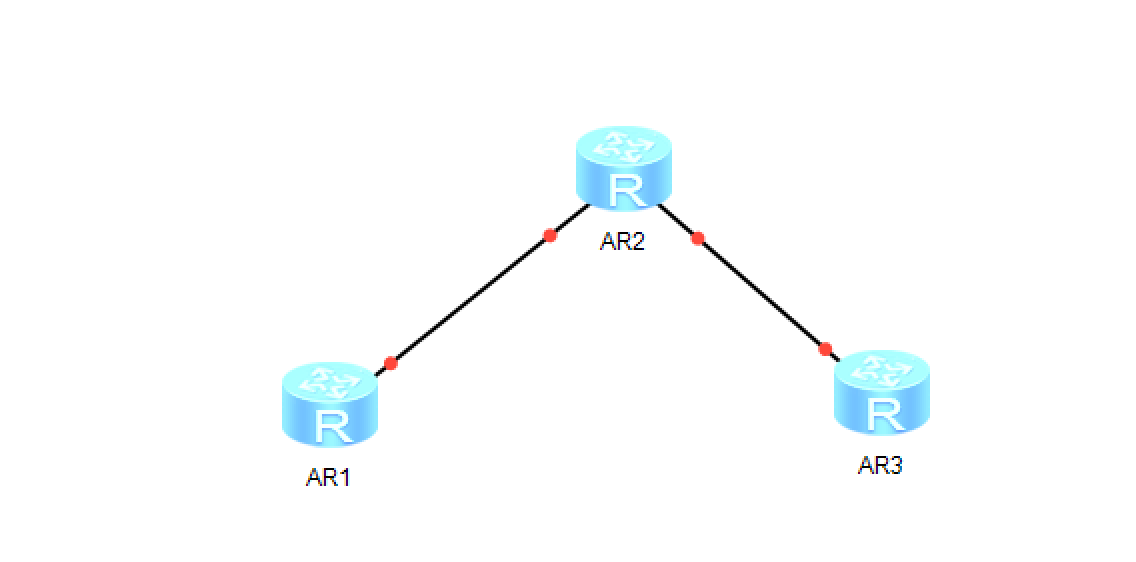
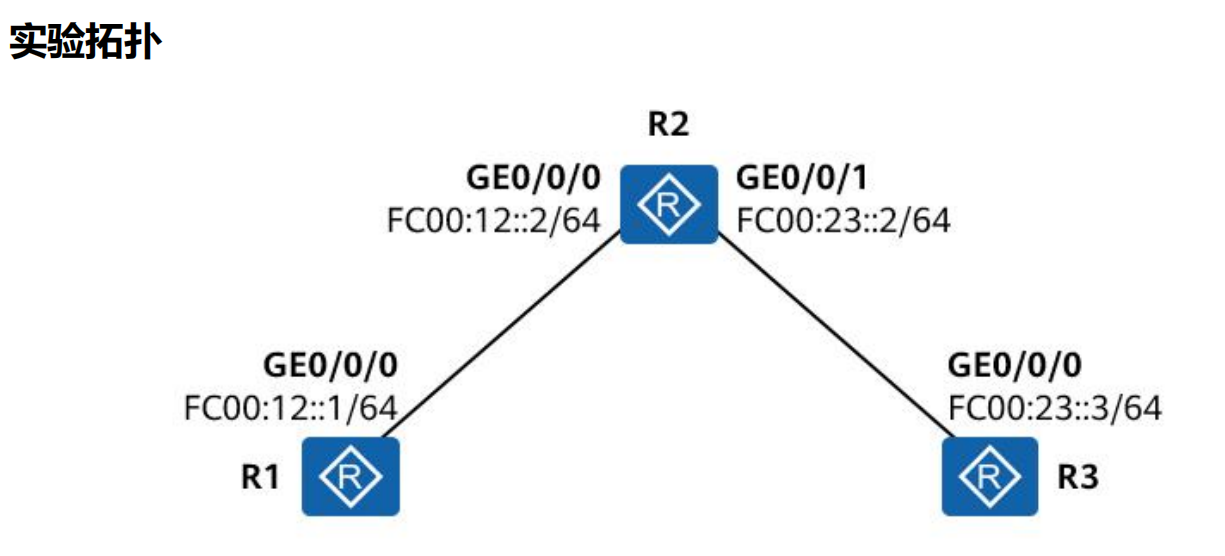
|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 王小龙 | 班级 | 2020211314 |
| 学号 | 2020211502 |  |  |

# 实验说明

**实验拓扑如下两张图**：



**实验目的**

1. 掌握数据报文捕获及分析方法。

2. 理解 RA 报文及无状态地址自动配置过程。

3. 理解 DAD 地址冲突检测机制工作过程。

4. 理解 IPv6 网络中的地址解析过程。

5. 分析 Ping 与 Tracert 应用所使用的 ICMPv6 报文及工作原理。

6. 理解 IPv6 PMTUD 机制及其工作原理。

**实验需求**

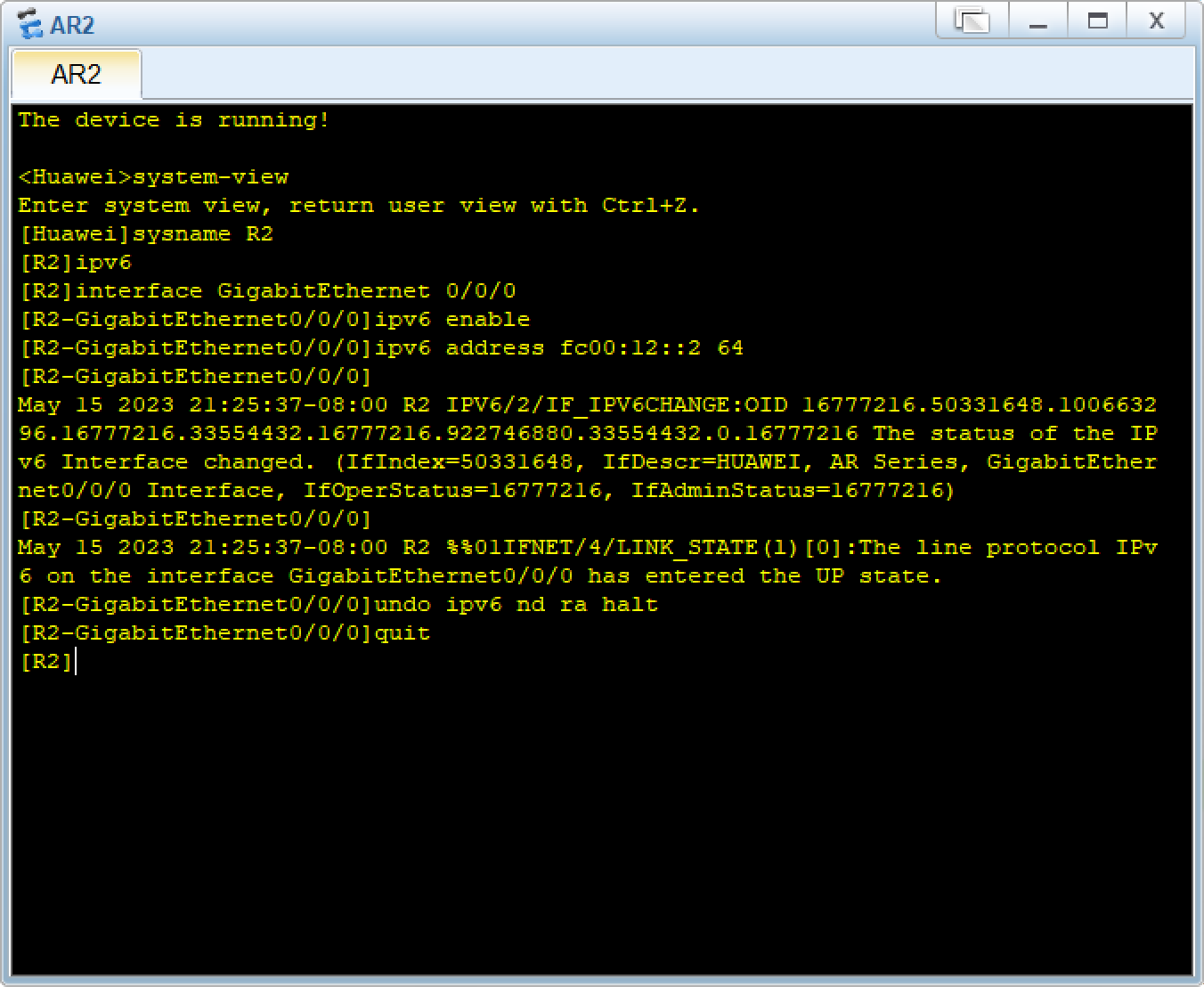
在本实验拓扑中完成基础 IPv6 配置，观察各类常见的 ICMPv6 报文在网络中的功能与

应用。

# 结果验证

1. 完成R2的基础配置

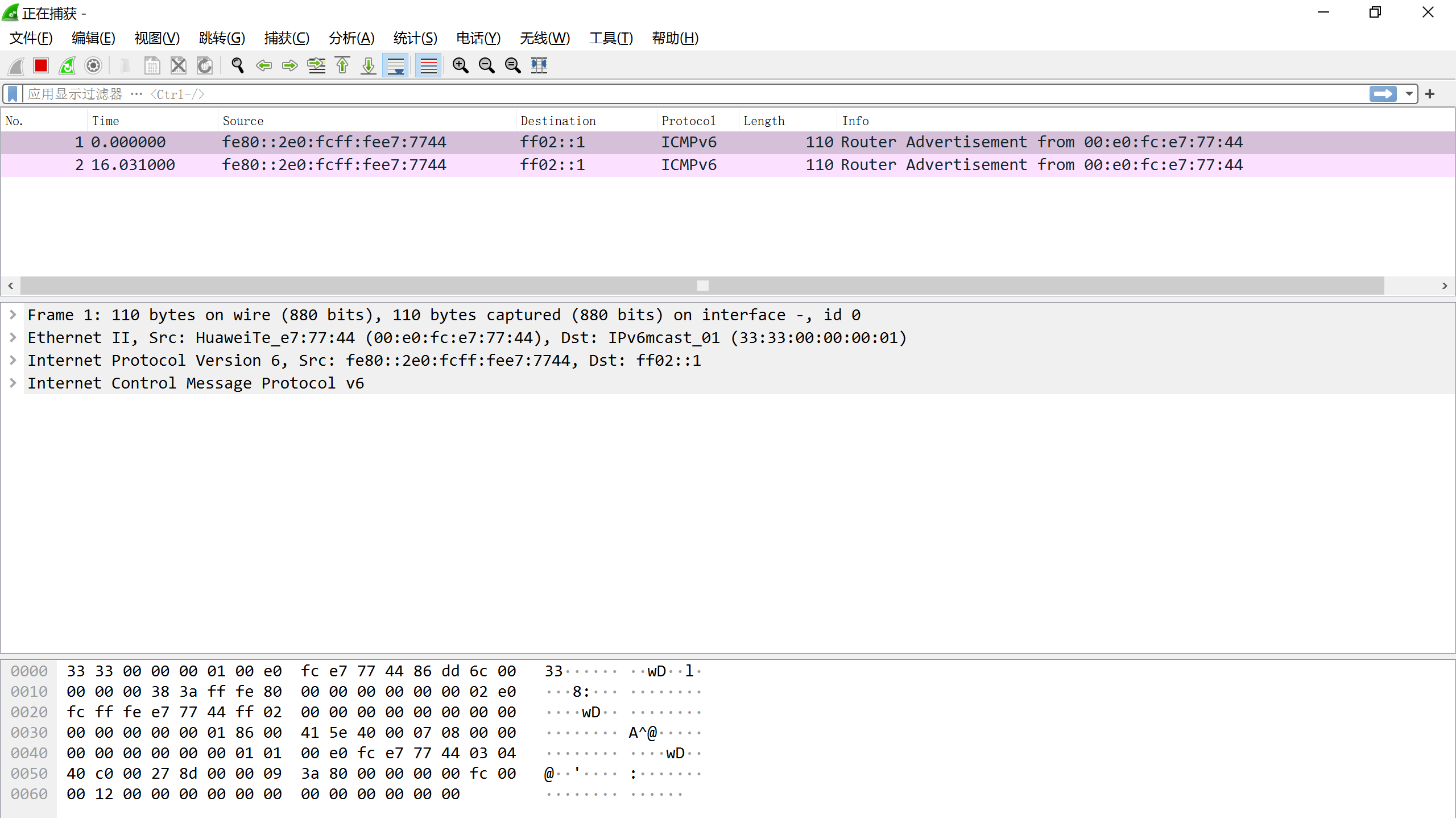
在 R2 上完成如下配置：



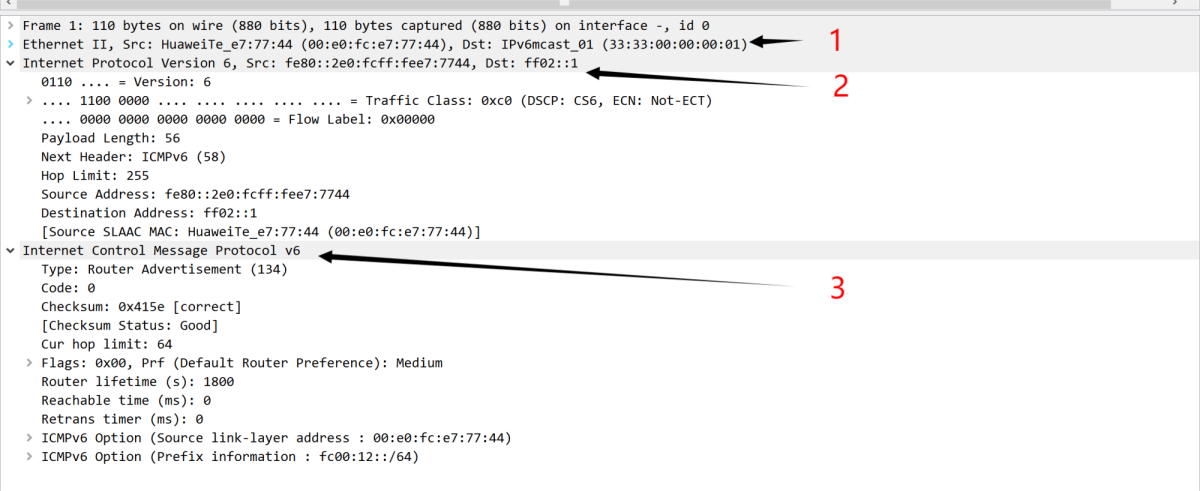
2. 观察RA报文与无状态地址自动配置过程

在 R2 的 GE0/0/0 接口上单机鼠标右键，点击“开始抓包”启动抓包程序Wireshark

此时，我们将观察到如下窗口：



展开相应内容查看 RA 报文的详细信息如下：



如上图所示，从数据帧头①可以看出，报文的目的 MAC 地址为 33:33:00:00:00:01，这

实际上是一个组播 MAC 地址，对应组播 IPv6 目的地址 FF02::1，这个组播地址对应本

链路上的所有 IPv6 节点，这表明该 RA 报文发往链路上的所有节点。

从 IPv6 包头②可以看出该报文发往 FF02::1，并且 NextHeader 为 58，对应 ICMPv6，

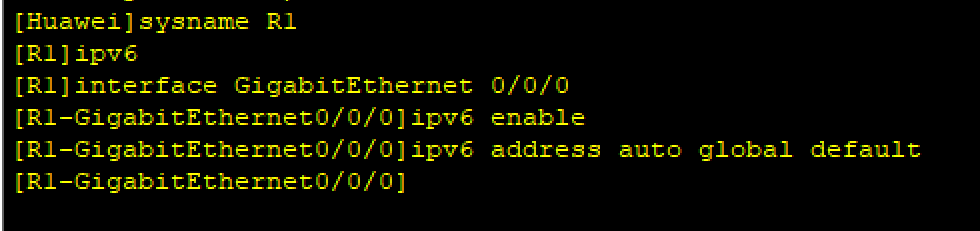
标明该头部后面跟随的是 ICMPv6 报文。

从 ICMPv6 报文③可以看出该报文的类型为 134（Router Advertisement，RA）报文，

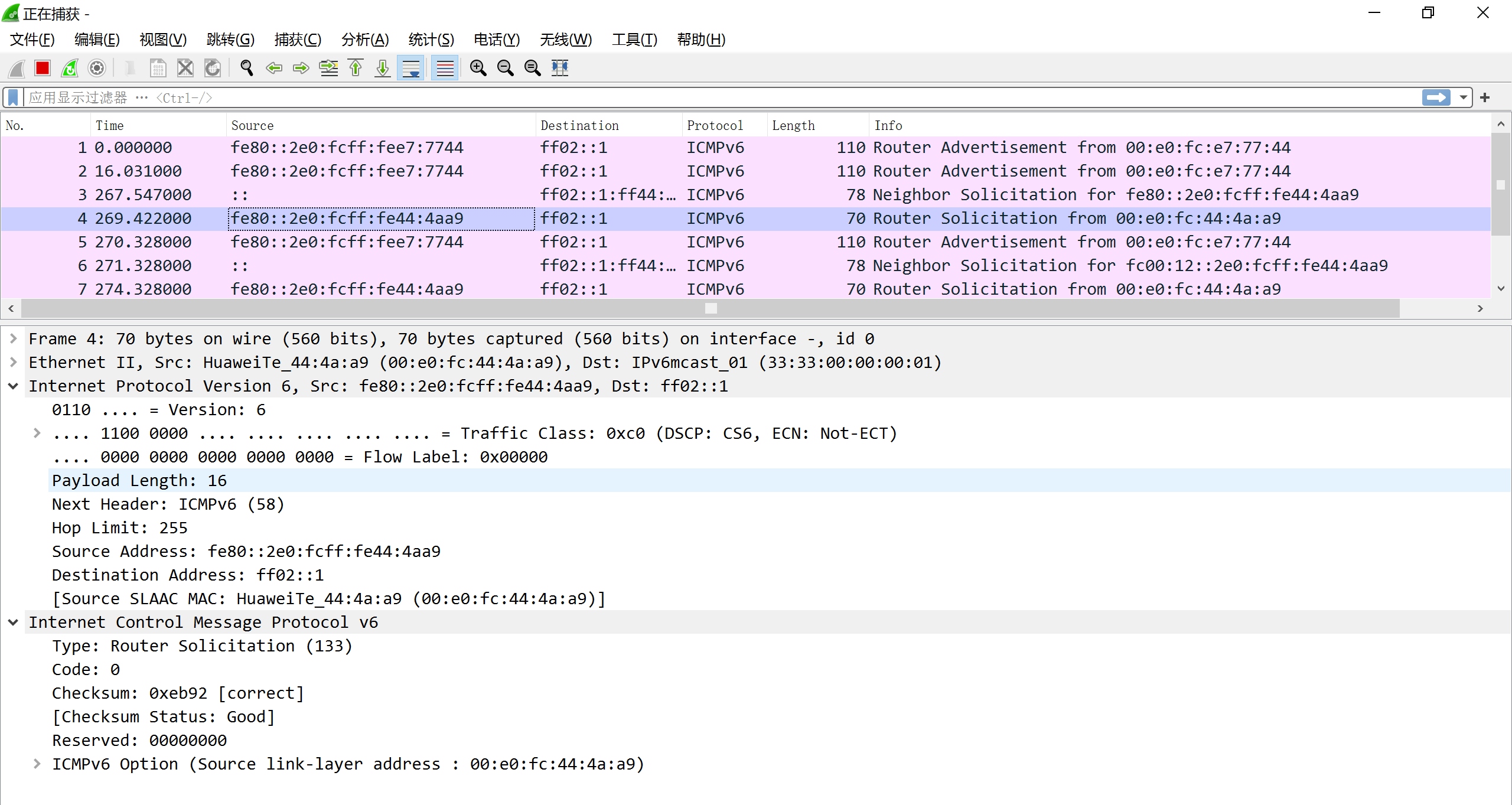
且报文携带两个可选字段（Option），其中一个描述 R2 的接口 MAC 地址，另一个则描

述 R2 通告的 IPv6 地址前缀 FC00:12::/64，该前缀可用于实现无状态地址自动配置。

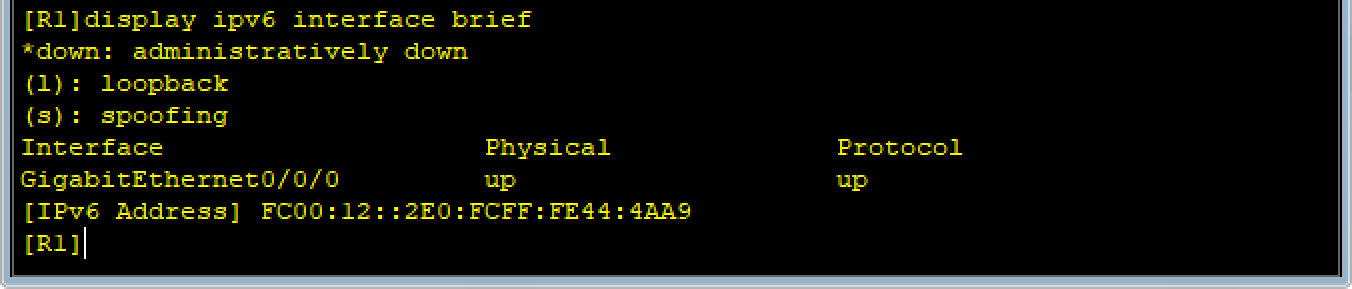
接下来我们在 R1 上配置其 GE0/0/0 接口：



完成配置后，R1 将主动发送 RS 报文，请求 R2 发送 RA 路由器通告报文：

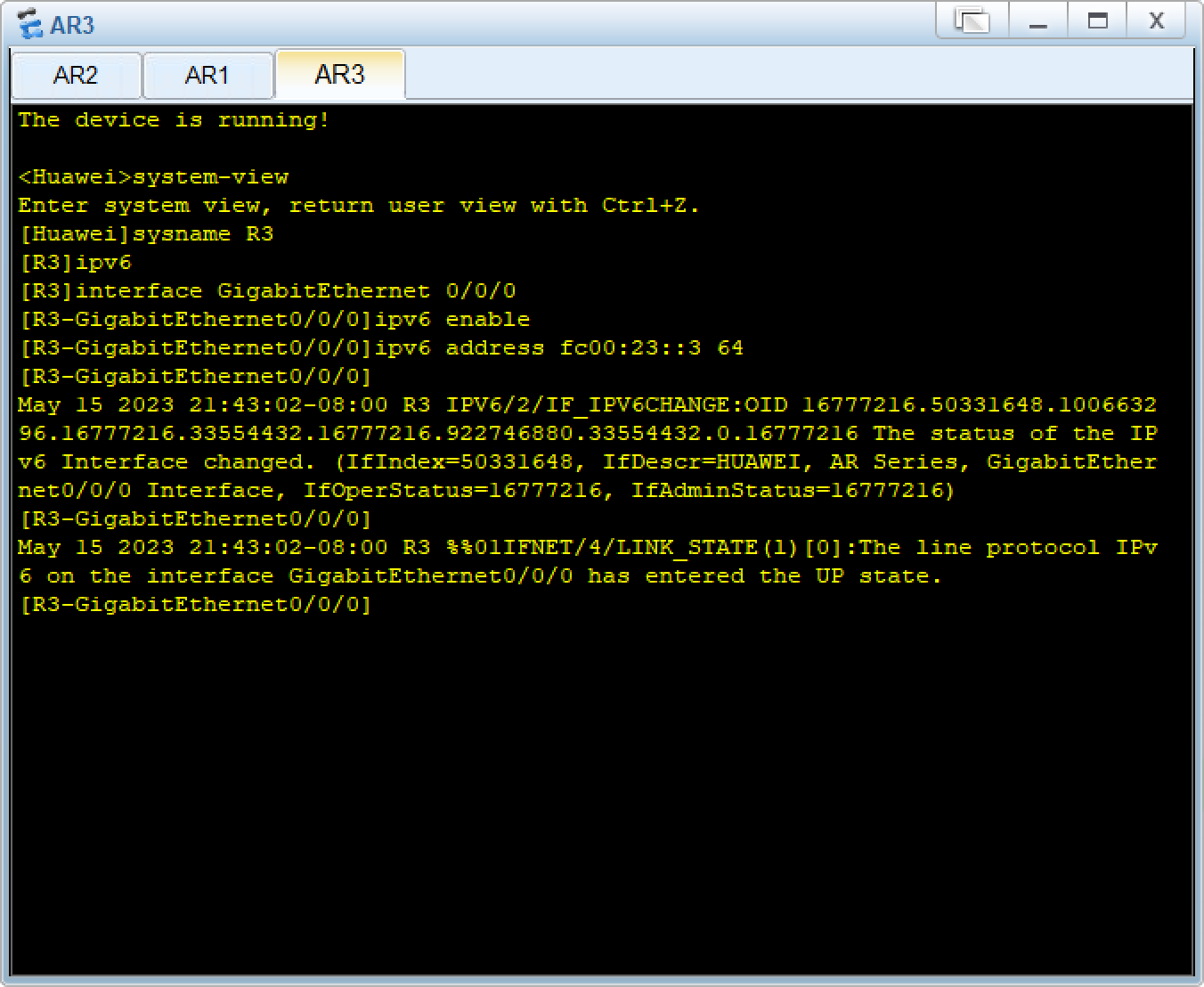


此时 R1 已经通过无状态地址自动配置方式获得 IPv6 地址：



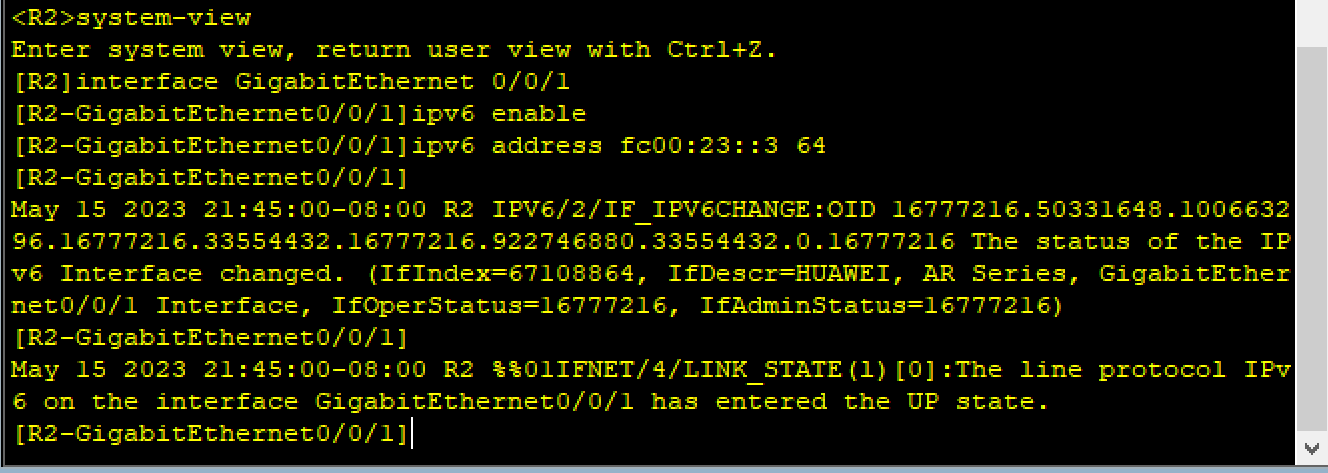
1. **观察DAD过程**

在 R3 上配置静态 IPv6 地址：

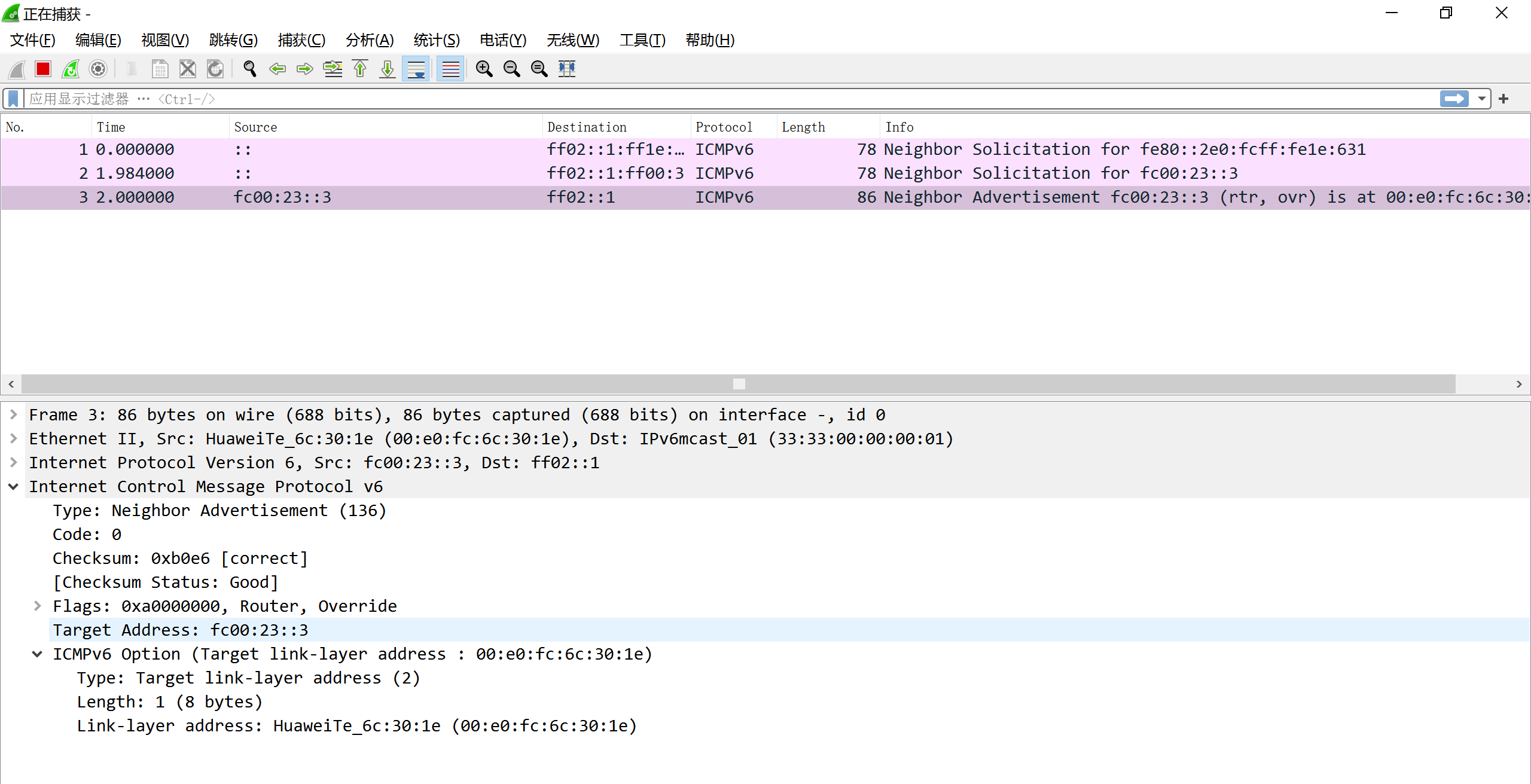


接下来在 R2 的 GE0/0/1 接口上开始抓包。

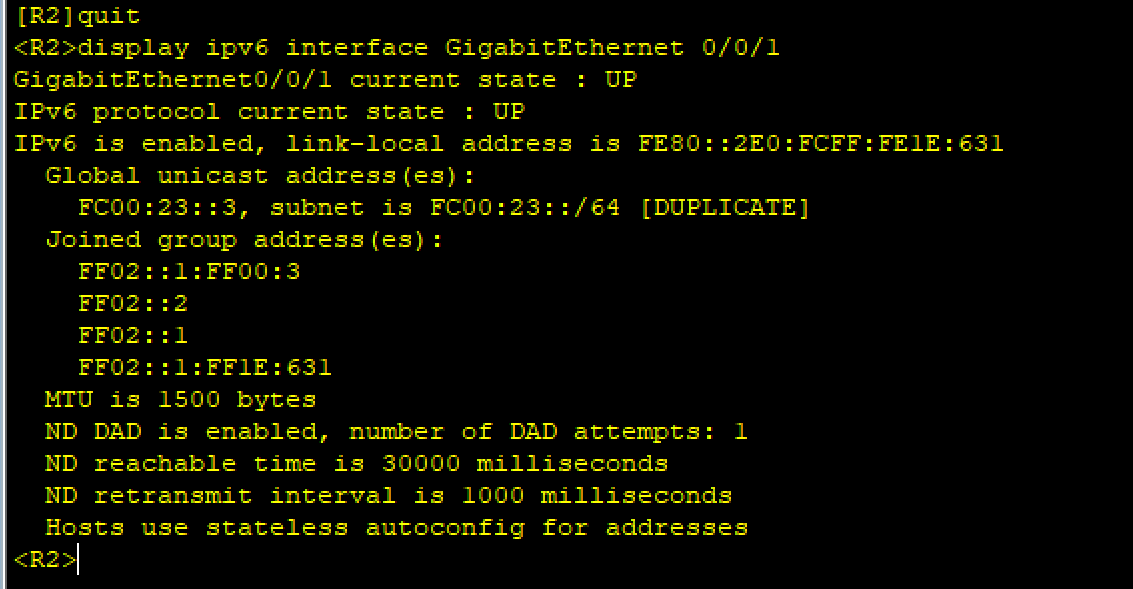
然后在 R2 上完成如下配置：



此时可以捕获到如下报文：

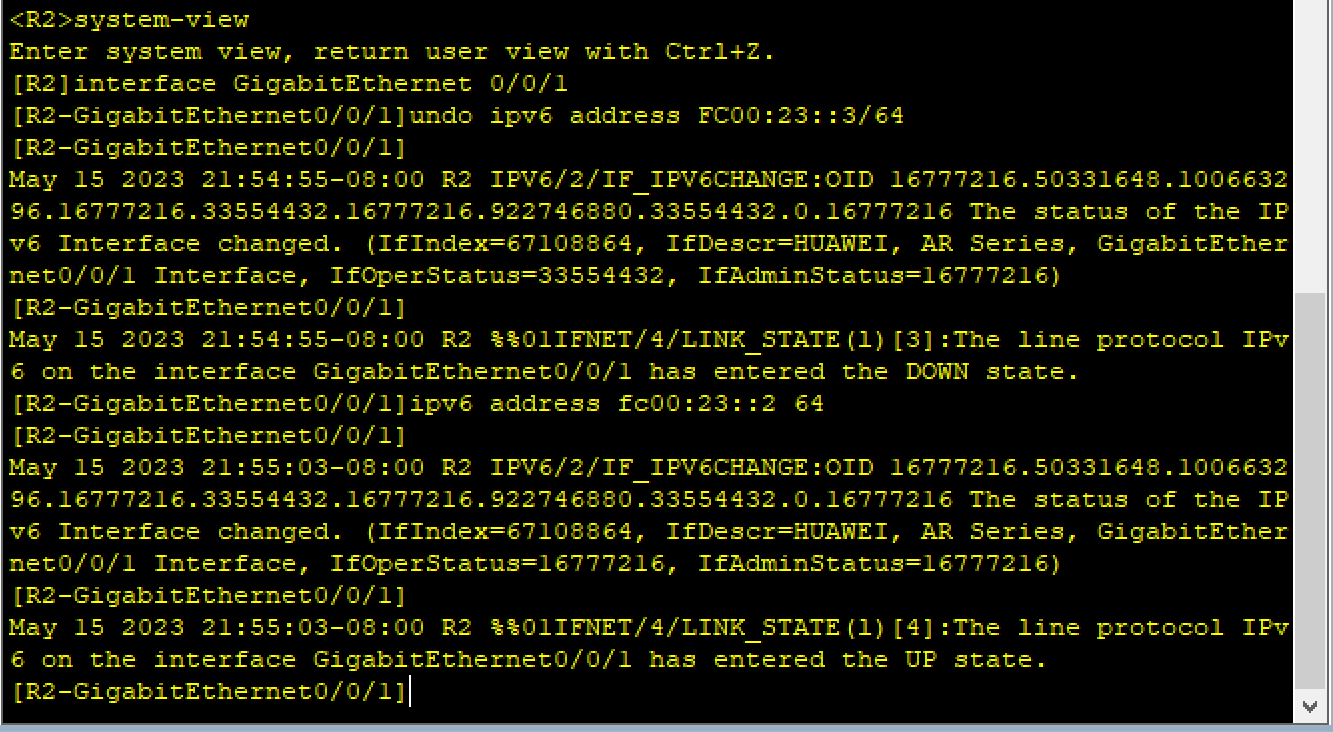


在 R2 上执行如下命令可观察到接口的地址状态：



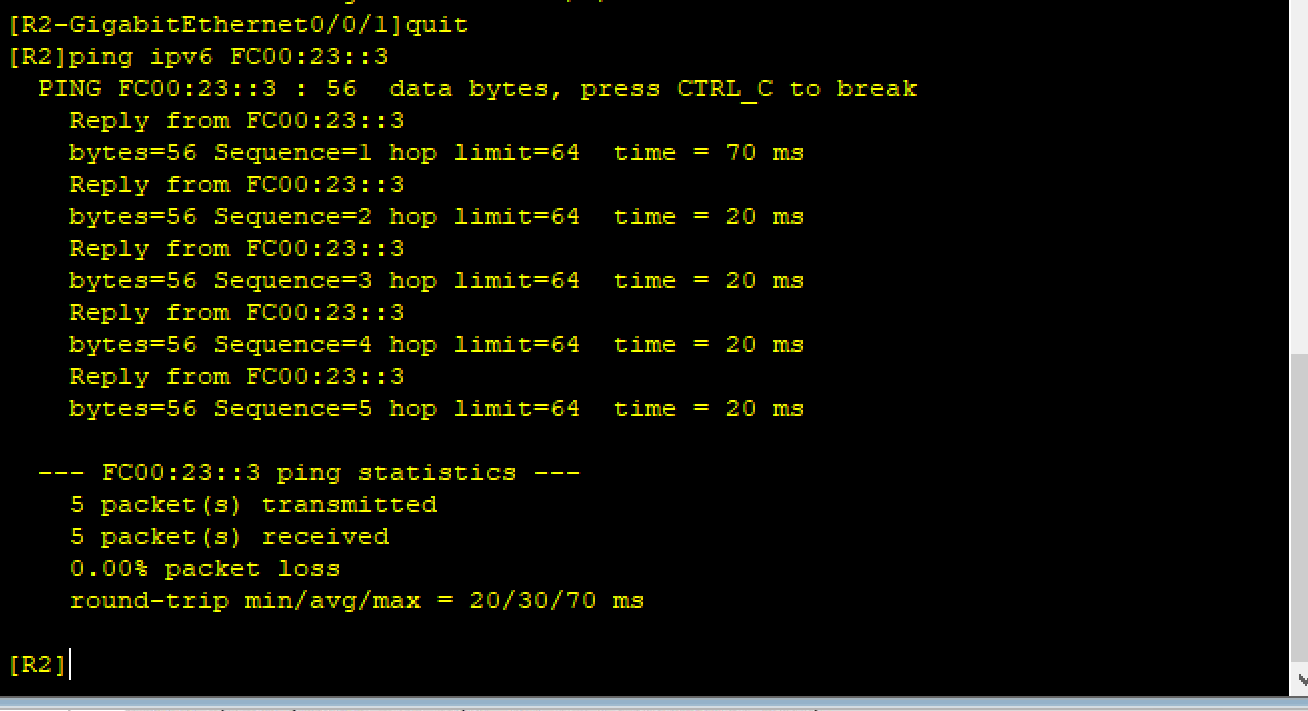
4. **观察地址解析过程**

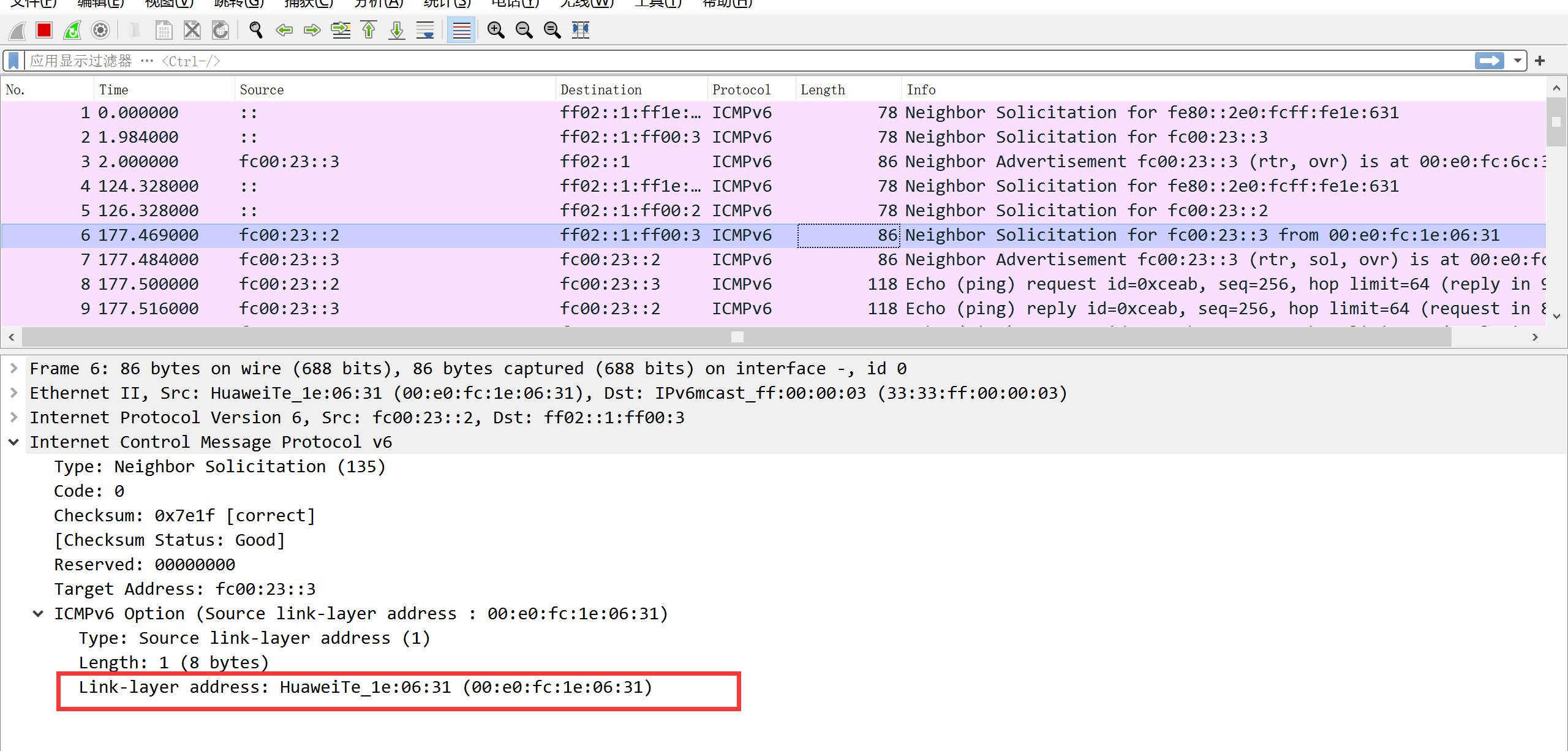
现在，将 R2 的接口地址修改为正确的地址：



在 DAD 检测通过后，R2 正式启用 FC00:23::2 地址，此时我们依然在 R2 的 GE0/0/1

接口上进行抓包，然后在R2上执行命令：ping ipv6 FC00:23::3

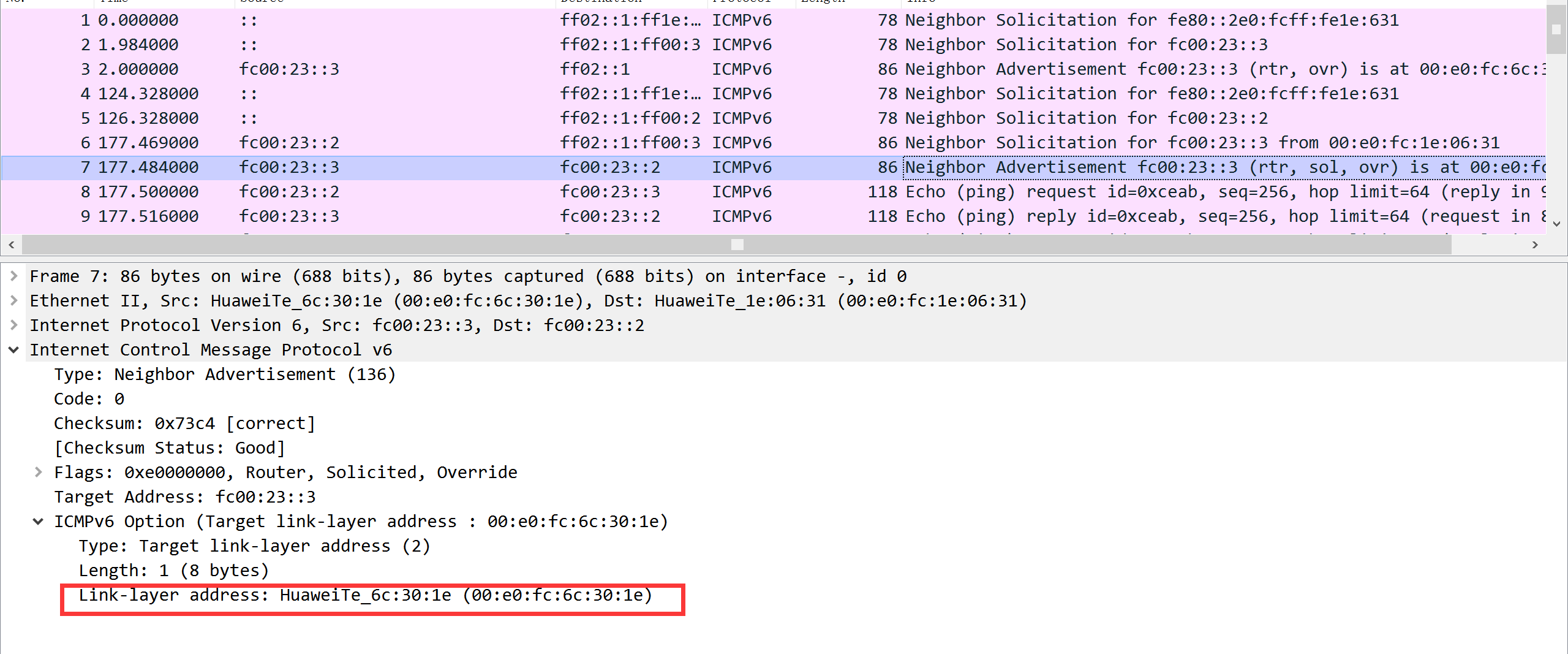




从上图可以看到，R2（FC00:23::2）首先发送了一个 NS 报文，该报文的 ICMPv6 载

荷中带有 R2 接口的 MAC 地址信息，这个报文发往目标地址 FC00:23::3 对应的被请求

节点组播地址 FF02::1:FF00:3，R3 恰恰在侦听这个地址，于是使用 NA 报文进行回应：



如上图所示，这个 NA 报文直接单播发给了 R2，其中填充着 R3 的接口 MAC 地址。

如此一来，R2 与 R3 便相互知晓了对方的 MAC 地址，可以正常交互 IPv6 报文。

5. **捕获Ping报文**

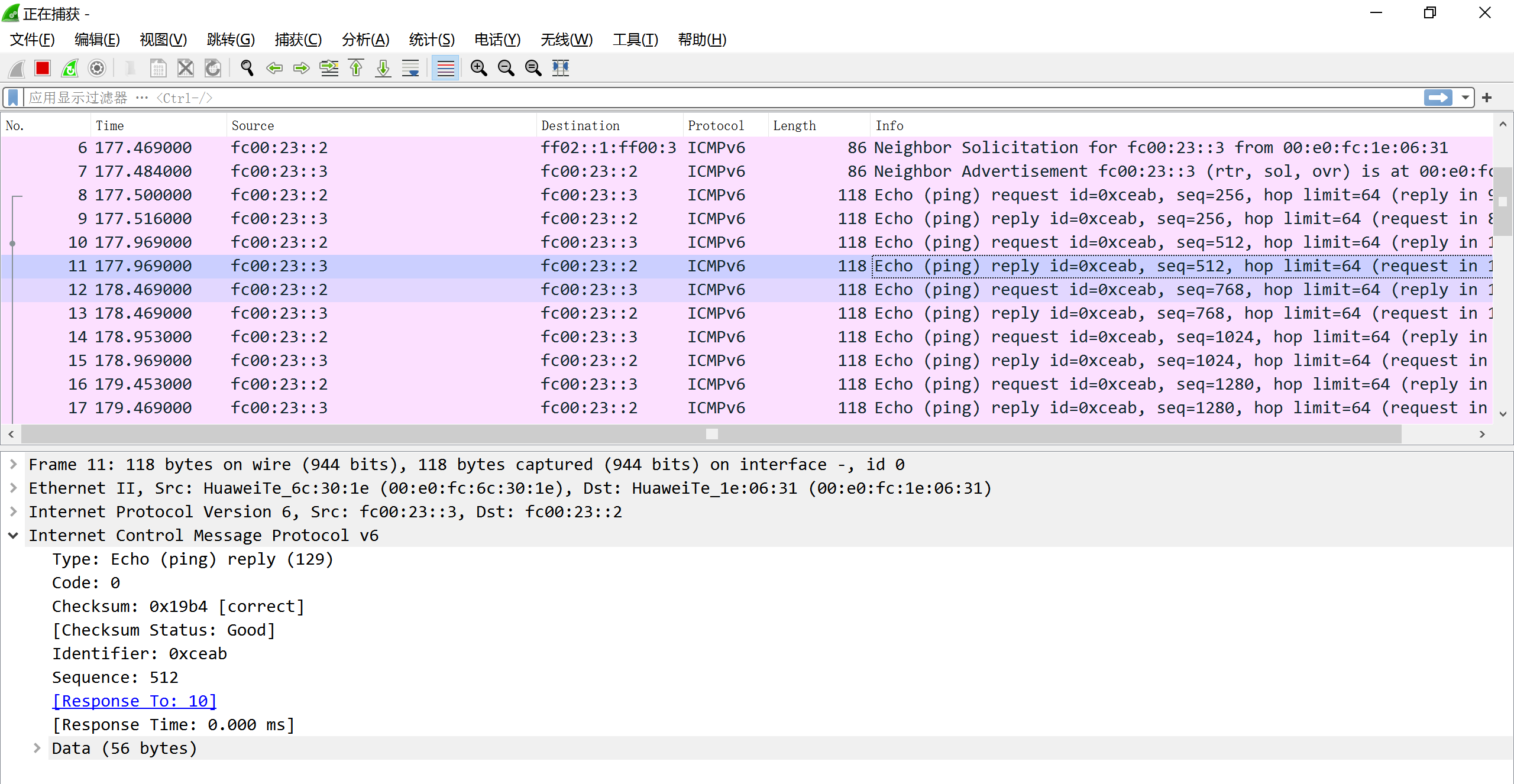
**在 ICMPv6 报文中，Echo Request 和 Echo Reply 报文是非常基础且重要的报文，被**

**用于 Ping 应用程序等，当我们在一个 IPv6 节点上执行 Ping 操作探测到某个目的地址**

**的可达性时，实际上该应用将触发一个 ICMPv6 Echo Request 报文发往目的地址，如**

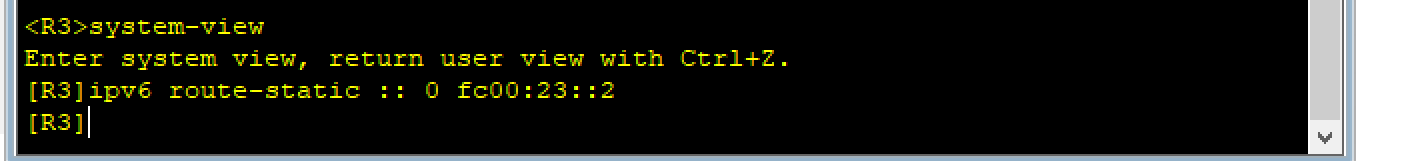
**果收到了对方回应的 Echo Reply，则认为网络是可达的。**

下图展示的是当 R2 ping R3的 FC00:23::3 地址时，捕获到的 Echo Request 和 Echo Reply 报文：



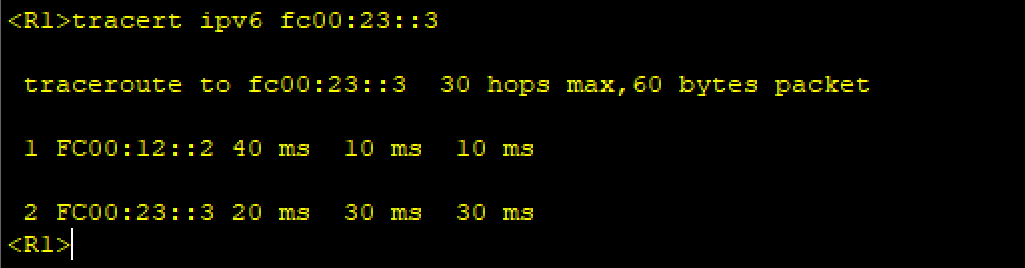
**6. 捕获Tracert报文**

在 R3 上添加默认路由，下一跳为 R2：



然后在 R1 的 GE0/0/0 接口上开始抓包。

此时我们在 R1 上执行如下命令：

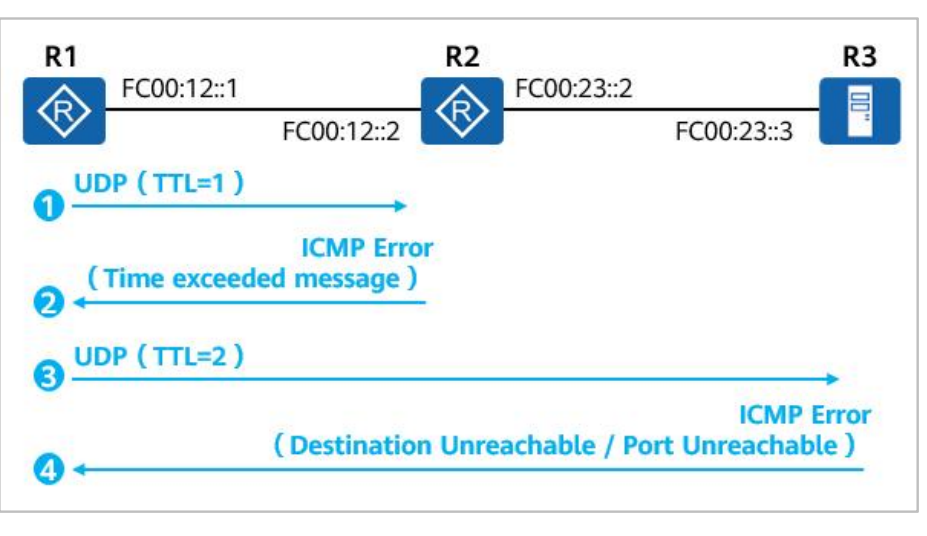


从上述结果可得知，从 R1 到 R3 经过了 FC00:12::2，最终到达 FC00:23::3。当源与目

的节点之间存在多跳设备时，Tracert 执行的结果更加直观。因此面对一个复杂的网络

时，这个工具可以方便地帮助网络管理员识别流量的转发路径。

Tracert 的实现原理如下：

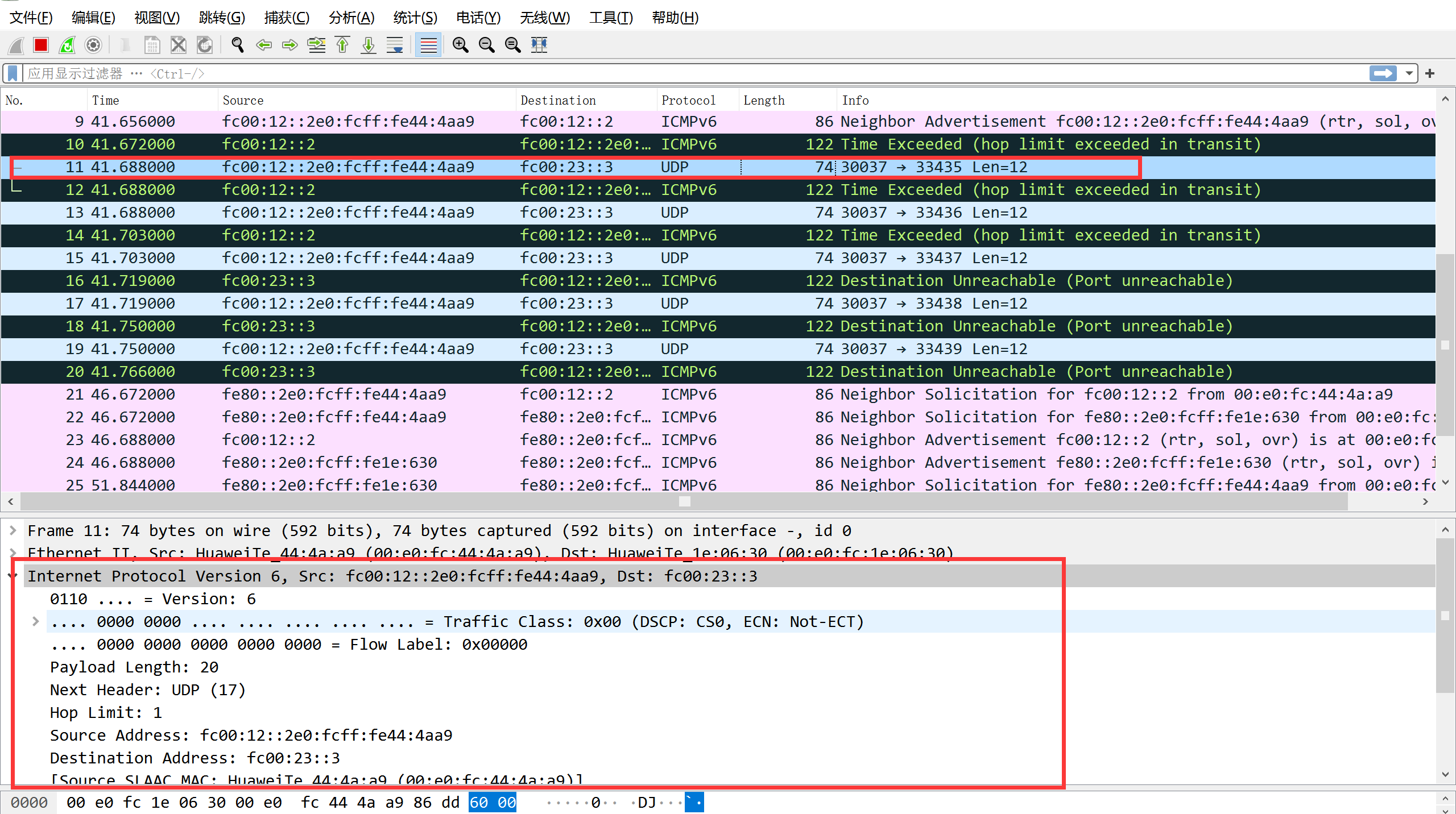


R1 首先构造第一个发往目标地址 FC00:23::3 的 UDP（UDP 目的端口为特殊的 33434，

该端口不会被具体的应用所使用）报文，这个报文的内容是随机填充的，没有实际意义，

但是在该报文的 IPv6 头部中，R1 将 Hop Limit 字段设置为 1，这意味着报文在发出去

之后，只能传递一跳。R1 可能一次会发出多个相同的 UDP 报文。如下图所示：



R2 收到该报文后将 Hop Limit 字段值减 1 后发现值已为 0，因此立即向 R1 发送 ICMPv6

错误消息，告知报文的生存时间截止，这个错误消息的源地址为 R2 的接口地址。

R1 收到这个报错消息后，获得了第一跳设备 R2 的接口地址，然后将该地址打印在回

显中。

接着 R2 以 Hop Limit=2 继续发送 UDP 报文，如此反复。

直到报文到达目的地 R3，由于 R1 在 Tracert 中所使用的 UDP 端口在 R3 处并未侦听，

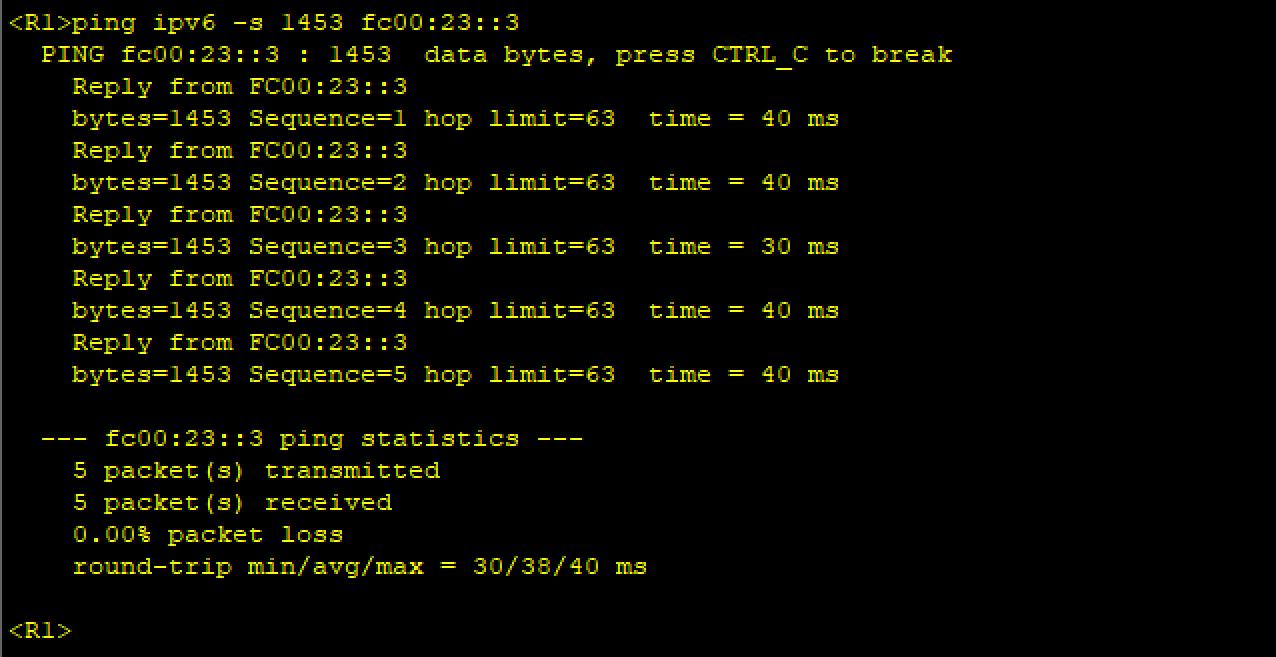
因此 R3 回应 ICMPv6 差错报文，告知 R1 目的端口不可达。

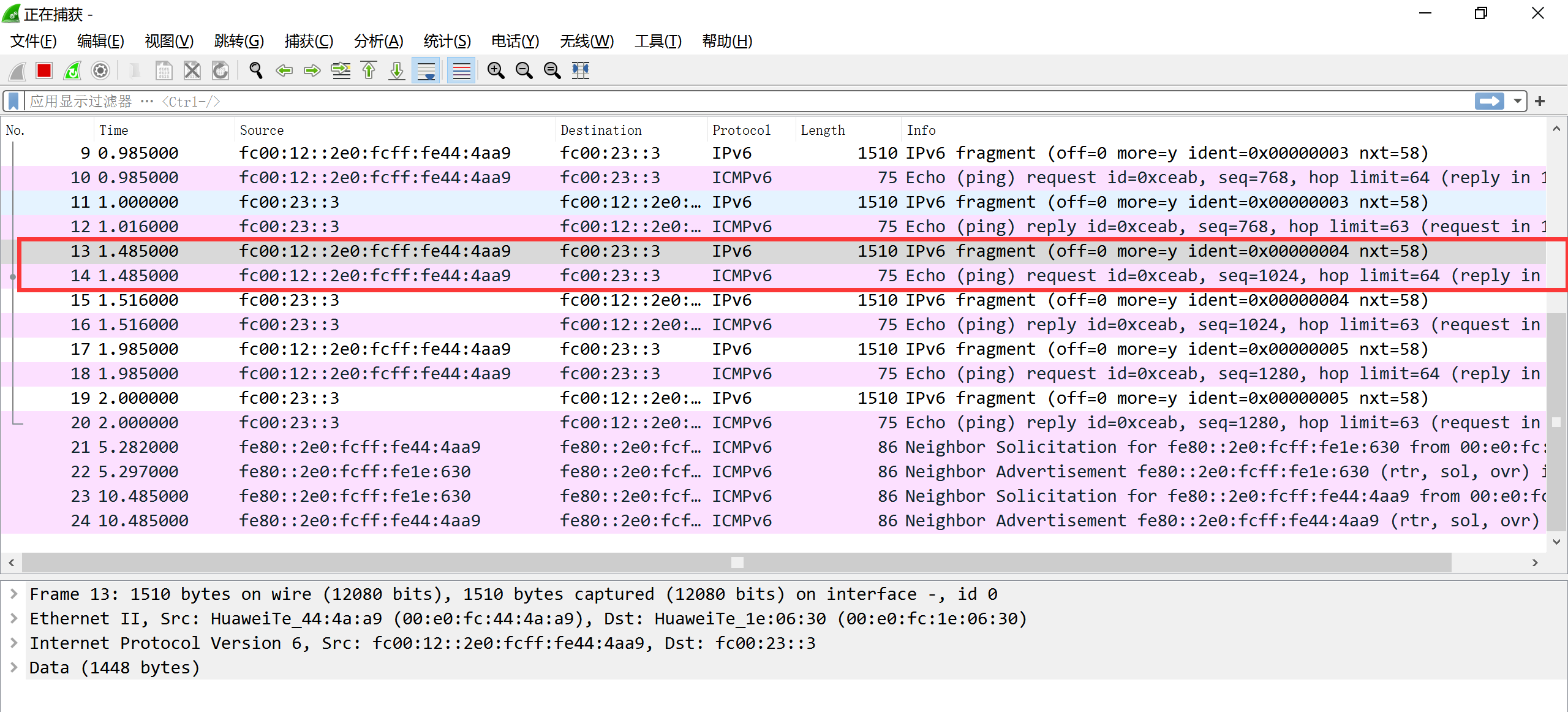
R1 收到该差错报文后即知晓最后一跳已到达。

**7. 观察IPv6 PMTUD机制**

完成上述配置后，R1 与 R3 已经能够相互通信。

此时在 R1 上执行 ping ipv6 -s 1453 fc00:23::3，则会在 R1 的 GE0/0/0 接口上捕获到如下报文：





从上图可以看出，R1 将一个载荷长度为 1453 字节的 ICMPv6 报文进行了分片，每个单独的报文被分为 2 片发往目的地 FC00:23::3。由于 R1 是以上报文的始发节点，因此它可以对报名进行分片，报文分片到达 R3 后，R3 再将分片进行重新组装。

值得注意的是，在 IPv6 中，中间转发设备不对 IPv6 报文进行分片，报文的分片将在始

发节点进行。因此，如果在本例中，若 R1 发出了长度超过 R2 的 GE0/0/1 接口 IPv6 MTU

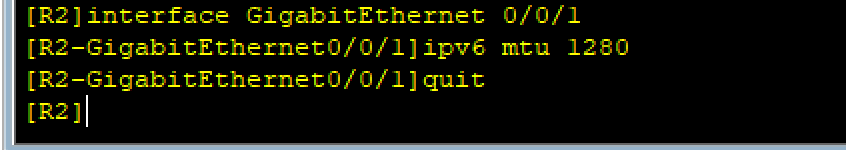
的报文，则 R2 是无法对其进行分片处理的，也无法转发该报文。

Path MTU 发现（PMTUD）机制用于解决该问题。PMTUD 的主要目的是发现路径上

的 MTU，当数据包被从源转发到目的地的过程中便可避免分片。

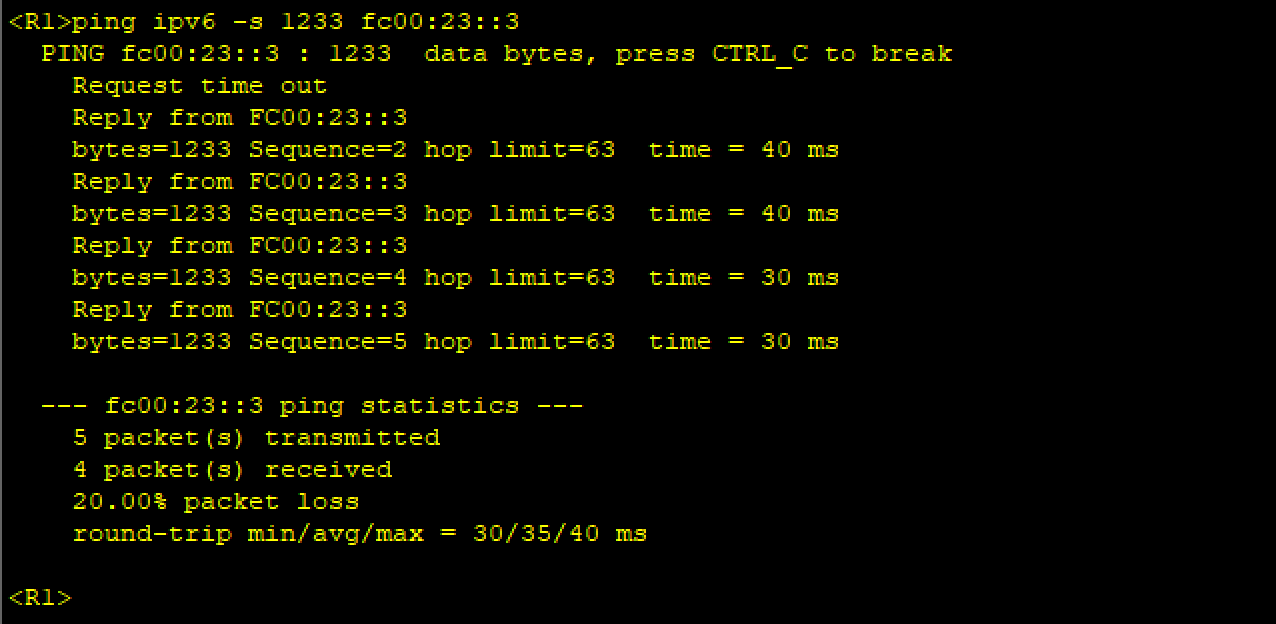
我们继续在 R1 的 GE0/0/0 接口上抓包，然后将 R2 的 GE0/0/1 接口的 IPv6 MTU 值修

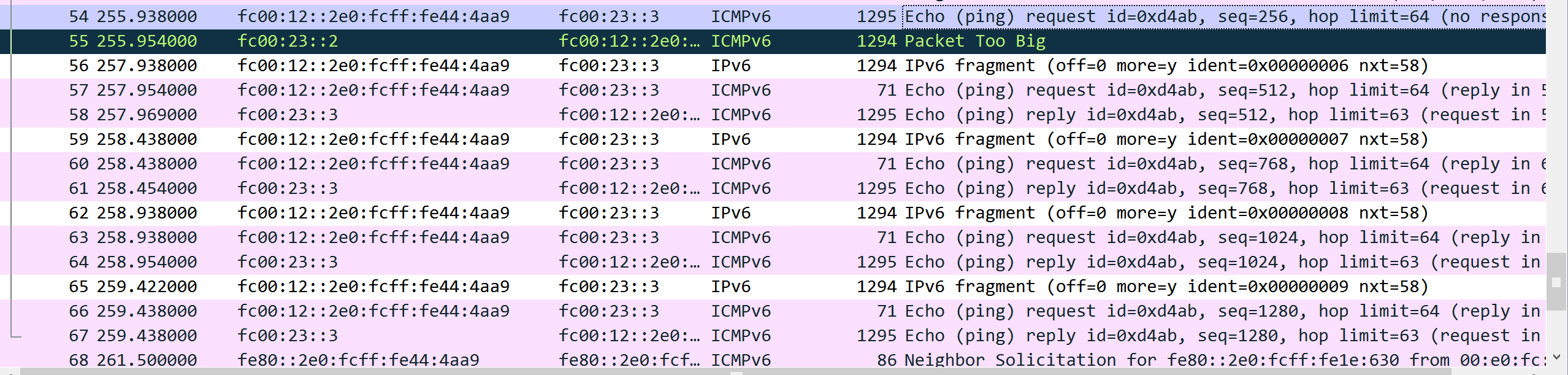
改为一个较小的值：1280 字节。



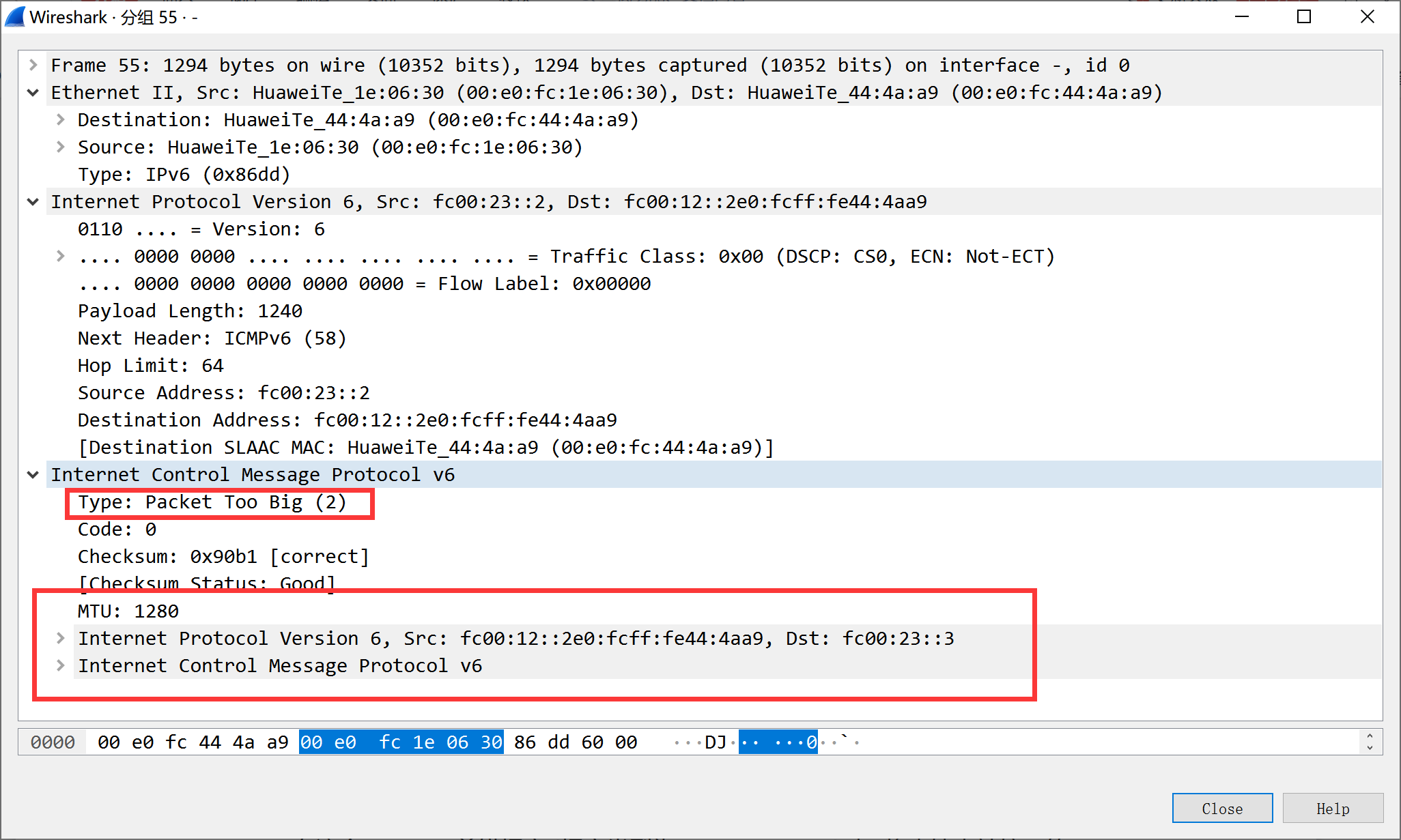
此时在 R1 上执行 ping ipv6 -s 1232 fc00:23::3 命令，可以触发 R1 产生一个人载荷长度 1232 字节的 ICMPv6 Echo Request 报文，这个长度加上 40 字节 IPv6 基本头部及 8 字节 ICMPv6 Echo Request 头部，正好是 1280 字节——等于报文到达 R3 的途中需经过的 R2 的 GE0/0/1 接口的 IPv6 MTU 值。

该命令执行后，从 R2 的 GE0/0/1 接口所捕获的报文中不会发现异常。接下来，将在R1 上执行的命令变更为 ping ipv6 -s 1233 fc00:23::3，会发现能够 Ping 通 R3，但是抓包的结果有了变化：



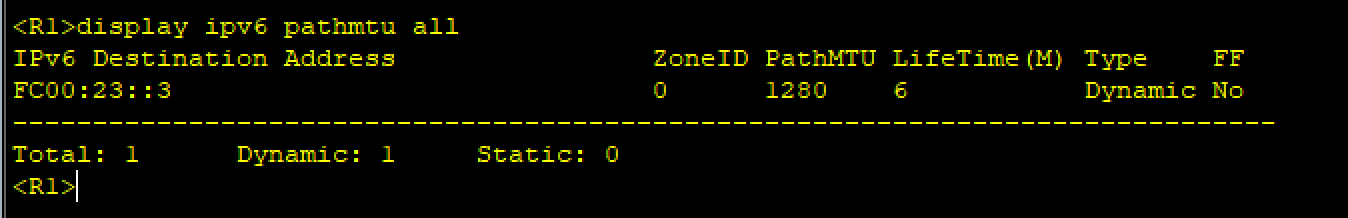


在上图中，第 55 个报文为 R1 发出的首个 ICMPv6 Echo Request 报文，这个报文到达R2 后，因为长度超出了其出站接口 GE0/0/1 的 IPv6 MTU，故被丢弃，R1 将无法收到对于这个 Echo Request 报文的应答。此时 R2 立即通过 ICMPv6 差错报文通知 R1，这个通知在第 56 个报文中体现，这个报文的详细内容如下：



R2 在这个 ICMPv6 差错报文（报文类型为 Packet Too Big）中，将本地出站接口的 IPv6MTU 值 1280 带给了 R1，同时也在该报文中将此前被其丢弃的、R1 所发出的 EchoRequest 报文附上了。

R1 收到上述报文后，得知自己发出的报文因为尺寸过大被丢弃，而且报文转发路径上目前探知的最小 MTU 为 1280，于是形成如下缓存表项：



如此一来，后续再发往 FC0023::3 的报文，将会以 1280 字节作为 MTU，如果报文的长度超出该值，则始发路由器 R1 将直接对齐进行分片，因此当 R1 Ping FC00:23::3时，首个 ICMPv6 报文被丢弃，后续的报文则可以被顺利转发。

# 思考题

1. 当我们在路由器的 IPv6 接口上执行 undo ipv6 nd ra halt 命令后，该接口将周期性地

发送 RA 报文，这些报文的目的 IPv6 地址是？该报文的载荷有什么内容？

解答：

这些报文的目的ipv6地址是节点多播地址FF02::1；

该报文的载荷有如下内容：

路由器的地址，地址前缀和其他参数。

1. 当一台设备的接口获得 IPv6 地址后，设备立即启动 DAD 过程并在接口上发送一个 NS报文用于检测该地址是否已被使用，这个 NS 报文的目的 IPv6 地址是什么？这个地址是如何形成的？

解答：

这个NS报文的目的ipv6地址是该地址对应的被请求节点的组播地址；

该地址的形成方法：

将地址的最后24位替换为FF02::1:FF00:0/104前缀而形成的。

1. IPv6 报文头部中的“Hop Limit”字段有什么用途？

RA报文中的跳数限制（Hop Limit字段）表示报文能够经过的最大跳数，每经过一个设备，该数值减去1。当该字段的值为0时，报文会被丢弃。