计算机网络实验报告

警示

1. 实验心得体会如有雷同，雷同各方当次实验心得体会成绩均以 0 分计。
2. 在规定时间内未上交实验报告的，不得以其他方式补交，当次心得体会成绩按 0 分计。
3. 报告文件以 PDF 文件格式提交。

本报告主要描述学生在实验中承担的工作、遇到的困难以及解决的方法、体会与总结等。

院系	计算机学院	班 级	19 级软工 1 班
学号	19335286		RIP 路由控制协议实验
学生	郑有为		

一、本人承担的工作

操作 PC1 和交换机，绘制拓扑图，分析实验问题

二、遇到的困难及解决方法

1、配置失误的修改

在实验的配置过程中，我们会常出现一些配置错误的事故，下面是本次实验配错的两个例子和他们各自的修改方法，另外，其他组员反应 clear ip route 无法有效地清楚路由表，在配错后继续查找修改方法，避免一次次的一键清。

1.1 交换机配置 VLAN 时 IP 配错，直接重新输入新的 ip address 覆盖掉原来的就可以了。

```
S5750(config)#interface vlan 10
S5750(config-if-VLAN 10)#*Jan 13 09:10:06: %LINEPROTO-5-UPDOWN: Line protocol on
 Interface VLAN 10, changed state to up.

S5750(config-if-VLAN 10)#ip address 10.10.1.2 255.255.255.0
S5750(config-if-VLAN 10)#no shutdown
S5750(config-if-VLAN 10)#exit
S5750(config)#interface vlan 50
S5750(config-if-VLAN 50)#*Jan 13 09:10:48: %LINEPROTO-5-UPDOWN: Line protocol on
 Interface VLAN 50, changed state to up.
```

1.2 RIP 网络划分时配错，同样地，可以选择重新输入，虽然原来的 Network 添加会保留，但是不会影响实验结果。

```
S5750#configure term
Enter configuration commands, one per line.  End with CNTL/Z.
S5750(config)#router rip
S5750(config-router)#version 1
S5750(config-router)#network 10.10.1.0 255.255.255.0
S5750(config-router)#network 10.10.5.0 255.255.255.0
S5750(config-router)#exit
S5750(config)#show ip route
```

2、设备配置软件的“死锁”

这个问题在之前实验一直有出现，就是由于实验过程中我们需要进行断开校园网，但配置路由器交换机的软件需要校园网才能打开，若我们没有关掉那个软件，如路由器交换机断开连接而直接断网，就会发生事故，首先那个软件会卡住，其次是再也无法从那个网站进入配置设备的软件了，除非一键清。

解决方法：断开校园网前小心翼翼，不然一键清。

三、体会与总结

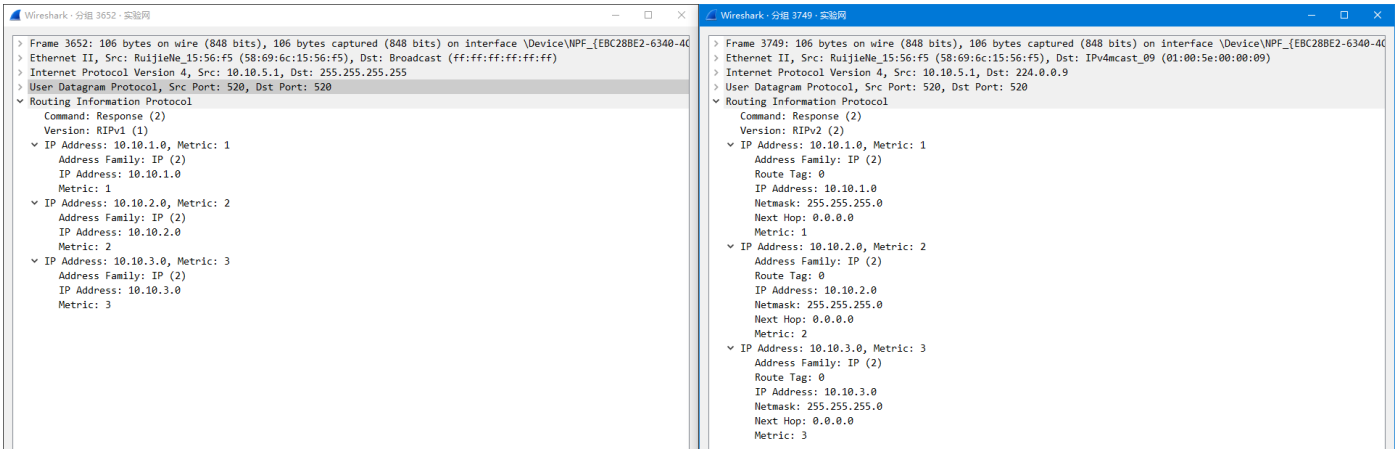
本次实验我们做了一整天，在实验思考的步骤中修改 IP 和掩码多次重复实验的配置和操作，掌握了 RIPv2 与 RIPv1 的配置方法，和两个 RIP 路由协议的特点和限制，学会了 RIPv2 和 RIPv1 报文的分析、明白了毒性反转的选路概念、并弄懂了自动汇总的作用和目的。除此之外，还通过 debug 技术掌握了如何通过 debug 指令产看 RIP 的工作状态。以下是对本次学习过程的概念的总结和自己对实验结果、数据报文的分析。

1. RIPv2 与 RIPv1 的异同

1.1. 提供的服务能力异同

RIPv1	RIPv2
有类别路由协议（类别为 A、B、C 类网络），故不支持可变长子网掩码（VLSM）	无类别路由协议，支持 VLSM，VLSM 由子网掩码标识
广播形式发送报文，广播 IP 为 255.255.255.255	组播发送报文，组播 IP 为 224.0.0.9，好处是可以不向没有运行 RIPv2 的网段发送更新报文
不支持认证	支持明文和 MD5 认证
必须使用自动汇总（自动汇总概念在后面总结）	可使用也可以禁用自动汇总（默认启用）
都支持路由毒化	
每隔 30 秒自动更新路由，在路由变化时能够立即发送报文	
管理一个路由数据库，记录路由项信息，信息包括：目的地址、下一跳地址、端口、度量值、定时器、路由标记	

2.1. 两种报文分析



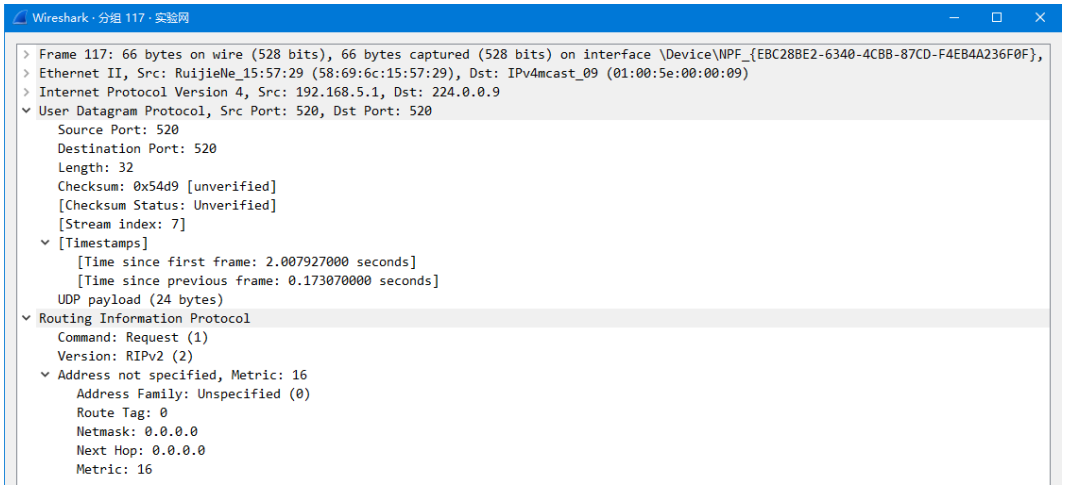
上图是两种报文，从上往下逐层分析，最上层是物理层、到以太网协议、到 IP 协议，IP 协议都

显示源 IP 为 10.10.5.1，而 RIPv1 的目的地址是广播地址 255.255.255.255，RIPv2 的目的地址是多播地址 224.0.0.9，往下分析到 UDP 协议，所使用的端口号都是 520，到路由协议 RIP：分别显示了两个不同的版本和路由信息，对比可以看到 RIPv2 包含路由标识 Route Tag 和子网掩码 Netmask，还有下一跳地址 Next Hop，而这些信息 RIPv1 包中都没有，但两个包路由的跳数 Metric 是相同的。

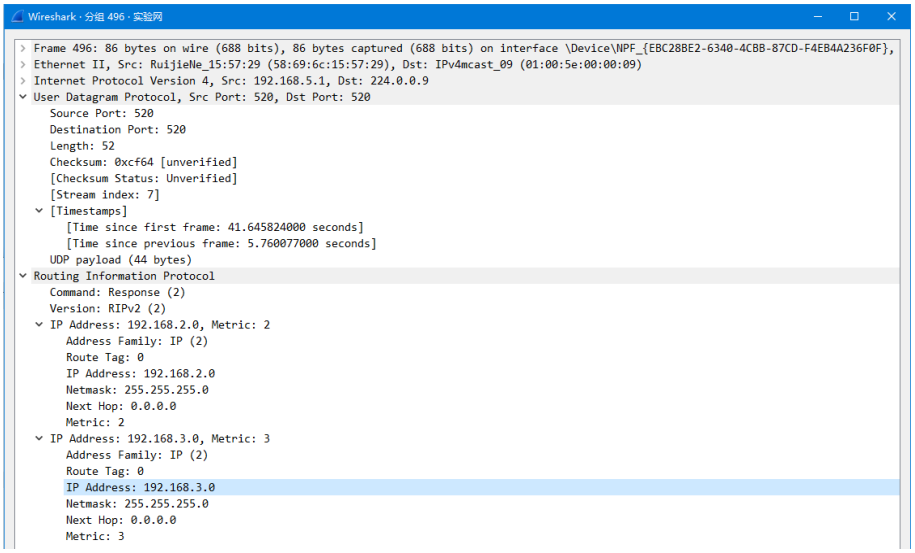
2. RIPv2 报文分析与毒性反转

2.1 毒性反转分析

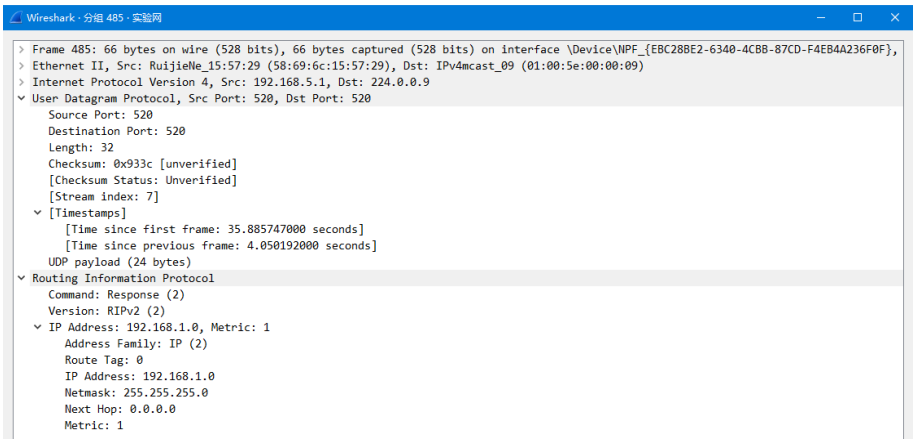
简单理解，毒性反转就是路由不可达的标志，以 Metric 等于 16 来标识。下面几张图是实验过程中 Wirershark 捕获到的毒性反转信息：



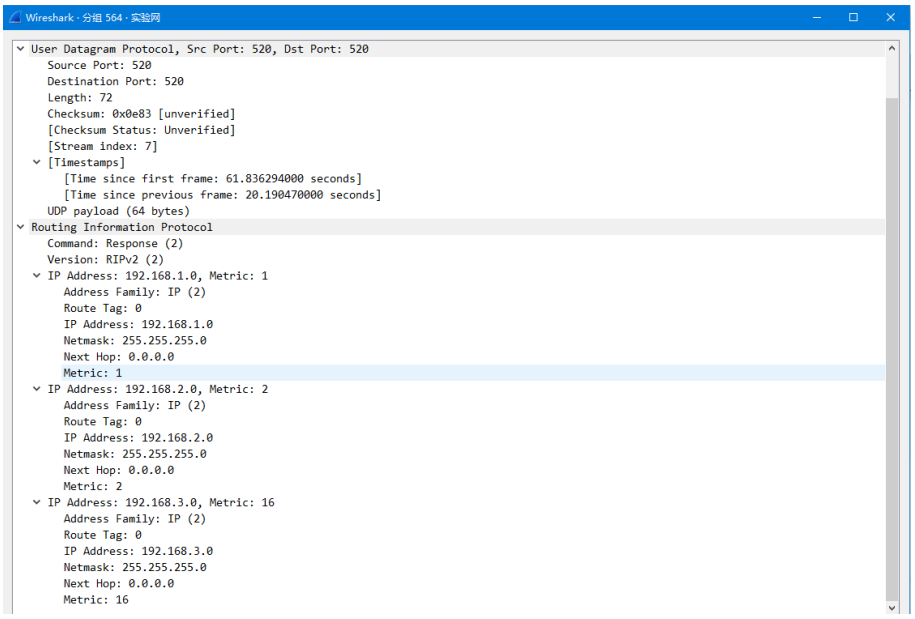
Packet117：是接线错了 PC2 交换机 0/5 接成了 0/3 连交换机都没有接对，所以没有 IP 地址



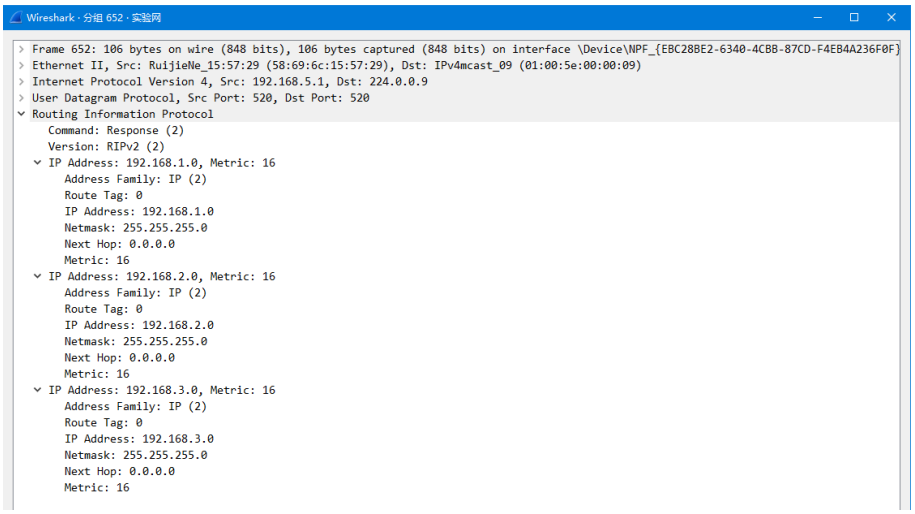
Packet485：是拔了路由器 1 和交换机的线导致的



Packet496：只有两段（第二第三段路由）的原因是因为第一段路没有更新



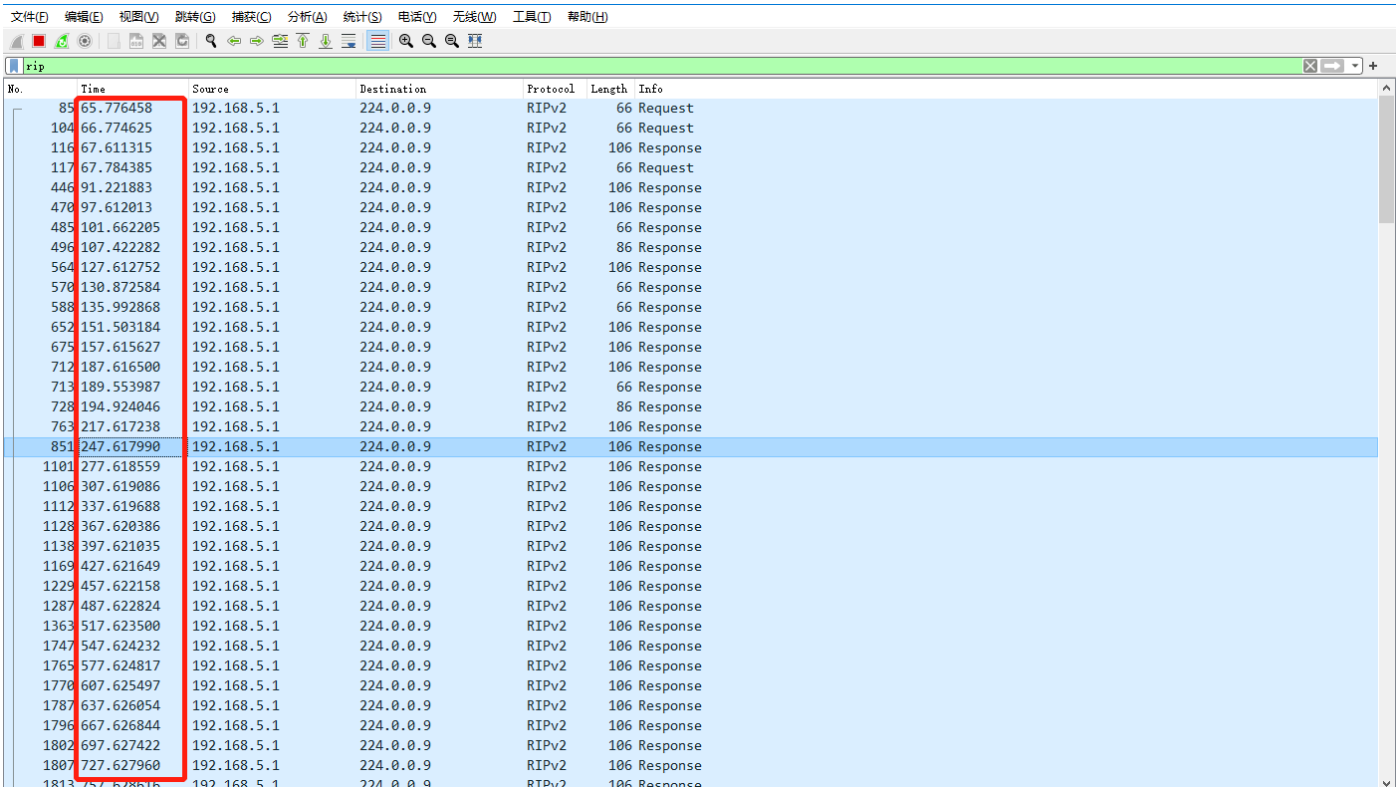
Packet564：拔了路由器 2 和 PC2 之间的线 第三段条数 16，毒性反转发生



Packet 652：拔了交换机和路由器之间的线，导致每一段都发生了毒性反转。

2.2 报文时序分析

可以看到前 851 个包的时间间隔不稳定是 30，因为拔线会导致 RIPv2 包，之后我们停止拔线，可以看到每隔 30 秒回自动发送一个 RIPv2 包。每隔 30 秒会发 RIPv2 包，在拔线的瞬间也会发送 RIPv2 包，但这个包只会发送更新（断掉）的网段信息，而 30 秒的周期包会显示所有网段的连接状态。



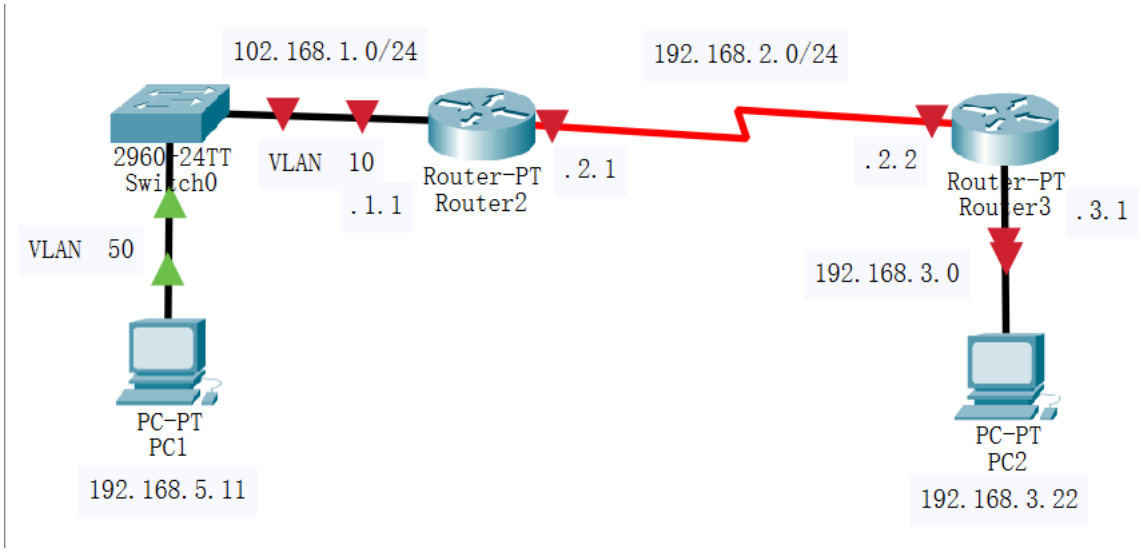
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

rip

No.	Time	Source	Destination	Protocol	Length	Info
85	65.776458	192.168.5.1	224.0.0.9	RIPv2	66	Request
104	66.774625	192.168.5.1	224.0.0.9	RIPv2	66	Request
116	67.611315	192.168.5.1	224.0.0.9	RIPv2	106	Response
117	67.784385	192.168.5.1	224.0.0.9	RIPv2	66	Request
446	91.221883	192.168.5.1	224.0.0.9	RIPv2	106	Response
470	97.612013	192.168.5.1	224.0.0.9	RIPv2	106	Response
485	101.662205	192.168.5.1	224.0.0.9	RIPv2	66	Response
496	107.422282	192.168.5.1	224.0.0.9	RIPv2	86	Response
564	127.612752	192.168.5.1	224.0.0.9	RIPv2	106	Response
570	130.872584	192.168.5.1	224.0.0.9	RIPv2	66	Response
588	135.992868	192.168.5.1	224.0.0.9	RIPv2	66	Response
652	151.503184	192.168.5.1	224.0.0.9	RIPv2	106	Response
675	157.615627	192.168.5.1	224.0.0.9	RIPv2	106	Response
712	187.616500	192.168.5.1	224.0.0.9	RIPv2	106	Response
713	189.553987	192.168.5.1	224.0.0.9	RIPv2	66	Response
728	194.924046	192.168.5.1	224.0.0.9	RIPv2	86	Response
763	217.617238	192.168.5.1	224.0.0.9	RIPv2	106	Response
851	247.617990	192.168.5.1	224.0.0.9	RIPv2	106	Response
1101	277.618559	192.168.5.1	224.0.0.9	RIPv2	106	Response
1106	307.619086	192.168.5.1	224.0.0.9	RIPv2	106	Response
1112	337.619688	192.168.5.1	224.0.0.9	RIPv2	106	Response
1128	367.620386	192.168.5.1	224.0.0.9	RIPv2	106	Response
1138	397.621035	192.168.5.1	224.0.0.9	RIPv2	106	Response
1169	427.621649	192.168.5.1	224.0.0.9	RIPv2	106	Response
1229	457.622158	192.168.5.1	224.0.0.9	RIPv2	106	Response
1287	487.622824	192.168.5.1	224.0.0.9	RIPv2	106	Response
1363	517.623500	192.168.5.1	224.0.0.9	RIPv2	106	Response
1747	547.624232	192.168.5.1	224.0.0.9	RIPv2	106	Response
1765	577.624817	192.168.5.1	224.0.0.9	RIPv2	106	Response
1770	607.625497	192.168.5.1	224.0.0.9	RIPv2	106	Response
1787	637.626054	192.168.5.1	224.0.0.9	RIPv2	106	Response
1796	667.626844	192.168.5.1	224.0.0.9	RIPv2	106	Response
1802	697.627422	192.168.5.1	224.0.0.9	RIPv2	106	Response
1807	727.627960	192.168.5.1	224.0.0.9	RIPv2	106	Response

3. Debug 信息解读 RIP 工作状态

3.1 debug ip rip 取一处实验截图进行分析（路由器 1）



（拓扑图，以便下面的毒性反转分析）

```
*Jun  4 09:23:41: %7:  [RIP] RIP recveived packet, sock=32979 src=10.10.2.2 len=24
*Jun  4 09:23:41: %7:  [RIP] Received version 2 response packet on Serial 2/0
*Jun  4 09:23:41: %7:  [RIP] Cancel peer[10.10.2.2] remove timer
*Jun  4 09:23:41: %7:  [RIP] Peer[10.10.2.2] remove timer shedule...
*Jun  4 09:23:41: %7:  [RIP] Both do not need auth, Auth ok
*Jun  4 09:23:41: %7:        route-entry: family 2 tag 0 ip 10.10.3.0 mask 255.255.255.0 nhop 0.0.0.0 metric 16
*Jun  4 09:23:41: %7:  [RIP] [10.10.3.0/24] RIP route disabling...
*Jun  4 09:23:41: %7:  [RIP] [10.10.3.0/24] cancel route timer
*Jun  4 09:23:41: %7:  [RIP] [10.10.3.0/24] route timer schedule...
*Jun  4 09:23:41: %7:  [RIP] Trigger timer Schedule, by instance 0
*Jun  4 09:23:41: %7:  [RIP] [10.10.3.0/24] ready to add into kernel...
*Jun  4 09:23:41: %7:  [RIP] NSM delete: IPv4 Route 10.10.3.0/24
*Jun  4 09:23:43: %7:  [RIP] Update timer expired via interface Serial 2/0[10.10.2.1/24]
*Jun  4 09:23:43: %7:  [RIP] Update timer schedule via interface Serial 2/0[10.10.2.1/24]
*Jun  4 09:23:43: %7:  [RIP] Prepare to send MULTICAST response...
*Jun  4 09:23:43: %7:  [RIP] Building update entries on Serial 2/0
*Jun  4 09:23:43: %7:        10.10.1.0/24 via 0.0.0.0 metric 1 tag 0
*Jun  4 09:23:43: %7:        10.10.5.0/24 via 0.0.0.0 metric 2 tag 0
*Jun  4 09:23:43: %7:  [RIP] Send packet to 224.0.0.9 Port 520 on Serial 2/0
*Jun  4 09:23:43: %7:  [RIP] Trigger timer expired, by instance 0
*Jun  4 09:23:43: %7:  [RIP] Prepare to send MULTICAST response...
*Jun  4 09:23:43: %7:  [RIP] Building update entries on Serial 2/0
*Jun  4 09:23:43: %7:  [RIP] Skip route[10.10.1.0/24] in trigger
*Jun  4 09:23:43: %7:  [RIP] Skip route[10.10.2.0/24] in trigger
*Jun  4 09:23:43: %7:  [RIP] Skip route[10.10.5.0/24] in trigger
*Jun  4 09:23:43: %7:  [RIP] Skip send response packet...
*Jun  4 09:23:43: %7:  [RIP] Prepare to send MULTICAST response...
*Jun  4 09:23:43: %7:  [RIP] Building update entries on GigabitEthernet 0/1
*Jun  4 09:23:43: %7:  [RIP] Skip route[10.10.1.0/24] in trigger
*Jun  4 09:23:43: %7:  [RIP] Skip route[10.10.2.0/24] in trigger
*Jun  4 09:23:43: %7:        10.10.3.0/24 via 0.0.0.0 metric 16 tag 0
*Jun  4 09:23:43: %7:  [RIP] Skip route[10.10.5.0/24] in trigger
*Jun  4 09:23:43: %7:  [RIP] Send packet to 224.0.0.9 Port 520 on GigabitEthernet 0/1
```

原截图的信息比较多，这里截取了两个时间点的 debug 信息进行分析。

1-4 行：首先路由器接收到来自 10.10.2.2（路由器 2）的一个包，接口来自 Serial2/0，重置 10.10.2.2 的计时器，判断都不需要认证（RIPv2 支持认证，所以需判断）。

5-13 行：10.10.3.0 发生了毒性反转（因为我们把路由器 2 与 PC2 的线给拔了），路由 10.10.3.0 的路由信息置为失效，并关闭路由器，进入内核删除 10.10.3.0 的路由信息，更新 Serial2/0 的计时器。

14-19 行：构建发送内容，包含网段 10.10.1.0 和 10.10.5.0 的跳数等信息。因为接口连着就是 10.10.2.0，所以无需更新改网段的跳数信息，然后将这些内容从 Serial2/0 发送出去。

20-26 行：从 Serial2/0 发送多播信息，构建发送内容，因为刚发送，跳过这些路由，并跳过发送包这个过程。

27-33 行：构建发送内容，跳过 10.10.1.0、10.10.2.0 和 10.10.5.0 更新，进更新 10.10.3.0（发生的毒性反转），然后将这些内容从 Gi0/1 发送出去。

3.2 debug ip packet 取一处实验截图进行分析

```
Router1#debug ip packet
Router1#*Jun  4 09:04:11: %7: IP: s=10.10.2.2 (Serial 2/0), d=224.0.0.9,vrf=global(0),len=52,received
*Jun  4 09:04:13: %7: IP: s=10.10.2.1 (local), d=224.0.0.9 (Serial 2/0),vrf=global(0), g=224.0.0.9,len=72,sent ip pkt to link_layer  222
*Jun  4 09:04:15: %7: IP: s=10.10.1.1 (local), d=224.0.0.9 (GigabitEthernet 0/1),vrf=global(0), g=224.0.0.9,len=72,sent ip pkt to link_layer --> raw send
*Jun  4 09:04:26: %7: IP: s=10.10.1.2 (GigabitEthernet 0/1), d=224.0.0.9,vrf=global(0),len=52,received
*Jun  4 09:04:41: %7: IP: s=10.10.2.2 (Serial 2/0), d=224.0.0.9,vrf=global(0),len=52,received
*Jun  4 09:04:43: %7: IP: s=10.10.2.1 (local), d=224.0.0.9 (Serial 2/0),vrf=global(0), g=224.0.0.9,len=72,sent ip pkt to link_layer  222
*Jun  4 09:04:45: %7: IP: s=10.10.1.1 (local), d=224.0.0.9 (GigabitEthernet 0/1),vrf=global(0), g=224.0.0.9,len=72,sent ip pkt to link_layer --> raw send
*Jun  4 09:04:56: %7: IP: s=10.10.1.2 (GigabitEthernet 0/1), d=224.0.0.9,vrf=global(0),len=52,received
*Jun  4 09:05:11: %7: IP: s=10.10.2.2 (Serial 2/0), d=224.0.0.9,vrf=global(0),len=52,received
*Jun  4 09:05:13: %7: IP: s=10.10.2.1 (local), d=224.0.0.9 (Serial 2/0),vrf=global(0), g=224.0.0.9,len=72,sent ip pkt to link_layer  222
*Jun  4 09:05:15: %7: IP: s=10.10.1.1 (local), d=224.0.0.9 (GigabitEthernet 0/1),vrf=global(0), g=224.0.0.9,len=72,sent ip pkt to link_layer --> raw send
*Jun  4 09:05:26: %7: IP: s=10.10.1.2 (GigabitEthernet 0/1), d=224.0.0.9,vrf=global(0),len=52,received
*Jun  4 09:05:41: %7: IP: s=10.10.2.2 (Serial 2/0), d=224.0.0.9,vrf=global(0),len=52,received
*Jun  4 09:05:43: %7: IP: s=10.10.2.1 (local), d=224.0.0.9 (Serial 2/0),vrf=global(0), g=224.0.0.9,len=72,sent ip pkt to link_layer  222
*Jun  4 09:05:45: %7: IP: s=10.10.1.1 (local), d=224.0.0.9 (GigabitEthernet 0/1),vrf=global(0), g=224.0.0.9,len=72,sent ip pkt to link_layer --> raw send
*Jun  4 09:05:56: %7: IP: s=10.10.1.2 (GigabitEthernet 0/1), d=224.0.0.9,vrf=global(0),len=52,received
*Jun  4 09:06:11: %7: IP: s=10.10.2.2 (Serial 2/0), d=224.0.0.9,vrf=global(0),len=52,received
*Jun  4 09:06:13: %7: IP: s=10.10.2.1 (local), d=224.0.0.9 (Serial 2/0),vrf=global(0), g=224.0.0.9,len=72,sent ip pkt to link_layer  222
*Jun  4 09:06:15: %7: IP: s=10.10.1.1 (local), d=224.0.0.9 (GigabitEthernet 0/1),vrf=global(0), g=224.0.0.9,len=72,sent ip pkt to link_layer --> raw send
*Jun  4 09:06:26: %7: IP: s=10.10.1.2 (GigabitEthernet 0/1), d=224.0.0.9,vrf=global(0),len=52,received
*Jun  4 09:06:41: %7: IP: s=10.10.2.2 (Serial 2/0), d=224.0.0.9,vrf=global(0),len=52,received
*Jun  4 09:06:43: %7: IP: s=10.10.2.1 (local), d=224.0.0.9 (Serial 2/0),vrf=global(0), g=224.0.0.9,len=72,sent ip pkt to link_layer  222
*Jun  4 09:06:45: %7: IP: s=10.10.1.1 (local), d=224.0.0.9 (GigabitEthernet 0/1),vrf=global(0), g=224.0.0.9,len=72,sent ip pkt to link_layer --> raw send
```

这里反映的是每个端口接收到的包信息和 Wireshark 捕获很像，但是他区分了各个端口，每一行 d

表示源 IP，p 表示目的 IP，vrf 指虚拟路由转发表。

3.3 debug ip rip 与 debug ip packet 的区别

命令	[no]debug ip rip	[no]debug ip packet
功能	显示或不显示发送和接收到的 RIP 路由选择更新	打开或关闭 ip 报文调试开关
命令模式	特用户模式	特权用户模式
使用指南	可以查看路由器使用了 RIP 的 V1 版还是 V2 版本，还有发送和接收的更新信息。	显示接收或发送的 ip 数据包的内容，包括：源地址、目的地址、字节数等。

4. 自动汇总的概念与使用条件

路由汇总是一种优化策略，用于减少路由器必须维护的路由数，它是一种用单个汇总地址代表一系列网络号的方法。

本地 IP 路由的所有子网，在发出去将汇总成一个主类网路。例如 192.168.1.11 与 192.168.2.22 会在边界上汇总成 192.168.0.0 并发布出去。边界指网络地址不同的边界路由器。如 192.168.1.0 与 192.168.2.0 之间的路由器。

当网络不连续时，需要关闭自动汇总。因为如果不关闭自动汇总，如 x.x.x.x/24 与 x.x.x.x/24 被 x.x.x.x/30(我们的实验思考第三问的结构)，不关闭自动汇总会在两个边界汇总成/30 发送出去，这样包含目的节点的网段地址/24 会不确定发到哪个子网，导致路由错误，这也是我们在实验思考修改 IP 后实验观测到的结果。

最后总结路由汇总的使用前提：

1. 多个 IP 地址的最左边几位必须相同，必须是连续网络；
2. 指定的选路协议必须根据 IP 地址和子网掩码出选路决策；
3. 选路决策中路由的更新必须包含 IP 地址和子网掩码。