计算机网络实验

实验 2：配置 Web 服务器，编写简单页面，分析交互过程

姓名： 蒋浩南　　学号： 2012948

一、实验要求

（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、实验过程

（一）服务器搭建、网页编写

1、在 ubuntu（虚拟机）上通过 sudo apt-get apache2 安装 apache。虚拟机的 ip 地址为 192.168.62.131。在本机访问该 ip 地址，显示 apache 的正确访问网页，证明服务器运行成功。

2、编写 web 网页，在/var/www 目录下新建 lab 文件夹，在该文件夹下放置编写好的 lab.html 文件和网页所需的图片。

3、在本机访问 http://192.168.62.131/lab/lab.html，网页正确显示。该阶段工作完成。

（二）Wireshark 捕获交互过程

1、打开 wireshark，选择虚拟机连接的捕获过滤器。

2、选择条件为 ip.addr ==192.168.62.131。

3、在本机浏览器访问 http://192.168.62.131/lab/lab.html，。

4、在 wireshark 可以查看交互过程，保存捕获文件。

三、交互过程分析

（一）TCP 三次握手

```
  7 3.046627     192.168.62.1      192.168.62.131    TCP      66 1428 → 80 [SYN] Seq=0 Win=64240 L
  8 3.046867     192.168.62.131    192.168.62.1      TCP      66 80 → 1428 [SYN, ACK] Seq=0 Ack=1
  9 3.046950     192.168.62.1      192.168.62.131    TCP      54 1428 → 80 [ACK] Seq=1 Ack=1 Win=1
```

1、第一次握手：由客户端向服务器发送 tcp，标志位为 SYN，序列号为 0，代表客户端请求建立连接（同步）。

```
Transmission Control Protocol, Src Port: 1428, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 1428
    Destination Port: 80
    [Stream index: 0]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 3817147656
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x002 (SYN)
```

```
  7 3.046627     192.168.62.1      192.168.62.131    TCP      66 1428 → 80 [SYN] Seq=0 Win=64240 L
```

相对序列号为 0，Sequence Number (raw)：3817147656。（seq=X）。初始化序列号。

2、第二次握手：服务器发回确认包，标志位为 SYN，ACK。

```
Transmission Control Protocol, Src Port: 80, Dst Port: 1428, Seq: 0, Ack: 1, Len: 0
    Source Port: 80
    Destination Port: 1428
    [Stream index: 0]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 0    (relative sequence number)
    Sequence Number (raw): 1392149513
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 3817147657
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x012 (SYN, ACK)
```

```
  8 3.046867     192.168.62.131    192.168.62.1      TCP      66 80 → 1428 [SYN, ACK] Seq=0 Ack=1
```

（1）设置确定序列号为 x+1（0+1=1），实际为 3817147675+1=3817147677。

（2）seq=y，Sequence Number (raw)：1392149513

3、第三次握手：客户端发送确认包，标志位为 ACK。

```
Transmission Control Protocol, Src Port: 1428, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
    Source Port: 1428
    Destination Port: 80
    [Stream index: 0]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 3817147657
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1    (relative ack number)
    Acknowledgment number (raw): 1392149514
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
```

```
  9 3.046950      192.168.62.1        192.168.62.131      TCP        54 1428 → 80 [ACK] Seq=1 Ack=1 Win=1
```

（1）ACK=Y+1，即将上一包的 y+1。即 0+1=1，实际为 1392149513+1=1392149514

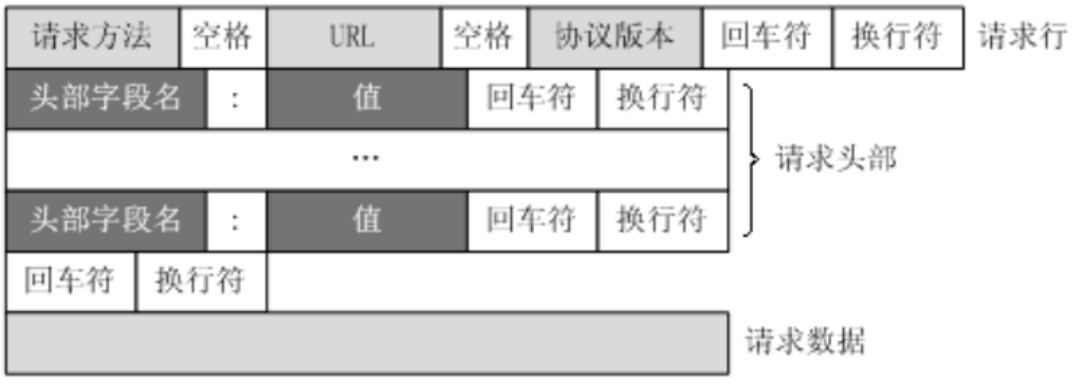
（2）seq 为上一包的 ack，即 3817147677。

（二）http 请求应答

```
10 3.055442      192.168.62.1    192.168.62.131  HTTP     625 GET /lab/lab.html HTTP/1.1
11 3.055660      192.168.62.131  192.168.62.1    TCP       60 80 → 1428 [ACK] Seq=1 Ack=572 Win=64128 Len=0
12 3.057804      192.168.62.131  192.168.62.1    HTTP     543 HTTP/1.1 200 OK  (text/html)
13 3.108804      192.168.62.1    192.168.62.131  TCP       54 1428 → 80 [ACK] Seq=572 Ack=490 Win=130816 Len=0
14 3.138173      192.168.62.1    192.168.62.131  HTTP     570 GET /lab/picture.jpg HTTP/1.1
15 3.138651      192.168.62.131  192.168.62.1    HTTP     304 HTTP/1.1 304 Not Modified
16 3.185832      192.168.62.1    192.168.62.131  TCP       54 1428 → 80 [ACK] Seq=1088 Ack=740 Win=130560 Len=0
17 3.268251      192.168.62.1    192.168.62.131  HTTP     478 GET /favicon.ico HTTP/1.1
18 3.268676      192.168.62.131  192.168.62.1    HTTP     546 HTTP/1.1 404 Not Found  (text/html)
19 3.309355      192.168.62.1    192.168.62.131  TCP       54 1428 → 80 [ACK] Seq=1512 Ack=1232 Win=130048 Len=0
```

1、浏览器向服务器发送 GET 请求

请求报文结构：



报文内容：

```
Hypertext Transfer Protocol
    GET /lab/lab.html HTTP/1.1\r\n
    Host: 192.168.62.131\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    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
Edg/106.0.1370.52\r\n
    Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,imag
e/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-
US;q=0.6\r\n
    If-None-Match: "b4-5ebda871e30db-gzip"\r\n
    If-Modified-Since: Tue, 25 Oct 2022 11:51:25 GMT\r\n
    \r\n
    [Full request URI: http://192.168.62.131/lab/lab.html]
    [HTTP request 1/3]
    [Response in frame: 12]
    [Next request in frame: 14]
```

请求行：
请求方法为 GET，URL 为/lab/lab.html，协议版本为 HTTP/1.1。
请求头：
Host 为请求的主机名
Connection 客户端与服务端指定的请求，响应有关选项
Upgrade-Insecure-Requests:1 让浏览器自动升级请求从 http 到 https
User-Agent 为发送请求的操作系统、及浏览器信息
Accept 为客户端可识别的内容类型列表，用于指定客户端接受哪些类型的信息
Accept-Encoding 为客户端可识别的数据编码
Accept-language 为浏览器所支持的语言类型

查看 tcp，可知[TCP Segment Len: 571]。

```
Transmission Control Protocol, Src Port: 1428, Dst Port: 80, Seq: 1, Ack: 1, Len: 571
  Source Port: 1428
  Destination Port: 80
  [Stream index: 0]
  [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 571]
  Sequence Number: 1    (relative sequence number)
  Sequence Number (raw): 3817147657
  [Next Sequence Number: 572    (relative sequence number)]
  Acknowledgment Number: 1    (relative ack number)
  Acknowledgment number (raw): 1392149514
  0101 .... = Header Length: 20 bytes (5)
> Flags: 0x018 (PSH, ACK)
```

之后服务器发送确定报文，

```
60 80 → 1428 [ACK] Seq=1 Ack=572 Win=64128 Len=0
```

SEQ 为客户端 GET 请求的 ACK。

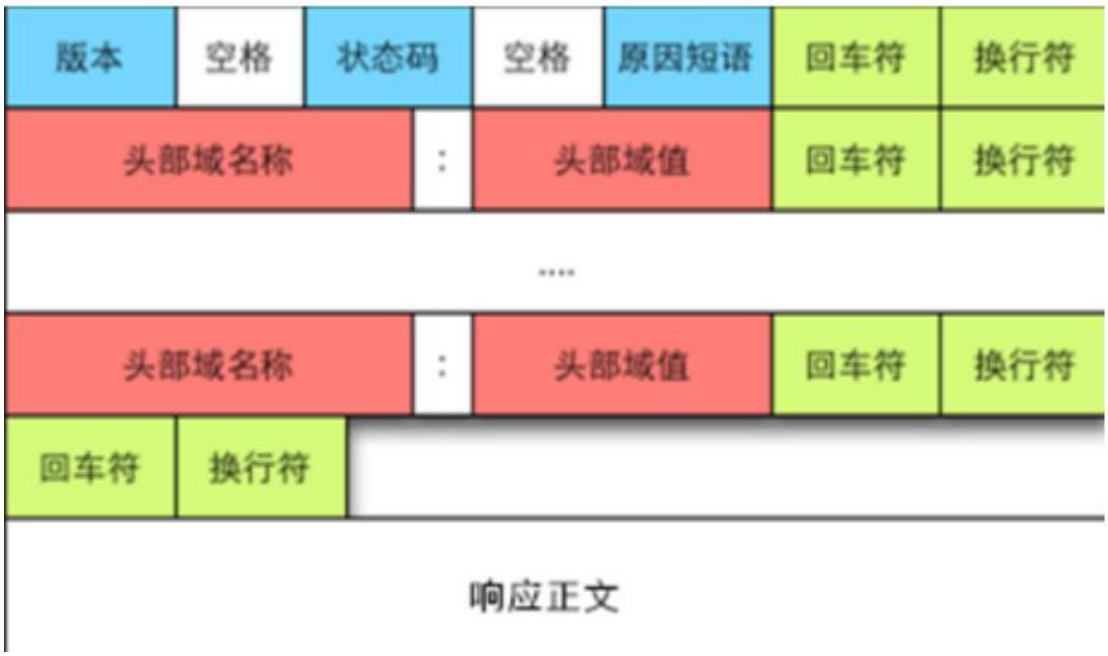
ACK 为客户端 GET 请求的 SEQ+length=1+571=572。

确认收到客户端的请求报文。

2、服务器回应

服务器在向客户端发送 GET 请求的确定报文的同时，发送响应报文。

响应报文结构：



响应报文：

Hypertext Transfer Protocol

```
    HTTP/1.1 200 OK\r\n
    Date: Thu, 27 Oct 2022 02:22:50 GMT\r\n
    Server: Apache/2.4.52 (Ubuntu)\r\n
    Last-Modified: Tue, 25 Oct 2022 11:51:25 GMT\r\n
    ETag: "b4-5ebda871e30db-gzip"\r\n
    Accept-Ranges: bytes\r\n
```

```
Vary: Accept-Encoding\r\n
Content-Encoding: gzip\r\n
Content-Length: 153\r\n
Keep-Alive: timeout=5, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html\r\n
\r\n
[HTTP response 1/3]
[Time since request: 0.002362000 seconds]
[Request in frame: 10]
[Next request in frame: 14]
[Next response in frame: 15]
[Request URI: http://192.168.62.131/lab/lab.html]
Content-encoded entity body (gzip): 153 bytes -> 180 bytes
File Data: 180 bytes
```

响应行：

版本 HTTP/1.1，状态码 200，原因短语 OK

响应头：

Data ： 日期

Server:表示服务器信息

ETag:资源的特定版本的标识符

Last-Modified:请求资源的最后修改时间

Accept-Ranges:用于标识下载中断时，可以尝试中断了的下载，值一般是 0，或 byte,0 表示不支持

Content-Type:告诉客户端实际返回的内容类型

Content-length:消息主体的大小

File Data:响应报文大小

Line-based text data:响应报文的主体，即 http 传送的内容

```
Line-based text data: text/html (12 lines)
    <html>\n
    <head>\n
    \t<title>network</title>\n
    </head>\n
    <body>\n
    \t<p>name:    JiangHaoNan</p>\n
    \t<p>major:   computer science</p>\n
    \t<p>id:      2012948</p>\n
    \t<img src="picture.jpg">\n
    </body>\n
    \n
    </html>\n
```

查看 tcp 可知，长度 489。

```
[TCP Segment Len: 489]
```

3、客户端发送确定报文

```
192.168.62.1    192.168.62.131  TCP      54 1428 → 80 [ACK] Seq=572 Ack=490 Win=130816 Len=0
```

SEQ 为服务器报文的 ACK。为 572。

ACK 为服务器报文 SEQ+length=1+489=490。

4、剩余应答分析

```
192.168.62.1    192.168.62.131  HTTP     570 GET /lab/picture.jpg HTTP/1.1
192.168.62.131  192.168.62.1    HTTP     304 HTTP/1.1 304 Not Modified
192.168.62.1    192.168.62.131  TCP       54 1428 → 80 [ACK] Seq=1088 Ack=740 Win=
192.168.62.1    192.168.62.131  HTTP     478 GET /favicon.ico HTTP/1.1
192.168.62.131  192.168.62.1    HTTP     546 HTTP/1.1 404 Not Found  (text/html)
192.168.62.1    192.168.62.131  TCP       54 1428 → 80 [ACK] Seq=1512 Ack=1232 Wir
```

（1）客户端请求（GET）/lab/picture.jpg，Seq：572，Ack：490，Len：516。

（2）服务器端由于请求资源未更改，返回 304。Seq： 490， Ack:572+16=1088，Len：250

（3）客户端发送确认报文。Seq：1088，Ack：490+250=740，Len：0

（4）客户端请求（GET）/favicon.ico。Seq：1088，Ack：740，Len：424

（5）服务器端由于请求资源不存在，返回 404。Seq：740，Ack：1088+424=1512，Len：492

（6）客户端发送确定报文。Seq：1512，Ack:740+492= 1232，Len：0

（三）TCP 四次挥手

192.168.62.131	192.168.62.1	TCP	60	80 → 1428 [FIN, ACK] Seq=1232 Ack=1512 Win=64128 Len=0
192.168.62.1	192.168.62.131	TCP	54	1428 → 80 [ACK] Seq=1512 Ack=1233 Win=130048 Len=0
192.168.62.1	192.168.62.131	TCP	54	1428 → 80 [FIN, ACK] Seq=1512 Ack=1233 Win=130048 Len=0
192.168.62.131	192.168.62.1	TCP	60	80 → 1428 [ACK] Seq=1233 Ack=1513 Win=64128 Len=0

可以看到为服务器主动提出的 FIN，我们设申请 FIN 的为 A，被动接受的为 B。

起初 A 和 B 处于 ESTABLISHED 状态——A 发出连接释放报文段并处于 FIN-WAIT-1 状态——B 发出确认报文段且进入 CLOSE-WAIT 状态——A 收到确认后，进入 FIN-WAIT-2 状态，等待 B 的连接释放报文段——B 没有要向 A 发出的数据，B 发出连接释放报文段且进入 LAST-ACK 状态——A 发出确认报文段且进入 TIME-WAIT 状态——B 收到确认报文段后进入 CLOSED 状态——A 经过等待计时器时间 2MSL 后，进入 CLOSED 状态。

过程分析：

（1）服务器端发出 FIN 报文，标记位为 FIN，ACK。进入 FIN-WAIT-1 状态。

（2）客户端发送确定报文，标志位 ACK。SEQ 为 FIN 报文的 ACK（1512），Ack 为 FIN 报文的 Seq+1（1232+1=1233）。进入 CLOSE-WAIT 状态。

（3）服务端收到确认后，进入 FIN-WAIT-2 状态，等待客户端的 FIN 报文。

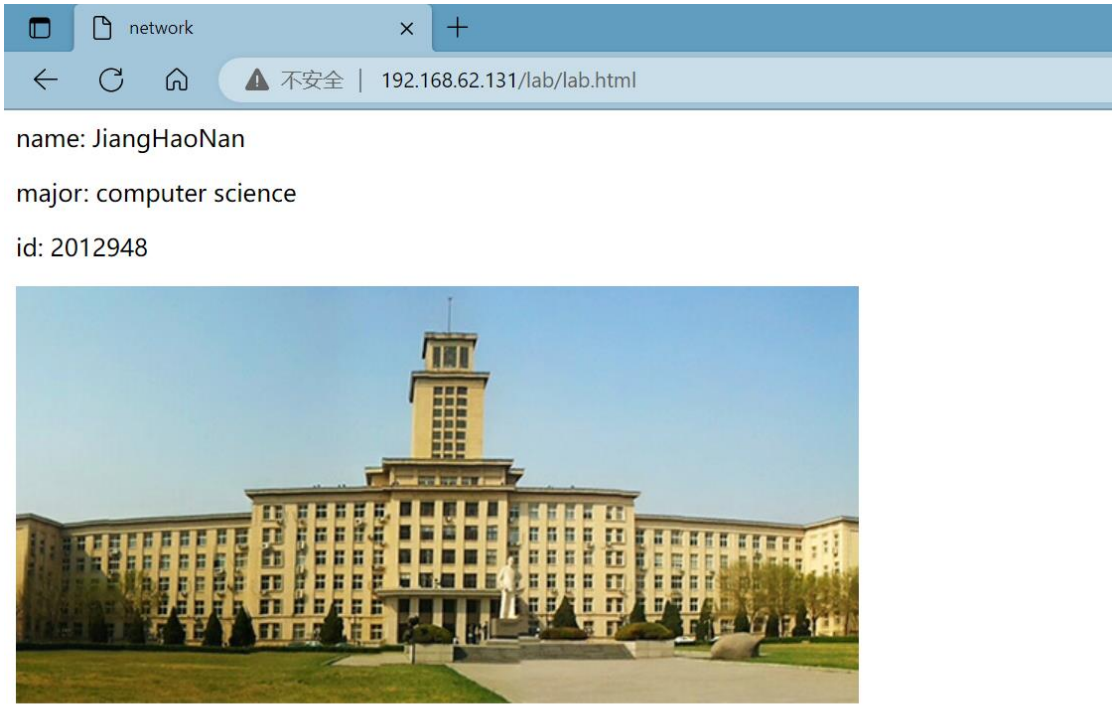
（4）客户端无要发的数据，发出 FIN 报文，标记位为 FIN，ACK。进入 LAST-ACK 状态。Seq 和 Ack 同（2）。

（5）服务器端发出确认报文且进入 TIME-WAIT 状态。Seq 为 FIN 报文的 ACK（1233），Ack 为 FIN 报文的 Seq+1（1512+1=1513）。

（6）客户端收到确认报文段后进入 CLOSED 状态

（7）服务器端经过等待计时器时间 2MSL 后，进入 CLOSED 状态。

四、网页展示



五、实验遇到的问题

1、初次接触，对序列号、确定序列号的累加和传输，在经过几遍梳理后会变得清晰。

2、对四次挥手，双方状态的变换也需要多次梳理熟悉。

3、吐槽下本地虚拟机网络和 Wireshark。