**实验三：RIP和OSPF路由协议的配置及协议流程分析**

目录

[**RIP配置** 1](#_Toc153040584)

[**RIP协议配置和分析实验** 1](#_Toc153040585)

[**实验目的** 1](#_Toc153040586)

[**实验环境** 2](#_Toc153040587)

[**实验拓扑** 2](#_Toc153040588)

[**配置过程** 2](#_Toc153040589)

[**步骤1：** 2](#_Toc153040590)

[**步骤2：** 3](#_Toc153040591)

[**步骤3：** 4](#_Toc153040592)

[**步骤4：** 5](#_Toc153040593)

[**问题和分析** 6](#_Toc153040594)

[**OSPF配置** 6](#_Toc153040595)

[**OSPF协议配置和分析** 7](#_Toc153040596)

[**实验目的** 7](#_Toc153040597)

[**实验环境** 7](#_Toc153040598)

[**实验拓扑** 7](#_Toc153040599)

[**配置过程** 7](#_Toc153040600)

[**步骤1：** 7](#_Toc153040601)

[**步骤2：** 8](#_Toc153040602)

[**问题和分析** 9](#_Toc153040603)

[**动态路由——复杂拓扑** 9](#_Toc153040604)

[**实验拓扑** 9](#_Toc153040605)

[**实验要求** 10](#_Toc153040606)

[**实验内容** 10](#_Toc153040607)

[**实验步骤** 10](#_Toc153040608)

[**配置路由器C和D之间的串口连接** 10](#_Toc153040609)

[**配置路由器A-J的端口IP地址** 11](#_Toc153040610)

[**实现RIP路由协议** 11](#_Toc153040611)

[**通过debug信息详细描述RIP工作过程** 11](#_Toc153040612)

[**水平分割** 13](#_Toc153040613)

[**实现OSPF路由协议** 15](#_Toc153040614)

[**通过debug信息详细描述OSPF工作过程** 16](#_Toc153040615)

[**数据库同步信息的格式和同步对象** 18](#_Toc153040616)

[**思考题** 20](#_Toc153040617)

**RIP配置**

1. 在路由器B的e1/2和路由器C的e1/0上关闭水平分割

|  |
| --- |
| interface e1/2 no ip split-horizon |

1. 被动掉路由器B的e1/2口，B就不能把e1/1被shutdown掉的信息以flash update的形式发给C

|  |
| --- |
| router rip passive-interface e1/2 |

1. 关闭e1/1端口

|  |
| --- |
| interface e1/1 shutdown |

1. 关闭路由器B的e1/2口的被动模式

|  |
| --- |
| router rip no passive-interface e1/2 |

1. 在debug ip rip下，能够看到回路路由的产生

**RIP协议配置和分析实验**

**实验目的**

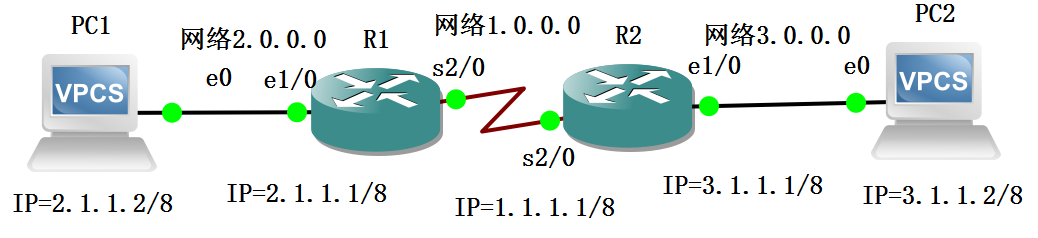
掌握路由器上RIP协议的配置方法，能够在模拟环境中进行路由器上RIP协议的配置，并能通过debug信息来分析RIP协议的工作过程。在RIP协议中观察没有配置水平分割和配置水平分割后协议工作流程的变化

**实验环境**

采用Dynamips仿真环境，前端采用GNS3进行图形拓扑设计

**实验拓扑**

采用2台路由器R1和R2将两台主机PC1和PC2互相连接在一起，两台路由器之间采用串口进行连接，两台PC与路由器之间都采用以太网进行连接



**配置过程**

首先在PC1、PC2、R1、R2的配置与静态路由配置完全一样，唯一不同的是在R1和R2上将静态路由删除掉，配置上RIP协议来自动计算路由

**步骤1：**

在R1的配置模式下配置RIP协议

|  |
| --- |
| R1(config) # Router rip；进入rip配置模式 R1(config) # version 2；配置版本为RIP v2 |

配置与R1直连的网络：

|  |
| --- |
| R1 (config-rip) # network 1.0.0.0 R1 (config-rip) # network 2.0.0.0 |

配置R1的邻居路由器：

|  |
| --- |
| R1 (config-rip) # neighbor 1.1.1.1 |

在R2上进行类似配置

|  |
| --- |
| R2 (config) # Router rip R2 (config) # version 2 R2 (config-rip) # network 1.0.0.0 R2 (config-rip) # network 3.0.0.0 R2 (config-rip) # neighbor 1.1.1.2 |

**步骤2：**

在R1和R2的特权模式下打开调试信息

|  |
| --- |
| R1 # Debug ip rip R2 # Debug ip rip |

这时会出现RIP协议的交互信息，根据这些信息对RIP协议的工作过程进行分析

***R1:***

*R1#debug ip rip*

*RIP protocol debugging is on*

*\*Mar 1 00:08:29.983: RIP: sending v2 update to 224.0.0.9 via Serial2/0 (1.1.1.1)*

*\*Mar 1 00:08:29.983: RIP: build update entries*

*\*Mar 1 00:08:29.983: 2.0.0.0/8 via 0.0.0.0, metric 1, tag 0*

*\*Mar 1 00:08:29.983: RIP: sending v2 update to 1.1.1.1 via Serial2/0 (1.1.1.1)*

*\*Mar 1 00:08:29.983: RIP: build update entries*

*\*Mar 1 00:08:29.983: 2.0.0.0/8 via 0.0.0.0, metric 1, tag 0*

*\*Mar 1 00:08:29.991: RIP: ignored v2 packet from 1.1.1.1 (sourced from one of our addresses)*

*\*Mar 1 00:08:31.255: RIP: received v2 update from 1.1.1.2 on Serial2/0*

*\*Mar 1 00:08:31.255: 3.0.0.0/8 via 0.0.0.0 in 1 hops*

*\*Mar 1 00:08:38.359: RIP: sending v2 update to 224.0.0.9 via Ethernet1/0 (2.1.1.1)*

*\*Mar 1 00:08:38.359: RIP: build update entries*

*\*Mar 1 00:08:38.359: 1.0.0.0/8 via 0.0.0.0, metric 1, tag 0*

*\*Mar 1 00:08:38.359: 3.0.0.0/8 via 0.0.0.0, metric 2, tag 0*

***R2:***

*R2#debug ip rip*

*RIP protocol debugging is on*

*\*Mar 1 00:08:46.083: RIP: received v2 update from 1.1.1.1 on Serial2/0*

*\*Mar 1 00:08:46.083: 2.0.0.0/8 via 0.0.0.0 in 1 hops*

*\*Mar 1 00:08:47.579: RIP: sending v2 update to 224.0.0.9 via Ethernet1/0 (3.1.1.1)*

*\*Mar 1 00:08:47.579: RIP: build update entries*

*\*Mar 1 00:08:47.579: 1.0.0.0/8 via 0.0.0.0, metric 1, tag 0*

*\*Mar 1 00:08:47.579: 2.0.0.0/8 via 0.0.0.0, metric 2, tag 0*

*\*Mar 1 00:08:48.203: RIP: sending v2 update to 224.0.0.9 via Serial2/0 (1.1.1.2)*

*\*Mar 1 00:08:48.203: RIP: build update entries*

*\*Mar 1 00:08:48.203: 3.0.0.0/8 via 0.0.0.0, metric 1, tag 0*

*\*Mar 1 00:08:48.203: RIP: sending v2 update to 1.1.1.2 via Serial2/0 (1.1.1.2)*

*\*Mar 1 00:08:48.203: RIP: build update entries*

*\*Mar 1 00:08:48.203: 3.0.0.0/8 via 0.0.0.0, metric 1, tag 0*

*\*Mar 1 00:08:48.219: RIP: ignored v2 packet from 1.1.1.2 (sourced from one of our addresses)*

根据debug输出信息，可以分析RIP协议的工作过程：

1. 发送更新：RIP路由器会定期（默认为30秒）向所有邻居发送其路由表中的所有路由信息。R1和R2都在发送v2更新到224.0.0.9。
2. 构建更新条目：在发送更新之前，RIP路由器会构建更新条目。R1和R2都在构建更新条目，包括2.0.0.0/8、1.0.0.0/8和3.0.0.0/8等网络的路由信息。
3. 接收更新：RIP路由器会接收来自邻居的更新，并根据这些更新刷新自己的路由表。例如，R1从1.1.1.2接收到了3.0.0.0/8网络的更新，而R2从1.1.1.1接收到了2.0.0.0/8网络的更新。
4. 忽略自己的更新：RIP路由器会忽略源自自己地址的更新。例如，R1忽略了从1.1.1.1接收到的更新，而R2忽略了从1.1.1.2接收到的更新。

**步骤3：**

在R1上将与R2相连的S2/0接口配置为禁止水平分割，然后观察RIP协议交互过程的变化情况

进入R1的s2/0接口的配置模式，禁止水平分割（默认是打开水平分割的）

|  |
| --- |
| R1 (config-if) # no ip split-horizon |

这时RIP协议的交互信息会发生变化，分析变化的原因

***R1:***

*R1(config-if)#no ip split-horizon*

*\*Mar 1 00:10:43.675: RIP: sending v2 update to 224.0.0.9 via Serial2/0 (1.1.1.1)*

*\*Mar 1 00:10:43.675: RIP: build update entries*

*\*Mar 1 00:10:43.675: 1.0.0.0/8 via 0.0.0.0, metric 1, tag 0*

*\*Mar 1 00:10:43.675: 1.1.1.2/32 via 0.0.0.0, metric 1, tag 0*

*\*Mar 1 00:10:43.675: 2.0.0.0/8 via 0.0.0.0, metric 1, tag 0*

*\*Mar 1 00:10:43.675: 3.0.0.0/8 via 0.0.0.0, metric 2, tag 0*

*\*Mar 1 00:10:43.675: RIP: sending v2 update to 1.1.1.1 via Serial2/0 (1.1.1.1)*

*\*Mar 1 00:10:43.675: RIP: build update entries*

*\*Mar 1 00:10:43.675: 1.0.0.0/8 via 0.0.0.0, metric 1, tag 0*

*\*Mar 1 00:10:43.675: 1.1.1.2/32 via 0.0.0.0, metric 1, tag 0*

*\*Mar 1 00:10:43.675: 2.0.0.0/8 via 0.0.0.0, metric 1, tag 0*

*\*Mar 1 00:10:43.675: 3.0.0.0/8 via 0.0.0.0, metric 2, tag 0*

*\*Mar 1 00:10:43.687: RIP: ignored v2 packet from 1.1.1.1 (sourced from one of our addresses)*

*\*Mar 1 00:10:45.071: RIP: received v2 update from 1.1.1.2 on Serial2/0*

*\*Mar 1 00:10:45.071: 3.0.0.0/8 via 0.0.0.0 in 1 hops*

在RIP协议中，水平分割是一种防止路由环路的机制。当水平分割启用时，路由器不会在接收到路由更新的接口上发送这些更新。这可以防止已经传播过的路由信息再次被传播回来，从而防止路由环路的产生。

禁止水平分隔后，R1的s2/0端口在收到R2的s2/0端口发来的数据包时，会将这些数据包发送给R2的s2/0端口，产生了循环。当在R1的S2/0接口上禁用水平分割时，R1现在会在所有接口上发送所有路由更新，包括它从R2接收到的更新。能看到R1现在在向224.0.0.9和1.1.1.1发送包含1.1.1.2/32网络的更新。

**步骤4：**

在R1中观察RIP协议动态学习到的路由表项。

在特权模式下输入：

R1 # show ip route

会显示如下信息：

*1.0.0.0/8 is variably subnetted, 2 subnets, 2 masks*

*C 1.0.0.0/8 is directly connected, Serial2/0*

*C 1.1.1.2/32 is directly connected, Serial2/0*

*C 2.0.0.0/8 is directly connected, Ethernet1/0*

*R 3.0.0.0/8 [120/1] via 1.1.1.2, 00:00:09, Serial2/0*

路由表显示了R1路由器通过RIP协议学习到的路由信息：

* 1.0.0.0/8和1.1.1.2/32网络是直接连接到Serial2/0接口的。"C"表示这些路由是直接连接的（Connected）。
* 2.0.0.0/8网络是直接连接到Ethernet1/0接口的。
* 3.0.0.0/8网络是通过1.1.1.2（即R2路由器）学习到的。"R"表示这个路由是通过RIP协议学习到的（RIP）。[120/1]表示这个路由的度量值是1，120是RIP协议的管理距离。

这些信息显示了R1路由器如何将数据包路由到不同的网络。

**问题和分析**

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？

在RIP协议中，"via 0.0.0.0"通常表示路由更新信息的来源。例如，当一个路由器接收到一个RIP更新包时，"via 0.0.0.0"表示这个更新包是直接从源路由器接收到的，而不是通过其他路由器转发的。

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？

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

**OSPF配置**

先去除RIP

* Conf
* No router rip
* R1
* Conf

Router ospf 10

Network 1.0.0.0 0.0.0.255 area 0

Network 2.0.0.0 0.0.0.255 area 0

Interface s1/1

Ip ospf hello-interval 5

Ip ospf dead-interval 20

* R2
* Conf

Router ospf 20

Network 1.0.0.0 0.0.0.255 area 0

Network 3.0.0.0 0.0.0.255 area 0

Interface s1/0

Ip ospf hello-interval 5

Ip ospf dead-interval 20

* Debug ip ospf events
* Debug ip ospf flood
* Sh ip ospf neighbor

**OSPF协议配置和分析**

**实验目的**

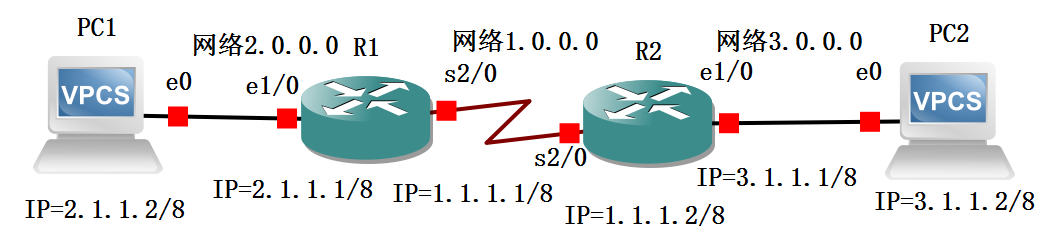
掌握路由器上OSPF协议的配置方法，能够在模拟环境中进行路由器上OSPF协议的配置，并能够通过debug信息分析OSPF协议的工作过程

**实验环境**

采用Dynamips仿真环境，使用GNS3进行图形拓扑设计

**实验拓扑**

采用两台路由器R1和R2将两台主机主机PC1、PC2连接在一起，两台路由器之间采用串口进行连接，两台主机和路由器之间都采用以太网进行连接



**配置过程**

首先PC1、PC2、R1和R2的配置与静态路由实验的配置完全一样，唯一不同的是在R1和R2上将静态路由删除，配置上用OSPF协议来自动计算路由

**步骤1：**

在R1的配置模式下配置OSPF协议

进入OSPF配置模式：

|  |
| --- |
| R1(config) # Router ospf 10 |

配置与R1直连的网络，并指明网络所属区域

|  |
| --- |
| R1 (config-ospf) # network 1.0.0.0 255.0.0.0 area 0 R1 (config-ospf) # network 2.0.0.0 255.0.0.0 area 0 |

进入s2/0接口配置模式：

|  |
| --- |
| R1 (config) # interface s2/0 |

配置从接口s2/0发送hello包的时间间隔，以及认为通过接口s2/0相连的邻居已经不存在的时间间隔

|  |
| --- |
| R1 (config-if) # ip ospf hello-interval 5 R1 (config-if) # ip ospf dead-interval 20 |

在R2上也进行类似配置。

**步骤2：**

在路由器的特权模式下打开调试信息：

|  |
| --- |
| R1 # debug ip ospf events |

可以观察到OSPF协议交互的所有事件信息，通过这些信息来分析OSPF协议的工作过程。

*R1#debug ip ospf events*

*OSPF events debugging is on*

*\*Mar 1 00:08:51.663: OSPF: Rcv hello from 1.1.1.2 area 0 from Serial2/0 1.1.1.2*

*\*Mar 1 00:08:51.663: OSPF: End of hello processing*

*\*Mar 1 00:08:52.123: OSPF: Send hello to 224.0.0.5 area 0 on Serial2/0 from 1.1.1.1*

*\*Mar 1 00:08:55.795: OSPF: Send hello to 224.0.0.5 area 0 on Ethernet1/0 from 2.1.1.1*

*\*Mar 1 00:08:56.655: OSPF: Rcv hello from 1.1.1.2 area 0 from Serial2/0 1.1.1.2*

*\*Mar 1 00:08:56.655: OSPF: End of hello processing*

根据debug输出信息，可以分析OSPF协议的工作过程：

1. 发送Hello报文：OSPF路由器会定期（通常为10秒）向所有邻居发送Hello报文。例如，R1在Serial2/0接口上向224.0.0.5发送了Hello报文。
2. 接收Hello报文：OSPF路由器会接收来自邻居的Hello报文，并进行处理。例如，R1从1.1.1.2接收到了Hello报文。
3. 处理Hello报文：在接收到Hello报文后，OSPF路由器会进行一些处理，如更新邻居信息、检查网络连接状态等。

**问题和分析**

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?

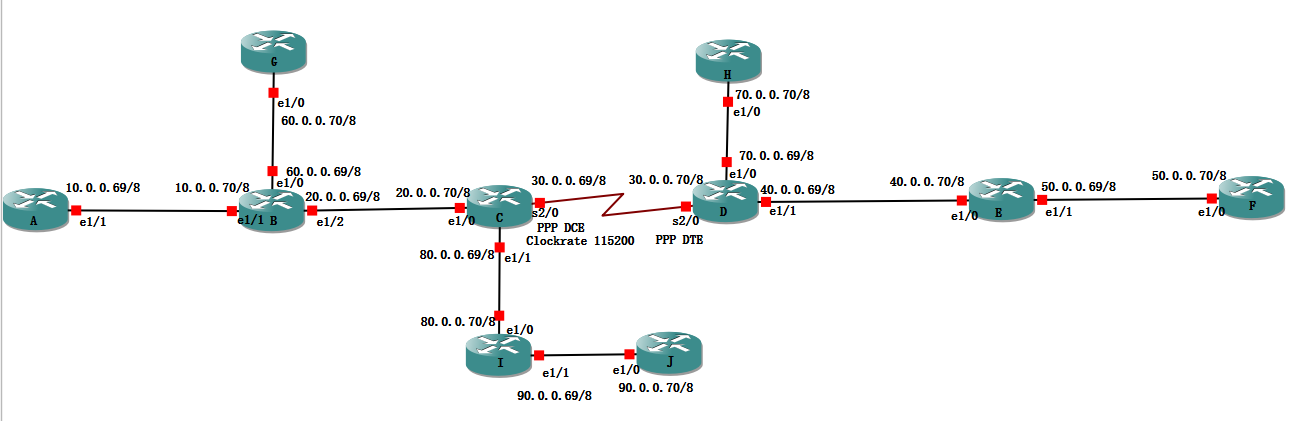
因为OSPF的主要协议交互过程是在刚配置完OSPF协议时就进行了，而在网络运行过程中，只要没有链路状态的变化就不再交互链路状态信息了。所以要观察OSPF协议的工作过程就要先打开debug信息，然后再配置OSPF协议。由于打开了debug信息，配置时就会弹出各种显示信息，影响配置的过程。

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？

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。

**动态路由——复杂拓扑**

**实验拓扑**



* 路由器地址的最后一个8位组与自己学号的后两位关联
* 我的学号是2021211269，那么路由器E的e1/1地址应该是50.0.0.69 路由器F的e1/0地址为50.0.0.70，其余路由器类似。

**实验要求**

* 按照上页复杂拓扑图搭建测试网络
* 该网络由10台C3640路由器组成
* 路由器C和D之间采用串口连接
* 通过配置静态路由，使得网络能够连通
* 路由器的地址使用自己的学号最后两位和学号+1，例子见图中的红字说明。

**实验内容**

* 在实验二的基础上实现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（需要修改部分链路，观察工作过程）
* 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（需要修改部分链路，观察区别，默认有水平分割）
* 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（需要修改部分链路，观察消息传递过程）

**实验步骤**

**配置路由器C和D之间的串口连接**

进入C的s2/0接口配置界面，配置时钟为115200

|  |
| --- |
| C(config-if)# clockrate 115200 |

将两端的串口的数据链路层都配置成PPP协议，如不配置，默认HDLC协议。进入C的s2/0接口配置界面和D的s2/0接口的配置界面，配置PPP协议。

|  |
| --- |
| C (config-if)# encapsulation PPP D (config-if)# encapsulation PPP |

两端串口各配置一个IP地址和子网掩码，保证这两个IP地址在同一个网段。

|  |
| --- |
| C (config-if)# ip address 30.0.0.69 255.0.0.0 D (config-if)# ip address 30.0.0.70 255.0.0.0 |

启动这两个串口，这样这两台路由器可以通过串口互相通信了。

**配置路由器A-J的端口IP地址**

例如路由器A：

|  |
| --- |
| config t interface e1/0 ip address 10.0.0.69 255.0.0.0 no shutdown |

**实现RIP路由协议**

配置各个路由器协议及相邻子网

例如路由器A：

|  |
| --- |
| A(config)#router rip A(config-router)#version 2 A(config-router)#network 10.0.0.0 |

C和D要配置各自的邻居路由器：

|  |
| --- |
| C(config-rip) # neighbor 30.0.0.69 |

**通过debug信息详细描述RIP工作过程**

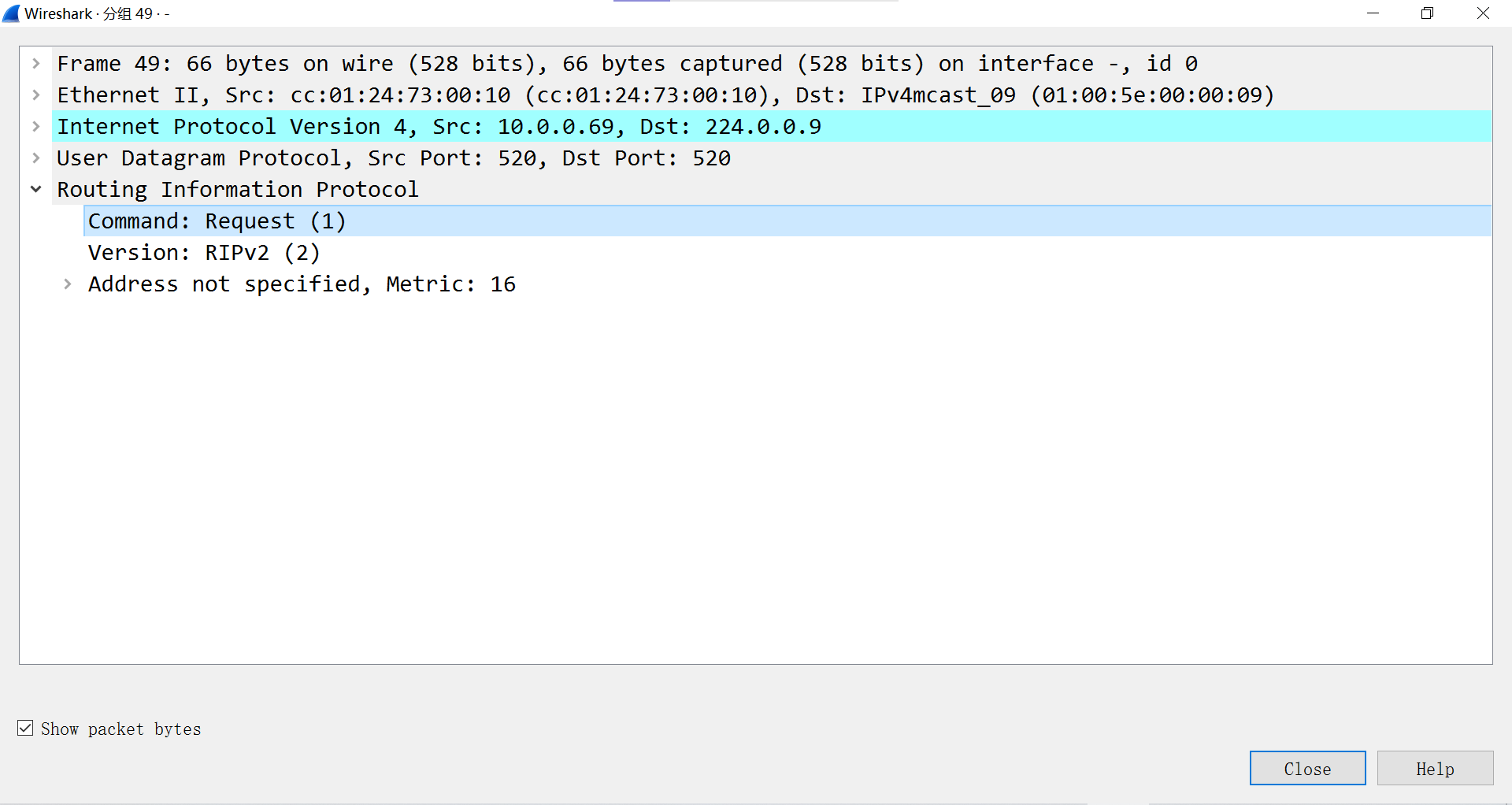
在各个路由器上打开调试信息：

|  |
| --- |
| A # Debug ip rip |

包括初始信息交互、路由计算、链路故障处理等部分（需要修改部分链路，观察工作过程）

* 初始信息交互：

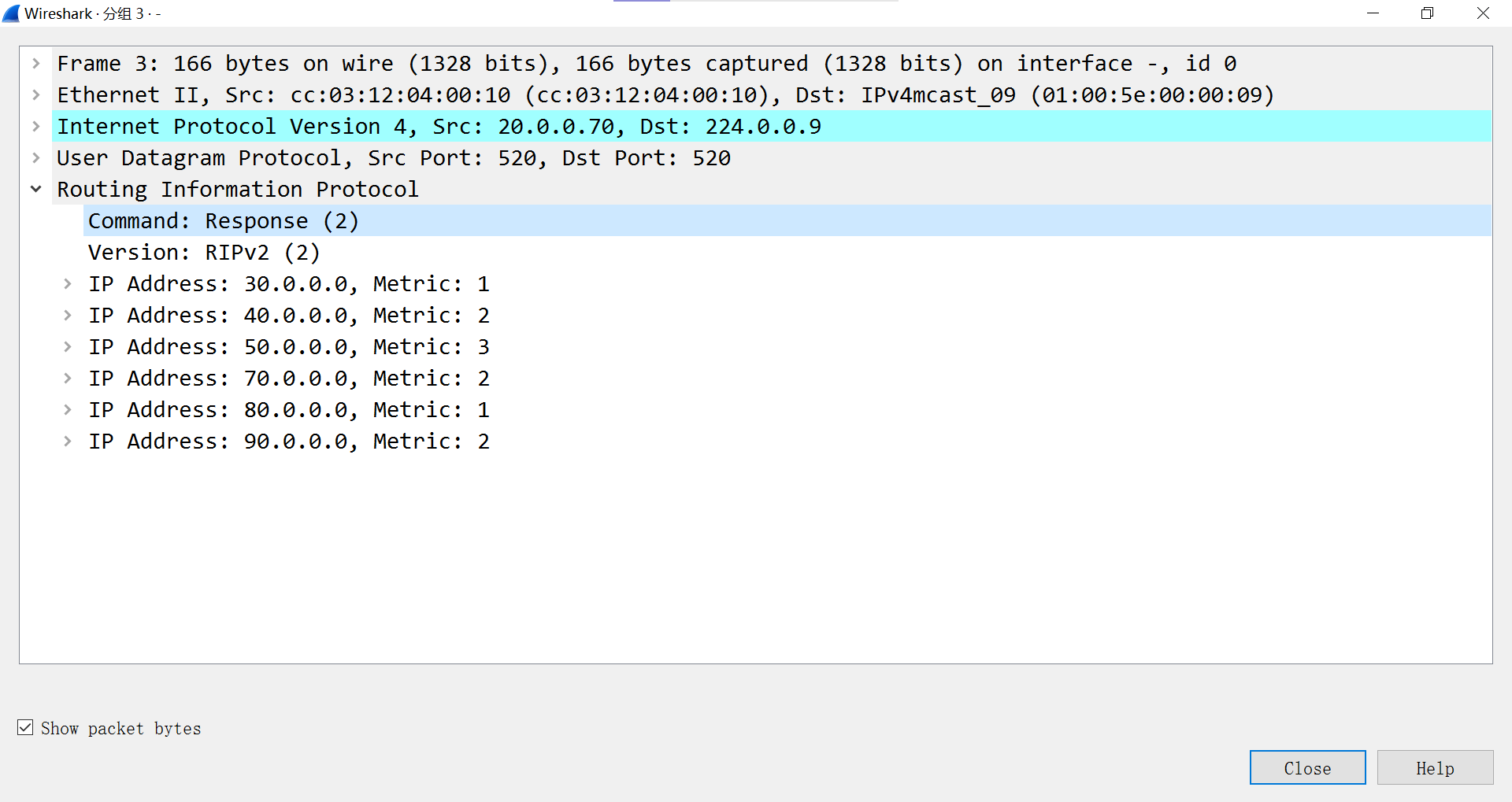
当路由器启动并运行RIP协议时，路由表中只会包含直连路由。路由器会发送Request报文，请求邻居路由器的RIP路由。运行RIP的邻居路由器收到该Request报文后，会根据自己的路由表，生成Response报文进行回复。路由器在收到Response报文后，会将相应的路由添加到自己的路由表中。在运行RIP协议时，路由器先发送Request报文，下图是路由器A的Request报文：



* 路由计算：

RIP网络稳定后，每个路由器会周期性地向邻居路由器通告自己的整张路由表中的路由信息，默认周期为30秒。邻居路由器根据收到的路由信息刷新自己的路由表。RIP使用跳数作为度量值来衡量到达目的网络的距离。在RIP中，路由器到与它直接相连网络的跳数为0，每经过一个路由器后跳数加1。该路由信息由Response报文携带。

下图是一个RIPv2的Response报文：



该报文由路由器C发出，包括命令类型（响应），版本号（RIPv2），以及多个路由条目。每个路由条目都包括一个IP地址和一个度量值（Metric，表示到达该网络的跳数）。例如，30.0.0.0网络的度量值为1，40.0.0.0网络的度量值为2，等等。

* 链路故障处理：

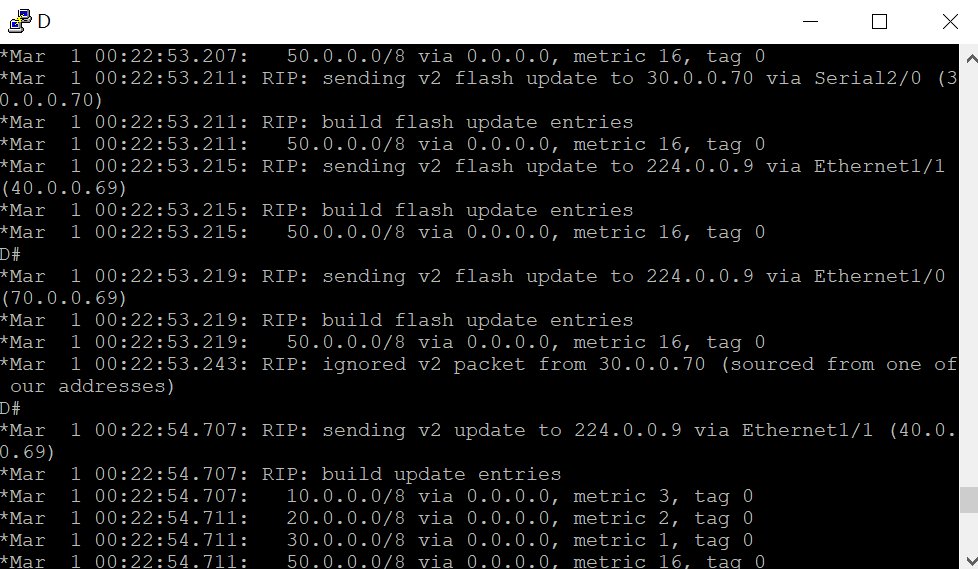
当网络发生故障时，RIP网络有可能产生路由环路。如果路由器E的直连网络50.0.0.0/8产生故障（关闭路由器E）：

路由器D会立即检测到该故障，并认为该路由不可达。

此时，路由器H还没有收到该路由不可达的信息，于是会继续向路由器D发送度量值为2的通往50.0.0.0/8的路由信息。

路由器D会学习此路由信息，认为可以通过路由器H到达50.0.0.0/8网络。

此后，路由器D发送的更新路由表，又会导致路由器H路由表的更新，路由器H会新增一条度量值为3的50.0.0.0/8网络路由表项，从而形成路由环路。这个过程会持续下去，直到度量值为16。



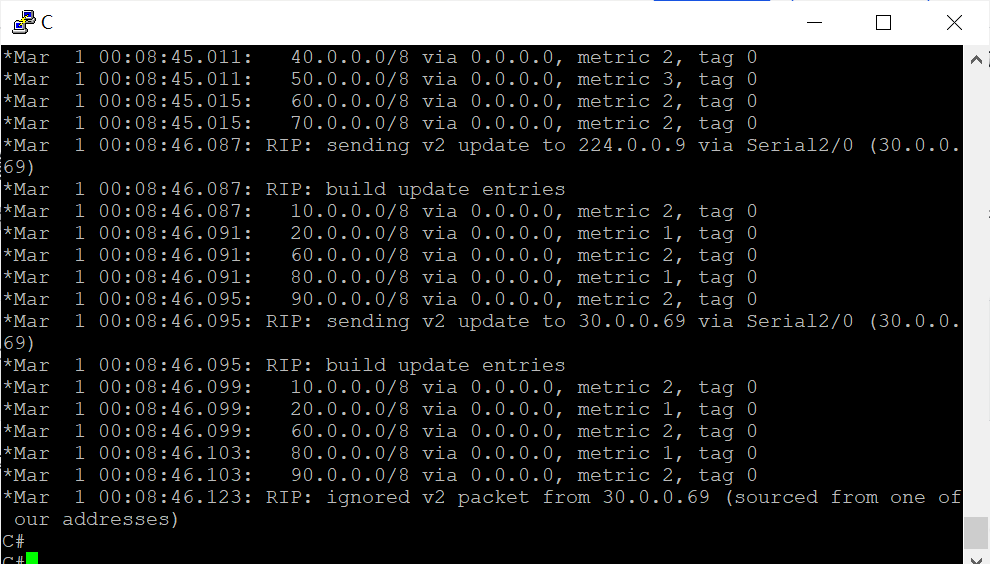
可以看到，这时候度量值为16，此后50.0.0.0/8子网表项被路由器删除。

**水平分割**

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（需要修改部分链路，观察区别，默认有水平分割）

* 有水平分割时：

C路由器调试输出：



观察到，C从s2/0收到的数据包不会再次发送到D，而会把其他端口收到的数据包发送给D，这避免了环路的产生。

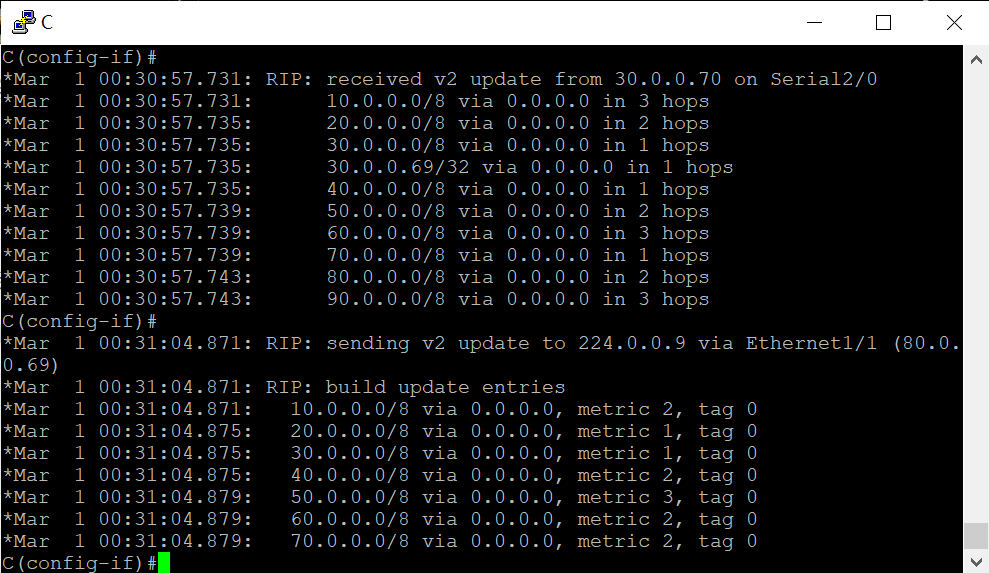
将路由器E关闭来模拟50.0.0.0/8子网故障，在子网故障后，观察路由器路由输出，D首先检测到路由器故障，发现故障处理过程中并没有出现环路，子网50.0.0.0/8在无效计时器超时后被各个路由器的路由表删除。

* 无水平分割时：

重新启动路由器E和F并打开RIP调试，各个路由器又增加了50.0.0.0/8子网表项。

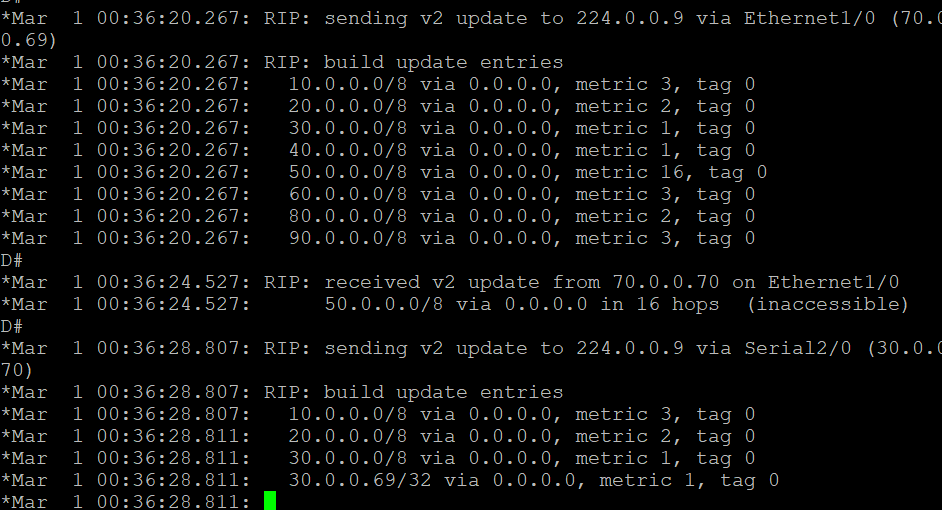
关闭C路由器s2/0和D路由器s2/0的水平分割：

|  |
| --- |
| C (config-if) # no ip split-horizon |



观察发现，C路由器将其余端口收到的数据包和从s2/0收到的数据包都发送给了端口s2/0，这样就产生了环路。

将E路由器和F路由器关闭来模拟50.0.0.0/8子网故障，在子网故障后，观察路由器路由输出：



发现路由器RIP调试出现环路，且在跳数递增到16时路由器才意识到不可达。因此，水平分割是必要的，这样会避免环路。

**实现OSPF路由协议**

进入OSPF配置模式：

|  |
| --- |
| A(config) # Router ospf 10 |

配置与A直连的网络，并指明网络所属区域，其余路由器类似配置。

|  |
| --- |
| A (config-ospf) # network 10.0.0.0 255.0.0.0 area 0 |

进入C路由器的s2/0接口配置模式：

|  |
| --- |
| C (config) # interface s2/0 |

配置从接口s2/0发送hello包的时间间隔，以及认为通过接口s2/0相连的邻居已经不存在的时间间隔

|  |
| --- |
| C (config-if) # ip ospf hello-interval 5 C (config-if) # ip ospf dead-interval 20 |

在D上也进行类似配置。

**通过debug信息详细描述OSPF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（需要修改部分链路，观察工作过程）

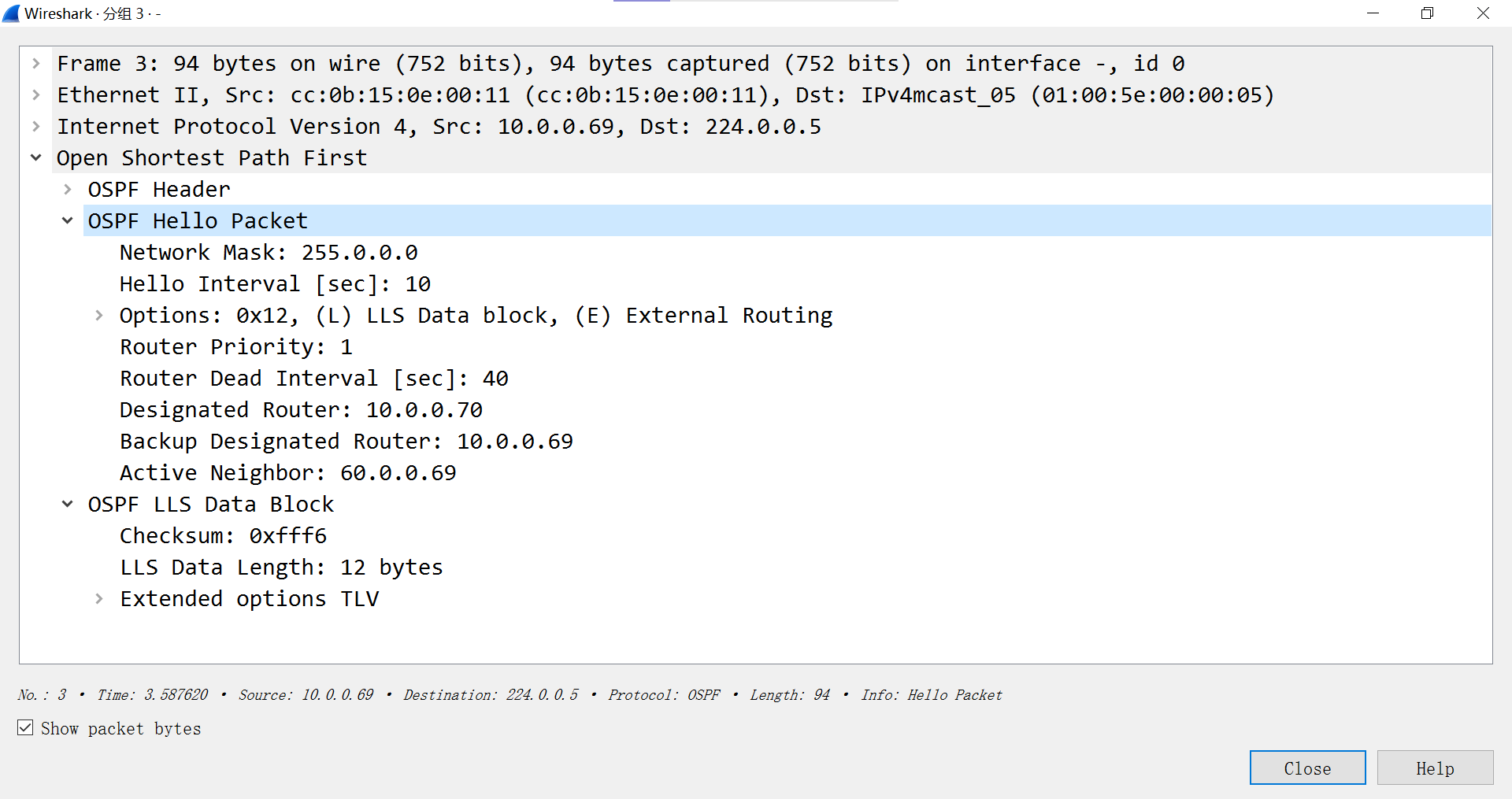
启动各个路由器的OSPF调试：

|  |
| --- |
| A # debug ip ospf events |

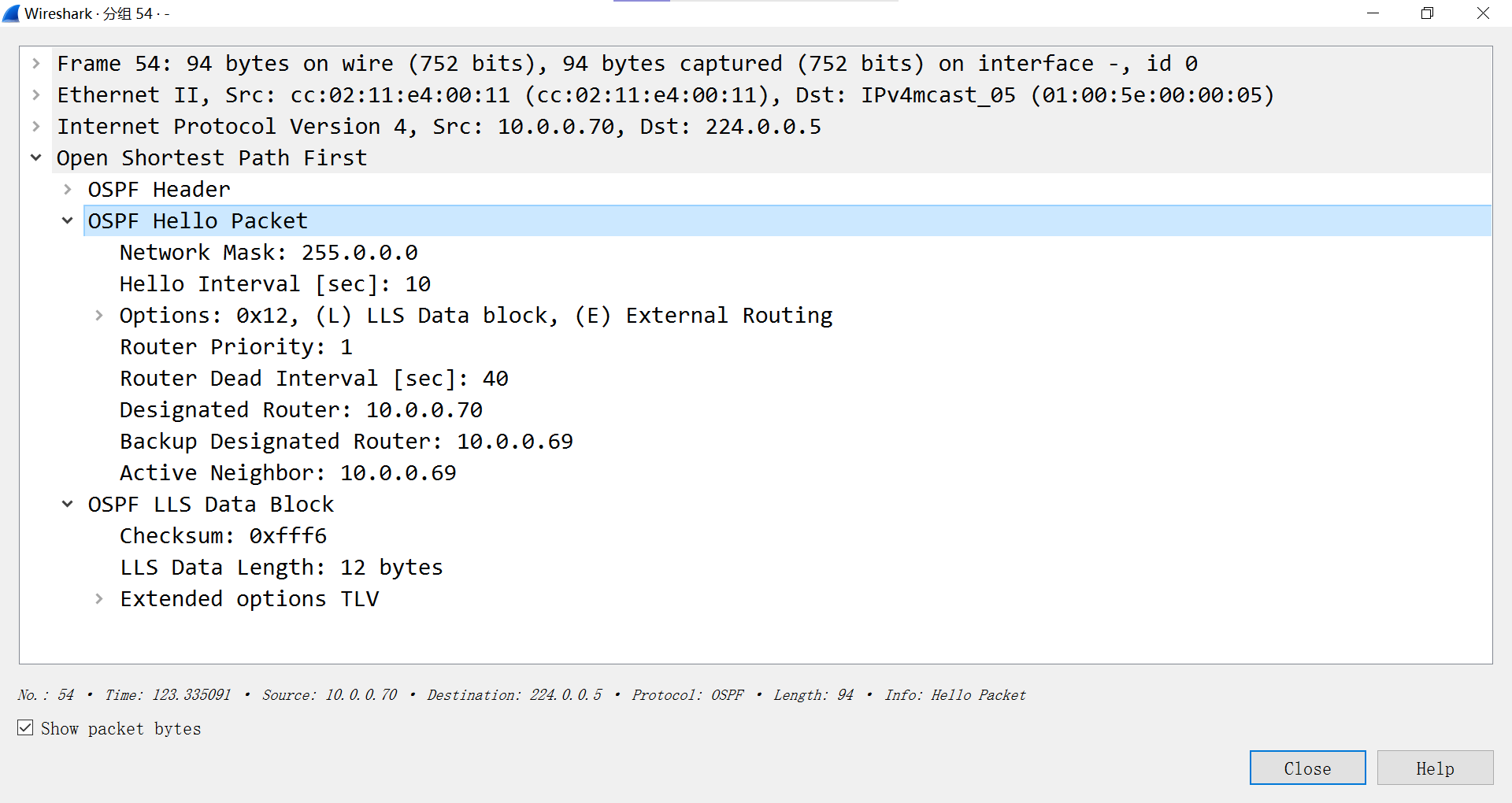
* 初始信息交互：

当路由器启动并运行OSPF协议时，路由器会向本地所有启动了OSPF协议的直连接口组播224.0.0.5发送hello包。本地hello包中携带本地的全网唯一的router-id。之后对端B运行OSPF协议的设备将回复hello包，该hello包中若携带了A的routerid，那么A/B建立为邻居关系，生成邻居表。

如图是A向组播发送的hello包：



对端B回复的hello包：



* 路由交换和计算：

OSPF路由器之间交换链路状态公告 (LSA)信息。OSPF的LSA中包含连接的接口、使用的Metric及其他变量信息。OSPF路由器收集链接状态信息并使用SPF算法来计算到各节点的最短路径。形成链路状态数据库表，表中记录了网络拓扑中链路状态的通告。

在获得完整链路状态数据库表后，进行SPF算法，形成最优路由加入路由表。SPF算法是以自身为根节点计算出一棵最短路径树，在这棵树上由根到各节点的累计开销最小，即由根到各节点的路径在整个网络中都是最优的，这样也就获得了由根去往各个节点的路由。

* 链路故障处理：

当网络发生故障时，OSPF网络有可能产生路由环路。例如，如果路由器B的直连网络10.0.0.0/8产生故障，路由器B会立即检测到该故障，并认为该路由不可达。此时，路由器C还没有收到该路由不可达的信息，于是会继续向路由器B发送度量值为2的通往10.0.0.0/8的路由信息。路由器B会学习此路由信息，认为可以通过路由器C到达10.0.0.0/8网络。此后，路由器B发送的更新路由表，又会导致路由器C路由表的更新，路由器C会新增一条度量值为3的10.0.0.0/8网络路由表项，从而形成路由环路。这个过程会持续下去，直到度量值为16。

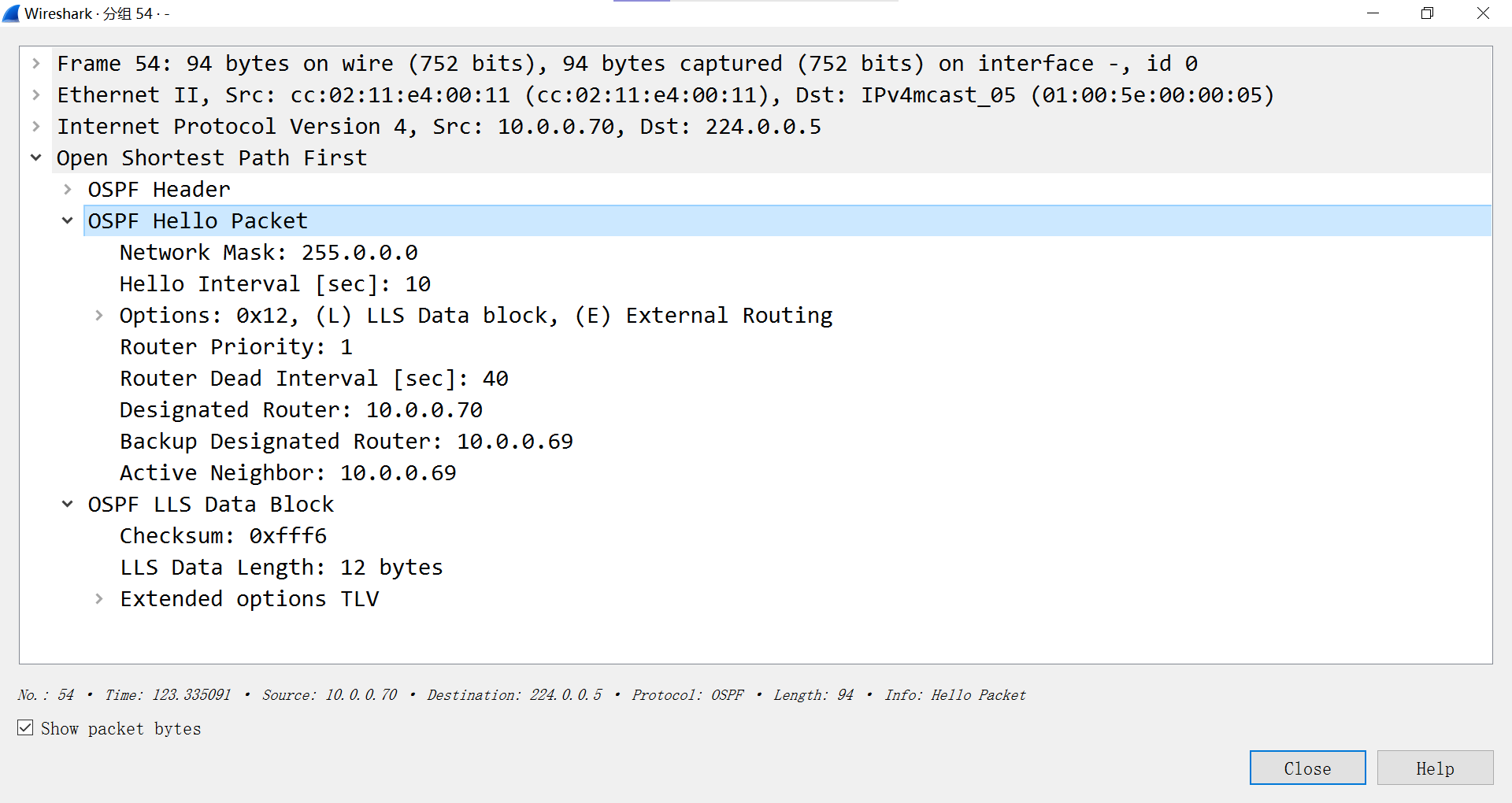
**数据库同步信息的格式和同步对象**

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（需要修改部分链路，观察消息传递过程）

在OSPF协议中，数据库同步信息和链路改变信息主要通过以下几种报文进行交互：

1. Hello报文：用于发现和维护邻居关系。在广播和NBMA（None-Broadcast Multi-Access）类型的网络中，还用于选举指定路由器（DR）和备份指定路由器（BDR）。

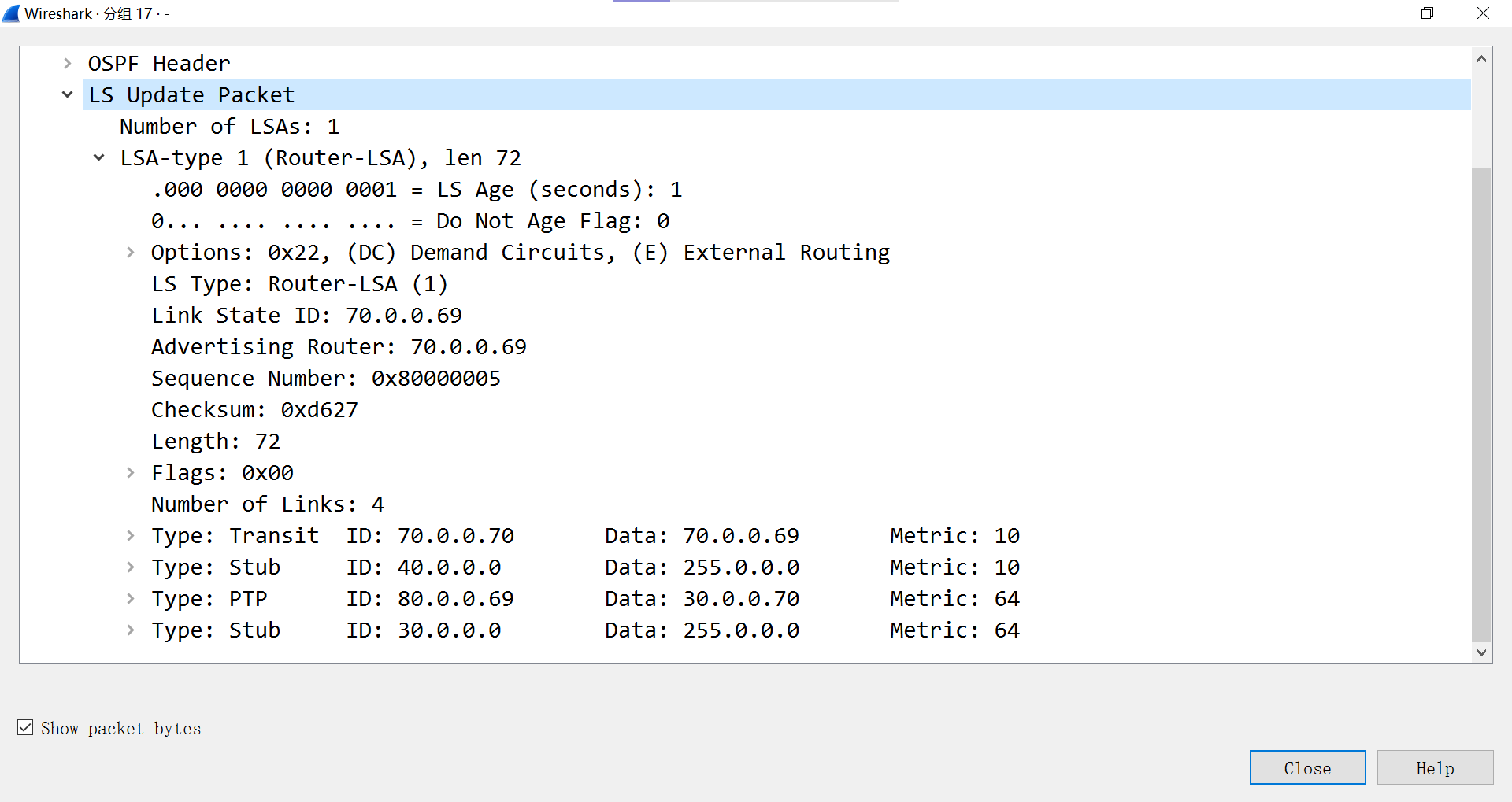
如图，在B的广播hello报文中，选举指定路由器是B，备份指定路由器是A：



1. Database Description (DD)报文：在两台路由器进行链路状态数据库（LSDB）同步时，用DD报文来描述自己的LSDB。DD报文的内容包括LSDB中每一条链路状态公告（LSA）的头部。LSA的头部可以唯一标识一条LSA，因此，这样可以减少路由器之间的协议报文流量。
2. Link State Request (LSR)报文：两台路由器互相交换过DD报文之后，知道对端的路由器有哪些LSA是本地LSDB所缺少的，这时需要发送LSR报文向对方请求缺少的LSA。LSR只包含了所需要的LSA的摘要信息。
3. Link State Update (LSU)报文：用来向对端路由器发送所需要的LSA。
4. Link State Acknowledgment (LSAck)报文：用来对接收到的LSU报文进行确认。

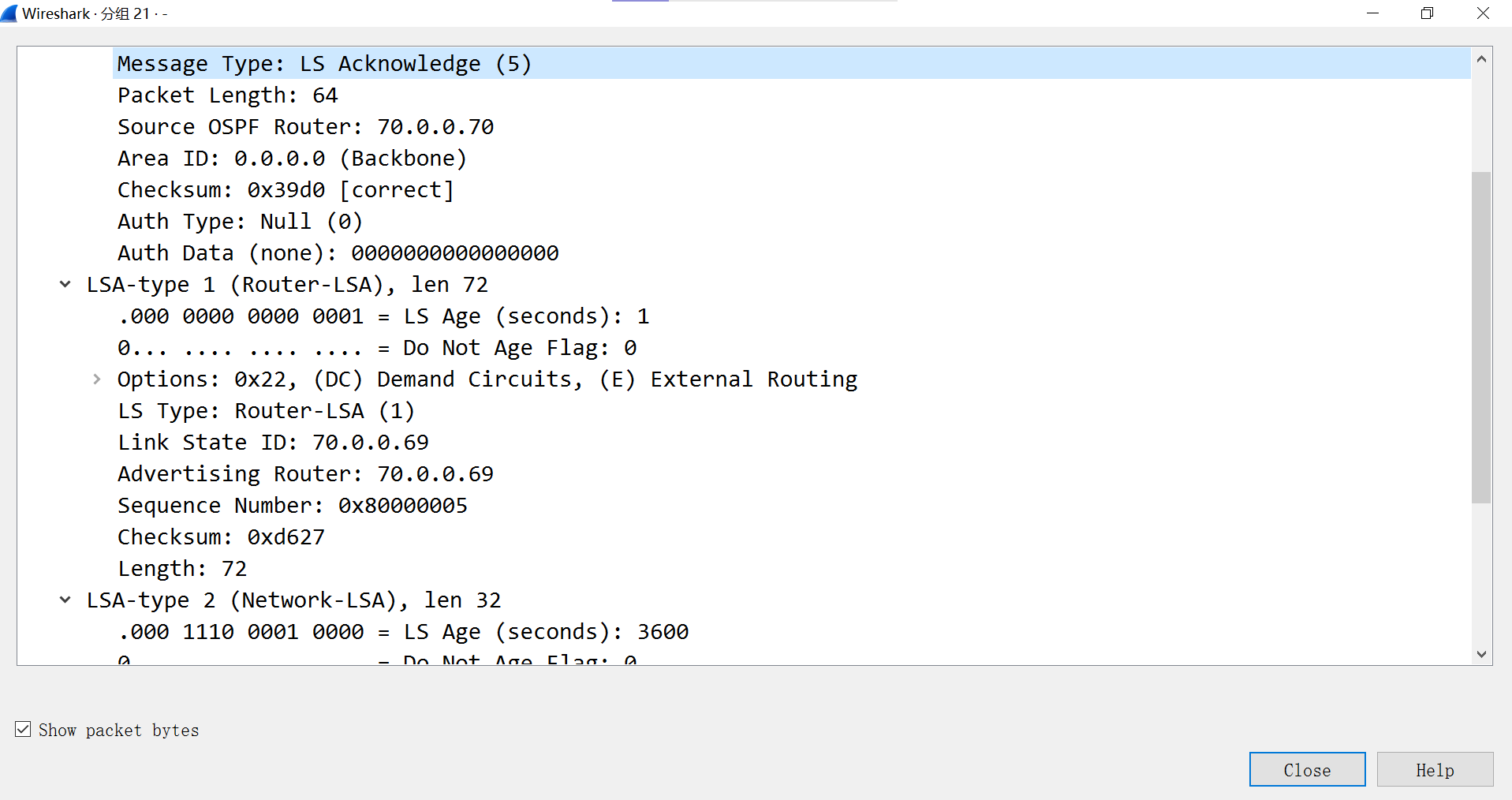
当链路发生改变时，路由器会生成一条新的LSA，并通过LSU报文发送给所有邻居。邻居路由器在收到LSU报文后，会更新自己的LSDB，并向其他邻居转发该LSA。这个过程被称为LSA的泛洪（flooding）。

将路由器E关闭来模拟50.0.0.0/8子网故障，在子网故障后，抓取路由器D发出的LSU报文：

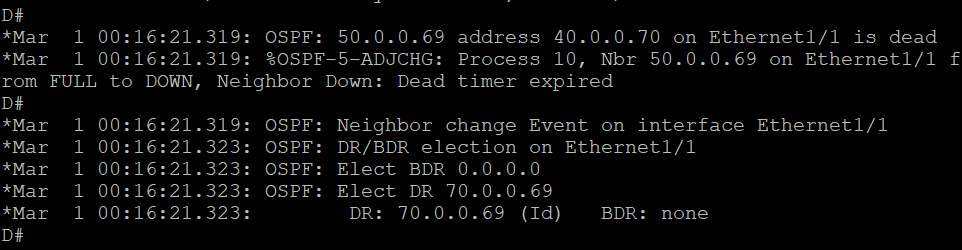


表示路由器D发出更新链路状态，且50.0.0.0子网被删除。

与D相邻的路由器H收到LSU后，更新自己的LSDB，并向其他邻居转发该LSA：



观察路由器D路由输出：



发现该协议可以检测到路由器关闭，定时器超时，50.0.0.0/8子网表项被路由器删除。

**思考题**

（1）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）

静态路由配置中的"下一跳"和"转发接口"这两种方式有以下区别：

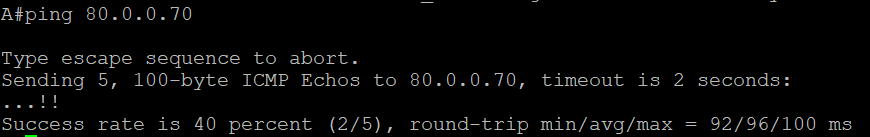
下一跳（Next Hop）：在配置静态路由时，指定下一跳IP地址，即数据包应该发送到哪个路由器的IP地址。这种方式更加灵活，因为它允许路由器有多个路径到达同一个目的地。如果一个路径不可用，路由器可以选择另一个路径。

转发接口：在配置静态路由时，指定转发接口，即数据包应该从路由器的哪个接口发送出去。这种方式在某些情况下可能更简单，例如在点对点连接中，因为这种情况下只有一个可能的下一跳。

在使用下一跳IP地址配置时，第一次ping可能会丢包，路由器需要进行ARP解析来获取下一跳IP地址对应的MAC地址。在ARP解析完成之前，第一次ping的请求可能会超时导致丢包。ARP解析完成后，ping的请求不会丢包。

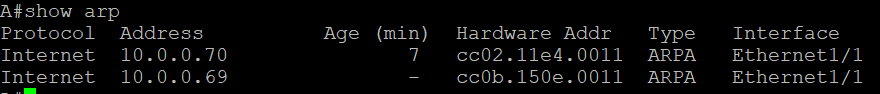
ARP表是路由器用来记录IP地址和MAC地址对应关系的表项。当我们需要转发数据的时候，除了需要对方的目的IP地址，还需要知道对方的MAC地址，那么正常情况下我们优先会在本地ARP表中查看是否有目的IP对应的MAC地址。如果ARP表中的条目不正确或者丢失，那么可能会导致数据包无法正确地发送到目的地，从而导致ping时的丢包。

在实验二中，使用下一跳配置，从路由器A第一次ping路由器I的e1/0端口：



路由器A和路由器I之间隔了两个路由器B和C，前三个包都丢了，这是因为第一个包在路由器A上需要解析ARP，第二个包需要在B上解析ARP，第三个包需要在C上解析ARP，从第四个包开始才能不丢失。

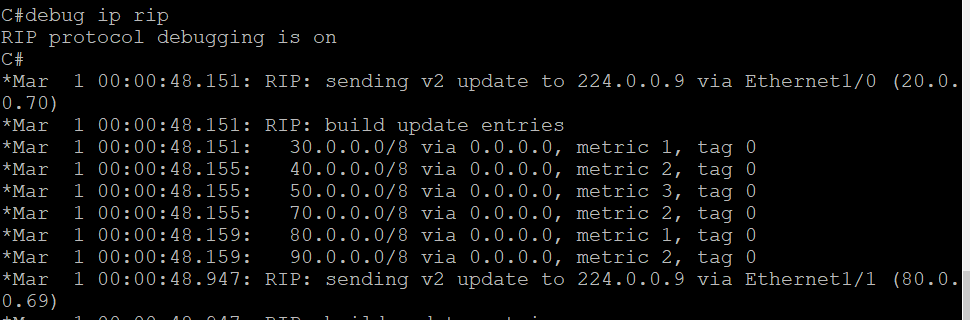
解析之后的ARP表项不为空，下次ping时能直接查该表来将对应的包直接送往对应端口，因此之后ping时不会丢包。



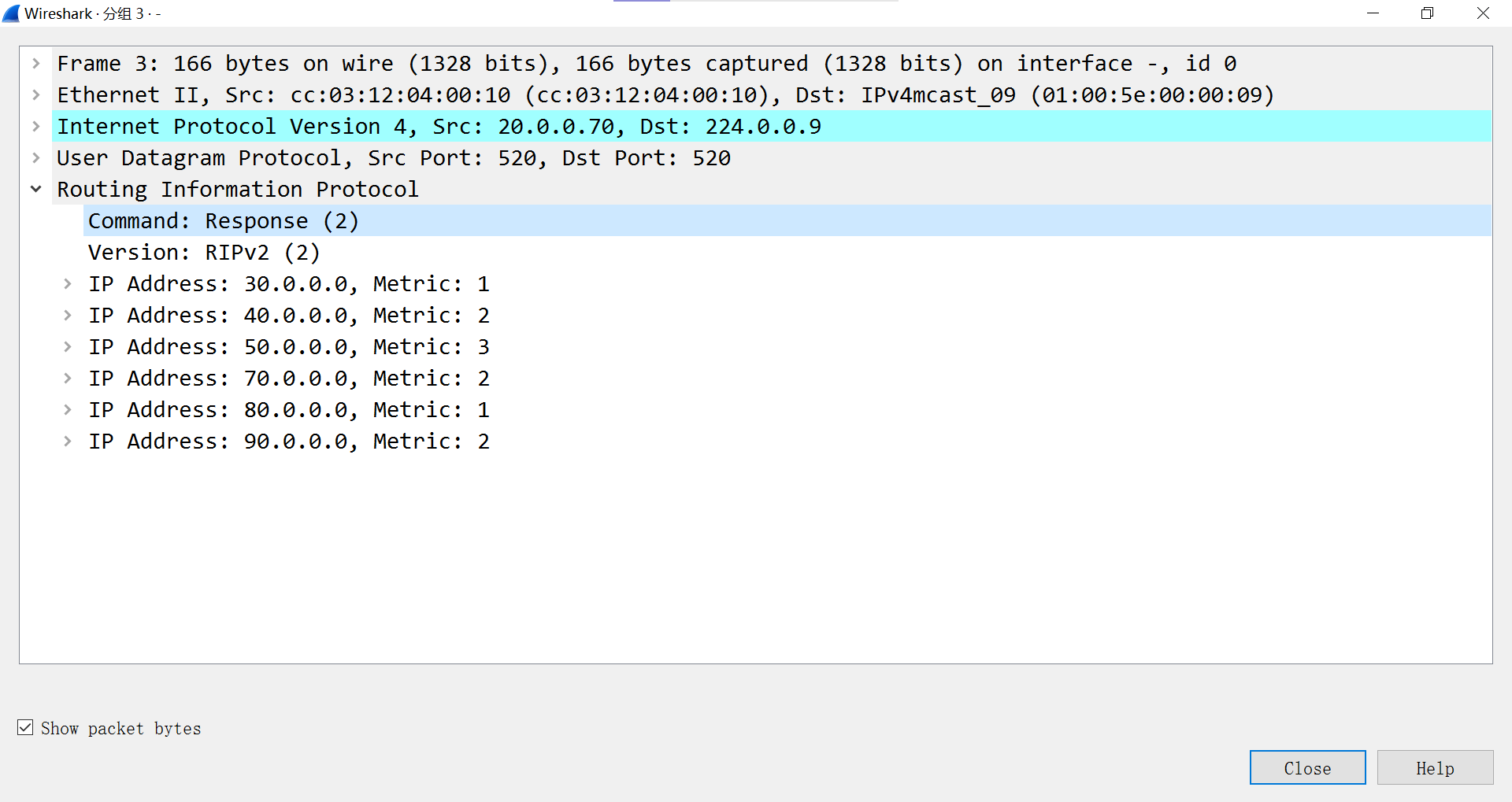
而如果使用转发接口来配置静态路由，则不用查询ARP表，因此不会出现上述丢包问题。

（2）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。

* 邻居发现：
* RIP协议：

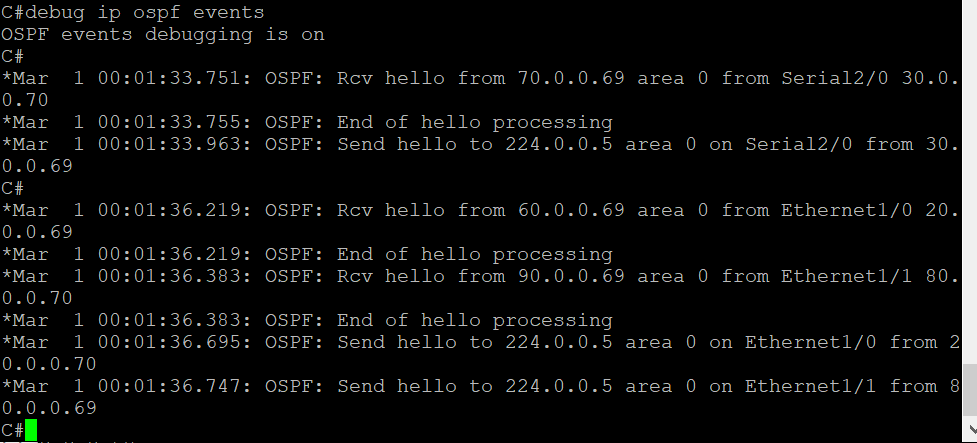


RIP协议通过定期广播或多播更新消息来发现邻居。每个RIP路由器会定期（默认为30秒）将其整个路由表发送给所有邻居。如图，路由器C定期将路由表发送给所有邻居。

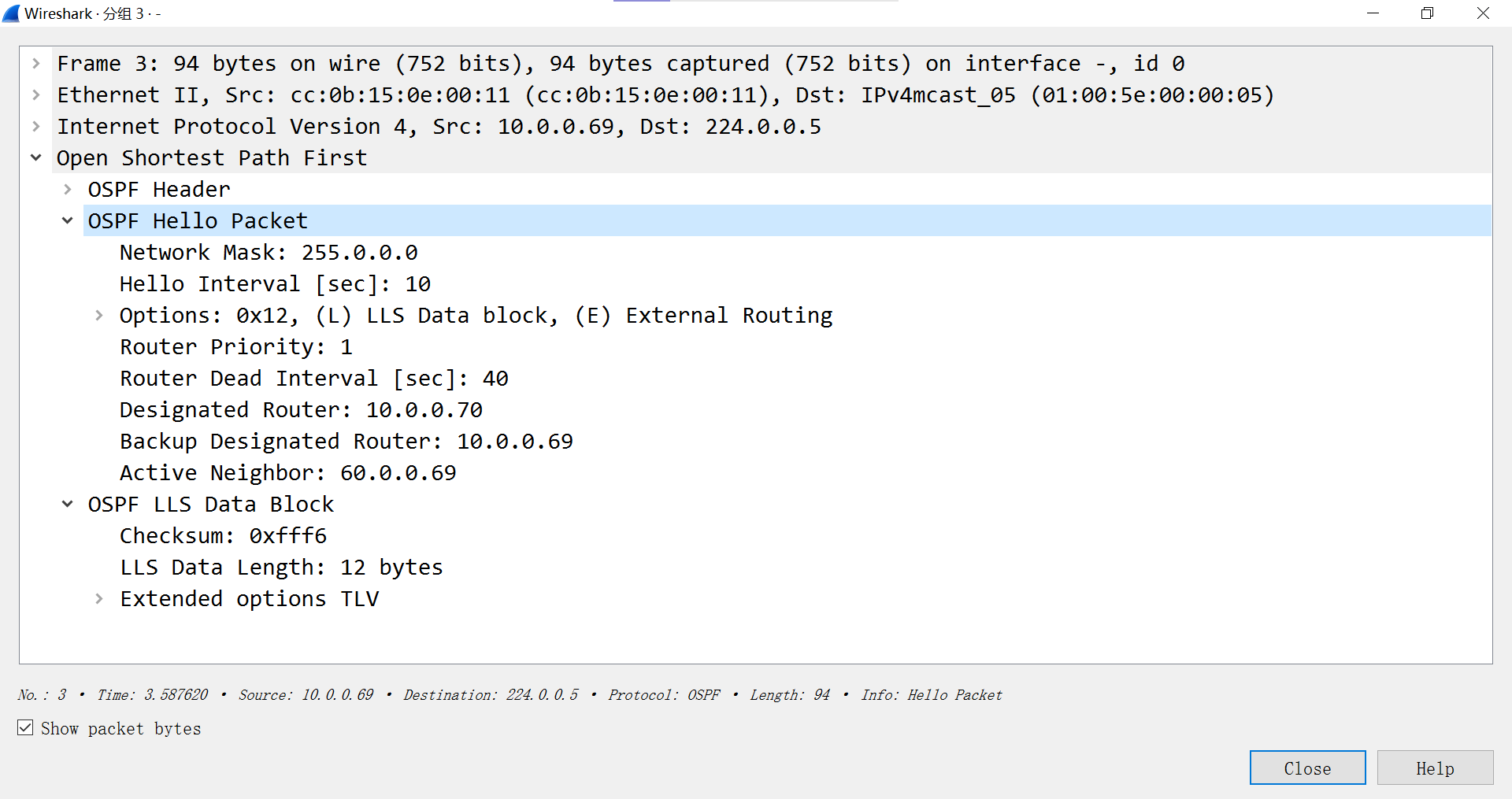


可以看到，包中包含路由的子网及跳数，这样邻居路由器就可以根据信息更新路由表项。

* OSPF协议：

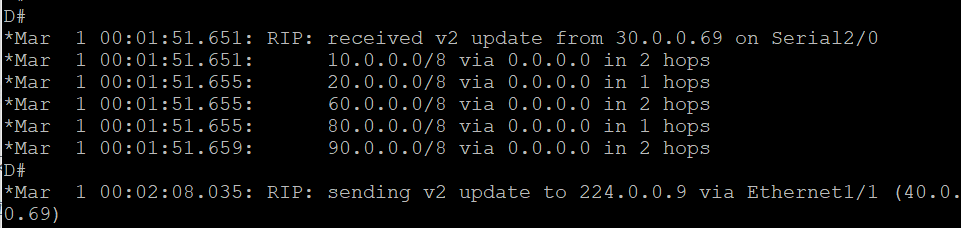


OSPF协议使用Hello报文来发现邻居。每个OSPF路由器会定期（通常为10秒）发送Hello报文，这些报文包含路由器的状态信息，如路由器的ID、当前的DR（指定路由器）和BDR（备份指定路由器）等。如上图，路由器C定期发送Hello报文，包含发送的端口和IP地址。



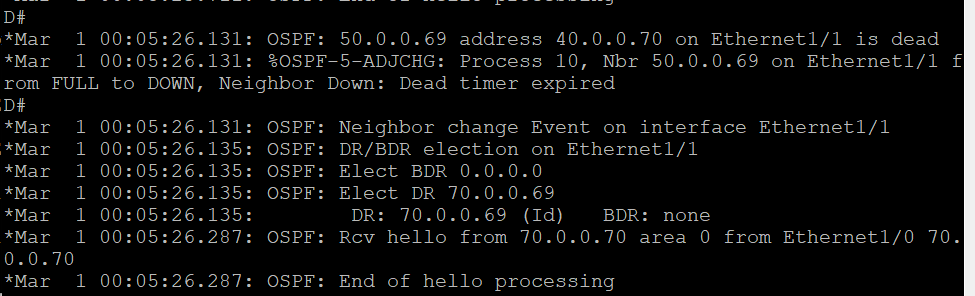
在路由器A发送的Hello报文中，包含了路由器的ID、当前的DR、BDR等。

* 数据库同步：
* RIP协议：

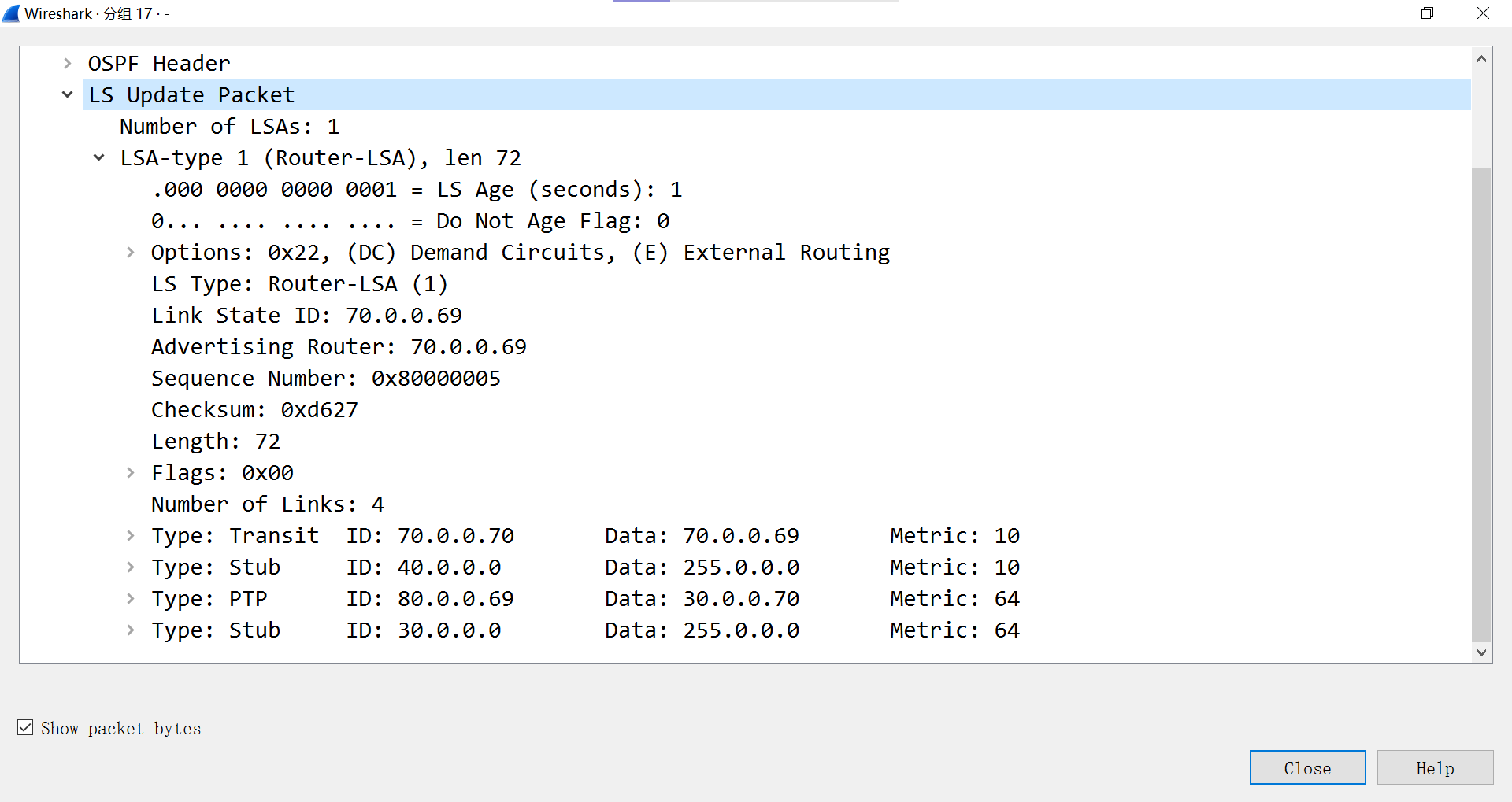


RIP协议通过定期广播或多播其整个路由表来同步数据库。当一个路由器收到来自邻居的路由更新时，它会更新自己的路由表，并将这些更改传播给其他邻居。如图，D收到来自C的路由表更新时，D更新了自己的路由表，并传播给其他邻居。

* OSPF协议：



OSPF协议使用洪泛法来同步链路状态数据库。当一个路由器的链路状态发生变化时，它会创建一个新的LSA（链路状态公告），并将其发送给所有邻居。邻居路由器在收到LSA后，会更新自己的数据库，并将LSA转发给其他邻居。如上图，关闭了路由器E后，D建立新的链路状态，并将其发送给所有邻居。

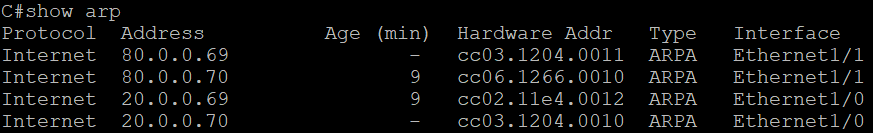


该LSA表示路由器D发出更新链路状态，且50.0.0.0子网被删除。

（3）写出在你的拓扑中，数据包从某台PC发送给其他PC的完整过程（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。

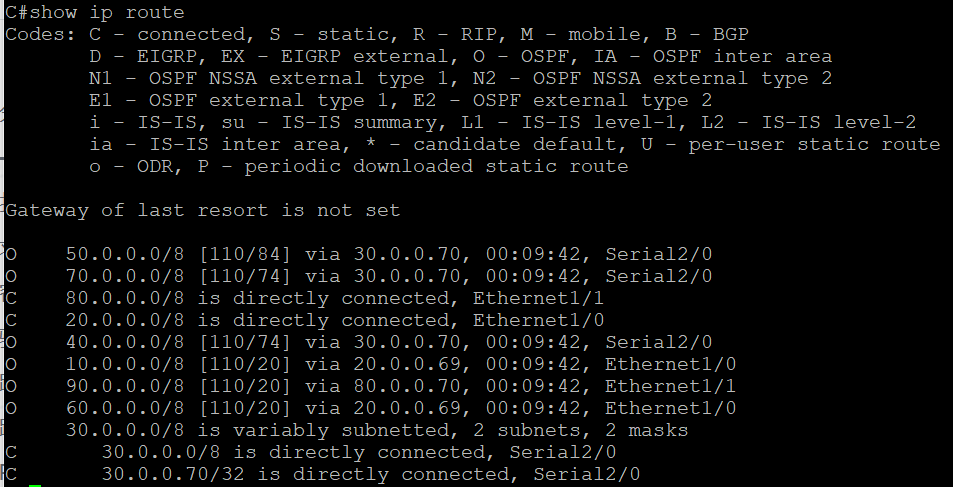
在从路由器A ping到路由器F时：

* ARP过程：当源PC需要发送数据包到目标PC时，它首先需要知道目标PC的MAC地址。这是通过ARP（地址解析协议）完成的。源PC首先检查其ARP缓存，看是否已经知道目标PC的MAC地址。如果已知，它将直接使用这个地址。如果未知，它将广播一个ARP请求，询问目标PC的MAC地址。目标PC收到ARP请求后，会回复一个ARP响应，包含其MAC地址。源PC收到响应后，会更新其ARP缓存，并使用目标PC的MAC地址来发送数据包。



如上图，路由器C的ARP表中包含相邻的端口的MAC地址，第一次访问相邻的端口时，先访问ARP表，如若没有，则广播一个ARP请求，收到回复时，ARP表项更新，之后可以直接使用相邻端口的MAC地址来发送数据包。

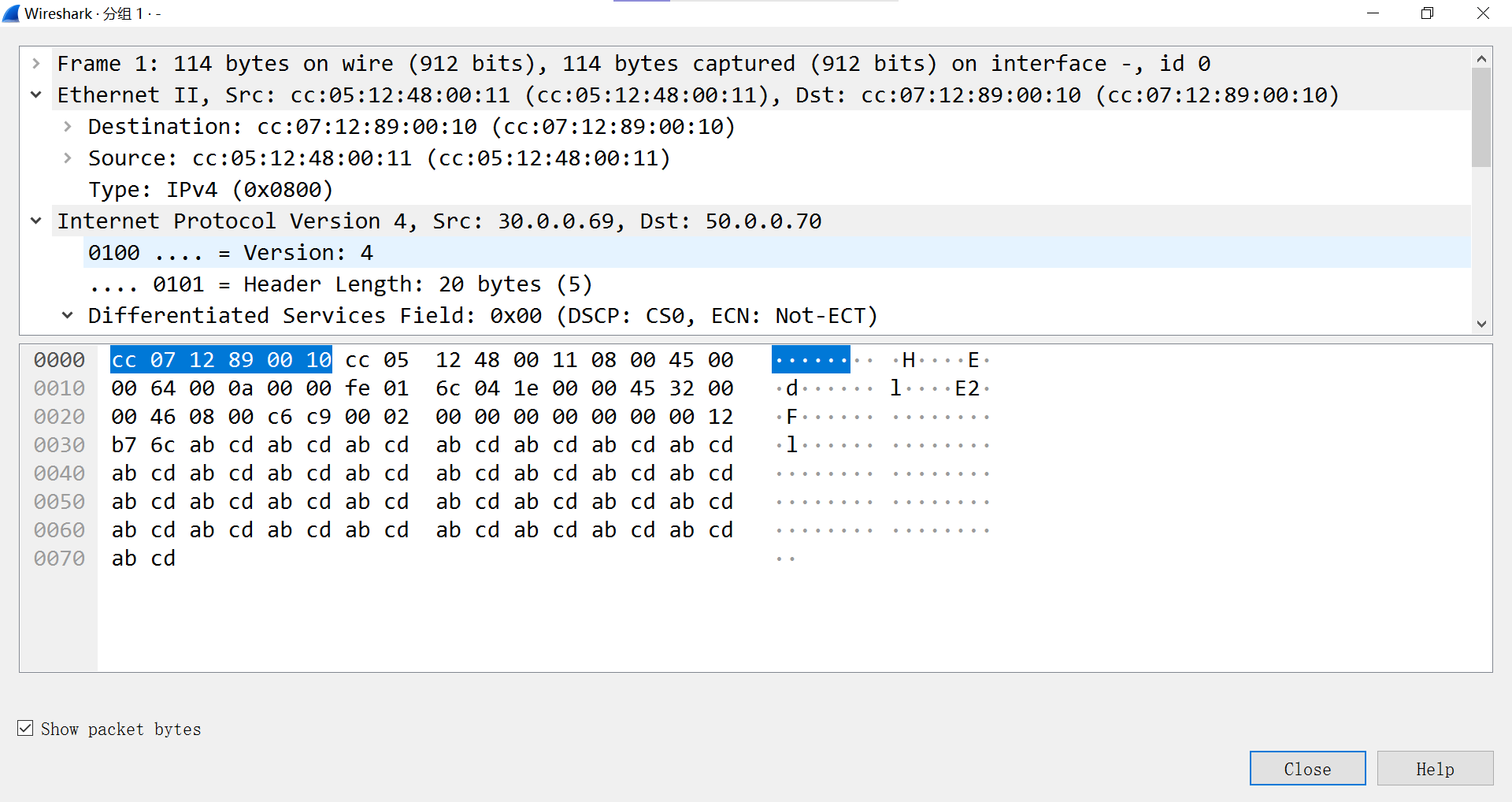
* 路由表匹配过程：在数据包从源PC发送到目标PC的过程中，可能需要经过一个或多个路由器。每个路由器都有一个路由表，用于决定如何转发数据包。当路由器收到一个数据包，它会查看数据包的目标IP地址，然后在路由表中查找匹配的条目。匹配的条目告诉路由器应该将数据包发送到哪个下一跳地址或网络接口。



如图，在路由器C收到数据包时，根据目标IP地址在路由表中选择下一条的端口，因此数据包可以正确的在路由线路中传播。

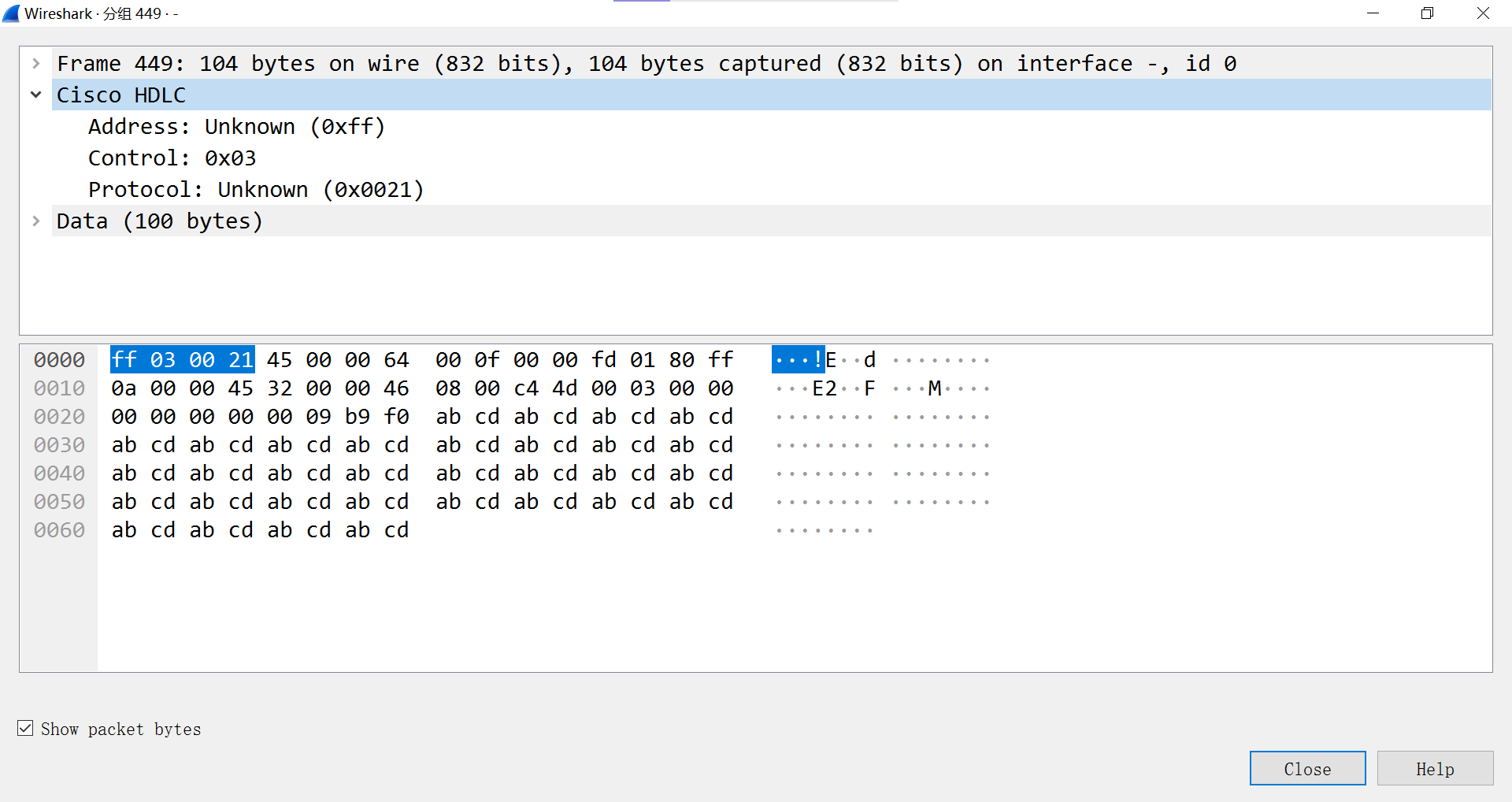
* 链路层协议封装过程：在数据包被发送之前，它需要被封装成一个链路层帧。这个过程通常由链路层协议（如以太网协议）完成。链路层帧包含源和目标的MAC地址，以及上层的数据（如IP数据包）。链路层帧还包含一些其他的控制信息，如帧类型和错误检测码。

在路由器C上ping路由器F的端口e1/0，抓取路由器E和路由器F之间线路上的包。



数据链路层帧头封装在IP帧头，数据链路帧头包含源和目的MAC地址，以及上层使用的数据包类型。上图中。目的MAC地址是0-5字节，源MAC地址是6-11字节，IP包类型是第12字节，代表上层数据类型是IPv4。

在C和D之间抓取数据帧，在从A ping B的过程中，抓取到如下的帧：



显示该帧的数据链路层协议是HDLC，但是我们设置的协议是PPP，这与预期不符，通过查阅Cisco官方文档，发现该帧头ff 03 00 21对应的数据链路层协议是PPP。因此，这是Wireshark的帧解析错误，Cisco确实是正常封装该帧来正常工作的。