

实验三：RIP和OSPF路由协议的配置及协议流程分析

计算机网络技术实践

实验报告

实验名称 RIP和OSPF路由协议的配置及协议流程分析

姓 名 _____

实验日期：11.26

学 号 20222113

实验报告日期：12.22

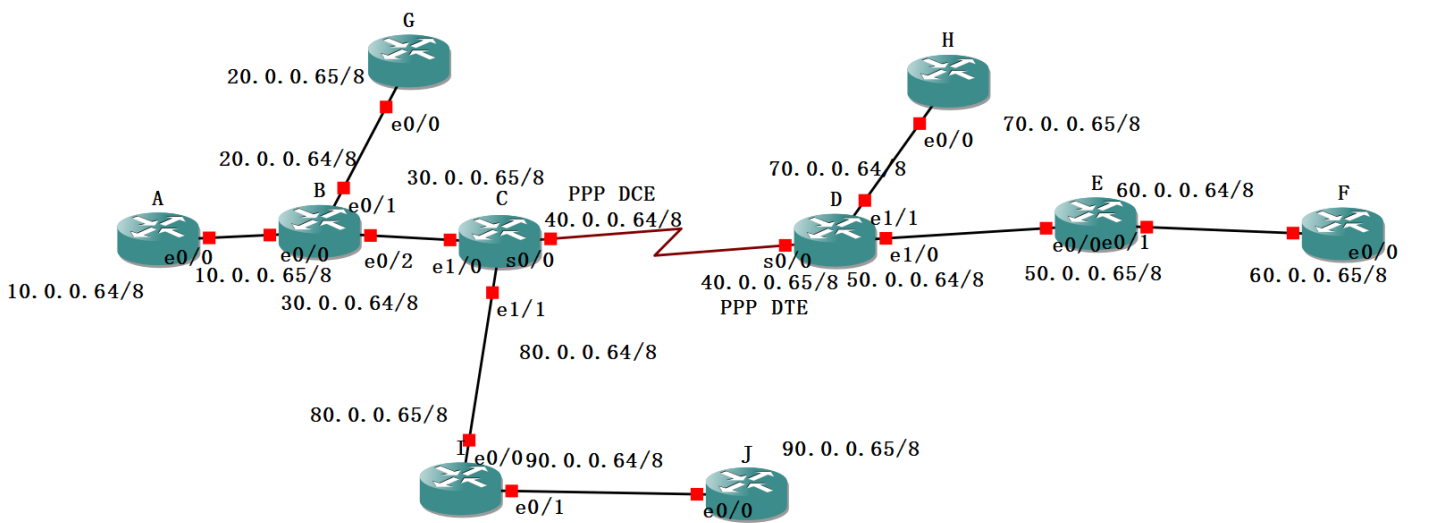
报告退发：

一、实验环境

- 1. 前端GNS3负责图形拓扑设计
- 2. 计算资源为 Windows 11.0 与 VMWare Ubuntu 20.04.6，后端Dynamips仿真环境负责硬件模拟
- 3. 抓包采用Wireshark 4.4.2

二、实验拓扑

拓扑图：



三、实验目的

- 1. 分析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。
- 2. 分析RIP协议水平分割的工作流程和路由信息传递方式。
- 3. 分析OSPF中数据库同步信息的格式和同步对象，链路改变信息的发送方式和格式。

四、实验内容

RIP协议

1.配置各个路由器接口IP地址

完成配置路由器单个接口的IP地址，指令如下

```
1 Router#configure terminal
2 Router(config)#interface [接口名: ex/x]
3 Router(config-interface)#ip address [ip地址: ip] [掩码: mask]
4 Router(config-interface)#no shutdown
5 Router#write
```

可用show命令来检视接口的状态

```
1 Router#show ip interface brief
```

这是对已经配置好的路由器D实行检视的结果

```
D#show ip interface brief
```

Interface	IP-Address	OK?	Method	Status	Protocol
Serial0/0	40.0.0.65	YES	NVRAM	up	up
Serial0/1	unassigned	YES	NVRAM	administratively down	down
Serial0/2	unassigned	YES	NVRAM	administratively down	down
Serial0/3	unassigned	YES	NVRAM	administratively down	down
Ethernet1/0	50.0.0.64	YES	NVRAM	up	up
Ethernet1/1	70.0.0.64	YES	NVRAM	up	up
Ethernet1/2	unassigned	YES	NVRAM	administratively down	down
Ethernet1/3	unassigned	YES	NVRAM	administratively down	down

可以看到，路由器D的IP配置与拓扑图一致

接着我们对所有路由器接口完成类似的配置

2.实现开启水平分割的RIP路由协议

水平分割（Split Horizon）是一种用于防止路由环路的机制，尤其常见于RIP（Routing Information Protocol）这类距离向量路由协议中。其主要作用是通过限制路由信息的传播方向，减少因错误路由信息而导致的网络环路，从而提高网络的稳定性与效率。

具体来说，水平分割规则规定：**一个路由器在接收到来自某个邻居的路由信息后，不会将该信息通过同一接口再发送回该邻居**。这意味着，路由器不会将从一个邻居处学到的路由信息通过该邻居的接口返回，从而有效防止了环路的形成。

我们配置RIP协议，默认会开启水平分割，指令如下

```
1 Router#configure terminal
2 Router(config)#router rip
3 Router(config-router)#version 2
4 Router(config-router)#network [参与RIP的网域: int]
5 Router#write
```

可用show命令来检视接口RIP协议的状态

```
1 show running-config | section router rip
```

这是对已经配置好的路由器D实行检视的结果

```
D#show running-config | section router rip
router rip
version 2
network 40.0.0.0
network 50.0.0.0
network 70.0.0.0
```

接着我们对所有路由器接口完成类似的配置

3.开启debug信息选项来描述RIP工作流程

3.1 RIP协议初始信息交互

我们开启debug信息选项，指令如下

```
1 Router#debug ip rip
```

以边缘路由器G为例，我们可以看到收发的更新路由信息如下

```
RIP protocol debugging is on
G#
*Mar  1 00:07:04.807: RIP:  sending v2 update to 224.0.0.9 via Ethernet0/0 (20
*Mar  1 00:07:04.807: RIP:  build update entries - suppressing null update
G#
*Mar  1 00:07:12.331: RIP:  received v2 update from 20.0.0.64 on Ethernet0/0
*Mar  1 00:07:12.331:          10.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 00:07:12.335:          30.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 00:07:12.335:          40.0.0.0/8 via 0.0.0.0 in 2 hops
*Mar  1 00:07:12.335:          50.0.0.0/8 via 0.0.0.0 in 2 hops
*Mar  1 00:07:12.339:          60.0.0.0/8 via 0.0.0.0 in 2 hops
*Mar  1 00:07:12.339:          70.0.0.0/8 via 0.0.0.0 in 2 hops
*Mar  1 00:07:12.339:          80.0.0.0/8 via 0.0.0.0 in 2 hops
*Mar  1 00:07:12.339:          90.0.0.0/8 via 0.0.0.0 in 2 hops
```

由此我们可以观察到RIP使用广播或多播方式定期发送路由更新

3.2 RIP协议路由计算

使用event带参命令来观察路由表的变化，使用clear命令来清空路由表，指令如下

```
1 Router#debug ip rip events
2 Router#clear ip route *
```

这里选择清空路由器B上的路由，可以看到RIP协议是如何进行路由计算来更新路由表的

```
RIP event debugging is on
B#
B#clear ip route *
B#
*Mar 1 00:09:05.559: RIP: sending request on Ethernet0/0 to 224.0.0.9
*Mar 1 00:09:05.563: rip_route_adjust for Ethernet0/0 coming up
*Mar 1 00:09:05.563: RIP: sending request on Ethernet0/0 to 224.0.0.9
*Mar 1 00:09:05.567: RIP: sending request on Ethernet0/1 to 224.0.0.9
*Mar 1 00:09:05.567: rip_route_adjust for Ethernet0/1 coming up
*Mar 1 00:09:05.567: RIP: sending request on Ethernet0/1 to 224.0.0.9
*Mar 1 00:09:05.571: RIP: sending request on Ethernet0/2 to 224.0.0.9
*Mar 1 00:09:05.571: rip_route_adjust for Ethernet0/2 coming up
*Mar 1 00:09:05.575: RIP: sending request on Ethernet0/2 to 224.0.0.9
*Mar 1 00:09:05.587: RIP: remove Ethernet0/0 from RIP idb list
*Mar 1 00:09:05.591: RIP: remove Ethernet0/1 from RIP idb list
*Mar 1 00:09:05.595: RIP: remove Ethernet0/2 from RIP idb list
*Mar 1 00:09:05.599: RIP: add Ethernet0/0 to RIP idb list
*Mar 1 00:09:05.599: RIP: sending request on Ethernet0/0 to 224.0.0.9
*Mar 1 00:09:05.603: RIP: add Ethernet0/1 to RIP idb list
*Mar 1 00:09:05.603: RIP: sending request on Ethernet0/1 to 224.0.0.9
*Mar 1 00:09:05.603: RIP: add Ethernet0/2 to RIP idb list
*Mar 1 00:09:05.607: RIP: sending request on Ethernet0/2 to 224.0.0.9
*Mar 1 00:09:05.763: RIP: received v2 update from 30.0.0.65 on Ethernet0/2
*Mar 1 00:09:05.775: RIP: Update contains 7 routes
*Mar 1 00:09:05.775: RIP: received v2 update from 30.0.0.65 on Ethernet0/2
*Mar 1 00:09:05.779: RIP: Update contains 7 routes
*Mar 1 00:09:05.779: RIP: received v2 update from 30.0.0.65 on Ethernet0/2
*Mar 1 00:09:05.779: RIP: Update contains 7 routes
*Mar 1 00:09:07.595: RIP: sending v2 flash update to 224.0.0.9 via Ethernet0/0
*Mar 1 00:09:07.599: RIP: Update contains 9 routes
*Mar 1 00:09:07.599: RIP: Update queued
*Mar 1 00:09:07.599: RIP: sending v2 flash update to 224.0.0.9 via Ethernet0/1
*Mar 1 00:09:07.603: RIP: Update contains 9 routes
*Mar 1 00:09:07.603: RIP: Update queued
*Mar 1 00:09:07.603: RIP: sending v2 flash update to 224.0.0.9 via Ethernet0/2
*Mar 1 00:09:07.607: RIP: Update contains 2 routes
*Mar 1 00:09:07.607: RIP: Update queued
*Mar 1 00:09:07.611: RIP: Update sent via Ethernet0/0
*Mar 1 00:09:07.611: RIP: Update sent via Ethernet0/1
*Mar 1 00:09:07.615: RIP: Update sent via Ethernet0/2
B#
```

等待一段时间后，我们可用show命令来检视路由器B各接口的状态

```

B#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 0.0.0.0 to network 0.0.0.0

R    50.0.0.0/8 [120/1] via 30.0.0.65, 00:00:58, Ethernet0/2
R    70.0.0.0/8 [120/1] via 30.0.0.65, 00:00:58, Ethernet0/2
R    80.0.0.0/8 [120/1] via 30.0.0.65, 00:00:58, Ethernet0/2
C    20.0.0.0/8 is directly connected, Ethernet0/1
R    40.0.0.0/8 [120/1] via 30.0.0.65, 00:00:58, Ethernet0/2
C    10.0.0.0/8 is directly connected, Ethernet0/0
R    90.0.0.0/8 [120/1] via 30.0.0.65, 00:00:58, Ethernet0/2
R    60.0.0.0/8 [120/1] via 30.0.0.65, 00:00:58, Ethernet0/2
C    30.0.0.0/8 is directly connected, Ethernet0/2

```

可见路由器B已经完成了路由表的重建

3.3 RIP协议应对故障路由

RIP协议遇到链路故障的话，会不断更新路由信息，尝试搜索合适的通路；而在路由恢复之后，也会更新路由信息

我们首先禁用B-C之间的接口，指令如下

```

1 Router#configure terminal
2 Router(config)#interface e0/2
3 Router(config-if)#shutdown
4 Router#write

```

在路由器B，C上观察debug信息，我们应该能看到B，C各自更新路由的信息

路由器B接口e0/2的路由信息

```

*Mar 1 00:21:23.623: %LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0/2, changed state to down
B(config-if)#
B(config-if)#
*Mar 1 00:21:25.555: RIP: received v2 update from 20.0.0.65 on Ethernet0/1
*Mar 1 00:21:25.555:      30.0.0.0/8 via 0.0.0.0 in 16 hops (inaccessible)
*Mar 1 00:21:25.559:      40.0.0.0/8 via 0.0.0.0 in 16 hops (inaccessible)
*Mar 1 00:21:25.559:      50.0.0.0/8 via 0.0.0.0 in 16 hops (inaccessible)
*Mar 1 00:21:25.559:      60.0.0.0/8 via 0.0.0.0 in 16 hops (inaccessible)
*Mar 1 00:21:25.563:      70.0.0.0/8 via 0.0.0.0 in 16 hops (inaccessible)
*Mar 1 00:21:25.563:      80.0.0.0/8 via 0.0.0.0 in 16 hops (inaccessible)
*Mar 1 00:21:25.563:      90.0.0.0/8 via 0.0.0.0 in 16 hops (inaccessible)
*Mar 1 00:21:25.563: RIP: Update contains 7 routes

```

路由器B稳定后的路由表如下

```
B#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

C    20.0.0.0/8 is directly connected, Ethernet0/1
C    10.0.0.0/8 is directly connected, Ethernet0/0
```

路由器C接口e1/0的路由信息

```
C#
*Mar  1 00:14:16.587: RIP: sending v2 update to 224.0.0.9 via Ethernet1/0 (30.0.0.65)
*Mar  1 00:14:16.587: RIP: build update entries
*Mar  1 00:14:16.587:   0.0.0.0/0 via 0.0.0.0, metric 1, tag 0
*Mar  1 00:14:16.591:   40.0.0.0/8 via 0.0.0.0, metric 1, tag 0
*Mar  1 00:14:16.591:   50.0.0.0/8 via 0.0.0.0, metric 1, tag 0
*Mar  1 00:14:16.591:   60.0.0.0/8 via 0.0.0.0, metric 1, tag 0
*Mar  1 00:14:16.595:   70.0.0.0/8 via 0.0.0.0, metric 1, tag 0
*Mar  1 00:14:16.595:   80.0.0.0/8 via 0.0.0.0, metric 1, tag 0
*Mar  1 00:14:16.595:   90.0.0.0/8 via 0.0.0.0, metric 1, tag 0
```

路由器C稳定后的路由表如下

```
C#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 0.0.0.0 to network 0.0.0.0

S    50.0.0.0/8 [1/0] via 40.0.0.65
      is directly connected, Serial0/0
S    70.0.0.0/8 [1/0] via 40.0.0.65
      is directly connected, Serial0/0
C    80.0.0.0/8 is directly connected, Ethernet1/1
S    20.0.0.0/8 is directly connected, Ethernet1/0
C    40.0.0.0/8 is directly connected, Serial0/0
S    10.0.0.0/8 is directly connected, Ethernet1/0
S    90.0.0.0/8 is directly connected, Ethernet1/1
S    60.0.0.0/8 is directly connected, Serial0/0
C    30.0.0.0/8 is directly connected, Ethernet1/0
```


我们恢复B-C之间的接口，指令如下

```
1 Router#configure terminal
2 Router(config)#interface e0/2
3 Router(config-if)#no shutdown
4 Router#write memory
```

在路由器B，C上观察debug信息，我们应该能看到B，C各自更新路由的信息

路由器B接口e0/2的路由信息

```
B(config-if)#
*Mar  1 00:28:55.019: RIP: sending request on Ethernet0/2 to 224.0.0.9
*Mar  1 00:28:55.023: rip_route_adjust for Ethernet0/2 coming up
*Mar  1 00:28:55.023: RIP: sending request on Ethernet0/2 to 224.0.0.9
*Mar  1 00:28:55.135: RIP: received v2 update from 30.0.0.65 on Ethernet0/2
*Mar  1 00:28:55.139:      0.0.0.0/0 via 0.0.0.0 in 1 hops
*Mar  1 00:28:55.139:      40.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 00:28:55.143:      50.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 00:28:55.143:      60.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 00:28:55.143:      70.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 00:28:55.143:      80.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 00:28:55.143:      90.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 00:28:55.143: RIP: Update contains 7 routes
```

路由器B稳定后的路由表如下

```
B#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 0.0.0.0 to network 0.0.0.0

R    50.0.0.0/8 [120/1] via 30.0.0.65, 00:00:20, Ethernet0/2
R    70.0.0.0/8 [120/1] via 30.0.0.65, 00:00:20, Ethernet0/2
R    80.0.0.0/8 [120/1] via 30.0.0.65, 00:00:20, Ethernet0/2
C    20.0.0.0/8 is directly connected, Ethernet0/1
R    40.0.0.0/8 [120/1] via 30.0.0.65, 00:00:20, Ethernet0/2
C    10.0.0.0/8 is directly connected, Ethernet0/0
R    90.0.0.0/8 [120/1] via 30.0.0.65, 00:00:20, Ethernet0/2
R    60.0.0.0/8 [120/1] via 30.0.0.65, 00:00:20, Ethernet0/2
C    30.0.0.0/8 is directly connected, Ethernet0/2
```

路由器C接口e1/0的路由信息


```
C#
*Mar 1 00:23:35.999: RIP: sending v2 update to 224.0.0.9 via Ethernet1/0 (30.0.0.65)
*Mar 1 00:23:35.999: RIP: build update entries
*Mar 1 00:23:35.999: 0.0.0.0/0 via 0.0.0.0, metric 1, tag 0
*Mar 1 00:23:36.003: 40.0.0.0/8 via 0.0.0.0, metric 1, tag 0
*Mar 1 00:23:36.003: 50.0.0.0/8 via 0.0.0.0, metric 1, tag 0
*Mar 1 00:23:36.003: 60.0.0.0/8 via 0.0.0.0, metric 1, tag 0
*Mar 1 00:23:36.007: 70.0.0.0/8 via 0.0.0.0, metric 1, tag 0
*Mar 1 00:23:36.007: 80.0.0.0/8 via 0.0.0.0, metric 1, tag 0
*Mar 1 00:23:36.007: 90.0.0.0/8 via 0.0.0.0, metric 1, tag 0
```

路由器C稳定后的路由表如下

```
C#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 0.0.0.0 to network 0.0.0.0

S    50.0.0.0/8 [1/0] via 40.0.0.65
      is directly connected, Serial0/0
S    70.0.0.0/8 [1/0] via 40.0.0.65
      is directly connected, Serial0/0
C    80.0.0.0/8 is directly connected, Ethernet1/1
S    20.0.0.0/8 is directly connected, Ethernet1/0
C    40.0.0.0/8 is directly connected, Serial0/0
S    10.0.0.0/8 is directly connected, Ethernet1/0
S    90.0.0.0/8 is directly connected, Ethernet1/1
S    60.0.0.0/8 is directly connected, Serial0/0
C    30.0.0.0/8 is directly connected, Ethernet1/0
```

由此我们可以知道RIP通过一定时间更新路由信息的方式，能够实现对链路故障的检测以及对故障链路恢复的检测

4.观察开启/关闭水平分割后RIP协议工作流程与路由消息传递

4.1 观察开启/关闭水平分割后路由信息传递的变化

我们开启路由器A接口e0/0，以及路由器B接口e0/0的水平分割

使用WIRESHARK抓包A-B的线路，可知路由消息如下

rip						
No.	Destination	Time	Source	Protocol	Length	Info
3	224.0.0.9	8.2968...	10.0.0.65	RIPv2	206	Response
9	224.0.0.9	37.580...	10.0.0.65	RIPv2	206	Response
14	224.0.0.9	66.085...	10.0.0.65	RIPv2	206	Response
21	224.0.0.9	92.868...	10.0.0.65	RIPv2	206	Response

我们关闭路由器A接口e0/0，以及路由器B接口e0/0的水平分割

使用WIRESHARK抓包A-B的线路，可知路由消息如下

128	224.0.0.9	618.37...	10.0.0.65	RIPv2	226	Response
134	224.0.0.9	635.79...	10.0.0.64	RIPv2	226	Response

分析以上两张图我们可以知道：

1. 开启水平分割时，只有从B到A的RIP包，没有从A到B的RIP包；
2. 而关闭水平分割后，两个方向的RIP包都存在。

按照水平分割的定义（上文），由于启用了水平分割，RIP不会通过接收消息的同一接口发送回去。换言之，A不会在接受到B的包修改了路由表之后，又向B发送路由表，以此避免死循环的发生

4.2 观察关闭水平分割后产生回路的路由传递情况

首先我们关闭路由器B接口e0/2，以及路由器C接口e1/0的水平分割，指令如下

```

1 B#
2 Router(config)#interface e0/2
3 Router(config-if)#no ip split-horizon
4 Router(config-if)#no shutdown
5 Router#write
6
7 C#
8 Router(config)#interface e1/0
9 Router(config-if)#no ip split-horizon
10 Router(config-if)#no shutdown
11 Router#write

```

再被动掉路由器B接口e0/2，B无法把e0/0链路断开的信息传递给C

```

1 B#
2 Router(config)#router rip
3 Router(config-router)#passive-interface e0/2

```

关闭路由器B接口e0/0（断开A-B链路）

```
1 B#
2 Router(config)#interface e0/0
3 Router(config-if)shutdown
4 Router#write
```

取消被动路由器B接口e0/2

```
1 B#
2 Router(config)#router rip
3 Router(config-router)#no passive-interface e0/2
```

C无法ping通A，可以判断一定有RIP路由信息一直在B和C之间传送

```
C#ping 10.0.0.64
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.64, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

观察路由器B和路由器C的路由表，可以发现两方都认为A是相通的，判断依据都是对方

```
B#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 0.0.0.0 to network 0.0.0.0

R    50.0.0.0/8 [120/1] via 30.0.0.65, 00:00:05, Ethernet0/2
R    70.0.0.0/8 [120/1] via 30.0.0.65, 00:00:05, Ethernet0/2
R    80.0.0.0/8 [120/1] via 30.0.0.65, 00:00:05, Ethernet0/2
C    20.0.0.0/8 is directly connected, Ethernet0/1
R    40.0.0.0/8 [120/1] via 30.0.0.65, 00:00:05, Ethernet0/2
R    10.0.0.0/8 [120/1] via 30.0.0.65, 00:00:05, Ethernet0/2
R    90.0.0.0/8 [120/1] via 30.0.0.65, 00:00:05, Ethernet0/2
R    60.0.0.0/8 [120/1] via 30.0.0.65, 00:00:05, Ethernet0/2
C    30.0.0.0/8 is directly connected, Ethernet0/2
```

```

C#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 0.0.0.0 to network 0.0.0.0

S    50.0.0.0/8 [1/0] via 40.0.0.65
      is directly connected, Serial0/0
S    70.0.0.0/8 [1/0] via 40.0.0.65
      is directly connected, Serial0/0
C    80.0.0.0/8 is directly connected, Ethernet1/1
S    20.0.0.0/8 is directly connected, Ethernet1/0
C    40.0.0.0/8 is directly connected, Serial0/0
S    10.0.0.0/8 is directly connected, Ethernet1/0
S    90.0.0.0/8 is directly connected, Ethernet1/1
S    60.0.0.0/8 is directly

```

可见路由器B和路由器C之间形成了环路，在用对方的路由信息更新自己

首先A-B链路断开了，A->B=INF

路由器C发送路由信息给路由器B，B:A=3

路由器B发送路由信息给路由器C，C:A=4

路由器C发送路由信息给路由器B，B:A=5

路由器B发送路由信息给路由器C，C:A=6

...

路由器C发送路由信息给路由器B，B:A=MAX-1

路由器B发送路由信息给路由器C，C:A=MAX

如此循环到最后距离达到最大跳数后，B与C判断A不可达，效率十分低下

我们开启B和C各自接口的水平分割，可以看到B用极少的跳数判断完了A不可达并回传给C，可见水平分割对于完成路由表收缩这一方面效果十分显著

```

B#
*Mar  1 01:16:02.659: RIP: received v2 update from 30.0.0.65 on Ethernet0/2
*Mar  1 01:16:02.659:      0.0.0.0/0 via 0.0.0.0 in 1 hops
*Mar  1 01:16:02.663:      40.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 01:16:02.663:      50.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 01:16:02.663:      60.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 01:16:02.663:      70.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 01:16:02.667:      80.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 01:16:02.667:      90.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 01:16:02.667: RIP: Update contains 7 routes

```

OSPF协议

1.实现OSPF协议

我们在OSPF配置之前，需要指定一个进程ID。进程ID是本地路由器内部使用的标识符，在这里我们选择10作为进程ID。

在OSPF配置模式下，声明参与OSPF的网络并指定它们所属的区域。区域通常使用区域编号，我们这里统一选择为0。配置指令如下

```
1 Router#configure terminal
2 Router(config)#router ospf 10
3 Router(config-router)#network [网域:x.x.x.0] [倒置掩码: 0.x.x.x] area [区域编号:int]
4 Router#write
```

可以输入以下指令检视OSPF

```
1 Router#show ip ospf neighbor
```

对路由器C执行上述指令，结果如下

```
C#show ip ospf neighbor
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
70.0.0.64	0	FULL/ -	00:00:15	40.0.0.65	Serial0/0
30.0.0.64	1	FULL/BDR	00:00:34	30.0.0.64	Ethernet1/0
90.0.0.64	1	FULL/DR	00:00:34	80.0.0.65	Ethernet1/1

可用show命令来检视接口的状态

```
C#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

O    50.0.0.0/8 [110/74] via 40.0.0.65, 00:04:08, Serial0/0
O IA 70.0.0.0/8 [110/74] via 40.0.0.65, 00:04:08, Serial0/0
C    80.0.0.0/8 is directly connected, Ethernet1/1
O IA 20.0.0.0/8 [110/20] via 30.0.0.64, 00:04:08, Ethernet1/0
C    40.0.0.0/8 is directly connected, Serial0/0
O    10.0.0.0/8 [110/20] via 30.0.0.64, 00:04:08, Ethernet1/0
O    60.0.0.0/8 [110/84] via 40.0.0.65, 00:04:08, Serial0/0
C    30.0.0.0/8 is directly connected, Ethernet1/0
```

2.开启debug信息选项来描述OSPF工作流程

2.1 OSPF协议初始信息交互

打开所有ospf类型的debug信息，指令如下

```
1 Router#debug ip ospf hello
2 Router#debug ip ospf events
3 Router#debug ip ospf packet
4 Router#debug ip ospf lsa
```

这里我们重启路由器A的接口e0/0，观察ospf协议的路由信息

```
1 Router(router)#interface e1/1
2 Router(router-if)#shutdown
3 Router(router-if)#no shutdown
```

可以看到，路由器A和B之间交换了Hello包，建立了邻居关系；之后会一直交换链路状态信息，保持链路状态数据库信息一致

```
A(config-if)#no shut
A(config-if)#
*Mar 1 00:09:09.843: %LINK-5-CHANGED: Interface Ethernet0/0, changed state to administratively down
*Mar 1 00:09:10.843: %LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0/0, changed state to down
*Mar 1 00:09:11.031: OSPF: Interface Ethernet0/0 going Up
*Mar 1 00:09:11.035: OSPF: Send hello to 224.0.0.5 area 0 on Ethernet0/0 from 10.0.0.64
*Mar 1 00:09:11.047: OSPF: rcv. v:2 t:1 l:48 rid:30.0.0.64
aid:0.0.0.0 chk:B9D9 aut:0 auk: from Ethernet0/0
*Mar 1 00:09:11.051: OSPF: Rcv hello from 30.0.0.64 area 0 from Ethernet0/0 10.0.0.65
*Mar 1 00:09:11.051: OSPF: 2 Way Communication to 30.0.0.64 on Ethernet0/0, state 2WAY
*Mar 1 00:09:11.051: OSPF: Backup seen Event before WAIT timer on Ethernet0/0
*Mar 1 00:09:11.055: OSPF: DR/BDR election on Ethernet0/0
*Mar 1 00:09:11.055: OSPF: Elect BDR 10.0.0.64
*Mar 1 00:09:11.055: OSPF: Elect DR 30.0.0.64
*Mar 1 00:09:11.055: OSPF: Elect BDR 10.0.0.64
*Mar 1 00:09:11.055: OSPF: Elect DR 30.0.0.64
*Mar 1 00:09:11.055: DR: 30.0.0.64 (Id) BDR: 10.0.0.64 (Id)
*Mar 1 00:09:11.055: OSPF: Send DBD to 30.0.0.64 on Ethernet0/0 seq 0x326 opt 0x52 flag 0x7 len 32
*Mar 1 00:09:11.055: OSPF: Send immediate hello to nbr 30.0.0.64, src address 10.0.0.65, on Ethernet0/0
*Mar 1 00:09:11.055: OSPF: Send hello to 10.0.0.65 area 0 on Ethernet0/0 from 10.0.0.64
*Mar 1 00:09:11.055: OSPF: End of hello processing
```

2.2 OSPF协议路由计算

OSPF协议交换LSA包来建立链路状态数据库，路由器则根据本地数据库计算最短路径

使用debug信息显示指令，可以看到各个路由器是如何进行最短路径计算的


```
1 Router#debug ip ospf spf
```

重启路由器A的接口e0/0之后，可以看到OSPF的路由更新信息，路由器会在本地保留路由拓扑，使用dijkstra算法计算出到达各ip的最短路径

```
*Mar 1 00:14:45.991: OSPF: putting LSA on the clist LSID 30.0.0.64, Type 1, Adv Rtr. 30.0.0.64
*Mar 1 00:14:45.991:   Add path: next-hop 10.0.0.65, interface Ethernet0/0
*Mar 1 00:14:45.991:   Processing router id 10.0.0.64
*Mar 1 00:14:45.991:   New newdist 10 olddist 0
*Mar 1 00:14:45.991: OSPF: downheap LSA on the clist LSID 30.0.0.64, Type 1, Adv Rtr. 30.0.0.64,
                        from index 1 to index 1
*Mar 1 00:14:45.995: OSPF: Add ABR Router Route to 30.0.0.64 via 10.0.0.65. Metric: 10. Area 0
*Mar 1 00:14:45.995: OSPF: Schedule SPF in area 0
                        Change in LS ID 30.0.0.64, LSA type SN, , spf-type Prefix Recalculation
*Mar 1 00:14:45.995: OSPF: insert route list LS ID 30.0.0.64, type 1, adv rtr 30.0.0.64
*Mar 1 00:14:45.995:   It is a router LSA 30.0.0.64. Link Count 2
*Mar 1 00:14:45.995:   Processing link 0, id 30.0.0.65, link data 30.0.0.64, type 2
*Mar 1 00:14:45.995:   Add better path to LSA ID 30.0.0.65, gateway 30.0.0.64, dist 20
*Mar 1 00:14:45.995: OSPF: putting LSA on the clist LSID 30.0.0.65, Type 2, Adv Rtr. 80.0.0.64
```

2.3 OSPF协议链路故障处理

禁用路由器B-C的接口，同RIP协议部分一样，营造链路故障与恢复，观察OSPF协议反馈

```
1 Router#configure terminal
2 Router(config)#interface e0/2
3 Router(config-if)#shutdown
4 Router#write
```

首先观察路由器B，C的debug信息，OSPF会检测到链路故障，并开始更新路由信息

路由器B的路由信息如下

```
B(config-if)#shut
B(config-if)#
*Mar 1 01:24:40.947: OSPF: Interface Ethernet0/2 going Down
*Mar 1 01:24:40.947: OSPF: Neighbor change Event on interface Ethernet0/2
*Mar 1 01:24:40.947: OSPF: DR/BDR election on Ethernet0/2
*Mar 1 01:24:40.947: OSPF: Elect BDR 0.0.0.0
*Mar 1 01:24:40.947: OSPF: Elect DR 80.0.0.64
*Mar 1 01:24:40.947: OSPF: Elect BDR 0.0.0.0
*Mar 1 01:24:40.947: OSPF: Elect DR 80.0.0.64
*Mar 1 01:24:40.947:   DR: 80.0.0.64 (Id)   BDR: none
*Mar 1 01:24:40.947: %OSPF-5-ADJCHG: Process 10, Nbr 80.0.0.64 on Ethernet0/2 from
own or detached
B(config-if)#
*Mar 1 01:24:40.947: OSPF: Neighbor change Event on interface Ethernet0/2
*Mar 1 01:24:40.947: OSPF: DR/BDR election on Ethernet0/2
*Mar 1 01:24:40.947: OSPF: Elect BDR 0.0.0.0
*Mar 1 01:24:40.947: OSPF: Elect DR 0.0.0.0
*Mar 1 01:24:40.947:   DR: none   BDR: none
*Mar 1 01:24:40.947: OSPF: Remember old DR 80.0.0.64 (id)
```

路由器B的路由表如下

```

B#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

C    20.0.0.0/8 is directly connected, Ethernet0/1
C    10.0.0.0/8 is directly connected, Ethernet0/0

```

路由器C的路由信息如下

```

*Mar  1 01:27:51.035: OSPF: 30.0.0.64 address 30.0.0.64 on Ethernet1/0 is dead
*Mar  1 01:27:51.035: %OSPF-5-ADJCHG: Process 10, Nbr 30.0.0.64 on Ethernet1/0 from FULL
expired
C#
*Mar  1 01:27:51.035: OSPF: Neighbor change Event on interface Ethernet1/0
*Mar  1 01:27:51.035: OSPF: DR/BDR election on Ethernet1/0
*Mar  1 01:27:51.039: OSPF: Elect BDR 0.0.0.0
*Mar  1 01:27:51.039: OSPF: Elect DR 80.0.0.64
*Mar  1 01:27:51.039:      DR: 80.0.0.64 (Id)    BDR: none
C#
*Mar  1 01:27:54.059: OSPF: rcv. v:2 t:5 l:64 rid:70.0.0.64
aid:0.0.0.0 chk:CDFE aut:0 auk: from Serial0/0
C#
*Mar  1 01:27:55.627: OSPF: rcv. v:2 t:1 l:48 rid:70.0.0.64
aid:0.0.0.0 chk:5633 aut:0 auk: from Serial0/0
*Mar  1 01:27:55.627: OSPF: Rcv hello from 70.0.0.64 area 0 from Serial0/0 40.0.0.65
*Mar  1 01:27:55.631: OSPF: End of hello processing

```

路由器C的路由表如下

```

C#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

O    50.0.0.0/8 [110/74] via 40.0.0.65, 00:07:55, Serial0/0
O IA 70.0.0.0/8 [110/74] via 40.0.0.65, 00:07:55, Serial0/0
C    80.0.0.0/8 is directly connected, Ethernet1/1
C    40.0.0.0/8 is directly connected, Serial0/0
O    60.0.0.0/8 [110/84] via 40.0.0.65, 00:07:55, Serial0/0
C    30.0.0.0/8 is directly connected, Ethernet1/0

```

OSPF协议将路由拓扑保存在了本地，再用dijkstra算法来计算最短路径，不会出现像RIP一样的循环

五、实验分析

1. RIP与OSPF协议消息传递方式与消息内容对比

1.1 RIP协议

邻居发现：利用定期RIP response消息发现

路由信息更新：采用触发性路由更新与定期路由更新

链路故障：检测到故障/恢复，会直接更新路由表且提醒邻居

首先关闭A-B接口，指令如下，等到C判断A不可达后，启动A-B接口，观察C如何和B交互

```
1 Router#configure terminal
2 Router(config)#interface e0/0
3 Router(config-if)#shutdown
4 Router#write
```

路由器B在链路断开后，向C发送A不可达的路由信息，抓包如下

```
> Frame 10: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface -, id 0
> Ethernet II, Src: cc:08:23:38:00:02 (cc:08:23:38:00:02), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
> Internet Protocol Version 4, Src: 30.0.0.64, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
> Routing Information Protocol
  Command: Response (2)
  Version: RIPv2 (2)
  > IP Address: 10.0.0.0, Metric: 16
```

路由器C的路由表如下所示

```
C#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 0.0.0.0 to network 0.0.0.0

S    50.0.0.0/8 [1/0] via 40.0.0.65
      is directly connected, Serial0/0
S    70.0.0.0/8 [1/0] via 40.0.0.65
      is directly connected, Serial0/0
C    80.0.0.0/8 is directly connected, Ethernet1/1
R    20.0.0.0/8 [120/1] via 30.0.0.64, 00:00:08, Ethernet1/0
C    40.0.0.0/8 is directly connected, Serial0/0
R    90.0.0.0/8 [120/1] via 80.0.0.65, 00:00:30, Ethernet1/1
R    60.0.0.0/8 [120/2] via 40.0.0.65, 00:00:29, Serial0/0
C    30.0.0.0/8 is directly connected, Ethernet1/0
```

这个时候路由器B只和路由器G接通，而水平机制确保了路由器B不会给C发送来自C的路由信息，那么B发给C的路由表只包含路由器G的地址，可以对B-C链路抓包得到RIP报文内容如下

```

> Frame 3: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface -, id 0
> Ethernet II, Src: cc:08:23:38:00:02 (cc:08:23:38:00:02), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
> Internet Protocol Version 4, Src: 30.0.0.64, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
▼ Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
    ▼ IP Address: 20.0.0.0, Metric: 1
        Address Family: IP (2)
        Route Tag: 0
        IP Address: 20.0.0.0
        Netmask: 255.0.0.0
        Next Hop: 0.0.0.0
        Metric: 1

```

关闭路由器B接口e0/2上的水平分割，B发给C的报文中就有B的全部路由表，如下图所示

```

> Frame 26: 186 bytes on wire (1488 bits), 186 bytes captured (1488 bits) on interface -, id 0
> Ethernet II, Src: cc:08:23:38:00:02 (cc:08:23:38:00:02), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
> Internet Protocol Version 4, Src: 30.0.0.64, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
▼ Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
    > IP Address: 20.0.0.0, Metric: 1
    > IP Address: 40.0.0.0, Metric: 16
    > IP Address: 50.0.0.0, Metric: 16
    > IP Address: 60.0.0.0, Metric: 16
    > IP Address: 70.0.0.0, Metric: 16
    > IP Address: 80.0.0.0, Metric: 16
    > IP Address: 90.0.0.0, Metric: 16

```

接下来我们开启A-B的链路，指令如下

```

1 Router#(config)#interface e0/0
2 Router#(config-if)#no shutdown
3 Router#write

```

B会立即检测到链路恢复，给C发送一个仅包含变化的路由报文，如下

```

> Frame 63: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface -, id 0
> Ethernet II, Src: cc:08:23:38:00:02 (cc:08:23:38:00:02), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
> Internet Protocol Version 4, Src: 30.0.0.64, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
▼ Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
    ▼ IP Address: 10.0.0.0, Metric: 1
        Address Family: IP (2)
        Route Tag: 0
        IP Address: 10.0.0.0
        Netmask: 255.0.0.0
        Next Hop: 0.0.0.0
        Metric: 1

```

C收到B发送的发现路由器A接通的报文之后，更新路由表如下

```
C#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 0.0.0.0 to network 0.0.0.0

S    50.0.0.0/8 [1/0] via 40.0.0.65
      is directly connected, Serial0/0
S    70.0.0.0/8 [1/0] via 40.0.0.65
      is directly connected, Serial0/0
C    80.0.0.0/8 is directly connected, Ethernet1/1
R    20.0.0.0/8 [120/1] via 30.0.0.64, 00:00:09, Ethernet1/0
C    40.0.0.0/8 is directly connected, Serial0/0
R    10.0.0.0/8 [120/1] via 30.0.0.64, 00:00:09, Ethernet1/0
R    90.0.0.0/8 [120/1] via 80.0.0.65, 00:00:05, Ethernet1/1
C    30.0.0.0/8 is directly connected, Ethernet1/0
```

1.2 OSPF协议

邻居发现：利用Hello包进行发现

路由信息更新：采用链路状态算法与交换链路状态广播LSA更新，进一步同步链路状态数据库

链路故障：检测到故障/恢复，会直接更新路由表且提醒邻居

我们首先分析一下B-C链路之间定期发送的Hello包，可以看到邻居信息与信息有效期，如下

```
> Frame 2: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface -, id 0
> Ethernet II, Src: cc:01:14:2f:00:10 (cc:01:14:2f:00:10), Dst: IPv4mcast_05 (01:00:5e:00:00:05)
> Internet Protocol Version 4, Src: 30.0.0.65, Dst: 224.0.0.5
▼ Open Shortest Path First
  > OSPF Header
  ▼ OSPF Hello Packet
    Network Mask: 255.0.0.0
    Hello Interval [sec]: 10
    ▼ Options: 0x12, (L) LLS Data block, (E) External Routing
      0... .... = DN: Not set
      .0.. .... = (O) Opaque: Not set
      ..0. .... = (DC) Demand Circuits: Not supported
      ...1 .... = (L) LLS Data block: Present
      .... 0... = (N) NSSA: Not supported
      .... .0.. = (MC) Multicast: Not capable
      .... ..1. = (E) External Routing: Capable
      .... ...0 = (MT) Multi-Topology Routing: No
    Router Priority: 1
    Router Dead Interval [sec]: 40
    Designated Router: 30.0.0.65
    Backup Designated Router: 0.0.0.0
  > OSPF LLS Data Block
```


我们接着重启B-C链接，观察一下完整的OSPF过程

53	30.0.0.65	190.37...	30.0.0.64	OSPF	78	DB Description
54	30.0.0.65	190.37...	30.0.0.64	OSPF	94	Hello Packet
55	30.0.0.64	190.38...	30.0.0.65	OSPF	78	DB Description
56	30.0.0.65	190.38...	30.0.0.64	OSPF	158	DB Description
57	30.0.0.64	190.39...	30.0.0.65	OSPF	238	DB Description
58	30.0.0.65	190.39...	30.0.0.64	OSPF	78	DB Description
59	30.0.0.64	190.40...	30.0.0.65	OSPF	78	DB Description
60	30.0.0.64	190.40...	30.0.0.65	OSPF	106	LS Request
61	30.0.0.65	190.40...	30.0.0.64	OSPF	154	LS Request
62	30.0.0.65	190.40...	30.0.0.64	OSPF	78	DB Description
63	30.0.0.65	190.40...	30.0.0.64	OSPF	206	LS Update
64	30.0.0.64	190.41...	30.0.0.65	OSPF	386	LS Update
65	224.0.0.5	190.87...	30.0.0.64	OSPF	110	LS Update
66	224.0.0.5	190.92...	30.0.0.65	OSPF	122	LS Update
67	224.0.0.5	190.96...	30.0.0.65	OSPF	94	LS Update
68	224.0.0.5	191.25...	30.0.0.64	OSPF	94	Hello Packet
70	224.0.0.5	192.92...	30.0.0.65	OSPF	158	LS Acknowledge
71	224.0.0.5	192.93...	30.0.0.64	OSPF	258	LS Acknowledge
73	224.0.0.5	200.36...	30.0.0.65	OSPF	94	Hello Packet
74	224.0.0.5	201.25...	30.0.0.64	OSPF	94	Hello Packet
77	224.0.0.5	210.35...	30.0.0.65	OSPF	94	Hello Packet

接下来我们分别看一下这些OSPF协议包的结构和报文

1. Database Description(DD)

本路由包描述链路状态数据库的摘要信息，用于邻居同步本地链路数据库

- DD包数据部分

字段	描述	长度（字节）
OSPF头部	包含通用的OSPF头部信息，如版本号、类型、长度等	24
接口ID	标识发送报文的接口	4
选项标志	用于记录OSPF选项，表示接口功能与支持的特性	1
标志	包含多个标志位，如初始化（Init）、更多（More）等	1
摘要项数量	指示摘要项（Summary）的数量	1

- DD包Summary部分

字段	描述	长度（字节）
LS类型	链路状态类型，如Router LSAs、Network LSAs等	1
LS ID	LSA标识符	4
广播路由ID	链路状态广告发出者的唯一标识符	4

序列号	用于确定LSA的最新版本	4
校验和	用于错误检测的校验和	2
长度	每个摘要项的总长度	16

以下是DD包的报文内容，包括了LSA（链路状态广告）类型报文的摘要

```
> Frame 56: 158 bytes on wire (1264 bits), 158 bytes captured (1264 bits) on interface -, id 0
> Ethernet II, Src: cc:08:23:38:00:02 (cc:08:23:38:00:02), Dst: cc:01:14:2f:00:10 (cc:01:14:2f:00:10)
> Internet Protocol Version 4, Src: 30.0.0.64, Dst: 30.0.0.65
▼ Open Shortest Path First
  > OSPF Header
  > OSPF DB Description
  ▼ LSA-type 1 (Router-LSA), len 36
    .000 0001 0010 0110 = LS Age (seconds): 294
    0... .. = Do Not Age Flag: 0
    > Options: 0x22, (DC) Demand Circuits, (E) External Routing
    LS Type: Router-LSA (1)
    Link State ID: 10.0.0.64
    Advertising Router: 10.0.0.64
    Sequence Number: 0x80000002
    Checksum: 0x708e
    Length: 36
  > LSA-type 1 (Router-LSA), len 48
  > LSA-type 2 (Network-LSA), len 32
  > LSA-type 3 (Summary-LSA (IP network)), len 28
  > OSPF LLS Data Block
```

2. Link State Request(LSR)

本路由包是包含多个LSA必要信息的请求包，向其他路由器请求更新其链路状态数据库

- LSR包数据部分

字段	描述	长度（字节）
OSPF头部	包含通用的OSPF头部信息，如版本号、类型、长度等	24
请求项数量	指示请求项（Request）的数量	2
请求项内容	记录不同的LSA标识信息	变长

- LSR包请求项部分

字段	描述	长度（字节）
LS类型	链路状态类型，如Router LSAs、Network LSAs等	1

LS ID	LSA标识符	4
广播路由ID	链路状态广告发出者的唯一标识符	4

以下是LSR包的报文内容，包含了非常简单的LS信息

```
> Frame 61: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits) on interface -, id 0
> Ethernet II, Src: cc:08:23:38:00:02 (cc:08:23:38:00:02), Dst: cc:01:14:2f:00:10 (cc:01:14:2f
> Internet Protocol Version 4, Src: 30.0.0.64, Dst: 30.0.0.65
▼ Open Shortest Path First
  > OSPF Header
  ▼ Link State Request
    LS Type: Router-LSA (1)
    Link State ID: 60.0.0.64
    Advertising Router: 60.0.0.64
  ▼ Link State Request
    LS Type: Router-LSA (1)
    Link State ID: 60.0.0.65
    Advertising Router: 60.0.0.65
  ▼ Link State Request
    LS Type: Router-LSA (1)
    Link State ID: 70.0.0.64
    Advertising Router: 70.0.0.64
  > Link State Request
  > Link State Request
  > Link State Request
  > Link State Request
```

3. Link State Update(LSU)

本路由包包含多个LSA，更新链路状态数据库

- LSU包数据部分

字段	描述	长度（字节）
OSPF头部	包含通用的OSPF头部信息，如版本号、类型、长度等	24
LSA头部信息	记录LSA的类型，接口ID等基本信息	变长
LSA内容信息	根据类型不同，记录不同的LSA广播	变长

- LSA头部

字段	描述	长度（字节）
LS类型	链路状态类型，如Router LSA=1、Network LSA=2等	1
选项标志	用于记录OSPF选项，表示接口功能与支持的特性	1

LS ID	LSA标识符	4
广播路由ID	链路状态广告发出者的唯一标识符	4
序列号	用于确定LSA的最新版本	4
LS寿命	LSA的生存时间，单位是s	2
校验和	用于错误检测的校验和	2
长度	每个摘要项的总长度	16

• LSA内容

LSA 类型	描述	作用	传播范围
Router-LSA (Type 1)	描述路由器及其直接连接的链路	描述路由器接口及其连接的网络信息	仅限当前区域
Network-LSA (Type 2)	描述广播和NBMA多访问网络中的所有路由器	描述多访问网络中的路由器拓扑	网络中的所有路由器
Summary-LSA (Type 3)	由ABR生成，描述其他区域的网络信息	描述其他区域的网络信息，进行路由汇总	跨区域传播
AS-External-LSA (Type 5)	描述来自外部自治系统的外部路由信息	描述外部路由信息，通常由ASBR引入OSPF网络	跨整个OSPF网络
NSSA External-LSA (Type 7)	描述NSSA区域中的外部路由信息	描述NSSA区域中的外部路由信息，由NSSA中的ASBR生成	仅限NSSA区域

重启B-C链路后，路由器B会向C发送LSU包，其中包含了所有B已知的路由拓扑

```
> Frame 64: 386 bytes on wire (3088 bits), 386 bytes captured (3088 bits) on interface -, id 0
> Ethernet II, Src: cc:01:14:2f:00:10 (cc:01:14:2f:00:10), Dst: cc:08:23:38:00:02 (cc:08:23:38:00:02)
> Internet Protocol Version 4, Src: 30.0.0.65, Dst: 30.0.0.64
  Open Shortest Path First
    OSPF Header
    LS Update Packet
      Number of LSAs: 8
      LSA-type 1 (Router-LSA), len 48
      LSA-type 1 (Router-LSA), len 36
      LSA-type 1 (Router-LSA), len 60
      LSA-type 1 (Router-LSA), len 60
      LSA-type 2 (Network-LSA), len 32
      LSA-type 2 (Network-LSA), len 32
      LSA-type 3 (Summary-LSA (IP network)), len 28
      LSA-type 3 (Summary-LSA (IP network)), len 28
```

这是其中一个LSA条目中的路由信息

```

    ▾ LSA-type 1 (Router-LSA), len 48
      .000 0001 0010 1000 = LS Age (seconds): 296
      0... .... .... = Do Not Age Flag: 0
      > Options: 0x22, (DC) Demand Circuits, (E) External Routing
      LS Type: Router-LSA (1)
      Link State ID: 60.0.0.64
      Advertising Router: 60.0.0.64
      Sequence Number: 0x80000003
      Checksum: 0x89ac
      Length: 48
      > Flags: 0x00
      Number of Links: 2
      > Type: Transit ID: 60.0.0.65      Data: 60.0.0.64      Metric: 10
      > Type: Transit ID: 50.0.0.64      Data: 50.0.0.65      Metric: 10

```

4. Link State Acknowledge(LSAck)

本路由包意味着确认收到特定的LSA，确保LSA的接收状态

- LSAck包数据部分

字段	描述	长度（字节）
OSPF头部	包含通用的OSPF头部信息，如版本号、类型、长度等	24
LSA数量信息	存储确认LSA的数量	2
LSA内容信息	记录不同的LSA标识信息	变长

- LSAck包Entry部分

字段	描述	长度（字节）
LS类型	链路状态类型，如Router LSAs、Network LSAs等	1
LS ID	LSA标识符	4
广播路由ID	链路状态广告发出者的唯一标识符	4

路由器C在接收到B发送的LSR后，向B发送LSAck信息

```

> Frame 71: 258 bytes on wire (2064 bits), 258 bytes captured (2064 bits) on interface -, id 0
> Ethernet II, Src: cc:08:23:38:00:02 (cc:08:23:38:00:02), Dst: IPv4mcast_05 (01:00:5e:00:00:05)
> Internet Protocol Version 4, Src: 30.0.0.64, Dst: 224.0.0.5
▼ Open Shortest Path First
  > OSPF Header
  > LSA-type 1 (Router-LSA), len 48
  > LSA-type 1 (Router-LSA), len 36
  > LSA-type 1 (Router-LSA), len 60
  > LSA-type 1 (Router-LSA), len 60
  > LSA-type 2 (Network-LSA), len 32
  > LSA-type 2 (Network-LSA), len 32
  > LSA-type 3 (Summary-LSA (IP network)), len 28
  > LSA-type 3 (Summary-LSA (IP network)), len 28
  > LSA-type 1 (Router-LSA), len 60
  > LSA-type 2 (Network-LSA), len 32

```

以下是LSAck某一项的报文内容，包含了非常简单的LS信息

```

▼ LSA-type 1 (Router-LSA), len 48
  .000 0001 0010 1000 = LS Age (seconds): 296
  0... .... .... .... = Do Not Age Flag: 0
  > Options: 0x22, (DC) Demand Circuits, (E) External Routing
  LS Type: Router-LSA (1)
  Link State ID: 60.0.0.64
  Advertising Router: 60.0.0.64
  Sequence Number: 0x80000003
  Checksum: 0x89ac
  Length: 48

```

在这一系列的OSPF LS包传输完成后，本地运行dijkstra算法得出最短路径记录为最优路由表

1.3 RIP与OSPF协议对比分析

方面	RIP	OSPF
邻居发现	<ul style="list-style-type: none"> - 定期发送更新来发现和维护路由 - 使用RIP协议的Request和Response - 使用224.0.0.9多播地址实现传输 	<ul style="list-style-type: none"> - 使用Hello包进行邻居发现和维护 - Hello包用于交换信息，内容包括区域ID、传输间隔以及寿命等 - 使用224.0.0.5/224.0.0.6的多播地址实现传输
数据库信息同步	<ul style="list-style-type: none"> - 基于距离向量算法，每个路由器通过广播路由表更新与邻居共享其路由信息 - 使用简单的RIP Request和RIP Response包进行沟通 - 路由表的更新是全局的，且是周期性的，不过数据库同步速度较慢 	<ul style="list-style-type: none"> - 基于链路状态算法，路由器发送LSA（链路状态广告）来同步链路状态数据库 - 先用DD包交换摘要信息，再用LSR发起请求，传输LSU包，最后用LSAck确认接收，再在本地更新最短路径 - LSA传播后，每个路由器都会重新计算最短路径，更新路由信息

消息传输	<ul style="list-style-type: none">- 采用周期性的路由更新消息广播（默认30秒），通过UDP端口520进行通信- 路由信息通过广播或单播传递给邻居，但是缺乏Ack，传输可靠性较差	<ul style="list-style-type: none">- 使用链路状态广告（LSA）来传递链路状态信息，LSA通过OSPF路由协议的多播地址进行传输- 支持广播、点对点以及多播网络，使用LSAck包确认已经接收的LSA
------	---	---

2.数据包从某PC发送到其他PC的过程

我们在RIP协议实现的拓扑上，使用ping命令的ICMP报文完成这一环节

2.1 两台直接相连机器(B-C)的ping

路由器B和C的地址状态如下

```
1 B-ip:30.0.0.64/8
2 C-ip:30.0.0.65/8
3 B-MAC:cc:08:23:38:00:02
4 C-MAC:cc:01:14:2f:00:10
```

ARP过程

目的：将目标IP地址解析为目标MAC地址，以便在同一局域网内进行通信。

步骤：

1. 通信需求产生

- 路由器B执行 `ping 30.0.0.65`，向路由器C发送数据包。

2. 路由表查找

- 路由器B检查其路由表，发现目标IP地址 `30.0.0.65` 属于本地网络，那么不使用网关转发。

3. ARP缓存检查

- 路由器B查看其ARP缓存表，查找IP地址 `30.0.0.65` 对应的MAC地址。
- 如果缓存表中存在对应的条目（例如，`00:11:22:33:44:55`），则直接使用该MAC地址发送数据。
- 如果缓存中没有对应条目，路由器B需要发送ARP请求。

4. 发送ARP请求

- 路由器B生成一个ARP请求帧，内容如下：
 - **目标IP地址：** `30.0.0.65`

- **源MAC地址：** AA:BB:CC:DD:EE:FF （路由器B的MAC地址）
- **目标MAC地址：** FF:FF:FF:FF:FF:FF （广播地址）

◦ ARP请求通过链路层的广播方式，发送到同一局域网内的所有设备。

5. ARP响应

- 路由器C接收到ARP请求，识别出目标IP地址 30.0.0.65 与自身的IP地址匹配。
- 路由器C生成一个ARP响应帧，内容如下：
 - **目标IP地址：** 30.0.0.65
 - **源MAC地址：** 00:11:22:33:44:55 （路由器C的MAC地址）
 - **目标MAC地址：** AA:BB:CC:DD:EE:FF （路由器B的MAC地址）
- ARP响应通过链路层直接发送给路由器B。

6. 更新ARP缓存

- 路由器B接收到ARP响应后，更新其ARP缓存表，将 30.0.0.65 对应的MAC地址设置为 AA:BB:CC:DD:EE:FF 。
- 路由器B使用该MAC地址作为目标MAC地址，封装数据包并发送给路由器C。

观察ARP过程：

通过执行 `clear arp-cache` 命令可以清除ARP缓存，然后再次进行 `ping` 测试，观察ARP过程的重新启动，包括ARP请求和ARP响应的交换。

344	cc:08:23:38:00:02	1118.4...	cc:01:14:2f:00:10	ARP	60	Who has 30.0.0.64? Tell 30.0.0.65
345	Broadcast	1118.4...	cc:01:14:2f:00:10	ARP	60	Gratuitous ARP for 30.0.0.65 (Reply)
346	cc:01:14:2f:00:10	1118.4...	cc:08:23:38:00:02	ARP	60	30.0.0.64 is at cc:08:23:38:00:02
347	cc:08:23:38:00:02	1121.3...	cc:08:23:38:00:02	LOOP	60	Reply
348	cc:01:14:2f:00:10	1126.3...	cc:01:14:2f:00:10	LOOP	60	Reply
349	224.0.0.9	1127.3...	30.0.0.65	RIPv2	186	Response
350	224.0.0.9	1128.1...	30.0.0.64	RIPv2	86	Response
351	cc:08:23:38:00:02	1131.6...	cc:08:23:38:00:02	LOOP	60	Reply
352	cc:01:14:2f:00:10	1136.5...	cc:01:14:2f:00:10	LOOP	60	Reply

```
▼ Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: cc:01:14:2f:00:10 (cc:01:14:2f:00:10)
  Sender IP address: 30.0.0.65
  Target MAC address: cc:08:23:38:00:02 (cc:08:23:38:00:02)
  Target IP address: 30.0.0.64
```

- ▼ Address Resolution Protocol (reply/gratuitous ARP)
 - Hardware type: Ethernet (1)
 - Protocol type: IPv4 (0x0800)
 - Hardware size: 6
 - Protocol size: 4
 - Opcode: reply (2)
 - [Is gratuitous: True]
 - Sender MAC address: cc:01:14:2f:00:10 (cc:01:14:2f:00:10)
 - Sender IP address: 30.0.0.65
 - Target MAC address: Broadcast (ff:ff:ff:ff:ff:ff)
 - Target IP address: 30.0.0.65
- ▼ Address Resolution Protocol (reply)
 - Hardware type: Ethernet (1)
 - Protocol type: IPv4 (0x0800)
 - Hardware size: 6
 - Protocol size: 4
 - Opcode: reply (2)
 - Sender MAC address: cc:08:23:38:00:02 (cc:08:23:38:00:02)
 - Sender IP address: 30.0.0.64
 - Target MAC address: cc:01:14:2f:00:10 (cc:01:14:2f:00:10)
 - Target IP address: 30.0.0.65

经过三次握手之后，路由器B得知了C的MAC地址，能够进行正常的RIP路由更新了

链路层协议封装过程

目的：将网络层的数据包封装到数据链路层的帧中，以便通过物理介质进行传输。

步骤：

1. 数据包生成

- 路由器B在网络层生成一个IP数据包，其中：
 - 源IP地址为 30.0.0.64
 - 目标IP地址为 30.0.0.65

2. 链路层封装

- **确定目标MAC地址：**通过先前的ARP过程，路由器B已经获取了目标IP 30.0.0.65 对应的MAC地址（例如， 00:11:22:33:44:55 ）。
- **创建以太网帧：**
 - **目标MAC地址：** 00:11:22:33:44:55 （路由器C的MAC地址）
 - **源MAC地址：** AA:BB:CC:DD:EE:FF （路由器B的MAC地址）
 - **类型字段：** 0x0800 （表示上层协议为IPv4）
 - **数据字段：** 封装IP数据包

3. 发送数据

- 路由器B通过其网络接口（如 `e0/2`）发送已经封装好的以太网帧。

4. 接收数据

- 路由器C接收到以太网帧后，首先检查目标MAC地址是否与自身的MAC地址匹配。
- 如果匹配，路由器C解封装以太网帧，并提取其中的IP数据包进行进一步处理。

5. 数据包处理

- 路由器C在网络层检查目标IP地址 `30.0.0.65` 发现目标设备是自己。
- 路由器C处理ICMP Echo Request，并生成一个ICMP Echo Reply响应。
- 路由器C重复上述封装过程，将ICMP Echo Reply通过链路层重新封装成以太网帧，然后发送回路由器B。

以太网帧报文格式如下

字段	长度 (字节)	示例值	描述	备注
目的MAC地址	6	AA:BB:CC:DD:EE:FF	目标设备的物理地址	通常用于标识数据包的接收者
源MAC地址	6	00:11:22:33:44:55	数据包发送设备的物理地址	表示发送者的网络接口的地址
类型字段	2	0x0800 (IP)	数据帧中封装的数据类型	0x0800表示IP协议
数据字段	可变	IP数据包	具体承载的网络层或其他协议数据	长度根据帧的最大传输单元而变化

我们抓包ICMP协议的ping包，得到这一时刻的情况如下

486	30.0.0.65	1588.3...	30.0.0.64	ICMP	114 Echo (ping) request	id=0x0000, seq=0/0, ttl=255 (reply in 487)
487	30.0.0.64	1588.3...	30.0.0.65	ICMP	114 Echo (ping) reply	id=0x0000, seq=0/0, ttl=255 (request in 486)
488	30.0.0.65	1588.4...	30.0.0.64	ICMP	114 Echo (ping) request	id=0x0000, seq=1/256, ttl=255 (reply in 489)
489	30.0.0.64	1588.4...	30.0.0.65	ICMP	114 Echo (ping) reply	id=0x0000, seq=1/256, ttl=255 (request in 488)

观察具体报文内容，与我们上方分析到的信息完全匹配

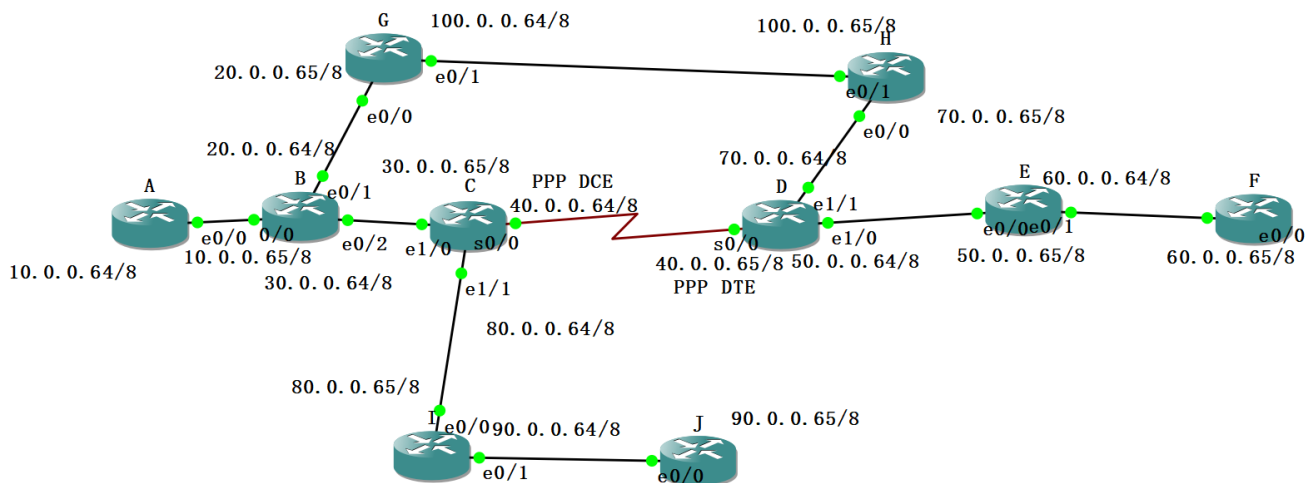
```

> Frame 486: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface -, id 0
▼ Ethernet II, Src: cc:08:23:38:00:02 (cc:08:23:38:00:02), Dst: cc:01:14:2f:00:10 (cc:01:14:2f:00:10)
  > Destination: cc:01:14:2f:00:10 (cc:01:14:2f:00:10)
  > Source: cc:08:23:38:00:02 (cc:08:23:38:00:02)
  Type: IPv4 (0x0800)
  [Stream index: 8]
▼ Internet Protocol Version 4, Src: 30.0.0.64, Dst: 30.0.0.65
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 100
  Identification: 0x0000 (0)
  > 000. .... = Flags: 0x0
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 255
  Protocol: ICMP (1)
  Header Checksum: 0x7f18 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 30.0.0.64
  Destination Address: 30.0.0.65
  [Stream index: 2]
> Internet Control Message Protocol

```

2.2 两台拓扑相连机器的ping

我们选择两台相距较远的路由器直接拓扑相连，形成环路，观察RIP协议如何寻找B---H的最优路径



我们完成G和H的配置之后，观察路由器B的路由表如下

```

B#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 0.0.0.0 to network 0.0.0.0

R    50.0.0.0/8 [120/1] via 30.0.0.65, 00:00:03, Ethernet0/2
R   100.0.0.0/8 [120/1] via 20.0.0.65, 00:00:26, Ethernet0/1
R    70.0.0.0/8 [120/1] via 30.0.0.65, 00:00:03, Ethernet0/2
R    80.0.0.0/8 [120/1] via 30.0.0.65, 00:00:03, Ethernet0/2
C   20.0.0.0/8 is directly connected, Ethernet0/1
R   40.0.0.0/8 [120/1] via 30.0.0.65, 00:00:03, Ethernet0/2
C   10.0.0.0/8 is directly connected, Ethernet0/0
R   90.0.0.0/8 [120/2] via 30.0.0.65, 00:00:03, Ethernet0/2
R   60.0.0.0/8 [120/3] via 30.0.0.65, 00:00:05, Ethernet0/2
C   30.0.0.0/8 is directly connected, Ethernet0/2

```

观察路由器G的路由表如下图

```

G#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 0.0.0.0 to network 0.0.0.0

R   50.0.0.0/8 [120/2] via 100.0.0.65, 00:00:00, Ethernet0/1
                [120/2] via 20.0.0.64, 00:00:10, Ethernet0/0
C  100.0.0.0/8 is directly connected, Ethernet0/1
R   70.0.0.0/8 [120/1] via 100.0.0.65, 00:00:00, Ethernet0/1
R   80.0.0.0/8 [120/2] via 20.0.0.64, 00:00:10, Ethernet0/0
C   20.0.0.0/8 is directly connected, Ethernet0/0
R   40.0.0.0/8 [120/2] via 100.0.0.65, 00:00:01, Ethernet0/1
                [120/2] via 20.0.0.64, 00:00:10, Ethernet0/0
R   10.0.0.0/8 [120/1] via 20.0.0.64, 00:00:13, Ethernet0/0
R   90.0.0.0/8 [120/3] via 20.0.0.64, 00:00:13, Ethernet0/0
R   60.0.0.0/8 [120/3] via 100.0.0.65, 00:00:03, Ethernet0/1
R   30.0.0.0/8 [120/1] via 20.0.0.64, 00:00:13, Ethernet0/0

```

路由操作详解

1. 路由表查找

1. 路由器B收到一个目的IP地址为 **100.0.0.65** 的数据包。为了决定下一步如何转发数据包，路由器B会查找其内部路由表，寻找与目标IP地址 **100.0.0.65** 最匹配的路由条目。
2. 路由表中，路由器B发现有一条与目标地址最匹配的路由条目：**100.0.0.0/8**。此路由条目表示目标地址位于网络 **100.0.0.0** 中，掩码为 **/8**，即匹配前8位的IP地址。
3. 根据这条路由信息，路由器B确定数据包的下一跳为直接连接的路由器G，转发接口为 **e0/0**。

2. 确定下一跳

1. 路由器B通过路由表查询和匹配，确定下一跳地址为 **20.0.0.65**，这是路由器G的接口地址。
2. 接着，路由器B将数据包发送到 **20.0.0.65**。在这个过程中，路由器B需要将数据链路层的目的MAC地址设置为路由器G的MAC地址，确保数据链路层的正确转发。
3. 此外，路由器B在转发数据包时，也会检查TTL（生存时间）字段，并将其减1。如果TTL减少到0，数据包将被丢弃，避免无限循环。

3. 选择最短路径

1. 路由协议RIP使用跳数作为衡量路径优劣的主要依据。跳数表示从源地址到目的地址所经过的路由器数量，跳数越少，路径越优。
2. 在当前网络中，路由器B到达路由器H的最短路径是通过路由器G直接到达H（路径为 **B-G-H**）。此路径的跳数为1，因此路由器B选择通过路由器G转发数据包。
3. 尽管网络中存在其他路径（如 **B-G-H-D-C-B**），但RIP协议通过水平分割、毒性逆转等防环路机制，避免了数据包在环路中循环转发，从而保障了网络的稳定性和效率。

4. 到达下一跳

1. 数据包到达下一跳路由器G后，路由器G会重复路由器B的路由查找和转发逻辑。
2. 路由器G的路由表中显示，它与路由器H直接相连，因此可以直接将数据包通过对应接口转发到路由器H，而无需额外的路由计算。
3. 如果路由器G未与目标路由器直接相连，则会重新进入路由查找流程，按照路由表中的最优匹配继续转发数据包，直到数据包到达最终目的地。

5. 数据包的最终到达

1. 数据包最终到达路由器H后，路由器H会根据IP地址进一步确定数据包的目标主机。
2. 如果数据包的目标地址属于H的直接连接子网，路由器H会直接将数据包通过网络接口转发到目标主机。如果目标地址不存在或不可达，路由器H会向源路由器发送ICMP不可达消息，通知数据包无法到达目标。

ARP过程

场景描述

路由器B需要向路由器H发送数据包，经过路由表查找可知，数据包会经过两个网络域 **20.0.0.0/8** 和 **100.0.0.0/8**。在此过程中，每个网络域内部需要通过ARP解析下一跳设备的MAC地址。具体过程如下：

1. 在20.0.0.0/8域中的ARP过程

- 路由器B需要通过接口e1/0将数据包发送到下一跳路由器G。
- 路由器B检查ARP缓存表，查找目标IP地址 **20.0.0.65** 是否存在对应的MAC地址。如果未找到，路由器B会发送ARP请求广播，询问“谁是20.0.0.65”。
- 路由器G收到ARP请求后，返回其MAC地址（例如 **AA:BB:CC:DD:EE:FF**）。
- 路由器B将该MAC地址记录到ARP缓存表中，并将数据包封装为以太网帧，通过接口e1/0发送到路由器G。

2. 在100.0.0.0/8域中的ARP过程

- 数据包到达路由器G后，路由器G需要将数据包转发到路由器H。
- 路由器G检查ARP缓存表，查找目标IP地址 **100.0.0.65** 是否存在对应的MAC地址。如果未找到，路由器G会发送ARP请求广播，询问“谁是100.0.0.65”。
- 路由器H收到ARP请求后，返回其MAC地址（例如 **00:11:22:33:44:55**）。
- 路由器G将该MAC地址记录到ARP缓存表中，并将数据包封装为以太网帧，发送到路由器H。

链路层协议封装过程

目标描述

链路层协议封装的目标是将网络层的数据包封装为链路层帧，并通过物理链路进行传输。封装过程中，以太网帧的内容会根据每一跳的传输链路进行动态更新。具体步骤如下：

1. 以太网帧的基本构成

- **目的MAC地址**：目标设备的MAC地址（下一跳设备）。
- **源MAC地址**：当前发送设备的MAC地址。
- **类型字段**：标识封装数据的协议类型，例如 **0x0800** 表示IPv4数据包。
- **数据字段**：封装的网络层IP数据包。
- **帧校验序列（FCS）**：用于校验帧数据是否完整。

2. B到G的以太网帧封装

```
▼ Ethernet II, Src: cc:08:23:38:00:01 (cc:08:23:38:00:01), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
  > Destination: IPv4mcast_09 (01:00:5e:00:00:09)
  > Source: cc:08:23:38:00:01 (cc:08:23:38:00:01)
  Type: IPv4 (0x0800)
```

3. G到H的以太网帧封装

```
▼ Ethernet II, Src: cc:0a:23:7c:00:01 (cc:0a:23:7c:00:01), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
  > Destination: IPv4mcast_09 (01:00:5e:00:00:09)
  > Source: cc:0a:23:7c:00:01 (cc:0a:23:7c:00:01)
  Type: IPv4 (0x0800)
```

IP包

在IP包中，必须包含发送端的IP地址和接收端的IP地址。路由器在数据包传输过程中会解析IP包的网络层内容，根据目的IP地址查找本地路由表以决定下一跳。IP包的内容在传输过程中有以下特点：

1. 固定的部分：

除了TTL（存活时间）和Checksum（校验和）外，IP包的其他内容在整个传输过程中不会发生变化。例如，发送端IP地址和接收端IP地址在数据包从源到目的的过程中始终保持不变。

2. 动态更新的部分：

- **TTL**: 每经过一个路由器, TTL值会减1, 用于防止数据包在网络中无限循环。如果TTL值减为0, 数据包将被丢弃, 并返回ICMP超时报文给源设备。
- **Checksum**: 每次TTL发生变化后, 路由器会重新计算校验和, 确保数据包的完整性和准确性。

3. B到G的IP帧封装

```

▼ Internet Protocol Version 4, Src: 20.0.0.64, Dst: 224.0.0.9
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
    Total Length: 192
    Identification: 0x0000 (0)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
  > Time to Live: 2
    Protocol: UDP (17)
    Header Checksum: 0xc324 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 20.0.0.64
    Destination Address: 224.0.0.9
    [Stream index: 1]

```

4. G到H的IP帧封装

```

▼ Internet Protocol Version 4, Src: 100.0.0.64, Dst: 224.0.0.9
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
    Total Length: 132
    Identification: 0x0000 (0)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
  > Time to Live: 2
    Protocol: UDP (17)
    Header Checksum: 0x7360 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 100.0.0.64
    Destination Address: 224.0.0.9
    [Stream index: 0]

```

5. B到H的IP包走最短路径

这是B-G的ping包抓包

1218	20.0.0.64	4058.4...	100.0.0.65	ICMP	114 Echo (ping) reply	id=0x0003, seq=0/0, ttl=254
1219	20.0.0.64	4058.4...	100.0.0.65	ICMP	114 Echo (ping) reply	id=0x0003, seq=1/256, ttl=254
1220	20.0.0.64	4058.5...	100.0.0.65	ICMP	114 Echo (ping) reply	id=0x0003, seq=2/512, ttl=254
1221	20.0.0.64	4058.5...	100.0.0.65	ICMP	114 Echo (ping) reply	id=0x0003, seq=3/768, ttl=254
1222	20.0.0.64	4058.5...	100.0.0.65	ICMP	114 Echo (ping) reply	id=0x0003, seq=4/1024, ttl=254

这是B-C的ping包抓包

icmp						
No.	Destination	Time	Source	Protocol	Length	Info

可知数据包从最短路径B-G-H而非B-C-D-H路径传输

2.3 两台最短路径故障，仍有其他路径机器的ping

接下来我们把路由器G接口e0/1关闭，模拟G-H链路故障

此时B向H发送的数据包，应该从路径B-C-D-H发送

B-G链路抓包如下，G已经检测到e0/1接口关闭，通知B路由器H已经不可达，也会通知通过H到达路由器D、E、F的路径断掉

```
> Frame 7: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface -, id 0
> Ethernet II, Src: cc:0a:23:7c:00:00 (cc:0a:23:7c:00:00), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
> Internet Protocol Version 4, Src: 20.0.0.65, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
▼ Routing Information Protocol
  Command: Response (2)
  Version: RIPv2 (2)
  > IP Address: 60.0.0.0, Metric: 16
  > IP Address: 70.0.0.0, Metric: 16
  > IP Address: 100.0.0.0, Metric: 16
```

B获得路由器G发送的报文之后，因为只有H是通过B-G-H产生的最短路径，而路由器D、E、F还可以通过C产生

最短路径，所以路由器D、E、F仍然可达

故路由器B向G发送RIP报文，表示路由器D、E、F可以通过B达到

```
> Frame 12: 226 bytes on wire (1808 bits), 226 bytes captured (1808 bits) on interface -, id 0
> Ethernet II, Src: cc:08:23:38:00:01 (cc:08:23:38:00:01), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
> Internet Protocol Version 4, Src: 20.0.0.64, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
▼ Routing Information Protocol
  Command: Response (2)
  Version: RIPv2 (2)
  > IP Address: 10.0.0.0, Metric: 1
  > IP Address: 30.0.0.0, Metric: 1
  > IP Address: 40.0.0.0, Metric: 2
  > IP Address: 50.0.0.0, Metric: 2
  > IP Address: 60.0.0.0, Metric: 4
  > IP Address: 70.0.0.0, Metric: 2
  > IP Address: 80.0.0.0, Metric: 2
  > IP Address: 90.0.0.0, Metric: 3
  > IP Address: 100.0.0.0, Metric: 16
```

路由器B也会把H不可达的消息传送给C

```

> Frame 6: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface -, id 0
> Ethernet II, Src: cc:08:23:38:00:02 (cc:08:23:38:00:02), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
> Internet Protocol Version 4, Src: 30.0.0.64, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
  Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
    > IP Address: 100.0.0.0, Metric: 16

```

不过路由器C到H的最短路中继是D而不是B，C仍然认为H可达，将其发给B

```

> Frame 11: 206 bytes on wire (1648 bits), 206 bytes captured (1648 bits) on interface -, id 0
> Ethernet II, Src: cc:01:14:2f:00:10 (cc:01:14:2f:00:10), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
> Internet Protocol Version 4, Src: 30.0.0.65, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
  Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
    > IP Address: 0.0.0.0, Metric: 1
    > IP Address: 40.0.0.0, Metric: 1
    > IP Address: 50.0.0.0, Metric: 1
    > IP Address: 60.0.0.0, Metric: 3
    > IP Address: 70.0.0.0, Metric: 1
    > IP Address: 80.0.0.0, Metric: 1
    > IP Address: 90.0.0.0, Metric: 2
    > IP Address: 100.0.0.0, Metric: 3

```

B收到后确认更新自己的路由表，将到达H的链路信息送达路由器G

```

> Frame 15: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface -, id 0
> Ethernet II, Src: cc:08:23:38:00:01 (cc:08:23:38:00:01), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
> Internet Protocol Version 4, Src: 20.0.0.64, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
  Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
    > IP Address: 100.0.0.0, Metric: 4

```

G在最后更新路由表，所有路由器的路由表就更新完成

路由器B的路由表如下

```

B#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 0.0.0.0 to network 0.0.0.0

R    50.0.0.0/8 [120/1] via 30.0.0.65, 00:00:27, Ethernet0/2
R   100.0.0.0/8 [120/3] via 30.0.0.65, 00:00:27, Ethernet0/2
R    70.0.0.0/8 [120/1] via 30.0.0.65, 00:00:27, Ethernet0/2
R    80.0.0.0/8 [120/1] via 30.0.0.65, 00:00:27, Ethernet0/2
C    20.0.0.0/8 is directly connected, Ethernet0/1
R    40.0.0.0/8 [120/1] via 30.0.0.65, 00:00:27, Ethernet0/2
C    10.0.0.0/8 is directly connected, Ethernet0/0
R    90.0.0.0/8 [120/2] via 30.0.0.65, 00:00:28, Ethernet0/2
R    60.0.0.0/8 [120/3] via 30.0.0.65, 00:00:30, Ethernet0/2
C    30.0.0.0/8 is directly connected, Ethernet0/2

```

路由器G的路由表如下

```

G#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 0.0.0.0 to network 0.0.0.0

R    50.0.0.0/8 [120/2] via 20.0.0.64, 00:00:22, Ethernet0/0
R   100.0.0.0/8 [120/4] via 20.0.0.64, 00:00:22, Ethernet0/0
R    70.0.0.0/8 [120/2] via 20.0.0.64, 00:00:22, Ethernet0/0
R    80.0.0.0/8 [120/2] via 20.0.0.64, 00:00:22, Ethernet0/0
C    20.0.0.0/8 is directly connected, Ethernet0/0
R    40.0.0.0/8 [120/2] via 20.0.0.64, 00:00:22, Ethernet0/0
R    10.0.0.0/8 [120/1] via 20.0.0.64, 00:00:22, Ethernet0/0
R    90.0.0.0/8 [120/3] via 20.0.0.64, 00:00:22, Ethernet0/0
R    60.0.0.0/8 [120/4] via 20.0.0.64, 00:00:25, Ethernet0/0
R    30.0.0.0/8 [120/1] via 20.0.0.64, 00:00:25, Ethernet0/0

```

作为验证，我们让Bping到H

可以看到B-C链路上有ICMP包

No.	Destination	Time	Source	Protoco	Lengt	Info
159	100.0.0.65	521.22...	30.0.0.64	ICMP	114	Echo (ping) request id=0x0004, seq=1/256, ttl=255 (reply in 160)
160	30.0.0.64	521.26...	100.0.0.65	ICMP	114	Echo (ping) reply id=0x0004, seq=1/256, ttl=253 (request in 159)
161	100.0.0.65	521.27...	30.0.0.64	ICMP	114	Echo (ping) request id=0x0004, seq=2/512, ttl=255 (reply in 162)
162	30.0.0.64	521.29...	100.0.0.65	ICMP	114	Echo (ping) reply id=0x0004, seq=2/512, ttl=253 (request in 161)
163	100.0.0.65	521.30...	30.0.0.64	ICMP	114	Echo (ping) request id=0x0004, seq=3/768, ttl=255 (reply in 164)
164	30.0.0.64	521.32...	100.0.0.65	ICMP	114	Echo (ping) reply id=0x0004, seq=3/768, ttl=253 (request in 163)
165	100.0.0.65	521.33...	30.0.0.64	ICMP	114	Echo (ping) request id=0x0004, seq=4/1024, ttl=255 (reply in 166)
166	30.0.0.64	521.35...	100.0.0.65	ICMP	114	Echo (ping) reply id=0x0004, seq=4/1024, ttl=253 (request in 165)

可以看到B-G链路上没有ICMP包

icmp						
No.	Destination	Time	Source	Protoco	Lengt	Info

2.4 两台不相连机器的ping

这次我们让B和路由器H完全断开，shut掉C的s0/0接口使得C-D断开

路由器C在检测到链路故障后通知B右侧不可达

```
> Frame 19: 166 bytes on wire (1328 bits), 166 bytes captured (1328 bits) on interface -, id 0
> Ethernet II, Src: cc:01:14:2f:00:10 (cc:01:14:2f:00:10), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
> Internet Protocol Version 4, Src: 30.0.0.65, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
▼ Routing Information Protocol
  Command: Response (2)
  Version: RIPv2 (2)
  > IP Address: 0.0.0.0, Metric: 16
  > IP Address: 40.0.0.0, Metric: 16
  > IP Address: 50.0.0.0, Metric: 16
  > IP Address: 60.0.0.0, Metric: 16
  > IP Address: 70.0.0.0, Metric: 16
  > IP Address: 100.0.0.0, Metric: 16
```

路由器B也向G发送了通过C不可达的路由消息

```
> Frame 17: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits) on interface -, id 0
> Ethernet II, Src: cc:08:23:38:00:01 (cc:08:23:38:00:01), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
> Internet Protocol Version 4, Src: 20.0.0.64, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
▼ Routing Information Protocol
  Command: Response (2)
  Version: RIPv2 (2)
  > IP Address: 40.0.0.0, Metric: 16
  > IP Address: 50.0.0.0, Metric: 16
  > IP Address: 60.0.0.0, Metric: 16
  > IP Address: 70.0.0.0, Metric: 16
  > IP Address: 100.0.0.0, Metric: 16
```

路由器G也向B发送了通过H不可达的路由消息

```
> Frame 18: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits) on interface -, id 0
> Ethernet II, Src: cc:0a:23:7c:00:00 (cc:0a:23:7c:00:00), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
> Internet Protocol Version 4, Src: 20.0.0.65, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
▼ Routing Information Protocol
  Command: Response (2)
  Version: RIPv2 (2)
  > IP Address: 40.0.0.0, Metric: 16
  > IP Address: 50.0.0.0, Metric: 16
  > IP Address: 60.0.0.0, Metric: 16
  > IP Address: 70.0.0.0, Metric: 16
  > IP Address: 100.0.0.0, Metric: 16
```

最后路由器B向路由器C发送H不可达的路由消息

```
> Frame 20: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits) on interface -, id 0
> Ethernet II, Src: cc:08:23:38:00:02 (cc:08:23:38:00:02), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
> Internet Protocol Version 4, Src: 30.0.0.64, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
✓ Routing Information Protocol
  Command: Response (2)
  Version: RIPv2 (2)
  > IP Address: 40.0.0.0, Metric: 16
  > IP Address: 50.0.0.0, Metric: 16
  > IP Address: 60.0.0.0, Metric: 16
  > IP Address: 70.0.0.0, Metric: 16
  > IP Address: 100.0.0.0, Metric: 16
```

我们查看一下B的路由表如下，可见B已经无法连接到 100.0.0.0/8 网域

```
B#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 0.0.0.0 to network 0.0.0.0

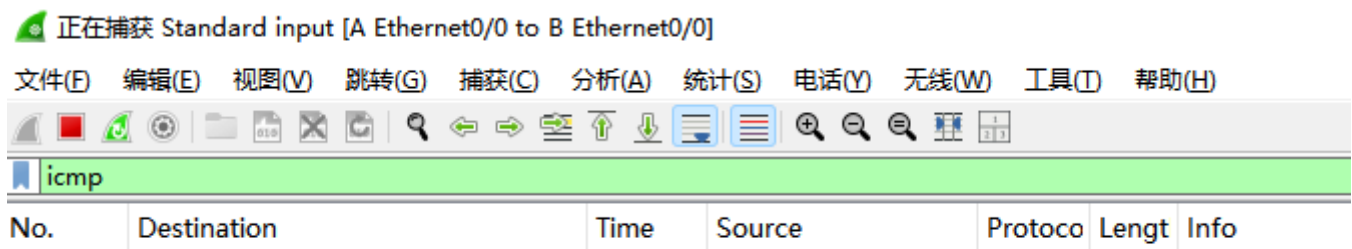
R    80.0.0.0/8 [120/1] via 30.0.0.65, 00:00:22, Ethernet0/2
C    20.0.0.0/8 is directly connected, Ethernet0/1
C    10.0.0.0/8 is directly connected, Ethernet0/0
R    90.0.0.0/8 [120/2] via 30.0.0.65, 00:00:23, Ethernet0/2
C    30.0.0.0/8 is directly connected, Ethernet0/2
```

Bping路由器H，可知无法到达

```
B#ping 100.0.0.65


Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 100.0.0.65, timeout is 2 seconds:
UUUUU
Success rate is 0 percent (0/5)
```

可见B到任何路由器的链路上都没有对应的ICMP包



正在捕获 Standard input [B Ethernet0/1 to G Ethernet0/0]

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)




icmp

No.	Destination	Time	Source	Protoco	Lengt	Info
-----	-------------	------	--------	---------	-------	------

正在捕获 Standard input [A Ethernet0/0 to B Ethernet0/0]

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)



icmp

No.	Destination	Time	Source	Protoco	Lengt	Info
-----	-------------	------	--------	---------	-------	------

2.5 通向不可达机器的ping

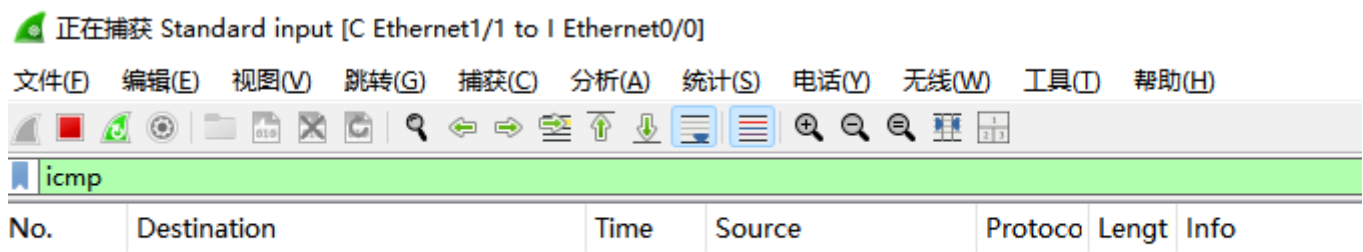
直接 `ping 80.0.0.100` 来完成测试

由于路由表是对于网域的路由，所以数据包会按照（B-C-I）的路径被发送，但是由于目标机器根本不存在，所以不会有回复报文被发送。我们可以看到B-C链路上有预期的ICMP包

正在捕获 Standard input [B Ethernet0/2 to C Ethernet1/0]							
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)							
icmp							
No.	Destination	Time	Source	Protoco	Lengt	Info	
15	80.0.0.100	38.813...	30.0.0.64	ICMP	114	Echo (ping) request id=0x000a, seq=0/0, ttl=255 (no response found!)	
16	80.0.0.100	40.797...	30.0.0.64	ICMP	114	Echo (ping) request id=0x000a, seq=1/256, ttl=255 (no response found!)	
17	80.0.0.100	42.686...	30.0.0.64	ICMP	114	Echo (ping) request id=0x000a, seq=2/512, ttl=255 (no response found!)	
19	80.0.0.100	44.686...	30.0.0.64	ICMP	114	Echo (ping) request id=0x000a, seq=3/768, ttl=255 (no response found!)	
21	80.0.0.100	46.716...	30.0.0.64	ICMP	114	Echo (ping) request id=0x000a, seq=4/1024, ttl=255 (no response found!)	

- > Frame 15: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface -, id 0
- ▼ Ethernet II, Src: cc:08:23:38:00:02 (cc:08:23:38:00:02), Dst: cc:01:14:2f:00:10 (cc:01:14:2f:00:10)
 - > Destination: cc:01:14:2f:00:10 (cc:01:14:2f:00:10)
 - > Source: cc:08:23:38:00:02 (cc:08:23:38:00:02)
 - Type: IPv4 (0x0800)
 - [Stream index: 6]
- ▼ Internet Protocol Version 4, Src: 30.0.0.64, Dst: 80.0.0.100
 - 0100 = Version: 4
 - 0101 = Header Length: 20 bytes (5)
 - > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 - Total Length: 100
 - Identification: 0x0032 (50)
 - > 000. = Flags: 0x0
 - ...0 0000 0000 0000 = Fragment Offset: 0
 - Time to Live: 255
 - Protocol: ICMP (1)
 - Header Checksum: 0x4cc3 [validation disabled]
 - [Header checksum status: Unverified]
 - Source Address: 30.0.0.64
 - Destination Address: 80.0.0.100
 - [Stream index: 2]

路由器C接收到路由消息后，发现目的网域自己已经接入了，然而并没有80.0.0.100这个ip地址，其不在路由器C的ARP表中，C会向I的网域广播ARP报文寻找目的IP机器MAC地址，但是没有收到回复，认为IP地址错误，那么C-I链路上没有数据包



3.分别启用 “下一跳” 与 “转发接口” 配置静态路由对比

3.1 下一跳方式的ping

配置下一跳方式的命令如下

```
1 Router#ip route 目标网络 掩码 下一跳IP
```

在规定下一跳IP之后，利用缺省路由补全这个拓扑静态路由

之后我们让JpingH，路径是J-I-C-D-H

```
J#ping 70.0.0.65
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 70.0.0.65, timeout is 2 seconds:
....!
Success rate is 20 percent (1/5), round-trip min/avg/max = 220/220/220 ms
```

可见丢包率为4/5，原因如下：

ICMP报文经过一条链路后，途经的路由器ARP表如果为空，就需要先执行一次ARP请求才可以知道下一跳的MAC地址来发送路由消息，则这个ICMP包未被发送产生丢包

这样的逐级递减过程可以在Wireshark抓包里看到

J-I链路如下

7	70.0.0.65	24.998...	90.0.0.65	ICMP	114	Echo (ping) request	id=0x0000, seq=1/256, ttl=255 (no response found!)
8	70.0.0.65	27.076...	90.0.0.65	ICMP	114	Echo (ping) request	id=0x0000, seq=2/512, ttl=255 (no response found!)
10	70.0.0.65	29.124...	90.0.0.65	ICMP	114	Echo (ping) request	id=0x0000, seq=3/768, ttl=255 (no response found!)
12	70.0.0.65	31.000...	90.0.0.65	ICMP	114	Echo (ping) request	id=0x0000, seq=4/1024, ttl=255 (no response found!)

I-C链路如下

5	70.0.0.65	22.170...	90.0.0.65	ICMP	114 Echo (ping) request	id=0x0000, seq=2/512, ttl=254 (no response found!)
7	70.0.0.65	24.224...	90.0.0.65	ICMP	114 Echo (ping) request	id=0x0000, seq=3/768, ttl=254 (no response found!)
8	70.0.0.65	26.101...	90.0.0.65	ICMP	114 Echo (ping) request	id=0x0000, seq=4/1024, ttl=254 (no response found!)

C-D链路如下

3	70.0.0.65	3.4324...	90.0.0.65	ICMP	104 Echo (ping) request	id=0x0000, seq=2/512, ttl=253 (no response found!)
4	70.0.0.65	5.4856...	90.0.0.65	ICMP	104 Echo (ping) request	id=0x0000, seq=3/768, ttl=253 (no response found!)

D-H链路如下

5	70.0.0.65	2.7007...	90.0.0.65	ICMP	114 Echo (ping) request	id=0x0000, seq=3/768, ttl=252 (no response found!)
---	-----------	-----------	-----------	------	-------------------------	--

之后如果我们执行同样的ping命令，丢包率为0/5，因为ARP表已经被确定下来了，途经的路由器在本地查询即可发送

```
J#ping 70.0.0.65

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 70.0.0.65, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 96/102/128 ms
```

可以查看路由器I的ARP表，指令与ARP表如下：

- 1 Router#show arp

```
I#show arp
Protocol Address      Age (min)  Hardware Addr  Type   Interface
Internet 70.0.0.65      2         cc01.142f.0011 ARPA   Ethernet0/0
Internet 80.0.0.64      2         cc01.142f.0011 ARPA   Ethernet0/0
Internet 80.0.0.65      -         cc06.22f0.0000 ARPA   Ethernet0/0
Internet 90.0.0.64      -         cc06.22f0.0001 ARPA   Ethernet0/1
Internet 90.0.0.65      2         cc07.2312.0000 ARPA   Ethernet0/1
```

3.2 转发接口方式的ping

配置转发接口方式的指令如下：

- 1 Router#no ip route 目标网络 掩码 下一跳IP
- 2 Router#ip route 目标网络 掩码 出口接口

我们对路由器J执行这样的命令，结果如下

完成后，我们让Aping路由器F，链路为A-B-C-D-E-F，结果如下

```
A#ping 60.0.0.65

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 60.0.0.65, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

丢包率达到5/5！原因同上，未知接收方，仍然需要ARP询问，消耗ICMPping包数量

B-C链路如下

6	60.0.0.65	16.230...	10.0.0.64	ICMP	114	Echo (ping) request	id=0x0000, seq=1/256, ttl=255 (no response found!)
7	60.0.0.65	18.309...	10.0.0.64	ICMP	114	Echo (ping) request	id=0x0000, seq=2/512, ttl=255 (no response found!)
9	60.0.0.65	20.359...	10.0.0.64	ICMP	114	Echo (ping) request	id=0x0000, seq=3/768, ttl=255 (no response found!)
10	60.0.0.65	22.409...	10.0.0.64	ICMP	114	Echo (ping) request	id=0x0000, seq=4/1024, ttl=255 (no response found!)

C-D链路如下

1	60.0.0.65	0.0000...	10.0.0.64	ICMP	104	Echo (ping) request	id=0x0000, seq=2/512, ttl=253 (no response found!)
2	60.0.0.65	2.0594...	10.0.0.64	ICMP	104	Echo (ping) request	id=0x0000, seq=3/768, ttl=253 (no response found!)
4	60.0.0.65	4.1003...	10.0.0.64	ICMP	104	Echo (ping) request	id=0x0000, seq=4/1024, ttl=253 (no response found!)

执行这次命令后，ARP表应该更新完成，我们查看路由器E的ARP表

```
E#show arp
Protocol Address      Age (min)  Hardware Addr  Type   Interface
Internet 10.0.0.64        0          cc02.1471.0010 ARPA    Ethernet0/0
Internet 50.0.0.64        0          cc02.1471.0010 ARPA    Ethernet0/0
Internet 50.0.0.65        -          cc03.148f.0000 ARPA    Ethernet0/0
Internet 60.0.0.64        -          cc03.148f.0001 ARPA    Ethernet0/1
Internet 60.0.0.65        0          cc04.14b1.0000 ARPA    Ethernet0/1
```

再次ping一遍，发现丢包率0/5

```
A#ping 60.0.0.65

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 60.0.0.65, timeout is 2 seconds:
!!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 92/113/160 ms
```

3.3 两种方式对比分析

对比维度	下一跳 (Next-Hop) 方式	转发接口 (Outbound Interface) 方式
适用场景	适合复杂网络结构，如多路径路由、跨域网络环境，或需要精确控制转发路径的场景	适合简单网络场景，如直连目标网络（例如点对点连接）的情况，不需要动态更新下一跳信息
配置内容	配置目标网络和对应的下一跳IP地址，例如 <code>ip route 10.0.0.0 255.0.0.0 192.168.1.1</code>	配置目标网络和对应的出口接口，例如 <code>ip route 10.0.0.0 255.0.0.0 Ethernet0/1</code>
依赖性	高灵活性，支持多个下一跳IP地址，便于实现路由冗余和负载均衡；适合大规模网络中的动态路由调整	灵活性较低，无法直接支持多路径负载均衡和路由冗余，完全依赖接口的连通性

维护复杂度	配置复杂，尤其是在需要维护大量目标网络和下一跳条目时，配置和调整成本较高	配置简单，适合小规模网络，但在大规模网络中可能需要频繁更改接口配置，增加维护负担
故障排查难度	排查较为复杂，需检查以下内容： <ul style="list-style-type: none">路由表的下一跳配置是否正确；ARP解析是否成功；下一跳IP地址是否可达	排查相对简单，只需检查： <ul style="list-style-type: none">出口接口的状态是否正常；ARP解析目标MAC地址是否成功
转发效率	转发效率高，通过路由表快速匹配目标IP地址并找到下一跳IP地址，但需依赖ARP缓存解析MAC地址完成传输	转发效率更高，直接使用出口接口发送数据，省去对下一跳IP地址的查找过程，但适用范围有限
路由协议兼容性	支持静态路由和动态路由（如RIP、OSPF、EIGRP等），适用于复杂的动态网络环境	通常用于静态路由场景，不适合动态路由协议，因为动态路由协议需要依赖下一跳地址的变化
ARP表依赖	ARP表中包含下一跳路由器的IP地址和MAC地址，若下一跳设备的ARP解析失败，转发过程会中断	ARP表中包含目标主机的IP地址和MAC地址，若目标设备的ARP解析失败，转发过程会中断
故障影响	若下一跳路由器不可达，所有依赖该路由的目标网络都会受影响，可能导致大范围的网络中断	若出口接口不可用，直接影响所有需要通过该接口的通信，范围较局限于该接口
Ping测试结果分析	<ul style="list-style-type: none">当下一跳IP地址可达时，Ping测试成功；当下一跳IP地址不可达或ARP解析失败时，Ping测试失败	<ul style="list-style-type: none">当出口接口可用且目标设备的ARP解析成功时，Ping测试成功；出口接口不可用或目标设备不可达时，Ping测试失败
安全性	提高安全性，通过ACL（访问控制列表）结合下一跳IP地址实现更细粒度的流量控制	安全性较低，基于接口的流量控制范围较大，难以实现精确的流量管理

核心差异

- 下一跳方式**更灵活，适用于复杂的动态网络，可支持多路径和负载均衡，但配置和维护成本较高，故障排查更复杂。
- 转发接口方式**配置简单，适用于小规模、直连场景，故障排查简单，但灵活性不足，难以支持复杂路由需求。

六、实验思考题

1. 下一跳与转发接口方式配置静态路由的区别及其对Ping丢包率的影响

1.1 配置差异

下一跳方式：

- 配置时直接指定**目标网络的下一跳路由器的IP地址**。
- 路由器根据目标IP地址来判断下一跳，转发数据包时通过该下一跳的路由器进行中转。

转发接口方式：

- 配置时指定**数据包应通过的本地出接口**（如某个物理接口或虚拟接口），而不是下一跳路由器的IP地址。
- 路由器根据出接口将数据包发送到该接口对应的网络段，数据包将在此接口上继续转发，直到到达目标网络或最终目的主机。

1.2 ARP表影响

下一跳方式：

- 路由器的ARP表将包含**下一跳路由器的IP与MAC地址**，因为在通过下一跳路由器转发数据时，必须知道下一跳路由器的MAC地址。
- ARP表会存储下一跳路由器的IP地址和它的MAC地址，直到TTL（生存时间）过期或网络发生变化。

转发接口方式：

- 路由器的ARP表通常也会包含下一跳路由器的IP与MAC地址，因为最终数据包的目标是通过下一个路由器转发的。
- 同样，ARP表中会保存目标网络段的MAC地址，直到TTL过期或网络发生变化。

1.3 Ping丢包情况

ARP表空时的丢包：

- **第一次执行Ping时**，无论采用哪种方式，都会发生丢包。原因是路由器的ARP表最初为空，在发送数据包时，路由器必须通过ARP请求获取下一跳路由器或目标主机的MAC地址。
- 在第一次Ping时，ARP请求会消耗一定时间，导致数据包丢失。只有当ARP表被更新并缓存了MAC地址，后续的Ping请求才不会丢包。

后续Ping请求：

- 一旦ARP表填充了正确的MAC地址，后续的Ping请求通常不会丢包，因为ARP表中的信息已经存在，数据包可以直接发送，无需等待ARP请求。
- 丢包情况通常只出现在首次发送数据包时，后续的Ping请求成功率较高。

与路由器负载的关系：

- 如果网络上路由器的负载较高，或者网络较为复杂，ARP请求的响应时间可能会延迟，导致Ping丢包的概率增加。
- 在配置路由时，如果使用多个静态路由或在复杂的拓扑中，路由器处理ARP请求的效率可能受到影响。

2.OSPF协议与RIP协议在邻居发现和数据库同步中的差异

方面	RIP	OSPF
邻居发现	<div>- 定期发送更新来发现和维护路由</div> <div>- 使用RIP协议的Request和Response</div> <div>- 使用224.0.0.9多播地址实现传输</div>	<div>- 使用Hello包进行邻居发现和维护</div> <div>- Hello包用于交换信息，内容包括区域ID、传输间隔以及寿命等</div> <div>- 使用224.0.0.5/224.0.0.6的多播地址实现传输</div>
数据库信息同步	<div>- 基于距离向量算法，每个路由器通过广播路由表更新与邻居共享其路由信息</div> <div>-使用简单的RIP Request和RIP Response包进行沟通</div> <div>- 路由表的更新是全局的，且是周期性的，不过数据库同步速度较慢</div>	<div>- 基于链路状态算法，路由器发送LSA（链路状态广告）来同步链路状态数据库</div> <div>-先用DD包交换摘要信息，再用LSR发起请求，传输LSU包，最后用LSAck确认接收，再在本地更新最短路径</div> <div>- LSA传播后，每个路由器都会重新计算最短路径，更新路由信息</div>
消息传输	<div>- 采用周期性的路由更新消息广播（默认30秒），通过UDP端口520进行通信</div> <div>- 路由信息通过广播或单播传递给邻居，但是缺乏Ack，传输可靠性较差</div>	<div>- 使用链路状态广告（LSA）来传递链路状态信息，LSA通过OSPF路由协议的多播地址进行传输</div> <div>- 支持广播、点对点以及多播网络，使用LSAck包确认已经接收的LSA</div>

3.数据包从PC发送到其他PC的完整过程

3.1 ARP过程：

步骤：

- **目标：**当PC需要向同一子网或不同子网的目标PC发送数据时，首先需要通过ARP请求获取目标IP地址对应的MAC地址。

- **ARP请求**：如果发送方PC的ARP缓存中没有目标IP地址的MAC地址，它会广播一个ARP请求包到局域网。ARP请求内容包括发送方的IP和MAC地址以及目标IP地址。
- **ARP响应**：接收到ARP请求的目标设备（如目标PC或路由器）会检查自己的IP地址是否与请求中的目标IP匹配。如果匹配，目标设备会发送一个ARP响应，其中包含目标IP对应的MAC地址。
- **ARP表更新**：发送方PC在收到ARP响应后，会将目标IP和对应的MAC地址添加到其ARP缓存表中，这样在后续的数据传输中就不再需要再次查询目标MAC地址。
- **缓存时间**：ARP表中的映射通常具有一定的生命周期（如几分钟），过期后会重新发起ARP请求来更新缓存。

3.2 路由表匹配过程：

查找最优路由：

- 路由器根据数据包的目标IP地址在其**路由表**中查找匹配项。
- 路由表中保存了不同网络段的信息，包括目标网络、子网掩码、下一跳IP地址、出接口等。
- 路由器会选择与目标IP最匹配的路由条目，通常是最长匹配原则，即子网掩码最长的匹配。

转发决策：

- 一旦匹配到最优路由，路由器决定数据包的**转发路径**，即选择数据包应该经过的**下一跳设备**（如另一个路由器）或直接发往目的设备。
- 路由器根据匹配结果和其接口信息，决定数据包的转发接口，并将数据包转发到相应的接口或下一跳路由器。

3.3 链路层协议封装过程：

封装：

- 数据包在经过路由器的处理后，会被封装成链路层的**帧**，这通常是以太网帧，帧中包含了源MAC地址、目标MAC地址、数据包（IP数据报）以及其他链路层信息（如类型字段、校验和等）。
- 对于局域网（LAN）中的通信，链路层帧会使用**MAC地址**来标识源设备和目标设备，而不是使用IP地址。
- 如果数据包需要跨越不同网络段，路由器会根据目标IP地址来决定是否需要封装到不同的链路层协议（例如，针对不同类型的网络，可能使用不同的封装方式，如PPP、ATM等）。

发送：

- 封装好的帧通过**物理接口**（如网卡、无线适配器）发送到目标设备或下一跳设备。
- 发送方式可以是**单播**（发送给单一目标设备）、**广播**（发送给网络中所有设备，如ARP请求）或**多播**（发送给一组设备，如OSPF的LSA广播）。
- 数据包通过物理媒介（如网线、无线信号）传输，直到到达目标设备或经过的路由器。

4.动态路由协议与静态路由在网络管理中的优势与劣势

4.1 动态路由协议的优势：

- **自动化：**动态路由协议（如OSPF和RIP）能够自动发现网络拓扑的变化并进行路由调整。当网络中的链路发生故障或新的设备加入时，协议可以自动计算并选择最优路径，减少了人工干预的需求。这使得网络能够灵活适应环境的变化，保持网络通信的畅通。
- **可扩展性：**动态路由协议适用于大规模和复杂的网络。随着网络的增长和拓扑的复杂化，手动配置静态路由将变得非常困难，而动态路由协议能够在大型网络中高效地进行路由选择和更新，保持网络的稳定性和可靠性。
- **冗余与负载均衡：**动态路由协议支持多路径路由，能够提升网络的冗余性和可靠性。例如，OSPF支持多路径选择，可以在多个路径间进行负载均衡，避免单一路由故障导致的网络中断。多路径路由不仅提高了网络的可用性，还提升了网络的整体性能。

4.2 动态路由协议的劣势：

- **资源消耗：**动态路由协议需要额外的处理能力（CPU）和内存资源，以维持路由表的更新和计算。在大规模网络中，路由器需要不断地处理路由更新和计算最优路径，这会导致设备资源的消耗。因此，动态路由协议对设备的要求较高，可能不适用于资源有限的设备。
- **复杂性：**配置和管理动态路由协议较为复杂。动态路由协议（如OSPF、RIP、BGP等）具有多种配置选项和特性，需要深入理解协议的工作原理，如链路状态算法、距离向量算法等。管理和排查故障时，需要较强的网络知识和技术支持，尤其是在大型网络中。

4.3 静态路由的优势：

- **简单：**静态路由的配置和管理非常简便，尤其适用于小型、稳定的网络环境。在静态路由中，管理员需要手动设置路由路径，一旦配置完成，网络路径便是固定的。对于拓扑简单、变动少的网络来说，静态路由是一个非常有效的选择。
- **低开销：**静态路由不依赖于任何动态路由协议，因此不会消耗CPU和内存资源。这对于设备资源有限的环境（如小型路由器或嵌入式设备）来说，非常适合。静态路由的网络流量也比较低，不会产生不必要的协议开销。
- **安全性：**由于静态路由路径是固定的，不会自动发生变化，因此具有较高的安全性。静态路由能够避免动态路由协议可能带来的安全隐患，例如路由劫持或路由欺骗等问题。此外，静态路由不依赖于外部协议，因此减少了被攻击的面。

4.4 静态路由的劣势：

- **缺乏自动化：**静态路由的一个主要劣势是缺乏自动化更新机制。当网络拓扑发生变化（如链路故障或设备新增）时，需要管理员手动更新路由表。这不仅增加了管理成本，而且在网络变动频繁的情况下，手动更新容易出错，导致网络通信中断。
- **不适用于大规模网络：**静态路由在大型、复杂网络中不具备良好的适应性。在这些网络中，手动配置和维护每一条路由变得非常困难。尤其是当网络拓扑频繁发生变化时，静态路由的维护工作量极大，且容易产生错误。

4.5 动态路由与静态路由的综合使用：

在实际网络环境中，动态路由协议和静态路由可以结合使用。对于大型网络，动态路由协议可以处理大部分的路由计算和更新，而在一些特定的情况下（如固定的出口链路、某些专用网络的路由），可以使用静态路由来确保流量的固定路径。这样既能提高网络的灵活性，又能在关键路径上保证网络的稳定性。

七、实验心得

在这次实验过程中，我深入探究了**RIP**和**OSPF**这两种常见的路由协议，并通过实际操作和观察，获得了许多有价值的认知和体会。通过对比这两种协议的工作原理、路由计算方式以及各自的优缺点，我对路由协议的理解更加深刻，以下是我的几点主要心得。

1. RIP协议的路由表计算机制

RIP协议作为一种基于**距离向量算法**的路由协议，通过交换路由表信息来计算最短路径。每个路由器都会以“跳数”作为衡量路径优先级的标准，跳数越少的路径，优先级越高。这种方式的优点是实现简单，但也带来了较慢的收敛速度和较高的路由开销。通过在GNS3中仿真，观察到RIP协议每30秒就会发送一次路由更新，这使得网络中的路由表能够保持最新。尽管RIP能够自动更新网络拓扑，但由于其较慢的收敛速度，网络拓扑变化时可能会造成一定的网络不稳定，直到路由器完成更新。

特点：

- **定期更新：**每30秒发送一次路由更新，保证路由信息的新鲜度。
- **距离向量算法：**以跳数作为路由度量单位，跳数越少，优先级越高。
- **收敛性慢：**网络拓扑变化时，RIP协议可能需要较长时间才能完成收敛，导致短时间内网络不稳定。

2. RIP协议中的路由环路及解决方法

在实验过程中，我还探讨了RIP协议可能出现的**路由环路**问题。由于RIP协议采用的是距离向量算法，在某些情况下，网络中可能会形成数据包无限循环的环路，特别是当网络拓扑发生变化时。例

如，当某一链路失效或设备崩溃时，路由器可能会误将错误的路由信息传递给其他路由器，从而导致路由环路的生产。

为了解决这个问题，RIP协议引入了**水平分割（Split Horizon）**机制，它能有效防止错误路由信息在环路中传播。具体来说，启用水平分割后，路由器会阻止将通过某一接口接收到的路由信息再次通过同一接口转发。这种机制显著减少了路由环路的可能性，并加速了路由表的收敛速度。

水平分割的效果：

- **防止错误信息传播：**通过阻止路由器将错误的路由信息在环路中传播，避免了环路的产生。
- **加速收敛速度：**有效避免了错误信息的干扰，使得路由表能够更快速地更新并找到最短路径。

3. OSPF协议的链路状态路由计算

与RIP协议相比，**OSPF**是基于**链路状态算法**的路由协议。在我的实验中，我深刻体会到了OSPF如何通过交换**链路状态信息（LSA）**来构建全网的拓扑图，并使用**Dijkstra算法**计算最短路径树（SPF）。与RIP的距离向量不同，OSPF的每个路由器都能了解整个网络的拓扑结构，这使得OSPF能够在路由计算时做出更为精确的决策。

OSPF的链路状态机制：

- **交换LSA：**每个OSPF路由器通过交换链路状态信息（LSA），构建整个网络的拓扑图，确保网络中每个路由器的视图是一致的。
- **Dijkstra算法：**基于LSA，OSPF路由器运用Dijkstra算法计算从路由器到其他节点的最短路径，生成最优的路由表。

4. OSPF的快速收敛机制

在OSPF中，路由信息的传播并非定期进行，而是通过**触发式更新（Trigger Updates）**来实现。也就是说，当网络拓扑发生变化时，路由器会立刻生成新的LSA并进行传播，这大大提高了网络的响应速度。OSPF还通过**LSA确认机制**确保每个LSA能够可靠传递，不会丢失信息，从而进一步提高了协议的稳定性。

OSPF的关键特点：

- **触发式更新：**网络拓扑变化时，路由器会即时生成新的LSA，迅速调整路由信息。
- **高效的LSA传播：**OSPF通过多播地址和LSA确认机制，确保信息快速且可靠地传播到网络中的每一个路由器。

5. OSPF防止错误最短路径信息传播

为了确保路由信息的准确性，OSPF采用了一些重要的机制，如**LSA的序列号和年龄机制**。每个LSA都带有序列号和生存时间（LS Age），路由器总是优先使用最新的LSA信息，确保路由表中不会使用过时的信息。此外，OSPF还支持认证机制，避免恶意路由器注入伪造的路由信息。

OSPF的防误机制：

- **LSA序列号与年龄：**确保每个路由器使用的是最新的、有效的链路状态信息。
- **认证机制：**通过认证确保LSA的可信度，防止恶意路由器注入错误信息。

总结与心得

通过这次对RIP和OSPF协议的深入学习和实验，我对两种路由协议的工作原理有了更加清晰的理解。RIP协议简单易用，但由于收敛速度慢、容易产生路由环路等问题，适合小型或稳定的网络环境。而OSPF协议作为链路状态协议，具有更高的灵活性和更快的收敛速度，适合大规模、复杂的网络。

我认识到，每种协议都有其适用的场景和优势，作为网络工程师，我们需要根据网络规模、拓扑结构以及管理需求选择合适的路由协议。RIP协议在一些简单的应用场景中仍然具有实用价值，而OSPF则适合高性能要求的企业级网络。在实际的网络设计和优化过程中，需要不断根据需求和条件做出合理的选择与调整。