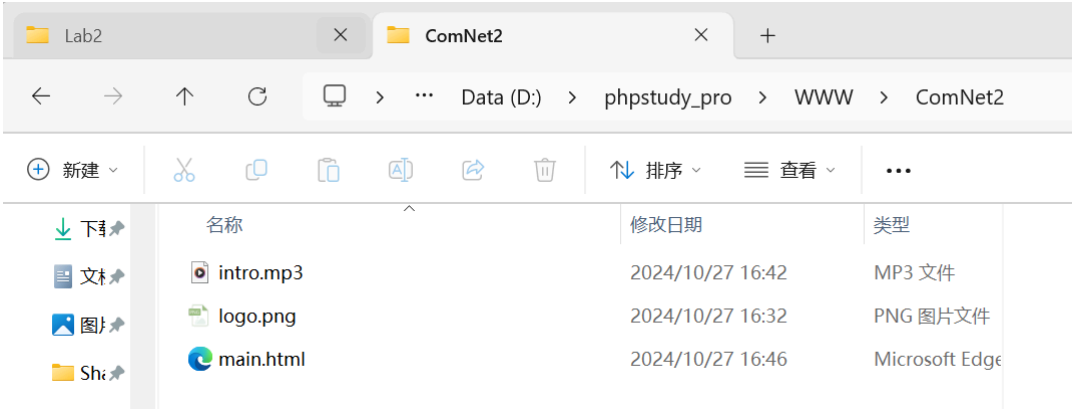


计网实验二——配置Web服务器，分析HTTP交互过程

一、Web服务器配置

本次实验中使用 phpstudy 中打包好的 Apache2.4.39 提供网页服务。

- 将编写好的main.html与相关图像音频文件打包放在WWW文件夹下。



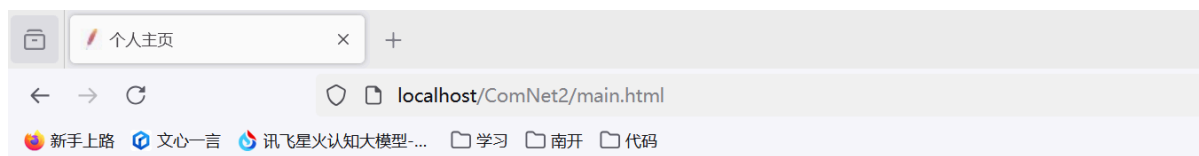
- 在phpstudy中打开Apache服务，即可在浏览器中访问我们编写的html。



- phpstudy中已经建立了一个默认的localhost站点，使用http协议，绑定端口为80，在后续实验中我们会用到这些信息。



- 按照路径访问页面即可。



个人主页

专业: 信息安全

学号: 2211532

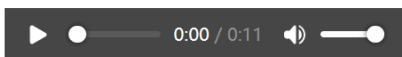
姓名: 石家伊

我的LOGO



介绍音频

以下是我的自我介绍音频:



页面源码附上:

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
  <meta charset="UTF-8">
```

```
<title>个人主页</title>
</head>
<body>
  <h1>个人主页</h1>
  <p>专业：信息安全</p>
  <p>学号：2211532</p>
  <p>姓名：石家伊</p>

  <h2>我的LOGO</h2>
  

  <h2>介绍音频</h2>
  <p>以下是我的自我介绍音频： </p>
  <audio controls>
    <source src="intro.mp3" type="audio/mpeg">
    您的浏览器不支持音频播放。
  </audio>
</body>
</html>
```

二、交互过程分析

- 开启服务后，我们就打开Wireshark准备捕获。
- 选择 Adapter for loopback traffic capture 接口，并按所用端口设置过滤器 (tcp.srcport == 80 or tcp.dstport == 80)，筛选源端口或目标端口为80的数据包。
- 用浏览器访问<http://localhost/ComNet2/main.html>，即可看到Wireshark的抓包信息。

No.	Time	Source	Destination	Protocol	Length	Info
995	27.221864	127.0.0.1	127.0.0.1	TCP	56	52334 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
996	27.221904	127.0.0.1	127.0.0.1	TCP	56	80 → 52334 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
997	27.221939	127.0.0.1	127.0.0.1	TCP	44	52334 → 80 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
1010	27.225443	127.0.0.1	127.0.0.1	HTTP	595	GET /ComNet2/main.html HTTP/1.1
1011	27.225476	127.0.0.1	127.0.0.1	TCP	44	80 → 52334 [ACK] Seq=1 Ack=552 Win=2160640 Len=0
1012	27.226781	127.0.0.1	127.0.0.1	HTTP	944	HTTP/1.1 200 OK (text/html)
1013	27.226823	127.0.0.1	127.0.0.1	TCP	44	52334 → 80 [ACK] Seq=552 Ack=901 Win=2160384 Len=0
1030	27.268162	127.0.0.1	127.0.0.1	HTTP	552	GET /ComNet2/logo.png HTTP/1.1
1031	27.268190	127.0.0.1	127.0.0.1	TCP	44	80 → 52334 [ACK] Seq=901 Ack=1060 Win=2160128 Len=0
1036	27.269201	127.0.0.1	127.0.0.1	TCP	65539	80 → 52334 [ACK] Seq=901 Ack=1060 Win=2160128 Len=65495 [TCP PDU reassembled in 1037]
1037	27.269227	127.0.0.1	127.0.0.1	HTTP	23632	HTTP/1.1 200 OK (JPEG JFIF image)
1038	27.269284	127.0.0.1	127.0.0.1	TCP	44	52334 → 80 [ACK] Seq=1060 Ack=89984 Win=2161152 Len=0
1131	27.283282	127.0.0.1	127.0.0.1	HTTP	573	GET /ComNet2/intro.mp3 HTTP/1.1
1132	27.283307	127.0.0.1	127.0.0.1	TCP	44	80 → 52334 [ACK] Seq=89984 Ack=1509 Win=2159616 Len=0
1137	27.284375	127.0.0.1	127.0.0.1	MPEG-2	43741	Audio Layer 3, 32 kb/s, 24 kHz
1138	27.284432	127.0.0.1	127.0.0.1	TCP	44	52334 → 80 [ACK] Seq=1509 Ack=133681 Win=2117632 Len=0
1147	27.285286	127.0.0.1	127.0.0.1	HTTP	544	GET /favicon.ico HTTP/1.1
1148	27.285308	127.0.0.1	127.0.0.1	TCP	44	80 → 52334 [ACK] Seq=133681 Ack=2089 Win=2159104 Len=0
1153	27.285976	127.0.0.1	127.0.0.1	HTTP	1312	HTTP/1.1 200 OK (JPEG JFIF image)
1154	27.286005	127.0.0.1	127.0.0.1	TCP	44	52334 → 80 [ACK] Seq=2089 Ack=134949 Win=2116352 Len=0
1313	32.290492	127.0.0.1	127.0.0.1	TCP	44	80 → 52334 [FIN, ACK] Seq=134949 Ack=2089 Win=2159104 Len=0
1315	32.290562	127.0.0.1	127.0.0.1	TCP	44	52334 → 80 [ACK] Seq=2089 Ack=134950 Win=2116352 Len=0
1316	32.290719	127.0.0.1	127.0.0.1	TCP	44	52334 → 80 [FIN, ACK] Seq=2089 Ack=134950 Win=2116352 Len=0
1317	32.290797	127.0.0.1	127.0.0.1	TCP	44	80 → 52334 [ACK] Seq=134950 Ack=2090 Win=2159104 Len=0

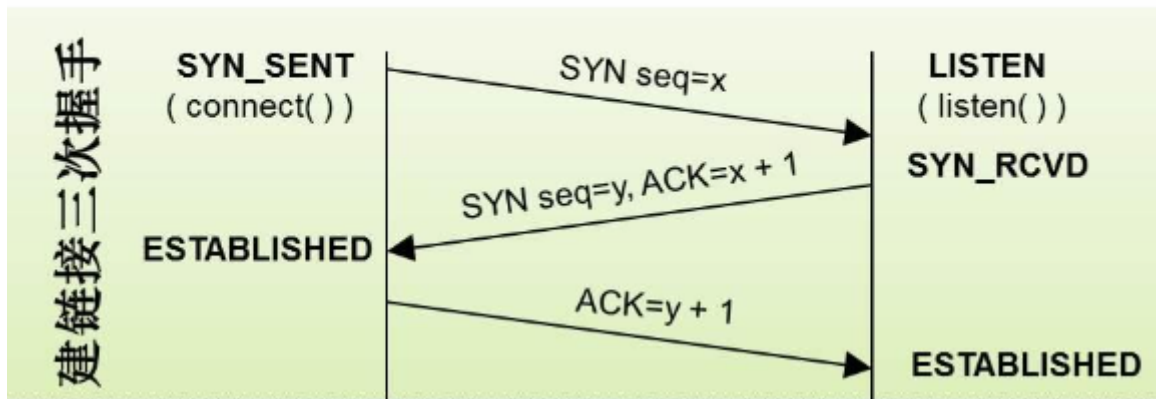
- 下面对整个交互过程进行分析。

建立连接

TCP通过三次握手与服务器建立连接。

Client

Server



在Wireshark捕获到的数据中可以看到这三次握手：

No.	Time	Source	Destination	Protocol	Length	Info
995	27.221864	127.0.0.1	127.0.0.1	TCP	56	52334 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
996	27.221994	127.0.0.1	127.0.0.1	TCP	56	80 → 52334 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
997	27.221939	127.0.0.1	127.0.0.1	TCP	44	52334 → 80 [ACK] Seq=1 Ack=1 Win=2161152 Len=0

第一次握手

客户端向服务器发送了一个**SYN**请求，即客户端向服务器请求连接。此时客户端进入**SYN_SENT（同步已发送）**状态。

```
> Frame 995: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_{...}
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
  > Transmission Control Protocol, Src Port: 52334, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 52334
    Destination Port: 80
    [Stream index: 51]
    [Stream Packet Number: 1]
    > [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0 (relative sequence number)
    Sequence Number (raw): 153437455
    [Next Sequence Number: 1 (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
    > Flags: 0x002 (SYN)
      0000 .... = Reserved: Not set
      ...0 .... = Accurate ECN: Not set
      ....0... = Congestion Window Reduced: Not set
      ....0... = ECN-Echo: Not set
      ....0... = Urgent: Not set
      ....0... = Acknowledgment: Not set
      ....0... = Push: Not set
      ....0... = Reset: Not set
      > ....0...1... = Syn: Set
      ....0...0... = Fin: Not set
      [TCP Flags: .....S.]
      Window: 65535
      [Calculated window size: 65535]
      Checksum: 0x5bf7 [unverified]
      [Checksum Status: Unverified]
      Urgent Pointer: 0
    > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP).
      > TCP Option - Maximum segment size: 65495 bytes
      > TCP Option - No-Operation (NOP)
      > TCP Option - Window scale: 8 (multiply by 256)
      > TCP Option - No-Operation (NOP)
      > TCP Option - No-Operation (NOP)
      > TCP Option - SACK permitted
```

可以看到报文的信息：

- Source Port 源端口，即客户端的端口：52334
- Destination Port 目标端口，即服务器的端口：80
- Sequence Number 序列号，表示数据流中该数据包的起始位置：初始为0
- Acknowledge Number 确认号，标识了已经成功接收到的数据字节：初始为0
- Flags 标志位：SYN标识，用于建立连接的初始握手，请求建立连接
- Window 窗口大小，当前接收方（客户端）可以接受的数据量：65535
- Maximum segment size 最大报文段大小，表示单个 TCP 报文段中可以传输的最大数据量（不含TCP头和Option）：65495
- Window scale 窗口缩放，声明了一个shift count，作为2的指数，再乘以TCP定义的接收窗口，得到实际的TCP窗口大小：8

第二次握手

服务器收到请求后，向客户端发送了一个ACK+SYN的回应，表明服务器收到了客户端的请求，并且同意了客户端的请求建立连接，之后进入SYN_RCVD（同步已接收）状态。

> Frame 996: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_{Loopback
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▼ Transmission Control Protocol, Src Port: 80, Dst Port: 52334, Seq: 0, Ack: 1, Len: 0
 Source Port: 80
 Destination Port: 52334
 [Stream index: 51]
 [Stream Packet Number: 2]
 > [Conversation completeness: Complete, WITH_DATA (31)]
 [TCP Segment Len: 0]
 Sequence Number: 0 (relative sequence number)
 Sequence Number (raw): 4151748463
 [Next Sequence Number: 1 (relative sequence number)]
 Acknowledgment Number: 1 (relative ack number)
 Acknowledgment number (raw): 153437456
 1000 = Header Length: 32 bytes (8)
 ▼ Flags: 0x012 (SYN, ACK)
 000. = Reserved: Not set
 ...0 = Accurate ECN: Not set
 ...0 = Congestion Window Reduced: Not set
 0. = ECN-Echo: Not set
 0. = Urgent: Not set
 1. = Acknowledgment: Set
 0. = Push: Not set
 0. = Reset: Not set
 >1. = Syn: Set
 0. = Fin: Not set
 [TCP Flags:A..S..]
 Window: 65535
 [Calculated window size: 65535]
 Checksum: 0xbcff [unverified]
 [Checksum Status: Unverified]
 Urgent Pointer: 0
 ▼ Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP).
 > TCP Option - Maximum segment size: 65495 bytes
 > TCP Option - No-Operation (NOP)
 > TCP Option - Window scale: 8 (multiply by 256)
 > TCP Option - No-Operation (NOP)
 > TCP Option - No-Operation (NOP)
 > TCP Option - No-Operation (NOP)
 > TCP Option - SACK permitted

0000 02 00 00 00 45 00 00 34 92 2d 40 00 00 06 00 00E..4..@....
0010 7f 00 00 01 7f 00 00 01 00 50 cc 6e f7 76 a7 6fP.n.v.o
0020 09 25 45 10 80 12 ff ff bc ff 00 00 02 04 ff d7%E.....
0030 01 03 03 08 01 01 04 02

报文信息中，可以看到：

- 源端口与目标端口号更换。
- Seq序列号仍为0，因为服务器现在还没有发送数据。
- Ack接收号变为1，因为接收到了客户端发来的SYN标志位。
- 发送的Flags为ACK与SYN，ACK用于确认数据的传输，接收方发送一个带有ACK标记的报文段回复发送方，确认已经收到了数据。

第三次握手

最后一次握手，客户端收到服务器端确认后，检查ack是否为x+1，即ACK是否为1。如果正确则客户端回应服务器一个ACK报文，告诉服务器我收到了你的确认，同时进入ESTABLISHED状态。

> Frame 997: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_{Loopback
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▼ Transmission Control Protocol, Src Port: 52334, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
 Source Port: 52334
 Destination Port: 80
 [Stream index: 51]
 [Stream Packet Number: 3]
 > [Conversation completeness: Complete, WITH_DATA (31)]
 [TCP Segment Len: 0]
 Sequence Number: 1 (relative sequence number)
 Sequence Number (raw): 153437456
 [Next Sequence Number: 1 (relative sequence number)]
 Acknowledgment Number: 1 (relative ack number)
 Acknowledgment number (raw): 4151748464
 0101 = Header Length: 20 bytes (5)
 ▼ Flags: 0x010 (ACK)
 000. = Reserved: Not set
 ...0 = Accurate ECN: Not set
 ...0 = Congestion Window Reduced: Not set
 0. = ECN-Echo: Not set
 0. = Urgent: Not set
 1. = Acknowledgment: Set
 0. = Push: Not set
 0. = Reset: Not set
 0. = Syn: Not set
 0. = Fin: Not set
 [TCP Flags:A.....]
 Window: 8442
 [Calculated window size: 2161152]
 [Window size scaling factor: 256]
 Checksum: 0xd6fc [unverified]
 [Checksum Status: Unverified]
 Urgent Pointer: 0
 > [Timestamps]
 > [SEQ/ACK analysis]

0000 02 00 00 00 45 00 00 28 92 2e 40 00 00 06 00 00E..(..@....
0010 7f 00 00 01 7f 00 00 01 cc 6e 00 50 09 25 45 10P.n.v.o
0020 f7 76 a7 70 50 10 20 fa d6 fc 00 00v.pP.....

- 源端口与目的端口分别为客户端与服务器端口号。
- Seq变为1，当前已经发送过一次数据（第一次握手）。

- Ack为1，当前接收到了服务器端发来的一字节数据。
- Flags发送的标志位为ACK。

至此，三次握手完毕，客户端与服务器的连接正确建立。

数据传输

建立连接后，就开始了数据传输的过程。

浏览器会给对应的服务器发送一个 HTTP 请求，对应的服务器收到这个请求之后，经过计算处理，就会返回一个 HTTP 响应。在HTTP协议中，客户端与服务器之间是一发一收、一问一答的。

在捕获的包中可以看到，一共发生了四次HTTP请求与响应。

1010	27.225443	127.0.0.1	127.0.0.1	HTTP	595 GET /ComNet2/main.html HTTP/1.1
1011	27.225476	127.0.0.1	127.0.0.1	TCP	44 80 → 52334 [ACK] Seq=1 Ack=552 Win=2160640 Len=0
1012	27.226781	127.0.0.1	127.0.0.1	HTTP	944 HTTP/1.1 200 OK (text/html)
1013	27.226823	127.0.0.1	127.0.0.1	TCP	44 52334 → 80 [ACK] Seq=552 Ack=901 Win=2160384 Len=0
1030	27.268162	127.0.0.1	127.0.0.1	请求 → HTTP	552 GET /ComNet2/logo.png HTTP/1.1
1031	27.268190	127.0.0.1	127.0.0.1	TCP	44 80 → 52334 [ACK] Seq=901 Ack=1060 Win=2160128 Len=0
1036	27.269201	127.0.0.1	127.0.0.1	TCP	65539 80 → 52334 [ACK] Seq=901 Ack=1060 Win=2160128 Len=65495 [TCP PDU reassembled in 1037]
1037	27.269227	127.0.0.1	127.0.0.1	响应 → HTTP	23632 HTTP/1.1 200 OK (JPEG JFIF image)
1038	27.269284	127.0.0.1	127.0.0.1	TCP	44 52334 → 80 [ACK] Seq=1060 Ack=89984 Win=2161152 Len=0
1131	27.283282	127.0.0.1	127.0.0.1	HTTP	573 GET /ComNet2/intro.mp3 HTTP/1.1
1132	27.283307	127.0.0.1	127.0.0.1	TCP	44 80 → 52334 [ACK] Seq=89984 Ack=1589 Win=2159616 Len=0
1137	27.284375	127.0.0.1	127.0.0.1	MPEG-2	43741 Audio Layer 3, 32 kb/s, 24 kHz
1138	27.284432	127.0.0.1	127.0.0.1	TCP	44 52334 → 80 [ACK] Seq=1589 Ack=133681 Win=2117632 Len=0
1147	27.285286	127.0.0.1	127.0.0.1	HTTP	544 GET /favicon.ico HTTP/1.1
1148	27.285308	127.0.0.1	127.0.0.1	TCP	44 80 → 52334 [ACK] Seq=133681 Ack=2089 Win=2159104 Len=0
1153	27.285976	127.0.0.1	127.0.0.1	HTTP	1312 HTTP/1.1 200 OK (JPEG JFIF image)

根据文件的名称，四次分别请求了html文件、网页中的图片文件、音频文件、浏览器图标。

请求报文

HTTP请求报文由请求行、请求头、空行与请求正文四部分组成。

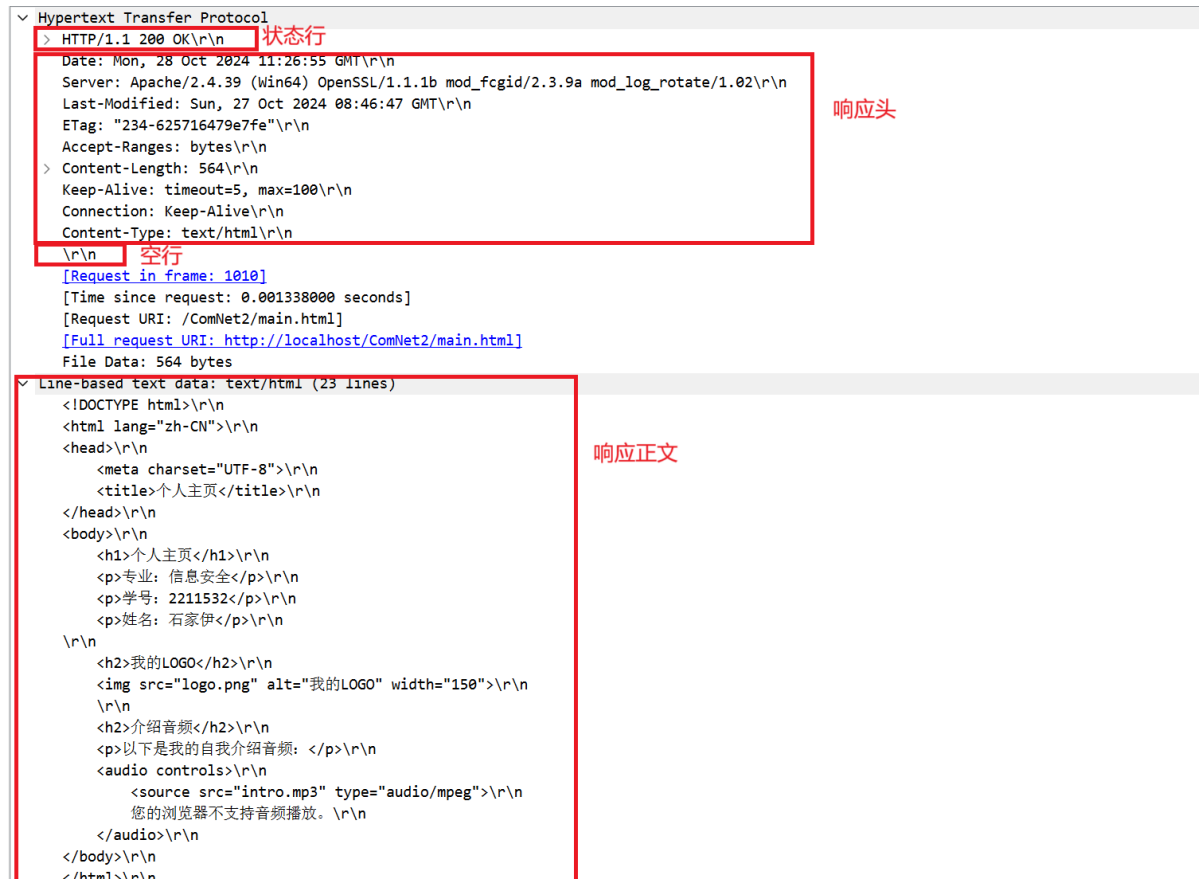
HTTP 请求	首行:	方法 URL 版本号
		GET https://www.sogou.com/ HTTP/1.1
	协议头 (header):	Host: www.sogou.com Connection: keep-alive Cache-Control: max-age=0 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="98", "Google Chrome";v="98" sec-ch-ua-mobile: ?0 sec-ch-ua-platform: "Windows" Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.102 Safari/537.36 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 Sec-Fetch-Site: none Sec-Fetch-Mode: navigate Sec-Fetch-User: ?1 Sec-Fetch-Dest: document Accept-Encoding: gzip, deflate, br Accept-Language: zh-CN,zh;q=0.9 Cookie: SUV=003B56A6DA4C2A82610BB3A8CFD5D583; SUID=DAE58074A021B00A0000000061CDAA89; ABTEST=0 1646288379 v17; IPLC
	空行:	空行是协议头结束的标志
正文 (body):		空行后面的部分就是正文。正文允许为空字符串。如果正文存在，则协议头中会出现一个 Content-Length 属性来标识正文的长度

- 请求行：指明客户端想要执行的操作。
 - 格式：<请求方法> <请求路径url> <HTTP版本>
- 请求头：包含一些关于请求的元信息，例如请求源、连接类型等，用于描述请求正文。
- 空行：用于分隔请求头和请求正文。空行后紧跟请求正文。
- 请求正文：主要用于向服务器传递数据，通常在 POST、PUT 等请求方法中使用，而 GET 请求一般不包含请求正文。

以捕获到的第一个请求为例：

- **500 Internal Server Error**: 服务器内部错误。
 - **503 Service Unavailable**: 服务器暂时无法处理请求。
- 响应头: 包含了关于响应的一些元信息, 例如内容类型、服务器信息、缓存控制等。
- 空行: 用于分隔响应头和响应正文。空行之后紧跟响应正文。
- 响应正文: 服务器发送给客户端的实际内容, 例如 HTML 页面、图片数据、JSON 数据等。
 - 响应正文在状态码 **200 OK** 等成功状态下会包含实际数据, 而在 **404 Not Found** 等错误状态下通常包含错误信息。

以捕获到的第一个请求的响应为例:



The image shows a Wireshark packet capture of an HTTP response. The packet list on the left shows packet 1012 as the first response. The packet details pane on the right shows the structure of the response:

- 状态行 (Status Line):** HTTP/1.1 200 OK
- 响应头 (Response Header):**
 - Date: Mon, 28 Oct 2024 11:26:55 GMT
 - Server: Apache/2.4.39 (Win64) OpenSSL/1.1.1b mod_fcgid/2.3.9a mod_log_rotate/1.02
 - Last-Modified: Sun, 27 Oct 2024 08:46:47 GMT
 - ETag: "234-625716479e7fe"
 - Accept-Ranges: bytes
 - Content-Length: 564
 - Keep-Alive: timeout=5, max=100
 - Connection: Keep-Alive
 - Content-Type: text/html
- 空行 (Empty Line):** A blank line separating the headers from the body.
- 响应正文 (Response Body):** The HTML document content, starting with <!DOCTYPE html> and containing a title, body text, and an audio player.

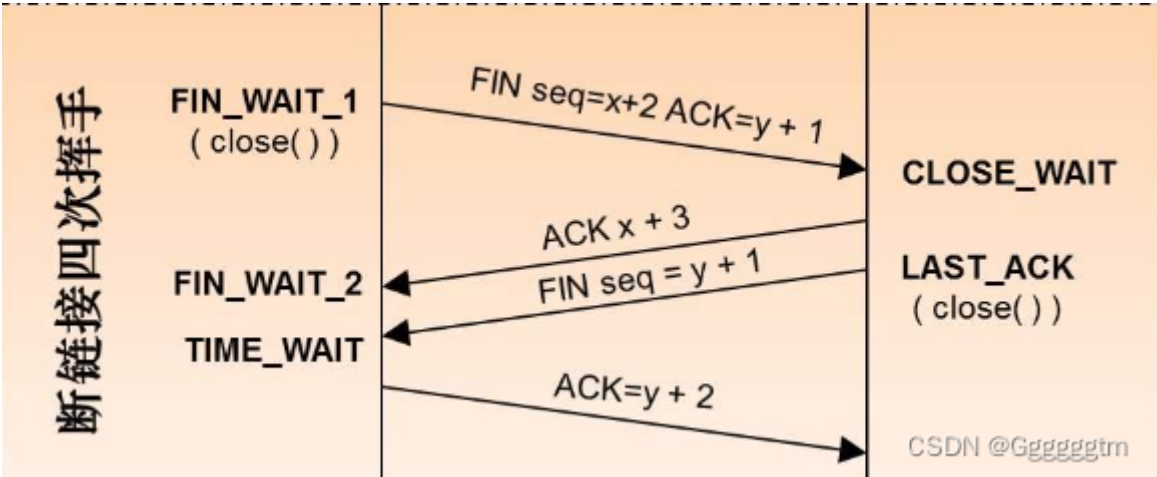
- 状态行: **HTTP/1.1 200 OK**。协议版本为HTTP/1.1, 状态码为200, 状态描述为OK。
- 响应头: 包含了多条信息, 如Date描述服务器响应的日期和时间, Server描述服务器的类型和版本信息等。
- 空行: 分隔响应头和响应正文。
- 响应正文: 服务器返回的 HTML 文档内容, 将显示在浏览器中。可以看到内容就是我们编写的html源码。

在每一次收到HTTP响应后, 客户端都会向服务器发送一个**ACK**表示收到。

1010	27.225443	127.0.0.1	127.0.0.1	HTTP	595 GET /ComNet2/main.html HTTP/1.1
1011	27.225476	127.0.0.1	127.0.0.1	TCP	44 80 → 52334 [ACK] Seq=1 Ack=552 Win=2160640 Len=0
1012	27.226781	127.0.0.1	127.0.0.1	HTTP	944 HTTP/1.1 200 OK (text/html)
1013	27.226823	127.0.0.1	127.0.0.1	TCP	44 52334 → 80 [ACK] Seq=552 Ack=901 Win=2160384 Len=0
1030	27.268162	127.0.0.1	127.0.0.1	HTTP	552 GET /ComNet2/logo.png HTTP/1.1
1031	27.268190	127.0.0.1	127.0.0.1	TCP	44 80 → 52334 [ACK] Seq=901 Ack=1060 Win=2160128 Len=0
1036	27.269201	127.0.0.1	127.0.0.1	TCP	65539 80 → 52334 [ACK] Seq=901 Ack=1060 Win=2160128 Len=65495 [TCP PDU reassembled in 1037]
1037	27.269227	127.0.0.1	127.0.0.1	HTTP	23632 HTTP/1.1 200 OK (JPEG JFIF image)
1038	27.269284	127.0.0.1	127.0.0.1	TCP	44 52334 → 80 [ACK] Seq=1060 Ack=89984 Win=2161152 Len=0
1131	27.283282	127.0.0.1	127.0.0.1	HTTP	573 GET /ComNet2/intro.mp3 HTTP/1.1
1132	27.283307	127.0.0.1	127.0.0.1	TCP	44 80 → 52334 [ACK] Seq=89984 Ack=1589 Win=2159616 Len=0
1137	27.284375	127.0.0.1	127.0.0.1	MPEG-2	43741 Audio Layer 3, 32 kb/s, 24 kHz
1138	27.284432	127.0.0.1	127.0.0.1	TCP	44 52334 → 80 [ACK] Seq=1589 Ack=133681 Win=2117632 Len=0
1147	27.285286	127.0.0.1	127.0.0.1	HTTP	544 GET /favicon.ico HTTP/1.1
1148	27.285308	127.0.0.1	127.0.0.1	TCP	44 80 → 52334 [ACK] Seq=133681 Ack=2089 Win=2159104 Len=0
1153	27.285976	127.0.0.1	127.0.0.1	HTTP	1312 HTTP/1.1 200 OK (JPEG JFIF image)
1154	27.286005	127.0.0.1	127.0.0.1	TCP	44 52334 → 80 [ACK] Seq=2089 Ack=134949 Win=2116352 Len=0

断开连接

数据传输完成后，服务端会进行四次挥手关闭连接。



在WireShark捕获到的数据中可以看到这四次挥手：

1313	32.290492	127.0.0.1	127.0.0.1	TCP	44 80 → 52334 [FIN, ACK] Seq=134949 Ack=2089 Win=2159104 Len=0
1315	32.290562	127.0.0.1	127.0.0.1	TCP	44 52334 → 80 [ACK] Seq=2089 Ack=134950 Win=2116352 Len=0
1316	32.290719	127.0.0.1	127.0.0.1	TCP	44 52334 → 80 [FIN, ACK] Seq=2089 Ack=134950 Win=2116352 Len=0
1317	32.290797	127.0.0.1	127.0.0.1	TCP	44 80 → 52334 [ACK] Seq=134950 Ack=2090 Win=2159104 Len=0

第一次挥手

服务器向客户端发送主动关闭报文**FIN+ACK**，表示服务器要关闭连接，不再发送数据。此时服务器进入 **FIN_WAIT_1** 状态。

> Frame 1313: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_{...} Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 80, Dst Port: 52334, Seq: 134949, Ack: 2089, Len: 0

Source Port: 80

Destination Port: 52334

[Stream index: 51]

[Stream Packet Number: 21]

> [Conversation completeness: Complete, WITH_DATA (31)]

[TCP Segment Len: 0]

Sequence Number: 134949 (relative sequence number)

Sequence Number (raw): 4151883412

[Next Sequence Number: 134950 (relative sequence number)]

Acknowledgment Number: 2089 (relative ack number)

Acknowledgment number (raw): 153439544

0101 ... = Header Length: 20 bytes (5)

> Flags: 0x011 (FIN, ACK)

000. = Reserved: Not set

...0 = Accurate ECN: Not set

...0 = Congestion Window Reduced: Not set

...0 = ECN-Echo: Not set

...0 = Urgent: Not set

...1 = Acknowledgment: Set

...0 = Push: Not set

...0 = Reset: Not set

...0 = Syn: Not set

> [TCP Flags:A....F]

Window: 8434

[Calculated window size: 2159104]

[Window size scaling factor: 256]

Checksum: 0xbfb5 [unverified]

[Checksum Status: Unverified]

Urgent Pointer: 0

> [Timestamps]

0000 02 00 00 00 45 00 00 28 93 69 40 00 00 06 00 00E..(..i@....

0010 7f 00 00 01 7f 00 00 01 00 50 cc 6e f7 78 b6 94P.n.x..

0020 09 25 4d 38 50 11 20 f2 bf b5 00 00 %BP.....

- 源端口与目的端口分别为服务器80与客户端52334。
- Seq与Ack再经过多次数据传输后都已改变，为134949与2089。
- Flags标志位为FIN与ACK。

第二次挥手

客户端要向服务器确认收到 **FIN** 包，于是发送 **ACK** 确认包。此时客户端进入 **CLOSE_WAIT** 状态，而服务器在收到ACK后进入 **FIN_WAIT_2** 状态。

```
> Frame 1315: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_{Loopback}
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
√ Transmission Control Protocol, Src Port: 52334, Dst Port: 80, Seq: 2089, Ack: 134950, Len: 0
  Source Port: 52334
  Destination Port: 80
  [Stream index: 51]
  [Stream Packet Number: 22]
  [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 0]
  Sequence Number: 2089 (relative sequence number)
  Sequence Number (raw): 153439544
  [Next Sequence Number: 2089 (relative sequence number)]
  Acknowledgment Number: 134950 (relative ack number)
  Acknowledgment number (raw): 4151883413
  0101 .... = Header Length: 20 bytes (5)
  √ Flags: 0x010 (ACK)
    000. .... = Reserved: Not set
    ...0 .... = Accurate ECN: Not set
    ....0... = Congestion Window Reduced: Not set
    ....0... = ECN-Echo: Not set
    ....0... = Urgent: Not set
    ....1... = Acknowledgment: Set
    ....0... = Push: Not set
    ....0... = Reset: Not set
    ....0... = Syn: Not set
    ....0... = Fin: Not set
    [TCP Flags: .....A....]
  Window: 8267
  [Calculated window size: 2116352]
  [Window size scaling factor: 256]
  Checksum: 0xc05c [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  [Timestamps]
  [SEQ/ACK analysis]
```

- 源端口与目的端口分别为客户端52334与服务器80。是由客户端发给服务器的。
- Seq与Ack, 为2089与134950。
- Flags标志位为ACK, 表示客户端收到了服务器的主动关闭报文, 并回复收到。

第三次挥手

客户端发送 **FIN+ACK** 包, 表示它也要关闭连接。此时客户端进入 **LAST_ACK** 状态, 等待服务器的最终确认。

```
> Frame 1316: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_{Loopback}
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
√ Transmission Control Protocol, Src Port: 52334, Dst Port: 80, Seq: 2089, Ack: 134950, Len: 0
  Source Port: 52334
  Destination Port: 80
  [Stream index: 51]
  [Stream Packet Number: 23]
  [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 0]
  Sequence Number: 2089 (relative sequence number)
  Sequence Number (raw): 153439544
  [Next Sequence Number: 2090 (relative sequence number)]
  Acknowledgment Number: 134950 (relative ack number)
  Acknowledgment number (raw): 4151883413
  0101 .... = Header Length: 20 bytes (5)
  √ Flags: 0x011 (FIN, ACK)
    000. .... = Reserved: Not set
    ...0 .... = Accurate ECN: Not set
    ....0... = Congestion Window Reduced: Not set
    ....0... = ECN-Echo: Not set
    ....0... = Urgent: Not set
    ....1... = Acknowledgment: Set
    ....0... = Push: Not set
    ....0... = Reset: Not set
    ....0... = Syn: Not set
    ....1... = Fin: Set
    [TCP Flags: .....A...F]
  Window: 8267
  [Calculated window size: 2116352]
  [Window size scaling factor: 256]
  Checksum: 0xc05b [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  [Timestamps]
```

- 源端口与目的端口分别为客户端52334与服务器80。是由客户端发给服务器的。
- Seq与Ack, 为2089与134950。
- Flags标志位为FIN与ACK, 表示客户端也要关闭连接。

第四次挥手

服务器发送 **ACK** 确认包, 表示已收到客户端的 FIN 包。此时服务器进入 **TIME_WAIT** 状态。客户端收到 ACK报文后, 就关闭连接, 也处于**CLOSED**状态了。连接完全关闭。

```
> Frame 1317: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_{...}
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
  Transmission Control Protocol, Src Port: 80, Dst Port: 52334, Seq: 134950, Ack: 2090, Len: 0
    Source Port: 80
    Destination Port: 52334
    [Stream index: 51]
    [Stream Packet Number: 24]
  > [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 0]
  Sequence Number: 134950 (relative sequence number)
  Sequence Number (raw): 4151883413
  [Next Sequence Number: 134950 (relative sequence number)]
  Acknowledgment Number: 2090 (relative ack number)
  Acknowledgment number (raw): 153439545
  0101 .... = Header Length: 20 bytes (5)
  Flags: 0x010 (ACK)
    000. .... = Reserved: Not set
    ...0 .... = Accurate ECN: Not set
    ....0. .... = Congestion Window Reduced: Not set
    ....0.. .... = ECN-Echo: Not set
    ....0. .... = Urgent: Not set
    ....1. .... = Acknowledgment: Set
    ....0. .... = Push: Not set
    ....0. .... = Reset: Not set
    ....0. .... = Syn: Not set
    ....0. .... = Fin: Not set
  [TCP Flags: .....A....]
  Window: 8434
  [Calculated window size: 2159104]
  [Window size scaling factor: 256]
  Checksum: 0xbfb4 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  [Timestamps]
  [SEQ/ACK analysis]
```

- 源端口与目的端口分别为服务器80与客户端52334。是由服务器发给客户端的。
- Seq与Ack，为134950与2090。
- Flags标志位为ACK，表示服务端收到了客户端的关闭连接请求，并回复确认关闭。

至此，连接断开。访问结束。

问题

1、遇到304状态码

396	13.197537	::1	::1	HTTP	874 GET /ComNet2/main.html HTTP/1.1
397	13.197577	::1	::1	TCP	64 80 → 57084 [ACK] Seq=1 Ack=811 Win=8436 Len=0
398	13.198588	::1	::1	HTTP	296 HTTP/1.1 304 Not Modified

因为进行了多次访问而没有清楚浏览器缓存。304状态码的含义是目前请求的信息与之前请求的内容相比没有改动，此时客户端从缓存读取即可，无须由服务器端再发送。

2、找不到 favicon.ico 文件

4595	26.698484	::1	::1	HTTP	709 GET /favicon.ico HTTP/1.1
4596	26.698519	::1	::1	TCP	64 80 → 62340 [ACK] Seq=133681 Ack=2623 Win=2157824 Len=0
4597	26.700427	::1	::1	HTTP	3066 HTTP/1.1 404 Not Found (text/html)

刚开始实验时抓到了一个GET失败的结果，上网查询一下是因为浏览器在访问时会默认请求一个网页图标文件，用于显示在标签栏上，但我的项目文件夹以及Apache的根目录中都没有这个图标文件。

于是从网上找了一张Apache的图标图片，用在线工具转为 .ioc 格式，并放在 \Apache2.4.39\htdocs 文件夹下，当项目根目录访问不到这个图标时，默认会去Apache的路径下访问，就可以访问到了。

