RIP

一 实验目的

学会设计较复杂的网络物理拓扑和逻辑网段。掌握路由器上 RIP 协议的配置方法，能够在环境中进行路由器上 RIP 协议的安装与配置，激活参与路由协议的接口使之可以发送接收路由的更新。理解 RIP 路由表的含义，通过网络的变化引起路由表的变化来分析 RIP 协议性能，总结 RIP 协议的适用范围。

二 预备知识

1. RIP

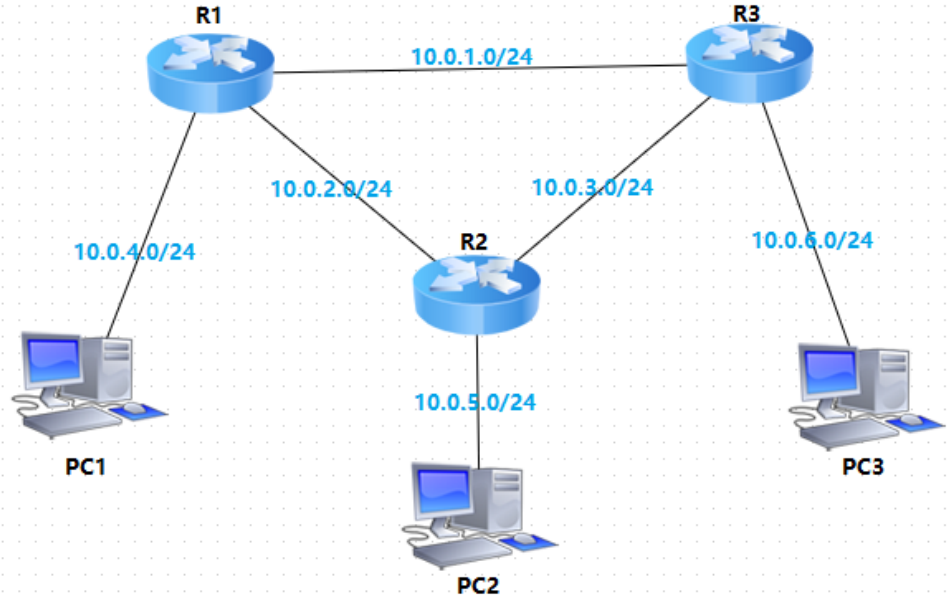
RIP（Routing Information Protocol，路由信息协议）作为一种较为简单的动态路由协议，在实际使用中有着广泛的应用。。RIP 采用距离矢量算法，即路由器根据距离选择路由，所以，也称为距离向量协议。路由器收集所有可到达目的地的不同路径，并且保存有关到达每个目的地的最少站点数的路径信息，除到达目的地的最佳路径外，任何其他信息均予以丢弃。同时，路由器也把所收集的路由信息用 RIP 协议通知相邻的其他路由器。这样，正确的路由信息逐渐扩散到了全网。

2. Zebra

Zebra 是一个运行于 Openwrt 之上的路由软件包，提供基于 TCP/IP 路由服务，支持 RIPv1, RIPv2, RIPng, OSPFv2, OSPFv3, BGP- 4 和 BGP-4+等众多路由协议。Zebra 还支持 BGP 特性路由反射器(Route Reflector)。除了传统的 IPv4 路由协议，Zebra 也支持 IPv6 路由协议。

三 实验环境

在右上方的实验拓扑图菜单中选择 **RIP 实验**，点击连线设置子网网段：



然后点击**提交实验**，等待资源分配成功后，各个设备 IP 地址信息表 1 所示。

表 1 IP 地址信息

设备	网卡接口	IP 地址
R1	eth0	10.0.2.11
	eth1	10.0.1.4
	eth2	10.0.4.4
R2	eth0	10.0.2.6
	eth1	10.0.3.8
	eth2	10.0.5.5
R3	eth0	10.0.1.6
	eth1	10.0.3.7
	eth2	10.0.6.5
PC1	eth0	10.0.4.7
PC2	eth0	10.0.5.10
PC3	eth0	10.0.6.3

四 实验内容

1. 路由器配置 RIP

在每个路由器上配置 Zebra 流程基本类似，这里以 R1 为例。在配置之前对网络进行测试，在 PC1 上进行测试，通过 Ping 命令可以发现 PC2 和 PC3 都是无法访问的。为方便起见，请在开始配置之分别进入三个 Route 中记录下各个设备的网卡端口以及对应的 IP 地址信息(使用命令 ifconfig)，记录成表 1 的形式。

```
eth0      Link encap:Ethernet  HWaddr FA:16:3E:50:79:3C
          inet addr:10.0.2.11  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fe50:793c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4299 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4388 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:341370 (333.3 KiB)  TX bytes:348308 (340.1 KiB)

eth1      Link encap:Ethernet  HWaddr FA:16:3E:CC:29:AD
          inet addr:10.0.1.4  Bcast:10.0.1.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fecc:29ad/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7890 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8168 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:637012 (622.0 KiB)  TX bytes:673720 (657.9 KiB)

eth2      Link encap:Ethernet  HWaddr FA:16:3E:BC:8D:27
          inet addr:10.0.4.4  Bcast:10.0.4.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:febc:8d27/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4152 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4396 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```

图 1 R1 中查看 ip 信息

(1) 关闭防火墙

OpenWrt 的防火墙默认开启，需要我们手动关闭(**/etc/init.d/firewall shutdown**)以防止系统过滤掉 PC 发来的数据包。

```
root@OpenWrt:/# /etc/init.d/firewall shutdown
Warning: Unable to locate ipset utility, disabling ipset support
 * Flushing IPv4 filter table
 * Flushing IPv4 nat table
 * Flushing IPv4 mangle table
 * Flushing IPv4 raw table
 * Flushing IPv6 filter table
 * Flushing IPv6 mangle table
 * Flushing IPv6 raw table
 * Flushing conntrack table ...
```

(2) 配置 Zebra

配置前后我们可以通过 route 查看 R1 中的路由表变化，根据我们前面所做的实验理解相应路由记录的含义，配置前的路由表内容见下图：

```
root@OpenWrt:/# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.0.1.1        0.0.0.0         UG    0      0        0 eth1
10.0.1.0        *               255.255.255.0   U     0      0        0 eth1
10.0.1.1        *               255.255.255.255 UH    0      0        0 eth1
10.0.2.0        *               255.255.255.0   U     0      0        0 eth0
10.0.2.1        *               255.255.255.255 UH    0      0        0 eth0
10.0.4.0        *               255.255.255.0   U     0      0        0 eth2
10.0.4.1        *               255.255.255.255 UH    0      0        0 eth2
```

图 2 路由表(配置前)

运行**/usr/sbin/quagga.init start** 启动 zebra。

```
root@OpenWrt:/# /usr/sbin/quagga.init start
quagga.init: Starting zebra ... found old pid file entry 1575 ... found zebra ru
nning (1575) - skipping zebra.
quagga.init: Starting ripd ... found old pid file entry 1580 ... found ripd runn
ing (1580) - skipping ripd.
quagga.init: Starting ospfd ... found old pid file entry 1585 ... found ospfd ru
nning (1585) - skipping ospfd.
quagga.init: Starting watchquagga ... found old pid file entry 1589 ... found wa
tchquagga running (1589) - skipping watchquagga.
```

telnet 127.0.0.1 2601 访问 zebra 并进行配置（默认密码 **zebra**）。

```
root@OpenWrt:/# telnet 127.0.0.1 2601

Entering character mode
Escape character is '^]'.


Hello, this is Quagga (version 1.1.0).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password:
OpenWrt>
```

enable 进入管理权限，并运行 **configure terminal** 进入终端设置模式。随后根据之前记录的 R1 的网卡接口与 IP 地址信息对各个网卡(eth0、eth1、eth2)进行配置，最后记得使用 **write** 命令将更改保存到配置文件中。

```
OpenWrt> enable
OpenWrt#
OpenWrt# configure terminal
OpenWrt(config)#
OpenWrt(config)# interface eth0
OpenWrt(config-if)#
OpenWrt(config-if)# ip address 10.0.2.11/24
OpenWrt(config-if)#
OpenWrt(config-if)# quit
OpenWrt(config)#
OpenWrt(config)# interface eth1
OpenWrt(config-if)#
OpenWrt(config-if)# ip address 10.0.1.4/24
OpenWrt(config-if)#
OpenWrt(config-if)# quit
```

```
OpenWrt(config)#
OpenWrt(config)# interface eth2
OpenWrt(config-if)#
OpenWrt(config-if)# ip address 10.0.4.4/24
OpenWrt(config-if)#
OpenWrt(config-if)# quit
OpenWrt(config)#
OpenWrt(config)# write
Configuration saved to /etc/quagga/zebra.conf
OpenWrt(config)#
OpenWrt(config)# quit
OpenWrt#
OpenWrt# quit

Connection closed by foreign host
```

(3) 配置 RIP

telnet 127.0.0.1 2602（注意此处与之前端口 2601 不一样）访问 rip 并进行配置（默认密码 **zebra**）。

```
root@OpenWrt:/# telnet 127.0.0.1 2602

Entering character mode
Escape character is '^]'.


Hello, this is Quagga (version 1.1.0).
Copyright 1996-2005 Kunihiro Ishiguro, et al.


User Access Verification

Password:
OpenWrt>
OpenWrt> _
```

与配置 Zebra 类似，**enable** 进入管理权限，并运行 **configure terminal**(可简写成 **conf t**)进入终端设置模式，配置与路由器 R1 直连的各个网段信息，最后记得使用 **write** 命令将更改保存到配置文件中。

```
OpenWrt> enable
OpenWrt#
OpenWrt# conf t
OpenWrt(config)#
OpenWrt(config)# router rip
OpenWrt(config-router)#
OpenWrt(config-router)# network 10.0.2.0/24
OpenWrt(config-router)#
OpenWrt(config-router)# network 10.0.1.0/24
OpenWrt(config-router)#
OpenWrt(config-router)# network 10.0.4.0/24
OpenWrt(config-router)#
OpenWrt(config-router)# version 2
OpenWrt(config-router)#
OpenWrt(config-router)# end
```

```
OpenWrt#
OpenWrt# write
Configuration saved to /etc/quagga/ripd.conf
OpenWrt#
OpenWrt# quit

Connection closed by foreign host
```

(4) 开启 zebra 服务

执行命令**/etc/init.d/quagga start** 开启相应服务。

```
root@OpenWrt:/# /etc/init.d/quagga start
quagga.init: Starting zebra ... found old pid file entry 1575 ... found zebra ru
nning (1575) - skipping zebra.
quagga.init: Starting ripd ... found old pid file entry 1580 ... found ripd runn
ing (1580) - skipping ripd.
quagga.init: Starting ospfd ... found old pid file entry 1585 ... found ospfd ru
nning (1585) - skipping ospfd.
quagga.init: Starting watchquagga ... found old pid file entry 1589 ... found wa
tchquagga running (1589) - skipping watchquagga.
```

请参照以上步骤完成 R2、R3 的相应配置。

2. 查看路由表

所有 Router 都配置完成之后，我们可以再次查看 R1 中的路由表，与配置前的路由表相对照发现除了与 R1 直连的网段信息，另外多出了关于网段 **10.0.3.0/24**、**10.0.5.0/24** 以及 **10.0.6.0/24** 的路由记录。

```
root@OpenWrt:/# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.0.1.1        0.0.0.0         UG    0      0        0 eth1
10.0.1.0        *               255.255.255.0   U     0      0        0 eth1
10.0.1.1        *               255.255.255.255 UH    0      0        0 eth1
10.0.2.0        *               255.255.255.0   U     0      0        0 eth0
10.0.2.1        *               255.255.255.255 UH    0      0        0 eth0
10.0.3.0        10.0.2.6        255.255.255.0   UG    20     0        0 eth0
10.0.4.0        *               255.255.255.0   U     0      0        0 eth2
10.0.4.1        *               255.255.255.255 UH    0      0        0 eth2
10.0.5.0        10.0.2.6        255.255.255.0   UG    20     0        0 eth0
10.0.6.0        10.0.1.6        255.255.255.0   UG    20     0        0 eth1
```

图 3 路由表(配置后)

另外，我们也可以进入 zebra(**telnet 127.0.0.1 2601**)中使用 **show ip route** 查看路由表的相关信息。

```
OpenWrt> show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, P - PIM, A - Babel,
       > - selected route, * - FIB route

K>* 0.0.0.0/0 via 10.0.1.1, eth1, src 10.0.1.4
C>* 10.0.1.0/24 is directly connected, eth1
K>* 10.0.1.1/32 is directly connected, eth1
C>* 10.0.2.0/24 is directly connected, eth0
K>* 10.0.2.1/32 is directly connected, eth0
R>* 10.0.3.0/24 [120/2] via 10.0.2.6, eth0, 02:46:36
C>* 10.0.4.0/24 is directly connected, eth2
K>* 10.0.4.1/32 is directly connected, eth2
R>* 10.0.5.0/24 [120/2] via 10.0.2.6, eth0, 02:46:32
R>* 10.0.6.0/24 [120/2] via 10.0.1.6, eth1, 02:38:30
C>* 127.0.0.0/8 is directly connected, lo
```

其中以 **R** 开头的路由记录表明了该条记录是由 RIP 协议动态生成的(Codes 字段给出了不同值所对应的协议)。如上图第一条 R 开头的路由记录所示，该记录指定了 R1 到达网段 10.0.3.0 的的网卡出口信息即通过 eth0（ip 地址为 10.0.2.6）。其中**[120/2]**指定了该路由记录的优先级（120）以及到达对应网段的路径中需经过的跳数（2）。

接下来我们来测试以下 RIP 对动态网络故障的响应速度，我们进入 R2 中关闭 eth0 端口（**ifconfig eth0 down**），此时 R1 将不能通过 R2 到达 10.0.3.0 和 10.0.5.0/24 网段，而 R1 中的路由表要**等待一段时间**（大概 2 分钟左右）才会更新相关的路由记录：

```
OpenWrt> show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, P - PIM, A - Babel,
       > - selected route, * - FIB route

K>* 0.0.0.0/0 via 10.0.1.1, eth1, src 10.0.1.4
C>* 10.0.1.0/24 is directly connected, eth1
K>* 10.0.1.1/32 is directly connected, eth1
C>* 10.0.2.0/24 is directly connected, eth0
K>* 10.0.2.1/32 is directly connected, eth0
R>* 10.0.3.0/24 [120/2] via 10.0.1.6, eth1, 00:00:44
C>* 10.0.4.0/24 is directly connected, eth2
K>* 10.0.4.1/32 is directly connected, eth2
R>* 10.0.5.0/24 [120/3] via 10.0.1.6, eth1, 00:00:44
R>* 10.0.6.0/24 [120/2] via 10.0.1.6, eth1, 03:18:50
C>* 127.0.0.0/8 is directly connected, lo
```

分析可见，到达 10.0.3.0/24 和 10.0.5.0/24 网段的下一跳地址更新为 10.0.1.6，且到达 10.0.5.0/24 的跳数变为 3，因为此时 R1 需要经过 R3 再到 R2 才能到达目的 10.0.5.0/24 网段（注意 R1 本身也算一跳）。而当我们在 R2 中重启 eth0 端口（**ifconfig eth0 up**）后，R1 会在相对较短的时间内更新相应的路由记录为更短的路径。

```
OpenWrt> show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, P - PIM, A - Babel,
       > - selected route, * - FIB route

K>* 0.0.0.0/0 via 10.0.1.1, eth1, src 10.0.1.4
C>* 10.0.1.0/24 is directly connected, eth1
K>* 10.0.1.1/32 is directly connected, eth1
C>* 10.0.2.0/24 is directly connected, eth0
K>* 10.0.2.1/32 is directly connected, eth0
R>* 10.0.3.0/24 [120/2] via 10.0.1.6, eth1, 00:10:29
C>* 10.0.4.0/24 is directly connected, eth2
K>* 10.0.4.1/32 is directly connected, eth2
R>* 10.0.5.0/24 [120/2] via 10.0.2.6, eth0, 00:00:00
R>* 10.0.6.0/24 [120/2] via 10.0.1.6, eth1, 03:28:35
C>* 127.0.0.0/8 is directly connected, lo
```

透过以上实验我们可以看到 RIP 协议的一个特点：**“好消息”传播的快，“坏消息”传播的慢。**以上述实验为例，当 R2 的 eth0 端口被开启之后，R2 会立即将该消息传播给 R1，R1 会立即据此来更新自己的路由表；而当 R2 的 eth0 端口被关闭之后，由于 R1 与 R2 之间的通讯链路已经被切断，该消息无法立即传递给 R1，R1 只能通过超时处理自行发现原来到达 R2 的链路已经被切断了，然后才会采取策略更新路由表。

3. PC 添加静态路由

此时我们再次测试 PC1 到或 PC3 间的连通性，发现还是不能 ping 通的，原因在于 PC1 和 PC3 中不存在到达相应网段的路由信息，以 PC1 为例，我们添加到网段 10.0.6.0/24（PC3 所在子网）的静态路由信息：

```
C:\Documents and Settings\admin>route add 10.0.6.0 mask 255.255.255.0 10.0.4.4

C:\Documents and Settings\admin>route print
===========================================================================
Interface List
0x1 ........................... MS TCP Loopback interface
0x10003 ...fa 16 3e 5b 57 42 ...... Red Hat VirtIO Ethernet Adapter - 数据包计划
程序微型端口
===========================================================================
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0         10.0.4.1        10.0.4.7       1
         10.0.4.0    255.255.255.0         10.0.4.7        10.0.4.7      10
         10.0.4.7  255.255.255.255        127.0.0.1       127.0.0.1      10
         10.0.6.0    255.255.255.0         10.0.4.4        10.0.4.7       1
   10.255.255.255  255.255.255.255         10.0.4.7        10.0.4.7      10
        127.0.0.0        255.0.0.0        127.0.0.1       127.0.0.1       1
  169.254.169.254  255.255.255.255         10.0.4.2        10.0.4.7       1
```

PC2 和 PC3 的添加方法类似，请自行完成。配置完成后 PC1 也以与 PC3（10.0.6.3）正常通讯了。

```
C:\Documents and Settings\admin>ping 10.0.6.3

Pinging 10.0.6.3 with 32 bytes of data:

Reply from 10.0.6.3: bytes=32 time=2ms TTL=126
Reply from 10.0.6.3: bytes=32 time=1ms TTL=126
Reply from 10.0.6.3: bytes=32 time=1ms TTL=126
Reply from 10.0.6.3: bytes=32 time=1ms TTL=126

Ping statistics for 10.0.6.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 2ms, Average = 1ms
```

五 实验结果

RIP 协议是基于距离矢量算法的内部动态路由协议。其基本工作原理是：每一个路由表维护一张路由表，记录到达该目标网络的时间开销或距离开销，以及对应的输出接口。所有路由器通过**周期性地**向外发送路由刷新报文，在接收到来自各个邻居路由器的路由表后，根据这些路由表来重新计算自己到达各个网络的最佳路由。

通过实验掌握了路由器上 RIP 协议的安装配置以及运作原理，即把自己所知道的整个路由表信息发送到相邻节点，然后相邻节点根据此路由表信息更新自己的路由表，使到达每个目的网络的距离最短。总之，RIP 协议最大的优点是实现简单，开销较小。但 RIP 协议的缺点也较多，除了上面提到的“**好消息传播的快，坏消息传播的慢**”之外。首先，RIP 限制了网络的规模，它采用**跳数**表示路由距离，而最大距离为 15（16 表示不可达）。其次，路由器交换的路由信息是路由器中的完整路由表，因而随着网络规模的扩大，开销也就增加，并且 RIP 是规定好每隔一定时间交换一次路由信息，即使网络拓扑没有变化。因此，RIP 比较适合在规模较小的网络中，而对于规模较大的网络就应该采用其他协议，下一个实验 OSPF 协议就适用于大规模网络中。