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_____黎昱彤_____

实　验　日　期：　2024.11.19

学　　号____2022211414____

实 验 报 告 日 期 ：　2024.11.29

报　告　退　发：（ 订正 、 重做 ）

目录

1 实验环境

- GNS3
- VMware
- Wireshark

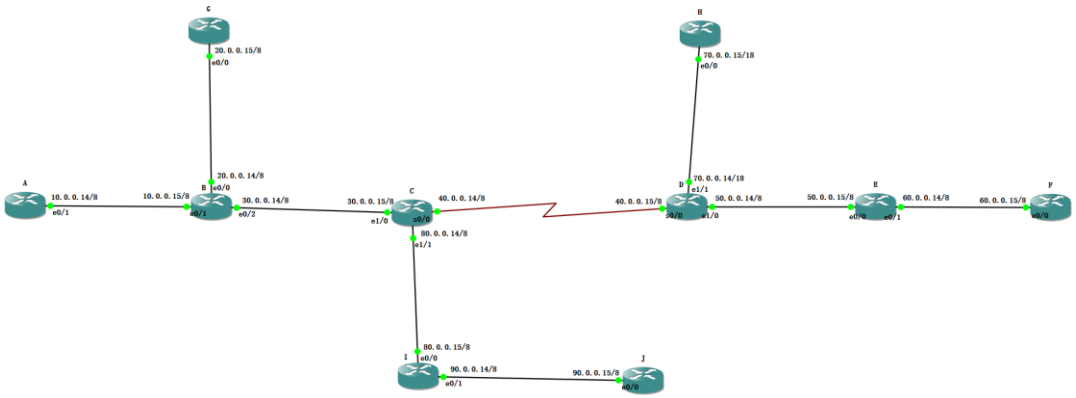
2 实验目的

- 实现 RIP 和 OSPF 路由协议；
- 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；
- OSPF 中数据库同步信息的格式和同步对象，链路改变信息如何发送，具体格式，修改部分链路，观察消息传递过程。

3 实验内容及步骤

3.1 IP

使用实验 2 的拓扑图和 IP 配置



3.2 RIP

以路由器 A 为例，指令如下：

```
A#config t
Enter configuration commands, one per line.  End with CNTL/Z.
A(config)#router rip
A(config-router)#version 2
A(config-router)#network 10.0.0.0
```

在路由器 A 上配置 RIP：在特权模式下，进入全局配置模式，再进入 RIP 路由器配置模式。将 RIP 协议版本设置为 2，以支持广播和多播传播。network 10.0.0.0 命令将接口 e0/1 所属的网络 10.0.0.14 添加到 RIP 路由表中，以便路由器 A 可以通过 RIP 协议向其他路由器广播此网络的路由信息。

```
A#show ip protocols
Routing Protocol is "rip"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Sending updates every 30 seconds, next due in 1 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Redistributing: rip
  Default version control: send version 2, receive version 2
    Interface             Send  Recv  Triggered RIP  Key-chain
    Ethernet0/1           2     2
  Automatic network summarization is in effect
  Maximum path: 4
  Routing for Networks:
    10.0.0.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    10.0.0.15             120      00:00:40
  Distance: (default is 120)
```

其余路由器配置同理。

3.3 OSPF

以路由器 A 为例，指令如下：

```
A#config t
Enter configuration commands, one per line.  End with CNTL/Z.
A(config)#no router rip
A(config)#router ospf 10
A(config-router)#network 10.0.0.0 0.255.255.255 area 0
A(config-router)#interface e0/1
A(config-if)#ip ospf hello-interval 5
A(config-if)#ip ospf dead-interval 20
```

在路由器 A 上配置 OSPF：在特权模式下，进入全局配置模式，停用之前配置的 RIP 路由协议。进入 OSPF 路由器配置模式，进程号为 10。network 10.0.0.0 0.255.255.255 area 0 命令将接口 e0/1 所属的网络 10.0.0.14 添加到 OSPF 路由表中（反子网掩码比子网掩码更灵活，支持非连续的 IP 地址匹配），并分配给区域 0，以便路由器 A 可以与同一区域的其他路由器交换路由信息。配置接口 e0/1 使用 ip ospf hello-interval 5 设置 Hello 报文发送间隔 5s，ip ospf dead-interval 20 设置认为通过接 e0/1 相连的邻居已经不存在的时间间隔为 20s。

```
A#show ip protocol
Routing Protocol is "ospf 10"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Router ID 100.0.0.14
  Number of areas in this router is 1. 1 normal 0 stub 0 nssa
  Maximum path: 4
  Routing for Networks:
    10.0.0.0 0.255.255.255 area 0
 Reference bandwidth unit is 100 mbps
  Routing Information Sources:
    Gateway         Distance      Last Update
    30.0.0.14            110      00:00:39
    60.0.0.14            110      00:00:39
    70.0.0.14            110      00:00:39
    90.0.0.14            110      00:00:39
    80.0.0.14            110      00:00:39
    110.0.0.15           110      00:00:39
  Distance: (default is 110)
```

其余路由器配置同理。

4 实验结果

4.1 RIP 路由协议流程分析

(1)初始信息交互

所有路由器配置好 RIP 协议后，在一个路由器上执行 debug ip rip

```
A#clear ip route *
A#debug ip rip
RIP protocol debugging is on
A#
*Mar  1 00:42:47.459: RIP: received v2 update from 10.0.0.15 on Ethernet0/1
*Mar  1 00:42:47.459:      20.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 00:42:47.463:      30.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 00:42:47.463:      40.0.0.0/8 via 0.0.0.0 in 2 hops
*Mar  1 00:42:47.463:      50.0.0.0/8 via 0.0.0.0 in 3 hops
*Mar  1 00:42:47.463:      60.0.0.0/8 via 0.0.0.0 in 4 hops
*Mar  1 00:42:47.467:      70.0.0.0/8 via 0.0.0.0 in 3 hops
*Mar  1 00:42:47.467:      80.0.0.0/8 via 0.0.0.0 in 2 hops
*Mar  1 00:42:47.467:      90.0.0.0/8 via 0.0.0.0 in 3 hops
A#
*Mar  1 00:43:09.319: RIP: sending v2 update to 224.0.0.9 via Ethernet0/1 (10.0.0.14)
*Mar  1 00:43:09.319: RIP: build update entries - suppressing null update
A#
```

分析：收到消息来自10.0.15，通过e0/1接口，路由器周期性（30s）广播它的已知路由表，包含多个目标网络的跳数信息：20.0.0.0/8：1跳（直连）；30.0.0.0/8：1跳；其他网络同理。发送广播时，224.0.0.9是RIP使用的组播地址，e0/1接口，本地地址为10.0.0.14，系统正在构建路由更新，但由于默认启用了水平分割，某些更新被抑制（suppressing null update）这种机制可以防止路由环路。

问题1：RIP协议的交互信息中会出现via 0.0.0.0，这个0.0.0.0表示什么意思？

回答： RIP将下一跳地址显示为0.0.0.0，表示数据包可以直接通过本地接口发送到目标网络，无需通过中间路由器转发，或者路由条目初始化时未配置明确的下一跳地址。

问题2：RIP的发送目的地址会有224.0.0.9的情况出现，这是什么地址？

回答：是RIP使用的组播地址。RIPv2采用组播方式路由，大大降低了非RIP设备的处理开销。

(2)路由计算

RIP协议基于距离矢量算法，使用**跳数**作为度量标准，消息传递过程具有以下特点：**周期性更新** 每隔30s路由器会广播其完整的路由表到所有邻居；**渐进更新** 当某条路径的跳数发生变化时，路由器会通过接收到的更新调整其路由表，并将变化通知其他路由器；**触发更新** 当网络拓扑发生变化（如链路断开），会立即生成触发更新（Triggered Updates），而无需等待下一个更新周期。

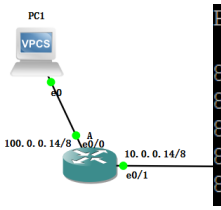
调试可观察到每次更新得到的每条路由的跳数，最终建立了完整路由表

```
A#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 [120/3] via 10.0.0.15, 00:01:00, Ethernet0/1
R    70.0.0.0/8 [120/3] via 10.0.0.15, 00:01:00, Ethernet0/1
R    80.0.0.0/8 [120/2] via 10.0.0.15, 00:01:00, Ethernet0/1
R    20.0.0.0/8 [120/1] via 10.0.0.15, 00:01:00, Ethernet0/1
R    40.0.0.0/8 [120/2] via 10.0.0.15, 00:01:00, Ethernet0/1
C    10.0.0.0/8 is directly connected, Ethernet0/1
R    90.0.0.0/8 [120/3] via 10.0.0.15, 00:01:00, Ethernet0/1
R    60.0.0.0/8 [120/4] via 10.0.0.15, 00:01:00, Ethernet0/1
R    30.0.0.0/8 [120/1] via 10.0.0.15, 00:01:03, Ethernet0/1
S*   0.0.0.0/0 is directly connected, Ethernet0/1
```

直连路由器A的PC1可以ping通网络中任意一个路由器




```
PC1> ping 60.0.0.14

84 bytes from 60.0.0.14 icmp_seq=1 ttl=251 time=110.441 ms
84 bytes from 60.0.0.14 icmp_seq=2 ttl=251 time=57.622 ms
84 bytes from 60.0.0.14 icmp_seq=3 ttl=251 time=78.349 ms
84 bytes from 60.0.0.14 icmp_seq=4 ttl=251 time=77.057 ms
84 bytes from 60.0.0.14 icmp_seq=5 ttl=251 time=75.910 ms
```

(3)链路故障处理

模拟链路故障，关闭路由器I的e0/0接口

观察故障影响

路由器A debug ip rip发现90.0.0.0网段消失，观察不到RIP的更新包

关闭路由器A e0/1接口的水平分割后，debug查看故障传播行为

```
A#debug ip rip
RIP protocol debugging is on
A#
*Mar  1 04:59:45.722: RIP: sending v2 update to 224.0.0.9 via Ethernet0/1 (10.0.0.14)
*Mar  1 04:59:45.722: RIP: build update entries
*Mar  1 04:59:45.722:   10.0.0.0/8 via 0.0.0.0, metric 1, tag 0
*Mar  1 04:59:45.726:   20.0.0.0/8 via 0.0.0.0, metric 2, tag 0
*Mar  1 04:59:45.726:   30.0.0.0/8 via 0.0.0.0, metric 2, tag 0
*Mar  1 04:59:45.726:   40.0.0.0/8 via 0.0.0.0, metric 3, tag 0
*Mar  1 04:59:45.726:   50.0.0.0/8 via 0.0.0.0, metric 4, tag 0
*Mar  1 04:59:45.730:   60.0.0.0/8 via 0.0.0.0, metric 5, tag 0
*Mar  1 04:59:45.730:   70.0.0.0/8 via 0.0.0.0, metric 4, tag 0
*Mar  1 04:59:45.730:   80.0.0.0/8 via 0.0.0.0, metric 3, tag 0
A#
*Mar  1 04:59:45.734:   90.0.0.0/8 via 0.0.0.0, metric 16, tag 0
*Mar  1 04:59:45.734:   100.0.0.0/8 via 0.0.0.0, metric 1, tag 0
```

```
A#
*Mar  1 05:00:20.586: RIP: received v2 update from 10.0.0.15 on Ethernet0/1
*Mar  1 05:00:20.586:      20.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 05:00:20.590:      30.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 05:00:20.590:      40.0.0.0/8 via 0.0.0.0 in 2 hops
*Mar  1 05:00:20.590:      50.0.0.0/8 via 0.0.0.0 in 3 hops
*Mar  1 05:00:20.590:      60.0.0.0/8 via 0.0.0.0 in 4 hops
*Mar  1 05:00:20.594:      70.0.0.0/8 via 0.0.0.0 in 3 hops
*Mar  1 05:00:20.594:      80.0.0.0/8 via 0.0.0.0 in 2 hops
*Mar  1 05:00:20.594:      90.0.0.0/8 via 0.0.0.0 in 16 hops  (inaccessible)
```

RIP会将原来的路由条目标记为无效，发送带有metric=16（不可达）的更新包，路由条目被逐步移除或更新，符合RIP的180s hold-down计时规则。

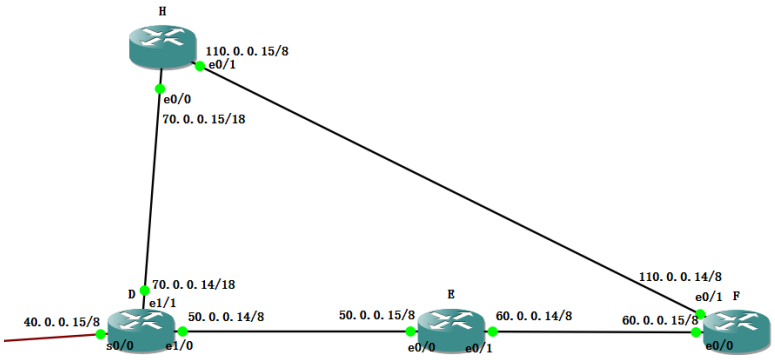
路由器I的e0/0接口no shutdown恢复链路，观察到路由器A重新学习了失效的路由条目

```
A#debug ip rip
RIP protocol debugging is on
A#
*Mar  1 05:08:58.066: RIP: sending v2 update to 224.0.0.9 via Ethernet0/1 (10.0.0.14)
*Mar  1 05:08:58.066: RIP: build update entries
*Mar  1 05:08:58.066:   10.0.0.0/8 via 0.0.0.0, metric 1, tag 0
*Mar  1 05:08:58.070:   20.0.0.0/8 via 0.0.0.0, metric 2, tag 0
*Mar  1 05:08:58.070:   30.0.0.0/8 via 0.0.0.0, metric 2, tag 0
*Mar  1 05:08:58.070:   40.0.0.0/8 via 0.0.0.0, metric 3, tag 0
*Mar  1 05:08:58.070:   50.0.0.0/8 via 0.0.0.0, metric 4, tag 0
*Mar  1 05:08:58.074:   60.0.0.0/8 via 0.0.0.0, metric 5, tag 0
*Mar  1 05:08:58.074:   70.0.0.0/8 via 0.0.0.0, metric 4, tag 0
*Mar  1 05:08:58.074:   80.0.0.0/8 via 0.0.0.0, metric 3, tag 0
A#
*Mar  1 05:08:58.078:   90.0.0.0/8 via 0.0.0.0, metric 4, tag 0
*Mar  1 05:08:58.078:   100.0.0.0/8 via 0.0.0.0, metric 1, tag 0
```

(4)水平分割的影响

水平分割方法让路由器记住每一条路由信息的来源，即标记收到该路由新的端口号，当路由器向外广播路由信息时，不会将该路由信息向收到该信息的端口转发。

构造一个冗余链路的拓扑，连接路由器H和F并配置RIP



默认启用水平分割，在路由器A上验证初始状态

```
A#debug ip rip
RIP protocol debugging is on
A#
*Mar  1 05:54:59.386: RIP: sending v2 update to 224.0.0.9 via Ethernet0/0 (100.0.0.14)
*Mar  1 05:54:59.386: RIP: build update entries
*Mar  1 05:54:59.386:   10.0.0.0/8 via 0.0.0.0, metric 1, tag 0
*Mar  1 05:54:59.390:   20.0.0.0/8 via 0.0.0.0, metric 2, tag 0
*Mar  1 05:54:59.390:   30.0.0.0/8 via 0.0.0.0, metric 2, tag 0
*Mar  1 05:54:59.390:   40.0.0.0/8 via 0.0.0.0, metric 3, tag 0
*Mar  1 05:54:59.390:   50.0.0.0/8 via 0.0.0.0, metric 4, tag 0
*Mar  1 05:54:59.394:   60.0.0.0/8 via 0.0.0.0, metric 5, tag 0
*Mar  1 05:54:59.394:   70.0.0.0/8 via 0.0.0.0, metric 4, tag 0
*Mar  1 05:54:59.394:   80.0.0.0/8 via 0.0.0.0, metric 3, tag 0
A#
*Mar  1 05:54:59.398:   90.0.0.0/8 via 0.0.0.0, metric 4, tag 0
*Mar  1 05:54:59.398:   110.0.0.0/8 via 0.0.0.0, metric 5, tag 0
A#
*Mar  1 05:55:10.742: RIP: sending v2 update to 224.0.0.9 via Ethernet0/1 (10.0.0.14)
*Mar  1 05:55:10.742: RIP: build update entries
*Mar  1 05:55:10.746:   100.0.0.0/8 via 0.0.0.0, metric 1, tag 0
```

```
A#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 [120/3] via 10.0.0.15, 00:00:16, Ethernet0/1
     100.0.0.0/24 is subnetted, 1 subnets
C       100.0.0.0 is directly connected, Ethernet0/0
R    70.0.0.0/8 [120/3] via 10.0.0.15, 00:00:16, Ethernet0/1
R    80.0.0.0/8 [120/2] via 10.0.0.15, 00:00:16, Ethernet0/1
R    20.0.0.0/8 [120/1] via 10.0.0.15, 00:00:16, Ethernet0/1
R    110.0.0.0/8 [120/4] via 10.0.0.15, 00:00:16, Ethernet0/1
R    40.0.0.0/8 [120/2] via 10.0.0.15, 00:00:16, Ethernet0/1
C    10.0.0.0/8 is directly connected, Ethernet0/1
R    90.0.0.0/8 [120/3] via 10.0.0.15, 00:00:18, Ethernet0/1
R    60.0.0.0/8 [120/4] via 10.0.0.15, 00:00:18, Ethernet0/1
R    30.0.0.0/8 [120/1] via 10.0.0.15, 00:00:18, Ethernet0/1
S*   0.0.0.0/0 is directly connected, Ethernet0/1
```

关闭路由器H和F各自的e0/1接口的水平分割

```
H#config t
Enter configuration commands, one per line.  End with CNTL/Z.
H(config)#interface e0/1
H(config-if)#no ip split-horizon
```

在路由器A上debug查看调试输出

```
A#debug ip rip
RIP protocol debugging is on
A#
*Mar  1 06:14:59.266: RIP: received v2 update from 10.0.0.15 on Ethernet0/1
*Mar  1 06:14:59.266:      20.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 06:14:59.270:      30.0.0.0/8 via 0.0.0.0 in 1 hops
*Mar  1 06:14:59.270:      40.0.0.0/8 via 0.0.0.0 in 2 hops
*Mar  1 06:14:59.270:      50.0.0.0/8 via 0.0.0.0 in 3 hops
*Mar  1 06:14:59.274:      60.0.0.0/8 via 0.0.0.0 in 4 hops
*Mar  1 06:14:59.274:      70.0.0.0/8 via 0.0.0.0 in 3 hops
*Mar  1 06:14:59.274:      80.0.0.0/8 via 0.0.0.0 in 2 hops
*Mar  1 06:14:59.274:      90.0.0.0/8 via 0.0.0.0 in 3 hops
*Mar  1 06:14:59.278:      110.0.0.0/8 via 0.0.0.0 in 4 hops
A#
*Mar  1 06:15:05.214: RIP: sending v2 update to 224.0.0.9 via Ethernet0/1 (10.0.0.14)
*Mar  1 06:15:05.214: RIP: build update entries
*Mar  1 06:15:05.218:   100.0.0.0/8 via 0.0.0.0, metric 1, tag 0
*Mar  1 06:15:05.314: RIP: sending v2 update to 224.0.0.9 via Ethernet0/0 (100.0.0.14)
*Mar  1 06:15:05.314: RIP: build update entries
*Mar  1 06:15:05.314:   10.0.0.0/8 via 0.0.0.0, metric 1, tag 0
*Mar  1 06:15:05.314:   20.0.0.0/8 via 0.0.0.0, metric 2, tag 0
*Mar  1 06:15:05.318:   30.0.0.0/8 via 0.0.0.0, metric 2, tag 0
*Mar  1 06:15:05.318:   40.0.0.0/8 via 0.0.0.0, metric 3, tag 0
*Mar  1 06:15:05.318:   50.0.0.0/8 via 0.0.0.0, metric 4, tag 0
A#
*Mar  1 06:15:05.322:   60.0.0.0/8 via 0.0.0.0, metric 5, tag 0
*Mar  1 06:15:05.322:   70.0.0.0/8 via 0.0.0.0, metric 4, tag 0
*Mar  1 06:15:05.322:   80.0.0.0/8 via 0.0.0.0, metric 3, tag 0
*Mar  1 06:15:05.326:   90.0.0.0/8 via 0.0.0.0, metric 4, tag 0
*Mar  1 06:15:05.326:   110.0.0.0/8 via 0.0.0.0, metric 5, tag 0
```

发现目标网络110.0.0.0/8的跳数一直不稳定，在4和5之间波动，说明路由器之间的更新并没有完全收敛。

分析：关闭了水平分割，导致RIP更新消息的反向传播。此时环路中的路由器会不断将路由更新重新发送给相邻路由器，形成循环。由于环路中存在多条路径，每次更新中，路由器选择的路径可能不同，导致跳数在一定范围内波动。此外，由于RIP协议周期性更新，网络中的链路状态传播和路由器处理时间并非完全同步，这可能进一步加剧了路由跳数的不稳定性。

重新启用水平分割，再次调试路由器A并验证路由表

```
A#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 [120/3] via 10.0.0.15, 00:00:39, Ethernet0/1
     100.0.0.0/24 is subnetted, 1 subnets
C       100.0.0.0 is directly connected, Ethernet0/0
R    70.0.0.0/8 [120/3] via 10.0.0.15, 00:00:39, Ethernet0/1
R    80.0.0.0/8 [120/2] via 10.0.0.15, 00:00:39, Ethernet0/1
R    20.0.0.0/8 [120/1] via 10.0.0.15, 00:00:39, Ethernet0/1
R    110.0.0.0/8 [120/4] via 10.0.0.15, 00:00:39, Ethernet0/1
R    40.0.0.0/8 [120/2] via 10.0.0.15, 00:00:39, Ethernet0/1
C    10.0.0.0/8 is directly connected, Ethernet0/1
R    90.0.0.0/8 [120/3] via 10.0.0.15, 00:00:41, Ethernet0/1
R    60.0.0.0/8 [120/4] via 10.0.0.15, 00:00:41, Ethernet0/1
R    30.0.0.0/8 [120/1] via 10.0.0.15, 00:00:41, Ethernet0/1
S*   0.0.0.0/0 is directly connected, Ethernet0/1
```

观察到目标网络110.0.0.0/8的跳数稳定在4，且路由表中目标网络的跳数同样为4。说明反向传播被阻止，环路内的更新传播被切断，路由器可以正确选择最优路径，更新快速收敛。

4.2 OSPF 路由协议流程分析

(1)初始信息交互

所有路由器配置好 OSPF 协议后，在路由器 A 上执行 debug ip ospf events

```
A#debug ip ospf events
OSPF events debugging is on
A#
*Mar  1 01:32:40.587: OSPF: Rcv hello from 30.0.0.14 area 0 from Ethernet0/1 10.
0.0.15
*Mar  1 01:32:40.587: OSPF: End of hello processing
*Mar  1 01:32:40.987: OSPF: Send hello to 224.0.0.5 area 0 on Ethernet0/1 from 1
0.0.0.14
```

分析：收到邻居 30.0.0.14 的 Hello 报文，接口为 e0/1，来源 IP 为 10.0.0.15；完成 Hello 报文的处理，确认与邻居的参数一致性，如果一致，邻居关系状态将进一步发展；多播从 e0/1 接口发送 Hello 报文，发送 Hello 报文是维持 OSPF 邻居关系的周期性任务。

问题 1：为什么打开 debug 信息后只能看到 hello 包，看不到 OSPF 协议的其他交互信息?

回答：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。由于打开了 debug 信息，配置时就会弹出各种显示信息，影响配置的过程。

确认邻居关系

```
A#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
30.0.0.14         1   FULL/BDR        00:00:15    10.0.0.15       Ethernet0/1
```

邻居状态为 FULL 意味着路由器 A 和邻居路由器已经完成 LSDB 的同步。

(2)路由计算

OSPF路由器的路由选择就是基于整个域中的所有链路状态，通过Dijkstra算法建立起以本路由器为根的最短路径树。然后通过计算域间路由、自治系统外部路由来完成整个网络上的路由表。

```
A#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

O    50.0.0.0/8 [110/94] via 10.0.0.15, 00:01:51, Ethernet0/1
     100.0.0.0/24 is subnetted, 1 subnets
C       100.0.0.0 is directly connected, Ethernet0/0
O    70.0.0.0/8 [110/94] via 10.0.0.15, 00:01:51, Ethernet0/1
O    80.0.0.0/8 [110/30] via 10.0.0.15, 00:01:51, Ethernet0/1
O    20.0.0.0/8 [110/20] via 10.0.0.15, 00:01:51, Ethernet0/1
O    40.0.0.0/8 [110/84] via 10.0.0.15, 00:01:51, Ethernet0/1
C    10.0.0.0/8 is directly connected, Ethernet0/1
O    90.0.0.0/8 [110/40] via 10.0.0.15, 00:01:52, Ethernet0/1
O    60.0.0.0/8 [110/104] via 10.0.0.15, 00:01:52, Ethernet0/1
O    30.0.0.0/8 [110/20] via 10.0.0.15, 00:01:52, Ethernet0/1
S*   0.0.0.0/0 is directly connected, Ethernet0/1
```

问题 2：为什么修改了某条链路的状态后，仍然看不到交互的链路状态信息？

回答：由于洪泛的信息量很大，默认情况下会将其禁止掉，而在 OSPF 协议中链路状态发生变化时，是采用洪泛的方式将变化信息向全网发送，所以要用 debug ip ospf flood 打开洪泛信息的调试，就可以看到链路状态更新时发送的信息。

(3)链路故障处理

在路由器A上查看初始状态，确保A和B之间的OSPF配置正常（邻居关系建立为FULL，网络处于正常状态）

并启用debug ip ospf adj, debug ip ospf lsa-generation, debug ip ospf events

```
A#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
30.0.0.14         1   FULL/BDR        00:00:17    10.0.0.15       Ethernet0/1
```

```
A#show ip ospf database

            OSPF Router with ID (100.0.0.14) (Process ID 10)

                Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum Link count
20.0.0.15       20.0.0.15       296         0x80000008 0x009EF8 1
30.0.0.14       30.0.0.14       486         0x80000009 0x00B216 3
60.0.0.14       60.0.0.14       359         0x80000009 0x0082DA 2
70.0.0.14       70.0.0.14       262         0x80000008 0x006773 4
80.0.0.14       80.0.0.14       440         0x80000009 0x00C71B 4
90.0.0.14       90.0.0.14       513         0x80000009 0x001196 2
90.0.0.15       90.0.0.15       222         0x80000008 0x008FED 1
100.0.0.14      100.0.0.14      310         0x80000008 0x000509 1
110.0.0.14      110.0.0.14      256         0x80000008 0x00C7CB 1
110.0.0.15      110.0.0.15      262         0x80000008 0x006C11 1

                Net Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum
10.0.0.14       100.0.0.14      310         0x80000007 0x000407
20.0.0.14       30.0.0.14       487         0x80000007 0x00ADE8
30.0.0.15       80.0.0.14       441         0x80000007 0x001D01
50.0.0.14       70.0.0.14       263         0x80000007 0x00817F
60.0.0.15       110.0.0.14      260         0x80000007 0x00A401
70.0.0.15       110.0.0.15      266         0x80000007 0x0094FA
80.0.0.15       90.0.0.14       517         0x80000007 0x00A401
90.0.0.15       90.0.0.15       227         0x80000007 0x0094FA
```

```
O    50.0.0.0/8 [110/94] via 10.0.0.15, 03:25:54, Ethernet0/1
     100.0.0.0/24 is subnetted, 1 subnets
C       100.0.0.0 is directly connected, Ethernet0/0
O    70.0.0.0/8 [110/94] via 10.0.0.15, 03:25:54, Ethernet0/1
O    80.0.0.0/8 [110/30] via 10.0.0.15, 03:25:54, Ethernet0/1
O    20.0.0.0/8 [110/20] via 10.0.0.15, 03:25:54, Ethernet0/1
O    40.0.0.0/8 [110/84] via 10.0.0.15, 03:25:54, Ethernet0/1
C    10.0.0.0/8 is directly connected, Ethernet0/1
O    90.0.0.0/8 [110/40] via 10.0.0.15, 03:25:55, Ethernet0/1
O    60.0.0.0/8 [110/104] via 10.0.0.15, 03:25:55, Ethernet0/1
O    30.0.0.0/8 [110/20] via 10.0.0.15, 03:25:55, Ethernet0/1
S*   0.0.0.0/0 is directly connected, Ethernet0/1
```

然后，关闭路由器A的e0/1接口模拟链路故障

```
A(config-if)#shutdown
A(config-if)#
*Mar  1 03:58:16.371: OSPF: Send hello to 224.0.0.5 area 0 on Ethernet0/1 from 10.0.0.14
*Mar  1 03:58:17.311: OSPF: Interface Ethernet0/1 going Down
*Mar  1 03:58:17.315: OSPF: 100.0.0.14 address 10.0.0.14 on Ethernet0/1 is dead, state DOWN
*Mar  1 03:58:17.315: OSPF: Neighbor change Event on interface Ethernet0/1
*Mar  1 03:58:17.315: OSPF: DR/BDR election on Ethernet0/1
*Mar  1 03:58:17.315: OSPF: Elect BDR 30.0.0.14
*Mar  1 03:58:17.315: OSPF: Elect DR 30.0.0.14
*Mar  1 03:58:17.319: OSPF: Elect BDR 30.0.0.14
*Mar  1 03:58:17.319: OSPF: Elect DR 30.0.0.14
*Mar  1 03:58:17.319:       DR: 30.0.0.14 (Id)   BDR: 30.0.0.14 (Id)
*Mar  1 03:58:17.319: OSPF: Flush network LSA immediately
*Mar  1 03:58:17.319: OSPF: Remember old DR 100.0.0.14 (id)
*Mar  1 03:58:17.323: OSPF: 30.0.0.14 address 10.0.0.15 on Ethernet0/1 is dead, state DOWN
*Mar  1 03:58:17.323: %OSPF-5-ADJCHG: Process 10, Nbr 30.0.0.14 on Ethernet0/1 from FULL to DOWN, Neig
hbor Down: Interface down or detached
*Mar  1 03:58:17.323: OSPF: Neighbor change Event on interface Ethernet0/1
*Mar  1 03:58:17.323: OSPF: DR/BDR election on Ethernet0/1
*Mar  1 03:58:17.323: OSPF: Elect BDR 0.0.0.0
*Mar  1 03:58:17.323: OSPF: Elect DR 0.0.0.0
*Mar  1 03:58:17.323:       DR: none    BDR: none
*Mar  1 03:58:17.323: OSPF: Remember old DR 30.0.0.14 (id)
*Mar  1 03:58:17.819: OSPF: We are not DR to build Net Lsa for interface Ethernet0/1
*Mar  1 03:58:17.819: OSPF: Build network LSA for Ethernet0/1, router ID 100.0.0.14
*Mar  1 03:58:17.819: OSPF: Build network LSA for Ethernet0/1, router ID 100.0.0.14
*Mar  1 03:58:17.823: OSPF: Build router LSA for area 0, router ID 100.0.0.14, seq 0x8000000A
A(config-if)#
A(config-if)#
*Mar  1 03:58:19.307: %LINK-5-CHANGED: Interface Ethernet0/1, changed state to administratively down
*Mar  1 03:58:20.307: %LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0/1, changed state to do
wn
A(config-if)#undebug
*Mar  1 03:58:27.819: OSPF: service_maxage: Trying to delete MAXAGE LSA
```

在A上观察到接口e0/1关闭，邻居关系发生变化变为DOWN。OSPF立即删除与e0/1相关的旧LSA，网络状态变化，生成新的LSA重新泛洪。接口e0/1被标记为administratively down，所有通过该接口的路由被撤销。最后，因为e0/1接口没有活跃邻居，OSPF无法为该接口生成LSA，意味着该链路不再对全网可见。

```
B#
*Mar  1 03:58:36.359: OSPF: 100.0.0.14 address 10.0.0.14 on Ethernet0/1 is dead
*Mar  1 03:58:36.359: OSPF: 100.0.0.14 address 10.0.0.14 on Ethernet0/1 is dead, state DOWN
*Mar  1 03:58:36.359: %OSPF-5-ADJCHG: Process 10, Nbr 100.0.0.14 on Ethernet0/1 from FULL to DOWN, Nei
ghbor Down: Dead timer expired
B#
*Mar  1 03:58:36.363: OSPF: Neighbor change Event on interface Ethernet0/1
*Mar  1 03:58:36.363: OSPF: DR/BDR election on Ethernet0/1
*Mar  1 03:58:36.363: OSPF: Elect BDR 30.0.0.14
*Mar  1 03:58:36.363: OSPF: Elect DR 30.0.0.14
*Mar  1 03:58:36.363: OSPF: Elect BDR 0.0.0.0
*Mar  1 03:58:36.363: OSPF: Elect DR 30.0.0.14
*Mar  1 03:58:36.367:       DR: 30.0.0.14 (Id)   BDR: none
*Mar  1 03:58:36.367: OSPF: Remember old DR 100.0.0.14 (id)
*Mar  1 03:58:36.755: OSPF: Rcv hello from 20.0.0.15 area 0 from Ethernet0/0 20.0.0.15
*Mar  1 03:58:36.755: OSPF: End of hello processing
*Mar  1 03:58:36.863: OSPF: Build router LSA for area 0, router ID 30.0.0.14, seq 0x8000000B
B#
*Mar  1 03:58:36.867: OSPF: No full nbrs to build Net Lsa for interface Ethernet0/1
```

在B上观察到邻居100.0.0.14被标记为dead，OSPF检测不到Hello报文，触发超时机制，将100.0.0.14标记为DOWN，即到A的链路失效。邻居状态变更同时触发网络的重新收敛过程。接口e0/1进行DE/BDR选举，此时网络中只有一个邻居30.0.0.14，它被选为DR和BDR，表明该链路上不再存在其他路由器。

接着，开启路由器A的e0/1接口恢复链路

```
*Mar  1 04:30:40.206: OSPF: Interface Ethernet0/1 going Up
*Mar  1 04:30:40.206: OSPF: Send hello to 224.0.0.5 area 0 on Ethernet0/1 from 10.0.0.14
*Mar  1 04:30:40.706: OSPF: Schedule SPF in area 0
      Change in LS ID 100.0.0.14, LSA type R, , spf-type Full
*Mar  1 04:30:40.706: OSPF: Build router LSA for area 0, router ID 100.0.0.14, seq 0x8000000C
```

链路已经恢复，触发OSPF开始对该接口执行邻居发现和状态更新的操作

```
*Mar  1 04:30:42.334: OSPF: 2 Way Communication to 100.0.0.14 on Ethernet0/1, state 2WAY
*Mar  1 04:30:42.338: OSPF: Neighbor change Event on interface Ethernet0/1
*Mar  1 04:30:42.338: OSPF: DR/BDR election on Ethernet0/1
```

恢复链路后，OSPF通过Hello消息重新发现邻居，并进入2WAY状态（建立了双向通信）2WAY是邻居关系进一步升级的前提，之后需要进行DR/BDR选举。

```
*Mar  1 04:30:42.342:        DR: 30.0.0.14 (Id)   BDR: 100.0.0.14 (Id)
*Mar  1 04:30:42.342: OSPF: End of hello processing
*Mar  1 04:30:42.342: OSPF: Rcv DBD from 100.0.0.14 on Ethernet0/1 seq 0x6AA opt 0x52 flag 0x7 len 32
 mtu 1500 state EXSTART
*Mar  1 04:30:42.342: OSPF: NBR Negotiation Done. We are the SLAVE
*Mar  1 04:30:42.342: OSPF: Send DBD to 100.0.0.14 on Ethernet0/1 seq 0x6AA opt 0x52 flag 0x2 len 392
*Mar  1 04:30:42.354: OSPF: Rcv DBD from 100.0.0.14 on Ethernet0/1 seq 0x6AB opt 0x52 flag 0x3 len 372
  mtu 1500 state EXCHANGE
*Mar  1 04:30:42.354: OSPF: Send DBD to 100.0.0.14 on Ethernet0/1 seq 0x6AB opt 0x52 flag 0x0 len 32
*Mar  1 04:30:42.366: OSPF: Rcv DBD from 100.0.0.14 on Ethernet0/1 seq 0x6AC opt 0x52 flag 0x1 len 32
 mtu 1500 state EXCHANGE
*Mar  1 04:30:42.366: OSPF: Exchange Done with 100.0.0.14 on Ethernet0/1
*Mar  1 04:30:42.370: OSPF: Send LS REQ to 100.0.0.14 length 12 LSA count 1
*Mar  1 04:30:42.370: OSPF: Send DBD to 100.0.0.14 on Ethernet0/1 seq 0x6AC opt 0x52 flag 0x0 len 32
*Mar  1 04:30:42.374: OSPF: Rcv LS REQ from 100.0.0.14 on Ethernet0/1 length 228 LSA count 17
*Mar  1 04:30:42.374: OSPF: Send UPD to 10.0.0.14 on Ethernet0/1 length 704 LSA count 17
*Mar  1 04:30:42.406: OSPF: Rcv LS UPD from 100.0.0.14 on Ethernet0/1 length 64 LSA count 1
*Mar  1 04:30:42.406: OSPF: Detect change in LSA type 1, LSID 100.0.0.14, from 100.0.0.14 area 0
*Mar  1 04:30:42.410: OSPF: Synchronized with 100.0.0.14 on Ethernet0/1, state FULL
*Mar  1 04:30:42.410: %OSPF-5-ADJCHG: Process 10, Nbr 100.0.0.14 on Ethernet0/1 from LOADING to FULL,
Loading Done
```

选举结果为DR：30.0.0.14；BDR：100.0.0.14

DBD包交换，协商完成进入LOADING状态

```
*Mar  1 04:30:42.414: OSPF: Schedule SPF in area 0
      Change in LS ID 100.0.0.14, LSA type R, , spf-type Full
*Mar  1 04:30:42.414: OSPF: Rcv LS UPD from 100.0.0.14 on Ethernet0/1 length 60 LSA count 1
*Mar  1 04:30:42.414: OSPF: Detect change in LSA type 2, LSID 10.0.0.14, from 100.0.0.14 area 0
*Mar  1 04:30:42.418: OSPF: Detect MAXAGE in LSA type 2, LS ID 10.0.0.14, from 100.0.0.14
*Mar  1 04:30:42.418: OSPF: Schedule SPF in area 0
      Change in LS ID 10.0.0.14, LSA type N, , spf-type Full
```

OSPF检测到链路状态的变化，触发了SPF计算，spf-type Full表明这是一次完整的SPF计算。多次调度SPF计算以重新生成最短路径树。

```
*Mar  1 04:30:47.422: OSPF: running SPF for area 0, SPF-type Full
*Mar  1 04:30:47.422: OSPF: Initializing to run spf
*Mar  1 04:30:47.422:  OSPF - spf_intra() - rebuilding the tree
*Mar  1 04:30:47.422:  It is a router LSA 30.0.0.14. Link Count 3
*Mar  1 04:30:47.426:   Processing link 0, id 30.0.0.15, link data 30.0.0.14, type 2
*Mar  1 04:30:47.426:    Add better path to LSA ID 30.0.0.15, gateway 30.0.0.14, dist 10
*Mar  1 04:30:47.426: OSPF: putting LSA on the clist LSID 30.0.0.15, Type 2, Adv Rtr. 80.0.0.14
*Mar  1 04:30:47.426:    Add path: next-hop 30.0.0.14, interface Ethernet0/2
```

spf_intra()通过对 LSDB 中的Router LSA和Network LSA执行Dijkstra算法，重建最短路径树，根据计算结果，更新路由表中的下一跳信息。

```
A#show ip ospf neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
30.0.0.14         1   FULL/DR         00:00:19    10.0.0.15       Ethernet0/1
```

查看路由器A，与B邻居关系为FULL。

问题1：OSPF中数据库同步信息的格式和同步对象？

回答：OSPF数据库同步主要通过LSA实现，数据同步通过DBD报文实现，用于交换LSDB的摘要信息。同步对象是LSDB，路由器通过LSA泛洪保证整个area的LSDB一致性。

问题2：链路改变信息如何发送，具体格式？

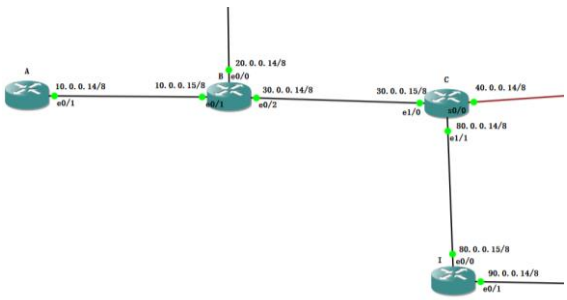
回答：链路断开：OSPF 通过 Hello 报文检测邻居是否存活，当超时且未收到 Hello 报文时，触发链路故障处理，Router LSA 更新, 受影响的接口和链路相关的 LSA 被标记为 MAXAGE, 并立即从 LSDB 中移除，同时触发 LSA 泛洪, 邻居关系状态变为 DOWN，引发 DR/BDR 重新选举；链路恢复：接口重新启用后，通过发送 Hello 报文重新发现邻居，进入 2WAY 状态，执行 DR/BDR 选举，发 DBD 包交换摘要信息，逐步进入 LOADING 和 FULL 状态。检测到 LSDB 更新后，通过 Dijkstra 算法重新生成最短路径树，更新路由表。

5 实验问题思考

思考题一：

实验中，采用下一跳和转发接口这两种方式配置的静态路由有什么区别？会导致在你的拓扑结构中从ping时的丢包数有什么变化？需要用你的拓扑路由器的ARP表中的内容来解释，要附截图。（ping时，间隔至少两台路由器）

在以太网中，两个相邻知道接口之间的通信是依靠MAC地址。相邻接口通信时，需要知道对方的MAC地址，根据MAC地址将通信数据转换成数据帧后交付给网络，进而发送到到对方。而对方MAC地址的获得，是通过第二层交换机广播，由ARP协议完成的。



使用实验2的拓扑图和IP配置，用路由器A ping 80.0.0.15

(1) 下一跳

```
A#show ip arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.0.0.14               -   cc07.1825.0001  ARPA   Ethernet0/1
A#ping 80.0.0.15

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 80.0.0.15, timeout is 2 seconds:
...!!
Success rate is 40 percent (2/5), round-trip min/avg/max = 108/108/108 ms
A#show ip arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.0.0.14               -   cc07.1825.0001  ARPA   Ethernet0/1
Internet  10.0.0.15               0   cc06.1803.0001  ARPA   Ethernet0/1
```

ARP表更新，显示加入下一跳（10.0.0.15）的MAC地址记录

```
2 2.909843    cc:07:18:25:00:01   Broadcast           ARP    60 Who has 10.0.0.15? Tell 10.0.0.14
3 2.919941    cc:06:18:03:00:01   cc:07:18:25:00:01   ARP    60 10.0.0.15 is at cc:06:18:03:00:01
```

广播询问的是下一跳地址

```
4 4.856129    10.0.0.14   80.0.0.15   ICMP   114 Echo (ping) request  id=0x0000, seq=1/256, ttl=255 (no response found!)
5 6.933846    10.0.0.14   80.0.0.15   ICMP   114 Echo (ping) request  id=0x0000, seq=2/512, ttl=255 (no response found!)
6 9.022743    10.0.0.14   80.0.0.15   ICMP   114 Echo (ping) request  id=0x0000, seq=3/768, ttl=255 (reply in 7)
7 9.077810    80.0.0.15   10.0.0.14   ICMP   114 Echo (ping) reply    id=0x0000, seq=3/768, ttl=253 (request in 6)
8 9.088102    10.0.0.14   80.0.0.15   ICMP   114 Echo (ping) request  id=0x0000, seq=4/1024, ttl=255 (reply in 9)
9 9.128555    80.0.0.15   10.0.0.14   ICMP   114 Echo (ping) reply    id=0x0000, seq=4/1024, ttl=253 (request in 8)
```

丢2个包

(2) 转发接口

```
A#show ip arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.0.0.14               -   cc07.1825.0001  ARPA   Ethernet0/1
A#ping 80.0.0.15

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 80.0.0.15, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
A#ping 80.0.0.15

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 80.0.0.15, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 92/115/132 ms
A#show ip arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  10.0.0.14               -   cc07.1825.0001  ARPA   Ethernet0/1
Internet  80.0.0.15               0   cc06.1803.0001  ARPA   Ethernet0/1
```

ARP表更新，直接通过接口转发，加入80.0.0.15的ARP表项

```
4 10.424396     cc:07:18:25:00:01    Broadcast            ARP         60 Who has 80.0.0.15? Tell 10.0.0.14
5 10.434850     cc:06:18:03:00:01    cc:07:18:25:00:01    ARP         60 80.0.0.15 is at cc:06:18:03:00:01
```

广播询问的是目标地址，丢4个包

思考题二：

对照所截获的消息，说明OSPF协议和RIP协议在邻居发现和数据库同步等部分中消息传递方式和消息内容上的差异。附截图和对消息的说明。

OSPF协议和RIP协议的消息传递过程在上文分析过，此处主要进行消息内容分析及协议对比。

(1) OSPF

邻居发现：通过定期发送 Hello 消息监测邻居路由器的连通性，并在邻居关系发生变化时触发 重新计算最短路径。邻居关系从 Init 到 2Way，再到 Full，通过多阶段状态变化完成。

Hello消息包括 网络掩码, Hello和Dead时间间隔, Router优先级, Router ID, 活跃邻居

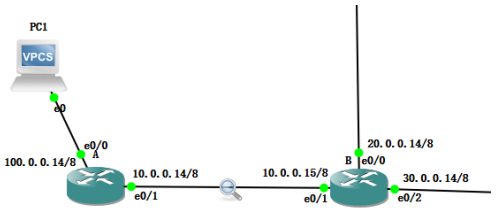
```
> Frame 5: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface -, id 0
> Ethernet II, Src: cc:06:18:03:00:01 (cc:06:18:03:00:01), Dst: IPv4mcast_05 (01:00:5e:00:00:05)
> Internet Protocol Version 4, Src: 10.0.0.15, Dst: 224.0.0.5
v Open Shortest Path First
  > OSPF Header
  v OSPF Hello Packet
      Network Mask: 255.0.0.0
      Hello Interval [sec]: 5
    > Options: 0x12, (L) LLS Data block, (E) External Routing
      Router Priority: 1
      Router Dead Interval [sec]: 20
      Designated Router: 10.0.0.14
      Backup Designated Router: 10.0.0.15
      Active Neighbor: 100.0.0.14
  > OSPF LLS Data Block
```

数据库同步：使用 LSA 更新数据库。邻居建立后，通过 DBD 包（包含LSAs的摘要）同步路由器的 LSDB。

LSA包括 Link State ID, Advertising Router（通告的路由器）, Metric（到目标的开销）有更新和确认报文。

```
> Frame 19: 110 bytes on wire (880 bits), 110 bytes captured (880 bits) on interface -, id 0
> Ethernet II, Src: cc:06:18:03:00:01 (cc:06:18:03:00:01), Dst: IPv4mcast_05 (01:00:5e:00:00:05)
> Internet Protocol Version 4, Src: 10.0.0.15, Dst: 224.0.0.5
v Open Shortest Path First
  > OSPF Header
  v LS Update Packet
      Number of LSAs: 1
    v LSA-type 1 (Router-LSA), len 48
        .000 0000 0000 0001 = LS Age (seconds): 1
        0... .... .... .... = Do Not Age Flag: 0
      > Options: 0x22, (DC) Demand Circuits, (E) External Routing
        LS Type: Router-LSA (1)
        Link State ID: 30.0.0.14
        Advertising Router: 30.0.0.14
        Sequence Number: 0x80000004
        Checksum: 0xaf90
        Length: 48
      > Flags: 0x00
        Number of Links: 2
      > Type: Transit  ID: 20.0.0.14       Data: 20.0.0.14       Metric: 10
      > Type: Transit  ID: 10.0.0.14       Data: 10.0.0.15       Metric: 10
```

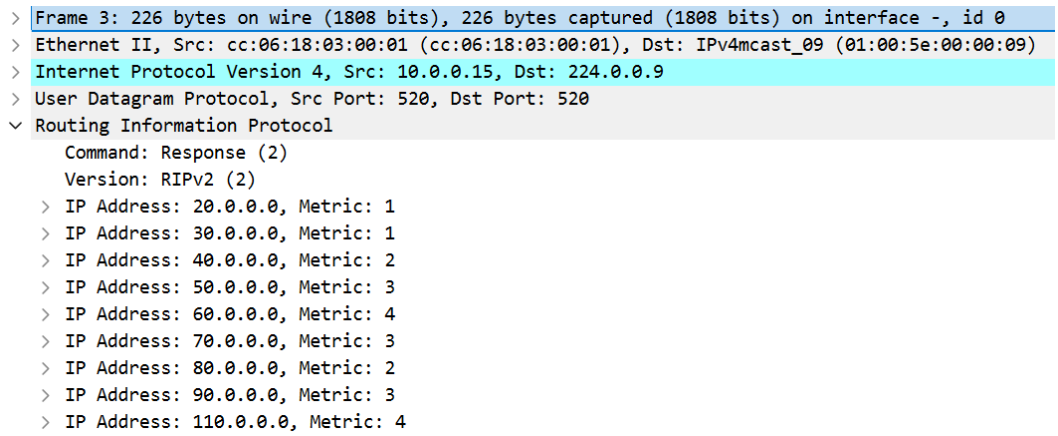
（此处进行了关闭路由器B的e0/2接口再恢复的操作，在路由器A debug，抓包A e0/1 to B e0/1）



19 24.274709	10.0.0.15	224.0.0.5	OSPF	110 LS Update
20 25.011880	10.0.0.14	224.0.0.5	OSPF	94 Hello Packet
21 25.507630	10.0.0.15	224.0.0.5	OSPF	94 Hello Packet
22 26.778588	10.0.0.14	224.0.0.5	OSPF	78 LS Acknowledge
23 29.990913	10.0.0.14	224.0.0.5	OSPF	94 Hello Packet
24 30.501840	10.0.0.15	224.0.0.5	OSPF	94 Hello Packet
25 30.936324	cc:07:18:25:00:01	cc:07:18:25:00:01	LOOP	60 Reply
26 30.976051	cc:06:18:03:00:01	cc:06:18:03:00:01	LOOP	60 Reply
27 34.999698	10.0.0.14	224.0.0.5	OSPF	94 Hello Packet
28 35.500733	10.0.0.15	224.0.0.5	OSPF	94 Hello Packet
29 38.075932	10.0.0.15	224.0.0.5	OSPF	122 LS Update
30 39.996598	10.0.0.14	224.0.0.5	OSPF	94 Hello Packet
31 40.523410	10.0.0.15	224.0.0.5	OSPF	94 Hello Packet
32 40.594612	10.0.0.14	224.0.0.5	OSPF	78 LS Acknowledge

(2) RIP

邻居发现：通过周期性广播（多播）路由更新消息进行邻居发现，不建立明确的邻居关系，所有接收到更新的设备都视为邻居。邻居关系是隐式的，不涉及状态转换。



数据库同步：每30s广播（多播）路由表更新，消息包括目标网络地址，距离向量（跳数）。直接覆盖旧路由表项，不校验数据一致性。

思考题三：

写出在你的拓扑中，数据包从某台PC发送给其他PC完整过程（考虑各种不同情况），阐述ARP过程和路由表匹配过程以及链路层协议封装过程，附相关截图。

(1) ARP过程

PC A通过查找ARP缓存判断是否有PC B的MAC地址：

如果ARP缓存中有MAC地址，直接使用。

如果没有，

- PC A和PC B在同一子网

  PC A发送广播ARP请求（包含PC B的IP地址）

- PC A和PC B不在同一子网

  PC A与默认网关通信

PC B收到ARP请求后，发送ARP回复，告知其MAC地址；PC A更新ARP缓存。

(2) 路由表匹配过程

- PC A和PC B在同一子网

PC A查找本地路由表，发现PC B属于同一子网，直接交付。

- PC A和PC B不在同一子网

PC A将数据包封装到以太网帧，目标MAC为网关MAC，网关处理解封装以太网帧，根据路由表匹配找到下一跳。经过路由器转发到接口（如果没有记录仍发送ARP查询）路由器通过ARP广播请求PC B的MAC地址并加入ARP表。

(3) 链路层协议封装过程

1. 从网络层接收数据包
2. 确定目标MAC地址（ARP）
3. 添加header
4. 添加数据部分
5. 计算并添加FCS
6. 传递到物理层

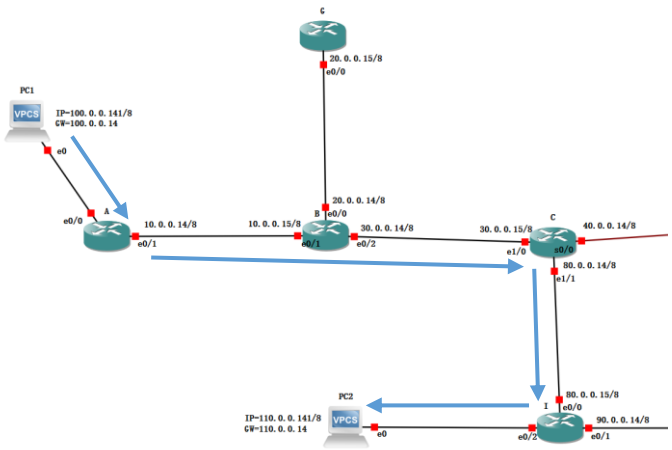
- PC A和PC B在同一子网

PC A封装以太网帧时，目标MAC为PC B的MAC。

- PC A和PC B不在同一子网

PC A封装以太网帧时，目标MAC为网关的MAC，路由器解封装帧，匹配目标IP地址后重新封装，目标MAC地址为PC B的MAC地址。

实验在RIP协议的拓扑基础上完成，PC1和PC2不在同一子网。实验截图：



```
PC1> ping 110.0.0.141

84 bytes from 110.0.0.141 icmp_seq=1 ttl=60 time=68.762 ms
84 bytes from 110.0.0.141 icmp_seq=2 ttl=60 time=58.826 ms
84 bytes from 110.0.0.141 icmp_seq=3 ttl=60 time=53.138 ms
84 bytes from 110.0.0.141 icmp_seq=4 ttl=60 time=390.010 ms
84 bytes from 110.0.0.141 icmp_seq=5 ttl=60 time=71.296 ms
```

11	36.256959	100.0.0.141	110.0.0.141	ICMP	98 Echo (ping) request id=0x16e4, seq=1/256, ttl=63 (reply in 12)
12	36.308250	110.0.0.141	100.0.0.141	ICMP	98 Echo (ping) reply id=0x16e4, seq=1/256, ttl=61 (request in 11)
13	37.326754	100.0.0.141	110.0.0.141	ICMP	98 Echo (ping) request id=0x17e4, seq=2/512, ttl=63 (reply in 14)
14	37.367334	110.0.0.141	100.0.0.141	ICMP	98 Echo (ping) reply id=0x17e4, seq=2/512, ttl=61 (request in 13)
15	38.379627	100.0.0.141	110.0.0.141	ICMP	98 Echo (ping) request id=0x18e4, seq=3/768, ttl=63 (reply in 16)
16	38.420363	110.0.0.141	100.0.0.141	ICMP	98 Echo (ping) reply id=0x18e4, seq=3/768, ttl=61 (request in 15)
17	39.131135	10.0.0.14	224.0.0.9	RIPv2	66 Response
18	39.433131	100.0.0.141	110.0.0.141	ICMP	98 Echo (ping) request id=0x19e4, seq=4/1024, ttl=63 (reply in 19)
19	39.545806	110.0.0.141	100.0.0.141	ICMP	98 Echo (ping) reply id=0x19e4, seq=4/1024, ttl=61 (request in 18)
20	40.832721	100.0.0.141	110.0.0.141	ICMP	98 Echo (ping) request id=0x1be4, seq=5/1280, ttl=63 (reply in 21)
21	40.883483	110.0.0.141	100.0.0.141	ICMP	98 Echo (ping) reply id=0x1be4, seq=5/1280, ttl=61 (request in 20)

Wireshark · 分组 11 · Standard input

```
> Frame 11: 98 bytes on wire (784 bits), 98 bytes captured (784 bits
> Ethernet II, Src: cc:07:4a:34:00:01 (cc:07:4a:34:00:01), Dst: cc:0
v Internet Protocol Version 4, Src: 100.0.0.141, Dst: 110.0.0.141
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0xe416 (58390)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 63
    Protocol: ICMP (1)
    Header Checksum: 0xc478 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 100.0.0.141
    Destination Address: 110.0.0.141
    [Stream index: 2]
> Internet Control Message Protocol
```

Wireshark · 分组 12 · Standard input

```
> Frame 12: 98 bytes on wire (784 bits), 98 bytes captured (784 bit
> Ethernet II, Src: cc:06:18:03:00:01 (cc:06:18:03:00:01), Dst: cc:
v Internet Protocol Version 4, Src: 110.0.0.141, Dst: 100.0.0.141
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0xe416 (58390)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 61
    Protocol: ICMP (1)
    Header Checksum: 0xc678 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 110.0.0.141
    Destination Address: 100.0.0.141
    [Stream index: 2]
> Internet Control Message Protocol
```

在OSPF/RIP路由下，由于路由器之间需要不断传递消息，所以路由器之间会长期持有ARP路由表。实验中路由器已经通过 RIP 确认了目标地址的下一跳和接口，则它可直接根据路由表进行转发，不需要发送新的ARP请求。

6 实验总结

本实验中实现了RIP和OSPF路由协议，对上学期学过的计算机网络理论知识有了更清晰的理解。OSPF协议和RIP协议无论从利用网络资源还是安全性来说都优于RIP，只是RIP更简单。一开始完成网络拓扑协议配置过程让我积累了一些配置和排错经验。这次实验需要从大量的debug信息深入分析协议工作过程，尤其是OSPF因为信息太多提取困难，需要对协议流程非常清楚才能准确定位。