

学号：202400130240	姓名：贾宗翰	班级：24.6
实验题目：Wireshark Lab: IP		
实验学时：2h	实验日期：2025.11.4	
实验目的：了解 ip 协议		
硬件环境：联想拯救者		
软件环境：wireshark		
<p>实验步骤与内容：</p> <p>在这个实验中，我们将研究著名的 IP 协议，重点关注 IPv4 和 IPv6 数据报。这个实验分为三个部分。第一部分，我们将分析由追踪路由程序发送和接收的 IPv4 数据报包（追踪路由程序本身在 Wireshark ICMP 实验中进行了更详细的探讨）。第二部分，我们将研究 IP 分片；第三部分，我们将简要介绍 IPv6。</p> <p>第 1 部分：基本 IPv4</p> <p>在你的跟踪中，你应该能够看到由追踪器发送到你计算机的一系列 UDP 段（对于 MacOS/Linux）或 ICMP 回显请求消息（对于 Windows），以及中间路由器返回到你计算机的 ICMP TTL 超限消息。在下面的问题中，我们假设你使用的是 MacOS/Linux 计算机；对于 Windows 机器的情况，相应的问题应该是显而易见的。你的屏幕应该类似于图 2 中的截图，在该图中我们使用了显示过滤器“udp icmp”（参见图 2 中的浅绿色填充显示过滤器字段），以便仅显示 UDP 和/或 ICMP 协议的数据包。</p> <p>回答以下问题 3。如果作为课堂的一部分进行这个实验，老师会提供关于如何提交作业的详细信息，无论是书面还是在 LMS 中。</p> <p>1. 选择您的计算机通过跟踪路由命令发送给 gaia.cs.umass.edu 的第一个 UDP 段。（提示：这是脚注 2 中 ipwireshark-tracel-1.pcapng 文件中跟踪文件的第 44 个数据包）在数据包详细信息窗口中展开该数据包的互联网协议部分。您的计算机的 IP 地址是什么？</p>		

```
44 1.865637 192.168.86.61 128.119.245.12 UDP 70 64928
45 1.868608 192.168.86.1 192.168.86.61 ICMP 98 Time-
46 1.869171 192.168.86.61 192.168.86.1 DNS 85 Stand
47 1.873594 192.168.86.1 192.168.86.61 DNS 85 Stand
48 1.874016 192.168.86.61 128.119.245.12 UDP 70 64928

Frame 44: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface en0
Ethernet II, Src: Apple_98:d9:27 (78:4f:43:98:d9:27), Dst: Google_89:0e:c8 (3c:28:6d)
Internet Protocol Version 4, Src: 192.168.86.61, Dst: 128.119.245.12
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0xfda1 (64929)
```

(ans)

可以看到，地址是 192.168.86.61

2. 此 IPv4 数据报的头部中的生存时间（TTL）字段的值是什么？

```
...0 0000 0000 0000 = Fr
Time to Live: 1
```

3. 此 IPv4 数据报的头部中上层协议字段的值是什么？[注意：此处 Linux/MacOS 的答案与 Windows 不同]。

```
Time to Live: 1
Protocol: UDP (17)
```

UDP (17)

4. IP 头中包含多少字节？

```
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: N, 20 个字节。
```

5. IP 数据报的有效载荷有多少字节？请说明你是如何确定有效载荷字节数的。

```
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: N
Total Length: 56
Identification: 0xfda1 (64929)
```

如图，

有效载荷是 $56 - 20 = 36$ 字节。

6. 此 IP 数据报是否被分片？请说明您如何确定数据报是否被分片。
没有，证据是：

```
Identification: 0xfda1 (64929)
000. .... = Flags: 0x0
  0... .... = Reserved bit: Not set
  .0.. .... = Don't fragment: Not set
  ..0. .... = More fragments: Not set
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 1
```

7. 在由您的计算机发送的、目的地为的这一系列 UDP 分段中，IP 数据报中的哪些字段始终从一个数据报到下一个数据报发生变化
128.119.245.12，通过跟踪路由？为什么？

观察发现，Identification，Header Checksum 是一直在变化的：

```
Identification: 0xfda1 (64929)
000. .... = Flags: 0x0
0... .... = Reserved bit: Not set
.0... .... = Don't fragment: Not set
..0... .... = More fragments: Not set
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 1
Protocol: UDP (17)
Header Checksum: 0x2faa [validation disabled]

Total length: 50
Identification: 0xfda2 (64930)
000. .... = Flags: 0x0
0... .... = Reserved bit: Not set
.0... .... = Don't fragment: Not set
..0... .... = More fragments: Not set
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 1
Protocol: UDP (17)
Header Checksum: 0x2fa9 [validation disabled]
```

因为这是把 UDP 分段了，每一段用这个 identification 区分，这也就导致 header checksum 变化。

8. 此 IP 数据报序列（包含 UDP 段）中的哪些字段保持不变？为什么？

除了 Identification, Header Checksum, Time to live.

前面两个变已经说明了，TTL 主要是因为传输越远，时间花费越大，确保数据的有效性；剩下的字段因为是同一个 UDP 的分段，所以内容不会变化。

9. 描述您在计算机发送的 IP 数据报的标识字段中看到的值的模式。

Identification: 0xfdaa (64938)

Identification: 0xfdaB (64939)

按照十六进制表示（括号里面是十进制），并且每次都+1.

现在让我们看看中间路由器返回到您计算机的 ICMP 数据包，其中 TTL 值被减小到零（因此导致 ICMP 错误消息返回到您的计算机）。您可以使用的显示过滤器是“ip.dst==192.168.86.61 和 ICMP”，以仅显示这些数据包。

10. 从路由器返回的 IP 数据报中指定的上层协议是什么？[注意：Linux/macOS 的答案与 Windows 不同]。

Time to Live: 64
Protocol: ICMP (1)

11. 标识字段中的值（来自所有路由器的所有 ICMP 数据包序列）的行为是否与您对上面问题 9 的回答相似？

是的，变的和不变的都一样

12. 所有路由器的所有 ICMP 数据包的 TTL 字段值是否相似？

不是，有 64，63，251 等等，跟路由器相关。

Time to Live: 251

Time to Live: 63

13. 查找包含发送给的分段的第一部分的第一个 IP 数据报

128.119.245.12 通过计算机使用跟踪路由命令发送到 gaia.cs.umass.

edu，在你指定跟踪路由数据包长度应为 3000。（提示：这是脚注 2 中 ipwireshark-trace1-1.pcapng 跟踪文件中的第 179 个数据包。第 179、180 和 181 个数据包是由第一个 3000 字节的 UDP 段分片生成的三个 IP 数据报，该段数据包发送至 128.119.145.12) 该段数据包是否被分成了多个 IP 数据报？

179	12.788154	192.168.86.61	128.119.245.12	IPv4	1514	Fragmented IP protocol (prot
180	12.788155	192.168.86.61	128.119.245.12	IPv4	1514	Fragmented IP protocol (prot

可以看到已经被分成了很多段。

14. IP 头中的哪些信息表明该数据报已被分片？

见上图“fragmented”以及：

```
Identification: 0x1da2 (64930)
▼ 001. .... = Flags: 0x1, More fragments
  0. .... = Reserved bit: Not set
```

15. 此数据包的 IP 头部中的哪些信息表明这是第一个分片还是后续分片？

```
Identification: 0x1da2 (64930)
001. .... = Flags: 0x1, More fragments
  0... .... = Reserved bit: Not set
  .0.. .... = Don't fragment: Not set
  ..1. .... = More fragments: Set
...0 0000 0000 0000 = Fragment Offset: 0
```

从 offset=0 看出是第一个，其他的是后续。

16. 此 IP 数据报（包括头部和有效载荷）有多少字节？

```
Differentiated Services Field: 0x00000000 (0)
Total Length: 1500
```

17. 现在检查包含分片 UDP 段的第二个分片的数据报。IP 头部中的哪些信息表明这不是第一个数据报分片？

```
001. .... = Flags: 0x1, More fragments
...0 0000 1011 1001 = Fragment Offset: 1480
```

可见 offset 不是 0。

18. 在第一和第二片段之间，IP 头中的哪些字段发生了变化？

Offset 和 header checksum。

19. 现在找到包含原始 UDP 段的第三个片段的 IP 数据报。IP 头部中的哪些信息表明这是该段的最后一个片段？

```
...0. .... = More fragments: Not set
```

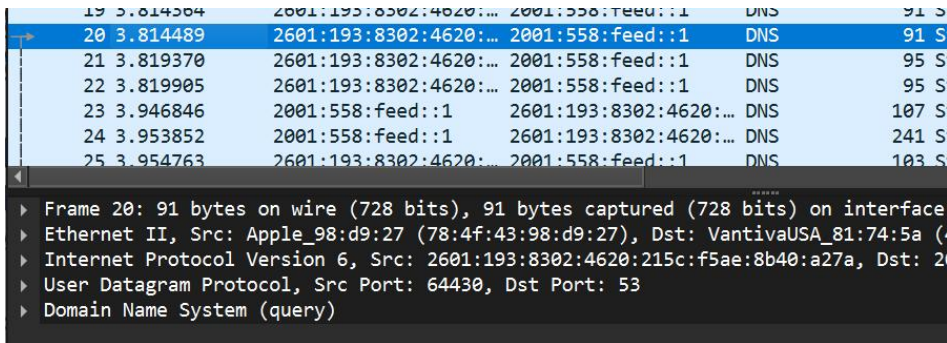
从这里看出，没有更多片段了。

第 3 部分：IPv6

在本节最后，我们将使用 Wireshark 快速查看 IPv6 数据报。你可能需要回顾教材中的第 4.3.4 节。由于互联网目前主要使用 IPv4 网络（见第 4.3.4 节），而且你的计算机或 ISP 可能未配置 IPv6，让我们来看一个已经捕获的包含一些 IPv6 数据包的跟踪记录。为了生成这个跟踪记录，我们的网页浏览器打开了 youtube.com 的主页。Youtube（和 Google）对 IPv6 的支持相当广泛。

打开文件 ip-wireshark-trace2-1.pcapng 位于脚注 2 中给出的 trace 的.zip 文件中。您的 Wireshark 显示应该类似于图 4。

20. 发出 DNS AAAA 请求的计算机的 IPv6 地址是什么？这是跟踪中第 20 个数据包的源地址。请以与 Wireshark 窗口 5 中显示的完全相同的格式给出该数据包的 IPv6 源地址。



The image shows a Wireshark packet capture. The packet list on the left shows packet 20 selected, with time 3.814489, source 2601:193:8302:4620::..., and destination 2001:558:feed::1. The details pane on the right shows the following information for packet 20:

Frame 20: 91 bytes on wire (728 bits), 91 bytes captured (728 bits) on interface
Ethernet II, Src: Apple_98:d9:27 (78:4f:43:98:d9:27), Dst: VantivaUSA_81:74:5a (44:1c:12:81:74:5a)
Internet Protocol Version 6, Src: 2601:193:8302:4620:215c:f5ae:8b40:a27a, Dst: 2001:558:feed::1
User Datagram Protocol, Src Port: 64430, Dst Port: 53
Domain Name System (query)

地址：2601:193:8302:4620:215c:f5ae:8b40:a27a

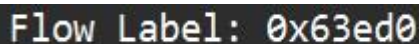
21. 此数据包的 IPv6 目标地址是什么？请以与 Wireshark 窗口中显示完全相同的格式给出此 IPv6 地址。



The image shows a close-up of the IPv6 destination address in the packet details pane: Dst: 2001:558:feed::1

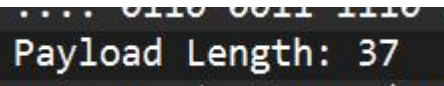
目标地址：

22. 此数据包的流标签值是什么？



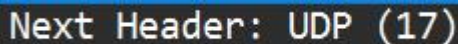
The image shows the flow label value in the packet details pane: Flow Label: 0x63ed0

23. 该数据包中携带了多少有效载荷数据？



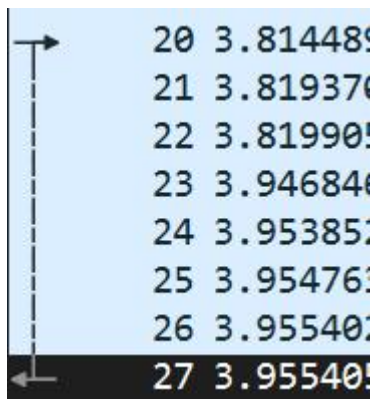
The image shows the payload length in the packet details pane: Payload Length: 37

24. 此数据包的有效载荷将在目的地交付到的上层协议是什么？



The image shows the next header value in the packet details pane: Next Header: UDP (17)

最后，在此跟踪的第 20 个数据包中找到 IPv6 DNS 对 IPv6 DNS AAAA 请求的响应。此 DNS 响应包含 youtube. com 的 IPv6 地址。

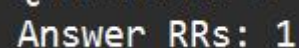


The image shows a Wireshark packet capture list. Packet 27 is selected, with time 3.955407, source 2601:193:8302:4620::..., and destination 2001:558:feed::1. The packet list shows packets 20 through 27.

20	3.814489	2601:193:8302:4620::...	2001:558:feed::1	DNS	91 S
21	3.819370	2601:193:8302:4620::...	2001:558:feed::1	DNS	95 S
22	3.819905	2601:193:8302:4620::...	2001:558:feed::1	DNS	95 S
23	3.946846	2001:558:feed::1	2601:193:8302:4620::...	DNS	107 S
24	3.953852	2001:558:feed::1	2601:193:8302:4620::...	DNS	241 S
25	3.954763	2601:193:8302:4620::...	2001:558:feed::1	DNS	103 S
26	3.955407	2601:193:8302:4620::...	2001:558:feed::1	DNS	103 S
27	3.955407	2601:193:8302:4620::...	2001:558:feed::1	DNS	103 S

对应的 27：

25. 此 AAAA 请求的响应中返回多少个 IPv6 地址？



The image shows the number of answer RRs in the packet details pane: Answer RRs: 1

26. DNS 返回的 youtube. com 的第一个 IPv6 地址是什么（在 ip-wiresharktrace2-1.pcapng 跟踪文件中，这也是该地址数值最小的）？请以与 Wireshark

窗口中显示完全相同的简写形式给出此 IPv6 地址。

```
119 Standard query response 0x920d AAAA youtube.com AAAA 2607:f8b0:4006:815::200e  
2607:f8b0:4006:815::200e
```

结论分析与体会：

题目和回答见上。通过此次实验，我对于 IP 的理解更进一步。