计算机网络：配置Web页面

2011360

牟迪

一.实验要求

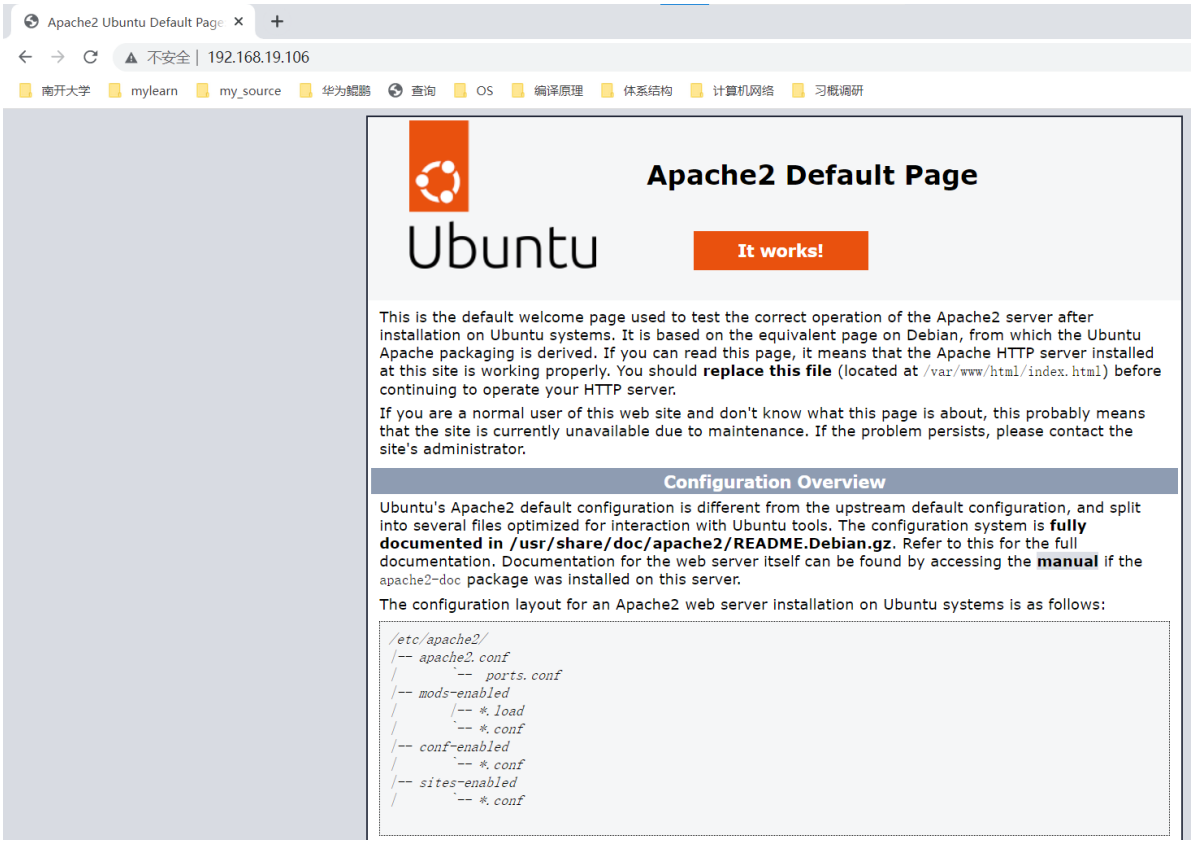
1. 搭建Web服务器，并制作简单的web页面，包含简单文本信息和自己的logo
2. 通过浏览器获取自己编写的web页面，使用wireshark捕获浏览器与web服务器的交互过程，并进行简单的分析说明
3. 提交实验报告

二.服务器搭建

本次实验通过Ubuntu虚拟机来作为要访问的目标服务器，在Ubuntu虚拟机中配置apache与php服务，在虚拟机服务器中查看虚拟机的虚拟网络ip地址，通过命令ifconfig可得：

```
kid@kid-virtual-machine:~/Desktop$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.19.106  netmask 255.255.255.0  broadcast 192.168.19.255
        inet6 fe80::31e9:698:df21:b6ce  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:5c:01:c0  txqueuelen 1000  (Ethernet)
        RX packets 1568  bytes 136981 (136.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 954  bytes 499184 (499.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 265  bytes 26055 (26.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 265  bytes 26055 (26.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

其ip地址为192.168.19.106，对于其Web页面的访问，默认端口号为80端口。通过访问192.168.19.106:80进入apache默认页面：



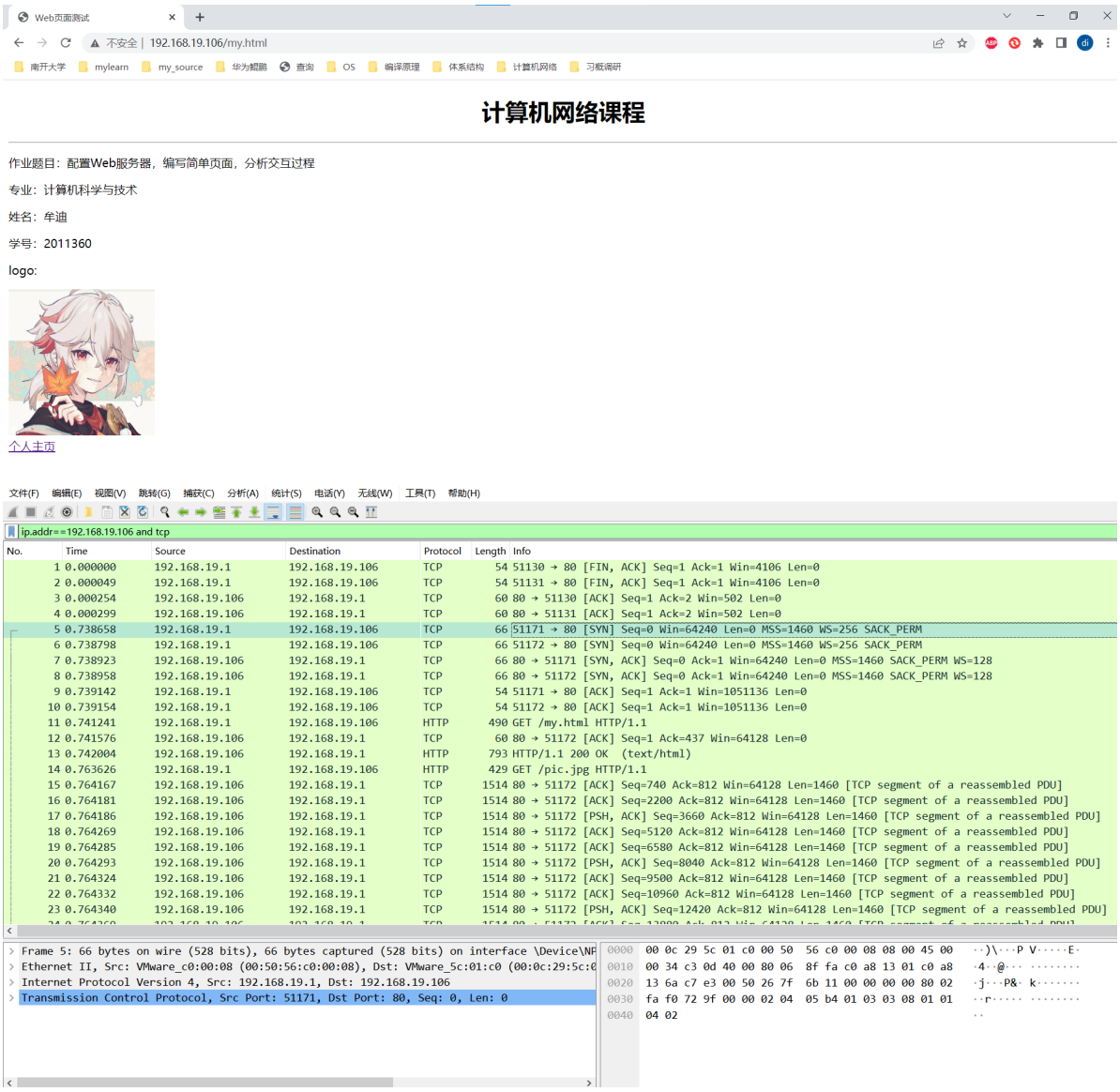
在apache服务下，可供访问的html资源存储在/var/www/html路径下，在对应路径下存储html文件与媒体资源，本次中用于测试页面文本如下：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title >Web页面测试</title>
</head>
<body>
        <h1 style="text-align:center;">计算机网络课程</h1>
        <hr>
        <p>作业题目：配置web服务器，编写简单页面，分析交互过程</p>
        <p>专业：计算机科学与技术</p>
        <p>姓名：牟迪</p>
        <p>学号：2011360</p>
        <p>logo:</p><img src="/pic.jpg" width="200" height="200">
        <br>
        <a href="https://github.com/KIDSSCC">个人主页</a>
</body>
</html>
```

三.交互过程分析

```
kid@kid-virtual-machine:~/Desktop$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.19.106  netmask 255.255.255.0  broadcast 192.168.19.255
        inet6 fe80::31e9:698:df21:b6ce  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:5c:01:c0  txqueuelen 1000  (Ethernet)
        RX packets 1568  bytes 136981 (136.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 954  bytes 499184 (499.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 265  bytes 26055 (26.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 265  bytes 26055 (26.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

其ip地址为192.168.19.106，对于其Web页面的访问，默认端口号为80端口。开启wireshark抓包过程，通过访问192.168.19.106:80/my.html即可访问预设的html页面。并获取捕获的TCP包的数据

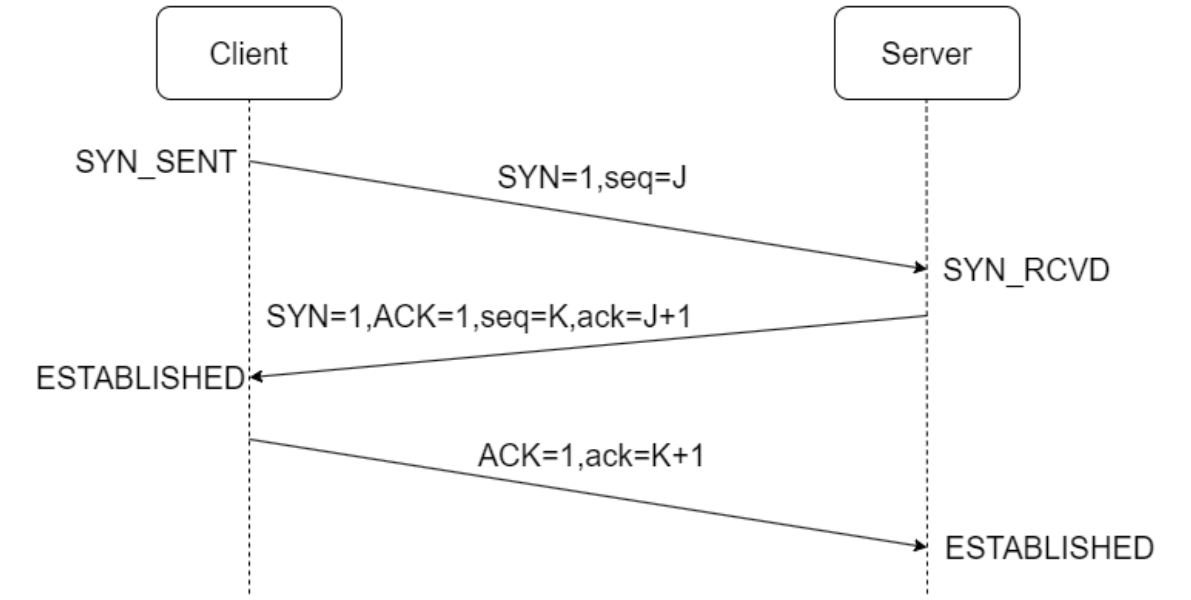


通过捕获到的数据包可发现，前四个包为上一次挥手过程结束时，由客户端（本机）端口51130和51131发起的挥手过程，服务端给予ack响应。由序号5开始，建立本次新的连接握手过程。通过wireshark的抓包结果可以发现，在本机端有两个端口与服务端建立的TCP的连接，进行了三次握手。这是因为在本次所采用的超文本传输协议为HTTP/1.1，为了防止头阻塞，会采用双端口进行连接，即51171和51172两个端口。这同时提高了传输的并发性。本次重点分析端口51172与服务端的传送过程。

wireshark捕获到的数据包信息共分为了四个部分，由上至下分别为物理帧，以太网关，IP包和TCP包。其中物理帧就是传输时的完整数据。在以太网关中，注明了连接的网关信息。在IP包中，包含的源ip和目的ip两个重要的ip地址信息，而TCP包即重点要分析的对象。

三次握手

TCP的三次握手过程：



No.	Time	Source	Destination	Protocol	Length	Info
6	0.738798	192.168.19.1	192.168.19.106	TCP	66	51172 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM

第一次握手，包序号为6，由客户端51172端口向服务端80端口发起，将标志位SYN置1，代表尝试建立连接，并产生一个随机值seq，将包发送给服务端。来看具体的报文信息：

```
> Frame 6: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{6A249A48-DC65-4A44-91D7-B35C3B3A43EB}, id 0
> Ethernet II, Src: VMware_c0:00:08 (00:50:56:c0:00:08), Dst: VMware_5c:01:c0 (00:0c:29:5c:01:c0)
> Internet Protocol Version 4, Src: 192.168.19.1, Dst: 192.168.19.106
v Transmission Control Protocol, Src Port: 51172, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 51172
    Destination Port: 80
    [Stream index: 3]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 913476523
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x002 (SYN)
    Window: 64240
    [Calculated window size: 64240]
0000  00 0c 29 5c 01 c0 00 50  56 c0 00 08 08 00 45 00   ··)\···P V·····E·
0010  00 34 c3 0e 40 00 80 06  8f f9 c0 a8 13 01 c0 a8   ·4··@··· ········
0020  13 6a c7 e4 00 50 36 72  8b ab 00 00 00 00 80 02   ·j···P6r ········
0030  fa f0 42 11 00 00 02 04  05 b4 01 03 03 08 01 01   ··B····· ········
0040  04 02                                             ··
```

通过wireshark查看完整的报文信息，由c7e4这两个字节开始，根据TCP报文的格式，首部的**四个字节**，c7e4和0050分别代表了**源端口号51172和目的端口号80**.随后**四个字节**为**序列号Seq**，此处真实的初始序号为36728bab即10进制下的913476523，相对的初始序号为0.随后的**四个字节**代表**确认序号**，此处为0.代表了在ACK位有效的情况下，期望对方的下一个数据包确认序号为0+1=1.确认序号后为**4位数据偏移**，即代表了TCP包头部的长度，即字节80中的8，而**80字节中的0与02字节中的0中的两个比特，共6个比特，作为保留位**，供以后应用。02字节中剩余的**6个比特**，在此处为000010，即**6位标志位**，此处10代表了标志位SYN被置为了1.随后的**2个字节**代表**滑动窗口**的大小，为64240.随后的**四个字节**分别为**检验和**与**紧急指针**，剩余部分为**选项与填充**，可发现其大小为**12字节**，截至此处，TCP报文的头部共有2（源端口）+2（目的端口）+4（初始序号）+4（确认序号）+2（偏移+保留位+长度）+2（窗口）+4（检验和与指针）+12（选项与填充），共32个字节，数据偏移处以4字节为单位，共32字节，与实际情况相对应。

<table>
<tr><td colspan="4">源端口号</td><td colspan="4">目的端口号</td></tr>
<tr><td colspan="8">初始序号Seq</td></tr>
<tr><td colspan="8">确认序号ack</td></tr>
<tr><td>偏移</td><td colspan="2">保留位</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="2">滑动窗口</td></tr>
<tr><td colspan="5">校验和</td><td colspan="3">紧急指针</td></tr>
<tr><td colspan="8">选项与填充</td></tr>
<tr><td colspan="8">数据</td></tr>
</table>

第二次握手，包序号为8，由服务端80端口发向客户端51172端口，在建连过程中，为保证双向的连通。需要服务端，客户端各自给出连接请求，并各自给出回应。即4次握手，但对于服务端来说，其对于客户端建连的响应ACK与自身建连的SYN可通过同一数据包进行发送，因此，可减少以此数据的传输，最终为三次握手。在此刻，服务端将**标志位ACK**置为1.同时根据收到的客户端的ack值0，将自身的**ack置为1**，同时也生成随机的序号Seq，在此基础上，再将自身的**标志位SYN**置位1。具体的报文分析与第一次握手相似。

```
> Internet Protocol Version 4, Src: 192.168.19.106, Dst: 192.168.19.1
v Transmission Control Protocol, Src Port: 80, Dst Port: 51172, Seq: 0, Ack: 1, Len: 0
    Source Port: 80
    Destination Port: 51172
    [Stream index: 3]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 765303156
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 913476524
    1000 .... = Header Length: 32 bytes (8)
  v 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: 64240
```

第三次握手过程，包序号为10，客户端对于服务器的SYN给予ACK响应，与第二次握手过程类似。值得注意的是，第三次握手与前两次握手相比，长度减少了12字节，这12字节是属于选项与填充的部分。通过三次握手过程，确认服务端与客户端均可以进行正常的数据接收与数据发送，可以进行后续的数据传输了。

```
> Internet Protocol Version 4, Src: 192.168.19.1, Dst: 192.168.19.106
v Transmission Control Protocol, Src Port: 51172, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
    Source Port: 51172
    Destination Port: 80
    [Stream index: 3]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 913476524
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 765303157
    0101 .... = Header Length: 20 bytes (5)
  v 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: 4106
    [Calculated window size: 1051136]
```

数据传输

包序号11，客户端请求访问一个html页面，向服务端发出了get请求。通过追踪HTTP流的信息，可得到客户端完整的请求信息：

```
GET /my.html HTTP/1.1
Host: 192.168.19.106
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 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
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
```

从中可以得到超文本传输协议的版本为HTTP/1.1，当前请求的页面为my.html，以及客户端自身的一些接收声明，诸如格式，编码等信息。服务端接收后回复以ACK予以确认，并通过HTTP协议开始传输对应的文本信息。

```
11 0.741241    192.168.19.1      192.168.19.106    HTTP     490 GET /my.html HTTP/1.1
12 0.741576    192.168.19.106    192.168.19.1      TCP       60 80 → 51172 [ACK] Seq=1 Ack=437 Win=64128 Len=0
13 0.742004    192.168.19.106    192.168.19.1      HTTP     793 HTTP/1.1 200 OK  (text/html)
```

同样对包13进行HTTP流的跟踪，得到其传输的信息：

```
HTTP/1.1 200 OK
Date: Fri, 28 Oct 2022 03:15:32 GMT
Server: Apache/2.4.52 (Ubuntu)
Last-Modified: Tue, 25 Oct 2022 06:03:48 GMT
ETag: "1dc-5ebd5abefd552-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 402
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title >Web页面测试</title>
</head>
<body>
          <h1 style="text-align:center;">计算机网络课程</h1>
          <hr>
          <p>作业题目：配置Web服务器，编写简单页面，分析交互过程</p>
          <p>专业：计算机科学与技术</p>
          <p>姓名：牟迪</p>
          <p>学号：2011360</p>
          <p>logo:</p><img src="/pic.jpg" width="200" height="200">
          <br>
          <a href="https://github.com/KIDSSCC">个人主页</a>
</body>
</html>
```

可得到首部为相关的传输信息，诸如内容类型text/html等，下部则为具体的数据信息，即对应的html页面，通过浏览器对齐进行解析即可进行网页的显示。

在获取到的页面中，包含一个多媒体资源，pic.jpg，该资源并未随文本信息一齐发送，因此，客户端再次向服务端发起get请求，请求pic.jpg资源。

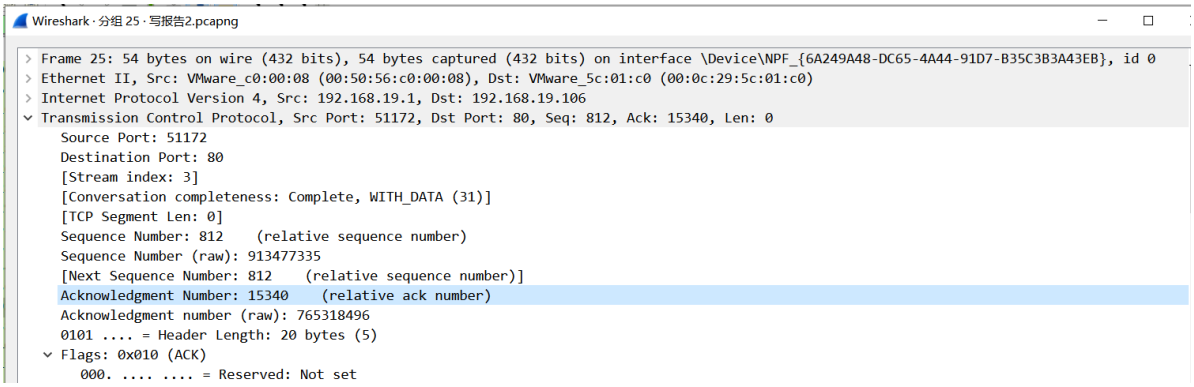
```
GET /pic.jpg HTTP/1.1
Host: 192.168.19.106
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/
537.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://192.168.19.106/my.html
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
```

14 0.763626	192.168.19.1	192.168.19.106	HTTP	429	GET /pic.jpg HTTP/1.1
15 0.764167	192.168.19.106	192.168.19.1	TCP	1514	80 → 51172 [ACK] Seq=740
16 0.764181	192.168.19.106	192.168.19.1	TCP	1514	80 → 51172 [ACK] Seq=220
17 0.764186	192.168.19.106	192.168.19.1	TCP	1514	80 → 51172 [PSH, ACK] Se
18 0.764269	192.168.19.106	192.168.19.1	TCP	1514	80 → 51172 [ACK] Seq=512
19 0.764285	192.168.19.106	192.168.19.1	TCP	1514	80 → 51172 [ACK] Seq=658
20 0.764293	192.168.19.106	192.168.19.1	TCP	1514	80 → 51172 [PSH, ACK] Se
21 0.764324	192.168.19.106	192.168.19.1	TCP	1514	80 → 51172 [ACK] Seq=950
22 0.764332	192.168.19.106	192.168.19.1	TCP	1514	80 → 51172 [ACK] Seq=109

图片资源较大，因此采取分组策略，将其分为不同的数据包进行发送，自包序号15起至包序号73间，大部分数据包均是由服务端发往客户端的图片资源的分组。其TCP部分长度固定.初始序号由740开始，每次发送长度为1460的部分，

16 0.764181	192.168.19.106	192.168.19.1	TCP	1514	80 → 51172 [ACK] Seq=2200 Ack=812 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
17 0.764186	192.168.19.106	192.168.19.1	TCP	1514	80 → 51172 [PSH, ACK] Seq=3660 Ack=812 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
18 0.764269	192.168.19.106	192.168.19.1	TCP	1514	80 → 51172 [ACK] Seq=5120 Ack=812 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
19 0.764285	192.168.19.106	192.168.19.1	TCP	1514	80 → 51172 [ACK] Seq=6580 Ack=812 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
20 0.764293	192.168.19.106	192.168.19.1	TCP	1514	80 → 51172 [PSH, ACK] Seq=8040 Ack=812 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
21 0.764324	192.168.19.106	192.168.19.1	TCP	1514	80 → 51172 [ACK] Seq=9500 Ack=812 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
22 0.764332	192.168.19.106	192.168.19.1	TCP	1514	80 → 51172 [ACK] Seq=10960 Ack=812 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
23 0.764340	192.168.19.106	192.168.19.1	TCP	1514	80 → 51172 [PSH, ACK] Seq=12420 Ack=812 Win=64128 Len=1460 [TCP segment of a reassembled PDU]
24 0.764369	192.168.19.106	192.168.19.1	TCP	1514	80 → 51172 [ACK] Seq=13880 Ack=812 Win=64128 Len=1460 [TCP segment of a reassembled PDU]

在服务端的发送过程中，客户端也会进行一定的响应。在包序号25处，客户端进行了第一次响应，其Seq为与服务端ack相同的812，ack值为15340，代表截至序号为15340的部分都已接收，从而带动服务端数据窗进行滑动。



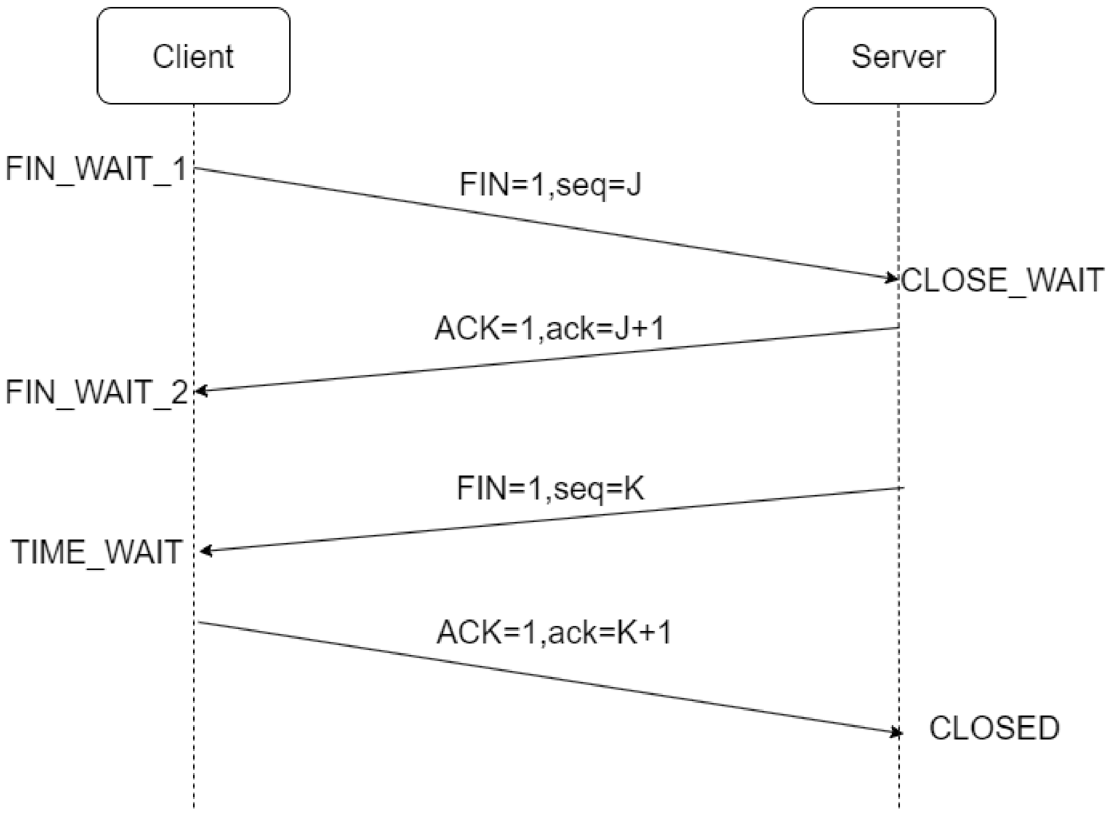
在包序号73处，服务端发送完成，此时通过对HTTP流进行追踪，可得到完整的发送数据，服务端也进行了响应，最终返回包，其ack值为81853，代表已全部接收。

73 0.766487	192.168.19.106	192.168.19.1	HTTP	867	HTTP/1.1 200 OK (JPEG JFIF image)
74 0.766998	192.168.19.1	192.168.19.106	TCP	54	51172 → 80 [ACK] Seq=812 Ack=59140 Win=1051136 Len=0
75 0.767202	192.168.19.1	192.168.19.106	TCP	54	51172 → 80 [ACK] Seq=812 Ack=73740 Win=1048064 Len=0
76 0.767357	192.168.19.1	192.168.19.106	TCP	54	51172 → 80 [ACK] Seq=812 Ack=81853 Win=1040128 Len=0

```
HTTP/1.1 200 OK
Date: Fri, 28 Oct 2022 03:15:32 GMT
Server: Apache/2.4.52 (Ubuntu)
Last-Modified: Tue, 25 Oct 2022 05:49:18 GMT
ETag: "13bb8-5ebd5781543b4"
Accept-Ranges: bytes
Content-Length: 80824
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: image/jpeg

......JFIF.............C.
.....
....

.....

......#.%$"."!&+7/&)4)!"0A149;>>>%.DIC<H7=>;...C.
...
.....;("(;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;......I.H.."...................................
....................}........!1A..Qa."q.2....#B...R..$3br.
.....
%&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz...........................................................................
.............................
....................w.......!1..AQ.aq."2...B....         #3R..br.
.$4.
%.....&'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz.......................................................................
..............?........4..+.?.J<...Vab<.4g.Ry_.4yG..4....<.G.}h...GJ..Oo..
$....b>(..........ZA....T.Q.J.G'...;.T.Q..?..Q.,2..'.i|.E.."......Q.......I..Q..Qp..(.I.}(...\,G.
```


四次挥手

TCP四次挥手过程



在完成传输后，tcp将通过四次挥手来断开连接。

第一次挥手，包序号78，由服务端发起，将标志位FIN置1，此时Seq为81853。客户端收到标志位FIN生效的数据包后，回复ACK予以确认，回复的ack值为seq+1即81854.两次挥手。

78	5.766743	192.168.19.106	192.168.19.1	TCP	60	80 → 51172 [FIN, ACK] Seq=81853 Ack=812 Win=64128 Len=0
79	5.766916	192.168.19.1	192.168.19.106	TCP	54	51172 → 80 [ACK] Seq=812 Ack=81854 Win=1051136 Len=0
138	50.767168	192.168.19.1	192.168.19.106	TCP	55	[TCP Keep-Alive] 51172 → 80 [ACK] Seq=811 Ack=81854 Win=1051136 Len=1
139	50.767360	192.168.19.106	192.168.19.1	TCP	60	[TCP Keep-Alive ACK] 80 → 51172 [ACK] Seq=81854 Ack=812 Win=64128 Len=0
147	55.278689	192.168.19.1	192.168.19.106	TCP	54	51172 → 80 [FIN, ACK] Seq=812 Ack=81854 Win=1051136 Len=0
149	55.278891	192.168.19.106	192.168.19.1	TCP	60	80 → 51172 [ACK] Seq=81854 Ack=813 Win=64128 Len=0

在达到最大响应时间后，由客户端发起第三次挥手，与第一次挥手相似，Seq为812。由服务器予以确认，回复的ack为813.通过四次挥手，断开连接。

四.实验总结

在本次实验中，掌握了通过wireshark工具进行网络抓包的方法。同时，通过对与TCP报文的逐步分析，加深了对于TCP协议的理解。