



南開大學
Nankai University

计算机网络课程实验报告

WireShark 抓包分析



学 院：网络安全学院

专 业：信息安全

学 号：2111252

姓 名：李佳豪

班 级：信安一班

1 实验内容

- 通过学习基本 web 页面的书写，自行建立 web 服务器
- 在理论课对网络协议方面的知识学习后，使用 wireshark 抓取流量

2 Web 服务器配置

2.1 Html 页面设计

本次实验页面不要求设计复杂，仅仅需要将个人 Logo、信息以及一个音频大方展示即可，这里我将所有展示内容放入一个 div 块内，并设计块边框以及字体。

效果如下：

7:27



姓名: 李佳豪

邮箱:
2111252@mail.nankai.edu.cn

电话: [13191110713](tel:13191110713)



源码如下：

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
6 <title>WebServer</title>
7 <style>
8     body {
9         font-family: Arial, sans-serif;
10        margin: 50px;
11    }
12    .profile {
13        /*border: 1px solid #e0e0e0;*/
14        background-image: linear-gradient(to right, #f3f7f7, #f4dada),
15            linear-gradient(90deg, #8F41E9, #578AEF);
16        padding: 20px;
17        border-radius: 5px;
18        box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
19        margin-bottom: 20px;
20    }
21
22 </style>
23 </head>
24 <body>
25     <div class="profile">
26         
28         <div class="profile-info">
29             <p><strong>姓名:</strong> 李佳豪</p>
30             <p><strong>邮箱:</strong> 2111252@mail.nankai.edu.cn</p>
31             <p><strong>电话:</strong> 13191110713</p>
32         </div>
33         <!-- 音频播放器 -->
34         <audio controls>
35             <source src="my_intro.mp3" type="audio/mpeg">
36             您的浏览器不支持音频元素。
37         </audio>
38     </div>
```

2.2 Server 设置

这里我使用 WampServer 在本机设置 Apache 服务器，WampServer 是一个 Windows 平台上的开发环境，它集成 Web 应用程序与 Apache2、PHP 和 MySQL 数据库，而内置的 Apache 服务器则使用起来也很方便。

将软件环境下载配置好，打开其 Apache 的 httpd-vhost.conf，设置权限 Require all granted；最

后，将页面移至 WampServer 目录 www 下面，在虚拟机或者局域网内访问、抓包。

3 WireShark 抓包

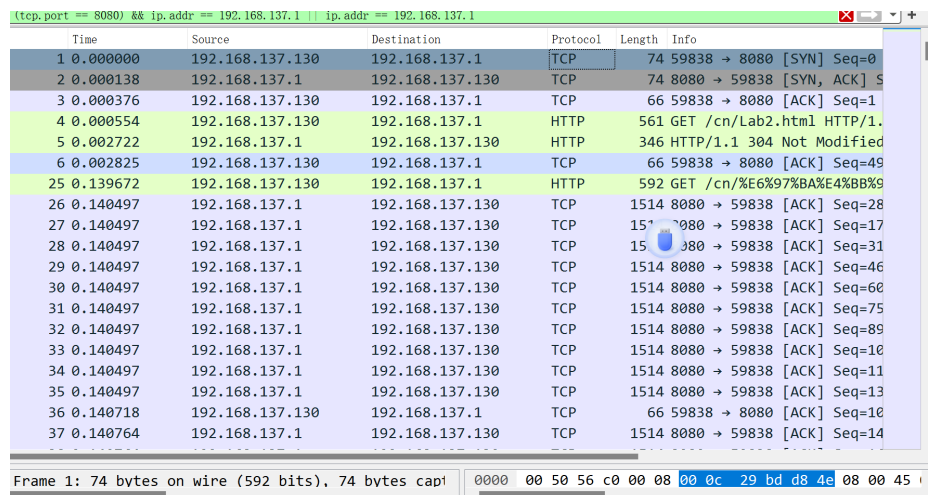
3.1 WireShark 设置

抓包前，设置抓取的网卡以及过滤器，这里我使用 Net 模式下的虚拟机访问页面，网卡选取对应虚拟机和主机转发数据的网卡，使用同时为了观察 Tcp 的三次握手、Http 协议传输数据和四次挥手，我需要仅仅协议为 tcp 且传输的源地址和目的地址是本机和虚拟机，发出端口为 8080(WampServer 开放的端口) 的包，因此过滤器设置为：

```
tcp && ((ip.src==192.168.137.130 && ip.dst==192.168.137.1) || (ip.dst==192.168.137.130 && ip.src==192.168.137.1))
```

同时还需要过滤长度 (tcp.len)、过滤包的 flags(tcp.flags.fin) 内字段等。

抓取界面如下，之后开始分析各种包的内容。



Time	Source	Destination	Protocol	Length	Info
1 0.000000	192.168.137.130	192.168.137.1	TCP	74	59838 → 8080 [SYN] Seq=0
2 0.000138	192.168.137.1	192.168.137.130	TCP	74	8080 → 59838 [SYN, ACK] S
3 0.000376	192.168.137.130	192.168.137.1	TCP	66	59838 → 8080 [ACK] Seq=1
4 0.000554	192.168.137.130	192.168.137.1	HTTP	561	GET /cn/Lab2.html HTTP/1.
5 0.002722	192.168.137.1	192.168.137.130	HTTP	346	HTTP/1.1 304 Not Modified
6 0.002825	192.168.137.130	192.168.137.1	TCP	66	59838 → 8080 [ACK] Seq=49
25 0.139672	192.168.137.130	192.168.137.1	HTTP	592	GET /cn/%E6%97%BA%E4%B8%9
26 0.140497	192.168.137.1	192.168.137.130	TCP	1514	8080 → 59838 [ACK] Seq=28
27 0.140497	192.168.137.1	192.168.137.130	TCP	1514	8080 → 59838 [ACK] Seq=17
28 0.140497	192.168.137.1	192.168.137.130	TCP	1514	8080 → 59838 [ACK] Seq=31
29 0.140497	192.168.137.1	192.168.137.130	TCP	1514	8080 → 59838 [ACK] Seq=46
30 0.140497	192.168.137.1	192.168.137.130	TCP	1514	8080 → 59838 [ACK] Seq=60
31 0.140497	192.168.137.1	192.168.137.130	TCP	1514	8080 → 59838 [ACK] Seq=75
32 0.140497	192.168.137.1	192.168.137.130	TCP	1514	8080 → 59838 [ACK] Seq=89
33 0.140497	192.168.137.1	192.168.137.130	TCP	1514	8080 → 59838 [ACK] Seq=10
34 0.140497	192.168.137.1	192.168.137.130	TCP	1514	8080 → 59838 [ACK] Seq=11
35 0.140497	192.168.137.1	192.168.137.130	TCP	1514	8080 → 59838 [ACK] Seq=13
36 0.140718	192.168.137.130	192.168.137.1	TCP	66	59838 → 8080 [ACK] Seq=10
37 0.140764	192.168.137.1	192.168.137.130	TCP	1514	8080 → 59838 [ACK] Seq=14

Frame 1: 74 bytes on wire (592 bits), 74 bytes captured on interface 0000 00 50 56 c0 00 08 00 0c 29 bd d8 4e 08 00 45

3.2 Tcp 三次握手

抓取包的前三个连续 Protocol 为 Tcp 的 entry，即为三次握手的内容。三次握手简单说：

第一次握手：客户端发送连接请求的数据包给服务器，告诉服务器我要建立连接，并且我可以发送数据。第二次握手：服务器收到请求后，确认自己可以接收到来自客户端的数据，同时告诉客户端我也可以发送数据。第三次握手：客户端告诉服务器，我可以接收到你的数据，这样双方都确认了各自既可以发送数据，也可以接收数据。

具体细节通过下面三个包逐个分析：

首先查一下 tcp 协议格式信息，

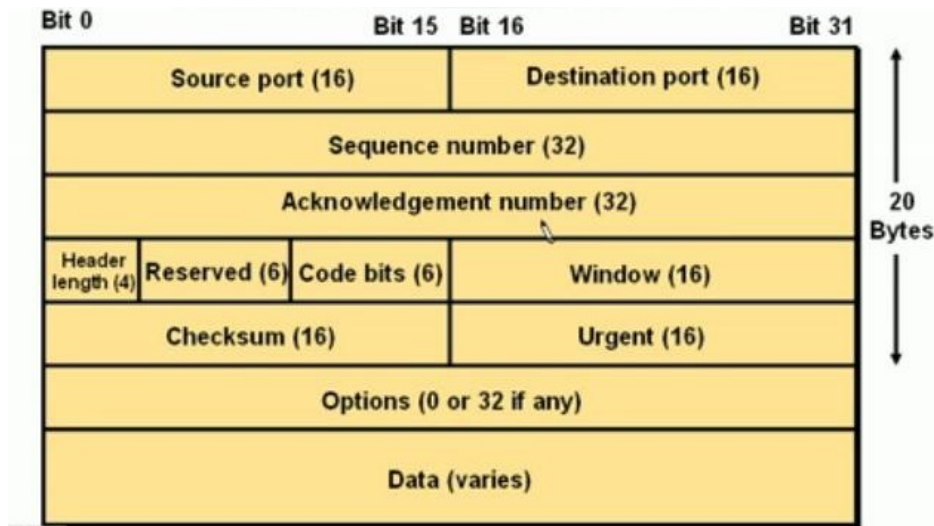


图 3.1: tcp 格式

三个包在 WireShark 已经自低向上解析过,三次握手(挥手)过程中关键的几个位置是 TCP Flags 的 Syn、Fin(在挥手时用)、Ack 位,Sequence Number,Acknowledgment Number,他们在传输层的信息如下:

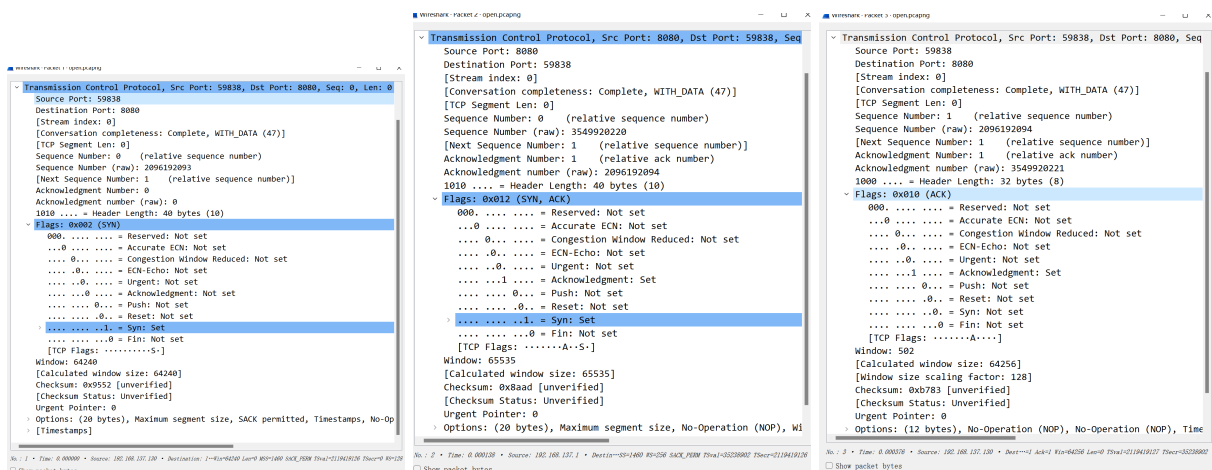


图 3.2: 三次握手内容对比

- 第一次握手,由上图,**SYN 位置为 1**,Sequence Number 为 2096192093;然后,客户端进入 SYN_SEND 状态,等待服务器的确认。
- 第二次握手,服务器收到 SYN 报文段。**SYN 和 Ack 位置 1**,服务器收到客户端的 SYN 报文段,需要对这个 SYN 报文段进行确认,设置 Acknowledgment Number 为 2096192094(**Sequence Number+1**);同时,发送 SYN 请求信息,将 SYN 位置为 1,Sequence Number 为 3549920220,此时服务器进入 SYN_RECV 状态。
- 第三次握手:客户端收到服务器的 SYN+ACK 报文段。**Ack 位置 1**,然后将 Acknowledgment Number 设置为 3549920220(Server 发来的 **Sequence Number+1**),Sequence 为 2096192094 向服务器发送 ACK 报文段,这个报文段发送完毕以后,客户端和服务端都进入 ESTABLISHED 状态,完成 TCP 三次握手。

- **值得注意的是**, 发现 client 发送第一次握手和 Server 第二次握手的 Options 20 字节, 但第三次握手并且后面所有的 tcp 包都是 12 字节。

具体内容而言, 第二次握手之后, **options 内只使用 10 个字节**记录时间戳相关内容, 而前两次还有 Maximum segment size、SACK permitted、Window scale。

经过查阅, 这三个通常都会出现 SYN 包 (可以观察 FLAGS 的 SYN 字段, 第二次挥手后的所有包的 SYN 位都置 0 了) MSS 用来指明 **TCP 通讯的最大段大小, 这里为 1460**。ws 用于扩大 TCP 的接收窗口大小, 允许更大的数据量被传输。SACK-Permitted, 表明端点支持 SACK 选项, SACK 本身用于指明已经成功接收的非连续数据段。

```

    Urgent pointer: 0
  ▾ Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps,
    > TCP Option - Maximum segment size: 1460 bytes
    > TCP Option - SACK permitted
    > TCP Option - Timestamps
    > TCP Option - No-Operation (NOP)
  ▾ TCP Option - Window scale: 7 (multiply by 128)
    Kind: Window Scale (3)
    Length: 3
    Shift count: 7
    [Multiplier: 128]

```

图 3.3: SYN 包-20 BYTES

3.3 Http 协议

抓取的第四、五个包便是 Http 协议包, 同样, 先学习 Http 报文格式

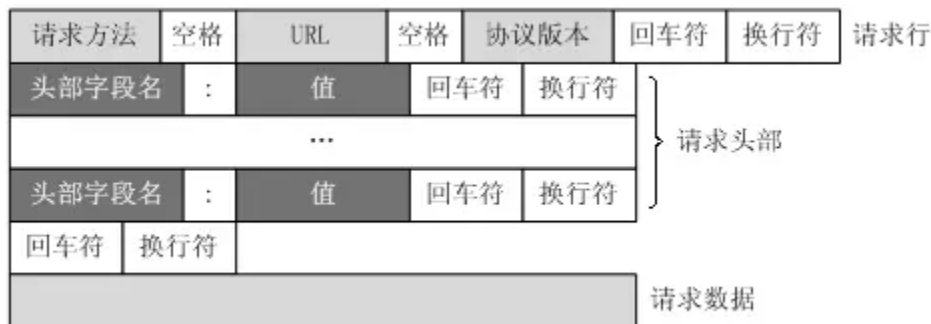


图 3.4: Http 格式

- 第一个 Http 包是 client 的 request, 可以看到除了正常的请求行、请求头和一个空行外, 最后有 WireShark 的提示, 说回复在第五个 frame, 第二个请求在 25frame, 但 request 请求大小并没有超过 mss 定义的值 (SYN 包定义的 1460), 因此这里我猜测跟 tcp 的滑动窗口机制有关或者其他原因, 当然后面会纠正。

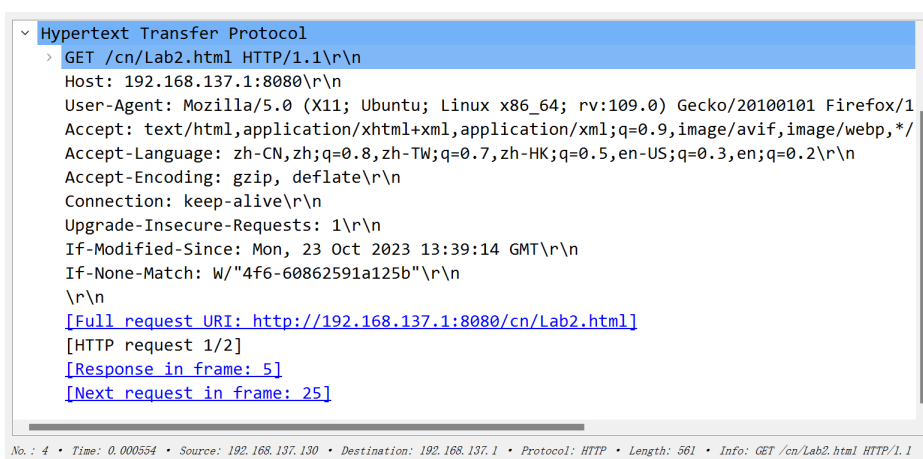


图 3.5: 第一个

- 第二个 Http 是 Server 的 reponse, 看到响应 304, 意思 Not Modified, 因为我之前测试网页的时候 client 浏览器有了缓存, 该缓存在 Server 判断为可以继续使用, 因此不发送页面源码。

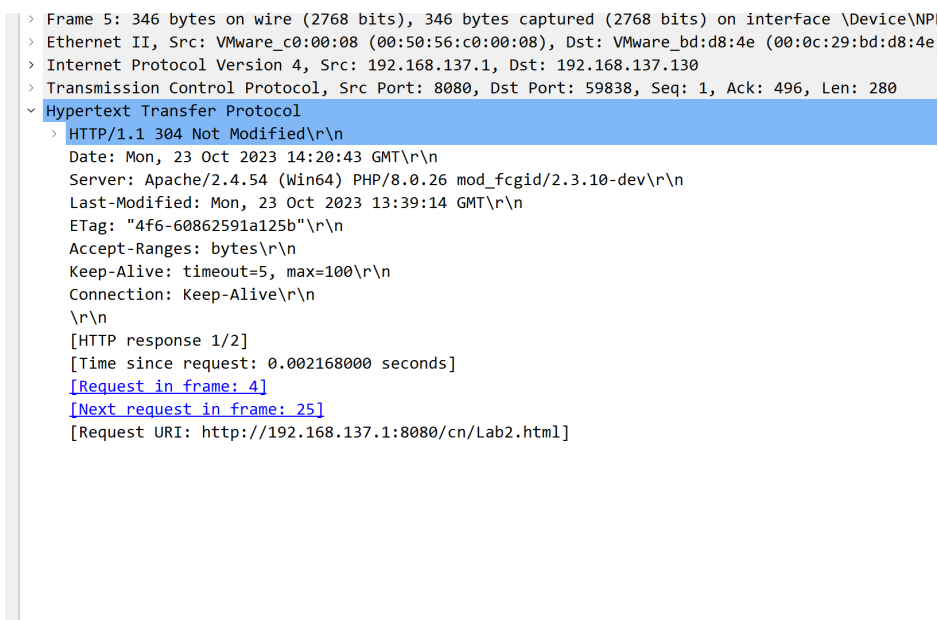


图 3.6: 第二个

- 根据指示找到了第二个 request, 它的请求是一个看似 url 加密的链接, 末尾是 MP3 格式, 可以看到它被单独的发送请求了, 经过搜索, 当浏览器解析到 html 音频、视频等文件后, 便自动向 Server 发送对这些资源的请求, 因此我会看到两个 request。

发现后面还有许多的 Tcp 包, 为了方便分析, 我对比有无音频数据的两个 Tcp Stream。也确实发现当我将页面重新把音频文件删掉后, 再次抓包不会出现这么多 Tcp 流。



图 3.7: 第三个

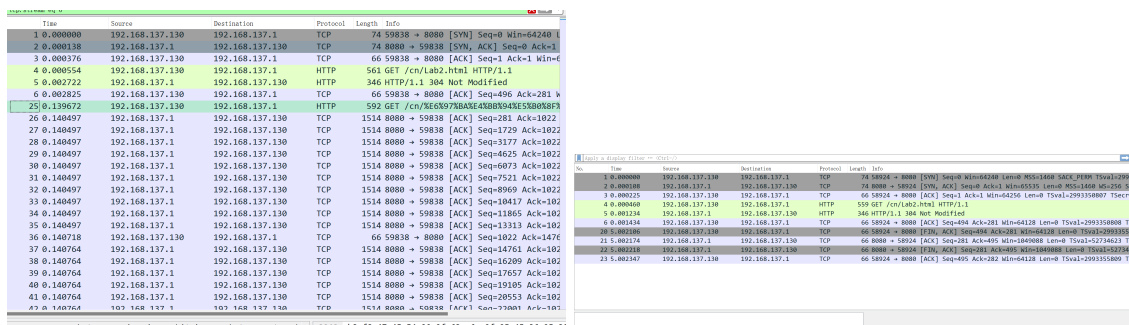


图 3.8: 有无音频的页面抓包数据包对比

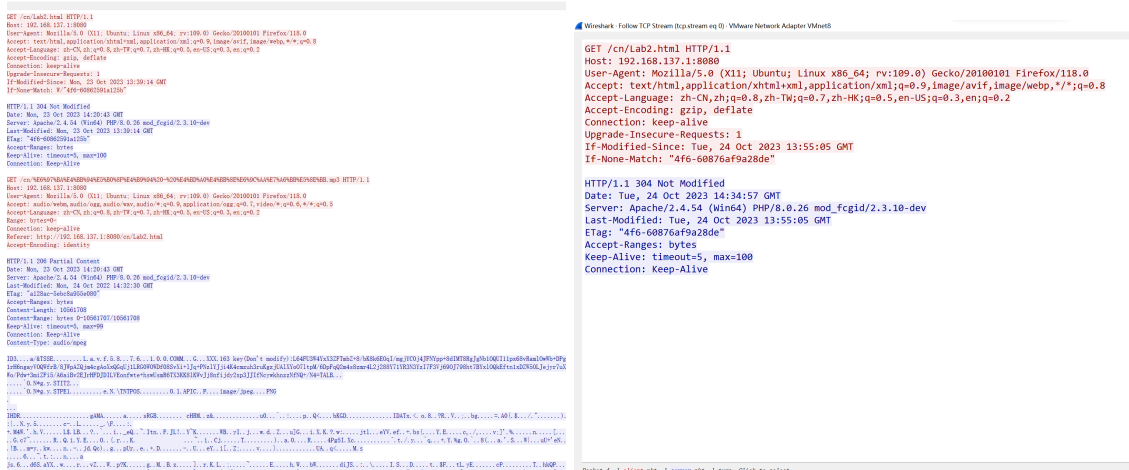


图 3.9: 有无音频的页面抓包 Tcp 流对比

- 有音频的页面，将由 Server 向 Client 传输大量的包，而普通的静态页面，仅仅只有所展示的几个数据包流量；值得注意的是，音频页面返回的 Tcp 流的大小都小于 **SYN 阶段 options 内定义的 1460**。
- 此外，从 Tcp 流整体看，**第一次发送对页面的 Request、Response 相同**，Client 发出相同的头部信息 get 页面，Server 发送 304（当然因为缓存原因...），但若是音频信息，Client **重新对音频文件的 url 链接 Get**，之后 Server 返回 206 的 response，继续将音频同个 Tcp 流分封为多个包发送。

4 0.000392	192.168.137.130	192.168.137.1	HTTP	599 GET /cn/Lab2.html HTTP/1.1
5 0.001142	192.168.137.1	192.168.137.130	HTTP	346 HTTP/1.1 304 Not Modified
6 0.001248	192.168.137.130	192.168.137.1	TCP	66 42614 → 8080 [ACK] Seq=534 Ack=281 Win=6412
10 5.001882	192.168.137.130	192.168.137.1	TCP	66 42614 → 8080 [FIN, ACK] Seq=534 Ack=281 Win=
11 5.001964	192.168.137.1	192.168.137.130	TCP	66 8080 → 42614 [ACK] Seq=281 Ack=535 Win=10496
12 5.002056	192.168.137.1	192.168.137.130	TCP	66 8080 → 42614 [FIN, ACK] Seq=281 Ack=535 Win=
13 5.002150	192.168.137.130	192.168.137.1	TCP	66 42614 → 8080 [ACK] Seq=535 Ack=282 Win=6412

Destination Port: 8080	0000 00 50 56 c0 00 08 00 0c 29 bd d0
Checksum: 01	0010 00 34 96 3f 40 00 40 06 10 b0 c0

图 3.10: 挥手阶段—音频已经缓存的情况

3.4 四次挥手

同样先简述一下过程，

- 第一次挥手客户端打算断开连接，Fin 置为 1，seq 设置当前为当前上一个发送 Client 发送 tcp 包 seq 值。
- 第二次挥手服务器收到连接 FIN 报文后 ACK 置为 1，ack 设置为接受 client 第一次挥手 Fin 的 seq+1，seq 设置为上一个 server 发送 tcp 包的 seq 的值 +1，进入 Fin_wait_2
- 第二次挥手后，服务器将剩余要发送的数据发送完毕，服务器进入 LASK ACK 状态，等待客户端的确认。服务器 FIN 和 ACK 置 1，序列号 Seq 在上一个 tcp 包的 seq 基础上 +1（应该就是第二次 seq 的值加上第二、三次挥手之间 server 发送 byte 数），ack 和第二次挥手相同。
- 第四次挥手，客户端收到来自服务器的连接释放 (FIN) 报文段后，FIN 和 ACK 置 1，Seq 继续 +1,ack 为第三次挥手的 seq+1。通话从此结束。

3.5 没有四次挥手问题

在实验过程中，发现很多次没有捕获到四次挥手的全流程；

多次测试，图一捕获到的页面并没有发送对音频 get 的请求，因此在退出时候 client 和 Server 的 ack 和 seq 的增量并不大

图二而在有大量 Client 对音频数据的 ack 包和 Server 对发送的大量大体积音频数据包的情况，且出现多次 <Fin,Ack> 包，这些包理论上只出现在四次挥手过程中的第三第四次挥手中。于此同时也在每一次对 MP3url 链接进行 get 前，都会进行一次三次握手，可以断定，这是和音频流数据的传输有关。

4 实验结论及心得体会

4.1 重新对完整的过程进行解读

询问过徐老师后，得知包处理过程和我服务器配置和浏览器的访问都有关系，且四次挥手效率不高，现代服务器架构会进行优化。最后我重新清理缓存，配合火狐 waterfall 进行一次解析。

1. 打开页面，对静态源码、图片发出 Get 请求，对于图片资源使用 Tcp 滑动窗口进行传输，中间 Client 发送 Ack 进行累计确认。（此时 get 请求都返回 200 了，因为无缓存）

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.137.130	192.168.137.1	TCP	74	46556 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=299...
2	0.000003	192.168.137.1	192.168.137.130	TCP	74	8080 → 46556 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 S...
3	0.000218	192.168.137.130	192.168.137.1	TCP	66	46556 → 8080 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2997659326 Tsecr...
6	0.001481	192.168.137.130	192.168.137.1	TCP	66	46556 → 8080 [ACK] Seq=534 Ack=281 Win=64128 Len=0 TSval=2997659327 T...
18	0.089457	192.168.137.130	192.168.137.1	TCP	66	46556 → 8080 [ACK] Seq=1060 Ack=14761 Win=55808 Len=0 TSval=299765941...
37	0.089685	192.168.137.130	192.168.137.1	TCP	66	46556 → 8080 [ACK] Seq=1060 Ack=40825 Win=48128 Len=0 TSval=299765941...
64	0.089895	192.168.137.130	192.168.137.1	TCP	66	46556 → 8080 [ACK] Seq=1060 Ack=72681 Win=44160 Len=0 TSval=299765941...
91	0.090102	192.168.137.130	192.168.137.1	TCP	66	46556 → 8080 [ACK] Seq=1060 Ack=78473 Win=61568 Len=0 TSval=299765941...
92	0.090155	192.168.137.130	192.168.137.1	TCP	66	46556 → 8080 [ACK] Seq=1060 Ack=116121 Win=40320 Len=0 TSval=29976594...
120	0.097989	192.168.137.130	192.168.137.1	TCP	66	46556 → 8080 [ACK] Seq=1060 Ack=145081 Win=46080 Len=0 TSval=29976594...
145	0.098245	192.168.137.130	192.168.137.1	TCP	66	46556 → 8080 [ACK] Seq=1060 Ack=155217 Win=84480 Len=0 TSval=29976594...
146	0.098286	192.168.137.130	192.168.137.1	TCP	66	46556 → 8080 [ACK] Seq=1060 Ack=189969 Win=153984 Len=0 TSval=2997659...
147	0.106675	192.168.137.130	192.168.137.1	TCP	66	46556 → 8080 [FIN, ACK] Seq=1060 Ack=189969 Win=153984 Len=0 TSval=29...
148	0.109468	192.168.137.130	192.168.137.1	TCP	74	46570 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=299...
149	0.109540	192.168.137.1	192.168.137.130	TCP	74	8080 → 46570 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 S...
150	0.109669	192.168.137.130	192.168.137.1	TCP	66	46570 → 8080 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2997659435 Tsecr...
162	0.110688	192.168.137.130	192.168.137.1	TCP	66	46570 → 8080 [ACK] Seq=533 Ack=14481 Win=55808 Len=0 TSval=2997659436...
181	0.110904	192.168.137.130	192.168.137.1	TCP	66	46570 → 8080 [ACK] Seq=533 Ack=40545 Win=48128 Len=0 TSval=2997659437...
208	0.111111	192.168.137.130	192.168.137.1	TCP	66	46570 → 8080 [ACK] Seq=533 Ack=72401 Win=44160 Len=0 TSval=2997659437...
235	0.111307	192.168.137.130	192.168.137.1	TCP	66	46570 → 8080 [ACK] Seq=533 Ack=115841 Win=36480 Len=0 TSval=299765943...
236	0.114155	192.168.137.130	192.168.137.1	TCP	66	46570 → 8080 [FIN, ACK] Seq=533 Ack=115841 Win=64128 Len=0 TSval=2997...
296	0.122660	192.168.137.130	192.168.137.1	TCP	60	46556 → 8080 [RST] Seq=1061 Win=0 Len=0
297	0.122788	192.168.137.130	192.168.137.1	TCP	60	46556 → 8080 [RST] Seq=1061 Win=0 Len=0
340	0.123061	192.168.137.130	192.168.137.1	TCP	60	46570 → 8080 [RST] Seq=534 Win=0 Len=0
341	0.123133	192.168.137.130	192.168.137.1	TCP	60	46570 → 8080 [RST] Seq=534 Win=0 Len=0
347	5.459038	192.168.137.130	192.168.137.1	TCP	74	41924 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=299...
348	5.459136	192.168.137.1	192.168.137.130	TCP	74	8080 → 41924 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 S...
349	5.459294	192.168.137.130	192.168.137.1	TCP	66	41924 → 8080 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2997664785 Tsecr...
361	5.460425	192.168.137.130	192.168.137.1	TCP	66	41924 → 8080 [ACK] Seq=533 Ack=14481 Win=55808 Len=0 TSval=2997664786...

图 3.11: 挥手阶段—音频未缓存的情况

No.	Time	Source	Destination	Protocol	Length	Info
8	0.357023	192.168.137.130	192.168.137.1	TCP	74	42792 → 8080 [SYN] Seq=0 Win=64240 Len=0
9	0.357114	192.168.137.1	192.168.137.130	TCP	74	8080 → 42792 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
10	0.357342	192.168.137.130	192.168.137.1	TCP	66	42792 → 8080 [ACK] Seq=1 Ack=1 Win=64256 Len=0
11	0.357575	192.168.137.130	192.168.137.1	HTTP	516	GET /cn/Lab2.html HTTP/1.1
12	0.359598	192.168.137.1	192.168.137.130	TCP	1514	8080 → 42792 [ACK] Seq=1 Ack=451 Win=65535 Len=0
13	0.359598	192.168.137.1	192.168.137.130	HTTP	206	HTTP/1.1 200 OK (text/html)
14	0.359858	192.168.137.130	192.168.137.1	TCP	66	42792 → 8080 [ACK] Seq=451 Ack=1589 Win=65535 Len=0
15	0.414404	192.168.137.130	192.168.137.1	HTTP	473	GET /cn/my_.jpg HTTP/1.1
16	0.415716	192.168.137.1	192.168.137.130	TCP	1514	8080 → 42792 [ACK] Seq=1589 Ack=858 Win=65535 Len=0
17	0.415716	192.168.137.1	192.168.137.130	TCP	1514	8080 → 42792 [ACK] Seq=3037 Ack=858 Win=65535 Len=0
18	0.415716	192.168.137.1	192.168.137.130	TCP	1514	8080 → 42792 [ACK] Seq=4485 Ack=858 Win=65535 Len=0
19	0.415716	192.168.137.1	192.168.137.130	TCP	1514	8080 → 42792 [ACK] Seq=5933 Ack=858 Win=65535 Len=0
20	0.415716	192.168.137.1	192.168.137.130	TCP	1514	8080 → 42792 [ACK] Seq=7381 Ack=858 Win=65535 Len=0
21	0.415716	192.168.137.1	192.168.137.130	TCP	1514	8080 → 42792 [ACK] Seq=8829 Ack=858 Win=65535 Len=0
22	0.415716	192.168.137.1	192.168.137.130	TCP	1514	8080 → 42792 [ACK] Seq=10277 Ack=858 Win=65535 Len=0
23	0.415716	192.168.137.1	192.168.137.130	TCP	1514	8080 → 42792 [ACK] Seq=11725 Ack=858 Win=65535 Len=0
24	0.415716	192.168.137.1	192.168.137.130	TCP	1514	8080 → 42792 [ACK] Seq=13173 Ack=858 Win=65535 Len=0
25	0.415716	192.168.137.1	192.168.137.130	TCP	1514	8080 → 42792 [ACK] Seq=14621 Ack=858 Win=65535 Len=0

图 4.12: Syn 以及 get 静态信息

- 当这些信息得到后，由服务器主动发送一个 <Fin,Ack> 报文，表示自己已经无资源给 Client 了;Client 接受后同时回复 <Fin,Ack> 报文；最后 Server 回复 Ack 结束这个端口的通话。

No.	Time	Source	Destination	Protocol	Length	Info
268	0.431504	192.168.137.1	192.168.137.130	TCP	1514	8080 → 42792 [ACK] Seq=349190 Ack=14621 Win=65535 Len=0
269	0.431564	192.168.137.1	192.168.137.130	TCP	1514	8080 → 42792 [ACK] Seq=350638 Ack=12666 Win=65535 Len=0
270	0.431564	192.168.137.1	192.168.137.130	HTTP	721	HTTP/1.1 200 OK (image/x-icon)
271	0.432212	192.168.137.130	192.168.137.1	TCP	66	42792 → 8080 [ACK] Seq=1266 Ack=2941 Win=65535 Len=0
272	0.432325	192.168.137.130	192.168.137.1	TCP	66	42792 → 8080 [ACK] Seq=1266 Ack=3527 Win=65535 Len=0
273	5.435036	192.168.137.1	192.168.137.130	TCP	66	8080 → 42792 [FIN, ACK] Seq=352741 Ack=12666 Win=0 Len=0
274	5.436047	192.168.137.130	192.168.137.1	TCP	66	42792 → 8080 [FIN, ACK] Seq=1266 Ack=352741 Win=0 Len=0
275	5.436137	192.168.137.1	192.168.137.130	TCP	66	8080 → 42792 [ACK] Seq=352742 Ack=12666 Win=0 Len=0

图 4.13: Fin

- 查看 waterfall，发现此时并没有音频资源的 get，但当点击播放按钮后，浏览器新端口 get 请求音频流（由图，新端口 35670，而之前关闭的 42792 端口处理静态其他属性），Server 通过这个端口通过 Tcp 滑动窗将资源传输到 Client 本地缓冲区，当 Client 点击音频播放按钮，便向 Server 发送 Get 请求，通过后续一些列 Tcp 流传到了 Client 缓存区。



图 4.14: WaterFall

470	140.336475	192.168.137.130	192.168.137.1	TCP	66 35670 → 8080 [ACK] Seq=1 Ack=1 Win=6
471	140.339823	192.168.137.130	192.168.137.1	HTTP	641 GET /cn/%E6%97%BA%E4%BB%94%E5%B0%8F%
472	140.340981	192.168.137.1	192.168.137.130	TCP	1514 8080 → 35670 [ACK] Seq=1 Ack=576 Win=
473	140.340981	192.168.137.1	192.168.137.130	TCP	1514 8080 → 35670 [ACK] Seq=1449 Ack=576
474	140.340981	192.168.137.1	192.168.137.130	TCP	1514 8080 → 35670 [ACK] Seq=2897 Ack=576
475	140.340981	192.168.137.1	192.168.137.130	TCP	1514 8080 → 35670 [ACK] Seq=4345 Ack=576
476	140.340981	192.168.137.1	192.168.137.130	TCP	1514 8080 → 35670 [ACK] Seq=5793 Ack=576
477	140.340981	192.168.137.1	192.168.137.130	TCP	1514 8080 → 35670 [ACK] Seq=7241 Ack=576
478	140.340981	192.168.137.1	192.168.137.130	TCP	1514 8080 → 35670 [ACK] Seq=8689 Ack=576
479	140.340981	192.168.137.1	192.168.137.130	TCP	1514 8080 → 35670 [ACK] Seq=10137 Ack=576

图 4.15: 新端口 Syn

4. 这个新端口, 当经过一定时间, 当 Server 发送了 206 Partial Content 表明正在进行流式传输。Client 回复 Ack, 并且这里还回复了 [TCP WINDOW UPdate] 的报文, 用来调整 TCP 接收窗口的大小, 以便能够更有效地接收更多的数据。又过了几个时间片, Server 确定无资源再给 Client 后, 发送 <Fin,Ack> 希望结束通讯, client 回复 <Fin,Ack>, 最后 Server 一次 Ack。

1	168.137.1	192.168.137.130	TCP	1514 8080 → 35670 [ACK] Seq=7412402 Ack=576 Win=1049088 Len=1448 T...
2	168.137.1	192.168.137.130	TCP	1514 8080 → 35670 [ACK] Seq=7413850 Ack=576 Win=1049088 Len=1448 T...
3	168.137.1	192.168.137.130	HTTP	1134 HTTP/1.1 206 Partial Content (audio/mpeg)
4	168.137.130	192.168.137.1	TCP	66 35670 → 8080 [ACK] Seq=576 Ack=7416366 Win=278272 Len=0 TSval=...
5	168.137.130	192.168.137.1	TCP	66 [TCP Window Update] 35670 → 8080 [ACK] Seq=576 Ack=7416366 Wi...
6	168.137.130	192.168.137.1	TCP	66 [TCP Window Update] 35670 → 8080 [ACK] Seq=576 Ack=7416366 Wi...
7	168.137.130	192.168.137.1	TCP	66 [TCP Window Update] 35670 → 8080 [ACK] Seq=576 Ack=7416366 Wi...
8	168.137.1	192.168.137.130	TCP	66 8080 → 35670 [FIN, ACK] Seq=7416366 Ack=576 Win=1049088 Len=0...
9	168.137.130	192.168.137.1	TCP	66 35670 → 8080 [FIN, ACK] Seq=576 Ack=7416367 Win=3059584 Len=0...
10	168.137.1	192.168.137.130	TCP	66 8080 → 35670 [ACK] Seq=7416367 Ack=577 Win=1049088 Len=0 TSva...

图 4.16: 音频流退出

5 实验结论及心得体会