**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。

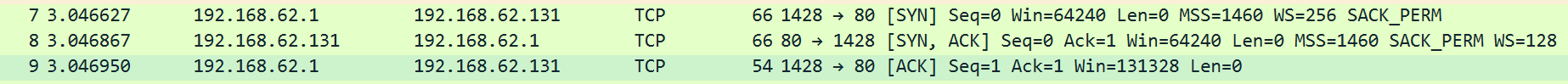
2、选择条件为ip.addr ==192.168.62.131。

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

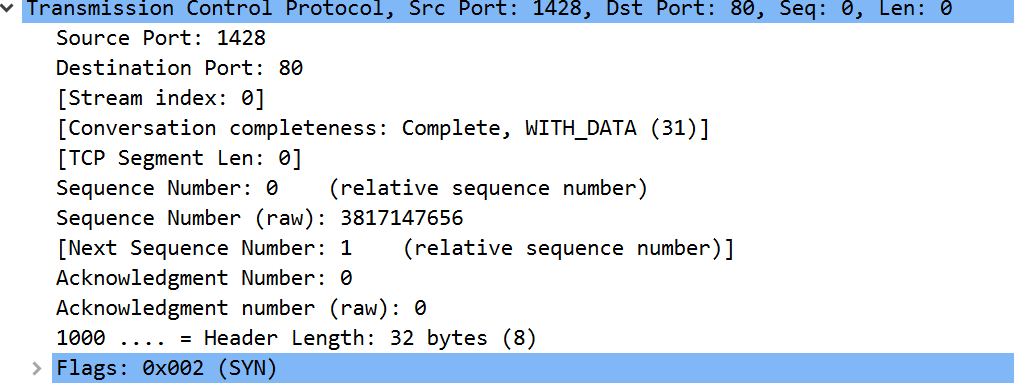
4、在wireshark可以查看交互过程，保存捕获文件。

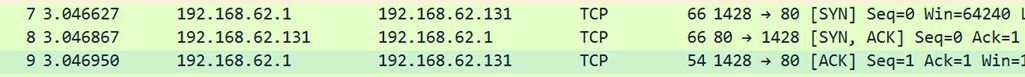
**三、交互过程分析**

## （一）TCP三次握手



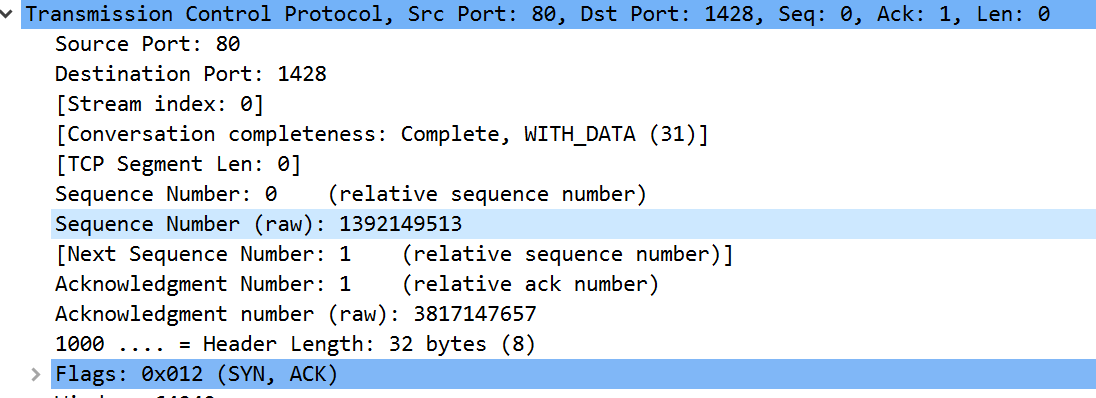
1、第一次握手：由客户端向服务器发送tcp，标志位为SYN，序列号为0，代表客户端请求建立连接（同步）。

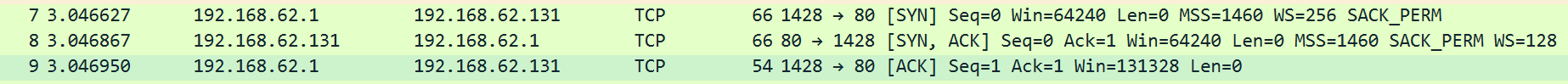




相对序列号为0，Sequence Number (raw): 3817147656。（seq=X）。初始化序列号。

2、第二次握手：服务器发回确认包，标志位为SYN，ACK。

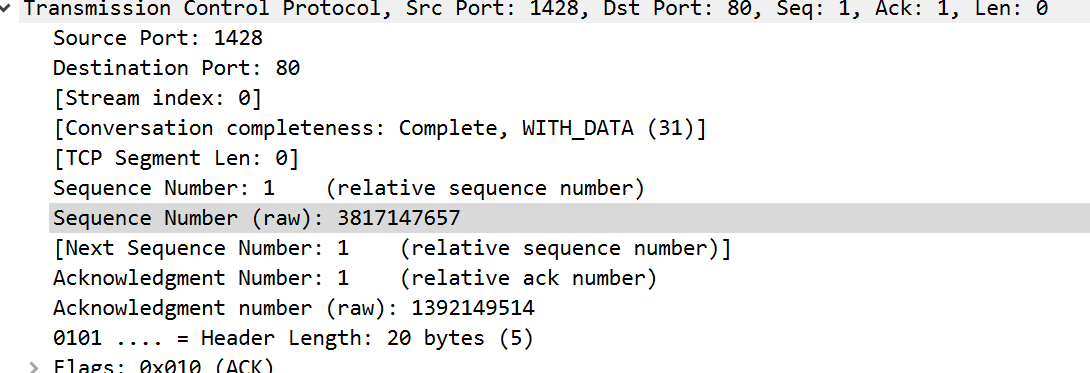


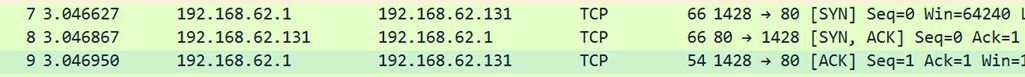


（1）设置确定序列号为x+1（0+1=1），实际为3817147675+1=3817147677。

（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。

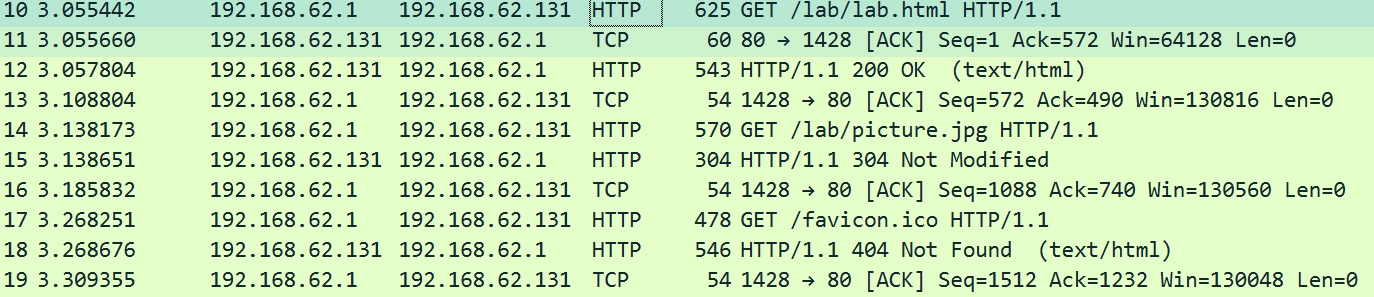




（1）ACK=Y+1，即将上一包的y+1。即0+1=1，实际为1392149513+1=1392149514

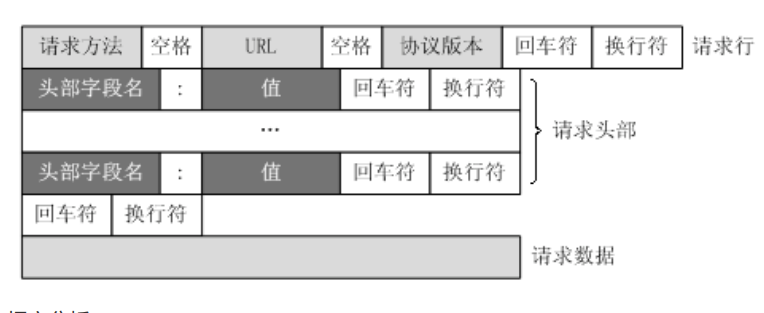
（2）seq为上一包的ack，即3817147677。

## （二）http请求应答



**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,image/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

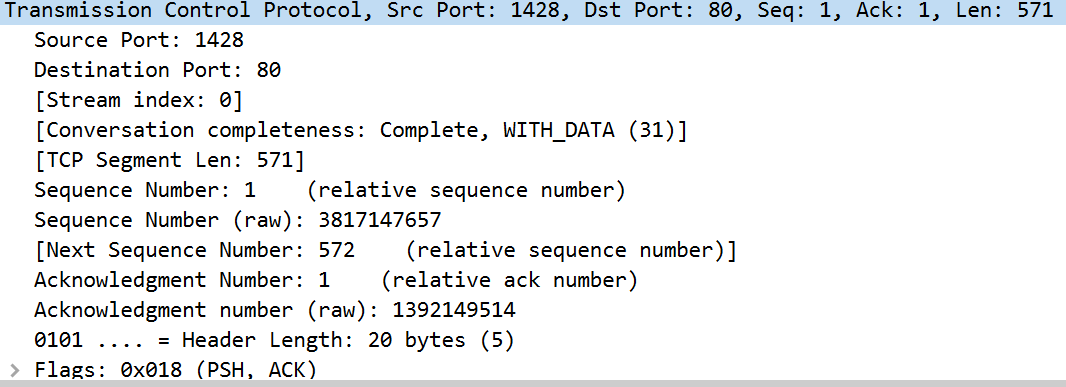
User-Agent为发送请求的操作系统、及浏览器信息

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]。



之后服务器发送确定报文，



SEQ为客户端GET请求的ACK。

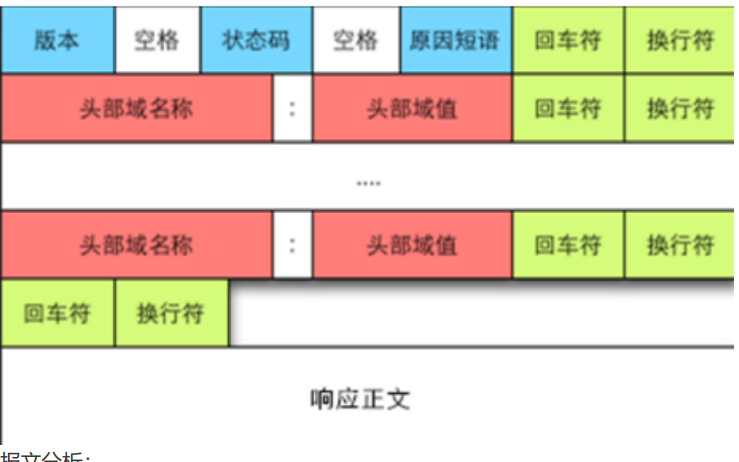
ACK为客户端GET请求的SEQ+length=1+571=572。

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

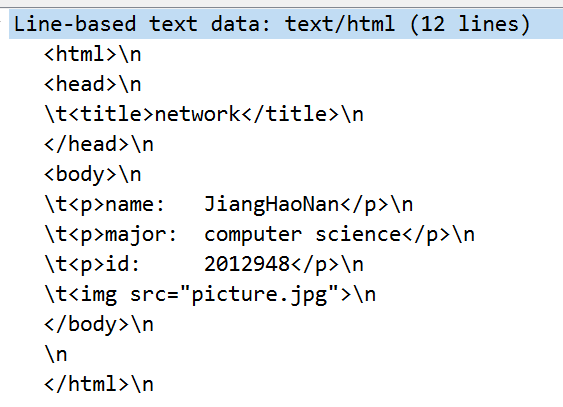
Accept-Ranges:用于标识下载中断时，可以尝试中断了的下载，值一般是0，或byte,0表示不支持

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



查看tcp可知，长度489。



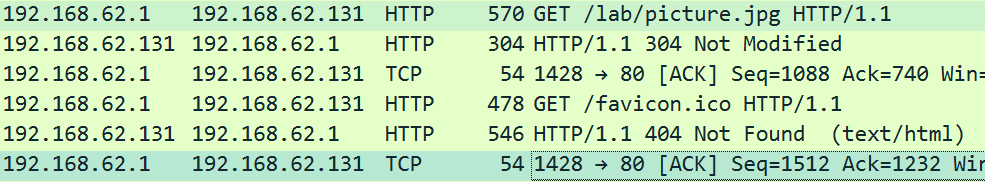
**3、客户端发送确定报文**



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**



（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

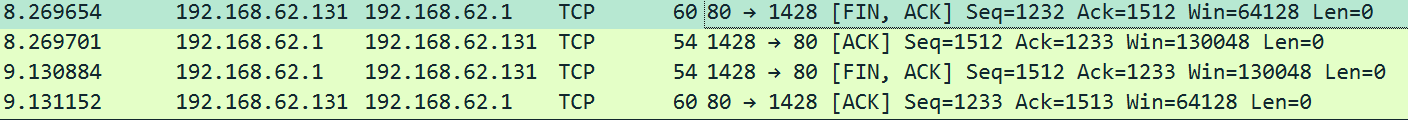
（3）客户端发送确认报文。Seq: 1088, Ack: 490+250=740, Len: 0

（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



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。

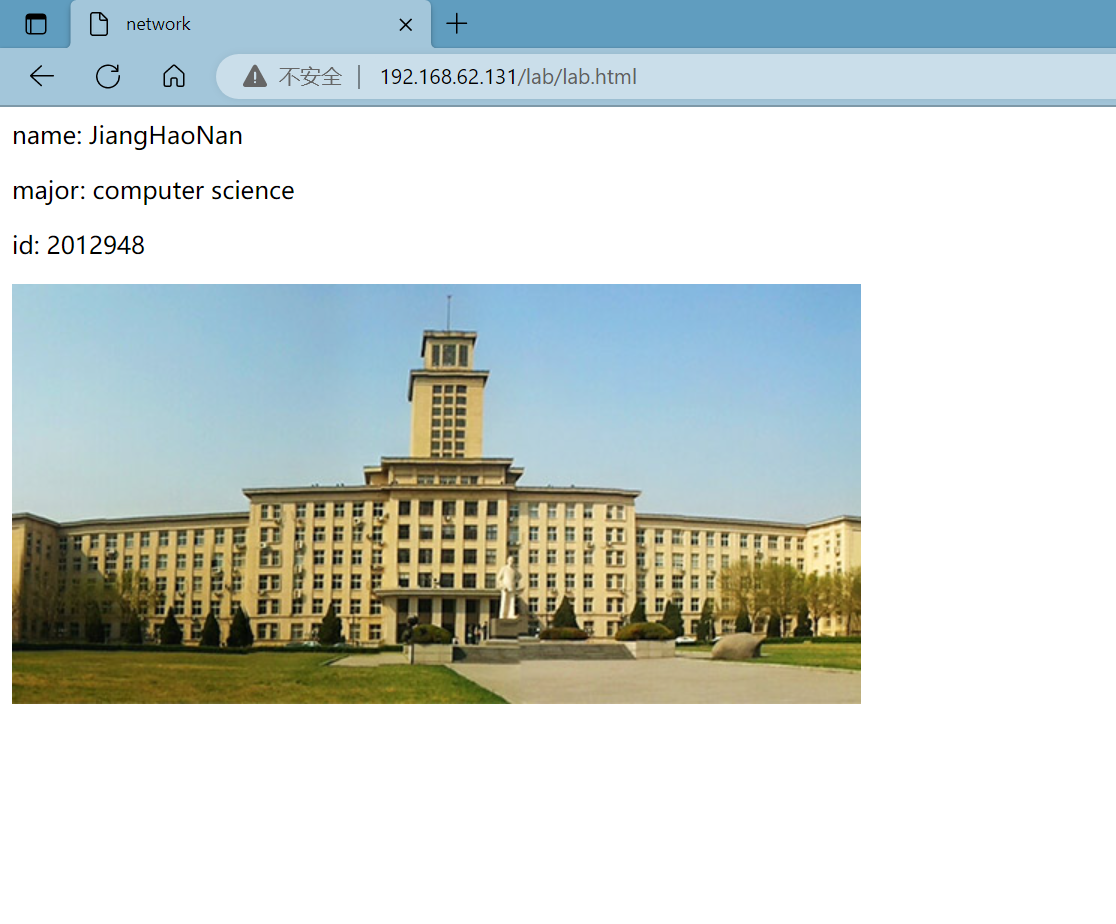
（4）客户端无要发的数据，发出FIN报文，标记位为FIN，ACK。进入LAST-ACK状态。Seq和Ack同（2）。

（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。