



南開大學
Nankai University

南 开 大 学

网络空间安全学院

计算机网络课程报告

第二次实验报告

学号：2011897

姓名：任薏霖

年级：2020 级

专业：物联网工程

2022 年 10 月 29 日

一、实验内容说明

（一）实验题目

实验 2: 配置 Web 服务器, 编写简单页面, 分析交互过程

（二）实验内容说明

1. 搭建 Web 服务器（自由选择系统），并制作简单的 Web 页面，包含简单文本信息（至少包含专业、学号、姓名）和自己的 LOGO。
2. 通过浏览器获取自己编写的 Web 页面，使用 Wireshark 捕获浏览器与 Web 服务器的交互过程，并进行简单的分析说明。
3. 提交实验报告。

二、实验准备

（一）配置 Web 服务器环境 --- Vscode

本实验使用 Vscode 的 Live Server 扩展来搭建代理服务器，让所访问域名、端口、路径相同。

1. 安装
2. 配置

安装完毕后需要对该扩展进行配置，其具体配置代码如下：

```
{  
    "verilog.ctags.path": "D:\\\\下载\\\\ctags\\\\ctags-p5.9.20221009.0-x64\\\\ctags.exe",  
    "verilog.linting.iverilog.arguments": "-i",  
    "workbench.colorTheme": "Visual Studio Light",  
    "workbench.iconTheme": "vscode-icons",  
    "vsicons.dontShowNewVersionMessage": true,  
    "code-runner.runInTerminal": true,  
    "open-in-browser.default": "edge",  
    "liveServer.settings.AdvanceCustomBrowserCmdLine": "",  
    "liveServer.settings.port": 9000,  
    "liveServer.settings.host": "localhost"  
}
```

三、实验过程

（一）制作 Web 页面

1. html 文件代码（部分）如下：

```

<body>
  <div class="content">
    
    <div class="content_r clearfix">
      <div class="content_l clearfix">
        <br>
        <br>
        <br>
        <h2>2011897 任意霖, Welcome here ! </h2>
        <br>
        <h3>计算机网络实验 2</h3>
        <br>
        <h3>实验要求</h3>
        <br>
        <p class="cc">(1) 搭建 Web 服务器 (自由选择系统) </p>
        <p class="cc">(2) 使用 Wireshark 捕获浏览器与 Web 服务器的
交互过程</p>
        <br>
      </div>
    </div>
  </div>
</body>

```

2. Web 页面如下:

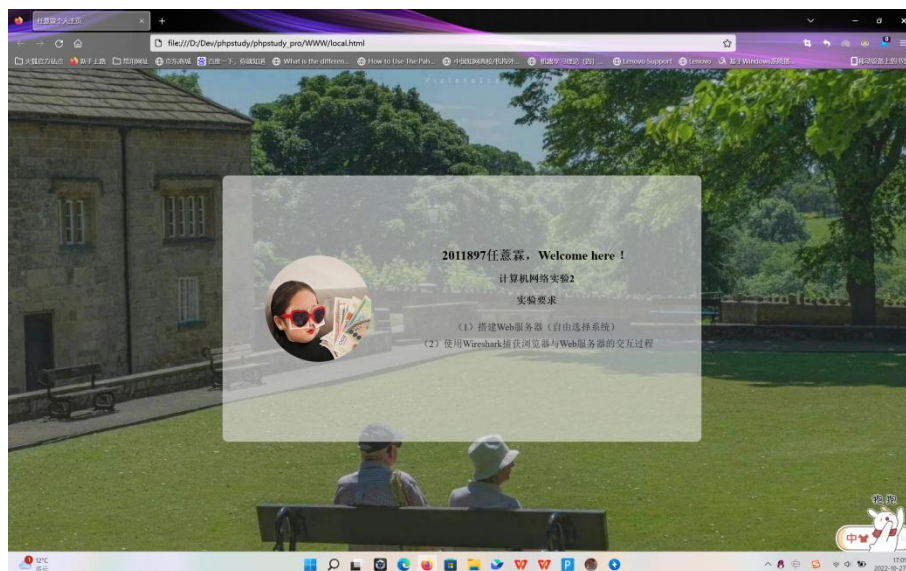


图 1 Web 服务器页面

(二) Wireshark 捕获交互过程

1. Wireshark+Npcap 实现抓包

由于本次实验中, 本机既作为客户端又作为服务器端, 使得本机自己和自己通信。但是

wireshark 此时是无法抓取到数据包的，需要通过安装 Npcap 对 Wireshark 进行配置。安装完成后启动 wireshark，可看到在网络接口列表中，多了一项 Adapter for loopback traffic capture，通过此接口来抓取本地回环包。

并在过滤器中输入：`tcp.port == 9000`，从而对本实验 Web 服务器进行抓包。

抓包结果如下图所示：

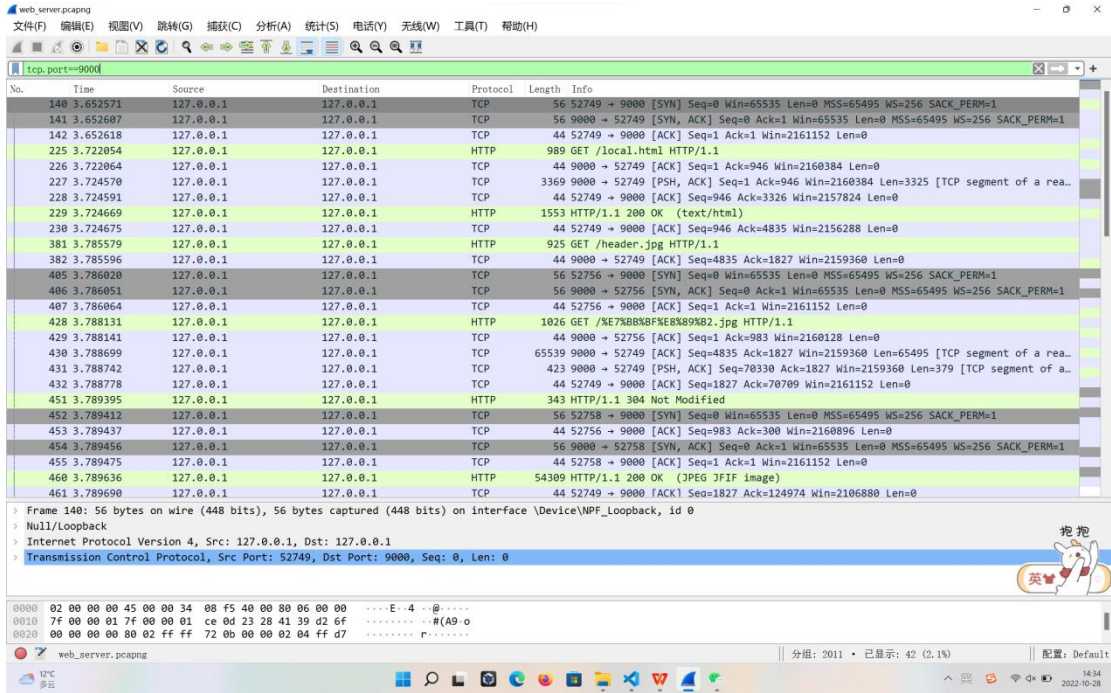


图 2 报文

2. 分析交互过程

(1) 分析中将会用到的 Flags:

Flags	意义
ACK	确认序号有效；
SYN	同步序号用来发起一个连接；
FIN	发端完成发送任务；

表 1 标志位

(2) 三次握手

① TCP 三次握手连接建立过程

Step1: 首先客户端向服务器发送一个 SYN 包，并等待服务器确认，这是第一次握手；

Step2: 服务器接收到客户端发来的 SYN 包后，对该包进行确认后结束 LISTEN 阶段，并返回一段 TCP 报文，这是第二次握手；

Step3: 客户端接收到发送的 SYN+ACK 包后, 明确了从客户端到服务器的数据传输是正常的, 从而结束 SYN-SENT 阶段, 并返回最后一段报文, 这就是第三次握手。

当服务器端收到来自客户端确认收到服务器数据的报文后, 得知从服务器到客户端的数据传输是正常的, 从而结束 SYN-RCVD 阶段, 进入 ESTABLISHED 阶段, 从而完成三次握手。

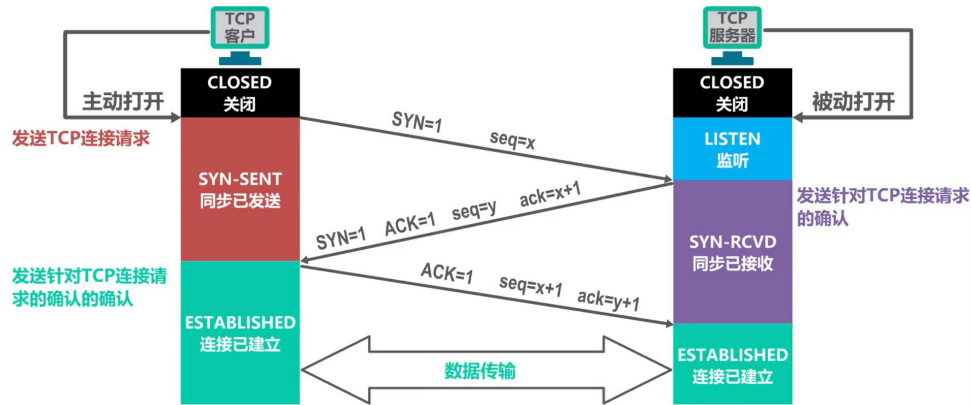


图 3 三次握手示意图

下图中可以看到 wireshark 截获到了三次握手的三个数据包, 而第四个包是 HTTP 的数据包, 这说明 HTTP 使用 TCP 建立连接。

No.	Time	Source	Destination	Protocol	Length	Info
140	3.652571	127.0.0.1	127.0.0.1	TCP	56	52749 → 9000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
141	3.652607	127.0.0.1	127.0.0.1	TCP	56	9000 → 52749 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
142	3.652618	127.0.0.1	127.0.0.1	TCP	44	52749 → 9000 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
225	3.722054	127.0.0.1	127.0.0.1	HTTP	989	GET /local.html HTTP/1.1

图 4 三次握手机报文

② 第一次握手数据包

客户端发送一个 TCP, 标志位为 SYN, 序列号为 0, 代表客户端请求建立连接。如下图:

140	3.652571	127.0.0.1	127.0.0.1	TCP	56	52749 → 9000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
141	3.652607	127.0.0.1	127.0.0.1	TCP	56	9000 → 52749 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
142	3.652618	127.0.0.1	127.0.0.1	TCP	44	52749 → 9000 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
225	3.722054	127.0.0.1	127.0.0.1	HTTP	989	GET /local.html HTTP/1.1

> Frame 140: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_{Loopback}, id 0 > Null/Loopback > Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1 > Transmission Control Protocol, Src Port: 52749, Dst Port: 9000, Seq: 0, Len: 0	
Source Port: 52749	Destination Port: 9000
[Stream index: 3]	
[Conversation completeness: Complete, WITH_DATA (31)]	
[TCP Segment Len: 0]	
Sequence Number: 0	(relative sequence number)
Sequence Number (raw): 1094308463	
[Next Sequence Number: 1	(relative sequence number)]
Acknowledgment Number: 0	
Acknowledgment number (raw): 0	
1000 ... = Header Length: 32 bytes (8)	
Flags: 0x002 (SYN)	
000. = Reserved: Not set	
...0 = Nonce: Not set	
...0... = Congestion Window Reduced (CWR): Not set	
...0... = ECN-Echo: Not set	
...0... = Urgent: Not set	
...0... = Acknowledgment: Not set	
...0... = Push: Not set	
...0... = Reset: Not set	
...1... = Syn: Set	

图 5 第一次握手机报文

数据包的关键属性如下:

SYN = 1: 标志位, 表示请求建立连接;

Seq = 0: 初始建立连接值为 0, 表示当前还没有发送数据;

Ack = 0: 初始建立连接值为 0, 已经收到包的数量, 表示当前没有接收到数据。

③ 第二次握手的数据包

TCP 服务器进程收到 TCP 连接请求报文段后, 如果同意建立连接, 则向 TCP 客户进程发送 TCP 连接请求确认报文段, 并进入同步已接收状态。如下图所示:

No.	Time	Source	Destination	Protocol	Length	Info
140	3.652571	127.0.0.1	127.0.0.1	TCP	56	52749 → 9000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
141	3.652607	127.0.0.1	127.0.0.1	TCP	56	9000 → 52749 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
142	3.652618	127.0.0.1	127.0.0.1	TCP	44	52749 → 9000 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
225	3.722054	127.0.0.1	127.0.0.1	HTTP	989	GET /local.html HTTP/1.1

> Frame 141: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_{...} id 0

> Null/Loopback

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

> Transmission Control Protocol, Src Port: 9000, Dst Port: 52749, Seq: 0, Ack: 1, Len: 0

Source Port: 9000

Destination Port: 52749

[Stream index: 3]

[Conversation completeness: Complete, WITH_DATA (31)]

[TCP Segment Len: 0]

Sequence Number: 0 (relative sequence number)

Sequence Number (raw): 3105468574

[Next Sequence Number: 1 (relative sequence number)]

Acknowledgment Number: 1 (relative ack number)

Acknowledgment number (raw): 1094308464

1000 ... = Header Length: 32 bytes (8)

Flags: 0x012 (SYN, ACK)

000. = Reserved: Not set

...0 = Nonce: Not set

...0... = Congestion Window Reduced (CWR): 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

图 6 第二次握手报文

数据包的关键属性如下:

报文段首部中的同步位 SYN = 1, ACK = 1: 表明这是一个 TCP 连接请求;

Seq=0: 初始建立值为 0, 表示当前还没有发送数据;

目标 IP 端进入获取数据状态。

④ 第三次握手的数据包

TCP 客户进程收到 TCP 连接请求确认报文段后, 还要向 TCP 服务器进程发送一个普通的 TCP 确认报文段并进入连接已建立状态。具体报文如下图所示:

No.	Time	Source	Destination	Protocol	Length	Info
140	3.652571	127.0.0.1	127.0.0.1	TCP	56	52749 → 9000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
141	3.652607	127.0.0.1	127.0.0.1	TCP	56	9000 → 52749 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
142	3.652618	127.0.0.1	127.0.0.1	TCP	44	52749 → 9000 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
225	3.722054	127.0.0.1	127.0.0.1	HTTP	989	GET /local.html HTTP/1.1

> Frame 142: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_{...} id 0

> Null/Loopback

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

> Transmission Control Protocol, Src Port: 52749, Dst Port: 9000, Seq: 1, Ack: 1, Len: 0

Source Port: 52749

Destination Port: 9000

[Stream index: 3]

[Conversation completeness: Complete, WITH_DATA (31)]

[TCP Segment Len: 0]

Sequence Number: 1 (relative sequence number)

Sequence Number (raw): 1094308464

[Next Sequence Number: 1 (relative sequence number)]

Acknowledgment Number: 1 (relative ack number)

Acknowledgment number (raw): 3105468575

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

Flags: 0x010 (ACK)

000. = Reserved: Not set

...0 = Nonce: Not set

...0... = Congestion Window Reduced (CWR): 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

图 7 第三次握手报文

源 IP 向目标 IP 发送一个确认报文，SEQ=1，ACK=1，SYN=0，三次握手结束。

(3) 发送与收取数据：浏览器与目的主机开始 HTTP 访问过程

① 客户端向服务器发送一个 SYN=0 ACK=1 acknumber=1 relativeSEQ=1 的数据包，请求服务器数据。其报文如下所示：

No.	Time	Source	Destination	Protocol	Length	Info
140	3.652571	127.0.0.1	127.0.0.1	TCP	56	52749 → 9000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
141	3.652607	127.0.0.1	127.0.0.1	TCP	56	9000 → 52749 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
142	3.652618	127.0.0.1	127.0.0.1	TCP	44	52749 → 9000 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
225	3.722054	127.0.0.1	127.0.0.1	HTTP	989	GET /local.html HTTP/1.1
226	3.722064	127.0.0.1	127.0.0.1	TCP	44	9000 → 52749 [ACK] Seq=1 Ack=946 Win=2160384 Len=0
227	3.724570	127.0.0.1	127.0.0.1	TCP	3369	9000 → 52749 [PSH, ACK] Seq=1 Ack=946 Win=2160384 Len=3325 [TCP segment of a reassembled
228	3.724591	127.0.0.1	127.0.0.1	TCP	44	52749 → 9000 [ACK] Seq=946 Ack=3326 Win=2157824 Len=0
229	3.724669	127.0.0.1	127.0.0.1	HTTP	1553	HTTP/1.1 200 OK (text/html)

Transmission Control Protocol, Src Port: 52749, Dst Port: 9000, Seq: 1, Ack: 1, Len: 945

Source Port: 52749
Destination Port: 9000
[Stream index: 3]
[Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 945]
Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 1094308464
[Next Sequence Number: 946 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 3105468575
0101 = Header Length: 20 bytes (5)
Flags: 0x018 (PSH, ACK)
0000 = Reserved: Not set
...0 = Nonce: Not set
....0... = Congestion Window Reduced (CWR): Not set
....0... = ECN-Echo: Not set
....0... = Urgent: Not set
...1... = Acknowledgment: Set
....1... = Push: Set
....0... = Reset: Not set
....0... = Syn: Not set
....0... = Fin: Not set

图 8 HTTP 协议报文

数据包的关键属性如下：

SYN = 0：表明不再需要请求连接；

ACK = 1：表明客户端回应服务器发来的请求；

Acknumber = 1：表明服务器只发送过长度为 1 的请求连接数据；

由于客户端只发送过长度为 1 的请求连接数据，所以 relativeSEQ=1，但此时要发送长度为 945 的请求数据包，这次是双方第一次发送具有真实含义的数据包。

② 获取服务器文字

HTTP/1.1 200 OK (text/html)

在 Line-based text data 中我们可以看到我们写的 html 源码，如下图所示：

229	3.724669	127.0.0.1	127.0.0.1	HTTP	1553	HTTP/1.1 200 OK (text/html)
230	3.724675	127.0.0.1	127.0.0.1	TCP	44	52749 → 9000 [ACK] Seq=946 Ack=4835 Win=2156288 Len=0

Frame 229: 1553 bytes on wire (12424 bits), 1553 bytes captured (12424 bits) on interface \Device\NPF_{Loopback}, id 0

Null/Loopback

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

Transmission Control Protocol, Src Port: 9000, Dst Port: 52749, Seq: 3326, Ack: 946, Len: 1509

[2 Reassembled TCP Segments (4834 bytes): #227(3325), #229(1509)]

Hypertext Transfer Protocol

Line-based text data: text/html (191 lines)

```
<!DOCTYPE html>\r\n<html lang="en">\r\n<head>\r\n  <meta charset="UTF-8">\r\n  <meta name="viewport" content="width=device-width, initial-scale=1.0">\r\n  <title>任慕霖个人主页</title>\r\n  <style type="text/css">\r\n    /* CSS Document */\r\n    {\r\n      margin: 0;\r\n      padding: 0;\r\n    }\r\n  \r\n
```

图 9 HTTP 获取服务器文字报文

③ 获取服务器图片

HTTP/1.1 200 OK (JPEG JFIF image)

在 JPEG File Interchange Format 中我们可以看到图片的十六进制编码，如下图所示：

460	3.789636	127.0.0.1	127.0.0.1	HTTP	54309 HTTP/1.1 200 OK (JPEG JFIF image)
461	3.789690	127.0.0.1	127.0.0.1	TCP	44 52749 → 9000 [ACK] Seq=1827 Ack=124974 Win=2106880 Len=0
464	3.789853	127.0.0.1	127.0.0.1	HTTP	1090 GET /local.html/ws HTTP/1.1
465	3.789862	127.0.0.1	127.0.0.1	TCP	44 9000 → 52758 [ACK] Seq=1 Ack=1047 Win=2160128 Len=0
472	3.790495	127.0.0.1	127.0.0.1	HTTP	173 HTTP/1.1 101 Switching Protocols

>	Frame 460: 54309 bytes on wire (434472 bits), 54309 bytes captured (434472 bits) on interface \Device\NPF_{Loopback}, id 0
>	Null/Loopback
>	Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
>	Transmission Control Protocol, Src Port: 9000, Dst Port: 52749, Seq: 70709, Ack: 1827, Len: 54265
>	[3 Reassembled TCP Segments (120139 bytes): #430(65495), #431(379), #460(54265)]
>	Hypertext Transfer Protocol
>	JPEG File Interchange Format
>	Marker: Start of Image (0xffd8)
>	Marker segment: Reserved for application segments - 0 (0xFFE0)
>	Marker segment: Reserved for application segments - 1 (0xFFE1)
>	Marker segment: Define quantization table(s) (0xFFDB)
>	Marker segment: Define quantization table(s) (0xFFDB)
>	Start of Frame header: Start of Frame (non-differential, Huffman coding) - Baseline DCT (0xFFC0)
>	Marker segment: Define Huffman table(s) (0xFFC4)
>	Marker segment: Define Huffman table(s) (0xFFC4)
>	Marker segment: Define Huffman table(s) (0xFFC4)
>	Marker segment: Define Huffman table(s) (0xFFC4)
>	Start of Segment header: Start of Scan (0xFFDA)
>	Entropy-coded segment (dissection is not yet implemented): f462b96079078c8ed4d0bc162bf9f1402db40e08ed9e314e05810369e791c5646a377a85...
>	Marker: End of Image (0xffd9)

图 10 HTTP 获取服务器图片报文

(4) 四次挥手

① TCP 四次挥手建立过程

Step1: 首先客户端向服务器发送一段 TCP 报文表明其想要释放 TCP 连接，随后客户端进入 FIN-WAIT-1 阶段，即半关闭阶段，并且停止向服务端发送通信数据。

Step2: 服务器接收到客户端请求断开连接的 FIN 报文后，结束 ESTABLISHED 阶段，进入 CLOSE-WAIT 阶段并返回一段 TCP 报文；

Step3: 服务器在发出 ACK 确认报文后，会将遗留的待传数据传送给客户端，待传输完成后即经过 CLOSE-WAIT 阶段，变准备释放服务器端到客户端，再次向客户端发出一段 TCP 报文；随后服务器接收 CLOSE-WAIT 阶段，进入 LAST-ACK 阶段，并停止向客户端发送数据；

Step4: 客户端收到从服务器发来的 TCP 报文，确认了服务器已经做好释放连接的准备，于是结束 FIN-WAIT-2 阶段，进入 TIME-WAIT 阶段，并向服务器发送一段报文。

服务器端收到从客户端发出的 TCP 报文之后结束 LAST-ACK 阶段，进入 CLOSED 阶段。由此正式确认关闭服务器端到客户端方向上的连接。客户端结束 TIME-WAIT 阶段，进入 CLOSED 阶段，由此完成四次挥手。

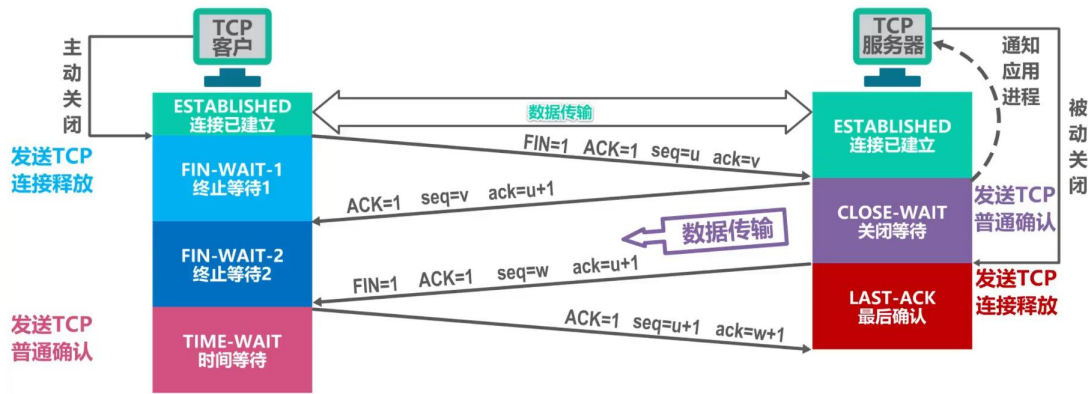


图 11 四次挥手示意图

② 第一次挥手报文分析

TCP 客户进程会发送 TCP 连接释放报文段，并进入终止等待 1(FIN-WAIT-1)状态。具体报文如下图所示：

554	3.812161	127.0.0.1	127.0.0.1	TCP	44	52756 → 9000	[FIN, ACK]	Seq=1865	Ack=300	Win=2160896	Len=0
555	3.812171	127.0.0.1	127.0.0.1	TCP	44	9000 → 52756	[ACK]	Seq=300	Ack=1866	Win=2159360	Len=0
558	3.812861	127.0.0.1	127.0.0.1	TCP	44	9000 → 52756	[FIN, ACK]	Seq=300	Ack=1866	Win=2159360	Len=0


```

Acknowledgment Number: 300 (relative ack number)
Acknowledgment number (raw): 188582523
0101 .... = Header Length: 20 bytes (5)
Flags: 0x011 (FIN, ACK)
 000. .... = Reserved: Not set
 ...0 .... = Nonce: Not set
 ...0 .... = Congestion Window Reduced (CWR): 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
  
```

图 12 第一次挥手报文

数据包的关键属性如下：

FIN = 1：表明这是一个 TCP 连接释放报文段；

ACK = 1：表明对之前收到的报文段进行确认；

③ 第二次挥手报文分析

554	3.812161	127.0.0.1	127.0.0.1	TCP	44	52756 → 9000	[FIN, ACK]	Seq=1865	Ack=300	Win=2160896	Len=0
555	3.812171	127.0.0.1	127.0.0.1	TCP	44	9000 → 52756	[ACK]	Seq=300	Ack=1866	Win=2159360	Len=0
558	3.812861	127.0.0.1	127.0.0.1	TCP	44	9000 → 52756	[FIN, ACK]	Seq=300	Ack=1866	Win=2159360	Len=0


```

Acknowledgment Number: 1866 (relative ack number)
Acknowledgment number (raw): 2964279990
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
 000. .... = Reserved: Not set
 ...0 .... = Nonce: Not set
 ...0 .... = Congestion Window Reduced (CWR): 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
  
```

图 13 第二次挥手报文

数据包的关键属性如下：

FIN = 0：表明在等待所有数据传输完成才能中断连接；

ACK = 1: 表明回应客户端发来的请求

④ 第三次挥手报文分析

554	3.812161	127.0.0.1	127.0.0.1	TCP	44	52756 → 9000 [FIN, ACK] Seq=1865 Ack=300 Win=2160896 Len=0
555	3.812171	127.0.0.1	127.0.0.1	TCP	44	9000 → 52756 [ACK] Seq=300 Ack=1866 Win=2159360 Len=0
558	3.812861	127.0.0.1	127.0.0.1	TCP	44	9000 → 52756 [FIN, ACK] Seq=300 Ack=1866 Win=2159360 Len=0
559	3.812883	127.0.0.1	127.0.0.1	TCP	44	52756 → 9000 [ACK] Seq=1866 Ack=301 Win=2160896 Len=0
708	3.892914	127.0.0.1	127.0.0.1	WebSock...	55	WebSocket Text [FIN]
709	3.892925	127.0.0.1	127.0.0.1	TCP	44	52758 → 9000 [ACK] Seq=1047 Ack=141 Win=2161152 Len=0
1883	8.795440	127.0.0.1	127.0.0.1	TCP	44	9000 → 52749 [FIN, ACK] Seq=124974 Ack=1827 Win=2159360 Len=0
1884	8.795466	127.0.0.1	127.0.0.1	TCP	44	52749 → 9000 [ACK] Seq=1827 Ack=124975 Win=2106880 Len=0
1885	8.795538	127.0.0.1	127.0.0.1	TCP	44	52749 → 9000 [FIN, ACK] Seq=1827 Ack=124975 Win=2106880 Len=0
1886	8.795555	127.0.0.1	127.0.0.1	TCP	44	9000 → 52749 [ACK] Seq=124975 Ack=1828 Win=2159360 Len=0

Flags: 0x011 (FIN, ACK)	
000. = Reserved: Not set
...0 = Nonce: Not set
....0 = Congestion Window Reduced (CWR): 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
[Expert Info (Chat/Sequence): Connection finish (FIN)]	
[TCP Flags:A...F]	

图 14 第三次挥手报文

数据包的关键属性如下:

FIN = 1: 表明服务器认为可以中断连接;

ACK = 1: 表明回应客户端发来的请求

⑤ 第四次挥手报文分析

554	3.812161	127.0.0.1	127.0.0.1	TCP	44	52756 → 9000 [FIN, ACK] Seq=1865 Ack=300 Win=2160896 Len=0
555	3.812171	127.0.0.1	127.0.0.1	TCP	44	9000 → 52756 [ACK] Seq=300 Ack=1866 Win=2159360 Len=0
558	3.812861	127.0.0.1	127.0.0.1	TCP	44	9000 → 52756 [FIN, ACK] Seq=300 Ack=1866 Win=2159360 Len=0
559	3.812883	127.0.0.1	127.0.0.1	TCP	44	52756 → 9000 [ACK] Seq=1866 Ack=301 Win=2160896 Len=0
708	3.892914	127.0.0.1	127.0.0.1	WebSock...	55	WebSocket Text [FIN]
709	3.892925	127.0.0.1	127.0.0.1	TCP	44	52758 → 9000 [ACK] Seq=1047 Ack=141 Win=2161152 Len=0
1883	8.795440	127.0.0.1	127.0.0.1	TCP	44	9000 → 52749 [FIN, ACK] Seq=124974 Ack=1827 Win=2159360 Len=0
1884	8.795466	127.0.0.1	127.0.0.1	TCP	44	52749 → 9000 [ACK] Seq=1827 Ack=124975 Win=2106880 Len=0
1885	8.795538	127.0.0.1	127.0.0.1	TCP	44	52749 → 9000 [FIN, ACK] Seq=1827 Ack=124975 Win=2106880 Len=0
1886	8.795555	127.0.0.1	127.0.0.1	TCP	44	9000 → 52749 [ACK] Seq=124975 Ack=1828 Win=2159360 Len=0

Acknowledgment Number: 301 (relative ack number)	
Acknowledgment number (raw): 188582524	
0101 = Header Length: 20 bytes (5)	
Flags: 0x010 (ACK)	
000. = Reserved: Not set
...0 = Nonce: Not set
....0 = Congestion Window Reduced (CWR): 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

图 15 第四次挥手报文

数据包的关键属性如下:

FIN = 0: 表明不需要再次终止连接;

ACK = 1: 表明回应服务器发来的请求

四、实验总结

- 通过本次实验实现了 Web 服务器的设计, 并使用 html 进行优化;
- 通过使用 wireshark 进行抓包分析, 更加细致地了解客户端与服务端的通信过程; 通过对报文的分析, 加深了对三次握手、四次挥手等相关知识的理解。