个用户上行和下行分别为 512k 和 1m。

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，之前版本需要自己手动添加：

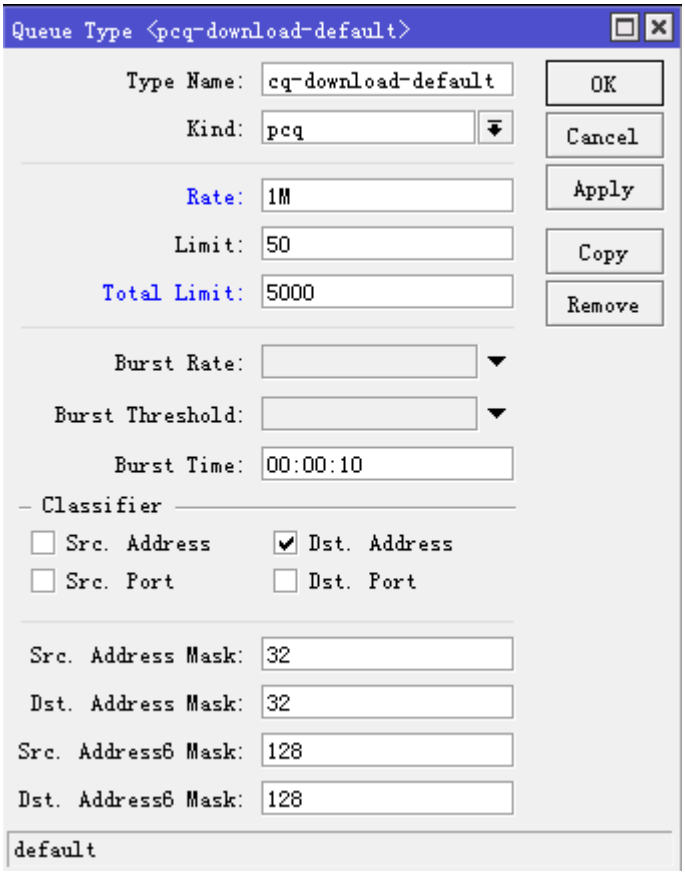




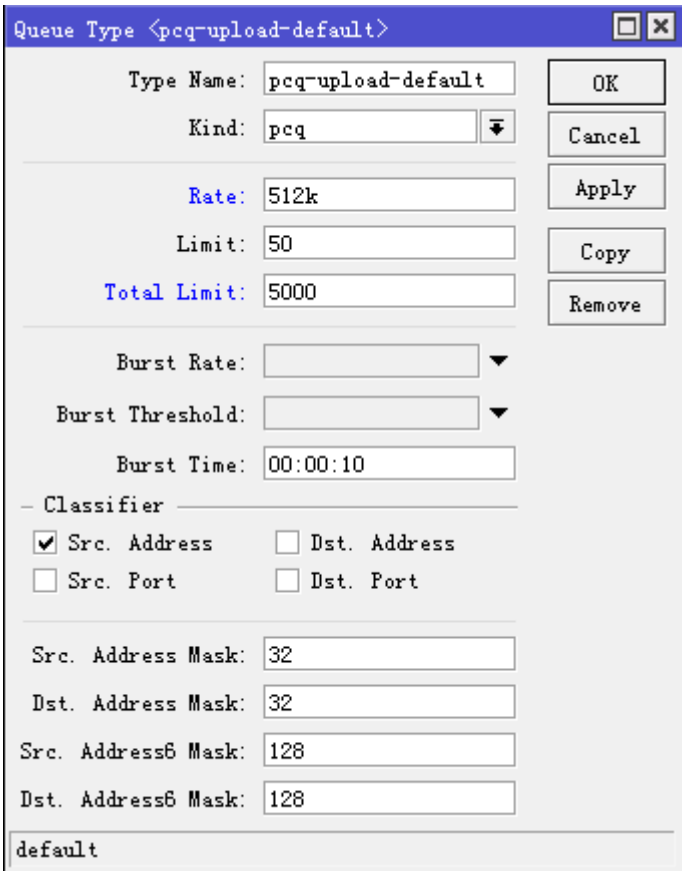
注： Limit 和 Total-Limit 的关系：

- 默认情况下 total-limit 是 2000，该规则仅能容纳 40 个用户，计算方法 total-limit/limit，即 2000/50=40
- 解决方法必须增加 total-limit 或者减少 limit
- 但必须保证每个用户队列(limit)获取 10-20 个数据包

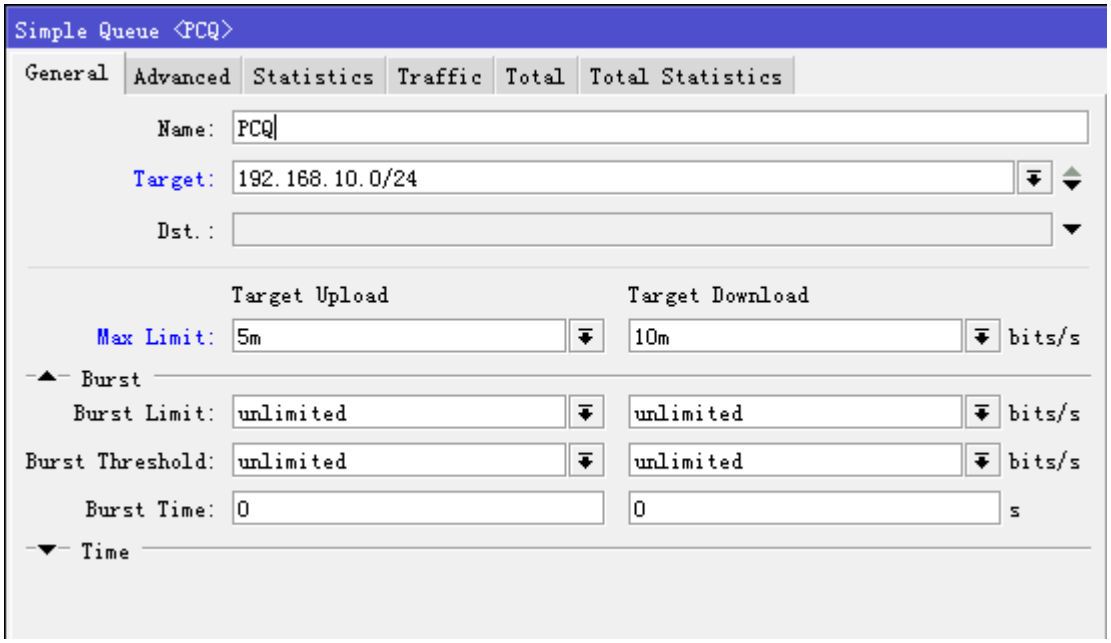
这里我们按照 RouterOS 6.0 版本来做配置，首先进入 Queue Type 中配置 PCQ 的上行和下行分别为 512k 和 1m，设置 pcq-download-default 为 1M，在线用户数 100 人，我们增加 total-limit，因此是 50×100=5000，下行分类为 dst-address



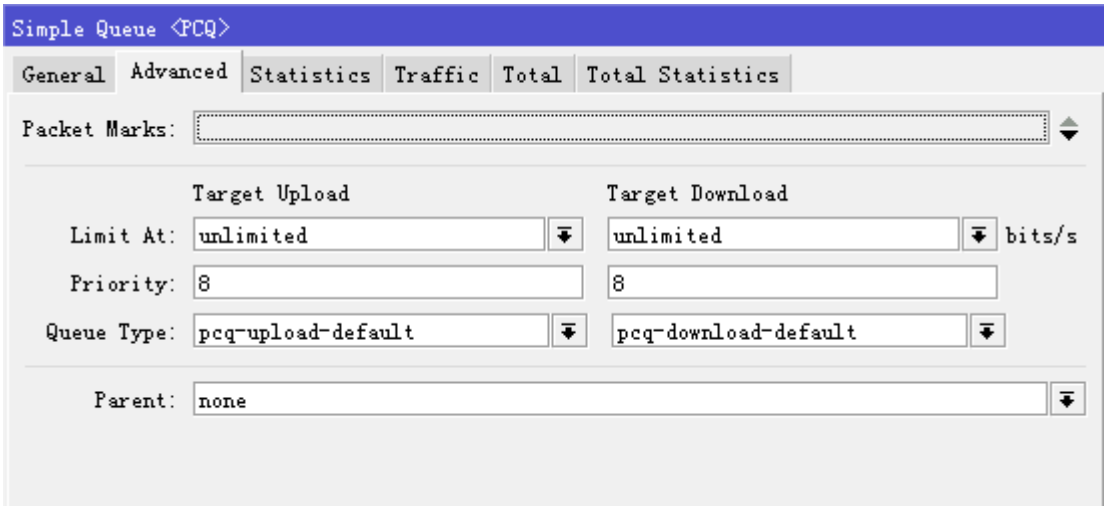
上行分类选择 src-address，并配置 512k 的上行流量配置如下：



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 10M，总上行带宽为 5M，内网地址段为 192.168.10.0/24：



进入 advanced 菜单，配置 Queue-type 类型，选择上行和下行为 PCQ 类型 pcq-download-default 和 pcq-upload-default：

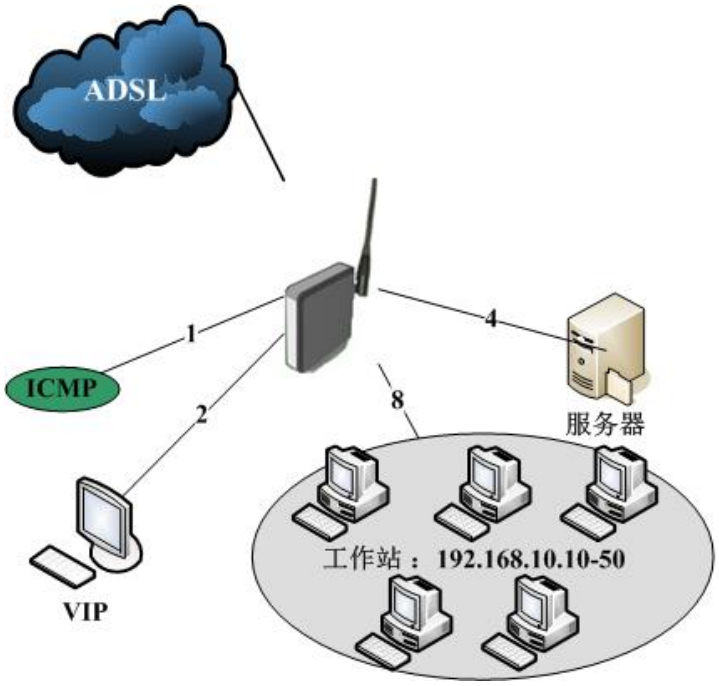


这样 PCQ 配置就完成，只需要在 simple queue 中配置一条规则，就可以控制所有 192.168.10.0/24 用户的流量。

12.10 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	员工工作电脑

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 标记连接和数据包

```
/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包（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

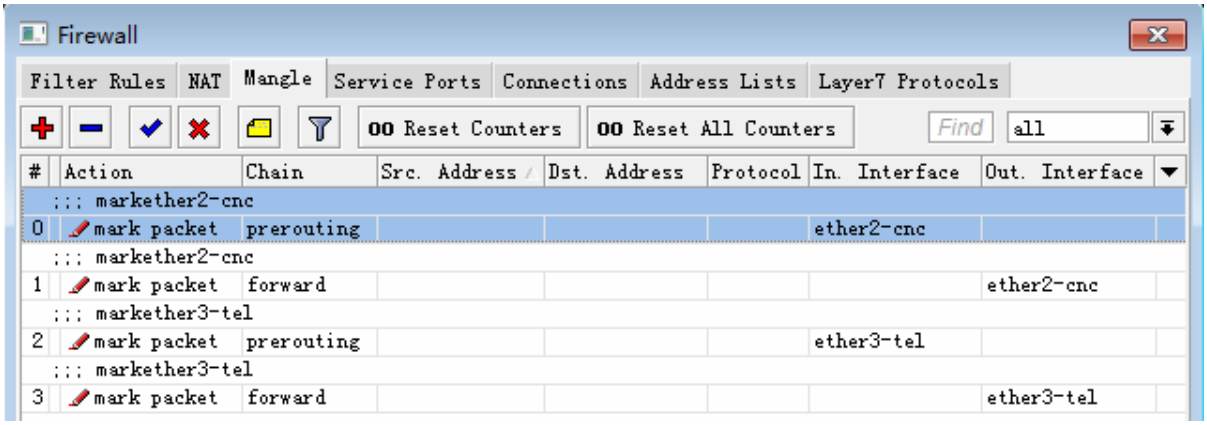
```
/queue tree add parent=Download queue=users-down packet-mark=users
/queue tree add parent=Upload queue=users-up packet-mark=users
```

12.11、网吧的 PCQ 与 HTB

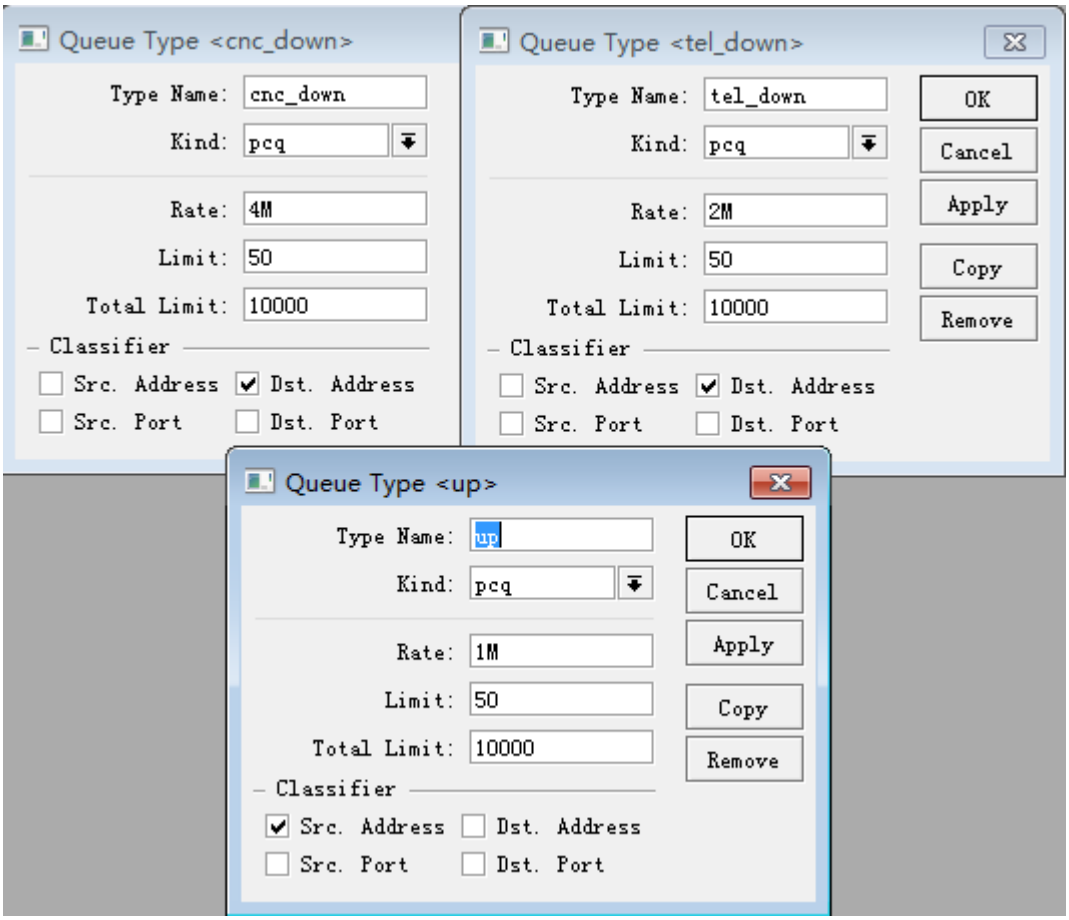
这里我们有一个实际环境，我们需要实现对带宽的动态分配；Tel 带宽为 6M，Un 带宽为 12M；

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

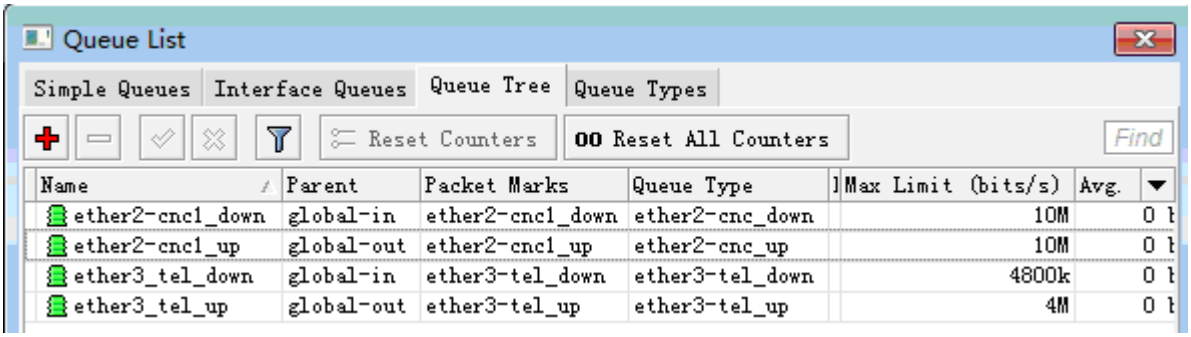
步骤 1：在 Mangle 标记上下行的标记，这里我们使用的下载标记链表为 prerouting，上传标记链表用的是 forward（为什么要用这两个链表处理数据包，可以参考第十章 RouterOS 对数据流的处理）。



步骤 2：在 Queue Type 里按照 200 台主机的数量，定义 PCQ 规则：



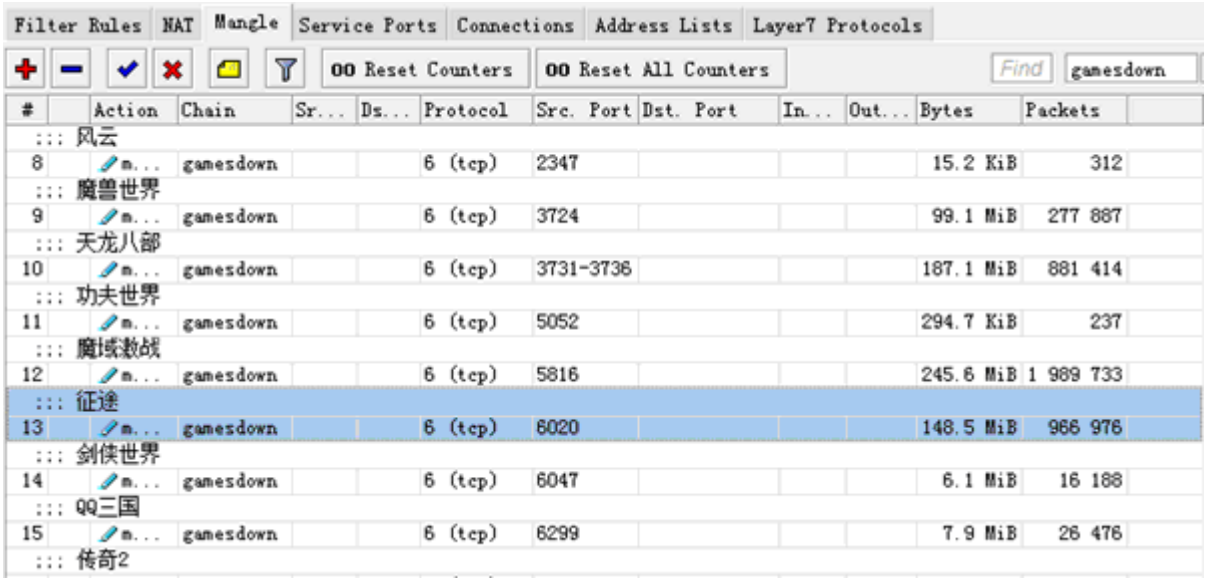
步骤 3：建立 Queue Tree 规则，记住保留一定带宽为缓冲，Un 我们保留 2M，Tel 我们保留 1.2M 带宽，这里下载使用的是 global-in，上传使用的是 global-out，记住 prerouting 和 input 链表标记的数据选择 global-in，其他两个链表 forward 和 output 则选择 global-out。



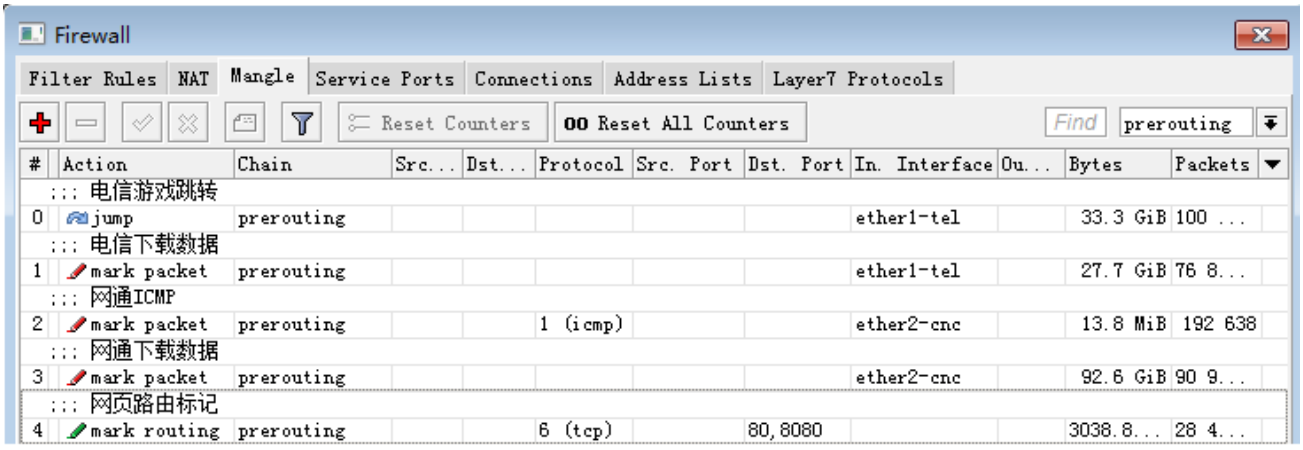
HTB 游戏优先

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：

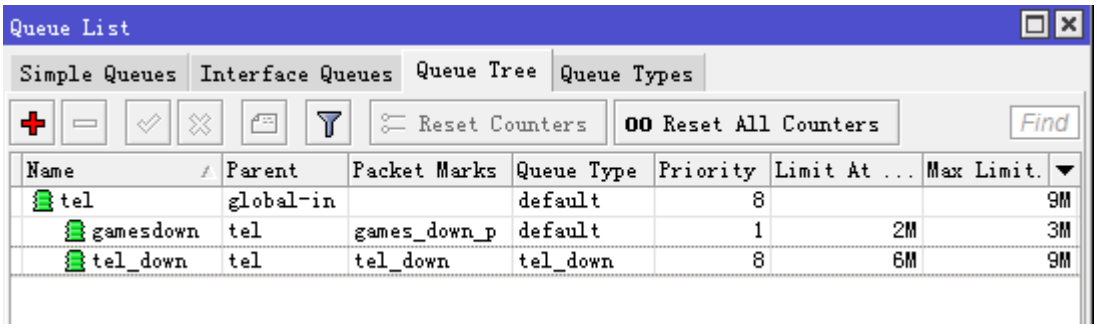
步骤 1：在原有的动态的 PCQ 流控规则上进行改进，首先导入游戏端口，建立新的 gamesdown 链表，将游戏与其他数据区分出来



通过将指定的数据转移到游戏链表进行过滤和数据包处理：



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。

12.12 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包到特定的项目，数据包的长度值（包括 IP 数据包头）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。

两个选项 "connection-bytes" 和"connection-rate" 工作只能在 TCP 和 UDP 传输。(你需要指定协议激活这些选项) 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，我们需要确定识别"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。)

下面实例让我们假设，我们有 6Mbps 上传和下载

```
per-connection-classifier=
PerConnectionClassifier ::= [!]ValuesToHash:Denominator/Remainder
  Remainder ::= 0..4294967295    (integer number)
  Denominator ::= 1..4294967295    (integer number)
  ValuesToHash ::= src-address|dst-address|src-port|dst-port[,ValuesToHash*]
```

实例脚本

```
/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，这样数据包标记从 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 中将标记数据包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。

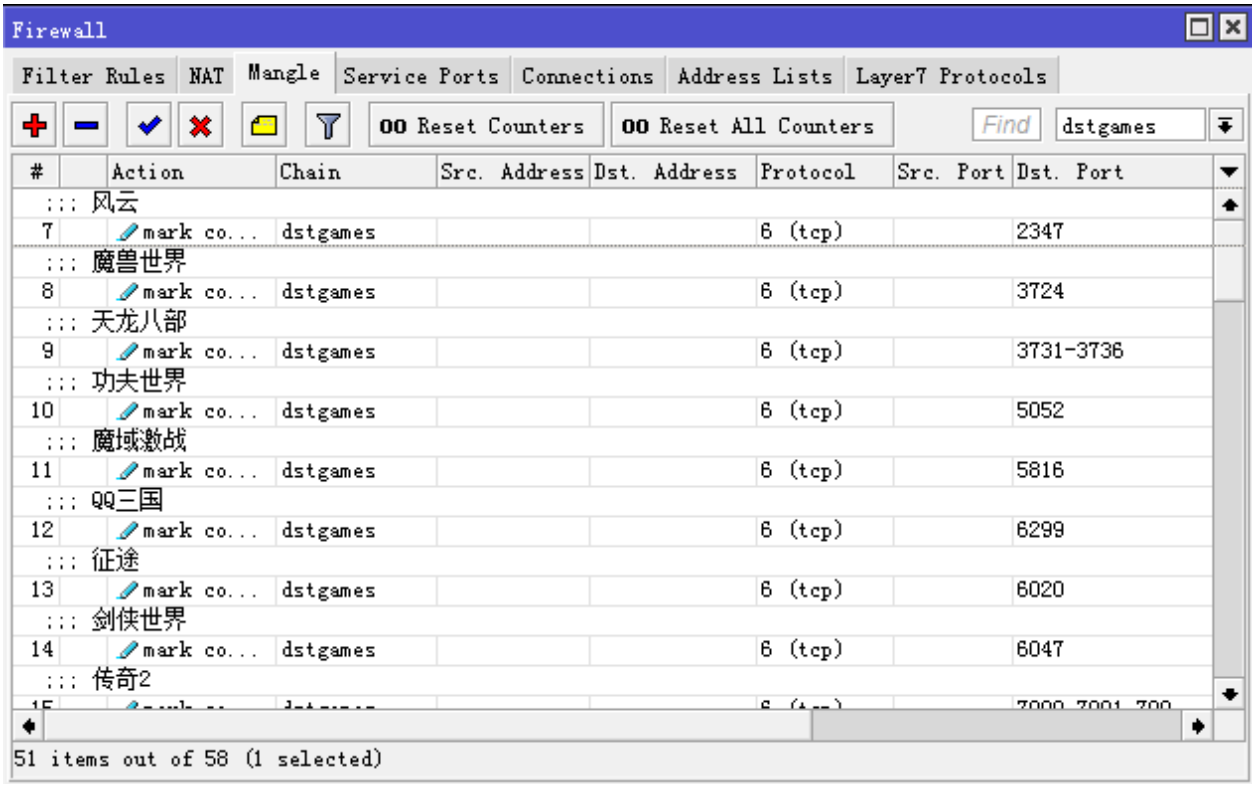
通过 Mangle 标记

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：脚本文件载入，并执行成功：

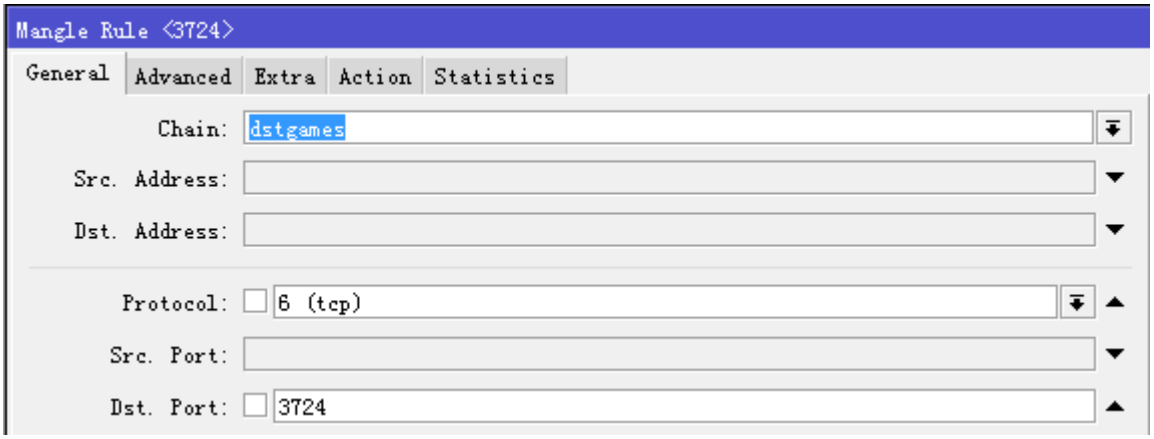
```
[admin@MikroTik] > import games_v2.rsc
Opening script file games_v2.rsc
Script file loaded and executed successfully
[admin@MikroTik] >
```

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

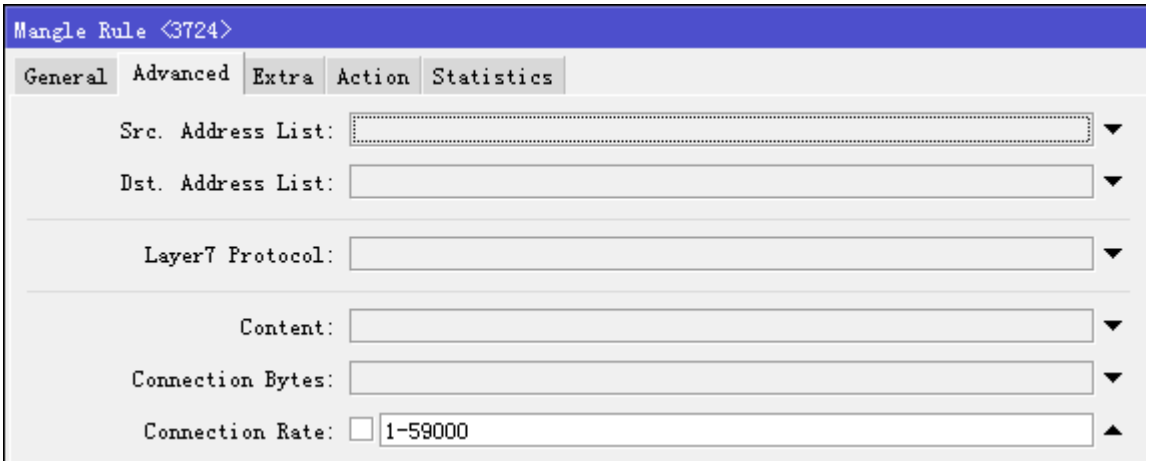


每条规则都标记的是目标端口，因为这里我将使用 forward 链表跳转数据到这个游戏标记，如果使用 prerouting 链表注意接口方向。

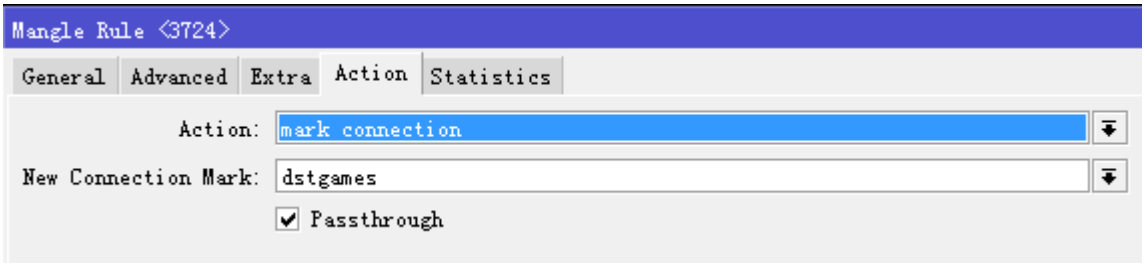
下面是魔兽世界的端口 tcp/3724，标记服务器的端口



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



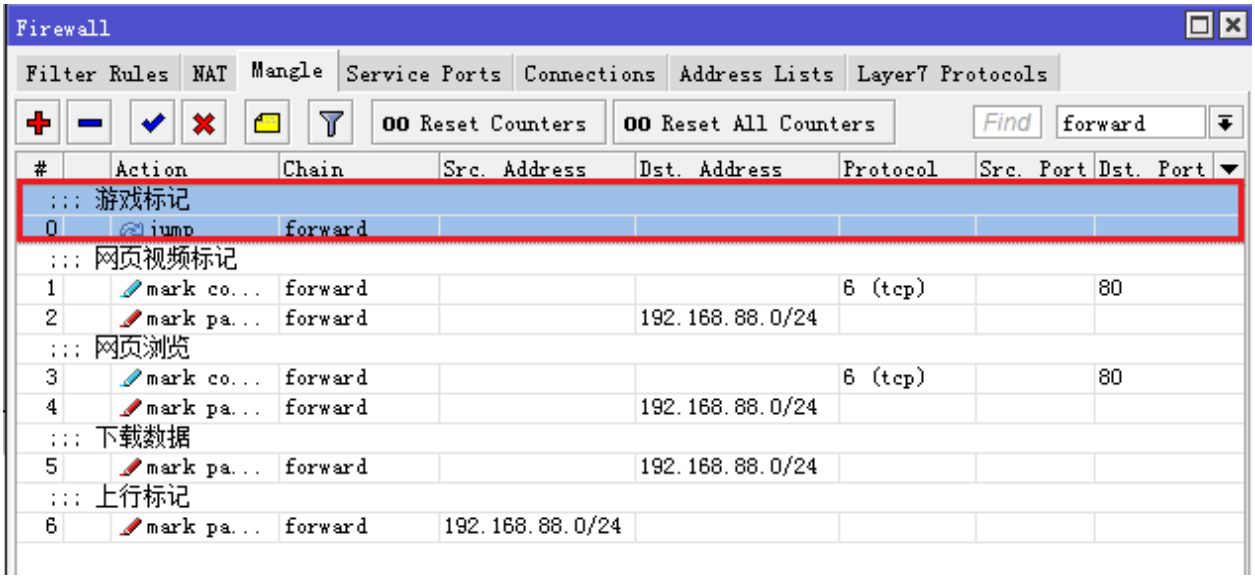
最后是在 action 里标记为 mark-connection，并填写标记名称 dstgames，passthrough=yes 要传递给后面的数据包标记规则



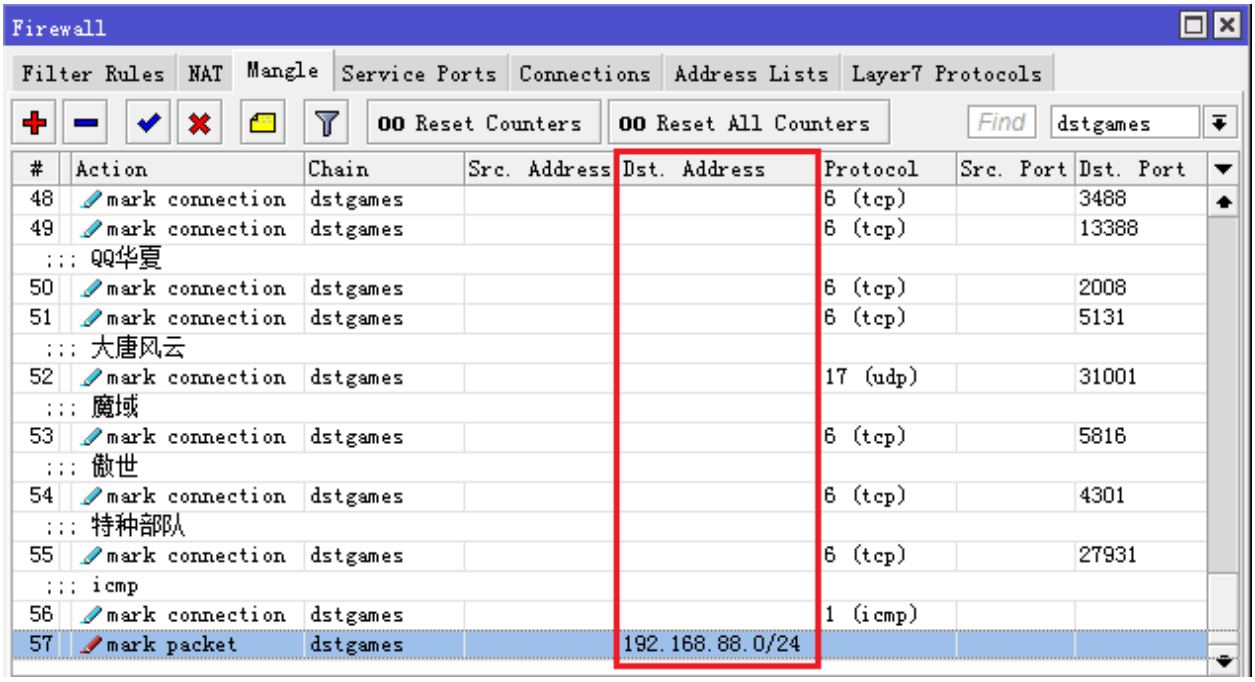
Forward 标记

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
```



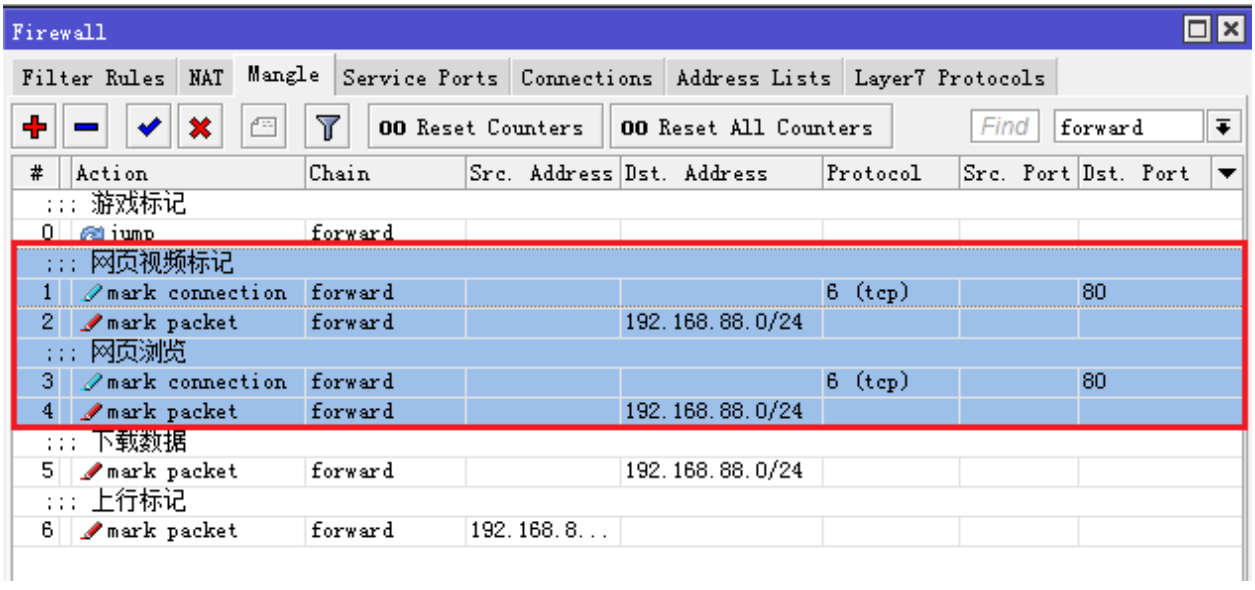
这条规则在 forward 链表里排在最前面，因为是首先处理的数据。这里我们的内网地址段是 192.168.88.0/24，我们需要通过地址来区分下载和上传，所以我们需要返回 dstgames 链表里，将数据包标记规则修改下



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。

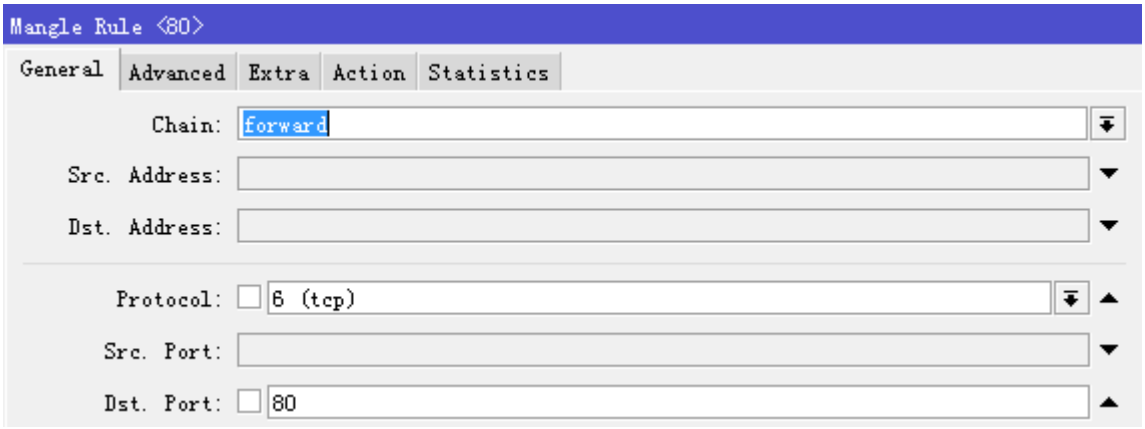
网页视频和浏览区分

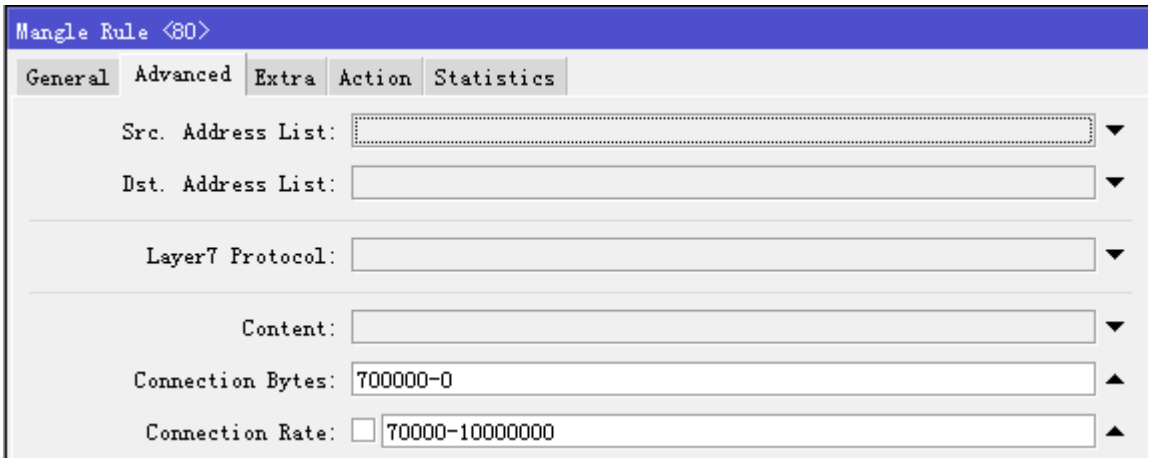
网页视频和网页浏览的区分，仍然采用 connection-rate 参数，区分网页的规则一共 4 条，如下图：

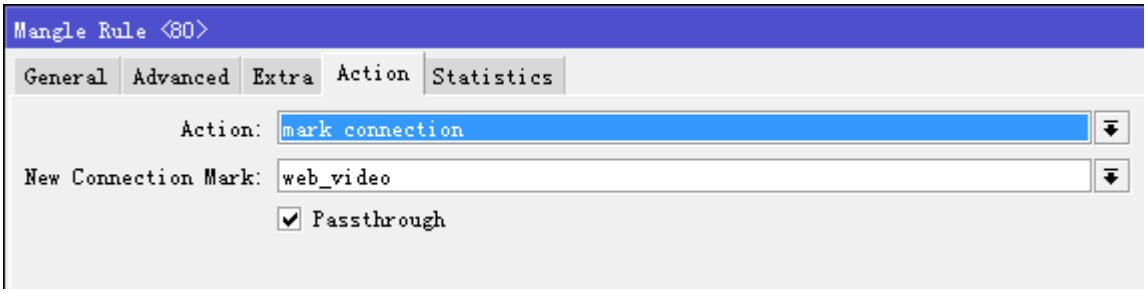


4 条规则里关键是在第一条连接标记，用于区分网页视频的速率，

```
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
```

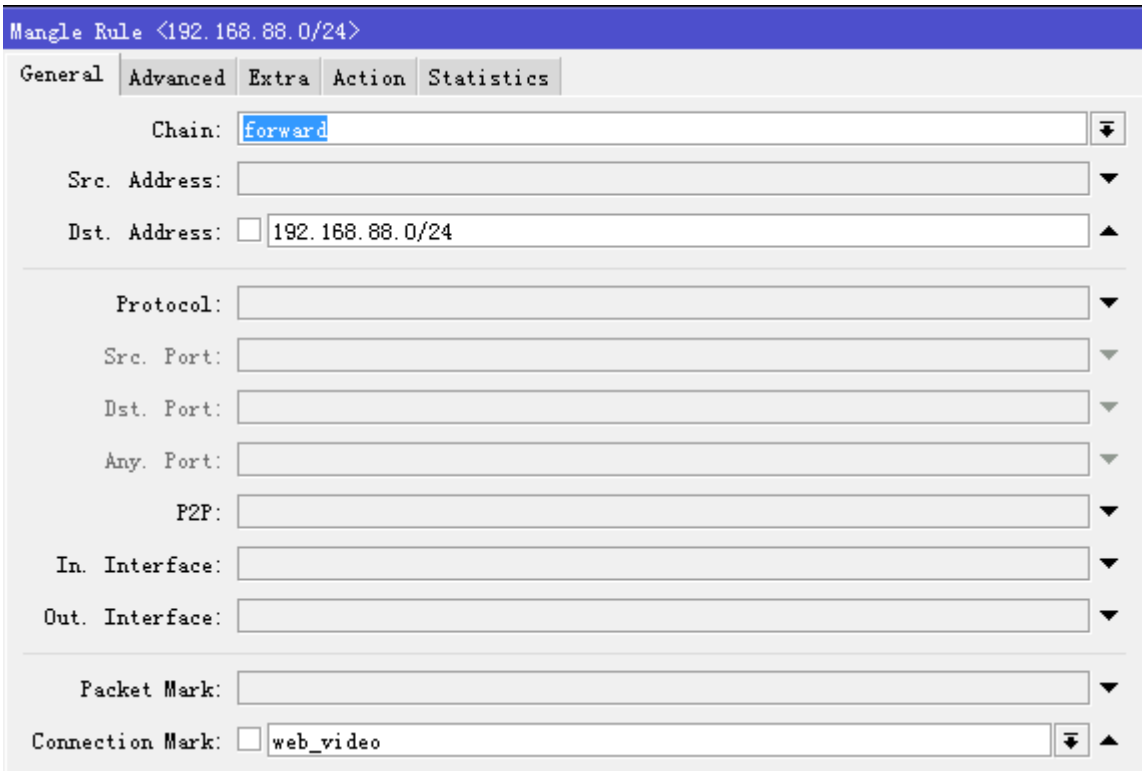


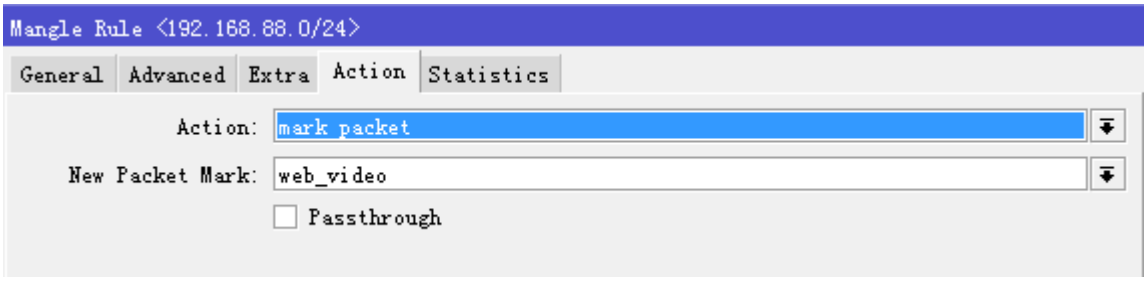




标记所有 tcp/80 端口，并设置 connection-bytes 为 700k 范围内，即初次使用了 700K 的字节，之后仍然保持 70kbps~10Mbps 的速率，就认为是网页视频，并标记名称为 web_video

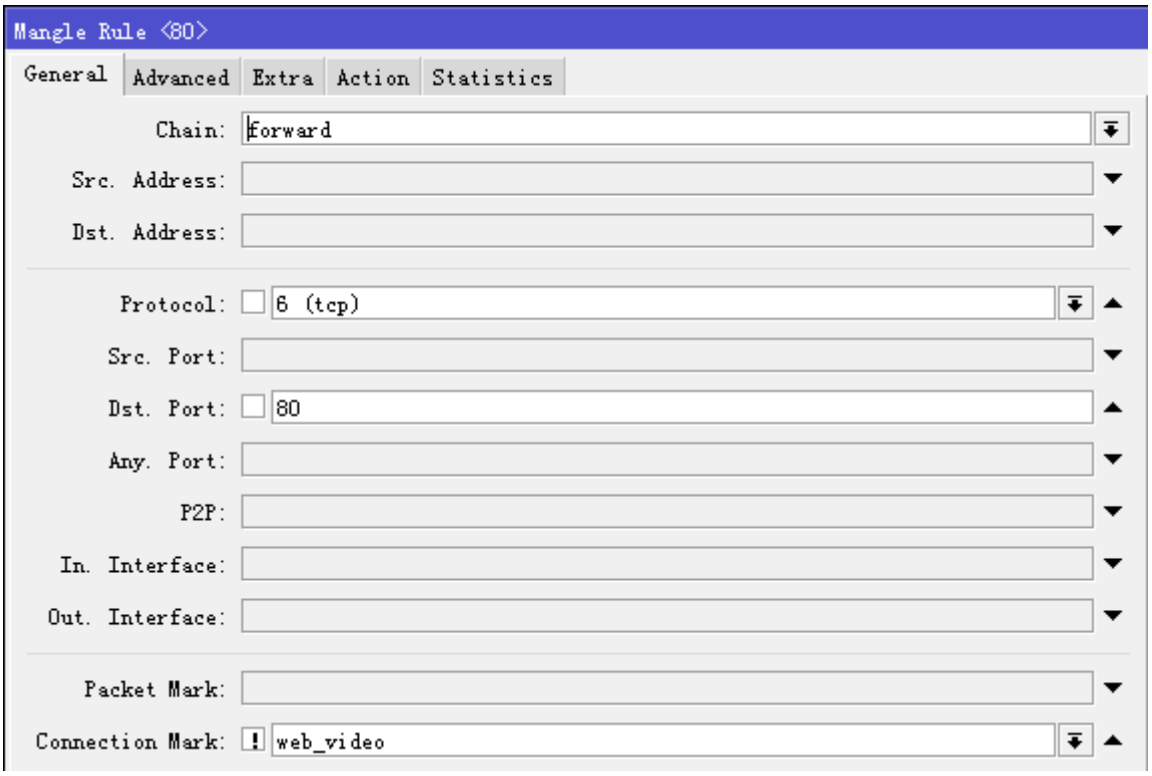
接下来的一条规则就只需要从这个连接里提取数据包，并指明目标地址是 192.168.88.0/24 的下行数据，action=mark-packet，取名标记为 web_video，注意这个 web_video 和连接的 web_video 并不相同，一个是连接一个是数据，这里的 passthrough=no，因为已经被数据标记处理，不需要交给后面的规则

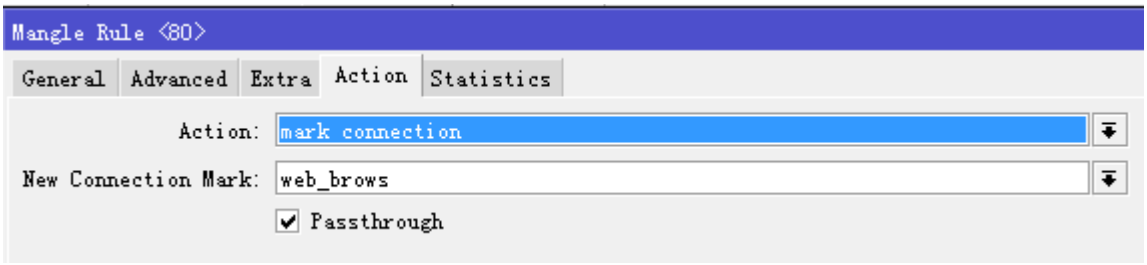




接下来我们标记网络浏览连接，仍然是是 tcp/80，但这里的 connection-mark 设置为非 web_video 的连接

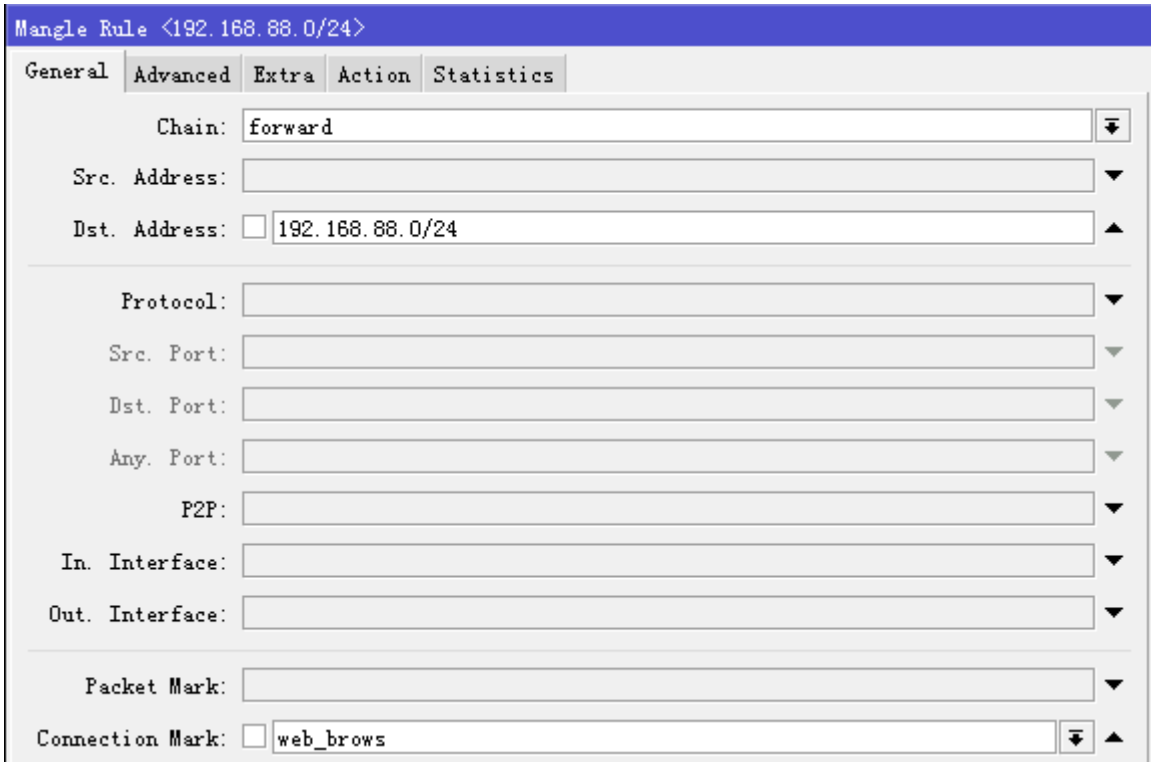
```
chain=forward action=mark-connection new-connection-mark=low_web passthrough=yes protocol=tcp
dst-port=80 connection-mark=!webhighspeed
```

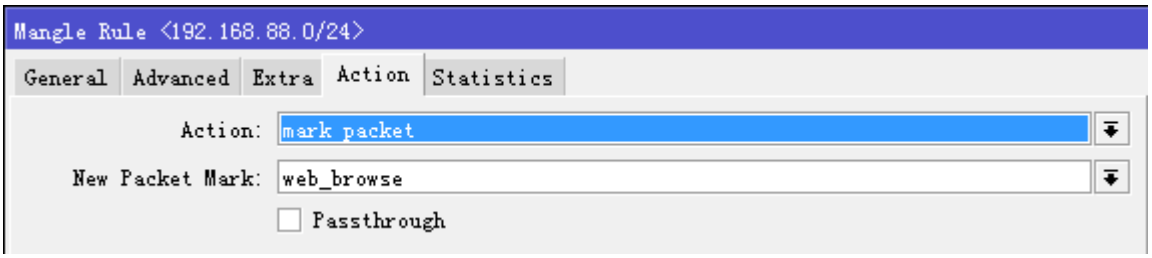




标记连接名称为 web_brows，并传递给后面的数据包标记规则

同样的方式，提取 web_brows 的数据包标记，选择目标地址并同样取名为 web_brows，passthrough=no

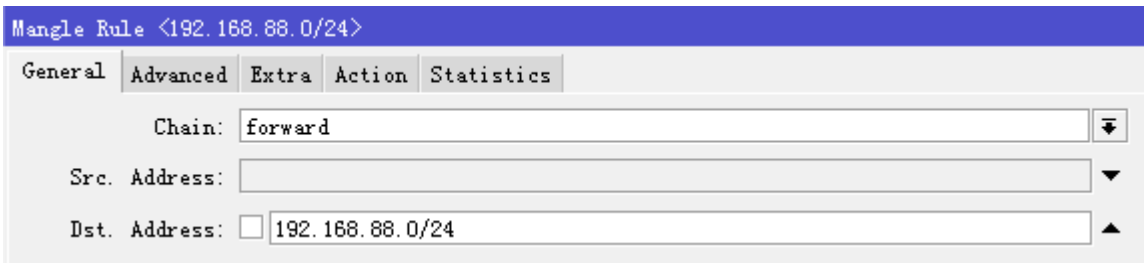


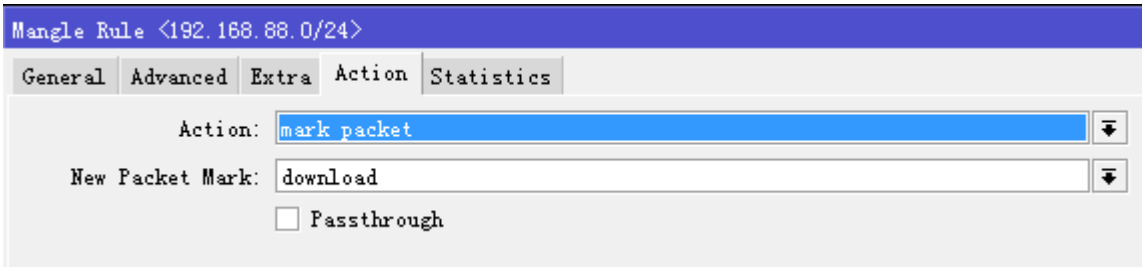


其他下载数据标记

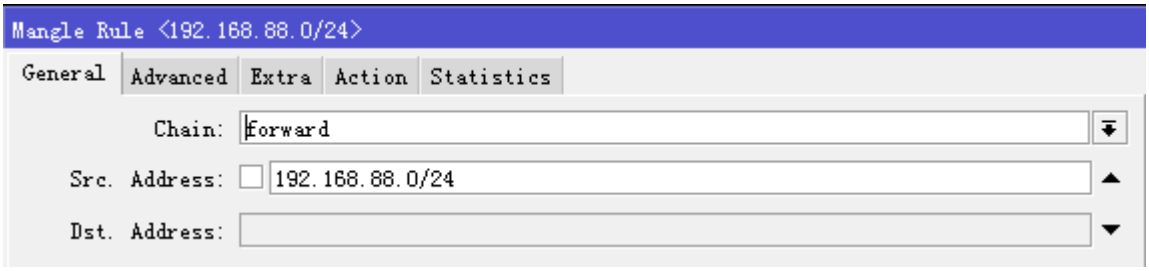
这样我们已经将部分游戏、网页视频和网页浏览区分出来，剩下的被认为是其他下载数据，直接做数据包标记

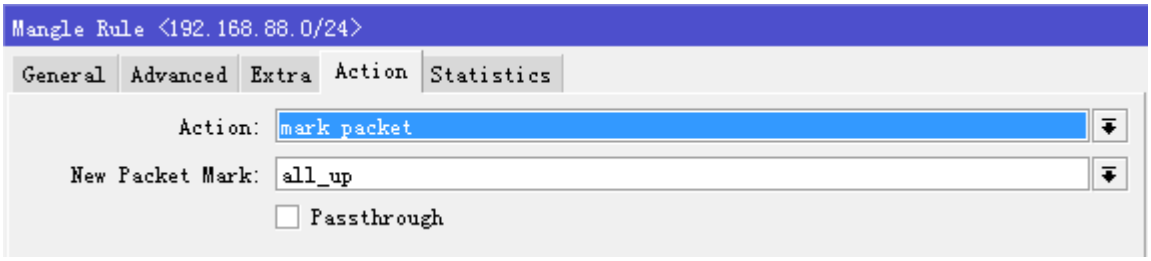
```
chain=forward action=mark-packet new-packet-mark=download passthrough=no
dst-address=192.168.88.0/24
```





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。

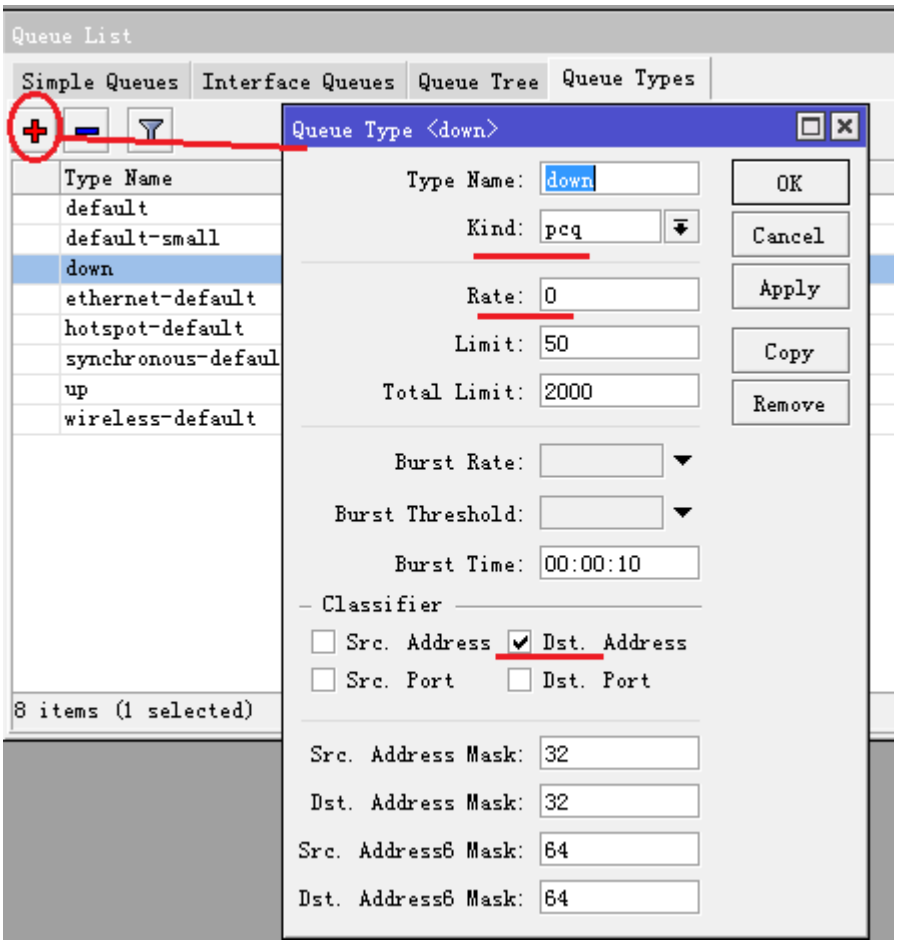




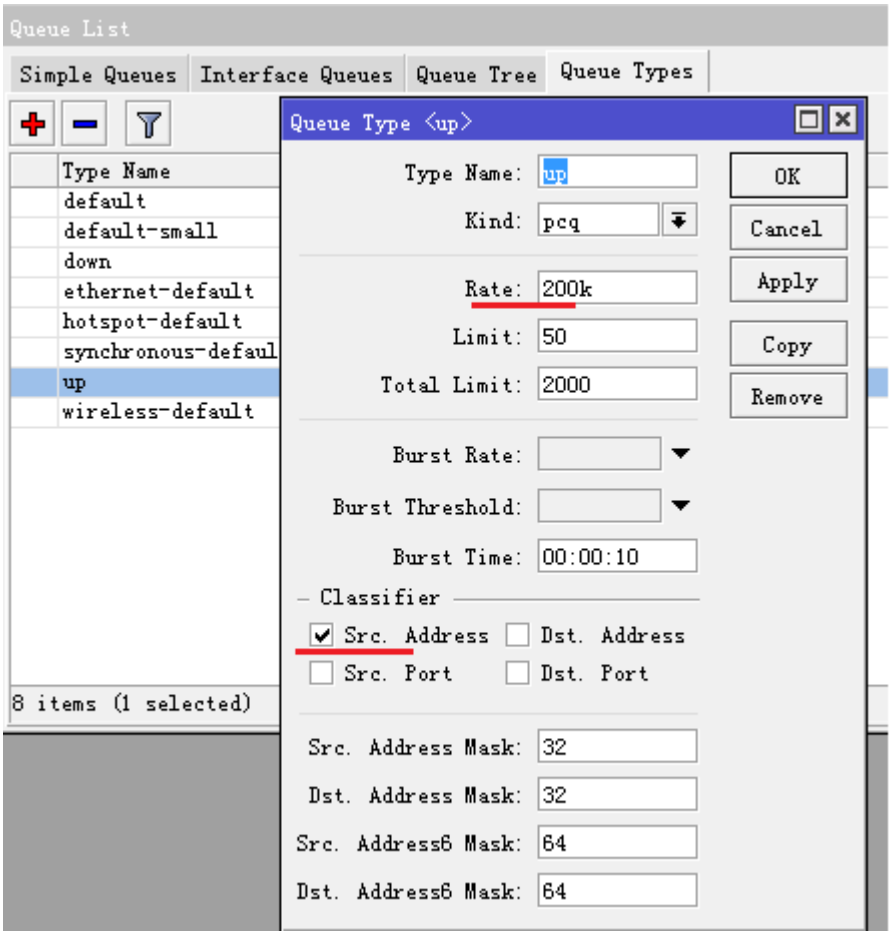
HTB 和 PCQ 设置

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

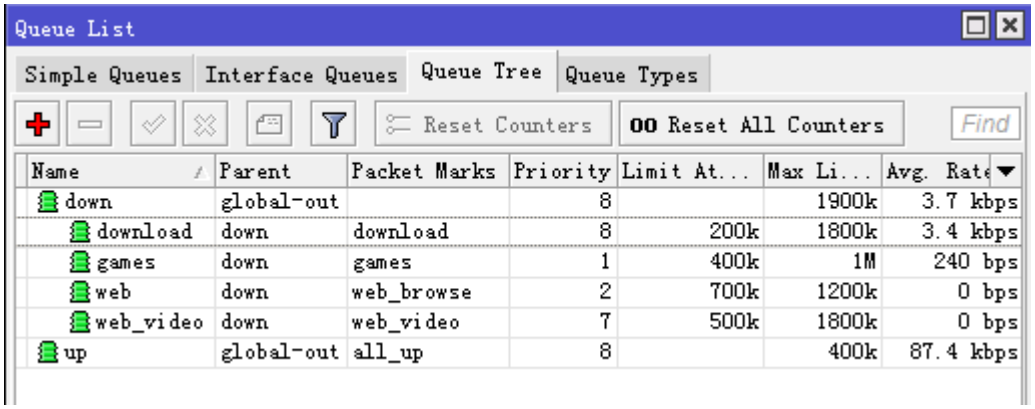
进入 queue type 添加 PCQ 规则，我们添加一个 down 的 PCQ，kind=pcq，rate 设置每台主机的带宽，这里我们选择 0，即动态分配带宽，你也可以设置一个固定值，以 k 或者 m 为单位，选择 dst-address 为下行的规则



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



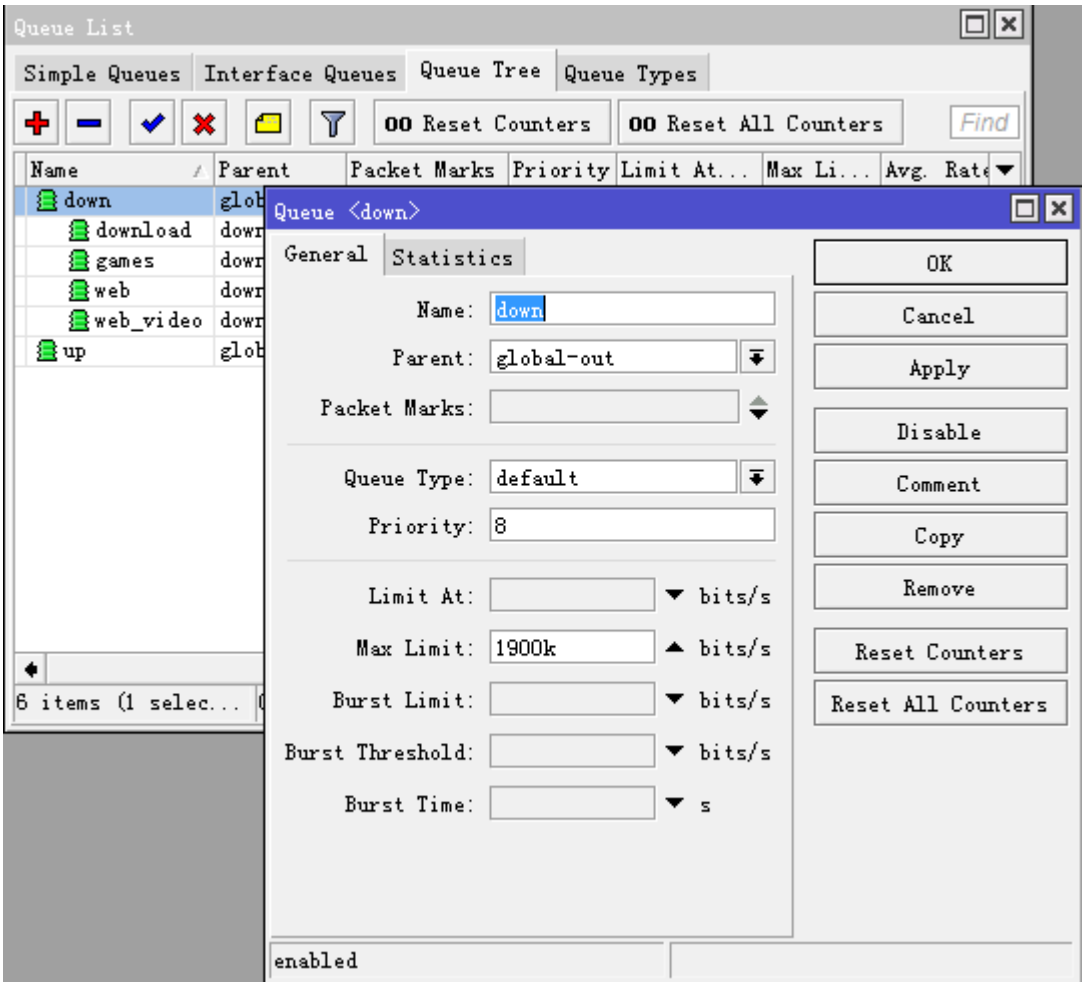
PCQ 设置完成后，我们接下来就需要配置 HTB，HTB 是由父级队列和子队列组成，我们只对下行做 HTB，上行数据则采用普通的 PCQ 限速（没有子队列的规则不能称为 HTB），如下图



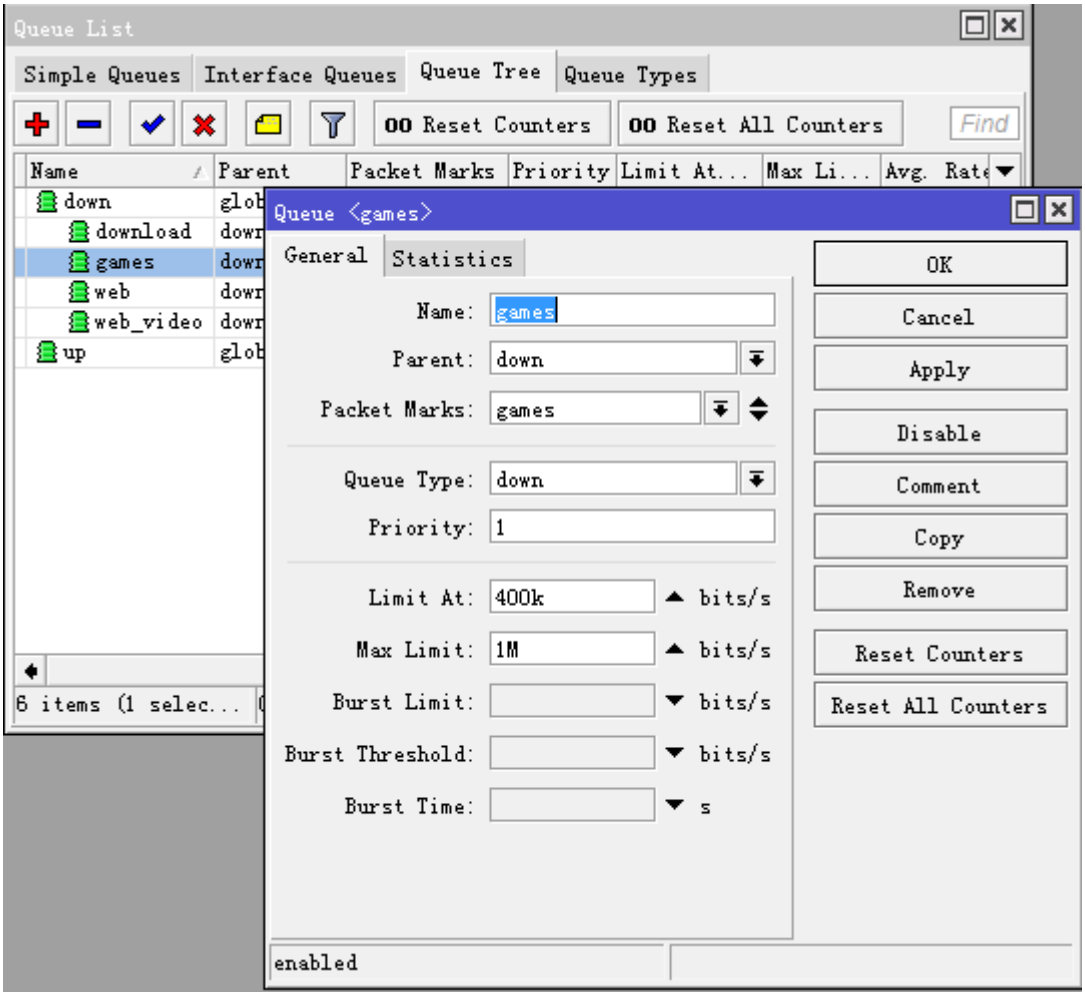
配置原则说明：

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）

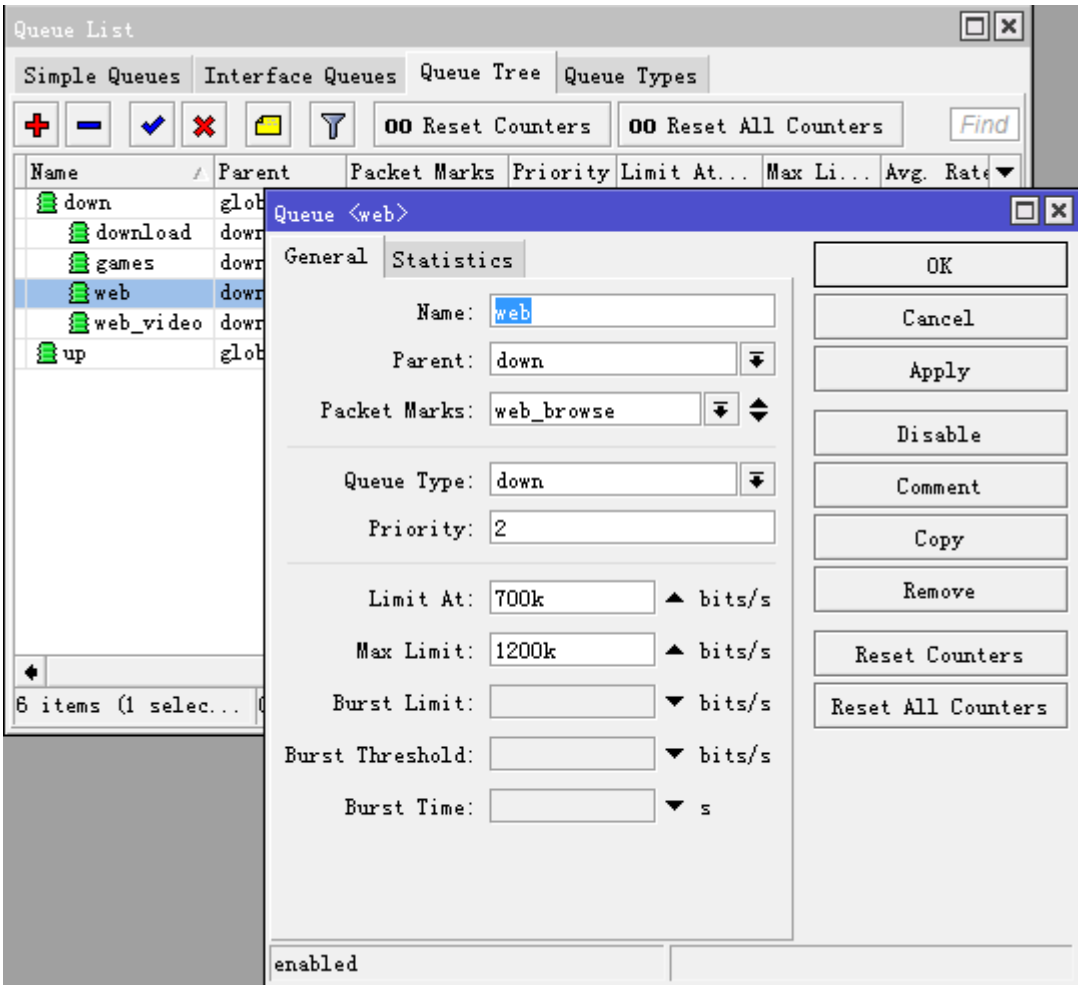
根据以上原则，添加 HTB 的规则的思路就明确了，首先添加父级队列，这里我们不设置任何 packet marks 标记，仅作为父级带宽分配给下面的子队列



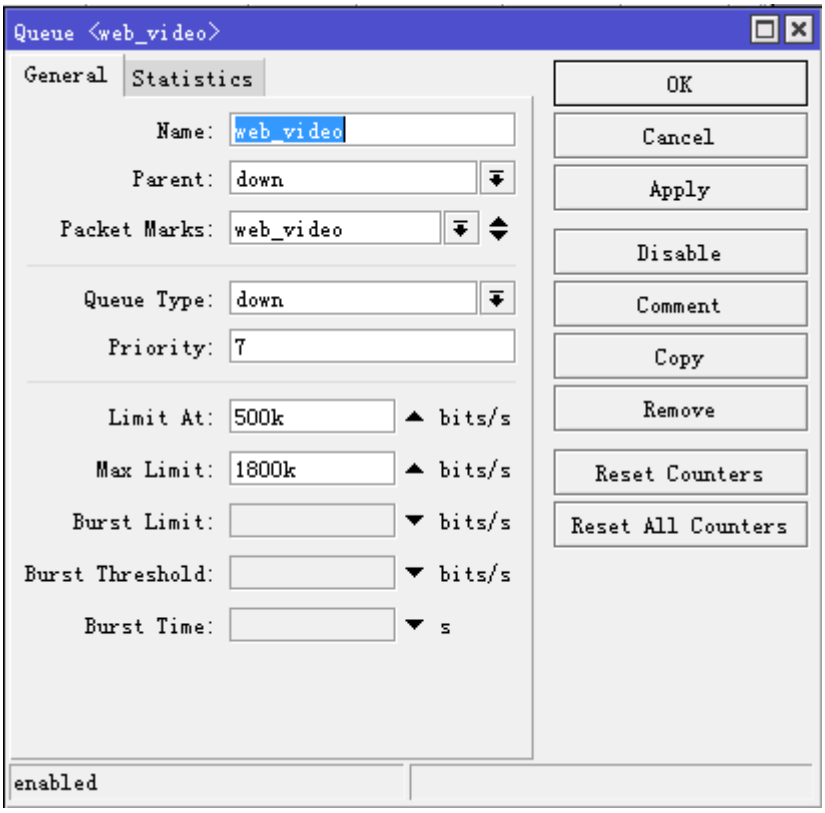
我们添加子队列，首先添加游戏子队列，我们需要设置父级 parent=down，需要从 down 获取带宽，packet-mark 为之前标记的 games，选择 queue type 为刚才的 PCQ 规则 down（给每个主机动态分配带宽），priority 优先级为 1 最高，我们给游戏分配 1M 带宽，最低保证 400kbps 带宽，如果有剩余带宽可以从父级获取得到最大 1Mbps



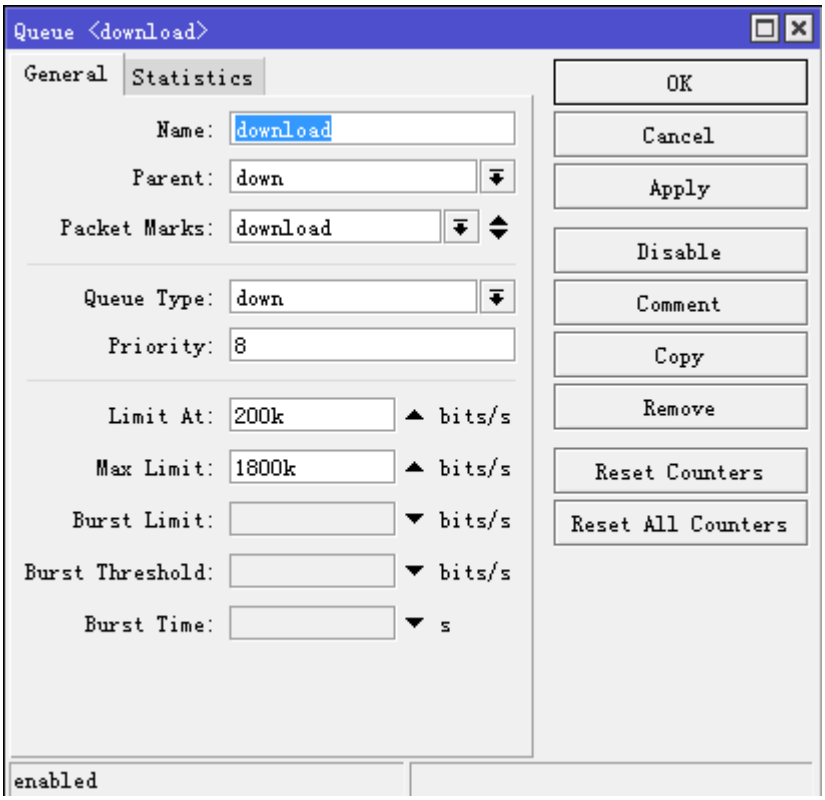
其次我们添加网页浏览的流控，使用同样的方法，priority 为 2，max-limit 为 1200k，保证最低获得 700k



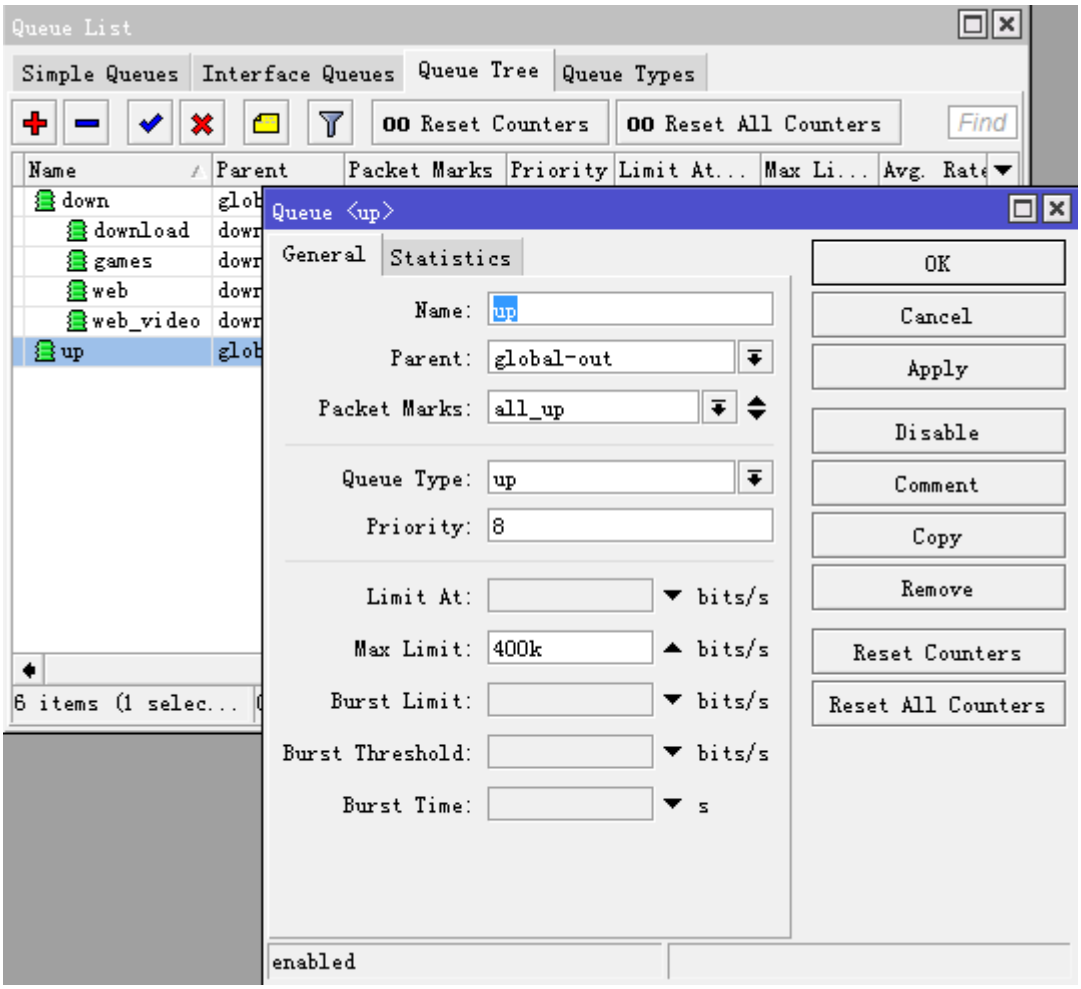
网页视频的优先级为 7，最大可以获得 1800k，最低保证 500k



其他下载数据优先级最低 8，最大带宽为 1800k，最低保证 200k



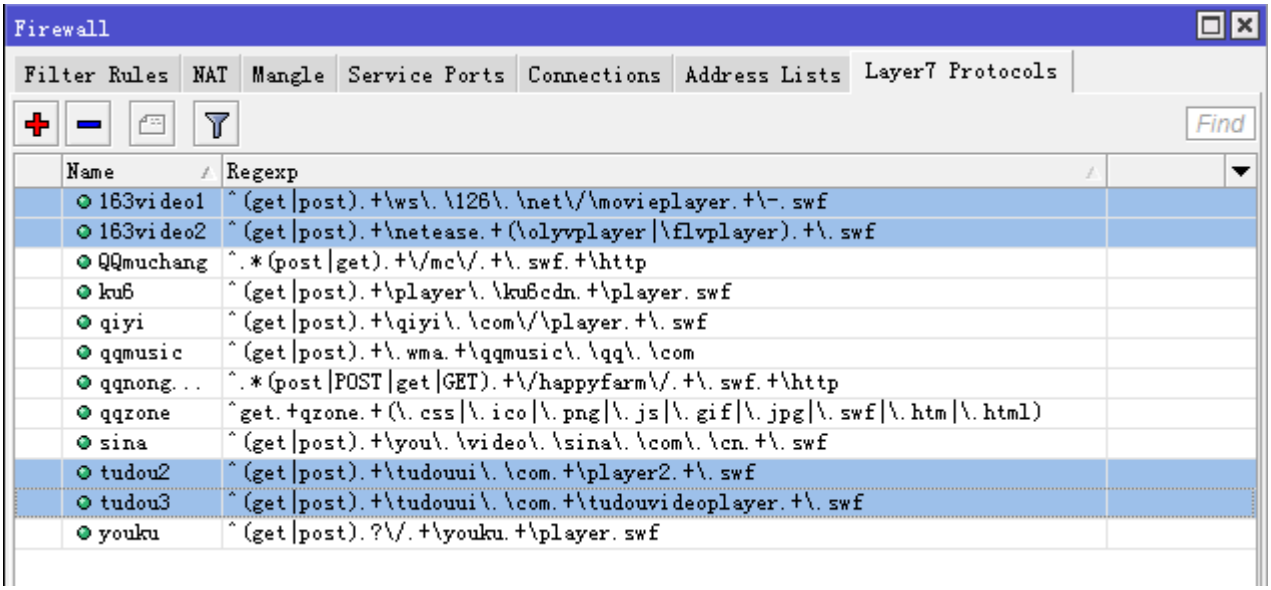
上行带宽的 PCQ 规则，取名为 up，因为上行是独立的 PCQ 规则，所以父级是 global-out，选择 packet-mark 为 all_up，queue-type 为刚才的 PCQ 规则 up，每台主机 200kbps 带宽，max-limit 为 400k（因为 2M 的 ADSL 上行带宽为 512k，保留一部分带宽）



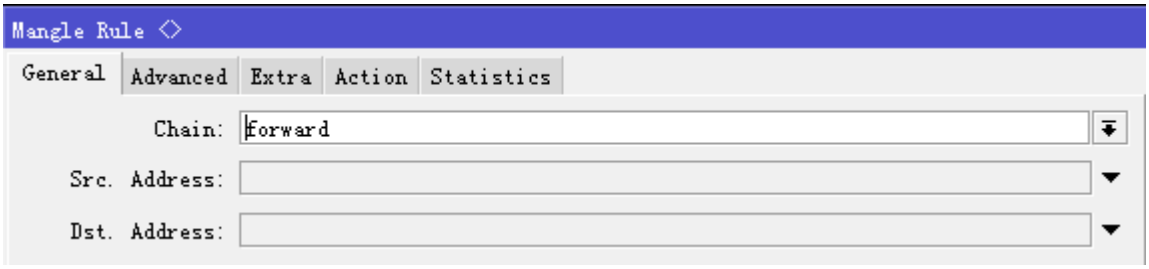
12.13 L7 流控

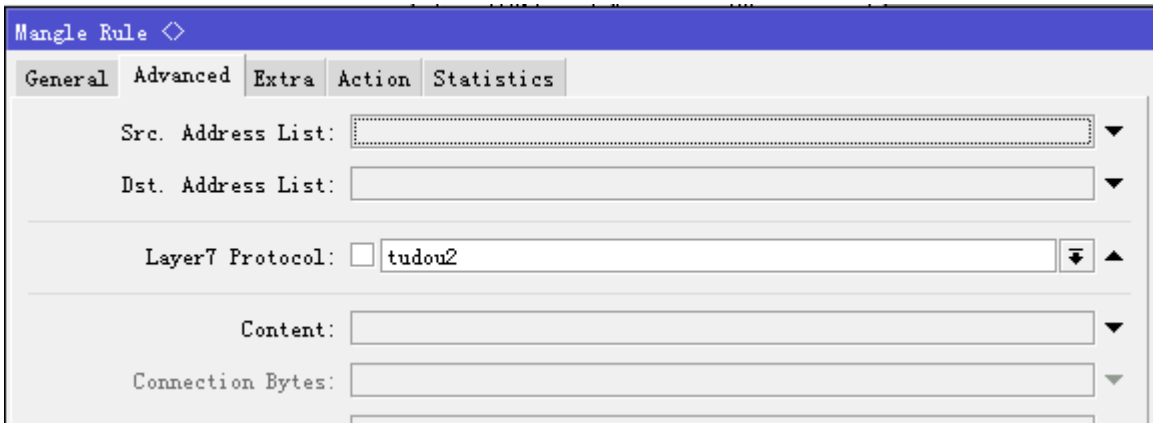
我们能通过 RouterOS 集成的 L7 协议识别一些网络软件，并对其进流量控制，例如我们对网页视频进行流控，通过 L7 协议抓取的网页视频比使用 Connection-rate 更精确，例如我们对网易和土豆网站的视频进行流控

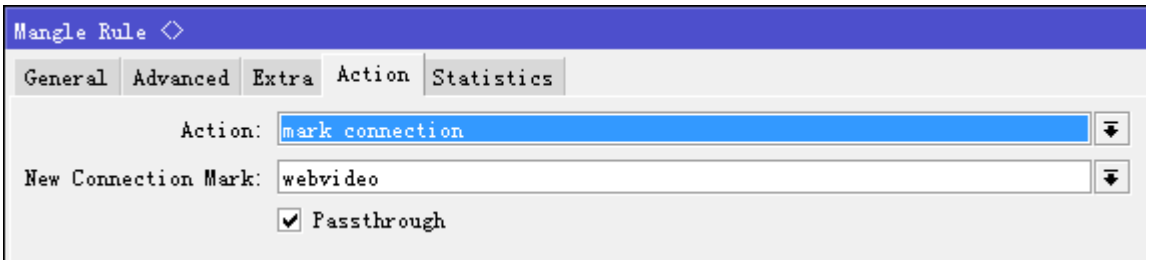
首先我们需要将网易和土豆的 L7 表达式写入/ip firewall layer7-protocol（具体的 L7 表达式，请参见防火墙章节的 L7 协议），如下图，我们选择 163 和 tudou 的视频 L7 规则



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：



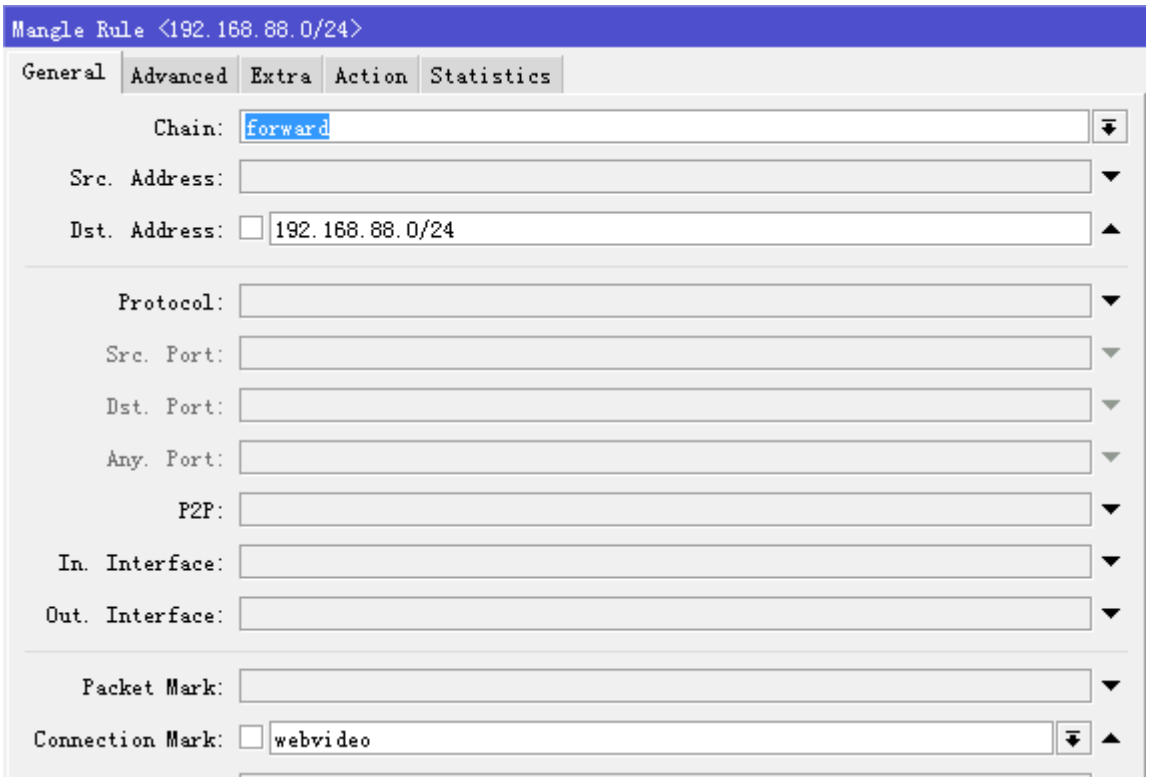


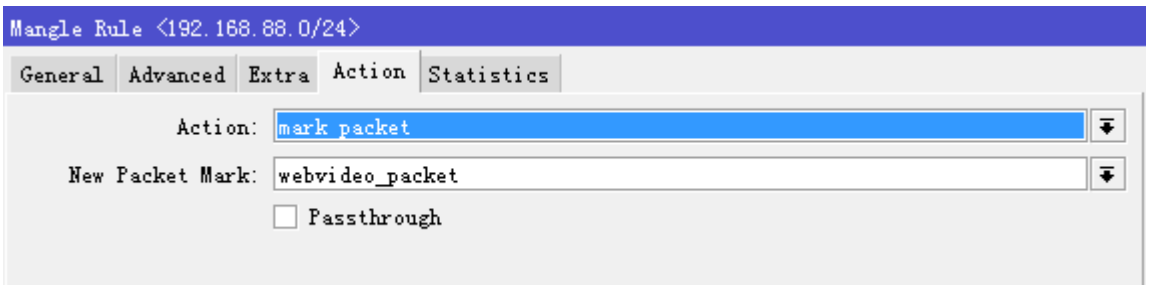


我们分别将其他几条，下面是连接标记的 mangle 脚本：

```
/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，我们需要指定数据包的方向，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：



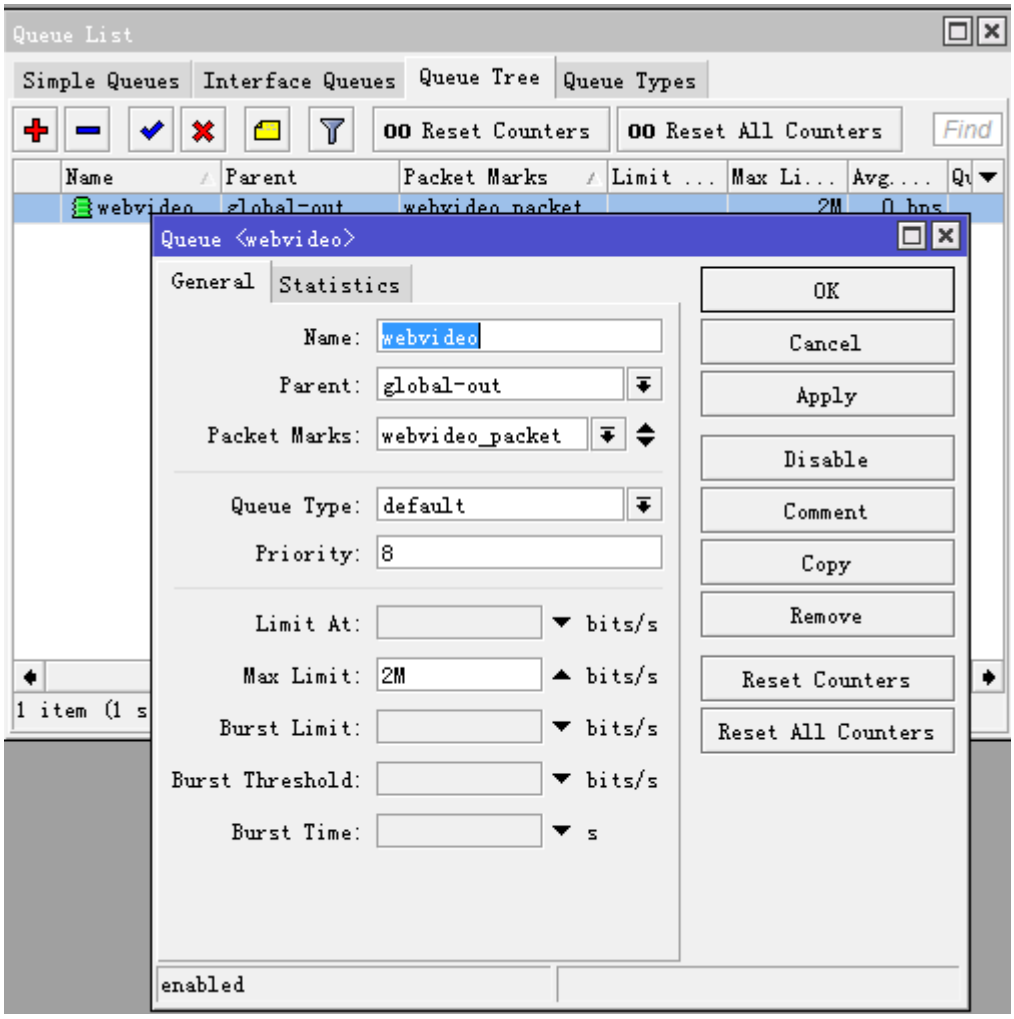


我们脚本如下：

```
/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，我们给网页视频带宽为 2M，在这里我们使用的是 forward 链表，parent 选择 global-out，具体配置如下：



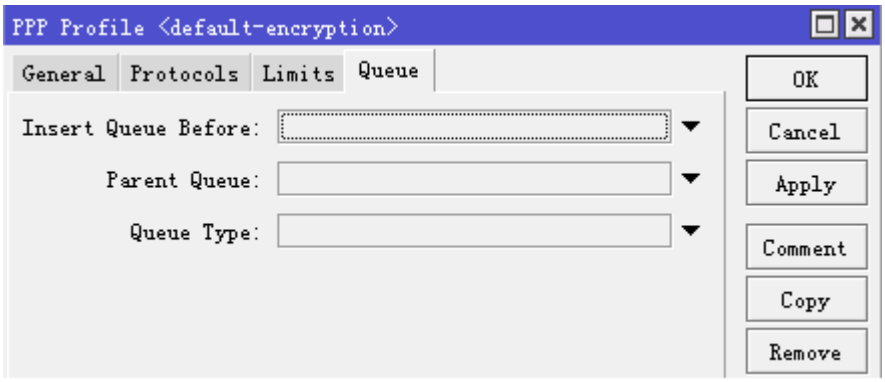
Queue tree 脚本如下

```
/queue tree
add max-limit=2M name=webvideo packet-mark=webvideo_packet parent=global-out
```

12.14 PPP Profile 动态 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

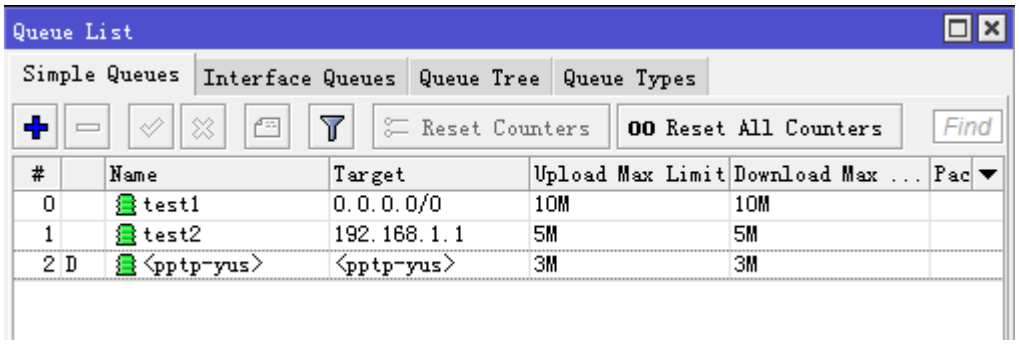
Parent Queue – 指定从属于父级的 simple queue 规则，有助于管理和优先级处理

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

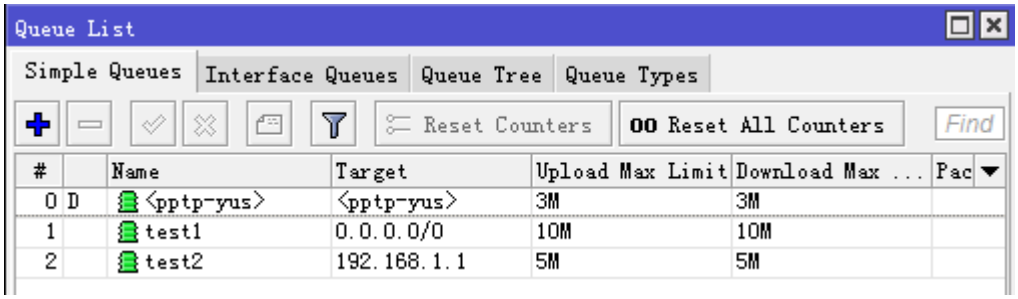
1、Bottom

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



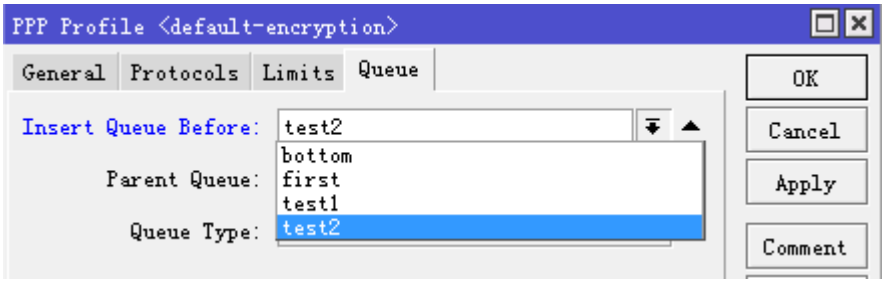
2、First

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

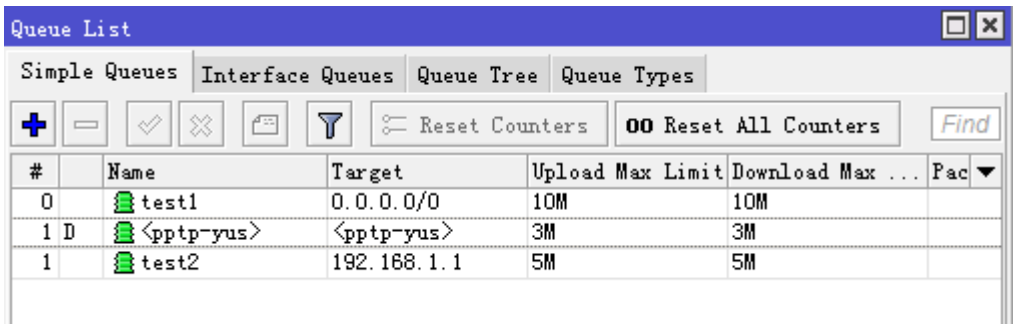


自定义：

自定义即，选择指定插入某条规则，如我插入 test2 之前



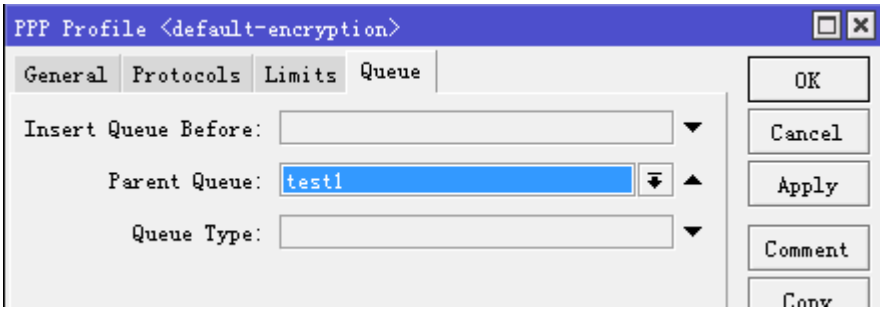
这样，pptp-yus 被放在 test2 之前



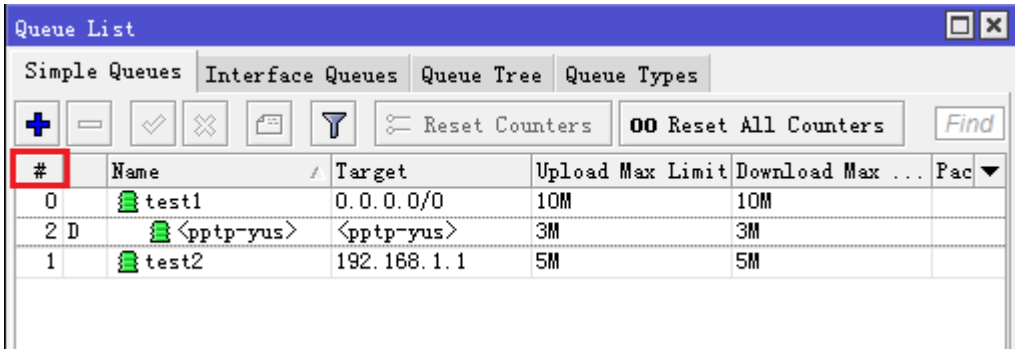
3、Parent 属性

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，

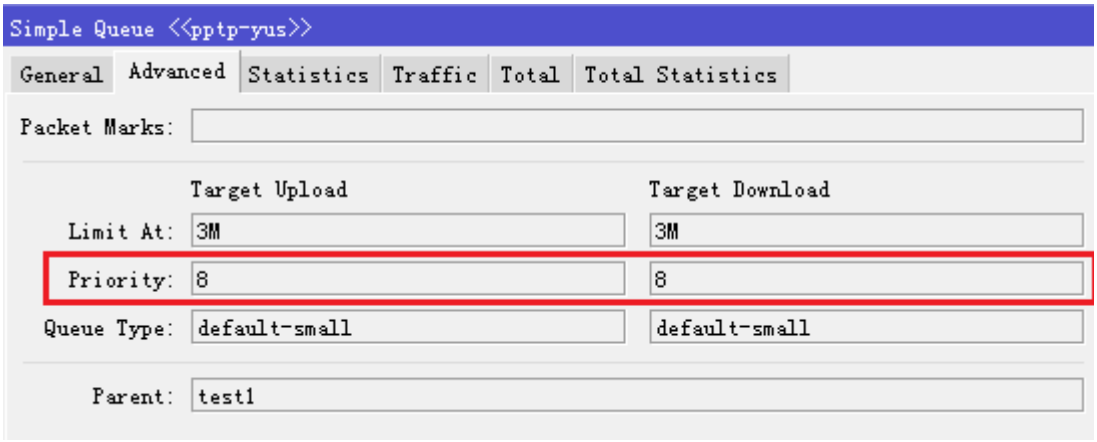
这里我们定义下面 PPP profile 的 Parent Queue 属性为 test1，即从属于 test1 规则下



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 数据包做特殊的标记，mangle 是通过修改指定的 IP 数据包头字段，去标记 IP 数据包的特征 能标记端口、IP、协议、TCP 协议和相应的 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。

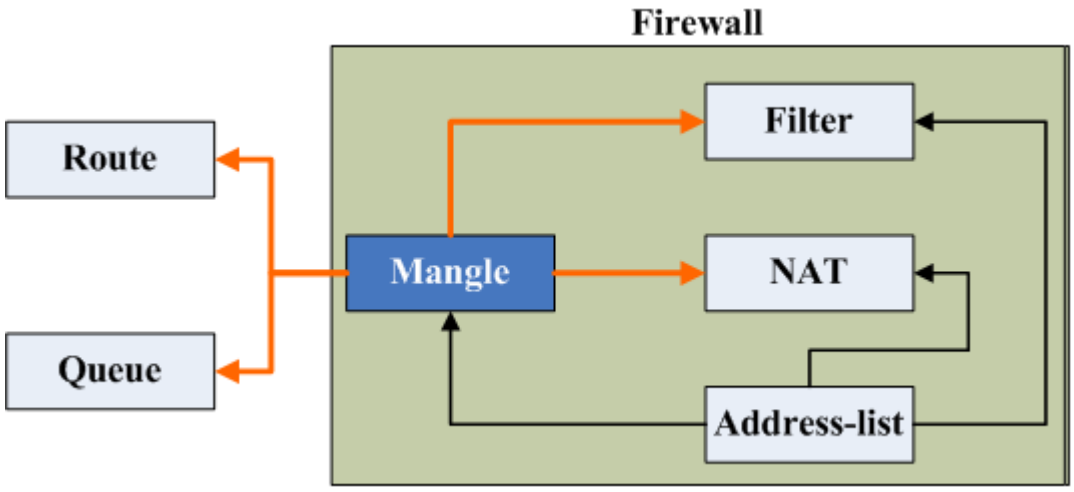
需要功能包: ***system***
需要等级: *Level1*
操作路径: ***/ip firewall mangle***
协议标准: IP

13.1 Mangle 介绍

Mangle 是一种标记器，标记特殊的数据包等待将来处理。在 RouterOS 中许多其他的功能组件会使用到他，如 queue-trees 和 nat，他们识别到一个数据包了标记的便会做相应的处理。Mangle 标记仅存在于该路由器中，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**
- **Prostrouting：路由后**

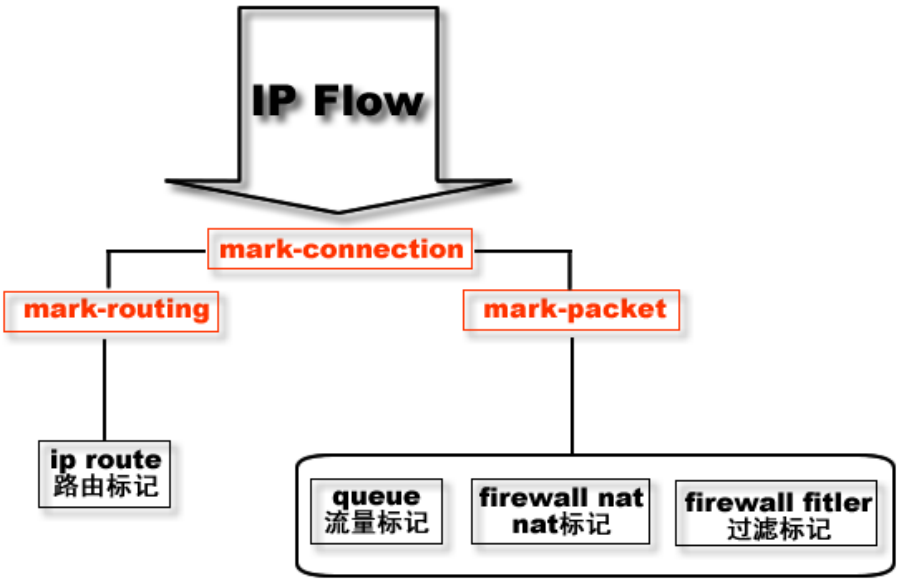
RouterOS 中的 IP firewall 主要由 3 个规则部分组成 Mangle、Filter、NAT ，而 Address-list 常用于地址列表分类。Mangle 通过标记特定的 IP 数据流后，为 Filter、NAT 和、路由、Queue 提供标记后的 IP 数据流



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 流中数据包**
- **Mark-routing：标记 IP 流中 IP 数据包的路由信息**

三种类型的关系，所有的在 IP 数据包传输前，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包，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
```

第十四章 Bridge 网桥

桥接（Bridge），是工作在 OSI 网络模型的第二层链路层，对网络数据包进行转发的过程。简单的说就是通过网桥可以把两个不同的物理局域网连接起来，是一种在链路层实现局域网互连的存储转发设备

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。

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
- 支持基于桥数据包过滤器

规格

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 STP/RSTP，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包倍增。RSTP(rapid spanning tree protocol)则是快速生成树协议收敛时间更短。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 wireless 无线教程 v6》

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，实现多个局域网直接的 MAC 层数据转发，类似与交换机的功能，即学习、存储和转发的功能，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。

快速配置指南

把接口 **ether1** 和 **ether2** 放在一个桥里：

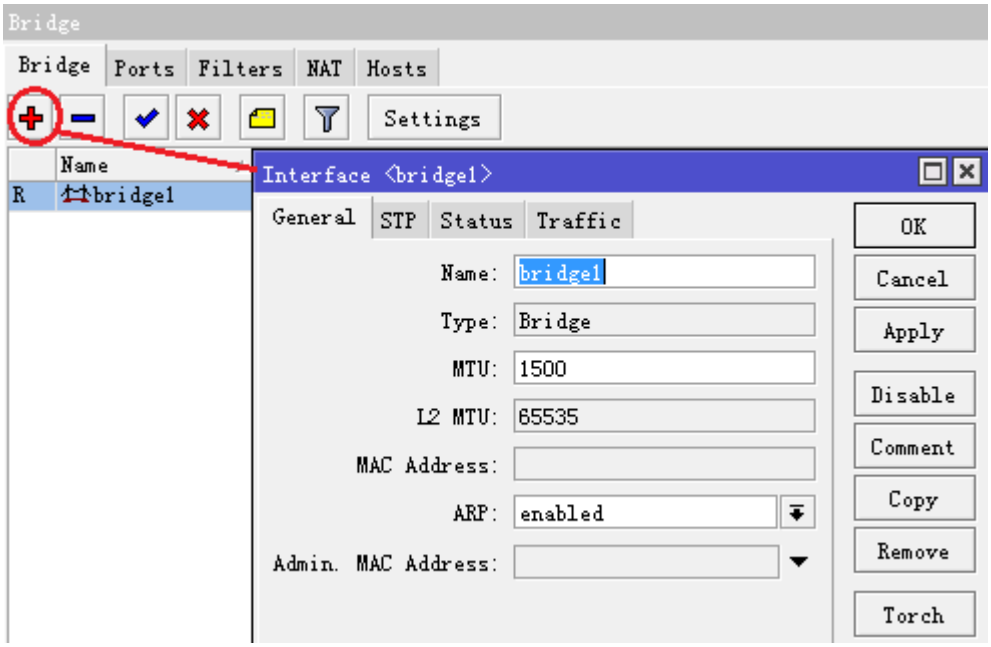
1. 添加一个桥接口，命名为 **MyBridge**：

```
/interface bridge add name="MyBridge" disabled=no
```

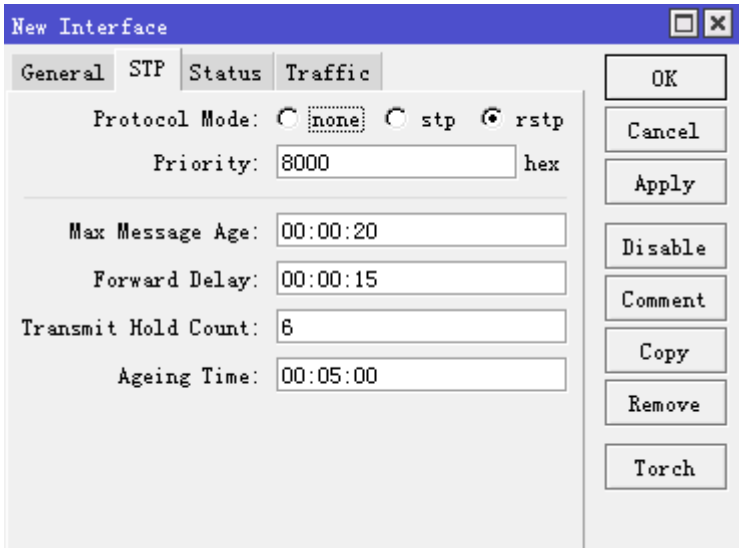
2. 把 **ether1 和 ether2** 添加到 **MyBridge** 接口：

```
/interface bridge port add interface=ether1 bridge=MyBridge
/interface bridge port add interface=ether2 bridge=MyBridge
```

用 winbox 进入 bridge 菜单，添加并启用一个网桥：

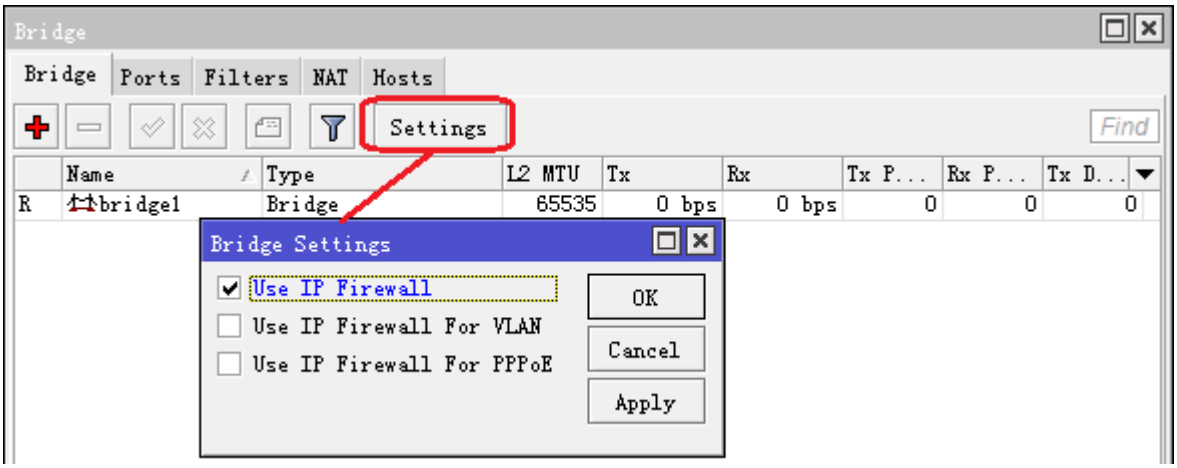


当创建一个 bridge 时，在菜单中可以选择 STP 配置，在该选项中可以选择启用 STP 或 RSTP：



Bridge setting

这个参数可以选择是否启用 ip firewall 的三层过滤规则，这个功能有别于 2.9 的桥接功能，如果关闭三层的 ip firewall 过滤，可以大大提高 RouterOS 的网桥转发率，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网桥过滤时，包括 ip firewall filter、mangle、nat 和 queue 流控等，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>
```

当然网桥由于打开了三层过滤，转发率会受到一定的影响，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` 中的数据包进行处理，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。

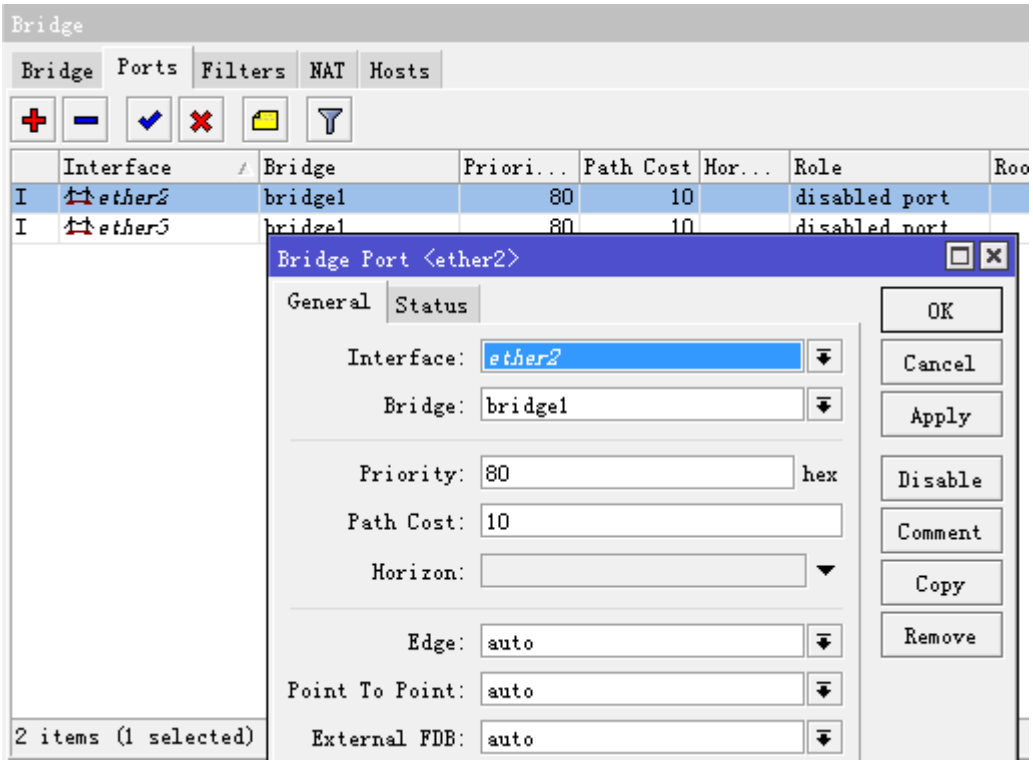
属性描述

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)，我们都以 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***

Host 是只读查看，用户查看各个接口上学习到的 MAC 地址

属性描述

age (*只读:时间*) - 从主机获得最后一个包开始的时间
bridge (只读*: 名称*) - 属于词条（entry）的桥
local (只读*: 标志*) - 主机词条是否是桥本身的
mac-address (只读*: MAC 地址*) - 主机 MAC 地址
on-interface (只读*: 名称*) - 主机所连接的桥接的接口

host 中获得活动的主机 MAC 列表和对应接口：

```
[admin@MikroTik] interface bridge host> print
Flags: L - local, E - external-fdb
    BRIDGE              MAC-ADDRESS       ON-INTERFACE       AGE
    bridge1             00:00:B4:5B:A6:58 ether1             4m48s
    bridge1             00:30:4F:18:58:17 ether1             4m50s
  L bridge1             00:50:08:00:00:F5 ether1             0s
  L bridge1             00:50:08:00:00:F6 ether2             0s
    bridge1             00:60:52:0B:B4:81 ether1             4m50s
    bridge1             00:C0:DF:07:5E:E6 ether1             4m46s
    bridge1             00:E0:C5:6E:23:25 prism1             4m48s
    bridge1             00:E0:F7:7F:0A:B8 ether1             1s
[admin@MikroTik] interface bridge host>
```

14.2 Bridge 防火墙

操作路径: ***/interface bridge filter***,

桥防火墙执行包过滤因此提供了用于管理数据流进，流出和流经桥的安全功能。

注：在桥接接口之间的数据包就像其他 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（进入桥设备的数据包，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包的源/目的 MAC 地址的方法。它有连条内置的链：
  - **scnat** - 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包
  - **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）中设置数据包标记，就像用 mangle 在 IP 防火墙中设置数据包标记一样。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（目的文件服务访问点） 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 (数据包类型)
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) - 数据包进入的桥接口
in-interface (名称) - 数据包进入的物理接口（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 协议(仅当 MAC 协议设置为 IPv4)
- **ipsec-ah** - IPsec AH 协议
- **ipsec-esp** - IPsec ESP 协议
- **ddp** - 数据报投递协议
- **egp** - 外部网关协议
- **ggp** – 网关-网关协议
- **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 通用包协议
- **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包
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包
- **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包

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包回应而设置的
stp-forward-delay (*time*: 0..65535) - forward delay timer 转发延迟计时器
stp-hello-time (*time*: 0..65535) - stp hello 数据包时间
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包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**) - 如果数据包匹配了其中一个规则就采取动作：
  accept - 接受包，无动作。例如：数据包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 变量指定的链
  log - 记录数据包
  mark – 标记数据包以便后面使用
  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*) - 数据包离开桥的接口

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**) - 如果数据包匹配了其中一个规则就采取动作：
- **accept** - 接受包，无动作。例如：数据包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** – 悄然丢弃数据包 (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 变量指定的链
- **log** - 记录数据包
- **mark** – 标记数据包以便后面使用
- **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** - 把数据包重新定位到桥本身（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

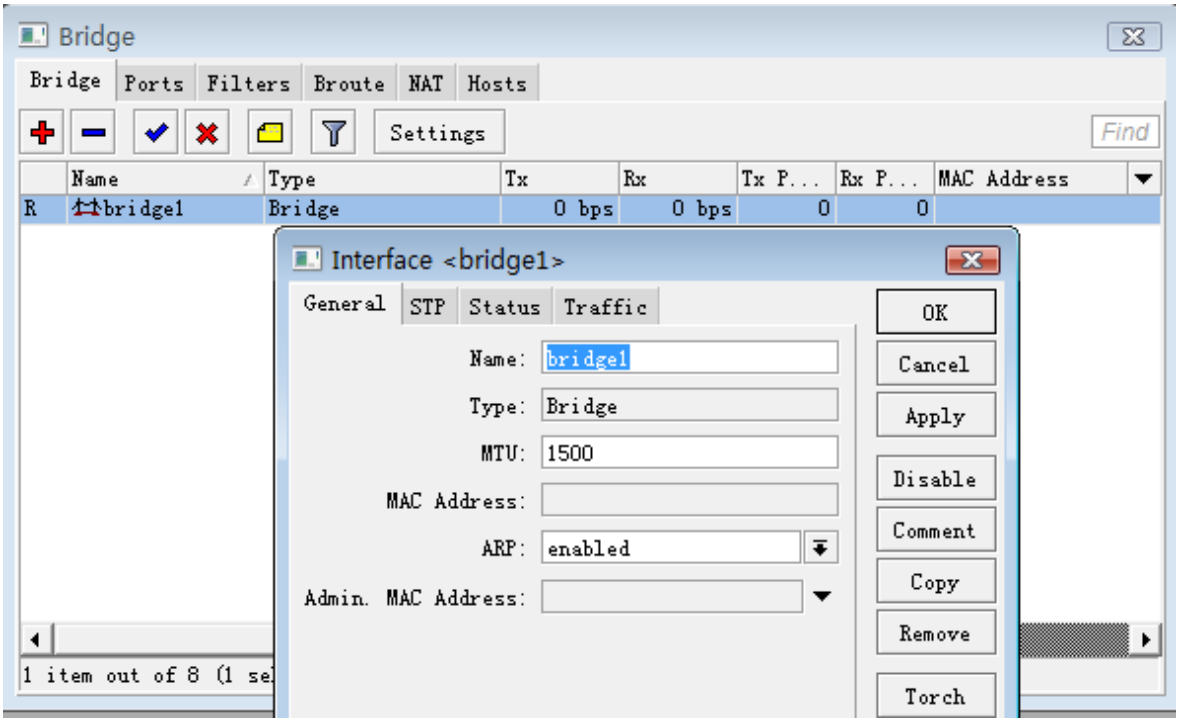
故障分析

- **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

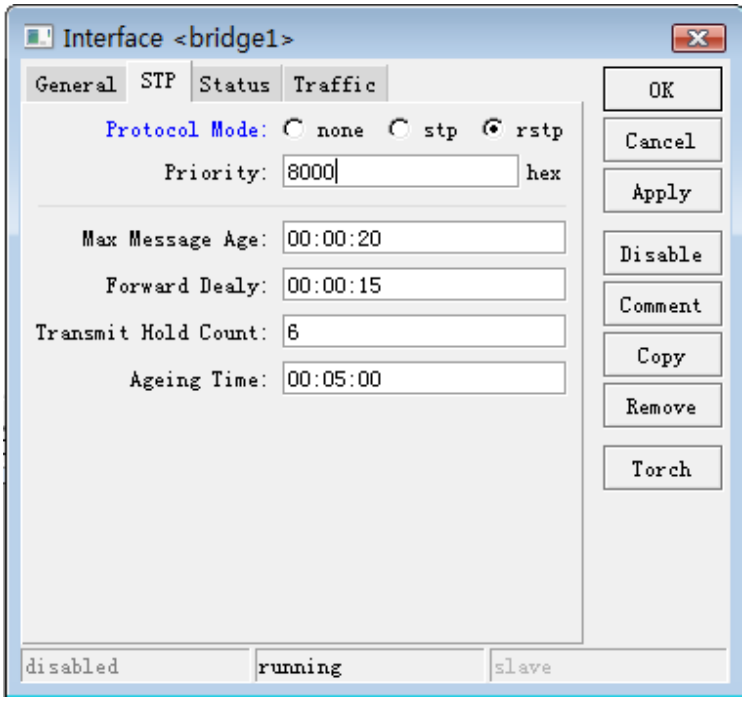
14.3 Bridge 实现二层端口隔离

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。

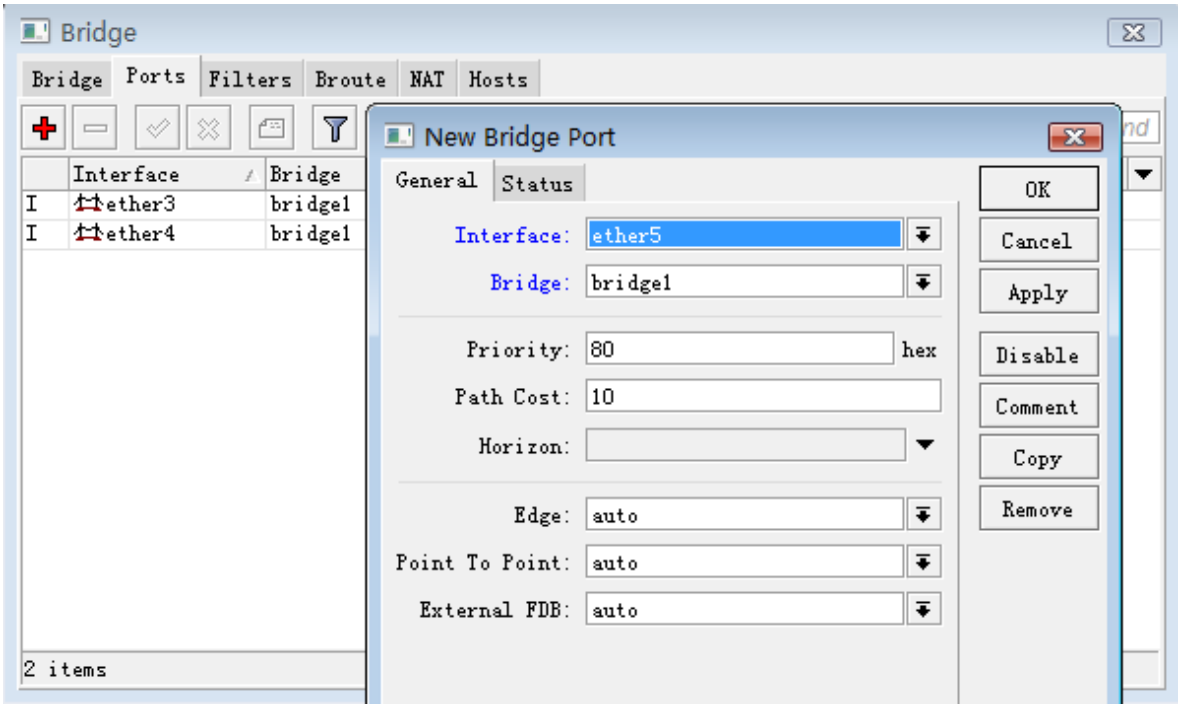
在这里我们通过 RB450 的操作为实例，配置二层端口隔离。首先我们在 Bridge 中添加一个网桥 bridge1：



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：

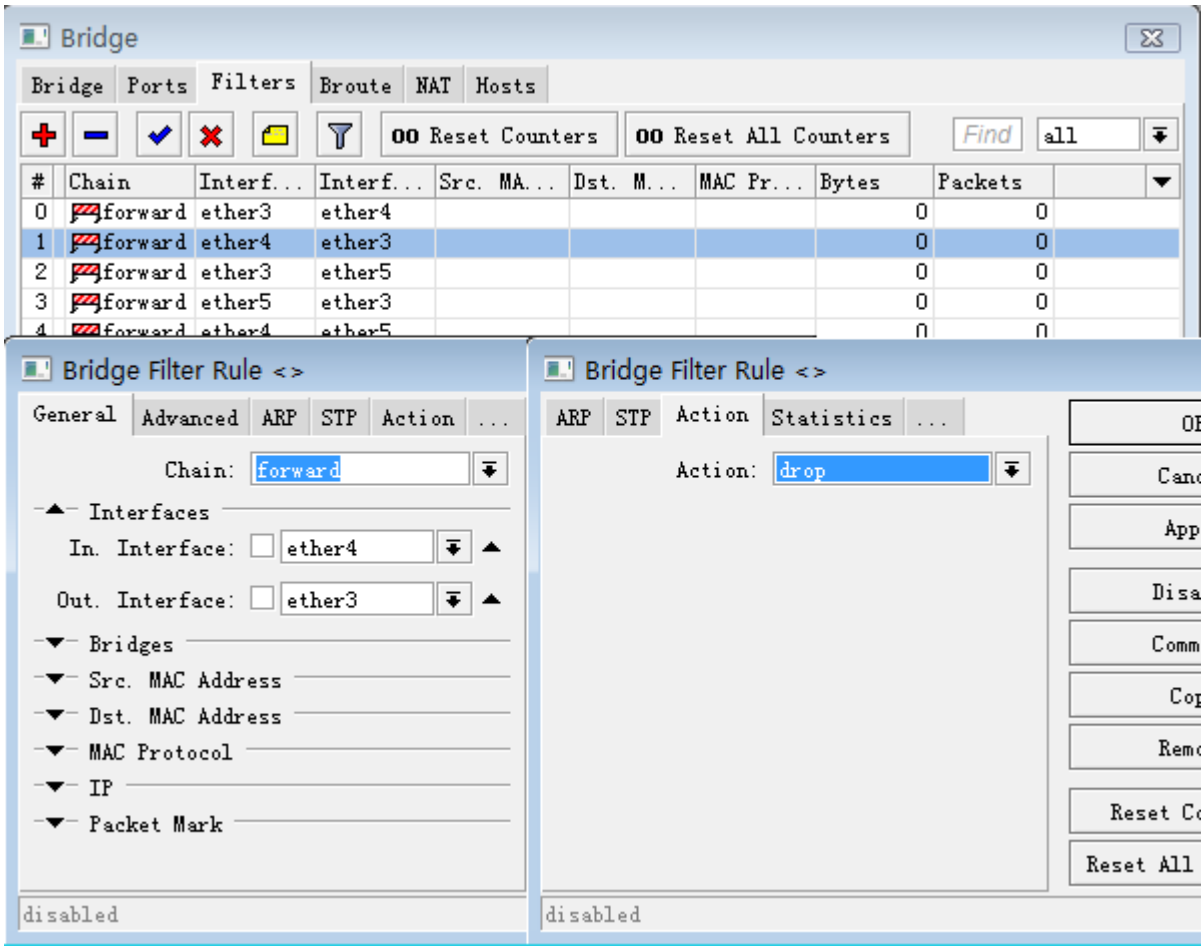


添加完桥接功能后，需要将对应的网卡添加入 bridge1 中，进入 Port 中设置，我们将 3 个网卡 ether3、ether4 和 ether5 一个一个添加到 bridge1 中：

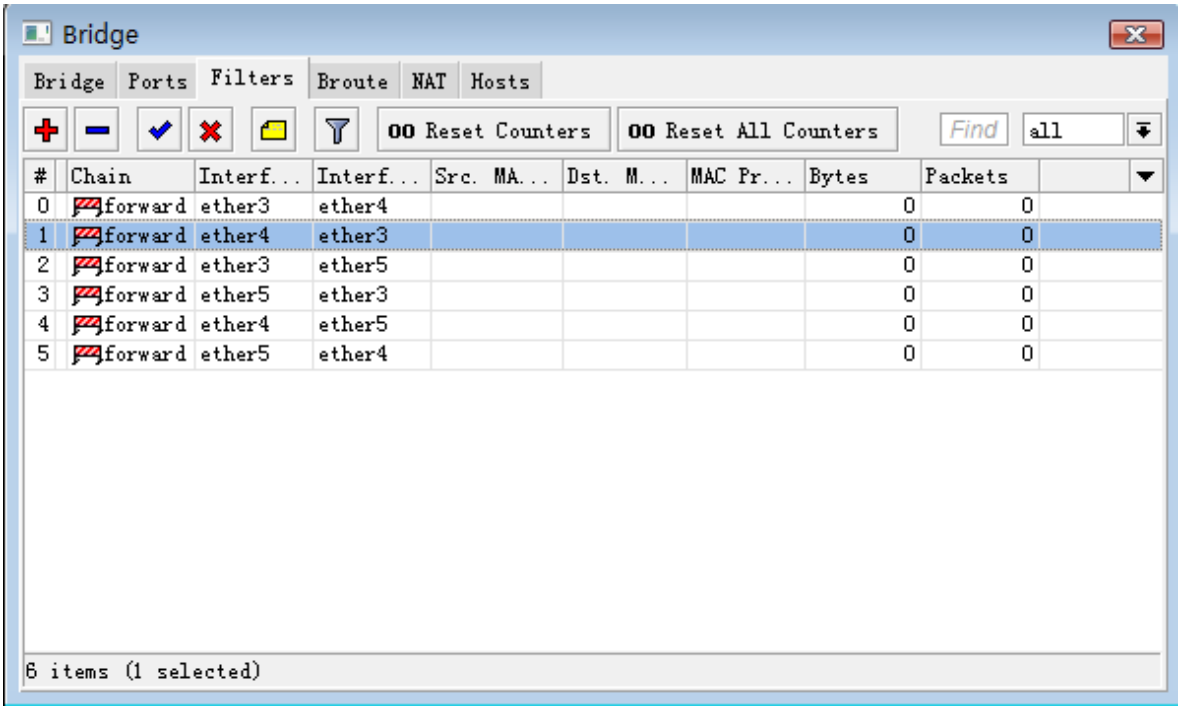


添加完每个端口后，现在 RB450 的 3 个以太网口，就完成了桥接的设置，这样 3 个口就实现了二层的交换功能。

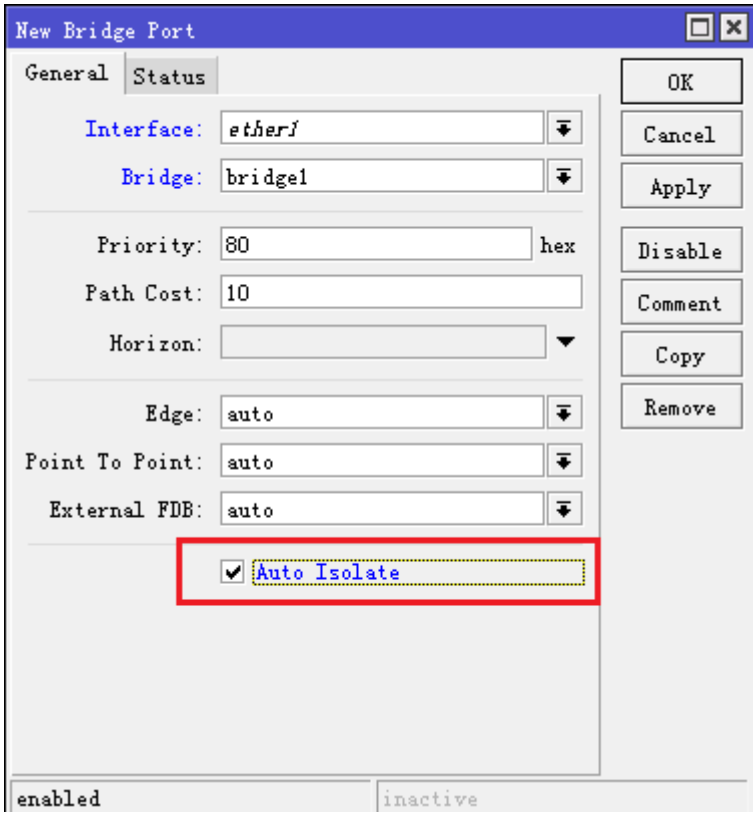
这里我们禁止 ether3、ether4 和 ether5 进行通信， 我们进入 filter 中设置防火墙过滤规则，我们首先配置 ether3 与 ether4 的数据隔离我们在 interface 选项中设置 In-interface 和 Out-interface（In-interface 为数据进入的网口，Out-interface 为数据出去的网口，数据是双向传输的，两个接口需要做两条规则），然后选择 action 设置 action 参数为 drop，丢弃数据：



下面是设置好的状态：



注意：RouterOS v6.19 后增加了自动隔离属性，当选择 auto Isolate 后，将自动对该接口与同一个桥接下的其他接口进行隔离，因此 v6.19 后可以选择更加简单的方式进行隔离



14.4 如何建立一个透明传输整形器

你想用在一个以太网中做一个 **MikroTik RouterOS** 透明传输整形器。你可以在两个网络中间加入。要达到这样 RouterOS™ 应该如下配置（这里假设为没有其他配置在整形器上，并且安装了两张以太网卡）：

1. 启用并命名以太网卡。连接到内部网络的网卡命令为 **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
```

2. 让我们假设 10.0.0.1 的 IP 地址是网关。那我们添加 IP 地址为 **10.0.0.2/24** 到相应的网卡上(以后你将需要这个地址远程配置整形器)，设置好后你可以通过 ping 来检查你的网关。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）**注：**如果一个都没有工作，可能在网关上设置了防火墙策略或是地址绑定，先暂时删除它们再试一次。

```
/ip address add interface=ext address=10.0.0.2/24
```

3. 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
```

注： 现在前面设置的 IP 地址应被改变到 **bridge** 接口上：

```
/ip address set [/ip address find] interface=bridge
```

现在你可以简单的添加期望的队列。**注：**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
```

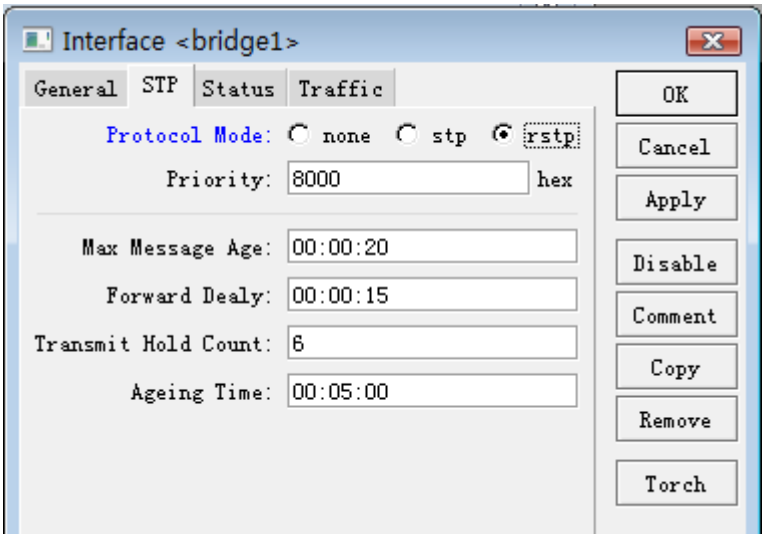
当然你也可以使用 Manlge 标记 ext 和 int 接口的数据包，通过 PCQ 进行流控。

14.5 通过 Bridge Filter 控制 MAC 地址

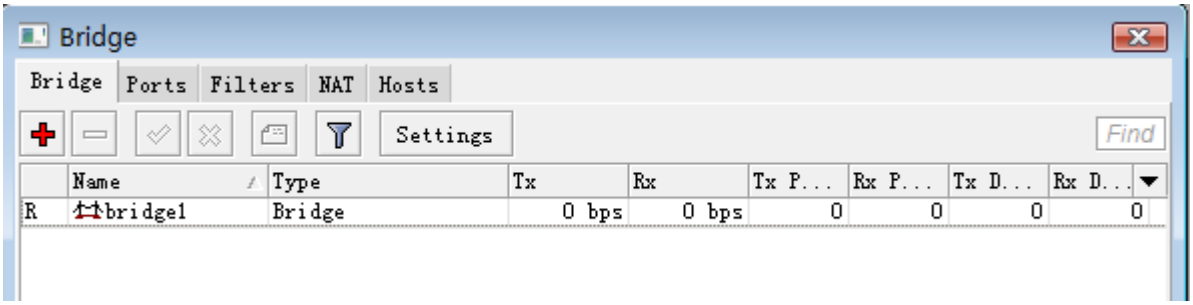
通过 bridge filter 控制 MAC 地址，如当我们把 RouterOS 设置为透明桥时，可以控制网络内的主机 MAC 通讯，这样我们可以从二层上控制客户端 PC。

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

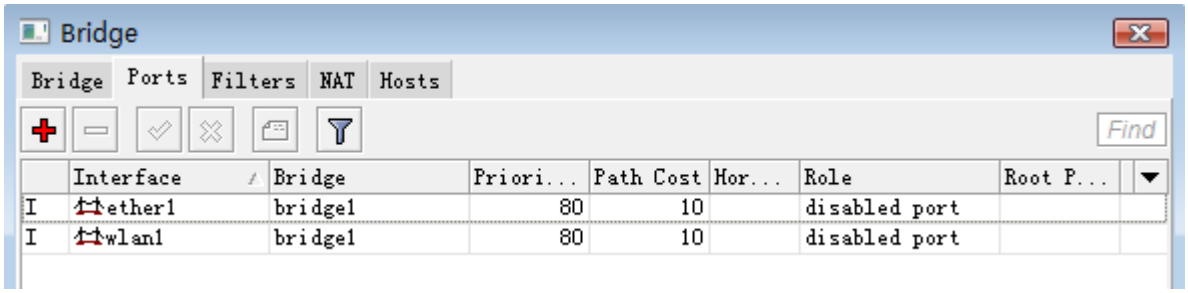
1、添加一个 bridge，默认的 bridge 的名称为 bridge1，并设置 RSTP（快速生成树协议）模式：



添加完 bridge 后，我们可以在 bridge 列表中查看到：



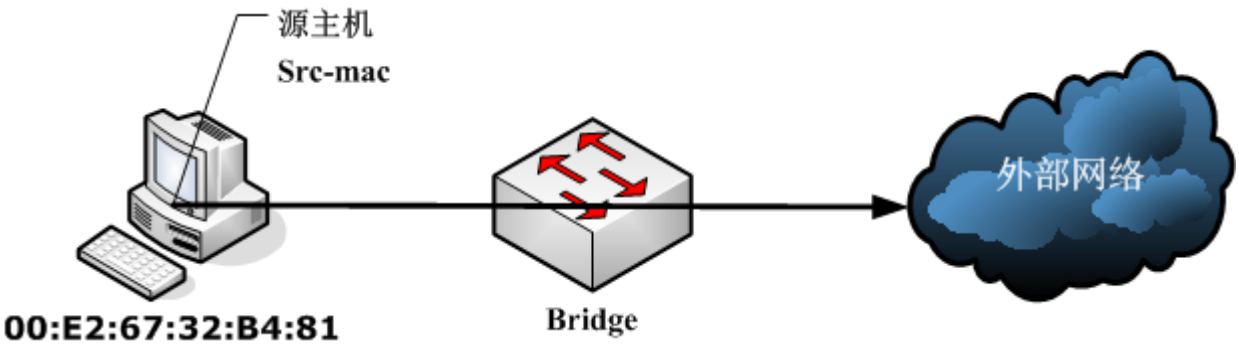
2、我们将 ether1 和 wlan1 网络接口添加到 bridge1 里，这样 2 个网络接口就实现了桥接功能



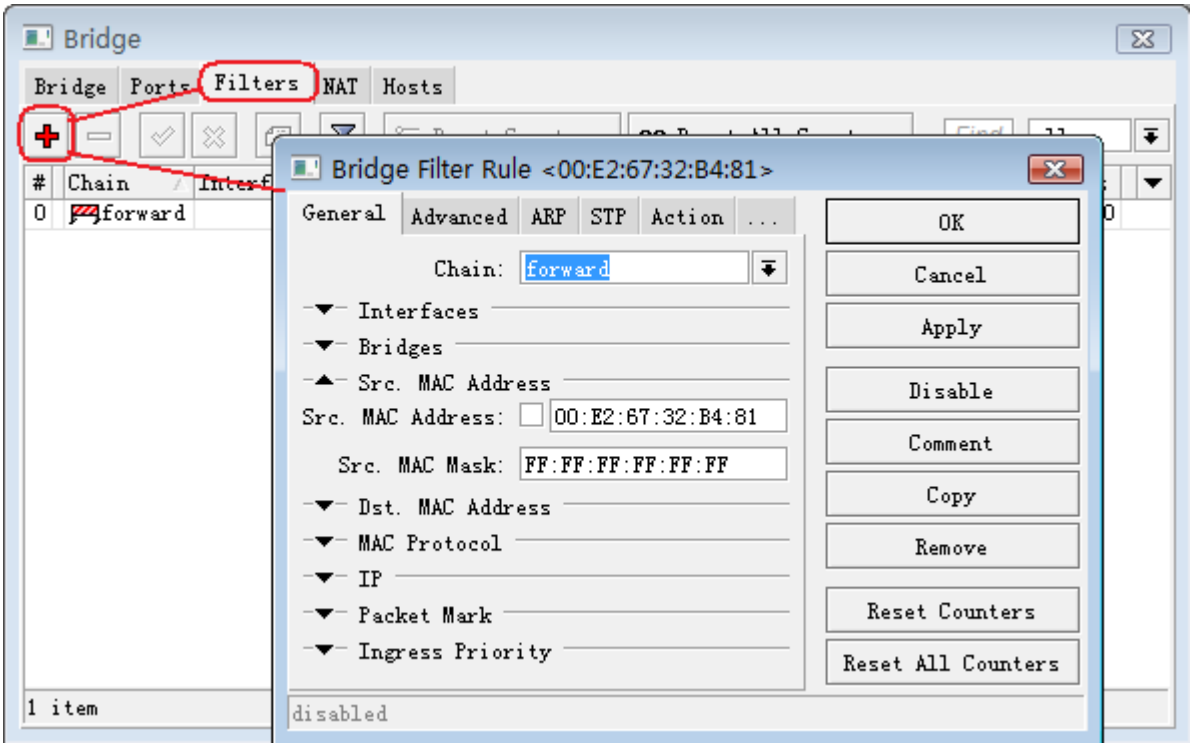
过滤源和目标 MAC 地址

1、源 MAC 地址过滤

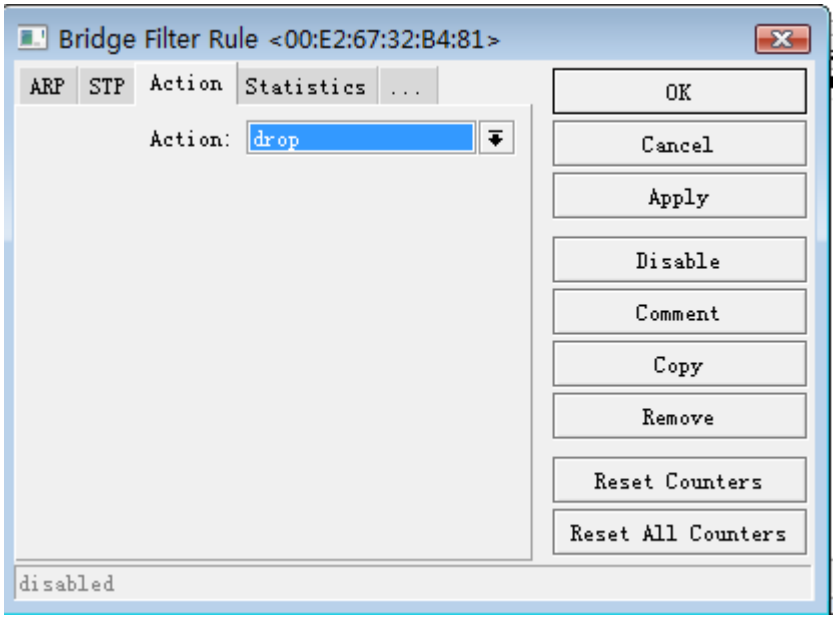
在设置完基本的 bridge 后，我们进入 bridge filter 中配置桥防火墙过滤，首先我们需要对指定的一台 PC 的 MAC：00:E2:67:32:B4:81 地址做过滤，不允许与 bridge 的外部网络连接，如下图：



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：



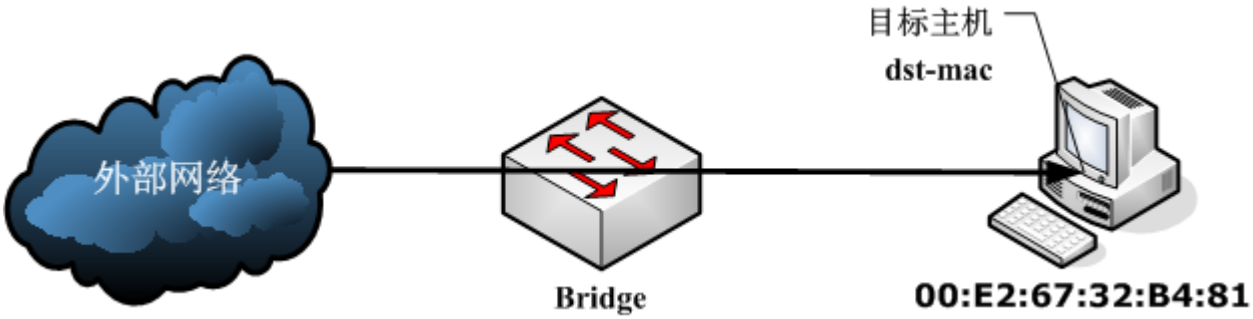
接下来选择 Action 为 drop，丢弃该 MAC 地址发出的数据：



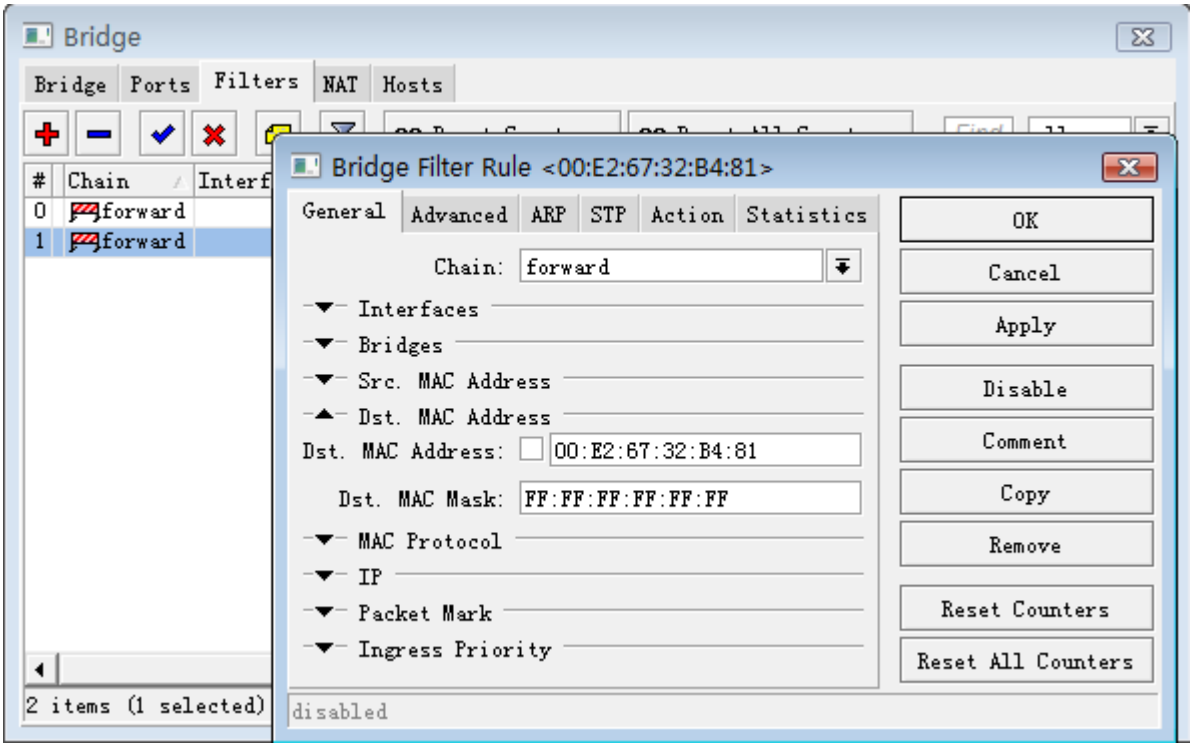
注：我们设置 src-mac-address 时，后面跟着 MAC 掩码，这个掩码和我们 IP 层的子网掩码类是，只是 MAC 掩码是按照十六进制换算，十六进制的 FF 与 IP 掩码的 255 是相同，规定网络范围，因为这里是过滤一个台主机的 MAC 地址，所以我们设置 MAC 子网掩码为 FF:FF:FF:FF:FF:FF。

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

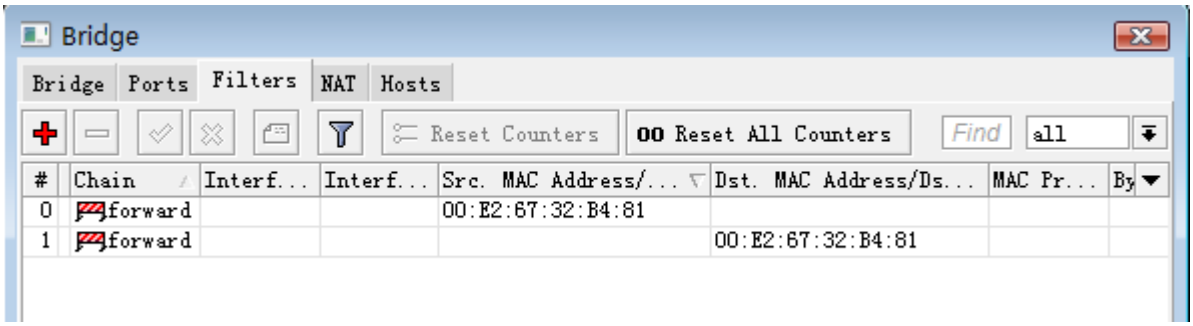


我们添加目标 MAC 地址过滤规则，选择 dst-mac-address=00:E2:67:32:B4:81，dst-mac-address 默认为全 FF。



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

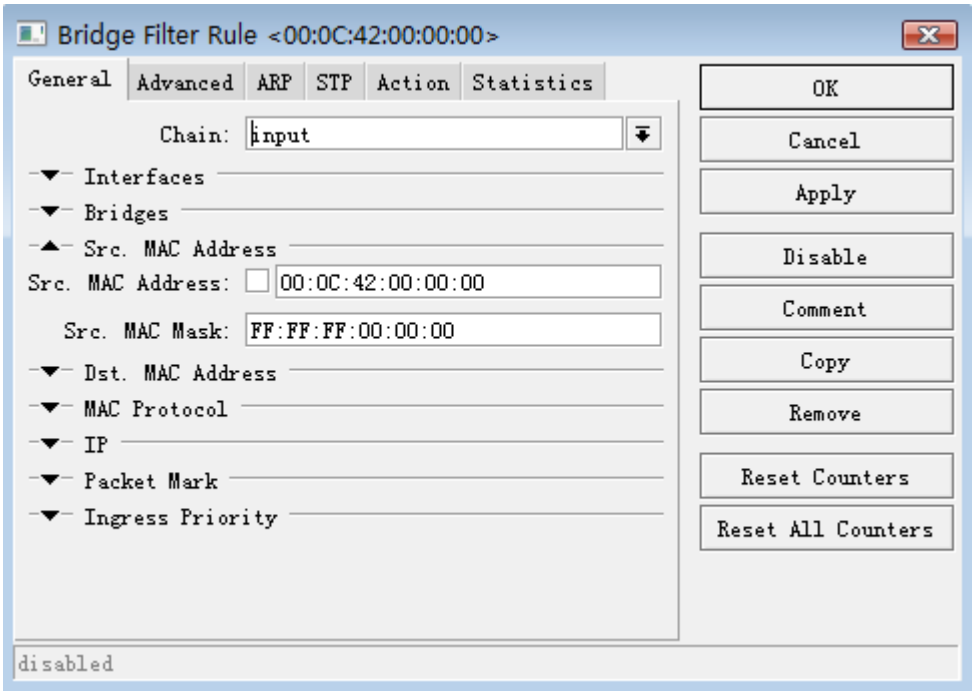
我们知道所有的网络设备都有一个 6 位的 MAC 地址，前 3 位为生产厂商标示，后 3 位为设备编号，当我们在做无线网桥的时候，只允许特定某一厂商的网卡连接到 RouterOS，可以通过 Bridge 的防火墙控制 MAC 地址，限制某一类的 MAC 不能连接到 RouterOS 设备，或者通过 RouterOS 设备。

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。

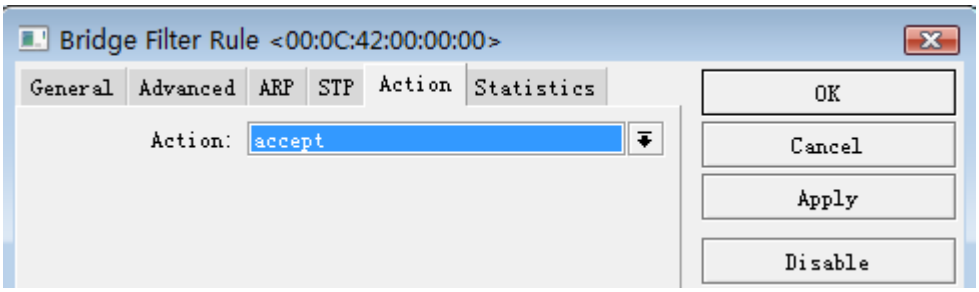
在设置为 bridge 的接口参数后，我们在 filter 中配置 2 条 input 规则，限制除了 MAC 地址前 3 位是 00:0C:42 能连接 RouterOS，其他的都拒绝掉。

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。

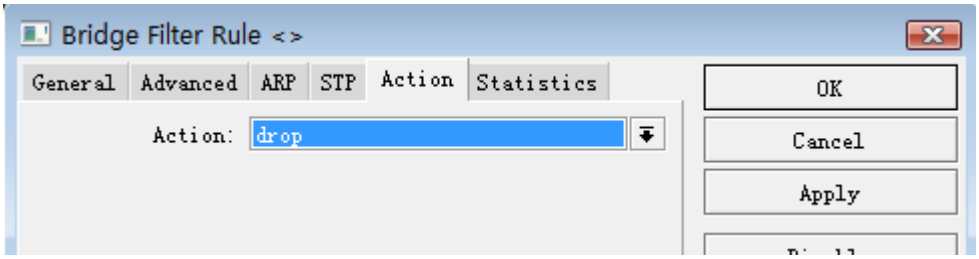
第一条规则，设置 src-mac-address=00:0C:42:00:00:00/src-mac-mask=FF:FF:FF:00:00:00



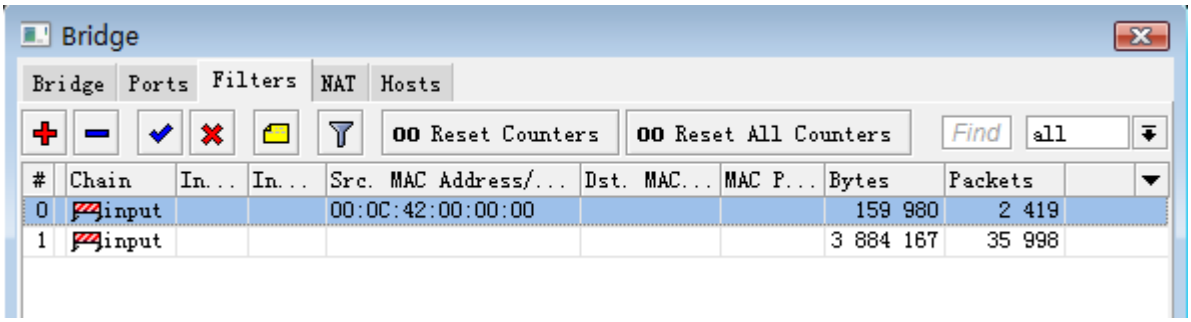
在 action 中选择 accpet 接受，数据通过



第二条规则，是丢弃其他所有的 MAC 数据



配置完成后，如下：


Bridge
Bridge
Ports
Filters
NAT
Hosts
00 Reset Counters
00 Reset All Counters
Find
all
Chain In... In... Src. MAC Address/... Dst. MAC... MAC P... Bytes Packets
0 input 00:0C:42:00:00:00 159 980 2 419
1 input 3 884 167 35 998

第十五章 VLAN

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。

你可以使用 MikroTik RouterOS (与 Cisco IOS、华为、Linux 和其他路由系统一样)都能处理标记 VLAN 数据包，VLAN 可以用于任何的网络中，没有任何限制，能成功通过以太网的桥接，你同样能将 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。

规格

功能包要求: ***system***
等级要求: *Level1 (limited to 1 vlan)* , *Level3*
子目录要求: ***/interface vlan***
标准与技术: VLAN (IEEE 802.1Q)

当前支持的以太网接口

这是一个 VLAN 经过测试并能工作的网络接口列表。注意也存在很多其他支持 VLAN 的接口，但它们并没有被检测。

- Realtek 8139
- Intel PRO/100
- Intel PRO1000 server adapter
- National Semiconductor DP83816 based cards (RouterBOARD200 onboard Ethernet, RouterBOARD 24 card)
- National Semiconductor DP83815 (Soekris onboard Ethernet)
- VIA VT6105M based cards (RouterBOARD 44 card)
- VIA VT6105
- VIA VT6102 (VIA EPIA onboard Ethernet)

这是一个 VLAN 经过测试并能工作的网络接口列表，**但不支持大数据包（>1496 字节）**：

- 3Com 3c59x PCI
- DEC 21140 (tulip)

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

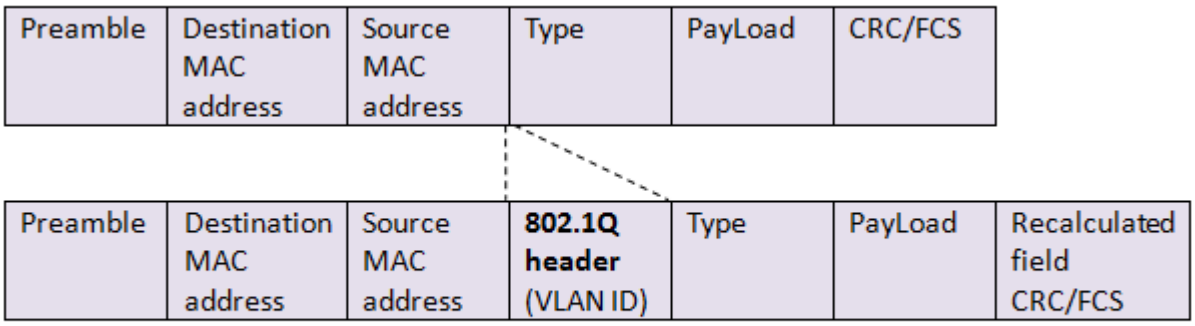
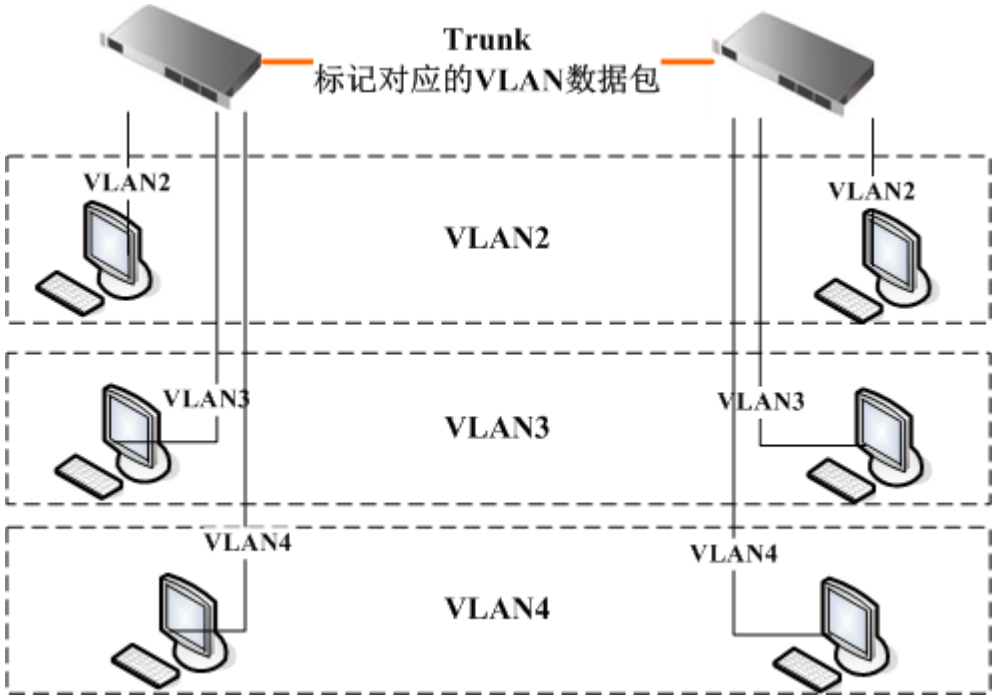


Figure 12.1. Insertion of 802.1Q Tag (VLAN ID) in Ethernet-II frame

每个 VLAN 被当做独立的子接口，即主机被指定到一个 VLAN 后，虽然他们在同一个交换机下，但不能连接其他 VLAN 的主机。因此你要实现 VLAN 之间的互访，你需要一台路由器，每一个 VLAN ID 对应一个独立的接口

当多个 VLAN 分部在多个交换机时，交换机内部的连接将必须使用 trunk，数据包将被打上属于自己 VLAN 标签，一个 trunk 装载多个 VLAN 的传输，如同点对点连接装载着打上标签的数据在交换机与交换机或路由器之间传输



Q-in-Q

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，在 RouterOS 里能被配置在一个 VLAN 接口上添加其他的 VLAN，如下外层 VLAN 是 11，内层 VLAN 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
```

如果数据包发向 vlan2 接口，2 个 vlan 标签将会被添加到以太网的包头里- "11" 和 "12"。

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

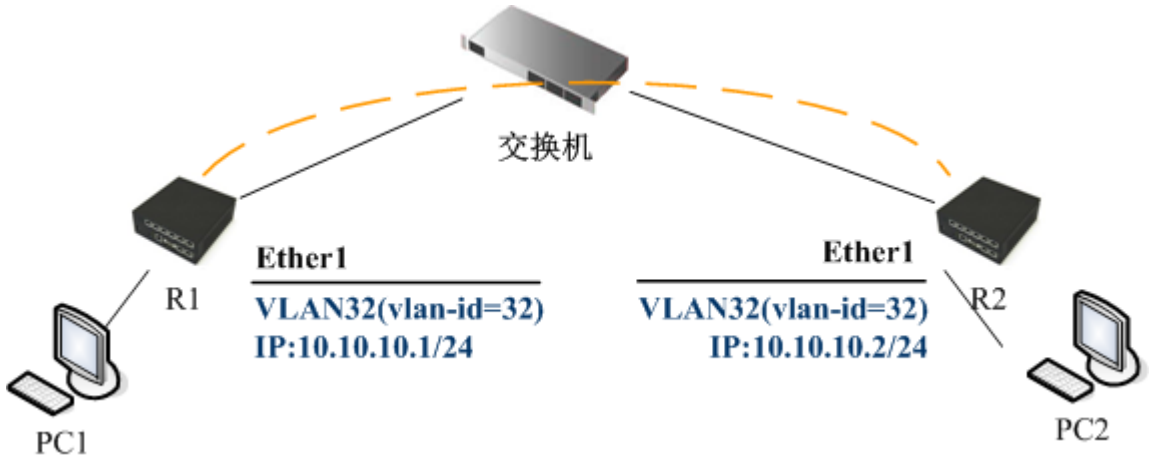
注：MTU 必须像在以太网接口那样设置为 1500 字节。但这样也可能不能与一些不支持接受/传输满长度带有 VLAN 标题的以太网数据包的以太网卡一起工作（1500 字节数据+4 字节 VLAN 标题+14 字节以太网标题）。这种情况下使用 MTU1496，但要注意如果较长的数据包要在接口发送的话这会引起数据包的分割。同时要记得如果路径 MTU 搜索在源和目的间不能正常工作，MTU1496 可能引起一些问题。

在接口 ether1 添加并启用名为 test 且 vlan-id=1 的 VLAN：

```
[admin@MikroTik] interface vlan> add name=test vlan-id=1 interface=ether1
[admin@MikroTik] interface vlan> print
Flags: X - disabled, R - running
 #      NAME                      MTU   ARP          VLAN-ID INTERFACE
 0 X    test                     1500 enabled     1          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 test                     1500 enabled     1          ether1
[admin@MikroTik] interface vlan>
```

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:

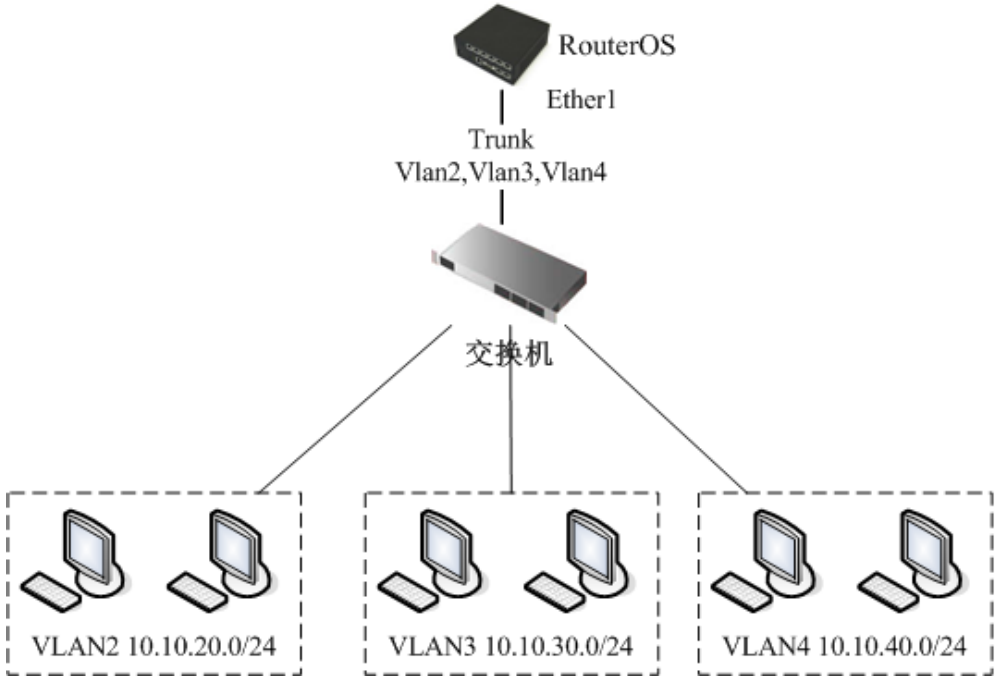
```
[admin@MikroTik] ip address> /ping 10.10.10.1
10.10.10.1 64 byte pong: ttl=255 time=3 ms
10.10.10.1 64 byte pong: ttl=255 time=4 ms
10.10.10.1 64 byte pong: ttl=255 time=10 ms
10.10.10.1 64 byte pong: ttl=255 time=5 ms
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 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 数据包，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
```

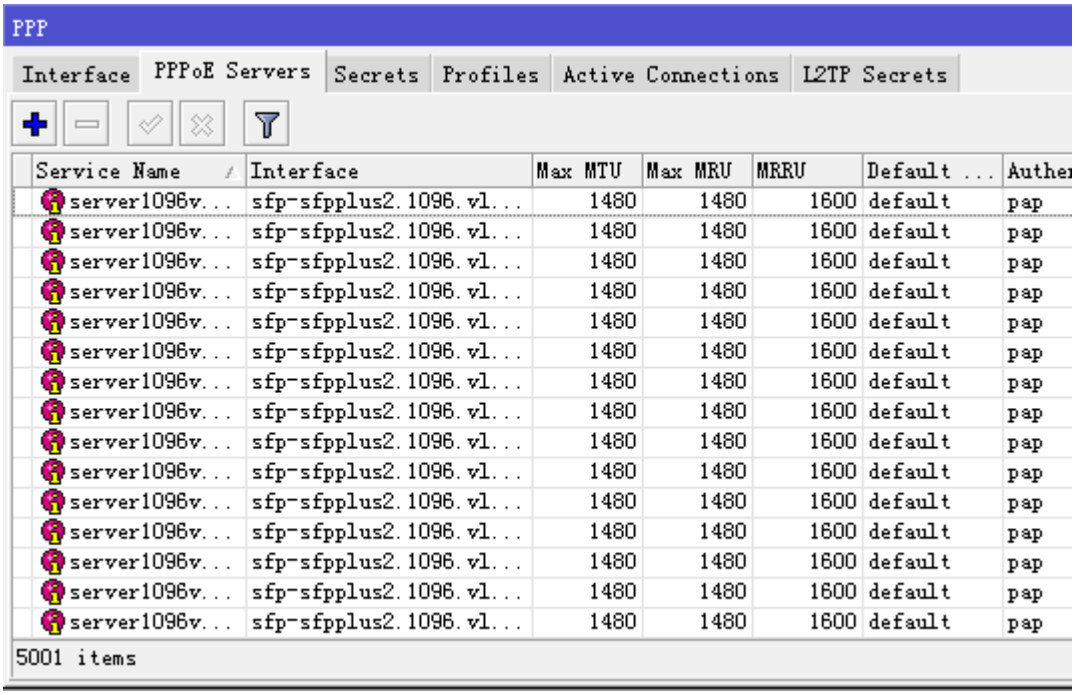
添加每个 VLAN 的 IP 地址:

```
/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
```

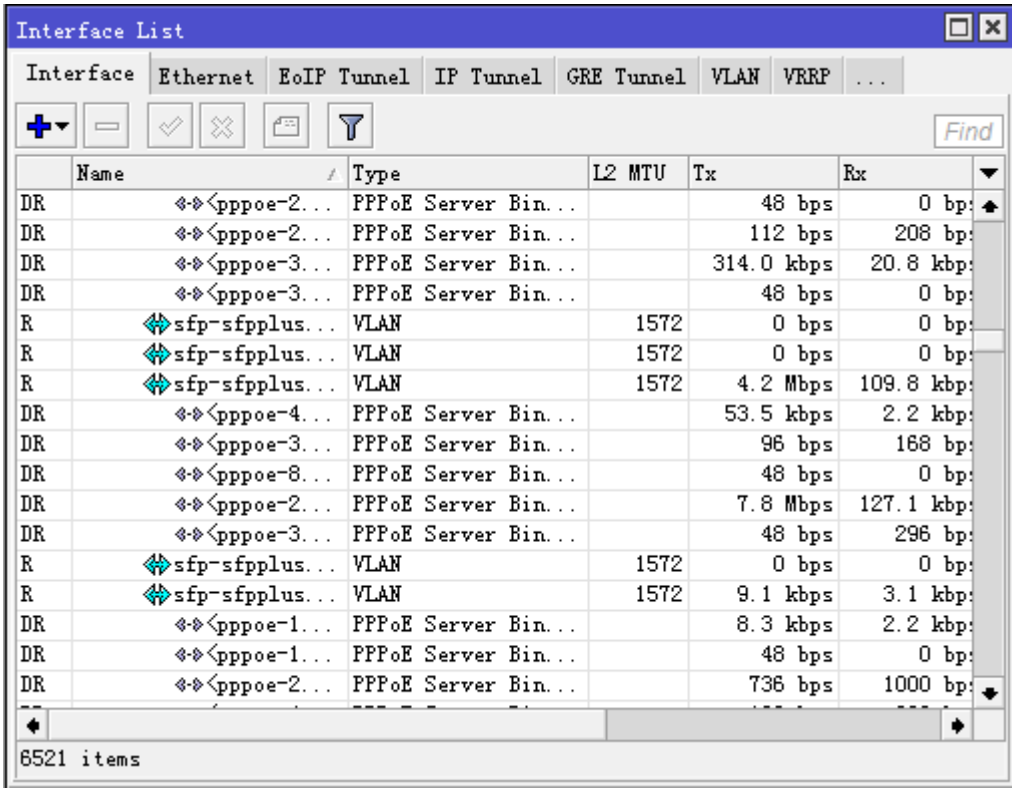
当然这种 VLAN 方式也可以应用于 Hotspot 认证，只是需要在每个 vlan 上设置 IP 地址。

15.4 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：



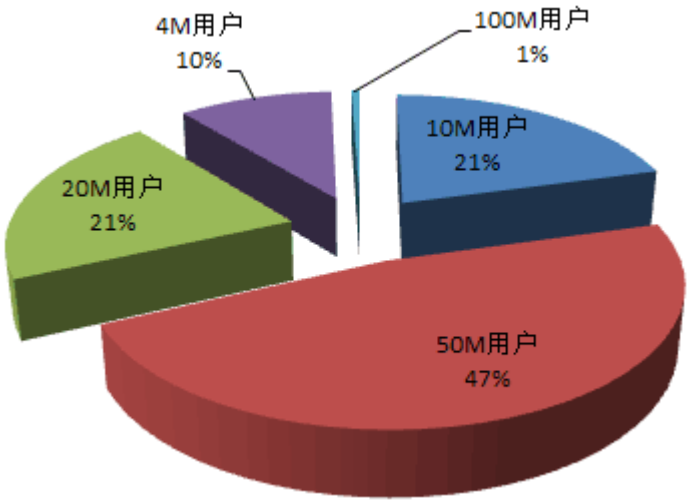
下面是在 interface 中独立 VLAN 下的 PPPoE 运行情况：



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。

下面是基于三款设备的 PPPoE 性能的测试实例情况，供大家参考：

首先网络环境是运营商级别，用户账号带宽分布比例如下图：



测试结果：

	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
运用环境	PPPoE server 5000 条 在线 571 人，CPU 平均 18%，流量 448Mb 在线 919 人，CPU 平均 15%，流量 285Mb	PPPoE server 5000 条 在线 1380 人，CPU 平均 9%，流量 288Mb 在线 1740 人，CPU 平均 40%（短时间 70%），流量 1.1Gb	PPPoE server 5000 条 在线 463 人，CPU 平均 30%，流量最高 412Mb
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
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	转发和软件优化比较出色 流量增长 CPU 变化不大	IRQ 网卡对应与实际网卡有差别 流量增长 CPU 增长较明显

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
   }
}
```

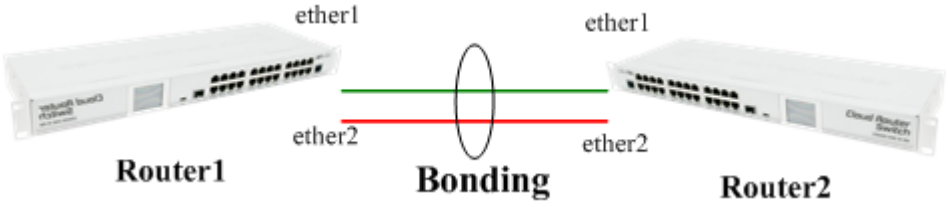
第十六章 Bonding

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。

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

让我们假设两台路由器（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
```

注：bonding 接口需要几秒钟时间的连通时间。

规格

需要功能包: ***system***
需要等级: *Level1*
操作路径: ***/interface bonding***

提供了最佳的失效转移管理，你需要指定 **link-monitoring** 参数:

- MII (媒体独立接口 Media Independent Interface) type1 or type2 - 媒体独立接口是一个在操作系统与 NIC 之间的理论层，探测连接是否运行（执行可以通过其他功能实现，但在我们的事例中这个是非常重要的）。
- ARP – 地址解析协议（通过 **arp-interval** 时间）检测连接状态。

link-monitoring 被用于检测是否连接。

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 协议
enabled – 启用 ARP 协议
proxy-arp – 使用 ARP 代理功能
reply-only – 将只回应/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。

16.2 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

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 安装普通的 access 口配置，加入 vlan8，无需任何特殊配置

```
[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]
```

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

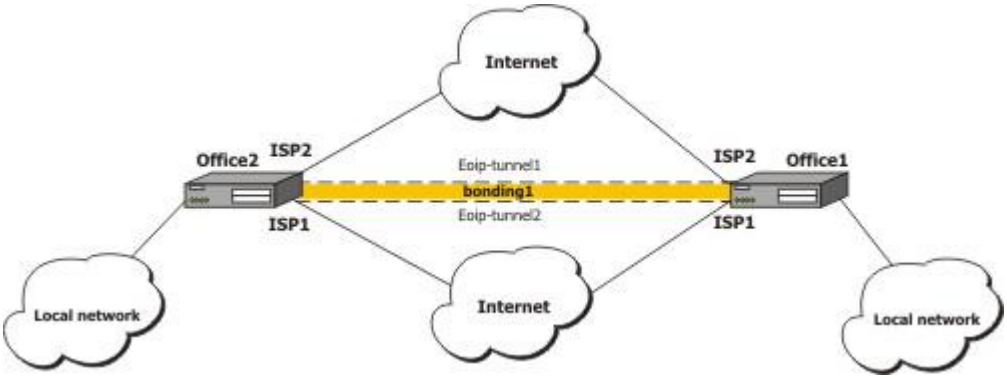
```
[5700]interface vlan 8
[5700-vlanif8]ip address 172.16.0.2 24
[5700-vlanif8]quit
```

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]
```

16.3 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，并提供失效管理。



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 设置请参阅本人编写的 RouterOS 无线手册

第十七章 虚拟路由冗余协议(VRRP)

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。

利用 VRRP 聚合功能提供高效的路由器运行方式，不在需要复杂的脚本 ping 监测

需要功能包: ***system***
软件等级: *Level1*
操作路径: ***/interface vrrp***

在一个网络中最大可用支持相同 VRID（虚拟路由 ID）255 个。每个路由器都必须设置一个优先值，每个 VRRP 配置通一个虚拟的网卡绑定在一个真实的网卡上。VRRP 地址放入虚拟的 VRRP 网卡上。VRRP **Master** 状态显示为 **running** 标志，虚拟网卡上的地址被激活，其他属于 **backup**（即优先级低的 VRRP 路由）停止运行。

虚拟路由冗余协议是一种为路由提供高效率的路由选择协议。一个或多个 IP 地址可以分配到一个虚拟路由上，一个虚拟路由节点应该具备以下状态：

- **MASTER** 状态，一个节点回答所有的请求给相应请求的 IP 地址。仅只有一个 MASTER 路由器在虚拟路由中。每隔一段时间这个主节点发出 VRRP 广播包给所有 backup 路由器。
- **BACKUP** 状态, VRRP 路由器监视 Master 路由器的状态。它不会回答任何来至相应 IP 地址的请求，当 MASTER 路由器无法工作时（假设至少三次 VRRP 数据连接丢失），选择过程发生，新的 MASTER 会根据优先级产生。

注：VRRP 不能运行在 VLAN 接口上，VLAN 的接口 MAC 地址于与运行在物理网卡 MAC 地址是不同的 。

17.1 VRRP 路由

操作路径: ***/interface vrrp***

属性描述

arp (disabled | enabled | proxy-arp | reply-only;默认: **enabled**) – 地址解析协议 Address Resolution Protocol
authentication (none | simple | ah; default: **none**) – 使用 VRRP 消息数据包的验证方式。
none – 没有证明
simple – 纯文本验证
ah – 验证头使用 HMAC-MD5-96 算法
backup (*只读: flag*) – 是否为备份状态
interface (*name*) – 运行接口的名称
interval (*整型*: 1..255; 默认 t: **1**) – VRRP 状态更新间隔秒钟。定义多少频率发送 VRRP 信息数据包。
mac-address (*MAC address*) – VRRP 的 MAC 地址 address。符合 RFC 协议，任何 VRRP 都应该只有唯一的 MAC 地址。
master (*只读: flag*) – 是否为 master 状态

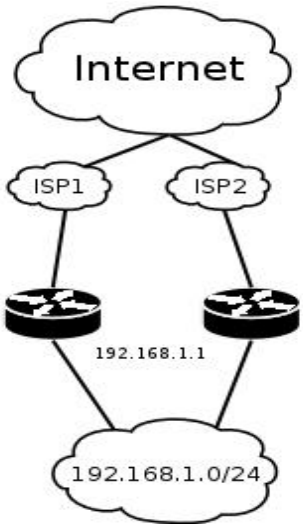
mtu (*整型*; 默认: **1500**) – 最大传输单位
name (*name*) – VRRP 分配的名称
on-backup (*name*; 默认: **""**) – 当节点为 backup 状态执行的脚本
on-master (*name*; 默认: **""**) -当节点为 master 状态执行的脚本
password (*文本*; 默认: **""**) – 需要验证时的密码，不使用验证时可以被忽略。8 位字符长文本字符串（为纯文本验证方式）；16 位字符长文本字符串（为需要 128 位 key 的 AH 验证）
preemption-mode (yes | no; 默认: **yes**) – 是否启用优先模式。
no – 一个 backup 节点在当前的 master 失效之前，是不会选择 master ，即使该 backup 的优先高于当前 master 的级别
yes – 该节点总是拥有最高优先级。
vrid (*整型*: 0-255; 默认: **1**) – 虚拟路由的身份号(必须是在 interface 上是唯一的)
priority (*整型*: 1-255; 默认: **100**) – 当前节点的优先级(高的数值代表高的优先级)

注:所有同一个集合的节点,必须使相同的 vrid, interval, preemption-mode, authentication 和 password. 第 255 的优先级被保留为真正的虚拟路由的主机 IP 地址。

添加一个 VRRP 事例在 ether1 的接口上，一个虚拟路由的 **vrid** 设置为 **1**，因为是虚拟路由的主机，所有优先级为 255：

```
[admin@MikroTik] interface vrrp> add interface=ether1 vrid=1 priority=255
[admin@MikroTik] interface vrrp> print
Flags: X - disabled, I - invalid, R - running, M - master, B - backup
 0   RM name="vrrp1" mtu=1500 mac-address=00:00:5E:00:01:01 arp=enabled
       interface=ether1 vrid=1 priority=255 interval=1 preemption-mode=yes
       authentication=none password="" on-backup="" on-master=""
[admin@MikroTik] ip vrrp>
```

17.2 简单的 VRRP 事例



VRRP 协议能被用于一个冗余的无缝 Internet 连接，让我们假设有 192.168.1.0/24 网络和我们需要提供高效的 Internet 连接。这个网络需要启用 NAT（VRRP 网络需要使用公网 IP，使用动态路由协议如 BGP 或 OSPF）。我们连接到两个不同的 ISP，且一个被设置为最优先(如，价格便宜或者速度更快的).。

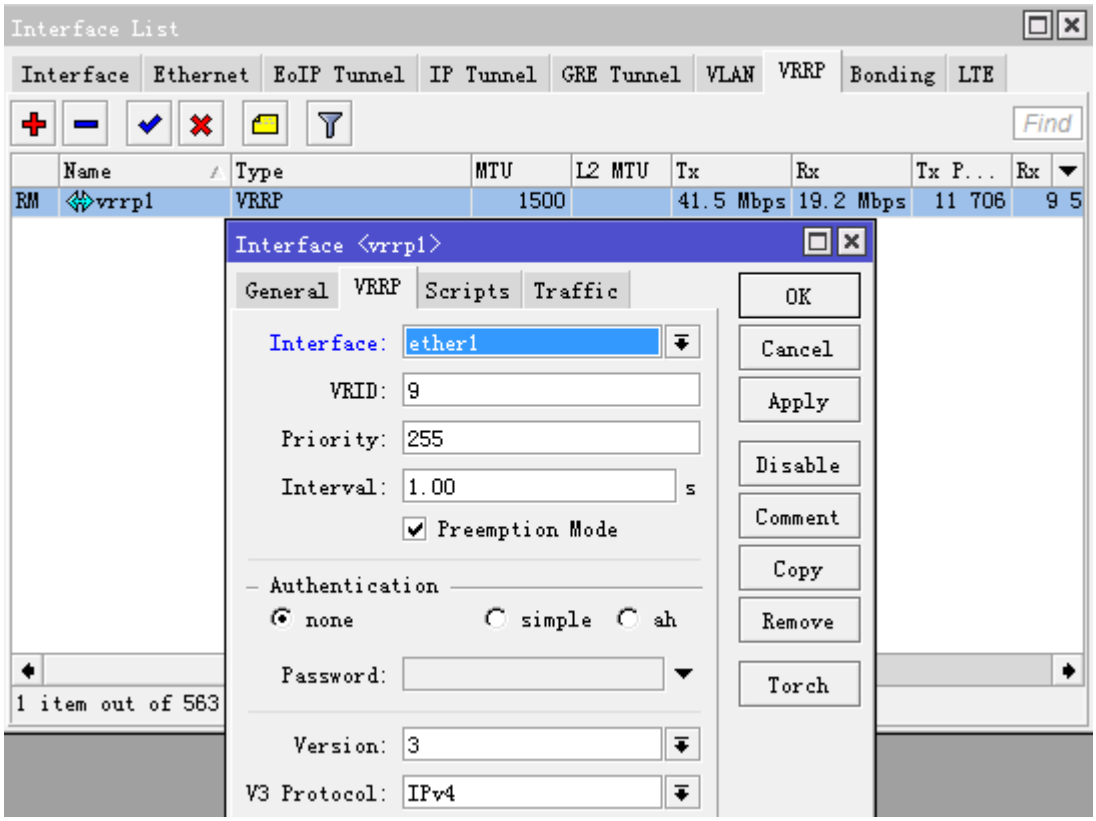
这个事例讲解如何配置 VRRP 在两个路由器上。路由器必须初始化配置：网卡已被启用、每个网卡配置好了 IP 地址、路由表这种正确（至少一个默认路由）。 SRC-NAT 或 masquerading（伪装）应配置好。具体设置请参见相关的内容

我们将 192.168.1.0/24 的网络连接到名为 **local** 网卡的两台 VRRP 路由器上

配置 Master VRRP 路由器

首先我们应创建一个 VRRP 在这个路由器上。我们将使用 255 的优先值，该路由器将被设置为优先路由器

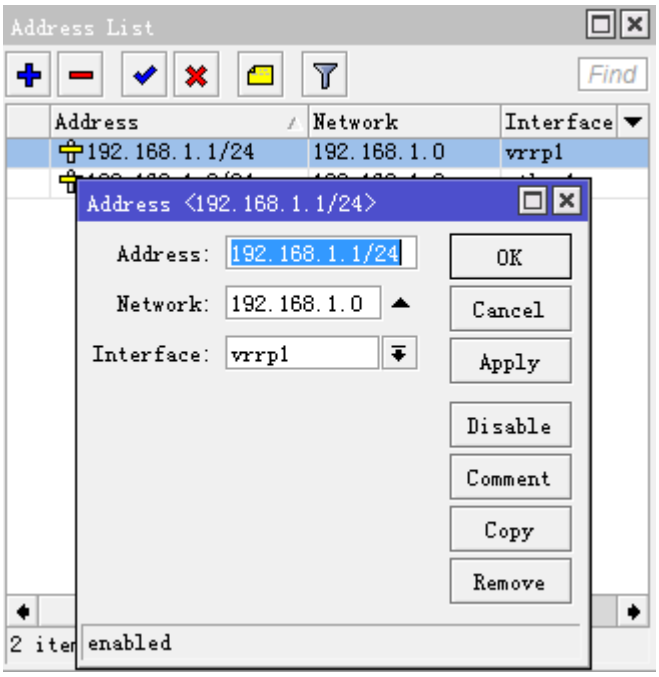
```
[admin@MikroTik] interface vrrp> add interface=local priority=255
[admin@MikroTik] interface vrrp> print
Flags: X - disabled, I - invalid, R - running, M - master, B - backup
 0   RM name="vrrp1" mtu=1500 mac-address=00:00:5E:00:01:01 arp=enabled
       interface=local vrid=1 priority=255 interval=1 preemption-mode=yes
       authentication=none password="" on-backup="" on-master=""
[admin@MikroTik] interface vrrp>
```



下一步，IP 地址应被添加到 VRRP 中

```
[admin@MikroTik] ip address> add address=192.168.1.1/24 interface=vrrp1
[admin@MikroTik] ip address> print
Flags: X - disabled, I - invalid, D - dynamic
  #   ADDRESS            NETWORK         BROADCAST       INTERFACE
  0   10.0.0.1/24        10.0.0.0        10.0.0.255      public
  1   192.168.1.2/24     192.168.1.0     192.168.1.255   local
  2   192.168.1.1/24     192.168.1.0     192.168.1.255   vrrp1
```

```
[admin@MikroTik] ip address>
```



配置 Backup VRRP 路由器

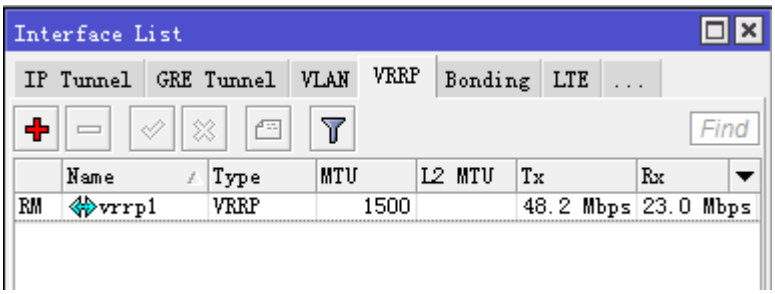
现在我们将创建一个低优先级的 VRRP 路由（我们可以使用默认值 **100**），因此路由器将优先选择 backup：

```
[admin@MikroTik] interface vrrp> add interface=local
[admin@MikroTik] ip vrrp> print
Flags: X - disabled, I - invalid, R - running, M - master, B - backup
 0    B name="vrrp1" mtu=1500 mac-address=00:00:5E:00:01:01 arp=enabled
       interface=local vrid=1 priority=100 interval=1 preemption-mode=yes
       authentication=none password="" on-backup="" on-master=""
[admin@MikroTik] interface vrrp>
```

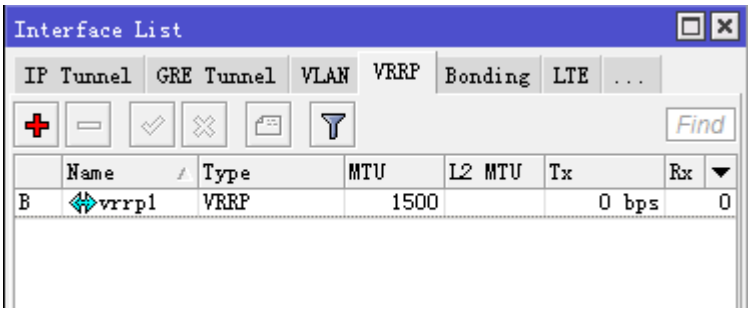
现在我们添加同样的地址到备份 VRRP 路由中：

```
[admin@MikroTik] ip address> add address=192.168.1.1/24 interface=vrrp1
```

在添加完成后，Master 作为主路由器生效，即 Master 接口生效,前缀显示 RM，



Backup 路由器则显示 B



我们断开 master 路由器，在几秒钟后备份路由将选择 master 状态：

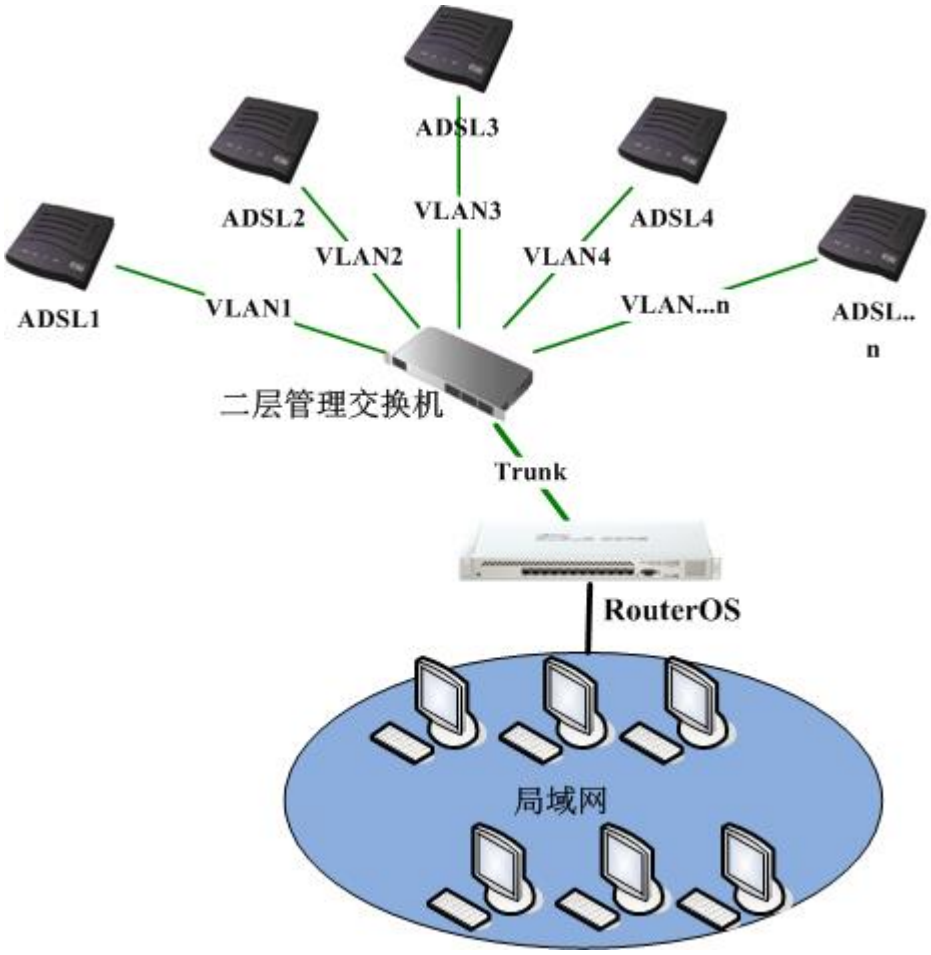
```
[admin@MikroTik] interface vrrp> print
Flags: X - disabled, I - invalid, R - running, M - master, B - backup
 0   RM name="vrrp1" mtu=1500 mac-address=00:00:5E:00:01:01 arp=enabled
       interface=local vrid=1 priority=100 interval=1 preemption-mode=yes
       authentication=none password="" on-backup="" on-master=""
[admin@MikroTik] interface vrrp>
```

17.3 VRRP 多 PPPoE 拨号接入配置

前几年 Nth 和 PCC 负载均衡技术的不断发展，为了实现更多条 ADSL 的负载均衡，就需要 RouterOS 路由器配置或安装更多的以太网卡，接入更多的 ADSL，如果 4-5 条 ADSL 拨号还能基本解决，当多达 10 条时候，路由器扩展网卡受到局限性，虽然 RB 出来 13 接口的 RB1100 但也不能那么浪费千兆接口，同样一台 PC 路由器的硬件配置会相应的增加网卡，这样复杂性、不稳定性、以及成本随之增加。

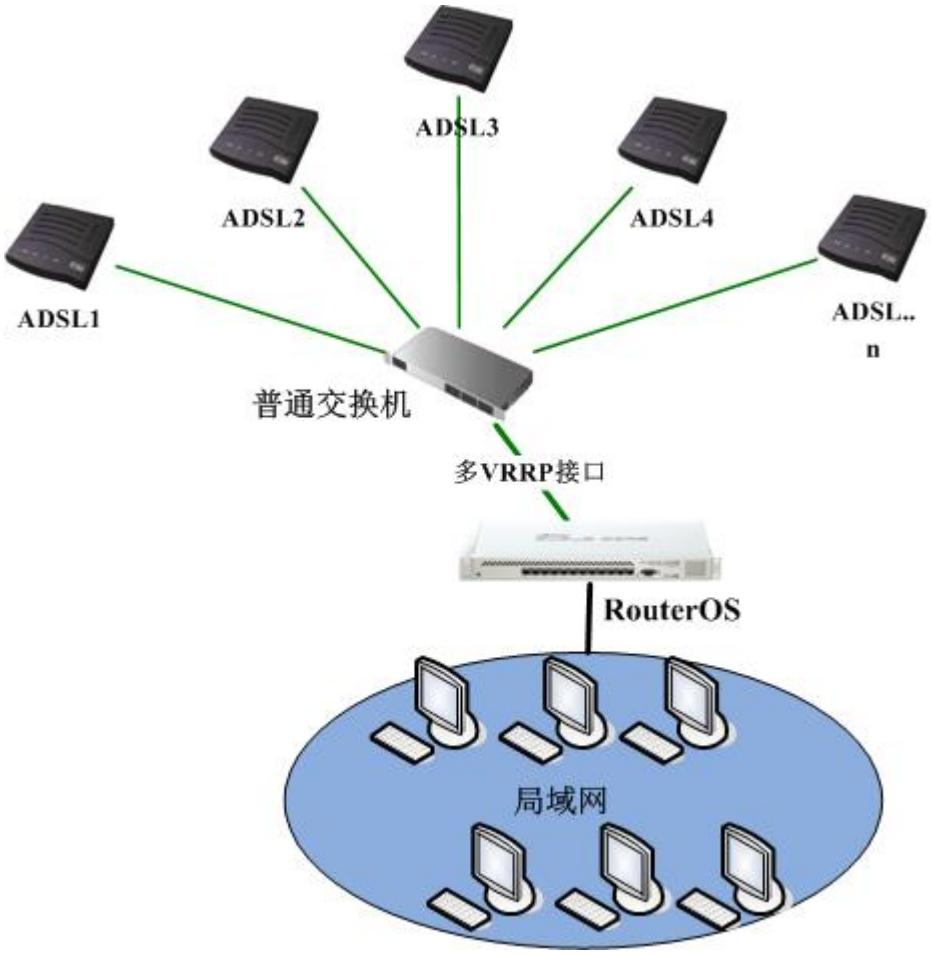
使用多网卡这个原因是在同一家 ISP 的 PPPoE 认证服务器上，客户端 MAC 地址不能相同，所以才要求使用不同的网卡，以确保采用不同 MAC 地址拨号。

后来聪明的 IT 精英们想到了通过二层管理交换机实现了划分 VLAN 的多 ADSL 拨号，通过 24 口、48 口管理交换机划分多个 vlan，Trunk 给 RouterOS，并建立对应的 VLAN，但每个 VLAN 接口的 MAC 地址是从属于实际物理网卡的，所以需要通过加入 bridge 方式，利用 bridge 可以设定 mac 地址的特点，将拨号的 MAC 区分开，达到多 ADSL 拨号能在一张网卡上实现。如下图：



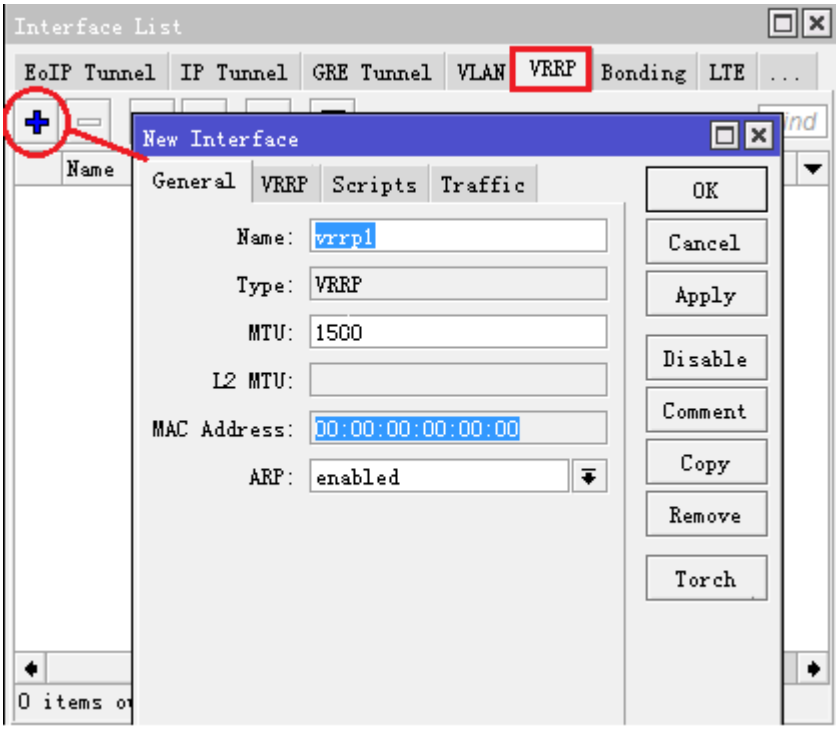
不过这个方式使用了管理交换机，投入的成本也相对较高，精明的 IT 精英又发现了 RouterOS 在 3.0 版本后修改的 VRRP 功能，每当启用一个 VRRP 虚机接口后，都会由于冗余路由协议特性自动生成一个独立与实际物理网卡的虚拟 MAC 地址，不再像之前使用 VLAN 和加入 bridge 的方式，简化了 RouterOS 的配置，这样连接 ADSL 的交换机也不用使用成本较高的管理交换机，仅需要一台普通的交换机就搞定。

警告：由于 VRRP 虚拟的 MAC 地址是 RouterOS 默认的，且无法修改，如果在运营商同一台 BRAS（PPPoE 认证设备）上存在其他的 RouterOS 设备也采用 VRRP 虚拟接口，会出现 MAC 一样，是无法完成拨号，因此可能出现配置 VRRP 后 PPPoE-client 无法拨号成功的情况。

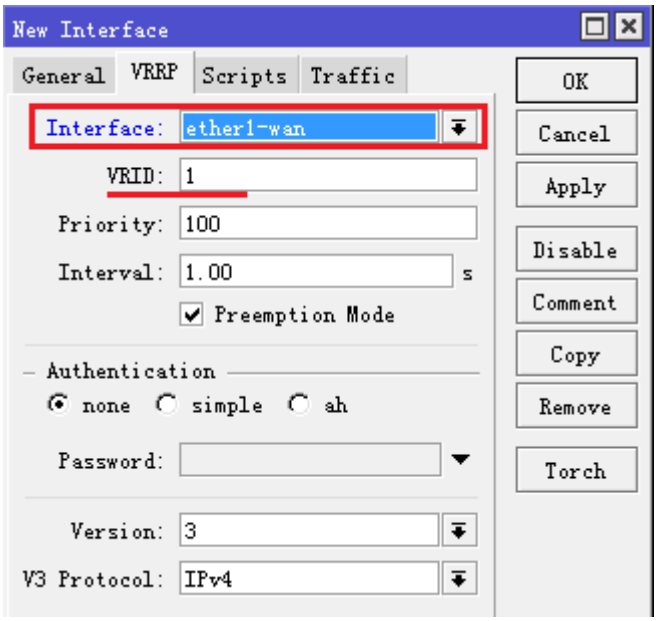


采用 VRRP 拨号 ADSL 这个方式，配合上 PCC 即可实现多线的负载均衡，由于 Nth 的功能缺陷，这里主要介绍 PCC 的负载均衡。

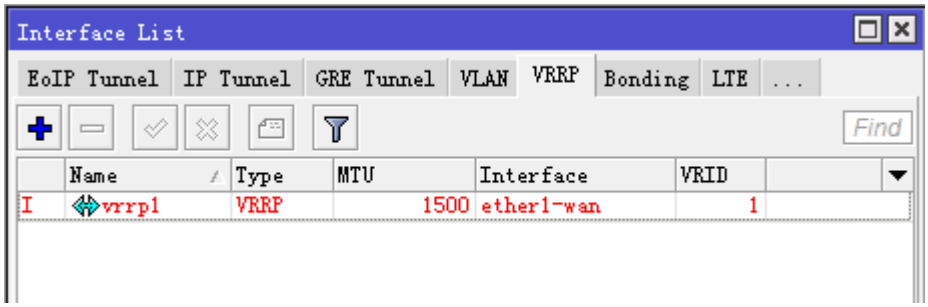
设置 VRRP 虚拟接口，进入 interface vrrp 菜单下，添加一个 vrrp 接口，



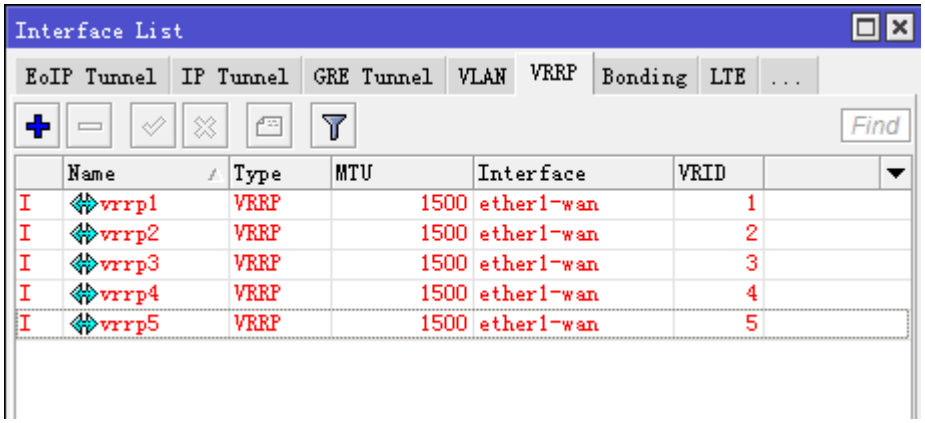
在添加项中，进入 VRRP 项设置 interface 和 VRID，这里 interface 是设置对应拨号的物理网卡，VRID 用于区分多个 VRRP 虚拟接口的身份，即每个用于拨号的 VRRP 虚拟接口 VRID 都不同，其他参数默认，这样 vrrp1 接口就在 ether1-wan 上生成了一个虚拟机口（如果你想仔细了解 VRRP，可以参考教材的 VRRP 章节）。



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



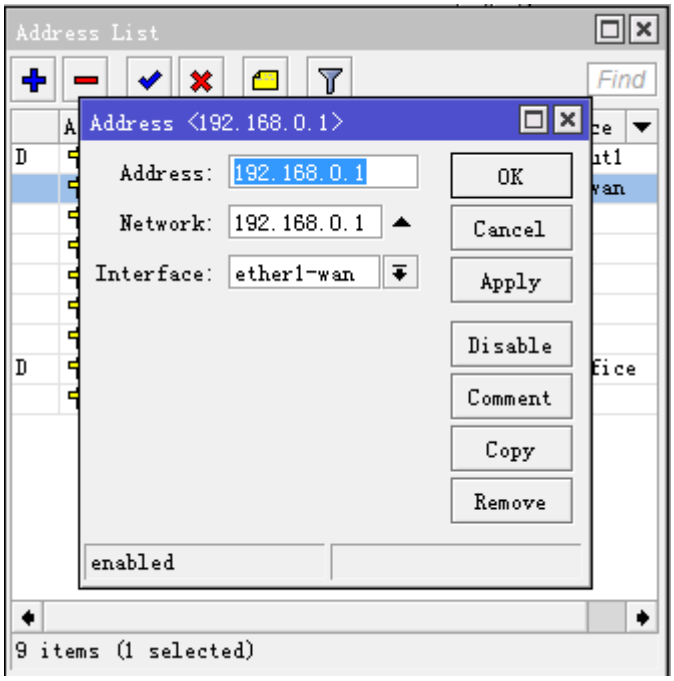
假设我们是 5 条 ADSL 拨号，这我们同样需要建立 5 条 vrrp 虚拟接口，他们只是 VRID 不同，其他参数一样，5 条 ADSL 的 VRID 分别是 1～5



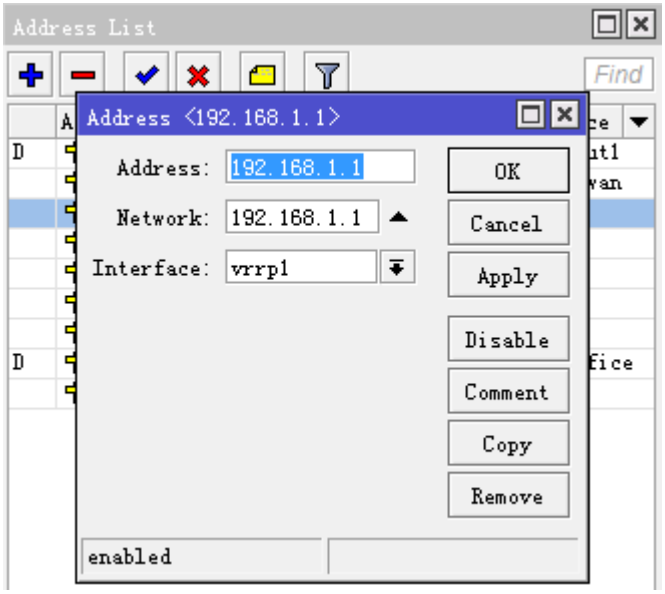
5 口虚拟的 VRRP 接口添加完成后，下面需要将它们激活，当前他们都不能正常使用，因为 VRRP 接口虚拟路由冗余协议，即对 2 台以上设备实现三层的冗余，当前 5 口接口都没有设置 IP 地址，即是不能生效使用的。

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。

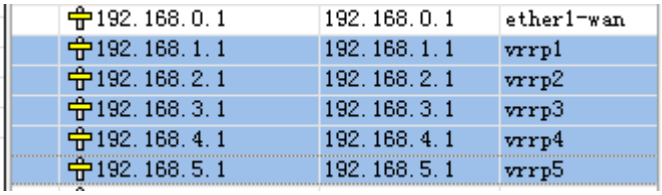
首先我们设置 ether1-wan 接口的 ip，192.168.0.1，这里我直接使用主机 IP 是可以生效的



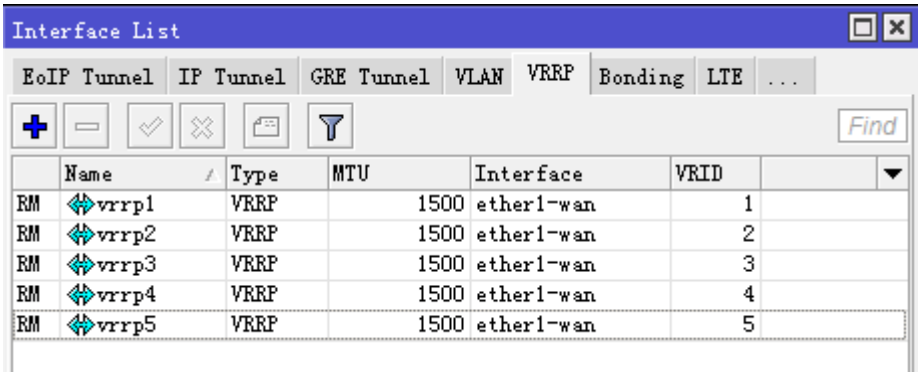
接着添加 vrrp1 虚拟接口的 ip 地址，同样设置为主机 ip，192.168.1.1，



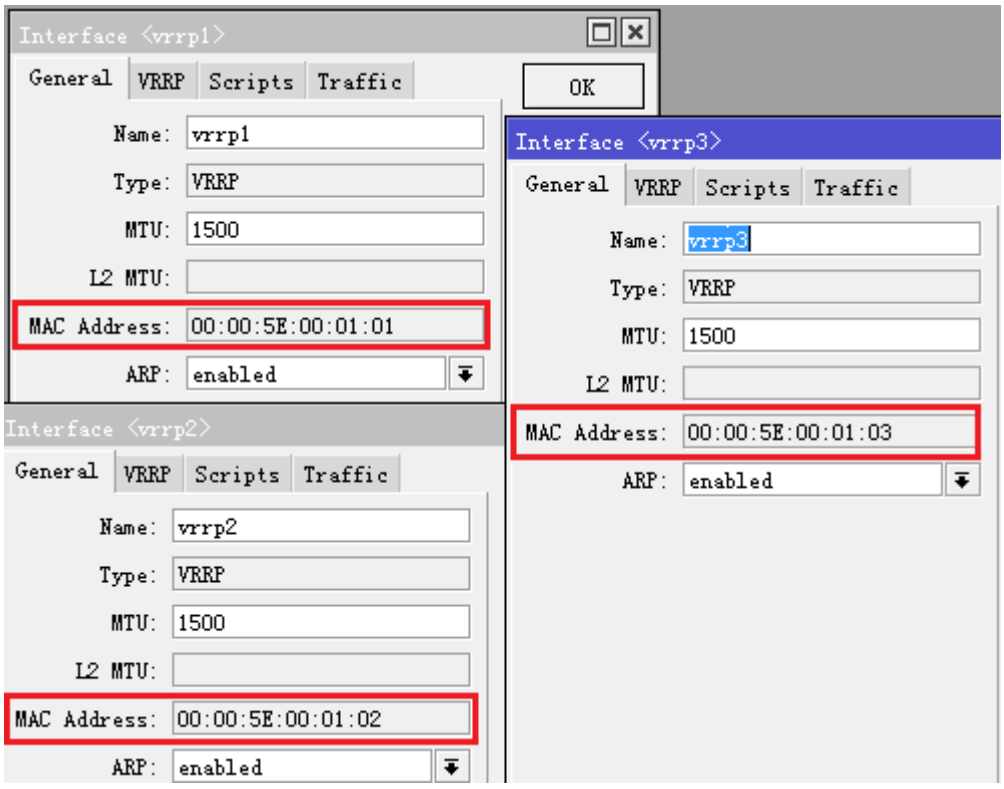
之后的配置以此类推，添加剩下的 4 个 vrrp 接口



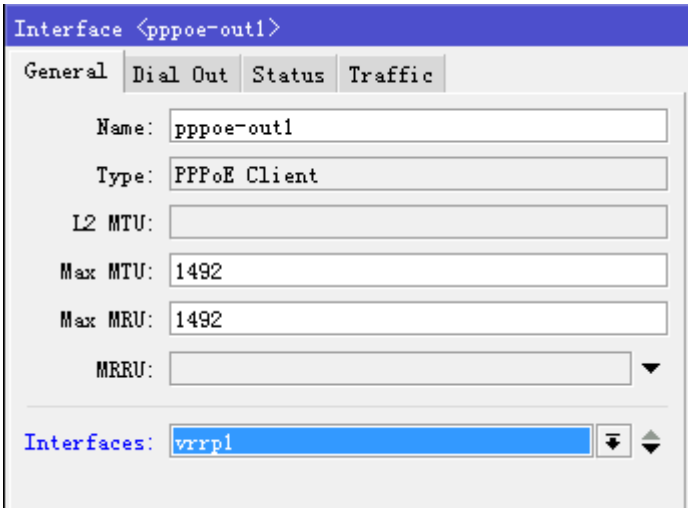
现在我们返回到 interface vrrp 菜单下，所有接口都进入了 RM 状态，即 VRRP 协议的 Master 状态



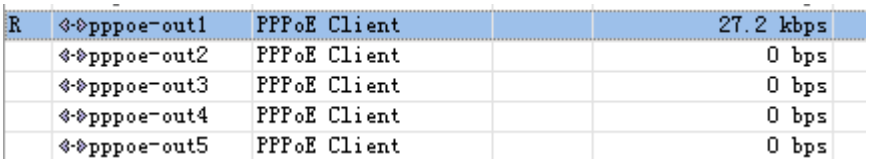
我们来对比下 VRRP 接口的 MAC 地址，都是不同的



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 个不同帐号的拨号了！



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

剩下的操作就设置 PCC 参数

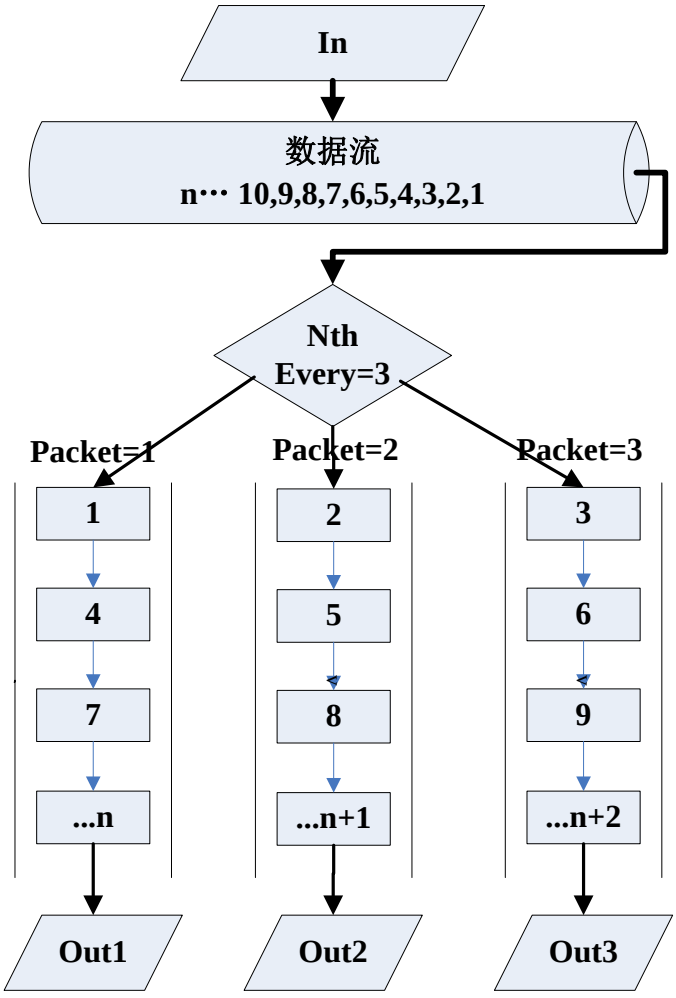
第十八章 RouterOS Nth

Nth 是 RouterOS 中对路由负载均衡一个重要的工具，不仅可以实现基于 IP 的负载均衡，还能实现对端口负载均衡和 nat 的有序指定访问

18.1 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”。每个规则都有自己的计数器。当规则收到数据包，当前规则的计数器会增加 1，如果计数器匹配值“every”与数据包匹配，计数器将重新设置为 0。 使用 Nth 我们可以将一串连接通过计数器分离，比如可以将连接分配为多个组，重新排列连接序列。

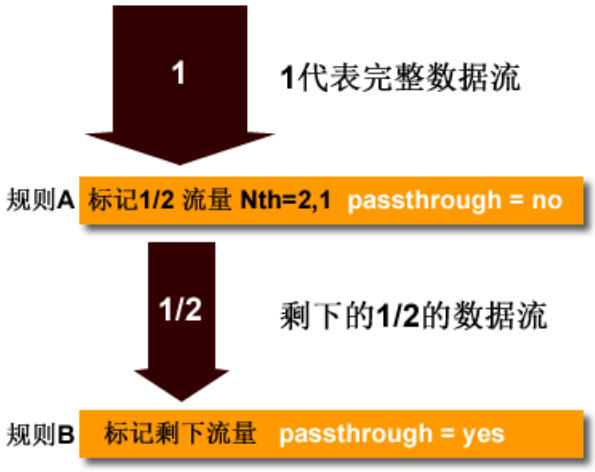
nth –匹配特定的第 N 次收到的数据包的规则。一个计数器最多可以计数 16 个数据包
Every – 匹配每 every 数据包，同时指定 **Counter**（计数器值）
Packet – 匹配给定的数据数，例如，Nth=3,1，匹配 3 个数据包的第 1 个

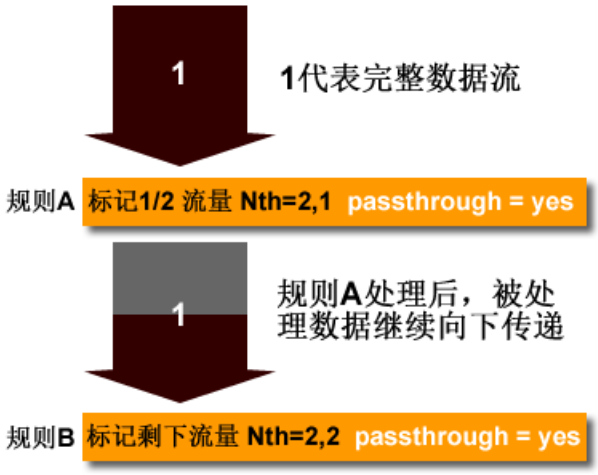


上图，可以看到数据流从 1-n 的数据，被 Nth 分为 3 个计数器，并根据 Packet 重新排列数据流的队列。Nth 我们可以应用的范围，包括多线路的负载均衡、内网多台 ftp 访问、以及其他的应用。

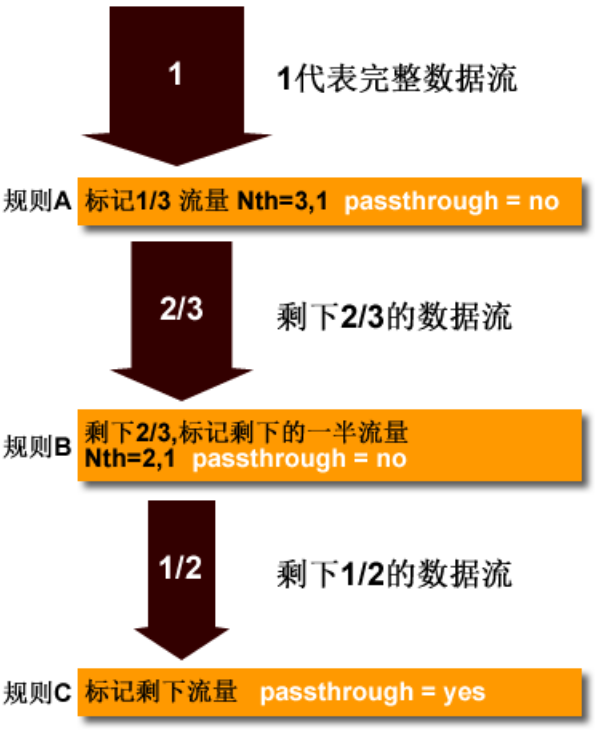
18.2 Passthrough 对 Nth 的控制

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包并对比所有流量的 1/3，第二条规则标记剩下 2/3 数据包的 50%，第三条规则标记和对比所有剩下的数据包（所有数据包的 1/3）

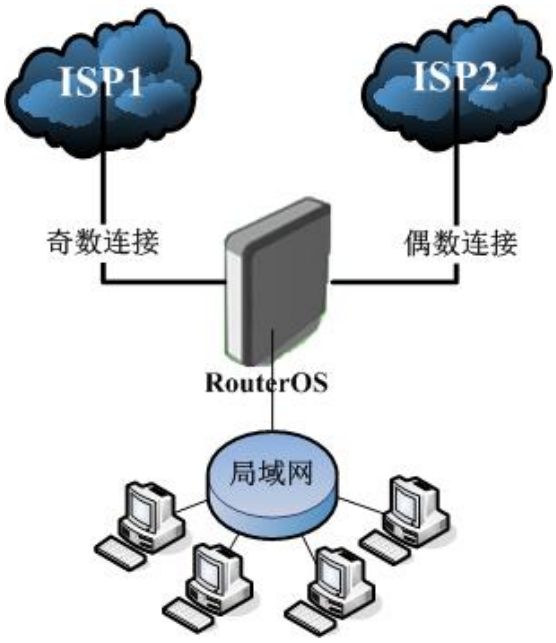
```
/ip firewall mangle
add action=mark-connection chain=prerouting new-connection-mark=AAA nth=3,1
passthrough=no;
add action=mark-connection chain=prerouting new-connection-mark=BBB nth=2,1
passthrough=no;
add action=mark-connection chain=prerouting new-connection-mark=CCC ;
```

同样我们有的数据包并且每个规则对比每 3 个数据包。

```
/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;
```

18.3 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。



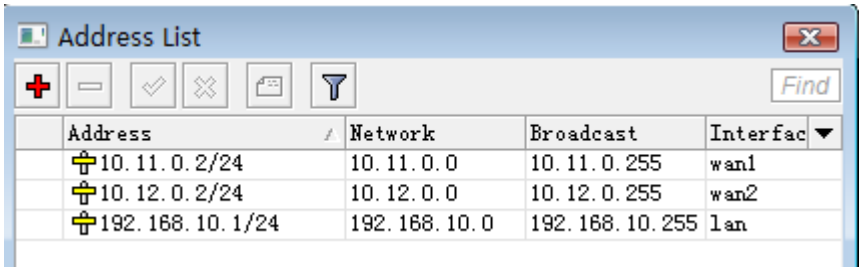
根据 Nth 的原理我们可以将来至内网的联接分为两组，即一组为奇数连接、一组为偶数连接，即奇数走一条线路，偶数走另外一条线路。因为我们定义的是连接状态为 new，即新建立的连接，对正常的访问没有任何影响，每个新建立所产生的后续数据都会按照原来的线路连接运行。

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。

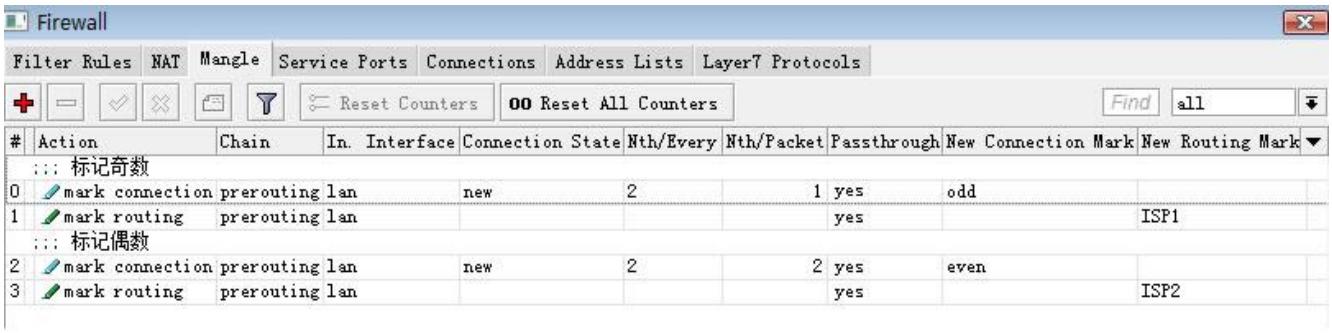
网络参数如下：

- 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：

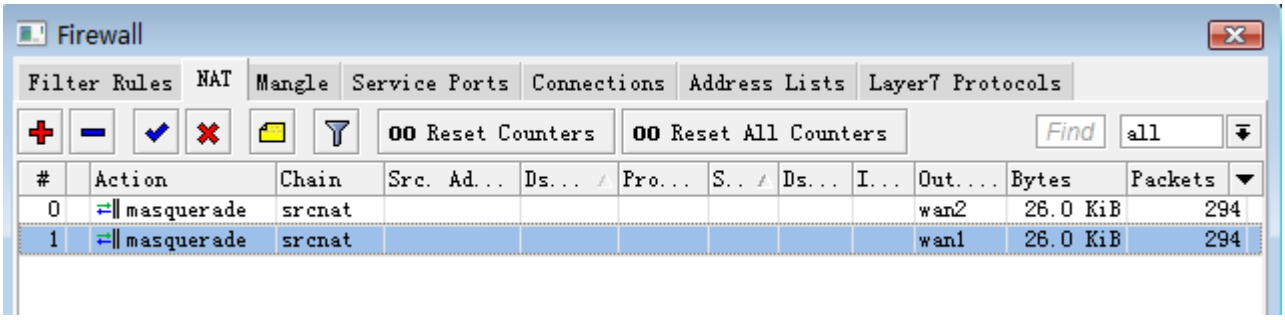


命令行配置如下：

```
[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
```

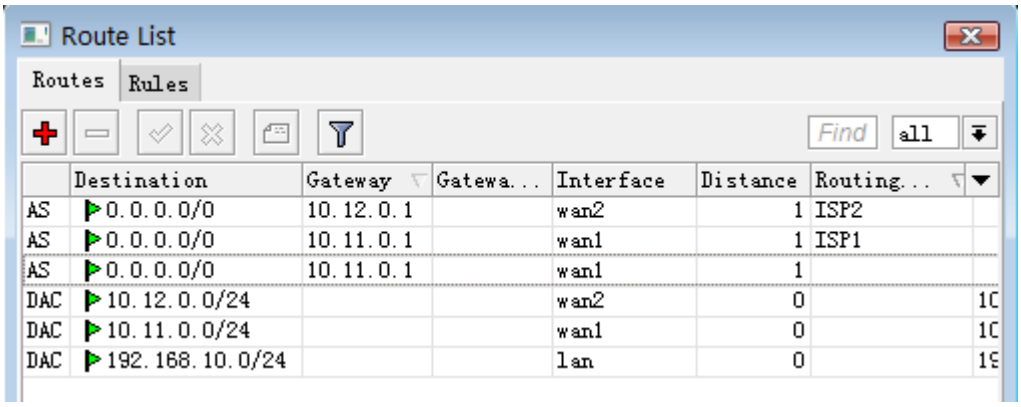
```
 3   chain=prerouting action=mark-routing new-routing-mark=ISP2 passthrough=yes
in-interface=lan  connection-mark=even
```

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
```

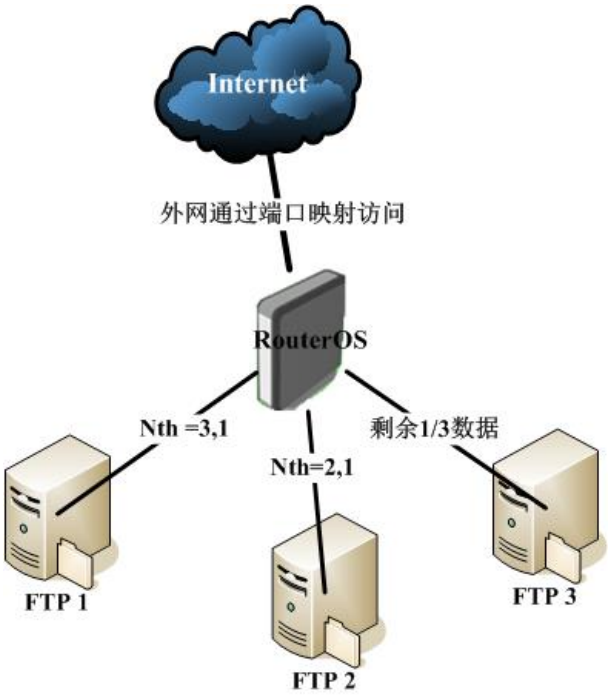
这样双线的 Nth 负载均衡就配置完成，建议这样的负载均衡使用在相同 ISP 的线路上，并且带宽接近。

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]

18.4 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。

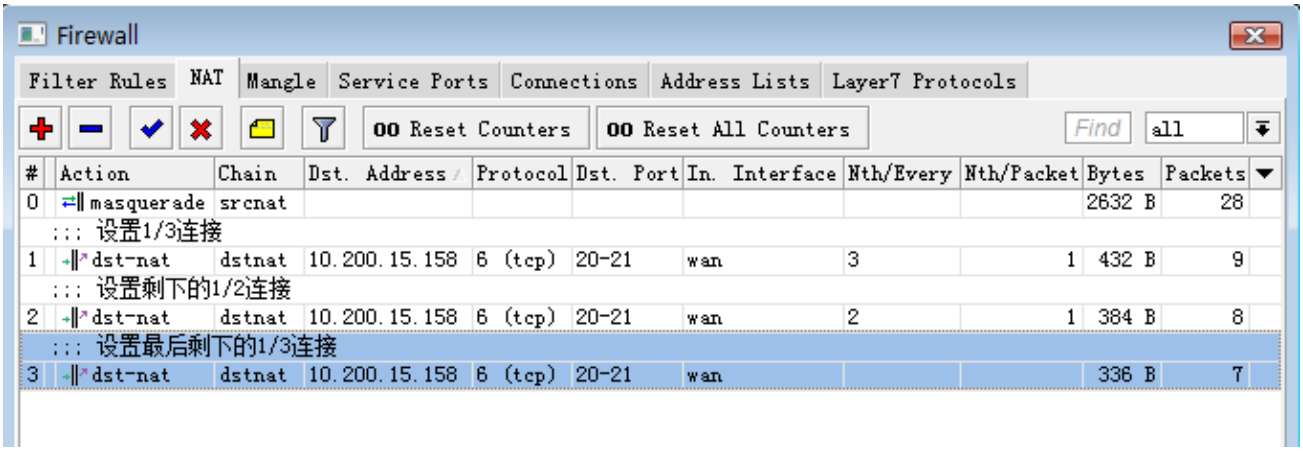
我们的网络环境如下

- 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。

第十九章 RouterOS 数据流(Packet Flow)

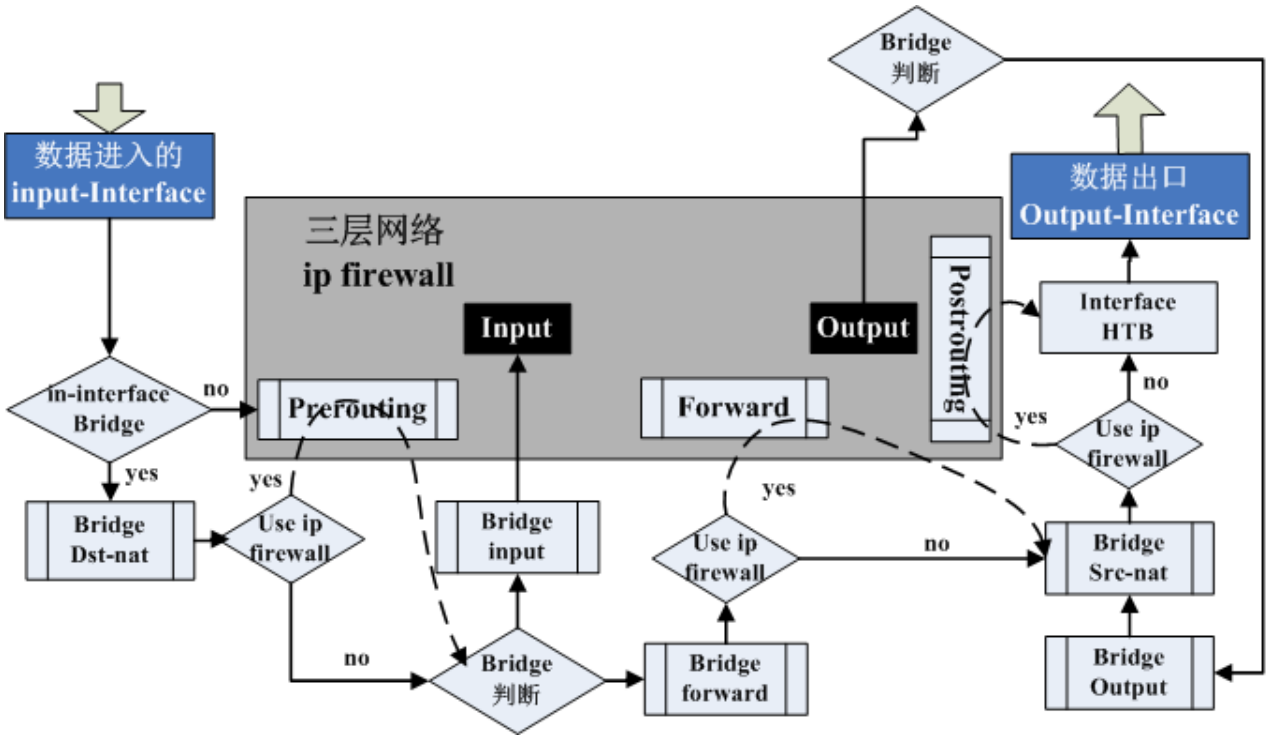
了解 RouterOS 的核心就需要对 IP 数据流进入 RouterOS 后是如何被处理的，这里将通过各种流程图来介绍 RouterOS 是如何处理 IP 数据流的过程，需要注意的是 RouterOS 在 3.x 和 6.x 对数据流处理都有重大调整。

19.1 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。

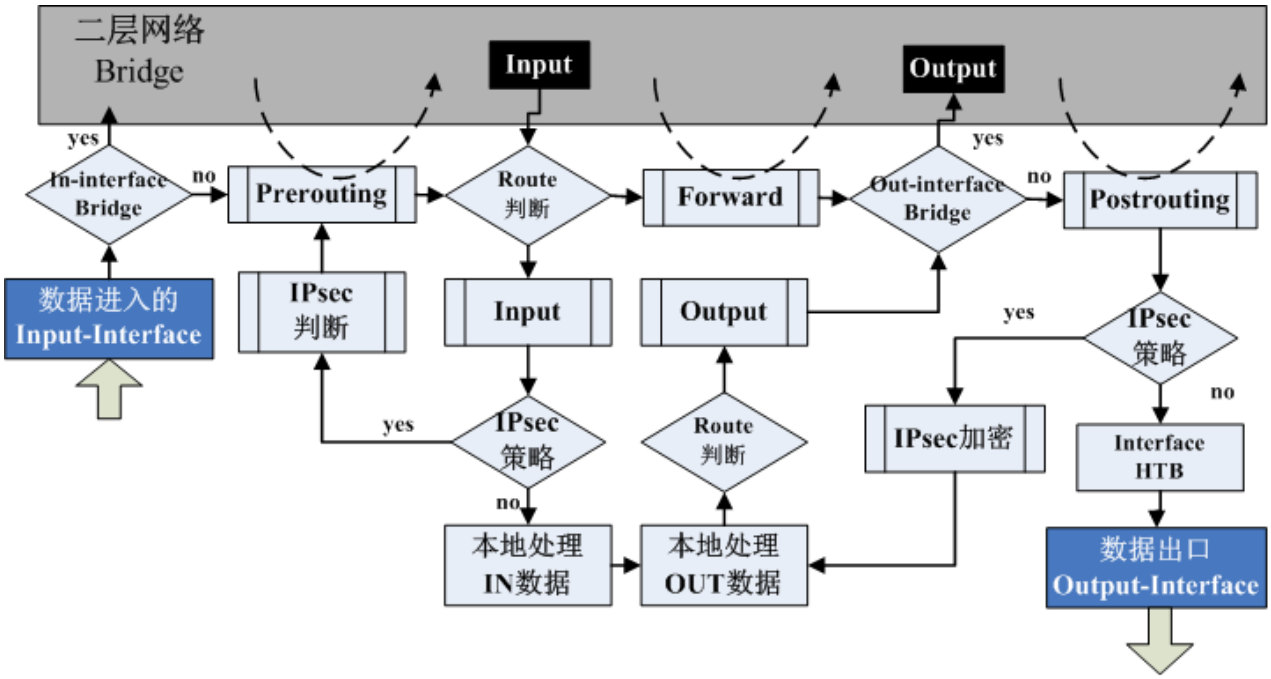
二层网络

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



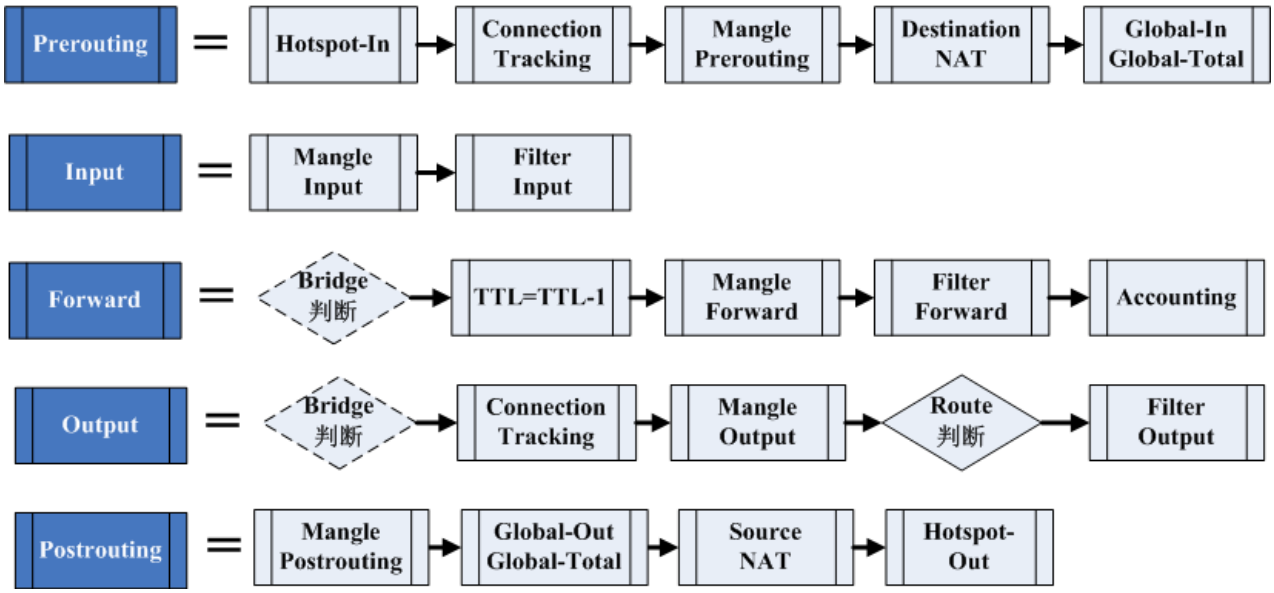
三层网络

在下面的图中，将二层网络或“Bridging”简化到框内



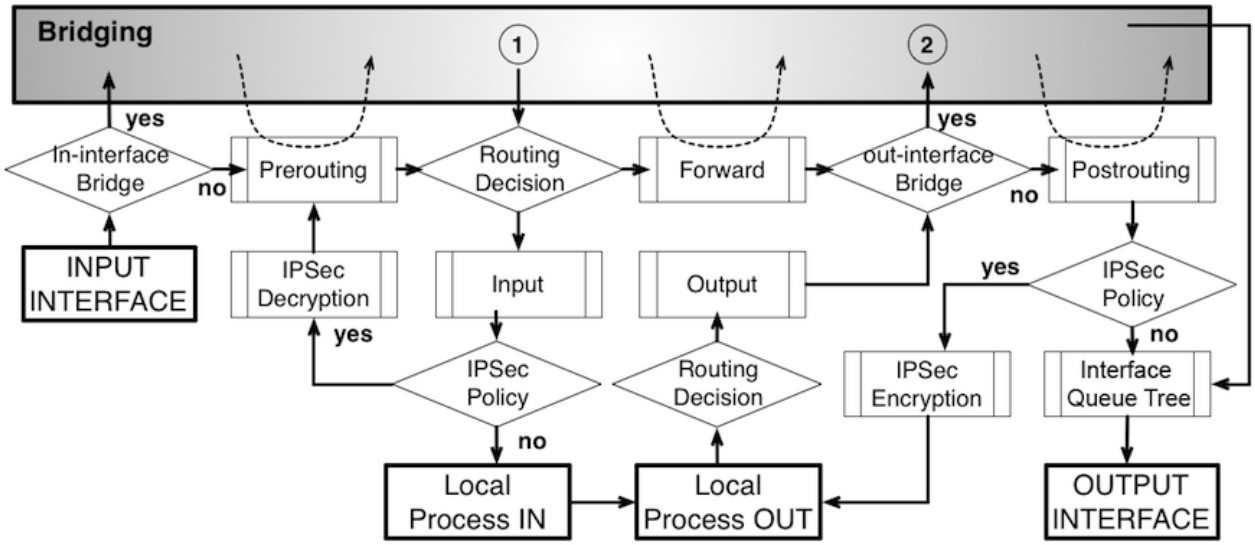
- **Input Interface** - 数据包进入路由器的起点，不论什么样的接口（物理接口或虚拟接口）数据包都会从这里开始进入路由器。
- **Output Interface** - 数据包离开路由器的终点，在之前经过或路由自发的数据包被发送出路由器。
- **Local Process IN** - 最终目的地是到达路由器自身的数据包。
- **Local Process OUT** - 由路由器自身发出的数据包。

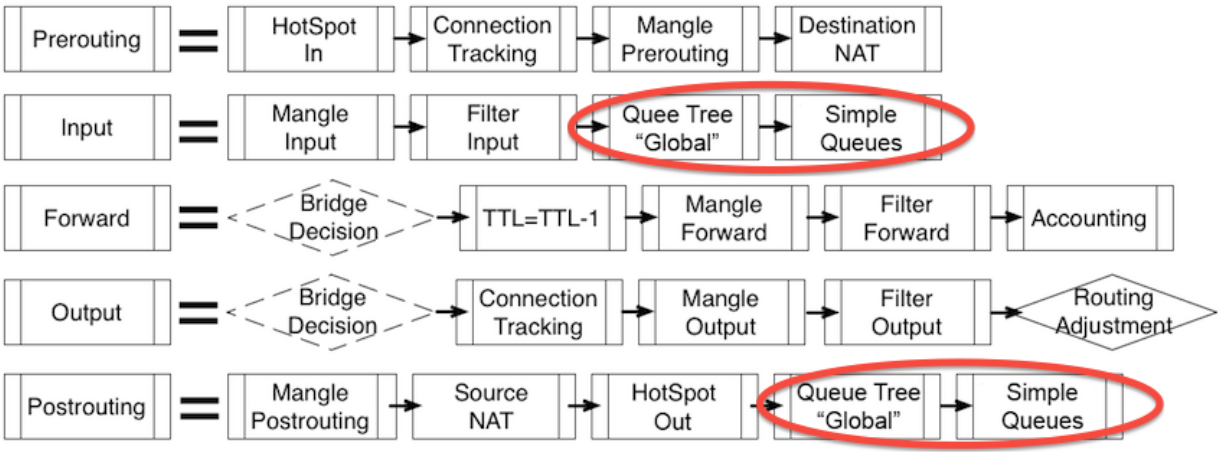
下面的图是 mangle 中 prerouting、input、forward、output 和 postrouting 链表所包含的各个功能组件，该流程是 v6.0 版本前：



19.2 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

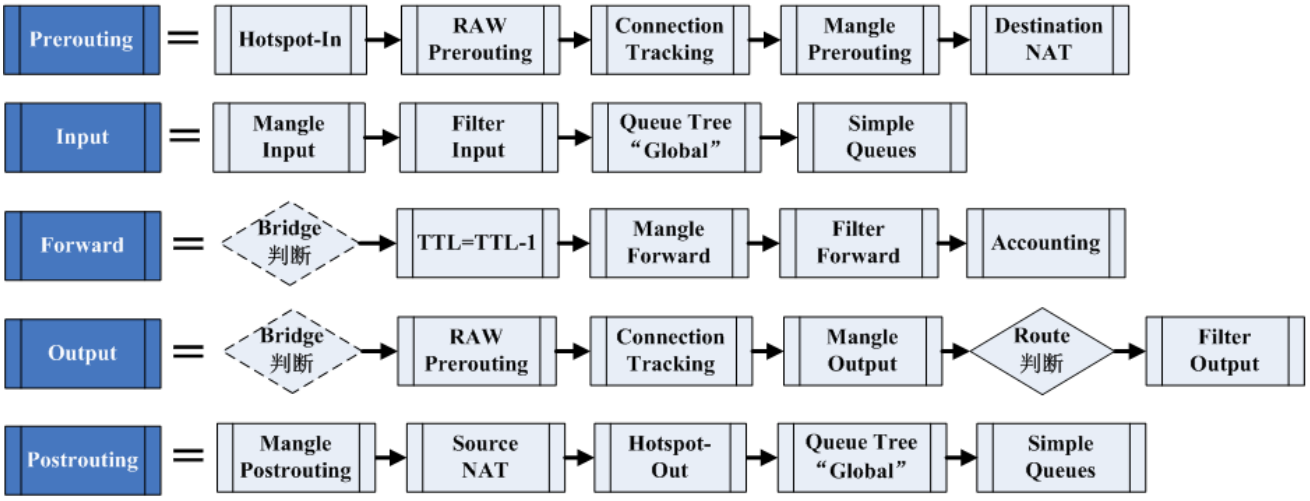




在流程图上可以看到，input（原来在 prerouting 链表）和 postrouting 下 simple queues 和 queue tree Global 从 Global-in/out 和 Global-total 中分离出来，官方的解释如下：

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



19.3 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

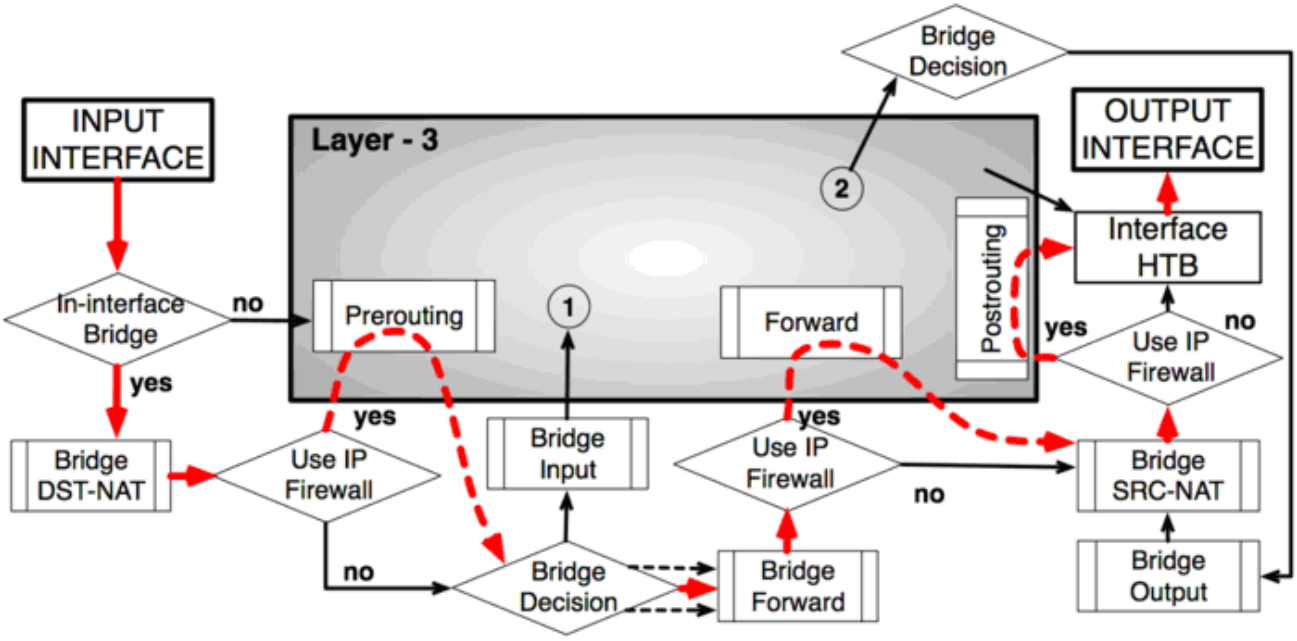
判断处理

- **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 地址列表查找数据包所匹配的目标 MAC 地址，当目标被找到，数据包将会被发送到相应的桥接口，如果没有匹配则会将数据包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包的目标 IP 地址。当查找到后，数据包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，数据包将会被丢弃。

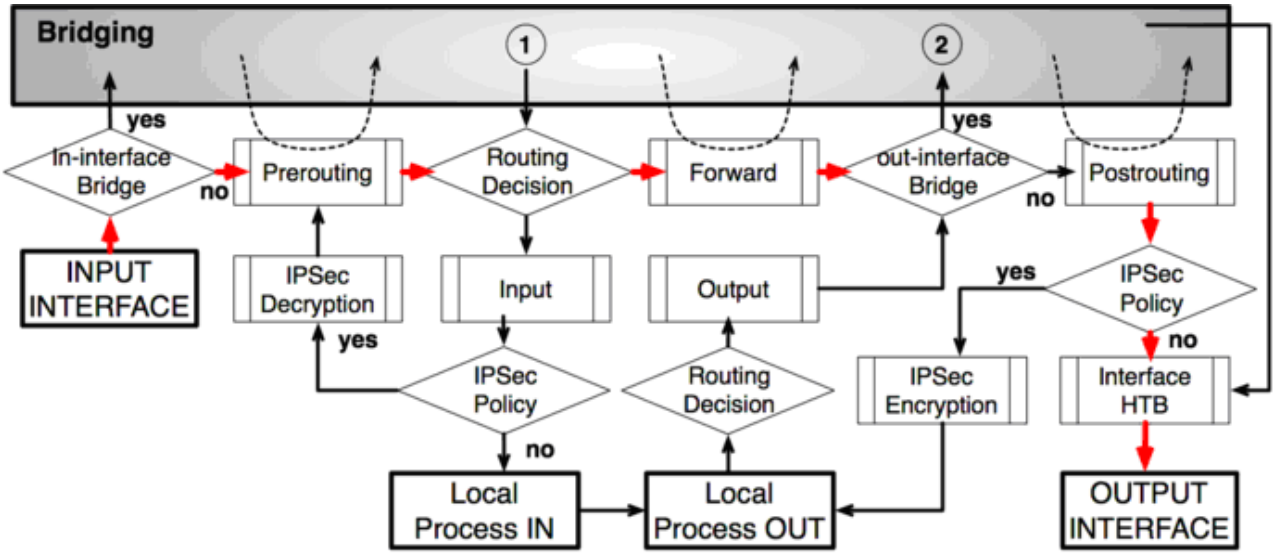
- **TTL-1** – 当路由得到准确的目的地，TTL 值被减少 1。如果 TTL 值变为 0，IP 包将会被丢弃掉。
- **IPSec Decryption/Encryption** - IPsec 判断，解密和加密。
- **Out-Interface Bridge** - 判断实际输出接口是否为桥接端口或判断输出接口是否为桥。
- **HotSpot-Out** - 撤销所有通过 Hotspot-in 的数据包操作，并发送回客户端。

19.4 功能处理流程

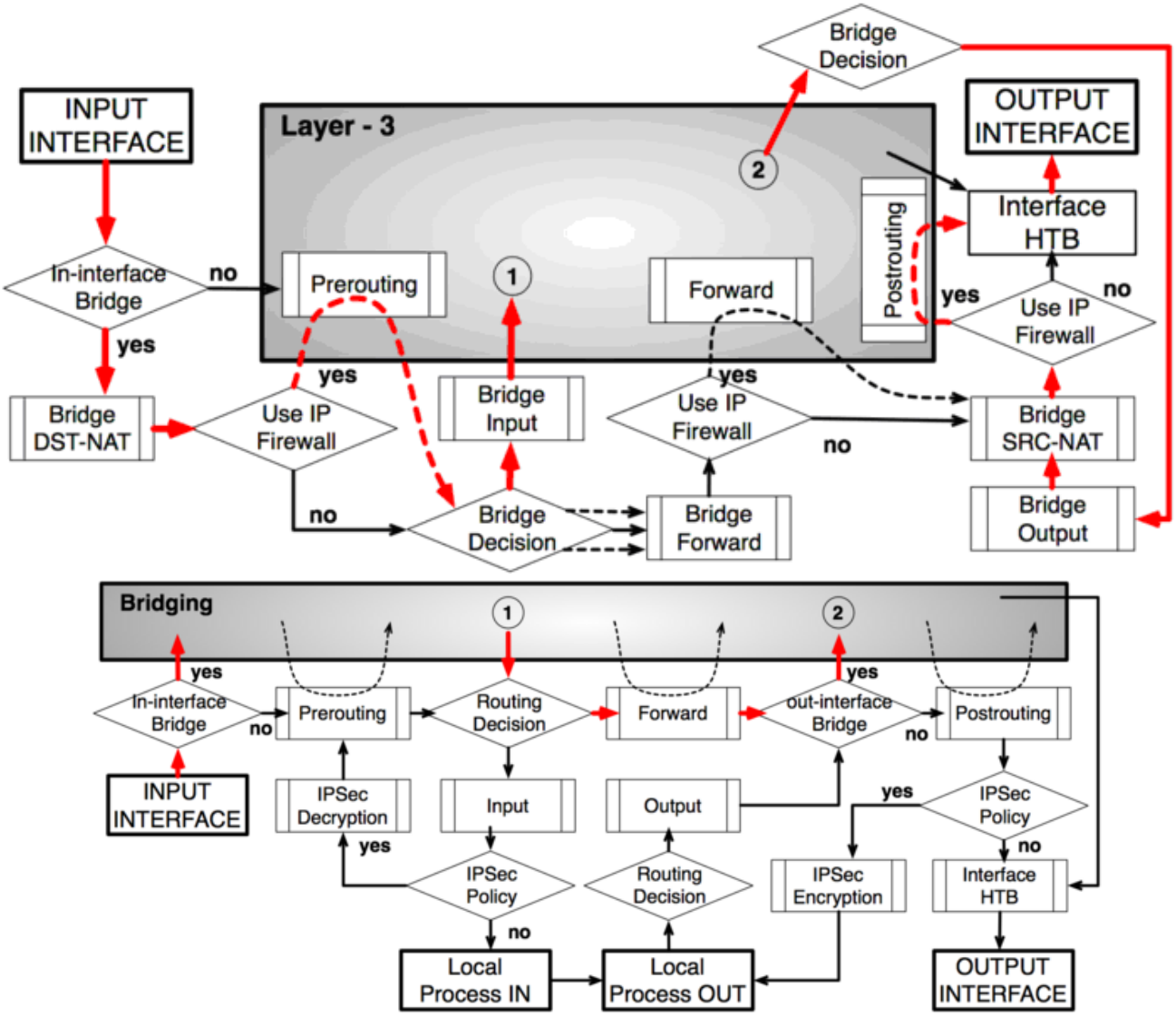
桥接设置 use-ip-firewall=yes



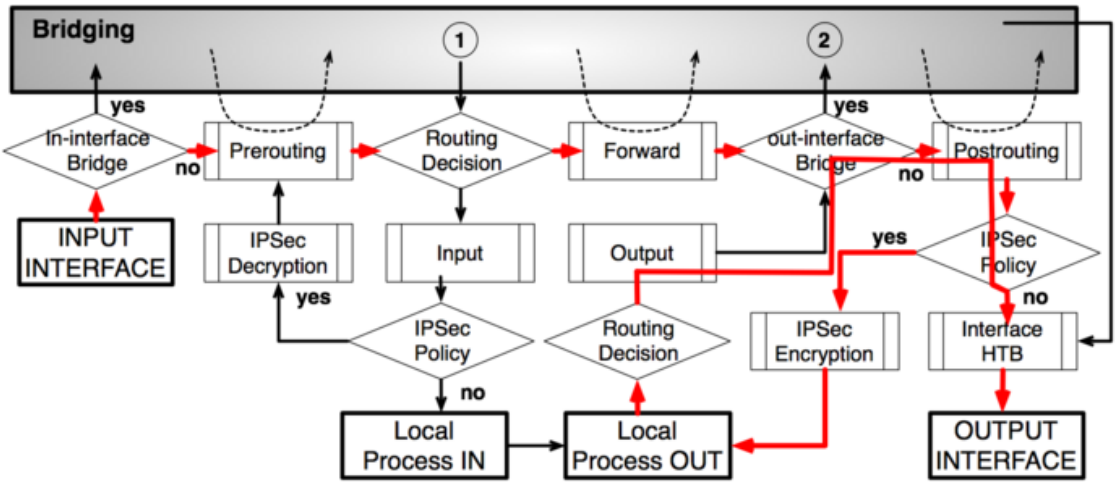
路由 – 从 Ethernet 到 Ethernet 接口



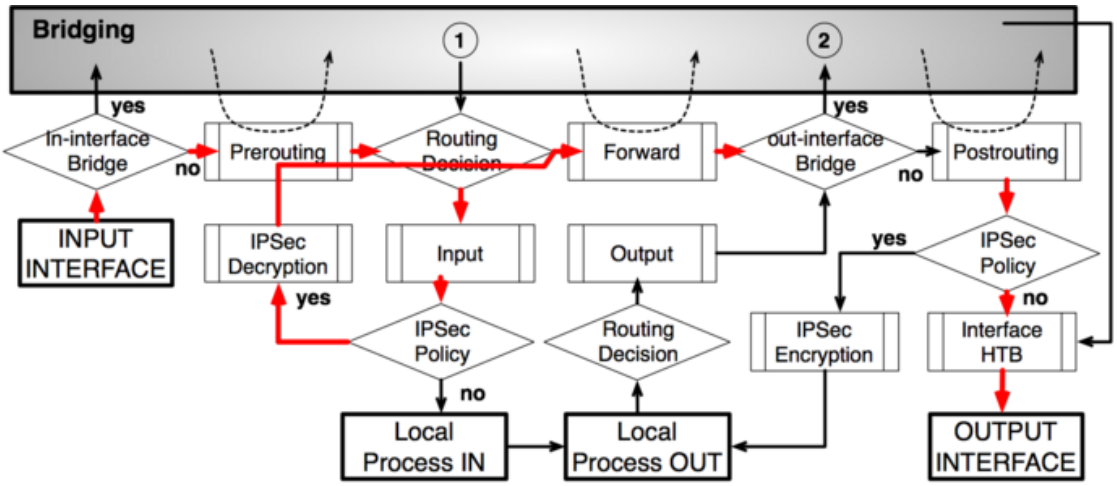
路由 - 从一个桥接口道不同的桥接口



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

第二十章 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帐号，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 提供了验证和帐号服务为 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 数据包向 RADIUS 服务器提交用户信息，包括用户名、密码等相关信息。

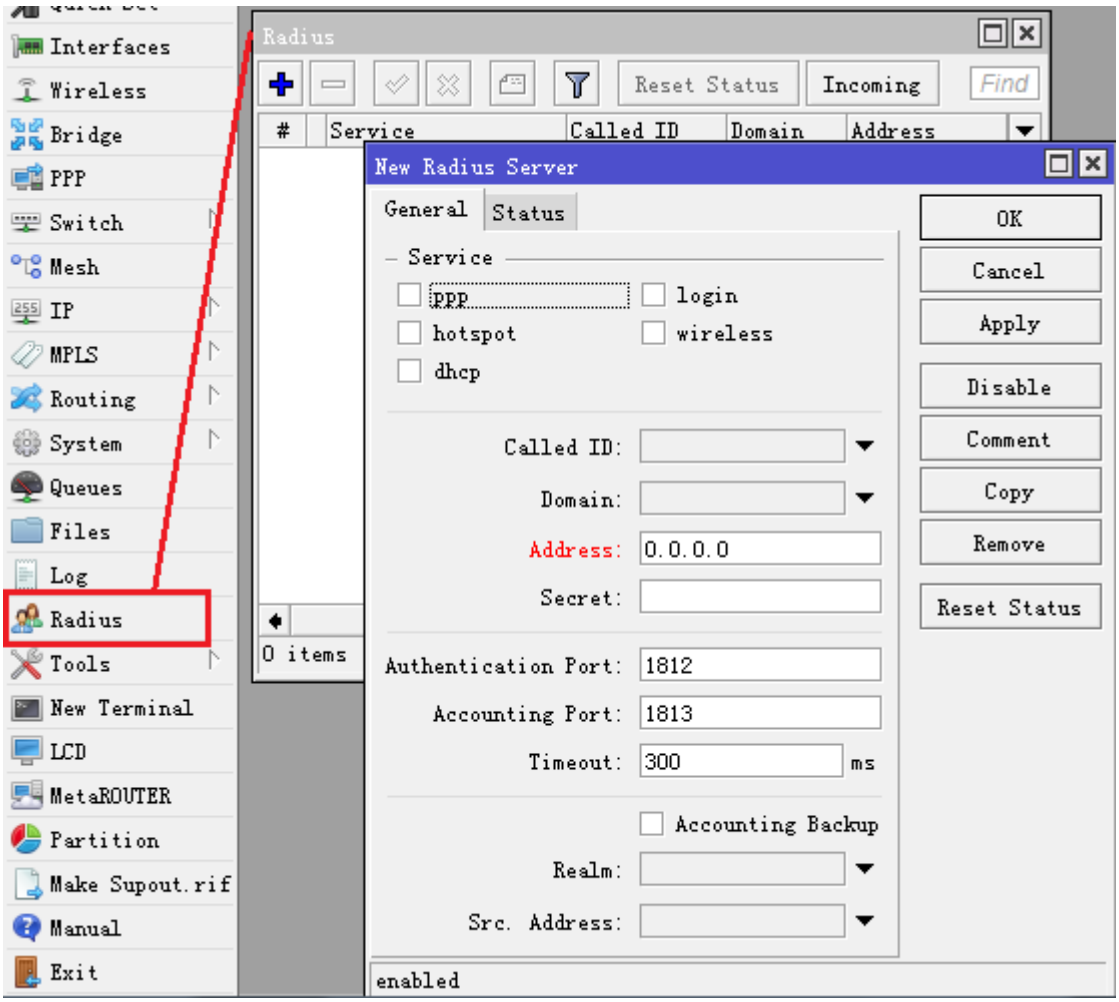
MikroTik RouterOS 提供了一个 RADIUS 客户端，能对接 RADIUS 服务器，并为 HotSpot, PPP、PPPoE、PPTP 和 L2TP 连接。 RADIUS 服务器通过本地数据库存储用户信息和策略参数，例如 RouterOS PPP 中的 Profile 配置有用户认证的相关信息和策略，当认证信息传输访问到 RADIUS 服务器，会从 RADIUS 数据库中查询相关信息，并返回。在 RouterOS 当本地数据库没有匹配到用户记录，RADIUS 服务器数据库才会被访问

20.1 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

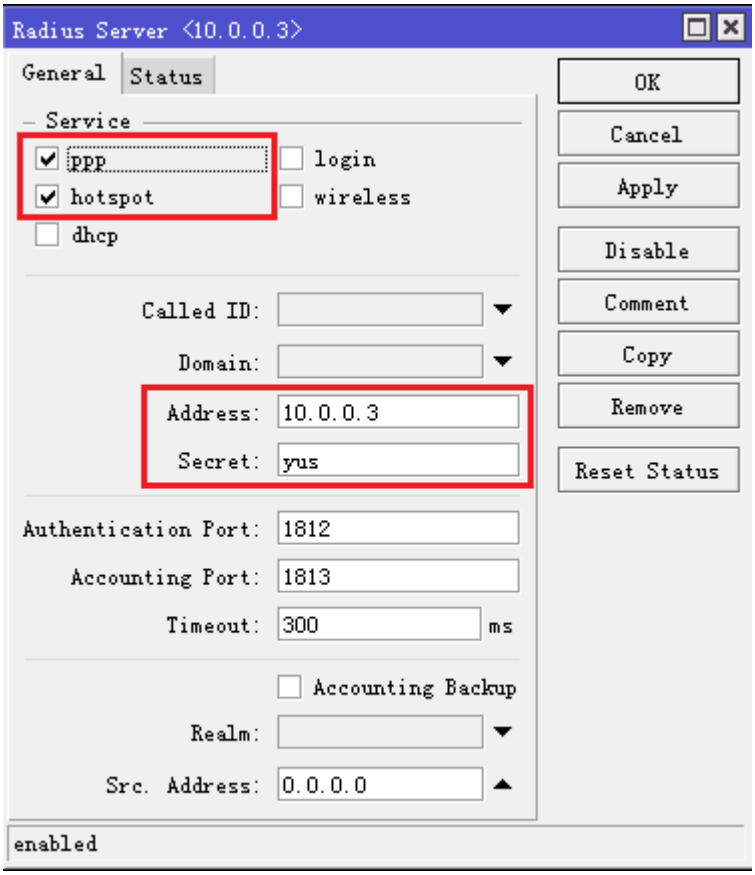
RADIUS 客户端属性

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 – 帐号登录路由器认证 ▪ 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，非请求报文时 RADIUS 协议的扩展命令，允许终止已经连接到 RADIUS 服务器的会话。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

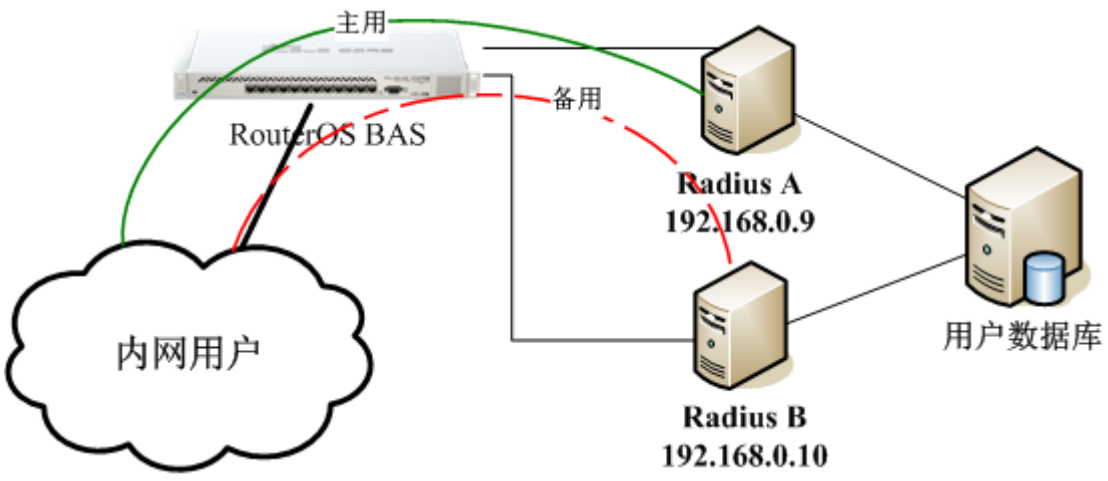
20.2 RouterOS 连接 RADIUS 备份

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 数据包，允许用户进行下一步工作，否则返回 Access-Reject 数据包，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

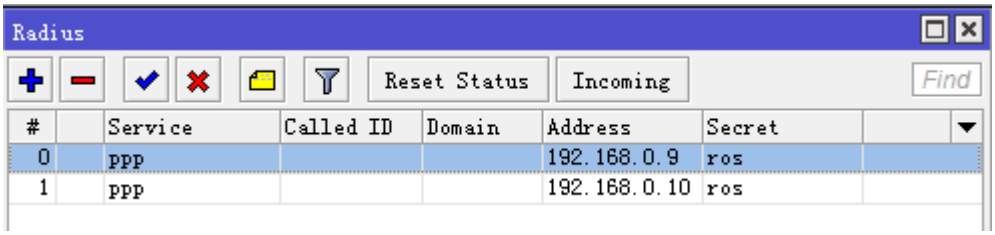
即 2 台或 2 台以上的 RADIUS 组成的 RADIUS 备份集群，如下图



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。

第二十一章 HotSpot 热点认证网关

21.1 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界面，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 将在数据包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要求认证（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（没有必要担心纯文本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 地址与用户帐号同时作为用户名。

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

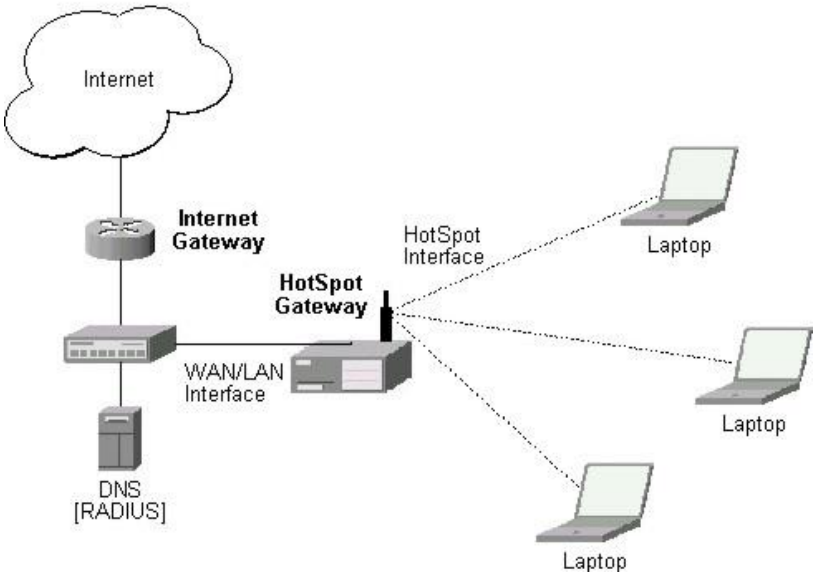
- **/ip hotspot** - HotSpot 上的特定界面（每个界面一个服务器）。HotSpot 服务器必须添加在这个目录中，HotSpot 系统才能够在一个界面上工作。
- **/ip hotspot profile** - HotSpot 服务器概要。影响 HotSpot 客户登陆过程的设置在这里进行。多个 HotSpot 服务器可以使用同样的概要信息。
- **/ip hotspot host** - 所有 HotSpot 接口上的活动网络主机的动态列表。在这里你可以找到 IP 地址与一对一 NAT 的绑定
- **/ip hotspot ip-binding** - 将 IP 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 用户的电脑和网关之间。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。

21.2 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>
```

查看根据向导一步一步创建的相关配置

```
[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>
```

向导提示了你设置对应的服务网络接口，IP 地址池、DHCP 服务、DNS 服务和 nat 规则配置

21.3 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**) - 对未认证客户的持活超时时间。用于探测客户的计算机是活动的并且是可达的。如果在这个期间探测失败，那么用户将被主机列表清除并且用户使用的地址也将被释放
    none - 不切断不可达用户
profile (*名称*; 默认: **default**) - 接口的默认 HotSpot 概要
reset-html (*名称*) - 以原始的 HTML 文件重新覆盖已有的 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>
```

21.4 HotSpot profile 策略

操作路径: ***/ip hotspot profile***

属性描述

dns-name (*文本*) - HotSpot 服务器的 DNS 名称。与 HotSpot 服务器名类似的 DNS 名。（它看起来像登陆页面位置）。这个名字会被自动地在 DNS 缓存中添加为一个静态 DNS。
hotspot-address (*IP address*; default: **0.0.0.0**) - HotSpot 服务器的 IP 地址
html-directory (*文本*; default: **""**) - 目录的名称（以 FTP 访问），它存储了 HTML servlet 页面（当改变路径时，如果路径不存在，默认页面会自动被复制到指定的目录中）
http-cookie-lifetime (*时间*; 默认: **3d**) - HTTP cookies 的有效时间
http-proxy (*IP 地址 s*; 默认: **0.0.0.0**) - HotSpot 服务器将作为一个代理服务器使用的对所有被通用代理系统打断并没在**/ip proxy direct** 列表中定义的代理服务器地址。如果没有特别指明，地址将在**/ip proxy** 下面的 **parent-proxy** 参数定义。如果这个也空缺，请求将被本地代理处理。
login-by (*多选项:* cookie | http-chap | http-pap | https | mac | trial; default: **cookie,http-chap**) - 使用的认证方法

cookie - 使用 HTTPcookie 认证，而不询问用户证明。以防客户没有 cookie，或者存储的用户名和密码对从上一次认证后不再合法，就将使用其他方法认证。可能仅和其他 HTTP 认证方法一同使用(HTTP-PAP, HTTP-CHAP 或 HTTPS)，因为第一次 cookie 是没有办法产生的。

http-chap - 对密码使用 MD5 散列算法的 CHAP 询问-回答的模式。这种方法很容易避免在一个不安全网络上发送清楚的文本密码。这个方法是默认的认证方法。

http-pap - 在网络中使用纯文本认证。请注意如果使用了这个模式，你的用户密码将在本地网络中暴露，所有可够侦听它们。

https - 使用加密了的 SSL 通道来传输用户与 HotSpot 服务器的通信。注意，为了使它能工作，必须对路由器输入一个合法的认证。

mac - 试着先使用客户的 MAC 地址作为它的用户名。如果与本地用户数据库或 RADIUS 服务器匹配了，那么客户将不会被要求填写登陆表格就可以通过认证。

trial - 在一定时间内不会要求认证

RADIUS-accounting (yes | no; 默认: **yes**) - 是否不时地在每个用户上发送 RADIUS 帐户管理信息（这个“不时”的时间是在 **RADIUS-interim-update** 属性中定义的）
RADIUS-interim-update (*time* | received; 默认: **received**) - 发送累计帐户报告的频率

0s – 与 **received** 相同

received - 使用接收自 RADIUS 服务器的任何值

rate-limit (*文本*; 默认: **""**) - 从路由器角度考虑以 **rx-rate[/tx-rate] [rx-burst-rate[/tx-burst-rate] [rx-burst-threshold[/tx-burst-threshold] [rx-burst-time[/tx-burst-time]]]**格式表示的速率限制(其中"rx" 是客户上传，"tx"是客户下载)。所有的速率都应该是带有‘k’ (1,000s)或‘M’ (1,000,000s)的数字。如果 tx-rate 没有指定，rx-rate 和 tx-rate 一样。对于 tx-burst-rate 和 tx-burst-threshold 以及 tx-burst-time 也同理。如果 rx-burst-threshold 和 tx-burst-threshold 都没有指定 (但是 burst-rate 已指定)， rx-rate 和 tx-rate 将被做为 burst threshold 使用。如果 rx-burst-time 和 tx-burst-time 都没有指定，那么 1s 将会作为默认值使用。
smtp-server (*IP 地址*; 默认: **0.0.0.0**) - 默认 SMTP 服务器无条件地用于重定向
split-user-domain (yes | no; 默认: **no**) - 当用户名以"user@domain"或"domain\user"格式给出时，是否把用户名从域名中分离出来
ssl-certificate (*名称* | none; 默认: **none**) - 对 HTTPS 认证使用的 SSL 认证名，不用于其他认证方式
trial-uptime (*时间/时间*; 默认: **30m/1d**) - 仅当认证方式为询问时使用。
trial-user-profile (*名称*; 默认: **default**) - 仅当认证方法为询问时使用。指定询问用户将使用的用户概要
use-RADIUS (yes | no; 默认: **no**) - 是否使用 RADIUS 认证 HotSpot 用户

注：如果 dns-name 参数没有指定，则 hotspot-address 将代替使用，如果 hotspot-address 也没有指定，那么将自动在路由器本地选择这两个值。如果启用了 RADIUS 验证，/RADIUS 下的参数应正确配置。

属性描述

domain (*只读: 文本*) - 域名(如果从用户名中分离出来的话)
expires-in (*只读: 时间*) - cookie 合法存在的时间
mac-address (*只读: MAC 地址*) – 用户的 MAC 地址
user (*只读: 名称*) – 用户名

注：可以在相同的 MAC 地址上有多重的 cookie。例如，在同一台电脑上对每个 web 浏览器都可以有一个单独的 cookie。

Cookie 是会过期的，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。

21.5 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帐单选项。

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**) - 如果数据包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：通配符属性(**dst-host** 和 **dst-path**)匹配一个完整的串 （如：若设置为"example"，则它们不会匹配"example.com")。可用的通配符为'*' (匹配任意字符的任意数量)并且 '?' (匹配任何一个字符)。正则表达式也在这里接受，但如果属性做为一个正则表达式对待，那么它应该以图标(':')开始。

关于使用正则表达式：：

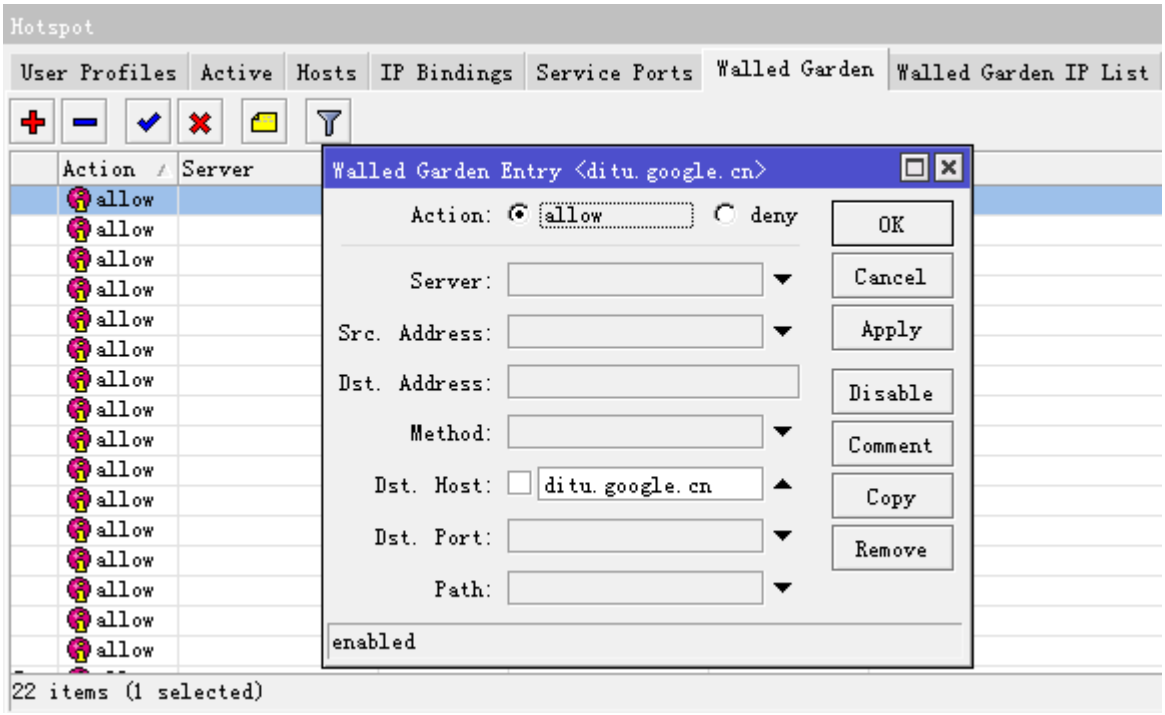
- **** 符号序列是用于在控制台输入****字符的
- **\.** 样式的意思为只是 **.** (在正则表达式单独的点表示任何符号)
- 显示在给出样式之前任何符号都不允许，我们在样式开始使用**^**符号
- 指定在给出样式之后任何符号都不允许，我们在样式结束的地方使用符号**$**

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）。

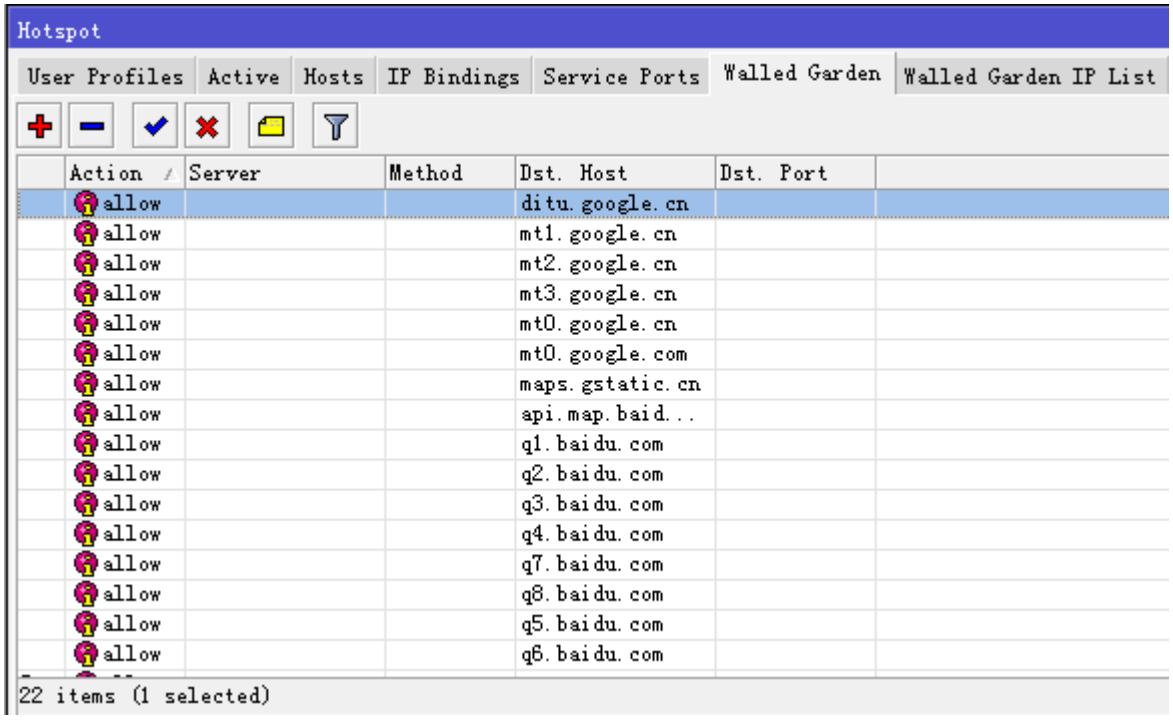
允许未认证用户到 **www.example.com** 域**/paynow.html** 页面的请求：

```
[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 需要访问的域名列表



IP 方式 Walled Garden

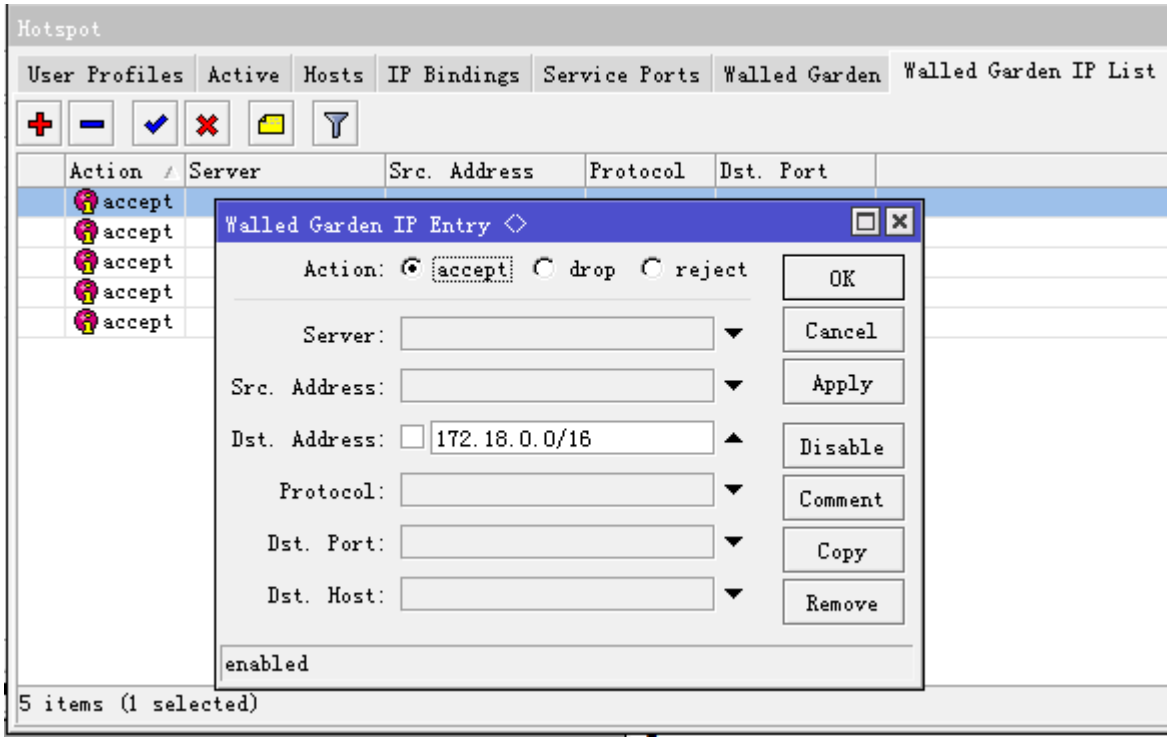
操作路径: ***/ip hotspot walled-garden ip***

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**) -如果数据包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>
```

21.6 Hotspot binding

操作路径: ***/ip hotspot ip-binding***

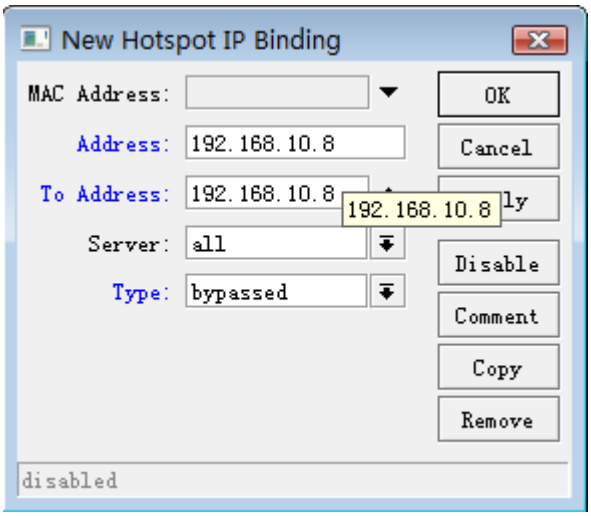
ip-binding 允许指定主机网络的静态规则，是否要求主机进行认证、绕过认证、阻止认证等。对该规则的网络主机设置源 IP 地址（或 IP 网络）或源 MAC 地址的 nat 翻译。你也可以允许一些地址绕过 HotSpot 认证（如：它们可以不必认证登陆就能访问外部资源），并阻止指定的地址认证登陆。

属性描述

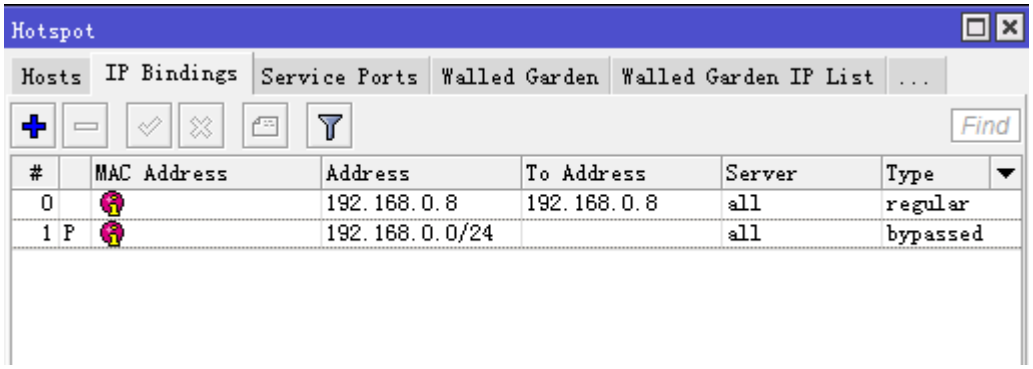
address (*IP 地址* /子网掩码; 默认: **""**) - 源 IP 地址或客户网络
mac-address (*MAC 地址*; 默认: **""**) - 客户的源 MAC 地址
server (*名称*|all; 默认: **all**) - 客户将连接到的服务器名
to-address (*IP 地址*; 默认: **""**) - 把原始客户地址翻译成的 IP 地址。如果 **address** 属性是作为一个网络给定，那么这个将是翻译的开始地址（例如：第一个 **address** 被翻译为 **to-address**，**address+1** 翻译为 **to-address+1**，以此类推）
type (regular | bypassed | blocked) – hotspot 指定静态绑定主机条目类型
- **regular** – 该规则要求做通过热点认证，并进行一对一 NAT 翻译
- **bypassed** – 绕过认证，即不需要通过 HotSpot 认证,扩音访问资源
- **blocked** - 不会执行 nat 翻译，即阻止该规则主机认证。

注：这是一个有序列表，即按照 FIFO 队列执行，所以你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。

下面是一给主机 192.168.10.8 在不通过认证情况下即可上网，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



21.7 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，这个列表包含所有一对一 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包（包括 ARP 回应，持活回应及用户流量）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登出。
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电脑在不可达状态多久会被自动登出
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包数
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？）

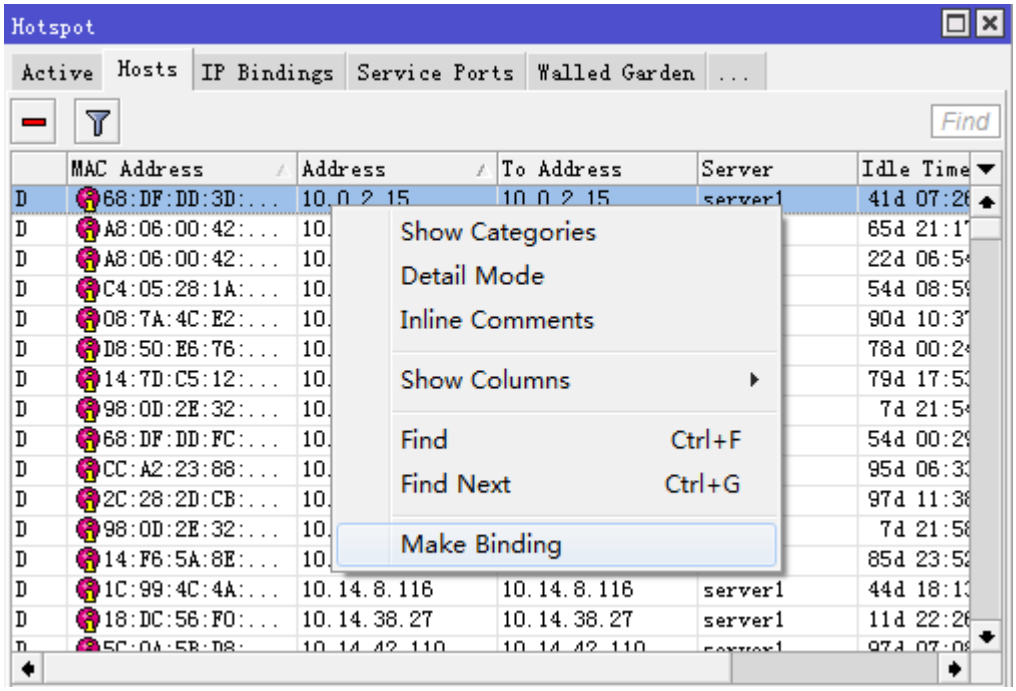
命令描述

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



21.8 HotSpot Users 管理

主要对 Hotspot 的用户帐号、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

address-pool (*名称* | none; 默认: **none**) – 为用户分配的地址池名称。
  none - 不向这个策略中的用户再分配 IP 地址池
advertise (yes | no; 默认: **no**) - 是否对此服务启用强制广告弹出
advertise-interval (*多选项: time*; 默认: **30m,10m**) - 显示广告弹出时间间隔的设置。
advertise-timeout (*time* | immediately never; 默认: **1m**) - 在使用 walled-garden 阻止网络访问之前等待广告显示的时间长度
advertise-url (*多选项: 文本*; 默认: **http://www.mikrotik.com/,http://www.routerboard.com/**) - 广告弹出显示的 URL 列表。这个列表是循环的推送。
idle-timeout (*time* | none; 默认: **none**) – 在线用户空闲超时时间（未活动状态的最长时间）。它用于探测用户是否在向外部网络或 hotspot 主机发送数据，比如：没有任何流量从用户进入或从路由器流出。当达到超时时间，用户会被注销，丢出主机在线列表，用户使用的地址也会被清空。
  none – 不切断空闲用户
incoming-filter (*名称*) - 应用于来自此策略组用户入方向数据包的防火墙链表名称
incoming-packet-mark (*名称*) – 应用于此策略组每个用户所有数据包入方向的包标记

keepalive-timeout (*time* | none; 默认值: **00:02:00**) – 在线用户的活动超时时间。用于探测用户的主机是否在线。如果在这个期间检测失败，那么用户会被注销，用户使用的地址也会被清空。

none – 关闭此功能。

name (*名称*) – 策略参考名

on-login (*文本*; 默认: **""**) - 用户登录后运行的脚本名

on-logout (*文本*; 默认: **""**) -用户注销后运行的脚本名

open-status-page (always | http-login; 默认值: **always**) - 是否为授权用户显示状态页面使用 MAC 登入方法。如果你想放一些信息（例如：弹出窗口）在 alogin.html 页面将会很有用，这样所有的用户都可以看到它。

http-login - 如果 http 登入打开状态页面（包括 cookie 和 http 登入方法）

always - 如果 mac 登入打开 http 状态页面

outgoing-filter (*名称*) - 应用于此服务用户的向外流出的包的防火墙链表名称

outgoing-packet-mark (*名称*) - 自动设置在此概要每个用户的所有数据包的包标记

rate-limit (*文本*; 默认: **""**) – 从路由器角度来看的 **rx-rate[/tx-rate]** 格式的速率限制。

[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]]]] (所以 "rx" 客户的上传， "tx"客户的下载)。所有速率必须以可选的'k' (1,000s) 或 'M' (1,000,000s)计算。如果 tx-rate 没有指定，则 rx-rate 和 tx-rate 一样。对于 tx-burst-rate 和 tx-burst-threshold 以及 tx-burst-time 也同理。如果 both rx-burst-threshold 和 tx-burst-threshold 都没有指定（但 burst-rate 指定了），那么 rx-rate 和 tx-rate 会作为脉冲串门限使用。如果 rx-burst-time 和 tx-burst-time 都没有指定，那么 1s 将设置为默认值。优先级从 1 到 8 取值，1 代表最高优先级，而 8 代表最低的。如果 rx-rate-min tx-rate-min 都没有指定那么 rx-rate 和 tx-rate 的值将被使用。rx-rate-min 和 tx-rate-min 的值不能超过 rx-rate 和 tx-rate。

session-timeout (*time*; 默认: **0s**) – 用户的会话超时时间(会话最大允许时间)，当该时间到后，用户将会被自动剔除在线列表，如果没有其他时间和流量限制，用户可以再次认证登录。

0 – 不剔除

shared-users (*整型*; 默认: **1**) – 同一用户账号，同时可以登录的最大数量，用于共享账号的数量设置;

status-autorefresh (*time* | none; 默认: **none**) – 热点 servlet 状态页面自动刷新间隔

transparent-proxy (yes | no; 默认: **yes**) - 是否对认证后的用户使用透明的 HTTP 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 下的用户账号信息。

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**) – 用户资料
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>
```

21.9 Hotspot active 在线管理

操作路径： ***/ip hotspot active***

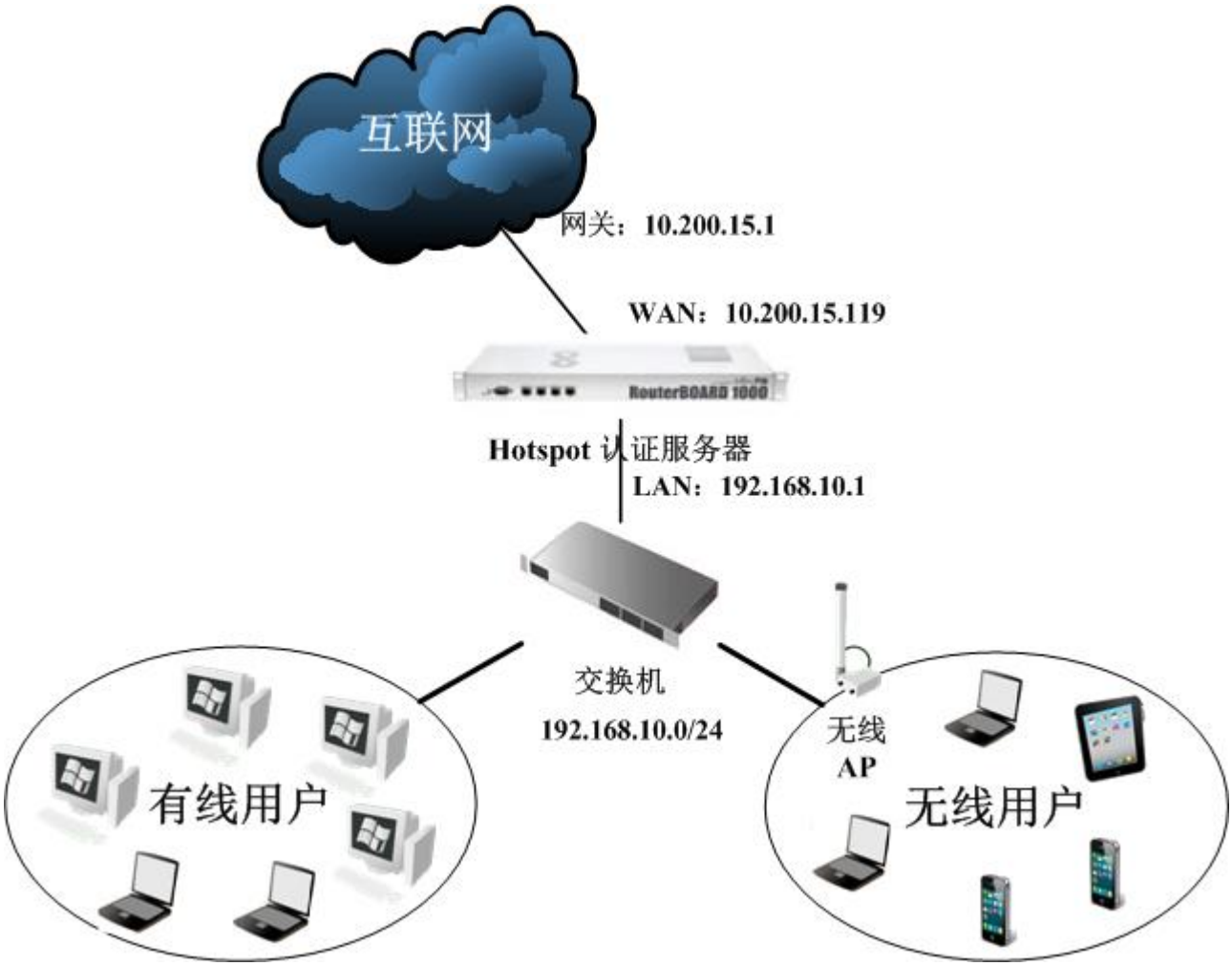
Active 列表及在线管理，显示用户当前已登陆了的信息。这里不能修改任何信息，除了使用 **remove** 命令用于剔除在线用户。

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登出的闲置时间
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电脑不可达多久才会被自动登出
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

21.10 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：

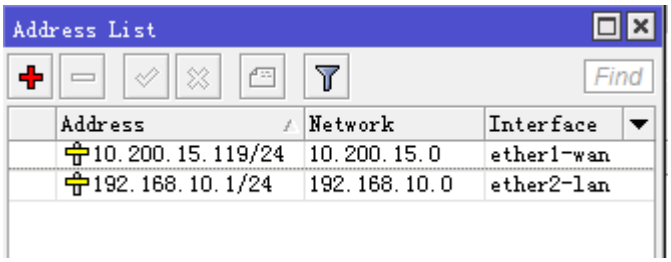


一个网关路由器的网络参数如下：

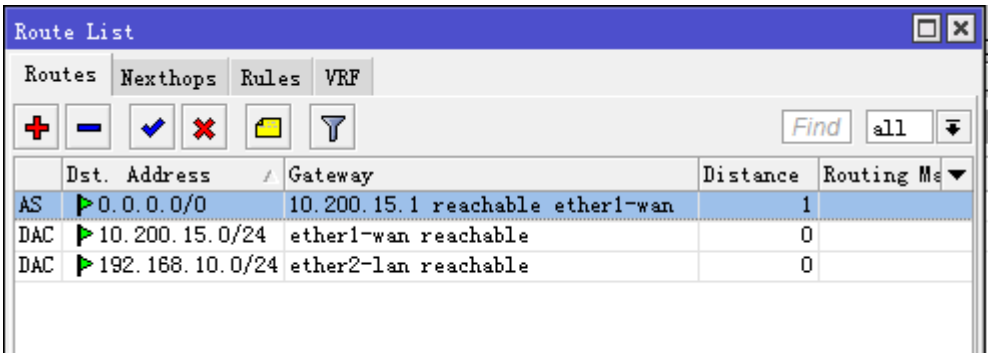
WAN 口对应外网 IP 为 10.200.15.119/24，网关为 10.200.15.1
LAN 口对应内网 IP 为 192.168.10.1/24
DNS：61.139.2.69

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。

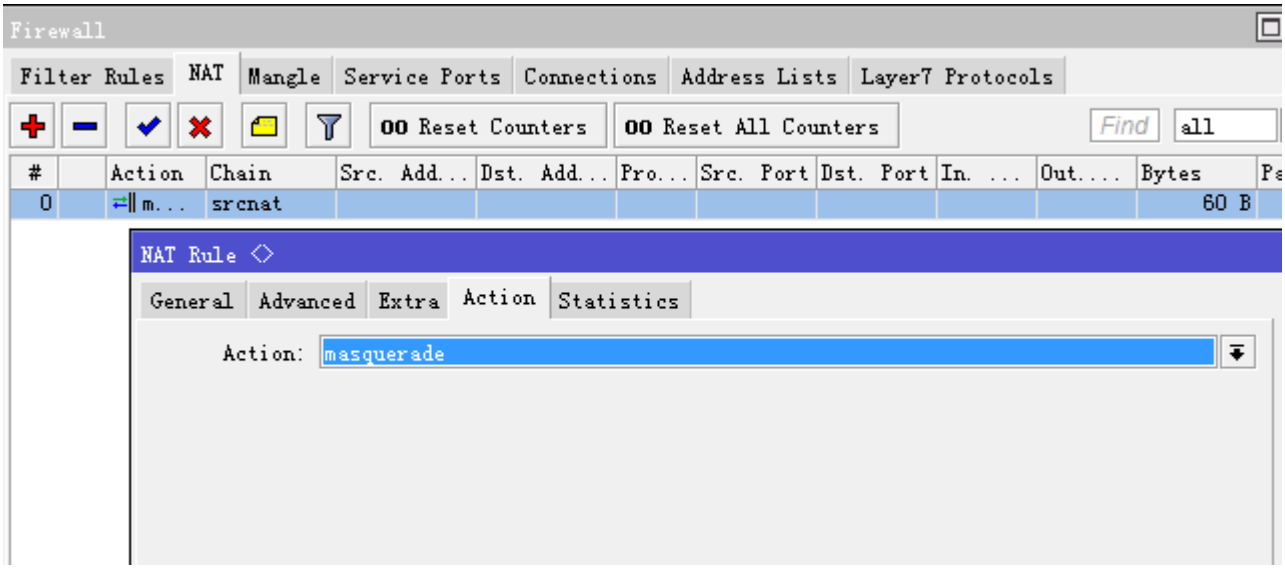
进入 ip address 配置 IP 地址：



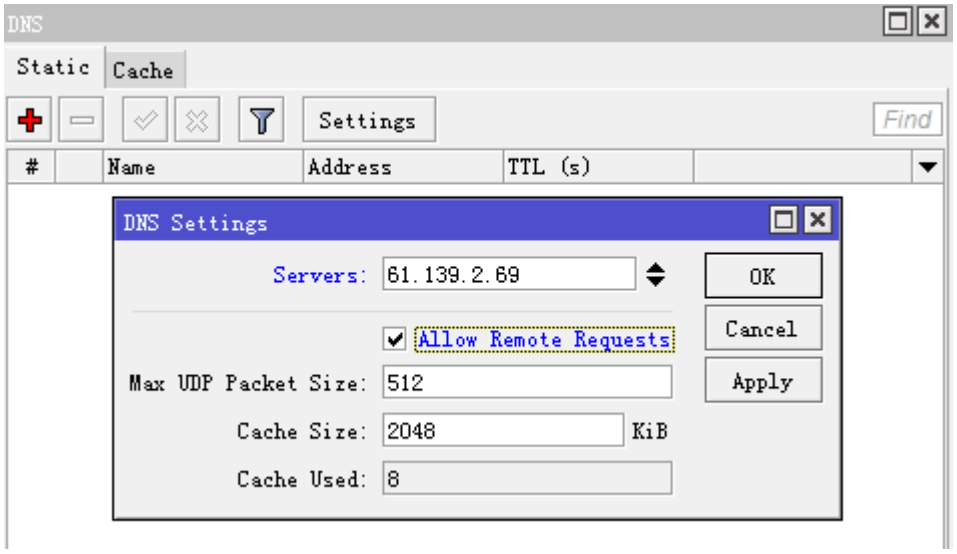
进入 ip route 配置网关



进入 ip firewall nat 设置好 NAT 伪装：



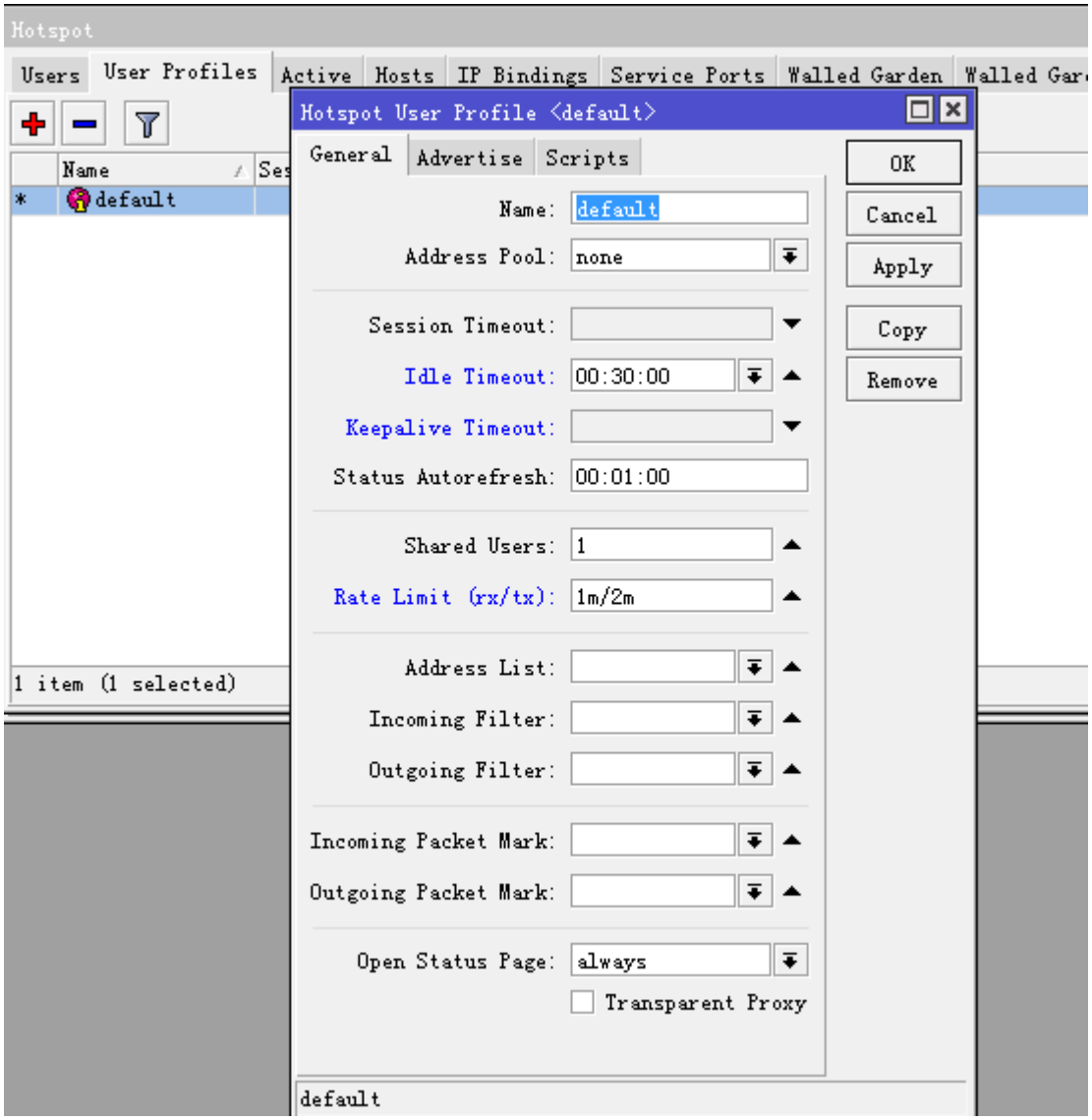
进入 ip dns 配置 DNS 缓存，DNS 和启用缓存对于 Hotspot 非常重要，如果 DNS 错误将导致用户认证跳转或无法浏览网页：



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 添加用户的帐号
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

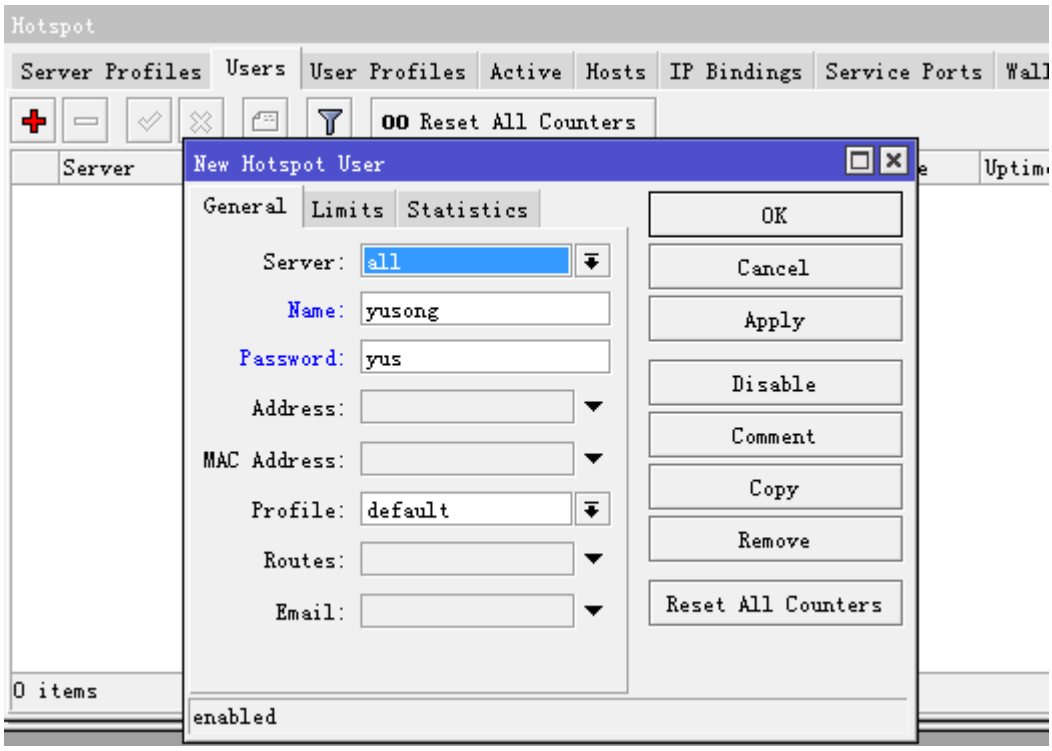


在 user profile 里面一般配置如下几个参数：

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：帐号的分享用户多少，默认为 1，即仅一个用户使用该帐号。
Rate-Limit：分配每个帐号带宽，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。

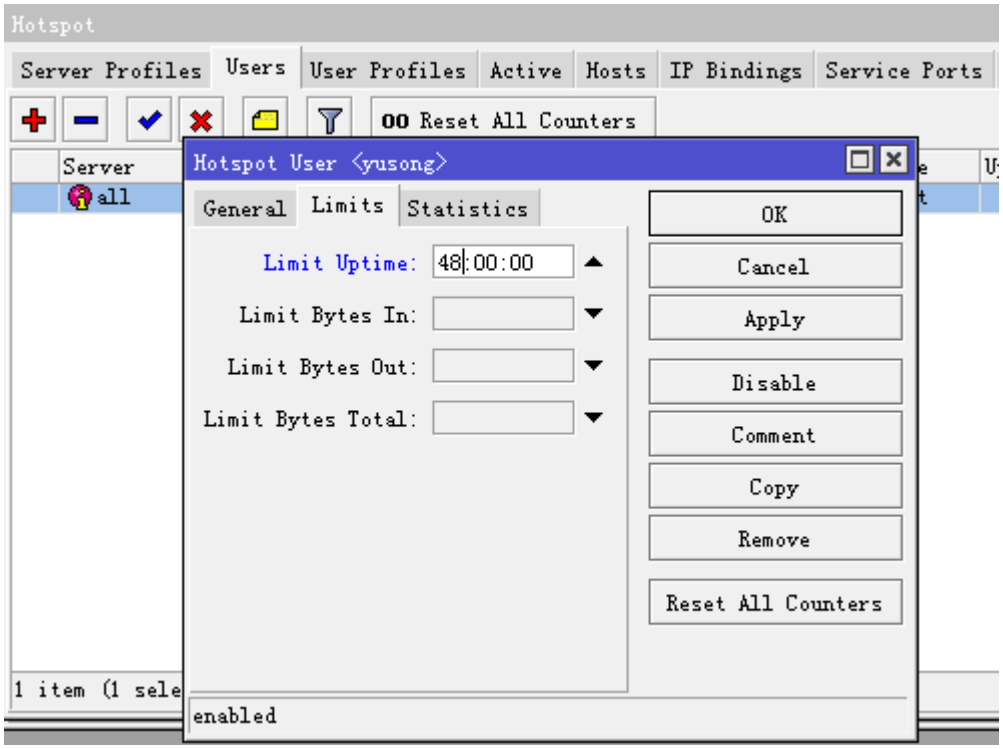
Address pool 这个是 DHCP 的地址池，给用户分配 IP，我们可以在 ip pool 中分配地址段，具体操作请参考 RouterOS 的 DHCP 操作。

在 user 配置用户登录帐号和密码，以及所属的 profile 类型，这里默认 `server` 服务器为 `all`：

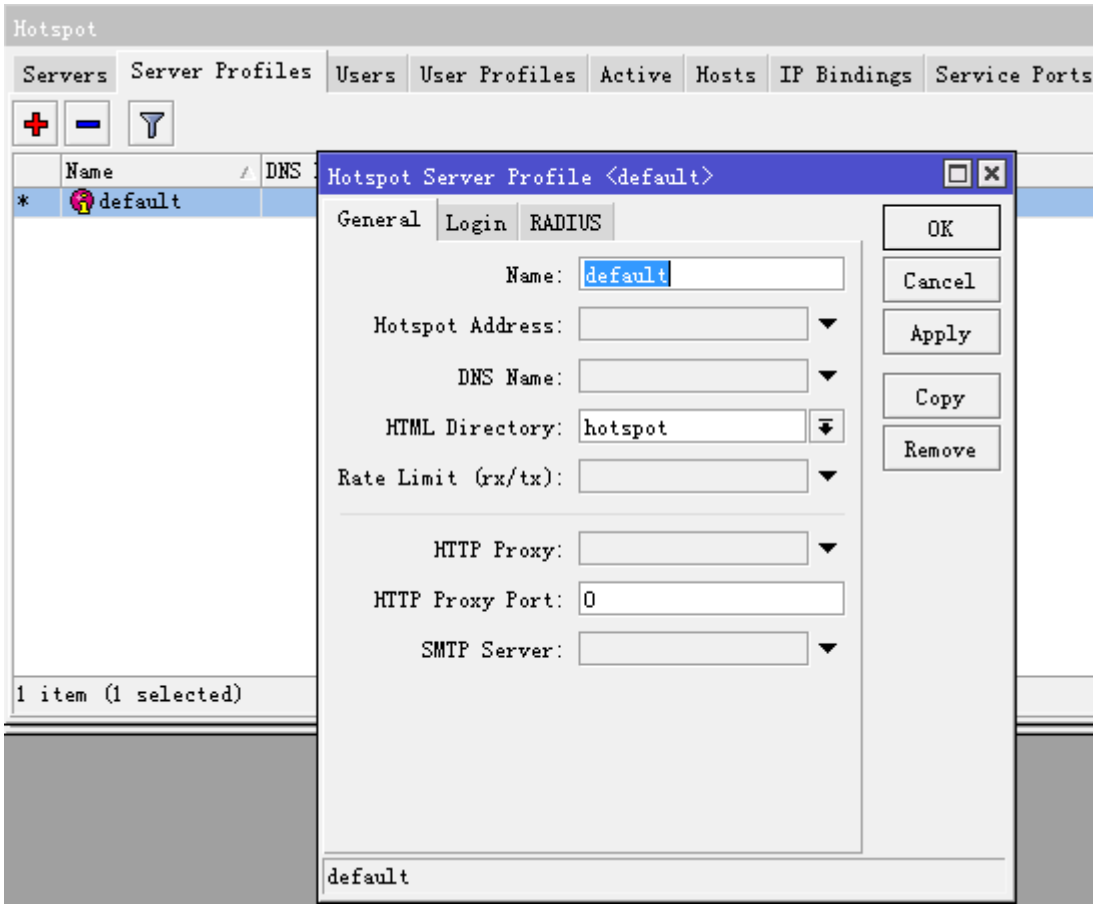


Name：用户名为 yusong
Password：密码为 yus
Profile：用户组规则，这里选择我们之前设置的 default 规则

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



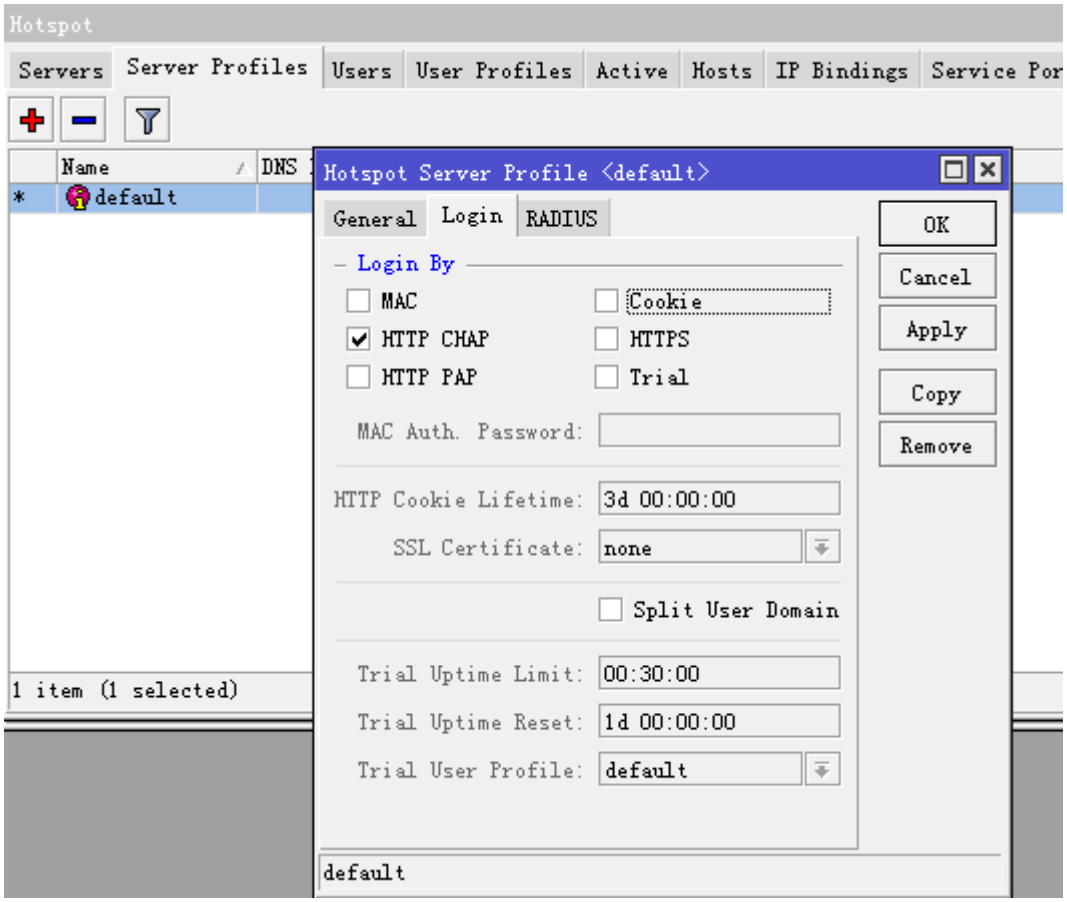
配置完用户规则后，进入 ip hotspot server profile，配置服务器器规则。



这里有几个参数我们需要说明：

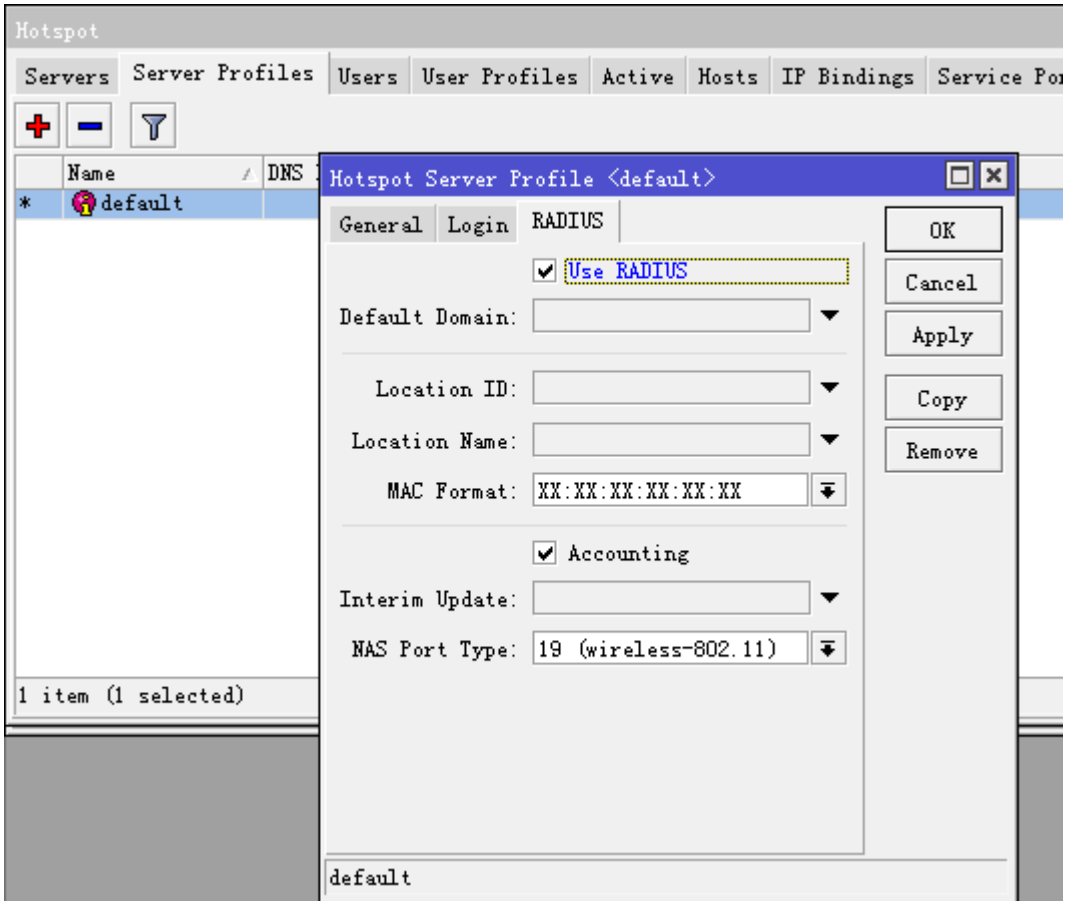
- **Hotspot address**：认证服务器的 IP 地址，如果你只有一个 hotspot 认证接口，这个参数可以默认不设置，但当你有多个接口的 hotspot 接口，且每个接口不同 IP 地址段，为了用户能都能同时访问一个 IP 地址认证，我们可以设置一个静态的地址，这样可以便于网络的管理。
- **HTML Directory**：该参数是指定 Hotspot 的认证也路径（在 Files 目录下可以找到），当你拥有多个 hotspot 认证接口时，你对不同接口的用户选择不同的认证页面，我们仅需要建立多个 server Profiles 规则。
- **Rate Limit**：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，一般默认启用 HTTP CHAP 即可，

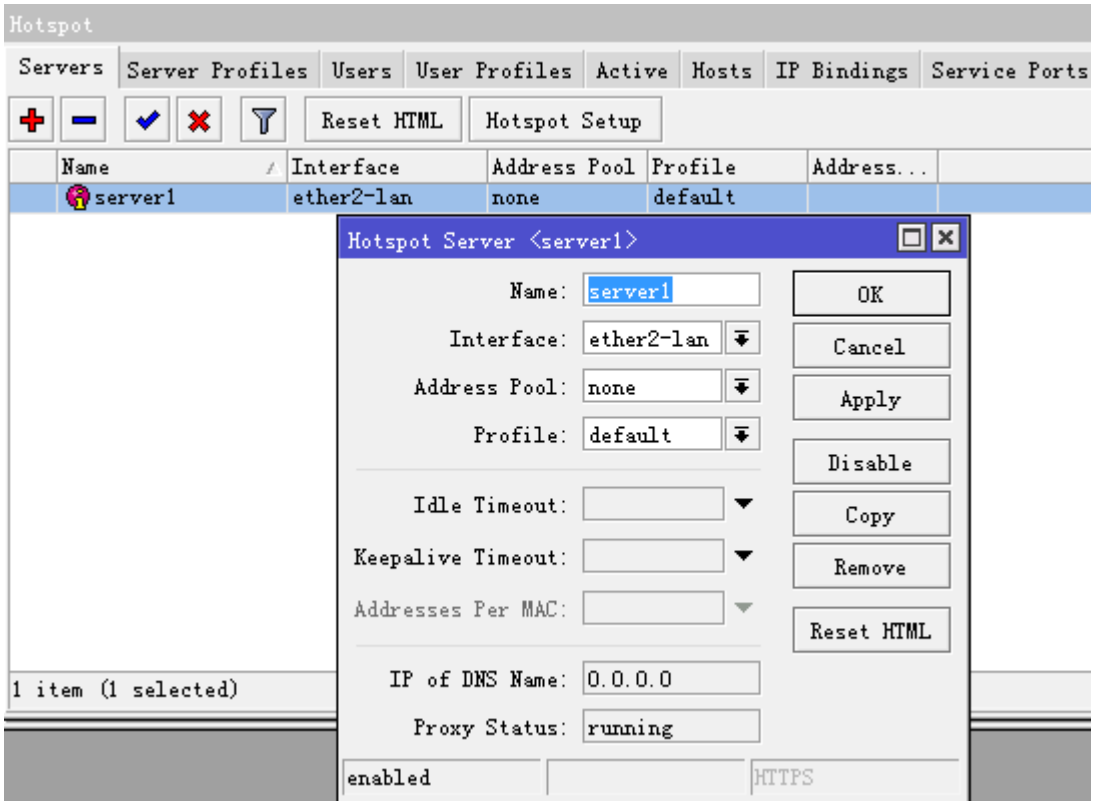


Cooke 等值一般不用选择，这里选择 HTTP CHAP 加密方式提交账号密码，HTTP PAP 是明文的账号密码提交。

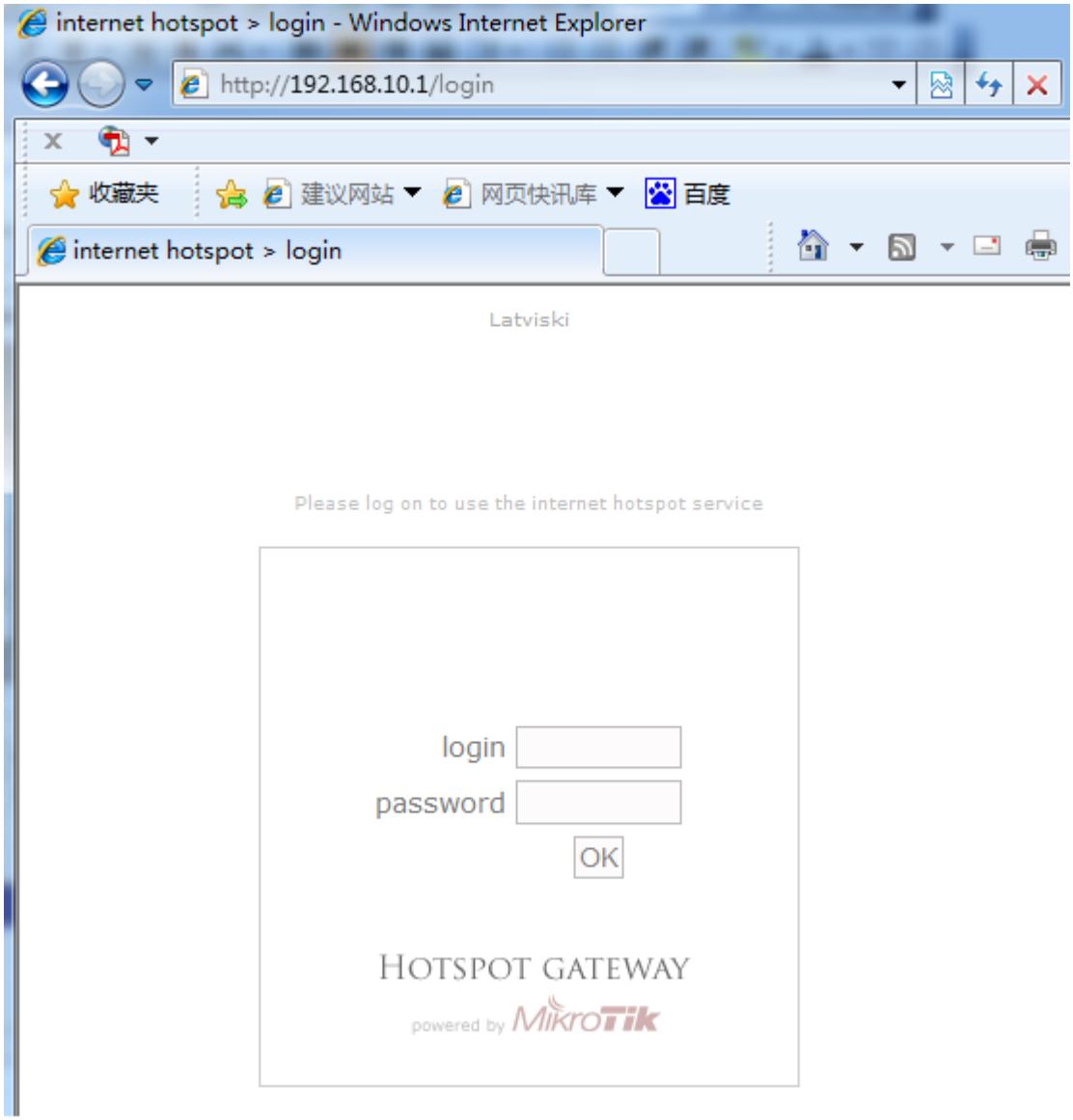
至于 RADIUS 根据需要开启，如果你有对应的 RADIUS 服务器，并要使用 RADIUS 计费，这里的 Use RADIUS 必须选择



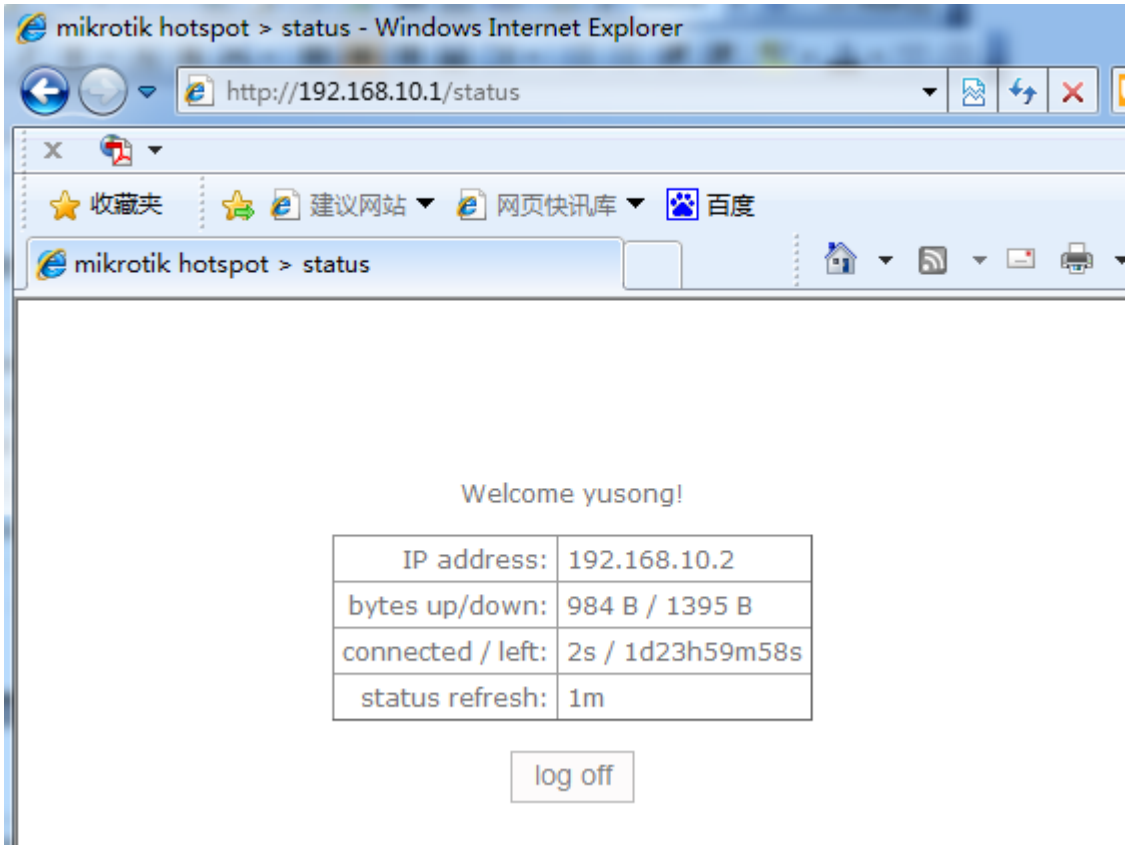
配置完成以上参数后，最后我们启用 Hotspot 服务器，并选择 interface 为我们的 lan 接口：



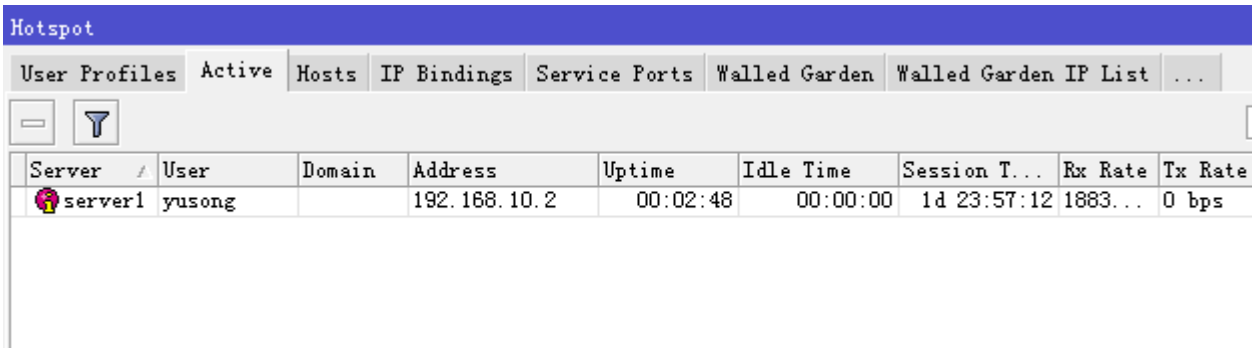
当启用完成后，所有对路由器或者外网访问都需要通过 web 认证，当用户随便输入一个网站都会跳转到认证页面，用户访问到的认证页面如下图：



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：



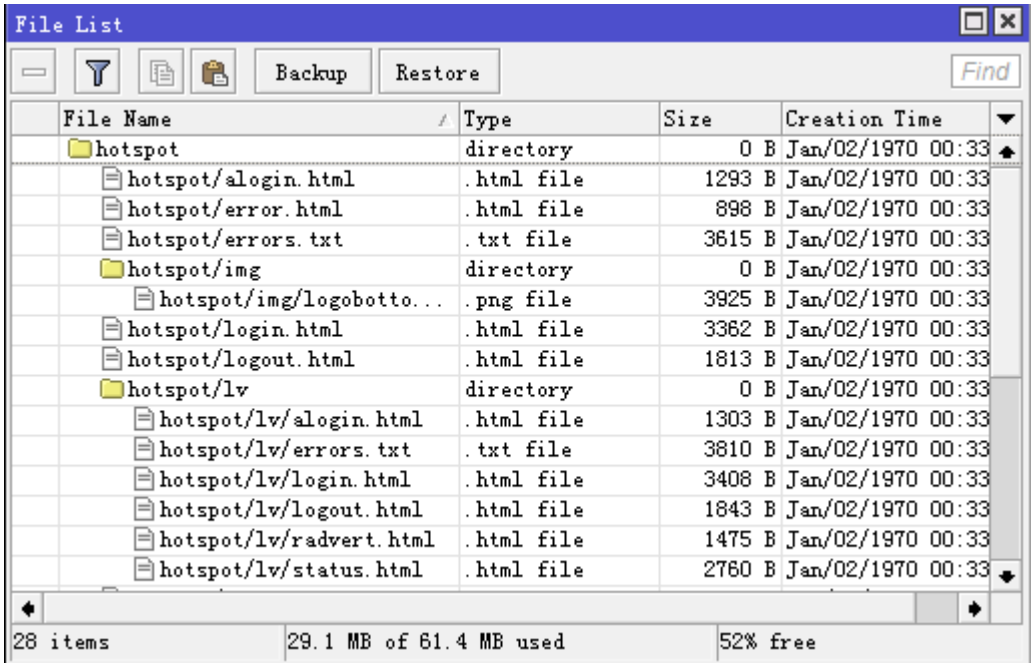
获取现时用户列表：

```
[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 界面得到你想要的网页画面或者 log。

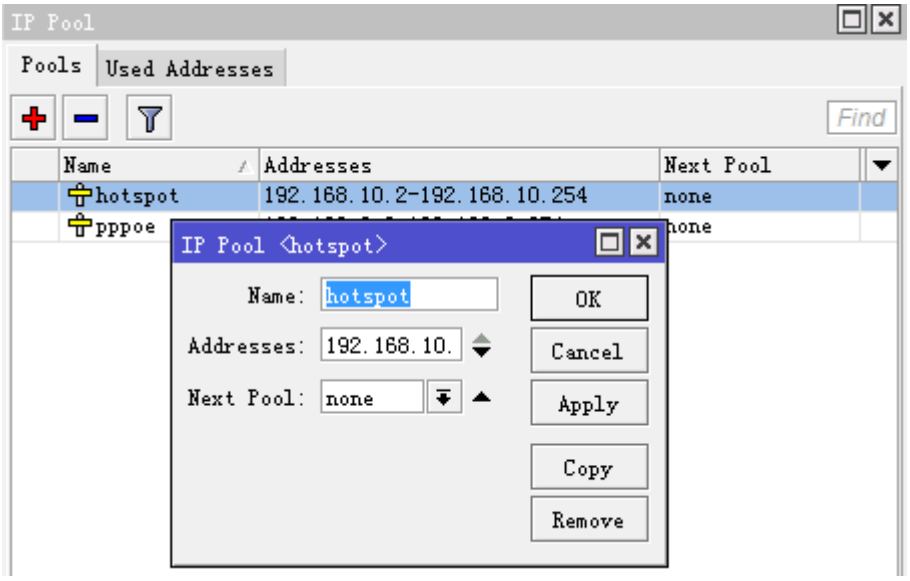
Hotspot 的 html 文件提供了较开放的设置，能实现网页认证的要求，如首次登陆看见一个页面，然后在跳转到认证认页、认证后在到另外一个页面等，都可以通过修改 html 文件代码实现。

21.11 Hotspot 即插即用功能

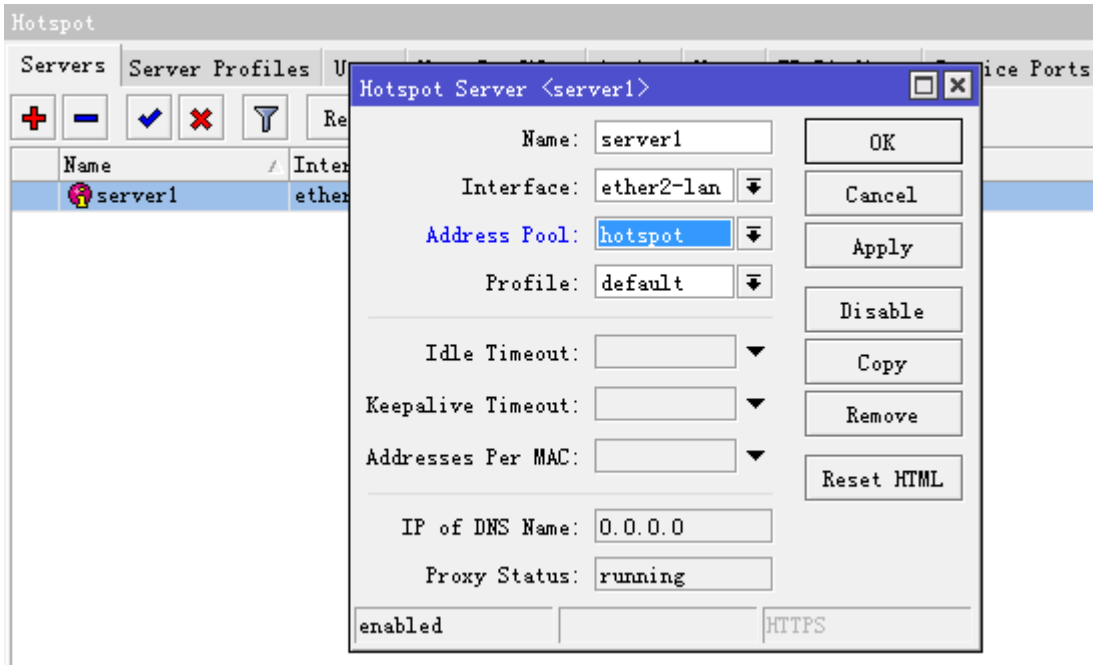
从 2.7 的版本就开始支持 upnp 的即插即用功能，即当用户和 Hotspot 认证服务器在同一局域网内，不管局域网用户设置任何的 IP 地址（前提是用户必须设置任意的 IP 地址、网关和 DNS）都可以被 Hotspot 认证服务器获取，并在 Hotspot 的 Host 中分配一个新的虚拟 IP 地址，并对用户作一对一的 NAT 转换。Hotspot 的即插即用方式分成适用于：流动性较强的公共场所，如机场、车站、公园，也可以应用到酒店和小区中。

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：

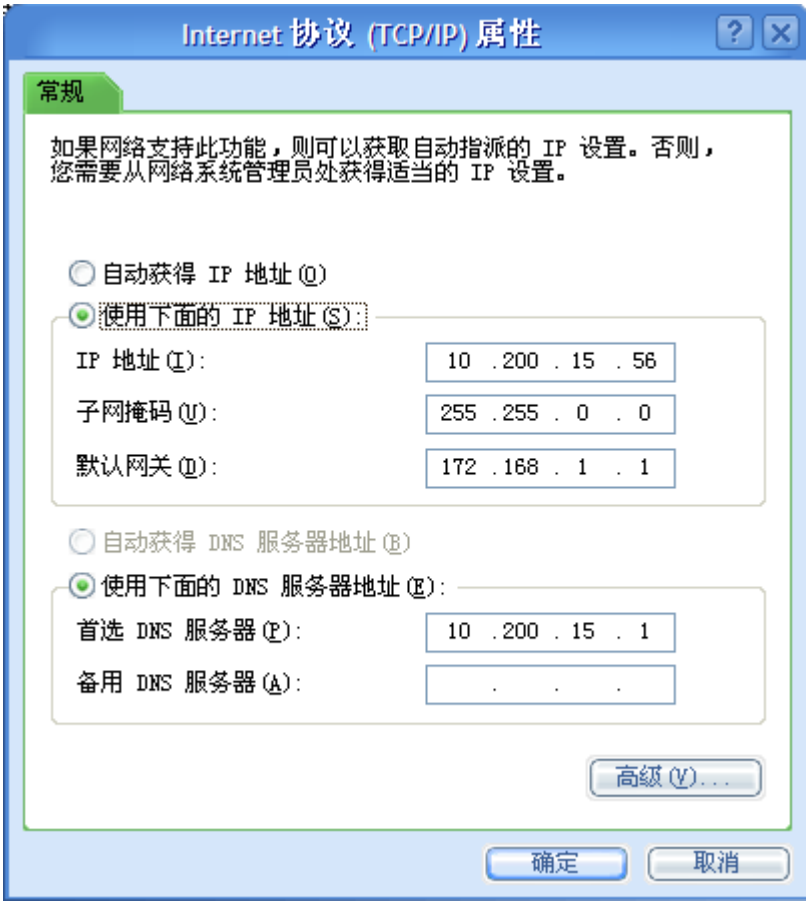


然后在 hotspot Severs 中选择 pool

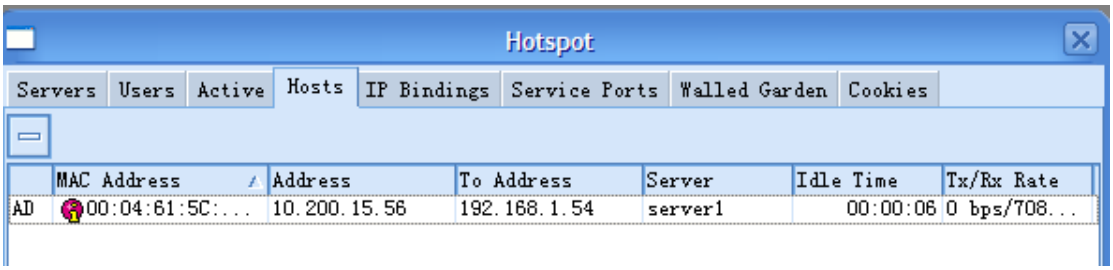


Addresses Per MAC 这个是每个 IP 对应的 MAC 地址，这里我们设置为 1，即一个 IP 对应一个 MAC 地址。

我们 windows 电脑的 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**

21.12 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激活一个 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
```

把对数据包的所有 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
```

所有其他的数据包除了 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包通过 **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 标志对数据包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电脑配置的 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电脑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包都将被发送到 **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。

第二十二章 PPPoE 配置

PPPoE 基于以太网的点对点协议(Point to Point Protocol over Ethernet)当前的 PPPOE 主要被 ISP 商用于 xDSL 和 cable modems 与用户端的连接，他们几乎与以太网一样。PPPoE 是一种标准的点对点协议(PPP) 他们之间只是传输上的差异：PPPoE 使用 modem 连接来代替普通的以太网。一般来说，PPPoE 是基于与用户认证和通过分发 IP 地址给客户端。

RouterOS 能做一个的 RADIUS 客户端 – 你能使用一台 RADIUS 服务器去验证 PPPoE 的客户端和对他们计费

一个 PPPoE 连接由客户端和一个访问集线服务器组成，客户端可以是一个安装了 PPPoE 协议的 windows 电脑。PPPoE 客户端和服务器能工作在任何以太网等级的路由器接口（interface） - wireless 802.11 (Aironet, Cisco, WaveLan, Prism, Atheros), 10/100/1000 Mbit/s Ethernet, RadioLan 和 EoIP (Ethernet over IP tunnel)都支持。

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 和大数据包）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。

注：当 RADIUS 服务器验证一个用户 CHAP、MS-CHAPv1 或 MS-CHAPv2，RADIUS 戏院不会使用共享密码（shared secret），仅验证回复（authentication reply）被使用。因此如果你有一个错误的共享密码，RADIUS 服务器将接受请求。你可以使用**/RADIUS monitor** 命令查看 **bad-replies** 参数，无论什么时候客户在试图连接时这个值都会增加。

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（如果限流被使用额外还需要增加 10KB），*

22.1 PPPoE Client 设置

操作路径: ***/interface pppoe-client***

属性描述

name (*名称*; 默认: **pppoe-out1**) –PPPoE 的接口名称
interface (*名称*) – 选择 PPPoE 服务器的接口使连接通过

mtu (*整型*; 默认: **1480**) – 最大传输单位。最适合的 MTU 值（以避免以太网连接的 1500-byte，设置为 1480 以避免数据包的重复存储）
mru (*整型*; 默认: **1480**) – 最大接收单位。最适合的 MRU 值（以避免以太网连接的 1500-byte，设置为 1480 以避免数据包的重复存储）
user (*文本*; 默认: **""**) – 连接在 PPPoE 服务器的用户帐号。
password (*文本*; 默认: **""**) – 连接 PPPoE 服务器的用户密码
profile (名称) – 连接的默认策略
allow (*多选项:* mschap2, mschap1, chap, pap; default: **mschap2, mschap1, chap, pap**) – 客户端使用的验证协议
service-name (*文本*; 默认: **""**) – 在访问集线器上设定指定服务名（AC）
ac-name (*文本*; 默认: **""**) – 这条可以为空白，当客户端与任何一个访问集线器相连，会选取该服务名。
add-default-route (yes | no; 默认: **no**) – 是否添加动态默认路由。
dial-on-demand (yes | no; 默认: **no**) – 当连接唯一的 AC 时，传输数据产生，在断开连接，没有传输数据时，idle-timeout 将被设置。
use-peer-dns (yes | no; 默认: **no**) – 是否分配对端服务器 DNS 在路由器本地"ip dns setting"中

注：如果存在一条默认的 pppoe 的路由，add-default-route 将不会创建一个新的路由

在 **gig** 接口上添加和启用客户端客户端，连接 AC 提供的 **testSN** 服务名，使用的用户帐号 **john** 和密码 **password** ：

```
[admin@RemoteOffice] interface pppoe-client> add interface=gig \
\... service-name=testSN user=john password=password disabled=no
[admin@RemoteOffice] interface pppoe-client> print
Flags: X - disabled, R - running
  0   R name="pppoe-out1" mtu=1480 mru=1480 interface=gig user="john"
        password="password" profile=default service-name="testSN" ac-name=""
        add-default-route=no dial-on-demand=no use-peer-dns=no
```

监视 PPPoE 客户端

命令名称: ***/interface pppoe-client monitor***

属性描述

status (文本) – 客户端的状态
  Dialing – 拨号连接的情况
  Verifying password... – 确认连接到服务器，密码正在核对处理
  Terminated – 接口没有启用，或是另一端未建立连接
encoding (*文本*) – 在该条连接中使用加密和编码。
uptime (*时间*) – 连接时间显示为天、时、分、秒
service-name (*文本*) – 客户端连接的服务器名称
ac-name (*文本*) – 客户端已经连接到的 AC 名称
ac-mac (*MAC 地址*) – 客户端已经连接的访问集线器（AC）MAC 地址。

监视 **pppoe-out1** 连接情况：

```
[admin@MikroTik] interface pppoe-client> monitor pppoe-out1
           status: "connected"
```

```
            uptime: 10s
          encoding: "none"
      service-name: "testSN"
           ac-name: "10.0.0.1"
            ac-mac: 00:C0:DF:07:5E:E6

[admin@MikroTik] interface pppoe-client>
```

22.2 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。

22.3 PPPoE Server 设置

操作路径: ***/interface pppoe-server server***

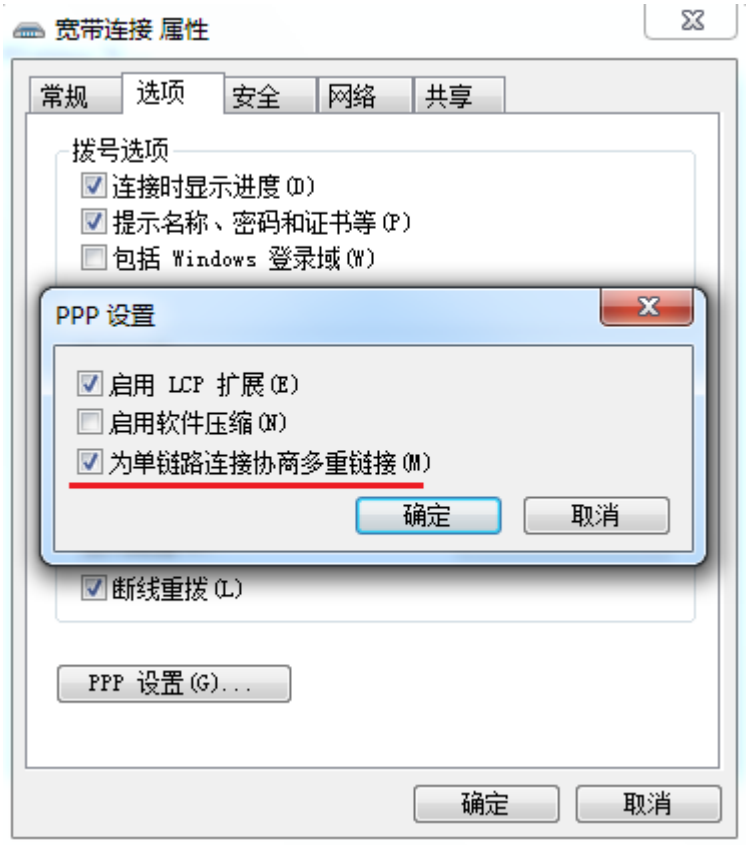
PPPoE server (access concentrator)支持在每一个接口上的多服务，需要设置不同的 service 名称 ，当前 PPPoE server 的吞吐量在一个 Celeron 600 CPU 测试达到 160 Mb/s ，如果使用更高性能的 CPU，吞吐量将会程比例的增加。

service-name (*文本*) – PPPoE 服务名称
mtu (*整型*; 默认: **1480**) – 最大传输单位。最适合的 MTU 值（以避免以太网连接的 1500-byte，设置为 1480 以避免数据包的重复存储）
mru (*整型*; default: **1480**) – 最大接受单位。最适合的 MRU 值（以避免以太网连接的 1500-byte，设置为 1480 以避免数据包的重复存储）
mrru (*整型*: 512..65535; 默认: **disabled**) – 在连接中能被接收的最大数据包长度。如一个数据包比隧道的 MTU 值大时，将会被分割到多个数据包中，允许实际大小的 IP 或以太网的数据包发送到隧道。
authentication (多种选择：mschap2 | mschap1 | chap | pap; 默认: **mschap2, mschap1, chap, pap**) – 验证算法
keepalive-timeout – 定义时间周期(秒) 连接开始后路由器每秒钟会发出 keepalive 数据包。如果在设定的时间周期内没有传输和没有 keepalive 回应，客户端将会被认为失去连接。
one-session-per-host (yes | no; 默认: **no**) – 每次只允许一个主机对话连接 (MAC 地址被确定). 如果主机将试着去建立一个新的对话连接，旧的一个将会被关闭。
default-profile (*名称*; 默认: **default**) – 使用默认的策略配置
max-sessions (*整形*; 默认: **0**) – AC 能服务的最大客户端数量
  0 – 没有限制
interface (*名称*) – 客户端连接的网卡接口

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帐号才会断开。解决这个问题，one-session-per-host 属性需启用。

安全提示：请不要分配一个 IP 地址到 PPPoE 的物理网卡上，避免出现用户不通过 PPPoE 验证即可上网的情况。

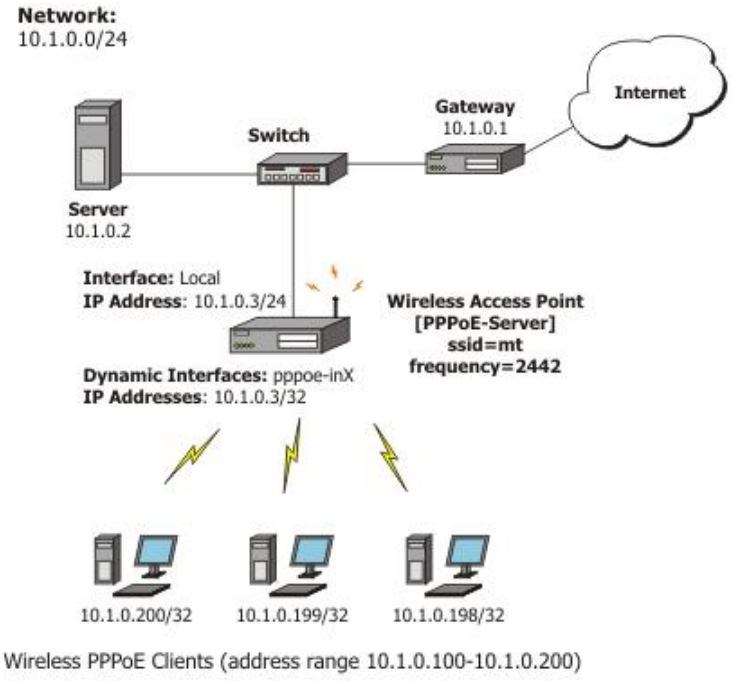
明确的讲 MRRU 意思为基于单连接的 MP，该协议被为拆分大数据包为更小的。在 windows 下在网络属性下，设置按钮中打开“为单链路连接协商多重连接”MRRU 是强行设置为 1614。这个设置有益于超载线路 MTU 探测失败。且 MP 协议应在双方都被启用。



22.4 基于 802.11g 无线网络的 PPPoE 服务

在无线网络中，服务器可以设置在一个访问节点（Access Point）,任意一个 RouterOS 客户端或是 Windows 客户端都可以连接到访问节点的 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 传输数据包，并避免 MTU 比 1500 低出现的任何问题。

让我们考虑下面的配置，MikroTik 无线 AP 能使无线用户端通过验证后访问到本地的网络：



首先，需要配置无线网卡：

```
[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
[admin@PPPoE-Server] ip route> /interface ethernet
[admin@PPPoE-Server] interface ethernet> set Local arp=proxy-arp
[admin@PPPoE-Server] interface ethernet> print
Flags: X - disabled, R - running
 #     NAME                                  MTU   MAC-ADDRESS       ARP
 0   R Local                                 1500  00:0C:42:03:25:53 proxy-arp
[admin@PPPoE-Server] interface ethernet>
```

添加 PPPoE server 到无线网卡上:

```
[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>
```

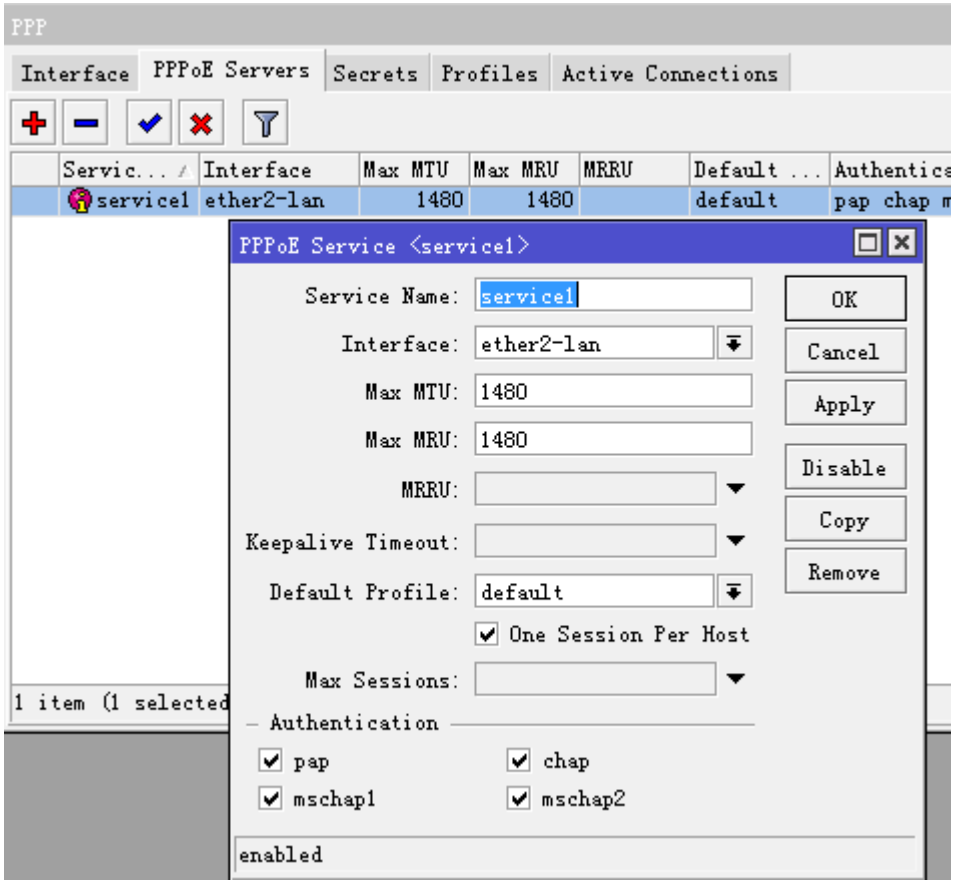
最后，设置 PPPoE clients:

```
[admin@PPPoE-Server] ip pool> add name=pppoe ranges=10.1.0.100-10.1.0.200
[admin@PPPoE-Server] ip pool> print
 # NAME                                      RANGES
 0 pppoe                                     10.1.0.100-10.1.0.200
[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
[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>
```

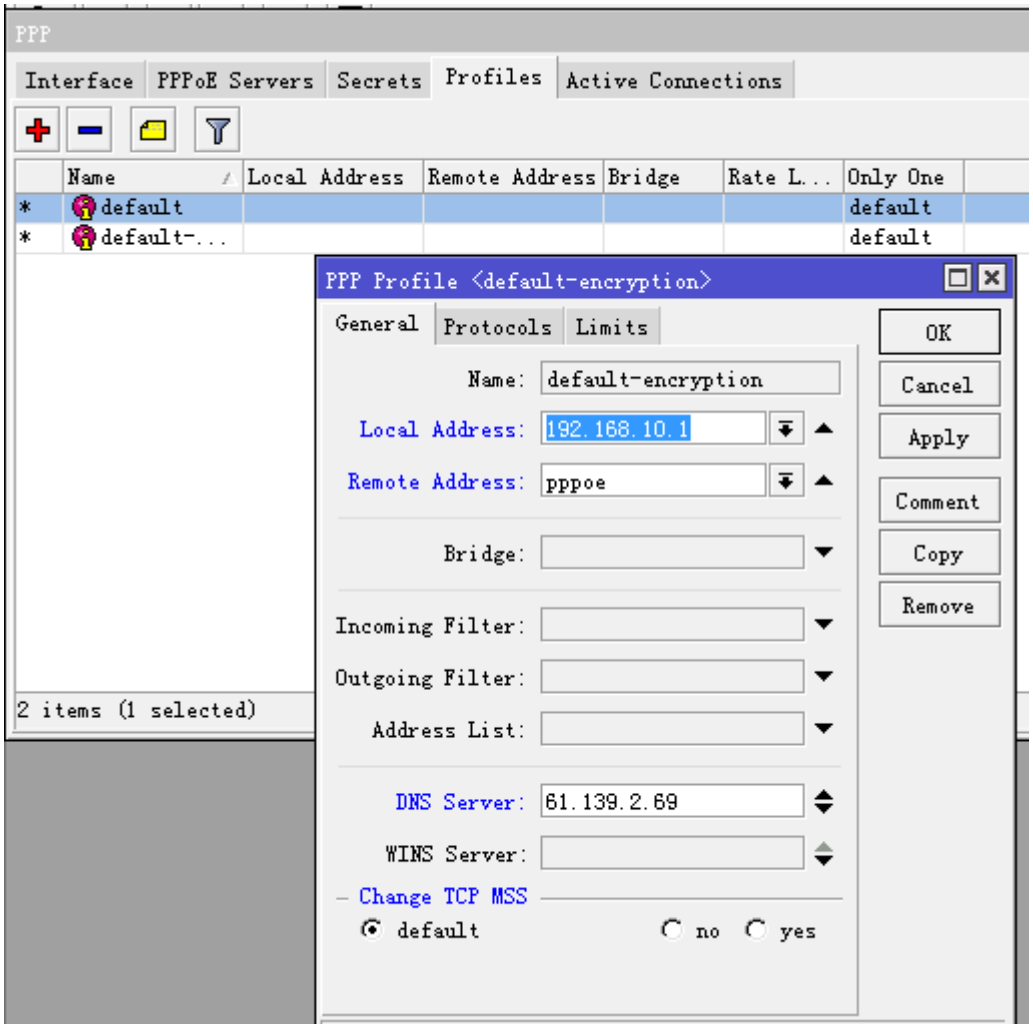
注: 在 Windows XP 中的 PPoE 客户端内建加密功能，但 RASPPPOE 没有。因此，如果计划不在支持比 Windows XP 老的 Windows 客户端，推荐在 **default** 规则配置把 **require-encryption** 值选择位 **yes**。在其他一些应用中，可以服务设置为接受为加密的数据。

22.5 Winbox 配置 PPPoE 服务

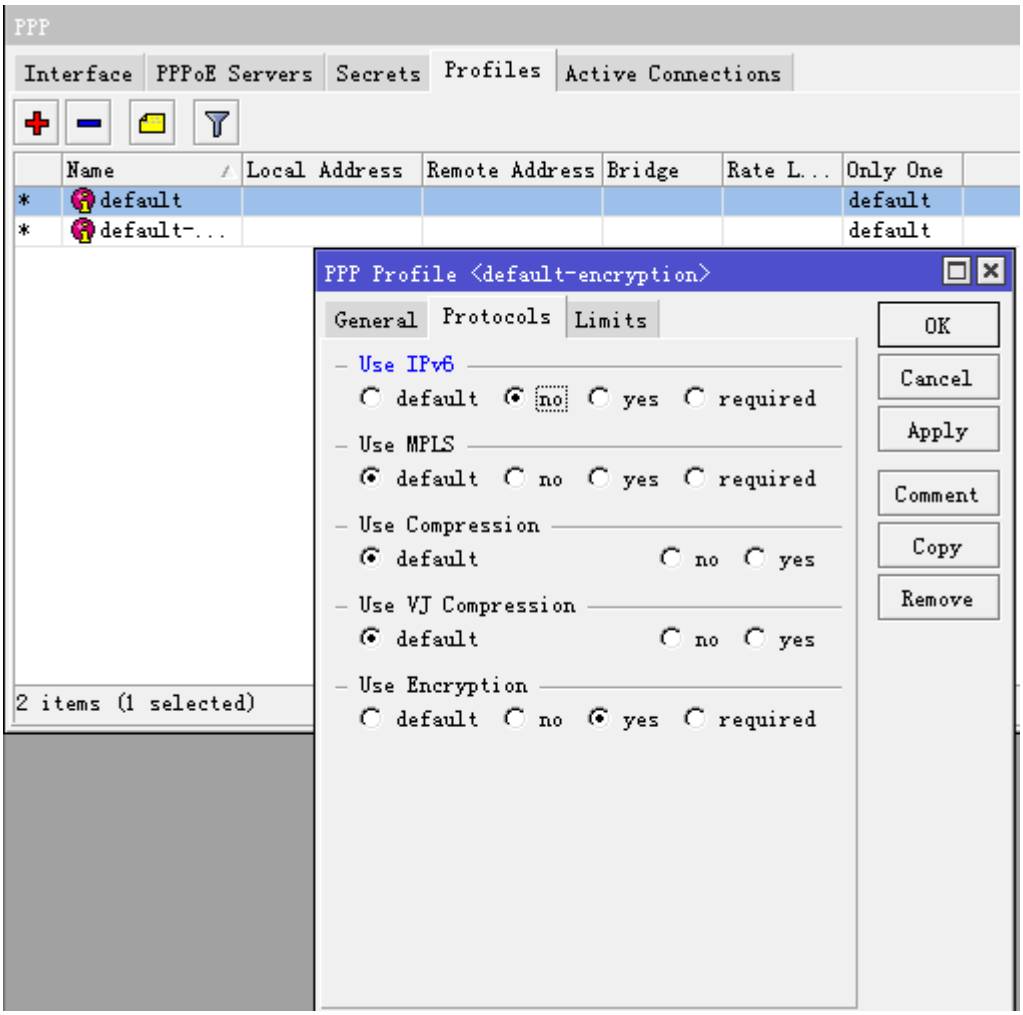
通过 Winbox 配置 PPPoE 服务器，这里我们首先通过进入 PPP 目录下的 PPPoE Server，配置 Service Name 为 MikroTik，用于 PPPoE 服务器名，并把 PPPoE 服务指向 ether2 的网卡上，其他参数如图所示：



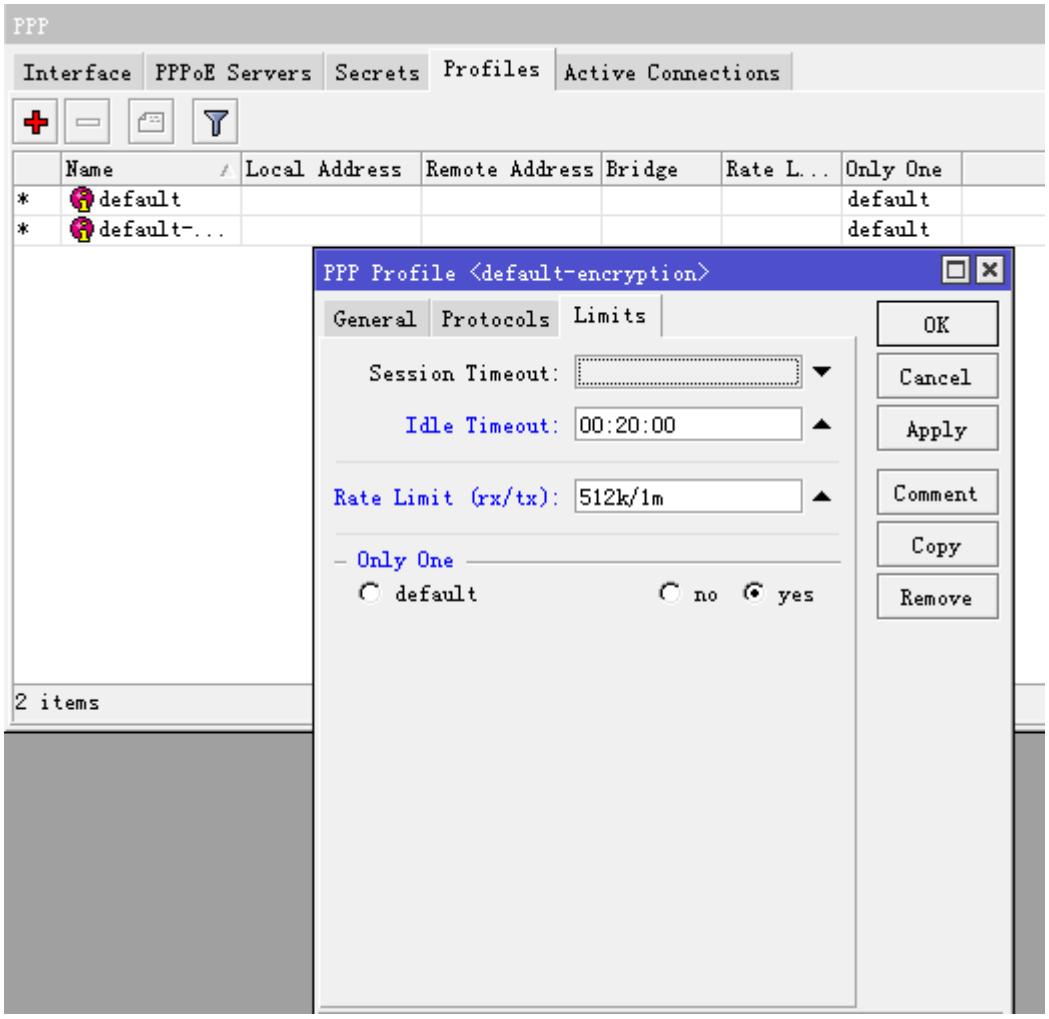
这里我们选择的是 default-encryption 的 profile 规则，所有我们需要进入 profiles 中配置该规则的参数，loacl-address 为本地路由器网关 IP，remote-address 则是远程客户端 IP 地址。这里我们设置 local-address 为 192.168.10.1，remote-address 添加在 ip pool 中设置好的地址池 pppoe，然后配置 DNS 参数，其他配置如图：



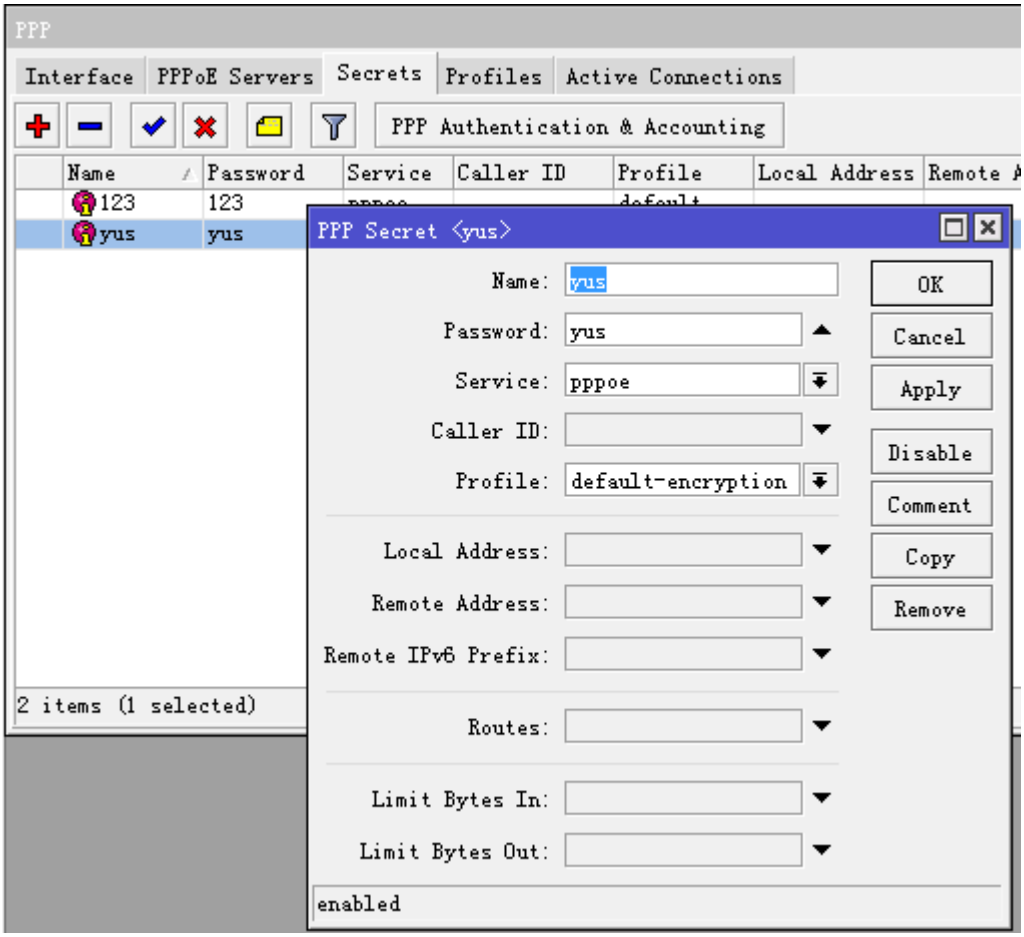
在 5.0 后增加了 IPv6 的支持，这里我们可以选择关闭 IPv6 的支持：



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 添加用户帐号：



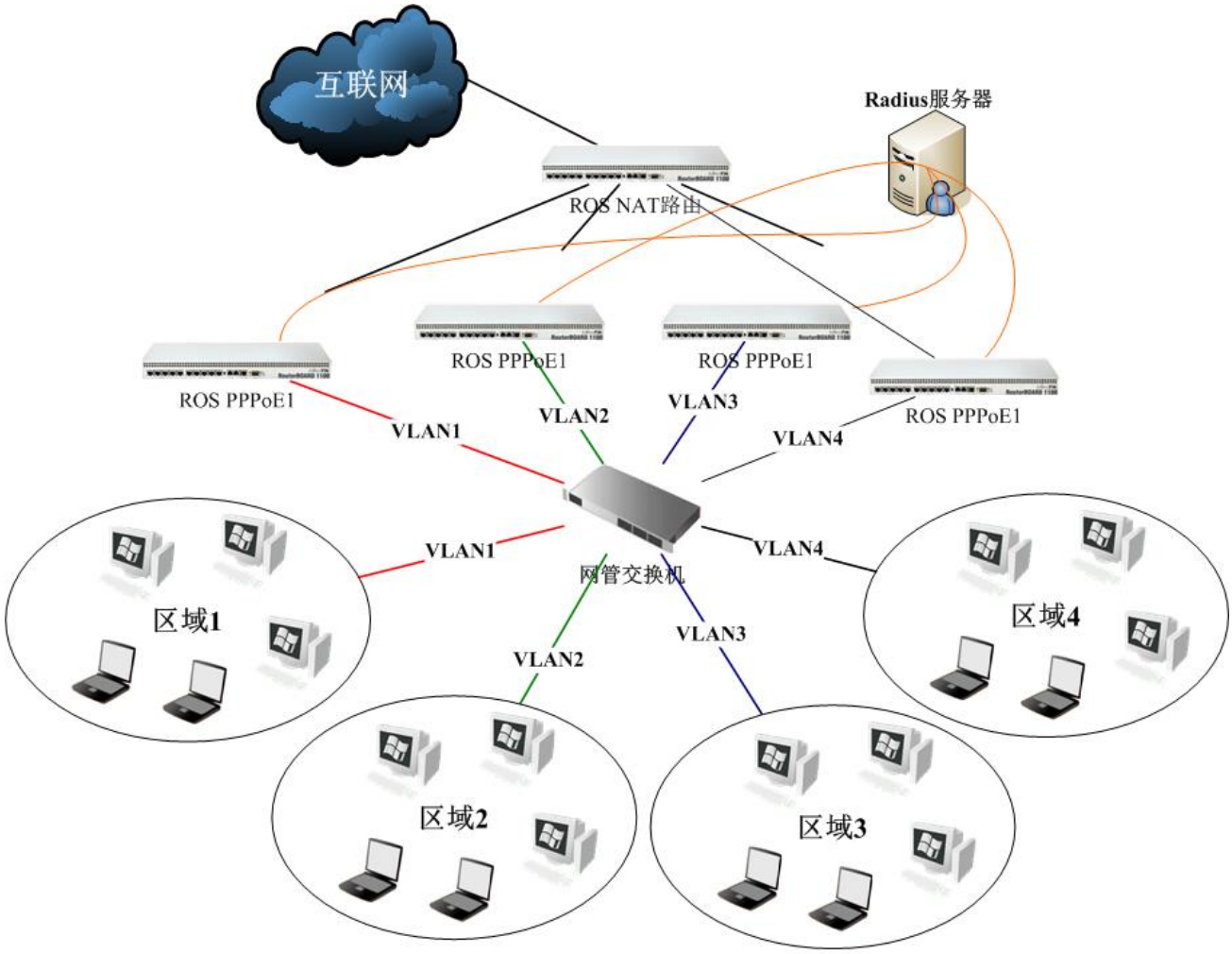
这里 Name 为用户帐号名，Password 为用户密码。Profile 选择刚才设置好的 default-encryption，根据情况也可以调用其它相应的 profile 规则。配置完用户的帐号和密码后，PPPoE 服务就可以启动。

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。

22.5 大型 PPPoE 服务应用

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 层数据的影响，特别是 ARP 协议，这样可以避免现在比较常见的 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。比起通过 IP 方式认证的 Web 页面要稳定安全的多。在一些网吧为了避免 ARP 病毒的侵扰，也在网吧内部建立的 PPPoE 认证方式，避免 ARP 对网吧带来上网电脑频繁掉线问题。

现在几乎所有的人都在使用 WindowsXP 或以上的操作系统，这些操作系统都自带了 PPPoE 拨号软件，即用户不需要太复杂的操作，就可以建立一个虚拟拨号连接。下面是一个 PPPoE 认证系统的网络结构：



- 首先采用 RouterOS 作为接入路由器和外网防火墙，这里我们采用 CCR1036 设备或者选择更强劲的 x86 服务器系统作为外网接入路由器，可以实现多线路多运营商的路由器，并作为 nat 转换设备，减轻 PPPoE 认证服务器的压力。
- 在 NAT 路由器下面，我们可以根据用户数量，建立多个 PPPoE 服务器，采用 PPPoE 集群服务器方式均衡用户，一般高性能的至强服务器，能支持 2000 个左右 PPPoE 认证用户同时在线（建议使用 RouterOS 6.0）。
- 通过核心交换 VLAN 的 Trunk 连接用户层交换机，并分配每个用户连接那个 PPPoE 服务器，这样可以通过 VLAN 划分用户区域，隔离不必要的数据，减小广播风暴。用户接入可以通过以太网的有线连接，也可以通过无线的 AP 接入网络，连接方式灵活多样化。
- 在实际使用中发现如果 2 台或者多台 RouterOS 的 PPPoE 认证服务器在同一个 VLAN 下，相同的 Service name 能实现用户拨号的自动负载均衡和冗余，只能使用在划分 802.1q 的网络，普通交换机不支持。
- 所有 PPPoE 服务器都采用同一个 RADIUS 服务器，这样帐号便于管理，特别在多 PPPoE 的集群认证下有利于冗余的工作，在一台 PPPoE 服务器停机后，其余的设备可以接替工作。配置 RADIUS 服务器也可以分担 RouterOS 在账号管理的负荷。

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 连接小点的数据包（例如 pings)**

  你需要改变所以经过 PPPoE 连接的 **mss** 数据包为 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>
```

- **我的 windows PPPoE 客户端得到了来至 MikroTik PPPoE server 的 IP 地址和默认网关，但不能出 PPPoE server 并且不能连接外部网络.**

  PPPoE 服务器没有与客户端连接， 为 PPPoE 客户端的地址配置伪装（masquerading）或是确定你为客户端分配的地址段指定了正确的路由,或是你在以太网卡上启用了 Proxy-ARP (请看 IP 地址和地址解析协议 Address Resolution Protocol (ARP))

- **我的 Windows XP 不能连接到 PPPoE 服务器**

  你要在 XP 的 pppoe 客户端属性中指明"Service Name"。或是没有在 MikroTik PPPoE 服务器配置服务名（service name），这样你会得到“line is busy”错误或是系统显示"verifying password - unknown error"

- **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

第二十三章 PPTP

PPTP（点对点隧道协议）支持 IP 上的加密隧道。MikroTik RouterOS 工具包含对 PPTP 客户和服务器的支持。PPTP 隧道的基本应用：

- 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）

每个 PPTP 连接都包含一个服务器和客户。MikroTik RouterOS 可能作为一个服务器或者客户工作——或者，对多种配置，它可以对某些连接是服务器而对其他连接是客户。例如，下面创建的客户可以连接到 Windows 2000 服务器，另一个 MikroTik Router，或另一个支持 PPTP 服务器的路由器。

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
```

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)

点对点隧道协议（PPTP）是一种支持多协议虚拟专用网络的网络技术。通过该协议，远程用户能够通过 windows 客户端或者路由器，以及其它装有点对点协议的系统安全访问公司网络，并能拨号连入本地 ISP，通过 Internet 安全连接到公司网络。

PPTP 包含了 PPP 认证及对每个 PPTP 连接的帐户管理。全部的认证和每个连接的帐户管理可以通过 RADIUS 客户或本地完成。支持 MPPE 40bit RC4 以及 MPPE 128bit RC4 加密。

PPTP 的连接采用的是 TCP 端口 1723 和 IP 协议 GRE（类属路由封装，IP 协议 ID 47）。PPTP 可以通过启用定为 TCP 端口 1723 和 47，注意让相应的路由器不会对这两个端口做防火墙过滤等操作，否则 PPTP 连接会失效。

23.1 PPTP 客户设置

操作路径: ***/interface pptp-client***

属性描述

add-default-route (yes | no; default: **no**) - 是否添加默认路由（网关）
allow (*多选项:* mschap2, mschap1, chap, pap; default: **mschap2, mschap1, chap, pap**) -允许客户启用验证的协议
connect-to (*IP address*) - 连接到 PPTP 服务器的 IP 地址
mru (*整型*; default: **1460**) - 最大接收单元。最优值是隧道工作的接口 MRU 减少 40（所以，1500 字节以太网连接设置 MRU 为 1460 以避免包的分割）
mtu (*整型*; default: **1460**) -最大传输单元。最优值是隧道工作的接口 MTU 减少 40（所以，1500 字节以太网连接设置 MTU 为 1460 以避免包的分割）
name (*名称*; default: **pptp-outN**) - pptp 客户端的名称
password (*文本*; default: **""**) 连接 PPTP 服务器的密码
profile (*名称*; default: **default**) -当连接到远程服务器时使用的概要简介
user (*文本*) 连接 PPTP 服务器的账号

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
```

属性描述

encoding (*文本*) -加密及编码（如果非对称，使用‘/’分隔）在该连接中使用
status (*文本*) - status of the client
Dialing – 试图进行连接
Verifying password... - 连接已建立到服务器，正在核实密码
Connected - 已连接状态
Terminated -没有启用借口或另一端不能建立连接
uptime (*time*) - 以天，小时，分钟以及秒钟显示的连接时间
命令名: ***/interface pptp-client monitor***

一个已建立连接的实例：

```
[admin@MikroTik] interface pptp-client> monitor test2
     uptime: 4h35s
   encoding: MPPE 128 bit, stateless
     status: Connected
[admin@MikroTik] interface pptp-client>
```

23.2 PPTP 服务器设置

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。

为了创建 PPTP 用户，你应该咨询 PPP secret 以及 PPP Profile 手册。也可以使用 MikroTik 路由器作为 RADIUS 客户来注册 PPTP 用户。

属性描述

authentication (*多选项:* pap | chap | mschap1 | mschap2; default: **mschap2**) - 认证算法
default-profile -默认概要信息
enabled (yes | no; default: **no**) - 定义 PPTP 服务器是否启用
keepalive-timeout (*time*; default: **30**) -定义路由器开始每秒发送持活时间数据包之后的时间段（以秒计算）。如果没有流量并且没有保持活动，在那段时间将出现反应（例如，2 * keepalive-timeout），没有反应的客户将被宣布为断开连接。
mru (*整型*; default: **1460**) - 最大接收单元。最优值是隧道工作的接口 MRU 减少 40（所以，1500 字节以太网连接设置 MRU 为 1460 以避免包的封装问题）
mtu (*整型*; default: **1460**) -最大传输单元。最优值是隧道工作的接口 MTU 减少 40（所以，1500 字节以太网连接设置 MTU 为 1460 以避免包的封装问题）

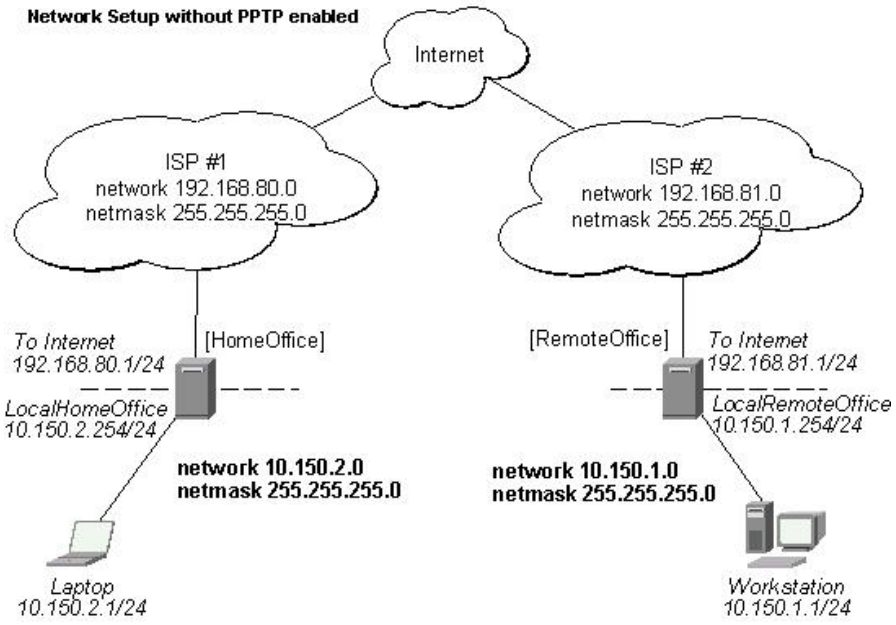
启用 PPTP 服务器：

```
[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>
```

23.3 PPTP 应用实例

Router to Router 安全隧道实例

以下是一个使用互联网上的加密 PPTP 隧道连接两个企业网局域网的例子：



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：

- **[HomeOffice]**

接口 LocalHomeOffice 10.150.2.254/24
接口 ToInternet 192.168.80.1/24

- **[RemoteOffice]**

接口 ToInternet 192.168.81.1/24
接口 LocalRemoteOffice 10.150.1.254/24

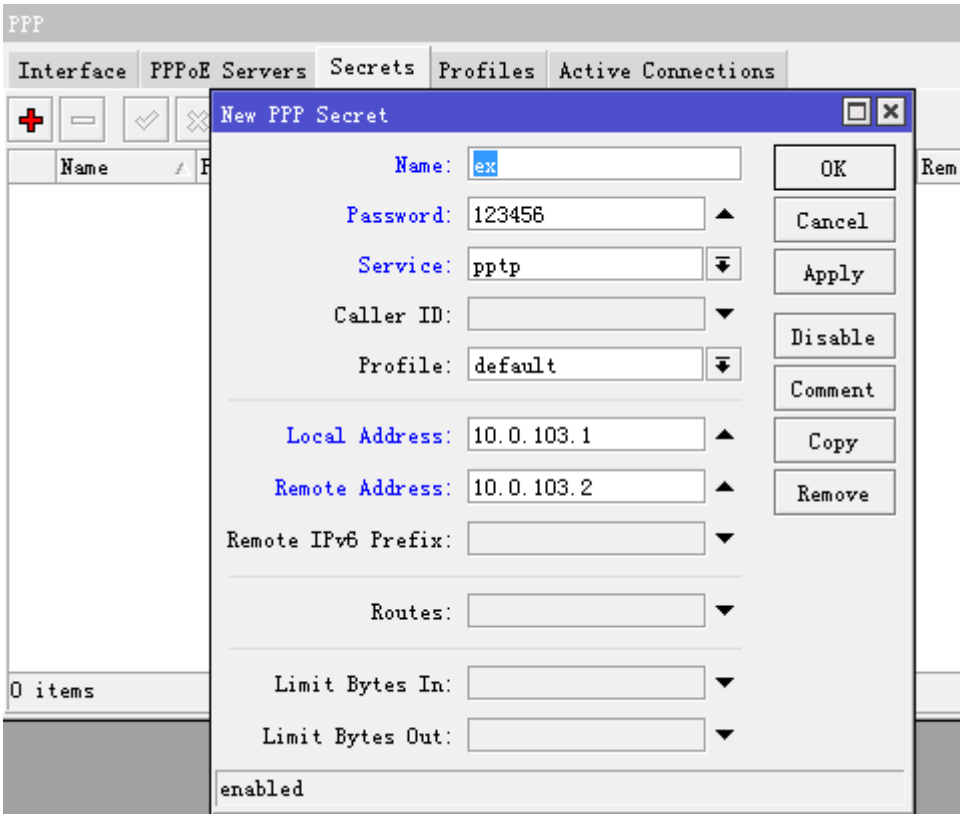
每个路由器连接到当地的 ISP，任何一个路由器可以通过互联网访问到对端的路由器。

HomeOffice 配置

在 HomeOffice 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：

```
[admin@HomeOffice] ppp secret> add name=ex service=pptp password=123456
local-address=10.0.103.1 remote-address=10.0.103.2
[admin@HomeOffice] 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@HomeOffice] ppp secret>
```

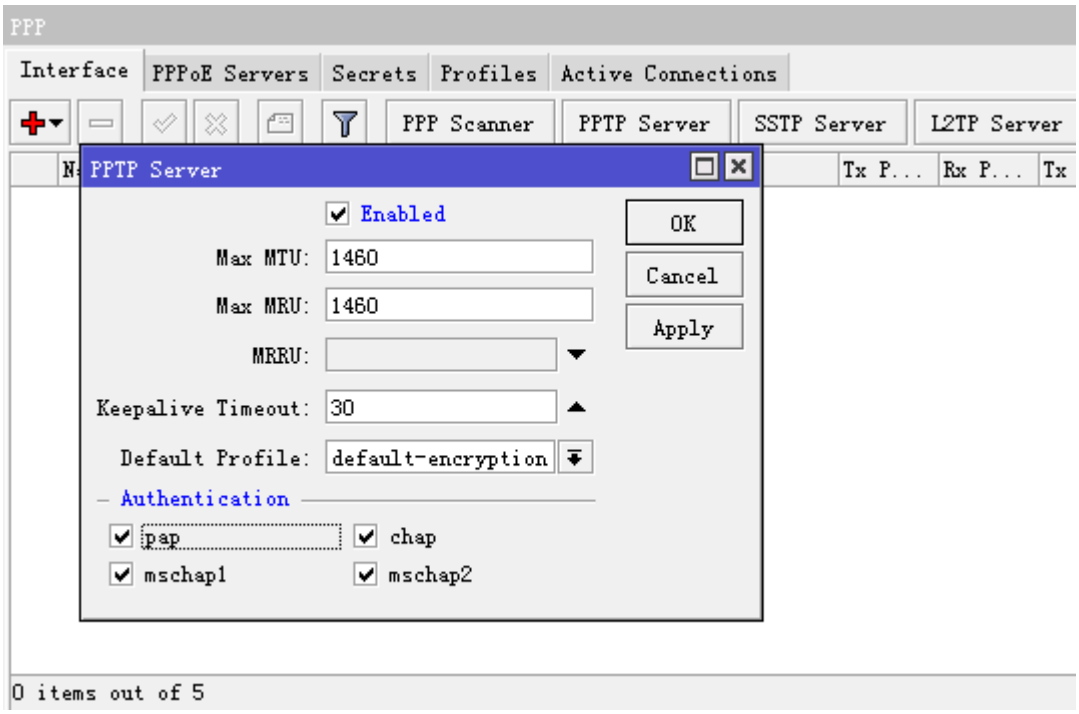
Winbox 操作如下：



在 interface pptp-server server 目录下，启用 pptp 服务器：

```
[admin@HomeOffice] interface pptp-server server> set enabled=yes
[admin@HomeOffice] 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：

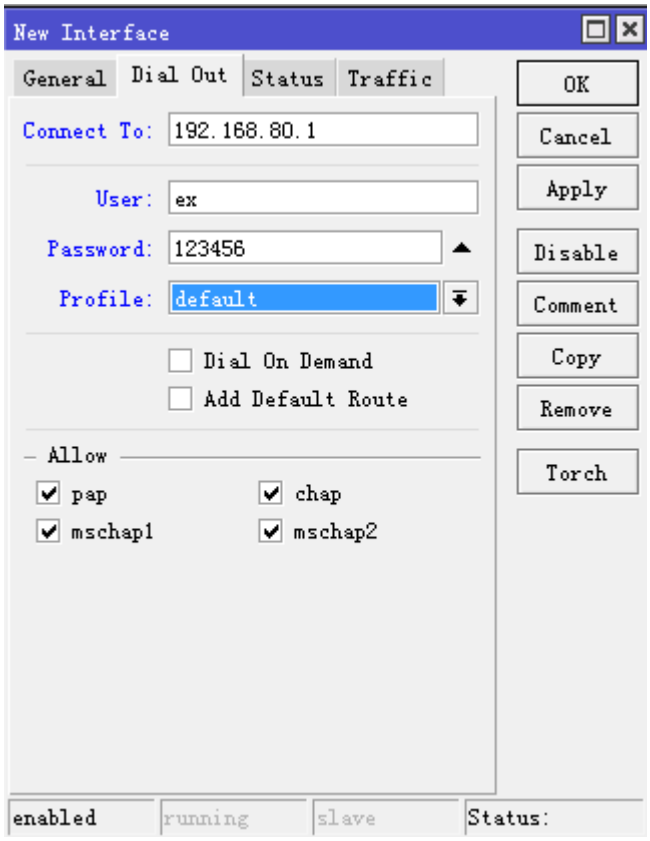


RemoteOffice 配置

在 RemoteOffice 路由器添加一个 PPTP 客户：

```
[admin@RemoteOffice] 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@RemoteOffice] interface pptp-client>
```

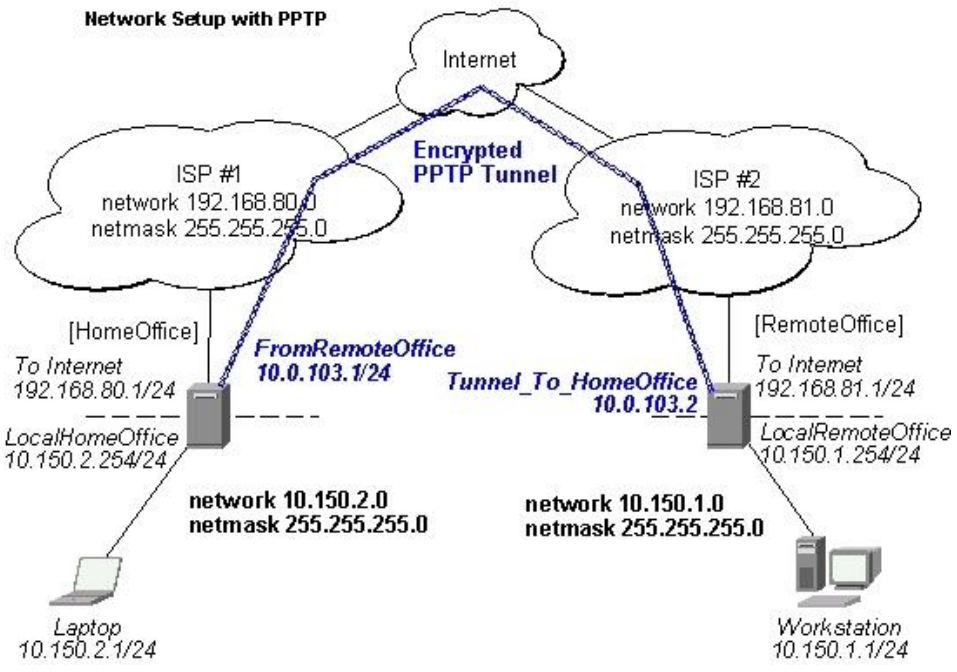
Winbox 在 interface 中添加 pptp-client



这样，一个 PPTP 隧道就在路由器之间创建好了。这个隧道就像在 IP 地址为 10.0.103.1 及 10.0.103.2 的路由器之间的三层点对点连接。

pptp 局域网的互访

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



为了在 PPTP 隧道上互访企业间本地网络，需要添加以下路由：

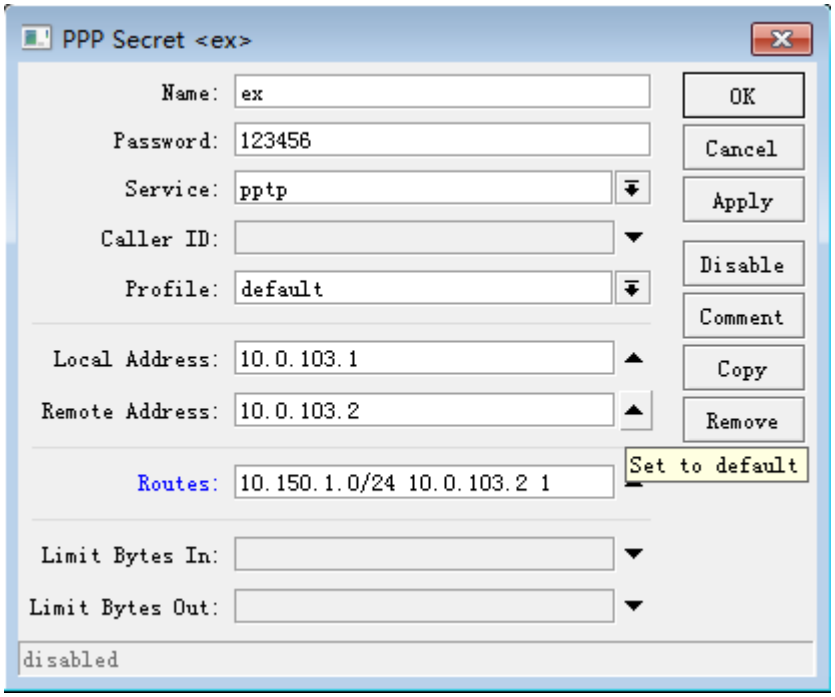
```
[admin@HomeOffice] > ip route add dst-address=10.150.1.0/24 gateway=10.0.103.2
[admin@RemoteOffice] > ip route add dst-address=10.150.2.0/24 gateway=10.0.103.1
```

或者也可以在 PPTP 服务器（HomeOffice）上通过用户配置的 routes 参数完成，RemoteOffice 还是需要在/ip route 中配置路由：

```
[admin@HomeOffice] 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@HomeOffice] ppp secret> set 0 routes="10.150.1.0/24 10.0.103.2 1"
[admin@HomeOffice] 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="10.150.1.0/24 10.0.103.2 1"

[admin@HomeOffice] ppp secret>
```

目的路由：10.150.1.0/24
pptp 的网关：10.0.103.2
Distance 路径： 1

Winbox 中修改 routes 参数



测试 PPTP 隧道连接：

```
[admin@RemoteOffice]> /ping 10.0.103.1
10.0.103.1 pong: ttl=255 time=3 ms
```

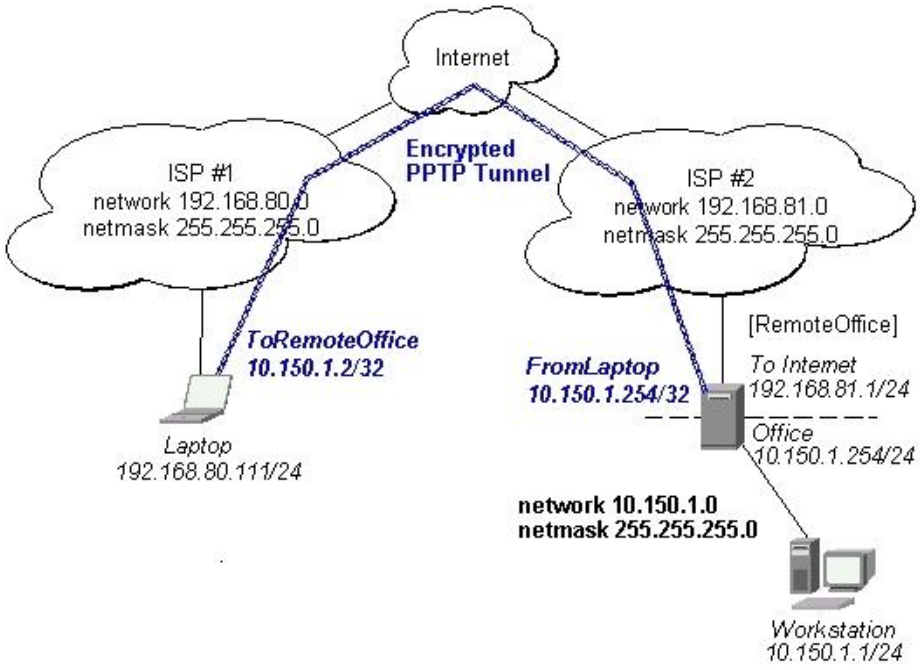
```
10.0.103.1 pong: ttl=255 time=3 ms
10.0.103.1 pong: ttl=255 time=3 ms
ping interrupted
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 3/3.0/3 ms
```

测试通过 PPTP 隧道到 LocalHomeOffice 接口的连接：

```
[admin@RemoteOffice]> /ping 10.150.2.254
10.150.2.254 pong: ttl=255 time=3 ms
10.150.2.254 pong: ttl=255 time=3 ms
10.150.2.254 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电脑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 ToInternet　192.168.81.1/24
  接口 Office　10.150.1.254/24

在 PPTP 服务器上设置用户帐号：

```
[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：

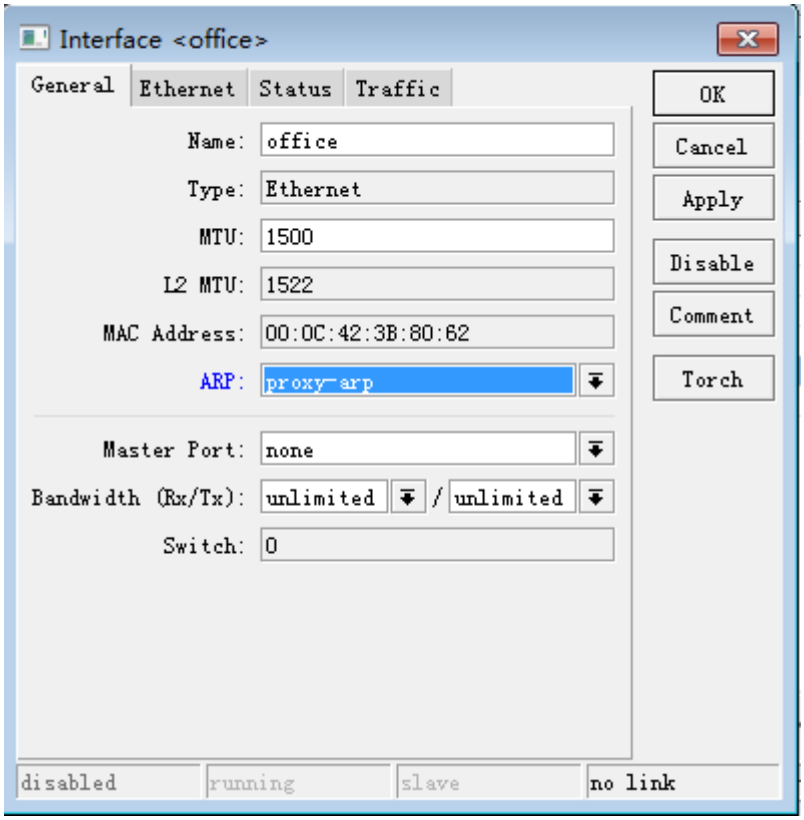
```
[admin@RemoteOffice] interface pptp-server server> set enabled=yes
[admin@RemoteOffice] interface pptp-server server> print
              enabled: yes
                  mtu: 1460
                  mru: 1460
      authentication: mschap2
     default-profile: default
[admin@RemoteOffice] interface 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>
```

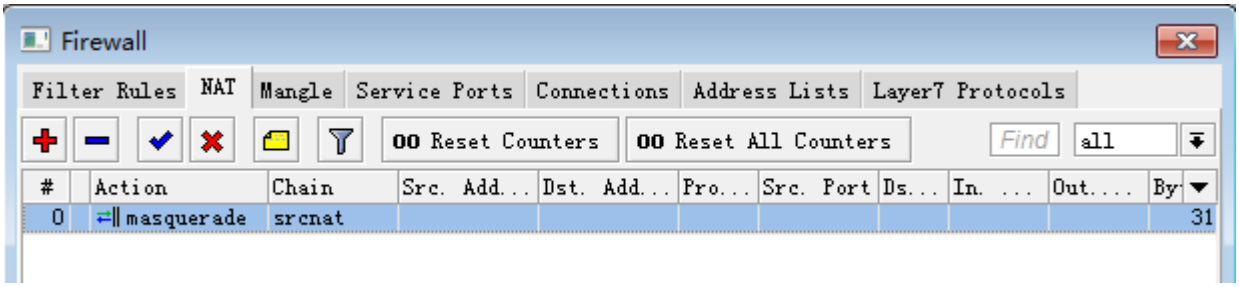
在 winbox 中进入 interface 目录下，选择 office 接口设置 arp 为 proxy-arp



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 都制作了帮助页面以帮助客户进行 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站点。而且，TCP 协议 47 应该通过。

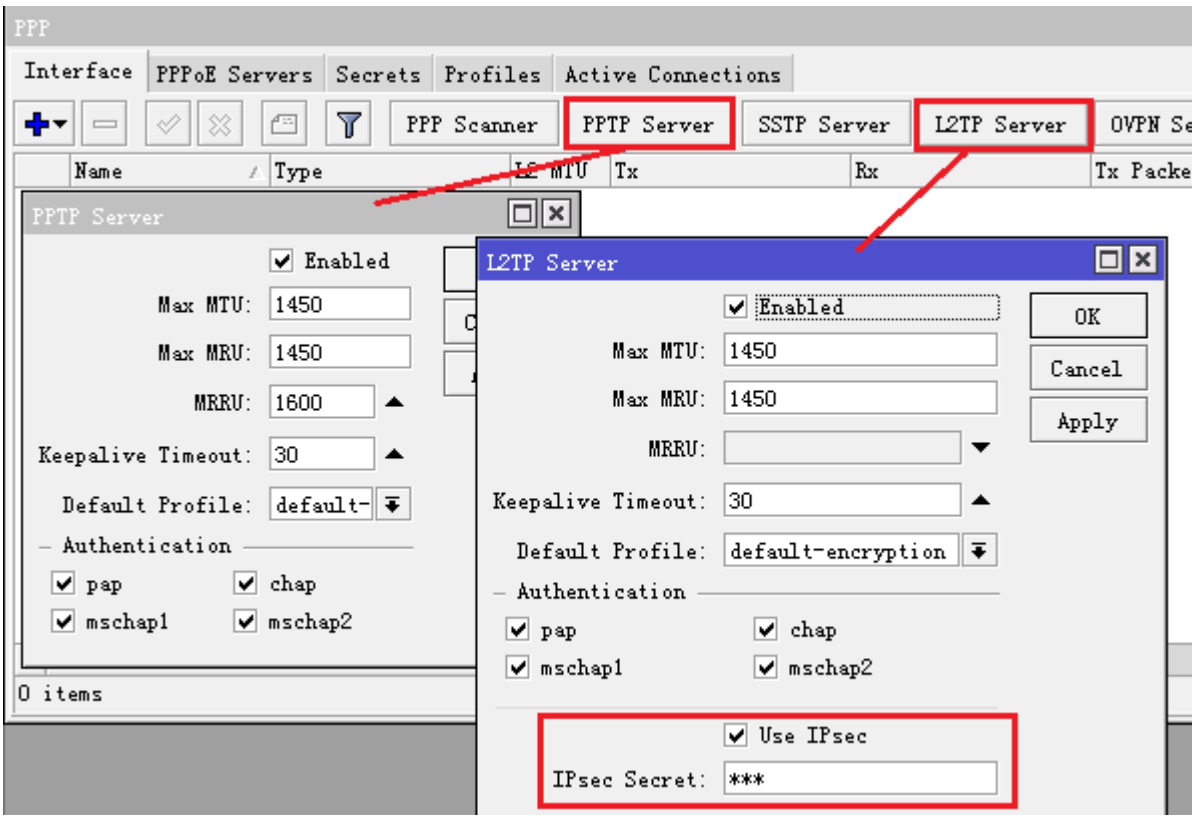
第二十四章 PPTP 与 L2TP 服务

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包的点对点的连接。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

24.1 同时建立 PPTP 和 L2TP 服务器

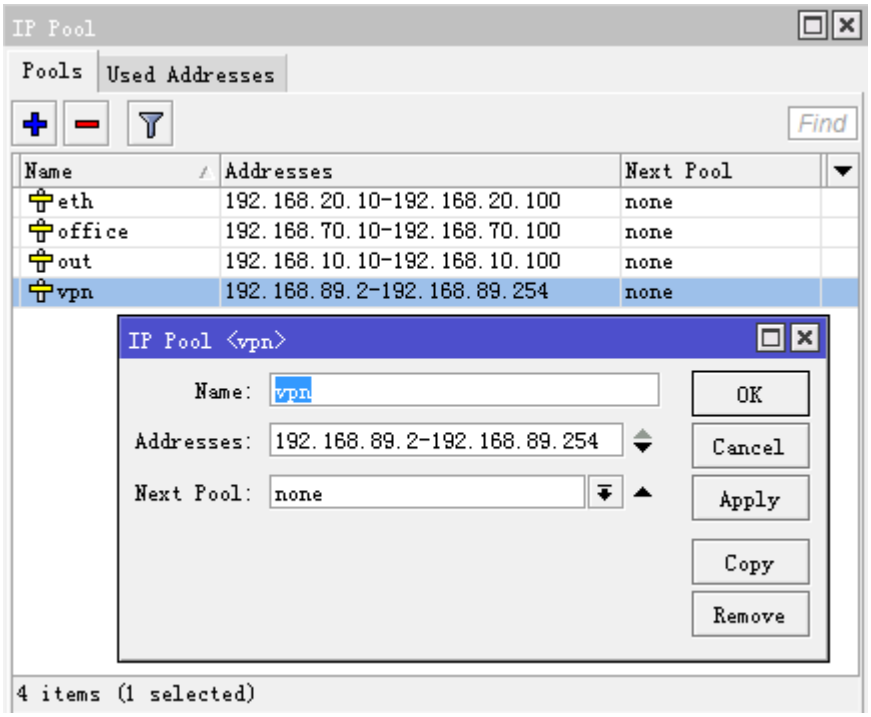
首先我看一下 PPTP 和 L2TP 的建立，同样是在 PPP 的目录下，interface 菜单中，选择 PPTP-Server 和 L2TP-Server 并启用服务：



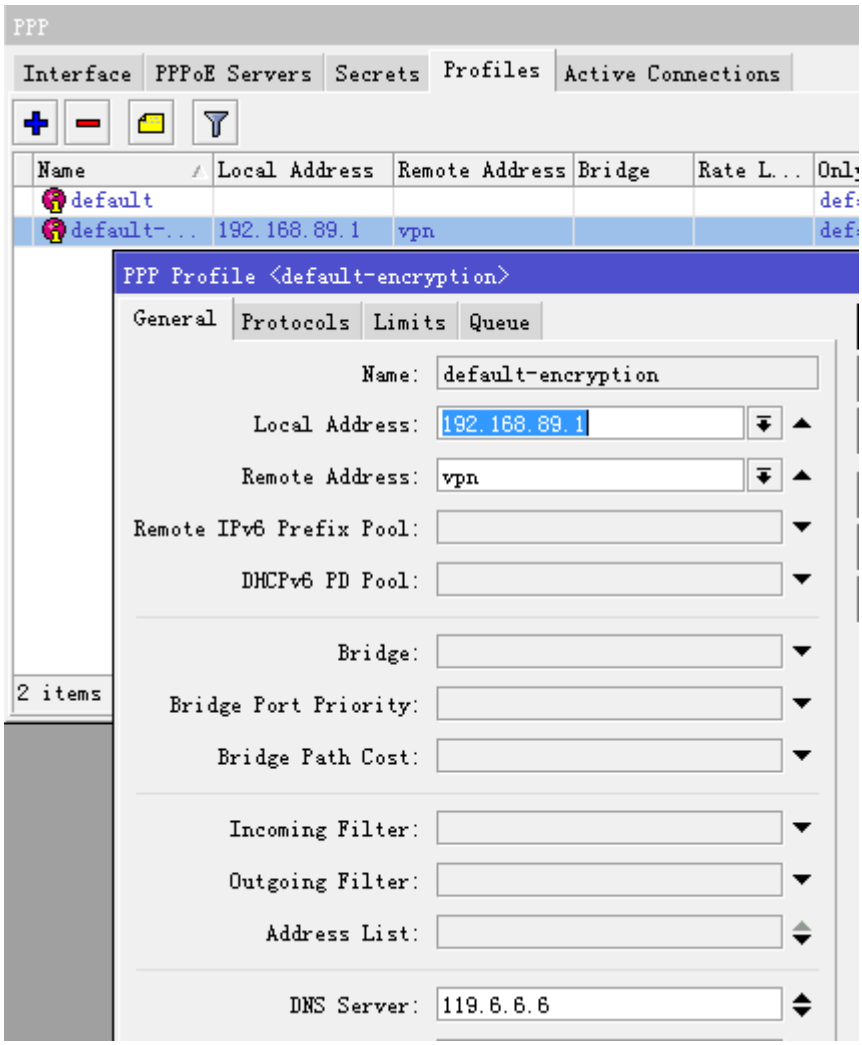
在 L2TP 服务器配置下多了 Use IPsec 选项，这是在 RouterOS 6.16 后加入的 IPsec 选项，方便 windows 客户端的连接，Default-Profile 类型选择 default-encryption，Autentication 认证方式也可以选择相同方式。

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。

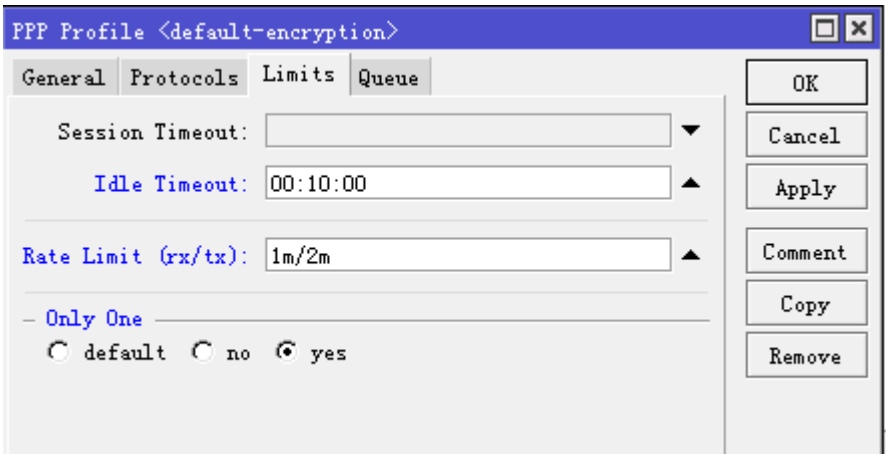
首先我进入 ip pool 中配置地址池：



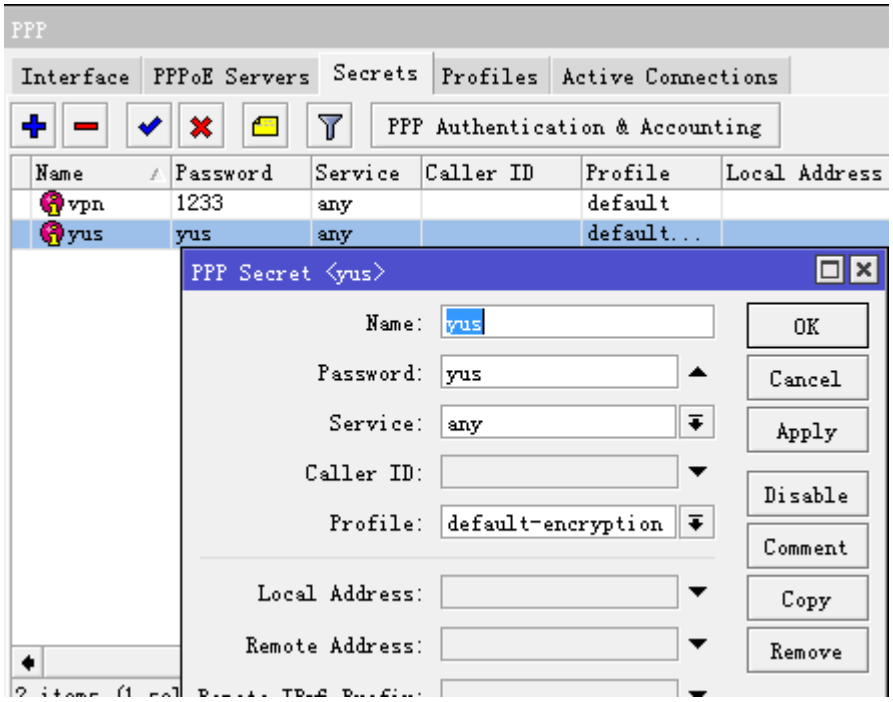
配置好地址池后，在 PPP Profiles 中使用默认的 default-encryption 组规则，配合本地 IP 地址 192.168.89.1，在远程 remote-address 种配置之前添加号的地址池 VPN，设置 DNS 为 119.6.6.6,其他配置参数如下：



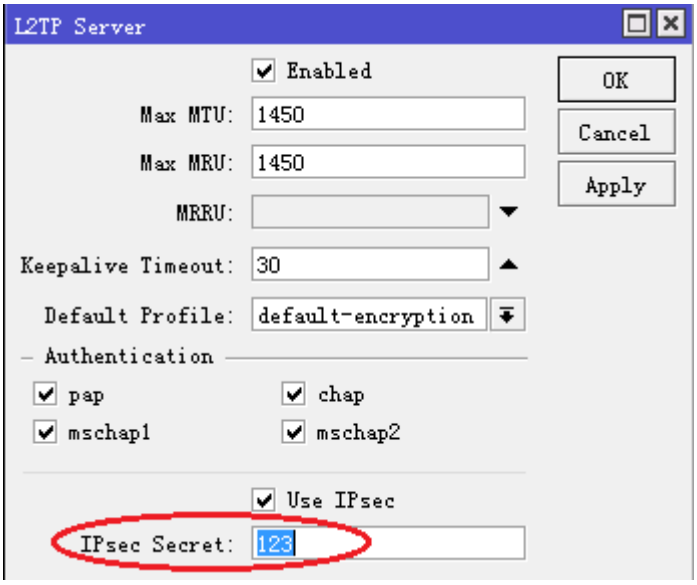
在 limits 选项中，配置相应的 Idle-timeout（空闲超时时间）为 10 分钟、Rate-limit（带宽）为 1M 上行和 2M 下行，配置 Only-one（帐号是否唯一性）：



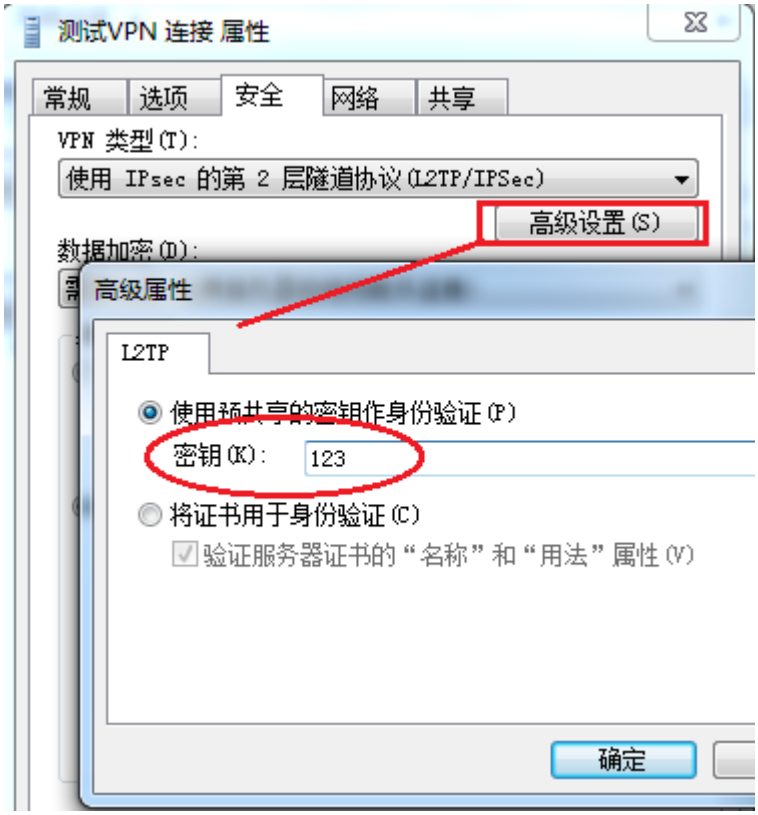
在 PPP secret 中配置用户，账号 yus，密码 yus，service 参数用于选择 VPN 类型，这里设置为默认 any 这样支持 L2TP 和 PPTP 登陆方式，profile 为 default-encryption，



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

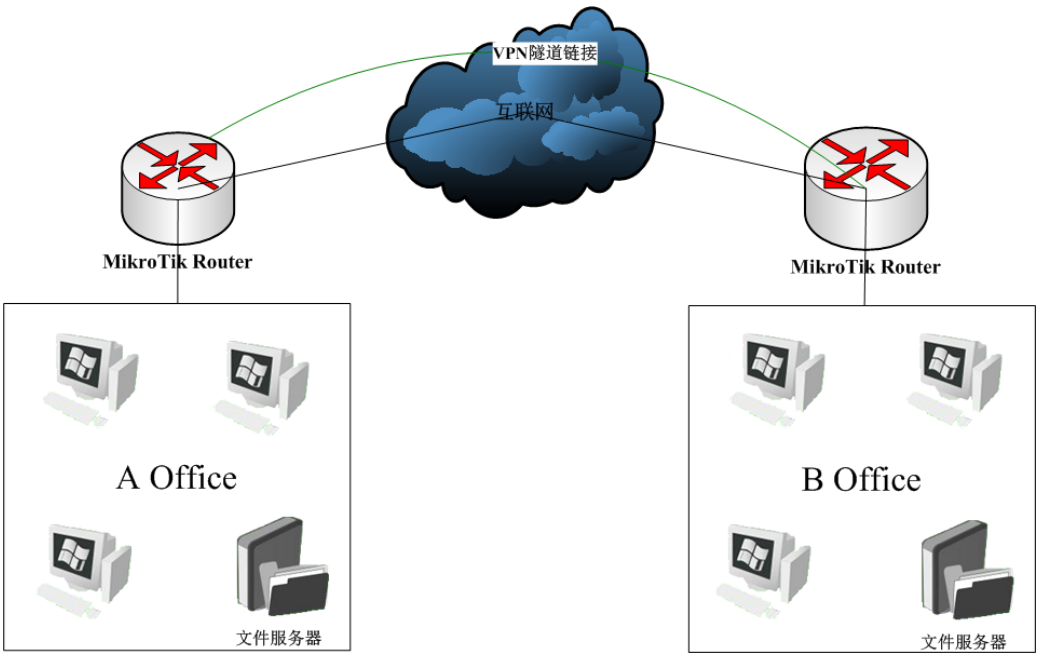
修改后即可通过 windows 正常连接到 L2TP 服务。在 RouterOS 6.16 后加入了 L2TP 服务中配置 IPsec 的选项，简化了 IPsec 配置操作，所以现在修改注册表会用到的人很少。

注：PPTP 和 L2TP 的连接端口都是固定的无法自定义，但 OVPN 和 SSTP 隧道 VPN 的连接端口可以自己定义，这有助于受到网络环境限制，采取灵活处理方法。

24.2 VPN 的几种应用方式

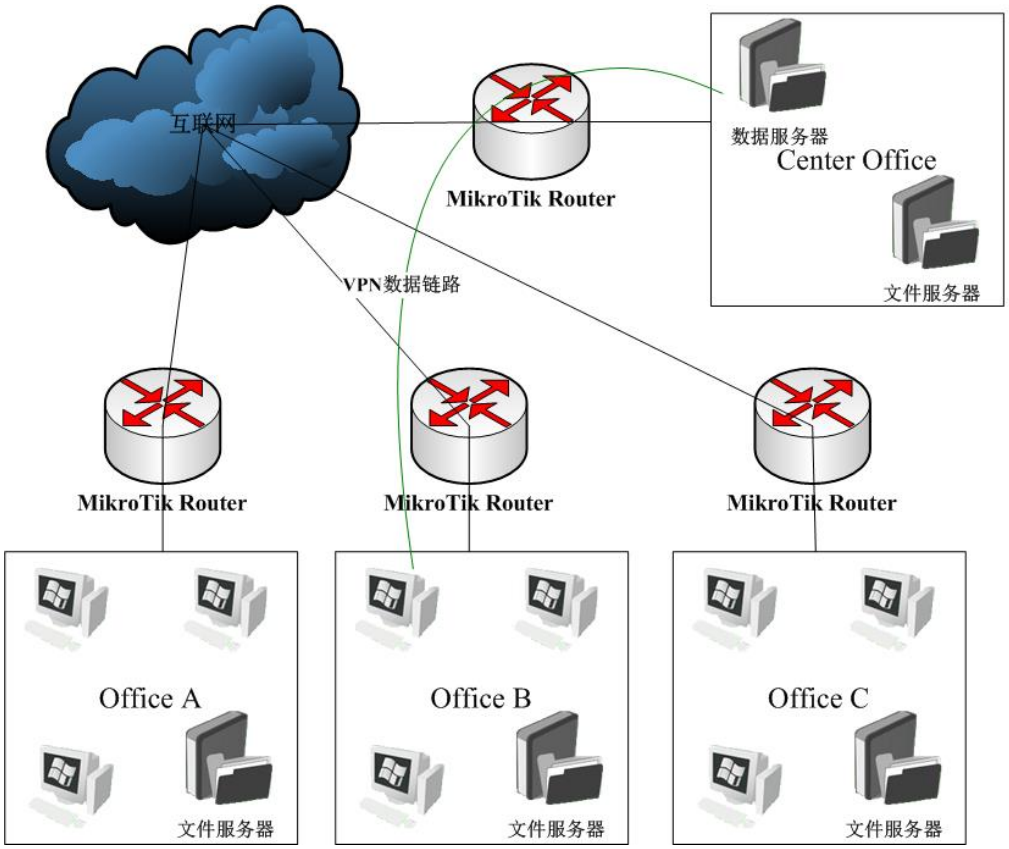
RouterOS 的 VPN 系统支持多种方式的应用，能实现企业对等访问、企业与多分支点、多点与移动办公和 VPN 数据转移等多种 VPN 连接方式。在 VPN 数据转移方面用处最多的方式是在 VoIP 方面，因为受某些 ISP 网络的限制，使得正常的 VoIP 通讯收到影响，所以可以通过 VPN 的方式实现数据的转移。

企业间对等互访



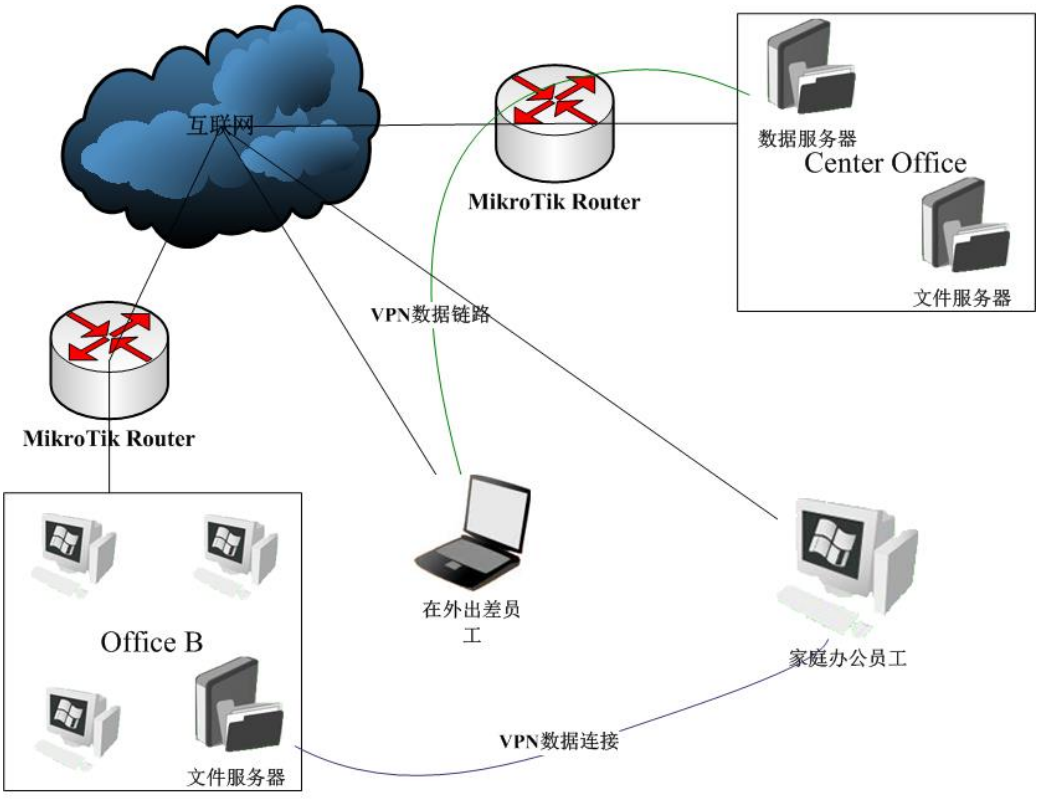
通过 VPN 隧道协议如 PPTP、L2TP 或者 IPIP 可以建立一个对等的隧道，使得两个办公室能互相访问公司数据和文件。这种方式我们首先建立 PPP 服务器，并给客户端分配帐号和固定 IP 地址、建立 PPP 的拨号和分配 IP 地址，最后设置两个远程局域网的 IP 地址路由（操作可以参考 PPTP 章节）

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，可以即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，特别对需要即时处理问题的公司员工最为适合。

24.3 BCP 协议

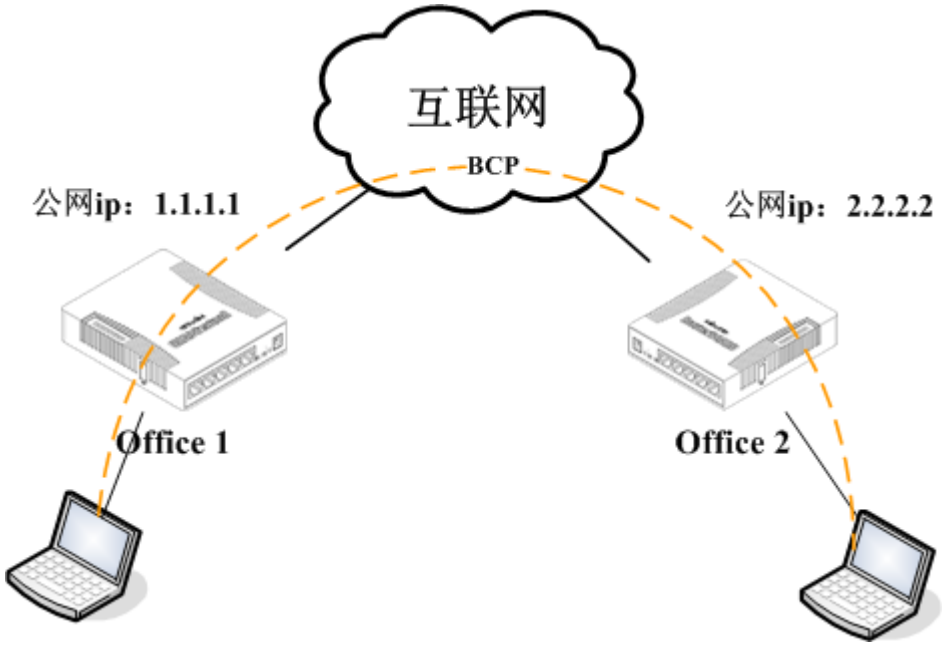
RouterOS 支持 BCP(Bridge Control Protocol)，即在 PPP、PPTP、L2TP 和 PPPoE 接口上的桥接（OVPN 和 SSTP 不支持）。BCP 协议通过 PPP 协议，将两个远端的以太网数据链路打通。BCP 建立后独立于 PPP 隧道，将不会与任何 PPP 的 IP 地址接口有关系。BCP 能用于替代 EoIP+VPN 隧道，EoIP 要求对等的网络连接，而 BCP 为网络提供另一种方式的解决，特别是一端在 nat 网络内，经过测试能正常透过 nat 网络透传二层数据。

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，奋力数据包到多个通道，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，回应在服务器端的配置

```
/ppp profile add name=ppp_bridging bridge=bridge_local use-encryption=yes
```

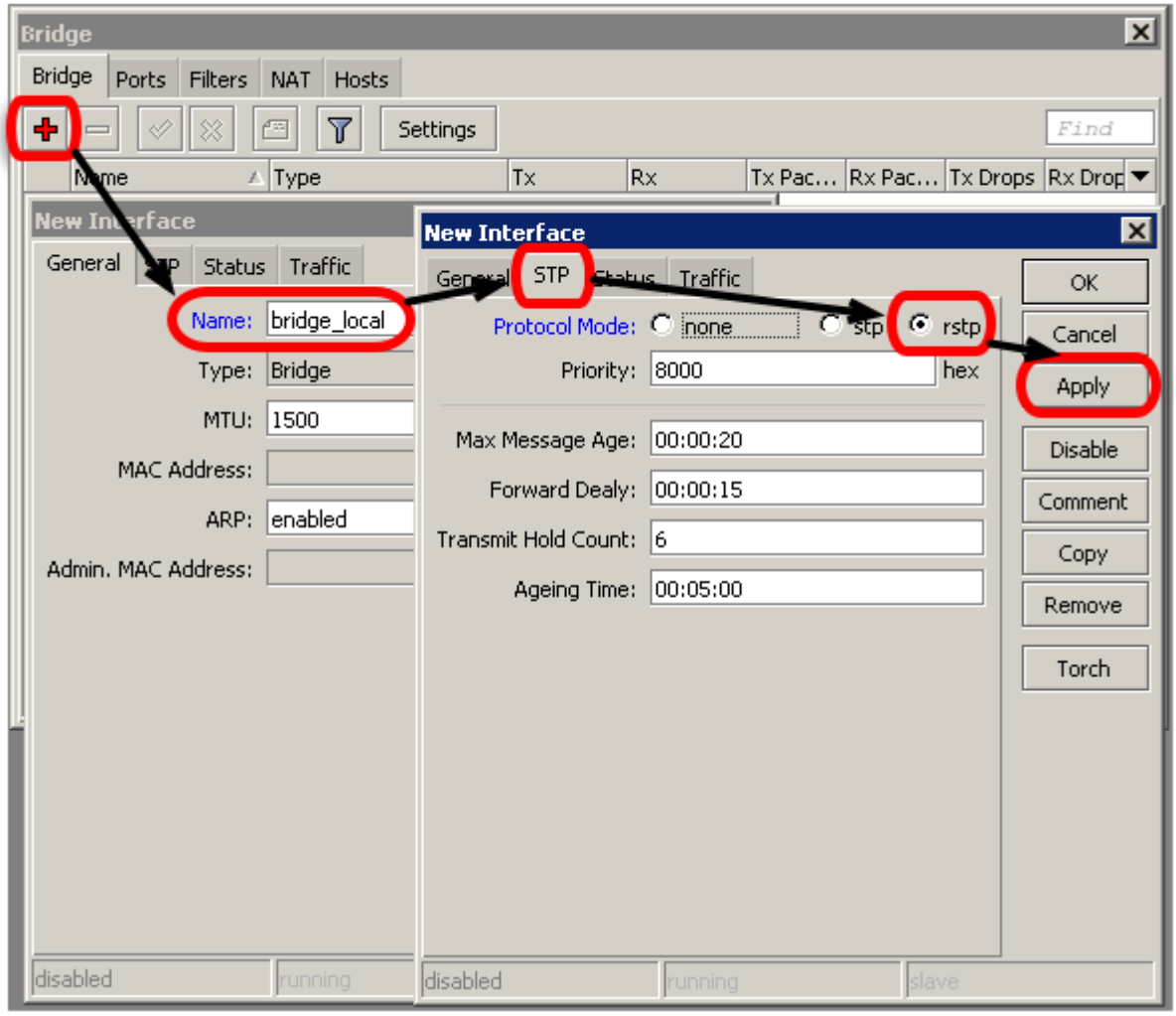
创建一个 pptp-client 接口，不要忘记配置 MRRU 选项，确保二层帧能通过 PPP 隧道。

```
/interface pptp-client add profile=ppp_bridging mrru=1600 connect-to=1.1.1.1 user=ppp1 password=ppp1
disabled=no
```

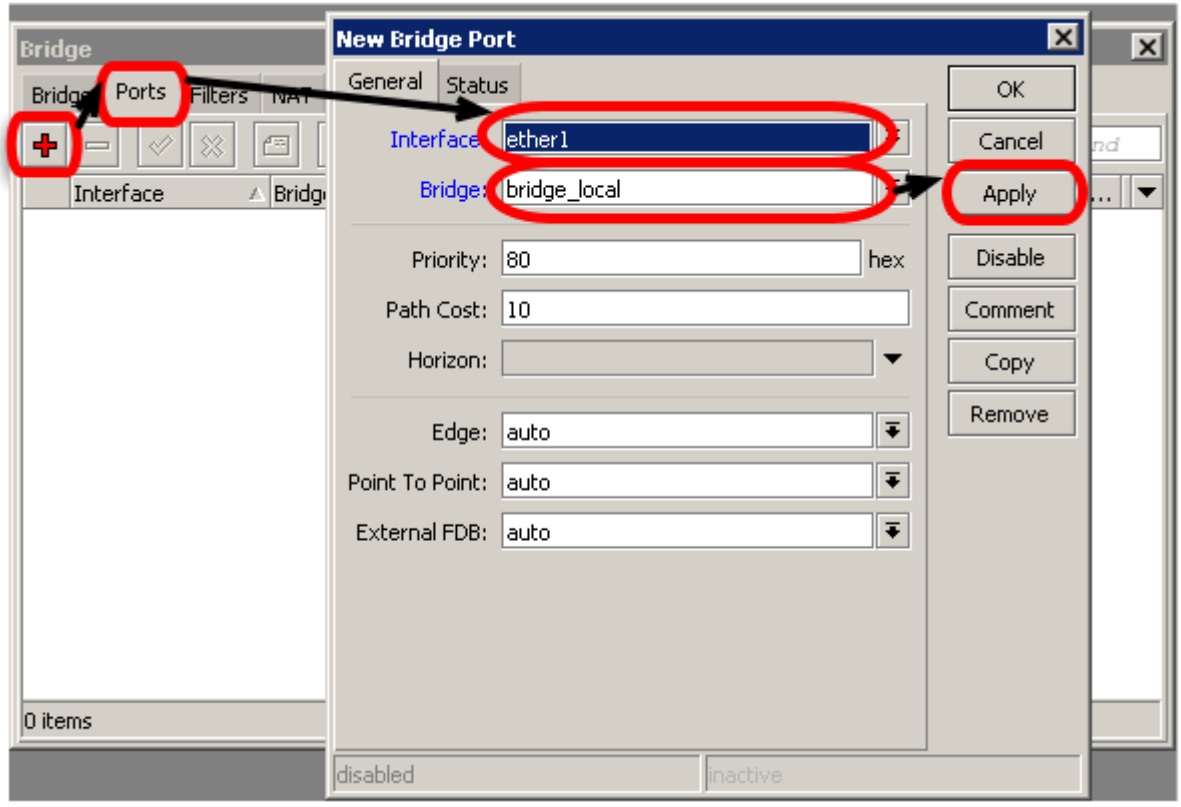
BCP winbox 配置

Office1 配置

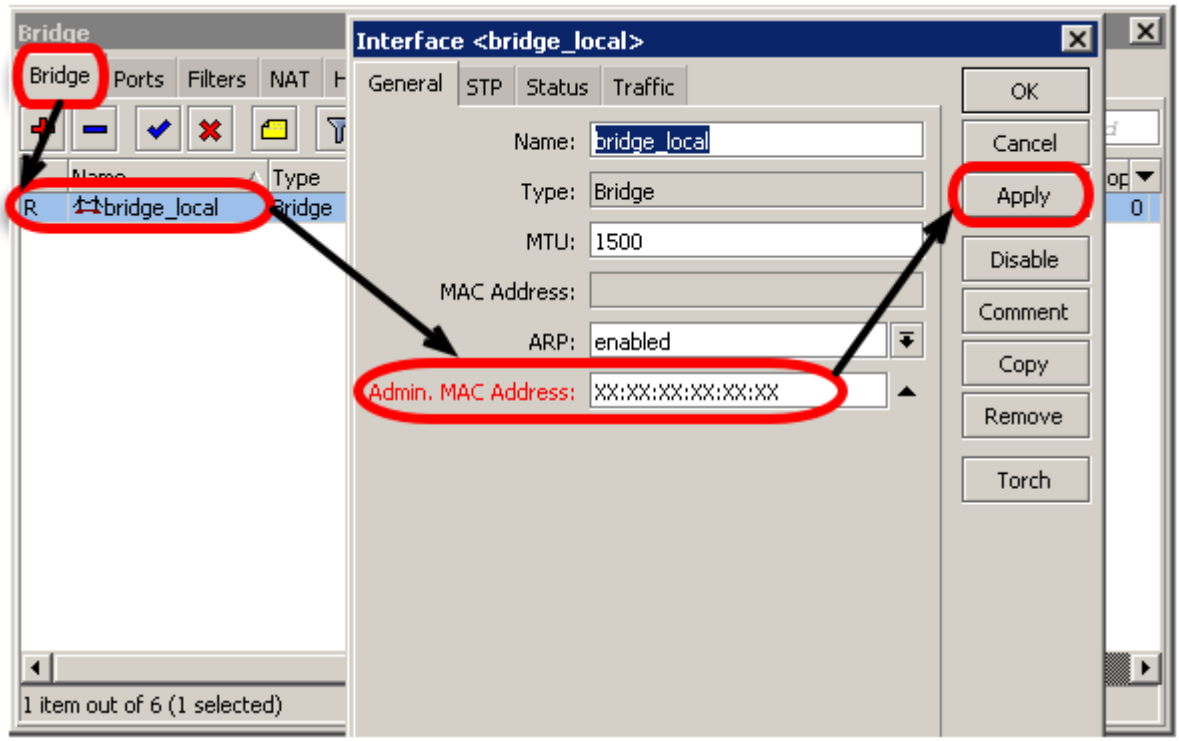
在 bridge 中添加一个桥，并设置 rstp：



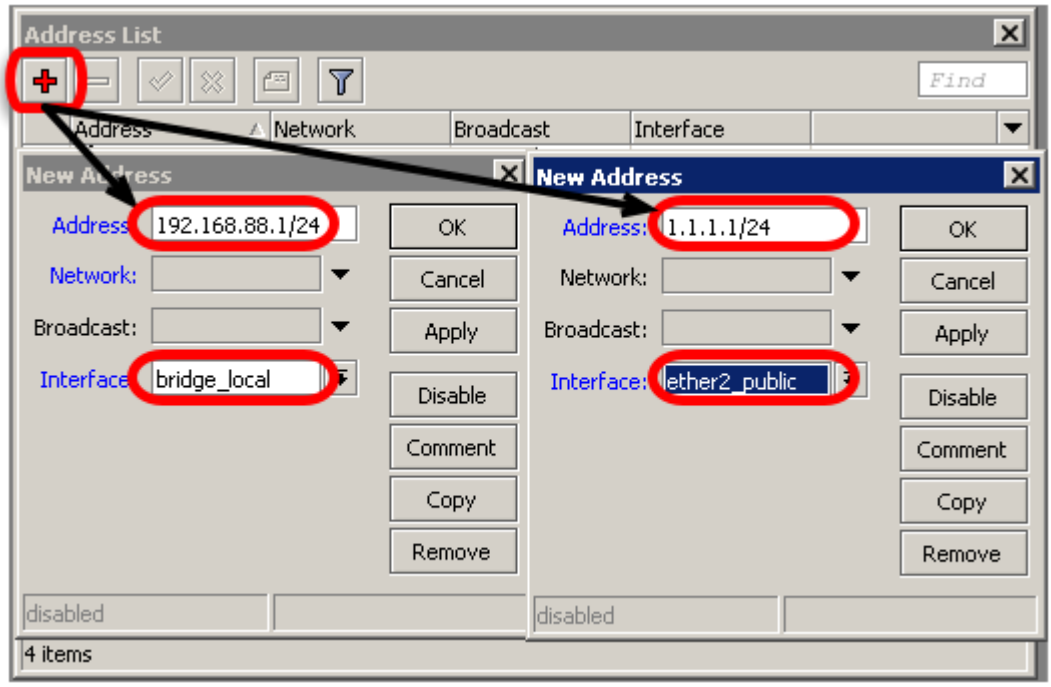
在 port 中，添加桥接的接口，我们添加 ether1 到 port 用于连接内网：



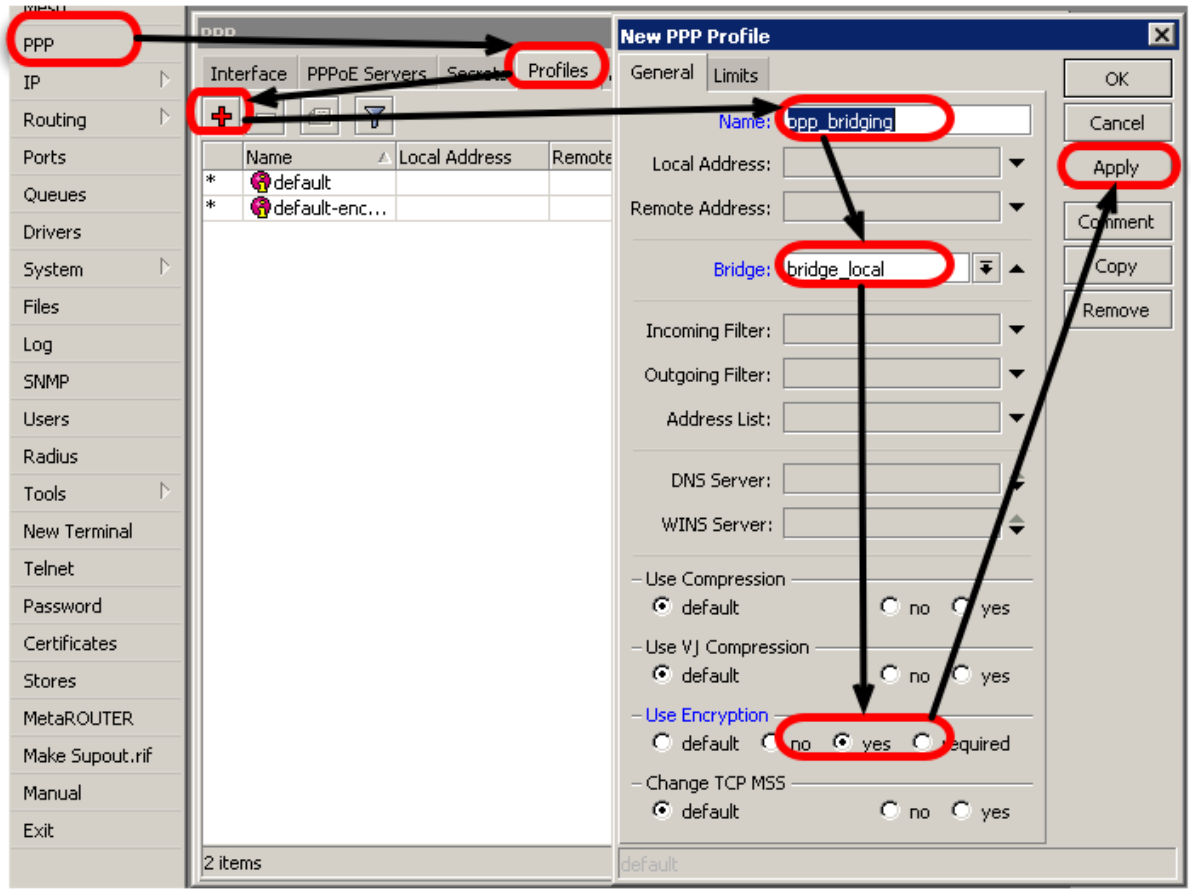
设置静态的 MAC-address：



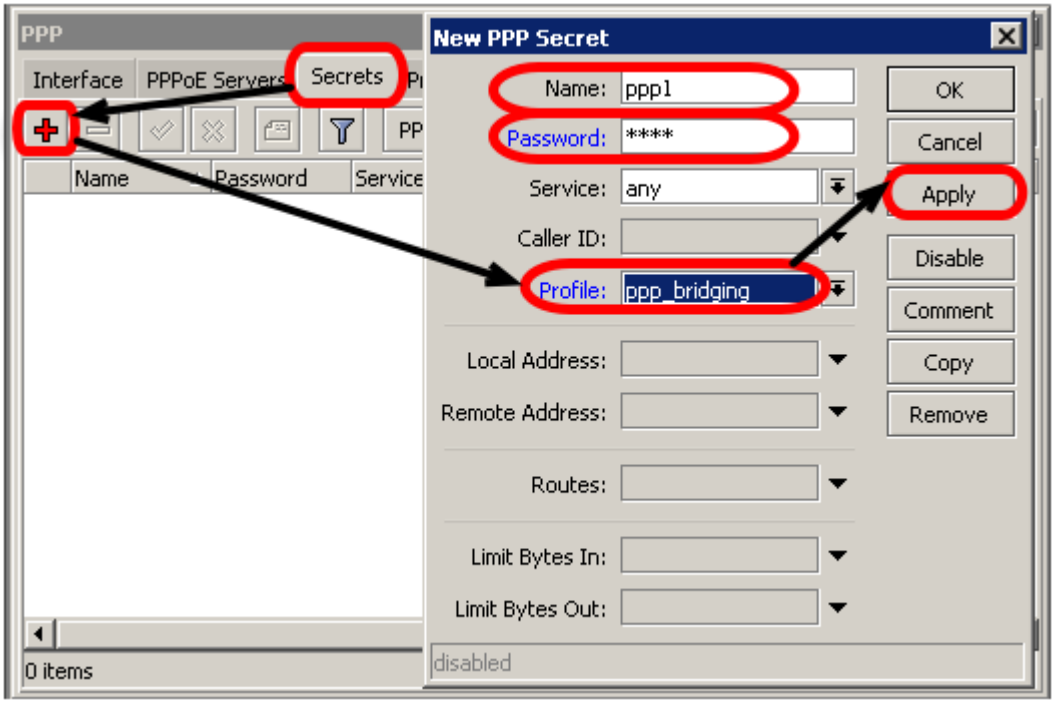
分配 IP addresses,



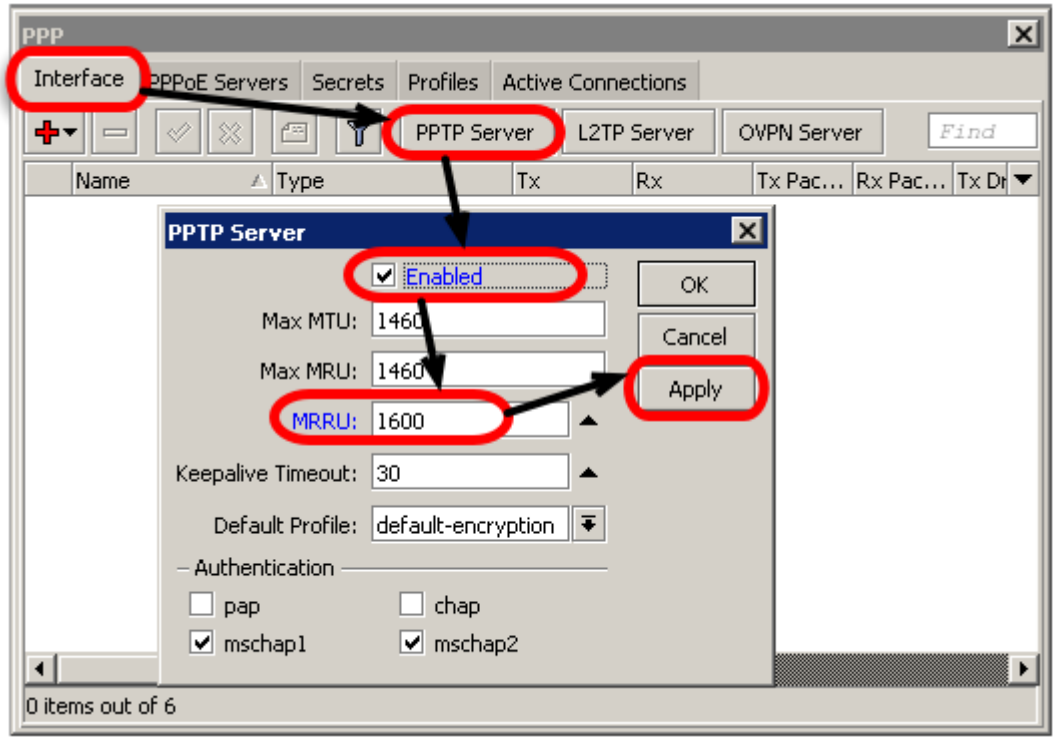
创建 PPP profile，并设置 bridge 参数，



添加 PPP 客户端

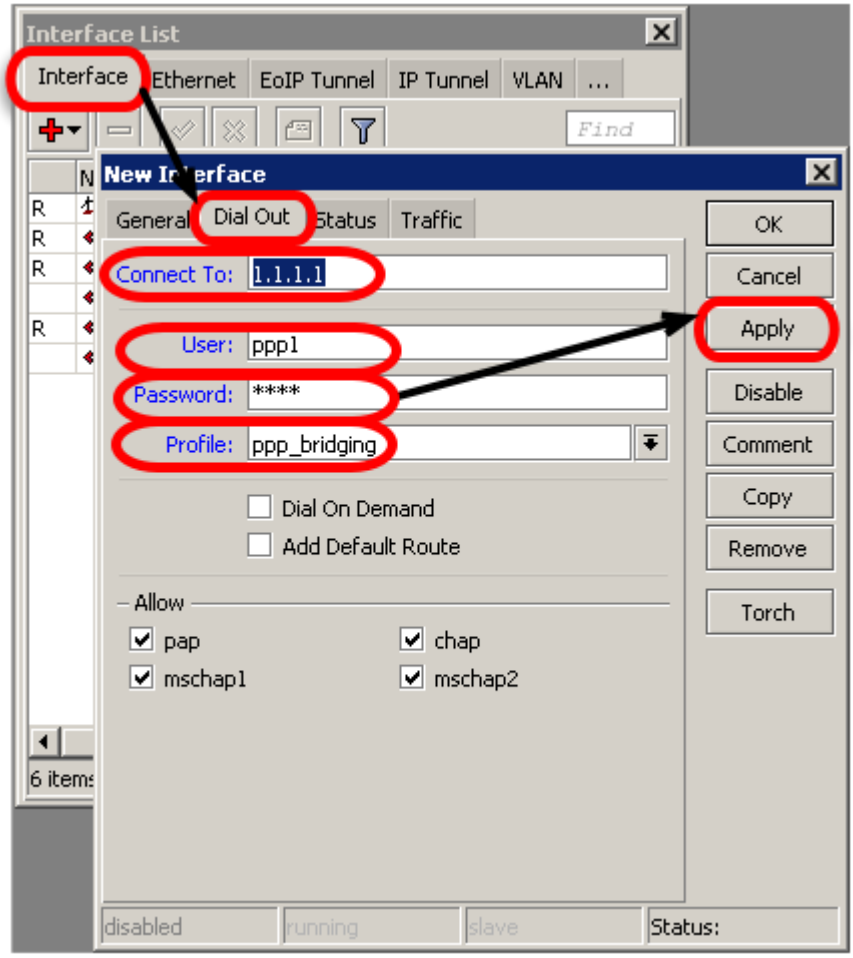


启用 PPTP-server，PPTP 服务器的 MRRU 一定要设置为 1600，否则会导致网页无法打开的情况。



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 来划分远端桥接的区域，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。

第二十五章 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 的单点使用共享金钥，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

25.1 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，命令行添加后默认为禁用
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 连接端口，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
```

25.2 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），例如我们需要将一个来至 test 用户的 OVPN 连接接口设置为一个路由网关，需要指定 gateway=ovpn-test，但如果是动态在连接中断后，gateway 将丢失，需要配置静态接口保持规则存在。
- 动态接口，即 OVPN 客户端连接后自动添加到接口列表，不论用户名是否匹配

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 回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

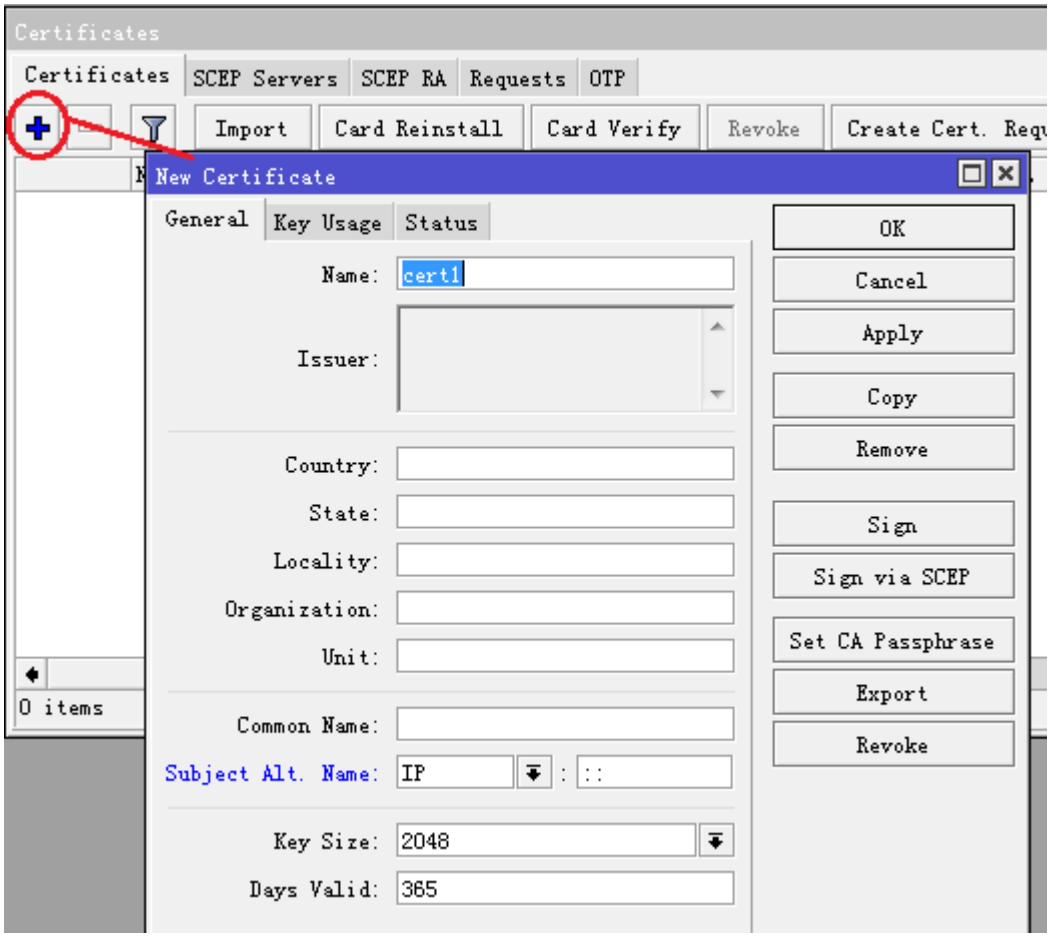
```
[admin@MikroTik] /interface ovpn-server server set enabled=yes
[admin@MikroTik] /interface ovpn-server server set certificate=server
[admin@MikroTik] /interface ovpn-server server print
                    enabled: yes
                       port: 1194
                       mode: ip
                    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。

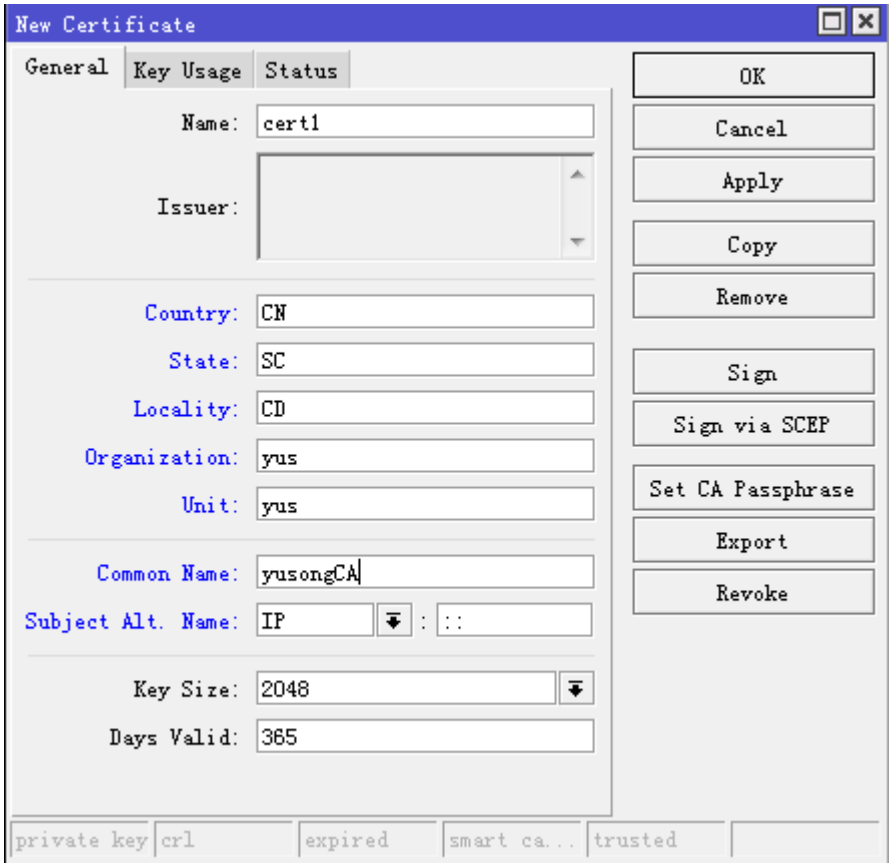
25.3 使用 certificate 创建 OVPN 证书

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。

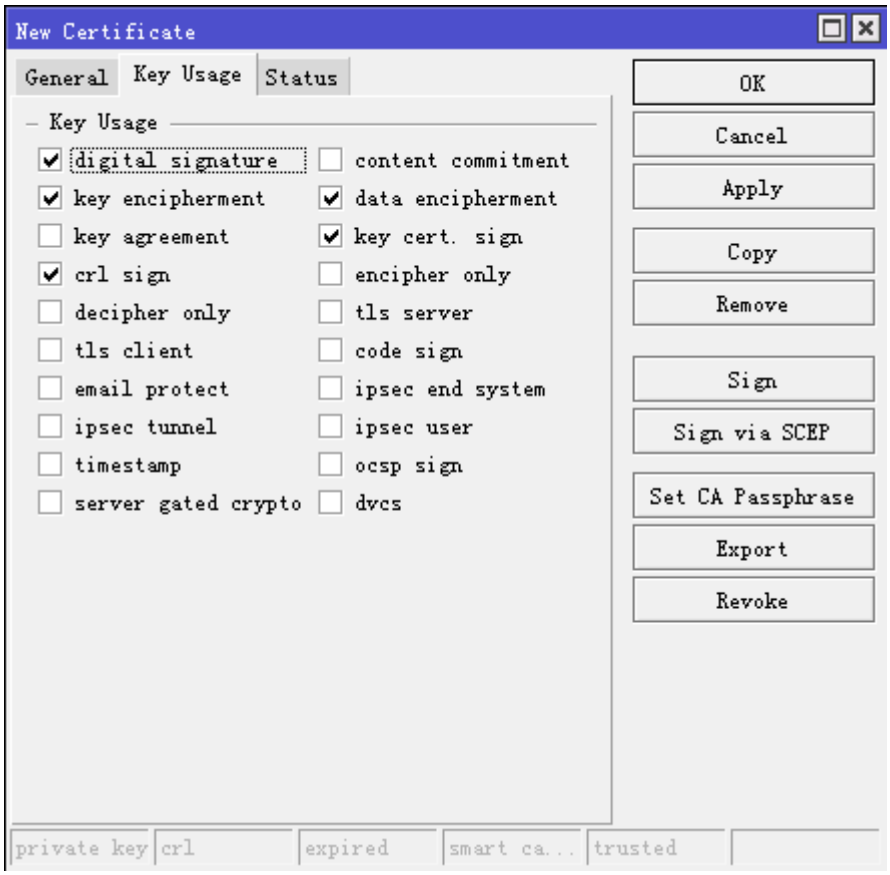
这里我们通过 CLI 命令行和 winbox 介绍，winbox 进入 system->certificates 后，点加号添加证书



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，其他参数默认，Key-size 设置密钥的长度为 2048，Days Valid 为有效日期长度，默认为 365 天。



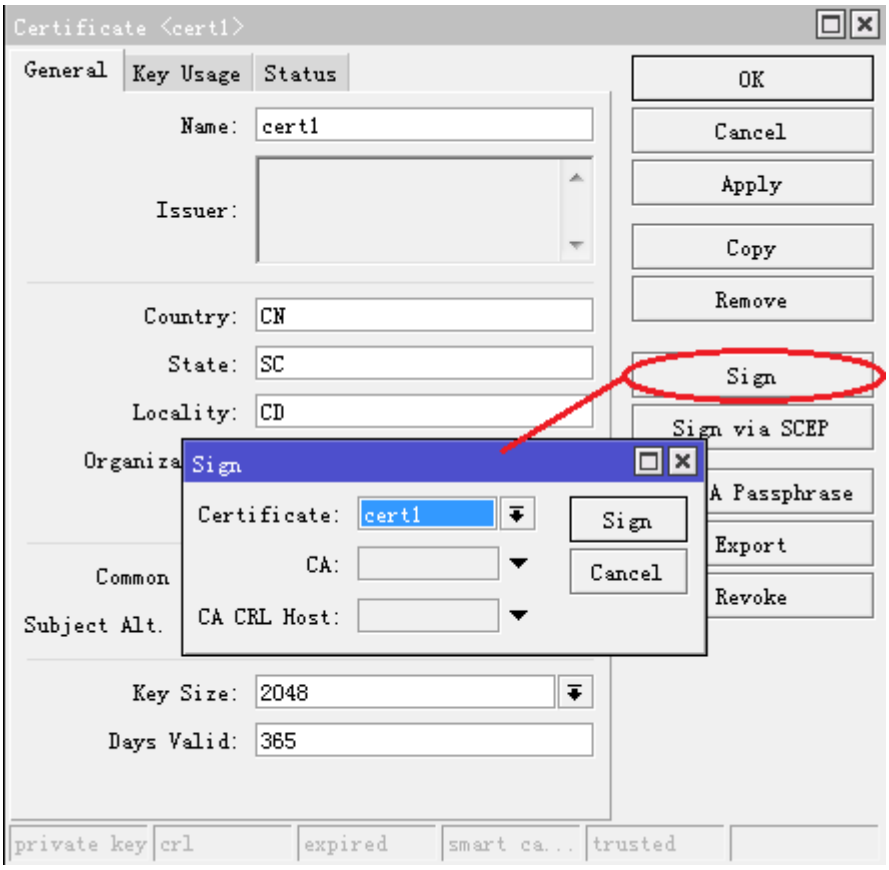
在 Key Usage 下选择默认即可：



命令行操作如下：

```
[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，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

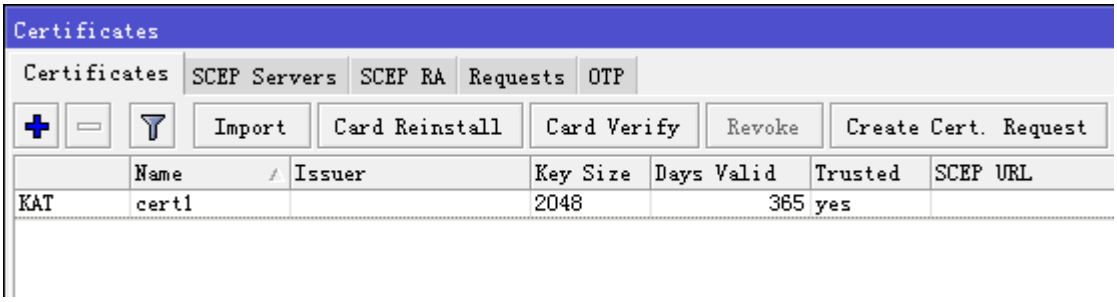


命令行操作如下：

```
[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。

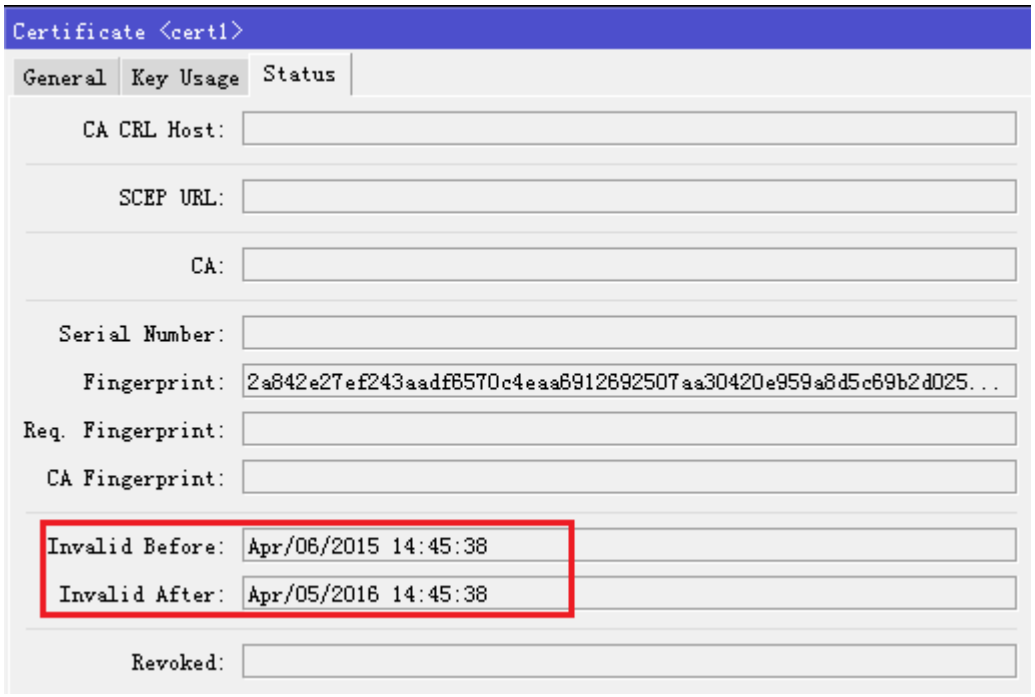
签发成功后，会显示 KAT 的前缀，



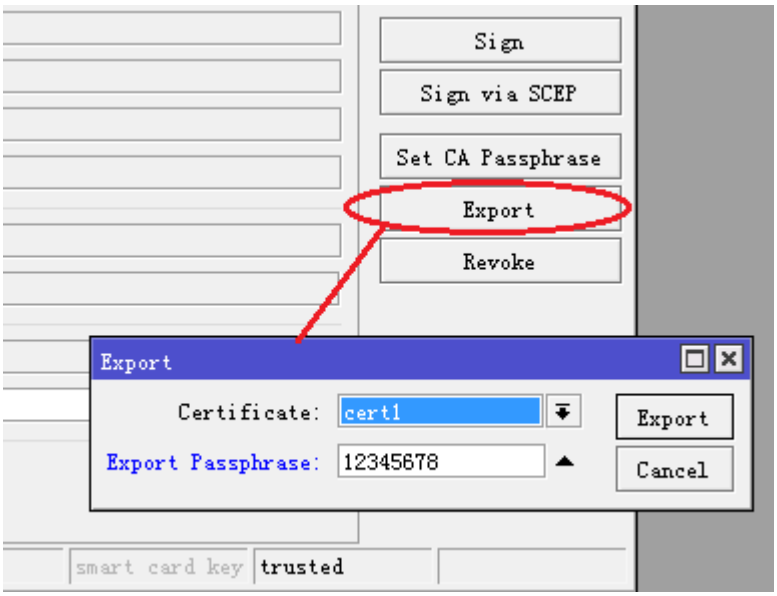
可以看到 Trusted 项为 yes，如果 Trusted 为 no 则代表不信任证书，我们需要在命令行设置为信任

```
[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



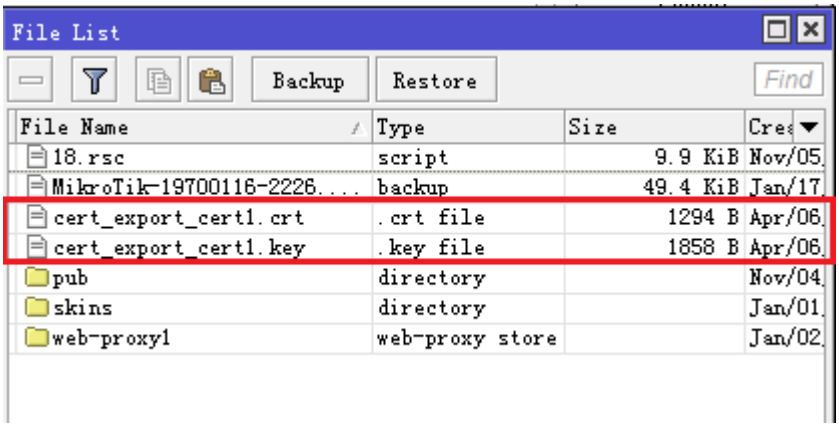
这样证书生成完成，可以使用 export 导出证书，并设置密码为 12345678



命令行操作如下：

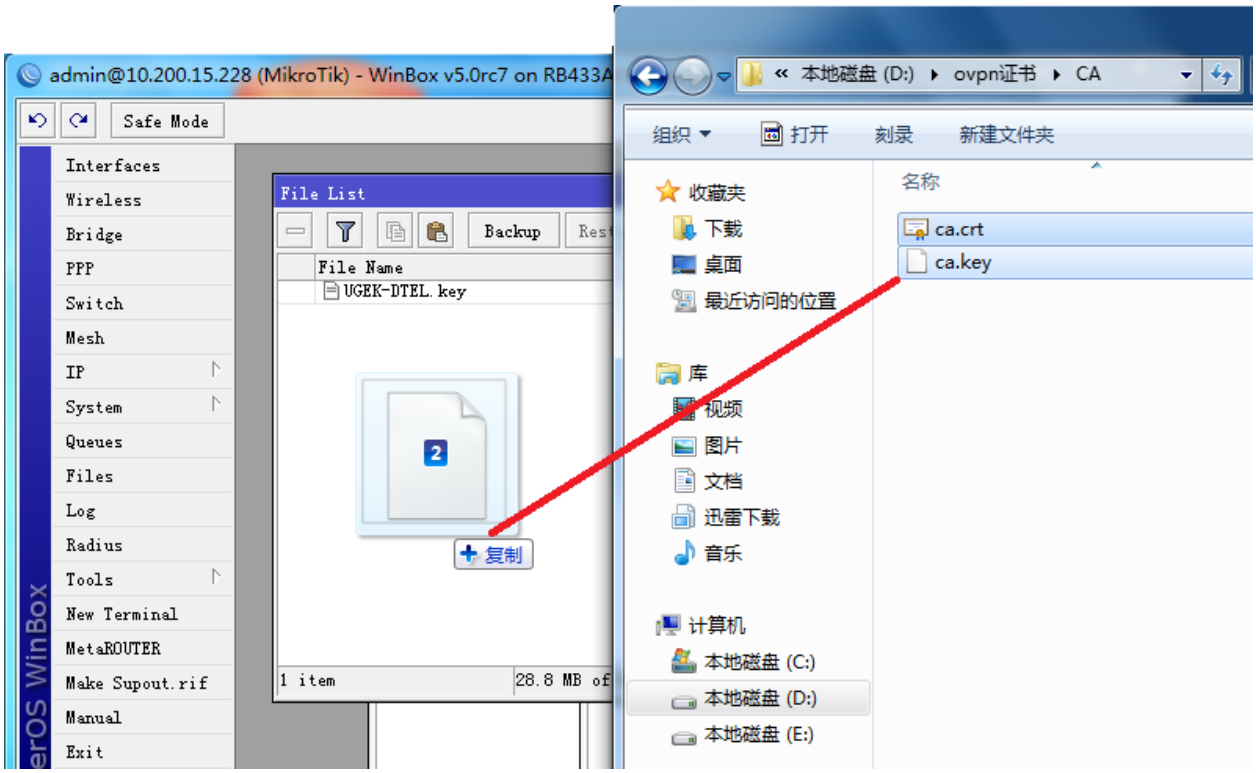
```
[admin@MikroTik] /certificate> export-certificate 0 export-passphrase=12345678
```

在 file 列表下，可以找到导出的证书 crt 和 key 两个文件，然后下载并上传到其他 OVPN 设备

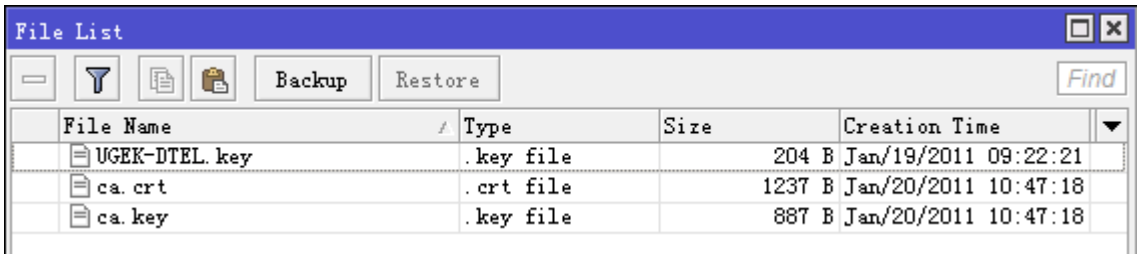


25.4 OVPN 配置

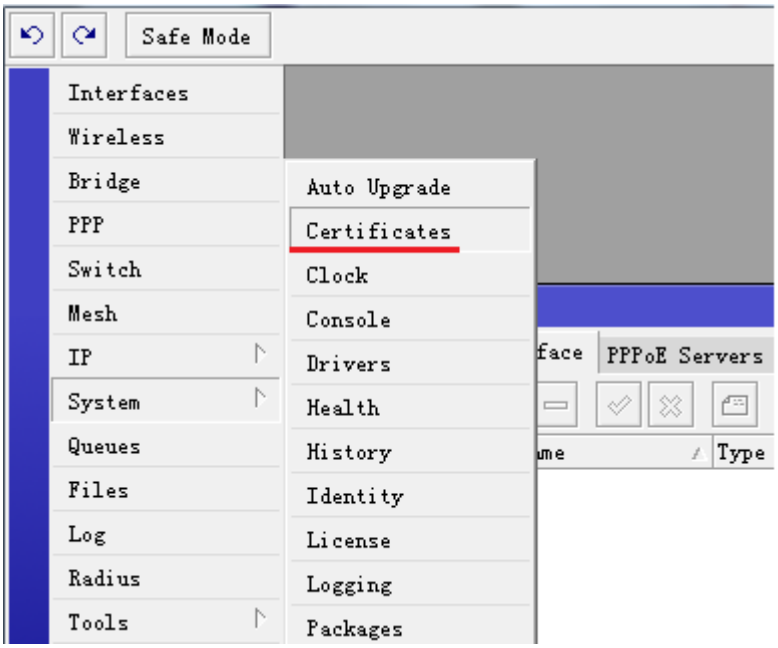
OVPN 服务器配置，首先要导入证书，否则 OVPN 与客户端是无法连接的，我们可以从网上下载已有的证书或者自己通过已经安装 OVPN 的 linux 系统生成新的证书，这里我们根据已有的证书进行操作，如下图：

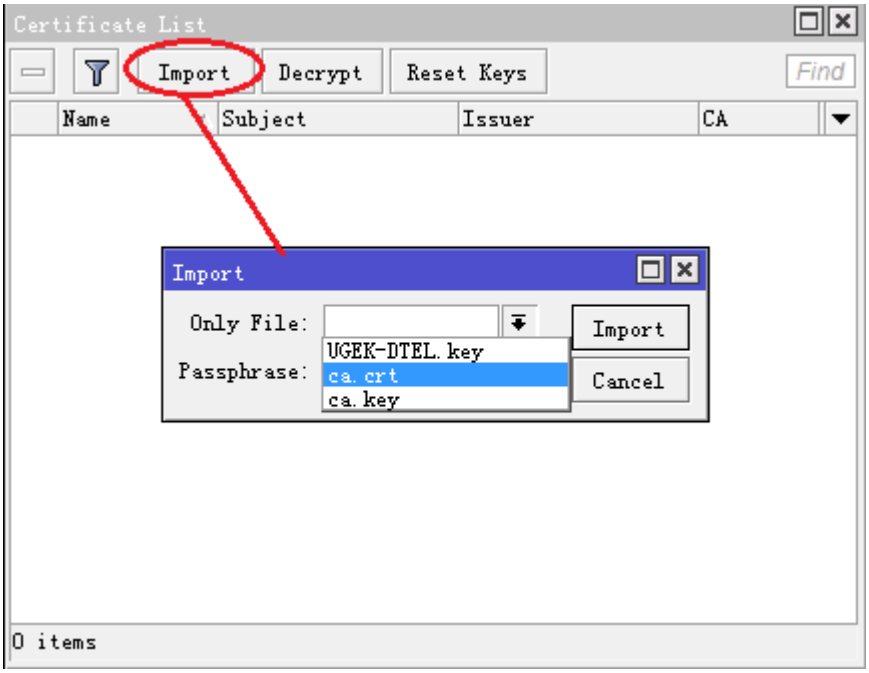


导入完成后，在 file list 可以看到证书

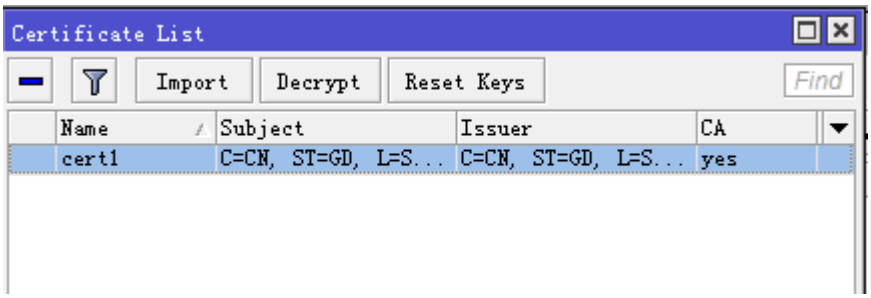


选择 system certificates 证书，并导入 crt 和 key 文件

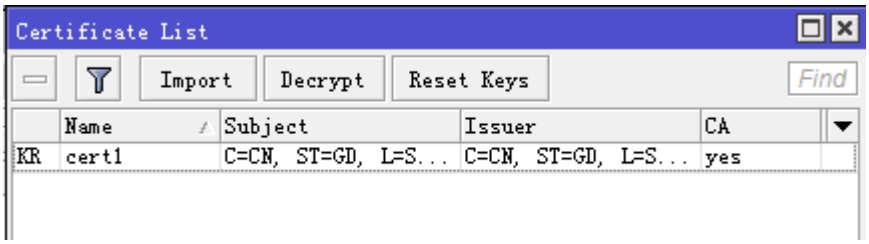




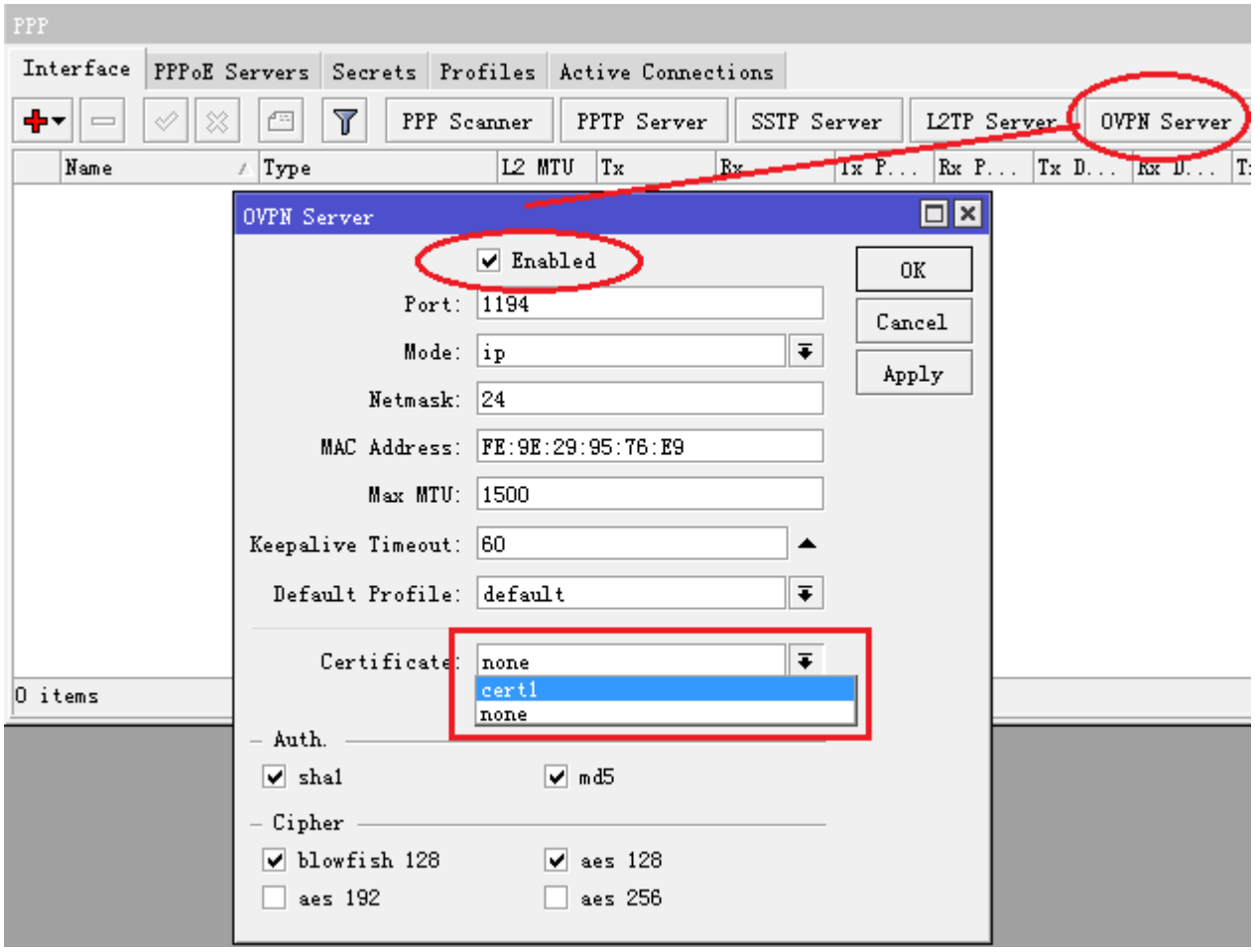
点击 import 导入 ca.crt 文件



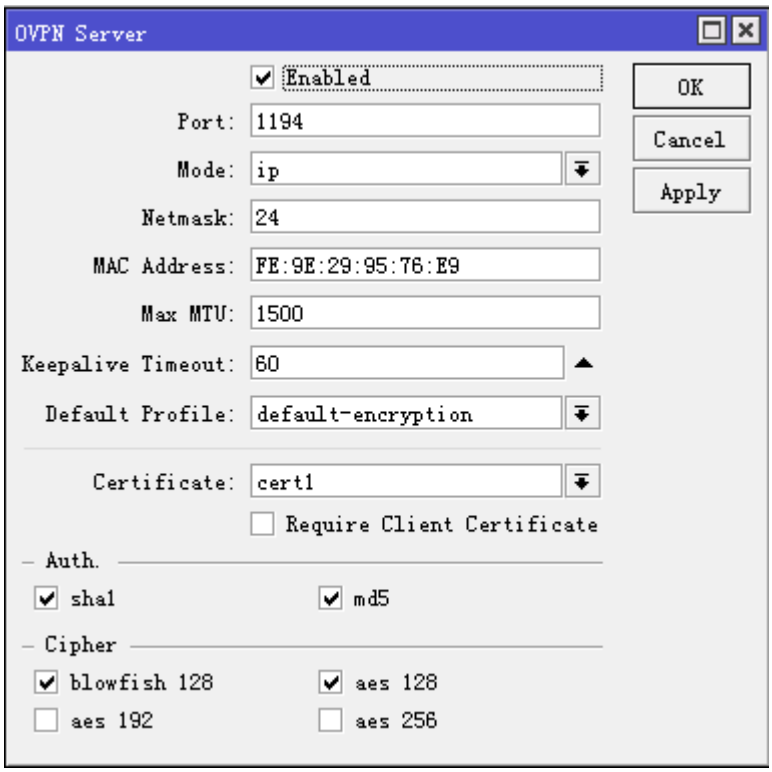
用同样的方式，导入 ca.key 文件，注意 ca.key 如果设置密码，需要在 Passphrase 选项输入密码，在当前目录下的项目出现了 KR 的前缀，表示已经导入 key 并运行



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

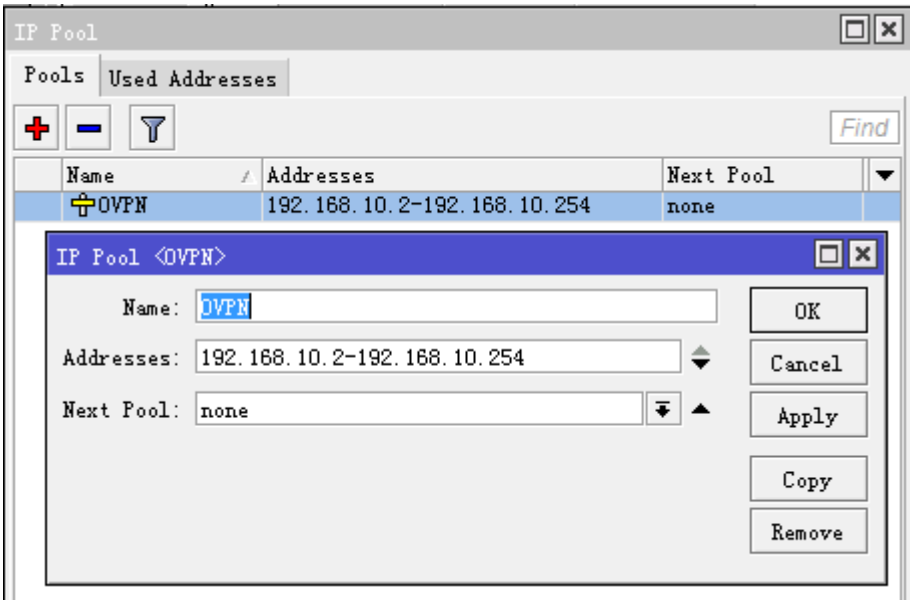


配置完成如下：

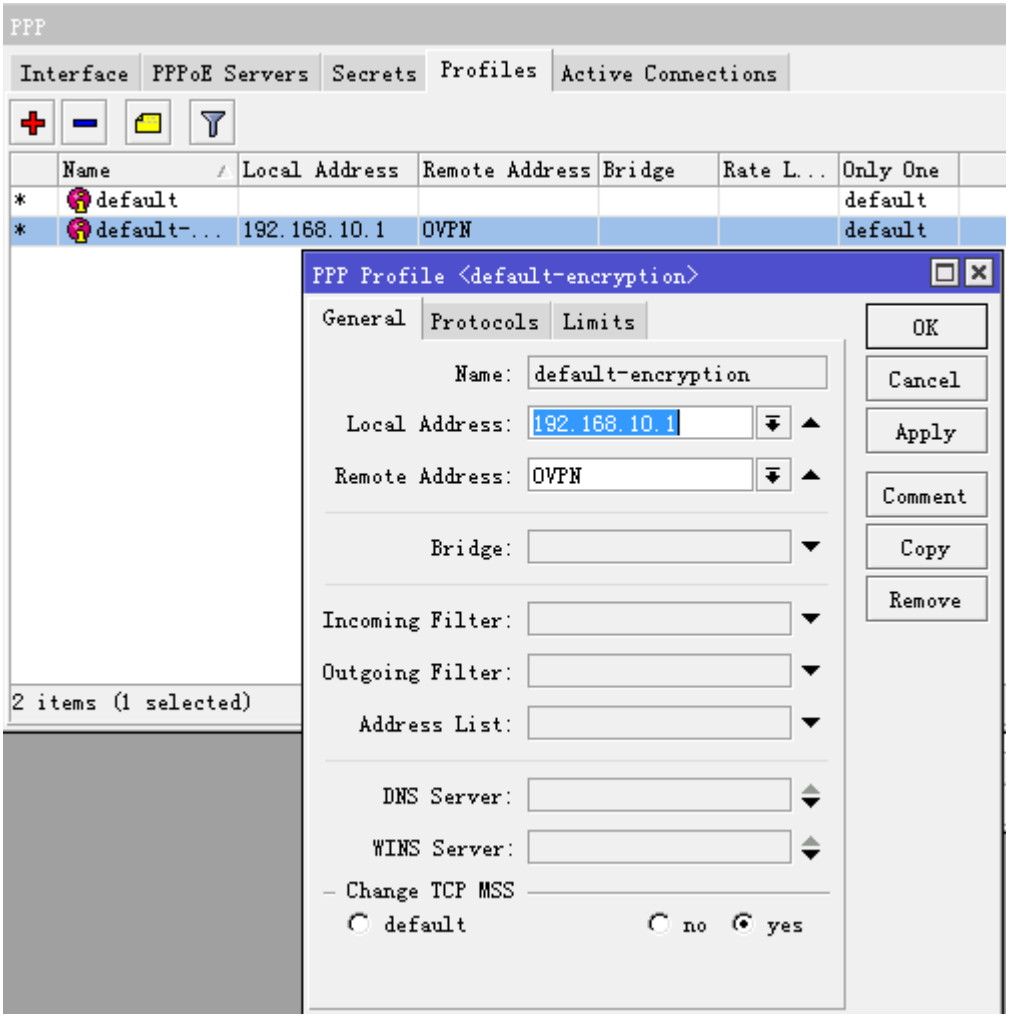


注： require-client-certificate 这里如果选择上了，客户端同样也要导入相同的证书

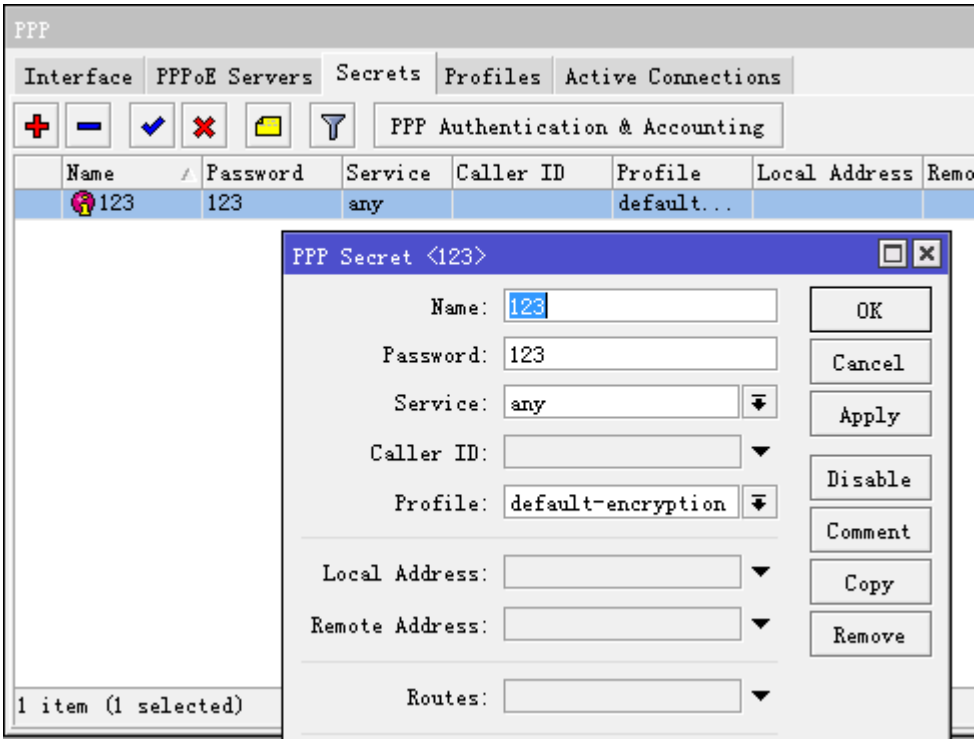
OVPN 服务器配置完成后，配置相应的规则，需要配置地址池 192.168.10.2-192.168.10.254，用于分配给用户的 IP 地址



进入 ppp profile 设置规则 local-address=192.168.10.1 和 remote-address=OVPN

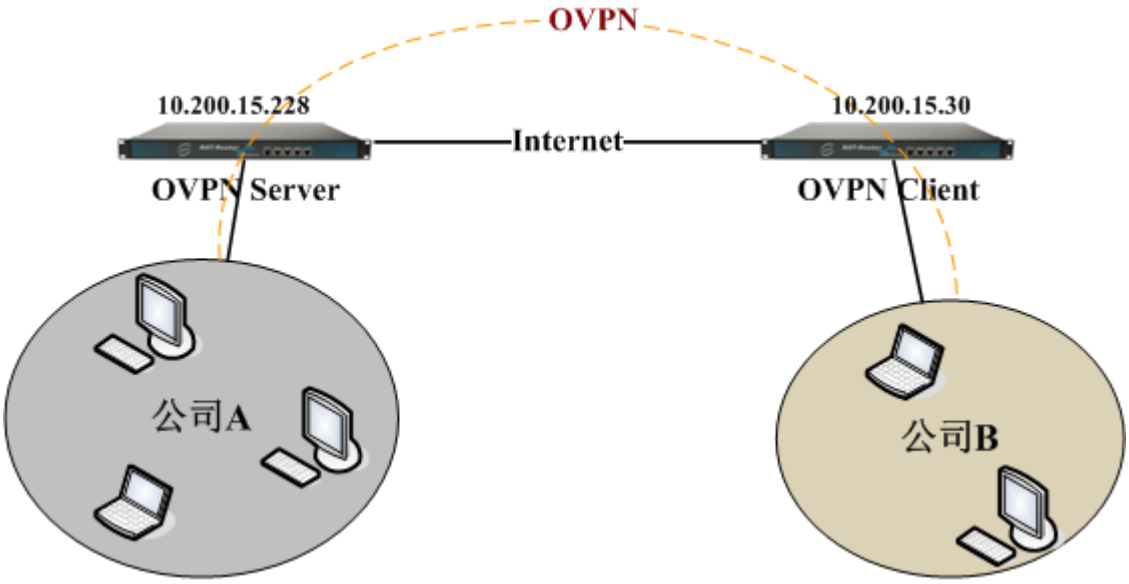


进入 ppp secret 添加用户账号



到此 OVPN 服务器配置完成

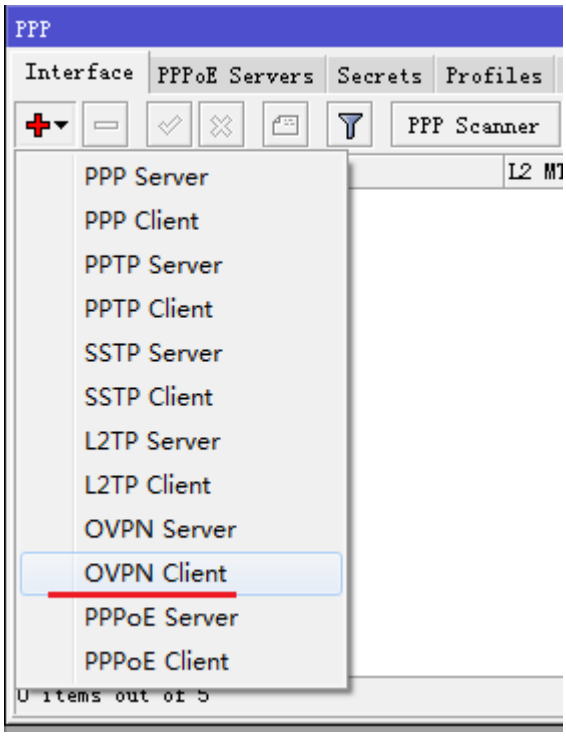
接下来我们需要在另外一台路由器配置 OVPN 客户端，如下面的拓扑图，将 2 台路由建立 OVPN 连接：



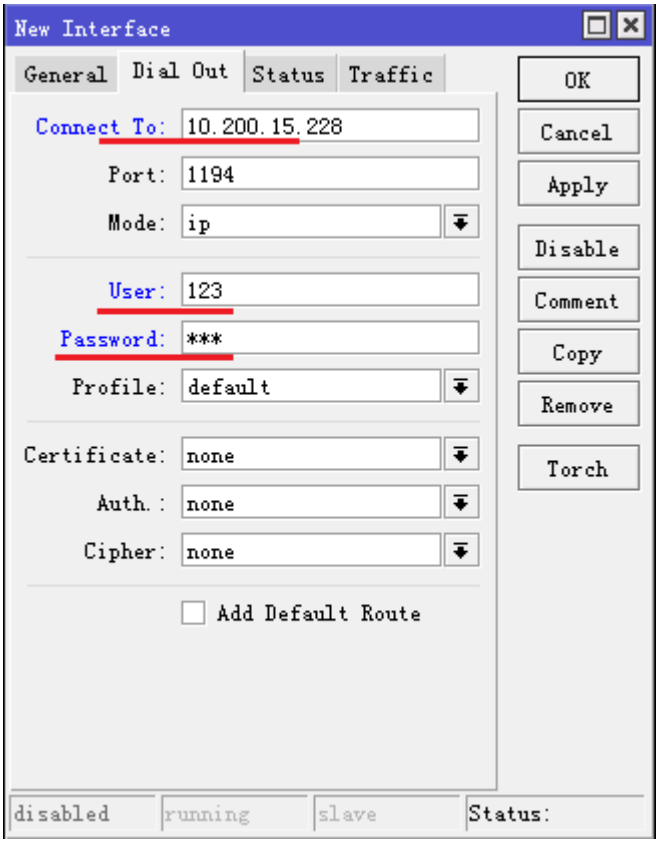
根据以上拓扑图我们假设以下网络环境：

OVPN Server 的 WAN 地址是 10.200.15.228，
OVPN Client 的 WAN 地址是 10.200.15.30

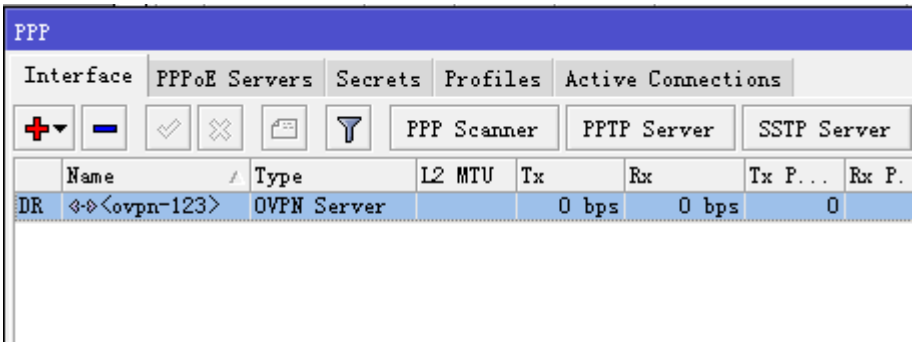
通过这两个 WAN 地址建立 OVPN 的互联，我们在 OVPN Client 建立 ovpn-client 拨号，打开 PPP， 在 interface 里点加号，选择 ovpn-client



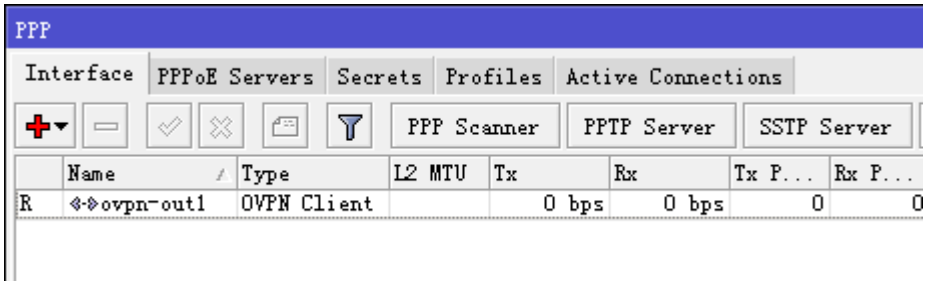
打开后，选择 Dial-out，设置 Connect-to=10.200.15.228，user=123，password=123，其他参数默认



确定后，连接 OVPN 服务器，服务器连接状态如下，显示 DR 前缀

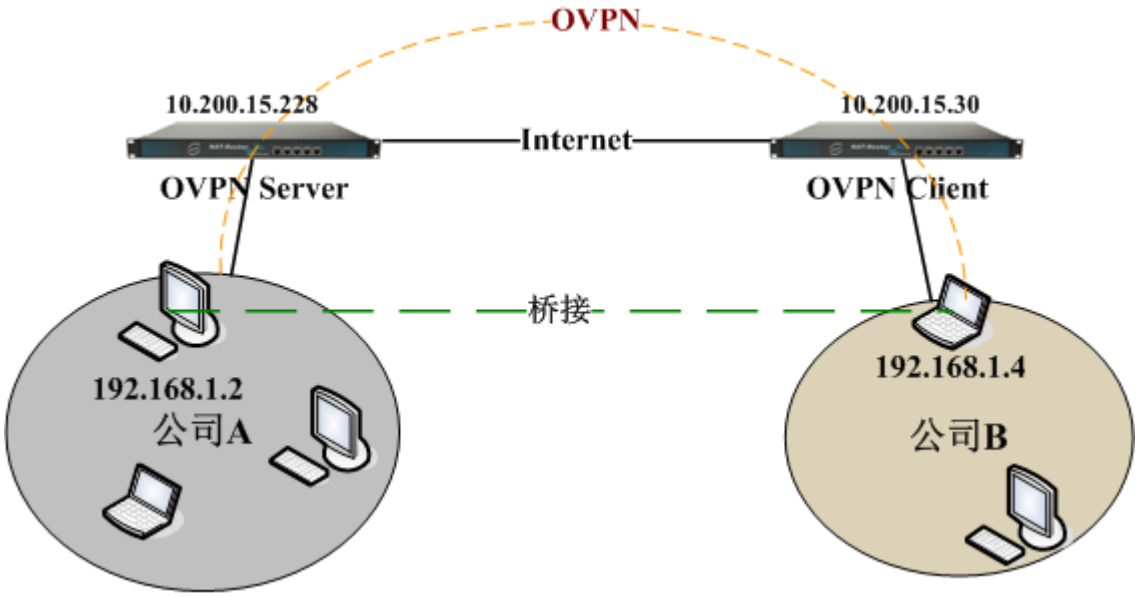


OVPN Client 显示前缀 R，表示连接成功

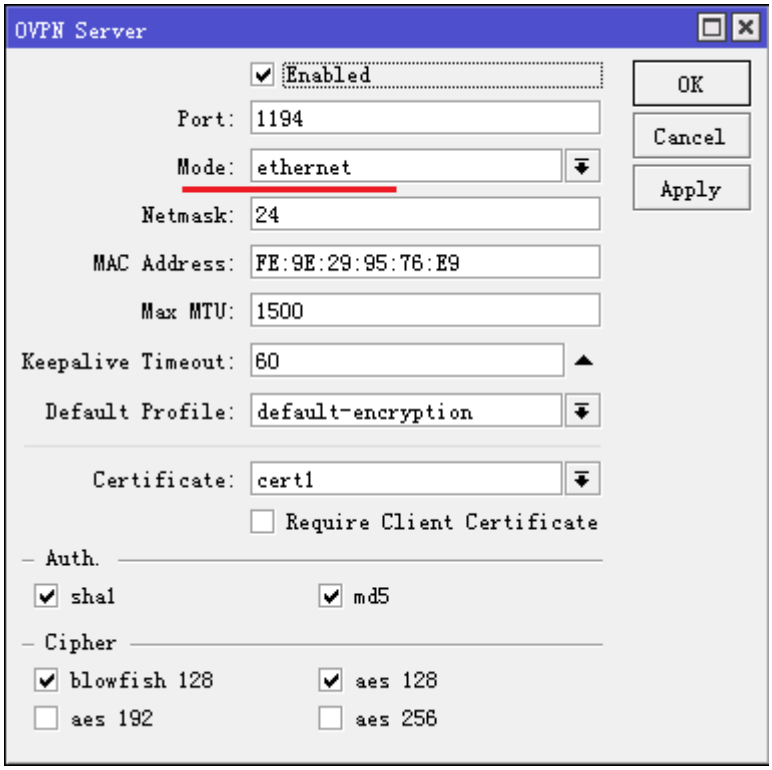


25.5 OVPN bridge 模式

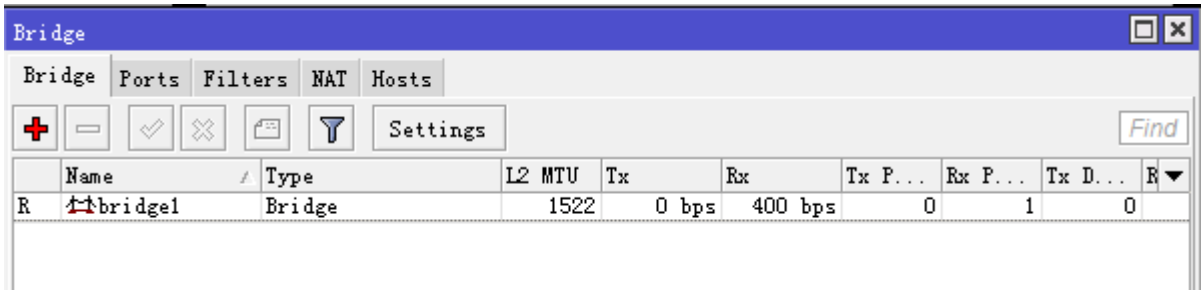
通过 RouterOS 提供的隧道 bridge 功能和 OVPN 的 ethernet 模式实现将两个远程网络建立二层的隧道透传连接，我们依然用之前的网络结构，如下图：



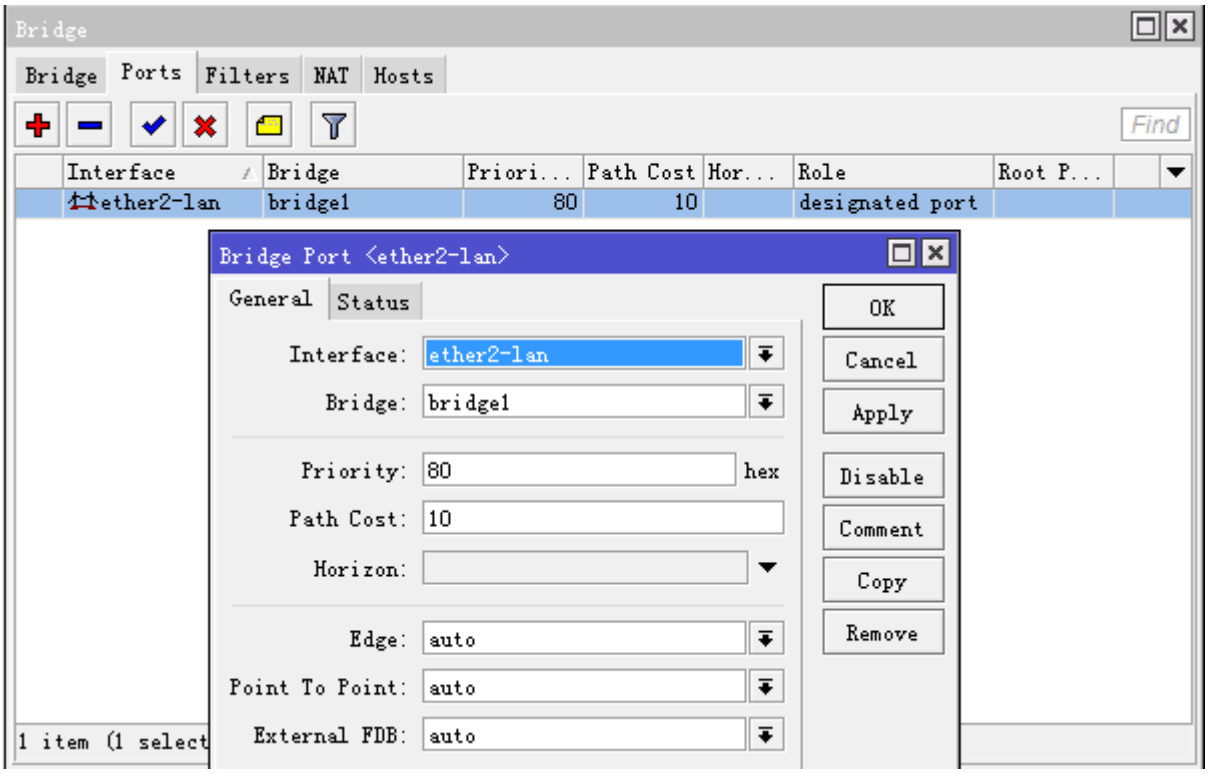
这里我们需要将 OVPN Server 和 OVPN Client 的模式修改为 ethernet



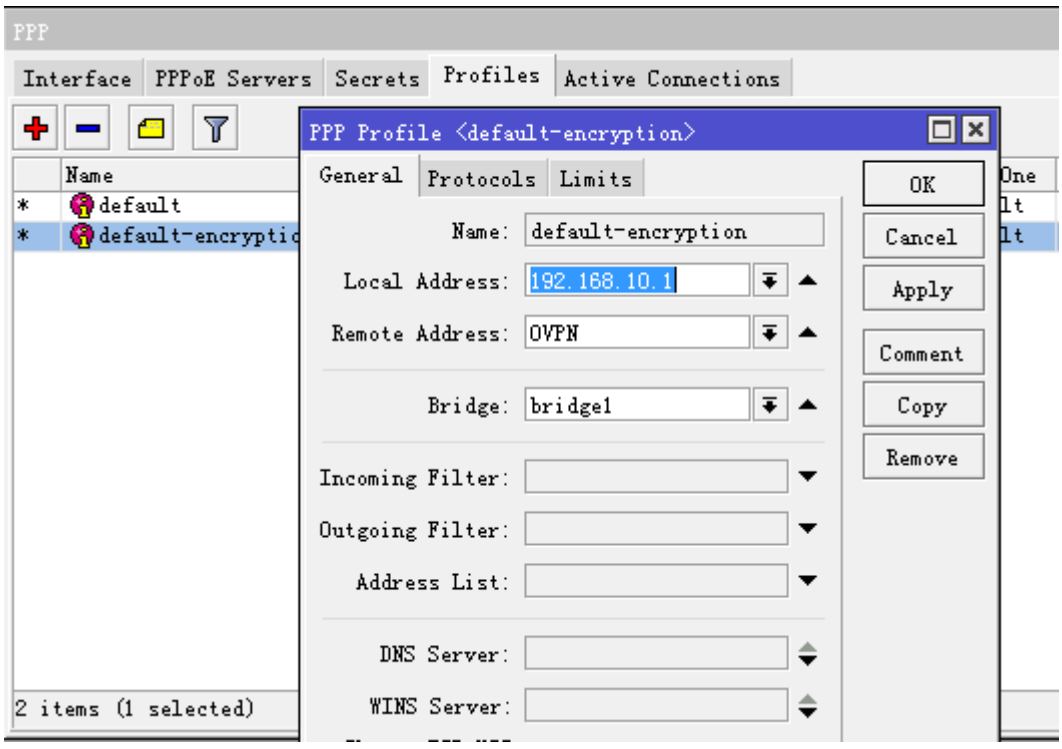
在 bridge 中添加一个桥配置



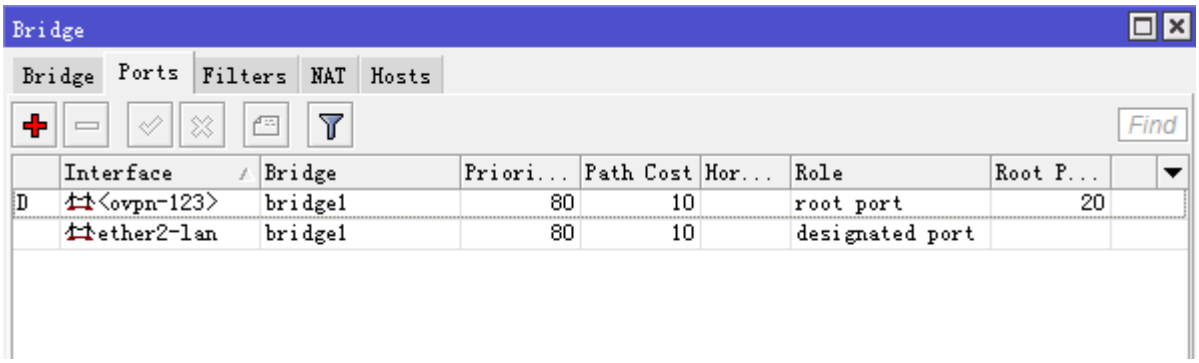
进入 ports 选择，将 ether2-lan 口添加入桥里



回到 ppp profile 里设置 default-encryption 的 bridge 选项设置为刚才我们添加的 bridge1



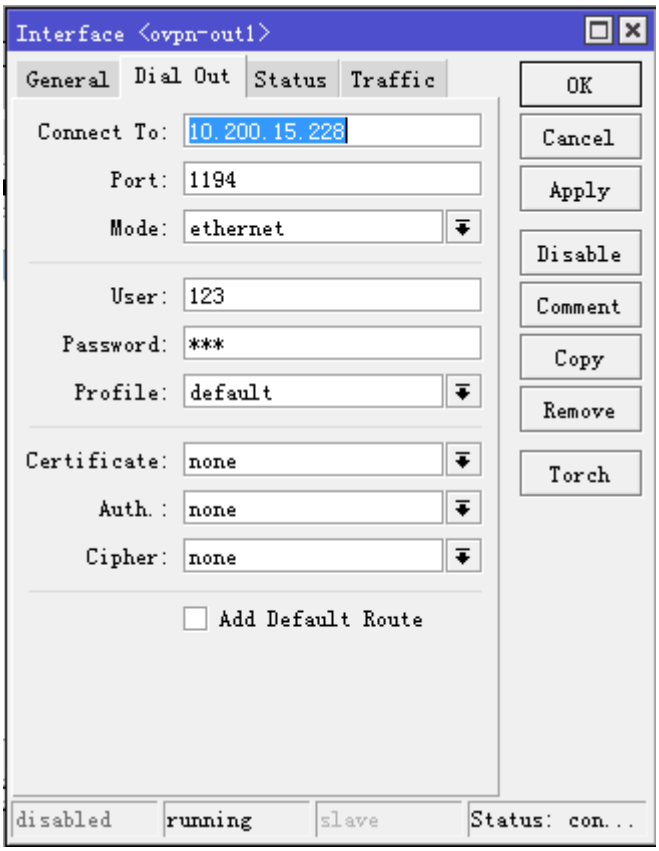
设置完成后，我们建立 OVPN Client 与 OVPN Server 的连接，在 bridge 的 port 里，我们可以看到 123 账号的连接在 port 被自动添加



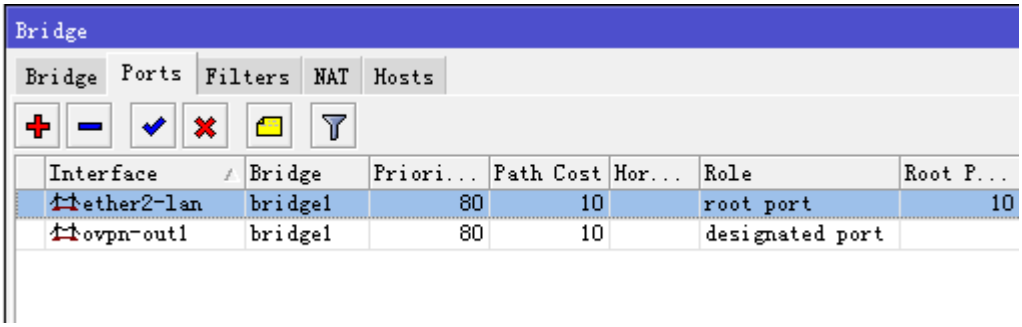
这样 OVPN Server 的 ether2-lan 内网口和 ovpn-123 隧道在同一个桥里

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，这样的方式相对于 EoIP 隧道更安全，但需要考虑到数据加密会产生一定的开销

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
```

第二十六章 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。但两者的数据包通过防火墙、NAT、WEB PROXY 时却都有可能发生一些连线方面的问题。

PPTP 数据包通过防火墙时，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，从而造成连线的问题;而当你的机器位于 NAT 之后，NAT 亦必需被设定成能转发 GRE 协议封装的数据包。否则就会造成只能建立 PPTP 的 TCP 连接，而无法接收 GRE 协议封装的数据包;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 在透传方面由于前两种 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。

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**)远端 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包长度。如果一个数据大于隧道的 MTU 值，将会被拆分多个数据包，允许最大长度的 IP 或者以太网数据包发送到隧道
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*; 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 数据包时钟周期（以秒为单位），如果没有数据，并且没有在指定的时钟周期回应，这些没有回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包长度。如果一个数据大于隧道的 MTU 值，将会被拆分多个数据包，允许最大长度的 IP 或者以太网数据包发送到隧道
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：

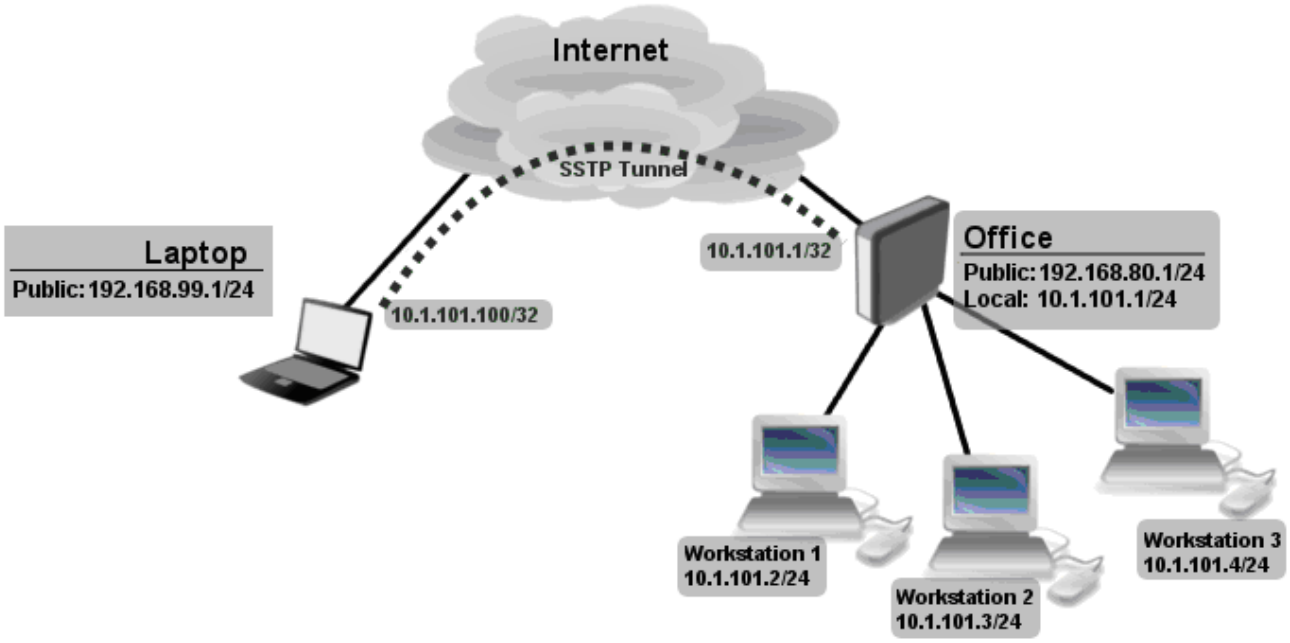
```
[admin@dzeltenais_burkaans] /interface sstp-server> monitor 0
    status: "connected"
    uptime: 17m47s
 idle-time: 17m47s
      user: "sstp-test"
 caller-id: "10.1.101.18:43886"
       mtu: 1500
```

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

26.1 端到服务器连接

下面事例显示了如何使用 SSTP 加密隧道通过一个电脑连接到远端办公网络，SSTP 服务器分配给电脑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
```

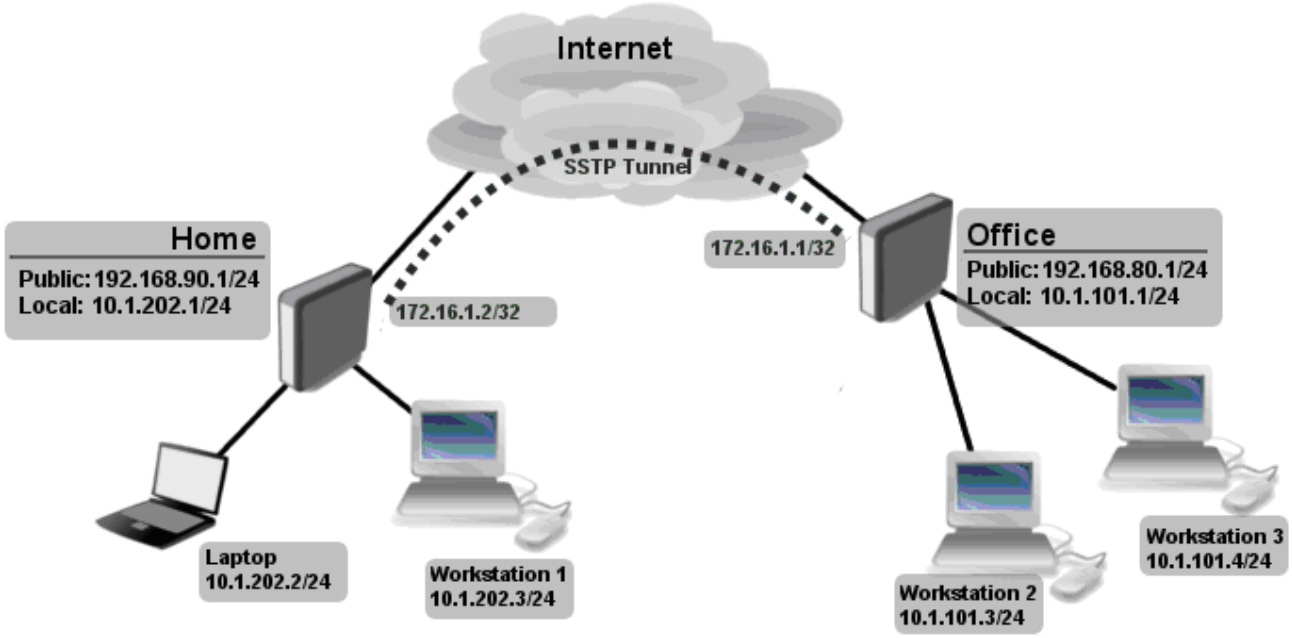
这里（当 SSTP 客户端连接成功），如果你想从笔记本电脑 ping 内网主机，将显示 timeout，因为笔记本电脑将不能从内网主机获取 ARP，解决方法是设置内网网卡的 arp 为 proxy-arp

```
[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

26.2 点对点的 SSTP

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 通远端的网络。

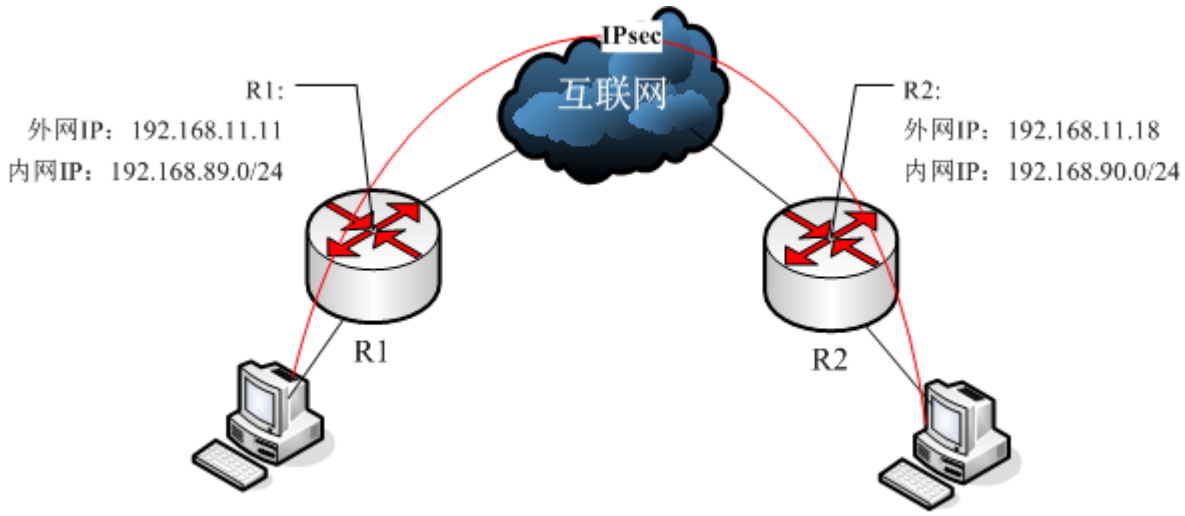
第二十七章 IPSec 配置

IPSec 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。服务的实施通过两种通信安全协议：认证头（AH）和封装安全负载（ESP）以及 Internet 密钥交换（IKE）协议来达到这些目标。

IPSec AH 协议提供数据源认证、无连接的完整性和可选的抗重放服务。ESP 协议提供数据保密性，有限的数据流保密性、数据源认证、无连接的完整性及抗重放服务。IKE 协议用于协商 AH 和 ESP 协议所使用的密码算法，并将算法所需的必备密钥放在合适的位置。IPSec 有两种模式：传输模式和隧道模式。它们都是对外出的数据包添加 IPSec 头进行加密和认证，而对于接收的 IPSec 数据包作解密认证处理和适当的转发传送。

27.1 IPSec 点对点配置实例

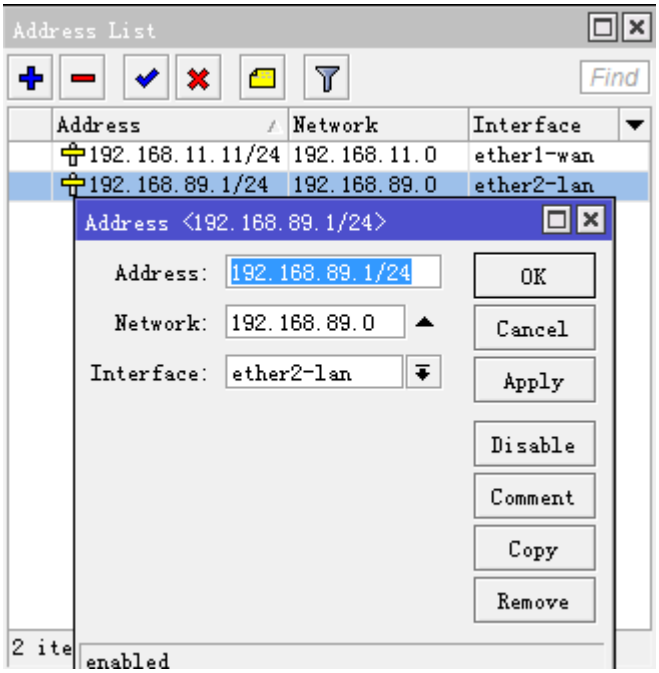
以下是一个使用 RouterOS 建立的 IPsec VPN 案例，网络拓扑图：



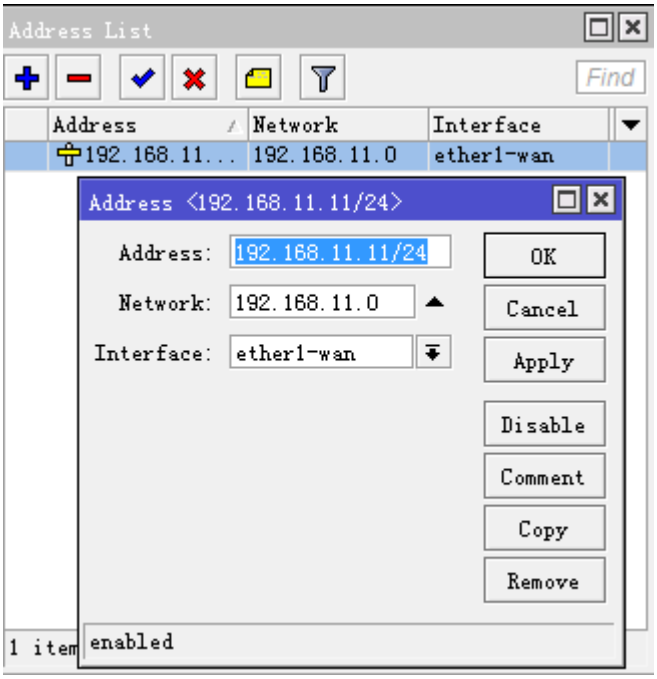
192.168.89.1/24--R1--192.168.11.11/24---互联网---192.168.11.18/24---R2---192.168.90.1/24

R1 配置

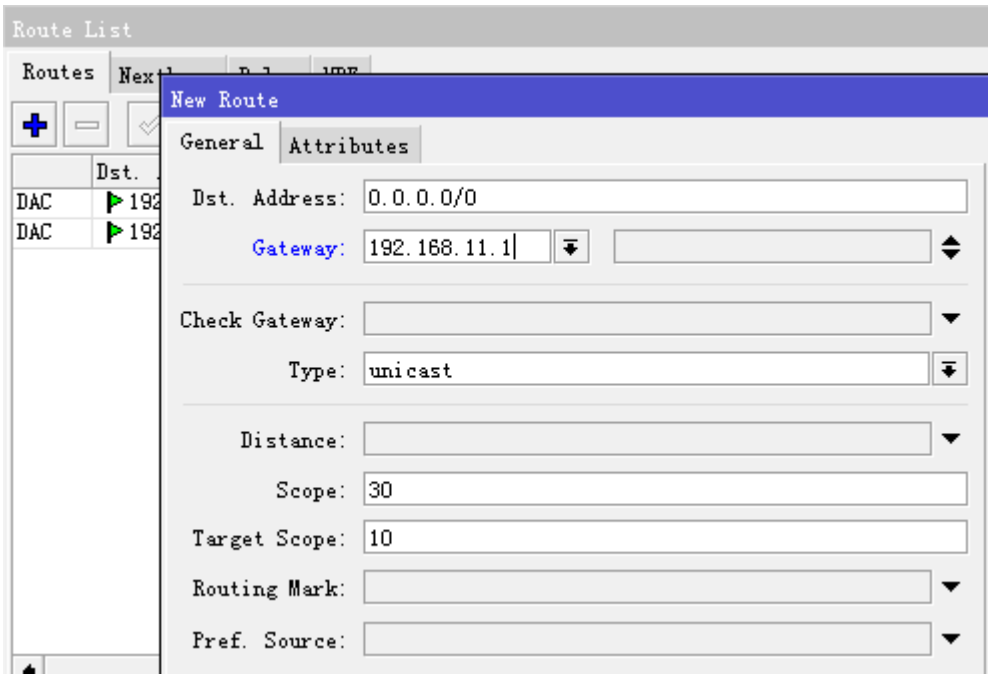
进入 ip address 添加各接口的 IP 地址，先配置内网接口地址：



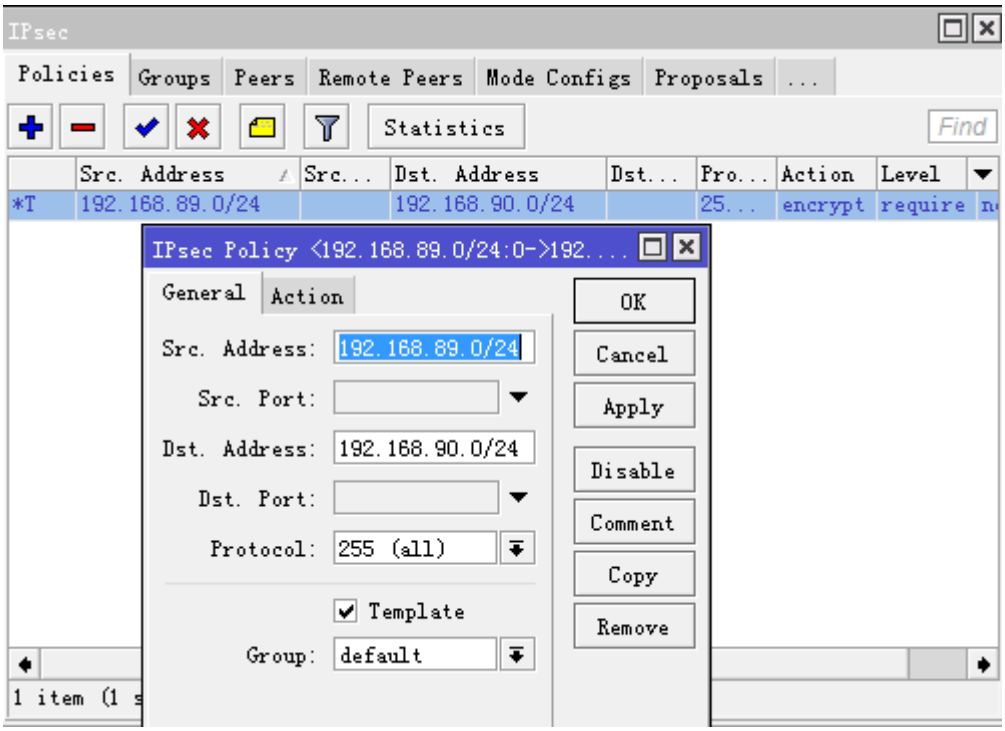
添加 ether1-wan 互联接口地址：



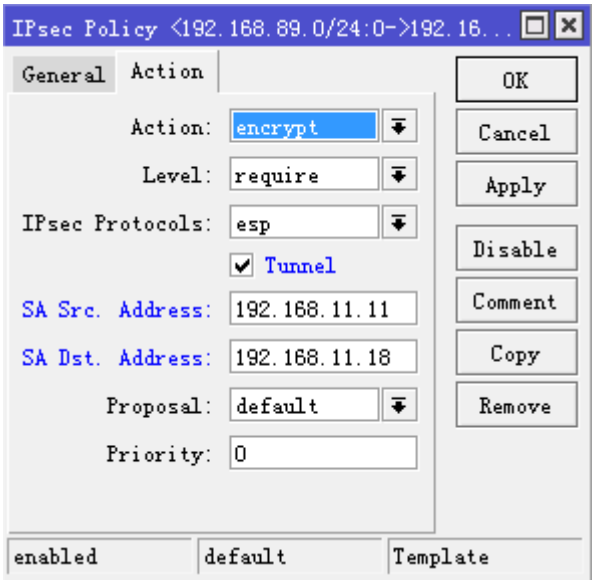
进入 ip routes 里面添加网关出口：



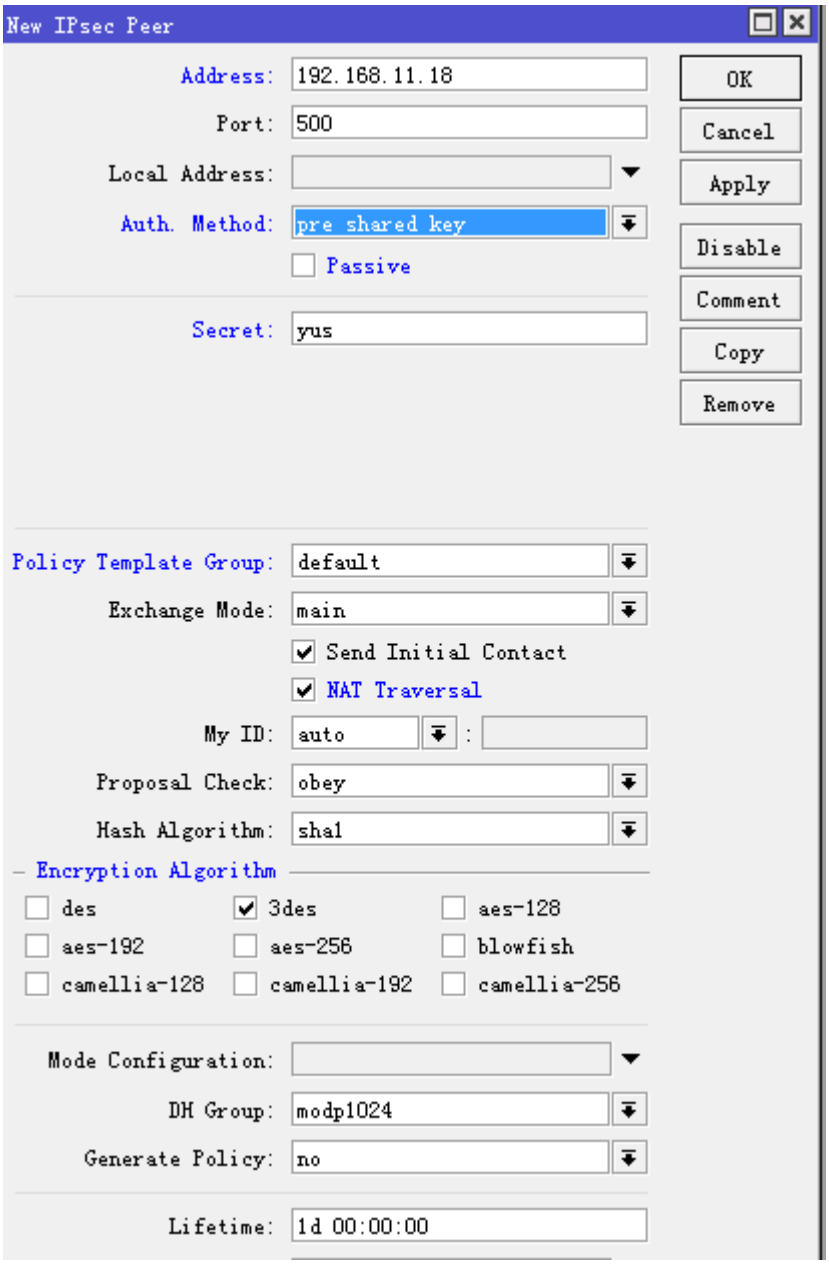
进入 ip ipsec 里的 policies 选项，先在 general 添加内网的源地址和对端网络的内网地址。



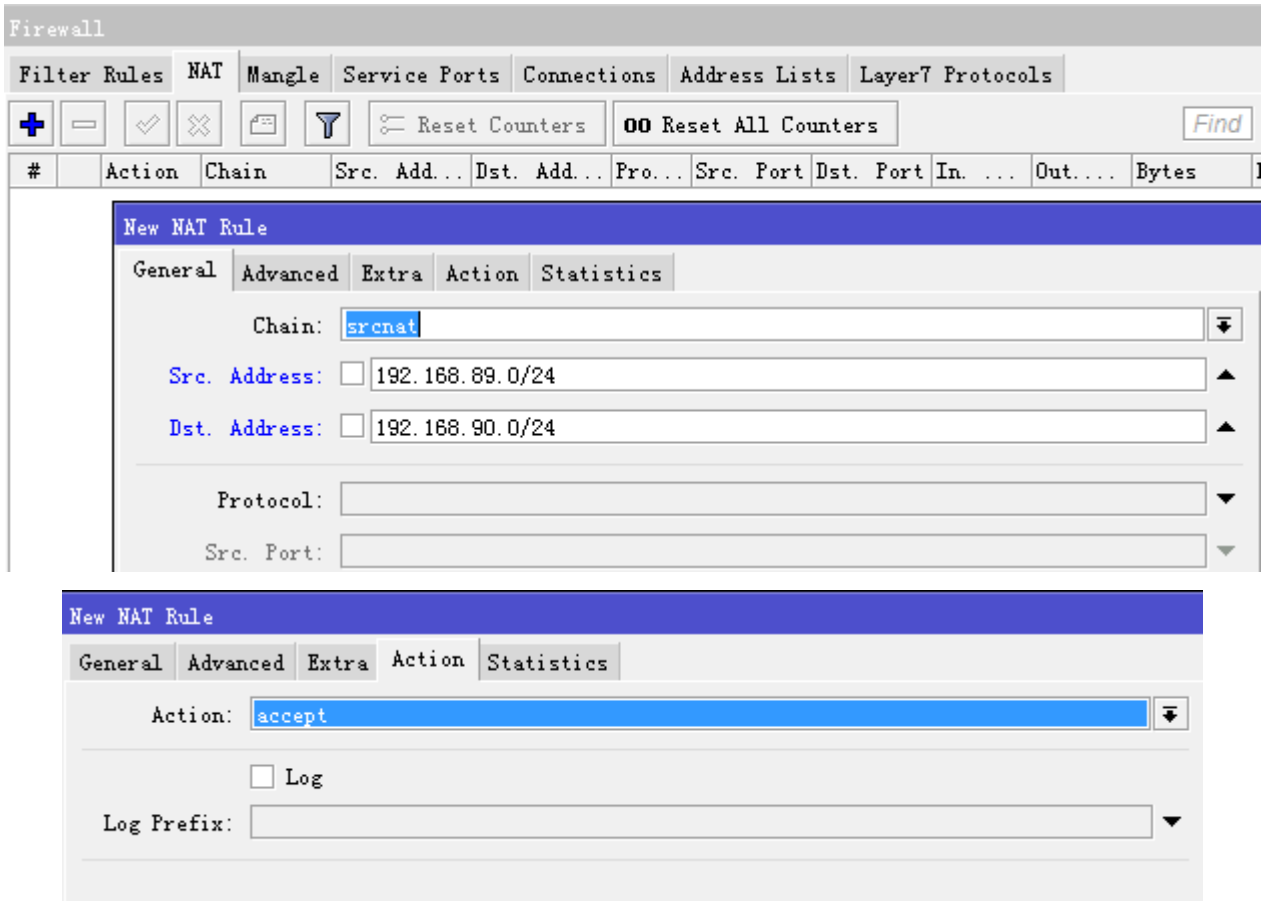
选择 action 选项里面添加源外网地址和对端外网地址和开启 tunnel 隧道协议



进入在 ip ipsec 里的 peers 标签里添加目标外网 ip 地址和 secert 密码：



在 ip firewall 里的 nat，建立一条 srcnat 规则为 accept，接受源内网地址和对端内网地址通过：



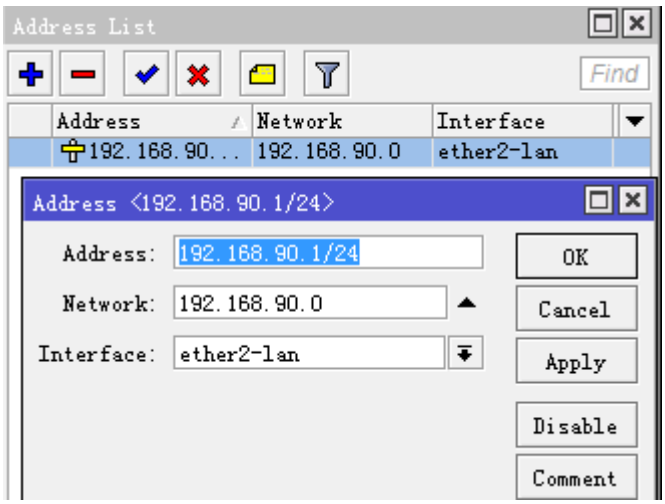
然后建立将本地内网隐藏上网的 masquerade 规则：

```
/ip firewall nat add src-address=192.168.89.0/24 action=masquerade
```

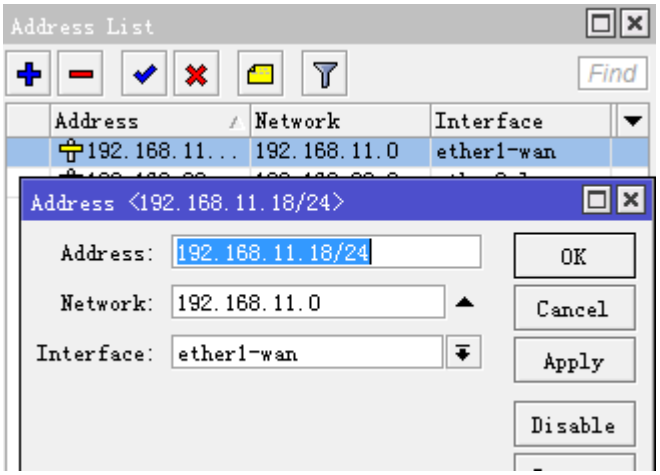
以上就是 R1 的相应配置过程，R1 已经配置完成现在就 R2 了。

R2 配置

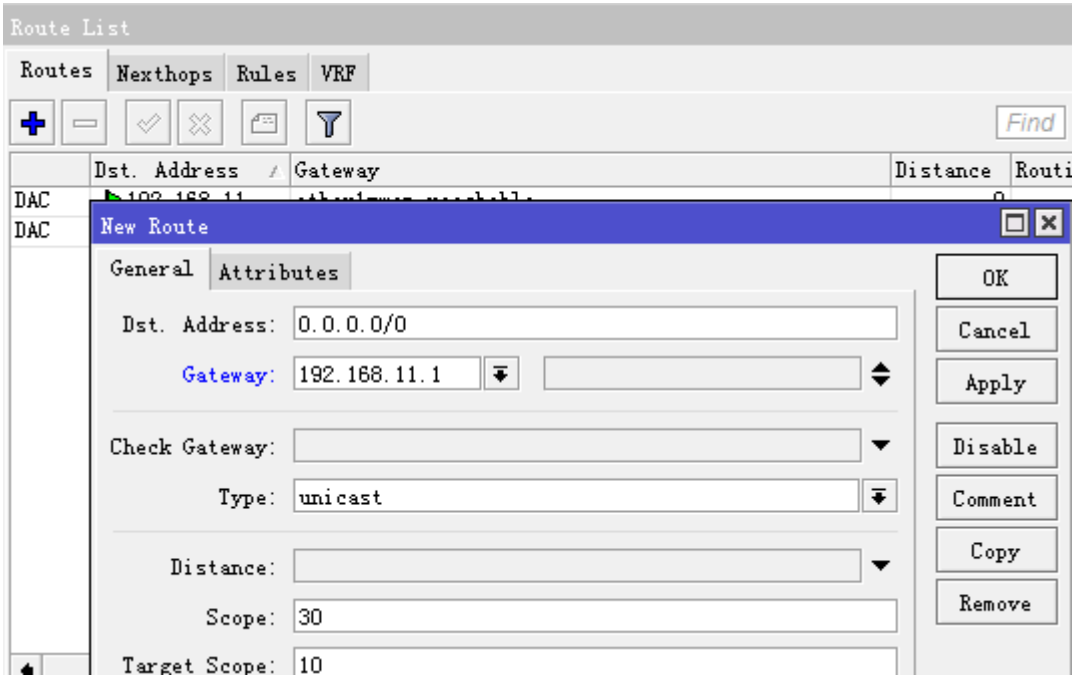
进入 ip address 添加 ether2-lan 内网接口地址：



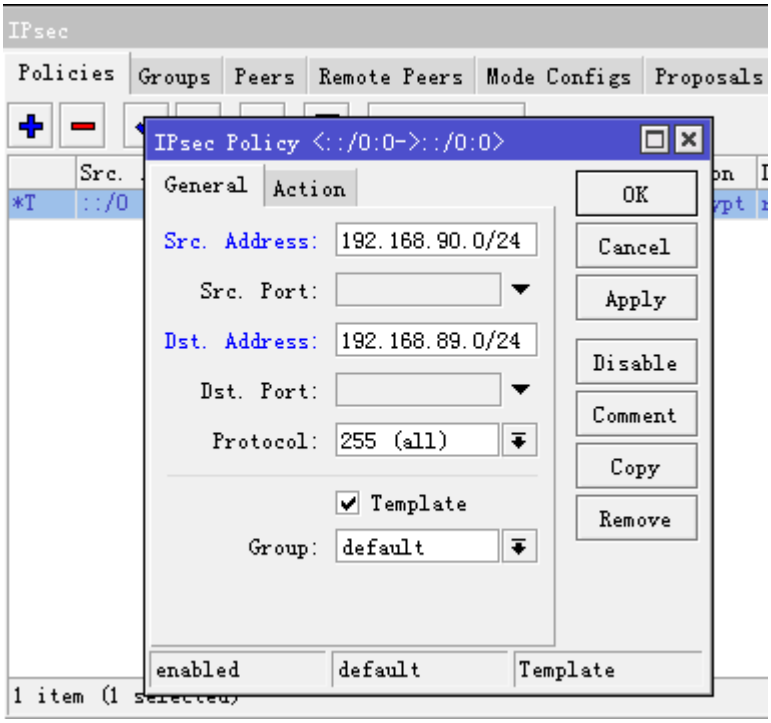
添加 ether1-wan 外网接口地址：



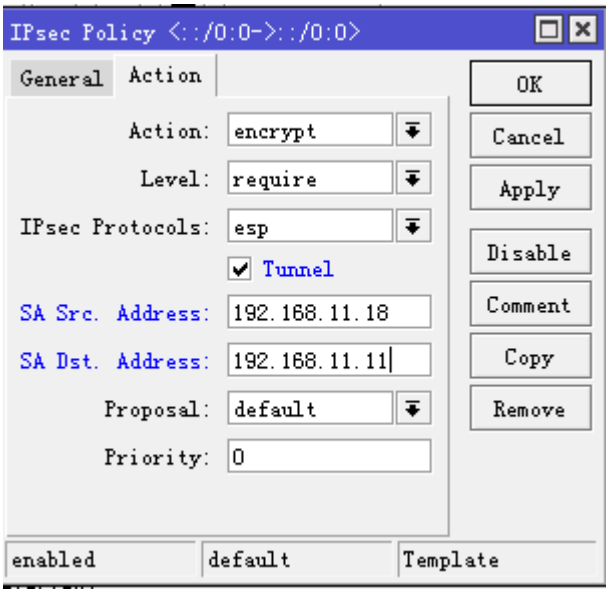
进入 ip routes 添加外网网关：



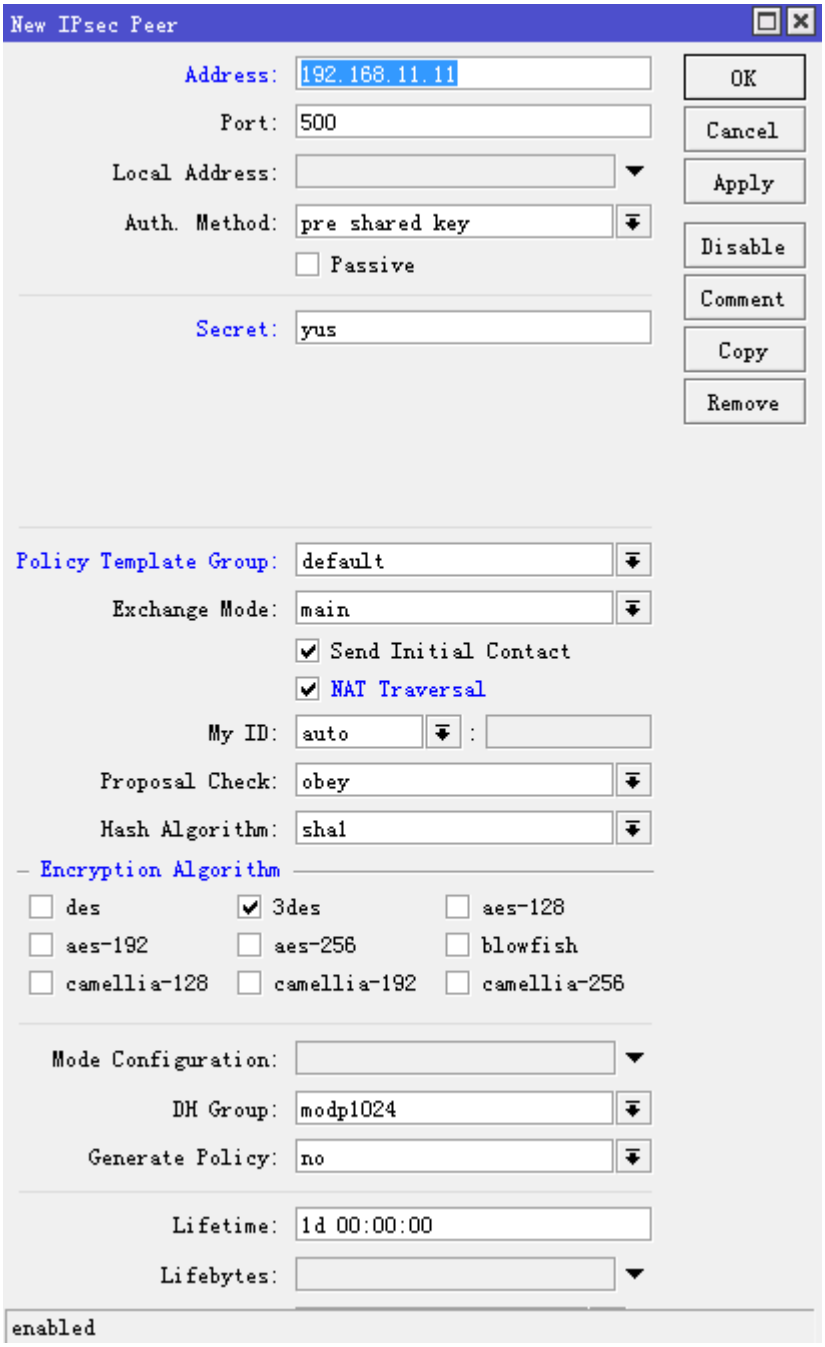
进入 ip ipsec 里的 policies 选项，先在 general 添加内网的源地址和对端网络的内网地址。



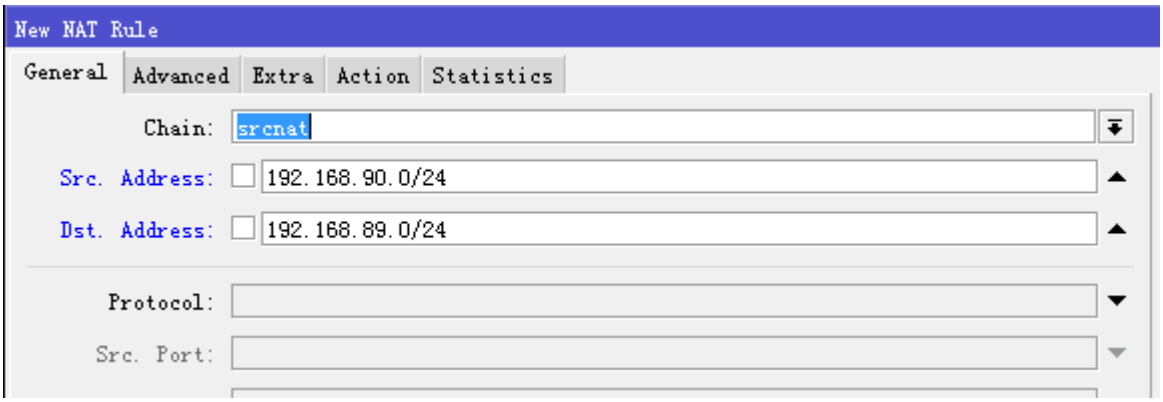
选择 action 选项里面添加源外网地址和对端外网地址和开启 tunnel 隧道协议

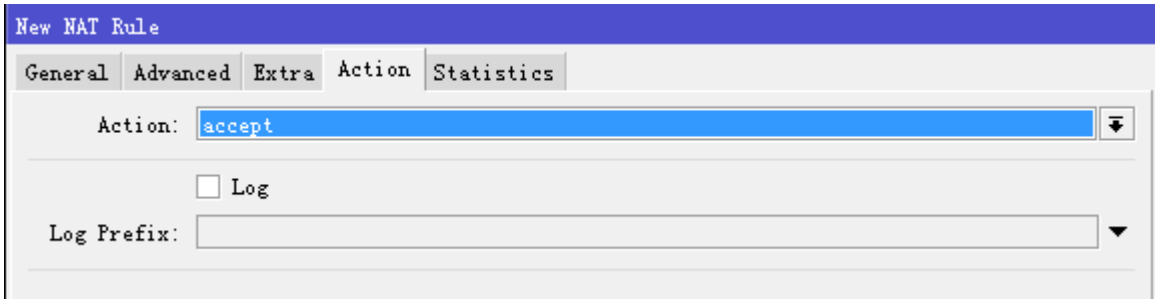


再在 ip ipsec 里的 peers 标签里添加对端外网 ip 地址和 secert 密码：



在 ip firewall 里的 nat，建立一条 srcnat 规则为 accept，接受源内网地址和对端内网地址通过：



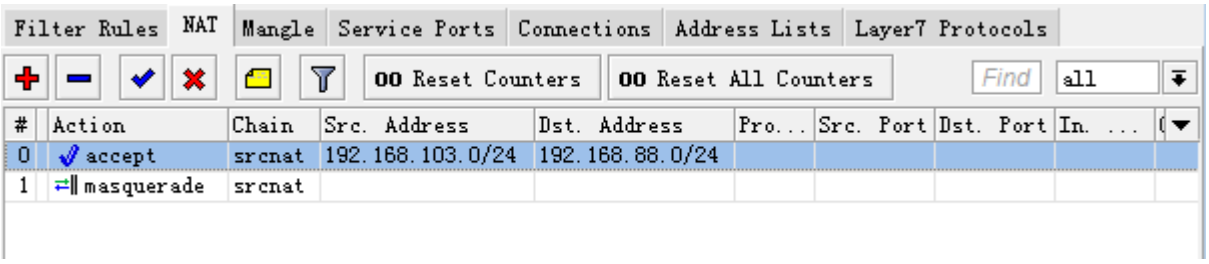


然后建立将本地内网隐藏上网的 masquerade 规则：

```
/ip firewall nat add src-address=192.168.90.0/24 action=masquerade
```

以上就是 R2 的配置过程。

注：NAT 规则的配置的上下顺序，accept 规则需在 masquerade 伪装规则前：



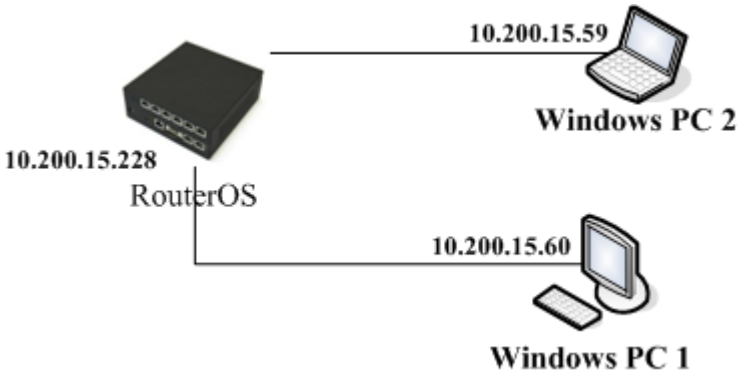
27.2 Windows L2TP/IPsec 连接

Microsoft Windows XP/Vista/win7 内建了 PPTP 客户端和和 L2TP/IPSec 客户端。PPTP 连接是不要 IPsec 加密的，而 windows 的 L2TP/IPsec 默认要求建立 IPsec 连接后，才能进行 L2TP 的拨号连接，这样的解决方法在早期采用的是修改 windows 的注册表，将 windows 默认的 IPsec 连接值修改并关闭，相对于终端客户操作繁琐，且安全性降低。为了能正常让 windows 的 L2TP/IPsec 与 RouterOS 连接，我们可以配置 RouterOS 启用 IPsec。

Windows 建立 L2TP/IPSec 连接，首先要求连接到对端的 IPSes，在 IPSec 建立完成后在允许 L2TP 连接，也就是 IPSec 连接在先，L2TP 其后，所以我们首先配置 IPSec 连接。

注：RouterOS 6.16 在 L2TP 服务配置中加入了 IPsec 选项，简化了 IPsec 的配置。

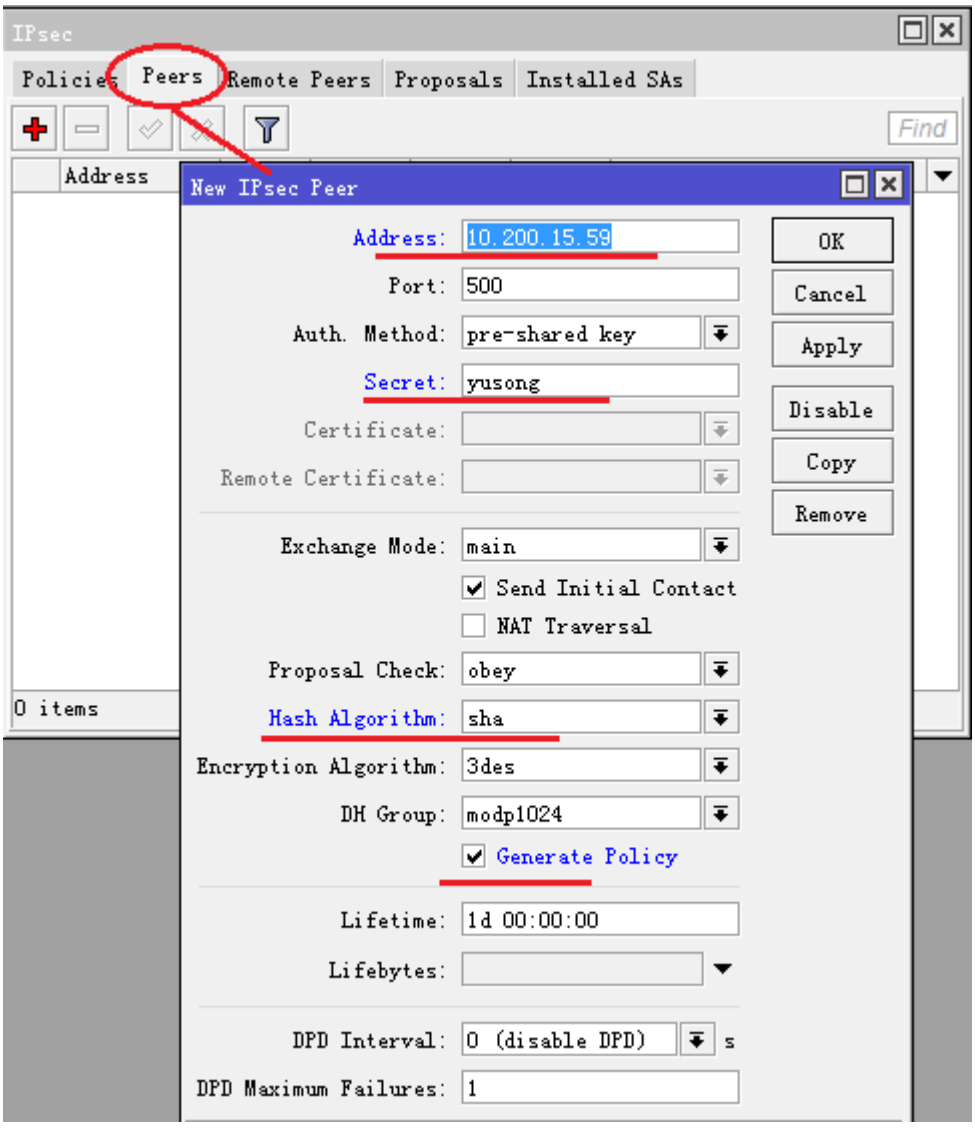
我们先确定一下网络结构：



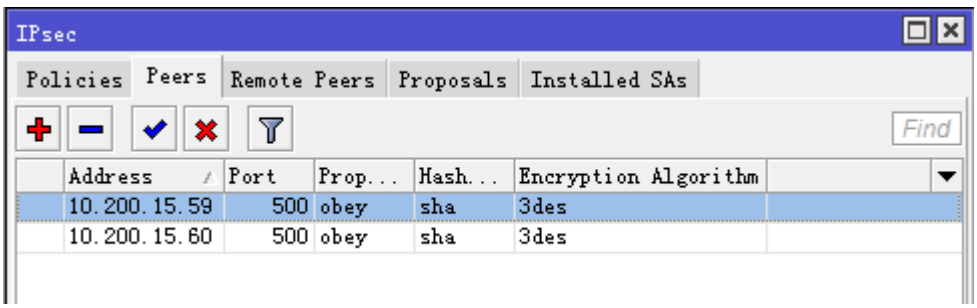
这里 RouterOS 的 IP 地址是 10.200.15.228，两台 PC 的 IP 地址分别是 10.200.15.59,和 10.200.15.60。两台 PC 的 IP 地址必须是固定，以便 IPsec 连接成功。在这个拓扑图里要求所有的地址是能被访问到的，即非 nat 转换的地址（也非 L2TP 隧道分配的 IP 地址）。

IPSec 配置

首先要将 IPsec 指向对端的 windows PC 的 IP 地址（非 L2TP 分配 IP 地址），进入/ip ipsec 菜单下（确定安装 security 功能包），选择 peer 标签，设置 address 为 PC 的 IP 地址，secret 设置共享密钥 yusong，Hash-algorithm 选择 sha，generate-policy 勾上，其他默认。



添加 10.200.15.60 的 peer 规则



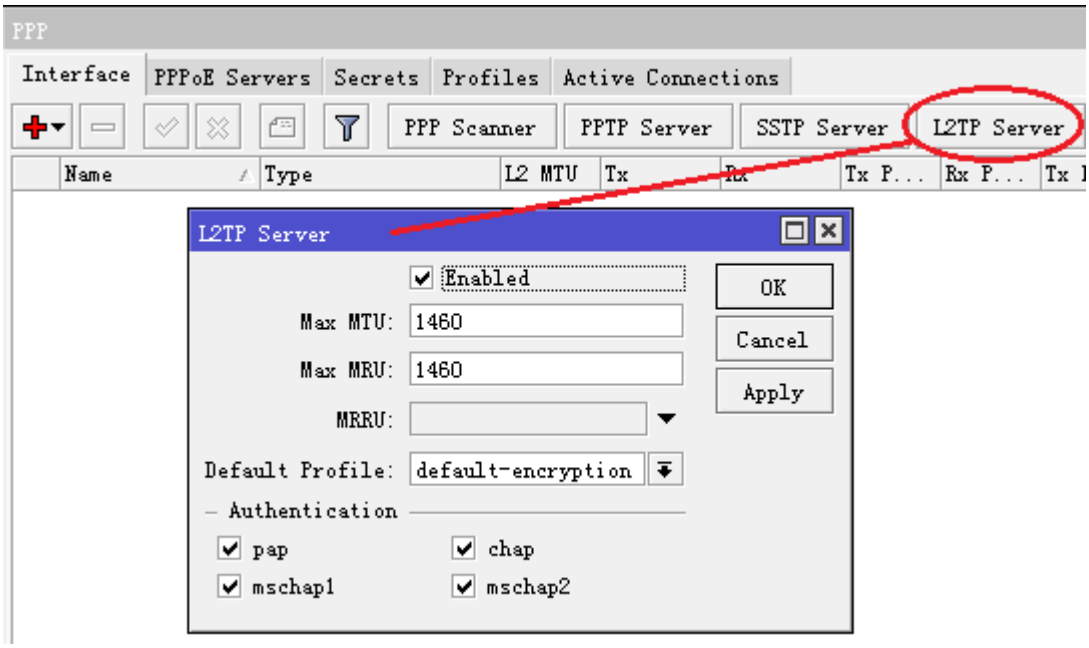
```
/ip ipsec peer add address=10.200.15.59:500 auth-method=pre-shared-key \
   secret=yusong hash-algorithm=sha enc-algorithm=3des generate-policy=yes
/ip ipsec peer add address=10.200.15.60:500 auth-method=pre-shared-key \
   secret=yusong hash-algorithm=sha enc-algorithm=3des generate-policy=yes
```

添加 IPSec peer 设置，

- **address=10.200.15.59** 是你的 windows 电脑的网卡实际地址。
- **:500** 端口号;
- **hash-algorithm=sha** 和 **enc-algorithm=3des** 是 windows 上的默认配置；
- **generate-policy=yes** 自动产生 IPSec 策略

RouterOS 配置

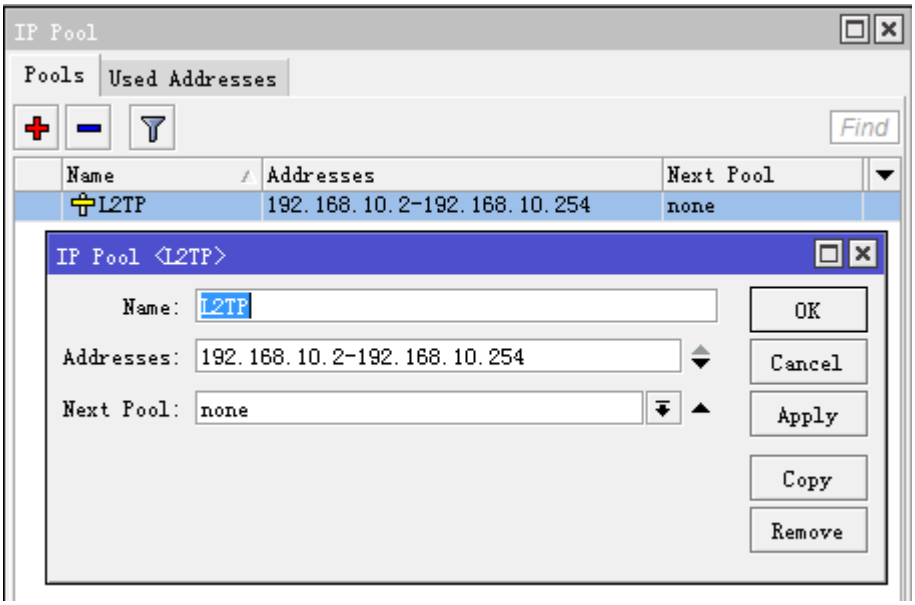
首先我们配置 RouterOS 的 L2TP 服务器，这个配置和普通的 PPTP 配置一样，在 PPP 里启用 L2TP 服务



命令行配置，记住这里的路径不同：

```
/ interface l2tp-server server set enabled=yes
```

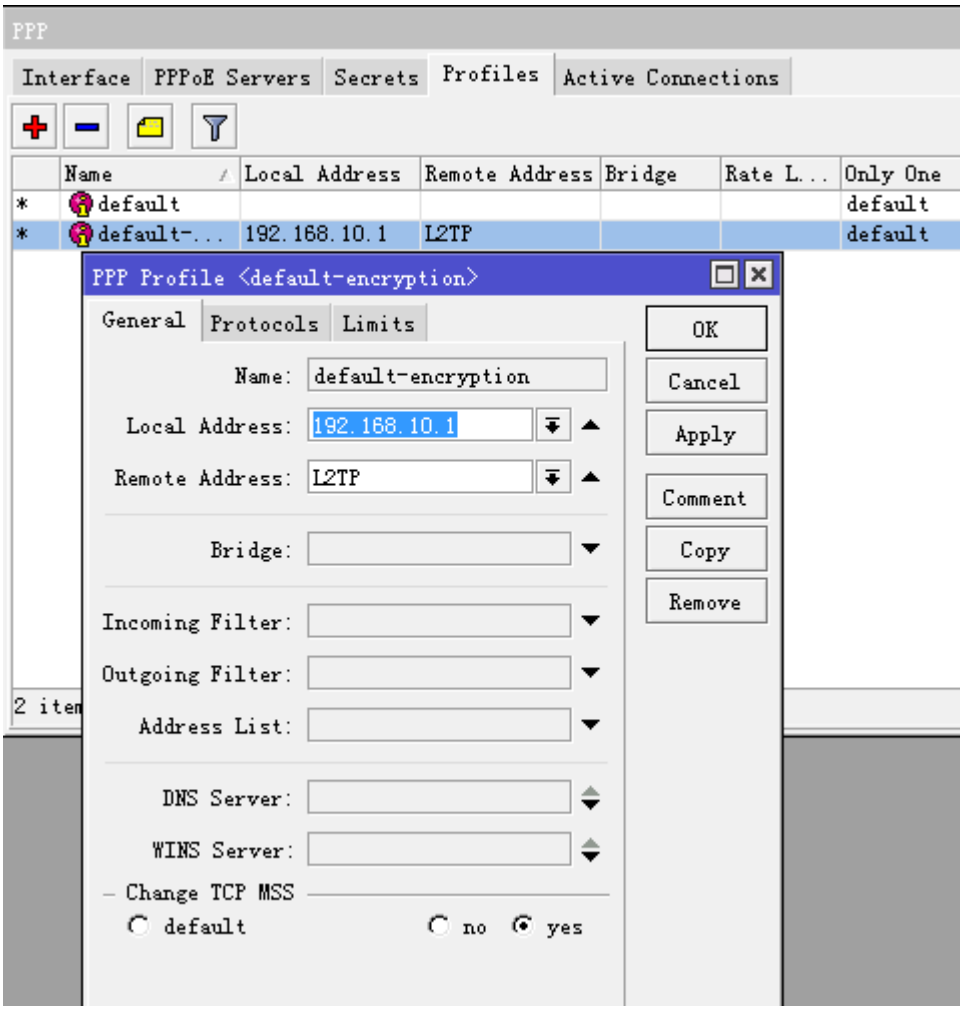
进入 ip pool 设置分配给用户的地址池：



命令操作如下：

```
/ip pool add name=L2TP ranges=192.168.10.2-192.168.10.254
```

进入/ppp profile 配置 default-encryption 的规则:

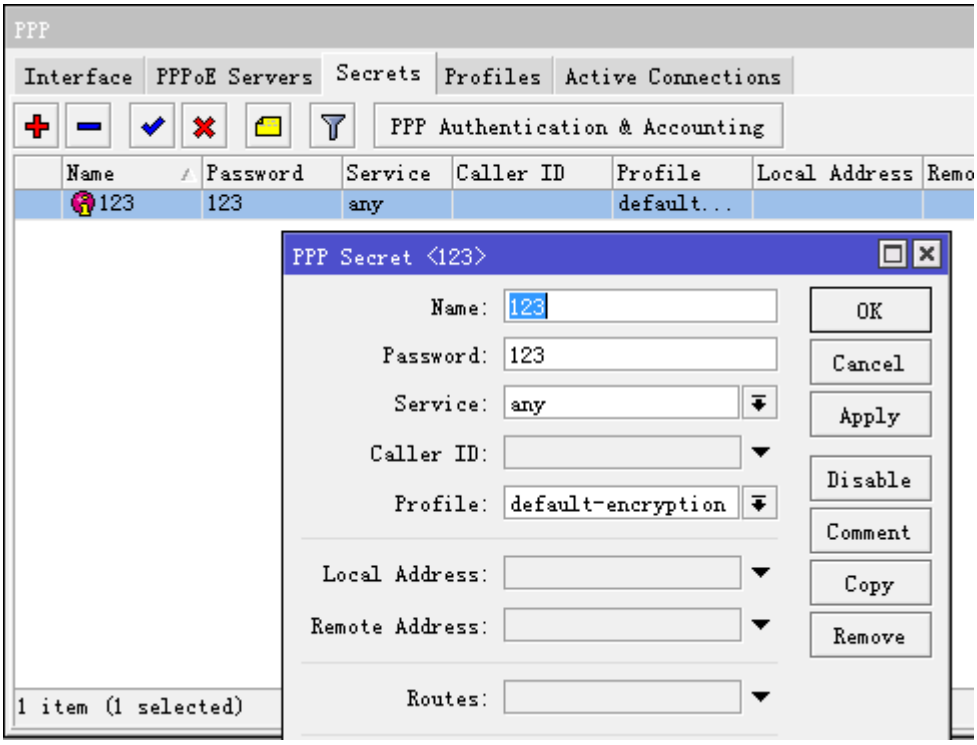


DNS 和 limit 选项里的 rate-limit、only one 参数根据需要设置，这里就不多讲解。

命令行配置

```
/ppp profile> set 1 local-address=192.168.10.1 remote-address=L2TP
```

进入/ppp secret 添加用户账号



命令行配置

```
/ ppp secret add name=123 password=123 profile=default-encryption
```

到这里 L2TP 服务器配置完成。

Windows 配置

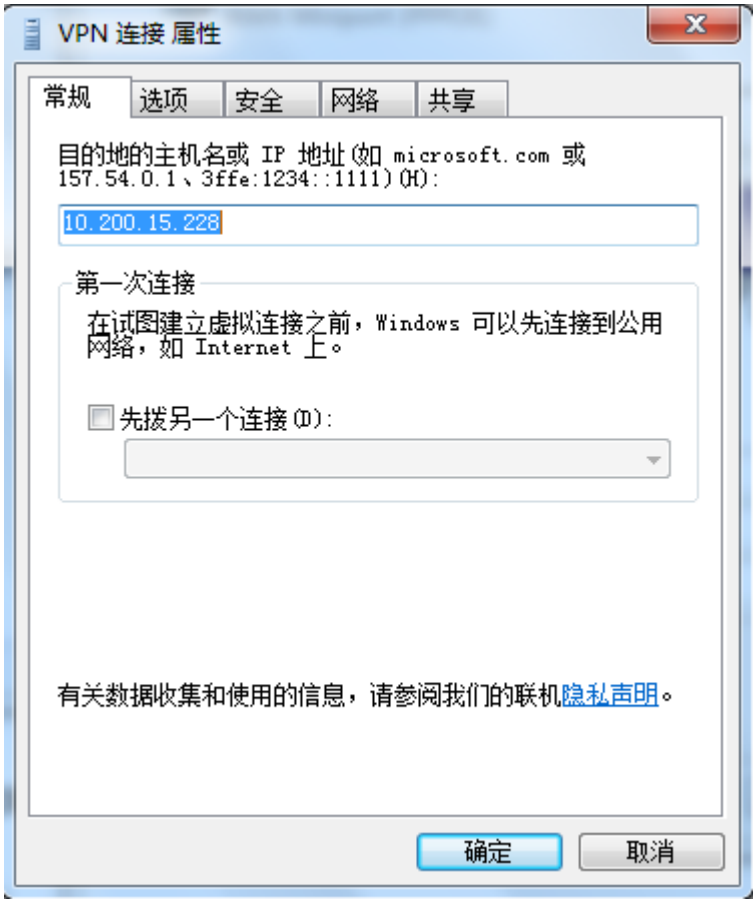
Windows 配置包含 2 个部分，第一个部分添加新的网络连接，第二个部分调整 IPSec 设置

Win7 配置步骤：

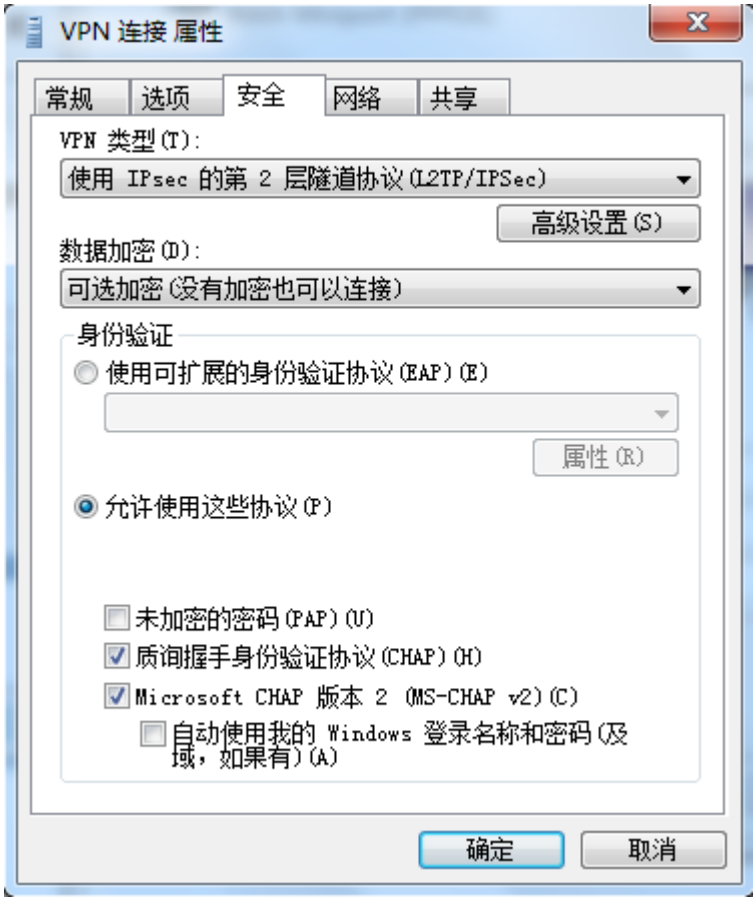
- **点开始菜单**;
- **控制面板\网络和 Internet\网络和共享中心**
- **设置新的连接或网络**;
- **添加一个 VPN 连接，**
- **目的地的主机或域名填写 10.200.15.228（具体操作跟着步骤走，不详细说明）**

接下来我们需要配置 VPN 连接的属性

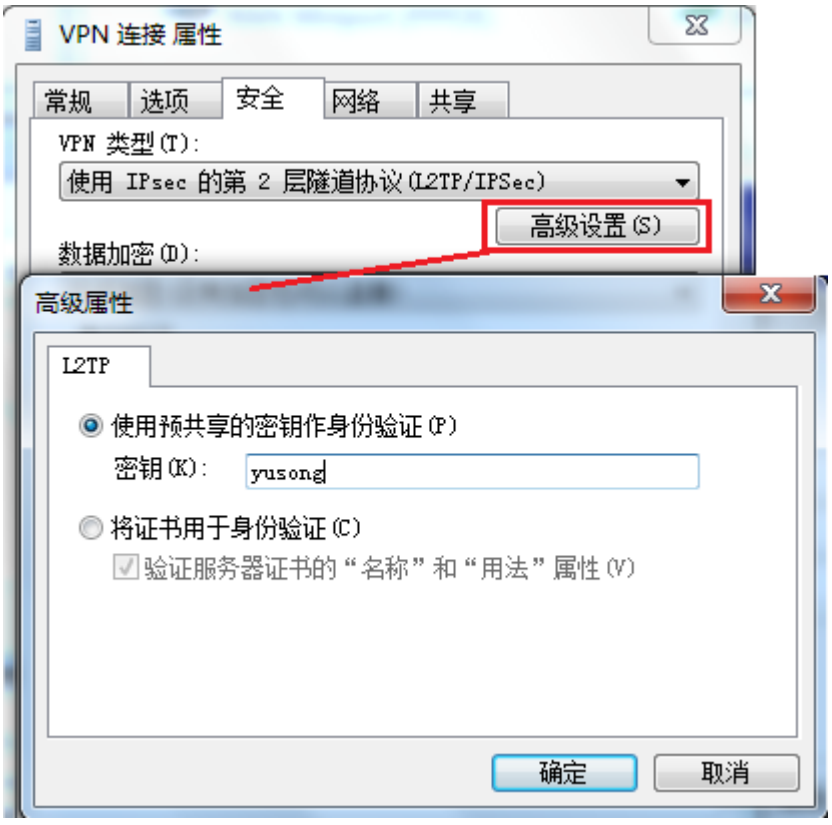
确定主机地址是 10.200.15.228



选择 VPN 类型为使用 ipsec 的第 2 层隧道协议（L2TP/IPSec）

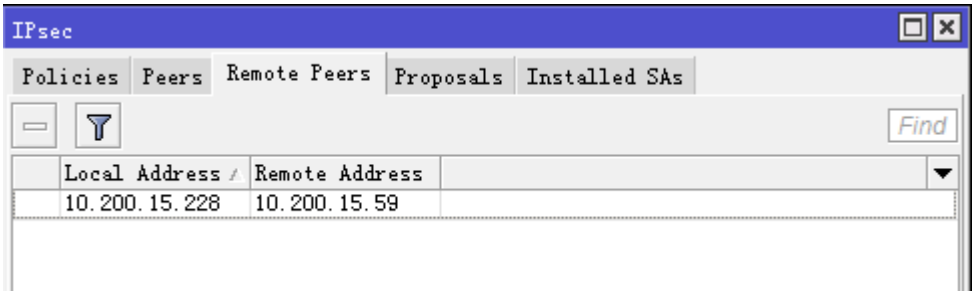


选择高级设置，并设置使用预共享的密钥作身份验证：输入相同的密钥：yusong

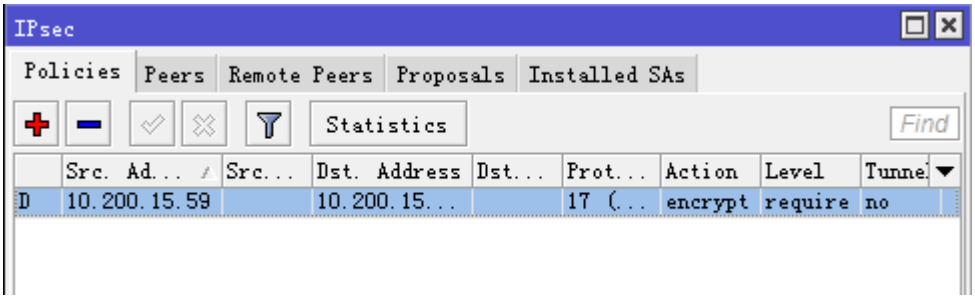


配置完成后，输入账号 123，密码 123，连接

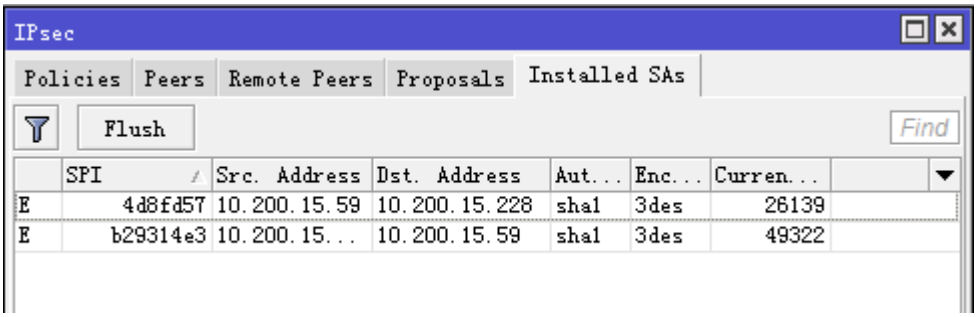
连接完成后，在 remote peers 可以看到连接的 IP 地址



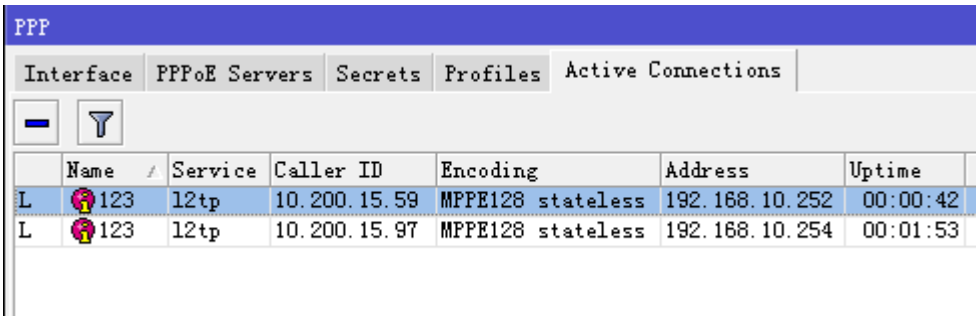
Policies 策略会被自动添加



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，第二次重播可能需要使用 Flush 清空状态



PPP 里的 active 状态



v6.16 后简化 L2TP/IPsec 配置

在 RouterOS 6.16 版本开始，L2TP 服务增加了 use IPsce 选项，可以直接在 L2TP 服务配置菜单下设置 IPsec，共享密钥设置后，会自动添加到 IPsec 配置中，简化了管理员操作，当然 RouterOS 必须同时安装 PPP 和 Security 功能包。

第二十八章 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 的自由协议。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。

有 EoIP 接口的网络设置：

- 可以在因特网上桥接 LAN
- 可以在加密的隧道桥接 LAN
- 可以在 802.11b 'ad-hoc'无线网络上桥接 LAN

快速设置向导

在 IP 地址为 **10.5.8.1** 和 **10.1.0.1** 的两个路由器之间做 EoIP 隧道：

1. 在 IP 地址为 **10.5.8.1** 的路由器上添加一个 EoIP 接口并设置它的 MAC 地址：

```
/interface eoip add remote-address=10.1.0.1 tunnel-id=1 mac-address=00-00-5E-80-00-01   disabled=no
```

2. 在 IP 地址为 **10.1.0.1** 的路由器上添加一个 EoIP 接口并设置它的 MAC 地址：

```
/interface eoip add remote-address=10.5.8.1 tunnel-id=1 mac-address=00-00-5E-80-00-02   disabled=no
```

现在你可以从同一子网添加 IP 地址以创建 EoIP 接口。

规格

功能包要求: ***system***
等级要求: *Level3*
操作路径: ***/interface eoip***

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）数据包中，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：WDS 在很大程度上比 EoIP 快（最多达可达到 10-20%，在 RouterBOARD 500 系统上），所以推荐在可能时使用 WDS。

28.1 EoIP 配置

操作路径: /interface eoip
arp (disabled | enabled | proxy-arp | reply-only; default: **enabled**) -地址解析协议
mac-address (*MAC 地址*) - EoIP 接口的 MAC 地址。你可以自由的使用从 **00-00-5E-80-00-00** 到 **00-00-5E-FF-FF-FF** 范围的 MAC 地址
mtu (*整型*; default: **1500**) -最大传输单元。默认值提供了最大的兼容性
name (*名称*; 默认: **eoip-tunnelN**) – 隧道接口名
remote-address - EoIP 隧道 IP 地址的另一端——必须是 MikroTik 路由器
tunnel-id (*整型*) – 隧道身份值

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（它允许类似以太网络的透明桥接，因此有可能在隧道上传输满长度的以太网帧）。

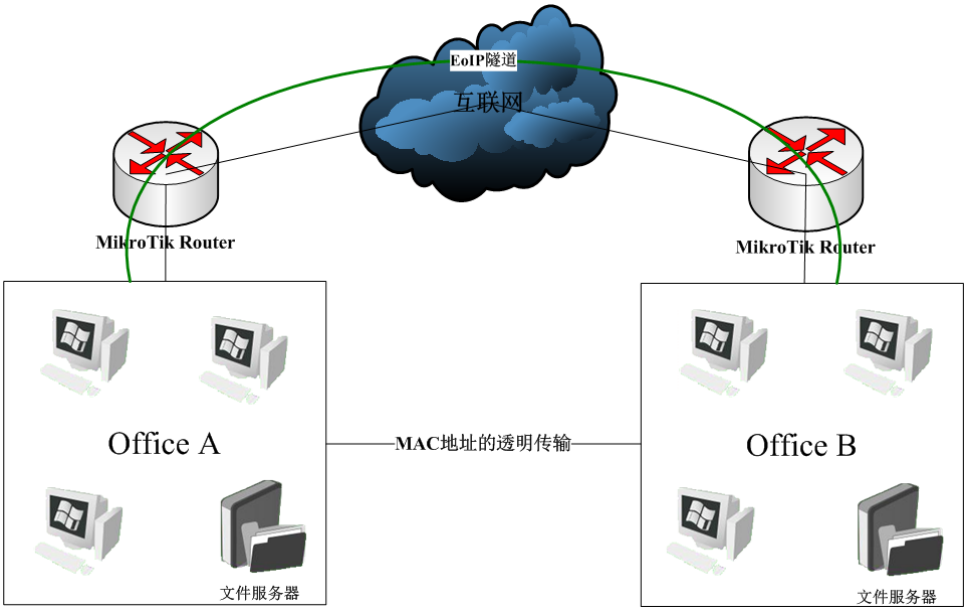
当桥接 EoIP 隧道时，推荐对每个隧道设置唯一的 MAC 地址以使桥接算法正常工作。对于 EoIP 接口你可以使用从 **00-00-5E-80-00-00** 到 **00-00-5E-FF-FF-FF** 范围的 MAC 地址，IANA 就是为这些情况保留的。或者，你可以设置第一字节的第二位来标记地址为由网络管理员指定的本地管理的地址，并使用任何 MAC 地址，你只需要确定它们在连接到一个桥的主机之间是唯一的。

添加并启用名为 **to_mt2** 连接到 **10.5.8.1** 路由器的 EoIP 隧道，指定 **tunnel-id** 为 **1**：

```
[admin@MikroTik] interface eoip> add name=to_mt2 remote-address=10.5.8.1 \
\... tunnel-id 1
[admin@MikroTik] interface eoip> print
Flags: X - disabled, R - running
  0 X   name="to_mt2" mtu=1500 arp=enabled remote-address=10.5.8.1 tunnel-id=1

[admin@MikroTik] interface eoip> enable 0
[admin@MikroTik] interface eoip> print
Flags: X - disabled, R - running
  0 R   name="to_mt2" mtu=1500 arp=enabled remote-address=10.5.8.1 tunnel-id=1

[admin@MikroTik] interface eoip>
```

28.2 EoIP 应用实例

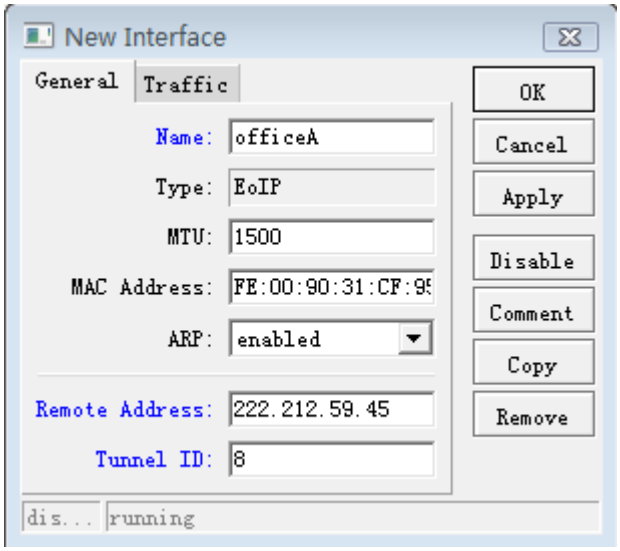
这里我们假设有两个异地的办公点，officeA 和 officeB，我们通过 Eoip 隧道将他们连接起来，建立 2 层的安全隧道通信，如下图：



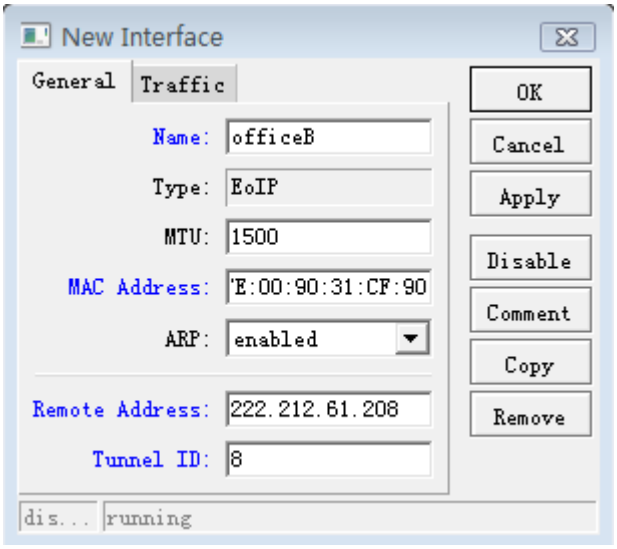
网络参数：

- OfficeA 的 IP 地址为 222.212.61.208，桥接分配地址 10.0.0.1
- OfficeB 的 IP 地址是 222.212.59.45，桥接分配地址 10.0.0.2

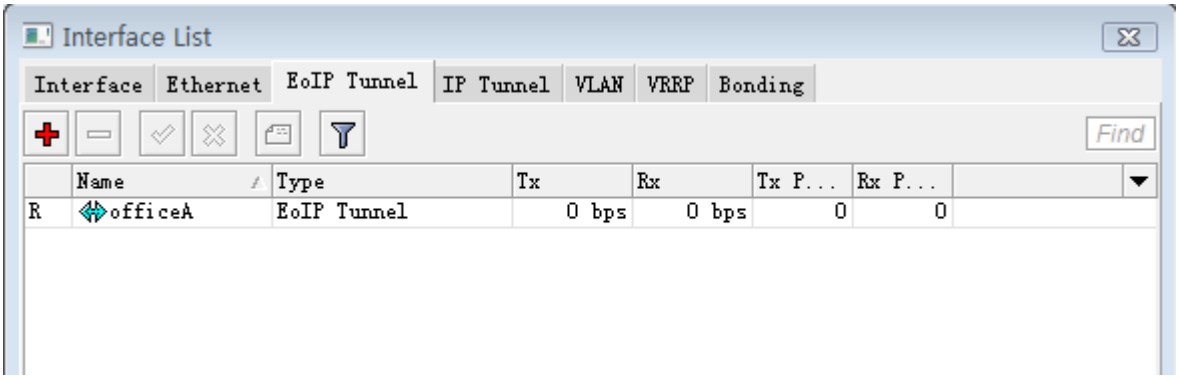
1、这里我们将配置两个 Eoip 隧道的 Tunnel ID 为 8，首先在 Interface 里建立 Eoip 隧道



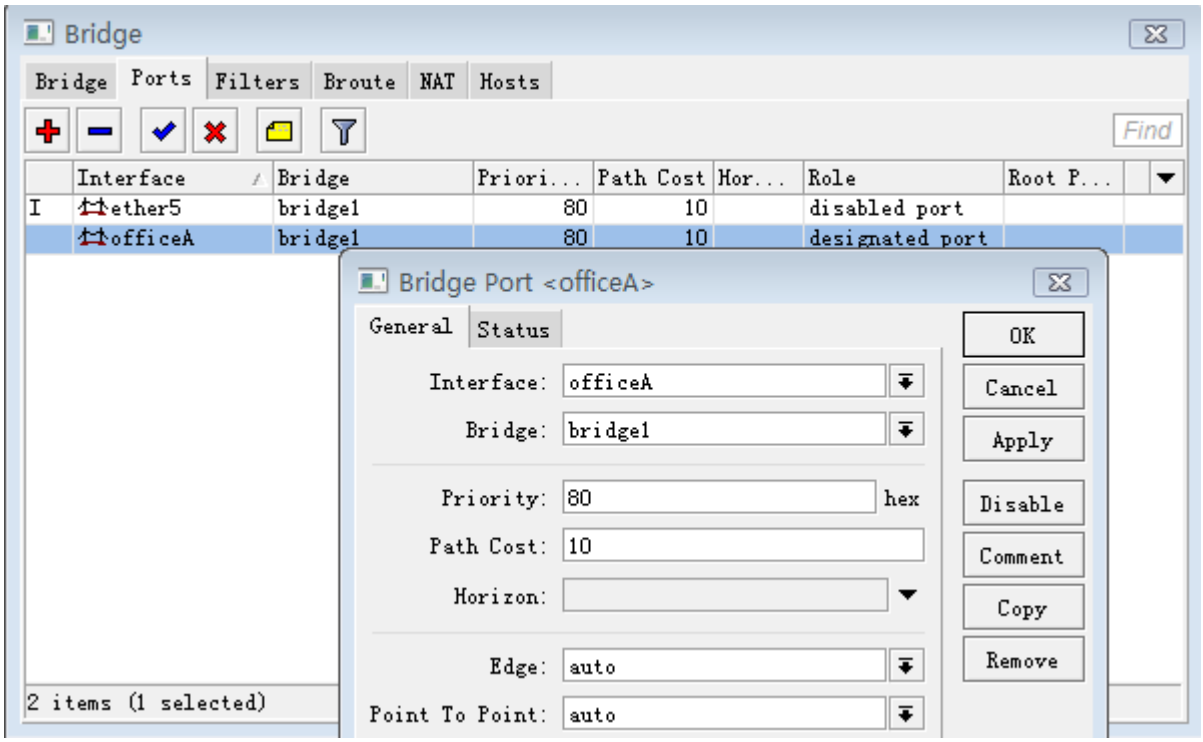
建立 officeB 的 Eoip 隧道：



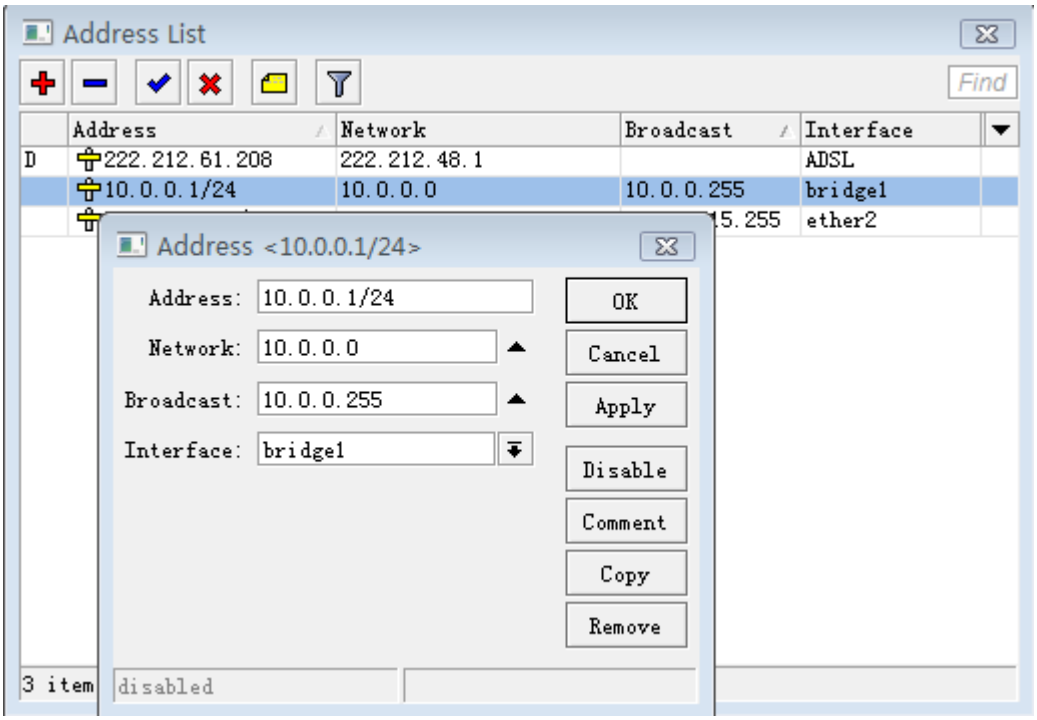
2、接下来，OfficeA 和 OfficeB 的配置基本相同，现在我们来看 officeA 配置，在 interface 中的 Eoip Tunnel 中可以看到 Eoip 隧道连接成功：



进入bridge，并在bridge 中添加一个bridge1的桥，然后并在port中将ether5网卡和建立Eoip隧道的officeA绑定到 Bridge1 中：



3、绑定完桥后，进入 ip address，设置桥 IP 地址为 10.0.0.1/24，同样在 OfficeB 的路由器则设置为 10.0.0.2/24：



注： 在做 NAT 规则的时候，特别是伪装，需要指明伪装的端口，如果默然伪装，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！

第二十九章 GRE 和 IPIP 隧道协议

29.1 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，即远端隧道中断，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）

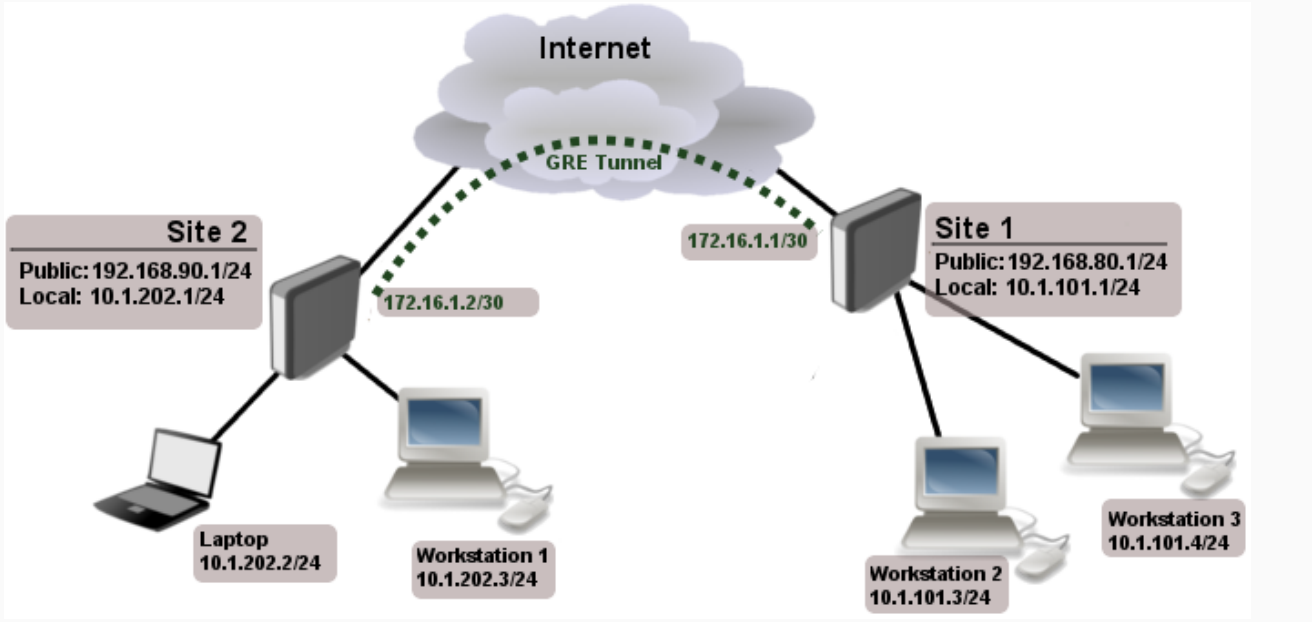
注意：GRE 隧道能转发 IP 和 IPv6 数据包（ethernet 800 和 86dd 类型）

属性

属性	描述
arp (*disabled* | *enabled* | *proxy-arp* | *reply-only*; 默认:)	地址解析协议
comment (*字符*; 默认:)	隧道属性描述。
disabled (*yes* | *no*; 默认: **no**)	禁用或启用隧道。
dscp (*继承* | *整型 [0-63]*; 默认:)	从 v5.6 开始，可设置 dscp 值到 GRE 包头，或从本地 dscp 继承到隧道传输中。
keepalive (*整型 [1..4294967295]*; 默认:)	隧道存活时间，单位秒，默认下存活时间被禁用。
l2mtu (*整型 [0..65536]*; 默认: **65535**)	二层网络最大传输单元。
local-address (*IP*; 默认: **0.0.0.0**)	用于隧道本地连接的 IP 地址，如果设置为 0.0.0.0，将会使用到远程终端出路由接口的 IP 地址。
mtu (*整型 [0..65536]*; 默认: **1476**)	三层网络最大传输单元。
name (*字符*; 默认:)	隧道名称
remote-address (*IP*; 默认:)	远端隧道连接 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站点，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

29.2 IPIP 隧道

操作路径: /interface ipip

IPIP 隧道一个简单隧道协议，将 IP 数据包封装到 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

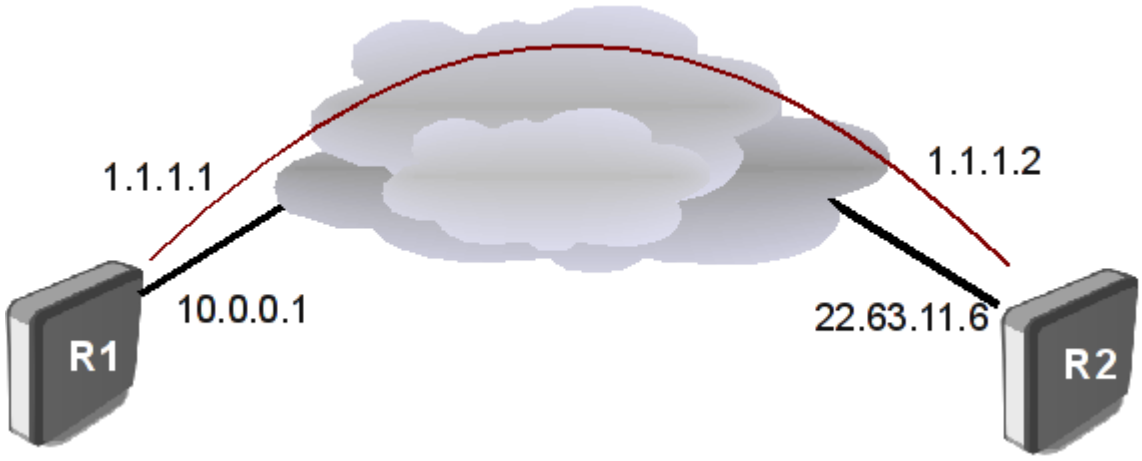
属性	描述
dscp (*继承* | *整型 [0-63]*; 默认:)	设置 dscp 值到 IPIP 包头，或从本地 dscp 继承到隧道传输中。
local-address (*IP*; 默认:)	用于隧道本地连接的 IP 地址
mtu (*整型*; 默认: **1500**)	三层网络最大传输单元
name (*字符*; 默认:)	IPIP 隧道名称
remote-address (*IP*; 默认:)	远端 IPIP 隧道路由器的 IP 地址

注意：IPIP 隧道没有 keepalive 值可以使用，因此在配置后不管远端是否能连接默认为运行状态，这点需要使用者特别注意。

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。

第三十章 OSPF

30.1 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包 LSA（Link State Advertisement）传送给在某一区域内的所有路由器，这一点与距离矢量路由协议不同。运行距离矢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矢量路由协议 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）

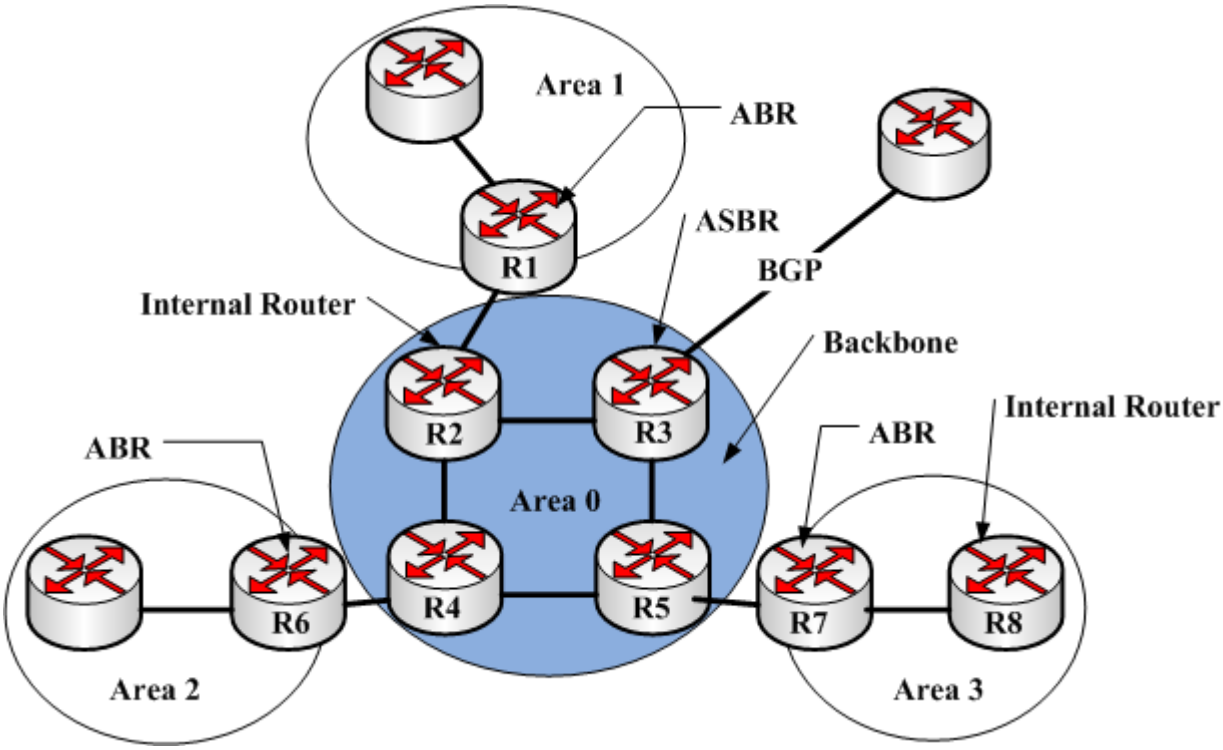
OSPF 术语

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.

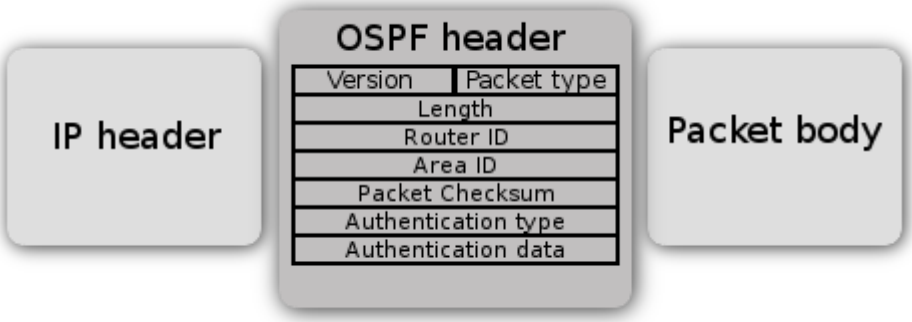
- **Type2:Network LSA**：在每个多路访问网络中，DR 都会产生这种 Network LSA，它只在产生这条 Network LSA 的区域泛洪描述了所有和它相连的路由器（包括 DR 本身）。
- **Type3:Network Summary LSA**：由 ABR 路由器始发，用于通告该区域外部的目的地址.当其他的路由器收到来自 ABR 的 Network Summary LSA 以后，它不会运行 SPF 算法，它只简单的加上到达那个 ABR 的开销和 Network Summary LSA 中包含的开销,通过 ABR，到达目标地址的路由和开销一起被加进路由表里，这种依赖中间路由器来确定到达目标地址的完全路由（full route）实际上是距离矢量路由协议的行为。
- **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 数据包都包含标准的 24byte 包头



字段	描述

Packet type	这里有几种 OSPF 数据包类型：Hello 数据包，Database Description (DD)数据包，Link state request 数据包，link State Update 数据包和 Link State Acknowledgment 数据包。
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 路由器关联的数据包到指定的 OSPF 区域
Checksum	校验数据包在传输中是否受损，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包内容没有被修改，且数据包确定来至正确的 Router ID 路由器

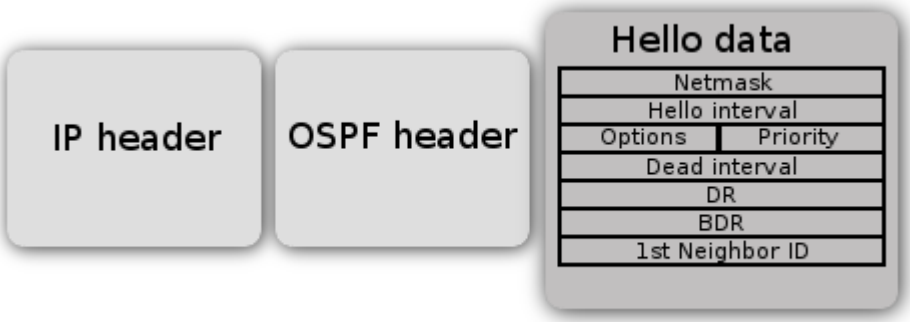
这里有 5 种不同的 OSPF 数据包类型用于确认 LSA 在 OSPF 网络中泛洪。

- **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 信息列表

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 数据包中；
- 外部路由和 NSSA 标识应在相同 Hello 数据包中。

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）数据包中。路由器每次发送 DD 数据包都需要等前一数据包是否确认达到，确认后在发生。 当所有 DD 数据包被收到，路由器需要更新 LSAs，这时会发送 Link-State Request (LSR) 数据包更新 LSAs，邻居以泛洪的方式通过 Link-state Upade（LSU）数据包回应 LSAs，当所有更新收到，邻居会收到 Link-State Acknowledgment (LSack)数据包，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 数据包，会发生一个确认数据包给发生者， 重新打包 LSA 到 LSU 数据包，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链接之间关联，如果转发等，也包含了每个链接的成本（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（数据包到目标地是一条路径，但回应则是不同路径），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：

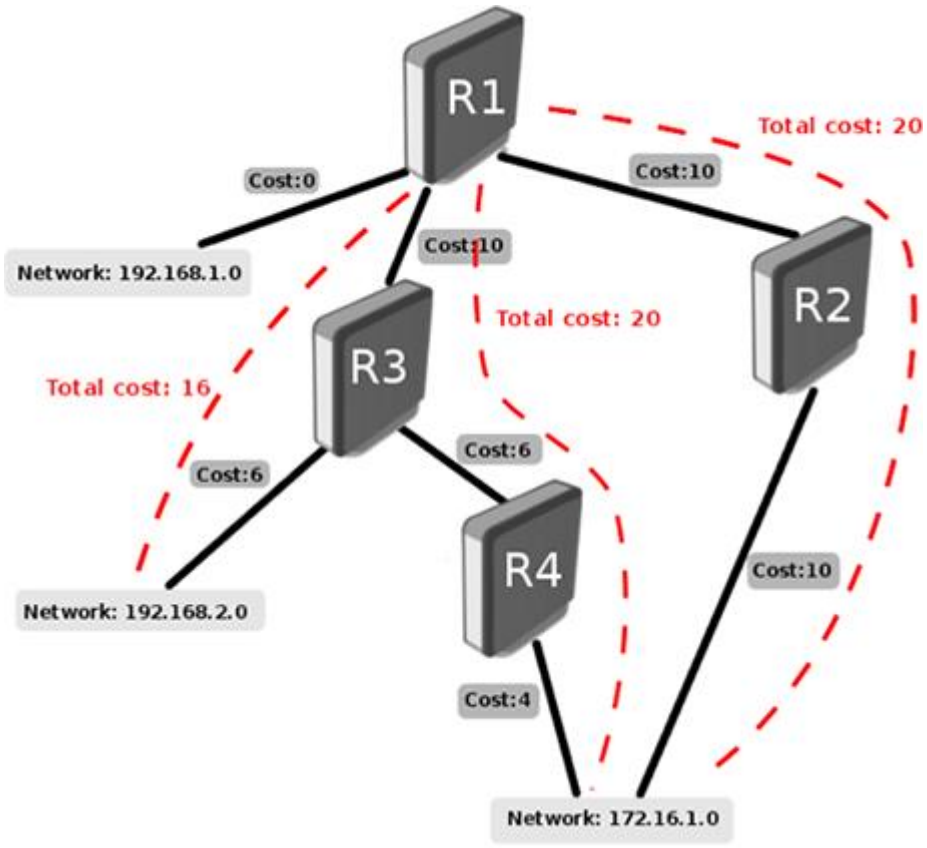
Cost = 100000000/带宽（bps）

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

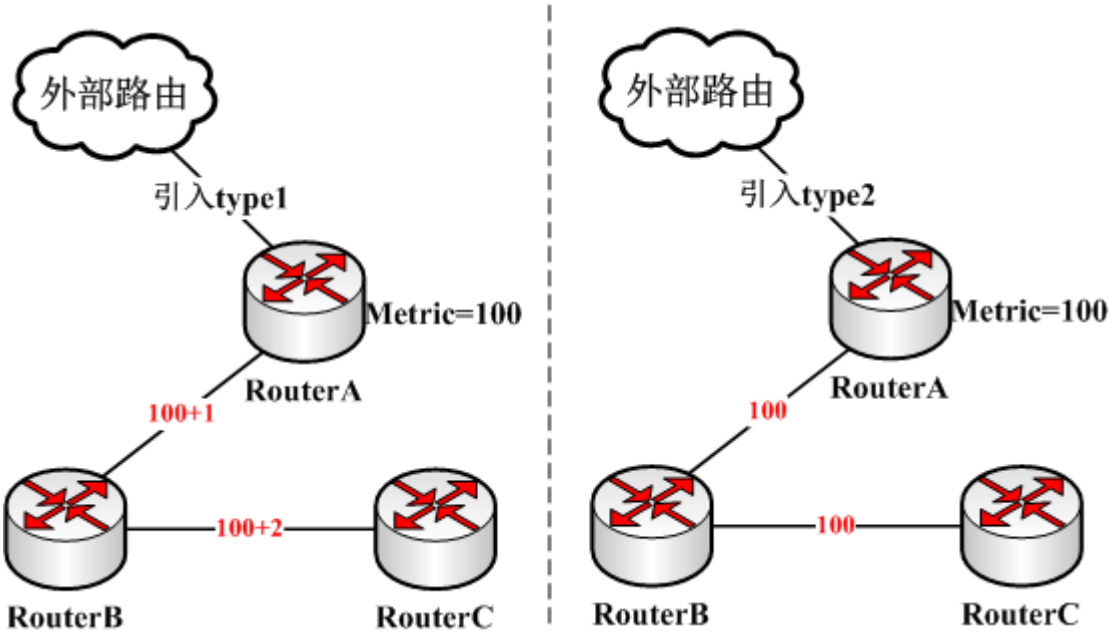
OSPF 路由类型

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

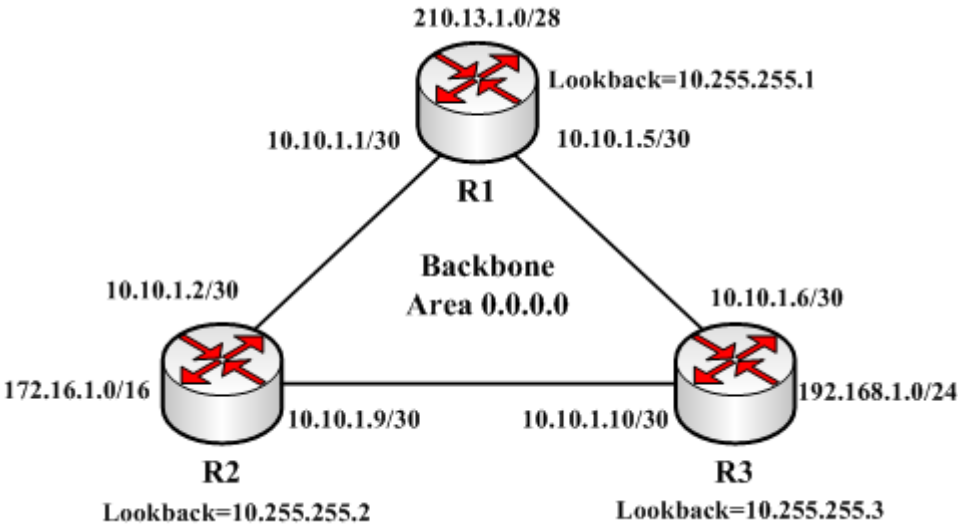
在 OSPF 选路时，去往同一目标网段的 OSPF 路由，开销相同的情况下，type1 的要优于 type2 的开销

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。



30.2 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

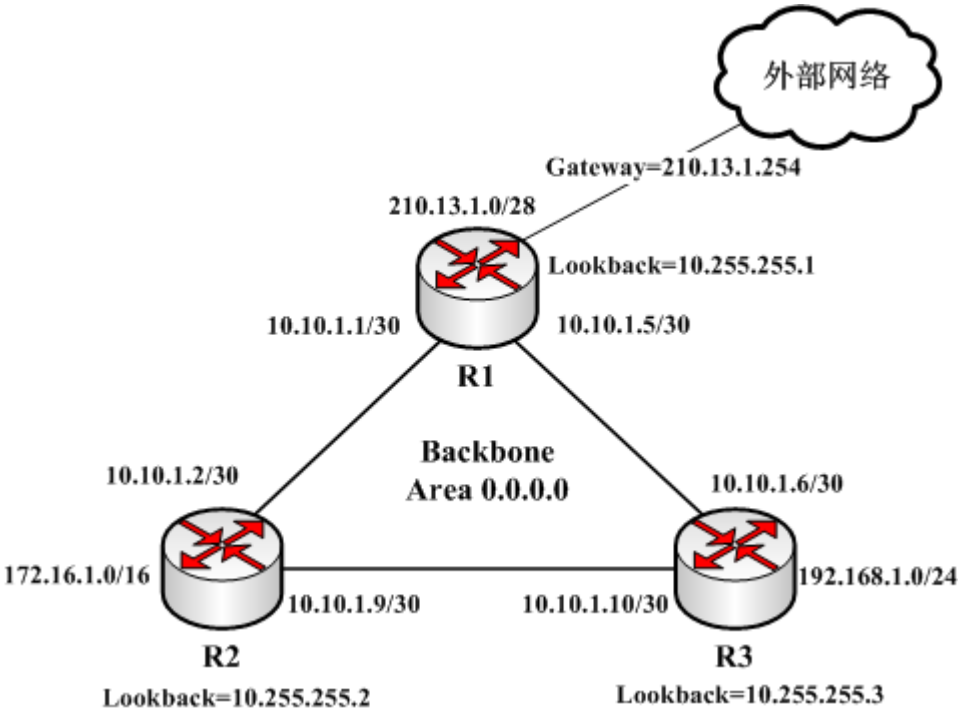
```
[admin@MikroTikR1] /routing ospf neighbor> print
```

- 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
```

30.3 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，然后是在是其他区域网络。

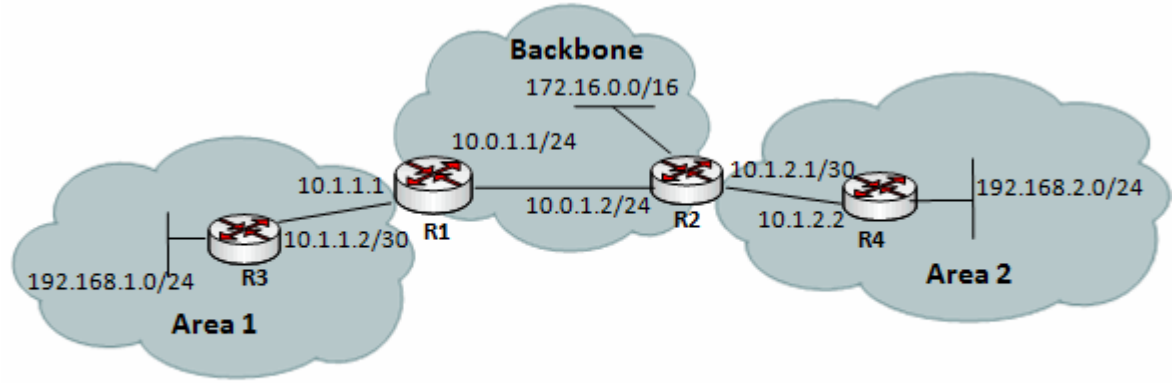


Figure 6.7. Example of multi- area OSPF network

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

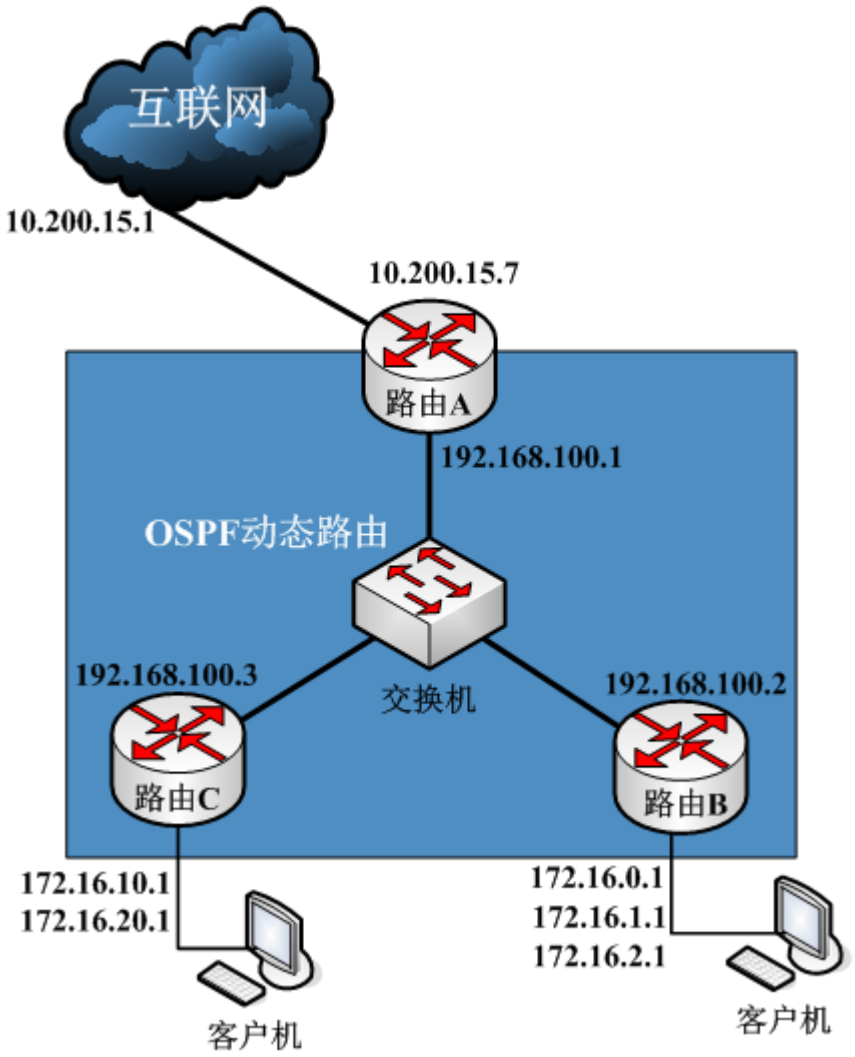
```
[admin@R3] > ip route print
Flags: X - disabled, A - active, D - dynamic,
C - connect, S - static, r - rip, b - bgp, o - ospf, m - mme,
B - blackhole, U - unreachable, P - prohibit
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 1 ADo  10.0.1.0/24                        10.1.1.1           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
```

30.4 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，



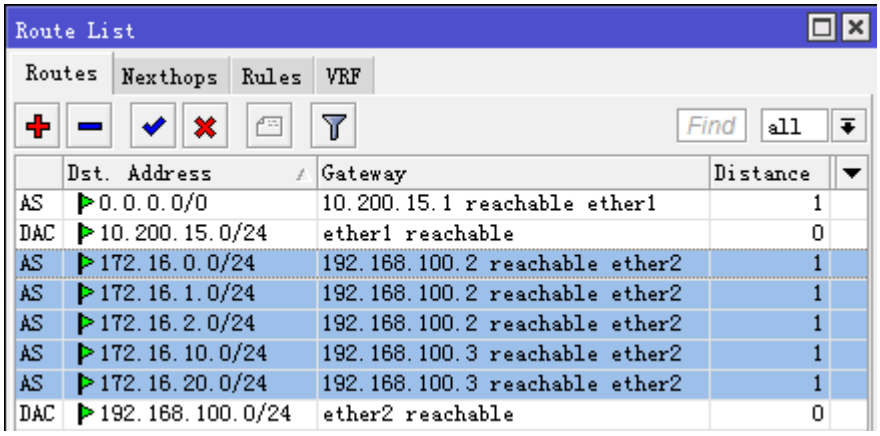
路由器 A 为接入路由器连接互联网，负责互联网接入、NAT 转发和防火墙等，采用静态路由与外网连接。

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：

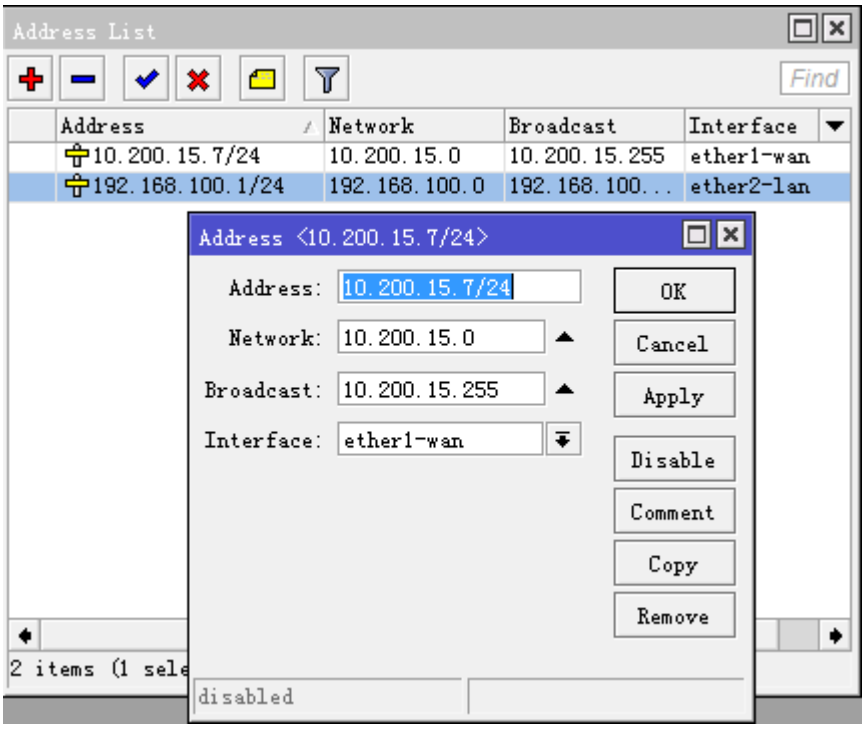


在路由 B 和 C 也要添加相应的路由，如果 IP 地址段有所变动都要修改静态路由表，而且路径也是被限制到一个出口上

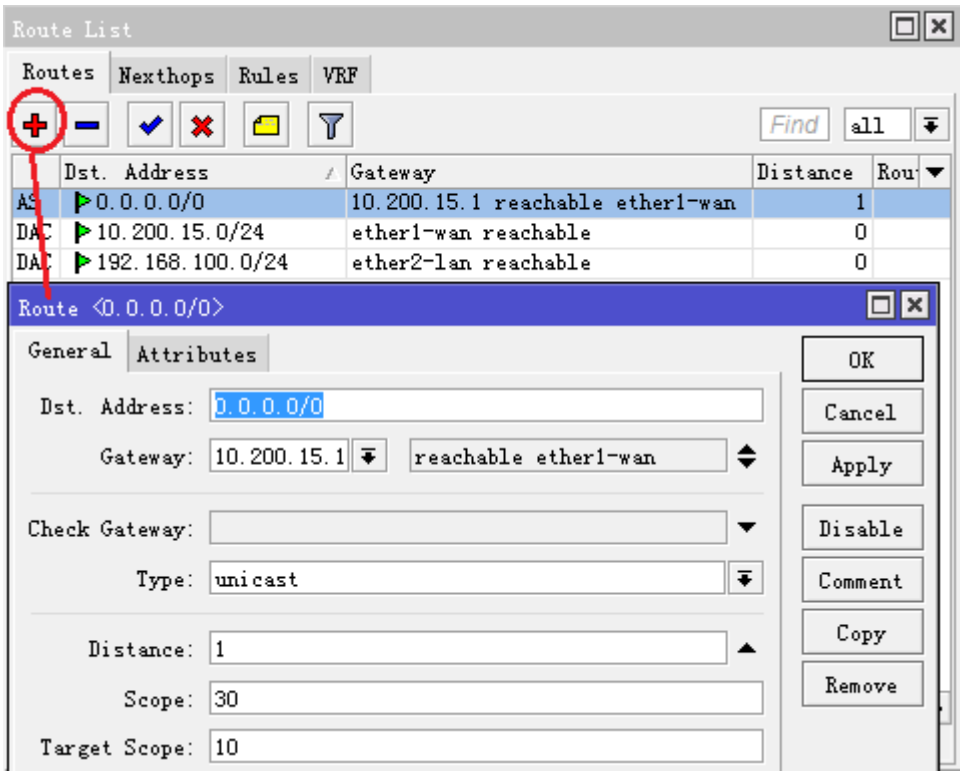
OSPF 配置

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

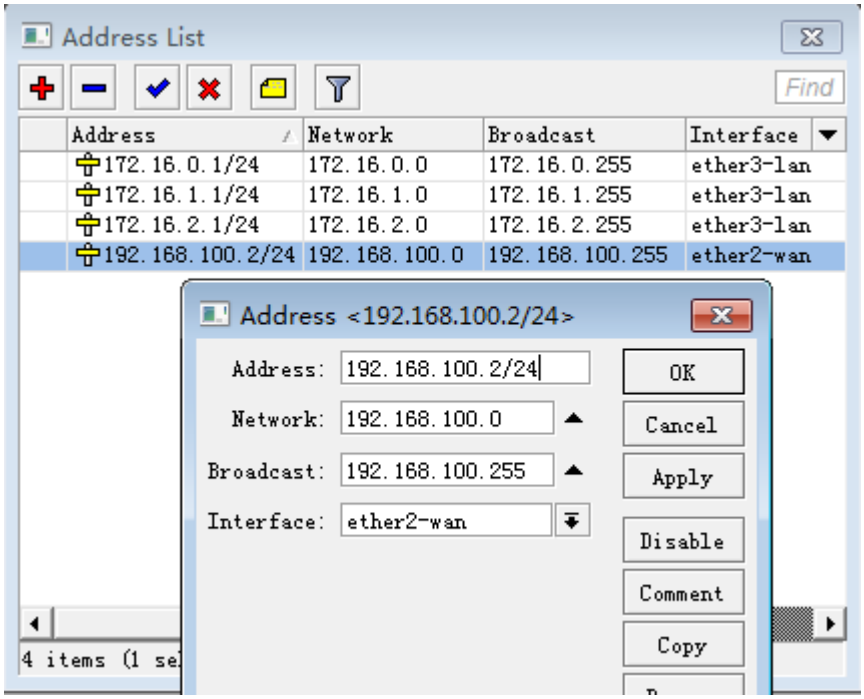
配置首先在路由器 A 配置 IP 地址和互联网的默认网关



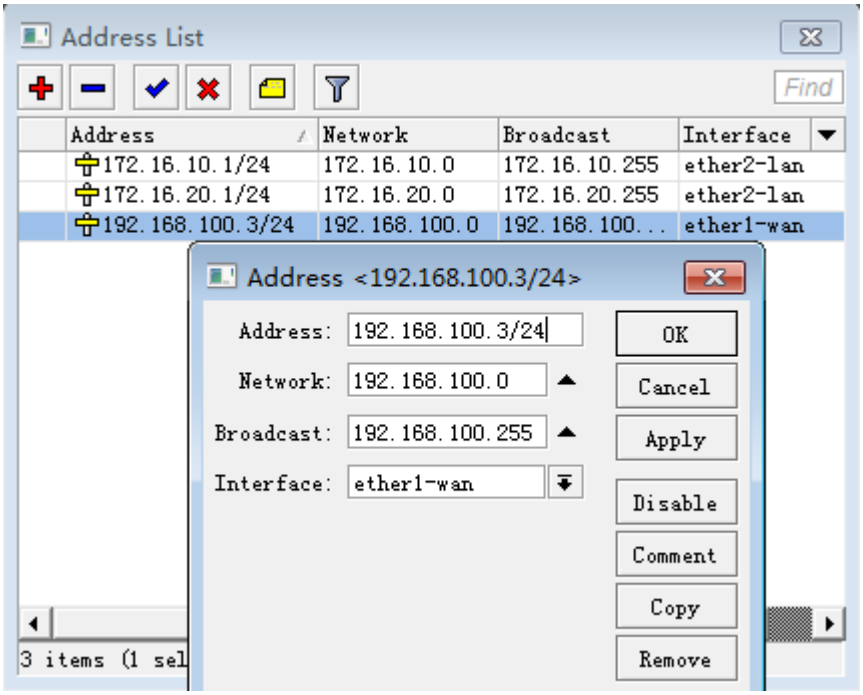
添加路由器 A 的互联网网关



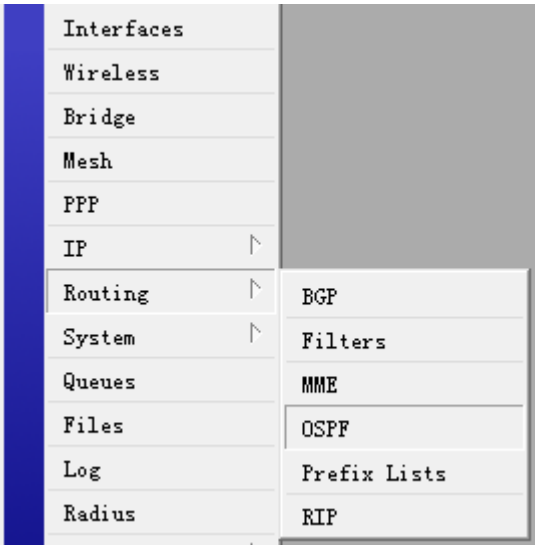
配置路由器 B 的 wan 口 IP 地址



配置路由 C 的 wan 口 IP 地址



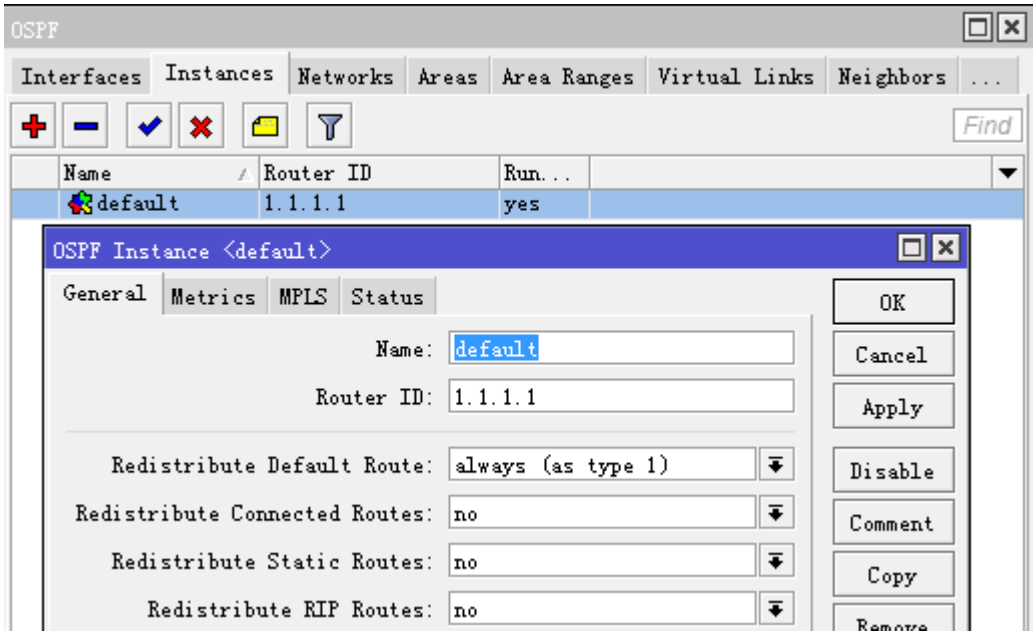
我们将 IP 地址都配置完成，接下来是我们要配置 OSPF 动态路由，我们采用的 RouterOS 版本分别是 5.0rc1 和 3.30，配置 OSPF 需要安装 routing 的功能包，在 winbox 的左边菜单可以看到 Routing 选项，选择 OSPF



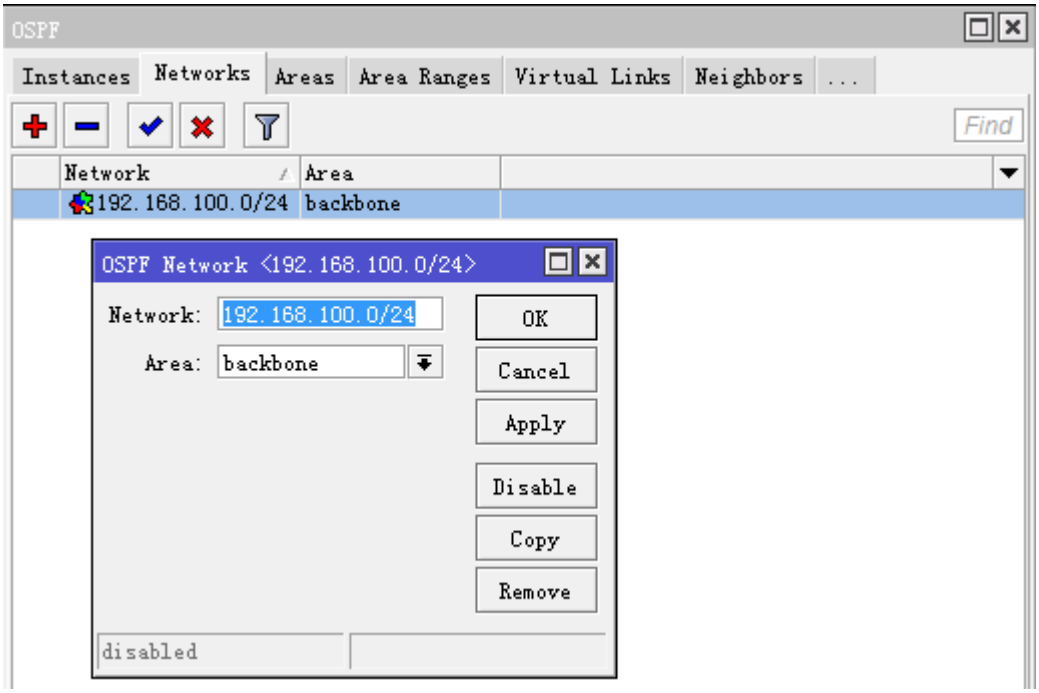
打开 OSPF 后，我们需要在 network 里添加本地路由的网段，申明自己的网络地址范围

路由器 A，

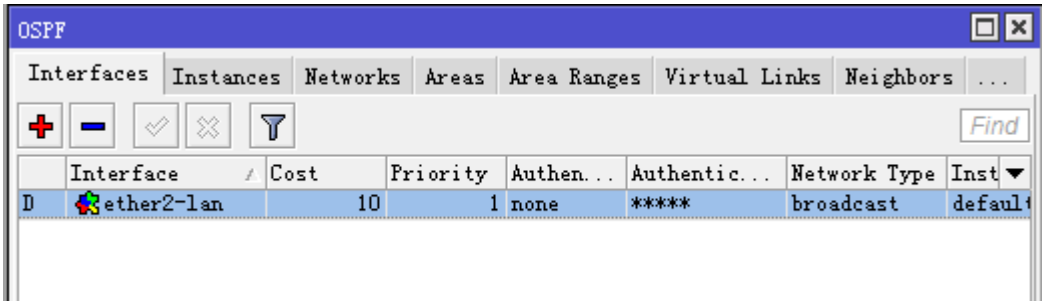
我们可以设置一个路由器的 ID 为 1.1.1.1，如果不设置也可以，OSPF 会自动选择本地接口的 IP 地址为路由器 ID，其他参数默认



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

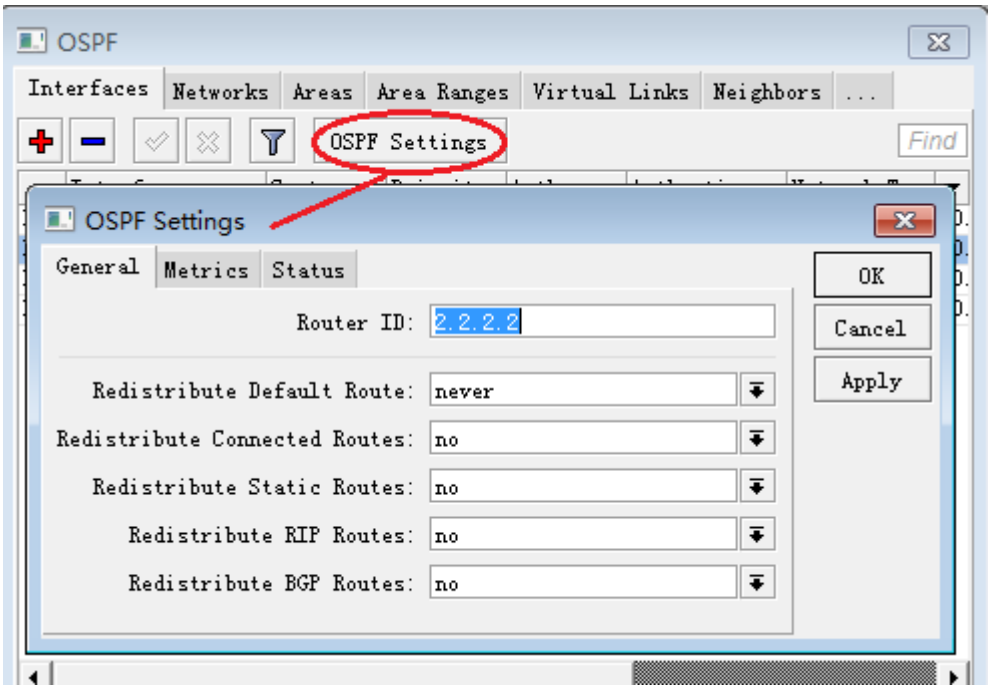


设置完成后，我们可以在 interface 里看到动态添加的接口

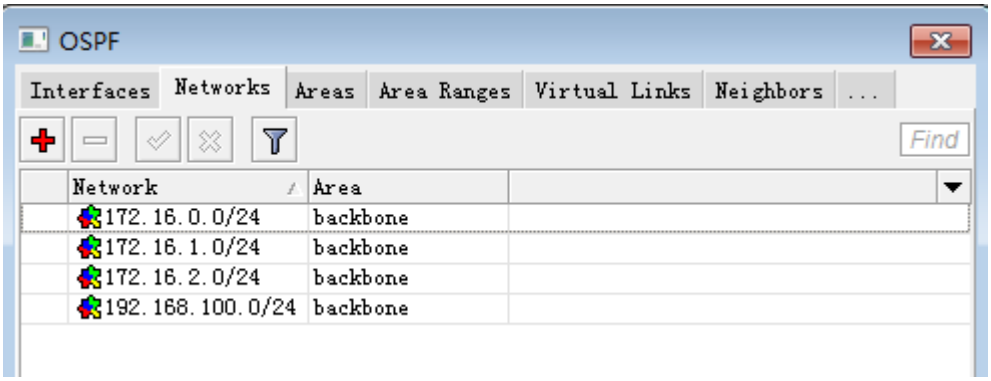


路由器 B

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

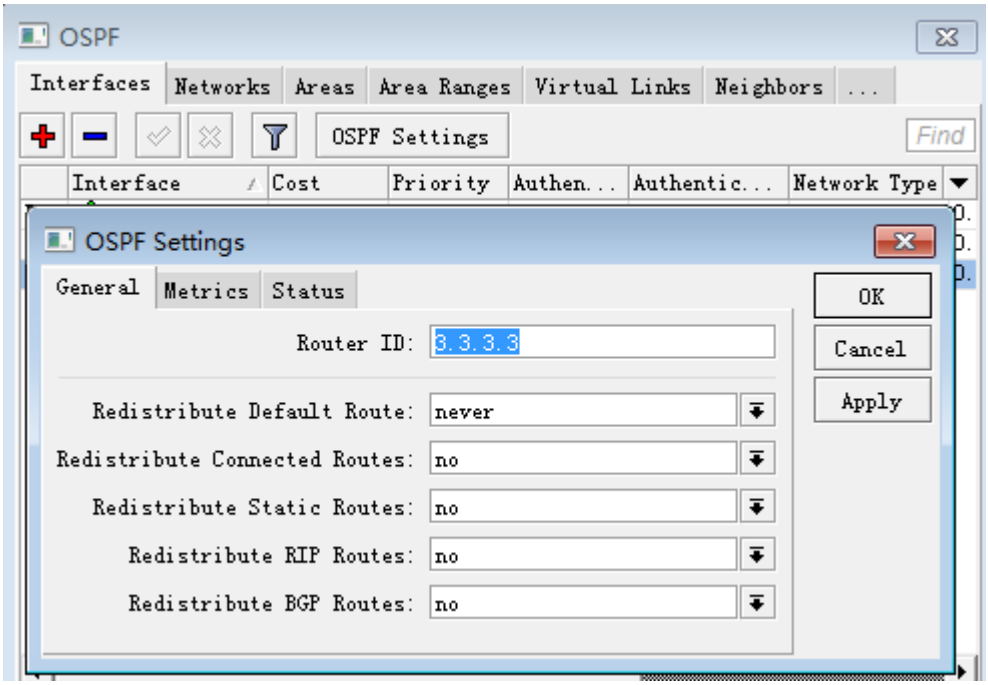


接下来我们同样设置 network 参数，申明自己的网络地址段，路由器 B 包含以下网络

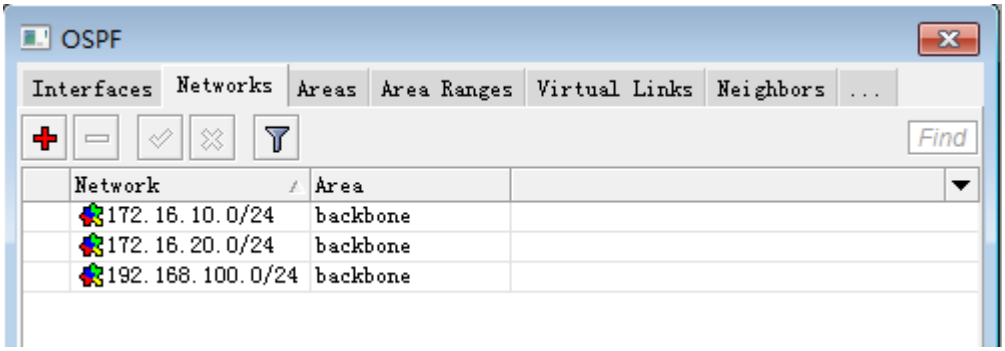


路由器 C

进入 OSPF，添加一个 OSPF 设置

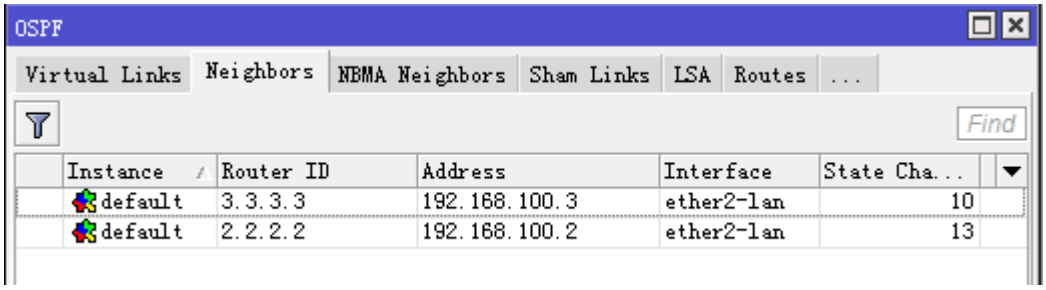


申明自己的 IP 地址段，添加本地网络的 IP 地址，我们都默认采用 Backbone 的域

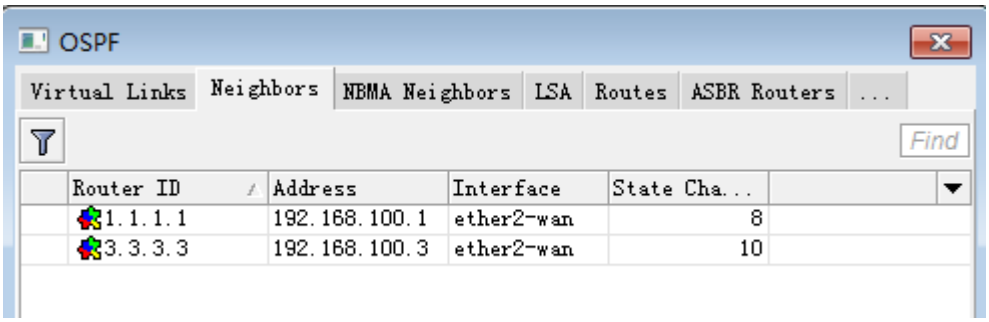


这样 3 台路由器配置完成，如果连接成功后，我们检查下是否建立连接，进入 neighbors 查看是否找到周围的设备

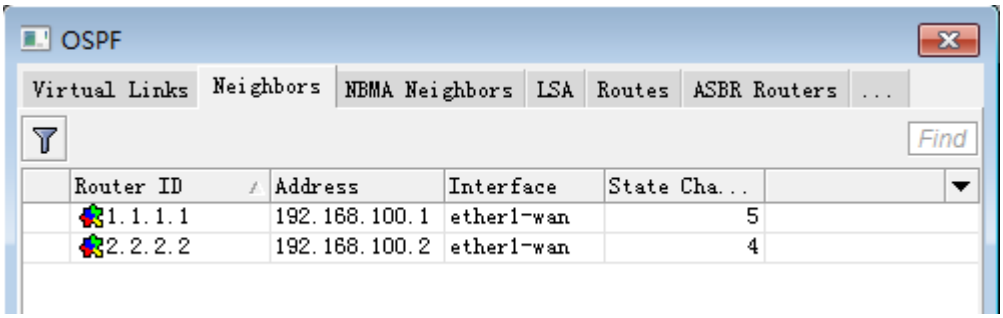
路由器 A，找到了另外两个设备显示如下



路由器 B

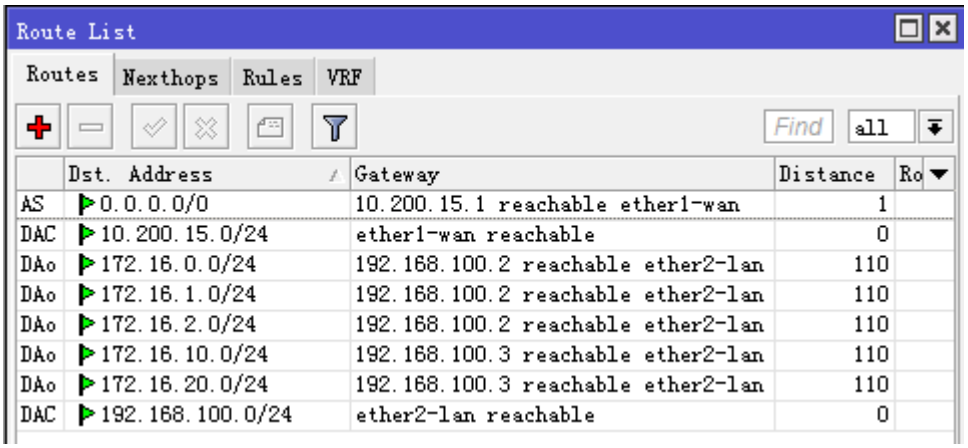


路由器 C

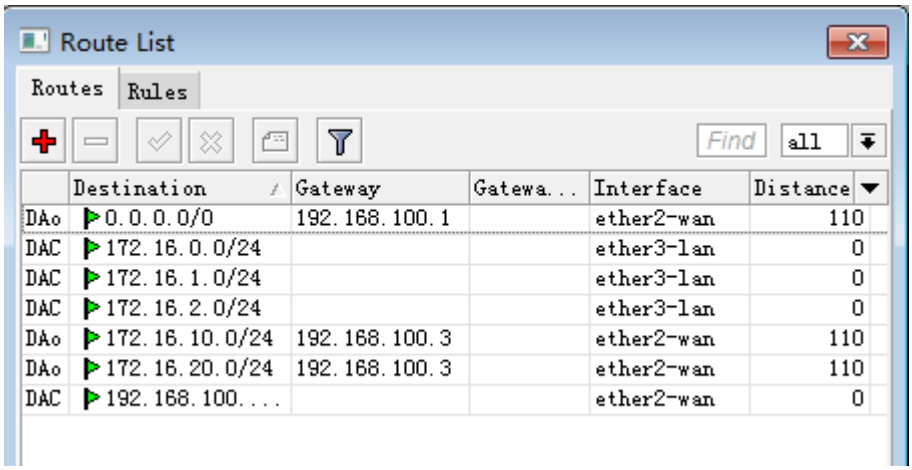


找到相互邻居后，我们可以在 ip route 里看到 OSPF 已经自动生成了路由列表，不需要像静态路由那样每条都手动添加设置

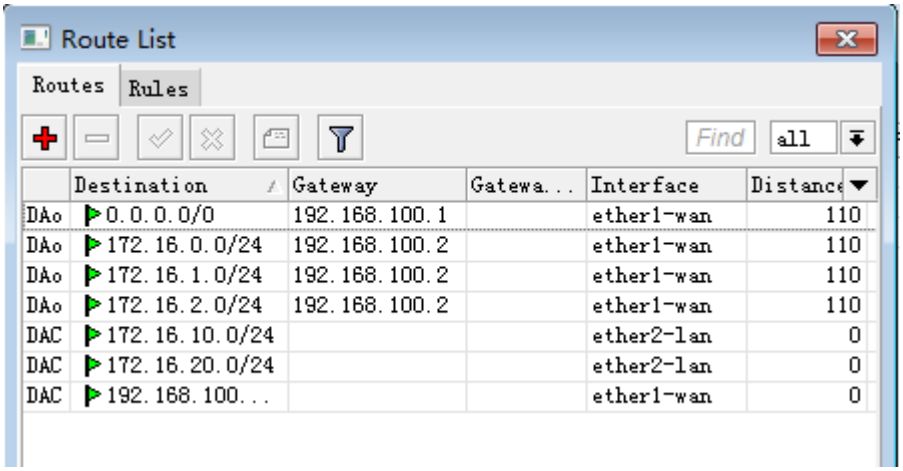
路由器 A 的路由表



路由器 B 路由表



路由器 C 路由表

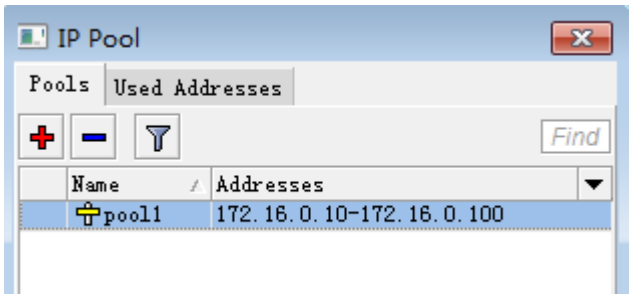


我们检查路由

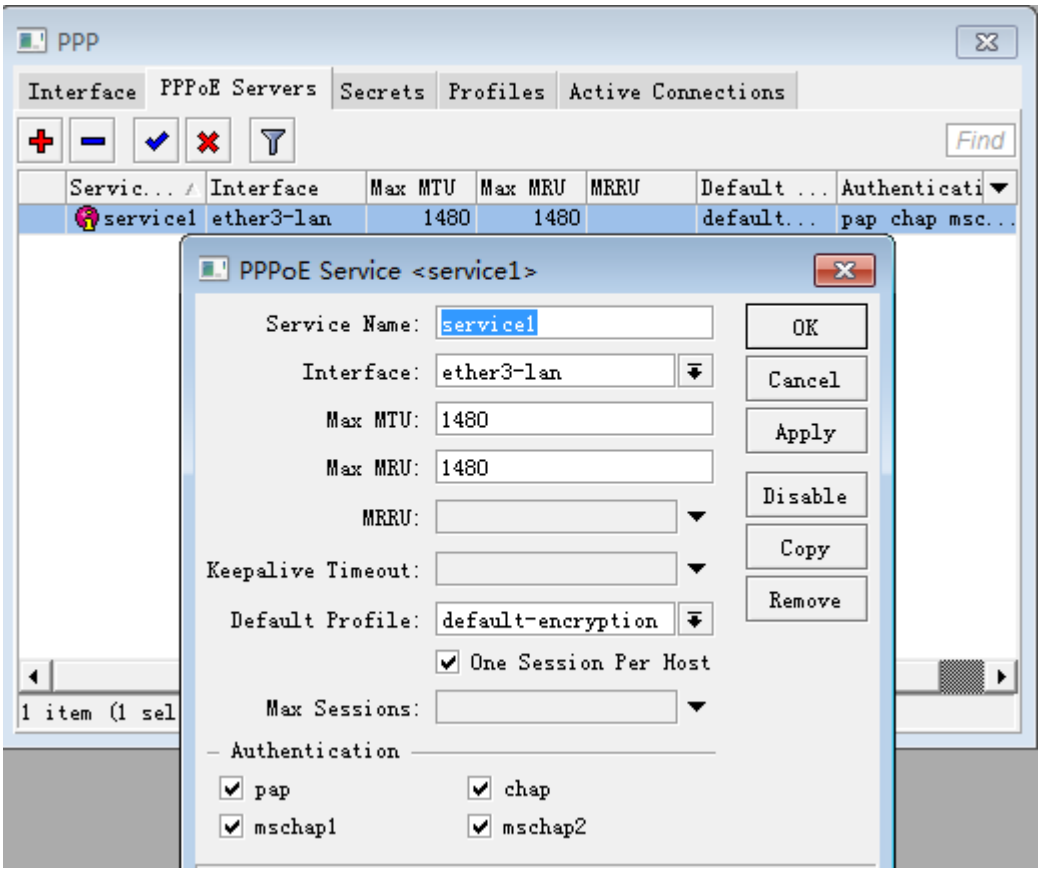
```
[admin@RouterA] > ping 172.16.1.1
HOST                                     SIZE  TTL TIME  STATUS
172.16.1.1                               56     64  1ms
172.16.1.1                               56     64  1ms
172.16.1.1                               56     64  0ms
172.16.1.1                               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，

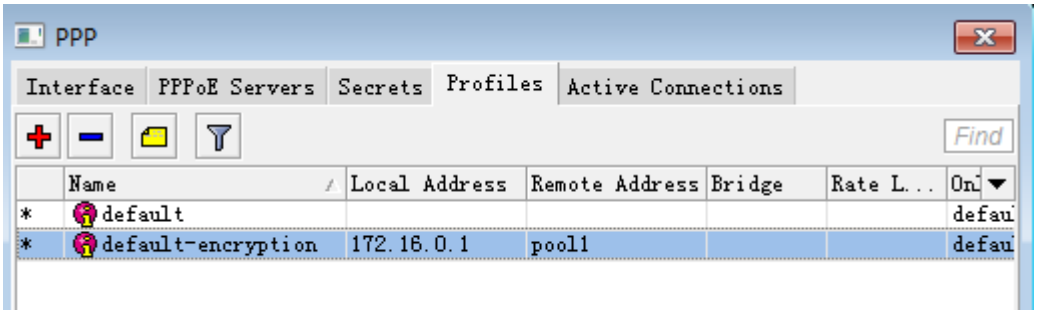
通过 ip pool 分配地址段为

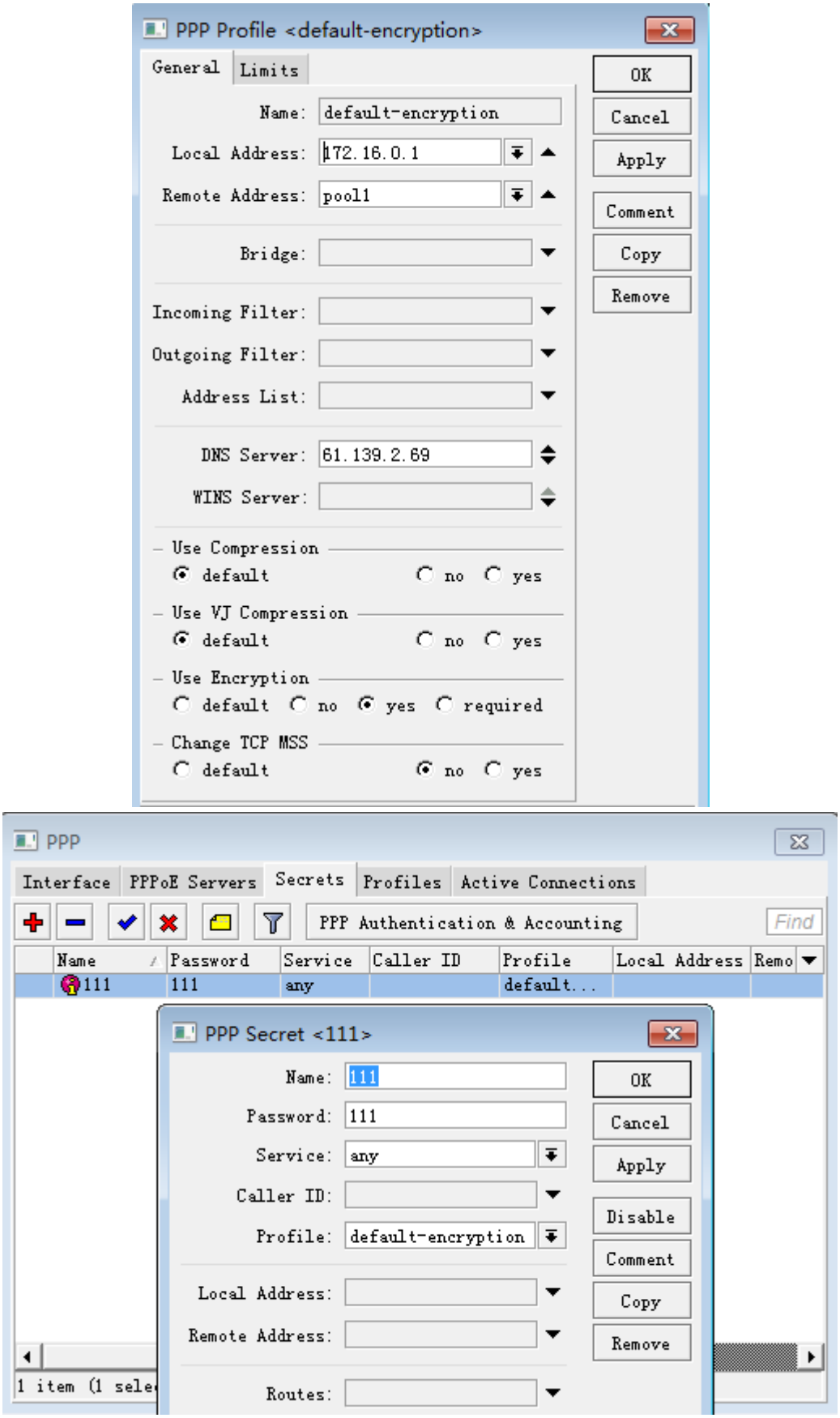


进入 winbox 的 PPP 菜单建立 PPPoE 服务器

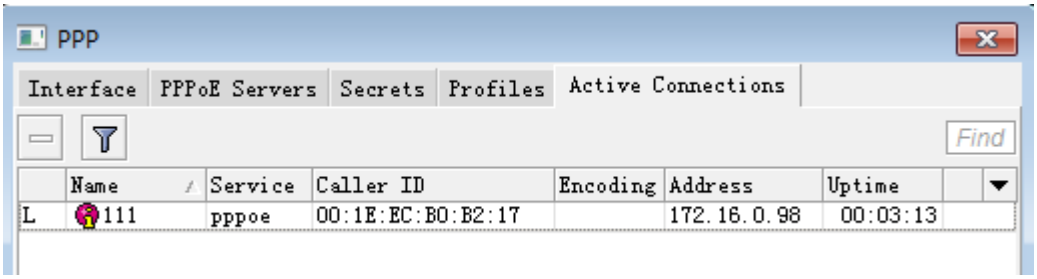


设置 PPPoE 服务器的 Profile 规则，并定义地址池 pool1

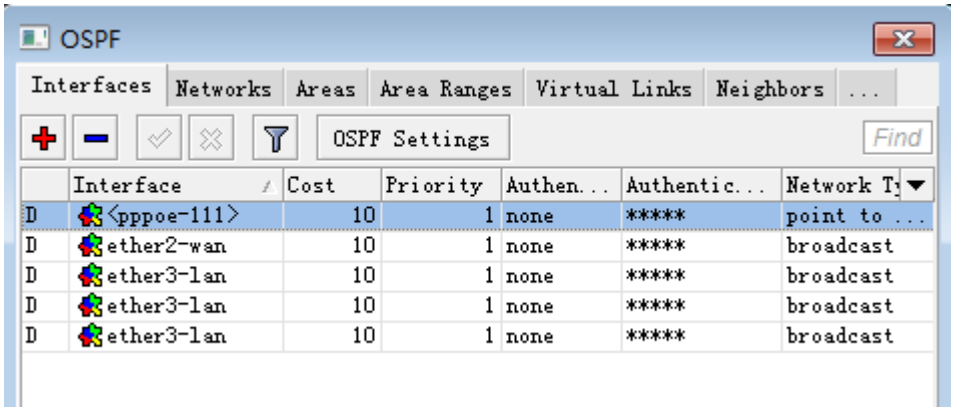




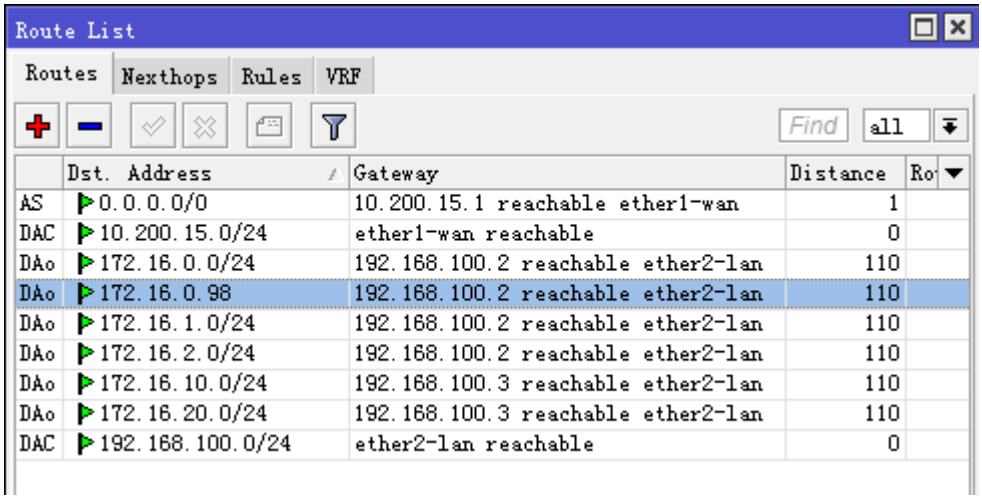
拨号成功后，获得的 IP 地址是 172.16.0.98



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



第三十一章 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。

31.1 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。

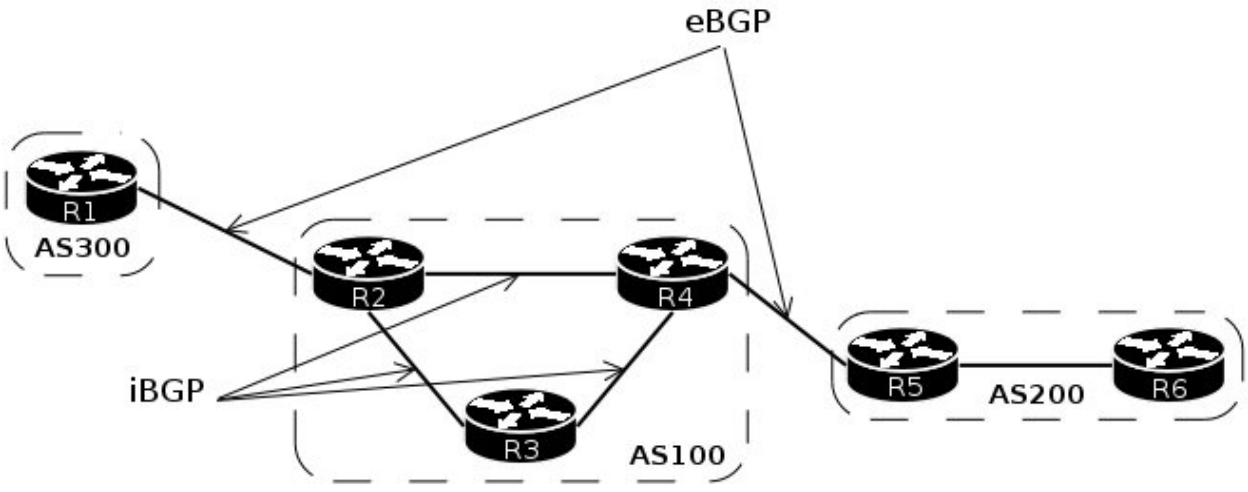
自治系统 AS（Autonomous System）

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 设备会发送和响应此报文。

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。

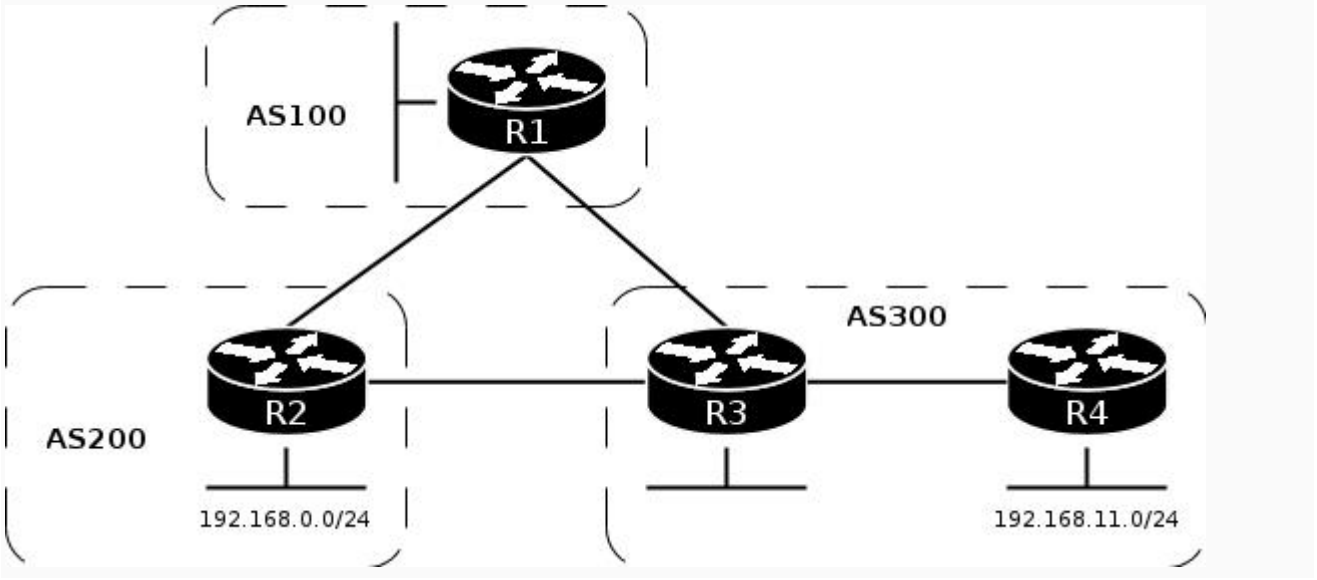
- **Idle**—BGP 进程被启动或被重置，这个状态是等待开始，比如等于指定一个 BGP peer，当收到 TCP 连接请求后，便初始化另外一个事件，当路由器或 peer 重置，都会回到 idle 状态。
- **Connect**—检测到有 peer 要尝试建立 TCP 连接。
- **Active**—尝试和对方 peer 建立 TCP 连接，如有故障，则回到 idle 状态
- **OpenSent**— TCP 连接已经建立，BGP 发送了一个 OPEN 消息给对方 peer，然后切换到 OpenSent 状态，如果失败，则切换到 Active 状态。
- **OpenReceive**— 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 默认不会重分布路由，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 前缀列表。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路由，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：

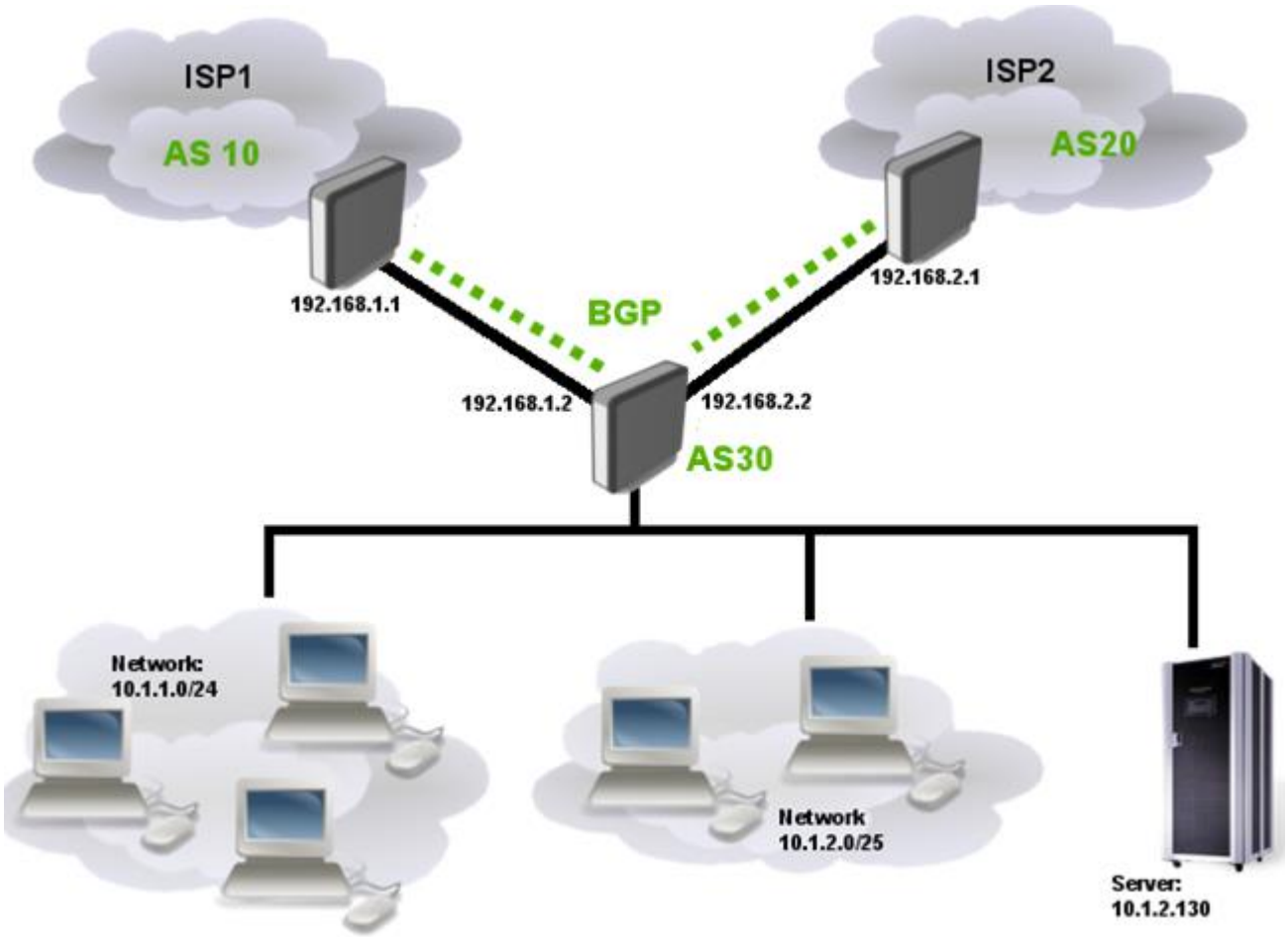
```
routing filter add chain=bgp-in set-in-nexthop=10.0.1.1,10.0.2.1
```

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
```

31.2 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
```

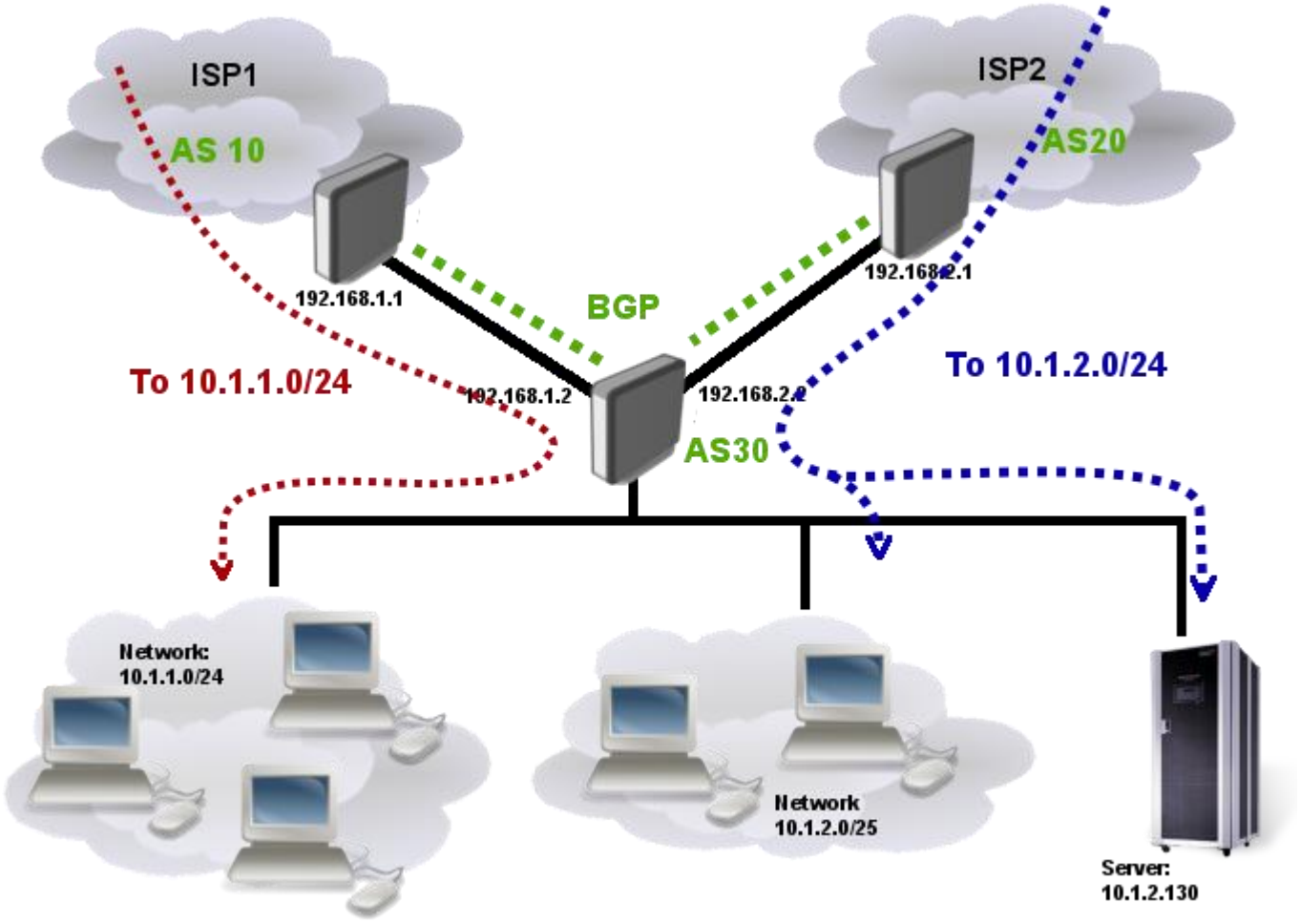
由于我们到两个 ISP 是建立默认路由，所以不需要接收，两个 ISP 向我们发布的任何路由，直接配置为拒绝接收，设置两条默认的静态路由，但到 ISP2 的默认路由 distance 设置为 30 作为备份链路，并开启网关 ping 检测

```
/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
```

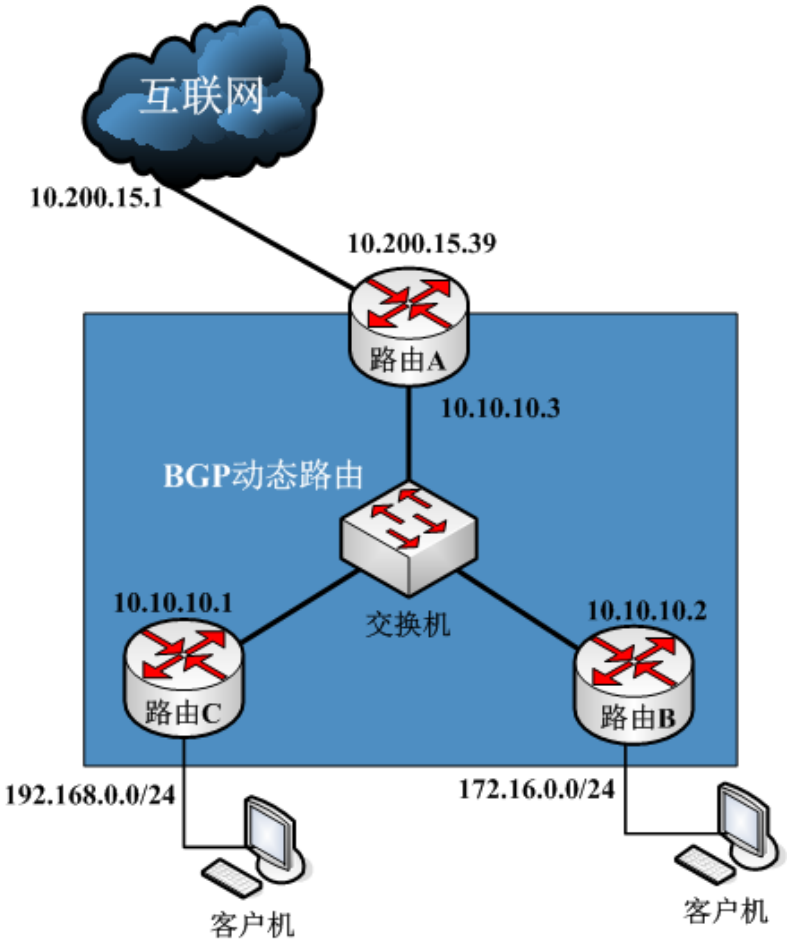
向 ISP2 发布路由过滤规则：

```
/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。

31.3 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：

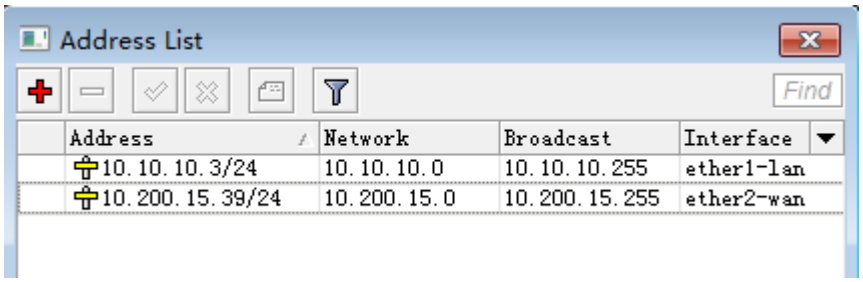


路由器 A 是连接互联网的出口路由器，路由器 B 和路由器 C 分别通过 10.10.10.0/24 与路由器 A 互联，并分别连接了 192.168.0.0/24 和 172.16.0.0/24 的网段

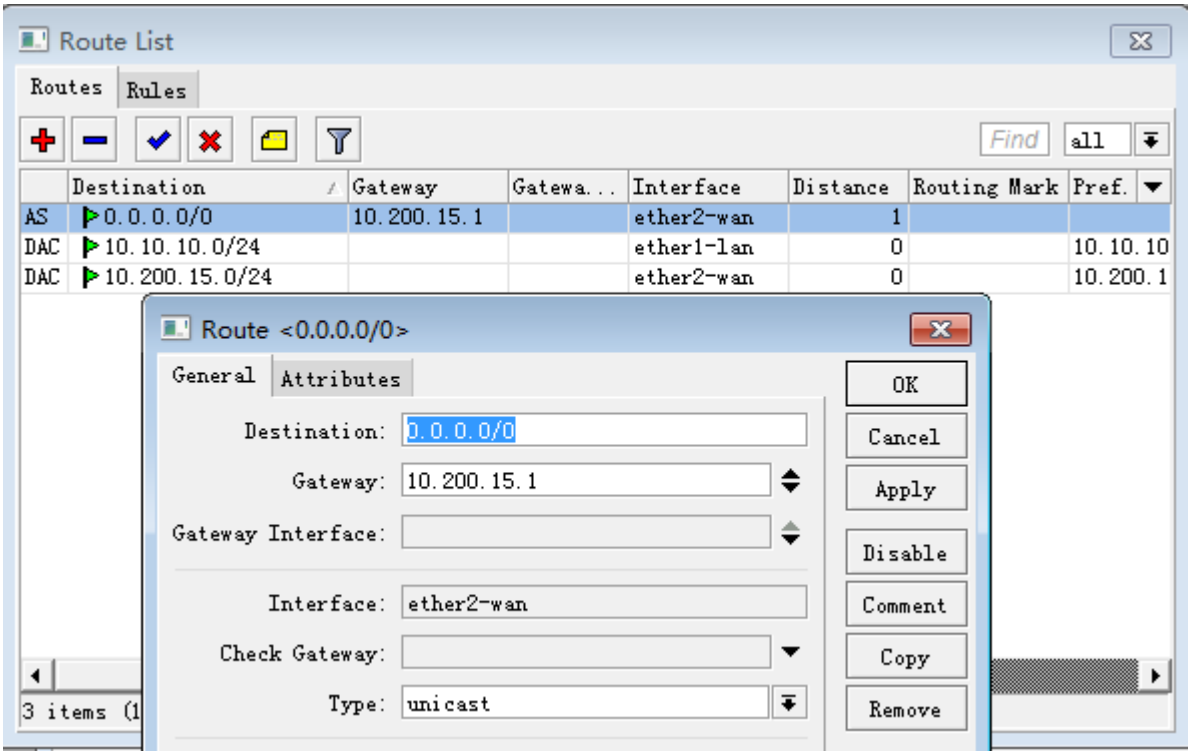
在这里我们使用默认的 AS 编号 65530，BGP 配置，我们首先设置每台路由器的 IP 地址和相关的参数

路由器 A 配置：

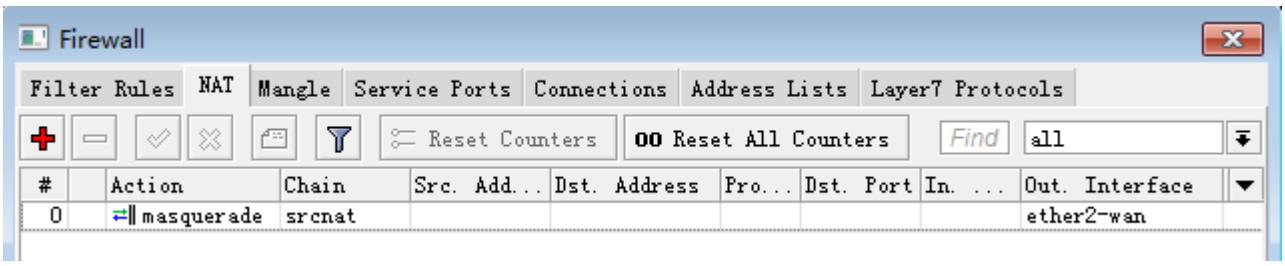
添加 IP 地址到 ip address 中



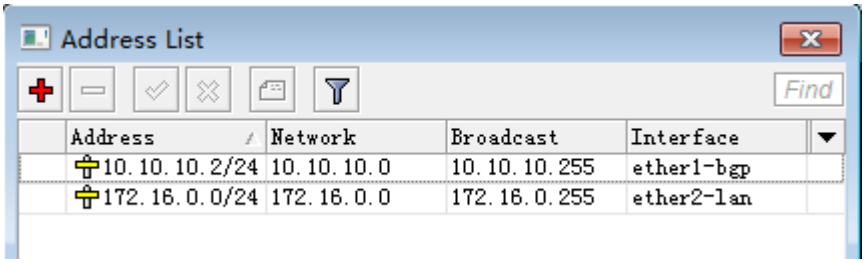
路由器 A 需要连接互联网，所以这里我们设置一个默认静态路由：



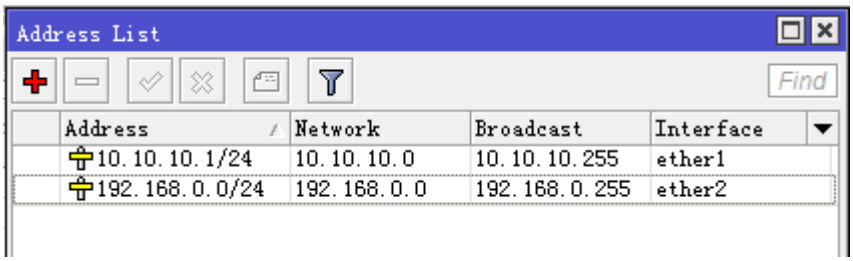
路由器 A 是网关出口，所以我们需要做一个 nat 的转换，进入 ip firewall nat 添加转换规则



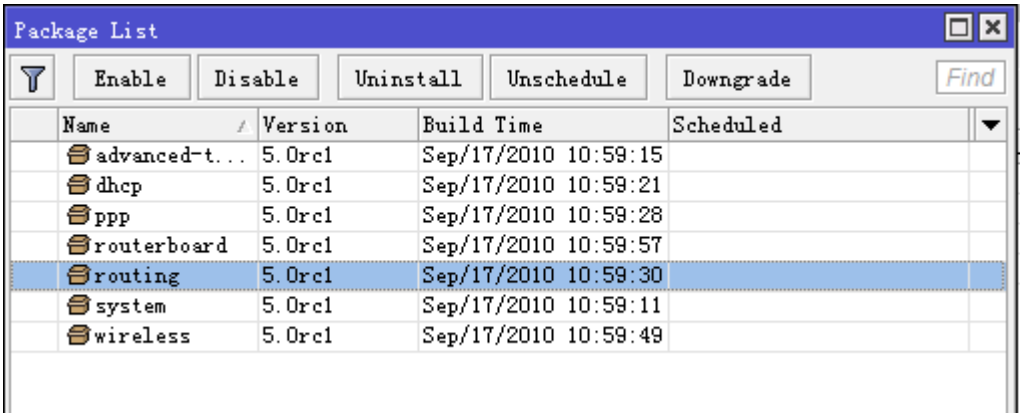
配置路由器 B 的 IP 地址



路由器 C 的 IP 地址



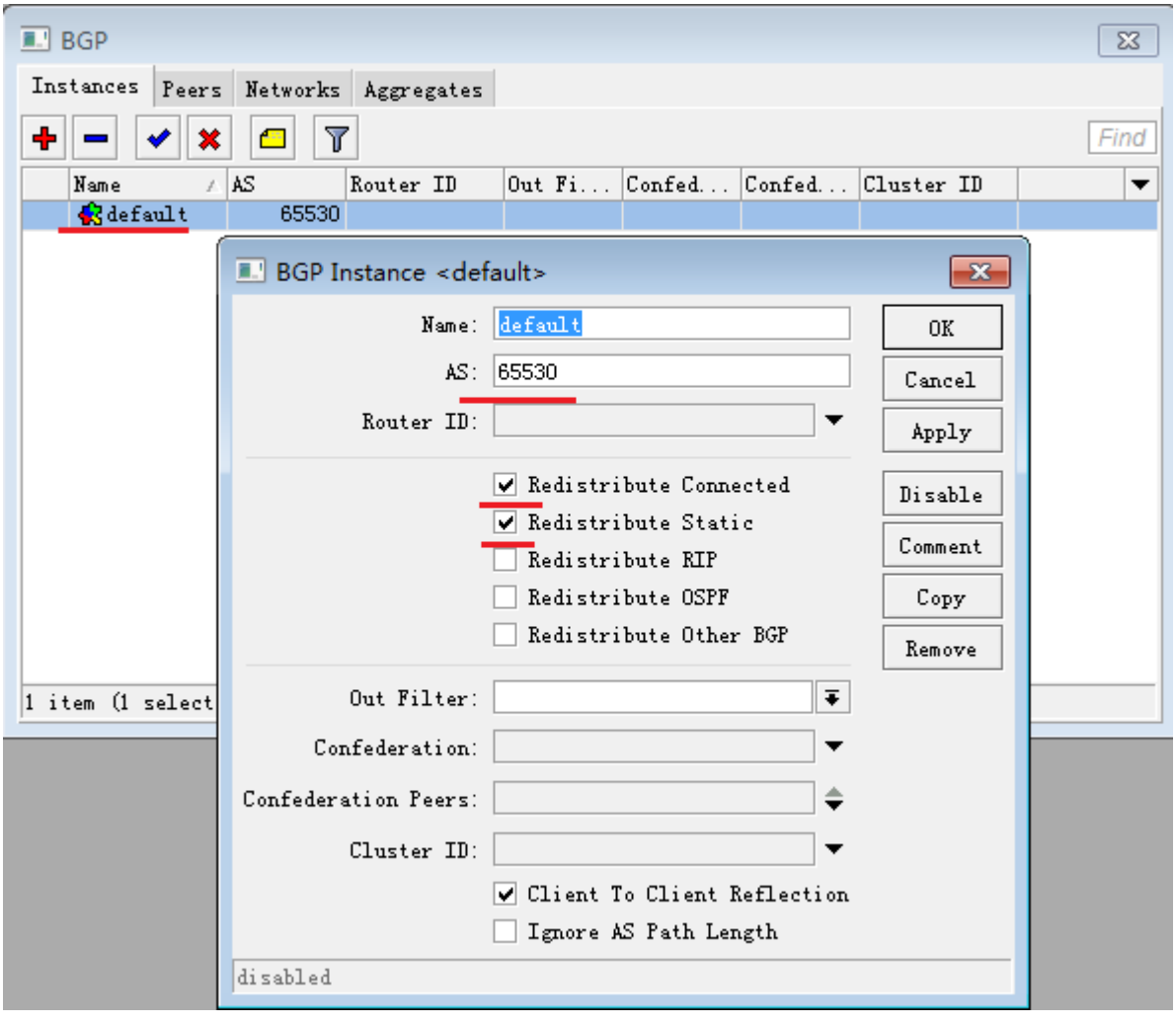
接下来就是配置 BGP 连接，我们首先确定每个路由器都安装了 routing 的功能包，如果没有安装就无法使用动态路由协议，确定是否安装我们可以查看 system package 里的参数：



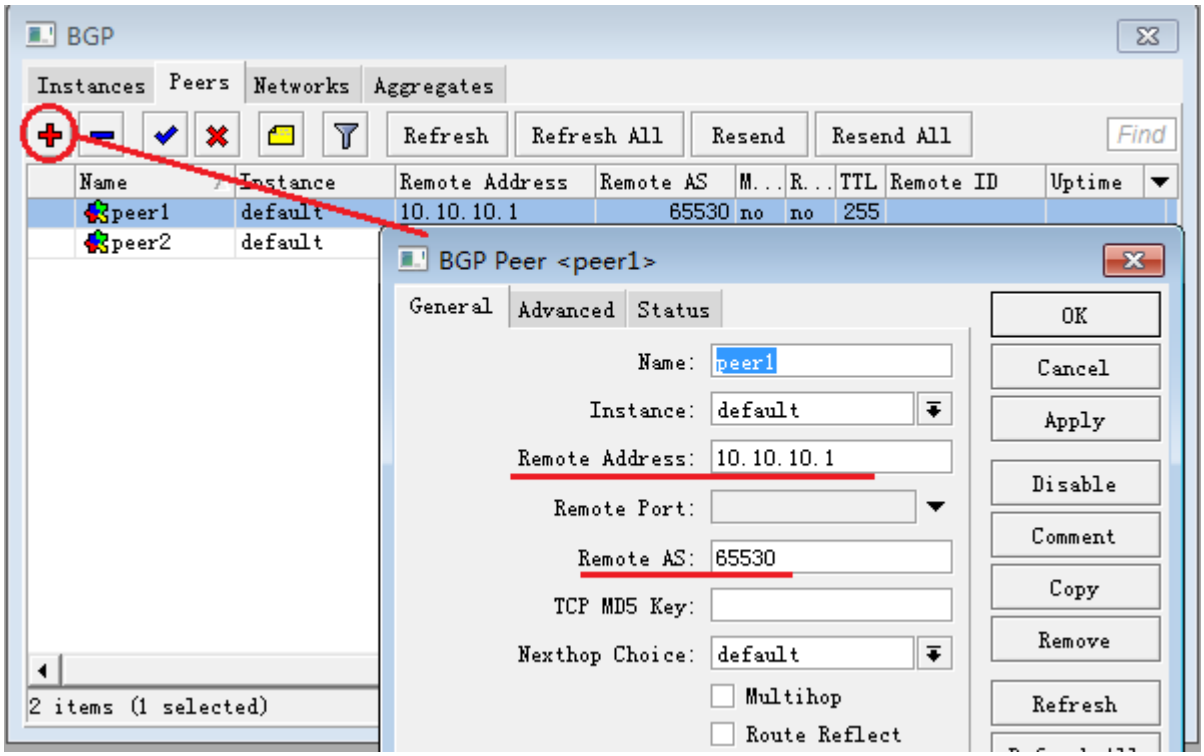
确认安装后，我们可以在 winbox 的左边菜单找到 routing 选项。

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，界面有所不同，但操作是一样的。

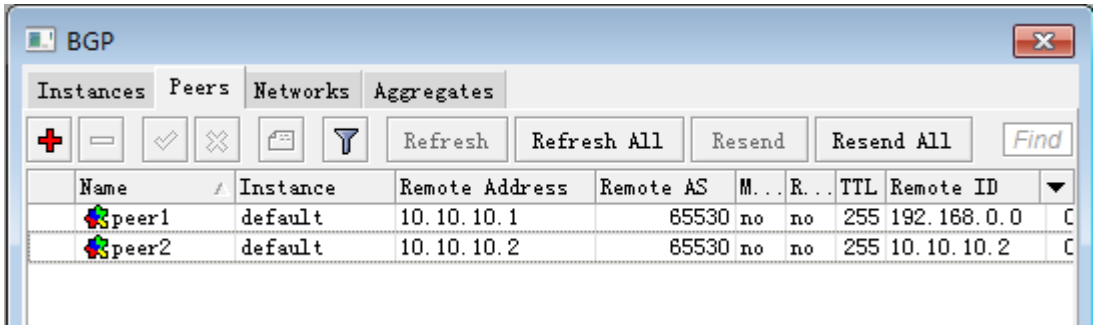
首先配置路由器 A 的 BGP 参数，打开 BGP 后我们选择 Instances，打开 default 选项，我们使用默认的 AS：65530，Router-ID 这里可以不用设置，路由器会自动使用本地连接地址作为 ID，因为路由器 A 是网关出口，所以我们需要分发静态路由给下面的 BGP 路由器在，redistribute-connected 和 redistribute-static 打上勾



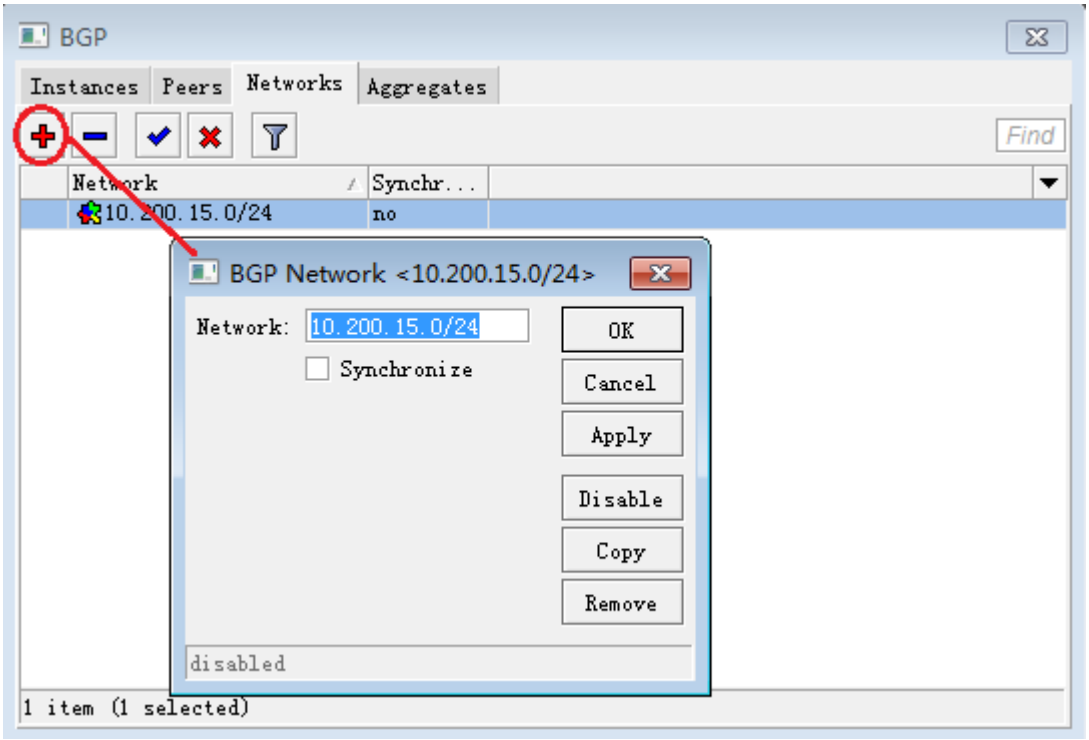
进入 peers 添加对端设备，路由器 A 的 IP 地址是 10.10.10.3，添加路由器 C 和 B 的 IP 地址，分别是 10.10.10.1，并设置 remote-as 参数 65530



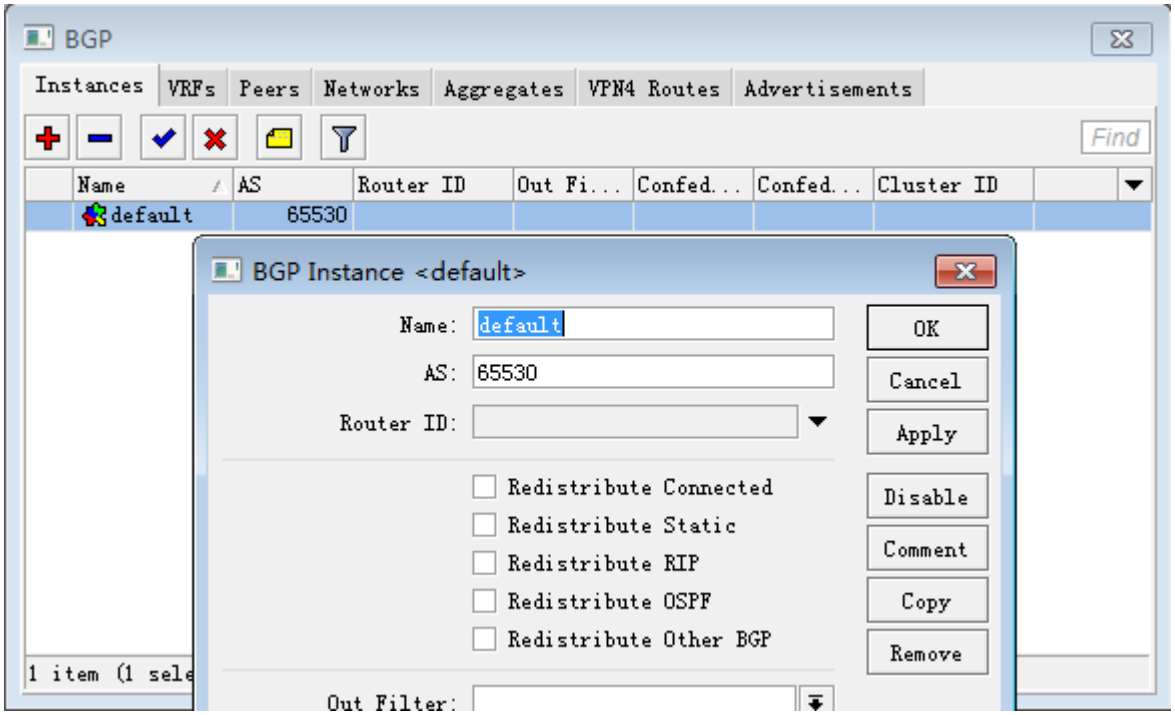
两个对应的路由器添加完成



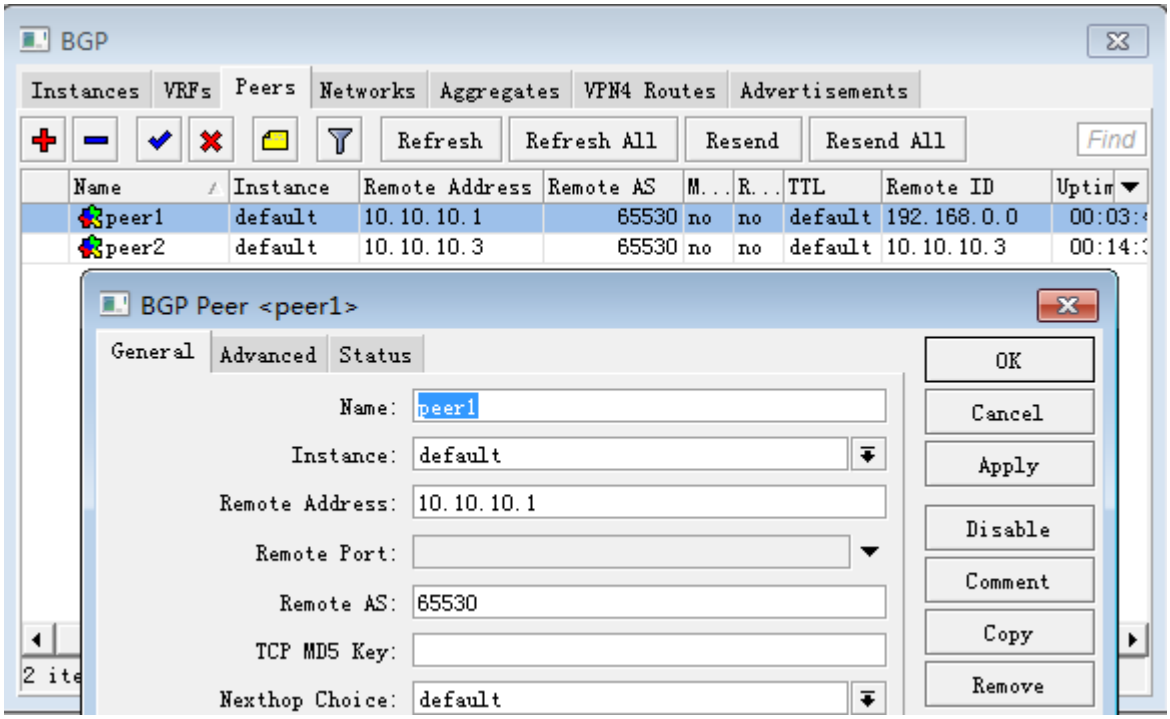
进入 network 申明自己的网段



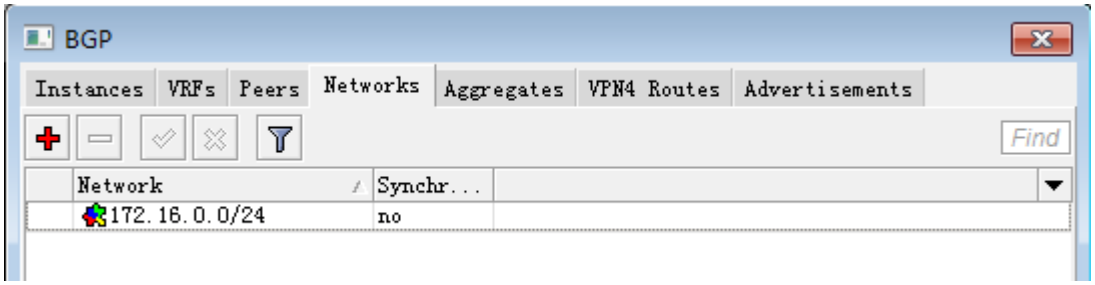
接下来我们配置路由器 B 的 BGP 参数：



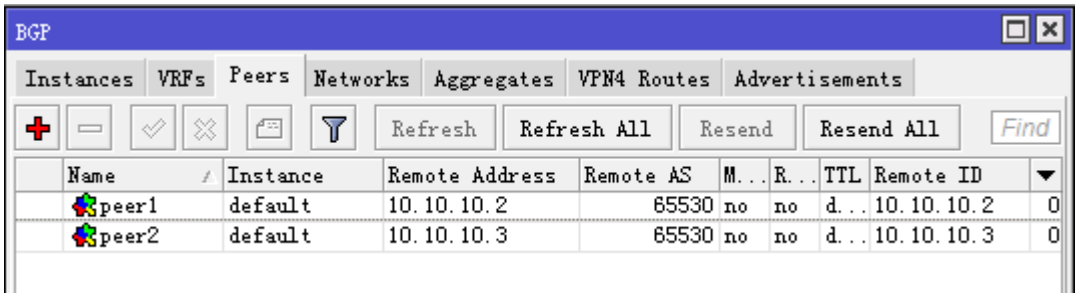
在 peers 的添加 2 个路由器的参数



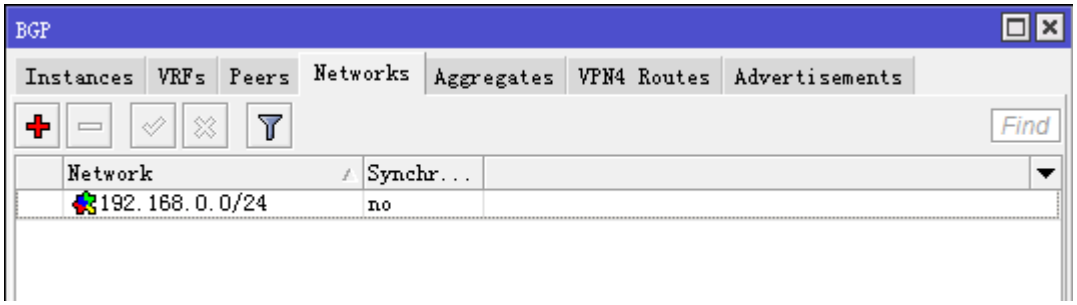
申明自己本地的网络地址段



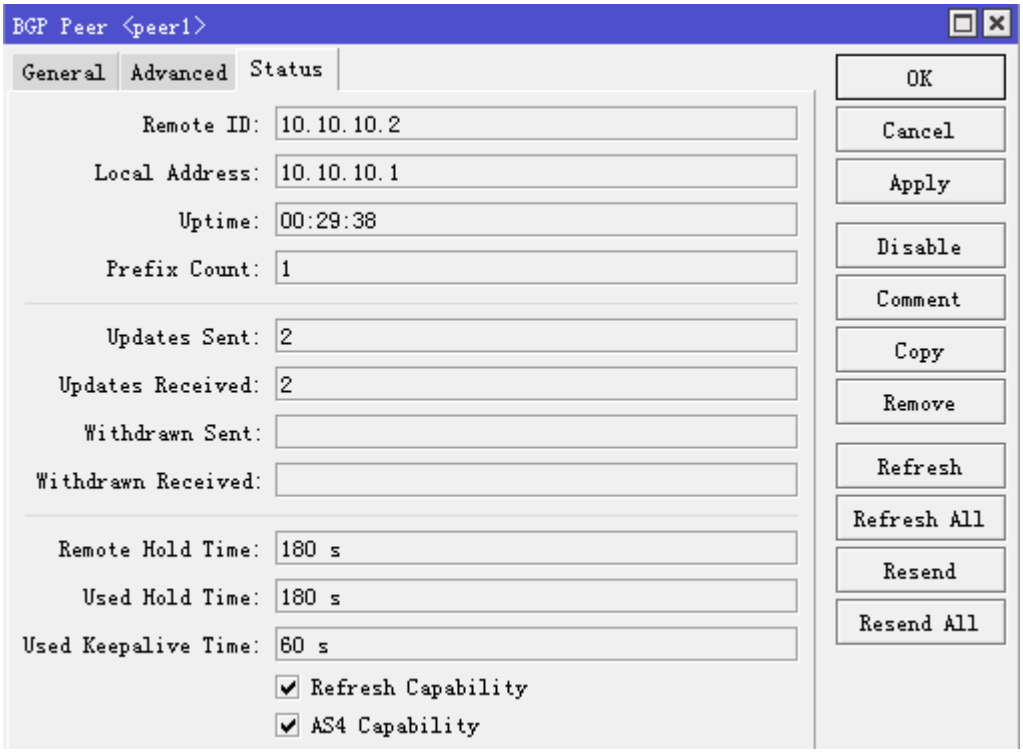
最后是路由器 C 的配置，instances 参数不变，同样在 peer 里，添加相应路由器的参数



申明自己本地的网络地址段

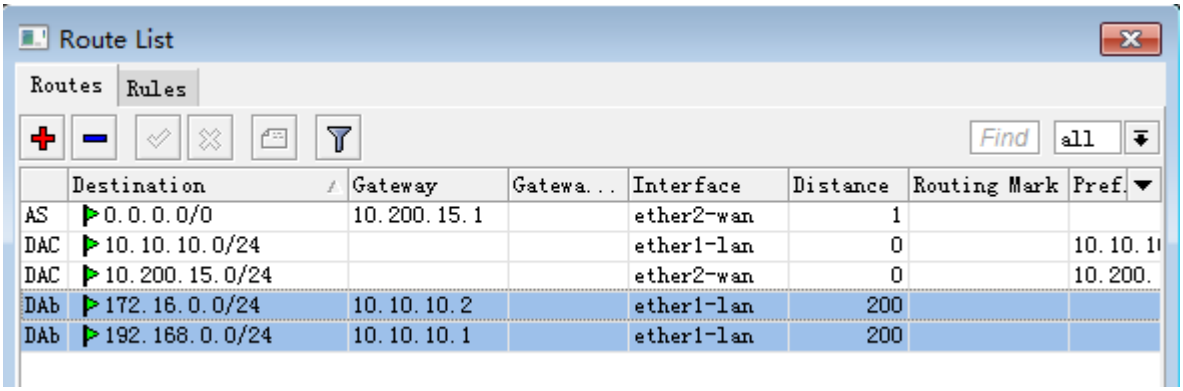


设置完成后，我们在路由器 C 上查看其中一个 BGP 连接状态

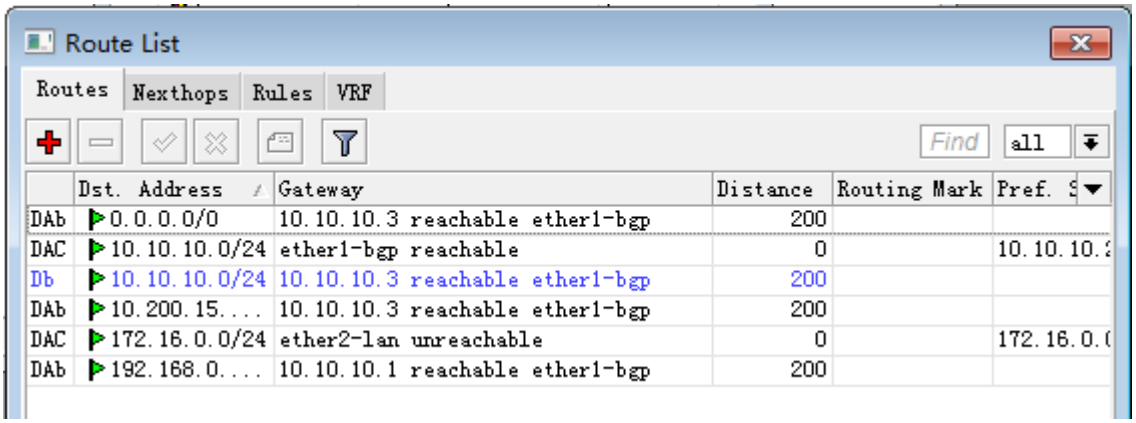


接下来我们可以到每个路由器的 ip route 里查看路由表

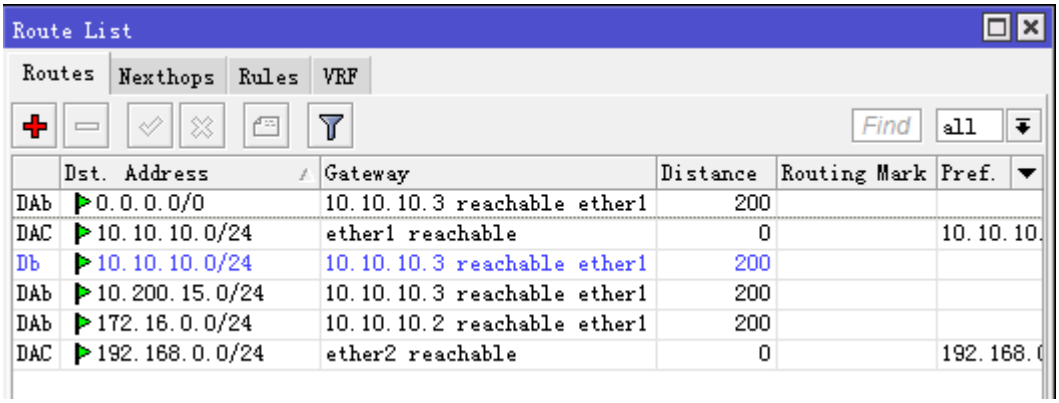
路由器 A：



路由器 B：



路由器 C：



这样一个 BGP 的动态路由建立完成

第三十二章 Proxy 代理

MikroTik RouterOS 支持下面的代理服务器功能：

- 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

设置 1GB 的 web 缓存，并通过 8000 端口监听，操作如下：

```
[admin@MikroTik] ip proxy> set enabled=yes port=8000 max-cache-size=1048576
[admin@MikroTik] ip proxy> print
                    enabled: yes
                src-address: 0.0.0.0
                       port: 8000
             parent-proxy: 0.0.0.0
       parent-proxy-port: 0
                cache-drive: system
    cache-administrator: "webmaster"
            max-cache-size: 1048576KiB
             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
[admin@MikroTik] ip proxy>
```

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>
```

规格

功能包需求：：***system***
许可等级: *Level3*
操作路径: ***/ip proxy（winbox：ip web-proxy）***
技术标准: HTTP/1.0, HTTP/1.1, FTP
硬件需求：需要内存和硬盘空间*(具体情况下面的属性)*

Proxy 能代理 HTTP 以及 HTTP 代理（对 FTP，HTTP 及 HTTPS 协议）请求。Web 代理通过存储被请求的因特网对象，以起到网页缓存功能的作用，例如，通过在一个网络数据产生的站点更接近接受者的系统上的 HTTP 及 FTP 协议数据的可用数据。这里“更接近”指的是增加的路径可靠度，或速度或者两者都有。Web 浏览器可以使用本地代理缓存来加快访问并减少带宽消耗。

当设置代理服务时，确定它只为你的客户服务，而不是误用为继电器。请阅读访问列表部分的安全注意。

注意保持 web 代理一直运行，即使当你想使用它作为像 HTTP 及 FTP 防火墙（例如，拒绝访问 mp3 文件）或把请求透明地重定向到外部代理时也没有缓存，这样做会是很有用处的。

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。你可以使用控制台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 站点
0.0.0.0 – 合适的 **src-address** 将会自动从路由列表中取出

注：这个 web 代理监听所有路由器 IP 地址列表中包含的 IP 地址。

在端口 8000 上启用代理：

```
[admin@MikroTik] ip proxy> set enabled=yes port=8000
[admin@MikroTik] ip proxy> print
                  enabled: yes
              src-address: 0.0.0.0
                    port: 8000
             parent-proxy: 0.0.0.0:0
              cache-drive: system
       cache-administrator: "dmitry@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-object-size: 2000KiB
           max-fresh-time: 3d
[admin@MikroTik] ip proxy>
```

32.1 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，规则从顶到底顺序处理。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 的网站链接，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：统配符属性（dst-host 和 dst-path 可以使用）匹配一个完整的字符串（例如，如果设置为"example"，则他们不会匹配"example.com"）。可用的统配符'*'（匹配任何数量的任何字符）以及'?'（匹配任何一个字符）。这里也接受常规表达，但是如果属性被当作常规表达处理，那就应该以冒号(':')开始。

在常规表达式中的低命中：

- **** 符号顺序用于在控制台中输入****字符
- **\.** 样式仅表示 **.** (在常规表达式中单独一个点表示任何符号)
- 表示在给定样式之前不允许任何符号，我们在样式的开头使用**^**符号
- 指定在给定样式之后不允许任何符号，我们在样式的结尾使用**$**
- 输入**[** or **]**符号，你可以用反斜杠对它们转义

强烈建议拒绝所有 IP 地址除了在路由器之后的那些，因为代理仍然可以访问你的 internal-use-only（企业网）web 服务器。在 Firewall Manual 中查询如果保护你的路由器。

32.2 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

32.3 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

32.4 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列表。

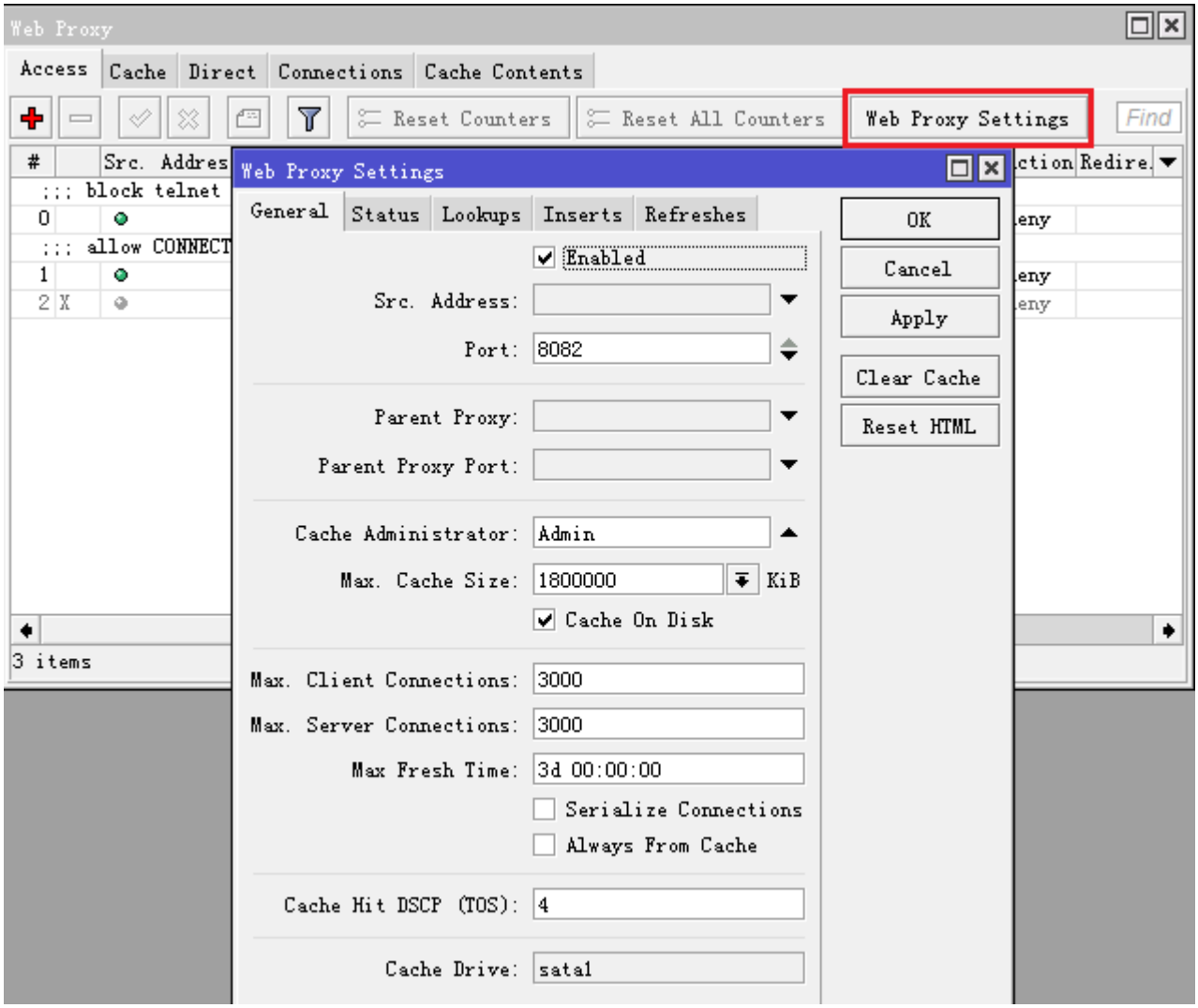
属性描述

dst-address (*只读: IP 地址*) – 连接的 IP 地址
protocol (*只读: 文本*) – 协议名
rx-bytes (*只读: 整型*) - 客户接收的字节量

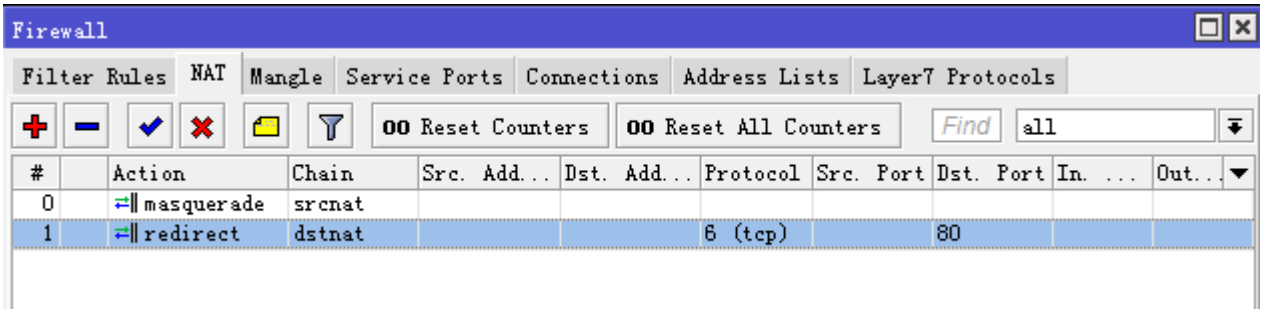
src-address (*只读: IP 地址*) – 连接源发站的 IP 地址
state (*只读:* closing | connecting | converting | hotspot | idle | resolving | rx-header | tx-body | tx-eof | tx-header | waiting |) – 打开连接的状态
closing - 数据传输完成，连接正在最终完成
connecting - 建立 toe 连接
hotspot – 检查是否 hotspot 认证允许继续（对 hotspot 代理）
idle - 闲置状态
resolving – 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

32.5 Web 代理应用事例

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：



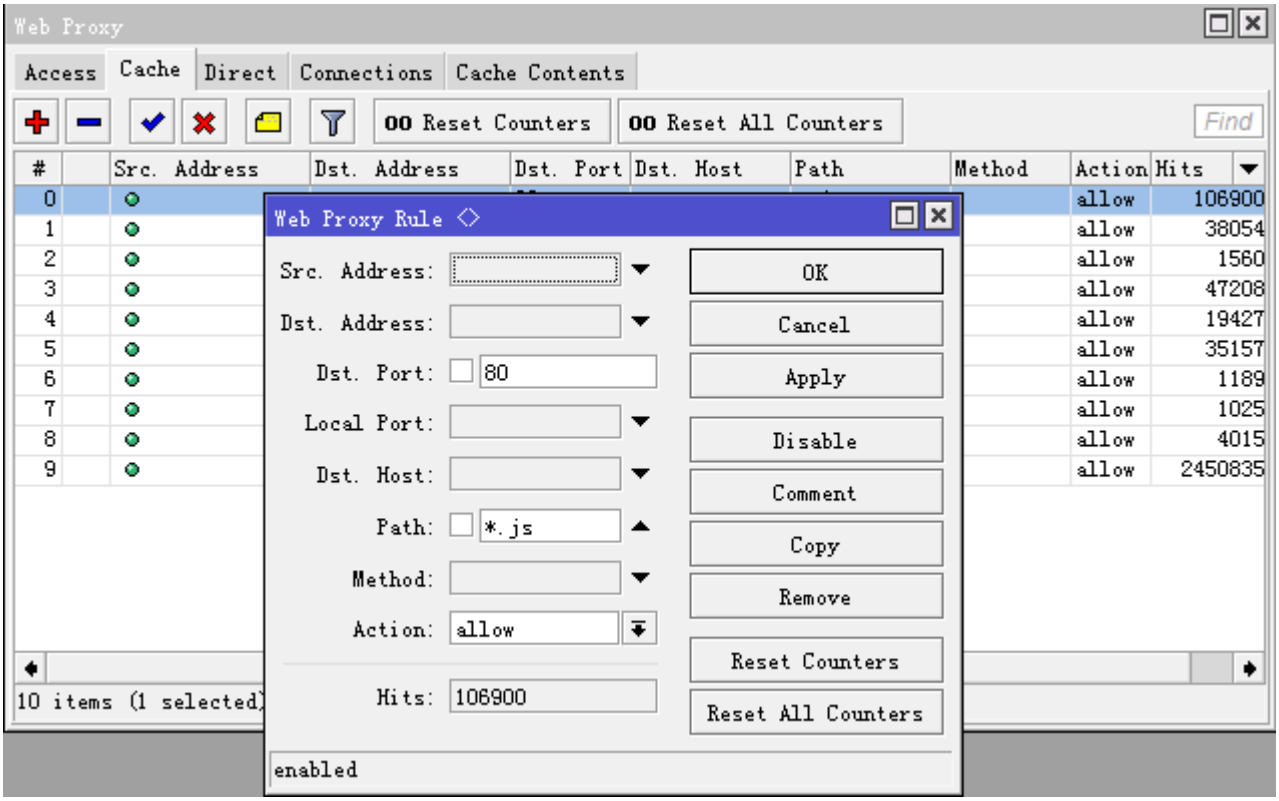
CLI 操作命令

```
/ip firewall nat
chain=dstnat protocol=tcp dst-port=80 action=redirect to-ports=8082
```

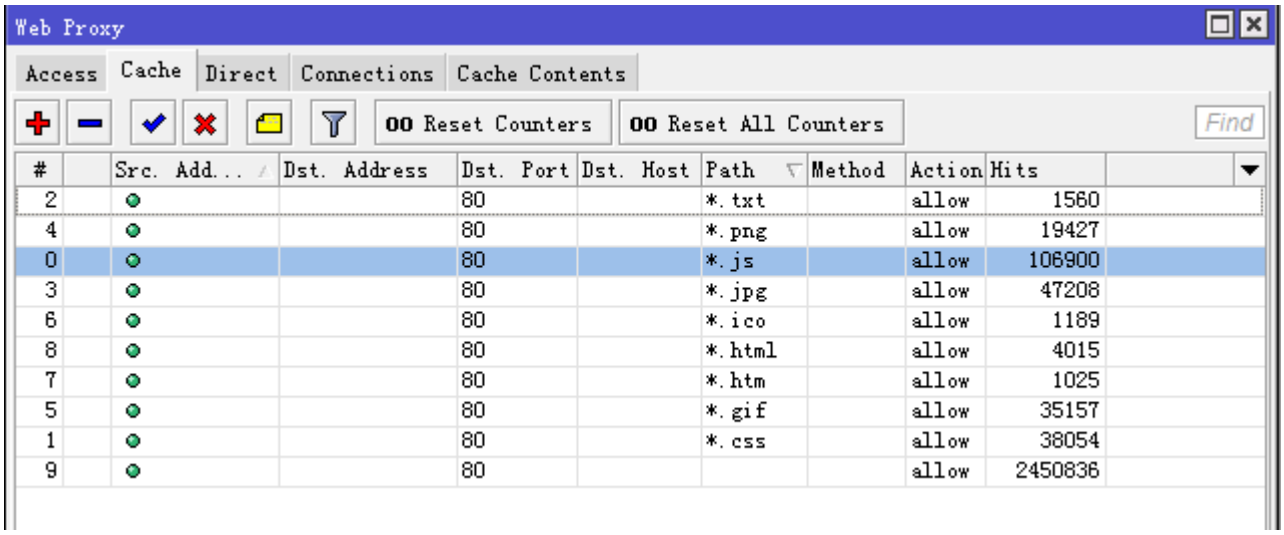
确定你路由器本地的 Proxy 没有打开代理，并禁止外网通过路由器代理上网：

```
/ip firewall filter
chain=input in-interface=<Your WAN Port> src-address=0.0.0.0/0 protocol=tcp dst-port=8082 action=drop
```

缓存我们需要的文件例如后缀名为：.js .css .html .jpg…等，这样有利于提高缓存命中率。



我们缓存的列表



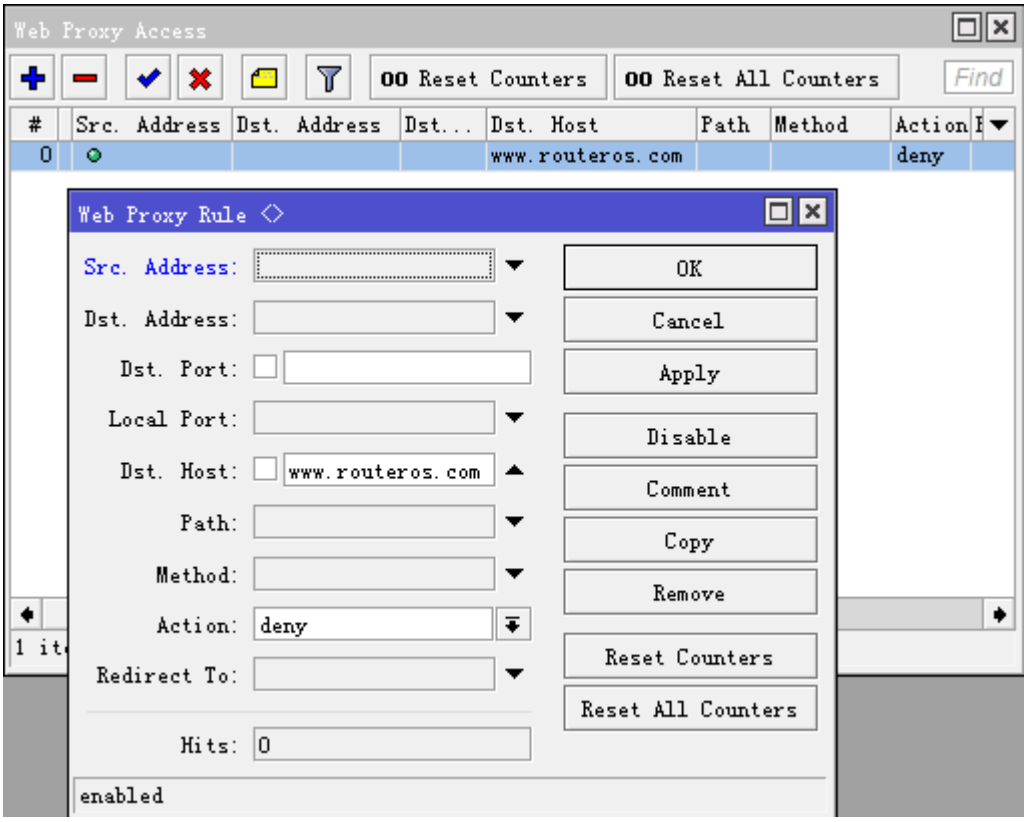
#	Src. Add...	Dst. Address	Dst. Port	Dst. Host	Path	Method	Action	Hits
2			80		*.txt		allow	1560
4			80		*.png		allow	19427
0			80		*.js		allow	106900
3			80		*.jpg		allow	47208
6			80		*.ico		allow	1189
8			80		*.html		allow	4015
7			80		*.htm		allow	1025
5			80		*.gif		allow	35157
1			80		*.css		allow	38054
9			80				allow	2450836

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。

proxy 访问控制

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
```



我们可用阻止文件如“.mp3, .exe, .dat, .avi”的下载。

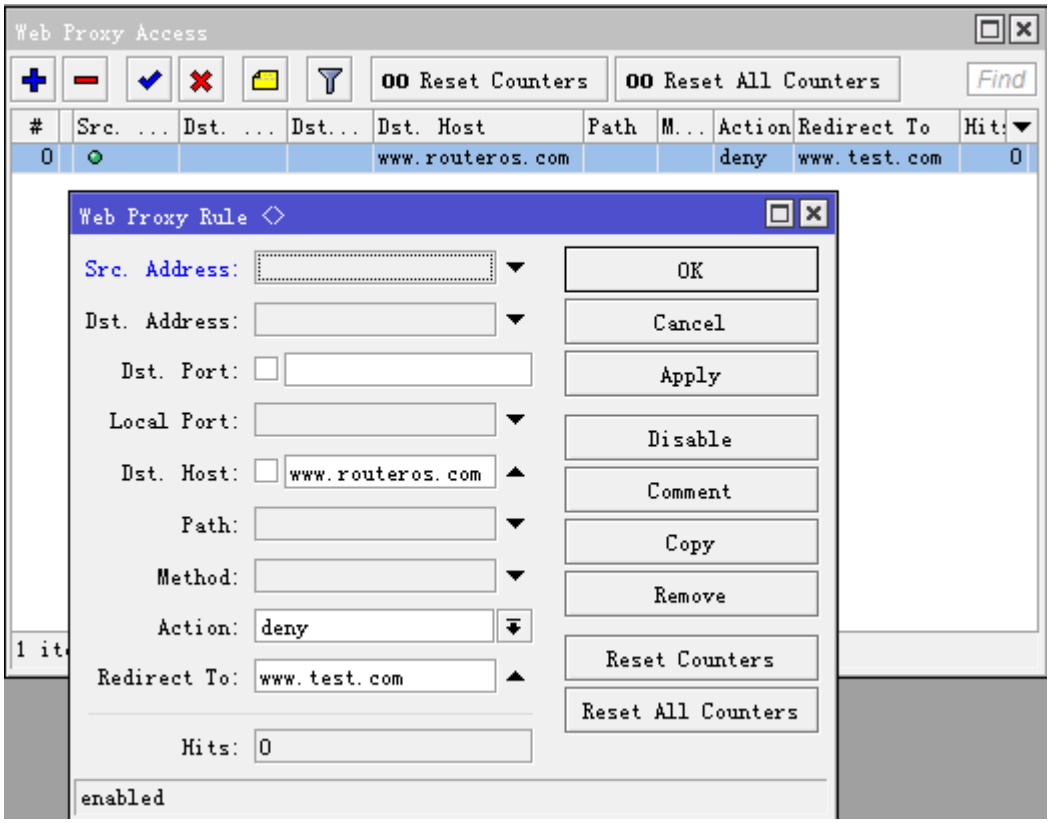
```
/ip proxy access
```

```
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”的关键字链接

```
/ip proxy access
dst-host=:mail action=deny
```

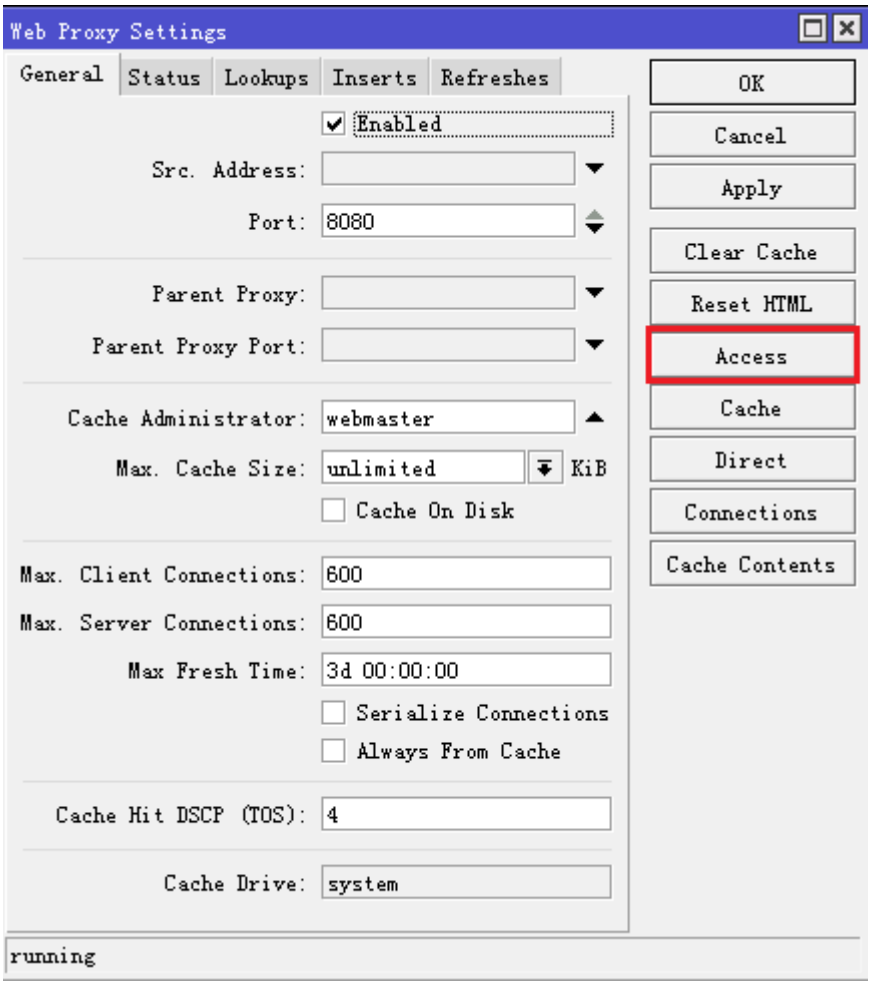
也可以通过重定向网站访问到指定页面，如重定向 www.routeros.com 到 www.test.com



32.6 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，

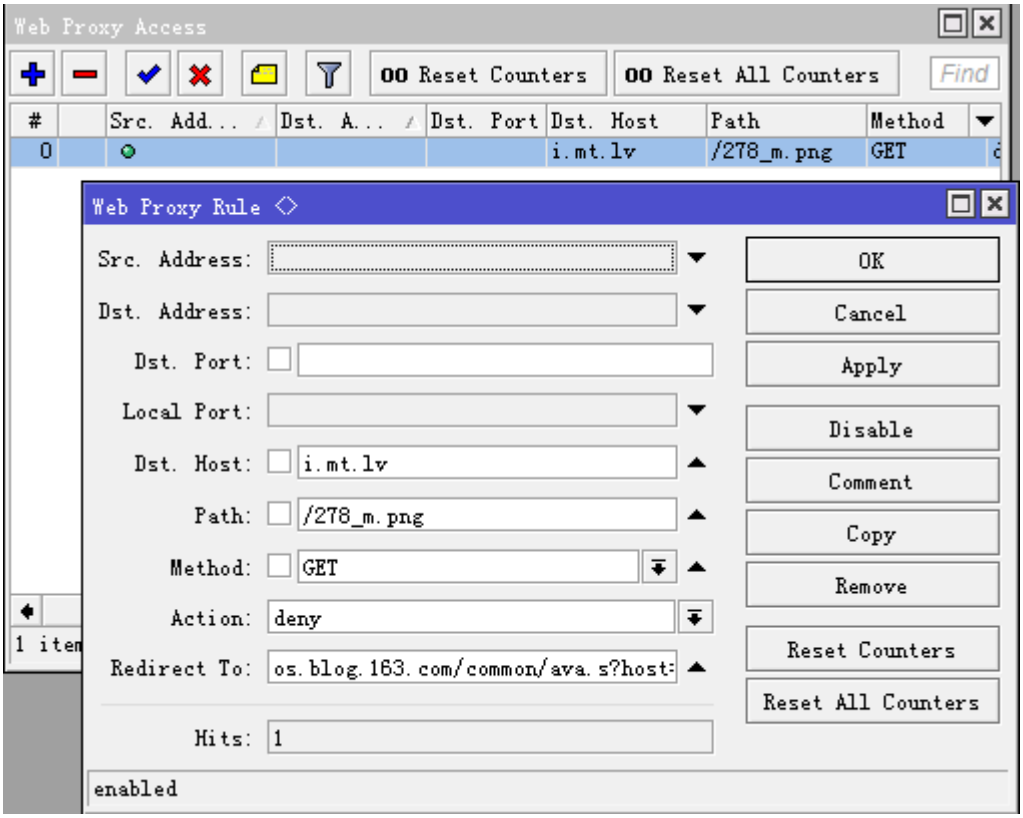
要实现重定向功能，需要使用到 proxy 的 access 功能，通过将访问数据重定向，在这个 proxy 配置里我们没有考虑 cache 设置。



设置好 nat 里的 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
- 重定向到：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。

第三十三章 虚拟化技术

RouterOS 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，

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文件。镜像文件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。

33.1 虚拟化技术应用

下面是一些虚拟机的可行方案（一些方案现在只指出 Xen，但 MetaRouter 将会添加更多的功能）：

数据管理中心

- 加强了一些路由器的硬件平台
- 加强路由器的服务，并更高等级的服务器如 VOIP 交换在同一台设备上
- 使用客户机上的一个路由器为定制功能，例如日志记录、LDAP 或者传统网络
- 冗余路由器更加简单和便宜

托管中心

- 通过 RouterOS 的虚拟化技术配合各种网络功能，如各种服务 Mail、Http、Ftp 等
- 提供虚拟路由器的 VPN 解决方案，这样能使网络管理员拥有自己的路由器，在高速骨干网络建立各种隧道或者 VPN 访问系统

无线 ISP 客户端

- 设置两个独立的路由器，并设置 WISP 的无线控制，并交由以太网端的客户进行控制

多客户端（例如办公楼）

- 分布在多个点的客户从一个骨干以太网连接（有线或无线），让每个客户能控制自己的独立的虚拟路由器，并配置自己的办公的路由。

网络规划与测试

- 建立一个虚拟的网络在一台设备上，相同环境的可对一个网络测试计划配置，进行微调，起到到在实验室的作用，而不需要在外假设，通过脚本和 The Dude 网络管理起模拟和监测网络

33.2 KVM 介绍

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文件。镜像文件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。

AMD 支持情况

在 2006 年 5 月 23 日，AMD 发行的 Athlon 64 ("Orleans"), Athlon 64 X2 ("Windsor") 和 Athlon 64 FX ("Windsor")首先支持这个技术

AMD-V 功能同样在 Athlon 64 and Athlon 64 X2 家族的“F”或“G”的 AM2（非 939 接口），Turion 64 X2 和第二代和第三代 Opteron，Phenom 与 Phenom II 处理器。只有 Sempron 处理器的 Sable 和 Huron 不支持 AMD-V.

Intel 支持情况

从 2009 年开始不是所有的 Intel 处理支持 VT-x 技术，VT-X 有不同的变化在相同型号，下面部分的列表是支持 VT-x，完整的列表请到 Intel 网页查询：

- Pentium 4 662 与 672
    - Pentium Extreme Edition 955 与 965 (非 Pentium 4 Extreme Edition HT)
- Pentium D 920-960 除 945, 935, 925, 915
- Core 2 Duo E6300, E6400, E6320, E6420, E6540, E6550, E6600, E6700, E6750, E6850 (Conroe)
- Core 2 Duo E5400, E7600, E8200, E8300, E8400, E8500, E8600 与一些 E7400 与 E7500 (Wolfdale)
- Mobile Core 2 Duo T5500, T5600, T6670, T7100, T7200, T7250, T7300, T7400, T7500, T7600G, T7700, T7800, U7500, L7200, L7300, L7400, L7500, L7700, U7500, U7600, U7700 (Merom)
- Mobile Core 2 Duo SU7300, SU9300, SU9400, SU9600, SL9300, SL9380, SL9400, SL9600, SP9300, SP9400, SP9600, P7370, P7550 , P7570, P8400, P8600, P8700, P8800, P9500, P9600, P9700, T8100, T8300, T9300, T9400, T9500, T9550, T9600, T9800, T9900 (Penryn)
- Core 2 Quad Q6600, Q6700 (Kentsfield)
- Core 2 Quad Q8400, Q8400S, Q9300, Q9400, Q9400S, Q9450, Q9550, Q9550S, Q9650 与一些 Q8300 (Yorkfield)
- Core 2 Extreme X6800 (Conroe XE)
- Core 2 Extreme QX6700, QX6800, QX6850 (Kentsfield XE)
- Core 2 Extreme QX9650, QX9770, QX9775 (Yorkfield XE)
- Xeon 3300 and +, 5000, 7000 series
- Atom Z520, Z530, Z540, Z550 (Silverthorne)

- 所有 Intel Core i3 processors
- 所有 Intel Core i5 processors
- 所有 Intel Core i7 processors
- Pentium Dual-Core E6300, E6500, E6600 与一些 E5300 与 E5400
- Celeron SU2300, E3200, E3300, E3400

在一些主板上，Intel 的 VT-x 功能必须在 BIOS 中启用

RouterOS 要求：安装 KVM 功能包，至少需要 system 与 KVM

配置

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

33.3 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 目录下给出了一下命令：

- 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：

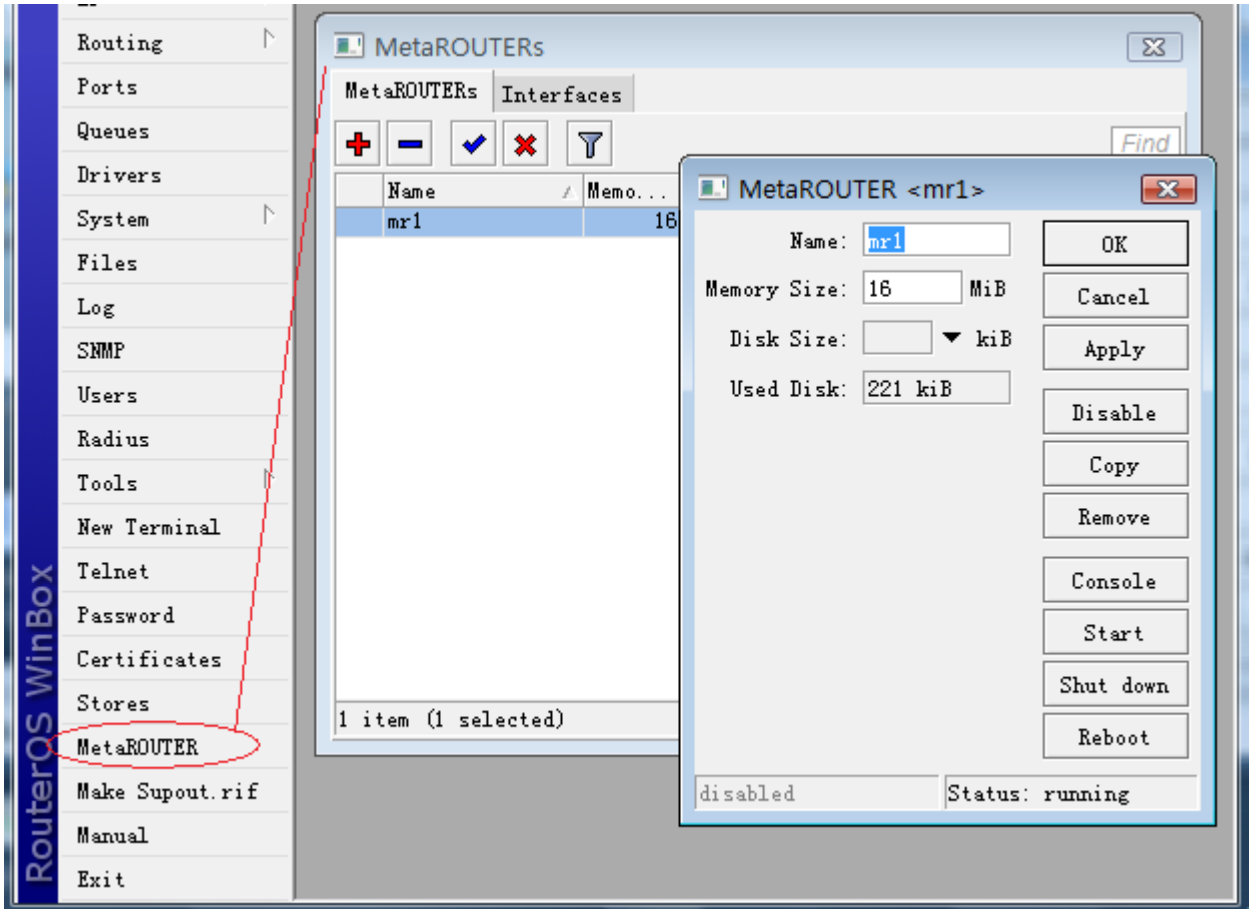
```
[admin@mr0] > interface print
 Flags: D - dynamic, X - disabled, R - running, S - slave
 #       NAME                                                    TYPE              MTU
 0   R   ether1                                                  ether             1500
```

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!
```

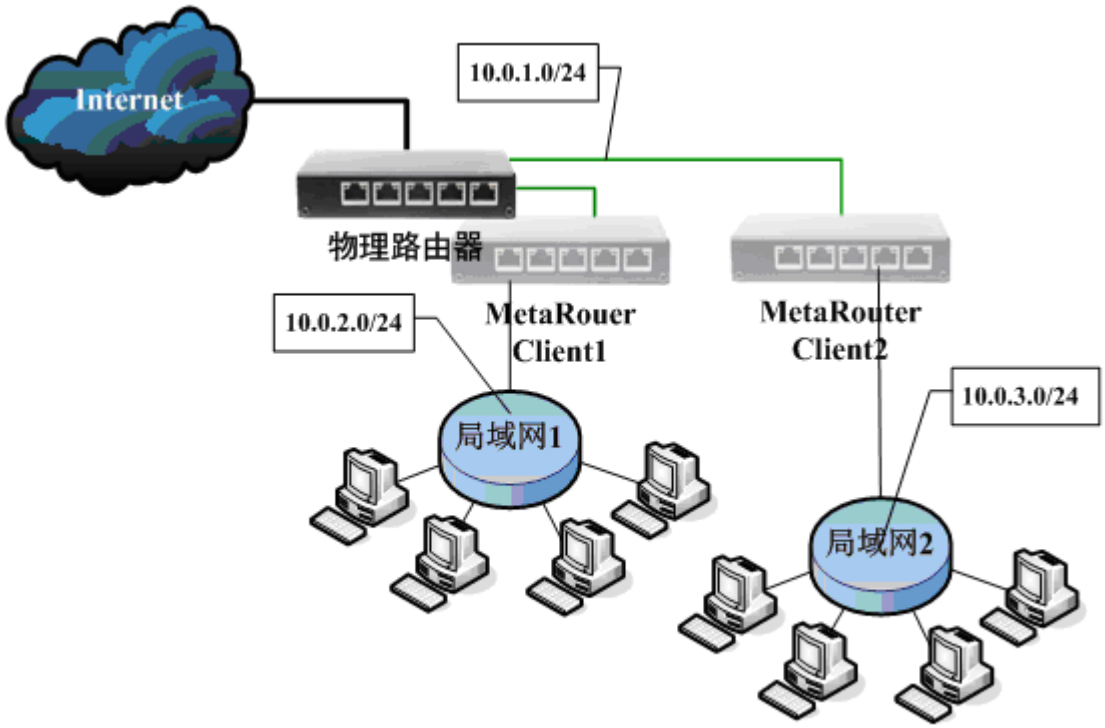
33.4 MetaRouter 事例

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 操作界面



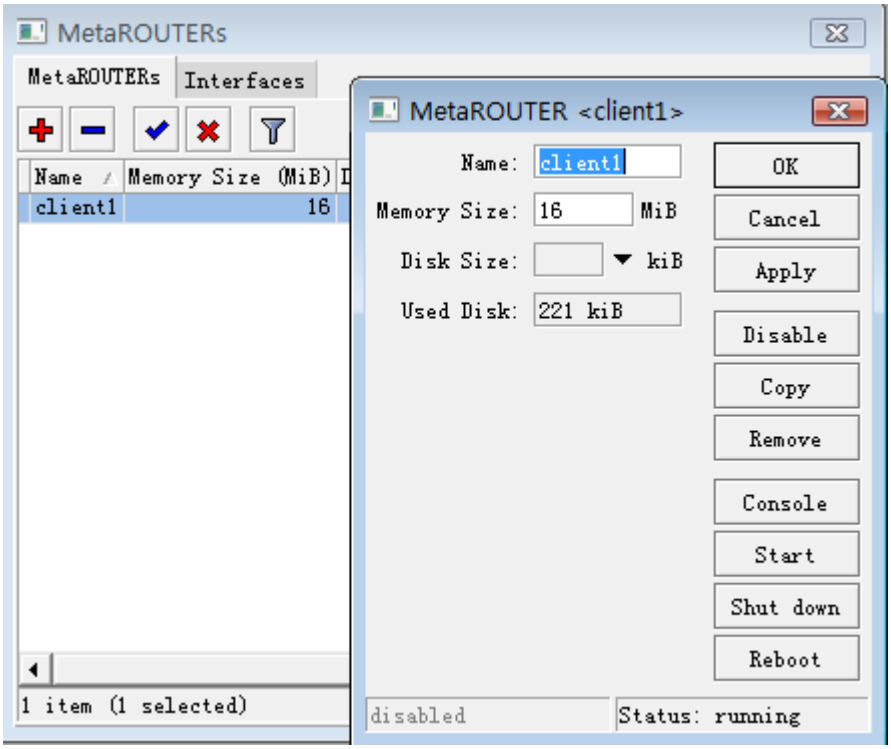
这个事例将介绍如何配置 MetaRouter 功能，为局域网内部独立 RouterOS 虚拟路由，基于 RB450 配置 MetaRouter，我们将建立两个 Client 虚拟路有，并管理两个不同的局域网。目的是让两个个局域网的管理员可

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）：



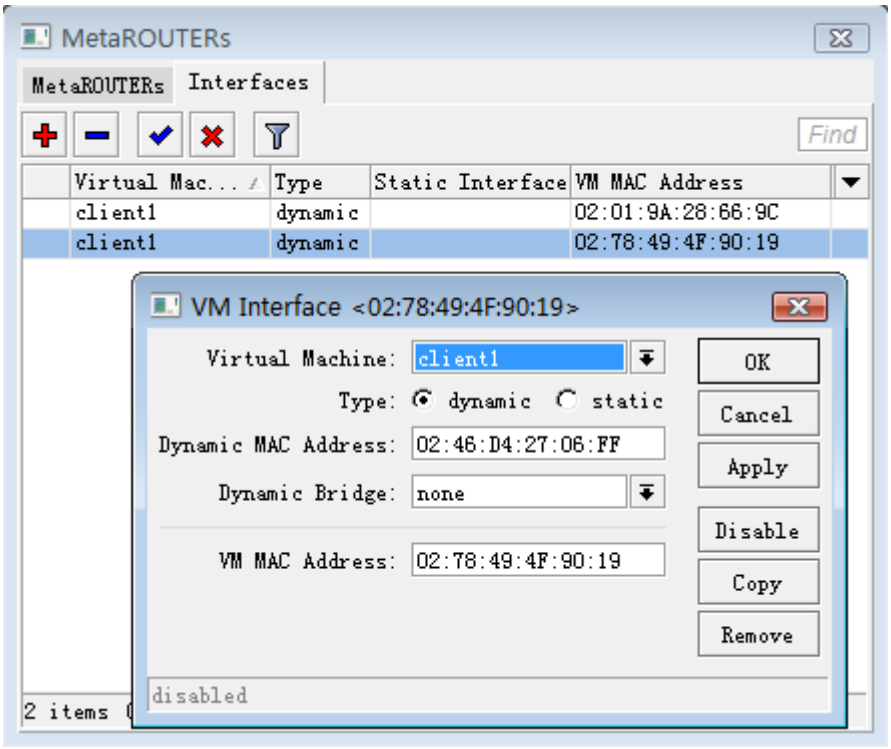
1.给客户添加一个 MetaRouter：

```
[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>
```



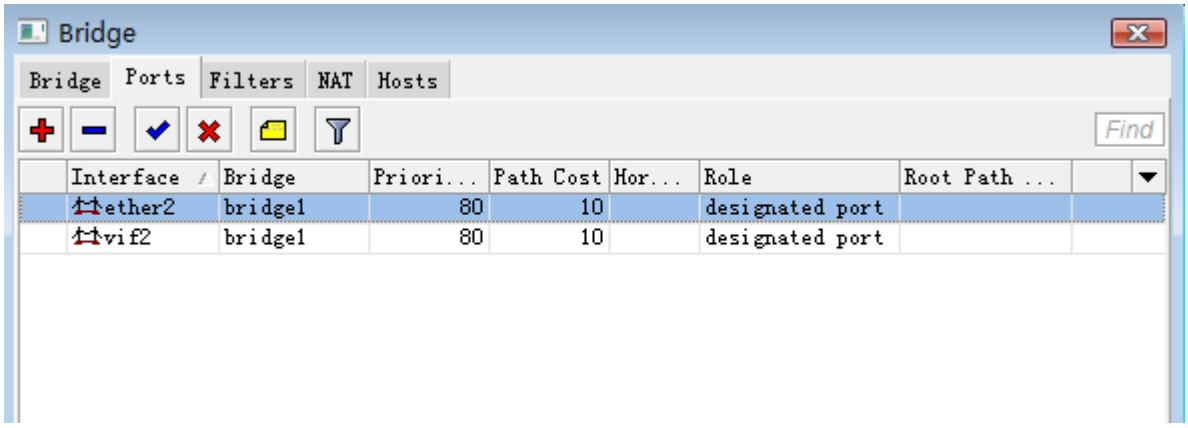
2. 添加 MetaRouter 虚拟机的接口：

```
[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>
```



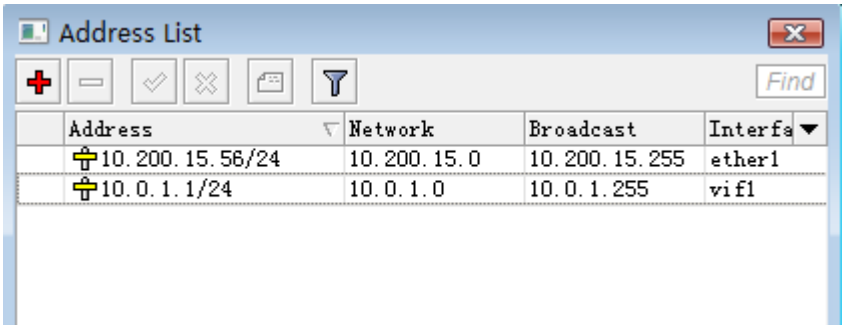
3．创建一个桥接口，将 MetaRouter 接口与以太网接口桥接，这里我们将 vif2 的虚拟接口放入内网的桥中，用于客户端通过物理接口连接(使用桥接目的是将虚拟路由的内网接口，通过桥接透穿到真实的网络中)：

```
[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
```

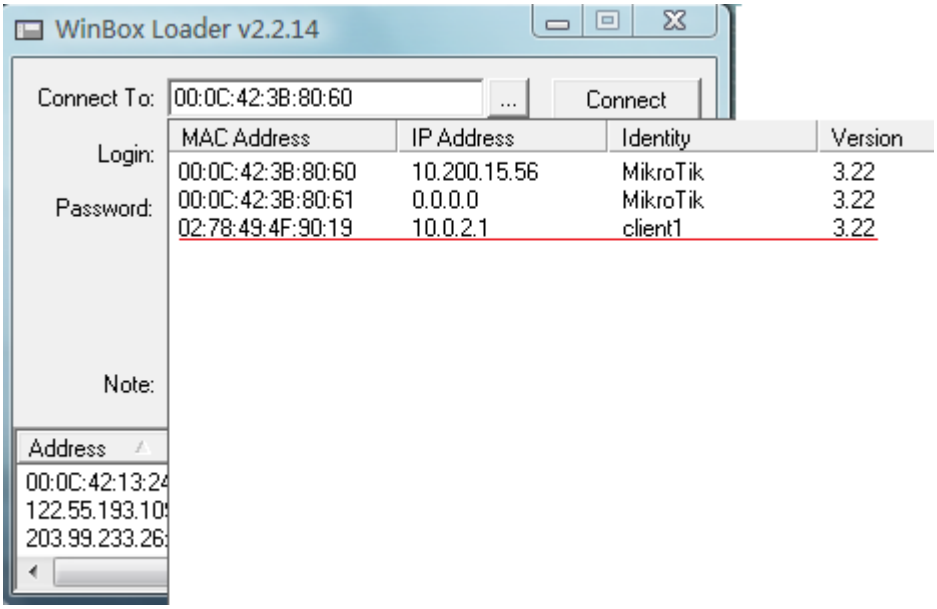
添加默认网关

```
[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>
```

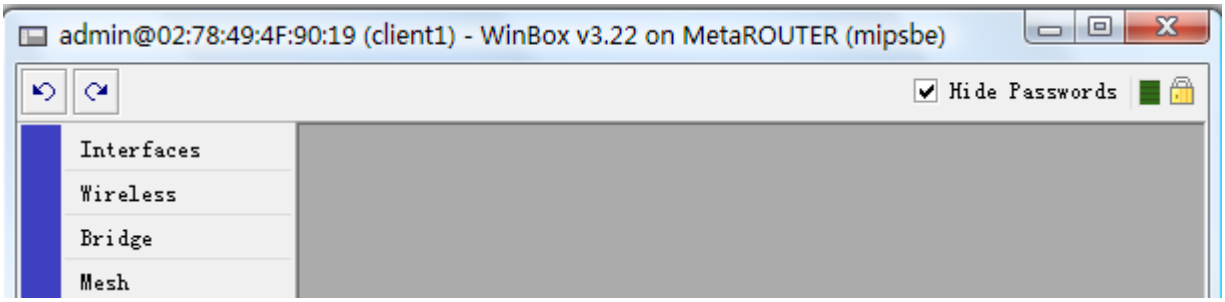
配置 nat 转换

```
[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

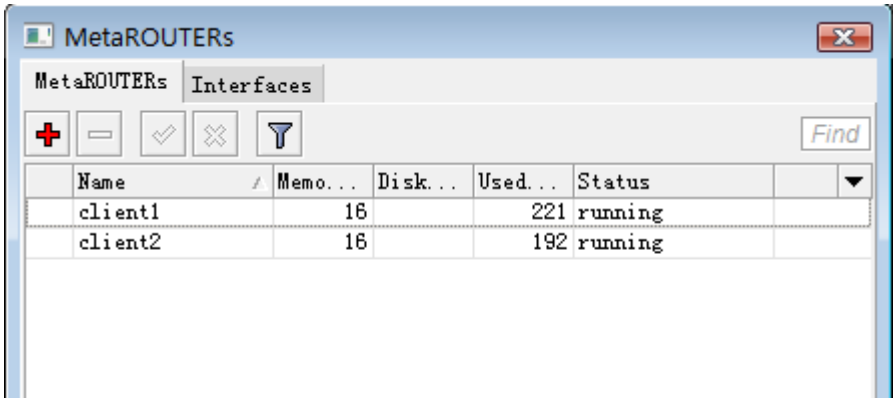


通过连接后，在 winbox 中显示 MetarRouter 信息



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：



33.5 CHR(Cloud Hosted Router)

MikroTik 推出了一项新功能 Cloud Hosted Router（CHR），是一个包含所有功能的 RouterOS 的虚拟机镜像，且没有像 RouterOS 那些的等级许可限制，但网络接口速率被限制，1Mbit 的网络接口是免费使用的，也提供不速率的网络接口，但需要有偿购买。MikroTik 对 CHR 的特点定义为：首先赋予了 x86 平台的新生命，其次支持 64bit 构架，能运行在 VMWare, Hyper-V, VirtualBox, KVM。

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 64 位构架，而传统 RouterOS 在 x86 硬件平台上则没有，还是 32bit 构架，这点应该和硬件平台多样化，要求兼容驱动过多等因素有关系，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。

MikroTik 提供一个通过格式镜像（RAW），能通过 qemu 或其他工具转换为其他格式。当前 MikroTik 已经提供了 VMDK (VMware virtual 磁盘镜像) 和 VDI (VirtualBox 磁盘镜像)。下面是不同许可的网口速率和价格：

许可	速率	价格
Free	1Mbit	免费
P1	1Gbit	$45
P10	10Gbit	$95
P-Unlimited	无限制	$250

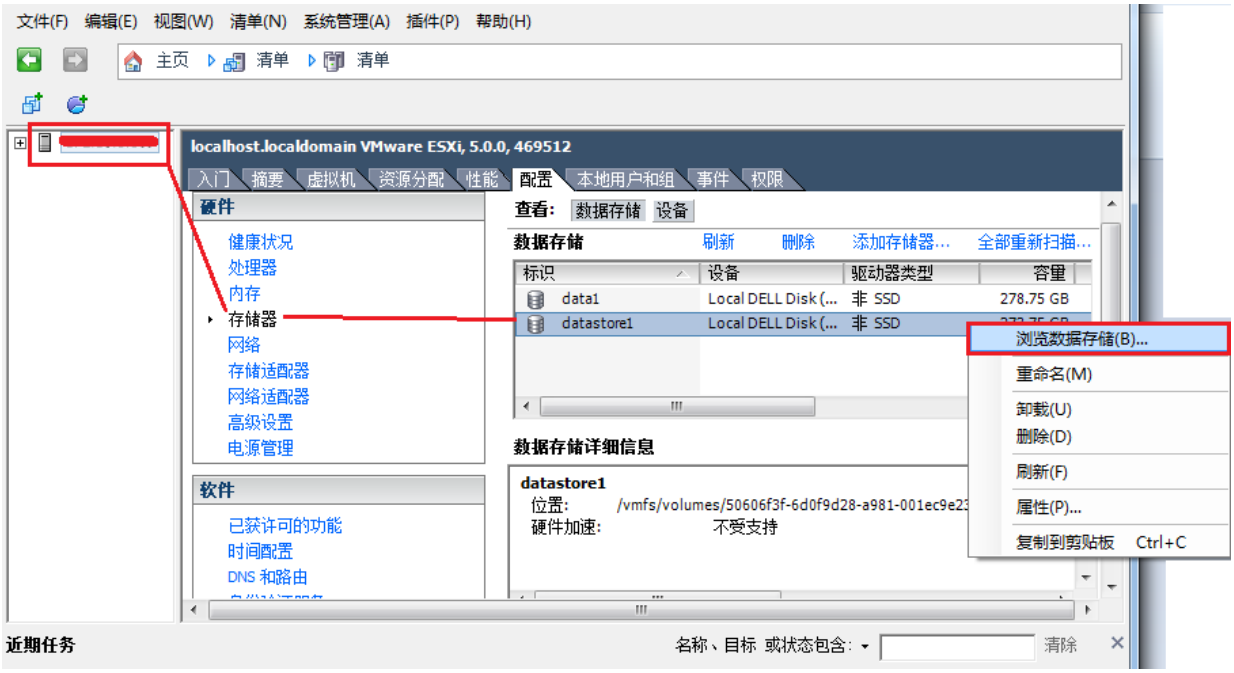
从免费版本的 CHR 看其实做了一个类似思科和华为的模拟器，让大家可以学习研究 RouterOS 做各种实验，而其他几个付费版本，就是顺应现在的虚拟化潮流，在虚拟机上搭建 RouterOS 平台，虽然我们可以把当前的 RouterOS 移植到这些虚拟平台上，但毕竟不是像 CHR 那样针对虚拟化平台设计的，而且 CHR 支持 64bit 构

架，不管兼容性，稳定性和性能上在虚拟化平台 CHR 都要比安装在 VM 一类的 RouterOS 强。当然要去不高的朋友，也可以继续使用安装在 VM 一类平台的 RouterOS。

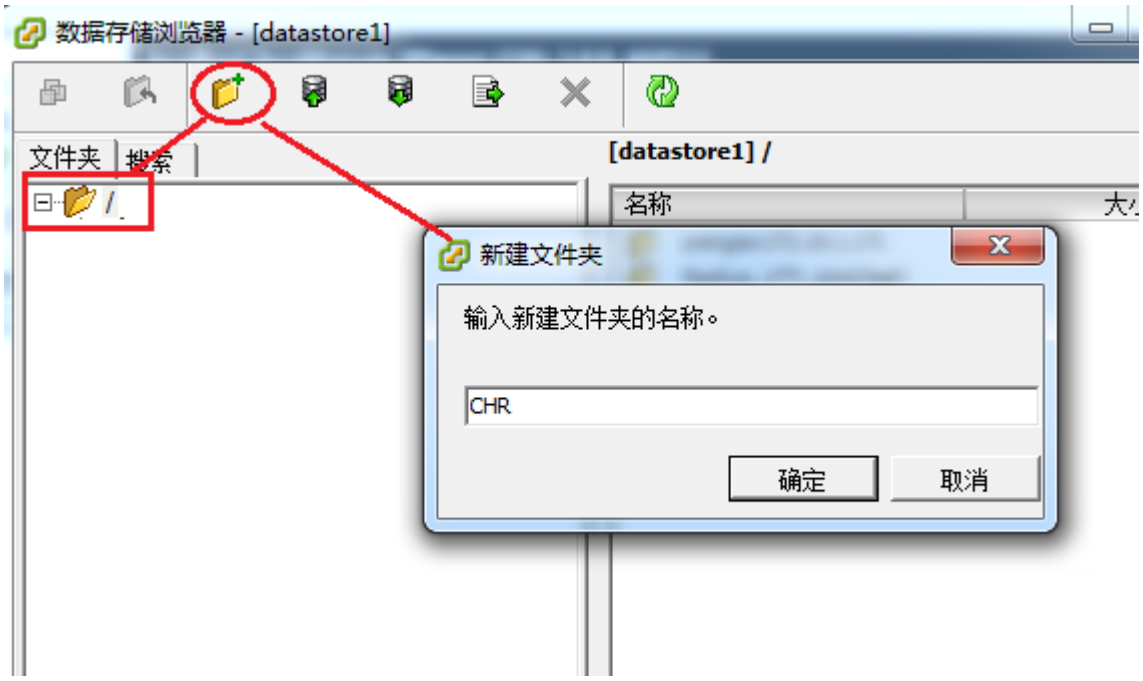
Cloud Host Router 安装到 VM EX 平台

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 的镜像文件 http://www.mikrotik.com/download/share/chr_6_31rc9.vmdk（此链接仅供参考）官方下载链接请访问 www.mikrotik.com/download

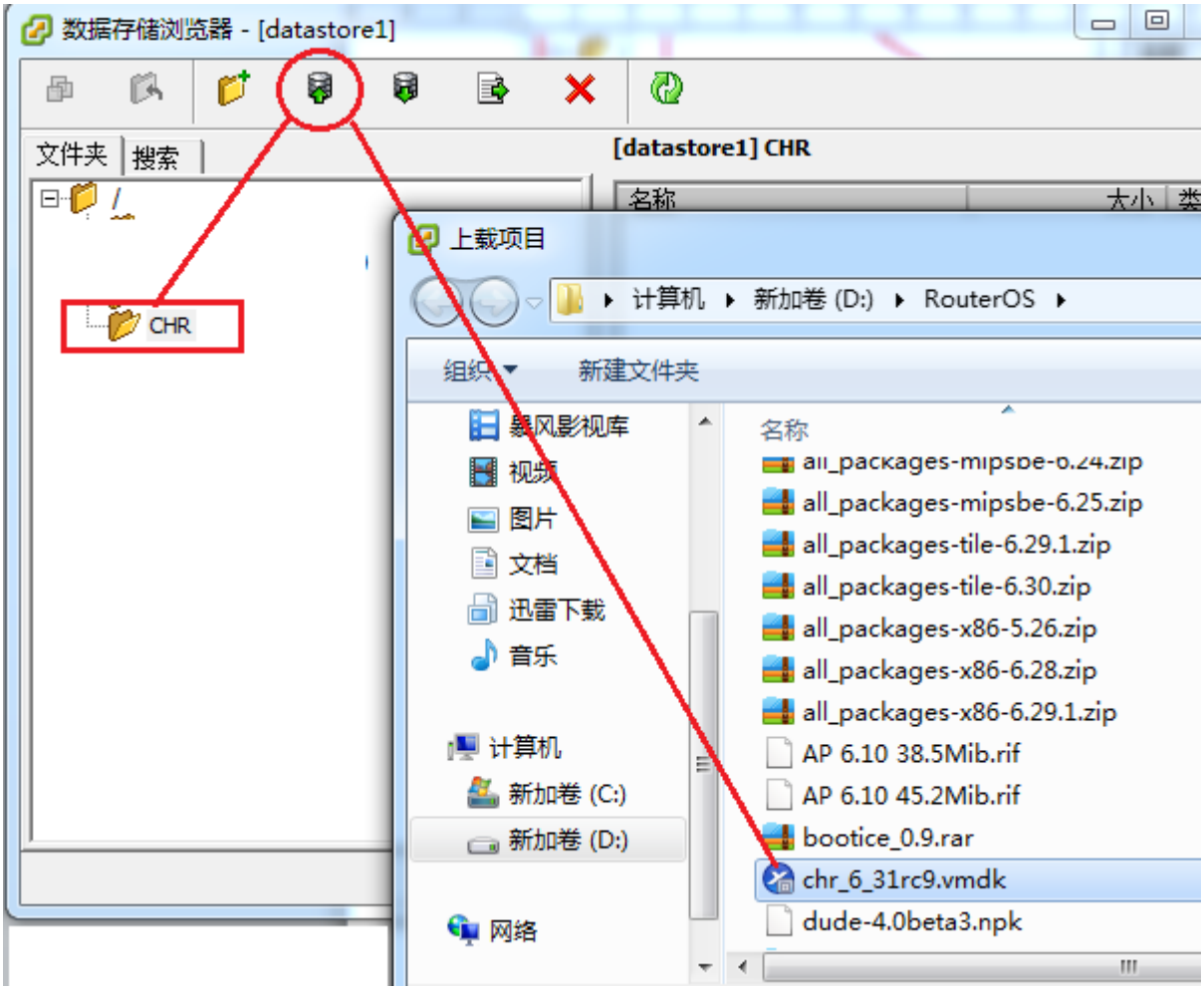
下载完成后，通过 vsphere client 登录 Vmware esxi，选择虚拟主机，并进入配置菜单，打开下面的存储器，选择一块 data 数据盘，右键选择“浏览数据存储”



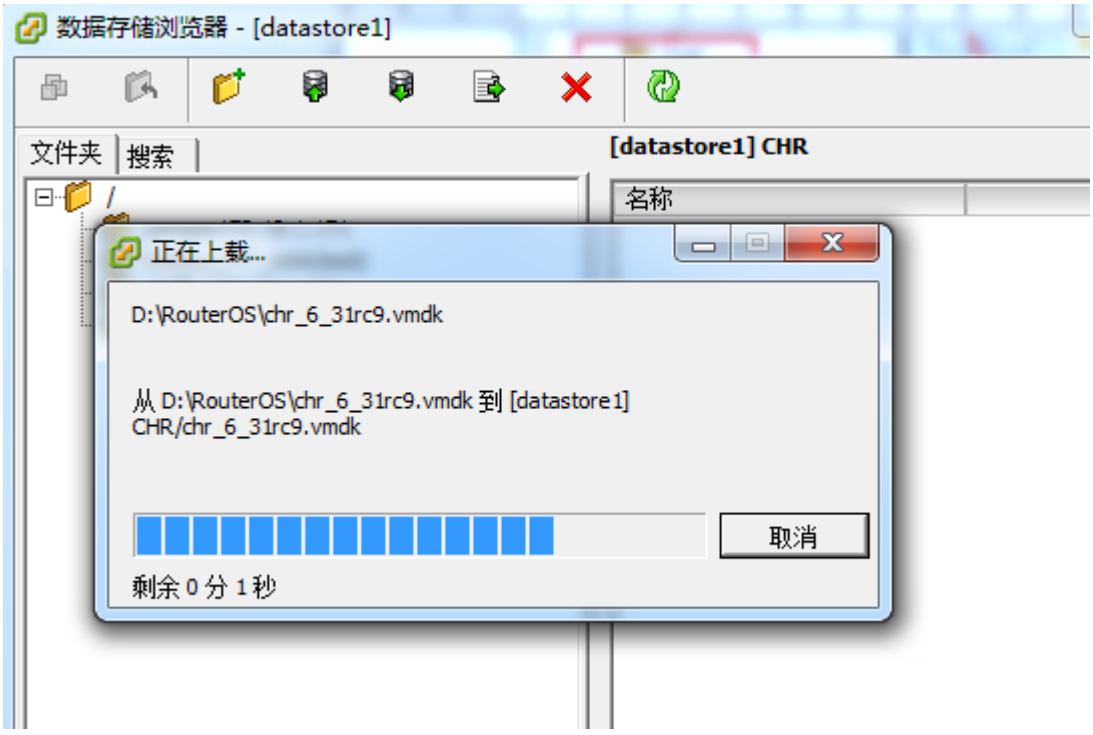
打开后，选择根目录，新建一个文件夹，取名 CHR

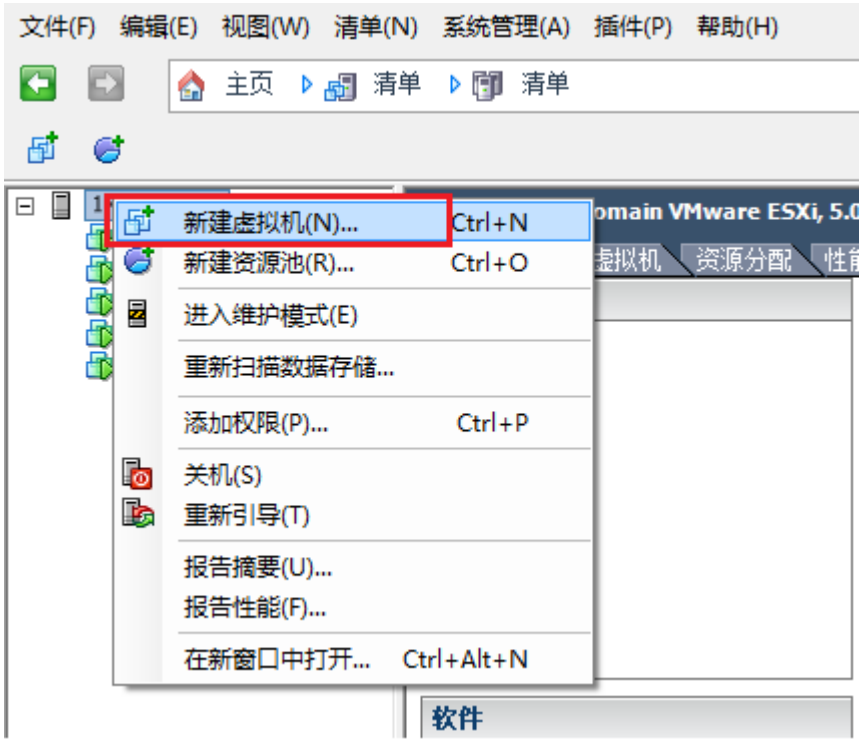


打开 CHR 文件夹，然后上传下载好的 chr_6_31rc9.vmdk 文件

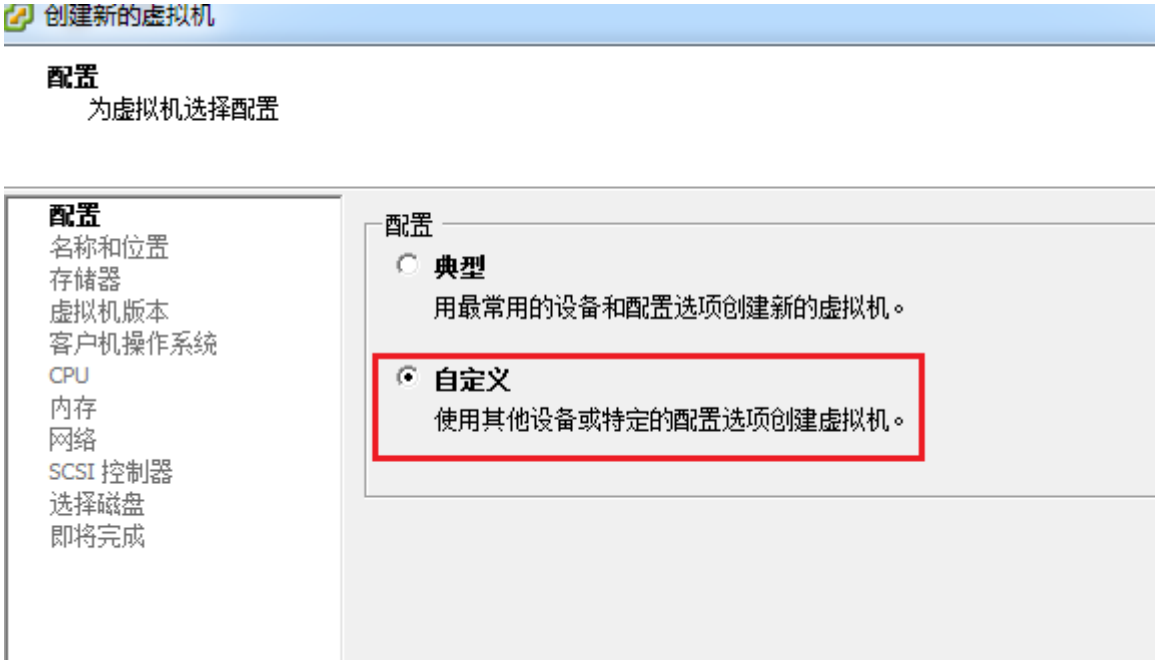


确认开始上传

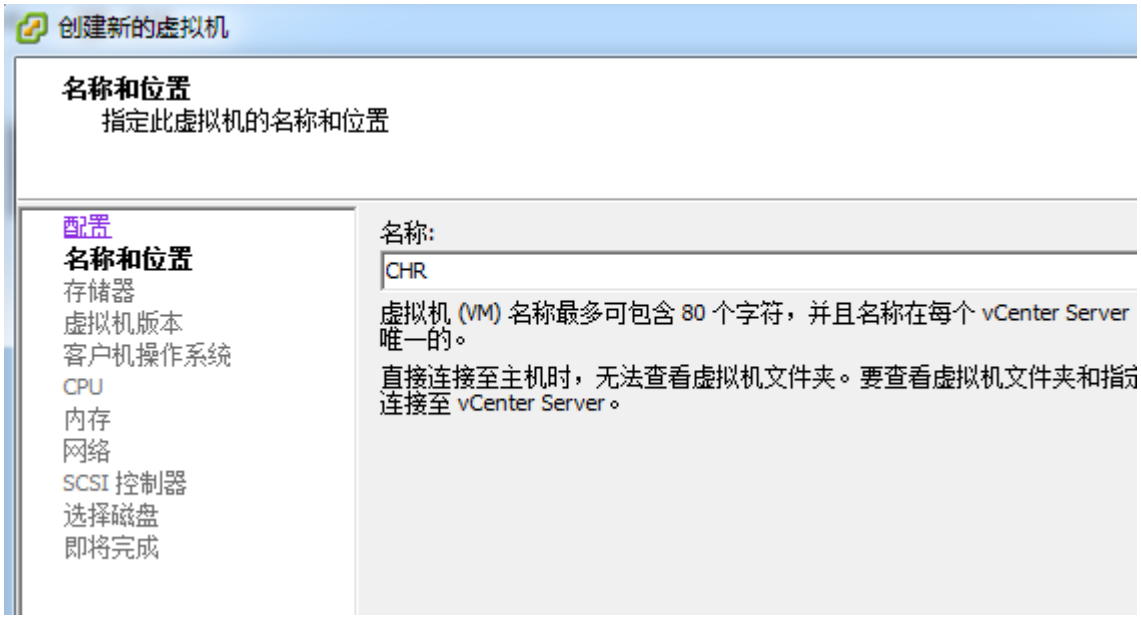




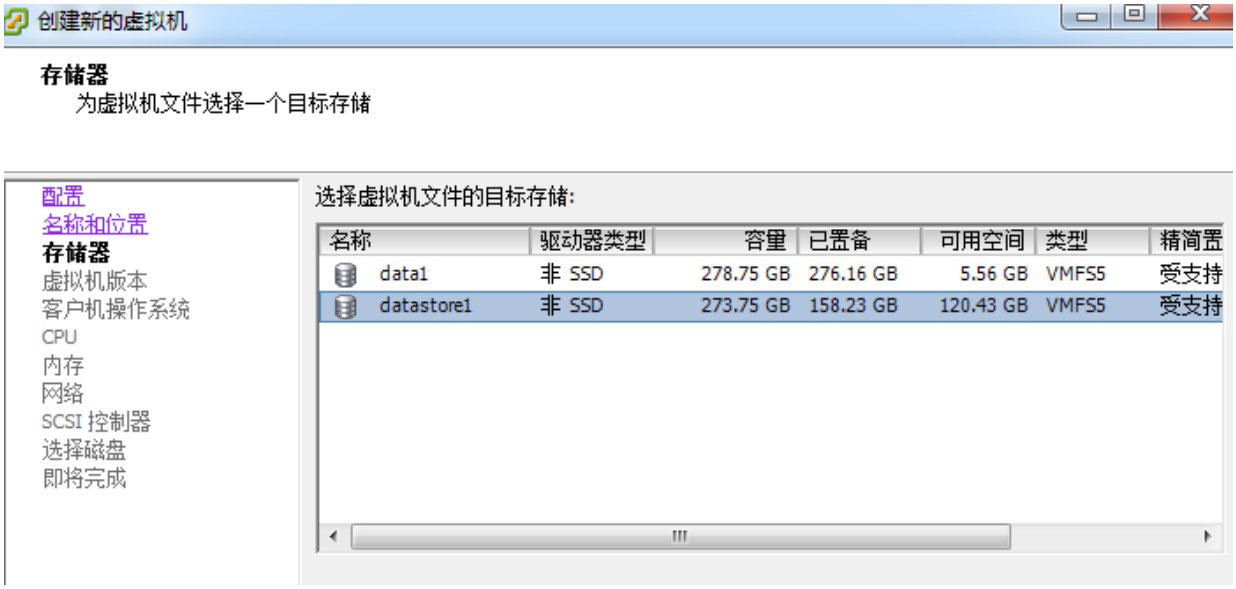
选择自定义，然后下一步

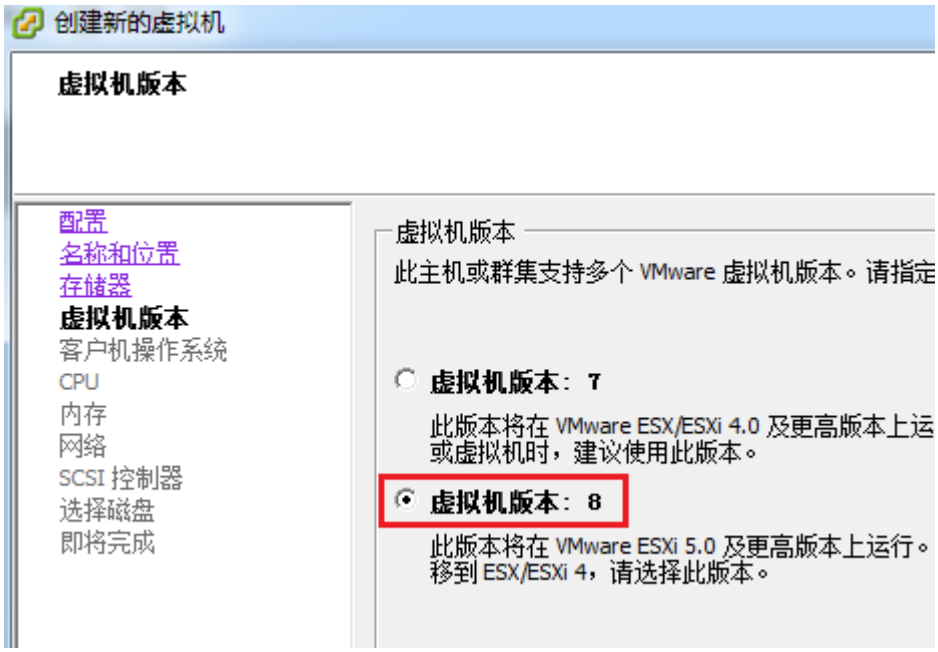


取名可以随意：


创建新的虚拟机
名称和位置
指定此虚拟机的名称和位置
配置
名称和位置
存储器
虚拟机版本
客户机操作系统
CPU
内存
网络
SCSI 控制器
选择磁盘
即将完成
名称:
CHR
虚拟机 (VM) 名称最多可包含 80 个字符，并且名称在每个 vCenter Server 唯一的。
直接连接至主机时，无法查看虚拟机文件夹。要查看虚拟机文件夹和指定
连接至 vCenter Server。

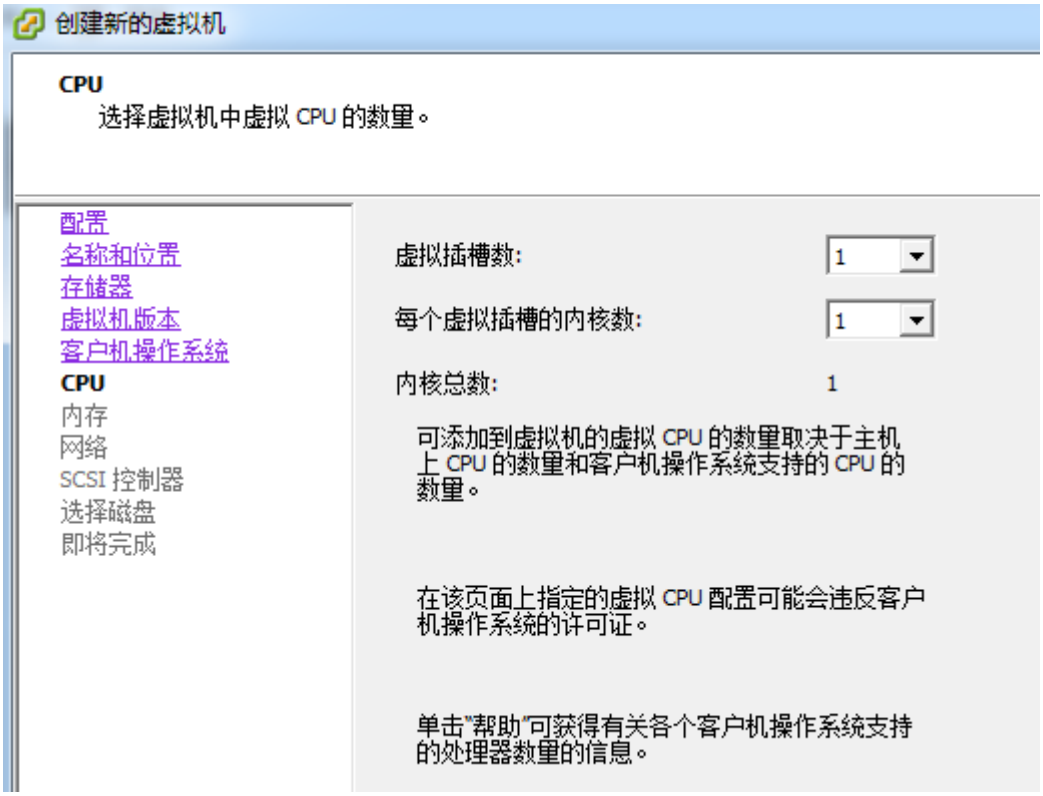
选择存储磁盘


创建新的虚拟机
存储器
为虚拟机文件选择一个目标存储
配置
名称和位置
存储器
虚拟机版本
客户机操作系统
CPU
内存
网络
SCSI 控制器
选择磁盘
即将完成
选择虚拟机文件的目标存储:
名称
驱动器类型
容量
已置备
可用空间
类型
精简置
data1
非 SSD
278.75 GB
276.16 GB
5.56 GB
VMFS5
受支持
datastore1
非 SSD
273.75 GB
158.23 GB
120.43 GB
VMFS5
受支持



选择操作系统 Linux，版本为：其他 linux（32 位），不过 CHR 支持是 64 位构架，这里演示仅供参考。

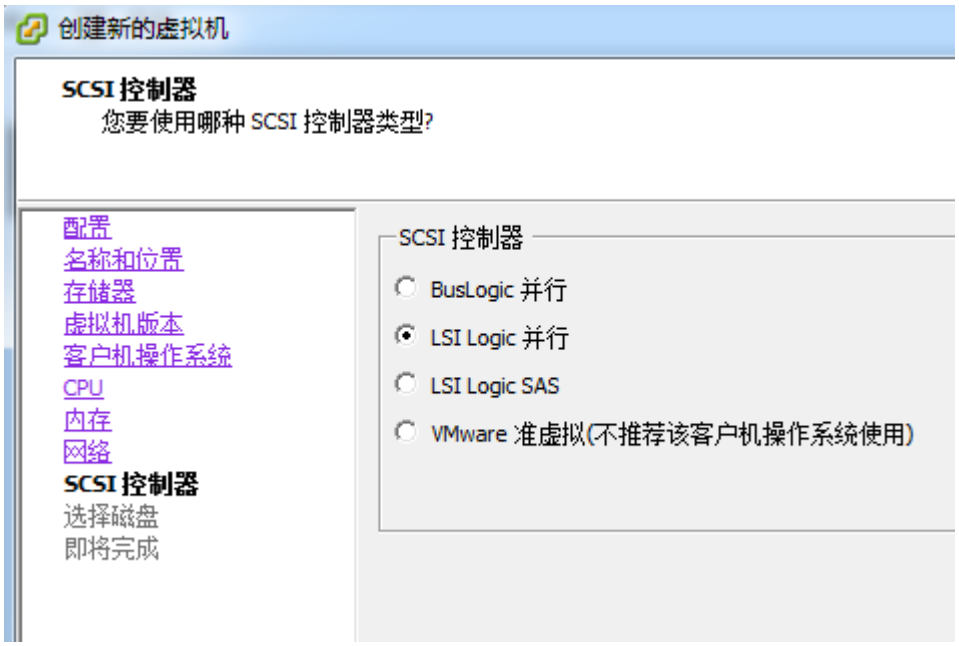
选择硬件配置


创建新的虚拟机
CPU
选择虚拟机中虚拟 CPU 的数量。
配置
名称和位置
存储器
虚拟机版本
客户机操作系统
CPU
内存
网络
SCSI 控制器
选择磁盘
即将完成
虚拟插槽数:
1
每个虚拟插槽的内核数:
1
内核总数:
1
可添加到虚拟机的虚拟 CPU 的数量取决于主机上 CPU 的数量和客户机操作系统支持的 CPU 的数量。
在该页面上指定的虚拟 CPU 配置可能会违反客户机操作系统的许可证。
单击"帮助"可获得有关各个客户机操作系统支持的处理器数量的信息。

创建新的虚拟机
内存
配置虚拟机的内存大小。
配置
名称和位置
存储器
虚拟机版本
客户机操作系统
CPU
内存
网络
SCSI 控制器
选择磁盘
即将完成
内存配置
1011 GB
512 GB
256 GB
128 GB
64 GB
32 GB
16 GB
8 GB
4 GB
2 GB
内存大小:
256
MB
为此客户机操作系统推荐的最大值(X): 64 GB。
为获取最佳性能推荐的最大值(R): 8188 MB。
为此客户机操作系统推荐的默认值(D): 256 MB。
为此客户机操作系统推荐的最小值(M): 32 MB。

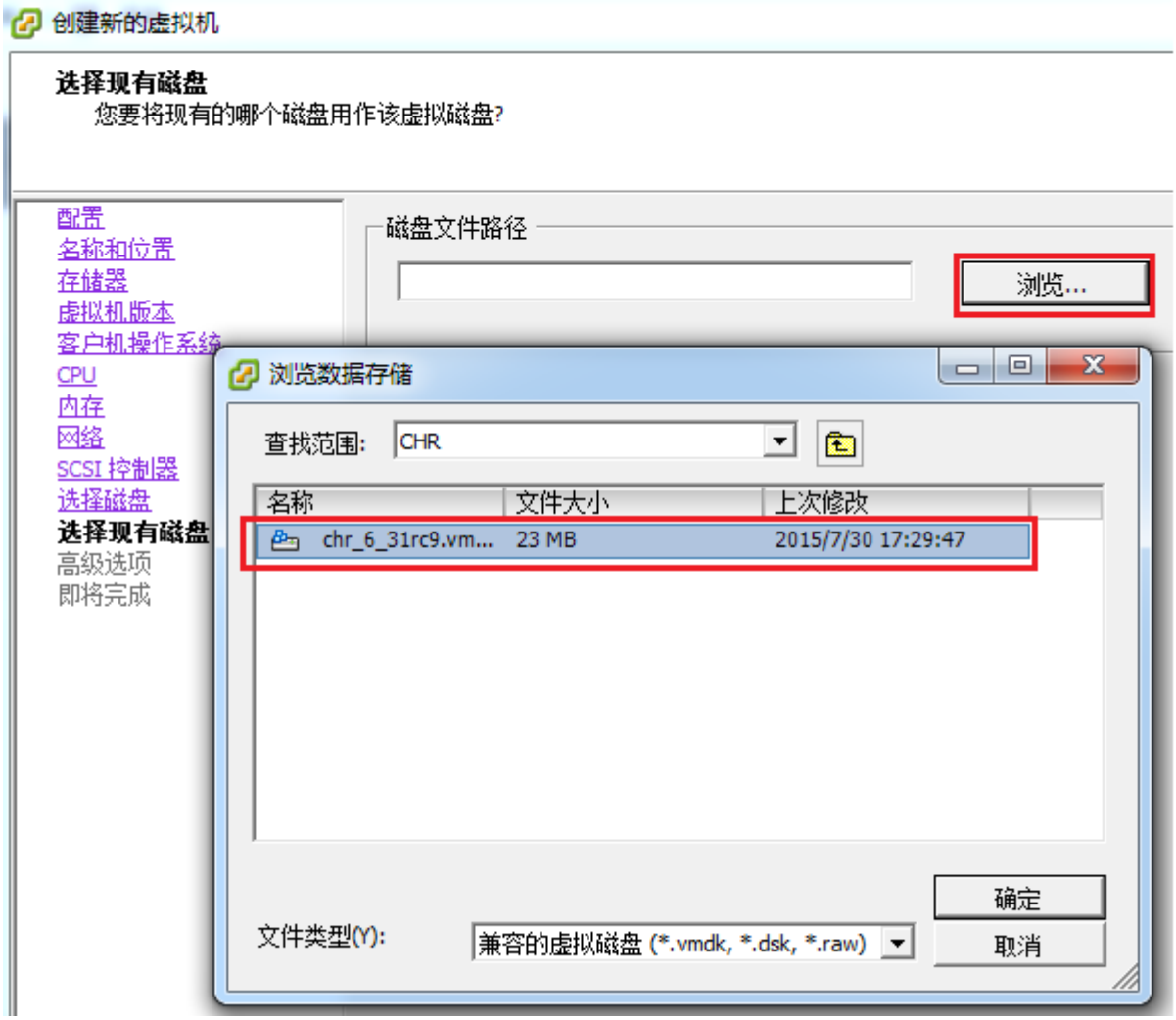
网络接口配置

下面是选择控制器



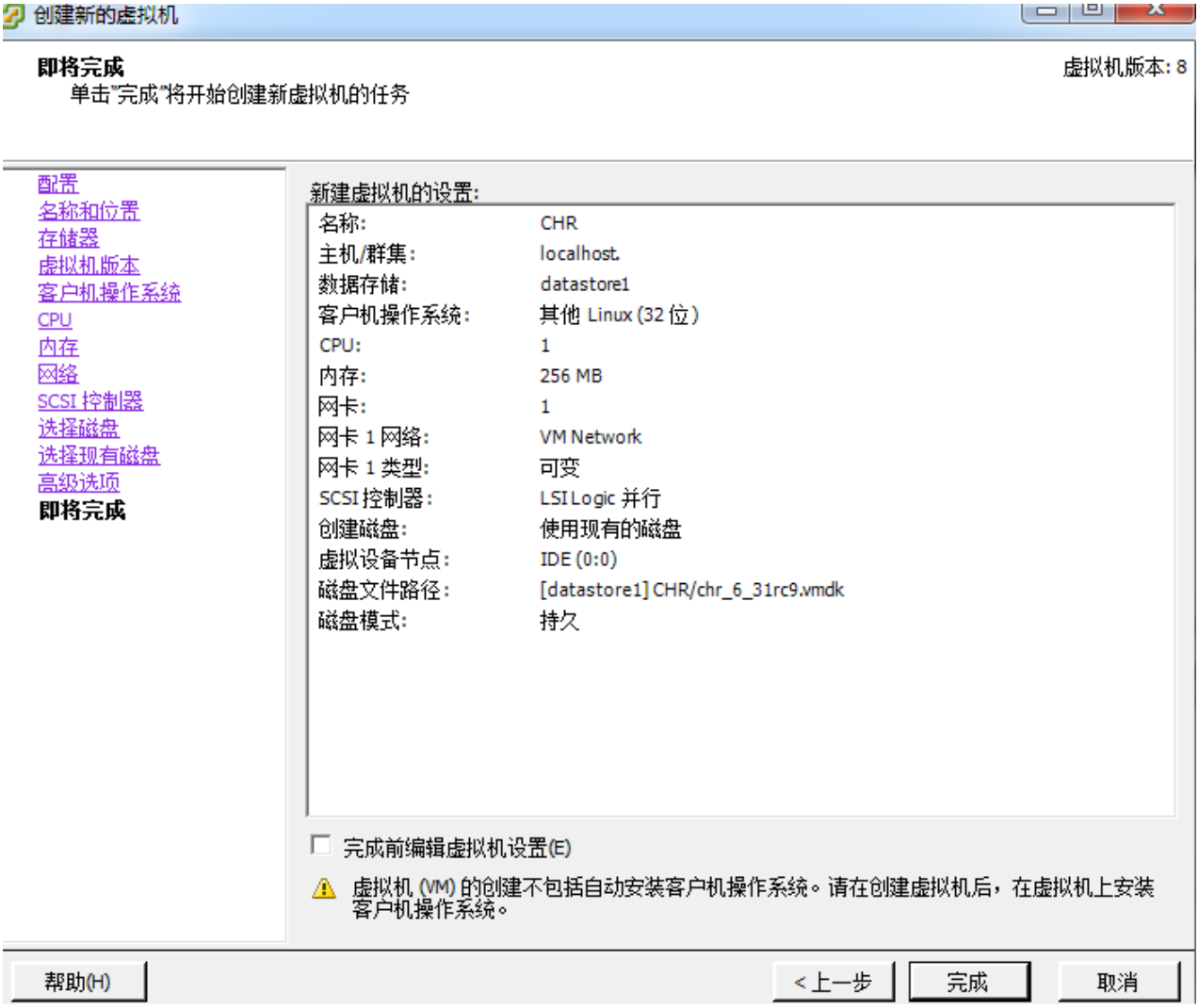
注意在选择磁盘的时候，我们需要选择“使用现有虚拟磁盘“

浏览磁盘文件路径，并加载我们的 CHR 文件

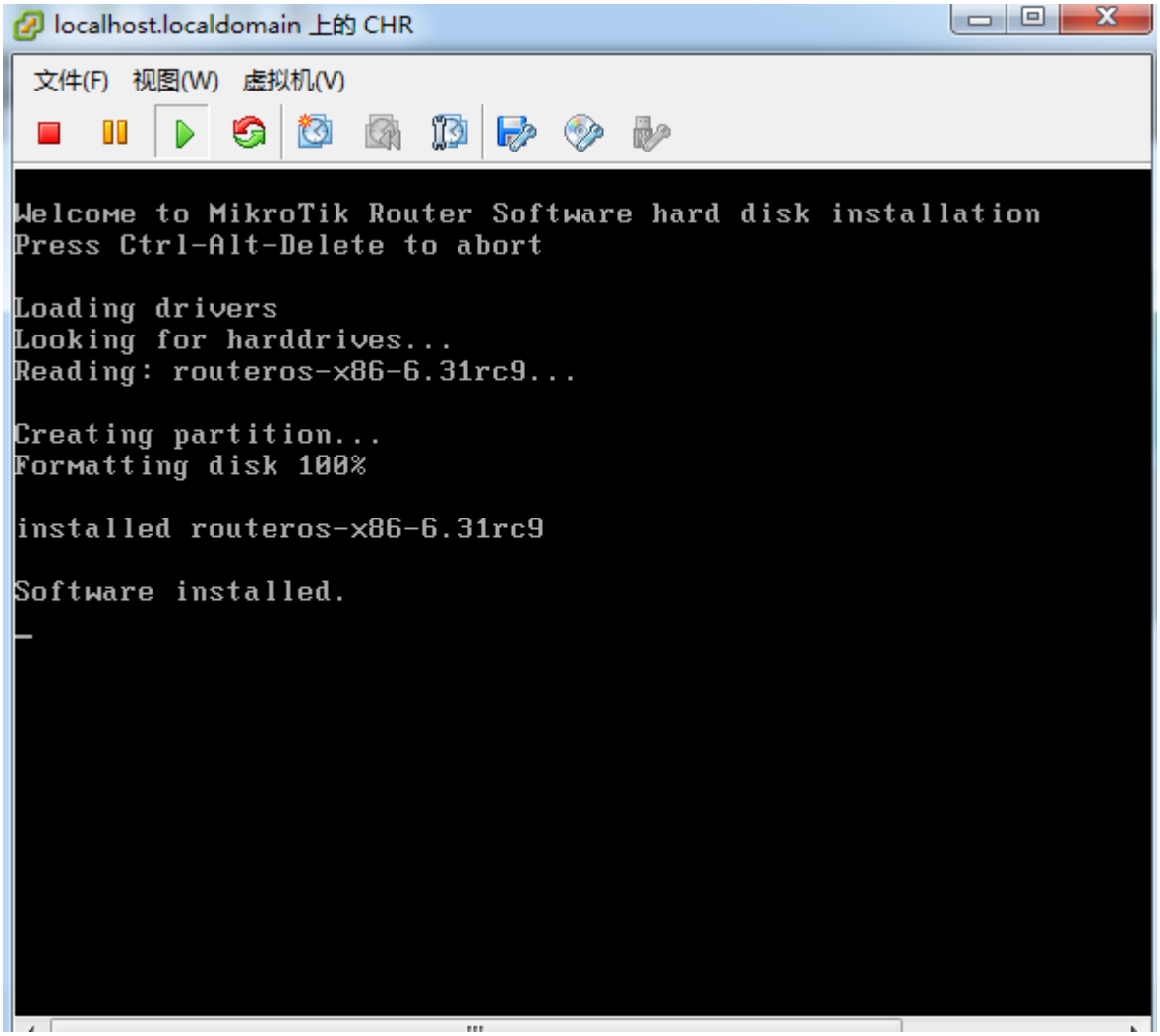


高级选项中，我们选择虚拟设备节点为 IDE(0:0)

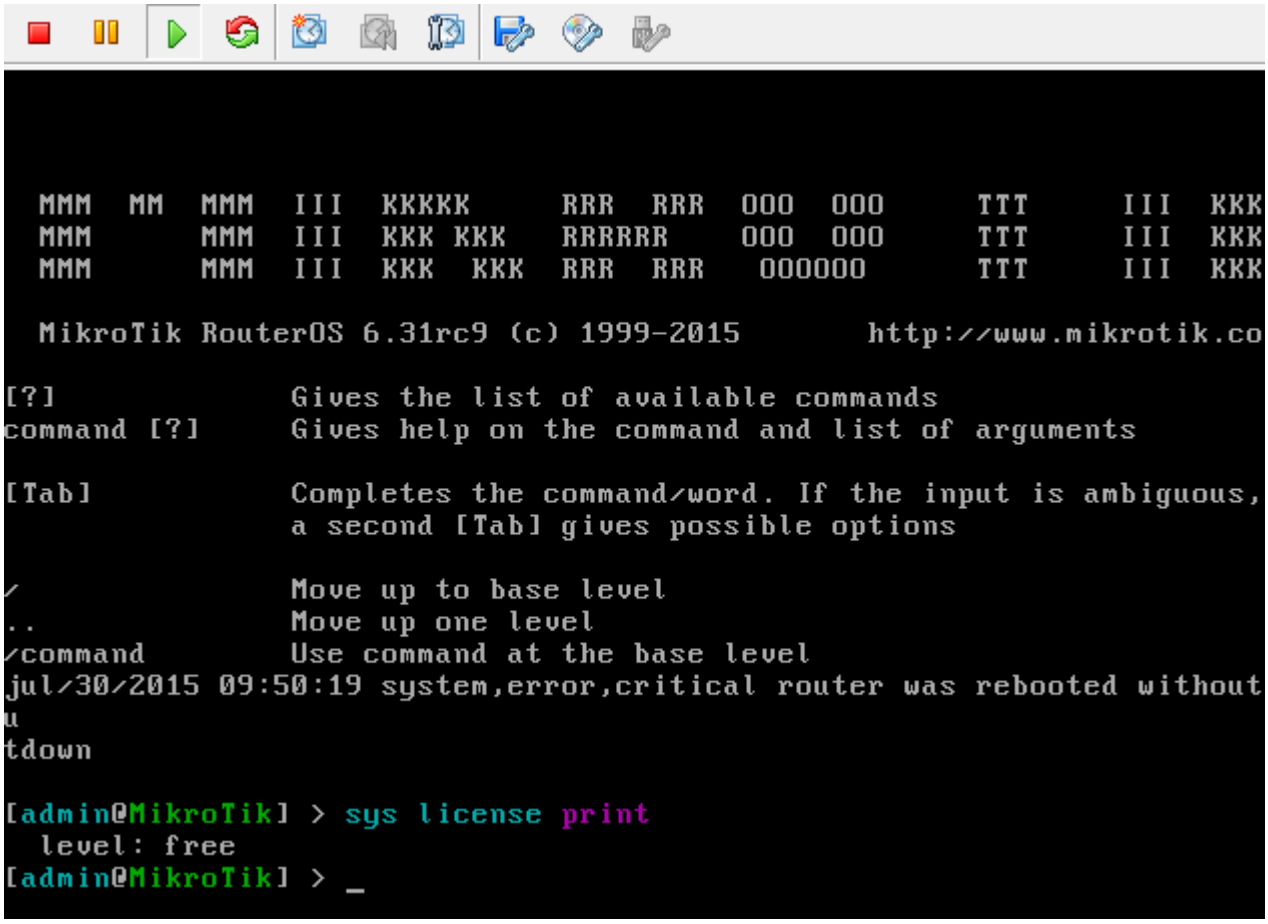
最后是创建完成



我们启动在 VM 的 CHR 虚拟机



启动完成后，进入界面，查看许可为 free


MMM MM MMM III KKKKK RRR RRR OOO OOO TTT III KKK
MMM MMM III KKK KKK RRRRRR OOO OOO TTT III KKK
MMM MMM III KKK KKK RRR RRR OOOOOO TTT III KKK
MikroTik RouterOS 6.31rc9 (c) 1999-2015 http://www.mikrotik.co
[?] Gives the list of available commands
command [?] Gives help on the command and list of arguments
[Tab] Completes the command/word. If the input is ambiguous,
a second [Tab] gives possible options
/ Move up to base level
.. Move up one level
/command Use command at the base level
jul/30/2015 09:50:19 system,error,critical router was rebooted without
u
tdown
[admin@MikroTik] > sys license print
level: free
[admin@MikroTik] > _

第三十四章 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、数据包和字节）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

34.1 IP 访问记录

操作路径: ***/ip accounting***

当每个包通过路由器时，匹配 IP 数据包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。数据包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。更经常的数据会被收集。

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>
```

34.2 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

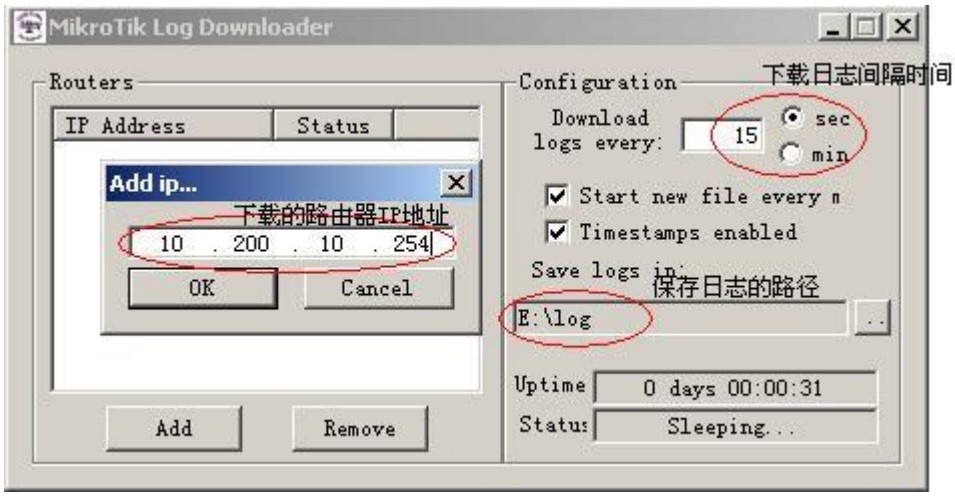
accessible-via-web (yes | no; 默认: **no**) – 是否 snapshot 通过 web 可用
address (*IP 地址*/子网掩码; 默认: **0.0.0.0**) -允许存取 snapshot 的 IP 地址范围

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

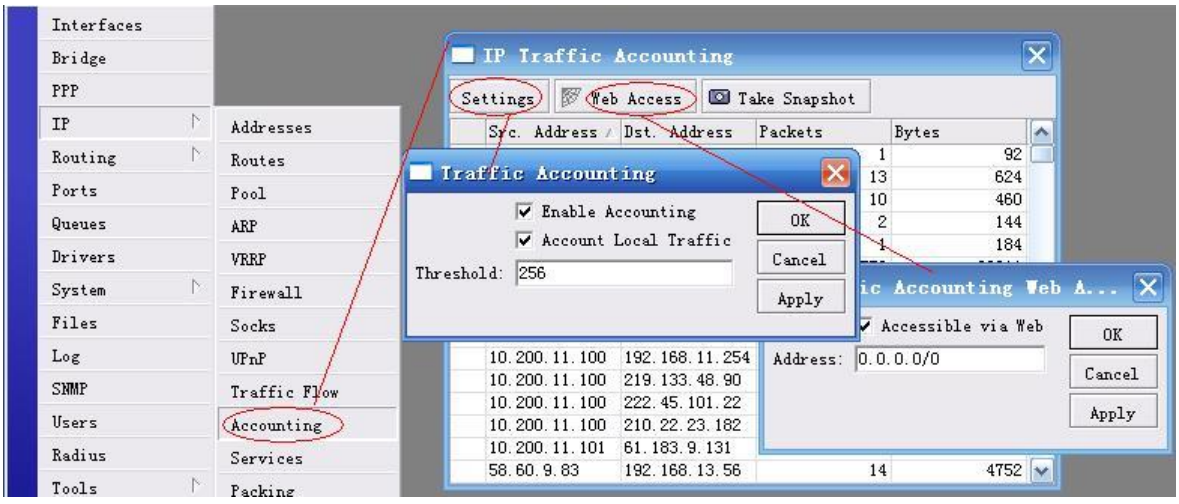
```
[admin@MikroTik] ip accounting web-access> set accessible-via-web=yes \
\... address=192.168.10.10/32
[admin@MikroTik] ip accounting web-access> print
      accessible-via-web: yes
                 address: 192.168.10.10/32
[admin@MikroTik] ip accounting web-access>
```

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：

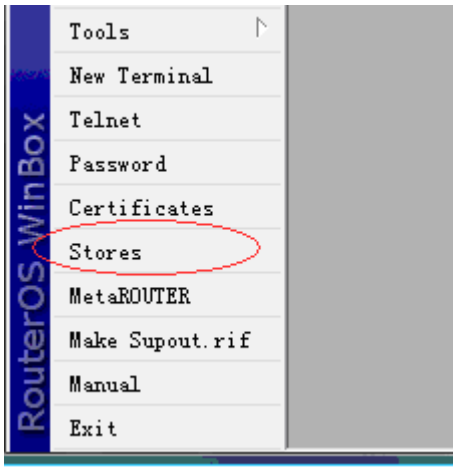


注：该软件只能通过 RouterOS 的 web 端口记录，即 web 端口默认必须是 80，最近在 bbs.routerclub.com 的论坛上有人发一个自己开发的日志记录软件解决了其他端口记录的问题。

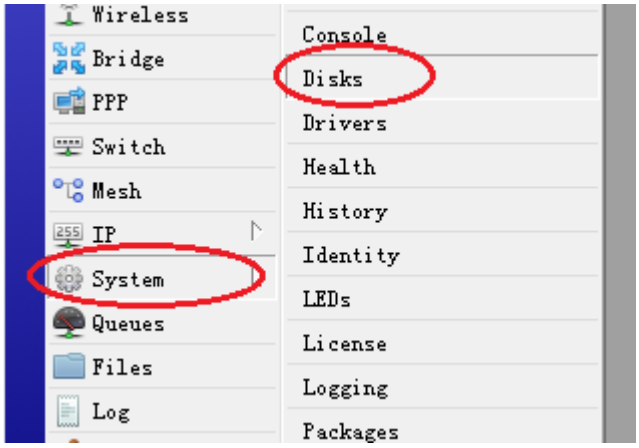
第三十五章 RouterOS Stores/disk 功能

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，



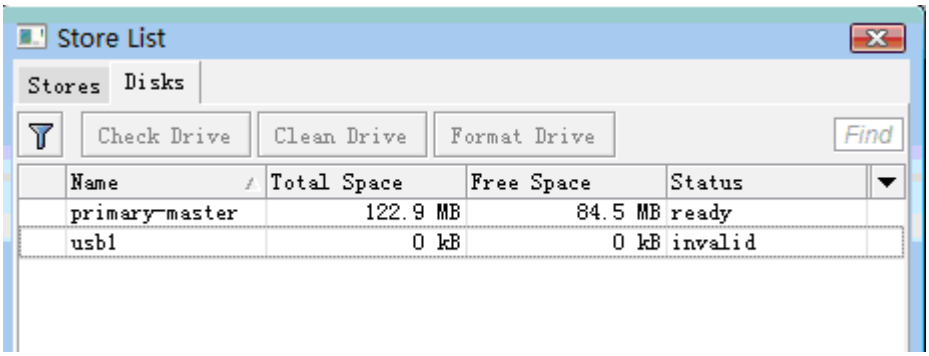
6.20 以前



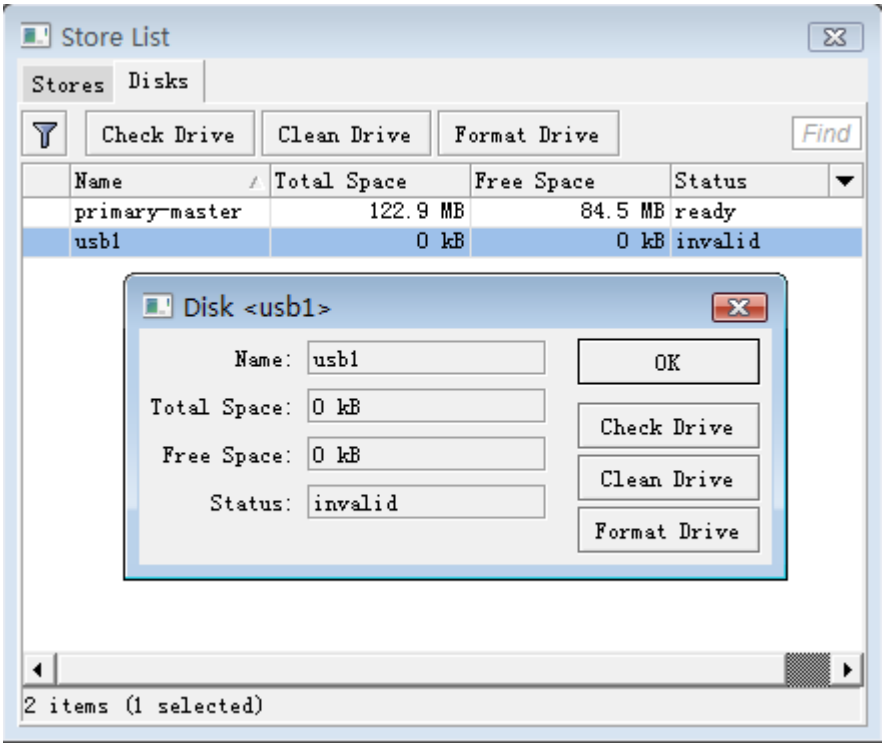
6.20 以后

35.1 RouterOS 使用 U 盘扩展存储

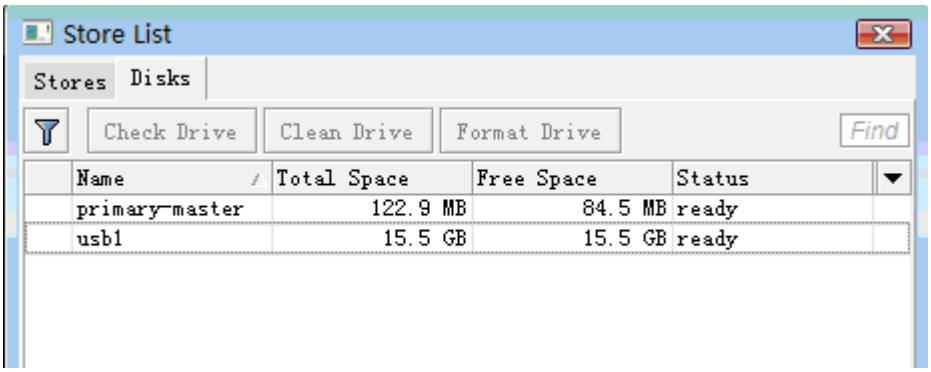
这里我们通过使用 U 盘来演示，在 PC 上增加外部存储的操作，我们将 1 个 16G 的 U 盘，插入 RouterOS PC 的 USB 接口，这个时候，我们可以在 Store 的 Disk 目录中找到 usb1 的硬盘信息：



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



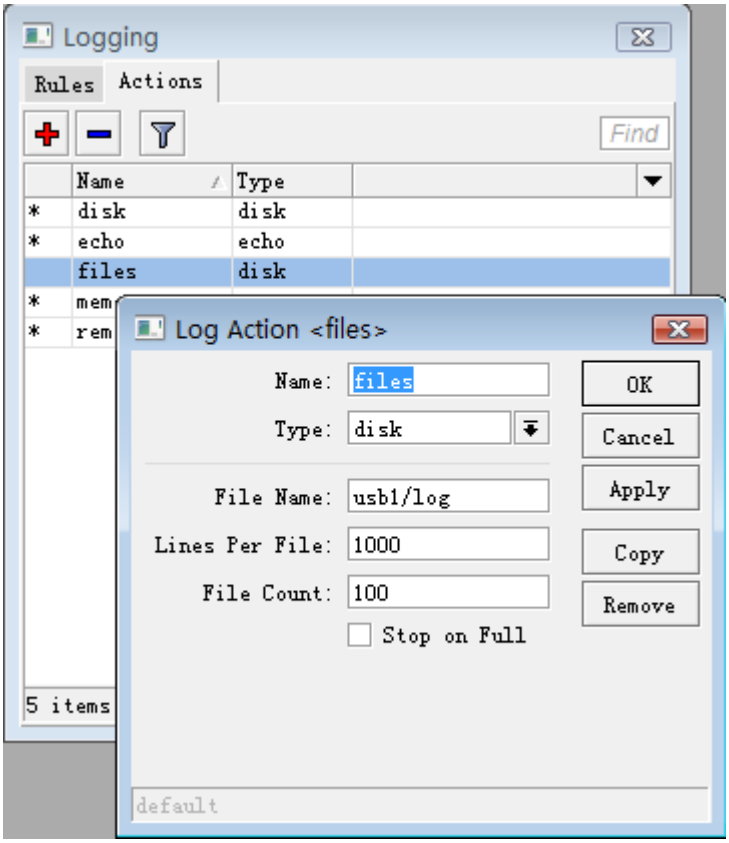
在格式化完成后，我们可以看到当前的 usb1 状态为 ready，能够正常识别到容量和空闲存储空间：



35.2 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：

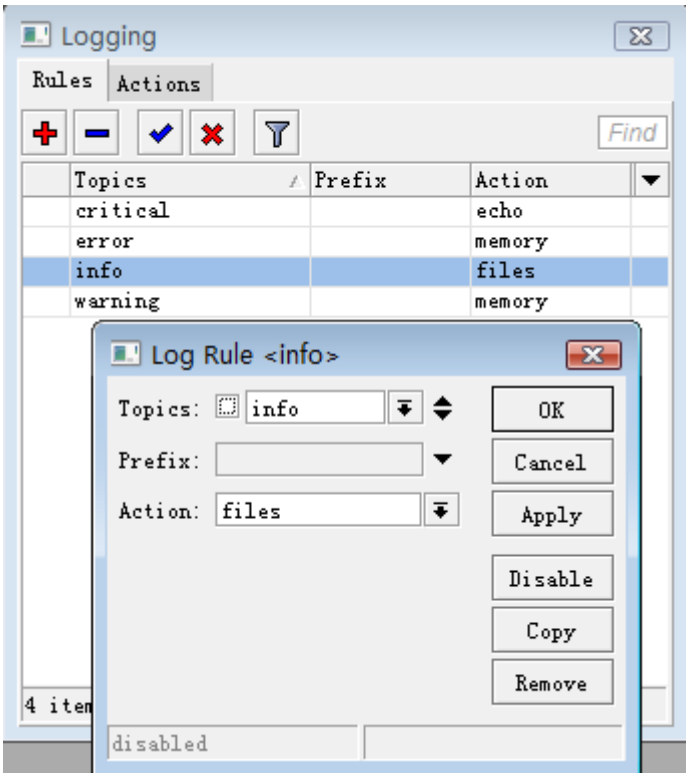


Disk 类型几个参数如下：

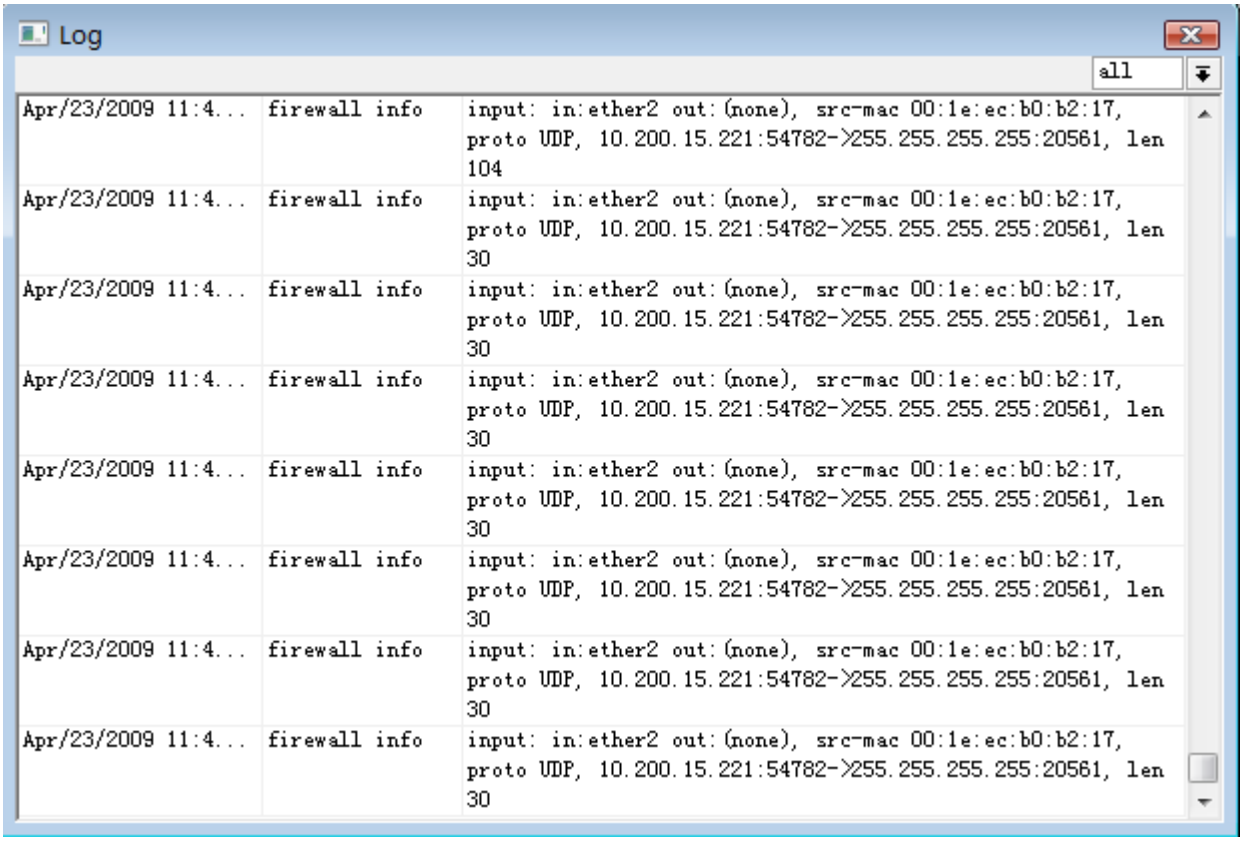
- **Type：**log 日志记录方式，这里我们选择 disk
- **File Name：**文件存储的路径，如果是 usb1 的 U 盘，我给的路径是 usb1/log
- **Lines Per File：**每个文件记录多少条信息
- **File Count：**log 日志一共建立多少个文件，如果日志记录超出文件数量，将会从 log0 号从头开始记录并覆盖原来的文件
- **Stop on Full：**当 log 建立的文件到达后，停止向文件写入 log 日志

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。

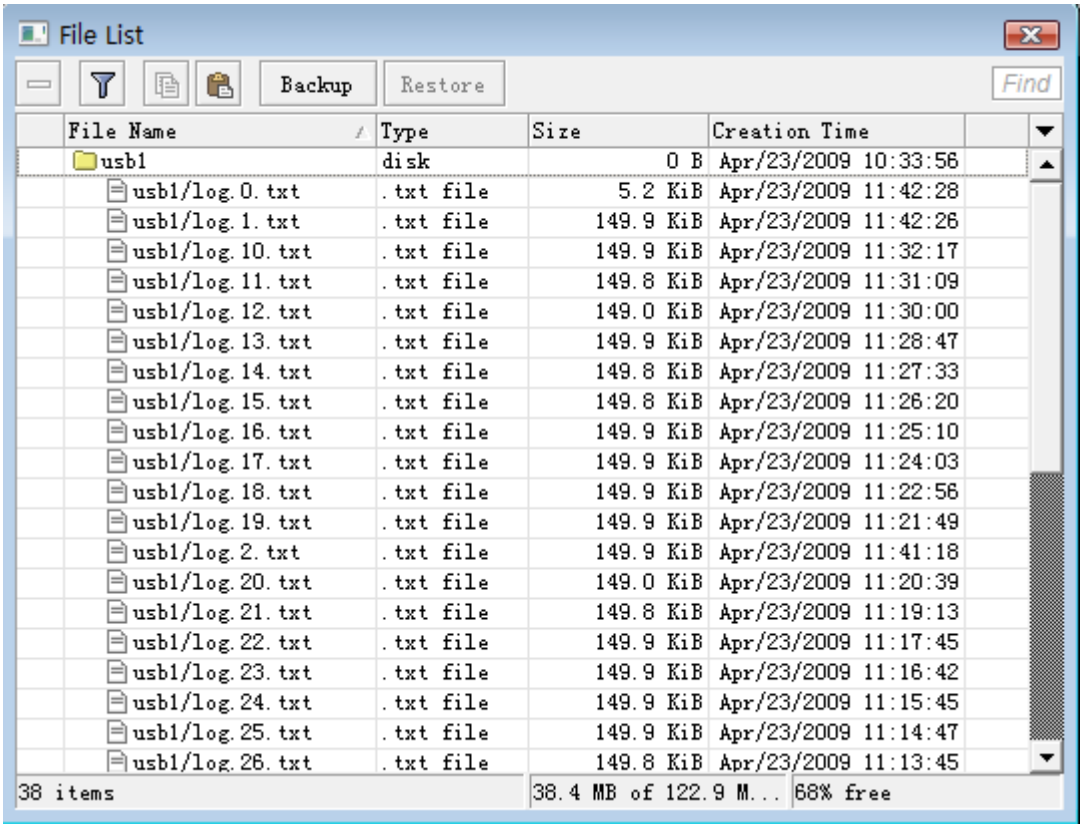
设置 logging 的 info（信息记录）为 files 操作，即记录到 usb1 中



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

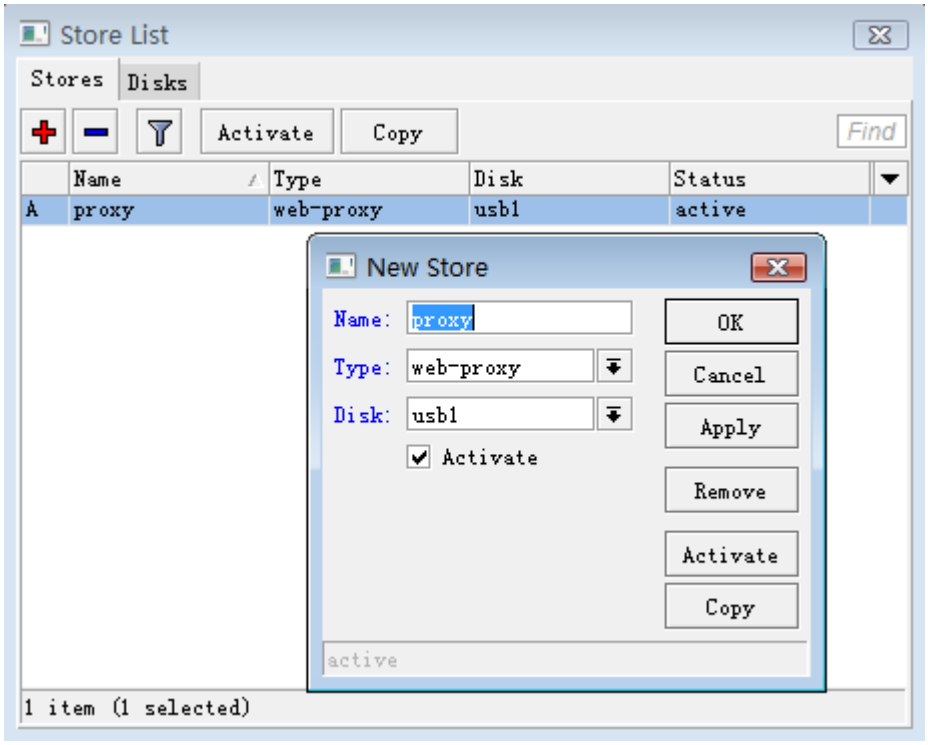


注：当在记录大量的日志信息，使用本地存储设备写入数据时，会出现 CPU 占用较大的情况，需要注意合理分配你的系统资源，建议尽量做远程日志记录的存储。

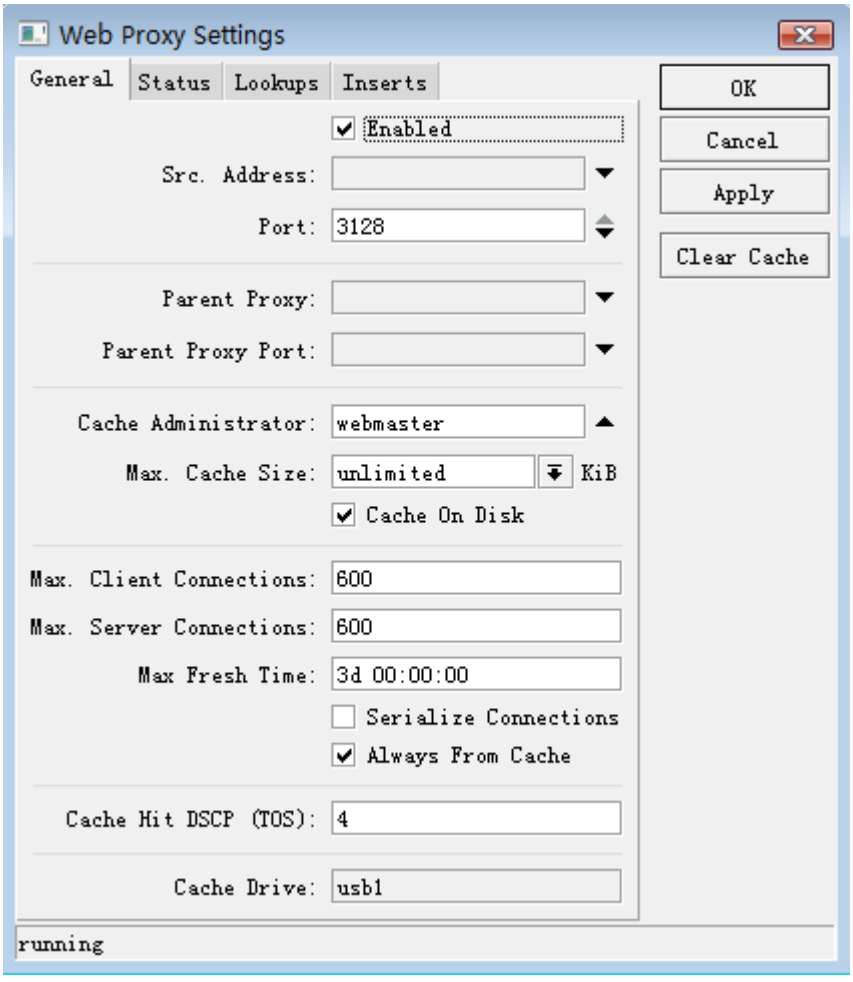
35.3 Web-Proxy 使用 U 盘存储

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。

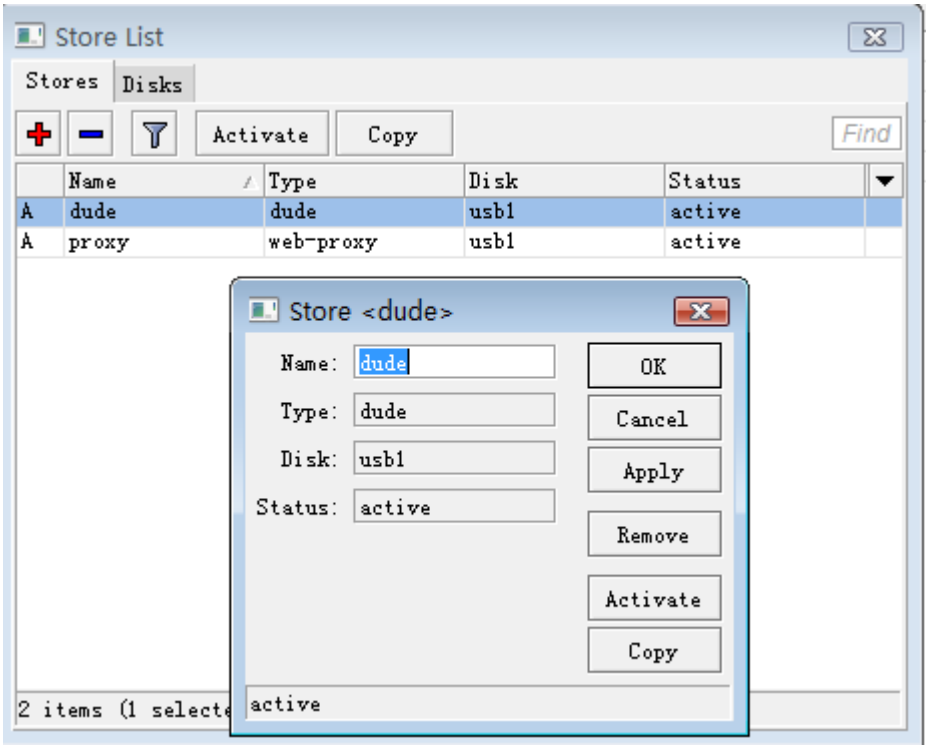


设置 Web-Proxy 的配置，这里可以看到 Cache Drive 会根据 Store 的配置，调用 usb1 的外部存储



35.4 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***
协议标准: none

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

down-script (*名称*) – 当一个主机的状态从 **unknown** 或 **up** 改变为 **down**。
host (*IP 地址*; 默认: **0.0.0.0**) – 需要监视的主机 IP 地址
interval (*时间*; 默认: **1s**) – ping 间隔时间。
status (*只读:* up | down | unknown) – 显示主机的当前状态
  up – 主机状态为 up
  down – 主机状态为 down
  unknown – 在列表项目属性被改变后或是项目被启用或禁用
timeout (时间; 默认: **1s**) – 每个 ping 的 timeout 值。在这个时钟周期内没有收到来至主机的回应，将认为该主机为 **down**
up-script (*名称*) -当一个主机的状态从 **unknown** 或 **down** 改变 **up**

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
{... [/ip route find dst 0.0.0.0] gateway 10.0.0.1}
[admin@MikroTik] system script> add name=gw_2 source={/ip route set
{.. [/ip route find dst 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=998ms \\... 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，如果网关变为无法到达改变默认路由。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，通常这种值为默认路由。用于代替**/ip route set** 命令后的第一个变量

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] 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>
```

36.2 绘图显示（Graphing）

Graphing 是一个监视工具，用于监视 RouterOS 在一段时期内不同参数的情况。

需要功能包: ***system***, ***routerboard****(optional)*
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标 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*/掩码; 默认: **0.0.0.0/0**) – 运行访问图形显示的网络地址段
store-on-disk (yes | no; 默认: **yes**) – 是否将信息存储到系统驱动上，如果选择为'no'，这些信息将存储到 RAM 中，重起后回丢失

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*/掩码; 默认: **0.0.0.0/0**) -运行访问图形显示的网络地址段，被允许的地址可以试着打开 **http://[Router_IP_address]/graphs/**，如果没有允许将无法看到
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驱动上，如果选择为'no'，这些信息将存储到 RAM 中，重起后回丢失

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*/掩码; 默认: **0.0.0.0/0**) -运行访问图形显示的网络地址段，被允许的地址可以试着打开 **http://[Router_IP_address]/graphs/**，如果没有允许将无法看到
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**) -是否将信息存储到系统驱动上，如果选择为'no'，这些信息将存储到 RAM 中，重起后回丢失

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

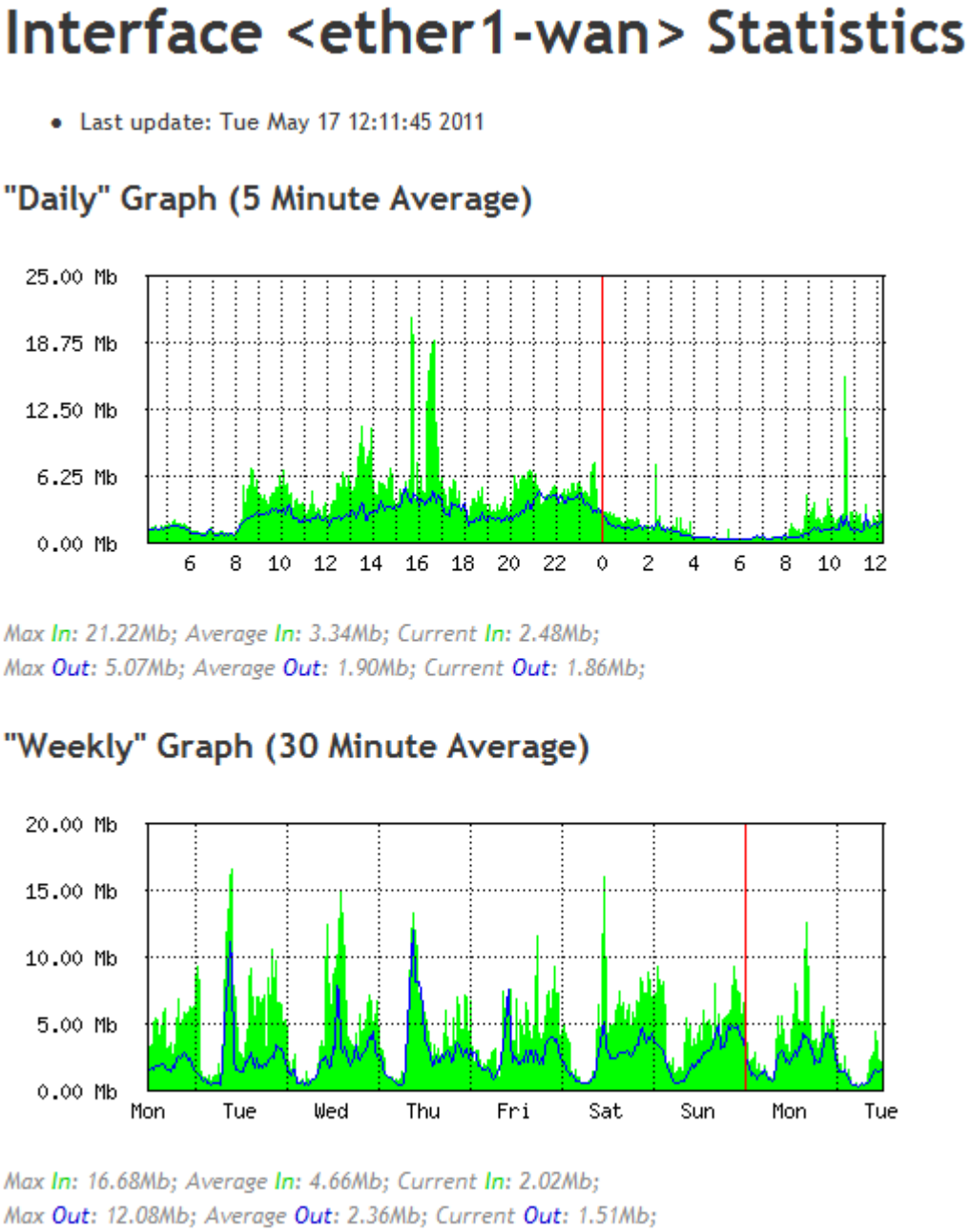
allow-address (*IP 地址*/掩码; 默认: **0.0.0.0/0**) -运行访问图形显示的网络地址段，被允许的地址可以试着打开 **http://[Router_IP_address]/graphs/**，如果没有允许将无法看到
store-on-disk (yes | no; 默认: **yes**) -是否将信息存储到系统驱动上，如果选择为'no'，这些信息将存储到 RAM 中，重起后回丢失

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-text 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 测试发送的数据包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，数据包要设置最大 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 发送最大数据包
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 测试数据包的有效载荷，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**) – 远端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) – 远端 UDP 发送最大数据包
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 数据包的带宽测试，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 (即时通信监听)

即时通信监听被称为 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：

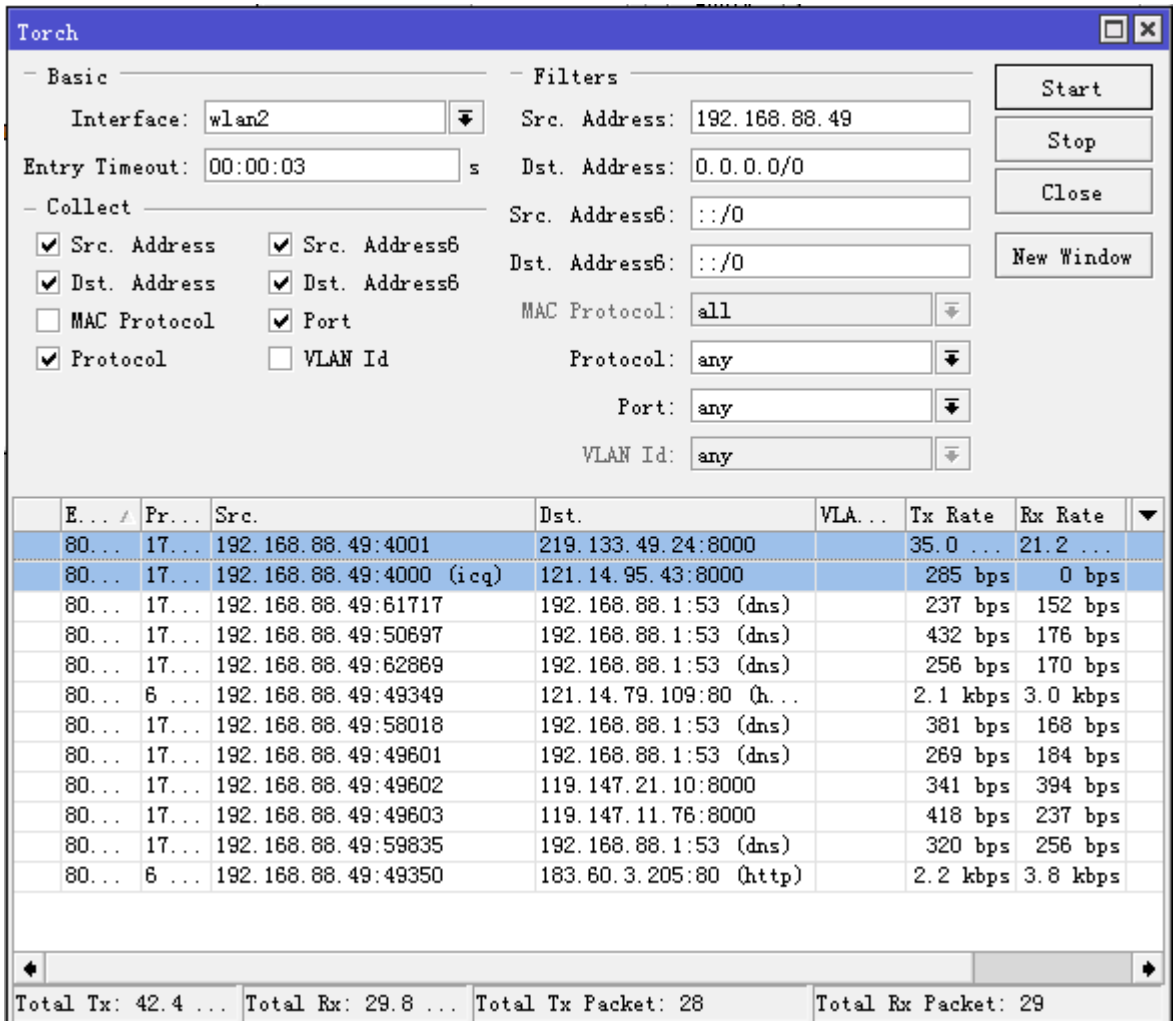
```
[admin@MikroTik] tool> torch ether1 protocol=any-ip port=any
 PRO.. SRC-PORT                DST-PORT                TX          RX
 tcp     3430                  22 (ssh)                1.06kbps    608bps
 udp     2812                  1813 (RADIUS-acct)      512bps      2.11kbps
 tcp     1059                  139 (netbios-ssn)       248bps      360bps
[admin@MikroTik] tool>
```

禁止 QQ 连接到服务器

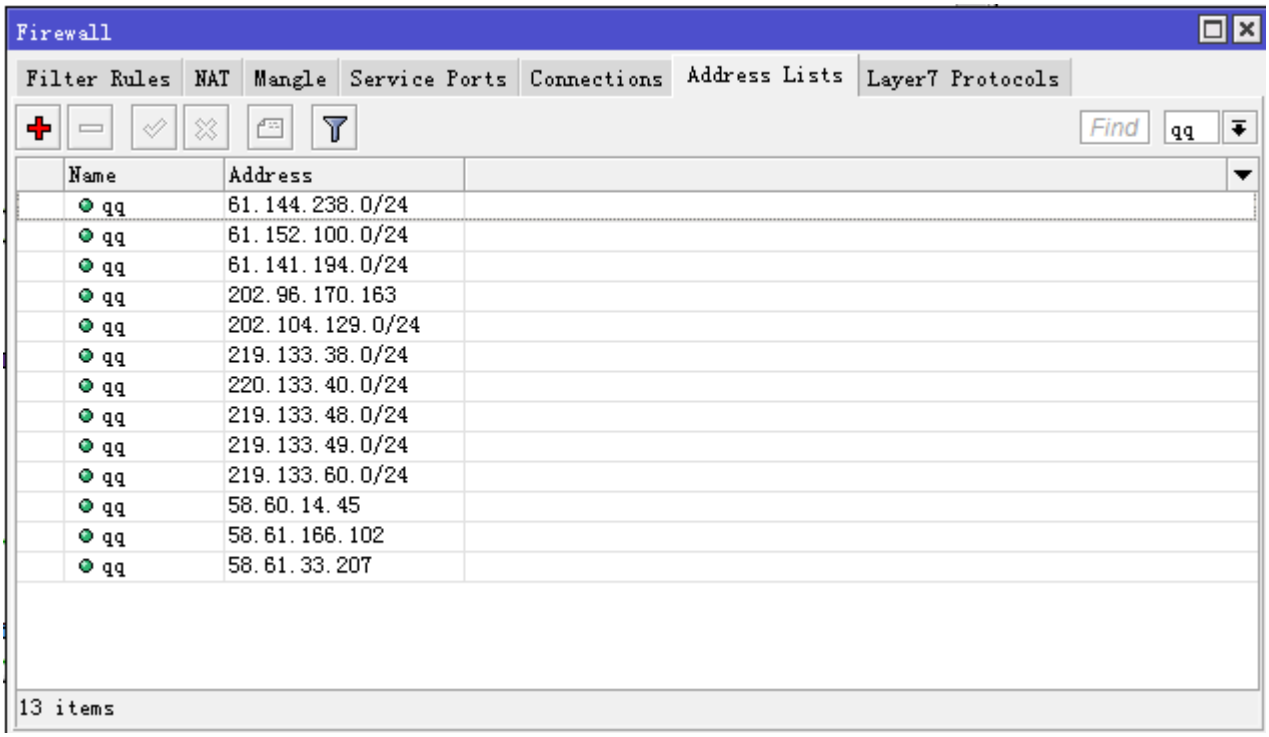
通过端口禁止 QQ 连接是没有作用的，因为 QQ 可以更改连接端口，最好的办法是通过禁止 QQ 连接相应的 QQ 服务器，实现对 QQ 的上网连接。

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 端口，如下图：



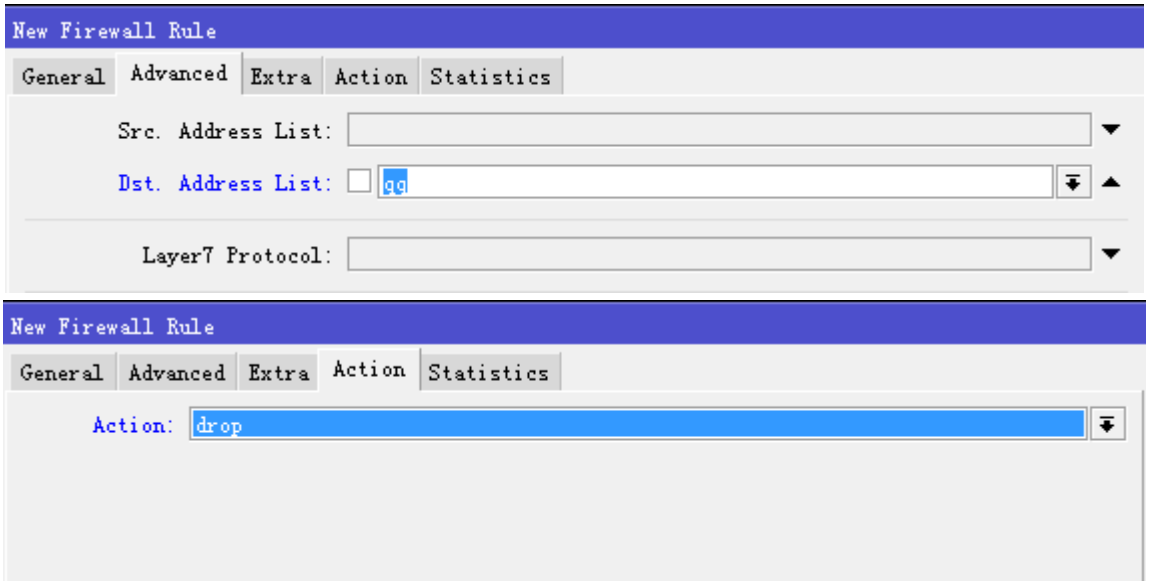
当看到 8000 端口的连接的 dst-address，可以判断为 QQ 的服务器，我通过 ip firewall address-list 填写 QQ 服务器的 IP 地址，比如 219.133.49.24 是 QQ 服务器地址，而添加到 address-list 名字取为 qq，填写 address：219.133.49.0/24，用于规则调用：



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 的附件文件名称
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
```

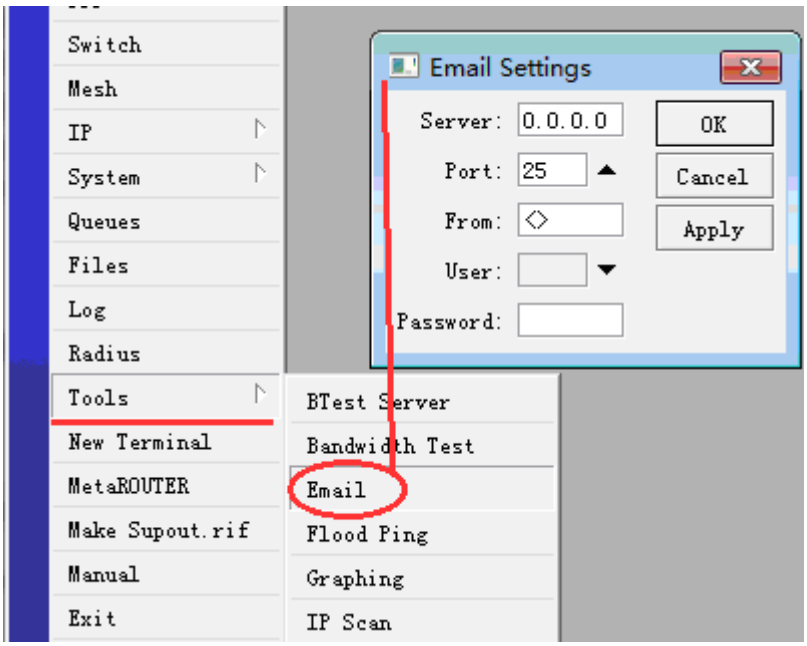
查看下面的接口列表：

```
[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 Feth 文件拷贝

Fetch 是一个 RouterOS 的 console 工具，该工具用于从网络设备拷贝文件到一个 MikroTik 设备，即由 RouterOS 下载指定的文件。

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：）	目标文件民初和路径
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，创建一个输入文件。
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：）	连接端口。
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，登陆远端设备的用户名。

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
```

36.7 3G 模块设置

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：

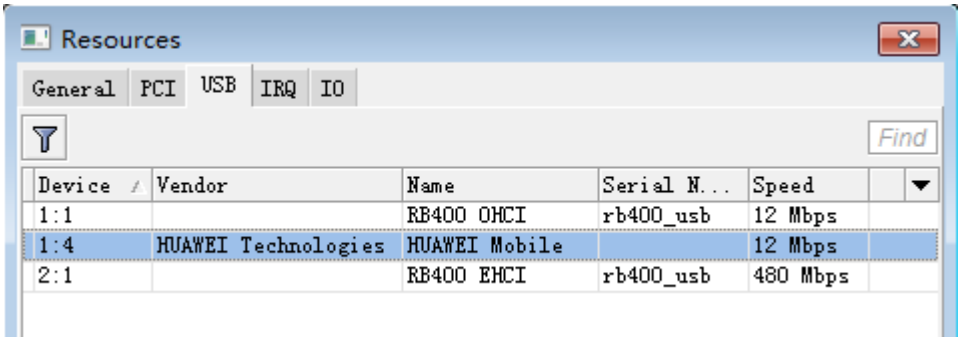
```
[admin@rb411u] > /port print
Flags: I - inactive
 #    NAME                    CHANNELS    USED-BY                    BAUD-RATE
 0    serial0                 1           Serial Console             auto
 1    usb2                    3                                      9600
[admin@rb411u] >
```

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

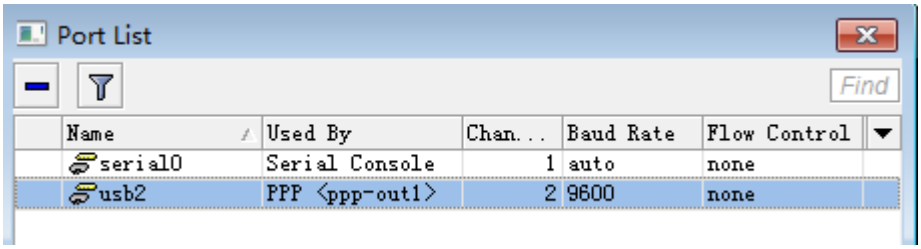
联通 3G 拨号测试

无线网卡型号：华为 3G USB 网卡 E220，基于联通的 WCDMA，测试主板：RB411U

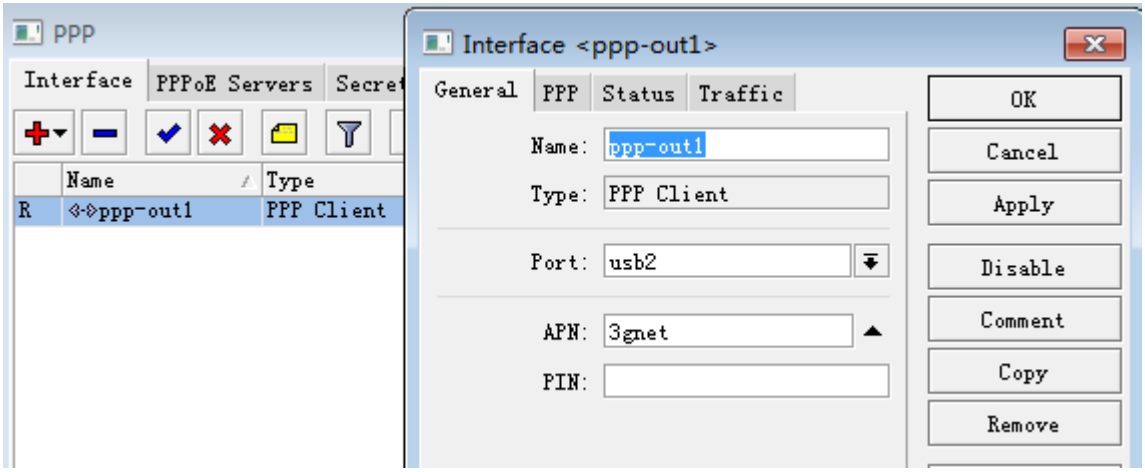
将华为的 USB 网卡，插入 RB411U 的 USB 接口，在 system resource 里可以找到网卡信息，如下显示的 USB 信息



然后我们进入 system port 里可以查看到一个拨号接口 usb2，参数默认即可，不用修改



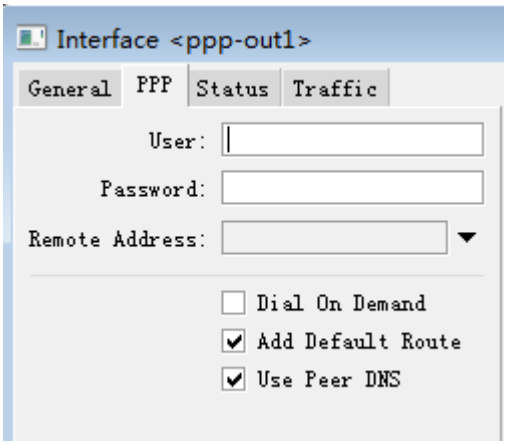
进入 ppp 目录下，添加一个 ppp-client，并设置 port 为 usb2，APN 设置为“3gnet”，这样 PPP-client 可拨号了



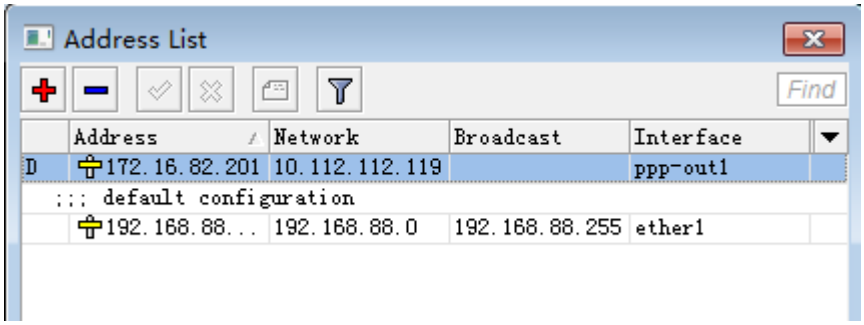
这里我们可以通过日志 log，查看拨号情况，显示如下：

```
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

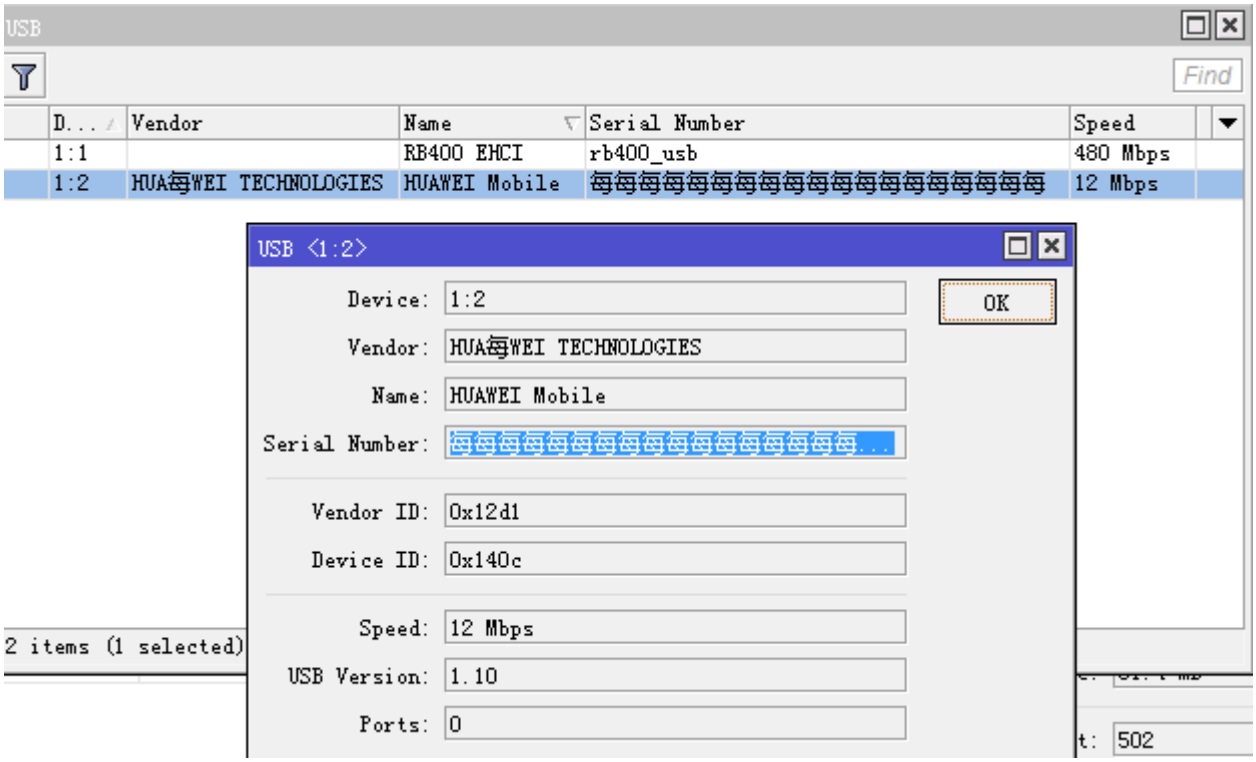


测试迅雷下载可以到 50-150KB 的范围波动，可能室内的原因，如果在室外或者信号好的地方带宽会相对稳定

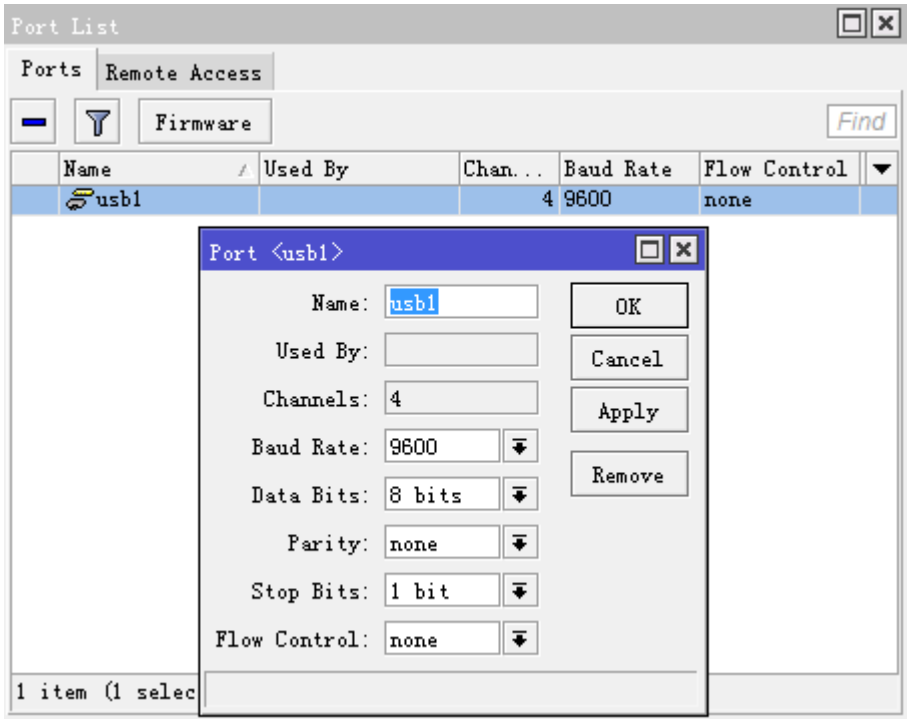
电信 CDMA 上网卡配置

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。

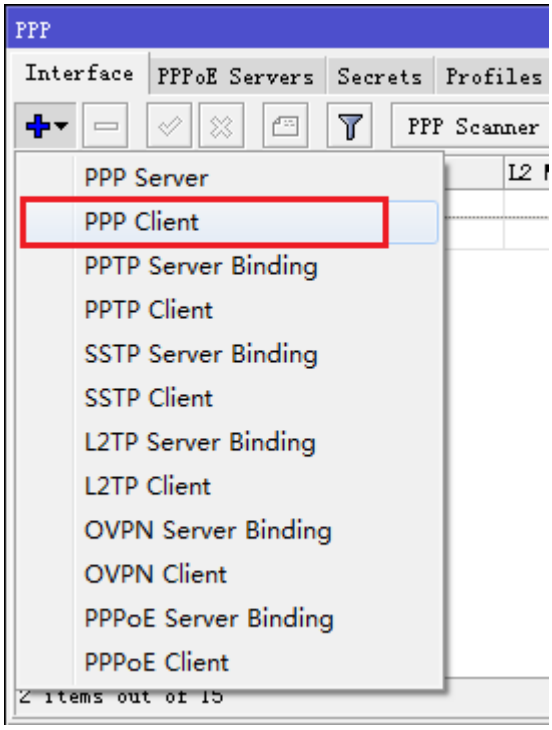
当将上 USB 网卡插到设备的 USB 接口后，我们可以在 system resource 中的 USB 菜单下找到设备，如下图，打开 USB 后看到，只不过我测试的这款华为 E177 在 USB 列表中显示错误，但并没有影响使用



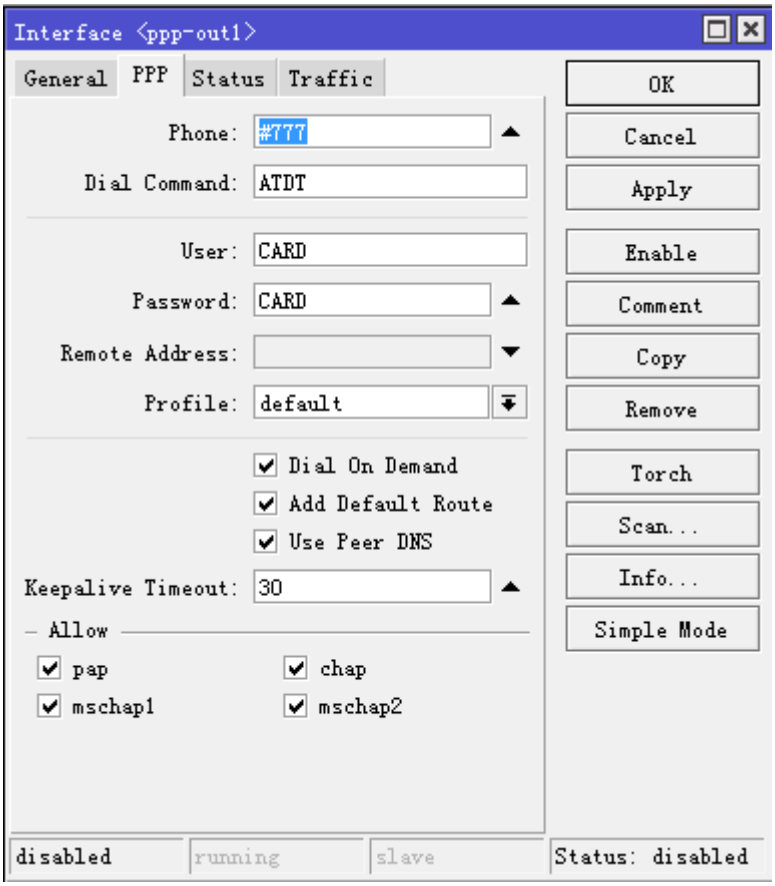
之后我们可以在 system port 中被识别为 usb1（这个根据设备接口情况具体识别）， 参数默认就可以，如下图



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”，其他参数默认（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。

36.8 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包（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: *""*)	文件名称，Sniffer 数据包将存储在这个文件中
filter-address1 (*IP 地址/子网:port*; 默认: 0.0.0.0/0: 0-65535)	抓取过滤的第一组 IP 地址
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 - 抓取所有数据包 • 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)	保持数据包的头部到内存，而非整个数据包
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包发到指定的 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包)。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

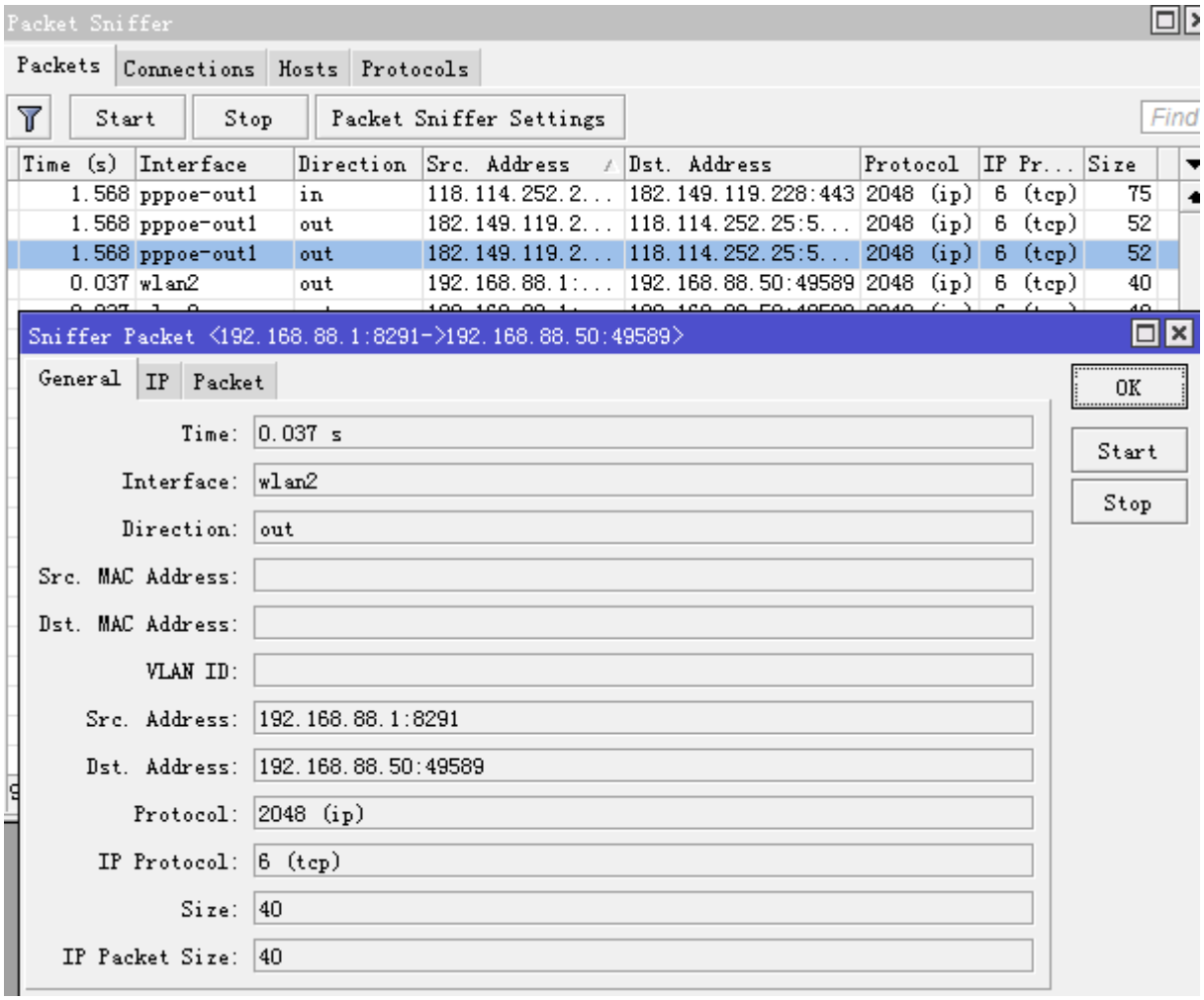
```
子菜单: /tool sniffer packet
```

这个子菜单允许查看抓到数据包的列表

属性	描述
data (*只读: text*)	表明数据包含的数据包

direction (*只读: in | out*)	数据包的方向，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*)	抓到数据包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*)	数据包达到的时间
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

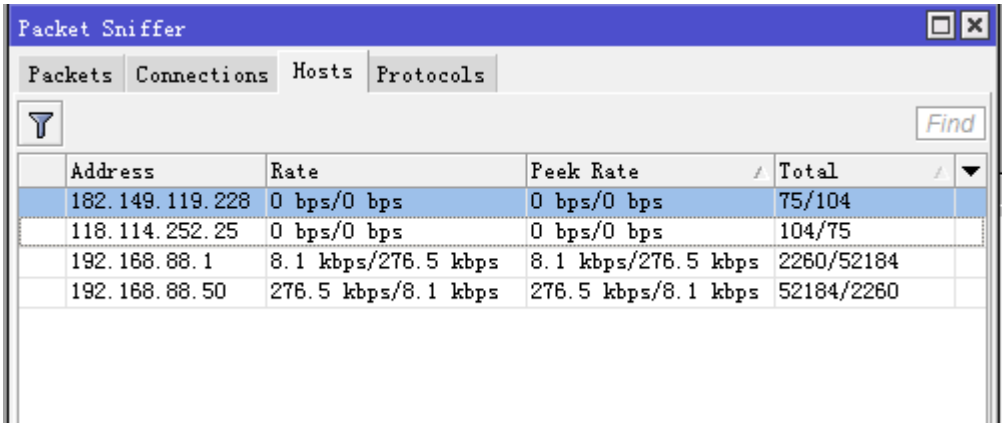
子菜单: /tool sniffer host

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*)	所有数据包的速率

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

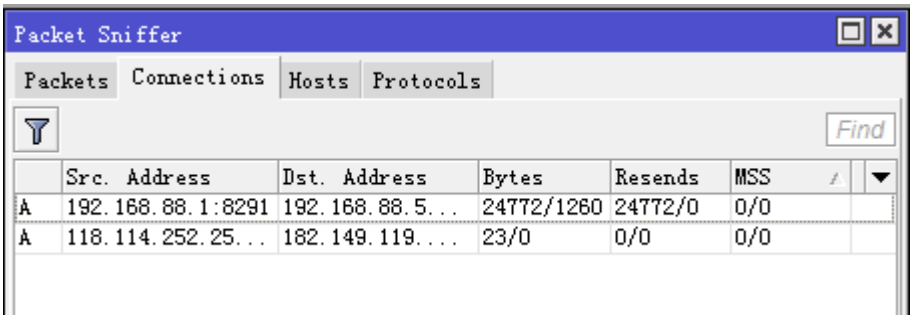
子菜单: /tool sniffer connection

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>
```



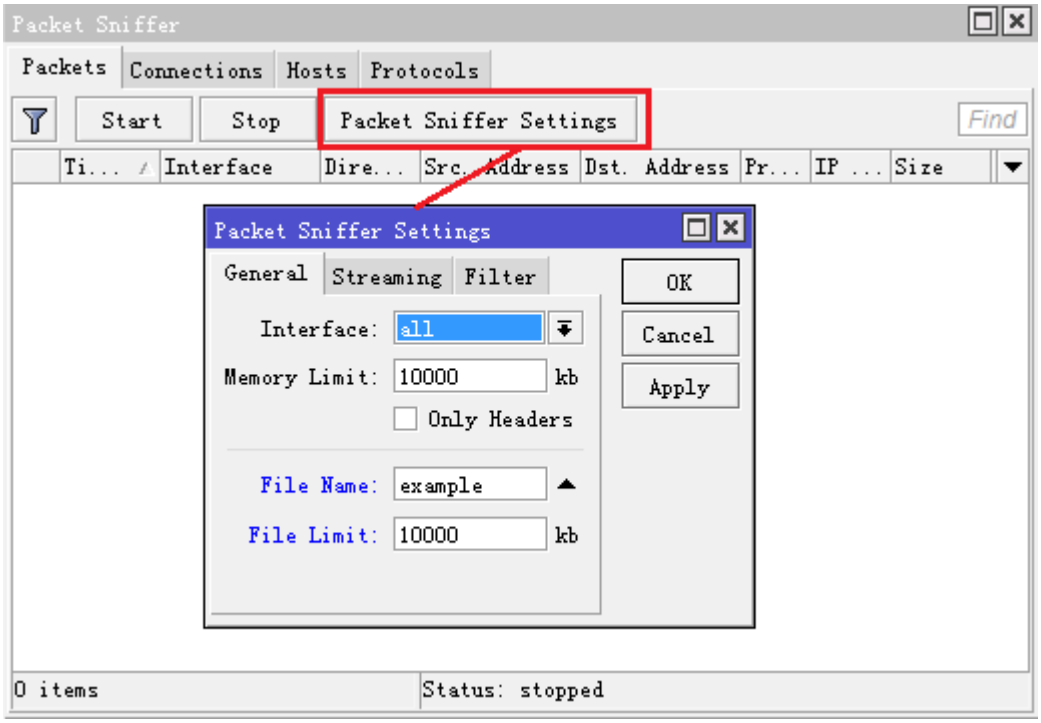
下载抓包结果

我们可以把抓包的结果保存到文件中，便于更详细的分析。

属性	描述
file-name (*字符*; 默认: **""**)	抓包存储的文件名称

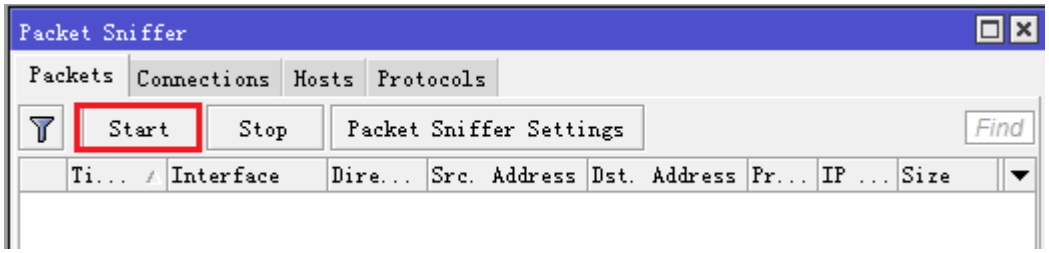
保存抓包的文件设置

```
[admin@MikroTik] /tool sniffer set file-name=example file-limit=10000
```



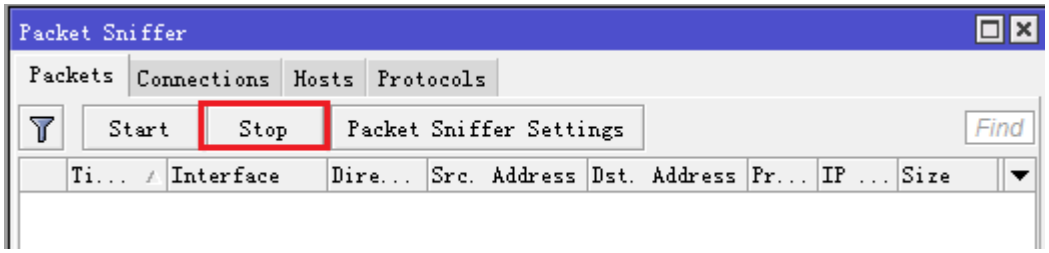
启动 sniffer

```
[admin@MikroTik] /tool sniffer start
```



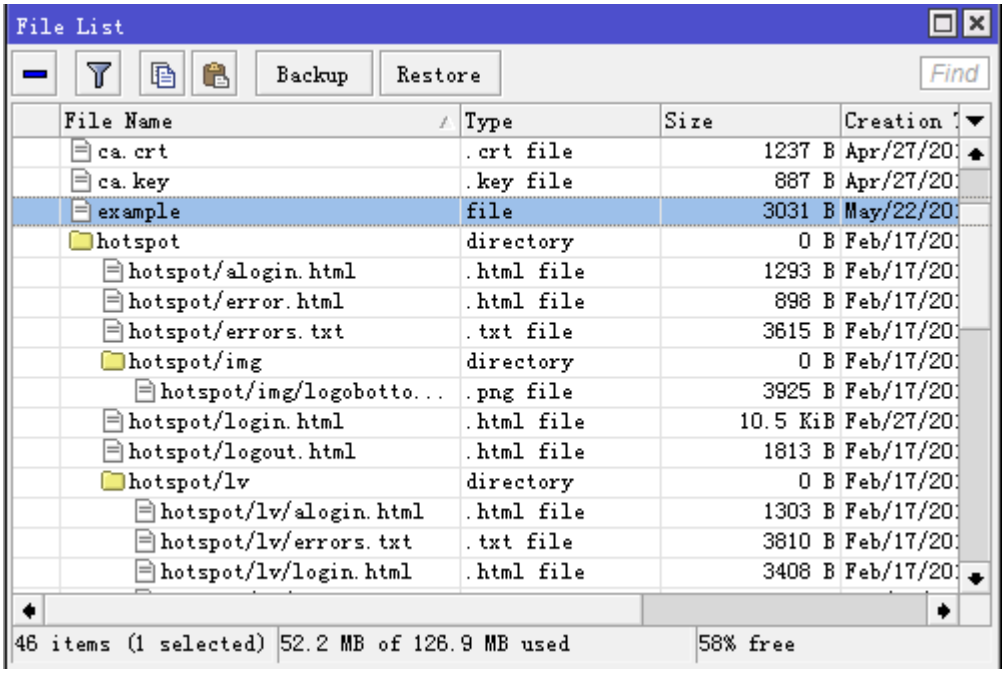
在分析一段时候后，停止抓包,

```
[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电脑上，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
```



当你获取抓包文件后，可以通过各种分析工具进行查看，例如 wireshark，这个文件可以被 wireshark 直接打开。

36.9 MikroTik SMB

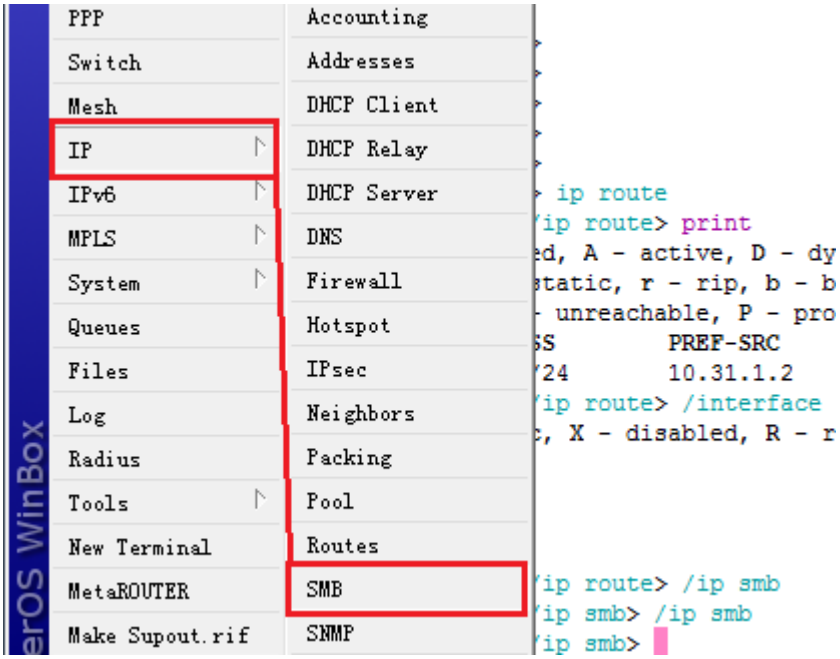
从 RouterOS v5.12 开始支持 SMB（Server Message Block）协议，SMB 通信协议是微软（Microsoft）和英特尔(Intel)在 1987 年制定的协议，主要是作为 Microsoft 网络的通讯协议。

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。

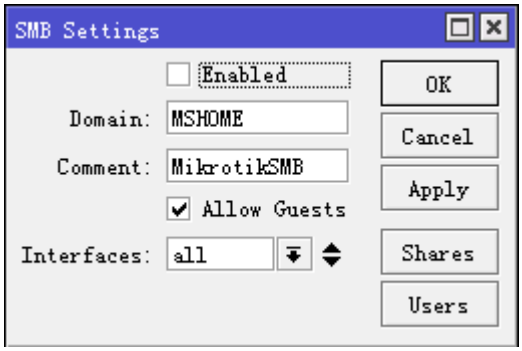
操作路径：**/ip smb**

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文件，也方便个人的文件共享。那下面是 RouterOS 的 SMB 操作

首先 5.12 版本在 ip 目录下增加了 SMB 选项，如下图：



Winbox 下的 SMB 设置界面，通过 enabled 开关 SMB 功能。



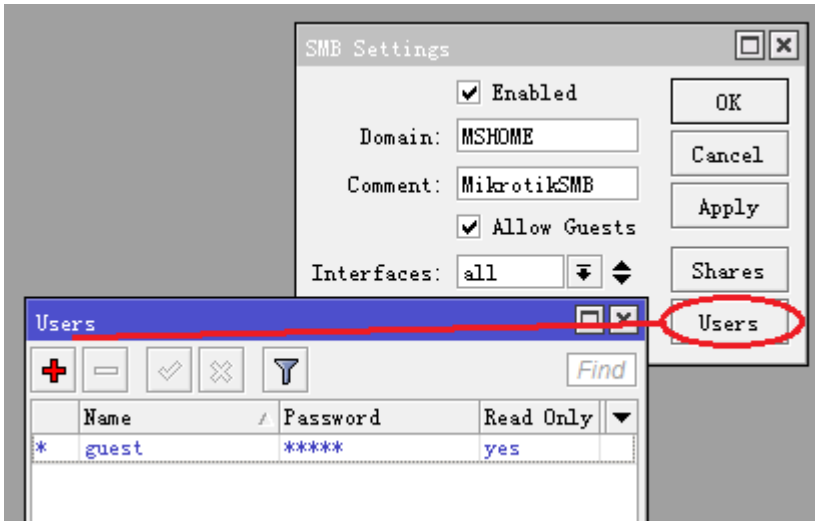
在命令行进入 ip smb，可以看到设置参数，通 set 命令启用 SMB 协议

```
[admin@MikroTik] /ip smb> print
               enabled: no
                domain: MSHOME
               comment: MikrotikSMB
         allow-guests: yes
            interfaces: all
[admin@MikroTik] /ip smb> set enabled=yes
```

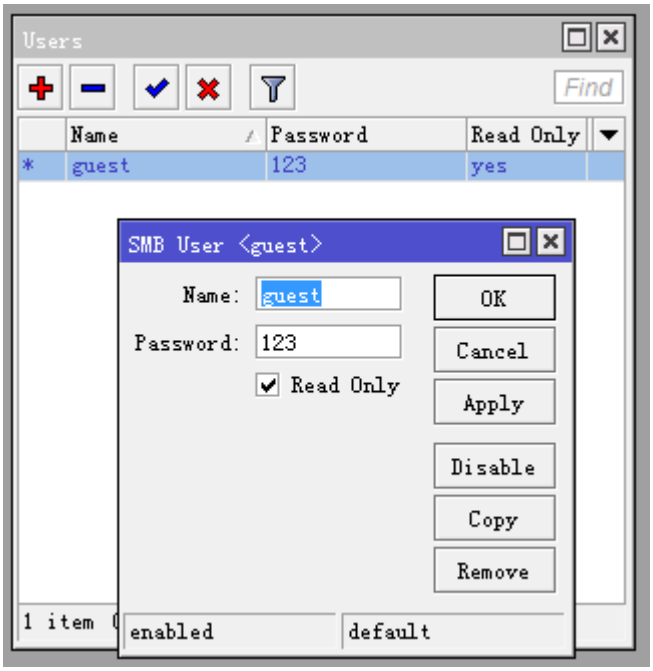
启用 SMB 协议后，以下参数：

- 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 就可以访问共享文件；
- 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。

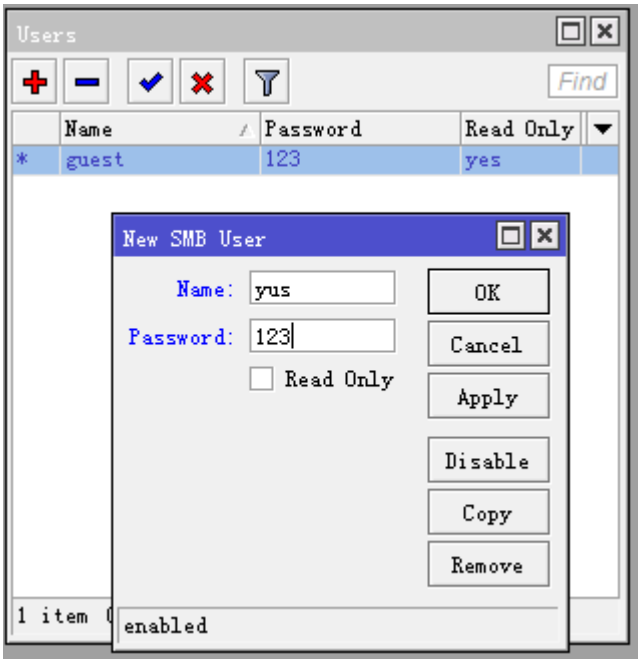
MikrotikSMB 允许设置登陆的账号和密码，默认账号是 guest，没有密码，read-only=yes 即 guest 账号登陆后，对共享目录下的文件只有只读权限



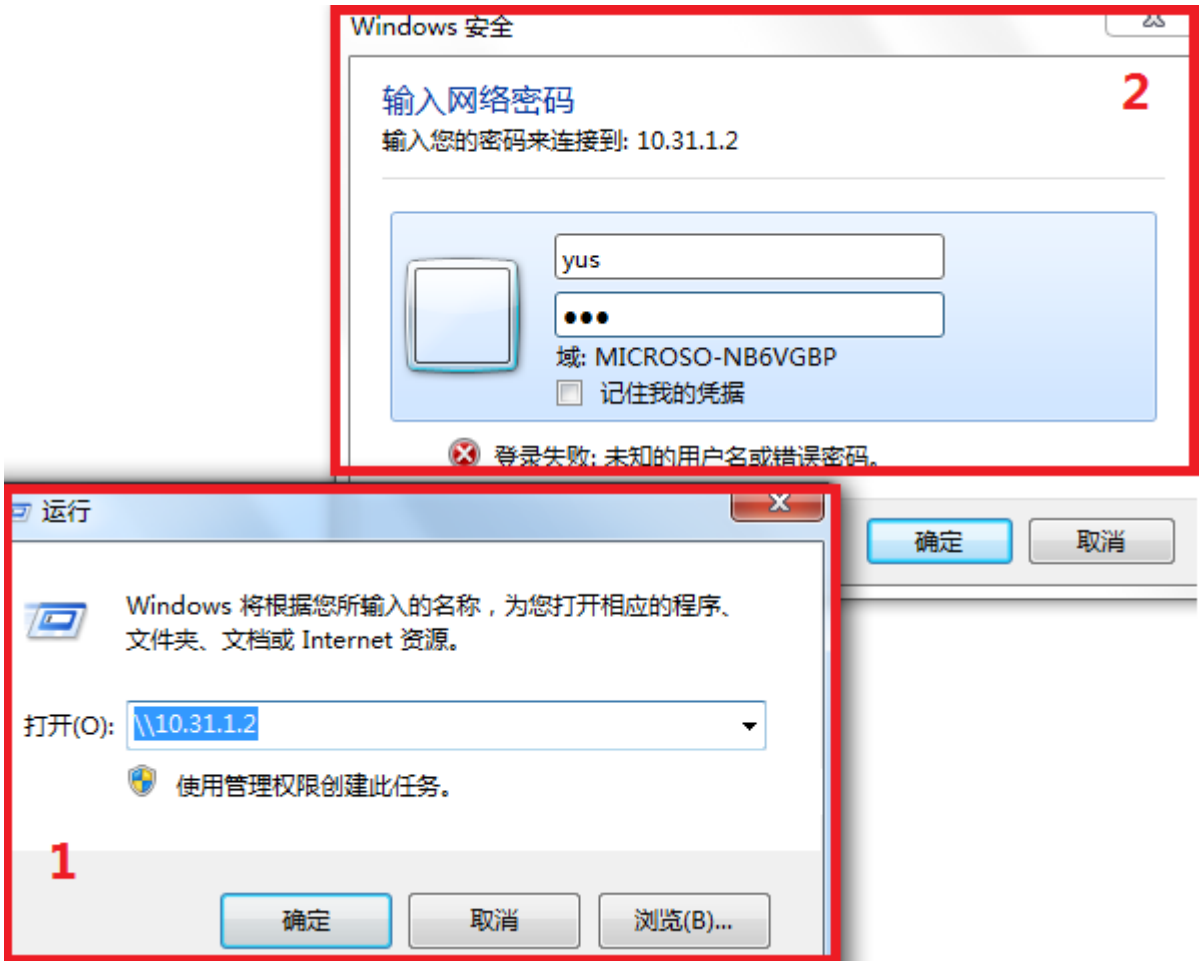
当然我们也可以设置 guest 的密码，或者新建一个用户账号和控制读写权限，下面是给 guest 账号配置一个密码 123。



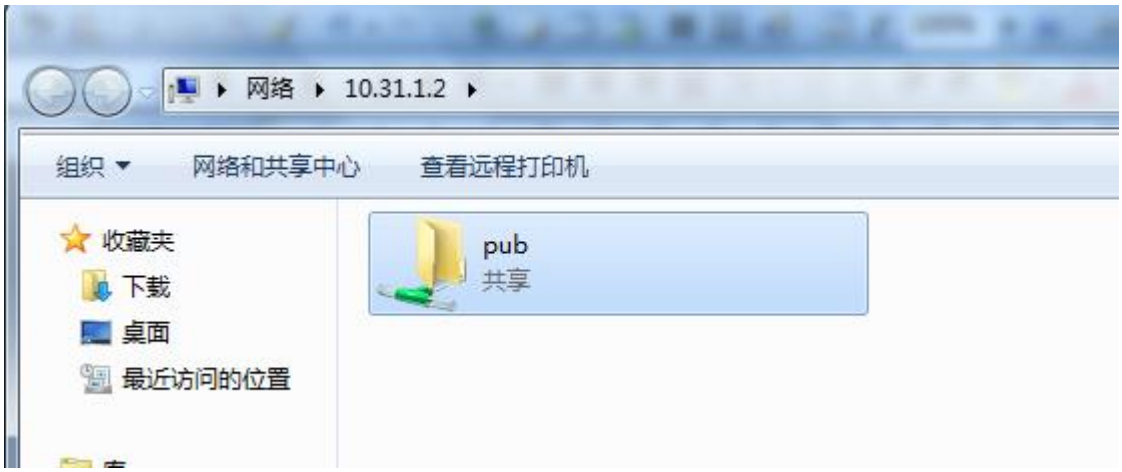
这里我也可以新建一个 yus 账号，密码 123，read-only=no，启用读写权限：



登陆 RouterOS 的文件共享，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

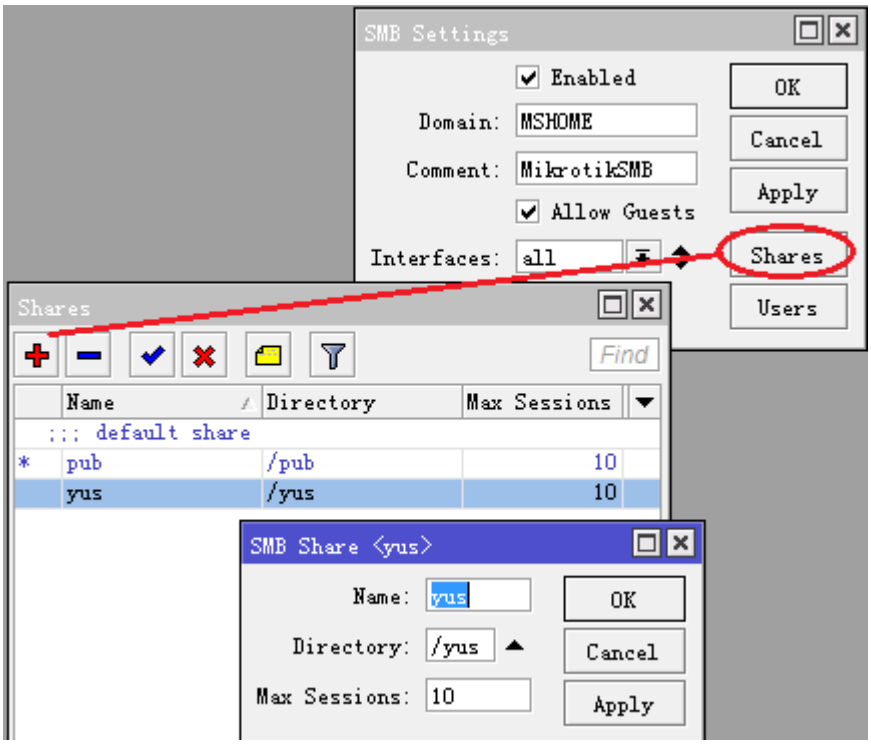


登陆后，可以看到以下的目录

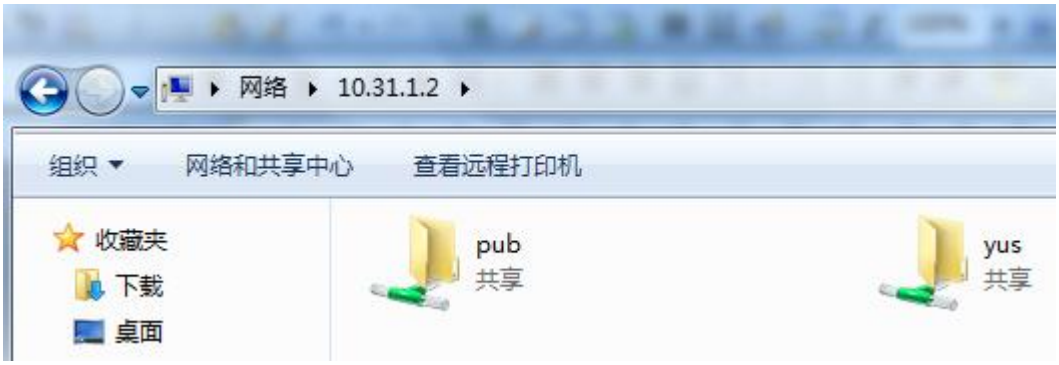


剩下的操作就不用我多说了吧，复制剪切粘贴和 windows 操作一样了，当然你也可以在 RouterOS 的 SMB 中分配各种账号给相关用户，共享各种资料！

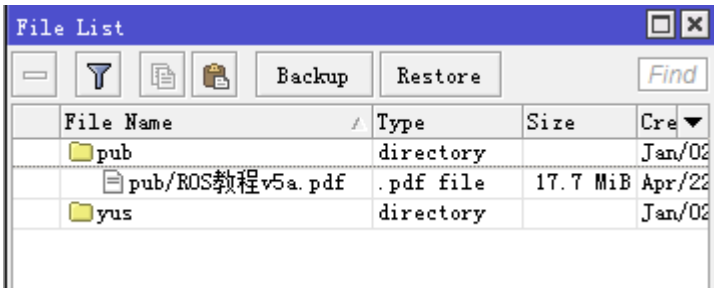
也可以为自己新建一个目录，但 SMB 还不具备不同目录下的权限管理：



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，一般选择重启电脑，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，大文件和频繁读写操作时需要特别考虑稳定性。

36.10 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 系统下查看

在多 CPU 系统下，系统允许查看每个 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%
```

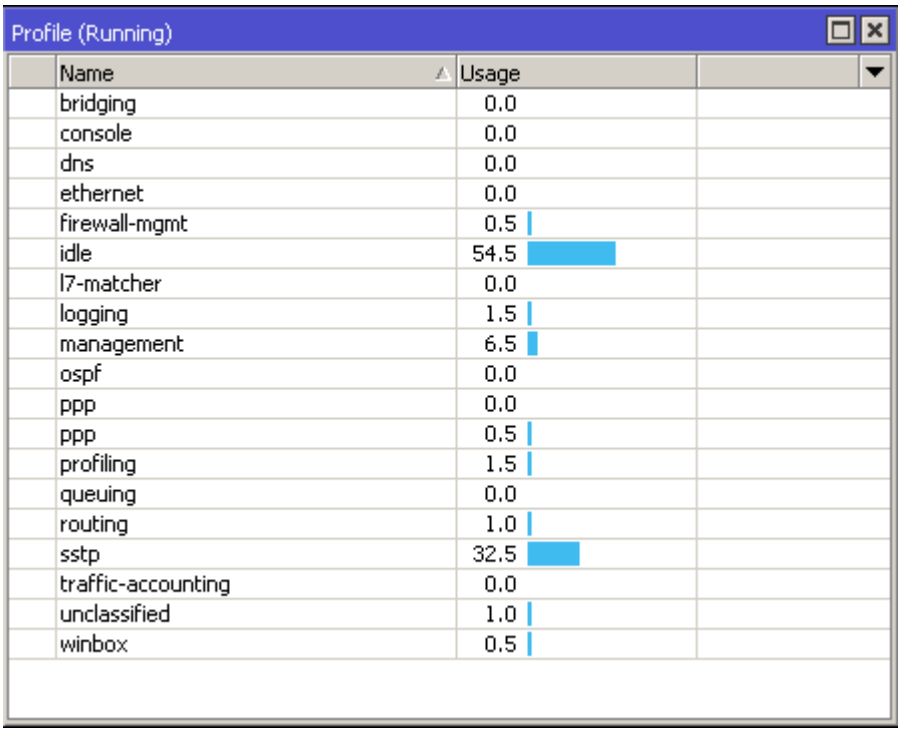
"**cpu**" 属性允许选择第几个 CPU，或其他两个属性 **all** 和 **total**

- **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：


Profile (Running)
Name
Usage
bridging 0.0
console 0.0
dns 0.0
ethernet 0.0
firewall-mgmt 0.5
idle 54.5
l7-matcher 0.0
logging 1.5
management 6.5
ospf 0.0
ppp 0.0
ppp 0.5
profiling 1.5
queuing 0.0
routing 1.0
sstp 32.5
traffic-accounting 0.0
unclassified 1.0
winbox 0.5

第三十七章 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 登录帐号管理

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。

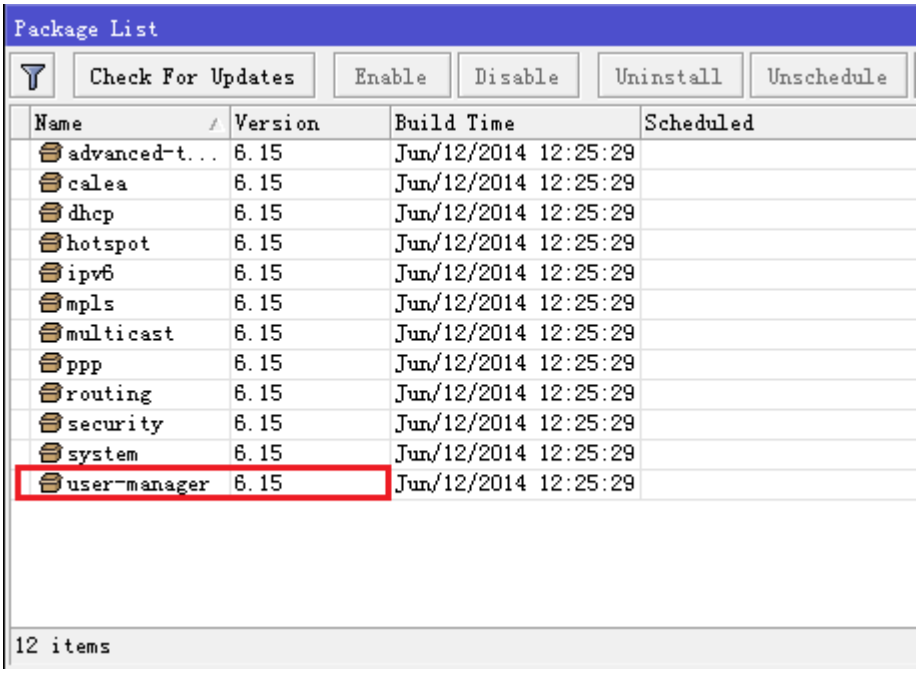
User Manager 操作主要通过 Web 界面进行管理对接 RouterOS，添加、删除和查询用户信息，同样也可以使用 CLI 操作，并且可以进行备份和导入。User Manager 通过安装包形式嵌入到 RouterOS 上，即 User Manager 版本与 RouterOS 版本一致。User Manager 和 RouterOS 能工作在 x86、MIPS、PowerPC 和 TILE 平台，路由器至少需要 32MB 内存和 2MB 的 HDD 空间，HDD 用于数据和日志信息存储，RouterOS 可以使用 USB 存储或者挂载硬盘（具体关于存储请参见 RouterOS Store 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

37.1 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。

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

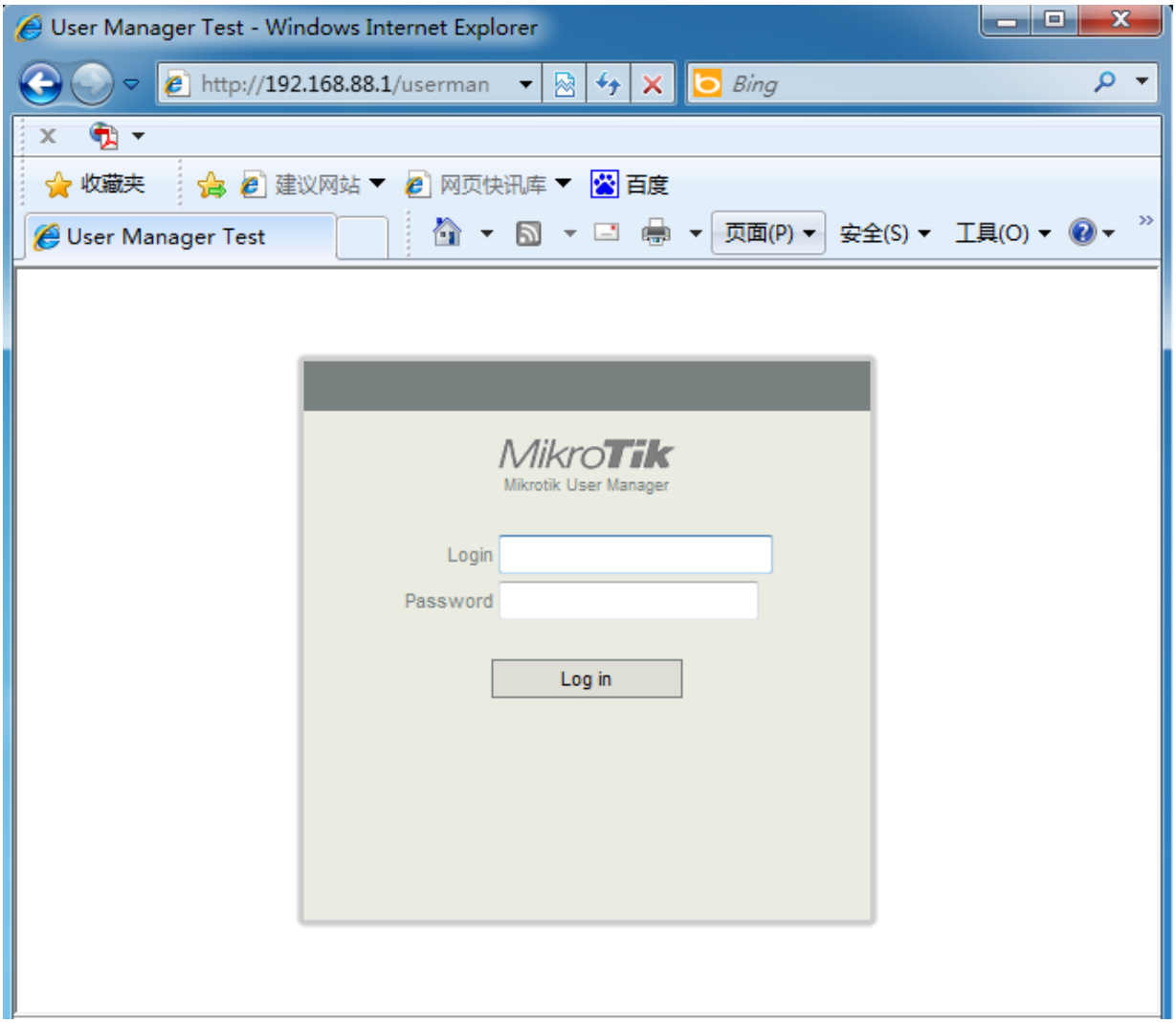
```
[admin@ MikroTik] /tool user-manager customer set admin password=PASSWORD
```

通过 **print** 命令查看添加或修改的信息。

```
[admin@ MikroTik] /tool user-manager customer> print
```

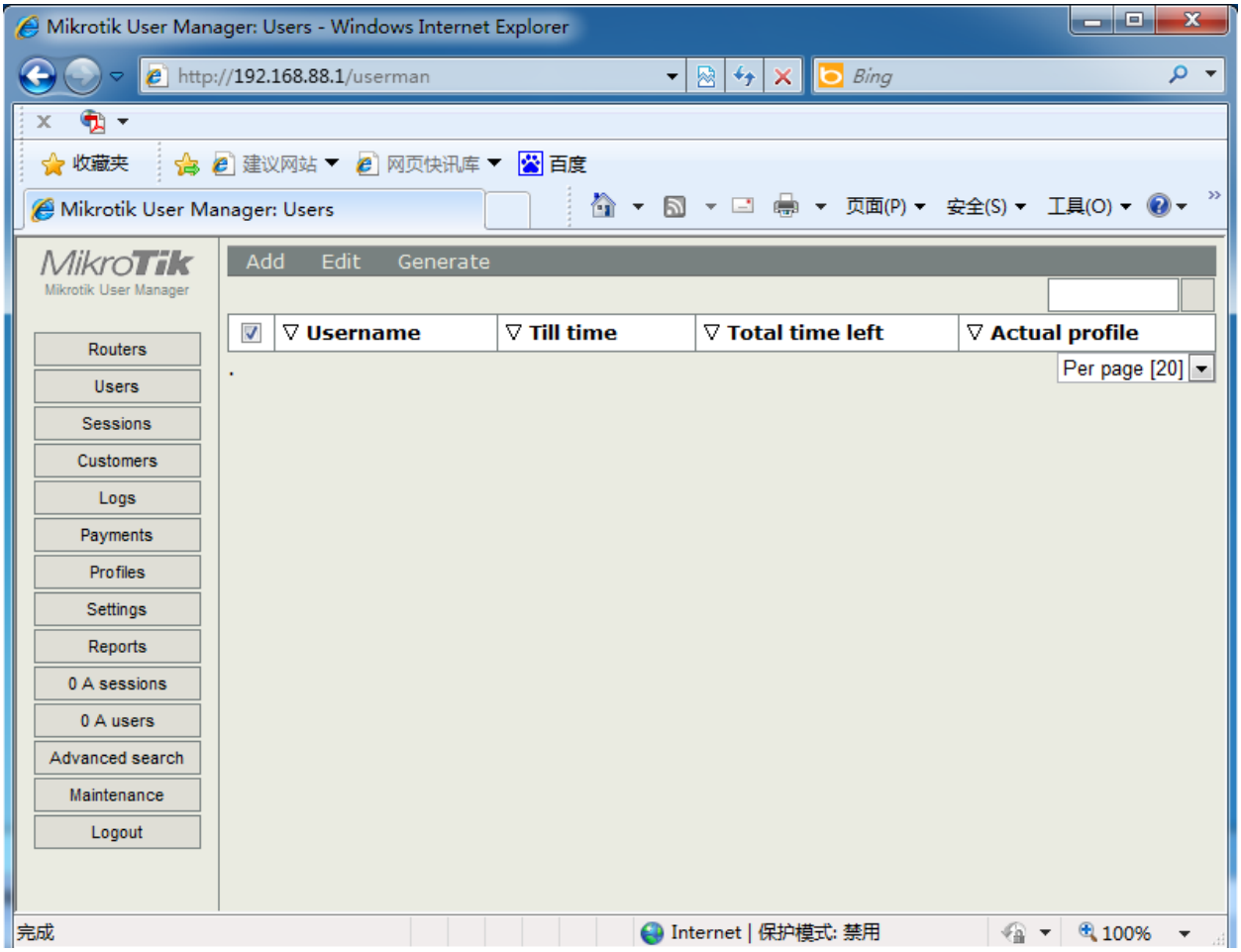
```
Flags: X - disabled
 0    login="admin" password="adminpassword" backup-allowed=yes currency="USD"
      time-zone=-00:00 permissions=owner signup-allowed=no paypal-allowed=no
      paypal-secure-response=no paypal-accept-pending=no
```

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（你可以在 ip service 里修改 www 的访问端口），访问路径 http://router_ip/userman，如下图：

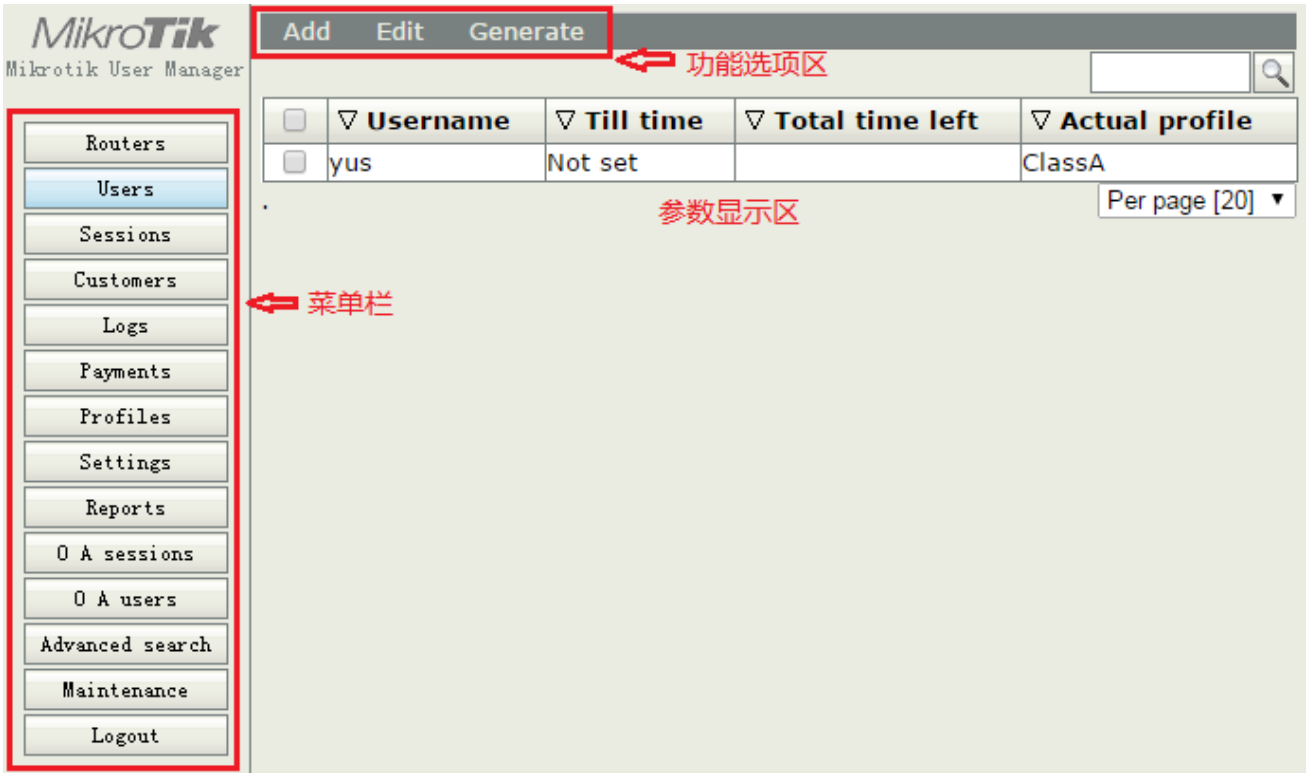


操作界面

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，登录后界面如下：



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：用户账号信息添加、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：导出 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 默认是英文界面，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

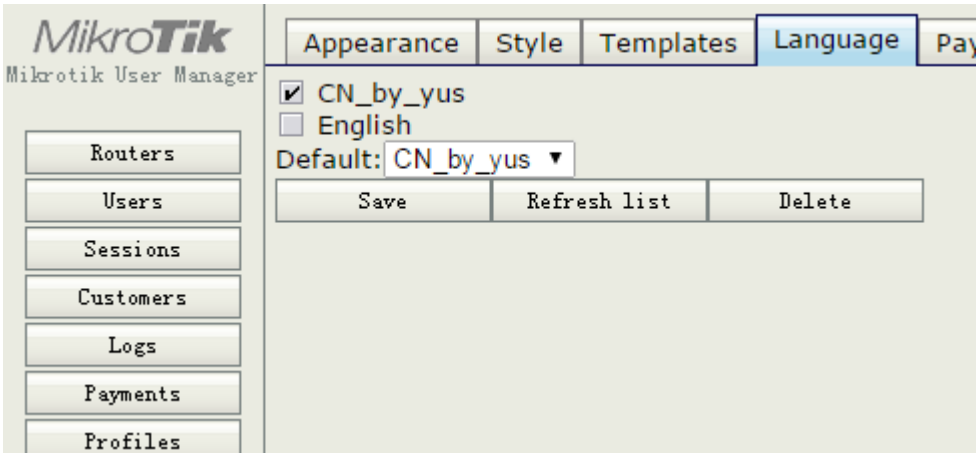
下载模板后，可以通过文本编辑，编辑请使用 UTF-8 编码，如通过文本打开后，编辑里面的内容，msgid 为字符 ID 默认代表对应的字段名，而 msgstr 就是可以修改的语言字符，可以根据这些内容修改。

```
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，保存文件后缀名为“.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。

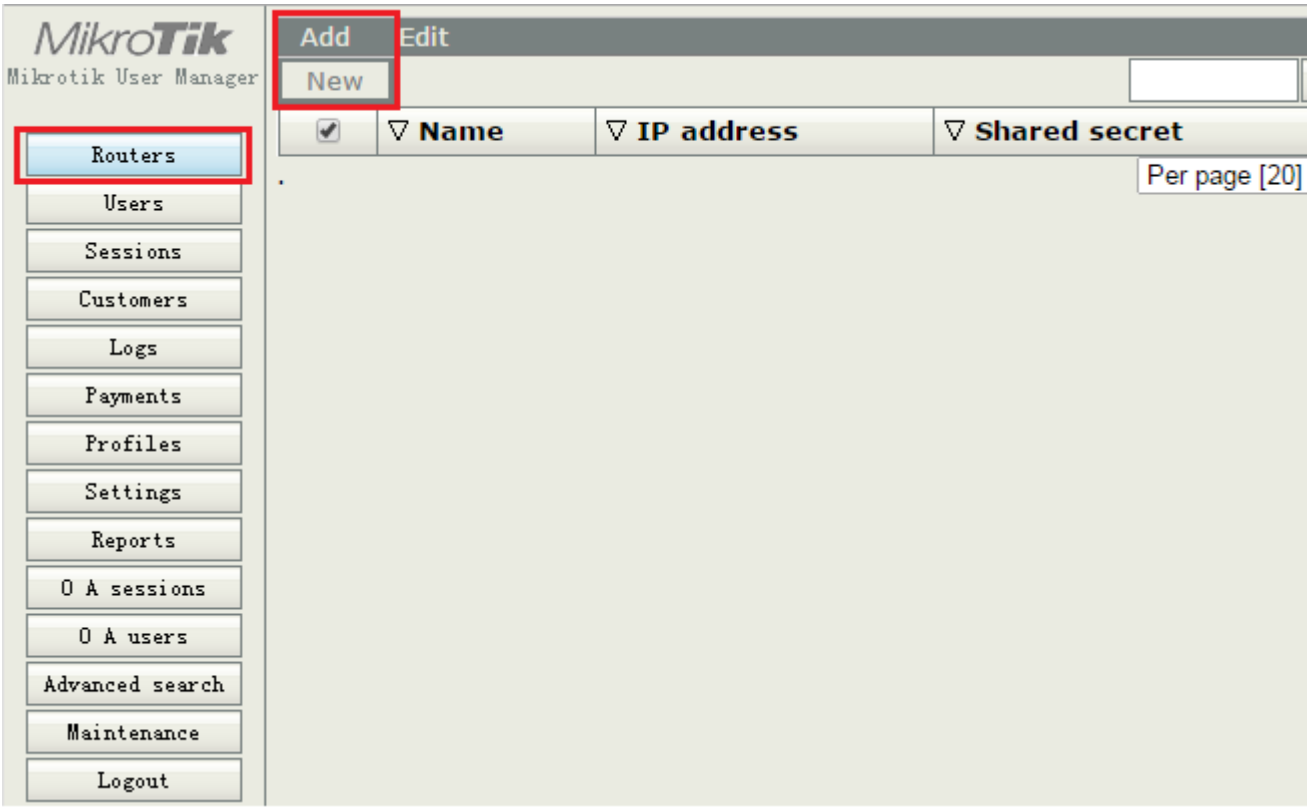


后期的版本在菜单栏的下面新增了语言选项，直接提供语言环境切换。

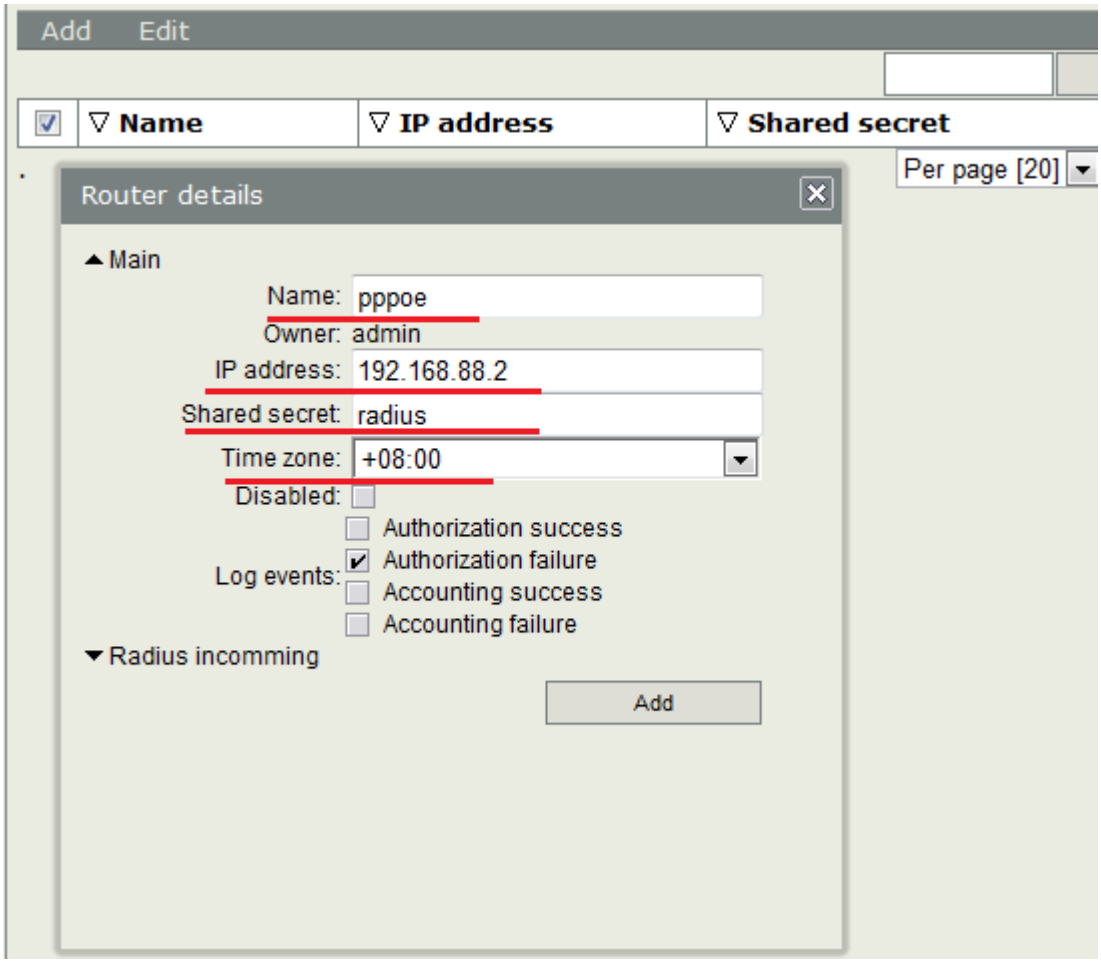
37.2 对接 RouterOS 配置事例

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

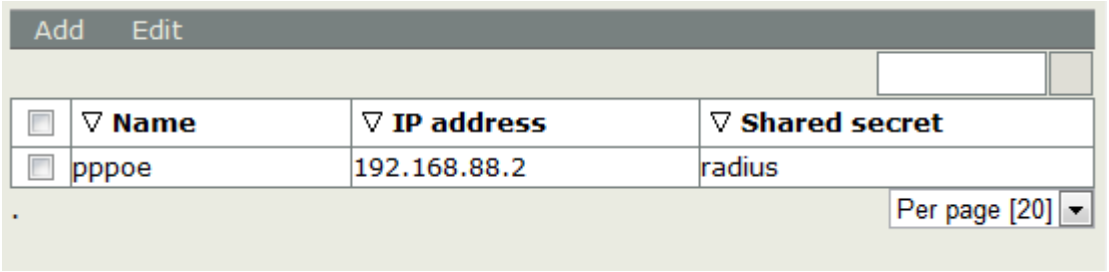


选择 Add->New 后，出现下面的选项，将新增 RouterOS 取名叫 pppoe，与该 RouterOS 对接 IP 地址是 192.168.88.2，Shared Secret（密钥）是 RADIUS，Time zone 选择+08:00，中国在+8 区



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

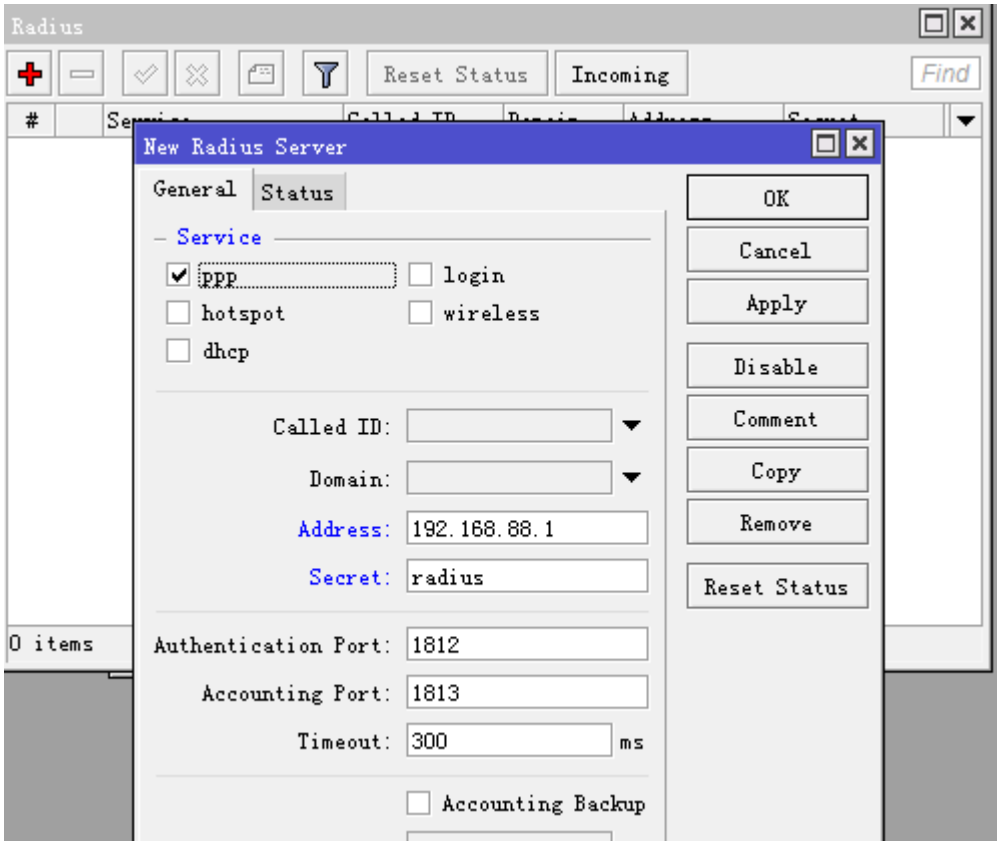


添加完成后，我们的 User manager 就与 192.168.88.2 的 RouterOS 认证设备建立连接，当然这台 RouterOS 也要在 RADIUS 里配置与该 User Manager 的对接参数。

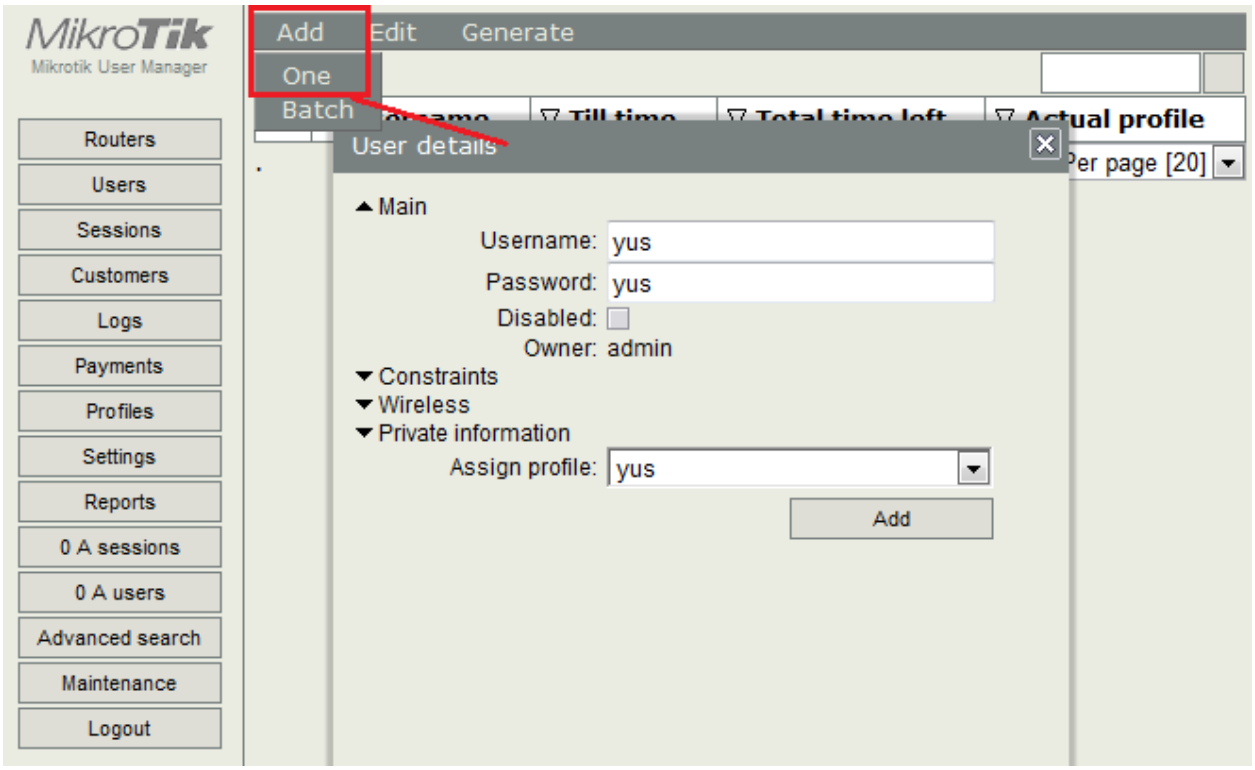
命令行操作

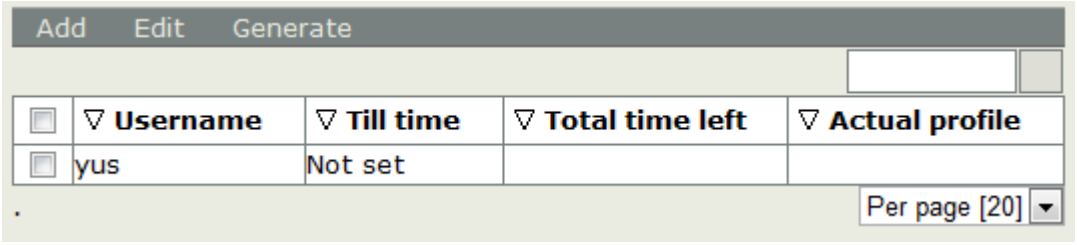
```
[admin@MikroTik] /tool user-manager router> add customer=admin ip-address=192.168.88.2
name=pppoe shared-secret=RADIUS
```

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
User details
Main
Username: yus
Password: yus
Disabled:
Owner: admin
Constraints
Wireless
Private information
Assign profile: yus
Add

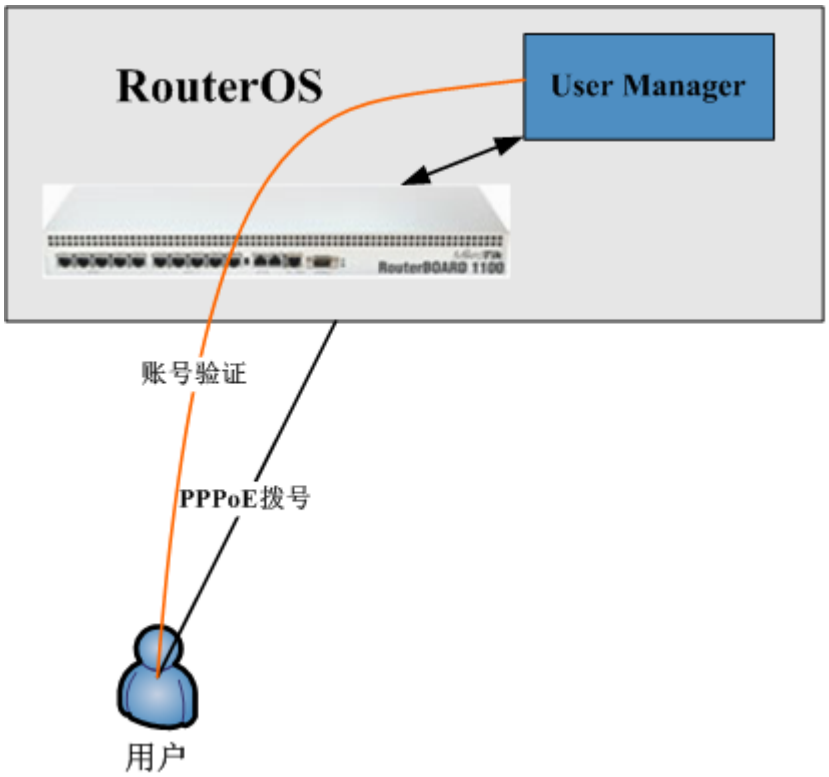


通过命令行操作

```
[admin@MikroTik] /tool user-manager user> add customer=admin name=yus password=yus
shared-users=1
```

37.3 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，

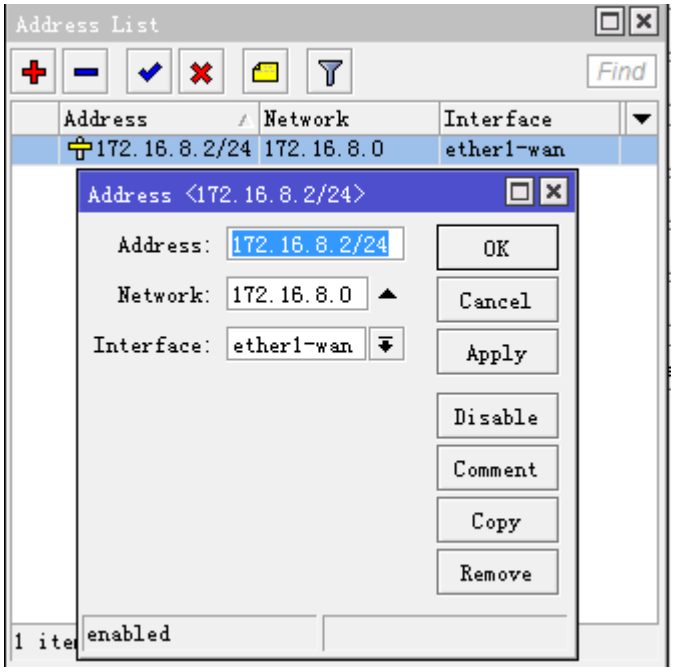


RouterOS 的配置如下：

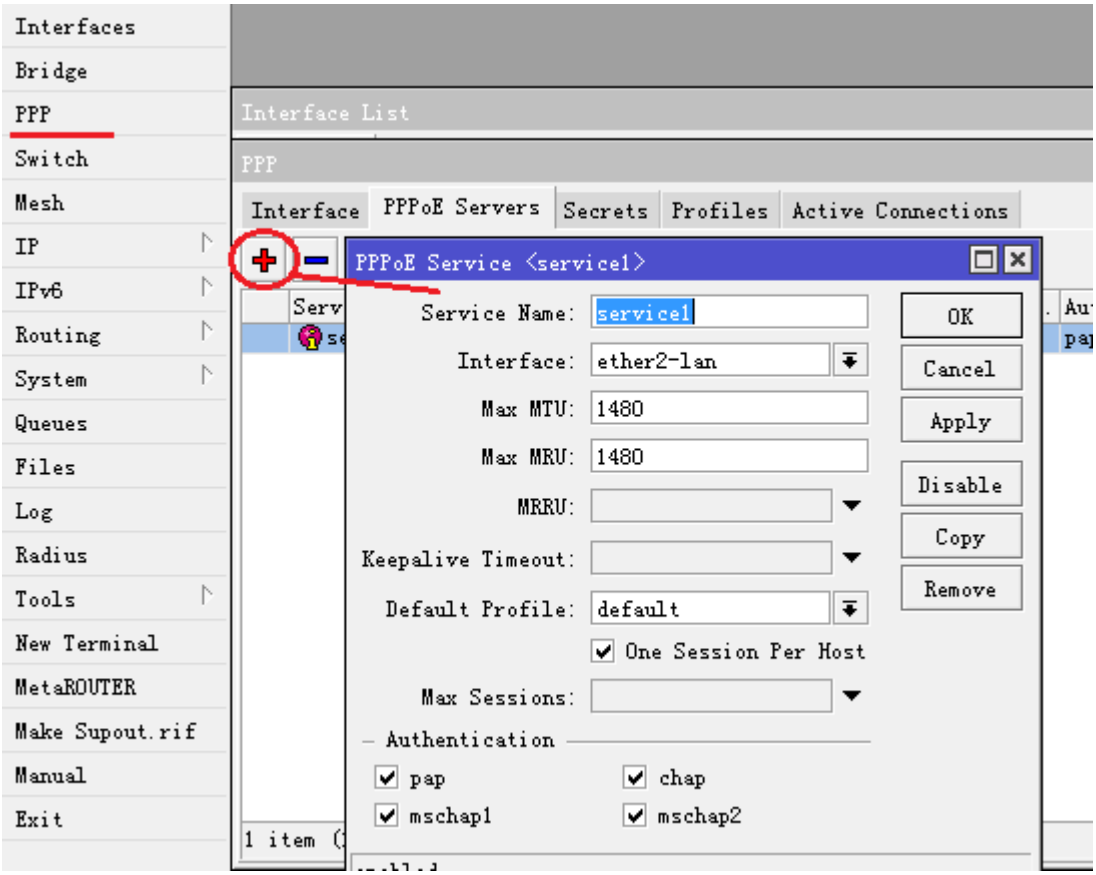
- WAN 口 IP 地址为 172.16.8.2，网关是 172.16.8.1
- LAN 口采用 PPPoE 验证，可以不需要设置 IP 地址
- 启用 User Manager，并与本地的 RouterOS 连接，用户账号通过 User Manager 分配

第一步：配置 RouterOS 的 PPPoE 认证

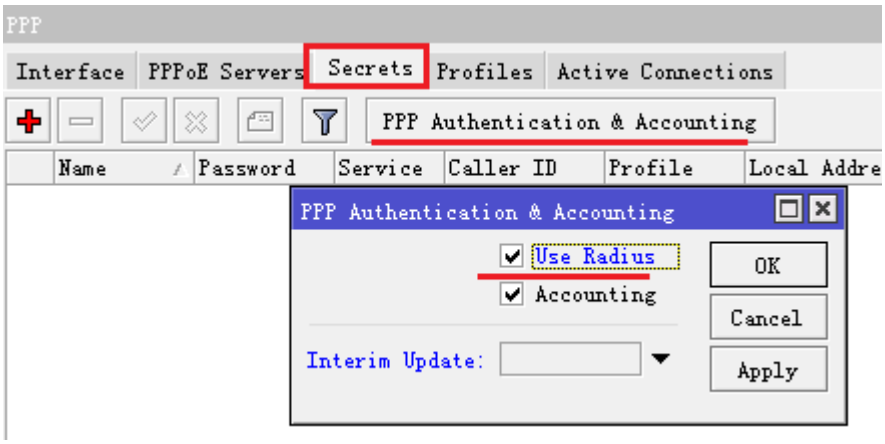
具体的 PPPoE 可以参考 PPPoE 一章，这里我们作简单的关于与 User Manager 配置的介绍，我们先进入 ip address 添加 WAN 口 IP 地址



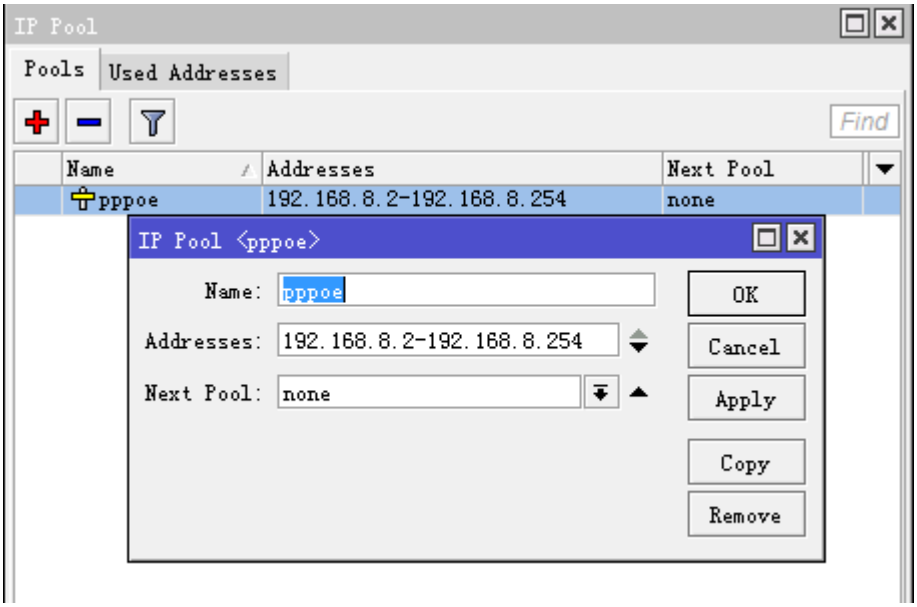
我们配置 PPPoE 服务器，进入 ppp 选择 PPPoE Service，添加并启用一个 interface=ether2-lan 的 PPPoE 服务器



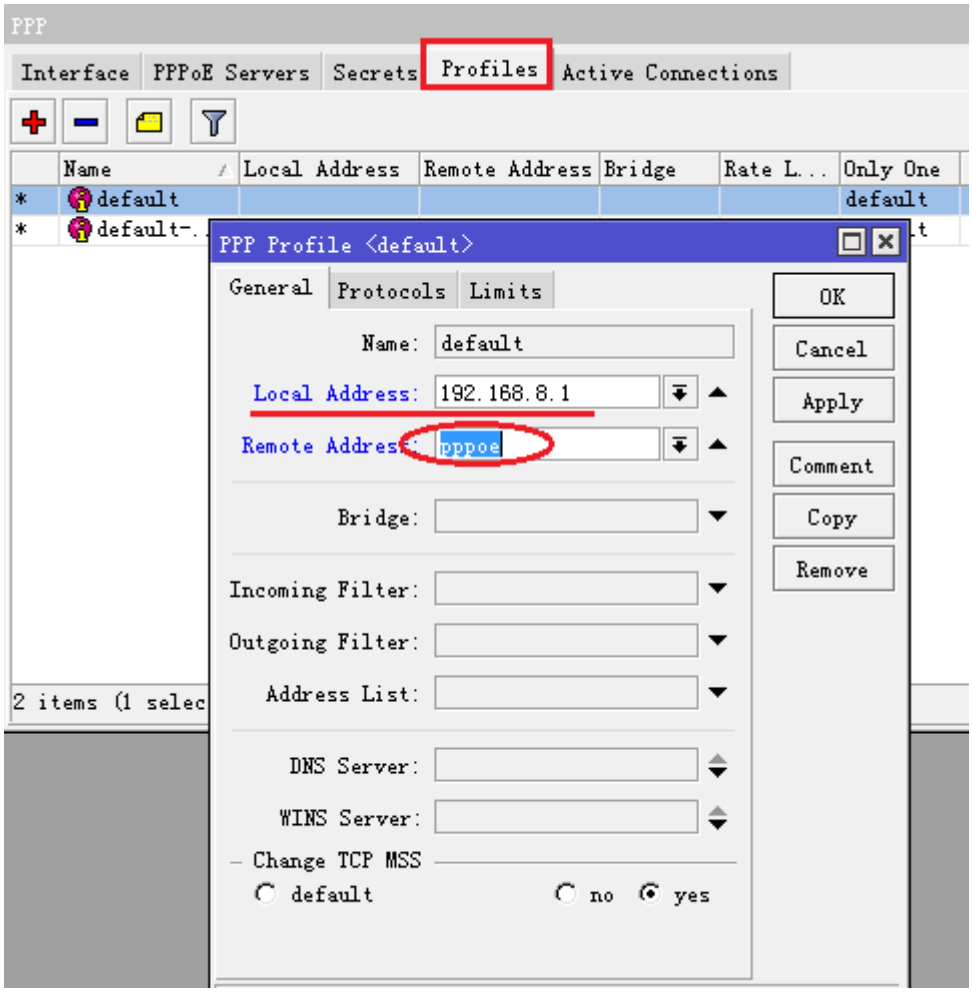
启用 PPPoE 服务器后，我们需要进入 Secrets 里将 PPP Authentication & Accounting 设置里的 Use RADIUS 选择上



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，我们给用户端分配的地址范围是 192.168.8.2-192.168.8.254，取名 pppoe



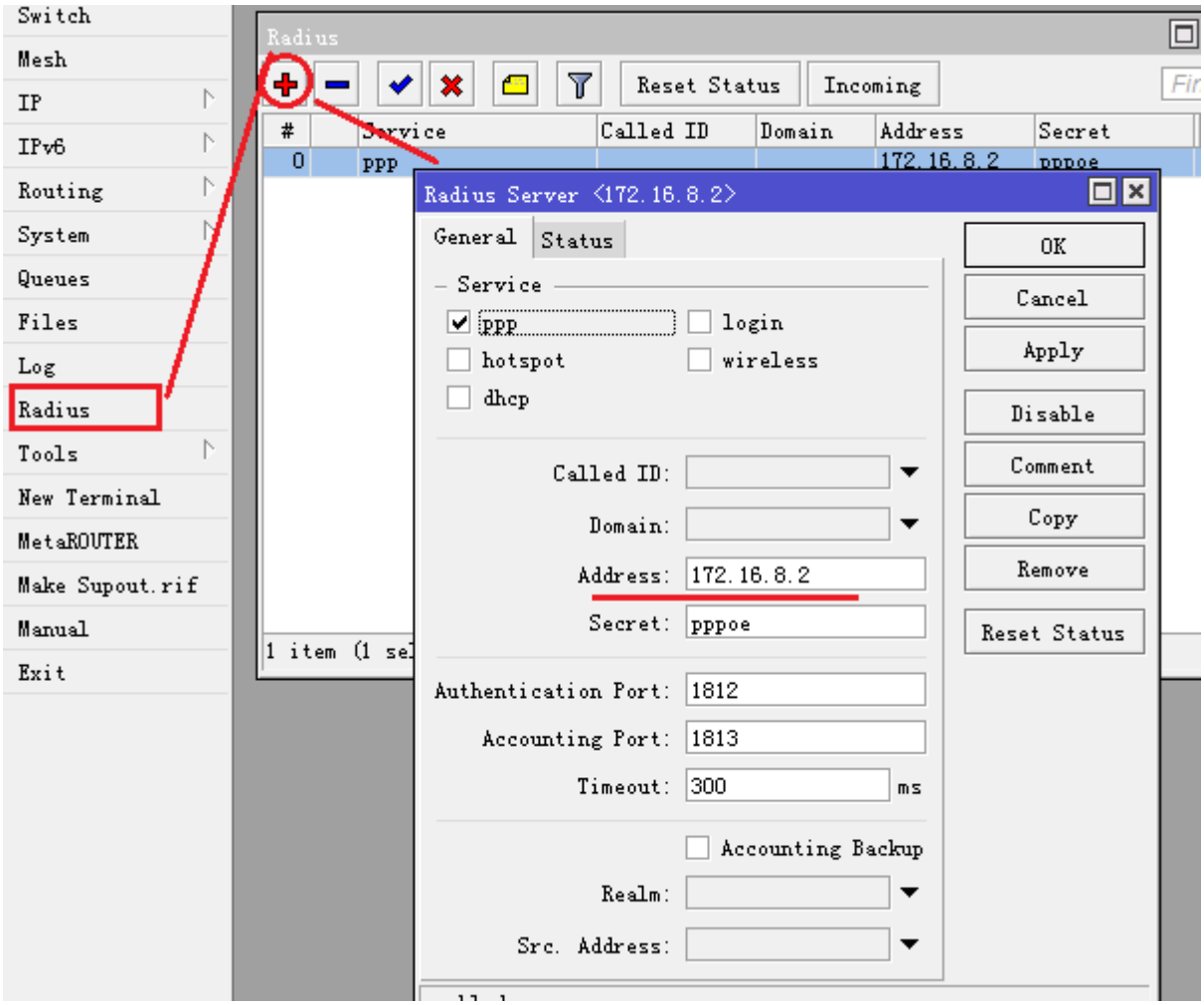
Pool 定义完成后，我们在 ppp profile 里 default 设置用户地址分配，我们设置用户端的网关是 192.168.8.1，



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

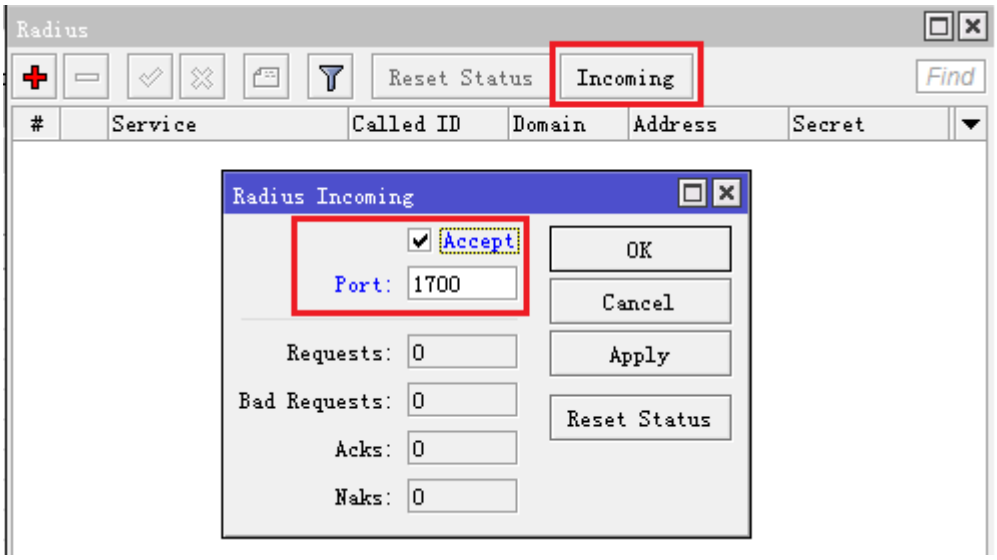
以上基本完成 PPPoE 服务器的配置，当然你需要设置其他的路由网关和 nat 转换，这里不再做介绍

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

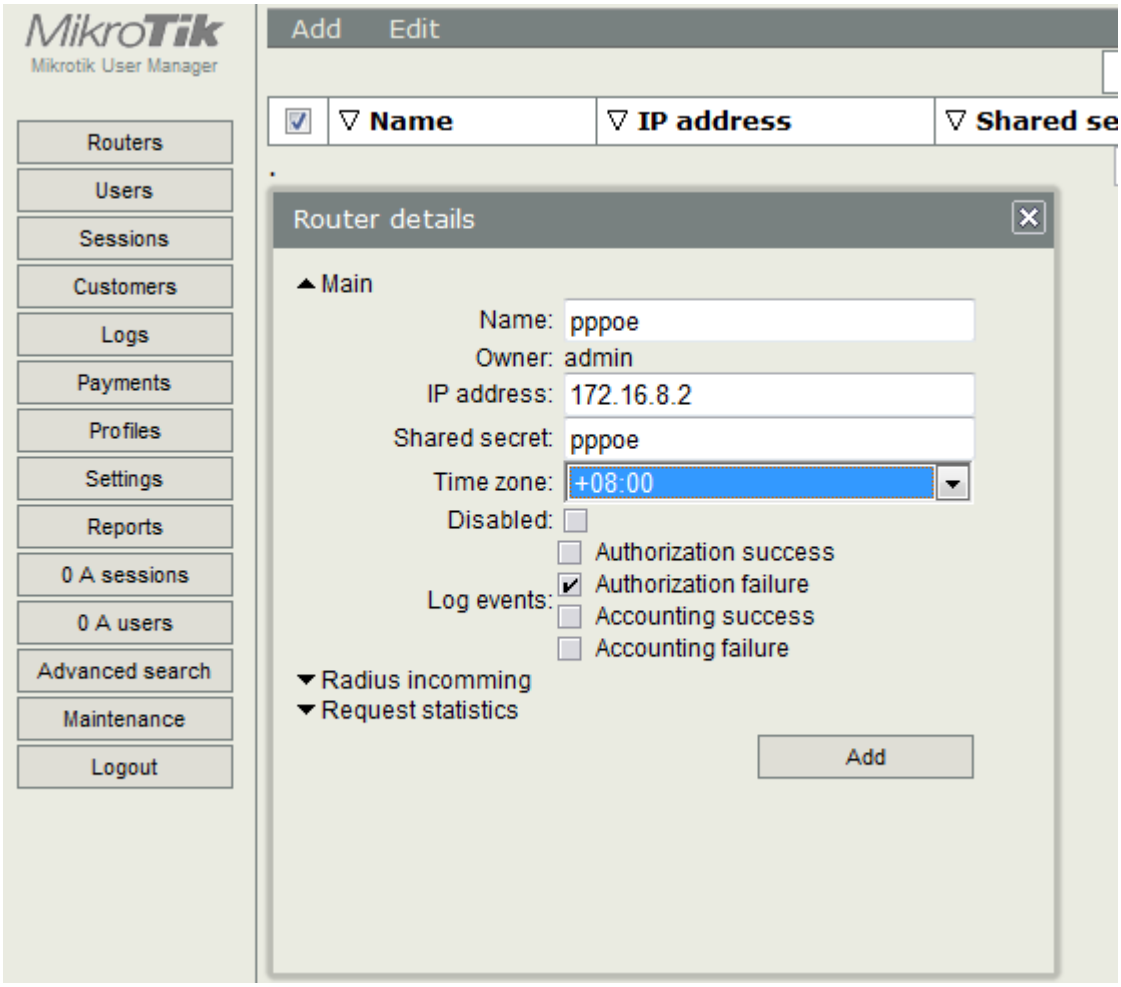
配置 incoming 参数，默认端口是 1700，incoming 参数是由 RADIUS 向 RouterOS 发送指令请求，例如在 User Manager 中剔除在线用户



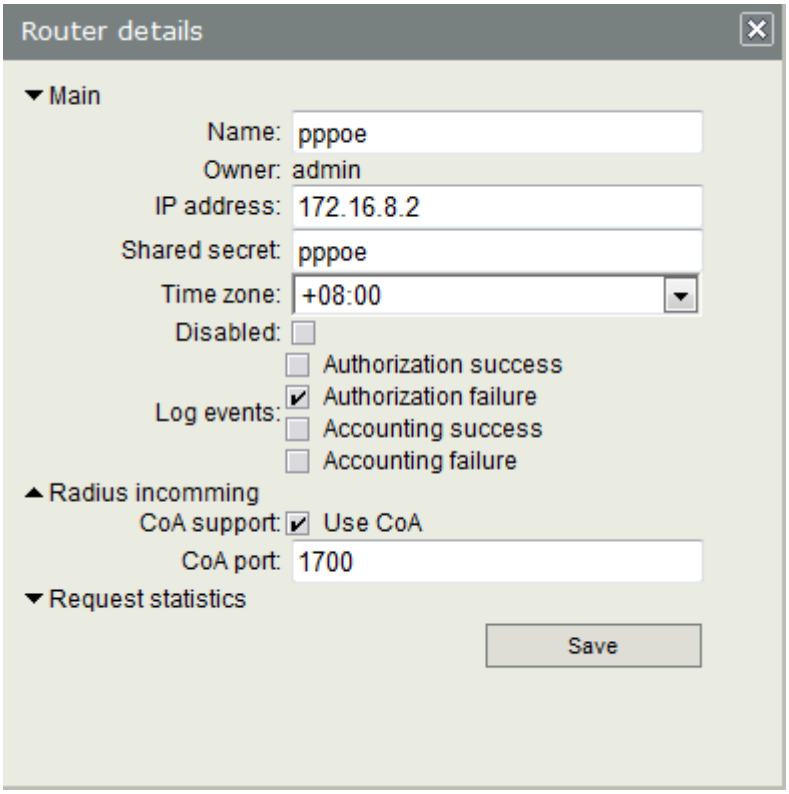
第三步，User Manager 配置

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。

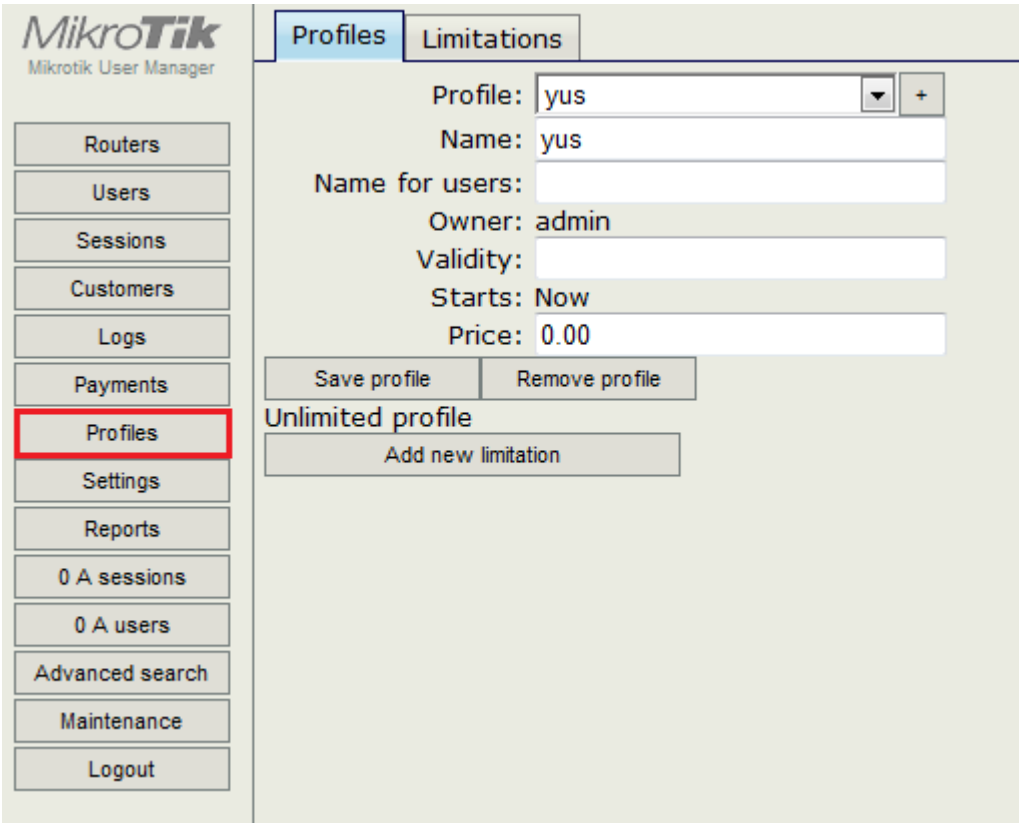
我们先进入 Router 里添加参数，IP address 同样是本地的 WAN 口 172.16.8.2，Shared secret 同样是 pppoe



在 RADIUS incomming 里开启端口

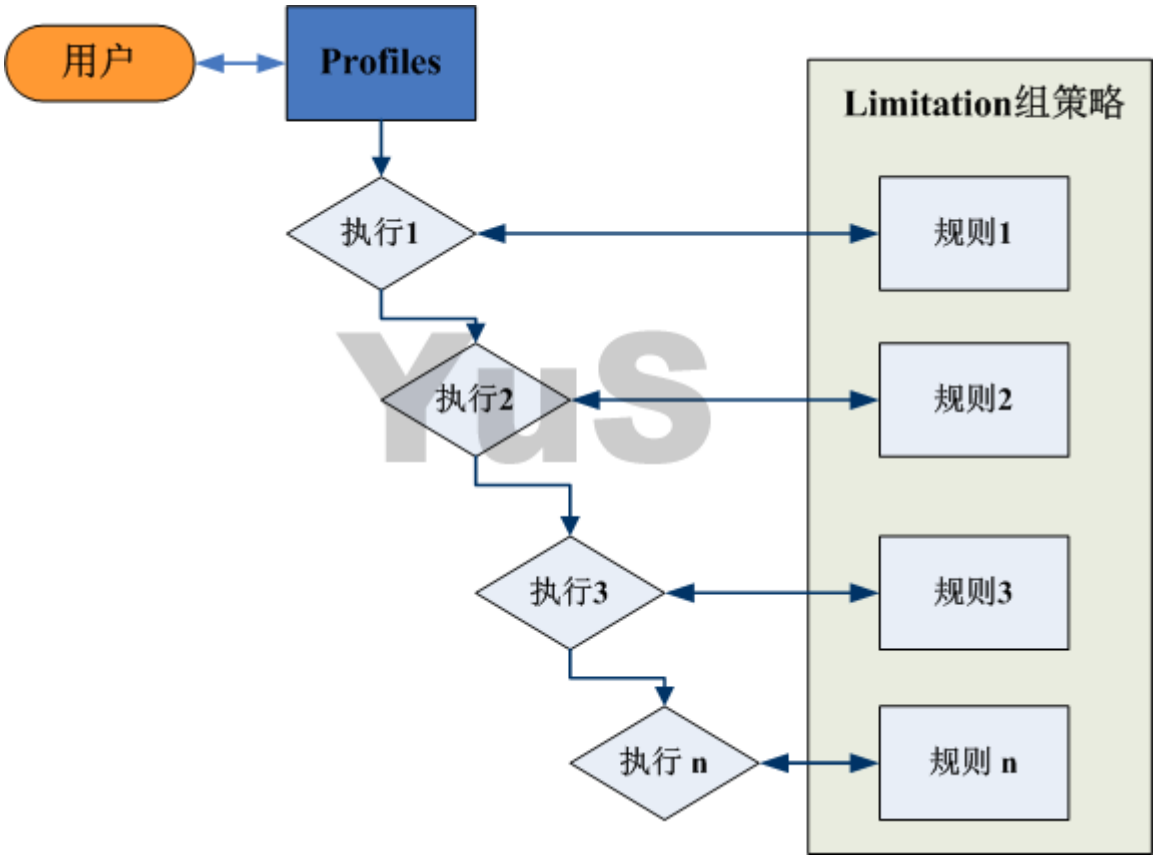


添加完成 Router 参数后，基本完成了 PPPoE 验证与 User Manager 的对接，接下来就是添加用户账号，在添加用户账号前我们需要设置对应 proflies 规则



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

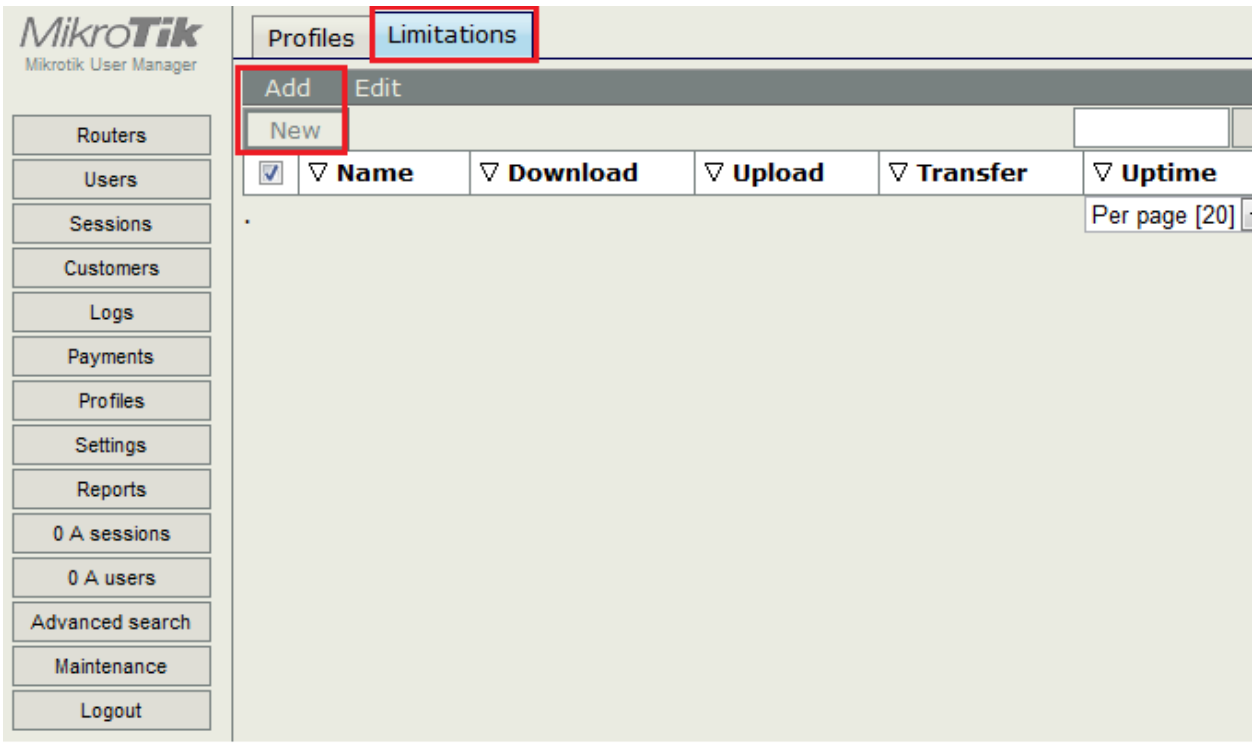
新的 User Manager 与以往的不同，引入了 Profiles 策略组，Profiles 组策略包括 Limitation 限制策略，我们可以建立多条 Limitation 策略，这些策略可以由 Profiles 调用，并设置他们的执行周期，即可以设置周一到周日我们可以选择执行那一条 Limitation 策略，实现不同时段的服务，如下图



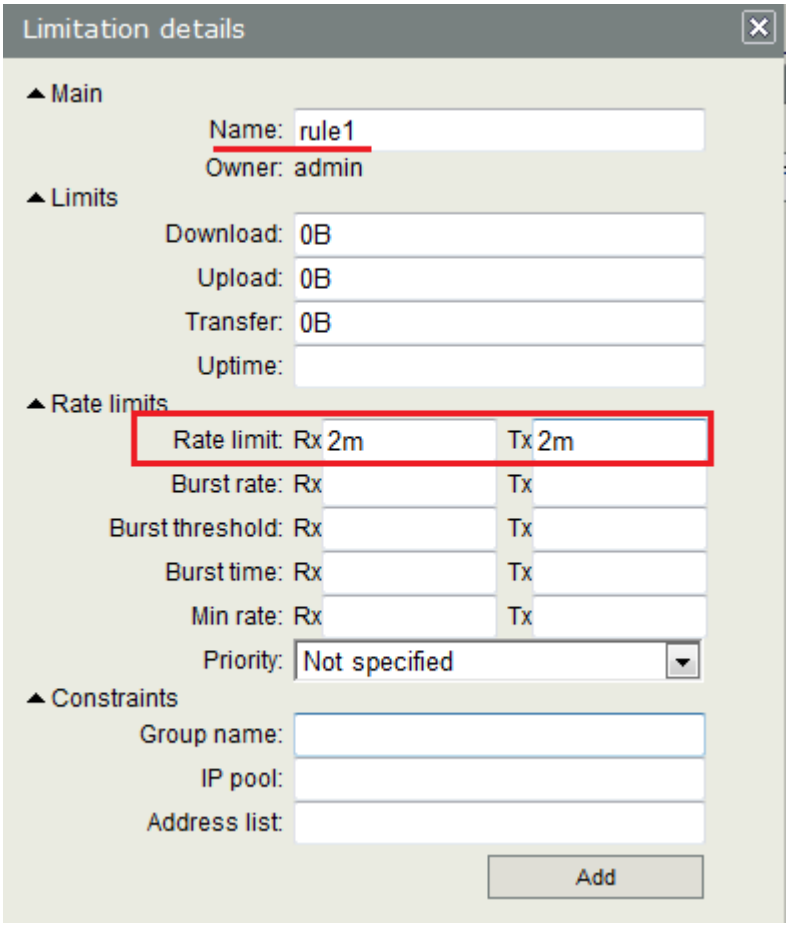
Limitation 的规则在添加后，并没有实际作用，只有在被 profiles 选择后才被调用，同时 User 里的用户账号也调用对应的 Profiles 策略，即一环扣一环的策略方式（其实在后面开发的 NAT RADIUS 也是如此，引入功能和策略上还要比这个复杂点，可惜没有找到好归属）

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）

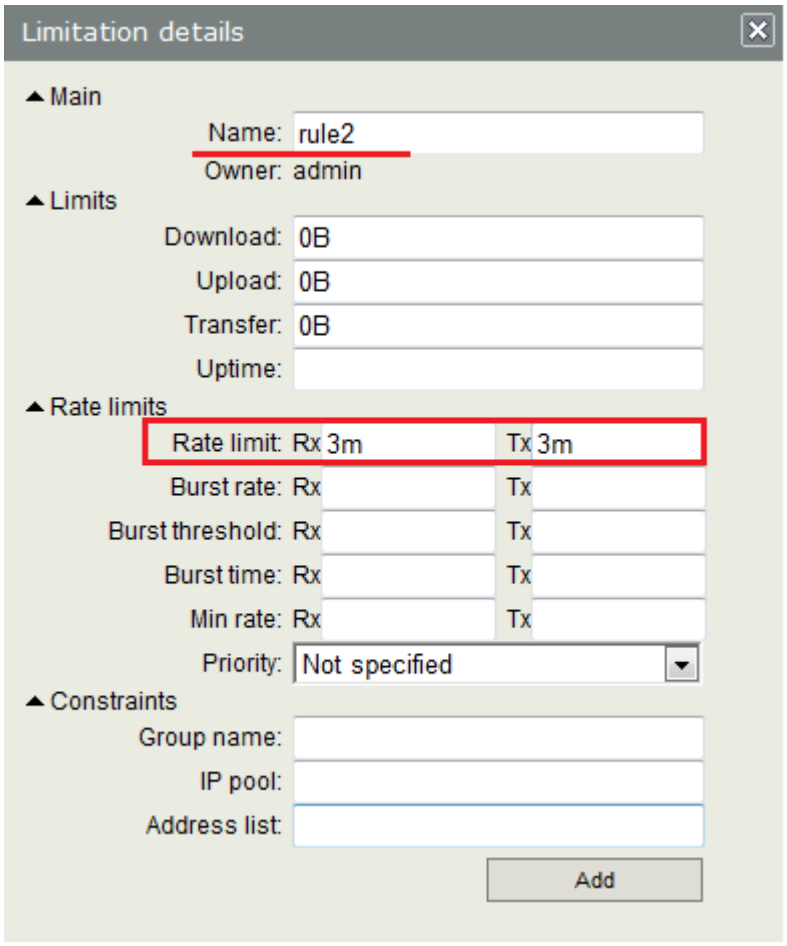
我们首先要在 limitation 里添加 2 个规则，定义星期一到星期五带宽 2M 为 rule1，星期六到星期天 3M 为 rule2，进入 Profiles 的 limitations

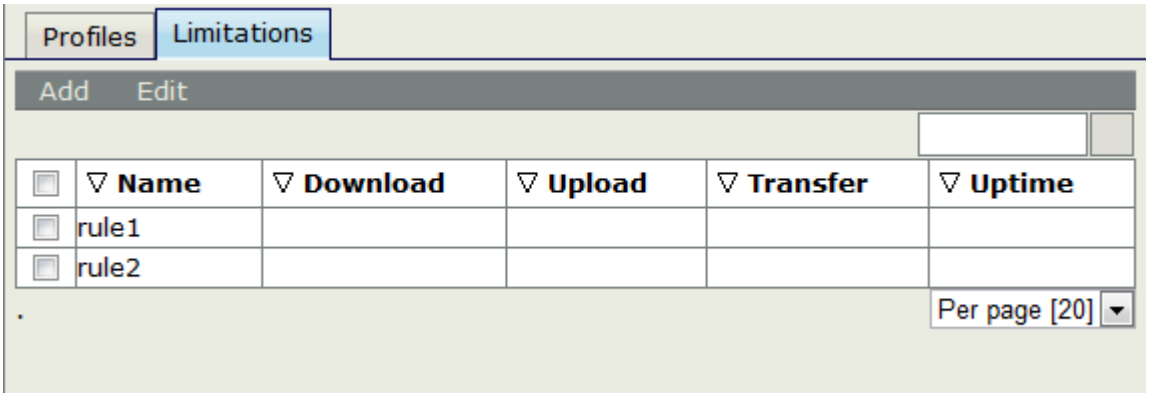


我们添加第一条规则，取名 rule1



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

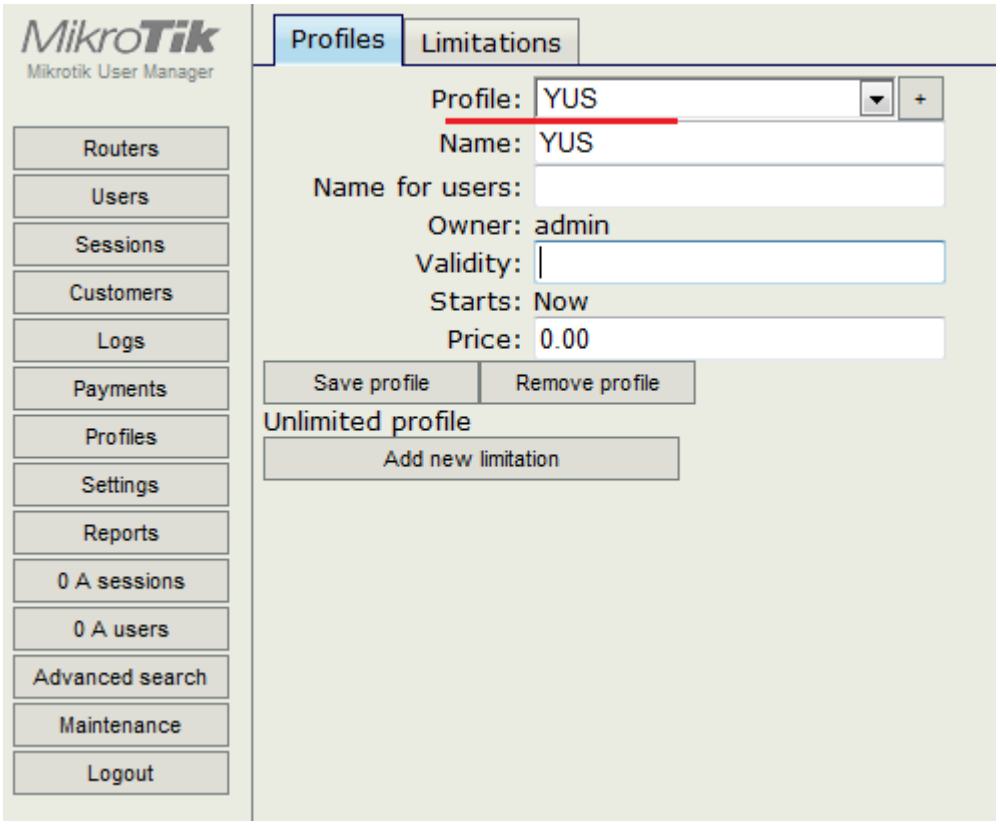
- Download 用户下载流量限制
- Upload：用户上传流量限制
- Transfer：用户总流量限制
- Uptime：用户上线时间限制

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

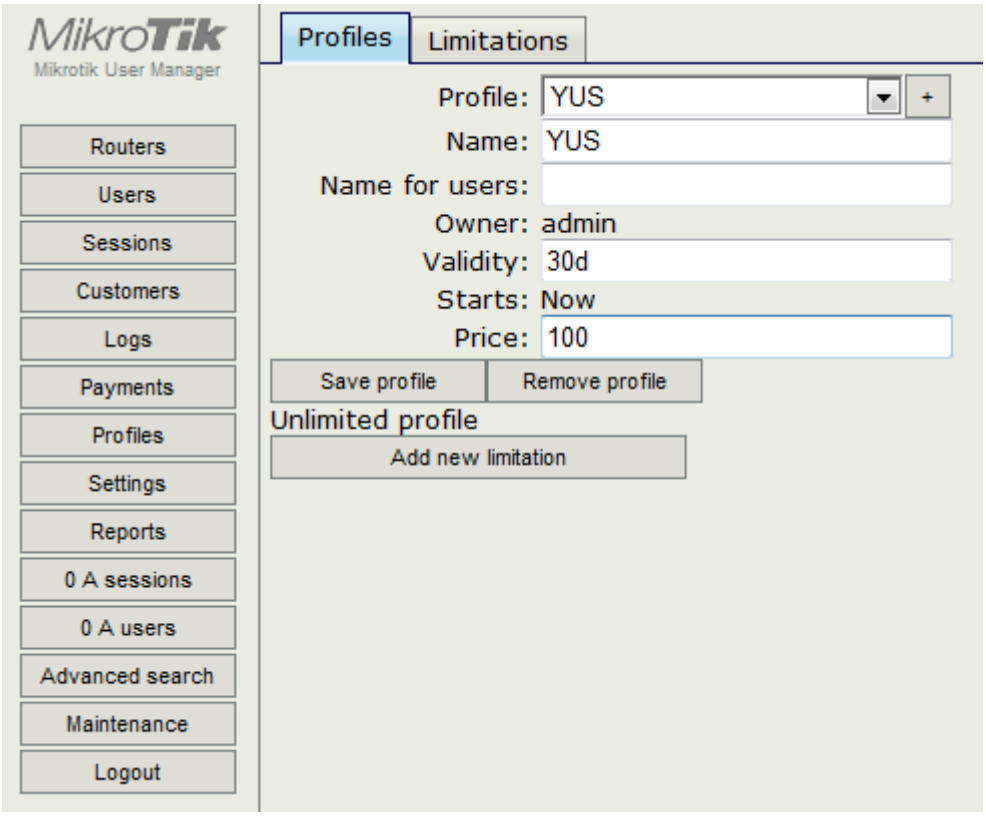
Constraints 栏，是约束限制，包括了 Hotspot 的组策略，IP Pool 地址池和 address-list，这些参数都是可以从 RouterOS 里直接调用

- 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

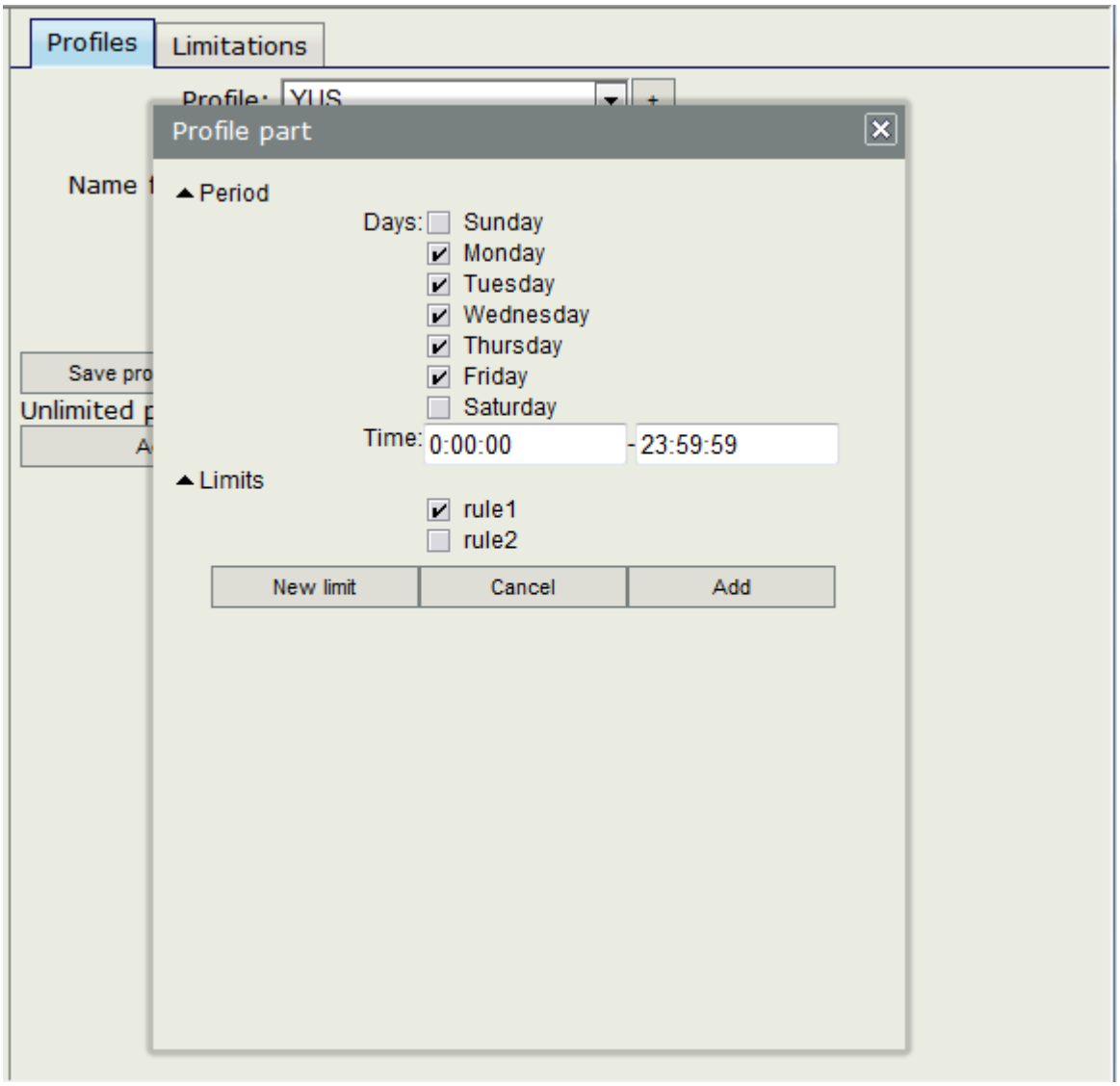
我回到 profile 页面，并添加一个 YUS 的 Profile 规则



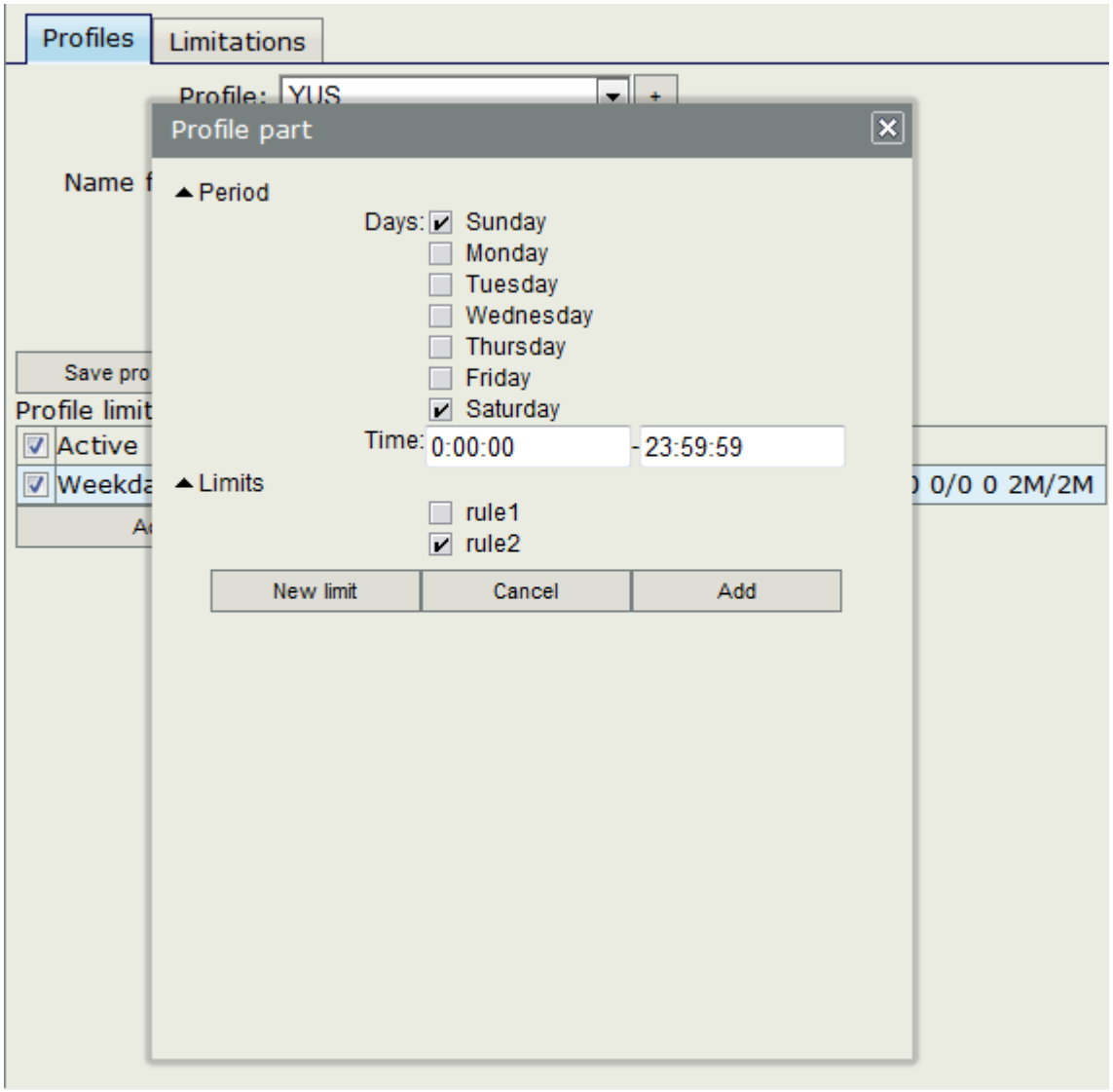
Validity 是有效期，我们设置 30d（30 天），Price：当然是给用户充值的金额，我们可以设置 100 元



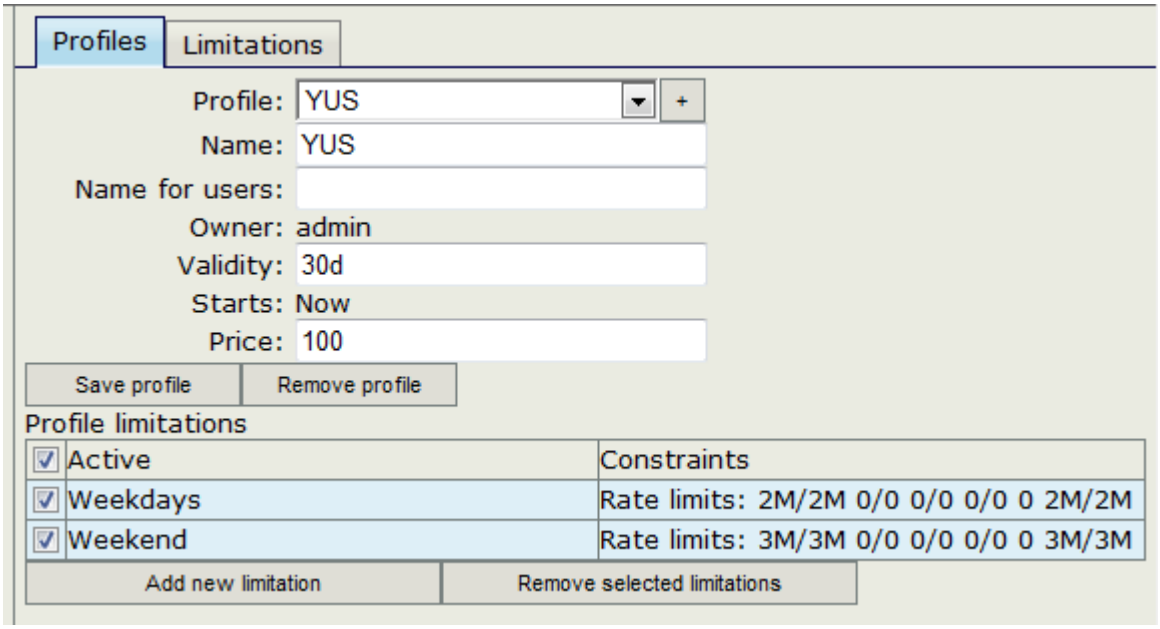
当设置完成后，我们在下面的 add new limitation 添加我们的组规则，首先我们添加星期一到星期五的策略，选择 rule1



接下来定义星期六和星期天的规则

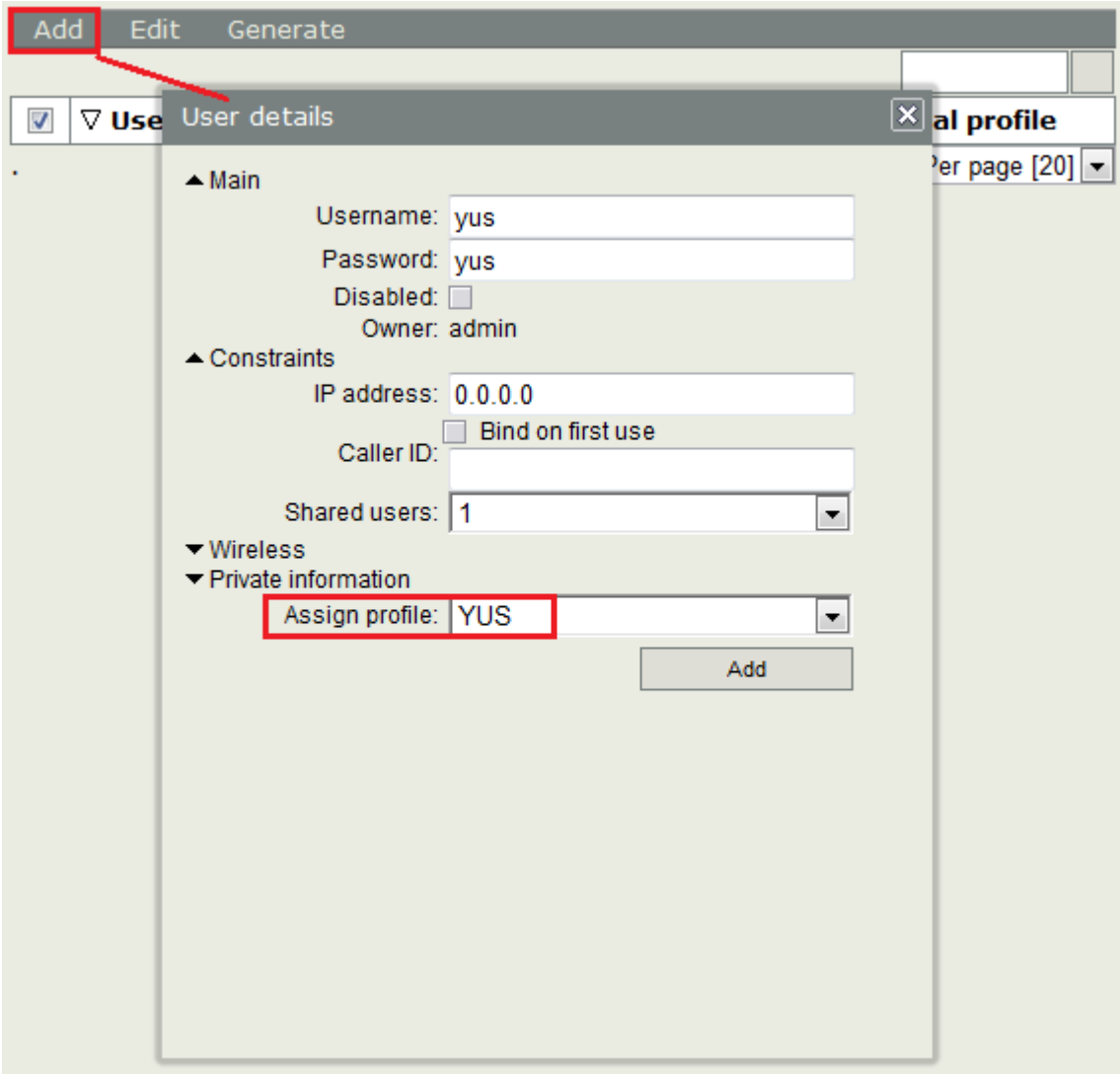


定义完成后，如下所示



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

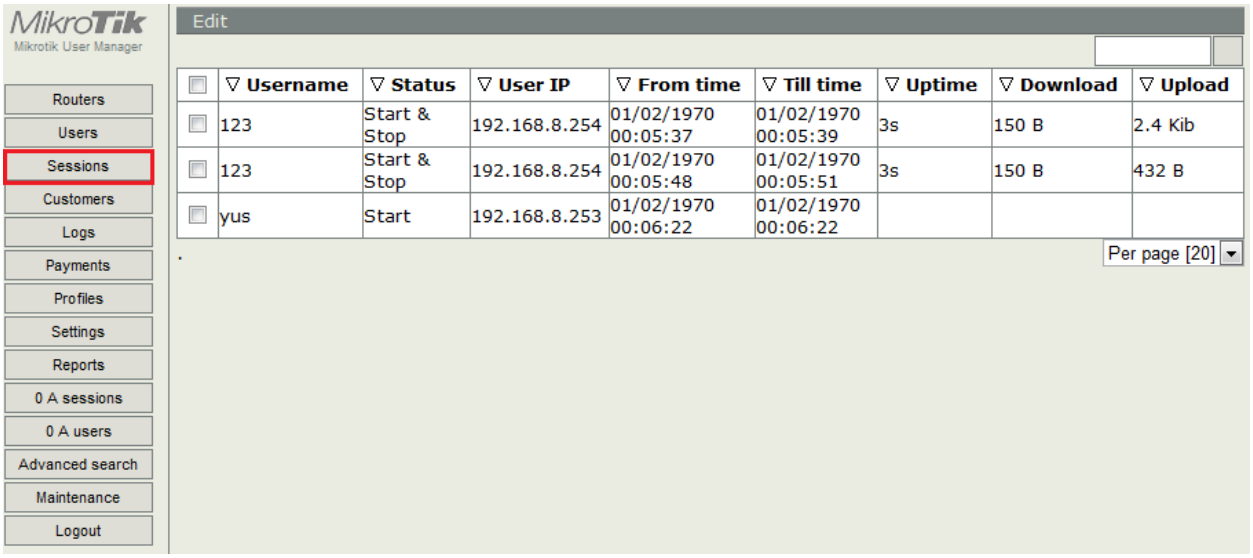
Profile 定义完成后，我们就在 Users 里添加我们需要新增的账号，取名 yus，并选择 Profile 为 YUS



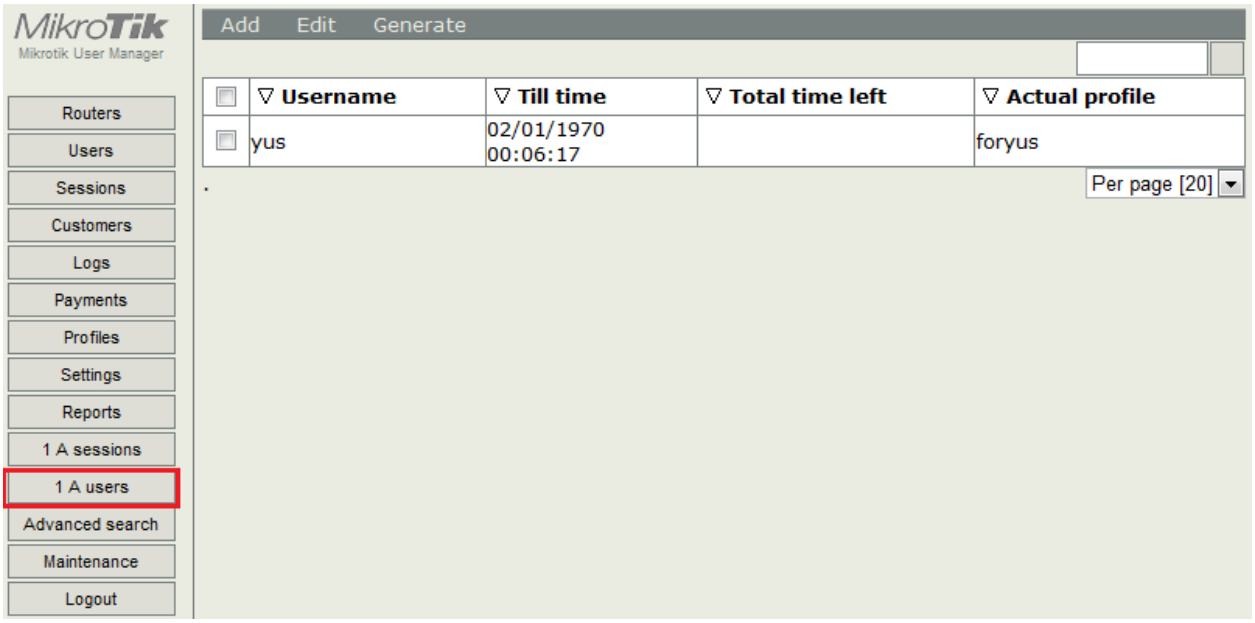
在 User 添加规则时，我们可以看到 constraints 有 3 个参数，他们可以根据你的需要定义

- 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

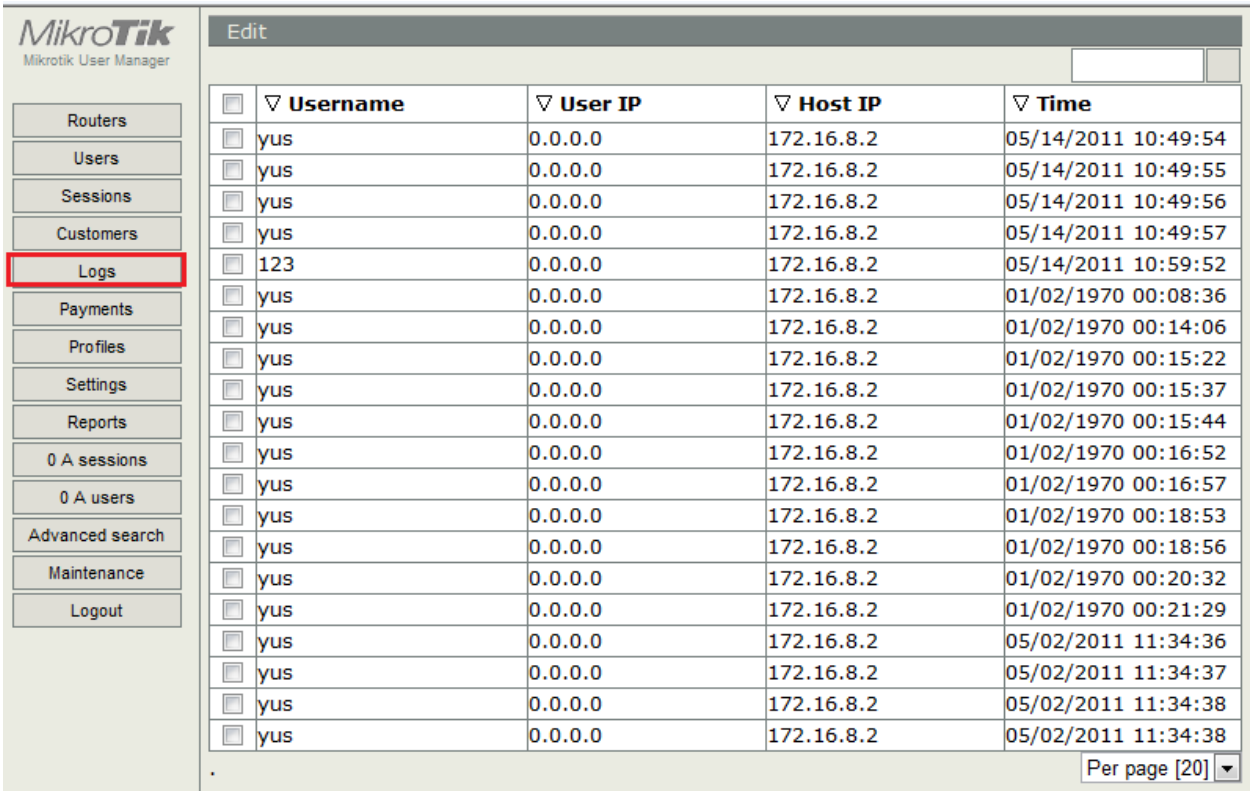
这样从设置 User Manager 连接到添加策略和用户账号基本完成，之后可以使用 yus 账号登录。我们可以在 Sessions 菜单里查看账号登录情况



在 A users 里可以看到当前在线用户

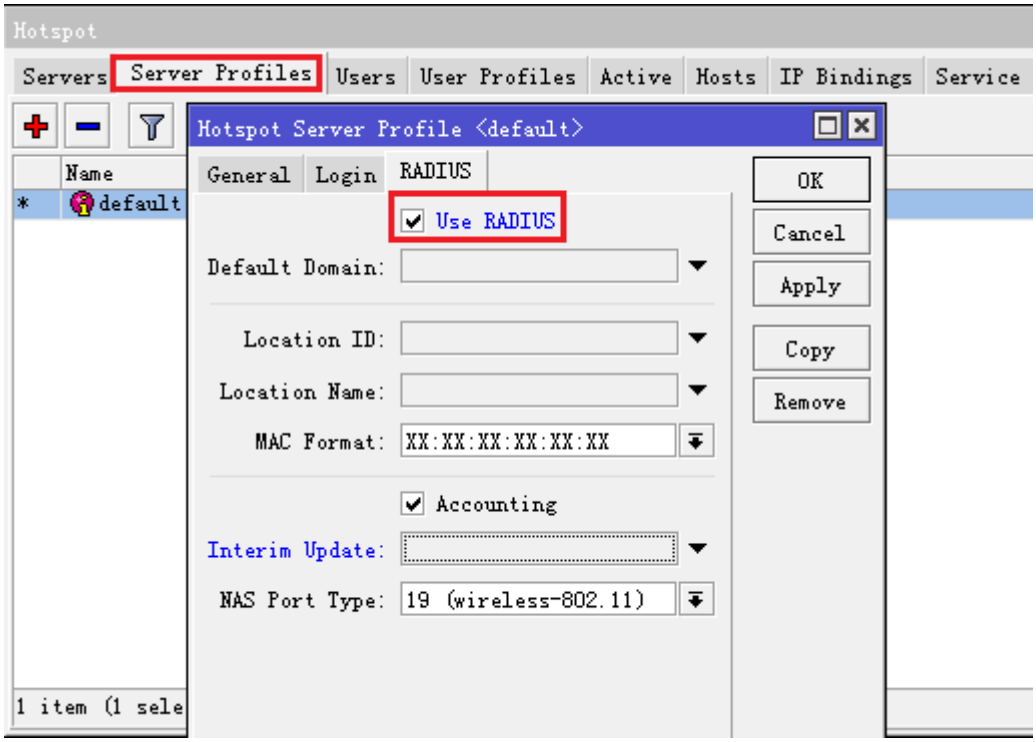


我们可以在 log 中查看用户登录日志

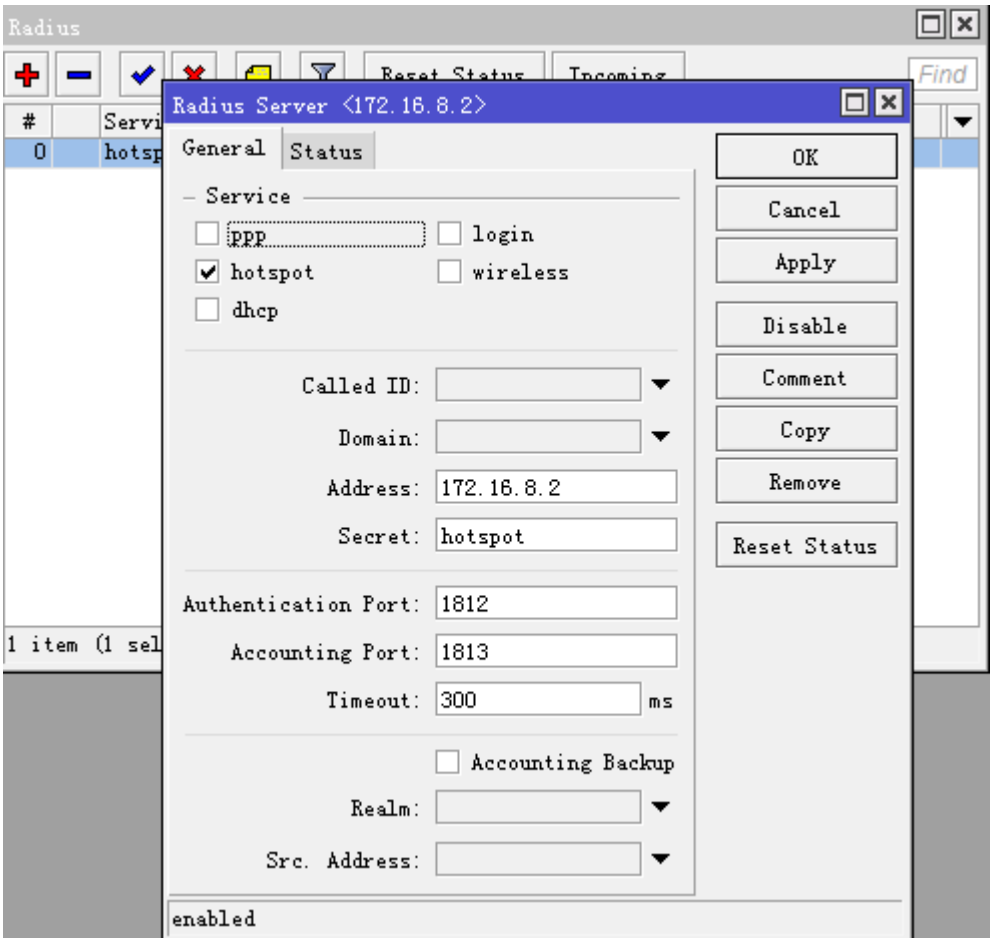


37.4 Hotspot 认证 User Manager 连接

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



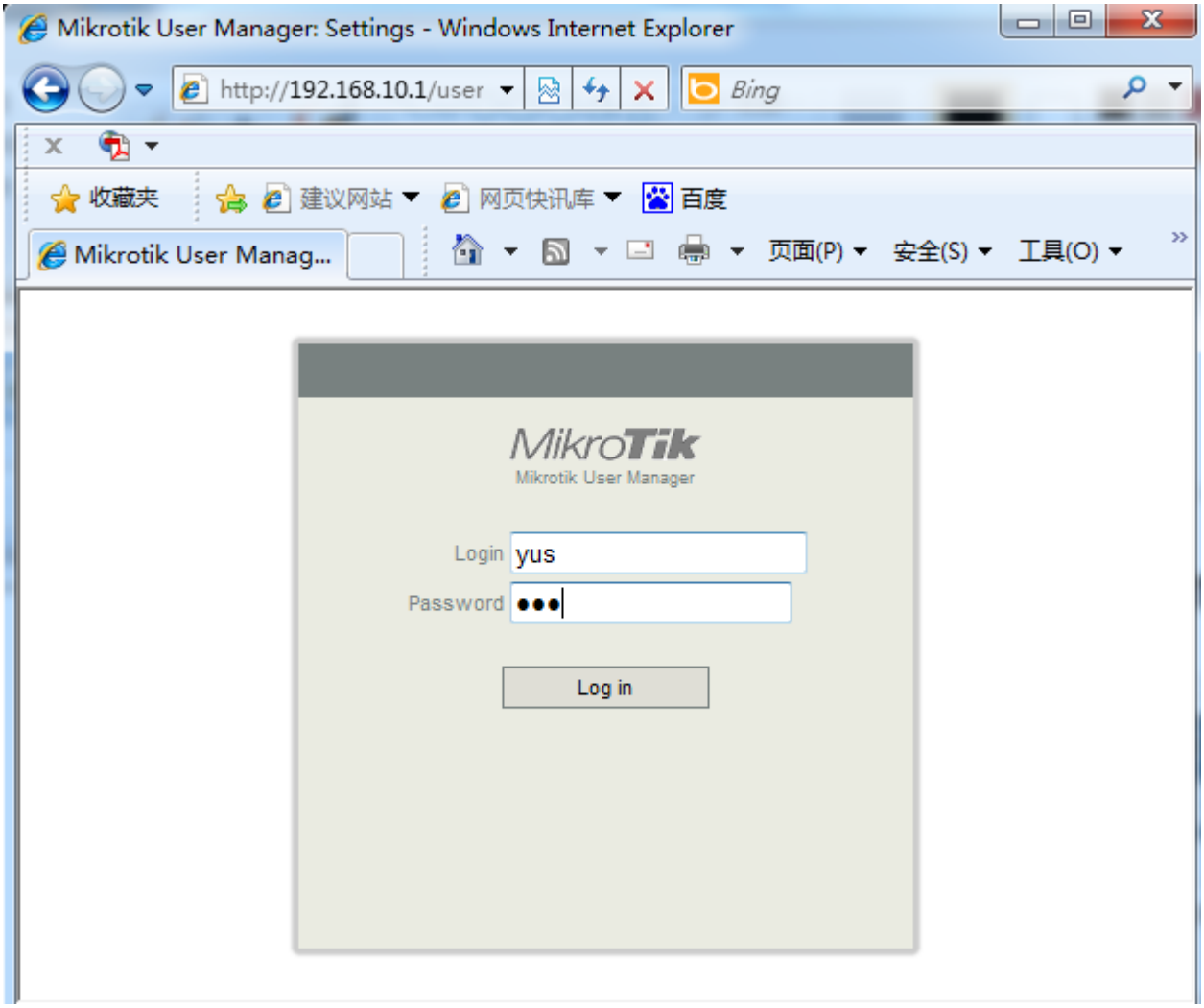
其他的 User Manager 参数相同，仅需要修改 Secret 参数。

37.5 用户自助服务

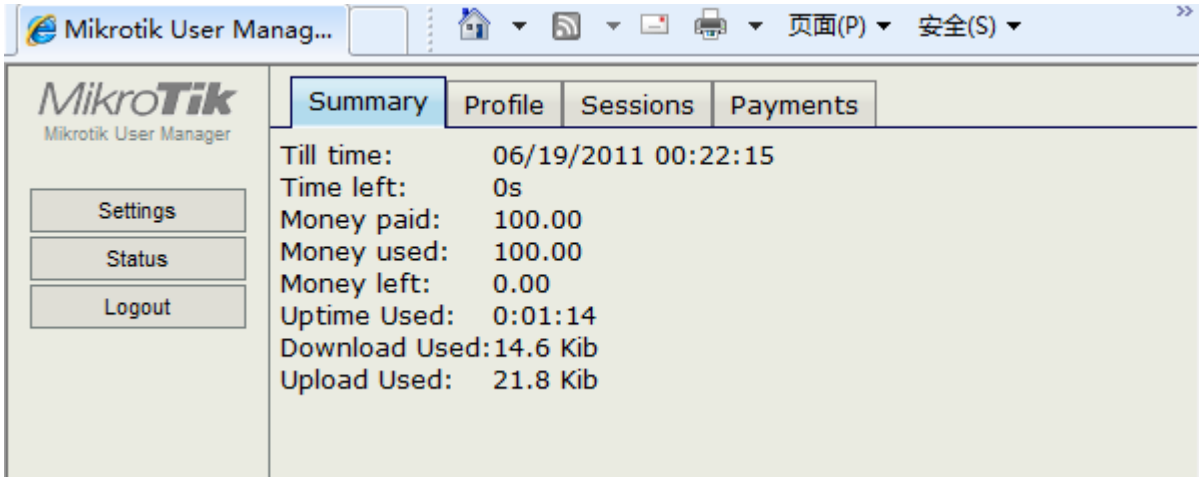
User Manager 支持用户通过网页访问，并让用户登陆到自己的账号进行自助查看和设置，用户通过 http://router_IP/user 页面下设置，你可以为用户自助访问页面做一个静态的域名，方便用户能直接访问。

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

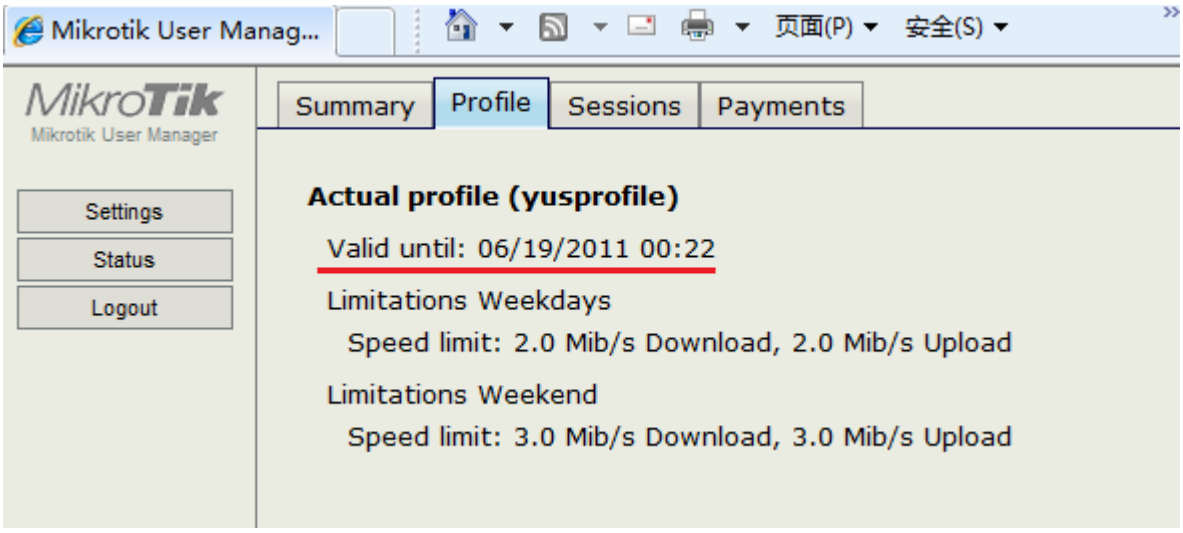
在 user 页面，我们使用的是用户自己的账号，如账号 yus，密码是 yus



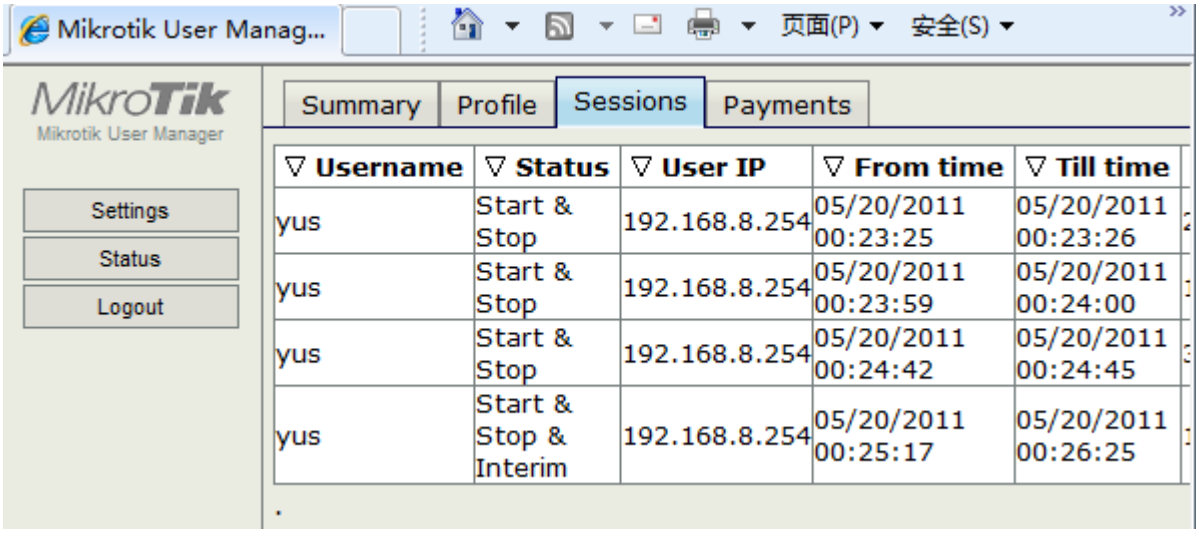
我们登录后，首先看到 Summary 页面，即用户基本使用情况



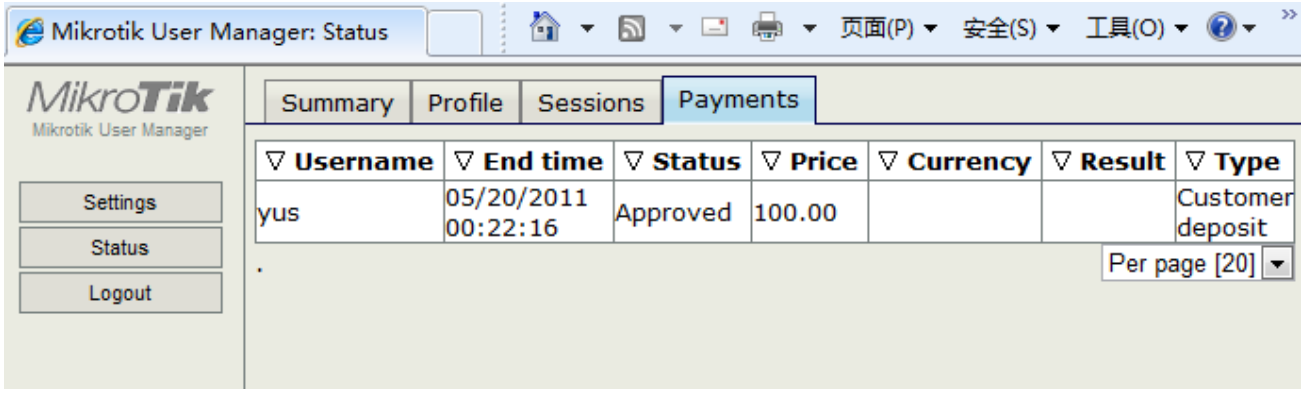
点击 Profile 标签，可以查看策略组的情况，如 Valid until 有效期和流量策略情况



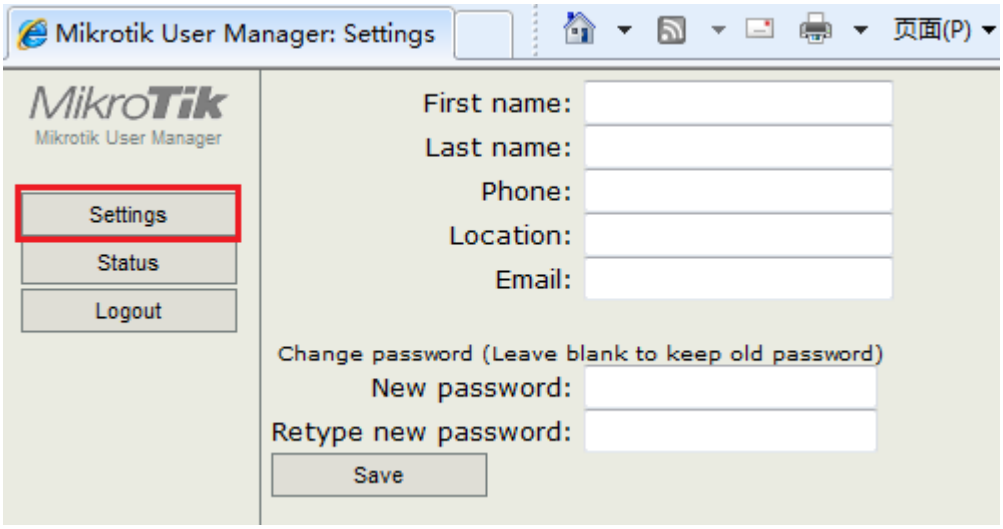
在 Sessions 里用户可以查看自己登录情况，如状态、使用的 IP 地址，登录时间和结束时间



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：


Mikrotik User Manager: Settings
页面(P)
MikroTik
Mikrotik User Manager
Settings
Status
Logout
First name:
Last name:
Phone:
Location:
Email:
Change password (Leave blank to keep old password)
New password:
Retype new password:
Save

第三十八章 Scheduler（计划任务）

设置的计划任务安排，通过预设值时间或周期安排执行相应的脚本操作，计划任务能有效的实现管理预期的任务，说的高级点就是自己编写智能路由器。在后面的 Script 章节将介绍如何使用脚本编程。

规格

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

38.1 计划任务介绍

计划任务列表能触发脚本执行，在指定的时间或者是在指定的时间周期执行任务。

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。

38.2 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
```

Schedule 在 Winbox 的配置如下：

事例 2：添加 2 个脚本改变带宽设置队列规则"cust0"，每天上午 9 点限制为 64kb/s，下午 5 点限制为 128kb/s。这个队列的规则、脚本和计划任务如下(**注：**在 2.9 种 cust0 是不需要加双引号的，但在 3.0 中需要注明字符串，要加上双引号"cust0")：

先添加 queue simple 规则：

```
[admin@MikroTik] queue simple> add name=Cust0 interface=ether1 \
\... target-address=192.168.0.0/24 limit-at=64000
[admin@MikroTik] queue simple> print
Flags: X - disabled, I - invalid
  0    name="Cust0" target-address=192.168.0.0/24 dst-address=0.0.0.0/0
       interface=ether1 limit-at=64000 queue=default priority=8 bounded=yes
```

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 \
\... "Cust0" limit-at=64000}
[admin@MikroTik] system script> add name=stop_limit source={/queue simple set \
\... "Cust0" limit-at=128000}
[admin@MikroTik] system script> print
```

```
 0 name="start_limit" source="/queue simple set "Cust0" limit-at=64000"
   owner=admin run-count=0

 1 name="stop_limit" source="/queue simple set "Cust0" limit-at=128000"
   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-64k" \
\... start-time=9:00:00 on-event=start_limit
[admin@MikroTik] system scheduler> add interval=24h name="set-128k" \
\... 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-64k    start... oct/30/2008 09:00:00    1d                    0
 1    set-128k   stop_... oct/30/2008 17:00:00    1d                     0
[admin@MikroTik] system scheduler>
```

事例 3：下面的例子安排了一个通过电子邮件发送每周备份路由器配置信息的脚本：

```
[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>
```

不要忘记去设置电子邮件参数，即 SMTP 服务的配置，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：下面的例子是从午夜 12 点到正午 12 点的每个小时里把“x”加进日志中：

```
[admin@MikroTik] system script> add name=enable-x source={/system scheduler
{... enable x}
[admin@MikroTik] system script> add name=disable-x source={/system scheduler
{... disable x}
[admin@MikroTik] system script> add name=log-x source={:log info "x"}
[admin@MikroTik] system script> .. scheduler
[admin@MikroTik] system scheduler> add name=x-up start-time=00:00:00 \
\... interval=24h on-event=enable-x
[admin@MikroTik] system scheduler> add name=x-down start-time=12:00:00
\... interval=24h on-event=disable-x
[admin@MikroTik] system scheduler> add name=x start-time=00:00:00 interval=1h \
\... on-event=log-x
[admin@MikroTik] system scheduler> print
Flags: X - disabled
  #    NAME       ON-EVENT START-DATE   START-TIME INTERVAL            RUN-COUNT
  0    x-up        enable-x   oct/30/2008      00:00:00       1d                 0
  1    x-down       disab...    oct/30/2008       12:00:00       1d                  0
  2    x            log-x       oct/30/2008      00:00:00       1h                 0
[admin@MikroTik] system scheduler>
```

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

6.2.1e 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

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

6.2.3e 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

6.2.4e 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 配置文档修正

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

6.3.1e 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

6.3.2e 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。

6.3.3e 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

参考文献：

http://wiki.mikrotik.com

http://www.routerboard.com