北京郵電大學

计算机网络技术实践

实验报告： RIP 和 OSPF 路由协议的配置及协议流程分析

学院：计算机学院（国家示范性软件学院）

专业：计算机科学与技术

班级：2022211305

学号：2022211683

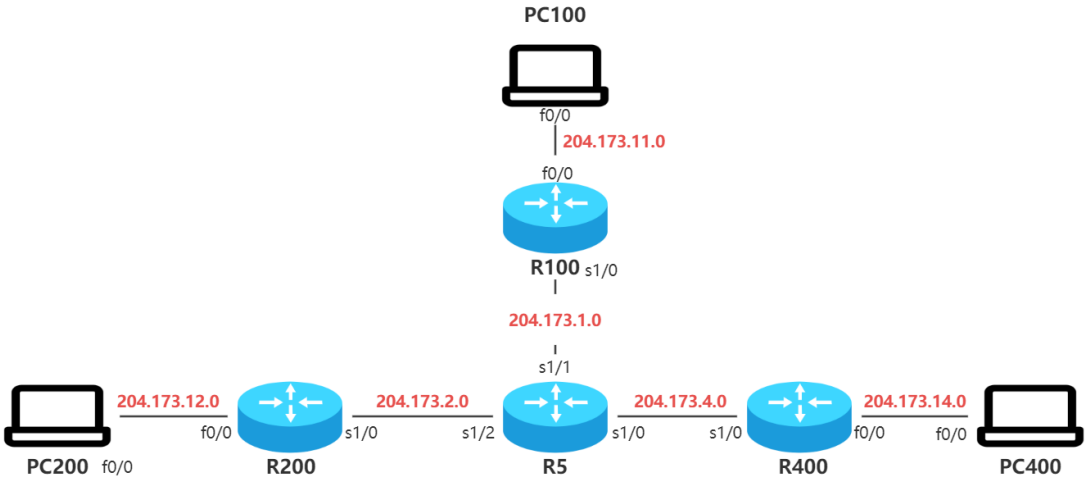
姓名：张晨阳

2024 年 12 月 4 号

目录

1.环境

1. **操作系统**：Windows11
2. **网络平台**：仿真软件 Dynamips
3. **网络拓扑图**：



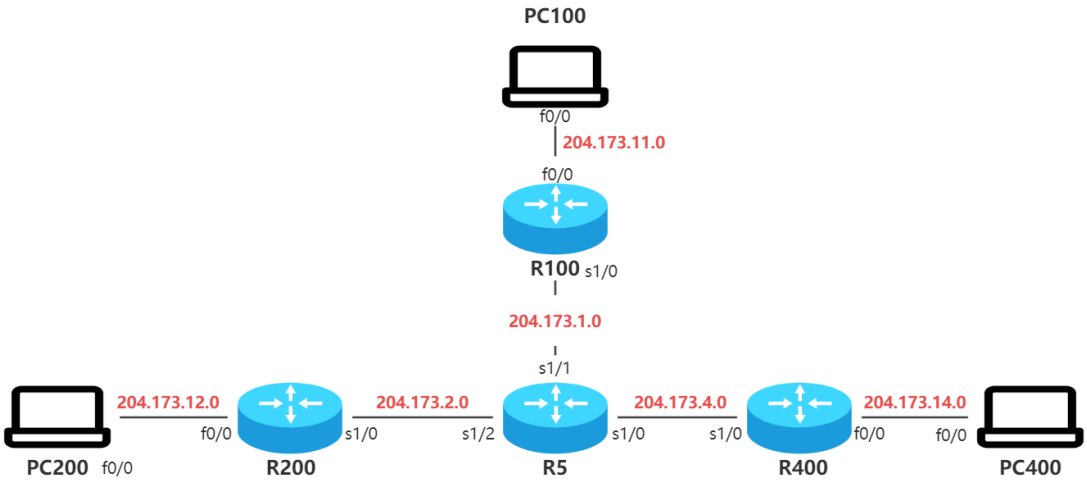
2.实验目的

- 在上一次实验的基础上实现 RIP 和 OSPF 路由协议
- 自己设计网络物理拓扑和逻辑网段，并在其上实现 RIP 和 OSPF 协议
- 通过 debug 信息详细描述 RIP 和 OSPF 协议的工作过程，包括初始信息交互、路由计算、链路故障处理等部分
- RIP 协议中观察没有配置水平分割和配置水平分割后协议的工作流程，和路由消息传递方式
- 观察 OSPF 中数据库同步信息的格式和同步对象以及链路改变信息如何发送

3.实验内容及步骤

3.1. 设计网络拓扑并修改拓扑文件

网络拓扑图设计如下：



由 4 个路由器和 3 台 PC 组成。接口情况以及逻辑网段设计也标在图中。

拓扑文件如下：

```
autostart = False
[localhost]
    port = 7200
    udp = 10000
    workingdir = .\tmp\

    [[router R100]]
    image = ..\ios\unzip-c7200-is-mz.122-37.bin
    model = 7200
    console = 3031
    npe = npe-400
    ram = 64
    confreg = 0x2102
    exec_area = 64
    mmap = false
    slot0 = PA-C7200-IO-FE
    slot1 = PA-4T
    f0/0 = PC100 f0/0
    s1/0 = R5 s1/1
```

```
[[router R200]]
image = ..\ios\unzip-c7200-is-mz.122-37.bin
model = 7200
console = 3032
npe = npe-400
ram = 64
confreg = 0x2102
exec_area = 64
mmap = false
slot0 = PA-C7200-IO-FE
slot1 = PA-4T
f0/0 = PC200 f0/0
s1/0 = R5 s1/2

[[router R400]]
image = ..\ios\unzip-c7200-is-mz.122-37.bin
model = 7200
console = 3034
npe = npe-400
ram = 64
confreg = 0x2102
exec_area = 64
mmap = false
slot0 = PA-C7200-IO-FE
slot1 = PA-4T
f0/0 = PC400 f0/0
s1/0 = R5 s1/0

[[router R5]]
image = ..\ios\unzip-c7200-is-mz.122-37.bin
model = 7200
console = 3035
npe = npe-400
ram = 64
confreg = 0x2102
exec_area = 64
mmap = false
slot0 = PA-C7200-IO-FE
slot1 = PA-4T

[[router PC100]]
    model = 2621
    ram = 20
    image = ..\ios\unzip-c2600-i-mz.121-3.T.bin
```

```
    mmap = False
    confreg = 0x2102
    console = 3036

[[router PC200]]
    model = 2621
    ram = 20
    image = ..\ios\unzip-c2600-i-mz.121-3.T.bin
    mmap = False
    confreg = 0x2102
    console = 3037

[[router PC400]]
    model = 2621
    ram = 20
    image = ..\ios\unzip-c2600-i-mz.121-3.T.bin
    mmap = False
    confreg = 0x2102
    console = 3039
```

3.2. 启动模拟器并指定 idlepc 值

在第一次打开一类机型的设备中的一台的时候，我们需要为其指定其 idlepc 值用于后续的匹配，获取 idlepc 值并将之保存的指令序列如下 (以路由器 R100 为例)：

```
idlepc get R100
idlepc save R100 db
```

其中，idlepc 选择标记*符号的即可。

使用 list 命令查看当前所有的设备：

```
=> list
Name        Type        State       Server           Console
R100        7200        stopped     localhost:7200   3031
R200        7200        stopped     localhost:7200   3032
R400        7200        stopped     localhost:7200   3034
R5          7200        stopped     localhost:7200   3035
PC100       2621        stopped     localhost:7200   3036
PC200       2621        stopped     localhost:7200   3037
PC400       2621        stopped     localhost:7200   3039
```

使用命令 start /all 启动所有设备：

```
=> start /all
100-VM 'R200' started
100-VM 'R5' started
100-VM 'PC200' started
100-VM 'PC100' started
100-VM 'PC400' started
100-VM 'R400' started
100-VM 'R100' started
=> list
Name        Type        State       Server           Console
R100        7200        running     localhost:7200   3031
R200        7200        running     localhost:7200   3032
R400        7200        running     localhost:7200   3034
R5          7200        running     localhost:7200   3035
PC100       2621        running     localhost:7200   3036
PC200       2621        running     localhost:7200   3037
PC400       2621        running     localhost:7200   3039
=>
```

所有设备的状态变为 running。

3.3. 使用 telnet 进行端口配置

对于基本的端口连接配置，内容基本相同，这里以 R100 的配置为例：

```
telnet R100
```

进行 R100 的配置：

- 打开端口
- 在 fastEthernet 口上配置 IP 地址使之能与 PC 机进行通讯
- 在 serial 口上配置 IP 地址以及时钟频率，使之能与别的路由器进行通信
- 在 serial 口上指定数据链路层的协议为 PPP 协议
- 保存配置

```
Router>en
Router#conf
Configuring from terminal, memory, or network [terminal]?
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface f0/0
Router(config-if)#ip add 204.173.11.1 255.255.255.0
Router(config-if)#no shutdown
Router(config-if)#exit
```

```
interface s1/0
ip add 204.173.1.2 255.255.255.0
encapsulation PPP
no shutdown
exit
exit
wr
```

下面再展示 PC 的基本端口配置以及默认静态路由，以 PC100 为例：

```
Router>en
Router#conf
Configuring from terminal, memory, or network [terminal]?
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface f0/0
Router(config-if)#ip add 204.173.11.2 255.255.255.0
Router(config-if)#no shutdown
Router(config-if)#exit
Router(config)#exit
Router#wr
Building configuration...

00:23:22: %SYS-5-CONFIG_I: Configured from console by console[OK]
```

```
Router(config)#ip route 0.0.0.0 0.0.0.0 f0/0
```

其他设备配置同理。

3.4. 配置 RIP 协议

在配置 RIP 协议时，我们需要指定：

1. 协议的版本
2. 路由器连接的网段号
3. 邻居的 IP 地址

这里以 R100 为例：

```
Router(config)#router rip
Router(config-router)#version 2
Router(config-router)#network 204.173.11.0
Router(config-router)#network 204.173.1.0
Router(config-router)#neighbor 204.173.1.1
Router(config-router)#exit
Router(config)#exit
Router#wr
Building configuration...
[OK]
```

依次配置即可。

3.5. 配置 OSPF 协议

配置 OSPF 协议时需要先去除 RIP 协议，然后再进行配置。

对于 OSPF 协议，我们不再需要指定邻居的 IP。因为 OSPF 协议会自动发现邻居。但是在配置 OSPF 协议时，我们还需要额外在路由器之间通信的 serial 口上指定两个延时：一个是发送问候信息的时间间隔 hello−interval，另一个是确认邻居是否还活着的延时 dead−interval。

下面是路由器 R5 的配置过程：

```
Router(config)#no router rip
Router(config)#router ospf 20
Router(config-router)#network 204.173.1.0 255.255.255.0 area 0
Router(config-router)#network 204.173.2.0 255.255.255.0 area 0
Router(config-router)#network 204.173.4.0 255.255.255.0 area 0
Router(config-router)#interface s1/1
Router(config-if)#ip ospf hello-interval 5
Router(config-if)#ip ospf dead-interval 20
Router(config-if)#exit
Router(config)#interface s1/2
```

```
Router(config-if)#ip ospf hello-interval 5
Router(config-if)#ip ospf dead-interval 20
Router(config-if)#exit
Router(config)#interface s1/0
Router(config-if)#ip ospf hello-interval 5
Router(config-if)#ip ospf dead-interval 20
Router(config-if)#exit
Router(config)#exit
Router#wr
00:57:53: %SYS-5-CONFIG_I: Configured from console by console
Building configuration...
[OK]
```

R100 的配置如下：

```
Router(config)#no router rip
Router(config)#router ospf 20
Router(config-router)#network 204.173.1.0 255.255.255.0 area 0
Router(config-router)#network 204.173.11.0 255.255.255.0 area 0
Router(config-router)#exit
Router(config)#interface s1/0
Router(config-if)#ip ospf hello-interval 5
Router(config-if)#
01:02:13: %OSPF-5-ADJCHG: Process 20, Nbr 204.173.4.1 on Serial1/0 from LOADING
to FULL, Loading Done
Router(config-if)#ip ospf dead-interval 20
Router(config-if)#exit
Router(config)#exit
Router#wr
01:02:31: %SYS-5-CONFIG_I: Configured from console by console
Building configuration...
[OK]
Router#
```

其他路由器依次配置即可。

4.实验结果与分析

4.1. RIP 协议

4.1.1.路由表信息

执行 show ip route 命令，查看路由器 R5 的路由表信息如下：

```
Router#show ip route
Codes: C - connected, S - static, I - IGRP, 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, E - EGP
       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

     204.173.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       204.173.1.0/24 is directly connected, Serial1/1
C       204.173.1.2/32 is directly connected, Serial1/1
     204.173.2.0/24 is variably subnetted, 2 subnets, 2 masks
C       204.173.2.2/32 is directly connected, Serial1/2
C       204.173.2.0/24 is directly connected, Serial1/2
     204.173.4.0/24 is variably subnetted, 2 subnets, 2 masks
C       204.173.4.0/24 is directly connected, Serial1/0
C       204.173.4.2/32 is directly connected, Serial1/0
R    204.173.11.0/24 [120/1] via 204.173.1.2, 00:00:07, Serial1/1
R    204.173.12.0/24 [120/1] via 204.173.2.2, 00:00:01, Serial1/2
R    204.173.14.0/24 [120/1] via 204.173.4.2, 00:00:07, Serial1/0
```

与我们的网络拓扑进行对比，发现路由表的结果完全正确。

4.1.2.Debug 信息

未关闭水平分割前，我们在 R5 上抓取了一部分的调试信息如下：

```
Router#debug ip rip
RIP protocol debugging is on
00:04:14: RIP: sending v2 update to 224.0.0.9 via Serial1/0 (204.173.4.1)
00:04:14: RIP: build update entries
00:04:14:      204.173.1.0/24 via 0.0.0.0, metric 1, tag 0
00:04:14:      204.173.2.0/24 via 0.0.0.0, metric 1, tag 0
00:04:14:      204.173.11.0/24 via 0.0.0.0, metric 2, tag 0
00:04:14:      204.173.12.0/24 via 0.0.0.0, metric 2, tag 0
00:04:14: RIP: sending v2 update to 224.0.0.9 via Serial1/1 (204.173.1.1)
00:04:14: RIP: build update entries
00:04:14:      204.173.2.0/24 via 0.0.0.0, metric 1, tag 0
00:04:14:      204.173.4.0/24 via 0.0.0.0, metric 1, tag 0
00:04:14:      204.173.12.0/24 via 0.0.0.0, metric 2, tag 0
00:04:14:      204.173.14.0/24 via 0.0.0.0, metric 2, tag 0
00:04:14: RIP: sending v2 update to 224.0.0.9 via Serial1/2 (204.173.2.1)
00:04:14: RIP: build update entries
00:04:14:      204.173.1.0/24 via 0.0.0.0, metric 1, tag 0
00:04:14:      204.173.4.0/24 via 0.0.0.0, metric 1, tag 0
00:04:14:      204.173.11.0/24 via 0.0.0.0, metric 2, tag 0
00:04:14:      204.173.14.0/24 via 0.0.0.0, metric 2, tag 0
00:04:14: RIP: sending v2 update to 204.173.1.2 via Serial1/1 (204.173.1.1)
00:04:14: RIP: build update entries
00:04:14:      204.173.2.0/24 via 0.0.0.0, metric 1, tag 0
00:04:14:      204.173.4.0/24 via 0.0.0.0, metric 1, tag 0
00:04:14:      204.173.12.0/24 via 0.0.0.0, metric 2, tag 0
00:04:14:      204.173.14.0/24 via 0.0.0.0, metric 2, tag 0
00:04:14: RIP: sending v2 update to 204.173.2.2 via Serial1/2 (204.173.2.1)
00:04:14: RIP: build update entries
00:04:14:      204.173.1.0/24 via 0.0.0.0, metric 1, tag 0
00:04:14:      204.173.4.0/24 via 0.0.0.0, metric 1, tag 0
00:04:14:      204.173.11.0/24 via 0.0.0.0, metric 2, tag 0
00:04:14:      204.173.14.0/24 via 0.0.0.0, metric 2, tag 0
00:04:14: RIP: sending v2 update to 204.173.4.2 via Serial1/0 (204.173.4.1)
00:04:14: RIP: build update entries
00:04:14:      204.173.1.0/24 via 0.0.0.0, metric 1, tag 0
00:04:14:      204.173.2.0/24 via 0.0.0.0, metric 1, tag 0
00:04:14:      204.173.11.0/24 via 0.0.0.0, metric 2, tag 0
00:04:14:      204.173.12.0/24 via 0.0.0.0, metric 2, tag 0
00:04:19: RIP: received v2 update from 204.173.1.2 on Serial1/1
00:04:19:     204.173.11.0/24 via 0.0.0.0 in 1 hops
```

```
00:04:19: RIP: received v2 update from 204.173.1.2 on Serial1/1
00:04:19:      204.173.11.0/24 via 0.0.0.0 in 1 hops
00:04:21: RIP: received v2 update from 204.173.2.2 on Serial1/2
00:04:21:      204.173.12.0/24 via 0.0.0.0 in 1 hops
00:04:21: RIP: received v2 update from 204.173.2.2 on Serial1/2
00:04:21:      204.173.12.0/24 via 0.0.0.0 in 1 hops
```

关闭 R5 s1/1 端口水平分割后的 debug 信息：

```
00:20:32: RIP: sending v2 update to 204.173.1.2 via Serial1/1 (204.173.1.1)
00:20:32: RIP: build update entries
00:20:32:       204.173.1.0/24 via 0.0.0.0, metric 1, tag 0
00:20:32:       204.173.1.2/32 via 0.0.0.0, metric 1, tag 0
00:20:32:       204.173.2.0/24 via 0.0.0.0, metric 1, tag 0
00:20:32:       204.173.4.0/24 via 0.0.0.0, metric 1, tag 0
00:20:32:       204.173.11.0/24 via 0.0.0.0, metric 2, tag 0
00:20:32:       204.173.12.0/24 via 0.0.0.0, metric 2, tag 0
00:20:32:       204.173.14.0/24 via 0.0.0.0, metric 2, tag 0
00:20:32: RIP: sending v2 update to 204.173.2.2 via Serial1/2 (204.173.2.1)
00:20:32: RIP: build update entries
00:20:32:       204.173.1.0/24 via 0.0.0.0, metric 1, tag 0
00:20:32:       204.173.4.0/24 via 0.0.0.0, metric 1, tag 0
00:20:32:       204.173.11.0/24 via 0.0.0.0, metric 2, tag 0
00:20:32:       204.173.14.0/24 via 0.0.0.0, metric 2, tag 0
00:20:32: RIP: sending v2 update to 204.173.4.2 via Serial1/0 (204.173.4.1)
00:20:32: RIP: build update entries
00:20:32:       204.173.1.0/24 via 0.0.0.0, metric 1, tag 0
00:20:32:       204.173.2.0/24 via 0.0.0.0, metric 1, tag 0
00:20:32:       204.173.11.0/24 via 0.0.0.0, metric 2, tag 0
00:20:32:       204.173.12.0/24 via 0.0.0.0, metric 2, tag 0
00:20:43: RIP: received v2 update from 204.173.2.2 on Serial1/2
00:20:43:      204.173.12.0/24 via 0.0.0.0 in 1 hops
00:20:43: RIP: received v2 update from 204.173.2.2 on Serial1/2
00:20:43:      204.173.12.0/24 via 0.0.0.0 in 1 hops
00:20:51: RIP: received v2 update from 204.173.1.2 on Serial1/1
00:20:51:      204.173.11.0/24 via 0.0.0.0 in 1 hops
00:20:51: RIP: received v2 update from 204.173.1.2 on Serial1/1
00:20:51:      204.173.11.0/24 via 0.0.0.0 in 1 hops
00:20:51: RIP: received v2 update from 204.173.4.2 on Serial1/0
00:20:51:      204.173.14.0/24 via 0.0.0.0 in 1 hops
00:20:51: RIP: received v2 update from 204.173.4.2 on Serial1/0
00:20:51:      204.173.14.0/24 via 0.0.0.0 in 1 hops
```

关闭 R5 的 s1/1 端口，记录信息如下：

```
00:36:35: RIP: received v2 update from 204.173.4.2 on Serial1/0
00:36:35:      204.173.1.0/24 via 0.0.0.0 in 16 hops  (inaccessible)
00:36:35:      204.173.11.0/24 via 0.0.0.0 in 16 hops  (inaccessible)
00:36:35:      204.173.14.0/24 via 0.0.0.0 in 1 hops
00:36:46: RIP: received v2 update from 204.173.2.2 on Serial1/2
00:36:46:      204.173.1.0/24 via 0.0.0.0 in 16 hops  (inaccessible)
00:36:46:      204.173.11.0/24 via 0.0.0.0 in 16 hops  (inaccessible)
00:36:46:      204.173.12.0/24 via 0.0.0.0 in 1 hops
00:36:46: RIP: received v2 update from 204.173.2.2 on Serial1/2
00:36:46:      204.173.1.0/24 via 0.0.0.0 in 16 hops  (inaccessible)
00:36:46:      204.173.11.0/24 via 0.0.0.0 in 16 hops  (inaccessible)
00:36:46:      204.173.12.0/24 via 0.0.0.0 in 1 hops
00:36:50: RIP: sending v2 update to 224.0.0.9 via Serial1/0 (204.173.4.1)
00:36:50: RIP: build update entries
00:36:50:       204.173.1.0/24 via 0.0.0.0, metric 16, tag 0
00:36:50:       204.173.2.0/24 via 0.0.0.0, metric 1, tag 0
00:36:50:       204.173.11.0/24 via 0.0.0.0, metric 16, tag 0
00:36:50:       204.173.12.0/24 via 0.0.0.0, metric 2, tag 0
00:36:50: RIP: sending v2 update to 224.0.0.9 via Serial1/2 (204.173.2.1)
00:36:50: RIP: build update entries
00:36:50:       204.173.1.0/24 via 0.0.0.0, metric 16, tag 0
00:36:50:       204.173.4.0/24 via 0.0.0.0, metric 1, tag 0
00:36:50:       204.173.11.0/24 via 0.0.0.0, metric 16, tag 0
00:36:50:       204.173.14.0/24 via 0.0.0.0, metric 2, tag 0
00:36:50: RIP: sending v2 update to 204.173.2.2 via Serial1/2 (204.173.2.1)
00:36:50: RIP: build update entries
00:36:50:       204.173.1.0/24 via 0.0.0.0, metric 16, tag 0
00:36:50:       204.173.4.0/24 via 0.0.0.0, metric 1, tag 0
00:36:50:       204.173.11.0/24 via 0.0.0.0, metric 16, tag 0
00:36:50:       204.173.14.0/24 via 0.0.0.0, metric 2, tag 0
```

4.1.3.RIP 协议工作过程分析

初始信息交互

在 RIP 协议启动时，R5 路由器通过其各个接口向 RIP 组播地址 224.0.0.9 发送 RIP v2 更新信息。这些信息包含了本地网络的路由条目以及其距离其他网络的度量值。下面是 R5 在启动时发送的 RIP 更新信息：

```
00:04:14: RIP: sending v2 update to 224.0.0.9 via Serial1/0 (204.173.4.1)
00:04:14: RIP: build update entries
00:04:14:      204.173.1.0/24 via 0.0.0.0, metric 1, tag 0
00:04:14: RIP: sending v2 update to 224.0.0.9 via Serial1/1 (204.173.1.1)
00:04:14: RIP: build update entries
00:04:14:      204.173.2.0/24 via 0.0.0.0, metric 1, tag 0
00:04:14: RIP: sending v2 update to 224.0.0.9 via Serial1/2 (204.173.2.1)
00:04:14: RIP: build update entries
00:04:14:      204.173.1.0/24 via 0.0.0.0, metric 1, tag 0
```

在这个过程中的具体操作是，R5 通过各个接口（如 Serial1/0, Serial1/1, Serial1/2）向其他路由器和网络广播自己的路由信息。例如，通过接口 Serial1/0，R5 发送路由信息 204.173.1.0/24 via 0.0.0.0, metric 1，表明它知道通过自己直接连接的网络 204.173.1.0/24，距离为 1 跳。

路由计算与更新

RIP 协议采用跳数作为度量值，随着网络拓扑的变化，路由器会更新自己的路由表。在调试信息中，我们看到 R5 定期向邻居发送路由更新信息，这些信息包括了每个目标网络的下一跳地址和跳数。

例如，在 R5 接收到来自其他路由器的更新时，它会根据收到的路由信息来更新其路由表。以下是 R5 接收到更新并处理的示例：

```
00:04:19: RIP: received v2 update from 204.173.1.2 on Serial1/1
00:04:19:      204.173.11.0/24 via 0.0.0.0 in 1 hops
00:04:21: RIP: received v2 update from 204.173.2.2 on Serial1/2
00:04:21:      204.173.12.0/24 via 0.0.0.0 in 1 hops
```

在这个例子中，R5 收到了来自 204.173.1.2 和 204.173.2.2 的更新信息，分别添加了 204.173.11.0/24 和 204.173.12.0/24 的路由条目。R5 现在知道，网络

204.173.11.0/24 的距离是 1 跳，通过地址 0.0.0.0（意味着直接连接或下一跳），同样，网络 204.173.12.0/24 也通过 1 跳到达。

这种更新过程确保了 RIP 协议能够根据邻居路由器提供的信息及时调整自己的路由表，保持网络的连通性。

链路故障处理

当网络拓扑发生变化时，例如某个接口或链路失败，RIP 协议将更新其路由表，可能会增加某些网络的度量值，以反映这些网络不再可达。例如，在关闭 R5 的 Serial1/1 接口后，R5 收到了来自其他路由器的更新信息，其中一些路由的跳数变为了 16（表示不可达）。

```
00:36:35: RIP: received v2 update from 204.173.4.2 on Serial1/0
00:36:35:      204.173.1.0/24 via 0.0.0.0 in 16 hops  (inaccessible)
00:36:35:      204.173.11.0/24 via 0.0.0.0 in 16 hops  (inaccessible)
```

在这个例子中，R5 收到了来自 204.173.4.2 的更新，其中 204.173.1.0/24 和 204.173.11.0/24 的度量值变为 16，这表示这些网络不可达。这是因为 R5 的 Serial1/1 接口已经关闭，导致通过该接口到达这些网络的路径不再有效，RIP 协议将跳数标记为 16，表示不可达。

同时，R5 会根据新的拓扑重新计算路由，选择新的路径来恢复连接。

路由更新广播与最终稳定

在链路故障发生后，RIP 协议会继续向邻居广播更新信息，直到网络收敛并达到稳定状态。例如，在链路故障后，R5 继续向其他路由器发送更新信息，其中不可达的路由（如 204.173.1.0/24 和 204.173.11.0/24）被标记为 16 跳。

```
00:36:50: RIP: sending v2 update to 224.0.0.9 via Serial1/0 (204.173.4.1)
00:36:50: RIP: build update entries
00:36:50:       204.173.1.0/24 via 0.0.0.0, metric 16, tag 0
00:36:50:       204.173.11.0/24 via 0.0.0.0, metric 16, tag 0
```

R5 发送的更新信息中，204.173.1.0/24 和 204.173.11.0/24 的度量值已经变为 16，表示这些网络无法通过当前的路径访问。这是 RIP 协议自动恢复和调整的过程，经过几次更新广播，网络将最终稳定。

4.2. OSPF 协议

4.2.1.邻居信息

配置成功后显示 R5 的 OSPF 邻居信息：

```
Router#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
204.173.14.1      1   FULL/  -        00:00:17    204.173.4.2     Serial1/0
204.173.12.1      1   FULL/  -        00:00:16    204.173.2.2     Serial1/2
204.173.11.1      1   FULL/  -        00:00:16    204.173.1.2     Serial1/1
Router#
```

完全正确。

4.2.2.Debug 信息

我们依然选择在 R5 上抓取信息，首先是正常情况下的信息：

```
Router#debug ip ospf events
OSPF events debugging is on
Router#
01:12:50: OSPF: Rcv hello from 204.173.11.1 area 0 from Serial1/1 204.173.1.2
01:12:50: OSPF: End of hello processing
01:12:51: OSPF: Rcv hello from 204.173.12.1 area 0 from Serial1/2 204.173.2.2
01:12:51: OSPF: End of hello processing
01:12:51: OSPF: Rcv hello from 204.173.14.1 area 0 from Serial1/0 204.173.4.2
01:12:51: OSPF: End of hello processing
01:12:55: OSPF: Rcv hello from 204.173.11.1 area 0 from Serial1/1 204.173.1.2
01:12:55: OSPF: End of hello processing
01:12:56: OSPF: Rcv hello from 204.173.12.1 area 0 from Serial1/2 204.173.2.2
01:12:56: OSPF: End of hello processing
01:12:56: OSPF: Rcv hello from 204.173.14.1 area 0 from Serial1/0 204.173.4.2
01:12:56: OSPF: End of hello processing
01:13:00: OSPF: Rcv hello from 204.173.11.1 area 0 from Serial1/1 204.173.1.2
01:13:00: OSPF: End of hello processing
01:13:01: OSPF: Rcv hello from 204.173.12.1 area 0 from Serial1/2 204.173.2.2
01:13:01: OSPF: End of hello processing
01:13:01: OSPF: Rcv hello from 204.173.14.1 area 0 from Serial1/0 204.173.4.2
01:13:01: OSPF: End of hello processing
01:13:05: OSPF: Rcv hello from 204.173.11.1 area 0 from Serial1/1 204.173.1.2
01:13:05: OSPF: End of hello processing
01:13:06: OSPF: Rcv hello from 204.173.12.1 area 0 from Serial1/2 204.173.2.2
01:13:06: OSPF: End of hello processing
```

```
01:13:06: OSPF: Rcv hello from 204.173.14.1 area 0 from Serial1/0 204.173.4.2
01:13:06: OSPF: End of hello processing
01:13:10: OSPF: Rcv hello from 204.173.11.1 area 0 from Serial1/1 204.173.1.2
01:13:10: OSPF: End of hello processing
01:13:11: OSPF: Rcv hello from 204.173.12.1 area 0 from Serial1/2 204.173.2.2
01:13:11: OSPF: End of hello processing
01:13:11: OSPF: Rcv hello from 204.173.14.1 area 0 from Serial1/0 204.173.4.2
01:13:11: OSPF: End of hello processing
01:13:15: OSPF: Rcv hello from 204.173.11.1 area 0 from Serial1/1 204.173.1.2
01:13:15: OSPF: End of hello processing
01:13:16: OSPF: Rcv hello from 204.173.12.1 area 0 from Serial1/2 204.173.2.2
01:13:16: OSPF: End of hello processing
01:13:16: OSPF: Rcv hello from 204.173.14.1 area 0 from Serial1/0 204.173.4.2
01:13:16: OSPF: End of hello processing
```

之后我们关闭 R5 的 s1/1 端口再次抓取信息：

```
Router(config)#interface s1/1
Router(config-if)#shutdown
Router(config-if)#exit
01:15:38: %OSPF-5-ADJCHG: Process 20, Nbr 204.173.11.1 on Serial1/1 from FULL to
DOWN, Neighbor Down: Interface down or detached
01:15:40: %LINK-5-CHANGED: Interface Serial1/1, changed state to administratively
down
01:15:41: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/1, changed state
to down
Router(config)#exit
01:15:53: %SYS-5-CONFIG_I: Configured
Router#debug ip ospf events
OSPF events debugging is on
Router#
01:16:06: OSPF: Rcv hello from 204.173.12.1 area 0 from Serial1/2 204.173.2.2
01:16:06: OSPF: End of hello processing
01:16:06: OSPF: Rcv hello from 204.173.14.1 area 0 from Serial1/0 204.173.4.2
01:16:06: OSPF: End of hello processing
01:16:11: OSPF: Rcv hello from 204.173.12.1 area 0 from Serial1/2 204.173.2.2
01:16:11: OSPF: End of hello processing
01:16:11: OSPF: Rcv hello from 204.173.14.1 area 0 from Serial1/0 204.173.4.2
01:16:11: OSPF: End of hello processing
01:16:16: OSPF: Rcv hello from 204.173.12.1 area 0 from Serial1/2 204.173.2.2
01:16:16: OSPF: End of hello processing
01:16:16: OSPF: Rcv hello from 204.173.14.1 area 0 from Serial1/0 204.173.4.2
01:16:16: OSPF: End of hello processing
01:16:21: OSPF: Rcv hello from 204.173.12.1 area 0 from Serial1/2 204.173.2.2
01:16:21: OSPF: End of hello processing
```

4.2.3.OSPF 协议过程分析

初始化信息交互（Hello 包交换）

在 OSPF 协议启动时，路由器通过发送和接收 **Hello 包**来发现邻居路由器，并建立邻接关系。以下是 R5 路由器在正常情况下接收到的 Hello 包信息：

```
01:12:50: OSPF: Rcv hello from 204.173.11.1 area 0 from Serial1/1 204.173.1.2
01:12:50: OSPF: End of hello processing
01:12:51: OSPF: Rcv hello from 204.173.12.1 area 0 from Serial1/2 204.173.2.2
01:12:51: OSPF: End of hello processing
01:12:51: OSPF: Rcv hello from 204.173.14.1 area 0 from Serial1/0 204.173.4.2
01:12:51: OSPF: End of hello processing
```

- 在这些信息中，R5 通过三个不同的接口（Serial1/1，Serial1/2，Serial1/0）接收来自不同邻居（204.173.11.1，204.173.12.1，204.173.14.1）的 Hello 包。
- 每个 Hello 包包含信息，如发送 Hello 包的路由器的 IP 地址、接口、区域 ID 等。
- End of hello processing 表示该 Hello 包处理完成，OSPF 协议已经在此接口上成功接收到 Hello 包，并准备进入邻接状态的建立阶段。

这些 Hello 包是 OSPF 邻接形成的第一步，通过定期发送和接收 Hello 包，OSPF 路由器可以确认邻居的存活状态以及双方是否有能力建立邻接关系。

路由计算（LSA 数据库同步）

一旦邻接建立，OSPF 路由器会进行**数据库同步**，即交换 LSA（Link State Advertisement）信息。OSPF 使用 LSA 来传播每个路由器的链路状态信息，允许所有路由器在相同的链路状态数据库（LSDB）中保持一致。

在邻接关系建立后，路由器通过交换不同类型的 LSA（如网络 LSA、路由 LSA）来同步网络中的路由信息。例如，当 R5 接收到来自其他路由器的 LSA 时，它会更新自己的链路状态数据库，并使用 **Dijkstra 算法**重新计算路由表。

在调试中，并未直接看到 LSA 交换的具体内容，但可以推测，在每次 Hello 包的交换后，邻接状态会逐步发展，从 INIT 状态到 2-WAY，然后是 EXSTART、EXCHANGE、LOADING，最终达到 FULL 状态，完成路由信息的同步。

链路故障处理（邻接状态变化）

当链路出现故障时，OSPF 能够通过更新 LSA 并重新计算路由来迅速适应变化。在 R5 关闭 Serial1/1 接口时，OSPF 会处理这个链路故障并更新邻接状态。

从调试信息中可以看到以下内容：

```
01:15:38: %OSPF-5-ADJCHG: Process 20, Nbr 204.173.11.1 on Serial1/1 from FULL to DOWN, Neighbor Down: Interface down or detached
01:15:40: %LINK-5-CHANGED: Interface Serial1/1, changed state to administratively down
01:15:41: %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial1/1, changed state to down
```

- FULL to DOWN 表示 R5 与 204.173.11.1（邻居路由器）之间的邻接关系从“完全邻接”（FULL）状态转变为“DOWN”状态。这通常是因为接口 Serial1/1 被关闭或链路断开。
- 此时，R5 会根据链路故障更新其路由表，并重新计算最优路径。

链路断开后，R5 会继续通过其其他接口接收 Hello 包，确保与其他邻居的邻接关系保持正常，如下所示：

```
01:16:06: OSPF: Rcv hello from 204.173.12.1 area 0 from Serial1/2 204.173.2.2
01:16:06: OSPF: End of hello processing
01:16:06: OSPF: Rcv hello from 204.173.14.1 area 0 from Serial1/0 204.173.4.2
01:16:06: OSPF: End of hello processing
```

在接口故障后，R5 继续接收来自其他接口的 Hello 包，并保持与其他邻居的连接。由于接口故障，Serial1/1 的邻居 204.173.11.1 不会再发送 Hello 包。

OSPF 数据库同步信息的格式和同步对象

OSPF 协议中的数据库同步信息主要是 LSA（Link State Advertisement）。LSA 是路由器用来宣传自己链路状态的消息，主要有以下几种类型：

- **Router LSA**：描述一个路由器直接连接的链路。
- **Network LSA**：描述一个网络中的所有路由器。
- **Summary LSA**：描述不同区域之间的路由。
- **AS External LSA**：描述 OSPF 外部网络的路由信息。

数据库同步的过程通过交换这些 LSA 信息完成，每个路由器会将其本地的

链路状态广告（LSA）广播给其他路由器，从而确保网络中所有路由器的链路状态数据库一致。

链路状态变化的通知和更新

当链路状态发生变化（如接口关闭或链路断开）时，OSPF 会发送 **LSA 更新**，通知邻居更新其链路状态数据库。具体的格式和操作如下：

链路状态变化的 LSA 更新：当接口的状态发生变化时，路由器会生成新的 LSA 并广播。这些 LSA 会传递给邻居，邻居路由器接收到新的 LSA 后，会重新计算其路由表并更新链路状态数据库。

OSPF 链路状态变化时的具体格式会包含以下内容：

- **Link ID**：链路标识符。
- **Link Data**：链路数据。
- **Link Type**：链路类型（如点对点链路、广播链路等）。
- **Metric**：链路的度量值。
- **Neighbor ID**：邻居路由器的 ID。

当链路状态发生变化时，OSPF 协议会根据新的 LSA 重新计算路径，更新路由表，确保路由器始终知道最佳路径。

4.3. PC 端互 Ping

PC100 ping PC200 结果如下：

```
Router#ping 204.173.12.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 204.173.12.2, timeout is 2 seconds:
...!!
Success rate is 40 percent (2/5), round-trip min/avg/max = 116/118/120 ms
```

经测试，均连通。

5.实验中的问题及心得

最初设计的网络结构过于复杂，导致配置过程出现很多莫名其妙的错误，包括环境崩溃，后改为 4 台路由器，3 台 PC。

在进行端口链接的时候，由于不清楚每个 slot 的接口类型、个数，导致许多错误。后来在 PPT 中学习到相关知识，依次修改，解决了不匹配的端口问题。

最初进行 PC 端的互相 ping 时，没有设置默认静态路由，导致一直无法 ping 通，多次重新配置，浪费了很多时间。后来在同学的帮助下发现了这个问题，成功地 ping 通了。

在分析 RIP 协议时，我发现了一个总是出现的地址：**224.0.0.9**。经过查阅资料，我才知道：

在 RIP 协议中，224.0.0.9 是一个组播地址，用于 RIP 版本 2 的路由信息交换。所有运行 RIP 版本 2 的设备都会监听这个地址，这样可以减少对广播域中其他设备的影响。当 RIP 路由设备收到 Request 报文后，会使用 Response 报文进行响应，在该报文中携带对方所请求的路由信息。这种方式可以减少不必要的网络负载，因为只有需要这些信息的设备（即运行 RIP 协议的设备）才会接收这些组播报文。

在对 OSPF 协议进行 debug 的过程中，我产生了这样的问题：为什么打开 debug 信息后只能看到 hello 包？经过查阅资料，我发现：因为 OSPF 的主要协议交互过程是在刚配置完 OSPF 协议时就进行了，而在网络运行过程中，只要没有链路状态的变化就不再交互链路状态信息了。所以要观察 OSPF 协议的工作过程就要先打开 debug 信息，然后再配置 OSPF 协议。

这个过程也让我了解了一些我们的实验不涉及的知识，让我对不同的协议有了更深的了解。

6.实验思考

1. **实验中，采用下一跳和转发接口这两种方式配置 PC1 和 PC2 的静态路由有什么区别？会导致在你的拓扑结构中从 PC1 ping PC2 时的丢包数有什么变化？**

下一跳配置：更适合复杂网络，但首次通信可能由于 ARP 请求导致丢包，尤其是在目标设备未及时响应的情况下。

转发接口配置：更适合简单网络，可以减少首次通信时的丢包，因为不需要额外的 ARP 解析。

这里使用 PC100 和 PC200

```
Router#show arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  204.173.11.2           -   c804.3100.0000  ARPA   FastEthernet0/0
Router#ping 204.173.12.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 204.173.12.2, timeout is 2 seconds:
...!!
Success rate is 40 percent (2/5), round-trip min/avg/max = 116/118/120 ms
Router#show arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  204.173.11.2           -   c804.3100.0000  ARPA   FastEthernet0/0
Internet  204.173.12.2           0   ca00.3100.0000  ARPA   FastEthernet0/0
```

(1) 初始 ARP 表内容

```
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  204.173.11.2           -   c804.3100.0000  ARPA   FastEthernet0/0
```

- 这里显示，R100 仅知道 204.173.11.2 的 MAC 地址（c804.3100.0000），没有 204.173.12.2 的 MAC 地址。
- 在使用下一跳配置的情况下，R100 会尝试通过 ARP 解析 204.173.12.2 的 MAC 地址，但如果下一跳未及时响应，可能导致 ARP 请求失败，从而丢包。

(2) Ping 结果

```
Sending 5, 100-byte ICMP Echos to 204.173.12.2, timeout is 2 seconds:
...!!
Success rate is 40 percent (2/5), round-trip min/avg/max = 116/118/120 ms
```

- R100 尝试 Ping 204.173.12.2，发现丢包率为 60%。
- 在首次 Ping 时，R100 需要通过 ARP 解析 204.173.12.2 的 MAC 地址。如果 ARP 请求未能及时完成（例如，网络延迟或目标设备未能及时响应 ARP 请求），会导致部分 ICMP 包丢失。
- 成功的 2 次 ICMP 回显表明，ARP 解析在一定时间内成功了，但未能完全满足 5 次 Ping 的所有请求。

(3) 更新后的 ARP 表

```
Internet  204.173.11.2           -   c804.3100.0000  ARPA   FastEthernet0/0
Internet  204.173.12.2           0   ca00.3100.0000  ARPA   FastEthernet0/0
```

- 在 Ping 完成后，R100 更新了 ARP 表，成功解析了 204.173.12.2 的 MAC 地址为 ca00.3100.0000。
- 在后续通信中，由于 ARP 解析已经完成，丢包率会降低。

2. 对照所截获的消息，说明 RIP 协议有无水平分割的工作流程；以及 OSPF 协议在点对点网络和广播型网络上工作流程。

Debug 信息和部分说明已记录在 **4. 实验结果与分析**。此处作为补充。

水平分割是一种防止路由环路的机制。它的规则是：通过某个接口学习到的路由信息，不会通过同一接口再次传播。

- **未启用水平分割**：路由器会将通过某个接口学到的路由信息，原封不动地再次发送回该接口。这可能导致路由环路的发生，特别是在多跳网络中。
- **启用水平分割**：路由器会过滤掉通过某接口学习到的路由信息，避免再次通过该接口发送，从而有效防止环路。

例如，从 Serial1/1 收到的路由更新中的网络条目不会包含在从 Serial1/1 发出的更新中。

从调试信息中可以看到：

```
00:04:14: RIP: sending v2 update to 204.173.1.2 via Serial1/1 (204.173.1.1)
00:04:14: RIP: build update entries
00:04:14:      204.173.2.0/24 via 0.0.0.0, metric 1, tag 0
00:04:14:      204.173.4.0/24 via 0.0.0.0, metric 1, tag 0
00:04:14:      204.173.12.0/24 via 0.0.0.0, metric 2, tag 0
00:04:14:      204.173.14.0/24 via 0.0.0.0, metric 2, tag 0
```

这里更新包没有包含通过 Serial1/1 学习到的 204.173.11.0/24 的条目。

这表明水平分割功能成功阻止了环路的形成。

对于 OSPF 协议在两种网络上的工作流程，这里主要补充不同点：

邻居发现与邻接

- **点对点**：直接与唯一邻居建立邻接。
- **广播型**：与所有邻居建立邻接。

路由信息传播

- **点对点**：直接在两端交换 LSA。
- **广播型**：所有路由器直接互相交换 LSA，可能增加网络负载

3. 写出在你的拓扑中，数据包从某台 PC 机 A 发送给其他 PC 机 B 完整过程

我们仍然选择 PC100 ping PC200

在 PC100 端执行下面命令，得到：

```
Router#debug ip packet
IP packet debugging is on
Router#ping 204.173.12.2

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 204.173.12.2, timeout is 2 seconds:
!!!!
00:30:35: IP: s=204.173.11.2 (local), d=204.173.12.2 (FastEthernet0/0), len 100,
 sending
00:30:35: IP: s=204.173.12.2 (FastEthernet0/0), d=204.173.11.2 (FastEthernet0/0)
, len 100, rcvd 3
00:30:35: IP: s=204.173.11.2 (local), d=204.173.12.2 (FastEthernet0/0), len 100,
 sending
00:30:35: IP: s=204.173.12.2 (FastEthernet0/0), d=204.173.11.2 (FastEthernet0/0)
, len 100, rcvd 3
00:30:35: IP: s=204.173.11.2 (local), d=204.173.12.2 (FastEthernet0/0), len 100,
 sending
00:30:36: IP: s=204.173.12.2 (FastEthernet0/0), d=204.173.11.2 (FastEthernet0/0)
, len 100, rcvd 3
00:30:36: IP: s=204.173.11.2 (local), d=204.173.12.2 (FastEthernet0/0), len 100,
 sending!
Success rate is 100 percent (5/5), round-trip min/avg/max = 116/124/132 ms
Router#
00:30:36: IP: s=204.173.12.2 (FastEthernet0/0), d=204.173.11.2 (FastEthernet0/0)
, len 100, rcvd 3
00:30:36: IP: s=204.173.11.2 (local), d=204.173.12.2 (FastEthernet0/0), len 100,
 sending
00:30:36: IP: s=204.173.12.2 (FastEthernet0/0), d=204.173.11.2 (FastEthernet0/0)
, len 100, rcvd 3
Router#
```

在 PC2 端也执行 debug ip packet 命令，得到：

```
Router#debug ip packet
IP packet debugging is on
Router#
00:30:33: IP: s=204.173.11.2 (FastEthernet0/0), d=204.173.12.2 (FastEthernet0/0)
, len 100, rcvd 3
00:30:33: IP: s=204.173.12.2 (local), d=204.173.11.2 (FastEthernet0/0), len 100,
 sending
00:30:33: IP: s=204.173.11.2 (FastEthernet0/0), d=204.173.12.2 (FastEthernet0/0)
, len 100, rcvd 3
00:30:33: IP: s=204.173.12.2 (local), d=204.173.11.2 (FastEthernet0/0), len 100,
 sending
00:30:33: IP: s=204.173.11.2 (FastEthernet0/0), d=204.173.12.2 (FastEthernet0/0)
, len 100, rcvd 3
00:30:33: IP: s=204.173.12.2 (local), d=204.173.11.2 (FastEthernet0/0), len 100,
 sending
00:30:33: IP: s=204.173.11.2 (FastEthernet0/0), d=204.173.12.2 (FastEthernet0/0)
, len 100, rcvd 3
00:30:33: IP: s=204.173.12.2 (local), d=204.173.11.2 (FastEthernet0/0), len 100,
 sending
00:30:34: IP: s=204.173.11.2 (FastEthernet0/0), d=204.173.12.2 (FastEthernet0/0)
, len 100, rcvd 3
00:30:34: IP: s=204.173.12.2 (local), d=204.173.11.2 (FastEthernet0/0), len 100,
 sending
Router#
```

数据包跨越的路径：PC100 → R100 → R5 → R200 → PC200

PC100 生成数据包并发送：

- PC100 执行 ping 204.173.12.2，向 PC200（IP 地址 204.173.12.2）发送 ICMP Echo 请求数据包。
- PC100 检查自身路由表，确定数据包应通过默认网关 R100 转发。

R100 的处理：

- R100 接收到 PC100 的数据包（源地址：204.173.11.2，目的地址：204.173.12.2）。
- 调试信息显示 R100 对数据包的处理：

```
00:30:35: IP: s=204.173.11.2 (local), d=204.173.12.2 (FastEthernet0/0), len 100,
sending
```

R100 使用路由表查找下一跳为 R5，并通过 FastEthernet0/0 接口转发数据包。

R5 的处理：

- R5 接收到从 R100 转发的数据包后，查找路由表，确认下一跳为 R200。
- R5 将数据包通过合适的接口转发给 R200。

R200 的处理：

- R200 接收到从 R5 转发的数据包，查找路由表，发现目的地址 204.173.12.2 属于其直连网络。
- 调试信息显示：

```
00:30:33: IP: s=204.173.11.2 (FastEthernet0/0), d=204.173.12.2 (FastEthernet0/0),
len 100, rcvd 3
```

R200 将数据包通过 FastEthernet0/0 接口转发至 PC200。

PC200 的处理：

- PC200 接收到 ICMP Echo 请求后，生成 ICMP Echo 回复（源地址：204.173.12.2，目的地址：204.173.11.2），并将其发送给默认网关 R200。