YesLab CCNA 实验手册

目录

此实验手册为测试版，难免有出错和不妥的地方，敬请谅解。如果你在做实验的过程中有发现任何错误，或者你有任何建议。请你记录下来联系我，我会视情况做出改正，并且为你署名。谢谢！此实验手册由 YESLAB 实验室苏函出品。（此实验手册最终会出版成书，想一想如果你能够署名，嘿嘿）

Yeslab -Learning on demand
www.yeslab.net
RS & ISP QQ Group: 168121608 苏函 QQ:149397864
Tel: 010:82864660
Fax: 010-82684760

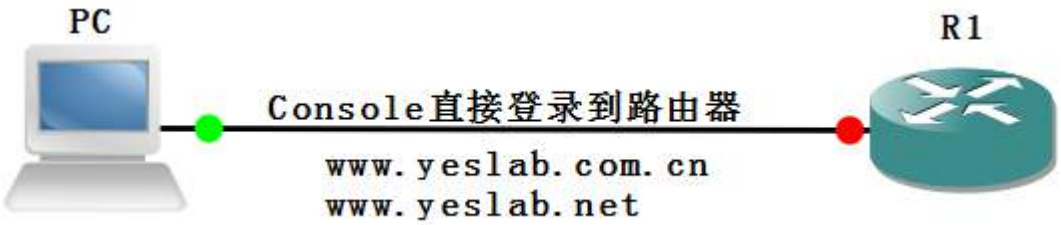
实验第一部分 路由器的基本配置

实验 1.1 认识路由器几种配置模式

实验目的

掌握进入路由器各种模式

拓扑



实验步骤

1：以下为刚进入路由器时的默认模式--用户模式，用大于号表示。

```
Router>
```

2：用户模式进入特权模式：

```
Router>enable
Router#
```

//以上是特权模式，用#号表示。

3：从特权模式进入全局模式：

```
Router#configure terminal
Router(config)#
```

//以上为全局模式，有 config 字样。

4:：为路由器设置主机名：

```
Router(config)#hostname Yeslab
Yeslab(config)#
```

//可以发现路由器的名字已经变成了 Yeslab

5：退出到特权模式：
Yeslab(config)#exit
Yeslab#

6:退出到用户模式：
Yeslab#disable
Yeslab>

7:退出控制台登陆
Yeslab>quit

说明：

路由器模式大致可分为：
1：用户模式：权限最低，通常只能使用少量的查看性质的命令，如需进入特权模式一般需要密码。
2：特权模式：可以使用更多的查看性质的命令和做一些少量的修改路由器参数命令，但是由此进入全局配置模式，并且不需任何密码。
3：全局配置模式：不能使用查看性质的命令，在此模式下做大量的对路由器的配置的模式，还可再此基础上进入接口配置模式，线路配置模式，路由进程配置模式等等。

配置实例：

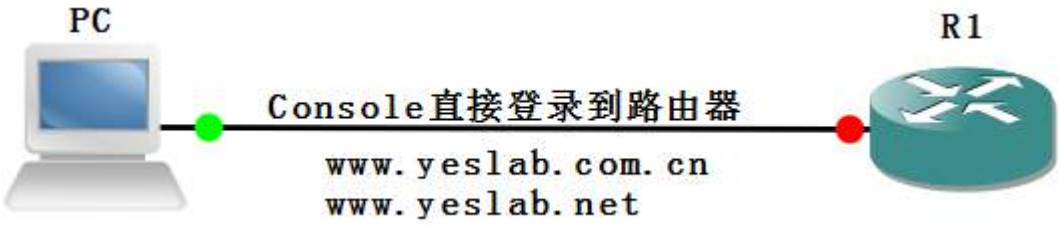
```
Router>
Router>enable
Router# configure terminal
Enter configuration commands, one per line.   End with CNTL/Z.
Router(config)#
Router(config)#hostname Yeslab
Yeslab(config)#
Yeslab(config)#exit
Yeslab#
*Mar   1 00:33:34.023: %SYS-5-CONFIG_I: Configured from console by console
Yeslab#disable
Yeslab>quit

Yeslab con0 is now available




Press RETURN to get started.
```

实验 1.2 为路由器配置一个时间

实验目的

熟练掌握为路由器配置一个时间

实验拓扑



实验步骤

先进入路由器的特权模式下
Yeslab#clock set 10:10:10 31 jul 2011
//以上是为路由设置一个时间
Yeslab(config)#clock timezone BeiJing +8
//以上是为路由器配置一个时区
Yeslab#show clock
//以上是查看我们配好的时间

配置实例

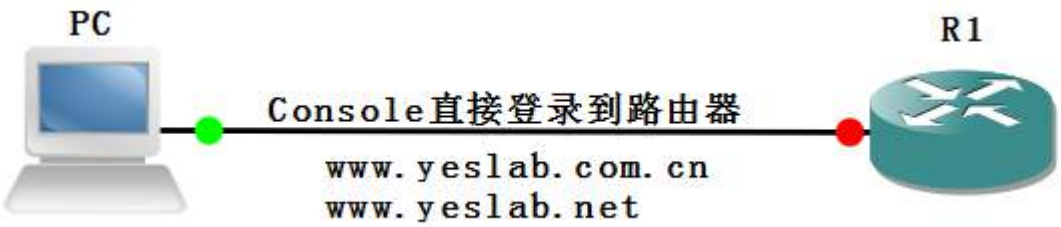
```
Yeslab#clock set 10:10:10 31 jul 2011
Yeslab#
Jul 31 02:10:10.000: %SYS-6-CLOCKUPDATE: System clock has been updated from 18:14:11 BeiJing Sun Jul 31 2011 to 10:10:10 BeiJing Sun Jul 31 2011, configured from console by console.
Yeslab(config)#clock timezone BeiJing +8
Yeslab(config)#
Jul 31 02:10:40.707: %SYS-6-CLOCKUPDATE: System clock has been updated from 10:10:40 BeiJing Sun Jul 31 2011 to 10:10:40 BeiJing Sun Jul 31 2011, configured from console by console.
Yeslab#show clock
10:25:42.135 BeiJing Sun Jul 31 2011
```

实验 1.3 设置路由器超时退出时间

实验目的

熟练掌握为路由器设置超时时间，以及开启光标跟随。

实验拓扑



实验步骤

进入路由器的全局模式下

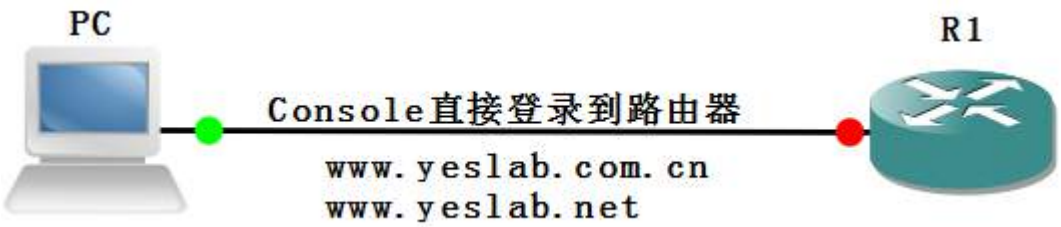
```
Yeslab(config)#line console 0
Yeslab(config-line)#
//以上是进入路由器的控制台线路模式下
Yeslab(config-line)#exec-timeout 30 30
//以上是设置路由器在 30 分钟 30 秒后无操作就自动弹出到用户模式外，如果设置为 0 0 则表示永不超时
Yeslab(config-line)#logging synchronous
//以上为启用光标跟随。所谓光标跟随，是指在我们输入命令的时候，不会被一些日志信息所打断。
```


实验 1.4 为路由器配置密码

实验目的

熟练掌握为路由器配置各种密码

实验拓扑



实验步骤

进入路由器全局模式

```
Yeslab(config)# enable password yeslab
//以上是在全局模式下为进入特权模式设置一个明文密码。
Yeslab(config)# enable secret yeslab
//以上是为特权模式设置一个密文的密码，如果以上二个密码都被设置的时候，只有密文的生效
```

```
Yeslab(config)# line console 0
Yeslab(config-line)# password yeslab
Yeslab(config-line)#login
//以上是进入 console 接口模式配置一个密码，后来的 login 是应用密码
```

```
Yeslab(config-line)#exit
Yeslab(config)# line vty 0 4
Yeslab(config-line)#password yeslab
Yeslab(config-line)#login
//以上是进入 VTY 接口模式下为 TELNET 上的用户设置一个密码
```



```
Yeslab(config-line)#exit
Yeslab(config)# line aux 0
Yeslab(config-line)#password yeslab
Yeslab(config-line)#login
//以上是为辅助接口配置一个密码
```

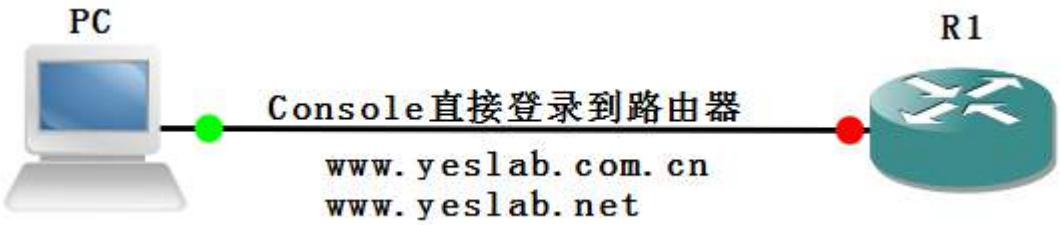
```
Yeslab(config)# service password-encryption
//以上是为一些明文显示的密码进行简单的显示加密
```

实验 1.5 关闭路由器的 DNS 命令解析功能

实验目的

熟练掌握关闭路由器的命令解析功能

实验拓扑



实验步骤

进入全局模式下

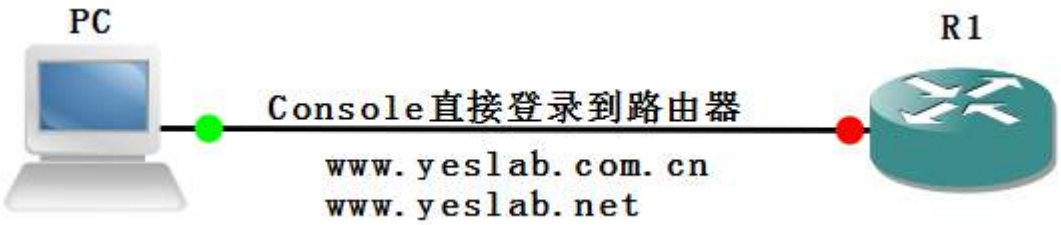
```
Yeslab(config)#no ip domain lookup
```
//以上是关闭路由器命令解析功能，默认情况下，当你输入一条 Cisco IOS 软件无法识别的命令时，路由器会把这个命令向解析服务器发出查询，一般实验性的环境并无解析服务器，因而输入错误的命令而造成的查询非常耗时，因此我们会关闭这一功能。

实验 1.6 为路由器配置描述信息

实验目标

了解如何为路由器设置标语信息和为接口设置描述信息即可

实验拓扑



实验步骤

进入全局模式下

```
Yeslab(config)#banner motd # If you are not YESLAB students, exit landing. Otherwise the consequences #
```
//以上是设置一条警告信息，这条信息将在一登陆上路由器即显示。是以#号键开始，再以#号结束。
```
Yeslab(config)# interface ethernet 0/1
```
//以上是进入接口 E0/1 下
```
Yeslab(config)# description This is the RS lab access YESLAB
```
//以上是为此接口配置一条描述信息。

```
Yeslab #show running-config interface ethernet 0/1
Building configuration...
```

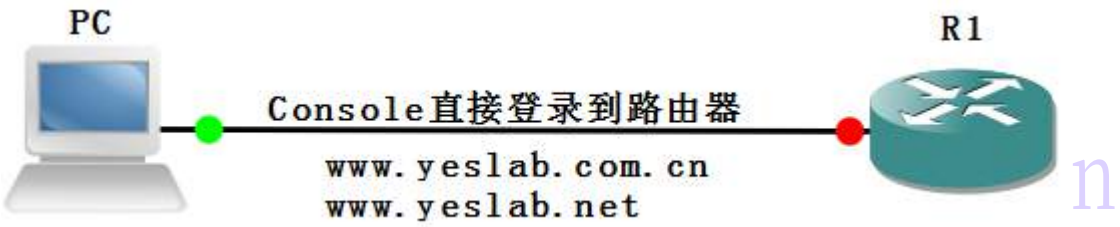
```
Current configuration : 113 bytes
interface Ethernet0/1
 description This is the RS lab access YESLAB
 no ip address
 shutdown
 half-duplex
end
```
//以上是查看我们配上的信息

实验 1.7 配置命令别名

实验目标

了解如何配置命令的别名即可

实验拓扑



实验步骤

进入全局模式下

```
Yeslab(config)# alias exec open show ip route
```
//以上是为查看路由信息的命令（show ip route）配置了一个更简单的别名（open）

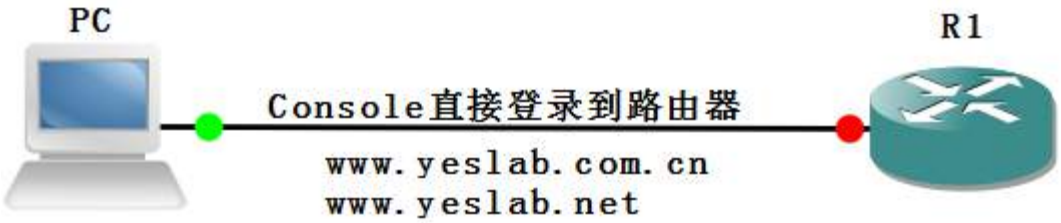
```
Yeslab#open
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is not set
Yeslab#
```
//以上是测试是否生效，可发现已经生效了

实验 1.8 清空路由器配置文件

实验目标

掌握如何清除路由器的配置

实验拓扑



实验步骤

进入特权模式下：

```
Yeslab# erase startup-config
Erasing the nvram filesystem will remove all configuration files! Continue? [confirm]
[OK]
Erase of nvram: complete
Yeslab#
Jul 31 04:08:56.495: %SYS-7-NV_BLOCK_INIT: Initialized the geometry of nvram
Yeslab# reload
```

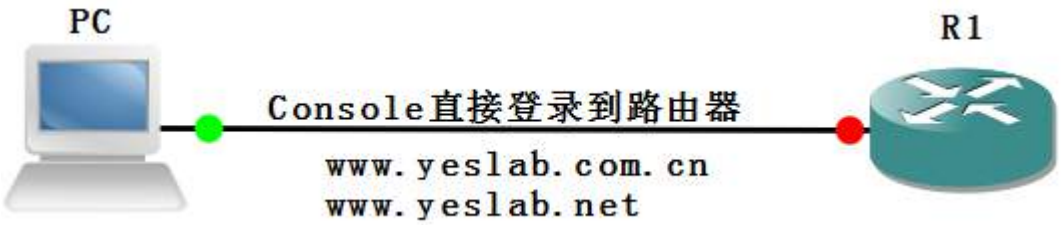
//以上是清空路由器的配置文件，再重启后，路由器就恢复了默认配置了。

实验 1.9 配置一台路由器模拟 PC

实验目标

熟练掌握如果用一台路由去模拟一台 PC，因为这在以后的学习过程中会经常用到

实验拓扑



实验步骤

进入全局模式下

```
Yeslab(config)# no ip routing
//以上是关闭路由器的路由功能，即成为一个 PC,非常简单！
```

```
Yeslab(config)# ip default-gateway 10.1.1.1
//以上是为此 PC 设置一个默认网关
```

```
Yeslab(config)# interface ethernet 0/1
Yeslab(config-if)#ip address 10.1.1.2 255.255.255.0
//以上是为我们模拟的 PC 某一个接口配一个 IP 地址。
```

```
Yeslab#show ip route
Default gateway is 10.1.1.1
//以上是查看模拟 PC 的网关
```

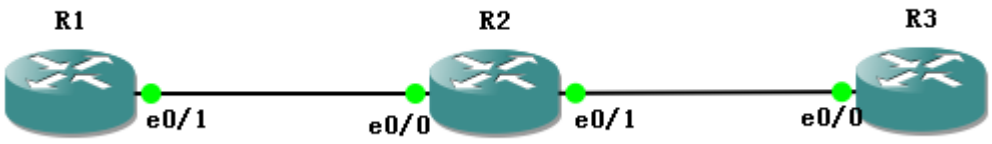

实验 1.10 Cisco 的 CDP 协议

CDP协议是Cisco的邻居发现协议。在默认情况下就是开启的，并不需要做配置，只用把接口UP起来，就可以去查看他。

实验目标

1：理解CDP协议的作用，
2：开启和关闭CDP

实验拓扑



实验步骤

首先修改三个路由器的名字如拓扑所示：R1 R2 R3
然后把拓扑所示的接口都UP起来

在R1上的全局模式下：

```
R1(config)# interface ethernet 0/1
R1(config-if)# no shutdown
```

然后在R2上

```
R2(config)# interface ethernet 0/0
R2(config-if)# no shutdown
R2(config)# interface ethernet 0/1
R2(config-if)# no shutdown
```

然后在R3上：

```
R3(config)# interface ethernet 0/0
R3(config-if)# no shutdown
```

当所有的接口都UP起来的时候，我们就可以去查看CDP协议告诉我们的一些信息了

在R2上的特权模式下：

```
R2# show cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
R3               Eth 0/1           131           R S I      3640      Eth 0/0
R1               Eth 0/0           131           R S I      3640      Eth 0/1
```

//以上是查看CDP协议带给我们的一些邻居信息。

```
R2# show cdp neighbors detail
-------------------------
Device ID: R3
Entry address(es):
Platform: Cisco 3640,  Capabilities: Router Switch IGMP
Interface: Ethernet0/1,  Port ID (outgoing port): Ethernet0/0
Holdtime : 148 sec
Version :
Cisco IOS Software, 3600 Software (C3640-JK9O3S-M), Version 12.4(10a), RELEASE SOFTWARE (fc2)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2006 by Cisco Systems, Inc.
Compiled Wed 11-Oct-06 20:52 by prod_rel_team
advertisement version: 2
VTP Management Domain: ''
```

```
Duplex: half
-------------------------
Device ID: R1
Entry address(es):
Platform: Cisco 3640,    Capabilities: Router Switch IGMP
Interface: Ethernet0/0,    Port ID (outgoing port): Ethernet0/1
Holdtime : 147 sec
Version :
Cisco IOS Software, 3600 Software (C3640-JK9O3S-M), Version 12.4(10a), RELEASE SOFTWARE (fc2)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2006 by Cisco Systems, Inc.
Compiled Wed 11-Oct-06 20:52 by prod_rel_team
advertisement version: 2
VTP Management Domain: ''
Duplex: half
```

//以上是查看一些更详细的信息，可以包含邻居路由器使用的IOS版本。

有的时候我们需要去关闭路由器的CDP协议，命令如下：

```
R2 (config)# no cdp run
R2# show cdp neighbors
% CDP is not enabled
```

//以上是关闭CDP协议后，再查看CDP信息的时候，就提示我们CDP协议没有开启。

如果想再启用的话再打下面的命令：

```
R2 (config)# cdp run
```

有的时候只想关闭一些特定接口的CDP协议，这可以在接口模式下完成：

```
R2 (config)# interface ethernet 0/1
R2 (config -if)#no cdp enable
```

这样就关闭了E0/1接口下的CDP协议，但是并不影响别的接口。想再开启直接去掉NO就可以了。

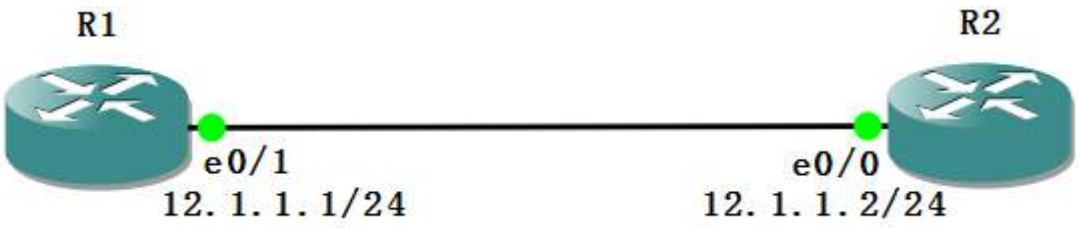
```
R2 (config -if)#cdp enable
```

实验 1.11 用户登陆任务管理

实险目标

1：掌握如何远程登陆一台路由器
2：如何查看和清除连接。
注意，此实验对初学者可能有些难度。

实验拓扑



实验步骤

首先如拓扑所示配置好接口IP地址，确保直连的连通性。

在R1上：

```
R1 (config)# interface ethernet 0/1
R1 (config-if)#ip address 12.1.1.1 255.255.255.0
R1(config-if)#no shutdown
```

在R2上：

```
R2 (config)# interface ethernet 0/0
R2 (config-if)#ip address 12.1.1.2 255.255.255.0
R2(config-if)#no shutdown
```

回到R1上去pingR2的直连接口地址：

```
R1 #ping 12.1.1.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 12.1.1.2, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 32/43/60 ms
R1#
//以上表示直连已经通了
```


注意：然后要按照上面的实验1.4为路由器配置上各种密码，因为如果没有enable密码和vty密码的时候是不能够远程去管理一台路由器的。

回到R2上配置密码：

```
R2 (config)#enable password yeslab
R2(config)#line vty 0 4
R2(config-line)#password yeslab
```

然后我们去R1上telnetR2

```
R1 # telnet 12.1.1.2
Trying 12.1.1.2 ... Open

User Access Verification
```

```
Password:
//以上表示R1已经成功telnet上R2了，这时再输出刚才配的密码yeslab
R2>enable
Password:
R2 #
//现在我们在R1上看到的提示符已经变为R2了。
```

然后我们来到R2上查看连接的情况：

```
R2 # show users
    Line       User       Host(s)              Idle       Location
*   0 con 0                idle                 00:00:00
  130 vty 0                idle                 00:00:11 12.1.1.1

  Interface    User               Mode         Idle     Peer Address
//以上是在R2上查看本路由器的所有连接，我们可以看到从CON口上的本地连接和从vty口上的R1连接
```

回到R1上

```
R1#同时接下<Ctrl+Shift+6>，然后全部放开，再按下x
//以上是挂起R1上对R2的连接。这时能回到R1上来，如果再接二次回车就能再回到R2上
在R1上可以查看我本地发起的所有连接：
R1# show sessions
Conn Host              Address            Byte  Idle Conn Name
*  1 12.1.1.2          12.1.1.2              0     0 12.1.1.2
//以上是显示我连接上了12.1.1.2这个IP地址。
要断开这个连接可以直接登陆后打EXIT通出，也可以有另外二种方式
```


一种是在发起连接的一方，也就是R1打下面命令：

```
R1#disconnect
```

```
还可以在被连接的一方，也就是R2上打下面的命令：
R2#clear line vty 0
[confirm]
 [OK]
//以上命令也可以直接清楚别人对我的连接。
```

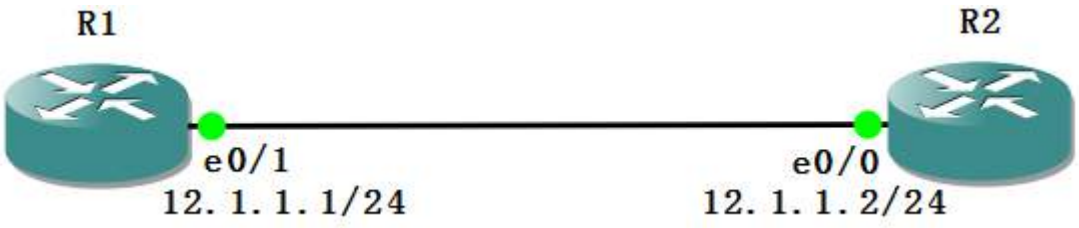
不要关闭模拟器，因为下面的实验可以在此基础上进行。

实验 1.12 创建静态 Hostname 表项

实验目标

掌握静态Hostname表项的作用，

实验拓扑



实验步骤

在R1的全局模式下：

```
R1(config)#ip host yeslab 12.1.1.2
R1(config)#exit
R1 telnet yeslab
Trying yeslab (12.1.1.2)... Open

User Access Verification

Password:
```

//以上表示可以直接telnet我们定义的名字就可以对应上IP地址12.1.1.2了。

实验第二部分 静态路由实验

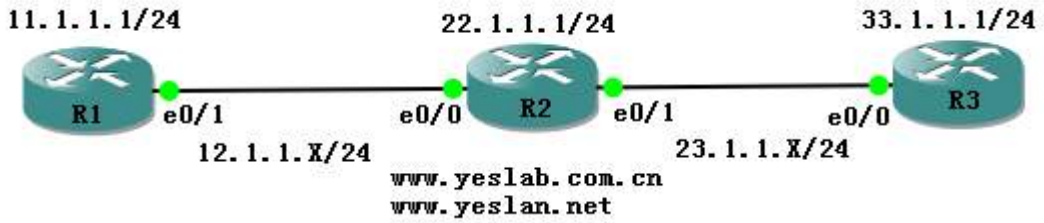
转发数据包和寻址是路由器的最主要功能。路由器转发数据包时需要查找路由表，管理员可以通过手工的方法在路由器中直接配置路由表，这就是静态路由。虽然静态路由不适合于在大的网络中使用，但是由于静态路由简单、路由器负载小、可控性强等原因，在许多场合中还经常被使用。本章将介绍静态路由的配置，同时为以后配置动态路由奠定基础。

实验 2.1 静态路由

实验目标

1：熟练掌握静态路由的原理与配置
2：确保全网连通

实验拓扑



实验步骤

如图所示，先配好各个接口的IP地址，确保直连连通性。

在R1上：

```
R1(config)#interface ethernet 0/1
R1(config-if)#ip add 12.1.1.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#interface loopback 0
R1(config-if)#ip address 11.1.1.1 255.255.255.0
```

然后在R2上：

```
R2(config)#interface ethernet 0/0
R2(config-if)#ip address 12.1.1.2 255.255.255.0
R2(config-if)#no shutdown
R2(config-if)#exit
R2(config)#interface loopback 0
R2(config-if)#ip address 22.1.1.1 255.255.255.0
R2(config-if)#exit
R2(config)#interface ethernet 0/1
R2(config-if)#ip address 23.1.1.2 255.255.255.0
R2(config-if)#no shutdown
```


然后到R3上：

```
R3(config)#interface ethernet 0/0
R3(config-if)#ip address 23.1.1.3 255.255.255.0
R3(config-if)#no shutdown
R3(config-if)#exit
R3(config)#interface loopback 0
R3(config-if)#ip address 33.1.1.1 255.255.255.0
R3(config-if)#exit
```

然后在R2上去测试直接是否通

```
R 2#ping 12.1.1.1
```

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 12.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/46/80 ms
R2#ping 23.1.1.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 23.1.1.3, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 48/61/76 ms
//以上表示直连已经没有问题了。
```

现在我们开始写静态路由让我们R1能ping通R2
在R1上：

```
R1(config)#ip route 22.1.1.0 255.255.255.0 12.1.1.2
R1#ping 22.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 22.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/47/76 ms
//以上表示我们已经ping通R2的环回口了
```


然后我们开始写去往R3的环回口路由

```
R1(config)#ip route 33.1.1.0 255.255.255.0 12.1.1.2
//以上的路由只是能把包路由到R2上，但是R2上并没有去往R3的路由，所以R2上还要写：
R2(config)#ip route 33.1.1.0 255.255.255.0 23.1.1.3
//这时数据包就能正常到达R3了，但是R3上并没有回到R1的路由，所以R3上也要加上：
R3(config)#ip route 12.1.1.0 255.255.255.0 23.1.1.2
```

然后我们再到R1上去pingR3的环回口

```
R1#ping 33.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 33.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 80/108/144 ms
//以上表示R1顺利ping通R3了。
```

按照以上的思路，自己试试让R2，R3分别能ping通别的路由器的环回口，如果不行，再看下面的配置：
先让R2能ping通R1,R3的环回口：

```
R2(config)#ip route 11.1.1.0 255.255.255.0 12.1.1.1
//这样R2就能ping通R1了，因为刚才已经配了通往R3的路由，所以R2上不用再配了。
```

在R3上：

```
R3(config)#ip route 22.1.1.0 255.255.255.0 23.1.1.2
R3(config)#ip route 11.1.1.0 255.255.255.0 23.1.1.2
//这里还要加上R1去往R2与R3之间网段的路由。
```

```
R1(config)#ip route 23.1.1.0 255.255.255.0 12.1.1.2
//做完以上的配置以后，全网都可以相互通信了。
```

在排错的时候，我们可以去查看本地的路由表，如在R1上

```
R1#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     33.0.0.0/24 is subnetted, 1 subnets
S       33.1.1.0 [1/0] via 12.1.1.2
     23.0.0.0/24 is subnetted, 1 subnets
S       23.1.1.0 [1/0] via 12.1.1.2
     22.0.0.0/24 is subnetted, 1 subnets
S       22.1.1.0 [1/0] via 12.1.1.2
     11.0.0.0/24 is subnetted, 1 subnets
C       11.1.1.0 is directly connected, Loopback0
     12.0.0.0/24 is subnetted, 1 subnets
C       12.1.1.0 is directly connected, Ethernet0/1
//S表示的就是我们刚才配置的静态路由了。
```

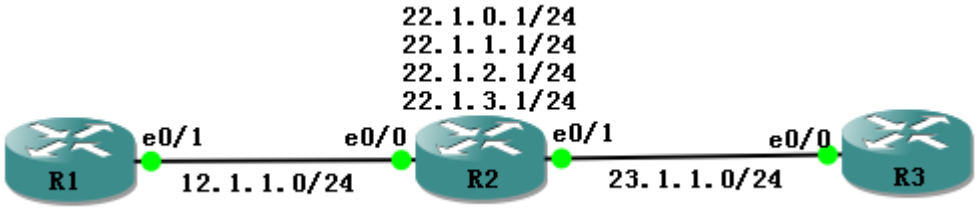

请不要关闭模拟器，因为接下来的实验是在此基础之上的。

实验 2.2 静态汇总路由

实验目标

掌握如何做静态的汇总路由

实验拓扑



实验步骤

在上一个实验的基础上，再在 R2 上起二个环回口。

```
R2(config)#interface loopback 1
R2(config-if)#ip add 22.1.0.1 255.255.255.0
R2(config-if)#exit
R2(config)#interface loopback 2
R2(config-if)#ip address 22.1.2.1 255.255.255.0
R2(config-if)#exit
R2(config)#interface loopback 3
R2(config-if)#ip address 22.1.3.1 255.255.255.0
```


查看一下 R2 的路由表是否正常。

```
R2#show ip route
————此部分省略————
      23.0.0.0/24 is subnetted, 1 subnets
C        23.1.1.0 is directly connected, Ethernet0/1
      22.0.0.0/24 is subnetted, 4 subnets
C        22.1.3.0 is directly connected, Loopback3
C        22.1.2.0 is directly connected, Loopback1
C        22.1.1.0 is directly connected, Loopback0
C        22.1.0.0 is directly connected, Loopback2
      12.0.0.0/24 is subnetted, 1 subnets
C        12.1.1.0 is directly connected, Ethernet0/0
```

//以上是 R2 上的路由表，这时 R1 和 R3 用上面的方法去往 R4 的还回口路由要写四条才可以，这时就可以用汇总的方式来写默认路由了。

在 R1 上：

```
R1(config)#ip route 22.1.0.0 255.255.252.0 12.1.1.2
```

```
//用以上一条路由代汇总了 R2 上的四条路由，这就是静态汇总路由
```

在 R3 上：

```
R3(config)#ip route 22.1.0.0 255.255.252.0 23.1.1.2
//然后我们可以在 R1 上和 R3 上分别去 pingR2 的四个环回口地址，以测试是否能通。在这不再做演示。
```

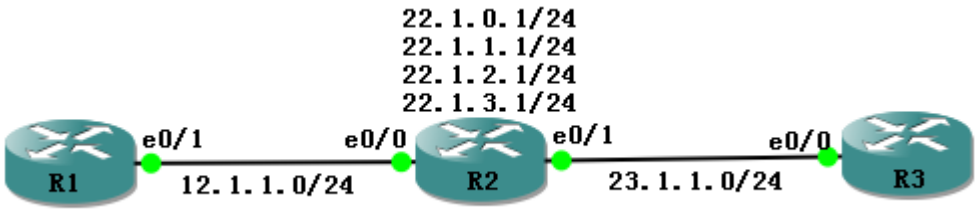
先不要半闭模拟器，因为下面的实验可在此基础上完成。

实验 2.3 静态默认路由

实验目标

掌握静态默认路由的配置

实验拓扑



实验步骤

先去掉我们刚才做好的汇总静态路由，然后我们用一条默认路由来代替它。静态默认路由是由 0.0.0.0 0.0.0.0 来表示的，它代表了所有路由。

在 R1 上，先去掉刚才配置的汇总静态：

```
R1(config)#no ip route 22.1.0.0 255.255.252.0 12.1.1.2
```

在 R3 上也去掉

```
R3(config)#no ip route 22.1.0.0 255.255.252.0 23.1.1.2
```

然后到 R1 和 R3 上分别写二条默认路由

```
R1(config)# ip route 0.0.0.0 0.0.0.0 12.1.1.2
R3(config)# ip route 0.0.0.0 0.0.0.0 23.1.1.2
```

在 R1 上去查看这条默认路由，带*号的就是默认路由了，并且以后我们会学到并不是以 8 个 0 表示的默认路由，我们只要看到带*号的那就是默认路由。

```
R1 #show ip route
————此部分省略————
      11.0.0.0/24 is subnetted, 1 subnets
C        11.1.1.0 is directly connected, Loopback0
```

```
     12.0.0.0/24 is subnetted, 1 subnets
C       12.1.1.0 is directly connected, Ethernet0/1
S*   0.0.0.0/0 [1/0] via 12.1.1.2
```

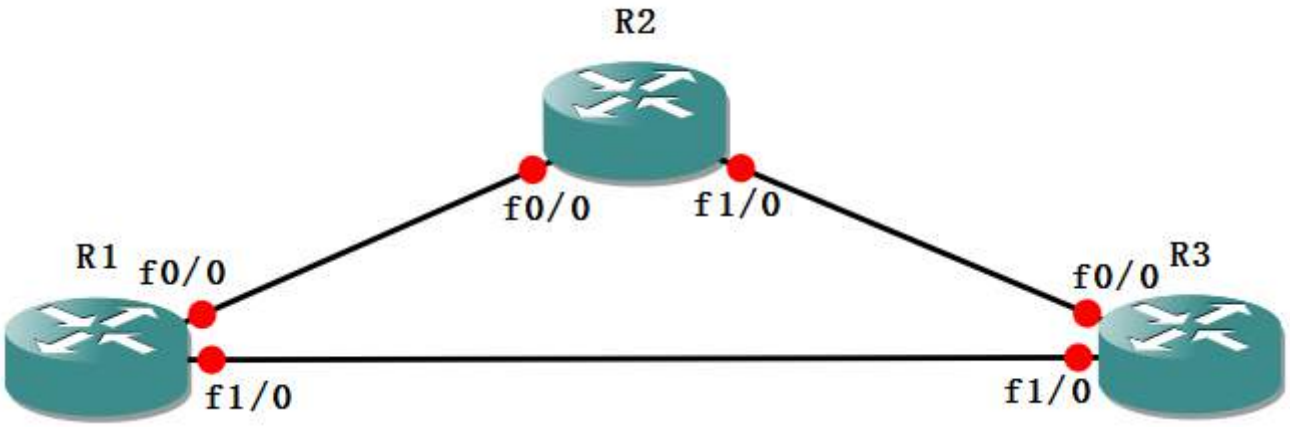
这时我们就可以去 pingR2 的所有环回口地址了。这里不再演试。

实验 2.4 选择静态路由

实验目的

根据路由表转发的最长匹配原则，理解选择静态的实现方法。

实验拓扑



实验步骤

1：先配置好各个接口的 IP 地址，保证直连能通。

在 R1 上：

```
R1(config)#interface fastEthernet 0/0
R1(config-if)#ip address 12.1.1.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#interface fastEthernet 1/0
R1(config-if)#ip address 13.1.1.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#exit
```

在 R2 上

```
R2(config)#interface fastEthernet 0/0
R2(config-if)#ip address 12.1.1.2 255.255.255.0
```

```
R2(config-if)#no shutdown
R2(config-if)#exit
R2(config)#interface fastEthernet 1/0
R2(config-if)#ip address 23.1.1.2 255.255.255.0
R2(config-if)#no shutdown
```

在 R3 上：

```
R3(config)#interface fastEthernet 0/0
R3(config-if)#ip address 23.1.1.3 255.255.255.0
R3(config-if)#no shutdown
R3(config-if)#exit
R3(config)#interface fastEthernet 1/0
R3(config-if)#ip address 13.1.1.3 255.255.255.0
R3(config-if)#no shutdown
```

在 R3 上再起一个环回口，并配置多个 IP 地址：

```
R3(config)#interface loopback 0
R3(config-if)#ip address 33.1.1.1 255.255.255.0
R3(config-if)#ip address 33.1.1.2 255.255.255.0 secondary
R3(config-if)#ip address 33.1.1.3 255.255.255.0 secondary
R3(config-if)#ip address 33.1.1.4 255.255.255.0 secondary
```

//加 secondary 可以使一个接口配置多个辅助地址。

2:在 R1,R2,R3 上做配置，让 R1 能通过 R2 去 ping 通 R3 上的环回口地址。

在 R1 上：

```
R1(config)#ip route 33.1.1.0 255.255.255.0 12.1.1.2
```

在 R2 上：

```
R2(config)#ip route 33.1.1.0 255.255.255.0 23.1.1.3
```

在 R3 上配置回来的路由：

```
R3(config)#ip route 12.1.1.0 255.255.255.0 23.1.1.2
```

在 R1 上去测试是否通：

```
R1#traceroute 33.1.1.1
Type escape sequence to abort.
Tracing the route to 33.1.1.1
  1 12.1.1.2 36 msec 72 msec 32 msec
  2 23.1.1.3 88 msec *    80 msec
```

//以上表示已经通了，并且是经过 R2。

这里我们想针对 33.1.1.1 这个 IP 地址让他走下面这条路，但是去访问 33.1.1.2，33.1.1.3，33.1.1.4 等地址依然走上面。要实现这个功能，就要用到我们的选择路由：

在 R1 上：

```
R1(config)#ip route 33.1.1.1 255.255.255.255 13.1.1.3
```

我们去查看一下 R1 的路由表：

```
R1#show ip route
————此部分省略————
      33.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
S        33.1.1.1/32 [1/0] via 13.1.1.3
S        33.1.1.0/24 [1/0] via 12.1.1.2
      12.0.0.0/24 is subnetted, 1 subnets
C        12.1.1.0 is directly connected, FastEthernet0/0
      13.0.0.0/24 is subnetted, 1 subnets
C        13.1.1.0 is directly connected, FastEthernet1/0
```

//可以发现针对 33.1.1.1 这个地址我们有一条 32 位的主机路由，这时我们再访问这个地址的时候就会走这一条而不会再选择刚才的 33.1.1.0/24 这条路由，原因是因为最长匹配的原则。

测试：

```
R1#traceroute 33.1.1.1
Type escape sequence to abort.
Tracing the route to 33.1.1.1
  1 13.1.1.3 68 msec *  44 msec
R1#traceroute 33.1.1.2
Type escape sequence to abort.
Tracing the route to 33.1.1.2
  1 12.1.1.2 84 msec 68 msec 40 msec
  2 23.1.1.3 40 msec *  80 msec
```

//通过最长匹配的原则，实现了处于同一网段的 IP 包选择不同的出口。

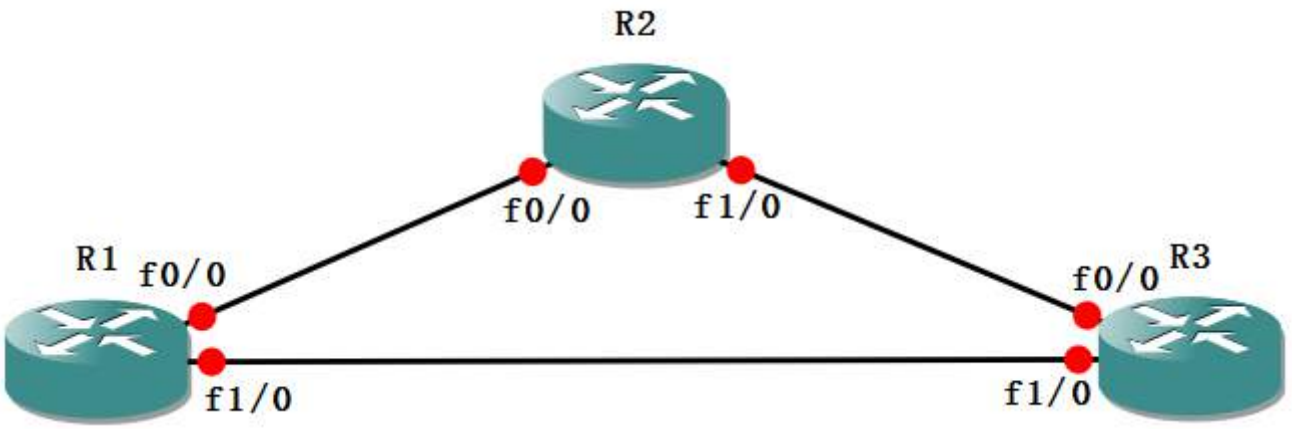
请不要删除配置，因为接下来的实验可在此基础上实现。

实验 2.5 浮动静态路由

实验目的

根据路由管理距离越小越优先的原则，实验路由自动的浮动选择。

实验拓扑



实验步骤

接着上面的 2.4 的配置向下做，先查看一下 R1 的路由表：

```
R1#show ip route
————此部分省略————
      33.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
S        33.1.1.1/32 [1/0] via 13.1.1.3
S        33.1.1.0/24 [1/0] via 12.1.1.2
      12.0.0.0/24 is subnetted, 1 subnets
C        12.1.1.0 is directly connected, FastEthernet0/0
      13.0.0.0/24 is subnetted, 1 subnets
C        13.1.1.0 is directly connected, FastEthernet1/0
```

可以看到 33.1.1.0/24 这条路由是指向 R2 的。

在 R1 上做以下配置：

```
R1(config)#ip route 33.1.1.0 255.255.255.0 13.1.1.3 10
```

//注意后面的 10，指的是为这条静态路由设定一个管理距离是 10.我们静态路由是默认情况下是 1

再查看路由表：

```
R1#show ip route
————此部分省略————
      33.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
S        33.1.1.1/32 [1/0] via 13.1.1.3
S        33.1.1.0/24 [1/0] via 12.1.1.2
      12.0.0.0/24 is subnetted, 1 subnets
C        12.1.1.0 is directly connected, FastEthernet0/0
      13.0.0.0/24 is subnetted, 1 subnets
C        13.1.1.0 is directly connected, FastEthernet1/0
```


//发现路由表中 33.1.1.0/24 并没有发现改变，这时我们再把路由表指的 R2 方向的接口 shutdown 后再查看。

```
R1(config)#interface fastEthernet 0/0
R1(config-if)#shutdown
R1#show ip route
————此部分省略————
      33.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
S        33.1.1.1/32 [1/0] via 13.1.1.3
S        33.1.1.0/24 [10/0] via 13.1.1.3
      13.0.0.0/24 is subnetted, 1 subnets
C        13.1.1.0 is directly connected, FastEthernet1/0
```

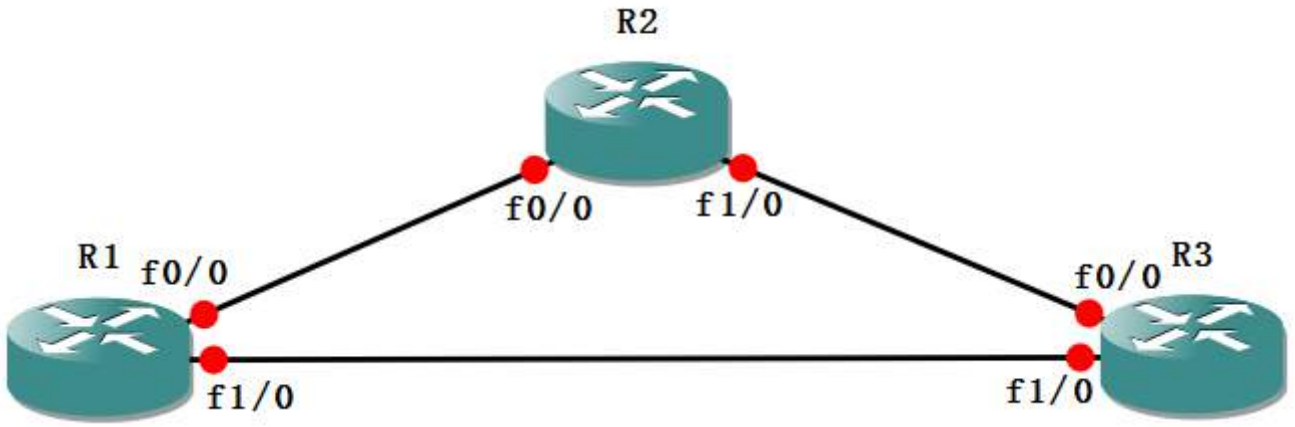
//我们发现路由表中 33.1.1.0/24 这条路由自动切换到下一跳为 R3 的 13.1.1.3,并且管理距离为 10.

实验 2.6 用静态路由实现负载均衡

实验目的

理解用静态路由来实现负载均衡的方法

实验拓扑



实验步骤

实验要求。在 R2 上去访问 R1 与 R3 直接的网段时。实现负载均衡。

步骤 1：先配置好 R1,R2,R3 的 IP 地址，保证直连通。

在R1上：

```
R1(config)#interface fastEthernet 0/0
R1(config-if)#ip address 12.1.1.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#interface fastEthernet 1/0
R1(config-if)#ip address 13.1.1.1 255.255.255.0
R1(config-if)#no shutdown
```

在R2上：

```
R2(config)#interface fastEthernet 0/0
R2(config-if)#ip address 12.1.1.2 255.255.255.0
R2(config-if)#no shutdown
R2(config-if)#exit
R2(config)#interface fastEthernet 1/0
R2(config-if)#ip address 23.1.1.2 255.255.255.0
R2(config-if)#no shutdown
```

在R3上：

```
R3(config)#interface fastEthernet 0/0
R3(config-if)#ip address 23.1.1.3 255.255.255.0
```

```
R3(config-if)#no shutdown
R3(config-if)#exit
R3(config)#interface fastEthernet 1/0
R3(config-if)#ip address 13.1.1.3 255.255.255.0
R3(config-if)#no shutdown
```

去测试连通性：

在R1上：

```
R1#ping 12.1.1.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 12.1.1.2, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 4/33/96 ms
R1#ping 13.1.1.3
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 13.1.1.3, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 8/37/68 ms
```

在R2上：


```
R2#ping 23.1.1.3
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 23.1.1.3, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 4/33/64 ms
```

此时R2并不能PING通R1与R3之间的13.1.1.0/24网段，因为路由表里默认只有直连的路由。但是R2去往13.1.1.0/24网段可以通过R1和R3可达，所以我们可以实现负载均衡。

在R2上：

```
R2(config)#ip route 13.1.1.0 255.255.255.0 12.1.1.1
R2(config)#ip route 13.1.1.0 255.255.255.0 23.1.1.3
//所谓负载均衡就是去往同一个目的地路由有不同的下一跳。
```

查看路由表：

```
R2#show ip route static
     13.0.0.0/24 is subnetted, 1 subnets
S       13.1.1.0 [1/0] via 23.1.1.3
                 [1/0] via 12.1.1.1
//以上表示去往13.1.1.0/24的路由有二个下一跳，这种情况称之为负载均衡
```

在测试之前我们要先保证R1可达23.1.1.0/24网段。

在R1上：

```
R1(config)#ip route 23.1.1.0 255.255.255.0 13.1.1.3
```

我们去R2上去PING R1的13.1.1.1地址观察负载均衡，
在R2上：

```
R2#debug ip icmp
//以上是打开ICMP的提示信息也便我们观察
R2(config)#no ip cef
//关闭CISCO的CEF转发，以方便我们观察
```

```
R2#ping 13.1.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 13.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/42/92 ms
R2#
*Mar  1 00:20:00.779: ICMP: echo reply rcvd, src 13.1.1.1, dst 12.1.1.2
*Mar  1 00:20:00.831: ICMP: echo reply rcvd, src 13.1.1.1, dst 23.1.1.2
*Mar  1 00:20:00.843: ICMP: echo reply rcvd, src 13.1.1.1, dst 12.1.1.2
*Mar  1 00:20:00.895: ICMP: echo reply rcvd, src 13.1.1.1, dst 23.1.1.2
*Mar  1 00:20:00.903: ICMP: echo reply rcvd, src 13.1.1.1, dst 12.1.1.2
```

//以上表示我们去往13.1.1.1这个目标的时候，源IP在不停的切换，分别从F0/0口和F1/0口出去，实现了负载均衡。

实验第三部分：RIP 路由协议

动态路由协议包括距离向量路由协议和链路状态路由协议。RIP（Routing InformationProtocols，路由信息协议）是使用最广泛的距离向量路由协议。RIP 是为小型网络环境设计的，因为这类协议的路由学习及路由更新将产生较大的流量，占用过多的带宽。

RIP 概述

RIP 是由Xerox 在70 年代开发的，最初定义在RFC1058 中。RIP 用两种数据包传输更新：更新和请求，每个有RIP 功能的路由器默认情况下每隔30 秒利用UDP 520 端口向与它直连的网络邻居广播（RIP v1）或组播（RIP v2）路由更新。因此路由器不知道网络的全局情况，如果路由更新在网络上传播慢，将会导致网络收敛较慢，造成路由环路。为了避免路由环路，RIP 采用水平分割、毒性逆转、定义最大跳数、闪式更新、抑制计时5 个机制来避免路由环路。

RIP 协议分为版本1 和版本2。不论是版本1 或版本2，都具备下面的特征：

1. 是距离向量路由协议；
2. 使用跳数（Hop Count）作为度量值；
3. 默认路由更新周期为30 秒；

4. 管理距离（AD）为120；
5. 支持触发更新；
6. 最大跳数为15 跳；
7. 支持等价路径,默认4 条,最大6 条；
8. 使用UDP520 端口进行路由更新。

而RIPv1 和RIPv2 的区别如表

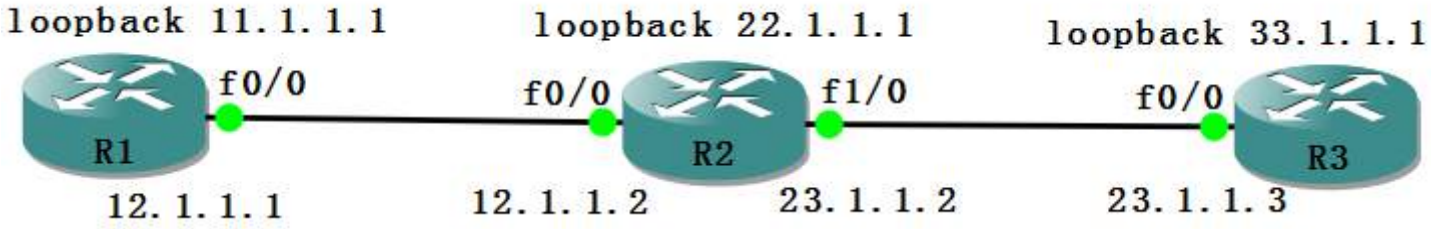
RIPv1	RIPv2
在路由更新的过程中不携带子网信息	在路由更新的过程中携带子网信息
不提供认证	提供明文和MD5认证
不支持VLSM和CIDR	支持VLSM和CIDR
采用广播更新	采用组播（224.0.0.9）更新
有类别（Classful）路由协议	无类别（Classless）路由协议

实验 3.1：RIP 版本 1

实验目的

熟练理解并掌握 RIP 版本 V1 的配置

实验拓扑



实验步骤

首先保证直连的连通性

在 R1 上：

```
R1(config)#interface fastEthernet 0/0
R1(config-if)#ip address 12.1.1.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#interface loopback 0
R1(config-if)#ip address 11.1.1.1 255.255.255.0
```

在 R2 上:

```
R2(config)#interface fastEthernet 1/0
R2(config-if)#ip address 23.1.1.2 255.255.255.0
R2(config-if)#no shutdown
R2(config-if)#exit
R2(config)#interface loopback 0
R2(config-if)#ip address 22.1.1.1 255.255.255.0
```

在 R3 上：

```
R3(config)#interface fastEthernet 0/0
R3(config-if)#ip address 23.1.1.3 255.255.255.0
R3(config-if)#no shutdown
R3(config-if)#exit
R3(config)#interface loopback 0
R3(config-if)#ip address 33.1.1.1 255.255.255.0
```

IP 地址配好以后开始在 R1 上配置 RIP

在 R1 上：

```
R1(config)#router rip
//以上是第一步起 RIP 的进程，
R1(config-router)#version 1
//以上是让 RIP 运行版本 1。
R1(config-router)#network 11.0.0.0
//以上是第二步宣告 loopback 0 进入 RIP 进程，在 RIP 里都是用主类宣告的方法。
R1(config-router)#network 12.0.0.0
//以上是宣告 fastEthernet 0/0 进入 RIP 进程。
```


在 R2 上：

```
R2(config)#router rip
R2(config-router)#version 1
R2(config-router)#network 12.0.0.0
R2(config-router)#network 23.0.0.0
R2(config-router)#network 22.0.0.0
```

在 R3 上：

```
R3(config)#router rip
R3(config-router)#version 1
R3(config-router)#network 23.0.0.0
R3(config-router)#network 33.0.0.0
```

宣告完以后，我们去但看路由表：

```
R1#show ip route
————此部分省略————
R       33.0.0.0/8 [120/2] via 12.1.1.2, 00:00:01, FastEthernet0/0
```

```
R       23.0.0.0/8 [120/1] via 12.1.1.2, 00:00:21, FastEthernet0/0
R       22.0.0.0/8 [120/1] via 12.1.1.2, 00:00:21, FastEthernet0/0
        11.0.0.0/24 is subnetted, 1 subnets
C           11.1.1.0 is directly connected, Loopback0
        12.0.0.0/24 is subnetted, 1 subnets
C           12.1.1.0 is directly connected, FastEthernet0/0
```

//以上表明我们已经正常的学习到了所有的路由条目，由于我们是运行的 RIP 版本 1，所以收到的都是主类的路由。由于版本 1 根本不会用到，所以相关的实验会比较少。

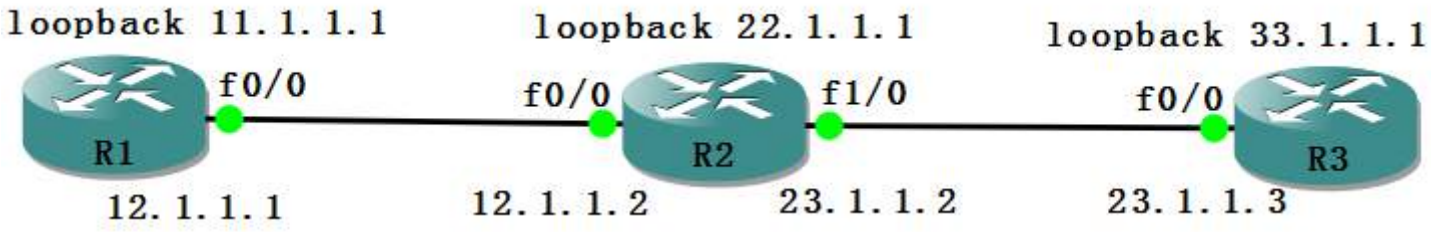
不要删除配置，因为接下来的实验是在此基础之上的。

实验 3.2：RIP 版本 2

实验目的

理解并熟练掌握 RIP 版本 2 的配置

实验拓扑



实验步骤

此实验是在 3.1 的基础之上做的，请保证直连已经连通，并且 RIP 版本 1 已经运行，因为 RIP 的版本 1 与版本 2 的配置只有一个命令不一样，单独拿出来告诉大家就可以了。
所以此实验请参考实验 3.1

在 R1 上：

```
R1(config)#router rip
R1(config-router)#version 2
```

//以上命令让 RIP 运行版本 2，如果不打的话默认情况是发送版本 1 的更新，但能接收版本 1 与版本 2

```
R1(config-router)#no auto-summary
```

//以上命令是关闭 RIP 的自动汇总功能，RIP 协议默认是自汇总的，如果不关闭的话，RIP 会在主类边界自动汇总到主类。

在 R2 上：

```
R2(config)#router rip
R2(config-router)#version 2
R2(config-router)#no auto-summary
```

在 R3 上：

```
R3(config)#router rip
R3(config-router)#version 2
R3(config-router)#no auto-summary
```

再去 R1 上查看路由表：

```
R1#show ip route
————此部分省略————
R       33.1.1.0 [120/2] via 12.1.1.2, 00:00:01, FastEthernet0/0
     23.0.0.0/24 is subnetted, 1 subnets
R       23.1.1.0 [120/1] via 12.1.1.2, 00:00:01, FastEthernet0/0
     22.0.0.0/24 is subnetted, 1 subnets
R       22.1.1.0 [120/1] via 12.1.1.2, 00:00:01, FastEthernet0/0
     11.0.0.0/24 is subnetted, 1 subnets
C       11.1.1.0 is directly connected, Loopback0
     12.0.0.0/24 is subnetted, 1 subnets
C       12.1.1.0 is directly connected, FastEthernet0/0
```

//以上表明已经正常学习到所有路由，并且都是明细的了。注意，在做此实验的时候，由于 RIP 收敛比较慢，可能我们开始会看到/8 位的路由，这时我们可以用命令：clear ip route * 去手动清空所有路由器的路由表，加快收敛。

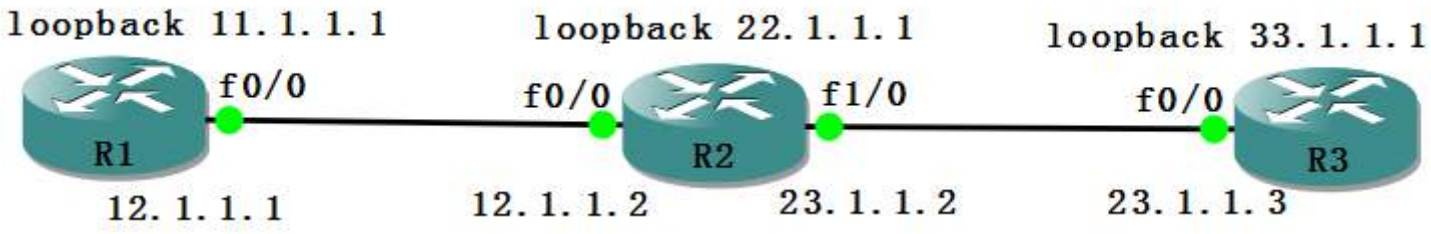
请不要删除配置，因为接下来的实验是在此基础之上的。

实验 3.3：RIP 版本 1 与版本 2 兼容

实验目的

熟练掌握 RIP 的版本 1 与版本 2 的共存问题

实验拓扑



实验步骤

在 3.2 的实验基础上，把 R3 重新配置成 RIP 版本 1

在 R3 上：

```
R3(config)#router rip
R3(config-router)#version 1
```

改完以后再去查看路由表：

```
R3#show ip route
————此部分省略————
      33.0.0.0/24 is subnetted, 1 subnets
C        33.1.1.0 is directly connected, Loopback0
      23.0.0.0/24 is subnetted, 1 subnets
C        23.1.1.0 is directly connected, FastEthernet0/0
```

//在 R3 上，我们发现已经没有学到任何 RIP 的路由了。

再到 R2 上去查看：

```
R2#show ip route
————此部分省略————
      23.0.0.0/24 is subnetted, 1 subnets
C        23.1.1.0 is directly connected, FastEthernet1/0
      22.0.0.0/24 is subnetted, 1 subnets
C        22.1.1.0 is directly connected, Loopback0
      11.0.0.0/24 is subnetted, 1 subnets
R        11.1.1.0 [120/1] via 12.1.1.1, 00:00:06, FastEthernet0/0
      12.0.0.0/24 is subnetted, 1 subnets
C        12.1.1.0 is directly connected, FastEthernet0/0
```

//以上发现 R2 上也看不到了 R3 的路由。

在 R2 上做以下配置：

```
R2(config)#interface fastEthernet 1/0
R2(config-if)#ip rip send version 1
R2(config-if)#ip rip receive version 1
```

//以上命令表示在 R2 上连接 R3 的接口上，我们不再发送和接收 RIP 版本 2 的更新，而是去发送和接收版本 1 的更新。以实现了 R2 的版本 2 与 R3 的版本 1 的兼容。

再到 R2 和 R3 上去查看路由表：

```
R2#show ip route
————此部分省略————
R       33.0.0.0/8 [120/1] via 23.1.1.3, 00:00:00, FastEthernet1/0
        23.0.0.0/24 is subnetted, 1 subnets
C          23.1.1.0 is directly connected, FastEthernet1/0
        22.0.0.0/24 is subnetted, 1 subnets
C          22.1.1.0 is directly connected, Loopback0
        11.0.0.0/24 is subnetted, 1 subnets
R          11.1.1.0 [120/1] via 12.1.1.1, 00:00:22, FastEthernet0/0
        12.0.0.0/24 is subnetted, 1 subnets
C          12.1.1.0 is directly connected, FastEthernet0/0
```

//以上表明 R2 又学到了 R3 的路由，并且是汇总的。

在 R3 上：

```
R3#show ip route
————此部分省略————
        33.0.0.0/24 is subnetted, 1 subnets
C          33.1.1.0 is directly connected, Loopback0
        23.0.0.0/24 is subnetted, 1 subnets
C          23.1.1.0 is directly connected, FastEthernet0/0
R       22.0.0.0/8 [120/1] via 23.1.1.2, 00:00:01, FastEthernet0/0
R       11.0.0.0/8 [120/2] via 23.1.1.2, 00:00:01, FastEthernet0/0
R       12.0.0.0/8 [120/1] via 23.1.1.2, 00:00:01, FastEthernet0/0
```


//以上表明 R3 也学到的所有的路由，都是主类的。

由此实现了不同版本之间相互学习路由。

--想一想：如果我们在 R3 上配置成收发版本 2 的路由，能不能实现与 R2 的兼容。

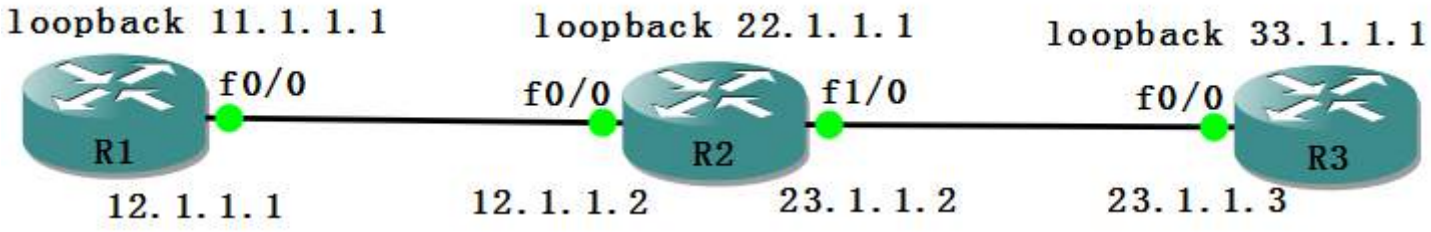
可恢复我们在本实验做的改变，把 R3 再成 RIP 版本 2，接着做下面的实验。

实验 3.4：在接口禁止运行 RIP

实验目的

理解并掌握 passive-interface 命令在 RIP 中的作用。

实验拓扑



实验步骤

根据拓朴图保证直连的连通性。
根据实验 3.2，配置好 3 个路由器。让他们能相互学习到 RIP 版本 2 的路由。如下：

在 R1 上：

```
R1(config)#router rip
R1(config-router)#version 2
R1(config-router)#no auto-summary
```

```
R1(config-router)#network 12.0.0.0
R1(config-router)#network 11.0.0.0
```

在 R2 上：

```
R2(config)#router rip
R2(config-router)#version 2
R2(config-router)#no auto-summary
R2(config-router)#network 12.0.0.0
R2(config-router)#network 23.0.0.0
R2(config-router)#network 22.0.0.0
```


在 R3 上：

```
R3(config)#router rip
R3(config-router)#version 2
R3(config-router)#no auto-summary
R3(config-router)#network 23.0.0.0
R3(config-router)#network 33.0.0.0
```

完成以去去查看路由表：

在 R1 上：

```
R1#show ip route
————本部分省略————
      33.0.0.0/24 is subnetted, 1 subnets
R        33.1.1.0 [120/2] via 12.1.1.2, 00:00:05, FastEthernet0/0
      23.0.0.0/24 is subnetted, 1 subnets
R        23.1.1.0 [120/1] via 12.1.1.2, 00:00:05, FastEthernet0/0
      22.0.0.0/24 is subnetted, 1 subnets
```

```
R        22.1.1.0 [120/1] via 12.1.1.2, 00:00:05, FastEthernet0/0
     11.0.0.0/24 is subnetted, 1 subnets
C        11.1.1.0 is directly connected, Loopback0
     12.0.0.0/24 is subnetted, 1 subnets
C        12.1.1.0 is directly connected, FastEthernet0/0
```

//以上表明已经通过 RIP 正常的学习到所有路由信息。

我们让 R1 连接 R2 的 F0/0 口禁止运行 RIP

```
R1(config)#router rip
R1(config-router)#passive-interface fastEthernet 0/0
```

//以上命令就是在接口 F0/0 上，RIP 不再发送任何 RIP 路由更新给 R2，但是可以接收 R2 发过来的路由更新。

再到 R2 上去看现像：

```
R2#clear ip route *
```

//以上是清空一下路由表，加快 RIP 的收敛

```
R2#show ip route
————此部分省略————
     33.0.0.0/24 is subnetted, 1 subnets
R        33.1.1.0 [120/1] via 23.1.1.3, 00:00:02, FastEthernet1/0
     23.0.0.0/24 is subnetted, 1 subnets
C        23.1.1.0 is directly connected, FastEthernet1/0
     22.0.0.0/24 is subnetted, 1 subnets
C        22.1.1.0 is directly connected, Loopback0
     12.0.0.0/24 is subnetted, 1 subnets
C        12.1.1.0 is directly connected, FastEthernet0/0
```

//以上表明 R2 上已经看不到 R1 的环回口路由了。

我们再回到 R1 上看路由表：

```
R1#clear ip route *
R1#show ip route
————此部分省略————
     33.0.0.0/24 is subnetted, 1 subnets
R        33.1.1.0 [120/2] via 12.1.1.2, 00:00:00, FastEthernet0/0
     23.0.0.0/24 is subnetted, 1 subnets
R        23.1.1.0 [120/1] via 12.1.1.2, 00:00:00, FastEthernet0/0
     22.0.0.0/24 is subnetted, 1 subnets
R        22.1.1.0 [120/1] via 12.1.1.2, 00:00:00, FastEthernet0/0
     11.0.0.0/24 is subnetted, 1 subnets
C        11.1.1.0 is directly connected, Loopback0
     12.0.0.0/24 is subnetted, 1 subnets
C        12.1.1.0 is directly connected, FastEthernet0/0
```

//以上表明 R1 能够正常收到 RIP 的路由更新。以上表明在 RIP 协议中，我们用 passive-interface 在一个接

口禁用掉 RIP 后，只是不发送 RIP 的更新，但是能正常学习到 RIP 的更新，这与 EIGRP，OSPF 有本质上的区别，EIGRP 和 OSPF 都是不发也不会收各自的报文。

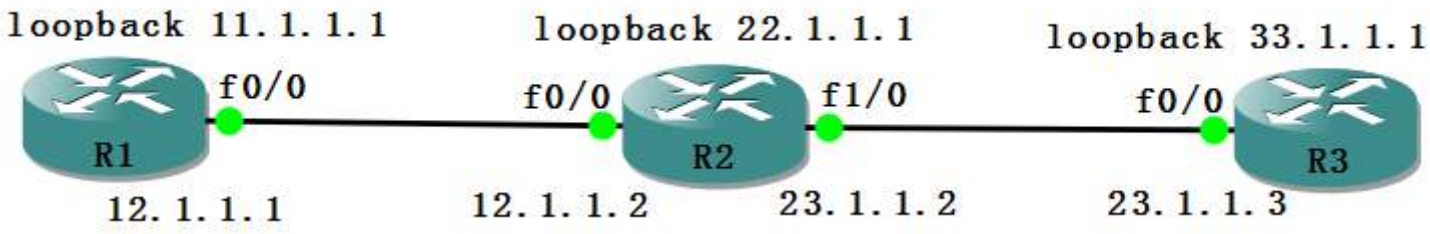
不要删除此配置，因为接下来的配置是在此基础之上的。

实验 3.5：RIPV2 单播更新

实验目的

掌握如何实验 RIP 的单播更新。

实验拓扑



实验步骤

此实验紧接着实验 3.4。

我们去 R2 上 passive 掉连接 R1 的 F0/0 口。

```
R2(config)#router rip
R2(config-router)#passive-interface fastEthernet 0/0
```

在 R1 上去看：

```
R1#clear ip route *
R1#show ip route
————————此部分省略————————
      11.0.0.0/24 is subnetted, 1 subnets
C        11.1.1.0 is directly connected, Loopback0
      12.0.0.0/24 is subnetted, 1 subnets
C        12.1.1.0 is directly connected, FastEthernet0/0
```

//由于我们 passive 掉 R2 的 F0/0 口，R2 不再发更新给 R1,R1 上也学不到了 RIP 的路由。

这时我们再去单播指邻居来实验路由器通过单播的方式学习路由：

在 R1 上：

```
R1(config)#router rip
R1(config-router)#neighbor 12.1.1.2
```

在 R2 上：

```
R2(config)#router rip
R2(config-router)#neighbor 12.1.1.1
```
//以上是在 R1,与 R2 上相互指对方为邻居。以单播的方式发送路由更新。

再到 R1 上去查看路由表：

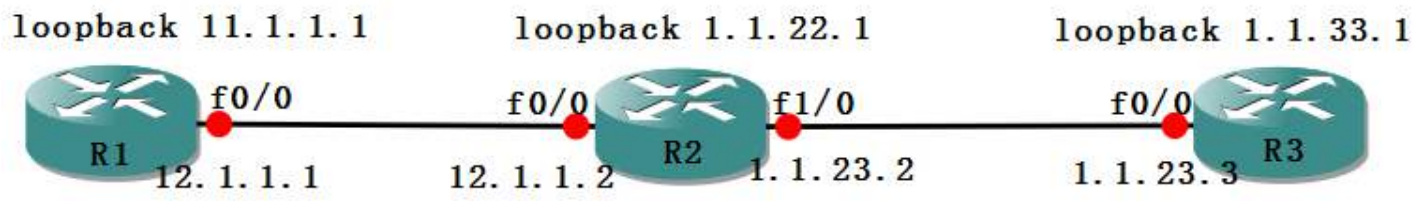
```
R1#show ip route
————此部分省略————

      33.0.0.0/24 is subnetted, 1 subnets
R        33.1.1.0 [120/2] via 12.1.1.2, 00:00:01, FastEthernet0/0
      23.0.0.0/24 is subnetted, 1 subnets
R        23.1.1.0 [120/1] via 12.1.1.2, 00:00:01, FastEthernet0/0
      22.0.0.0/24 is subnetted, 1 subnets
R        22.1.1.0 [120/1] via 12.1.1.2, 00:00:01, FastEthernet0/0
      11.0.0.0/24 is subnetted, 1 subnets
C        11.1.1.0 is directly connected, Loopback0
      12.0.0.0/24 is subnetted, 1 subnets
C        12.1.1.0 is directly connected, FastEthernet0/0
```
//以上表明 R1 上通过了单播更新学到了 RIP 的路由。

实验 3.6：RIPV2 自动汇总

实验目的

理解 RIP 的自动汇总的工作方式

实验拓扑



实验步骤

因为要理解 RIP 的自动汇总特性，所以此实验的 IP 地址规划有变，看配置：

在 R1 上：

```
R1(config)#interface fastEthernet 0/0
R1(config-if)#ip address 12.1.1.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#exit
```

```
R1(config)#interface loopback 0
R1(config-if)#ip address 11.1.1.1 255.255.255.0
```

在 R2 上：

```
R2(config)#interface fastEthernet 0/0
R2(config-if)#ip address 12.1.1.2 255.255.255.0
R2(config-if)#no shutdown
R2(config-if)#exit
R2(config)#interface fastEthernet 1/0
R2(config-if)#ip address 1.1.23.2 255.255.255.0
R2(config-if)#no shutdown
R2(config-if)#exit
R2(config)#interface loopback 0
R2(config-if)#ip address 1.1.22.1 255.255.255.0
R2(config-if)#exit
```

在 R3 上：

```
R3(config)#interface fastEthernet 0/0
R3(config-if)#ip address 1.1.23.3 255.255.255.0
R3(config-if)#no shutdown
R3(config-if)#exit
R3(config)#interface loopback 0
R3(config-if)#ip address 1.1.33.1 255.255.255.0
```

IP 地址配好以后，再配好 RIP。

在 R1 上:

```
R1(config)#router rip
R1(config-router)#version 2
R1(config-router)#auto-summary
```

//以上表示开始 RIP 的自动汇总功能，当然，默认就是开启的。

```
R1(config-router)#network 11.0.0.0
R1(config-router)#network 12.0.0.0
```

在 R2 上：

```
R2(config)#router rip
R2(config-router)#version 2
R2(config-router)#auto-summary
R2(config-router)#network 12.0.0.0
R2(config-router)#network 1.0.0.0
```

在 R3 上：

```
R3(config)#router rip
R3(config-router)#version 2
```

```
R3(config-router)#auto-summary
R3(config-router)#network 1.0.0.0
//配置完成以后分别去看路由表。
```

在 R1 上：

```
R1#show ip route rip
————此部分省略————
R       1.0.0.0/8 [120/1] via 12.1.1.2, 00:00:18, FastEthernet0/0
//以上表示 R1 上收到的 R2,R3 的路由都被汇总成了主类。
```

再去 R2 上：

```
R2#show ip route rip
      1.0.0.0/24 is subnetted, 3 subnets
R        1.1.33.0 [120/1] via 1.1.23.3, 00:00:11, FastEthernet1/0
R       11.0.0.0/8 [120/1] via 12.1.1.1, 00:00:02, FastEthernet0/0
//以上表示 R2 收到 R3 的环回口路由依然是明细的，原因是 R2 和 R3 处于同一个主类网络。在同一个主类网内，是传明细的。到主类网络的边界，会汇总。
```

在去 R3 上查看，我们可以推断出学到 R2 的环回口会是明细的，学到 R1 的环回口会是主类的。

```
R3#show ip route rip
      1.0.0.0/24 is subnetted, 3 subnets
R        1.1.22.0 [120/1] via 1.1.23.2, 00:00:05, FastEthernet0/0
R       11.0.0.0/8 [120/2] via 1.1.23.2, 00:00:05, FastEthernet0/0
R       12.0.0.0/8 [120/1] via 1.1.23.2, 00:00:05, FastEthernet0/0
//结果和我们想的一样。
```

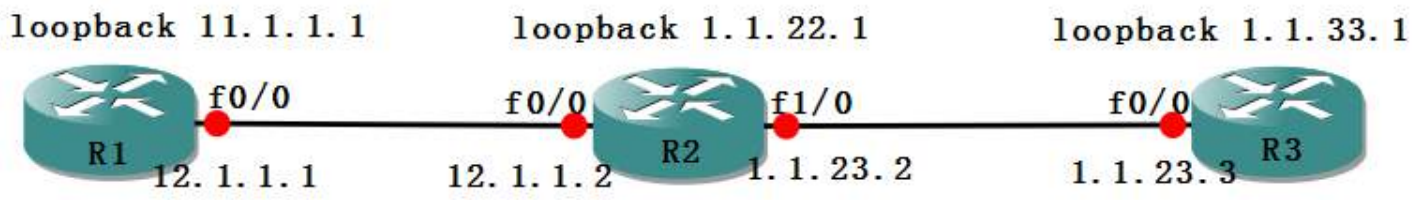

不要关闭模拟器，因为接下来的手动汇总是在此基础之上的。
我们先关闭 RIP 的自动汇总再接着完成以下的实验。

实验 3.7：RIPV2 手动汇总

实验目的

理解掌握 RIP 的手动汇总

实验拓扑



实验步骤

在 3.6 的基础之上关闭 RIP 的自动汇总。来完成此实验。

在 R1 上：

```
R1(config)#router rip
R1(config-router)#no auto-summary
```

在 R2 上：

```
R2(config)#router rip
R2(config-router)#no auto-summary
```

在 R3 上：

```
R3(config)#router rip
R3(config-router)#no auto-summary
```

都关闭自动汇总后，我们去查看 R1 的路由表：

```
R1#show ip route rip
     1.0.0.0/24 is subnetted, 3 subnets
R       1.1.22.0 [120/1] via 12.1.1.2, 00:00:00, FastEthernet0/0
R       1.1.23.0 [120/1] via 12.1.1.2, 00:00:00, FastEthernet0/0
R       1.1.33.0 [120/2] via 12.1.1.2, 00:00:00, FastEthernet0/0
//以上表明 R1 学到的又全是明细了。这时我们可以去 R2 上手动汇总这三条路由条目。
```


在 R2 上：

```
R2(config)#interface fastEthernet 0/0
//RIP 的手动汇总是在接口下做的，所以先进到 R2 连接 R1 的接口下
R2(config-if)#ip summary-address rip 1.1.0.0 255.255.0.0
//以上是汇总成一条 16 位的路由。
```

再去 R1 上查看路由表：

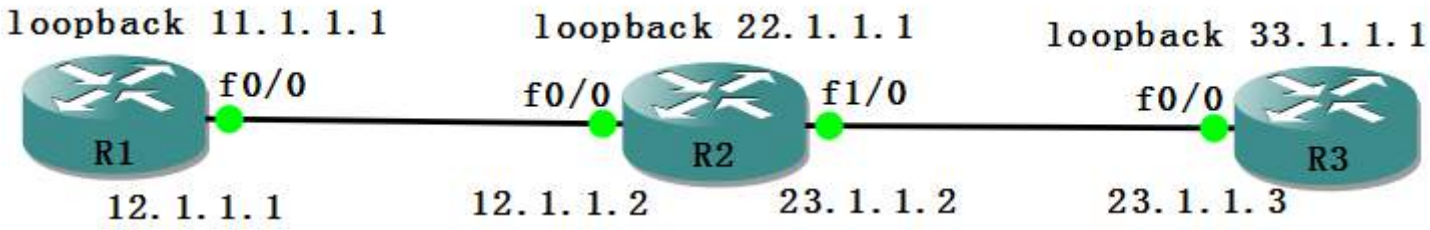
```
R1#clear ip route *
R1#show ip route rip
     1.0.0.0/16 is subnetted, 1 subnets
R       1.1.0.0 [120/1] via 12.1.1.2, 00:00:02, FastEthernet0/0
//以上表明刚才的三条路由都被我们汇总成了一条 16 位的路由了。
```

实验 3.8：RIPV2 产生默认路由

实验目的

掌握如何在 RIP 中产生一种默认路由

实验拓扑



实验步骤

1：先保证直连的连通性：

在 R1 上：

```
R1(config)#interface fastEthernet 0/0
R1(config-if)#ip address 12.1.1.1 255.255.255.0
R1(config-if)#no shutdown
R1(config)#interface loopback 0
R1(config-if)#ip address 11.1.1.1 255.255.255.0
```

在 R2 上：

```
R2(config)#interface fastEthernet 0/0
R2(config-if)#ip address 12.1.1.2 255.255.255.0
R2(config-if)#no shutdown
R2(config-if)#exit
R2(config)#interface loopback 0
R2(config-if)#ip address 22.1.1.1 255.255.255.0
R2(config-if)#exit
R2(config)#interface fastEthernet 1/0
R2(config-if)#ip address 23.1.1.2 255.255.255.0
R2(config-if)#no shutdown
```

在 R3 上：

```
R3(config)#interface fastEthernet 0/0
R3(config-if)#ip address 23.1.1.3 255.255.255.0
R3(config-if)#no shutdown
R3(config)#interface loopback 0
R3(config-if)#ip address 33.1.1.1 255.255.255.0
```

配好以后去相互 PING 一下对方看直连是否可达了。

2：接下来做 RIP 的配置：

在 R1 上：

```
R1(config)#router rip
R1(config-router)#version 2
R1(config-router)#no auto-summary
R1(config-router)#network 12.0.0.0
R1(config-router)#network 11.0.0.0
```

在 R2 上：

```
R2(config)#router rip
R2(config-router)#version 2
R2(config-router)#no auto-summary
R2(config-router)#network 12.0.0.0
R2(config-router)#network 23.0.0.0
```

在 R3 上：

```
R3(config)#router rip
R3(config-router)#version 2
R3(config-router)#no auto-summary
R3(config-router)#network 23.0.0.0
R3(config-router)#network 33.0.0.0
```

配好以后查看一下路由表，看是否有相互学习到对方的路由。

3：在 R3 上产生一条默认路由。共有四种方法。先用第一种方法：

①：用命令 default-information originate 产生

在 R3 上：

```
R3(config)#router rip
R3(config-router)#default-information originate
```

//以上是用一条命令的方法让 RIP 产生一条默认路由，此方法最简单易用。推荐使用。
再去 R1 和 R2 查看路由表是否学习到了此路由，这里只显示 R1 的查看结果。

在 R1 上：

```
R1#show ip route rip
      33.0.0.0/24 is subnetted, 1 subnets
R        33.1.1.0 [120/2] via 12.1.1.2, 00:00:09, FastEthernet0/0
      23.0.0.0/24 is subnetted, 1 subnets
R        23.1.1.0 [120/1] via 12.1.1.2, 00:00:09, FastEthernet0/0
R*    0.0.0.0/0 [120/2] via 12.1.1.2, 00:00:09, FastEthernet0/0
```

```
//以上表明 R1 学习到了这条默认路由。
```

```
R3(config)#router rip
R3(config-router)#no default-information originate
//以上是恢复到没产生默认路由的配置，以便我们再用第二种方法。
```

②用命令 ip default-network 产生：

注意，此命令后面跟的网络号必须是主类的。

先在 R3 上产生一条主类的路由，可以用直连产生，或者用静态的方法产生。

在 R3 上：

```
R3(config)#interface loopback 1
R3(config-if)#ip address 10.1.1.1 255.0.0.0
R3(config)#ip default-network 10.0.0.0
```

再去 R1 上查看：

```
R1#clear ip route *
R1#show ip route rip
      33.0.0.0/24 is subnetted, 1 subnets
R        33.1.1.0 [120/2] via 12.1.1.2, 00:00:03, FastEthernet0/0
      23.0.0.0/24 is subnetted, 1 subnets
R        23.1.1.0 [120/1] via 12.1.1.2, 00:00:03, FastEthernet0/0
R*    0.0.0.0/0 [120/2] via 12.1.1.2, 00:00:03, FastEthernet0/0
//以上表明 R1 已经学习到了这条默认路由
```


回到 R3 上恢复配置：

```
R3(config)#no ip default-network 10.0.0.0
R3(config)#interface loopback 1
R3(config-if)#shutdown
```

③用宣告的方式产生。

首先在 R3 上要手动写一条静态默认，再把它宣告进 RIP 里。

在 R3 上：

```
R3(config)#ip route 0.0.0.0 0.0.0.0 null 0
//此条默认只能指向一个接口
R3(config)#router rip
R3(config-router)#network 0.0.0.0
```

去 R1 查看结果

```
R1#clear ip route *
R1#show ip route rip
```

```
      33.0.0.0/24 is subnetted, 1 subnets
R        33.1.1.0 [120/2] via 12.1.1.2, 00:00:01, FastEthernet0/0
      23.0.0.0/24 is subnetted, 1 subnets
R        23.1.1.0 [120/1] via 12.1.1.2, 00:00:01, FastEthernet0/0
R*    0.0.0.0/0 [120/2] via 12.1.1.2, 00:00:01, FastEthernet0/0
```

//以上表明 R1 已经学习到了这条路由，用此方法有一个不好的地方，就是在 R3 上我们会把所有的接口都宣告进 RIP 进程里。

恢复配置，继续下面的实验：

```
R3(config)#router rip
R3(config-router)#no network 0.0.0.0
```

④把刚才写的那条默认路由重分布进 RIP 里。

在 R3 上：

```
R3(config)#router rip
R3(config-router)#redistribute static
```

再去 R1 上查看结果：

```
R1#show ip route rip
      33.0.0.0/24 is subnetted, 1 subnets
R        33.1.1.0 [120/2] via 12.1.1.2, 00:00:04, FastEthernet0/0
      23.0.0.0/24 is subnetted, 1 subnets
R        23.1.1.0 [120/1] via 12.1.1.2, 00:00:04, FastEthernet0/0
R*    0.0.0.0/0 [120/2] via 12.1.1.2, 00:00:03, FastEthernet0/0
```

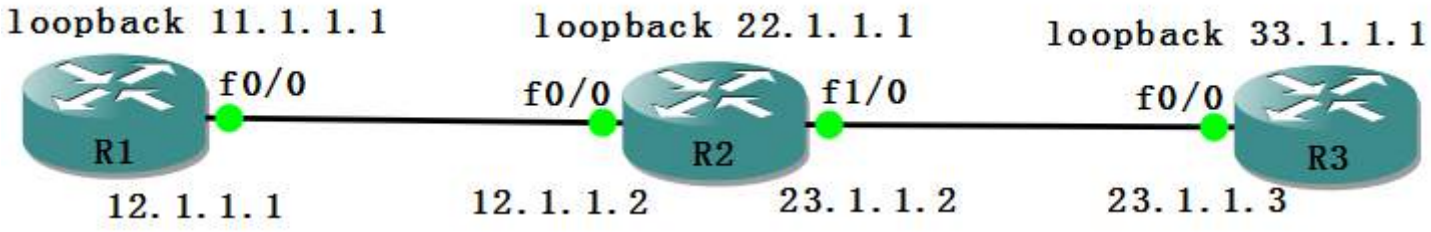

//以上便是在 RIP 里产生默认路由的 4 种方式。

实验 3.9：RIPV2 明文认证

实验目的

熟练掌握 RIP 的认证

实验拓扑



实验步骤

1：根据拓扑确保直连的可达性。

在 R1 上：

```
R1(config)#interface fastEthernet 0/0
R1(config-if)#ip address 12.1.1.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#interface loopback 0
R1(config-if)#ip address 11.1.1.1 255.255.255.0
```

在 R2 上：

```
R2(config)#interface fastEthernet 0/0
R2(config-if)#ip address 12.1.1.2 255.255.255.0
R2(config-if)#no shutdown
R2(config-if)#exit
R2(config)#interface loopback 0
R2(config-if)#ip address 22.1.1.1 255.255.255.0
R2(config-if)#exit
R2(config)#interface fastEthernet 1/0
R2(config-if)#ip address 23.1.1.2 255.255.255.0
R2(config-if)#no shutdown
R2(config-if)#exit
```


在 R3 上：

```
R3(config)#interface fastEthernet 0/0
R3(config-if)#ip address 23.1.1.3 255.255.255.0
R3(config-if)#no shutdown
R3(config-if)#exit
R3(config)#interface loopback 0
R3(config-if)#ip address 33.1.1.1 255.255.255.0
```

2：配置 RIP。

在 R1 上：

```
R1(config)#router rip
R1(config-router)#version 2
R1(config-router)#no auto-summary
R1(config-router)#network 11.0.0.0
R1(config-router)#network 12.0.0.0
```

在 R2 上：

```
R2(config)#router rip
R2(config-router)#version 2
R2(config-router)#no auto-summary
R2(config-router)#network 12.0.0.0
R2(config-router)#network 22.0.0.0
R2(config-router)#network 23.0.0.0
```

在 R3 上：

```
R3(config)#router rip
R3(config-router)#version 2
R3(config-router)#no auto-summary
R3(config-router)#network 23.0.0.0
R3(config-router)#network 33.0.0.0
```


3：检查是否正常学习到了路由。

在 R1 上：

```
R1#show ip route rip
      33.0.0.0/24 is subnetted, 1 subnets
R        33.1.1.0 [120/2] via 12.1.1.2, 00:00:02, FastEthernet0/0
      23.0.0.0/24 is subnetted, 1 subnets
R        23.1.1.0 [120/1] via 12.1.1.2, 00:00:02, FastEthernet0/0
      22.0.0.0/24 is subnetted, 1 subnets
R        22.1.1.0 [120/1] via 12.1.1.2, 00:00:02, FastEthernet0/0
```

4：我们在 R1 与 R2 之间做一个明文的认证。

在 R1 上：首先要配置一个钥匙

```
R1(config)#key chain yeslab
//配置钥匙链的名字
R1(config-keychain)#key 1
//配置 KEY ID
R1(config-keychain-key)#key-string cisco
//配置密匙
```

```
R1(config-keychain-key)#end
R1#conf terminal
R1(config)#interface fastEthernet 0/0
R1(config-if)#ip rip authentication mode text
//指定认证的类型为明文的。
R1(config-if)#ip rip authentication key-chain yeslab
//调用我们刚才配置的钥匙链
做完以后去查看一下路由表：
R1#clear ip route *
R1#show ip route rip
会发现路由表里显示为空，因为 R1 与 R2 的认证不通过导至 R1 学不到 R2 的路由。
```

再到 R2 上：

```
R2(config)#key chain yeslab1
//此钥匙链的名字可以不与 R1 的一样
R2(config-keychain)#key 1
R2(config-keychain-key)#key-string cisco
//KEY ID 的密匙必须与 R1 的一致
R2(config-keychain-key)#end
R2#configure terminal
R2(config)#interface fastEthernet 0/0
R2(config-if)#ip rip authentication mode text
R2(config-if)#ip rip authentication key-chain yeslab1
```


做完以上配置再到 R1 上去查看路由信息：

```
R1#clear ip route *
R1#show ip route rip
      33.0.0.0/24 is subnetted, 1 subnets
R        33.1.1.0 [120/2] via 12.1.1.2, 00:00:03, FastEthernet0/0
      23.0.0.0/24 is subnetted, 1 subnets
R        23.1.1.0 [120/1] via 12.1.1.2, 00:00:03, FastEthernet0/0
      22.0.0.0/24 is subnetted, 1 subnets
R        22.1.1.0 [120/1] via 12.1.1.2, 00:00:03, FastEthernet0/0
//以上表示 R1 重新学习到了路由信息，我们也做完了 RIP 的明文认证。
```

明文认证总结：只发送 KEY ID 最小的 KEY，并不携带 KEY ID，接收方与 KEY 列表中所有的 KEY 匹配，只要有一个能匹配上则通过认证。

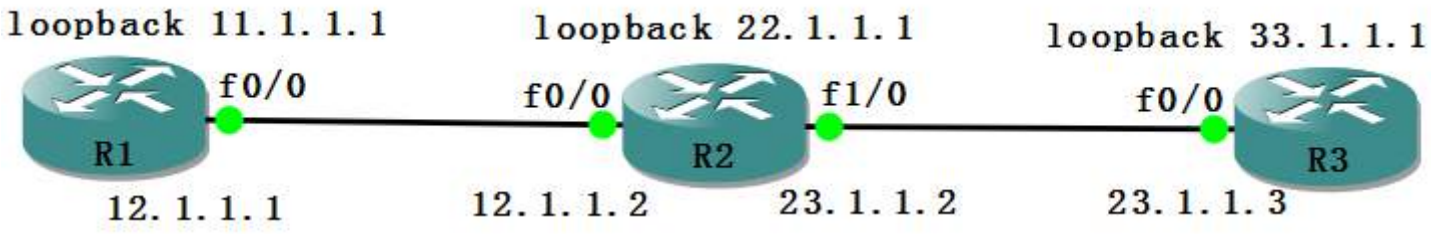
不要关闭模拟器，接着做下面的实验。

实验 3.10：RIPV2 MD5 认证

实验目的

熟练掌握 RIP 的 MD5 认证配置

实验拓扑



实验步骤

此实验是在实验 3.9 的基础之上做的。
请确保 RIPV2 已经正常工作。
我们要在 R2 与 R3 之间实施 RIPV2 的 MD5 认证：
RIP 的明文认证与 MD5 认证的区别只有一点，就是开启认证方式的时候。

在 R2 上：

```
R2(config)#key chain yeslab2
R2(config-keychain)#key 1
R2(config-keychain-key)#key-string yeslab
R2(config-keychain-key)#end
R2#configure terminal
R2(config)#interface fastEthernet 1/0
R2(config-if)#ip rip authentication mode md5
 //以上命令是开启 RIP 的 MD5 的认证方式
R2(config-if)#ip rip authentication key-chain yeslab2
```

在 R3 上：

```
R3(config)#key chain yeslab3
R3(config-keychain)#key 1
//此处的 KID 最好保持一致。
R3(config-keychain-key)#key-string yeslab
R3(config-keychain-key)#end
R3#configure terminal
R3(config)#interface fastEthernet 0/0
R3(config-if)#ip rip authentication mode md5
R3(config-if)#ip rip authentication key-chain yeslab3
```

去查看 R3 的路由表：

```
R3#show ip route rip
      22.0.0.0/24 is subnetted, 1 subnets
R        22.1.1.0 [120/1] via 23.1.1.2, 00:00:26, FastEthernet0/0
      11.0.0.0/24 is subnetted, 1 subnets
R        11.1.1.0 [120/2] via 23.1.1.2, 00:00:26, FastEthernet0/0
      12.0.0.0/24 is subnetted, 1 subnets
R        12.1.1.0 [120/1] via 23.1.1.2, 00:00:26, FastEthernet0/0
```

//R3 已经正常的学习到了 R2 的路由了。

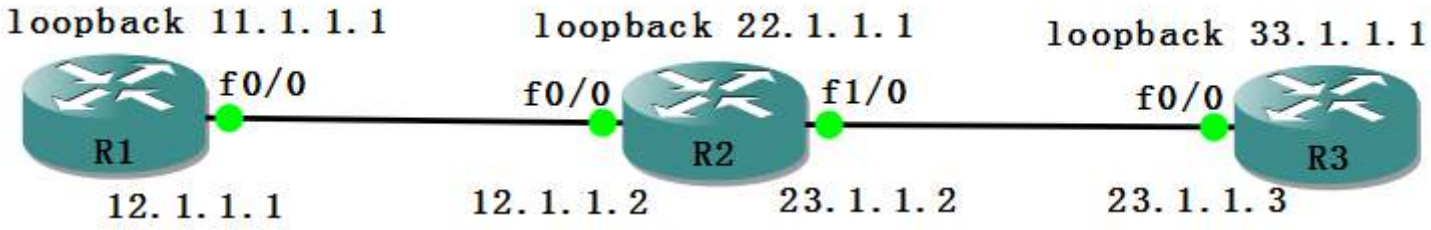
下面的实验可在此基础上完成

实验 3.11：RIP 过滤路由

实验目的

熟练运用访问控制列表在 RIP 中过滤一条路由。

实验拓扑



实验步骤

确保 RIPV2 已经正常运行。这里不再指导。

查看 R3 的路由表：

```
R3#show ip route rip
      22.0.0.0/24 is subnetted, 1 subnets
R        22.1.1.0 [120/1] via 23.1.1.2, 00:00:15, FastEthernet0/0
      11.0.0.0/24 is subnetted, 1 subnets
R        11.1.1.0 [120/2] via 23.1.1.2, 00:00:15, FastEthernet0/0
      12.0.0.0/24 is subnetted, 1 subnets
R        12.1.1.0 [120/1] via 23.1.1.2, 00:00:15, FastEthernet0/0
```

我们要去过滤掉路由表中的 11.1.1.0/24 这条路由。

步骤 1：写访问控制列表去拒绝这条路由：
在 R3 上:

```
R3(config)#access-list 10 deny 11.1.1.0
//以上是拒绝这条路由。
R3(config)#access-list 10 permit any
//以上是允许别的所有路由
```

```
R3(config)#router rip
R3(config-router)#distribute-list 10 in
//以上是在 R3 的 RIP 进程下去过滤，调用的 10 就是刚写的访问控制列表 10
这里我们用的方向是 in 方向
```

想一想，我们可以在 R2 的 out 方向去过滤吗？

```
R3#clear ip route *
R3#show ip route rip
      22.0.0.0/24 is subnetted, 1 subnets
R        22.1.1.0 [120/1] via 23.1.1.2, 00:00:03, FastEthernet0/0
      12.0.0.0/24 is subnetted, 1 subnets
R        12.1.1.0 [120/1] via 23.1.1.2, 00:00:03, FastEthernet0/0
//以上表示我们已经成功过滤掉了这条路由。
```


实验第四部分：EIGRP

EIGRP(Enhanced Interior Gateway Routing Protocol，增强型内部网关路由协议)是Cisco 公司开发的一个平衡混合型路由协议，它融合了距离向量和链路状态两种路由协议的优点，支持IP、IPX、AppleTalk 等多种网络层协议。由于TCP / IP 是当今网络中最常用的协议，因此本书只讨论IP 网络环境中的EIGRP。

EIGRP 概述

EIGRP 是一个高效的路由协议，它的特点如下:

1. 通过发送和接收Hello 包来建立和维持邻居关系，并交换路由信息;
2. 采用组播（224.0.0.10）或单播进行路由更新;
3. EIGRP 的管理距离为90 或170;
4. 采用触发更新，减少带宽占用;

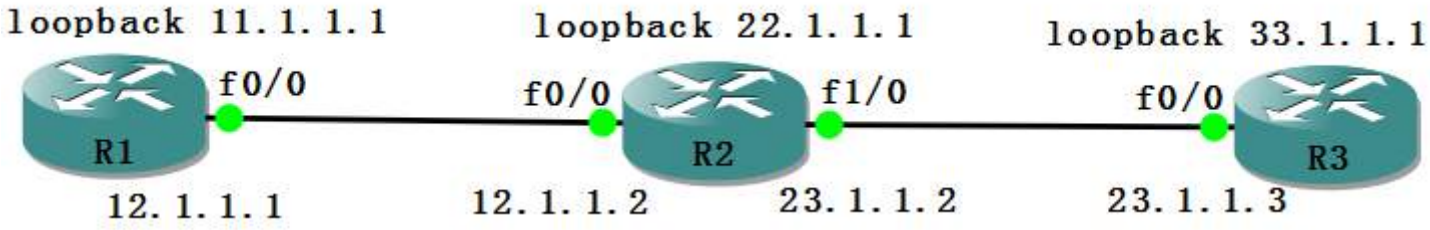
5. 支持可变长子网掩码（VLSM），默认开启自动汇总功能；
6. 支持IP、IPX、AppleTalk 等多种网络层协议；
7. 对每一种网络协议，EIGRP 都维持独立的邻居表、拓扑表和路由表；
8. EIGRP 使用Diffusing Update 算法(DUAL)来实现快速收敛.并确保没有路由环路；
9. 存储整个网络拓扑结构的信息，以便快速适应网络变化；
10. 支持等价和非等价的负载均衡；
11. 使用可靠传输协议（RTP）保证路由信息传输的可靠性；
12. 无缝连接数据链路层协议和拓扑结构，EIGRP 不要求对OSI 参考模型的2 层协议做特别的配置。

实验 4.1 EIGRP 基础配置

实验目的

理解并掌握如何配置 EIGRP 协议，为避免冗余，在 CCNA 的实验手册里，针对 EIGRP 我只会安排这一个实验。要想尝试更多的 EIGRP 实验，请参考即将与你见面的 CCNP 苏函出品。嘿嘿

实验拓扑



实验步骤
首先请按拓扑配置好 IP 地址，确保直连通.这里不再指导：

在 R1 上：

```
R1(config)#router eigrp 1
//起 EIGRP 的进程，注意后面的 EIGRP 的 AS 号必须一样才能相互学习路由
R1(config-router)#no auto-summary
//EIGRP 也有自动汇总的功能，默认开启，我们一般会关闭他。
R1(config-router)#network 11.0.0.0
//用主类宣告的方式宣告进一条路由。这与 RIP 是一样的。
R1(config-router)#network 12.1.1.0 0.0.0.255
//用子网宣告的方式宣告一条路由，这样会更精确
```

在 R2 上：

```
R2(config)#router eigrp 1
R2(config-router)#no auto-summary
R2(config-router)#network 12.1.1.2 0.0.0.0
//以上是用精确宣告一个 IP 地址的方式宣告进 EIGRP 进程
```

```
R2(config-router)#network 22.1.1.1 0.0.0.0
R2(config-router)#network 23.1.1.2 0.0.0.0
```

在 R3 上：

```
R3(config)#router eigrp 1
R3(config-router)#no auto-summary
R3(config-router)#network 23.1.1.0 0.0.0.255
R3(config-router)#network 33.1.1.0 0.0.0.255
```

去 R1 上查看从 EIGRP 学到的路由：
在 R1 上：

```
R1#show ip route eigrp
      33.0.0.0/24 is subnetted, 1 subnets
D        33.1.1.0 [90/158720] via 12.1.1.2, 00:01:09, FastEthernet0/0
      23.0.0.0/24 is subnetted, 1 subnets
D        23.1.1.0 [90/30720] via 12.1.1.2, 00:12:40, FastEthernet0/0
      22.0.0.0/24 is subnetted, 1 subnets
D        22.1.1.0 [90/156160] via 12.1.1.2, 00:12:44, FastEthernet0/0
```

在 R2 上：


```
R2#show ip route eigrp
      33.0.0.0/24 is subnetted, 1 subnets
D        33.1.1.0 [90/156160] via 23.1.1.3, 00:03:39, FastEthernet1/0
      11.0.0.0/24 is subnetted, 1 subnets
D        11.1.1.0 [90/156160] via 12.1.1.1, 00:15:21, FastEthernet0/0
```

在 R3 上：

```
R3#show ip route eigrp
      22.0.0.0/24 is subnetted, 1 subnets
D        22.1.1.0 [90/156160] via 23.1.1.2, 00:03:57, FastEthernet0/0
      11.0.0.0/24 is subnetted, 1 subnets
D        11.1.1.0 [90/158720] via 23.1.1.2, 00:03:57, FastEthernet0/0
      12.0.0.0/24 is subnetted, 1 subnets
D        12.1.1.0 [90/30720] via 23.1.1.2, 00:03:57, FastEthernet0/0
```

//以上表明所有的路由器都学习到了相互的路由条目。至于我们应该用什么样的方式来宣告进一条路由进 EIGRP。这根据我们的工程需要，没有必须。

更多 EIGRP 的实验请参考 CCNP 的实验手册。

实验第五部分：OSPF 基本配置

OSPF（Open Shortest Path First,开放最短链路优先）路由协议是典型的链路状态路由协议。OSPF 由IETF 在20世纪80 年代末期开发，OSPF 是SPF 类路由协议中的开放式版本。最初的OSPF 规范体现在RFC1131 中，被称为OSPF 版本1，但是版本1 很快被进行了重大改进的版本所代替，这个新版本体现在RFC1247 文档中。RFC1247 被称为OSPF 版本2,是为了明确指出其在稳定性和功能性方面的实质性改进。这个OSPF 版本有许多更新文档，每一个更新都是对开放标准的精心改进。接下来的一些规范出现在RFC1583 和2328 中。OSPF 版本2 的最新版体现在RFC 2328 中。而OSPF 版本3 是关于IPv6 的。OSPF 的内容多而复杂，所以本书分了多个章节来介绍。本章只讨论单区域的OSPF。

OSPF 概述

OSPF 作为一种内部网关协议（Interior Gateway Protocol，IGP），用于在同一个自治系统（AS）中的路由器之间交换路由信息。OSPF 的特性如下:

1. 可适应大规模网络；
2. 收敛速度快；
3. 无路由环路；
4. 支持VLSM 和CIDR；
5. 支持等价路由；
6. 支持区域划分，构成结构化的网络；
7. 提供路由分级管理；
8. 支持简单口令和MD5 认证；
9. 以组播方式传送协议报文；
10. OSPF 路由协议的管理距离是110；
11. OSPF 路由协议采用cost 作为度量标准；
12. OSPF 维护邻居表、拓扑表和路由表。

另外,OSPF 将网络划分为四种类型：广播多路访问型(BMA)、非广播多路访问型(NBMA)、点到点型(Point-to-Point)、点到多点型（Point-to-MultiPoint）。不同的二层链路的类型需要OSPF 不同的网络类型来适应。

下面的几个术语是学习OSPF 要掌握的：

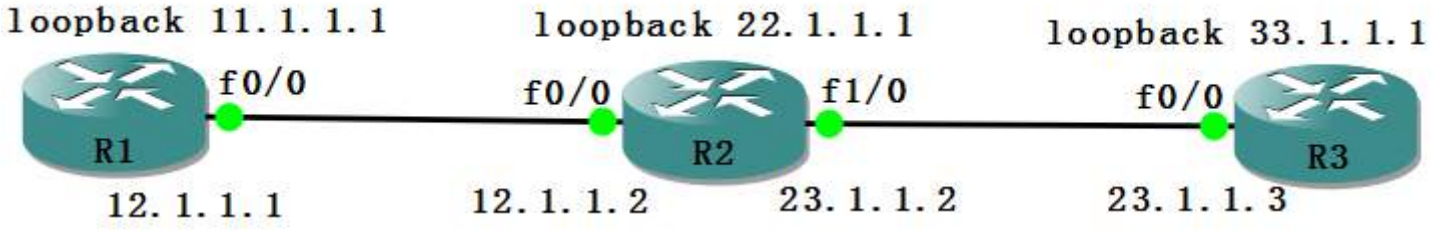
1. 链路：链路就是路由器用来连接网络的接口；
2. 链路状态：用来描述路由器接口及其与邻居路由器的关系。所有链路状态信息构成链路状态数据库；
3. 区域：有相同的区域标志的一组路由器和网络的集合。在同一个区域内的路由器有相同的链路状态数据库；
4. 自治系统：采用同一种路由协议交换路由信息的路由器及其网络构成一个自治系统；
5. 链路状态通告（LSA）：LSA 用来描述路由器的本地状态，LSA 包括的信息有关于路由器接口的状态和所形成的邻接状态；
6. 最短路经优先（SPF）算法：是OSPF 路由协议的基础。SPF 算法有时也被称为Dijkstra算法，这是因为最短路径优先算法(SPF)是Dijkstra 发明的。OSPF 路由器利用 SPF，独立地计算出到达任意目的地的最佳路由。

实验 5.1 OSPF 基本配置

实验目标

熟练掌握配置 OSPF

实验拓扑



实验步骤

先确定直连的连通性，这里将不再指导

在 R1 上：

```
R1(config)#router ospf 1
//激活一个 OSPF 的进程，后面的进程号只具有本地意义，不同路由器之间不用一样。
R1(config-router)#router-id 11.1.1.1
//这里是手动配置一个 OSPF 的 router-ID.
R1(config-router)#network 11.1.1.0 0.0.0.255 area 0
//宣告一个网段进 OSPF 的进程，并且属于区域 0
R1(config-router)#network 12.1.1.0 0.0.0.255 area 0
```


在 R2 上：

```
R2(config)#router ospf 1
R2(config-router)#router-id 22.1.1.1
R2(config-router)#network 12.1.1.0 0.0.0.255 area 0
R2(config-router)#network 23.1.1.0 0.0.0.255 area 0
R2(config-router)#network 22.1.1.0 0.0.0.255 area 0
```

在 R3 上：

```
R3(config)#router ospf 1
R3(config-router)#router-id 33.1.1.1
R3(config-router)#network 23.1.1.0 0.0.0.255 area 0
R3(config-router)#network 33.1.1.0 0.0.0.255 area 0
```

在 R2 上去查看 OSPF 的邻居

```
R2#show ip ospf neighbor
Neighbor ID     Pri   State           Dead Time   Address         Interface
33.1.1.1          1   FULL/DR         00:00:33    23.1.1.3        FastEthernet1/0
11.1.1.1          1   FULL/DR         00:00:30    12.1.1.1        FastEthernet0/0
//以上表示 R2 已经学到了二个邻居。
```

再去查看路由表：

```
R2#show ip route ospf
      33.0.0.0/32 is subnetted, 1 subnets
O        33.1.1.1 [110/2] via 23.1.1.3, 00:04:33, FastEthernet1/0
      11.0.0.0/32 is subnetted, 1 subnets
O        11.1.1.1 [110/2] via 12.1.1.1, 00:04:33, FastEthernet0/0
```

//以上表示路由也学到了。R1,R3 上的路由表这里不再展示。

OSPF 的更多实验请参考 CCNP 实验手册。

实验第六部分：交换机基础配置

交换机是局域网中最重要的设备，交换机是基于MAC 来进行工作的。和路由器类似，交换机也有IOS，IOS 的基本使用方法是一样的。本章将简单介绍交换机的一些基本配置，以及交换机独特的密码恢复、IOS 恢复步骤。关于VLAN、Trunk 等将在后面章节介绍。

交换机简介

交换机是第二层的设备，可以隔离冲突域。交换机是基于收到的数据帧中的源MAC 地址和目的MAC 地址来进行工作。交换机的作用主要有这么两个：一个是维护CAM（ContextAddress Memory）表，该表是MAC 地址和交换机端口的映射表；另一个是根据CAM 来进行数据帧的转发。交换机对帧的处理有三种：交换机收到帧后，查询CAM 表，如果能查询到目的计算机所在的端口，并且目的计算机所在的端口不是交换机接收帧的源端口，交换机将把帧从这一端口转发出去（Forward）；如果该计算机所在的端口和交换机接收帧的源端口是同一端口，交换机将过滤掉该帧（Filter）；如果交换机不能查询到目的计算机所在的端口，交换机将把帧从源端口以外的其他所有端口上发送出去，这称为泛洪（Flood），当交换机接收到的是帧是广播帧或者多播帧，交换机也会泛洪帧。

以太网交换机转发数据帧有三种交换方式，

（1） 存储转发（Store-and-Forward）

存储转发方式是先存储后转发的方式。它把从端口输入的数据帧先全部接收并存储起来；然后进行CRC（循环冗余码校验）检查，把错误帧丢弃；最后才取出数据帧目的地址，查找地址表后进行过滤和转发。存储转发方式延迟大；但是它可以对进入交换机的数据包进行高级别的错误检测。这种方式可以支持不同速度的端口间的转发。

（2） 直接转发（Cut-Through）

交换机在输入端口检测到一个数据帧时，检查该帧的帧头，只要获取了帧的目的地址，就开始转发帧。它的优点是：开始转发前不需要读取整个完整的帧，延迟非常小。它的缺点是：不能提供错误检测能力。

（3） 无碎片（Fragment-Free）

这是改进后的直接转发，是介于前两者之间的一种解决方法。无碎片方法在读取数据帧的长前64个字节后，就开始转发该帧。这种方式虽然也不提供数据校验，但是能够避免大多数的错误。它的数据处理速度比直接转发方式慢，但比存储转发方式快许多。

CISCO 交换机和路由器一样，本质上也是一台特殊的计算机，也有CPU、RAM 等部件。也采用IOS，所以交换机的很多基本配置（例如密码、主机名等）和路由器是类似的。

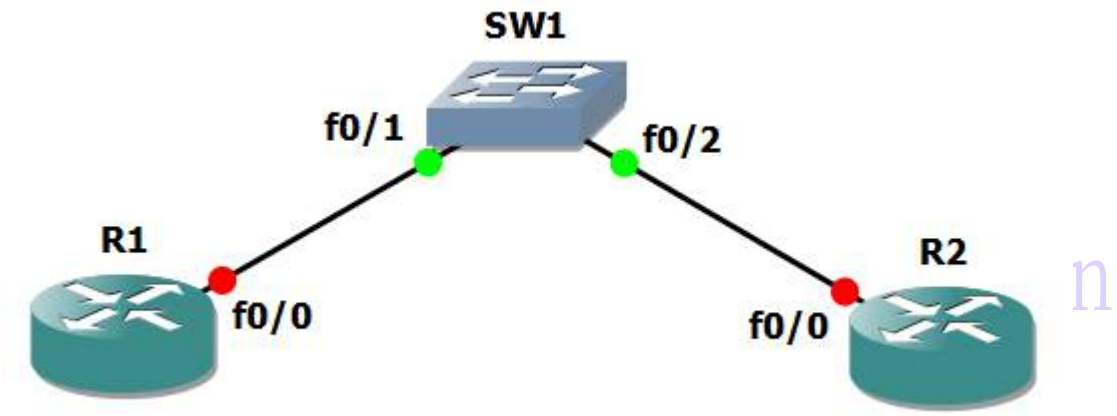
实验 6.1　VLAN 的划分

VLAN技术用于分割交换网络的广播域，使得广播域的范围减小。每一个VLAN是一个广播域，一个VLAN域的广播数据帧不会扩散到另一VLAN域

实验目的

熟练掌握在交换机上添加删除 VLAN
把交换机接口划分到特定 vlan

实验拓扑：



实验步骤

注意，SW1 可以用 GNS3 中的路由器插 NM-16ESW 模块来做此实验。
要配置 VLAN，首先要先创建 VLAN，然后才把交换机的端口划分到特定的端口上：
步骤 1：在 SW1 上创建 VLAN

```
SW1#vlan database
//在特权模式下进入 VLAN 配置模式。
SW1(vlan)#vlan 2
VLAN 2 added:
    Name: VLAN0002
//以上创建的 2 是 vlan 的编号，VLAN0002 是该 VLAN 的名字。
sw1(vlan)#vlan 3
VLAN 3 added:
    Name: VLAN0003
sw1(vlan)#apply
```

```
// apply 命令是应用刚才创建的 VLAN。
APPLY completed.
sw1(vlan)#exit
APPLY completed.
Exiting....
```
创建了 valn2、vlan3，并在退出后配置生效。

【提示】交换机中的VLAN 信息存放在单独的文件中flash:vlan.dat，因此如果要完全清除交换机的配置，除了使用“erase starting-config”命令外，还要使用“delete
flash:vlan.dat”命令把VLAN 数据删除。

【提示】新的IOS 版本中，可以在全局配置模式中创建VLAN，如下：

```
SW1(config)#vlan 2
SW1(config-vlan)#name VLAN2
SW1(config-vlan)#exit
SW1(config)#vlan 3
SW1(config-vlan)#name VLAN3
```

（3） 步骤 3：把端口划分在 VLAN 中

```
SW1(config)# interface fastEthernet 0/1
SW1(config-if)#switchport mode access
\\将接口配置为接入模式。
SW1(config-if)#switchport access vlan 2
\\将接口划入指定的 vlan2 中。
SW1(config-if)#interface fastEthernet 0/2
SW1(config-if)#switchport mode access
SW1(config-if)#switchport access vlan 3
```

R1 做如下配置：

```
R1(config-if)#interface fastEthernet 0/0
R1(config-if)#ip address 123.1.1.1 255.255.255.0
r1(config-if)#no shutdown
```

R2 做如下配置：

```
R2(config-if)# interface fastEthernet 0/0
R2(config-if)#ip address 123.1.1.2 255.255.255.0
R2(config-if)#no shutdown
```

（2）监测和测试：

查看 vlan 信息。

```
SW1#show vlan-switch brief
```

\\在路由器 3640 或者路由器 3725 上的交换模块上用这个命令，如果是在交换机 3550 或者交换机 3560 上，采用的命令 show vlan brief

```
VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/0, Fa0/3, Fa0/4, Fa0/5
                                                Fa0/6, Fa0/7, Fa0/8, Fa0/9
                                                Fa0/10, Fa0/11, Fa0/12, Fa0/13
                                                Fa0/14, Fa0/15
2    VLAN0002                         active    Fa0/1
3    VLAN0003                         active    Fa0/2
1002 fddi-default                     active
1003 token-ring-default               active
1004 fddinet-default                  active
1005et-default                        active
```

// Valn1、vlan2 已经处于活跃状态，且 f0/1、f0/2 已经非别属于 vlan2、vlan3 中。

测试 R1 和 R2 间的连通性。发现并不能通，原因是他们并不在一个 VLAN，用上面学到的方法把 SW1 上的 Fa0/2 口划到 VLAN 2 里面，R1 和 R2 就能相互 PING 通。

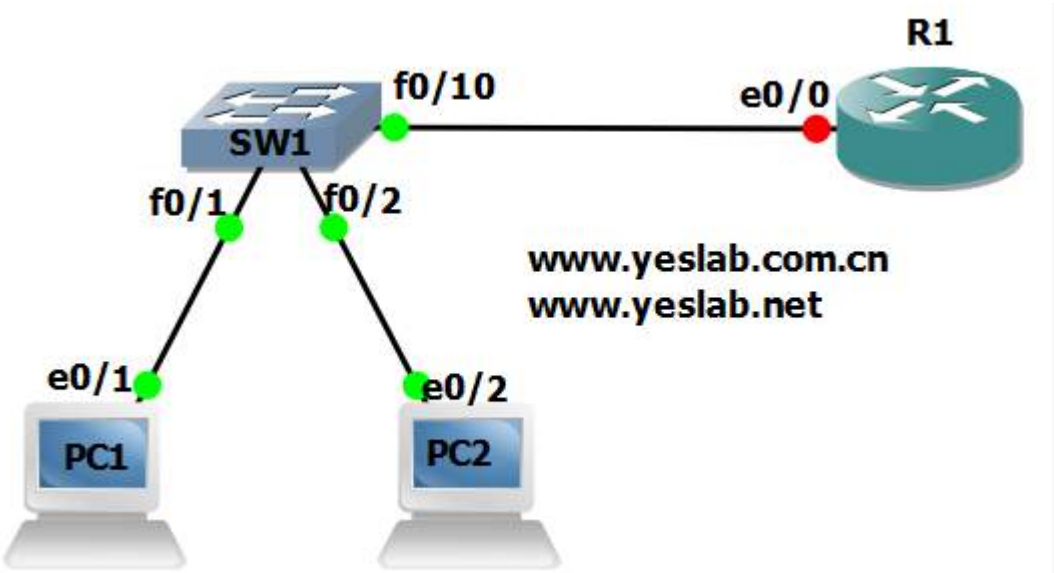
实验 6.2　VLAN 间路由

处于不同VLAN 的计算机即使它们是在同一交换机上，它们之间的通信也必须使用路由器。可以在每个VLAN 上都有一个以太网口和路由器连接。采用这种方法，如果要实现N个VLAN 间的通信，则路由器需要N 个以太网接口，同时也会占用了N个交换上的以太网接口。单臂路由提供另外一种解决方案。路由器只需要一个以太网接口和交换机连接，交换机的这个接口设置为Trunk 接口。在路由器上创建多个子接口，每个子接口封装在不同的VLAN，子接口是路由器物理接口上的逻辑接口。工作过程如下面的拓扑所示，PC1和PC2通信。PC1发送数据帧，当交换机收到VLAN10的计算机发送的数据帧后，通过Trunk接口发送数据给路由器，由于该链路是Trunk 链路，数据帧中带有VLAN10的标签，该数据帧到了路由器后，路由器去掉数据帧的VLAN10标签，然后查找路由表，发现数据需要要转发到VLAN20上，重新用VLAN20的标签进行封装，通过Trunk 链路发送到交换机上的Trunk 接口；交换机收到该数据帧，去掉VLAN20标签，发送给VLAN20上的计算机PC2，从而实现了VLAN 间的通信。

实验目的

掌握单臂路由的原理。
掌握单臂路由的配置。
掌握 trunk 的配置

实验拓扑



实验步骤

注意，此实验中的 SW1 像上面一样，可以用路由器模拟。PC1 与 PC2 可以用路由器模拟。祥见实验 1.9

配置实例：

先在路由器 R1 上

```
router(config)#hostname R1
R1(config)#interface ethernet0/0
R1(config-if)#no ip address
```

//以上是表示物理接口不需要配置 ip 地址

```
R1(config-if)#no shutdown
R1(config-if)# exit
R1(config)#interface ethernet0/0.10
```

//以上是进入子接口、配置子接口 f0/0.10

```
R1(config-if)#encapsulation dot1Q 10
```

//以上是配置子接口封装为 dot1q，要和 trunk 的封装类型统一。并指明该子接口承载 vlan10 的流量。

```
R1(config-if)#ip address 10.1.1.1 255.255.255.0
```

//以上是配置子接口的 IP 地址，子接口的 IP 地址就是 PC 的默认网关地址。

```
R1(config-if)#exit
R1(config)#interface ethernet0/0.20
R1(config-if)#encapsulation dot1Q 20
R1(config-if)#ip address 20.1.1.1 255.255.255.0
```

再在 SW1 上做配置：

先配置交换机与路由器之前的链路为 trunk，才能同时通过 VLAN10 和 VLAN20 的流量。

```
SW1(config)#interface fastEthernet 0/10
SW1(config-if)#switchport trunk encapsulation dot1q
```

```
SW1(config-if)#switchport mode trunk
```
然后配置 VLAN
```
SW1#vlan database
SW1(vlan)#vlan 10
SW1(vlan)#vlan 20
SW1(vlan)#apply
SW1(vlan)#exit
```
//再把相拓扑中的接口划入相对应的 VLAN
```
SW1(config)# interface fastEthernet 0/1
SW1(config-if)#switch mode access
SW1(config-if)#switch access vlan 10
SW1(config-if)#no shutdown
SW1(config-if)#exit
SW1(config)# interface fastEthernet 0/2
SW1(config-if)#switch mode access
SW1(config-if)#switch access vlan 20
SW1(config-if)#no shutdown
SW1(config-if)#exit
```

PC1：
```
Router(config)#hostname PC1
PC1(config)#no ip routing
PC1(config)#interface ethernet0/1
PC1(config-if)#ip address 10.1.1.2 255.255.255.0
PC1(config)#ip default-gateway 10.1.1.1
```
PC2:
```
router(config)#hostname PC2
PC2(config)#no ip routing
PC2(config)#interface ethernet0/2
PC2(config-if)#ip address 20.1.1.2 255.255.255.0
PC2(config)#ip default-gateway 20.1.1.1
```

（1） 监测和测试
```
PC1#ping 20.1.1.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 20.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
```

实验 6.3 生成树

为了减少网络的故障时间，我们经常会采用冗余拓扑。STP 可以让具有冗余结构的网络在故障时自动调整网络的数据转发路径。STP 重新收敛时间较长，通常需要30—50 秒，为了减少这个时间，引入了一些补充技术，例如uplinkfast、backbonefast 等。RSTP 则在协议上对STP作了改进，缩短了故障收敛时间。STP还有PVST、MST等改进型协议，以及一些保护STP树的安全feature，本章将介绍这些常用的配置。

简介：

基本STP

为了增加局域网的冗余性，我们常常会在网络中引入冗余链路，然而这样却会引起交换环路。交换环路会带来三个问题：广播风暴、同一帧的多个拷贝、交换机CAM 表不稳定。STP(Spanning Tree Protocol)可以解决这些问题，STP 基本思路是阻断一些冗余的交换机接口，构建一棵没有环路的转发树。STP 利用BPDU(Bridge Protocol Data Unit)和其他交换机进行通信，从而确定哪个交换机该阻断哪个接口。在BPDU 中有几个关键的字段，例如：根桥ID、路径代价、端口ID 等。为了在网络中形成一个没有环路的拓扑，交换网络中的交换机要进行以下三个步骤：（1）选举根桥、（2）每个交换机选取一个根端口、（3）每个网段选取一个指定端口。剩下的其它的接口都将被阻断，不转发数据包。这样网络就构建出一棵没有环路的二层转发树。

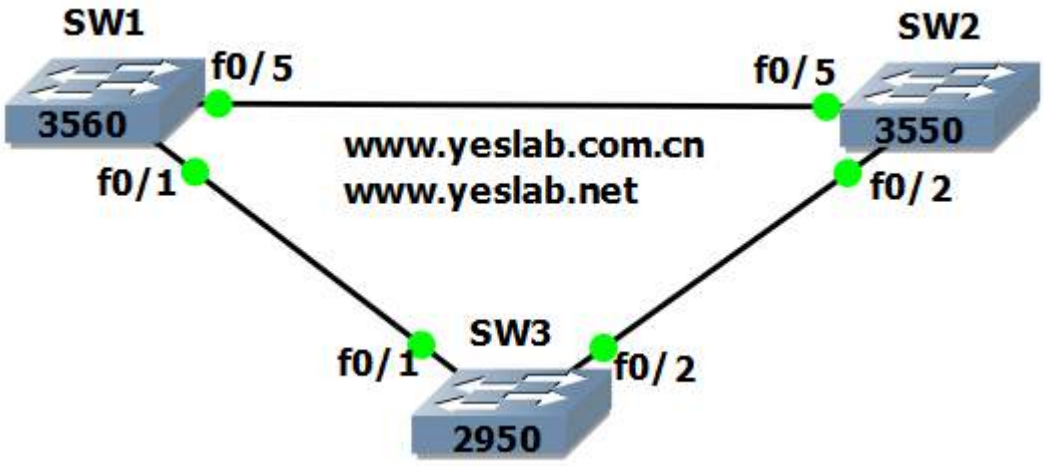
PVST

当网络上有多个VLAN 时，PVST(Per Vlan STP)会为每个VLAN 构建一棵STP树。此时端口的阻断基于VLAN，因此可以让一个端口为某些VLAN工作在forwarding状态，为某些VLAN工作在blocking状态，从而实现负载平衡。缺点是如果VLAN 数量很多，会给交换机带来沉重的负担。Cisco 交换机默认的模式就是PVST+。

实验目的

1）理解 STP 的工作原理；
2）掌握 STP 树的控制；
3）利用 PVST 进行负载均衡。
4）理解 trunk 的配置

实验拓扑



实验步骤

（1）配置三条链路为 trunk、在 sw1 上创建 vlan2。配置 VTP 协议让另二台交换机都学习到 VLAN2

步骤 1：

先手动配置三条链路为 trunk

```
SW1(config)#interface range fastEthernet 0/1 , fastEthernet 0/5
//以上是同时对二个接口进行配置
SW1(config-if-range)# switchport trunk encapsulation dot1q
//以上是配置 trunk 的封装类型
SW1(config-if-range)#switchport mode trunk
//配置一个接口为 trunk 模式
SW1(config-if-range)#exit
```

切换到 sw2 上做配置

```
SW2(config)#interface range fastEthernet 0/2 , fastEthernet 0/5
SW2(config-if-range)# switchport trunk encapsulation dot1q
SW2(config-if-range)#switchport mode trunk
SW2(config-if-range)#exit
```

切换到 sw3 上做配置

```
SW3(config)#interface range fastEthernet 0/1 , fastEthernet 0/2
SW3(config-if-range)# switchport trunk encapsulation dot1q
SW3(config-if-range)#switchport mode trunk
W3(config-if-range)#exit
```

配置步骤 2：增加 VLAN2 并在三台交换机上配置 VTP 协议

```
SW1#vlan database
SW1(vlan)#vlan 2
SW1(vlan)#vtp server
//以上是指定 VTP 的模式为服务器模式
SW1(vlan)#vtp password cisco
//以上是配置 VTP 的密码
SW1(vlan)#vtp domain yeslab
//以上是配置 VTP 的域名
```

```
SW1(vlan)#apply
SW1(vlan)#exit
```

SW1 配置完毕，然后在 SW2 上做如下配置：

```
SW2#vlan database
SW2(vlan)#vtp client
SW2(vlan)#vtp domain yeslab
SW2(vlan)#vtp password cisco
SW2(vlan)#apply
SW2(vlan)#exit
```

接着在 SW3 上做配置：

```
SW3#vlan database
SW3(vlan)#vtp client
SW3(vlan)#vtp domain yeslab
SW3(vlan)#vtp password cisco
SW3(vlan)#apply
SW3(vlan)#exit
```

说明：完成上面二大步骤的配置后，在另两台交换上应能看到 vlan2 和 vtp 的信息。请确认是否成功（此步骤不在本实验中演示）。

步骤 2：查看初始的 stp 状态。

```
SW1#show spanning-tree
VLAN0001
  Spanning tree enabled protocol ieee
  Root ID     Priority    32769
              Address     000b.5f85.1380
              This bridge is the root
```

//priority:根桥的优先级，默认情况下所有的网桥的优先级都是 32768，此处之所以显示了 32769 是因为加上了 vlan1 的 VLAN ID。如果是 vlan2 则加上 2，为 32770。

//Address: 根桥的 MAC 地址。

// This bridge is the root 表明该设备是 vlan1 是跟桥。

```
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
  Bridge ID   Priority    32769  (priority 32768 sys-id-ext 1)
              Address     000b.5f85.1380
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time 300
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- --------
--------------------------------
Fa0/1               Desg FWD 19        128.1    P2p
Fa0/5               Desg FWD 19        128.5    P2p
VLAN0002
  Spanning tree enabled protocol ieee
  Root ID     Priority    32770
```

```
              Address     000b.5f85.1380
              This bridge is the root
//通过查看表明 SW1 是 vlan1 和 vlan2 的跟桥。
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
  Bridge ID   Priority    32770  (priority 32768 sys-id-ext 2)
              Address     000b.5f85.1380
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time 300
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- --------
--------------------------------
Fa0/1               Desg FWD 19        128.1    P2p
Fa0/5               Desg FWD 19        128.5    P2p
```

步骤 3：控制 sw1 是 vlan1 的根桥、sw2 是 vlan2 的根桥。
SW1 是 vlan2 的备份根 SW2 是 vlan1 的备份根

```
SW1(config)#spanning-tree vlan 1 priority 4096
//以上是把 SW3560 在 vlan1 的优先级改成 4096 使之成为 vlan1 的根桥。
//也可以用命令 spanning-tree vlan 1 root primary 来实现，
SW1(config)#spanning-tree vlan 2 priority 8192
//以上是把 SW1 配置起 VLAN2 的备份根桥
也可以用命令 spanning-tree vlan 1 root secondary 来实现，
SW2(config)#spanning-tree vlan 2 priority 4096
//以上是把 SW3550 在 vlan1 的优先级改成 4096 使之成为 vlan2 的根桥。
//注意：此处是用改变优先级的方式来改变根桥属性。
//也可以用使用命令 spanning-tree vlan 2 root primary 来实现
SW2(config)#spanning-tree vlan 1 priority 8192
//以上是把 SW2 配置起 VLAN1 的备份根桥
也可以用命令 spanning-tree vlan 1 root secondary 来实现
```

查看调试后生成树的状态、验证配置
注意由于设备的不同，显示的细节可能不一致。

```
SW1#show spanning-tree
VLAN0001
  Spanning tree enabled protocol ieee
  Root ID     Priority    4097
              Address     000b.5f85.1380
              This bridge is the root
//以上表明该 SW1 在 vlan1 中是根桥。
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
  Bridge ID   Priority    4097   (priority 4096 sys-id-ext 1)
              Address     000b.5f85.1380
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time 300
```

```
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- --------
--------------------------------
Fa0/1               Desg FWD 19        128.1    P2p
Fa0/5               Desg FWD 19        128.5    P2p
//跟桥上的所有端口都是指定端口。
VLAN0002
  Spanning tree enabled protocol ieee
  Root ID     Priority    4098
              Address     000d.28fc.9b80
              Cost        19
              Port        5 (FastEthernet0/5)
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
  Bridge ID   Priority    8194  (priority 8192 sys-id-ext 2)
              Address     000b.5f85.1380
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time 300
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- --------
--------------------------------
Fa0/1               Desg FWD 19        128.1    P2p
Fa0/5               Root FWD 19        128.5    P2p
```



```
SW2#show spanning-tree
VLAN0001
  Spanning tree enabled protocol ieee
  Root ID     Priority    4097
              Address     000b.5f85.1380
              Cost        19
              Port        5 (FastEthernet0/5)
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    8194  (priority 8192 sys-id-ext 1)
              Address     000d.28fc.9b80
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time 300
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- --------
--------------------------------
Fa0/2               Desg FWD 19        128.2    P2p
Fa0/5               Root FWD 19        128.5    P2p
VLAN0002
  Spanning tree enabled protocol ieee
  Root ID     Priority    4098
```

```
              Address      000d.28fc.9b80
              This bridge is the root
//此处表明在 vlan2 中 SW2 是根桥。
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
  Bridge ID   Priority     4098   (priority 4096 sys-id-ext 2)
              Address      000d.28fc.9b80
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time 300
Interface           Role Sts Cost      Prio.Nbr Type
------------------- ---- --- --------- --------
--------------------------------
Fa0/2               Desg FWD 19        128.2    P2p
Fa0/5               Desg FWD 19        128.5    P2p
```

说明：以上表明 SW1 是 vlan1 的根桥、SW 是 vlan2 的根桥。即：通过控制根桥的属性实现了每个 vlan1 和 vlan2 有不同的根网桥，能够对 vlan1 和 vlan2 达到负载均衡的效果。

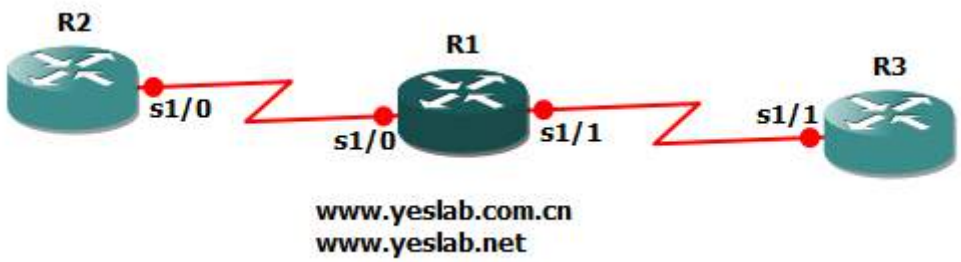
实验第七部分 NAT

实验 7.1 静态 NAT

实验目的：

1）理解 NAT 的工作原理；
2）掌握 NAT 的配置。

拓扑：



实验步骤：

说明：本配置将演示 NAT 将一个内部地址转换为全局的、唯一的外部地址。常应用于内部服务器对外提供服务，即能访问外部又能为外部所访问。

场景仿真，R2 为一个 ISP 接入路由器，客户端为 R1\R3，ISP 分配给客户端 1 个公网 IP 地址为：12.1.1.1。用户端有一台服务器希望为外网访问出口，在内网的地址为：33.1.1.1。
R1/R3 之间运行 OSPF 协议。通过默认路由访问外网。

步骤 1：配置路由协议保证 R1/R3 之间的能够互通信和路由的同步。

在 R1 上：

```
R1(config)#interface serial 1/1
R1(config-if)#ip address 13.1.1.1 255.255.255.0
R1(config-if)#no shutdown
R1(config)#interface serial 1/0
R1(config-if)#ip addressa 12.1.1.1 255.255.255.0
R1(config-if)#no shutdown
R1(config)#router ospf 1
R1(config-router)#router-id 11.1.1.1
R1(config-router)#network 13.1.1.0 0.0.0.255 area 0
R1(config-router)#default-information originate
R1(config)#ip route 0.0.0.0 0.0.0.0 12.1.1.2

R3 上：
R3(config)#interface Loopback0
R3(config-if)#ip address 33.1.1.1 255.255.255.0
R3(config-if)#ip ospf network point-to-point
R3(config)#interface Serial1/1
R3(config-if)#ip address 13.1.1.3 255.255.255.0
R3(config-if)#no shutdown
R3(config)#router ospf 1
R3(config-router)#router-id 33.1.1.1
R3(config-router)#network 33.1.1.0 0.0.0.255 area 0
R3(config-router)#network 13.1.1.0 0.0.0.255 area 0

R2 上：
R2(config)#interface serial 1/0
R2(config-if)#ip address 12.1.1.2 255.255.255.0
R2(config-if)#no shutdown
R2(config-if)#exit
R2(config)#interface loopback 0
R2(config-if)#ip address 22.1.1.1 255.255.255.0
R2(config-if)#exit
R2(config)#ip route 0.0.0.0 0.0.0.0 12.1.1.1
//写一条默认路由指向 R1，让数据包能够回去。
```


步骤 2：配置路由器 R1 提供 NAT 服务

```
R1(config)#ip nat inside source static 33.1.1.1 12.1.1.1
//配置静态 NAT 映射
R1(config)#interface Serial1/1
```

```
R1(config-if)#ip nat inside
//配置NAT 内部接口
R1(config)#interface Serial1/0
R1(config-if)#ip nat outside
//配置 NAT 外部接口
```

步骤 3：监测和测试

R1#debug ip nat

//在 R1 上打开 NAT 的 debug 调试开关

R3# ping 22.1.1.1 source 33.1.1.1

//然后在 R3 上加源（IP 地址：33.1.1.1）ping 外网地址（R2 的 loopback0：22.1.1.1）验证 NAT 是否已经成功转换地址。

```
r3#ping 22.1.1.1 source 33.1.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 22.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 33.1.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 36/64/108 ms
```

在 R1 上观察：

```
IP NAT debugging is on
May  1 23:28:55.663: NAT*: s=33.1.1.1->12.1.1.1, d=22.1.1.1 [43]
May  1 23:28:55.663: NAT*: s=22.1.1.1, d=12.1.1.1->33.1.1.1 [43]
May  1 23:28:55.815: NAT*: s=33.1.1.1->12.1.1.1, d=22.1.1.1 [44]
May  1 23:28:55.839: NAT*: s=22.1.1.1, d=12.1.1.1->33.1.1.1 [44]
May  1 23:28:55.871: NAT*: s=33.1.1.1->12.1.1.1, d=22.1.1.1 [45]
May  1 23:28:55.899: NAT*: s=22.1.1.1, d=12.1.1.1->33.1.1.1 [45]
May  1 23:28:55.911: NAT*: s=33.1.1.1->12.1.1.1, d=22.1.1.1 [46]
May  1 23:28:55.931: NAT*: s=22.1.1.1, d=12.1.1.1->33.1.1.1 [46]
May  1 23:28:55.951: NAT*: s=33.1.1.1->12.1.1.1, d=22.1.1.1 [47]
May  1 23:28:55.975: NAT*: s=22.1.1.1, d=12.1.1.1->33.1.1.1 [47]
```

//以上表示 NAT 把内网地址 33.1.1.1 转换为了外网地址 12.1.1.1

再测试从外网作为源与内网的通信情况：

```
R2#ping 12.1.1.1 source 22.1.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 12.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 22.1.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 48/84/120 ms
```

在 R1 上观察：

```
May  1 23:45:56.431: NAT*: s=22.1.1.1, d=12.1.1.1->33.1.1.1 [30]
May  1 23:45:56.447: NAT*: s=33.1.1.1->12.1.1.1, d=22.1.1.1 [30]
May  1 23:45:56.591: NAT*: s=22.1.1.1, d=12.1.1.1->33.1.1.1 [31]
May  1 23:45:56.611: NAT*: s=33.1.1.1->12.1.1.1, d=22.1.1.1 [31]
May  1 23:45:56.623: NAT*: s=22.1.1.1, d=12.1.1.1->33.1.1.1 [32]
May  1 23:45:56.659: NAT*: s=33.1.1.1->12.1.1.1, d=22.1.1.1 [32]
```

```
May  1 23:45:56.723: NAT*: s=22.1.1.1, d=12.1.1.1->33.1.1.1 [33]
May  1 23:45:56.759: NAT*: s=33.1.1.1->12.1.1.1, d=22.1.1.1 [33]
May  1 23:45:56.791: NAT*: s=22.1.1.1, d=12.1.1.1->33.1.1.1 [34]
May  1 23:45:56.807: NAT*: s=33.1.1.1->12.1.1.1, d=22.1.1.1 [34]
//以上表示 NAT 把地址 12.1.1.1 这个外网地址转换为了 33.1.1.1 内网地址
```

```
R1#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
--- 12.1.1.1           33.1.1.1           ---                ---
```

静态映射时，NAT 表一直存在。

【术语】

① 内部局部（inside local）地址：在内部网络使用的地址，往往是RFC1918 地址；
② 内部全局（inside global）地址：用来代替一个或多个本地IP地址的、对外的、向NIC 注册过的地址；
③ 外部局部（outside local）地址：一个外部主机相对于内部网络所用的IP地址，不一定是合法的地址；
④ 外部全局（outside global）地址：外部网络主机的合法 IP 地址。

不要重起路由器，接下来的实验可在此基础上进行。把静态 NAT 的配置 NO 掉即可：

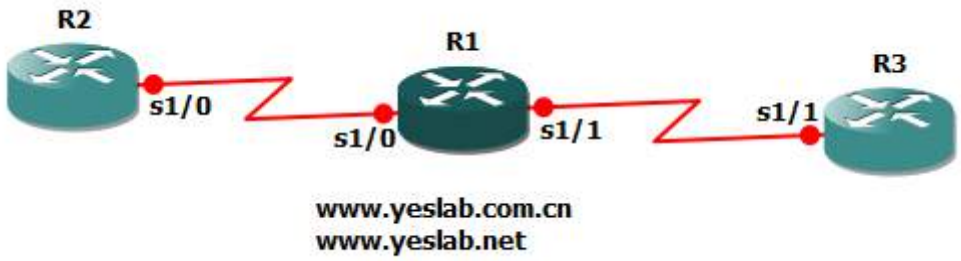
```
R1(config)#no ip nat inside source static 33.1.1.1 12.1.1.1
```

实验 7.2 动态 NAT

实验目的：

1）理解动态 NAT 的概念
2）掌握动态 NAT 的配置

拓扑：



实验步骤：

说明：动态 NAT 地址转换本质上与静态是一致的，只是先定义一个转换地址池，当 PC 有一个向外的连接请求时，从地址池中取出一个 IP 地址。当连接断开时将把取出的 IP 重新放入池中，以供其他 PC 向外连接时使用。动态转换的效率非常高，因为一个公网 IP 可以让不同的站点使用多次。这比静态只能让一个特定站点使用更高效。

注意：虽然动态地址转换是高效且利于管理，但外部用户不能访问内部特定的地址。因为他们之间没有静态映射。当每个会话结束后，主机再次发起连接，很可能分配不同于上次的本地全局地址。所以不可能用一个本地全局地址访问内部特定的地址。

以下实验仿真客户从 ISP 处申请了 10 个公网 IP（12.1.1.1-10/24）

步骤 1：配置路由协议保证 R1/R3 之间的能够互通信和路由的同步，这里不再重复，如有不明白请参考上一个实验。

步骤 2：配置路由器 R1 提供 NAT 服务

```
R1(config)#ip nat pool YESLAB 12.1.1.1 12.1.1.10 netmask 255.255.255.0
//配置动态 NAT 转换的地址池，起始和结束地址范围。
R1(config)#access-list 1 permit 33.1.1.0 0.0.0.255
//允许动态 NAT 转换的内部地址范围
R1(config)#ip nat inside source list 1 pool YESLAB
//配置动态 NAT 映射
R1(config)#interface Serial1/1
R1(config-if)#ip nat inside
R1(config-if)#interface Serial1/0
R1(config-if)#ip nat outside
```

4：监测和测试

```
先在 R3 上起多个 IP 地址：
r3(config-if)#ip address 33.1.1.2 255.255.255.0 secondary
r3(config-if)#ip address 33.1.1.3 255.255.255.0 secondary
r3(config-if)#ip address 33.1.1.4 255.255.255.0 secondary
r3(config-if)#ip address 33.1.1.5 255.255.255.0 secondary
然后分别用这些地址去 ping 外网
R3# ping 22.1.1.1 source 33.1.1.1
R3# ping 22.1.1.1 source 33.1.1.2
R3# ping 22.1.1.1 source 33.1.1.3
R3# ping 22.1.1.1 source 33.1.1.4
R3# ping 22.1.1.1 source 33.1.1.5
到 R1 去看 NAT 转换表项：
R1#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
icmp 12.1.1.1:7        33.1.1.1:7         22.1.1.1:7         22.1.1.1:7
--- 12.1.1.1           33.1.1.1           ---                ---
--- 12.1.1.2           33.1.1.2           ---                ---
--- 12.1.1.3           33.1.1.3           ---                ---
--- 12.1.1.4           33.1.1.4           ---                ---
--- 12.1.1.5           33.1.1.5           ---                ---
```


（1） 演示如何查看动态 NAT 的有效时间

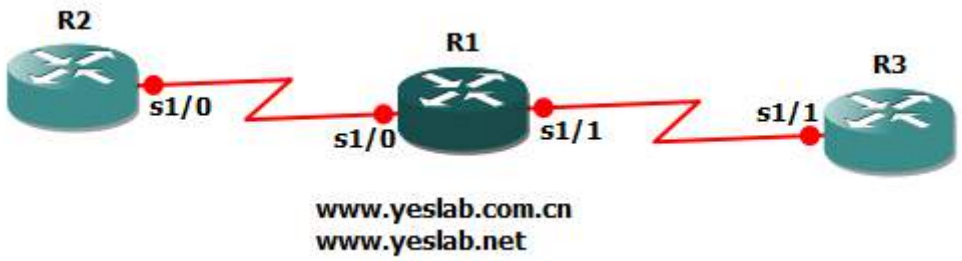
```
R1#show ip nat translations verbose
Pro Inside global      Inside local       Outside local      Outside global
--- 12.1.1.1           33.1.1.1           ---                ---
    create 00:08:50, use 00:07:00 timeout:86400000, left 23:52:59, Map-Id(In): 1,
//以上显示会话建立时间为 8：50，使用计时为 7：00，超时时间为 23：52：59
```

实验 7.3 复用内部全局地址的 NAT（PAT）

实验目的：

1）理解复用全局地址 NAT 的工作原理；
2）掌握复用全局地址的 NAT 的配置。

拓扑：



以上实验都是一对一的映射,CISCO 支持复用全局地址，因此避免了本地地址和全局地址的一对一映射的需求。复用全局地址是 NAT 应用里最常见的一种用法。如：局域网中共享上网。
说明：当多个本地地址映射到同一个全局地址时候，用端口号来区别不同的本地地址。
执行 NAT 步骤：
1）内部主机 A 访问 Internet 站点 Web-server
2）路由器收到主机 A 的第一个报文，检查 NAT 表
3）如果不存在转换，则路由器用全局地址 B 代替源地址 A，并建立 NAT 表项，NAT 表项包含内部全局地址和端口号、外部全局地址和端口号、内部本地地址和端口号、外部本地地址和端口号。
4）当路由器收到从 Web-server 返回的报文后，路由器利用 NAT 表，路由器将目的地址为全局地址 B，转换为内部地址 A，然后将报文转发到主机。

实验步骤：

步骤 1：配置路由协议保证 R1/R3 之间的能够互通信和路由的同步，这里不再重复，如有不明白请参考上一个实验。

步骤 2：配置路由器 R1 提供 PAT 服务。

在 R1 上：

```
R1(config)#ip nat pool YESLAB 12.1.1.1 12.1.1.1 netmask 255.255.255.0
//配置动态 NAT 转换的地址池，现在这个池里只有一个地址。
R1(config)#access-list 1 permit 33.1.1.0 0.0.0.255
//允许动态 NAT 转换的内部地址范围
R1(config)#ip nat inside source list 1 pool YESLAB overload
//配置动态 NAT 映射，事实上 PAT 与动态 NAT 的配置区别就是多了 overload 关键字
R1(config)#interface Serial1/1
R1(config-if)#ip nat inside
R1(config-if)#interface Serial1/0
R1(config-if)#ip nat outside
```

监测和测试：

在 R3 在上分别用 33.1.1.1—33.1.1.5 作为源地址 ping 22.1.1.1

```
R3#ping 22.1.1.1 source 33.1.1.1
R3#ping 22.1.1.1 source 33.1.1.2
R3#ping 22.1.1.1 source 33.1.1.3
R3#ping 22.1.1.1 source 33.1.1.4
R3#ping 22.1.1.1 source 33.1.1.5
到 R1 上去看 NAT 转换表项：
R1#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
icmp 12.1.1.1:26       33.1.1.1:26        22.1.1.1:26        22.1.1.1:26
icmp 12.1.1.1:27       33.1.1.2:27        22.1.1.1:27        22.1.1.1:27
icmp 12.1.1.1:28       33.1.1.3:28        22.1.1.1:28        22.1.1.1:28
icmp 12.1.1.1:29       33.1.1.4:29        22.1.1.1:29        22.1.1.1:29
icmp 12.1.1.1:30       33.1.1.5:30        22.1.1.1:30        22.1.1.1:30
//12.1.1.1 被转换多次，在地址的后面跟上了端口号来区分不同的内部地址
```

实验第八部分：PPP 与 HDLC

路由器经常用于构建广域网，广域网链路的封装和以太网上的封装有着非常大的差别。常见的广域网封装有HDLC、PPP、Frame-relay 等，本章介绍HDLC 和PPP。相对而言，PPP比起HDLC 有较多的功能。

HDLC 和PPP 简介

HDLC 介绍

HDLC 是点到点串行线路上（同步电路）的帧封装格式，其帧格式和以太网帧格式有很大的差别，HDLC 帧没有源MAC 地址和目的MAC 地址。Cisco 公司对HDLC 进行了专有化，Cisco的HDLC 封装和标准的HDLC 不兼容。如果链路的两端都是Cisco 设备，使用HDLC 封装没有问题，但如果Cisco 设备与非Cisco 设备进行连接，应使用PPP 协议。HDLC 不能提供验证，缺少了对链路的安全保护。默认时，Cisco 路由器的串口是采用Cisco HDLC 封装的。如果串口的封装不是HDLC，要把封装改为HDLC 使用命令“encapsulation hdlc”。

PPP 介绍

1. PPP 概述

和HDLC 一样，PPP 也是串行线路上（同步电路或者异步电路）的一种帧封装格式，但是PPP 可以提供对多种网络层协议的支持。PPP 支持认证、多链路捆绑、回拨、压缩等功能。PPP 经过4 个过程在一个点到点的链路上建立通信连接：

· 链路的建立和配置协调：通信的发起方发送LCP 帧来配置和检测数据链路· 链路质量检测：在链路已经建立、协调之后进行，这一阶段是可选的· 网络层协议配置协调：通信的发起方发送NCP 帧以选择并

配置网络层协议·关闭链路：通信链路将一直保持到LCP 或NCP 帧关闭链路或发生一些外部事件

2. PPP 认证：PAP 和CHAP

（1）PAP——密码验证协议

PAP（Password Authentication Protocol）利用2 次握手的简单方法进行认证。在PPP 链路建立完毕后，源节点不停地在链路上反复发送用户名和密码，直到验证通过。PAP的验证中，密码在链路上是以明文传输的，而且由于是源节点控制验证重试频率和次数，因此PAP 不能防范再生攻击和重复的尝试攻击。

（2）CHAP——询问握手验证协议

CHAP(Challenge Handshake Authentication Protocol)利用3 次握手周期地验证源端节点的身份。CHAP 验证过程在链路建立之后进行，而且在以后的任何时候都可以再次进行。这使得链路更为安全；CHAP 不允许连接发起方在没有收到询问消息的情况下进行验证尝试。CHAP 每次使用不同的询问消息，每个消息都是不可预测的唯一的值，CHAP 不直接传送密码，只传送一个不可预测的询问消息，以及该询问消息与密码经过MD5 加密运算后的加密值。所以CHAP 可以防止再生攻击，CHAP 的安全性比PAP 要高。

实验 8.1 HDLC 与 PPP 的封装

实验目标

通过本实验，读者可以掌握如下技能：

（1） 串行链路上的封装概念

（2） HDLC 封装

（3） PPP 封装

实验拓扑

实验步骤

先按照拓扑配置好 IP 地址，保证直连能通。

在 R1 上：

```
R1(config)#interface serial 0/0
R1(config-if)#ip address 12.1.1.1 255.255.255.0
R1(config-if)#no shutdown
```

在 R2 上：

```
R2(config)#interface serial 0/0
R2(config-if)#ip address 12.1.1.2 255.255.255.0
R2(config-if)#no shutdown
```

测试直连是否通：
在 R1 上：

```
R1#ping 12.1.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 12.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/28/68 ms
```

我们去查看接口的封装类型：
在 R1 上：

```
R1#show interfaces serial 0/0
Serial0/0 is up, line protocol is up
  Hardware is M4T
  Internet address is 12.1.1.1/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, crc 16, loopback not set
```

//以上显示表明此接口的封装类型为 HDLC，这也是 CISCO 在串口上的默认封装方式。
————以下部分省略————

我们去修改此接口的封装类型为 PPP
在 R1 上：

```
R1(config)#interface serial 0/0
R1(config-if)#encapsulation ppp
```

//进入到接口下打以上命令，改变了此接口的封装类型为 PPP。

```
*Mar  1 00:15:30.819: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed state to down
```

//以上提示是说我们接口的协议层 down 掉了，想一想原因是什么？
是线路的双端封装类型不一致。这时我们要到对方去改变接口的封装也为 PPP.

在 R2 上：

```
R2(config)#interface serial 0/0
R2(config-if)#encapsulation ppp
*Mar  1 00:19:47.355: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed state to up
```

//双端都改成 PPP 以后，线路就起来了。

【注意】用命令show interfaces serial 0/0显示串行接口时，常见以下几种状态：
Serial0/0 is up, line protocol is up
//链路正常

```
Serial0/0 is administratively down, line protocol is down
//没有打开该接口，执行“no shutdown”可以打开接口
Serial0/0 is up, line protocol is down
//物理层正常，数据链路层有问题，通常是没有配置时钟、两端封装不匹配、PPP 认证错误
Serial0/0 is down, line protocol is down
//物理层故障，通常是连线问题
```

不要关闭设备，接下来的实验可在此基础上完成

实验 8.2 PPP 的 PAP 认证

实验目标

掌握 PAP 认证的配置方法

实验拓扑

实验步骤

步骤 1:先确保 R1 与 R2 的封装类型为 PPP，直连可达。
在 R1 上：

```
R1(config)#interface serial 0/0
R1(config-if)#ip address 12.1.1.1 255.255.255.0
R1(config-if)# encapsulation ppp
R1(config-if)#no shutdown
```

在 R2 上：

```
R2(config)#interface serial 0/0
R2(config-if)#ip address 12.1.1.2 255.255.255.0
R2(config-if)# encapsulation ppp
R2(config-if)#no shutdown
```

先摸拟 R2 为认证方，R1 为被认证方。

步骤 2:开启 R2 的认证功能。
在 R2 上：

```
R2(config)#interface serial 0/0
R2(config-if)#ppp authentication pap
*Mar  1 01:02:09.539: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed state to down
//以上表示，我们在 R2 开启 PAP 的认证后，链路就 DOWN 掉了。
再去 PING 对方，会发现不通。
```

步骤 3：在 R2 上添加用用户名和密码

```
R2(config)#username R1 password cisco
```

步骤 4:去 R1 上发送用户名和密码来通过认证。

```
R1(config)#interface serial 0/0
R1(config-if)#ppp pap sent-username R1 password cisco
*Mar  1 01:07:03.915: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial0/0, changed state to up
//配置好以后，马上就发现链路已经 UP 起来了。再去 PING 通测试。
```

现在就做好了一个单向的认证关系，即 R2 去认证 R1。我们可以实施一个双向的认证，就是说再让 R1 去认证 R2。如下：

在 R1 上：

```
R1(config)#interface serial 0/0
R1(config-if)#ppp authentication pap
R1(config)#username R2 password yeslab
//双向 PAP 认证的时候两边的密码可以不一致
```


在 R2 上：

```
R2(config)#interface serial 0/0
R2(config-if)#ppp pap sent-username R2 password yeslab
```

//以上便实验了 PAP 的双向认证。

实验 8.3 PPP 的 CHAP 认证

实验目标

掌握CHAP 认证的配置方法

实验拓扑

实验步骤

先确保链路是 PPP 封装，并且直连已经能通。不会请参考 8.1，这里不再指导。
我们先模拟由 R2 是认证方，R1 是被认证方。

在 R2 上：

```
R2(config)#interface serial 0/0
R2(config-if)#ppp authentication chap
//开启 PPP 的 CHAP 认证方式。
```

```
R2(config)#username R1 password cisco
R1(config)#username R2 password cisco
//分别配置对方的用户名和对应的密码
```


如果想要实验双向认证，在 R1 上也开启认证即可：

```
R1(config)#interface serial 0/0
R1(config-if)#ppp authentication chap
```


上面是CHAP 验证的最简单配置，也是实际应用中最常用的配置方式。配置时要求用户名为对方路由器名，而双方密码必须一致。原因是：由于CHAP 默认使用本地路由器的名字做为建立PPP 连接时的识别符。路由器在收到对方发送过来的询问消息后，将本地路由器的名字作为身份标识发送给对方；而在收到对方发过来的身份标识之后，默认使用本地验证方法，即在配置文件中寻找，看看有没有用户身份标识和密码；如果有，计算加密值，结果正确则验证通过；否则验证失败，连接无法建立。

实验第九部分：帧中继

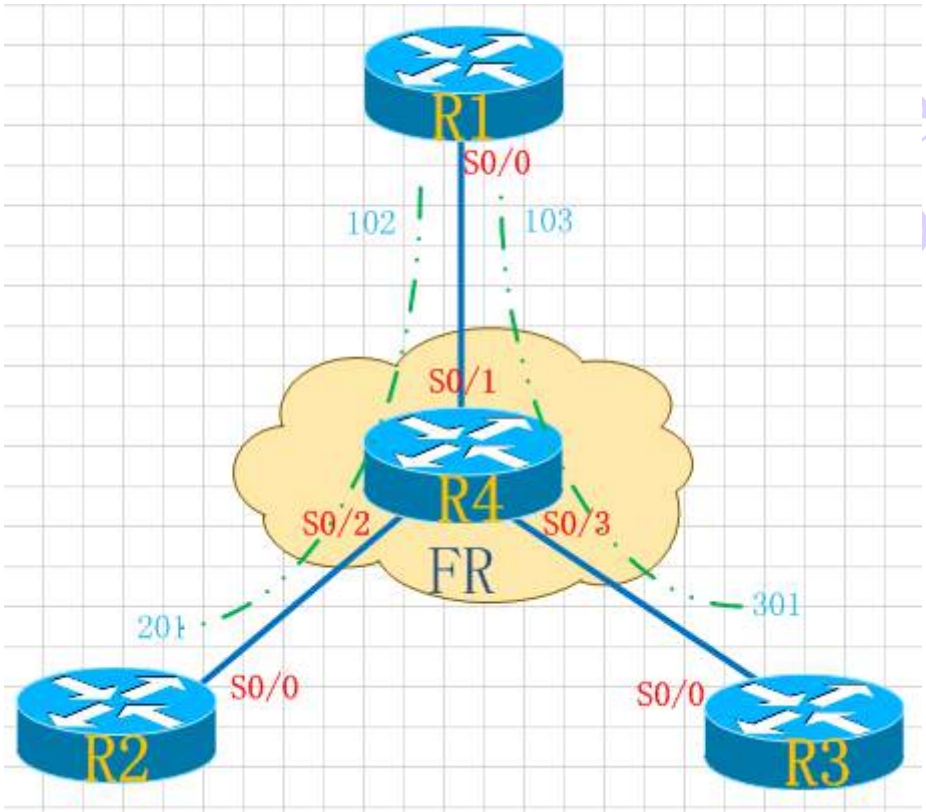
帧中继线路是中小企业常用的广域网线路，其通信费用较低。由于帧中继技术的一些特殊性使得帧中继的配置较为复杂，特别是在帧中继上运行路由协议时更是如此。作为入门，对帧中继的理解应着重放在DLCI、PVC、帧中继映射和子接口等概念上。本章通过几个实验详细介绍了帧中继的关键概念。

实验 9.1:把一台 Cisco 路由器配置为帧中继交换机

实验目标

掌握用路由器充当帧中继交换机的配置

实验拓扑



实验步骤

本实验只关心 R4 上的配置，把 R4 配置成一个帧中继的交换机。

```
R4(config)#frame-relay switching
```

```
//开启帧中继交换机功能，要注意哦，别忘记打了。
```

```
R4(config)#interface serial 0/1
R4(config-if)#encapsulation frame-relay
//在接口下改封装类型为帧中继
R4(config-if)#frame-relay lmi-type cisco
//配置 LMI 的类型为 cisco
R4(config-if)#frame-relay intf-type dce
//配置本端为 DCE 端
R4(config-if)#frame-relay route 102 interface serial 0/2 201
//配置帧中继交换表的，告诉路由器如果从该接口收到DLCI=102 的帧，要从s0/2 交换出去，并且DLCI 改为201
R4(config-if)#frame-relay route 103 interface serial 0/3 301
R4(config-if)#no shutdown
R4(config-if)#exit
```

```
R4(config)#interface serial 0/2
R4(config-if)#encapsulation frame-relay
R4(config-if)#frame-relay lmi-type cisco
R4(config-if)#frame-relay intf-type dce
R4(config-if)#frame-relay route 201 interface serial 0/1 102
R4(config-if)#exit
```



```
R4(config)#interface serial 0/3
R4(config-if)#encapsulation frame-relay
R4(config-if)#frame-relay lmi-type cisco
R4(config-if)#frame-relay intf-type dce
R4(config-if)#frame-relay route 301 interface serial 0/1 103
R4(config-if)#no shutdown
```

用以下命令查看配置结果：

```
R4#show frame-relay route
Input Intf        Input Dlci      Output Intf      Output Dlci      Status
Serial0/1         102             Serial0/2        201              inactive
Serial0/1         103             Serial0/3        301              inactive
Serial0/2         201             Serial0/1        102              inactive
Serial0/3         301             Serial0/1        103              inactive
```

配好以后接着完成下面的实验。

实验 9.2：帧中继基本配置、帧中继映射

实验目的

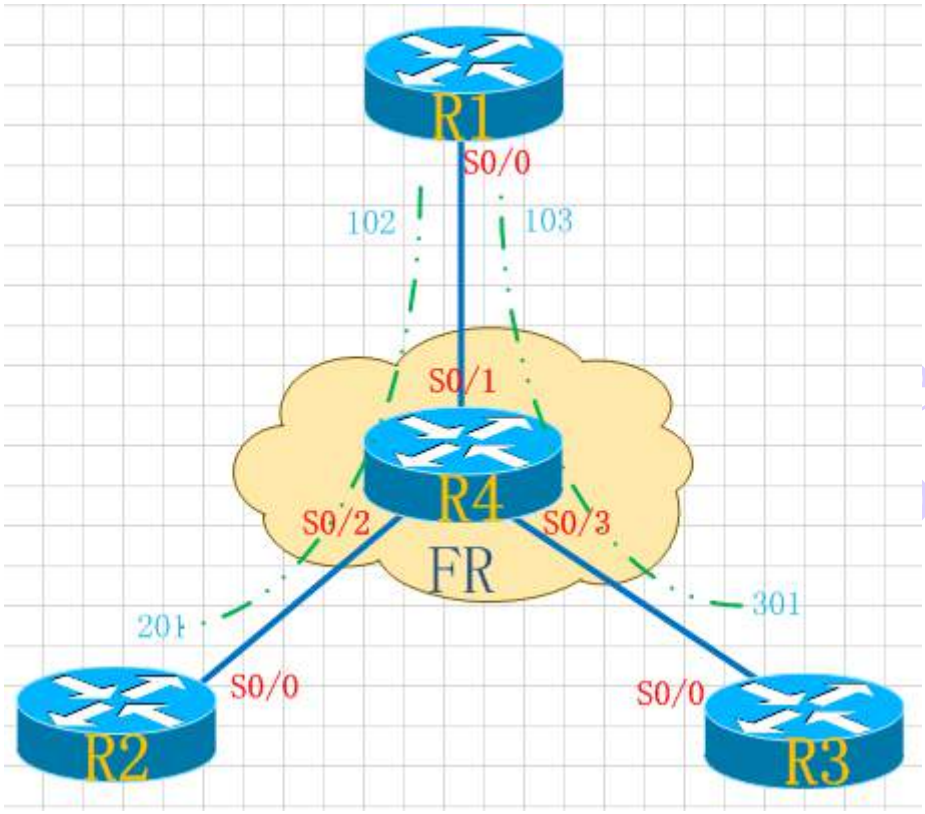
通过本实验，你可以掌握如下技能：

（1） 帧中继的基本配置

（2） 帧中继的动态映射

（3） 帧中继的静态映射

实验拓扑



实验步骤

此实验要在 9.1 的基础之上完成。请先完成 9.1

9.1已完成了帧中继的配置，现配置R1、R3、R4，使得它们能够互相通信，配置步骤如下：

在R1上：

```
R1(config)#interface serial 0/0
R1(config-if)#ip address 123.1.1.1 255.255.255.0
R1(config-if)#encapsulation frame-relay
//使用命令“encapsulation frame-relay [ ietf ]”。帧中继有两种封装类型：cisco 和
ietf（Internet Engineering Task Force）。对于cisco 路由器，cisco 是它的默认值；对
于非cisco 路由器，须选用ietf 类型。但国内帧中继线路一般为ietf 类型的封装，我们这
```

里由于上面的帧中继交换机中封装类型是cisco，所以这里选择cisco。
R1(config-if)#frame-relay lmi-type cisco
//如果采用的是cisco 路由器且IOS 是11.2 及以后版本的，路由器可以自动适应LMI 的类型，则本步骤可不做。国内帧中继线路一般采用ansi 的LMI 信令类型，我们这里采用的是cisco。因为上面配的是CISCO，DTE端必须要去适应DCE端。
R1(config-if)#no shutdown

在 R2 上：

```
R2(config)#interface serial 0/0
R2(config-if)#encapsulation frame-relay
R2(config-if)#ip address 123.1.1.2 255.255.255.0
R2(config-if)#no shutdown
```

在 R3 上：

```
R3(config)#interface serial 0/0
R3(config-if)#encapsulation frame-relay
R3(config-if)#ip address 123.1.1.3 255.255.255.0
R3(config-if)#no shutdown
```

去 R1 上测试：

```
R1#ping 123.1.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 123.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/60/116 ms
R1#ping 123.1.1.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 123.1.1.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/70/148 ms
```


去查看帧中继的二层映射信息：

```
R1#show frame-relay map
Serial0/0 (up): ip 123.1.1.3 dlci 103(0x66,0x1860), dynamic,
                broadcast,, status defined, active
Serial0/0 (up): ip 123.1.1.2 dlci 102(0x67,0x1870), dynamic,
                broadcast,, status defined, active
```

//默认时，帧中继接口开启了动态映射，会自动建立帧中继映射，“dynamic”表明这是动态映射。

去查看 PVC 的状态信息：

```
R1#show frame-relay pvc
```

```
PVC Statistics for interface Serial0/0 (Frame Relay DTE)

                Active      Inactive      Deleted      Static
  Local            2            0            0           0
  Switched         0            0            0           0
  Unused           0            0            0           0

DLCI = 102, DLCI USAGE = LOCAL, PVC STATUS = ACTIVE, INTERFACE = Serial0/0
————以下部分省略————
```

//可以看到 PVC 的状态为 Active，这是正常状态。如果为 Inactive，表示 PVC 的对端有问题，如果是 Deleted 表示本地的 PVC 有问题。

接下来要手工的去配置帧中继的映射。
首先去关闭帧中继的逆向 ARP，帧中继的自动映射就是靠 IARP 生成的。

```
R1(config)#interface serial 0/0
R1(config-if)#no frame-relay inverse-arp
```

//如果我们要再开启用命令 frame-relay inverse-arp。

```
R1#clear frame-relay inarp
```

//以上是清除刚才自动生成的映射关系


```
R1(config)#interface serial 0/0
R1(config-if)#frame-relay map ip 123.1.1.2 102 broadcast
R1(config-if)#frame-relay map ip 123.1.1.3 103 broadcast
```

//注意，这里的IP地址是对方的IP地址，DLCI号是本地的。broadcast是允许该帧中继链路通过多播或广播包，如果帧中继链路上要运行路由协议，该参数非常重要。

在 R2 上：

```
R2(config)#interface serial 0/0
R2(config-if)#no frame-relay inverse-arp
R2(config-if)#frame-relay map ip 123.1.1.1 201 broadcast
```

在 R3 上：

```
R3(config)#interface serial 0/0
R3(config-if)#no frame-relay inverse-arp
R3(config-if)#frame-relay map ip 123.1.1.1 301 broadcast
```

```
R1#show frame-relay map
Serial0/0 (up): ip 123.1.1.3 dlci 103(0x67,0x1870), static,
                broadcast,
                CISCO, status defined, active
Serial0/0 (up): ip 123.1.1.2 dlci 102(0x66,0x1860), static,
```

```
        broadcast,
        CISCO, status defined, active
```

//static 表示是手动产生的映射关系。

再去 PING 测试：

```
R1#ping 123.1.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 123.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 48/68/104 ms
R1#ping 123.1.1.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 123.1.1.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/71/112 ms
```

想一想，如果去 R2 上 PING R3 能通吗？为什么？

不要删除配置，接下来的实验是在此基础之上的。

实验 9.3：在帧中继的链路上运行 RIP V2

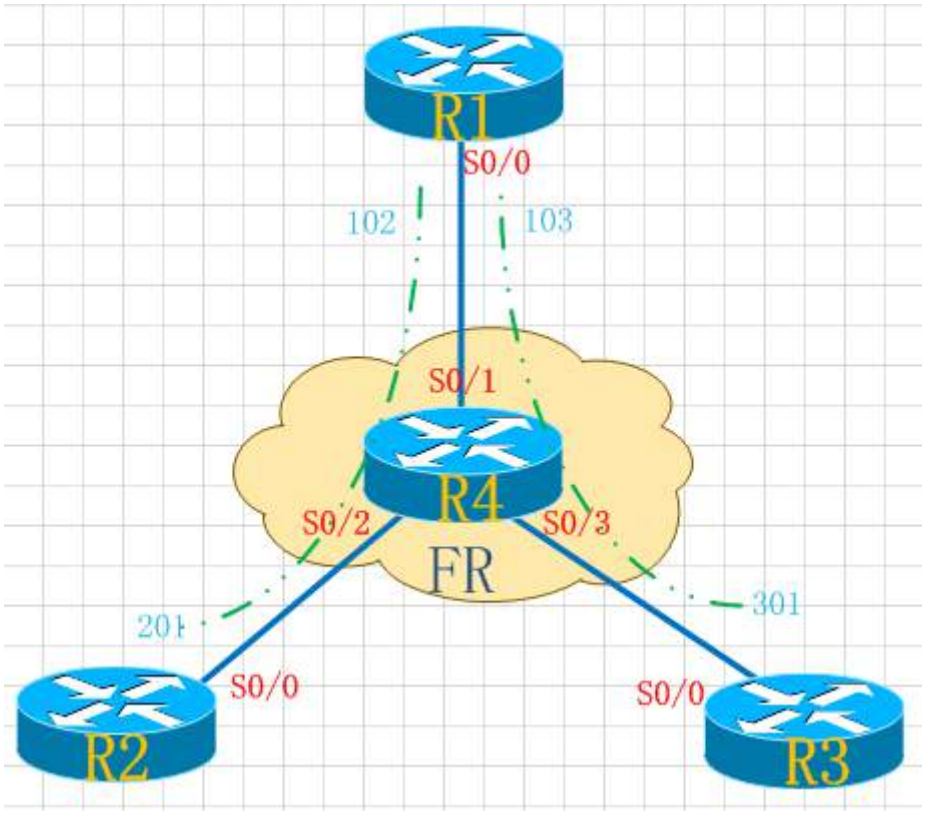
实验目标

通过本实验，读者可以掌握如下技能：

（1） 帧中继上路由协议运行的特殊性

（2） 水平分割

实验拓扑



实验步骤

本实验在实验 9.2 的基础上完成。
每个路由器都有一个环回口地址分别为 XX.1.1.1/24 位。
步骤 1：先配置好环回口：

在 R1 上：

```
R1(config)#interface loopback 0
R1(config-if)#ip address 11.1.1.1 255.255.255.0
```

在 R2 上：

```
R2(config)#interface loopback 0
R2(config-if)#ip address 22.1.1.1 255.255.255.0
```

在 R3 上：

```
R3(config)#interface loopback 0
R3(config-if)#ip address 33.1.1.1 255.255.255.0
```

步骤 2：在 R1,R2,R3 上分别去配置 RIP 协议。
在 R1 上：

```
R1(config)#router rip
R1(config-router)#version 2
R1(config-router)#no auto-summary
R1(config-router)#network 11.0.0.0
R1(config-router)#network 123.0.0.0
```

在 R2 上：

```
R2(config)#router rip
R2(config-router)#version 2
R2(config-router)#no auto-summary
R2(config-router)#network 22.0.0.0
R2(config-router)#network 123.0.0.0
```

在 R3 上：

```
R3(config)#router rip
R3(config-router)#version 2
R3(config-router)#no auto-summary
R3(config-router)#network 33.0.0.0
R3(config-router)#network 123.0.0.0
```

配好以后分别去 R1,R2,R3 查看路由表：
在 R1 上:

```
R1#show ip route rip
      33.0.0.0/24 is subnetted, 1 subnets
R        33.1.1.0 [120/1] via 123.1.1.3, 00:00:17, Serial0/0
      22.0.0.0/24 is subnetted, 1 subnets
R        22.1.1.0 [120/1] via 123.1.1.2, 00:00:08, Serial0/0
```


在 R2 上：

```
R2#show ip route rip
      33.0.0.0/24 is subnetted, 1 subnets
R        33.1.1.0 [120/2] via 123.1.1.3, 00:00:01, Serial0/0
      11.0.0.0/24 is subnetted, 1 subnets
R        11.1.1.0 [120/1] via 123.1.1.1, 00:00:01, Serial0/0
```

在 R3 上：

```
R3#show ip route rip
      22.0.0.0/24 is subnetted, 1 subnets
R        22.1.1.0 [120/2] via 123.1.1.2, 00:00:06, Serial0/0
      11.0.0.0/24 is subnetted, 1 subnets
R        11.1.1.0 [120/1] via 123.1.1.1, 00:00:06, Serial0/0
```

发现路由表都学习正常，注意红色部分路由的下一跳,再去 R2 上 PING R3 的环回口。

```
R2#ping 33.1.1.1

Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 33.1.1.1, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```
//发现虽然 R2 上有 R3 环回口路由，但是 PING 不通。

再去查看二层帧中继的映射关系。

```
R2#show frame-relay map
Serial0/0 (up): ip 123.1.1.1 dlci 201(0xC9,0x3090), static,
                broadcast,
                CISCO, status defined, active
```

//通过以上显示可以知道不通的原因是二层帧中继的映射中并没有 33.1.1.1/24 这条路由的下一跳 123.1.1.3 所以我们必须去添加。这也解决了在上一个实验结束留下的那个问题。

在 R2 上：

```
R2(config-if)#frame-relay map ip 123.1.1.3 201 broadcast
```

同理，我们要去 R3 上添加：

```
R3(config-if)#frame-relay map ip 123.1.1.2 301 broadcast
```

再去测试：

```
R2#ping 33.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 33.1.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 60/144/280 ms
```


在 R1 上用以下命令去查看接口状态：

```
R1#show ip interface serial 0/0
Serial0/0 is up, line protocol is up
  Internet address is 123.1.1.1/24
  Broadcast address is 255.255.255.255
 ————以上部分省略————
  Split horizon is disabled
```

//以上表示当在一个接口封装帧中继以后，水平分割默认被关闭了。

```
 ————以下部分省略————
```

我们去 R1 开启水平分割再观察路由表：

```
R1(config)#interface serial 0/0
R1(config-if)#ip split-horizon
```

去 R2 上：

```
R2#clear ip route *
```

```
R2#show ip route rip
      11.0.0.0/24 is subnetted, 1 subnets
R        11.1.1.0 [120/1] via 123.1.1.1, 00:00:01, Serial0/0
```

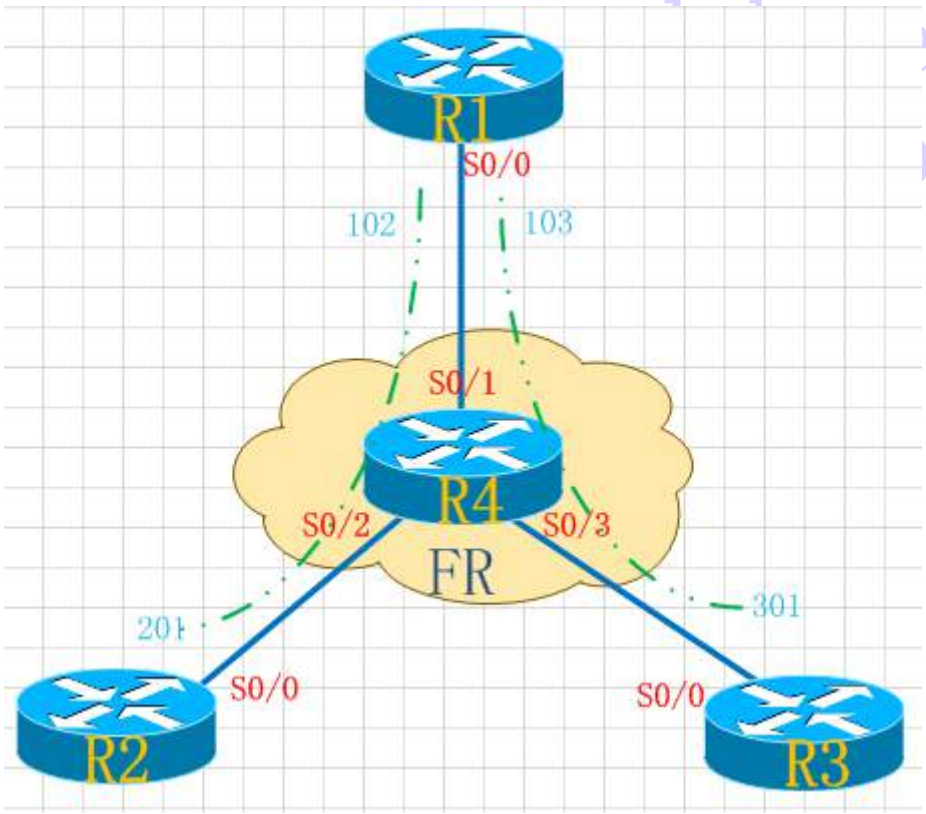
//我们发现 R2 不能再次学到 R3 的路由了，这是由于水平分割的原因。R1 从接口 S0/0 收到的路由不会再从 S0/0 口发出去，这种造成了 R1,R3 不能学习到相互的路由了。
当然，在 RIP 的环境下，水平分割被关闭了。但是请注意，在 EIGRP 中，需要我们手动去关闭 EIGRP 的水平分割才能相互学习路由。

实验 9.4: 帧中继的多点子接口

实验目标

通过本实验，读者可以掌握如下技能：
（1） 点到多点子接口的配置

实验拓扑



实验步骤

此实验在 9.1 的基础上完成

实验要求，R2,R3 依然使用主接口，在 R1 上使用多点子接口来连接 R2 与 R3。
因为在帧中继中的多点子接口与主接口的特性完全一样。

在 R2 上：

```
R2(config)#interface serial 0/0
R2(config-if)#no frame-relay inverse-arp
R2(config-if)#frame-relay map ip 123.1.1.1 201 broadcast
R2(config-if)#frame-relay map ip 123.1.1.3 201 broadcast
```

在 R3 上：

```
R3(config)#interface serial 0/0
R3(config-if)#no frame-relay inverse-arp
R3(config-if)#frame-relay map ip 123.1.1.1 301 broadcast
R3(config-if)#frame-relay map ip 123.1.1.2 301 broadcast
```

去 R1 上配置帧中继的多点子接口：

```
R1(config)#interface serial 0/0
R1(config-if)#encapsulation frame-relay
R1(config-if)#no ip address
//主接口上不需要再配 IP 地址了。
R1(config-if)#no shutdown
R1(config-if)#exit
```



```
R1(config)#interface serial 0/0.1 multipoint
//创建点到多点子接口
R1(config-subif)#ip address 123.1.1.1 255.255.255.0
R1(config-subif)#frame-relay map ip 123.1.1.2 102 broadcast
R1(config-subif)#frame-relay map ip 123.1.1.3 103 broadcast
//多点子接口上的配置与主接口上一致。
```

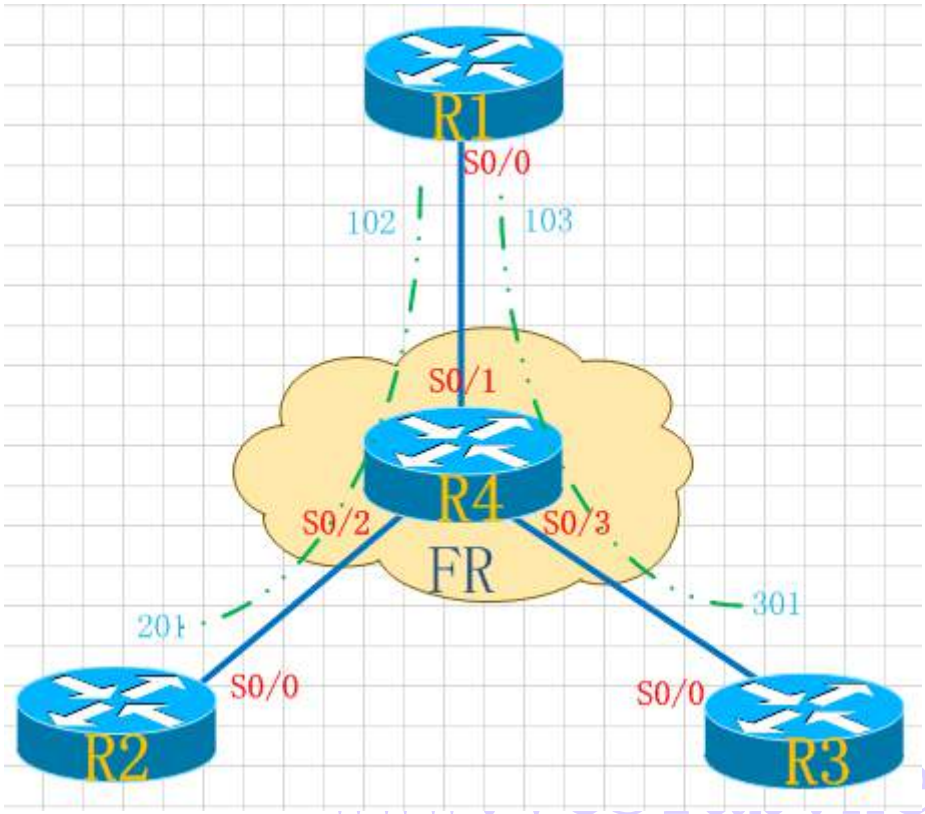
注意，子接口的水平分割并不会默认关闭。

实验 9.5：帧中继的点到点子接口

实验目标

通过本实验，读者可以掌握如下技能：
（1） 点到点子接口的配置

实验拓扑



实验步骤

此实验是在实验 9.1 的基础上完成的
我们在 R1,R2,R3 上分别起点对点的子接口。

在 R1 上：

```
R1(config)#interface serial 0/0
R1(config-if)#encapsulation frame-relay
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#interface serial 0/0.2 point-to-point
//以上是起一个连接 R2 的点对点的子接口
```

```
R1(config-subif)#ip address 12.1.1.1 255.255.255.0
R1(config-subif)#frame-relay interface-dlci 102
//以上是为这个接口配置一个 DLCI 号，和主接口的映射不同。
```

```
R1(config-fr-dlci)#exit
R1(config-subif)#exit
R1(config)#interface serial 0/0.3 point-to-point
R1(config-subif)#ip address 13.1.1.1 255.255.255.0
R1(config-subif)#frame-relay interface-dlci 103
R1(config-fr-dlci)#exit
```

在 R2 上：

```
R2(config)#interface serial 0/0
R2(config-if)#encapsulation frame-relay
R2(config-if)#no shutdown
R2(config-if)#exit
R2(config)#interface serial 0/0.1 point-to-point
R2(config-subif)#ip address 12.1.1.2 255.255.255.0
R2(config-subif)#frame-relay interface-dlci 201
R2(config-fr-dlci)#end
```

在 R3 上：

```
R3(config)#interface serial 0/0
R3(config-if)#encapsulation frame-relay
R3(config-if)#no shutdown
R3(config-if)#exit
R3(config)#interface serial 0/0.1 point-to-point
R3(config-subif)#ip address 13.1.1.3 255.255.255.0
R3(config-subif)#frame-relay interface-dlci 301
```

去 R1 上测试：


```
R1#ping 12.1.1.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 12.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/64/148 ms
R1#ping 13.1.1.3

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 13.1.1.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/61/148 ms
```

实验第十部分：访问控制列表

随着大规模开放式网络的开发，网络面临的威胁也就越来越多。网络安全问题成为网络管理员最为头疼的问题。一方面，为了业务的发展，必须允许对网络资源的开发访问，另一方面，又必须确保数据和资源的尽可能安全。网络安全采用的技术很多，而通过访问控制列表（ACL）可以对数据流进行过滤，是实现基本的网络安全手段之一。本章只研究基于IP的ACL。

ACL 概述

访问控制列表简称为ACL，它使用包过滤技术，在路由器上读取第三层及第四层包头中的信息如源地址、目的地址、源端口、目的端口等，根据预先定义好的规则对包进行过滤，从而达到访问控制的目的。ACL 分很多种，不同场合应用不同种类的ACL。

1. 标准ACL

标准ACL 最简单，是通过使用IP 包中的源IP 地址进行过滤，表号范围1-99 或1300-1999；

2. 扩展ACL

扩展ACL 比标准ACL 具有更多的匹配项，功能更加强大和细化，可以针对包括协议类型、源地址、目的地址、源端口、目的端口、TCP 连接建立等进行过滤，表号范围100-199 或2000-2699；

3. 命名ACL

以列表名称代替列表编号来定义ACL，同样包括标准和扩展两种列表。在访问控制列表的学习中，要特别注意以下两个术语。

1. 通配符掩码：一个32 比特位的数字字符串,它规定了当一个IP 地址与其他的IP 地址进行比较时，该IP 地址中哪些位应该被忽略。通配符掩码中的“1”表示忽略IP 地址中对应的位，而“0”则表示该位必须匹配。两种特殊的通配符掩码是“255.255.255.255”和“0.0.0.0”，前者等价于关键字“any”，而后者等价于关键字“host”；

2. Inbound 和outbound:当在接口上应用访问控制列表时,用户要指明访问控制列表是应用于流入数据还是流出数据。

总之，ACL 的应用非常广泛，它可以实现如下的功能：

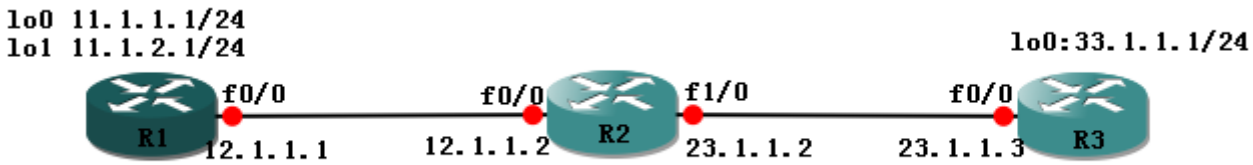
1. 拒绝或允许流入（或流出）的数据流通过特定的接口；
2. 为DDR 应用定义感兴趣的数据流；
3. 过滤路由更新的内容；
4. 控制对虚拟终端的访问；
5. 提供流量控制。

实验 10.1 标准访问控制列表

实验目标

学会使用标准的 ACL 控制网络流量

实验拓扑



实验步骤

实验要求：
要求 1:拒绝 R1 用环回口地址 11.1.2.1 去 ping 通 33.1.1.1.

配置好 IP 地址，并用你最熟悉的路由协议保证全网 IP 可达。这里不再指导。
配好以后，先测试 IP 连通性：
在 R1 上：

```
R1#ping 33.1.1.1 source 11.1.2.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 33.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 11.1.2.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 12/53/76 ms
```

//以上表示现在 R1 可以用地址 11.1.2.1 去 PING 通 33.1.1.1

在 R2 上去拒绝他。
在 R2 上：

```
R2(config)#access-list 10 deny 11.1.2.0 0.0.0.255
```

//以上是拒绝 11.1.2.0/24 这个网段

```
R2(config)#access-list 10 permit any
```

//以上是允许其它所有数据包通过

```
R2(config)#interface fastEthernet 1/0
R2(config-if)#ip access-group 10 out
```

//以上是到接口上去调用刚才写的 ACL

再去 R1 上测试：

```
R1#ping 33.1.1.1 source 11.1.2.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 33.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 11.1.2.1
U.U.U
Success rate is 0 percent (0/5)
```

//以上表示现在不通了。说明刚才的配置已经生效。
想一想：我们在 R2 的 F0/0 口上能实现要求吗？如果可以，要用什么方向的访问控制列表呢？

要求 2：只允许用 R1 的 LO0:11.1.1.1 去 telnet 登陆 R3。
在 R3 上：

```
R3(config)#access-list 11 permit 11.1.1.1
R3(config)#line vty 0 4
R3(config-line)#access-class 11 in
```

去 R2 上试试：

```
R2#telnet 33.1.1.1
Trying 33.1.1.1 ...
% Connection refused by remote host
//被拒绝了
```

在 R1 上测试：

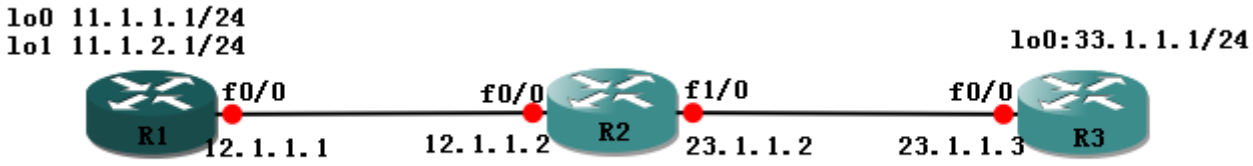
```
R1#telnet 33.1.1.1
Trying 33.1.1.1 ...
% Connection refused by remote host
//以上表示也被拒绝了，
我们再用允许的那个 IP 地址去 TELNET
R1#telnet 33.1.1.1 /source-interface loopback 0
Trying 33.1.1.1 ... Open


User Access Verification

Password:
//以上表示成功的 telnet 上了。也达到了实验目的。
```


实验 10.2 扩展访问控制列表

实验目标

定义和应用扩展ACL

实验拓扑



实验步骤

配置好 IP 地址，并用你最熟悉的路由协议保证全网 IP 可达。这里不再指导。
配好以后，先测试 IP 连通性。
实验要求：只拒绝 R1 用 LO0 去 ping R2,并且只允许 R1 用 LO0 地址去 telnet R2。用一个扩展 ACL 在 R2 的 F0/0 去实现。

在 R2 上：

```
R2(config)#access-list 100 deny icmp host 11.1.1.1 any echo
R2(config)#access-list 100 deny icmp host 11.1.1.1 any echo-reply
//以上是拒绝了 11.1.1.1 的 PING 报文
```

```
R2(config)#access-list 100 permit tcp host 11.1.1.1 any eq telnet
R2(config)#access-list 100 deny tcp any any eq telnet
//以上是只允许了 11.1.1.1 的 telnet 报文
```

```
R2(config)#access-list 100 permit ip any any
//最后允许了其它的所有报文
```

```
R2#show access-lists
Extended IP access list 100
    10 deny icmp host 11.1.1.1 any echo
    20 deny icmp host 11.1.1.1 any echo-reply
    30 permit tcp host 11.1.1.1 any eq telnet
    40 deny tcp any any eq telnet
    50 permit ip any any
```

去接口下调用：

```
R2(config)#interface fastEthernet 0/0
R2(config-if)#ip access-group 100 in
```


去 R1 上测试：

```
R1#ping 12.1.1.2 source 11.1.1.1

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 12.1.1.2, timeout is 2 seconds:
Packet sent with a source address of 11.1.1.1
U.U.U
Success rate is 0 percent (0/5)
//以上表示 11.1.1.1 已经不能 PING 通 R2 了
```

```
R1#telnet 12.1.1.2 /source-interface loopback 0
Trying 12.1.1.2 ... Open

User Access Verification

Password:
//以上表示 11.1.1.1 能 telnet R2.
```

你可以再用别的地址去 ping 和 telnet R2.看是否符合实验的要求。

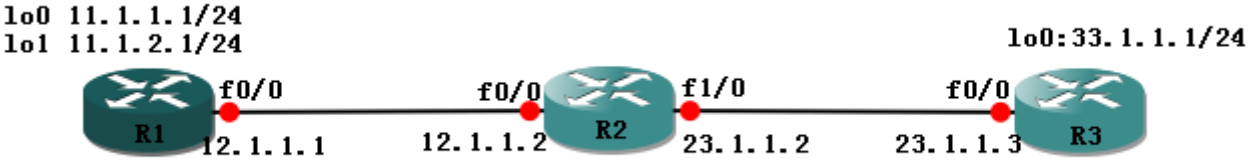
实验 10.3 命名访问控制列表

命名ACL 允许在标准ACL 和扩展ACL 中，使用字符串代替前面所使用的数字来表示ACL。命名ACL 还可以被用来从某一特定的ACL 中删除个别的控制条目，这样可以让网络管理员方便地修改ACL。

实验目标

定义和应用命名ACL

实验拓扑



实验步骤

本实验给出如何用命名 ACL 来实现实验 10.1 中和实验 10.2 中的要求
要求 1: 拒绝 R1 用环回口地址 11.1.2.1 去 ping 通 33.1.1.1.
要求 2：只允许用 R1 的 LO0:11.1.1.1 去 telnet 登陆 R3。
要求 3：只拒绝 R1 用 LO0 去 ping R2,并且只允许 R1 用 LO0 地址去 telnet R2。用一个扩展 ACL 在 R2 的 F0/0 去实现。

要求 1：

```
R2(config)#ip access-list standard yeslab
R2(config-std-nacl)#deny 11.1.2.0 0.0.0.255
R2(config-std-nacl)#permit any
R2(config-std-nacl)#exit
R2(config)#interface fastEthernet 1/0
R2(config-if)#ip access-group yeslab out
```

要求 2：

```
R3(config)#ip access-list standard yeslab
R3(config-std-nacl)#permit 11.1.1.1
R3(config-std-nacl)#exit
R3(config)#line vty 0 4
R3(config-line)#access-class yeslab in
```

要求 3：

```
R2(config)#ip access-list extended yeslab1
R2(config-ext-nacl)#deny icmp host 11.1.1.1 any echo
R2(config-ext-nacl)#deny icmp host 11.1.1.1 any echo-reply
```

```
R2(config-ext-nacl)#permit tcp host 11.1.1.1 any eq telnet
R2(config-ext-nacl)#deny tcp any any eq telnet
R2(config-ext-nacl)#permit ip any any
R2(config-ext-nacl)#exit
R2(config)#interface fastEthernet 0/0
R2(config-if)#ip access-group yeslab1 in
```

以上的CCNA实验指导手册由苏函精心编写，YESLAB出品，希望你好好做实验，早日学有所成。做实验的过程中有任何问题，请直接与我联系。如果你提出了好的意见和建议。我将在修订版中为你冠名。嘿嘿！