南開大學
Nankai University

配置 Web 服务

及分析 HTTP 交互过程

姓名：李娅琦

学号：2213603

专业：计算机科学与技术

2024 年 10 月 29 日

目录

1 实验要求

1. 搭建 Web 服务器并制作简单的 Web 页面，包含文本信息（至少包含专业、学号、姓名）、自己的 LOGO、自我介绍的音频。
2. 通过浏览器访问 Web 服务器，获取自己编写的 HTML 文档，并显示 Web 页面。
3. 使用 Wireshark 捕获与 Web 服务器的交互过程，设置过滤器使 Wireshark 仅显示 HTTP 报文，并详细说明 HTTP 交互过程。

2 实验环境

2.1 服务器搭建与启动

本次实验个人选择了在 windows 本机系统之下，搭建 Apache 服务器。

2.1.1 搭建

在官网下载 Apache 压缩包并安装。通过设置 http.conf 文件中的 ip 和端口号指定了服务器的 ip 为：127.0.0.1，端口号为：8089。

```
#
ServerName 127.0.0.1:8089
#
```

```
#Listen 12.34.56.78:80
Listen 8089
```

2.1.2 入栈规则

由于更改了 Apache 服务器的默认端口 (80)，因此需要在 Windows 防火墙中添加新的入站规则，以允许流量通过这个新端口 8089。

2.1.3 启动

可通过在 bin 文件目录下在终端输入：**httpd -t start**

或直接双击 bin 目录下的 **ApacheMonitor.exe** 以开启 apache 服务。

2.2 客户端设置与访问

因为在本机的浏览器中打开，因此客户端 ip 同服务器为：127.0.0.1，端口号由浏览器窗口随机生成 (本次为：65009)。

将 html 文件等放于 htdos 目录下并命名为 index 后，使用本机的浏览器访问http://127.0.0.1:8089即可看到 html 界面。

3 web 页面

利用 html 设计 Web 页面，包含文本信息（至少包含专业、学号、姓名）、自己的 LOGO、自我介绍的音频。代码参考给出的文件，效果如下 (可以调节声音以及播放倍速)：

4 wireshark 捕获

安装 wireshark 并进行监视对应网卡（**Adapter for loopback traffic capture**）即可看到捕获的包的相关信息。

4.1 抓包过滤

利用：**tcp.port == 8089&& tcp.port == 65009** 对抓包结果进行过滤。

4.2 捕获结果

捕获结果如下图所示：

tcp.port == 8089&&tcp.port==65009

No.	Time	Source	Destination	Protocol	Lengt	Info
131	208.681738	127.0.0.1	127.0.0.1	TCP	56	65009 → 8089 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
132	208.681771	127.0.0.1	127.0.0.1	TCP	56	8089 → 65009 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
133	208.681797	127.0.0.1	127.0.0.1	TCP	44	65009 → 8089 [ACK] Seq=1 Ack=1 Win=327424 Len=0
134	208.689706	127.0.0.1	127.0.0.1	HTTP	842	GET / HTTP/1.1
135	208.689737	127.0.0.1	127.0.0.1	TCP	44	8089 → 65009 [ACK] Seq=1 Ack=799 Win=2160384 Len=0
136	208.690090	127.0.0.1	127.0.0.1	HTTP	3938	HTTP/1.1 200 OK (text/html)
137	208.690120	127.0.0.1	127.0.0.1	TCP	44	65009 → 8089 [ACK] Seq=799 Ack=3895 Win=323328 Len=0
141	208.737976	127.0.0.1	127.0.0.1	HTTP	645	GET /%E5%89%8D%E5%A5%8F.mp3 HTTP/1.1
142	208.738032	127.0.0.1	127.0.0.1	TCP	44	8089 → 65009 [ACK] Seq=3895 Ack=1400 Win=2159872 Len=0
143	208.739553	127.0.0.1	127.0.0.1	TCP	65539	8089 → 65009 [ACK] Seq=3895 Ack=1400 Win=2159872 Len=65495 [TCP PDU reassembled in 150]
144	208.739574	127.0.0.1	127.0.0.1	TCP	65539	8089 → 65009 [ACK] Seq=69390 Ack=1400 Win=2159872 Len=65495 [TCP PDU reassembled in 150]
145	208.739607	127.0.0.1	127.0.0.1	TCP	65539	8089 → 65009 [ACK] Seq=134885 Ack=1400 Win=2159872 Len=65495 [TCP PDU reassembled in 150]
146	208.739616	127.0.0.1	127.0.0.1	TCP	65539	8089 → 65009 [ACK] Seq=200380 Ack=1400 Win=2159872 Len=65495 [TCP PDU reassembled in 150]
147	208.740021	127.0.0.1	127.0.0.1	TCP	44	65009 → 8089 [ACK] Seq=1400 Ack=265875 Win=327424 Len=0
148	208.740212	127.0.0.1	127.0.0.1	TCP	65539	8089 → 65009 [ACK] Seq=265875 Ack=1400 Win=2159872 Len=65495 [TCP PDU reassembled in 150]
149	208.740220	127.0.0.1	127.0.0.1	TCP	65539	8089 → 65009 [ACK] Seq=331370 Ack=1400 Win=2159872 Len=65495 [TCP PDU reassembled in 150]
150	208.740228	127.0.0.1	127.0.0.1	MPEG-2	1654	Audio Layer 3, 112 kb/s, 16 kHz
151	208.740318	127.0.0.1	127.0.0.1	TCP	44	65009 → 8089 [ACK] Seq=1400 Ack=398475 Win=194816 Len=0
152	208.741475	127.0.0.1	127.0.0.1	TCP	44	[TCP Window Update] 65009 → 8089 [ACK] Seq=1400 Ack=398475 Win=327424 Len=0
154	213.745051	127.0.0.1	127.0.0.1	TCP	44	8089 → 65009 [FIN, ACK] Seq=398475 Ack=1400 Win=2159872 Len=0
155	213.745087	127.0.0.1	127.0.0.1	TCP	44	65009 → 8089 [ACK] Seq=1400 Ack=398476 Win=327424 Len=0
209	236.028674	127.0.0.1	127.0.0.1	TCP	44	65009 → 8089 [FIN, ACK] Seq=1400 Ack=398476 Win=2161152 Len=0
210	236.028716	127.0.0.1	127.0.0.1	TCP	44	8089 → 65009 [ACK] Seq=398476 Ack=1401 Win=2159872 Len=0

可以看出，其中包括 TCP 连接的三次握手和四次挥手，以及 HTTP 协议。下面对捕获结果进行详细解释。

5 封包信息

首先可以看到每个包均具有以下几种信息:

- Frame: 物理层的数据帧
- Internet Protocol Version 4: 互联网层 IP 包头部信息
- Transmission Control Protocol: 传输层数据段头部信息
- Hypertext Transfer Protocol: 应用层信息

```
> Frame 7: 292 bytes on wire (2336 bits), 292 bytes captured (2336 bits) on interfa
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 8089, Dst Port: 65164, Seq: 1, Ack: 799,
> Hypertext Transfer Protocol
```

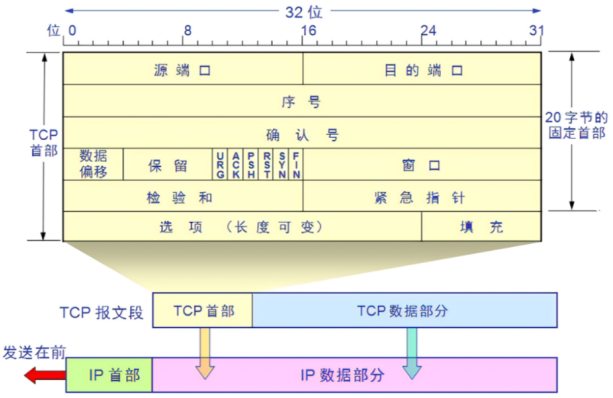
但是实际上，还有 Ethernet II 所对应的数据链路层。但由于在本实验中由于直接访问本机的服务器，因此不存在。

6 三次握手

6.1 理论基础

6.1.1 TCP 报文

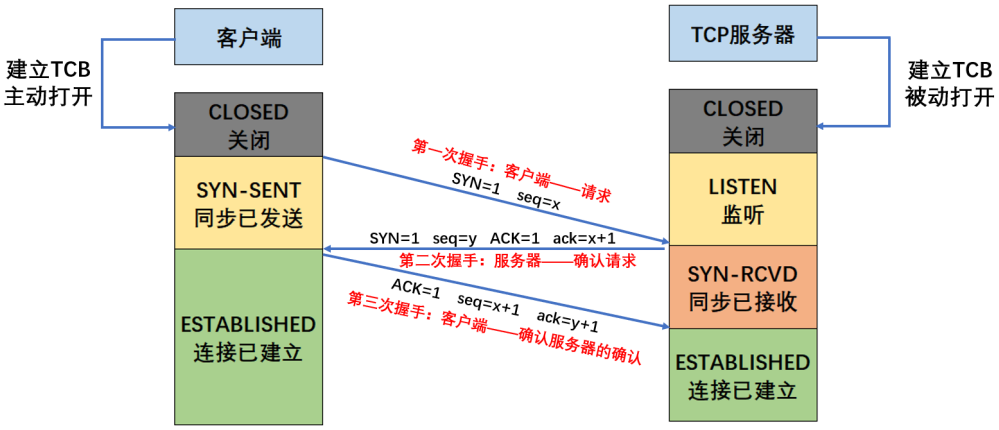
TCP 报文，是 TCP 协议中传输的数据单元。一个 TCP 段由首部和数据两部分组成。首部包含了一系列的字段，用于实现 TCP 协议的各种功能，如序列号、确认号、窗口大小等。数据部分则是应用层传下来的数据。



6.1.2 三次握手详解

为了保证客户端和服务器端的可靠连接，TCP 建立连接时必须要进行三次会话，也叫 TCP 三次握手。

这个过程涉及到三个步骤，如下图:



可以知道的是在其中用到的标志位和序列号有：

- TCB: 传输控制块，打开后服务器/客户端进入监听 (LISTEN) 状态
- SYN: 置 1 表示发起一个新连接
- ACK: 置 1 表示确认接收到消息。TCP 规定，连接建立后所有报文传输 ACK 置 1
- seq: 报文初始序列号代表发送的第一个字节的序号
- ack: 报文确认序号代表希望收到的下个数据第一个字节序号

从图中可以看出：

- **第一次握手：**

客户端发送一个 TCP 报文到服务器，这个报文的 SYN 标志位被设置为 1，客户端会随机选择一个初始序列号 seq=x，并进入 SYN-SENT 状态。

- **第二次握手：**

服务器收到这个 SYN 报文后，如果同意建立连接，则会发送一个 SYN 报文作为响应，这个报文的 SYN 和 ACK 标志位都被设置为 1，确认号 ack=x+1，同时服务器也会选择一个初始序列号 seq=y，并进入 SYN-RCVD 状态。

- **第三次握手：**

客户端收到服务器的 SYN+ACK 报文后，会发送一个 ACK 报文作为响应，这个报文的 ACK 标志位被设置为 1，确认号 ack=y+1，序列号 seq=x+1。此时，客户端和服务器都进入 ESTABLISHED 状态，表示连接已经建立。

6.1.3 三次握手原因

进行三次握手的主要原因是为了防止已经失效的连接请求报文突然又传送到了服务器，从而产生错误。每次握手都是为了确定信息：

- 第一次握手：证明客户端的发送能力正常
- 第二次握手：证明服务器端的接收能力、发送能力正常
- 第三次握手：证明客户端的接收能力正常

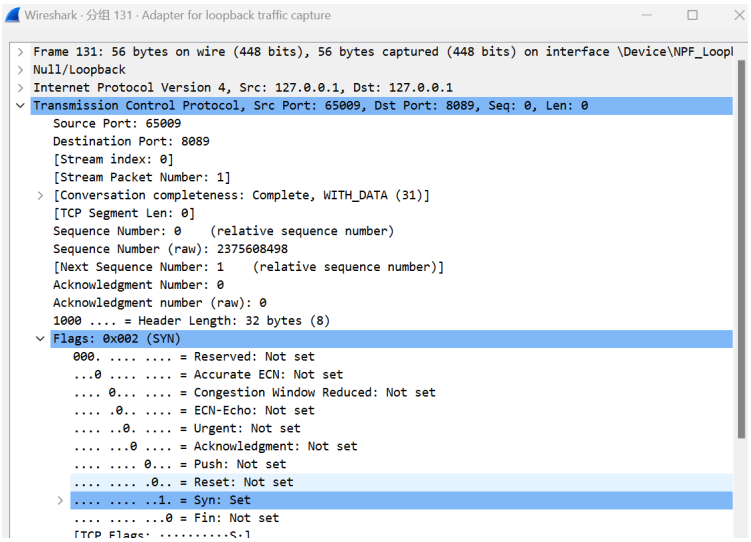
这样的话，通过**确认和同步双方的序列号和确认号**确保了连接的可靠性。如果只采用两次握手，**无法防止历史连接的建立，会造成双方资源的浪费，也无法可靠的同步双方序列号**。

6.2 实验验证

捕获的前三个包即为与三次握手相关的包，如下图：

```
131 208.681738   127.0.0.1       127.0.0.1       TCP        56 65009 → 8089 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
132 208.681771   127.0.0.1       127.0.0.1       TCP        56 8089 → 65009 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
133 208.681797   127.0.0.1       127.0.0.1       TCP        44 65009 → 8089 [ACK] Seq=1 Ack=1 Win=327424 Len=0
```

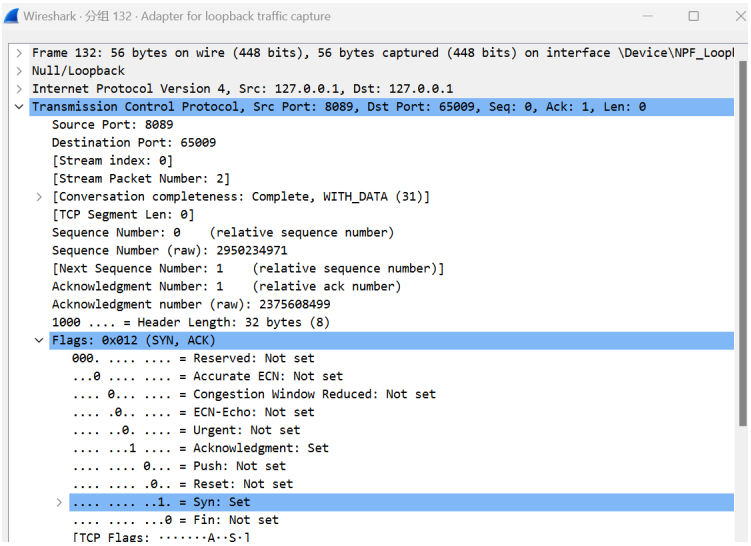
第一次握手：由客户端向服务器发送 TCP 连接请求。



可以看到:

- 源端口为 65009，目的端口为 8089(代表客户端发向服务器)
- 客户端随机生成初始序列号:Seq=2375608498(绝对序列号)
- SYN 被置位为 1，ACK 无置位

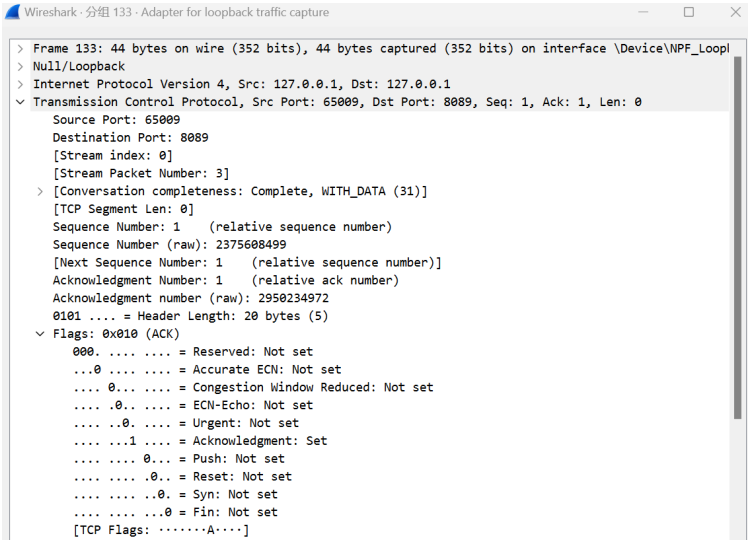
第二次握手：服务器向客户端发送确认包



可以看到:

- 源端口为 8089，目的端口为 65009(代表服务器发向客户端)
- 服务器随机生成初始序列号:Seq=2950234971(绝对序列号)
- SYN 和 ACK 都被置位为 1，确认号 (ack 字段) 为第一次握手的 seq+1，即 2375608499

第三次握手：客户端向服务器发送确认包



可以看到:

- 源端口为 65009，目的端口为 8089(代表客户端发向服务器)
- SYN 和 ACK 都被置位为 1，确认号 (ack 字段) 为第一次握手的 seq+1，即 2950234972
- 确认号 (seq 字段) 为第一次握手的 seq+1，即 2375608499

至此，客户端和服务器都进入 ESTABLISHED 状态，连接成功建立。

7 http 请求

本次实验采用了 http 而不是 https，同时使用 http/1.1 协议。

7.1 理论基础

7.1.1 请求格式

Http 请求分为以下三部分：

- 请求行 (包括请求方法,URL, 版本号)
- 请求头 (包含元数据，如 Host、User-Agent、Accept 等)
- 请求体

7.1.2 响应格式

Http 响应分为以下三部分：

- 响应行 (包括版本号, 状态码及其解释)
- 响应头 (包含描述响应的元数据)
- 响应体

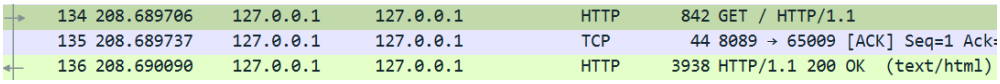
7.1.3 状态码

状态码表示这次请求所获得的响应是一个什么样的状态，表示这次请求是成功，还是失败，还是其他。状态码的种类非常多，常用的有如下：

- 200：请求成功
- 302(临时重定向)：表示请求资源的 URL 被暂时更改
- 404：服务器找不到请求的资源
- 504：无法在规定的时间内获得想要的响应

7.2 实验验证

可以看到其中的一个请求与它的响应如下图：



下面展开请求与响应进行详细的查看：

7.2.1 请求

在 Hypertext Transfer Protocol 中展开可以发现请求的具体内容：

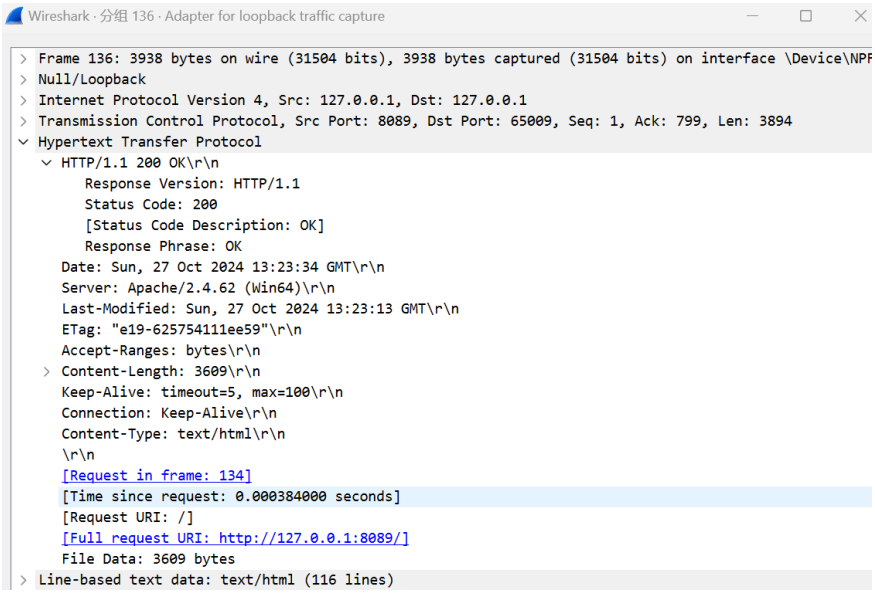
(1) 请求行

- 请求方法：GET
- 请求 URL：/
- 请求协议：http/1.1

(2) 请求头

- Host: 请求的服务器地址
- Connection：keep-Alive，表示为长连接可处理多个请求
- User-Agent: 用户代理，表示客户端使用的浏览器信息
- Accept：客户端能够接收的内容类型

7.2.2 响应



可以看到响应的具体内容：

(1) 响应行

- 响应协议：HTTP/1.1
- 状态码：200(请求成功)
- 响应描述：OK

(2) 响应头

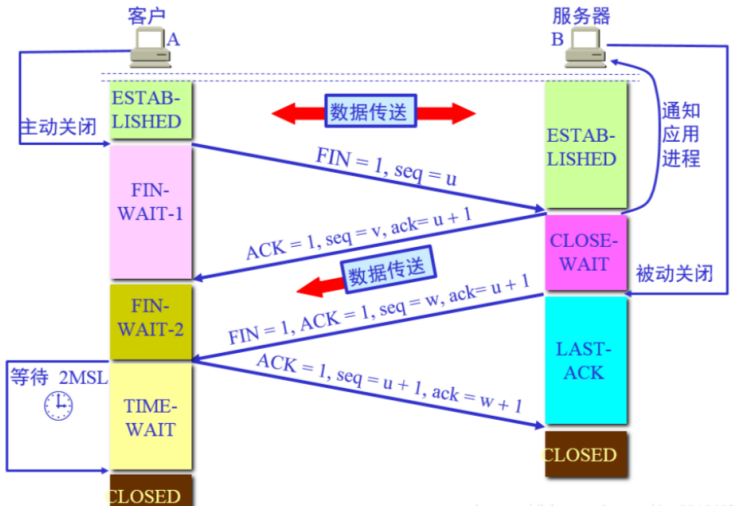
- Date：响应的时间
- Server：服务器信息
- Content-Type：内容类型
- Content-Length：响应体字节数

8 四次挥手

8.1 理论基础

8.1.1 四次挥手详解

释放连接的过程一般需要四次挥手，如下图：



以客户端主动释放连接为例：

- **第一次挥手：**

客户端向服务端发送连接释放请求的 FIN 报文。报文中会指定一个序列号 seq=u，并停止再发送数据，但依然能够接收数据。此时客户端处于 FIN-WAIT-1 状态，等待服务端确认。

- **第二次挥手：**

服务器接收到客户端请求报文后进入 CLOSE-WAIT 阶段并返回一段 TCP 报文。随后服务器开始准备释放服务器到客户端方向上的连接，客户端收到后进入 FIN-WAIT-2 阶段。

- **第三次挥手：**

服务器在发出 ACK 确认报文后，将遗留的待传数据传送给客户端完成后即经过 CLOSE-WAIT 阶段，再次向客户端发出一段 TCP 报文。随后进入 LAST-ACK 阶段，并且停止向客户端发送数据。

- **第四次挥手：**

客户端收到服务器的连接释放报文后，一样发送一个 ACK 报文作为应答，此时客户端处于 TIME-WAIT 状态，并在这个状态等待 2MSL。

服务端收到从客户端发出的 TCP 报文后进入 CLOSED 阶段。客户端等待完 2MSL 之后，结束 TIME-WAIT 阶段，进入 CLOSED 阶段。

8.1.2 等待 2MSL

客户端在 TIME-WAIT 阶段要等 2MSL，原因如下：

(1) 保证客户端发送的最后一个 ACK 报文段能够到达服务器端，确保服务端能正常进入 CLOSED 状态。

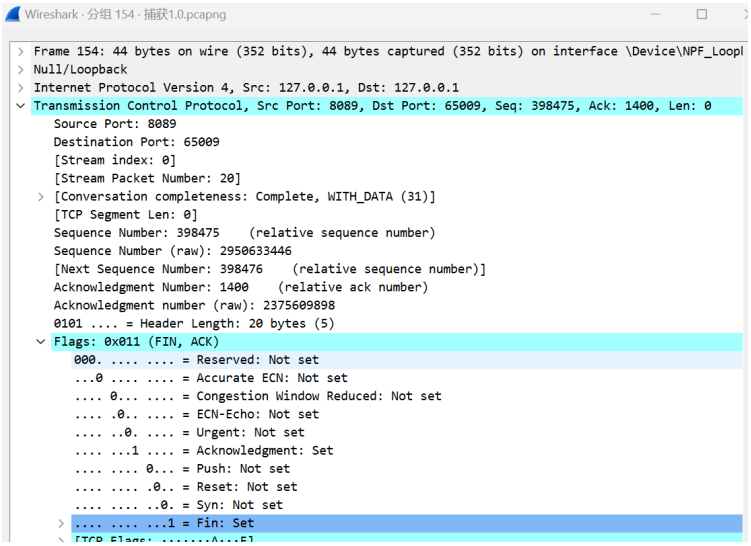
(2) 避免新旧连接混淆。由于网络滞留，客户端可能发送了多次请求建立连接的请求。

8.2 实验验证

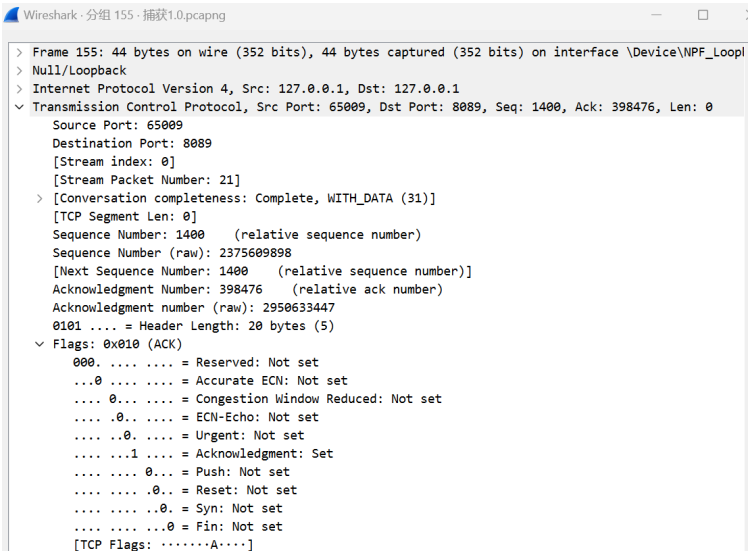
后四个包即为与四次挥手相关的包，如下图：

```
152 208.741475   127.0.0.1   127.0.0.1   TCP   44 [TCP Window Update] 65009 → 8089 [ACK] Seq=1400 Ack=398475 Win
154 213.745051   127.0.0.1   127.0.0.1   TCP   44 8089 → 65009 [FIN, ACK] Seq=398475 Ack=1400 Win=2159872 Len=0
155 213.745087   127.0.0.1   127.0.0.1   TCP   44 65009 → 8089 [ACK] Seq=1400 Ack=398476 Win=327424 Len=0
209 236.028674   127.0.0.1   127.0.0.1   TCP   44 65009 → 8089 [FIN, ACK] Seq=1400 Ack=398476 Win=2161152 Len=0
210 236.028716   127.0.0.1   127.0.0.1   TCP   44 8089 → 65009 [ACK] Seq=398476 Ack=1401 Win=2159872 Len=0
```

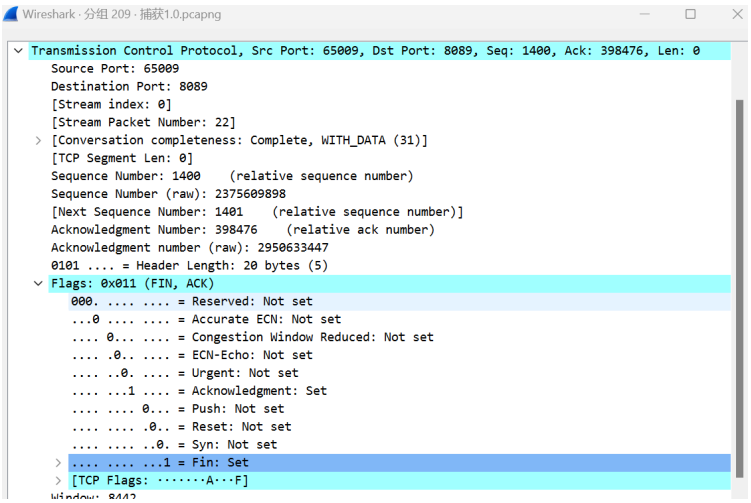
第一次挥手：服务器发向客户端



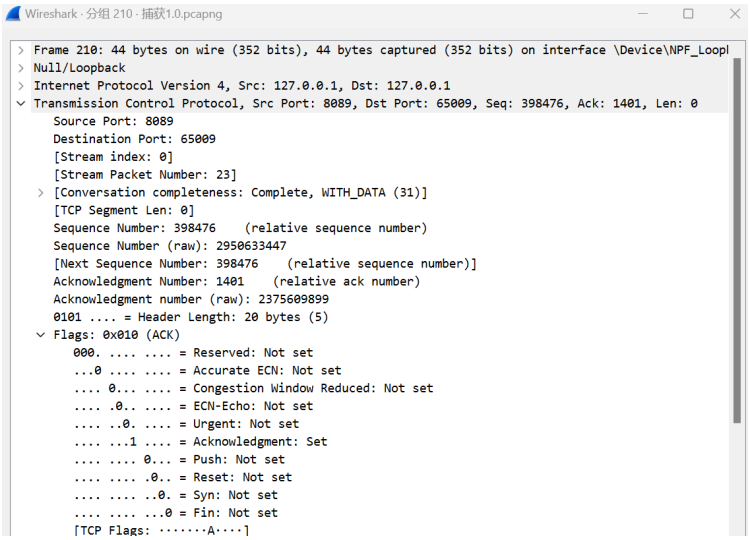
第二次挥手：客户端发向服务器



第三次挥手：客户端发向服务器



第四次挥手：服务器发向客户端



9 遇到问题

9.1 服务器先挥手

一般来说，四次挥手由客户端先发送断开连接的请求，但本次实验，由服务器主动断开连接。可能的原因有：

- 服务器完成了对客户端请求的处理，并且没有更多的数据需要发送给客户端
- 客户端在一定时间内没有发送任何数据，服务器因为超时而主动关闭连接
- 服务器检测到网络不稳定，主动关闭连接以避免数据传输问题

个人理解，可能是由于在 apache 服务器中，会每隔一段时间主动向客户端发送断开连接的请求，如果此时客户端没有保持 keep alive，服务器会主动断开连接以节约资源。

9.2 客户端端口号不一

在网络通信中，端口号用于区分不同的服务或进程。当在网页中打开客户端时，每次打开的端口都不一样，这可能因为**动态端口分配**。

许多操作系统在启动网络服务时，会为每个新的连接动态分配一个临时端口号。这些端口号通常是从 1024 到 65535 之间的未被占用的端口，这样可以提高安全性。

9.3 TCP 初始序列号不一

每次建立 TCP 连接时，初始化的序列号也不一样，可能是因为：

- 防止历史报文被下一个相同四元组的连接接收
- 防止黑客伪造的相同序列号的 TCP 报文被对方接收