Lab2——配置 Web 服务器，编写简单页面，分析交互过程

> **学号：2111033**
> **姓名：艾明旭**
> **专业：信息安全**

一、作业说明

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、配置 Web 服务器

(1)环境配置

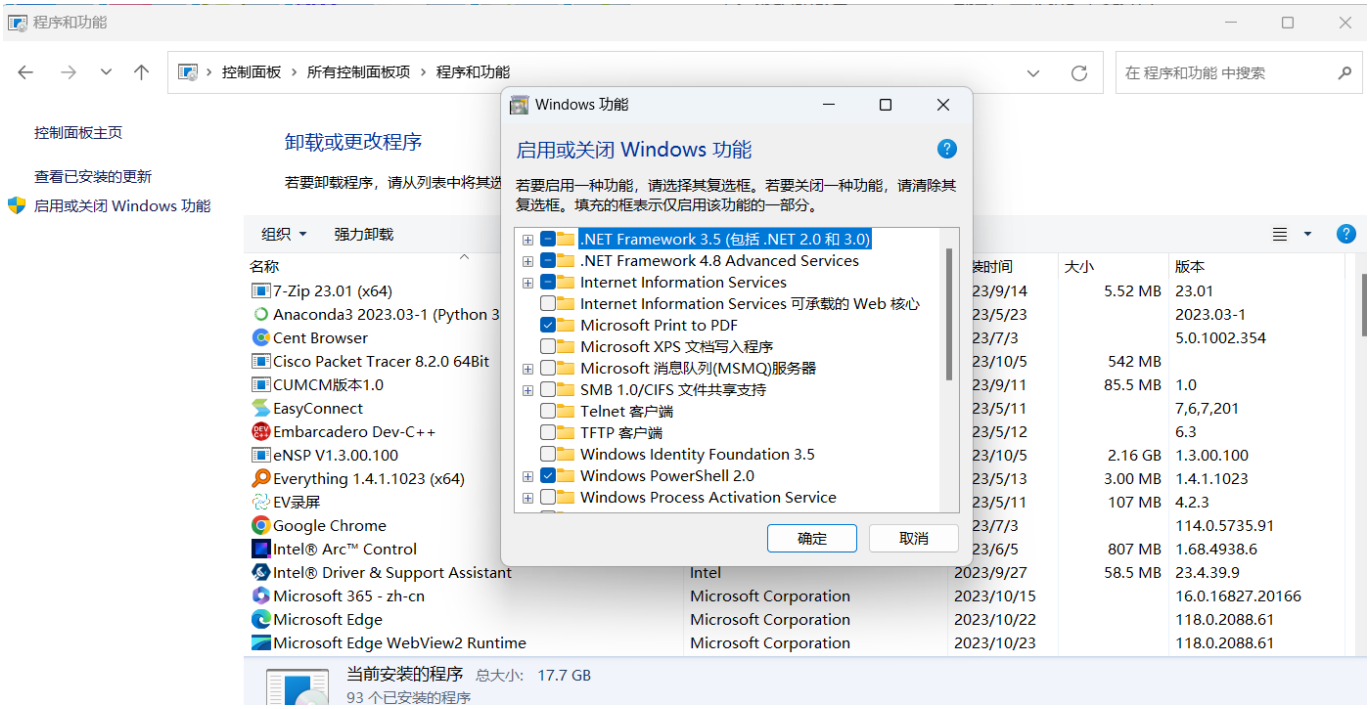
本人使用的是 windows 电脑，主机的环境如下：

- 操作系统： win11
- 抓包工具： Wireshark
- 语言： html
- 框架： node.js
- 浏览器： IE

(2)运行说明

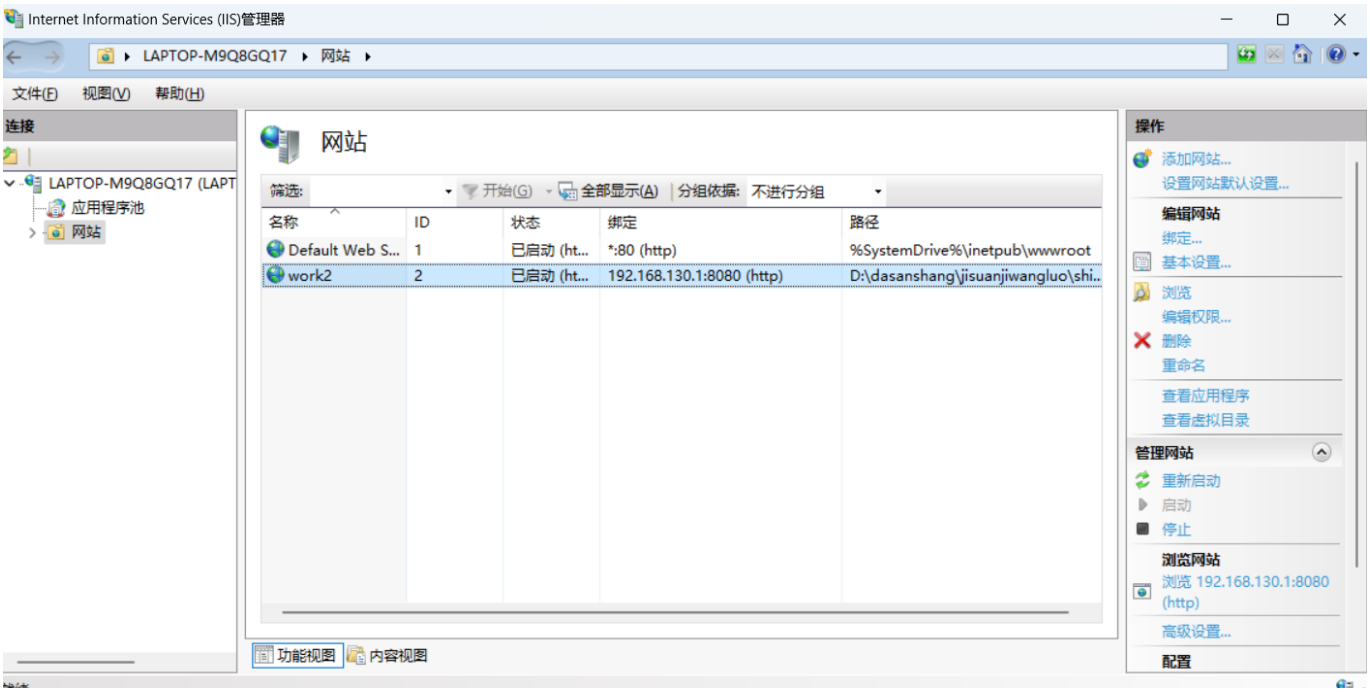
本次 web 服务器的框架使用的是 node.js 的 http-server 框架

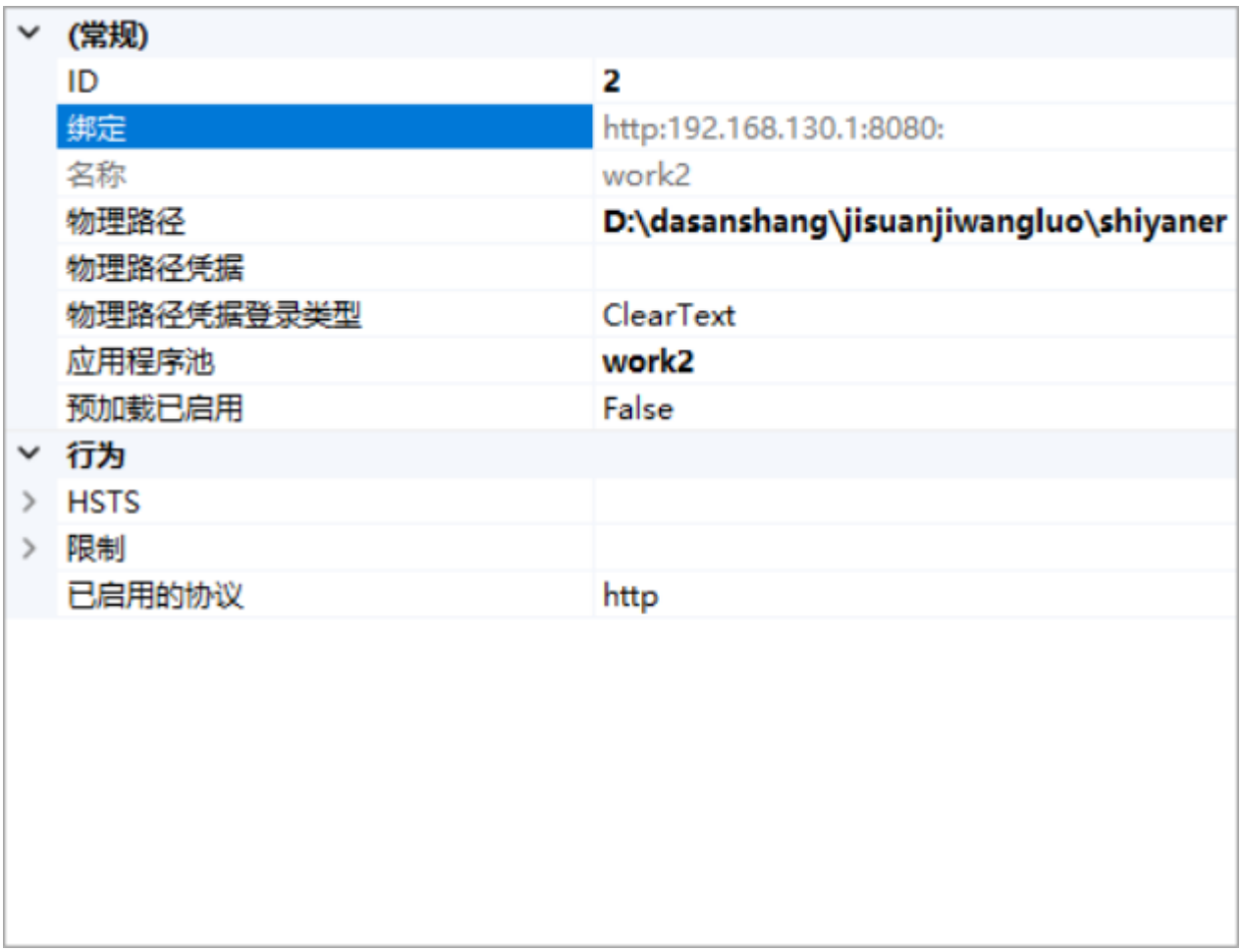
如果想要运行本次项目，首先在控制面板当中下载安装IIS



在程序与功能页面当中选择IIS添加后选择确定。

之后我们打开相关的网站，设定相关的信息。





(3)运行过程

首先是编辑html页面，按照题目要求输入信息。

D: > dasanshang > jisuanjiwangluo > shiyaner > index.html > html > body > audio

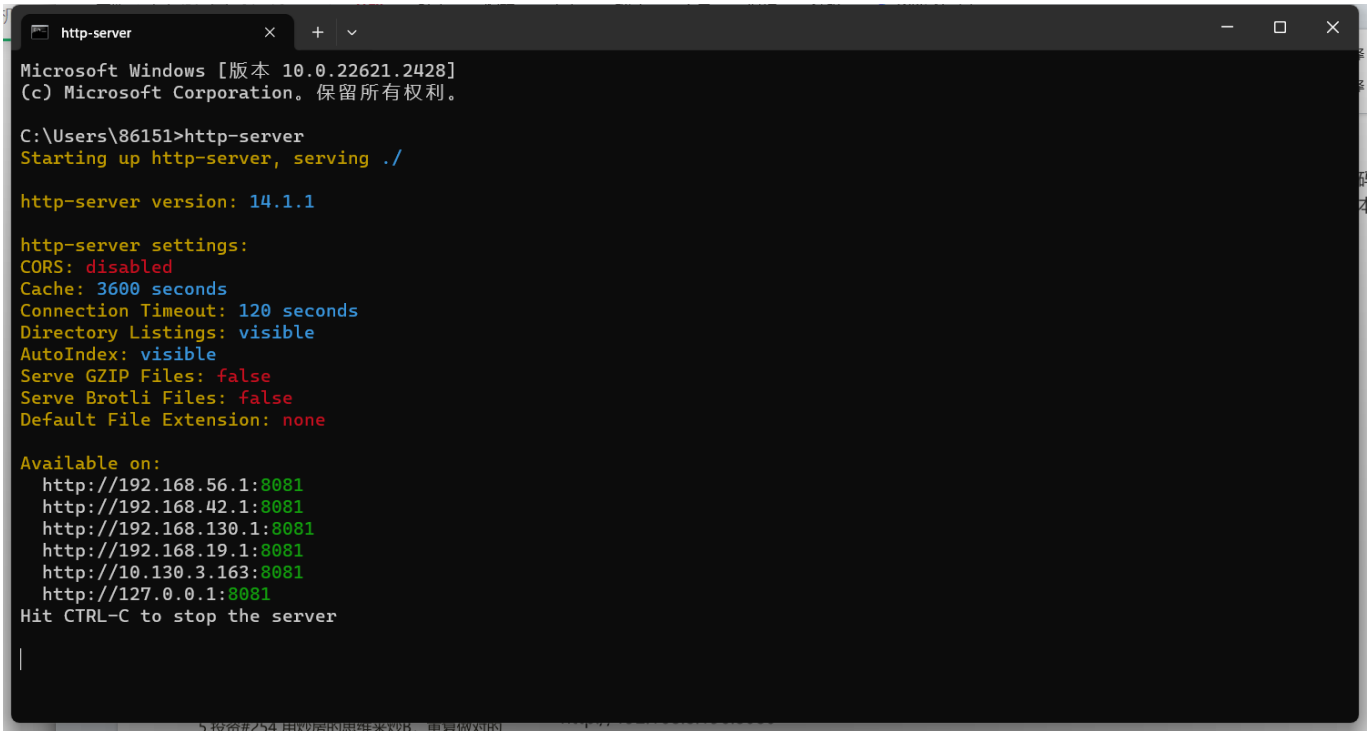
```html
<!DOCTYPE html>
<html>

<head>
    <title>My identity App</title>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8"/>
</head>

<body>
    <h1>Welcome to my identity view</h1>
    <h1>专业：信息安全</h1>
    <h1>学号：2111033</h1>
    <h1>姓名：艾明旭</h1>
    <img src="2.png" alt="logo" width="200" height="300">
    <audio>
        <source src="identity.mp3" type="audio/mp3">
        </source>
    </audio>


</body>

</html>
```

接下来在官网当中下载node.js之后按照所需要的方式安装并启动

之后在终端里下载http-server

```
npm install http-server -g
```

之后输入http-server以启动该程序



我们的服务器就能够成功启动，接下来我们就可以启动页面。

开启服务器后，在浏览器中输入提示的网址即可显示页面

Welcome to my identity view

专业：信息安全

学号：2111033

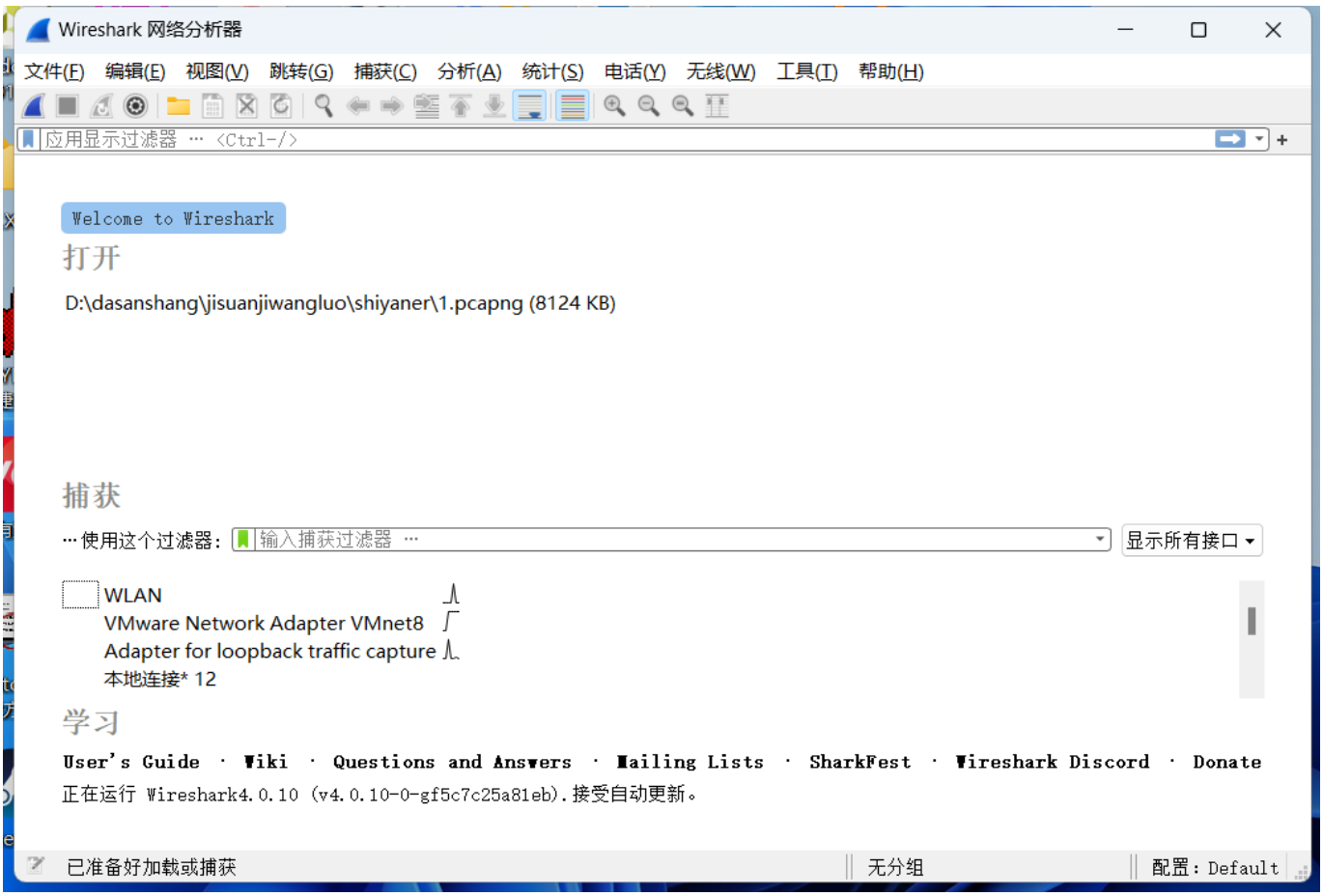
姓名：艾明旭

0:00 / 4:42

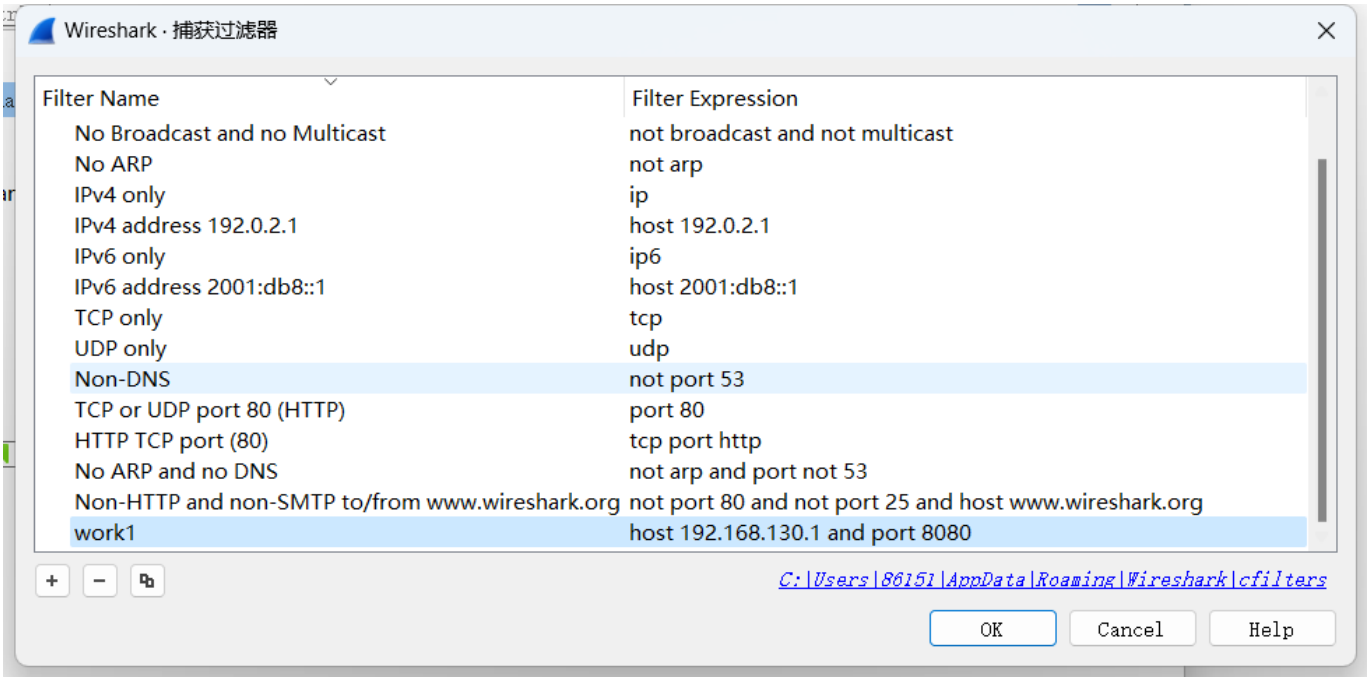
三、交互过程分析

(1)捕获数据报

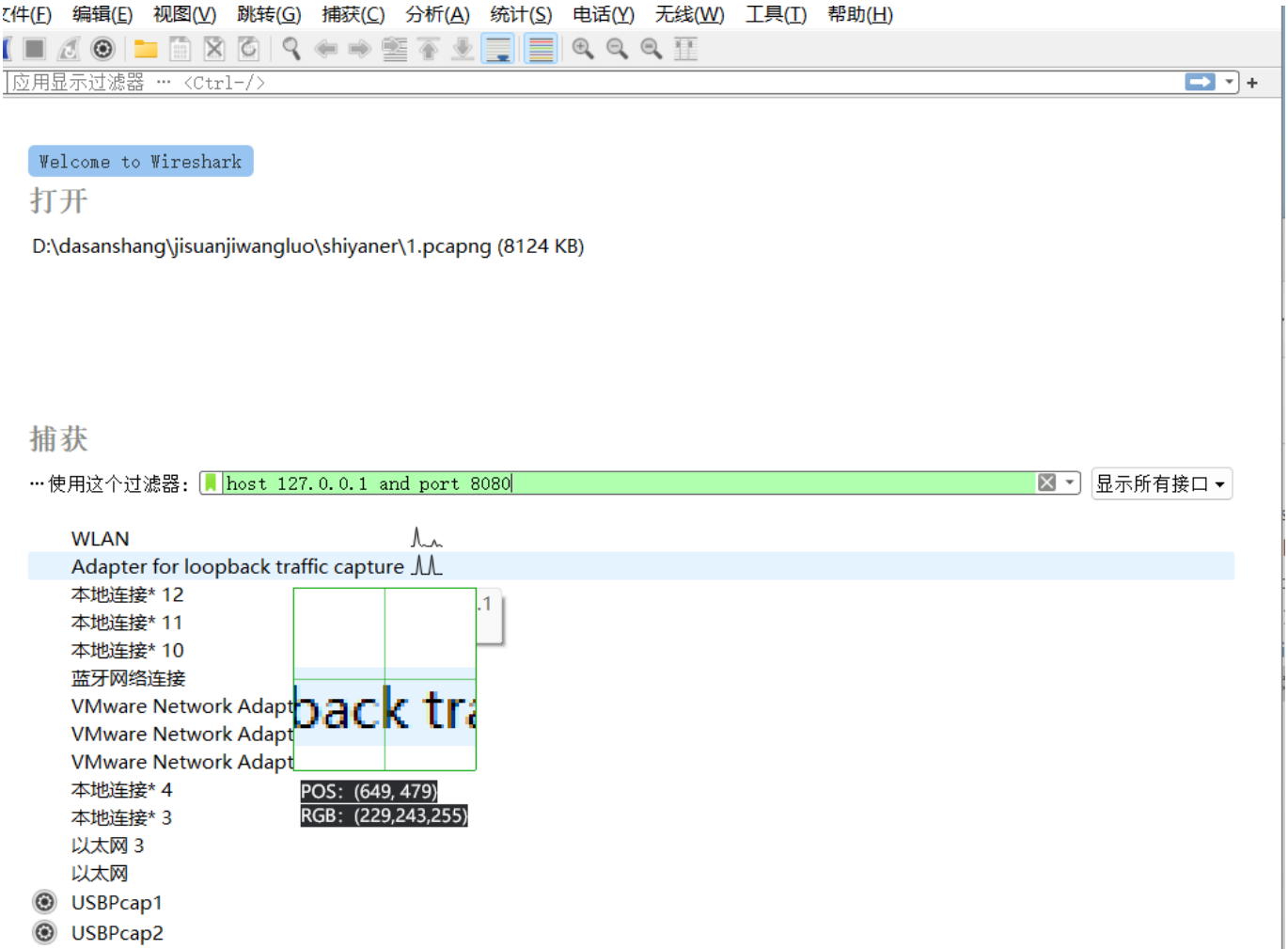
首先开启`Wireshark`程序，在浏览器中输入`http://127.0.0.1:8080`到本机开启的`web`服务器，可以看到如下界面：

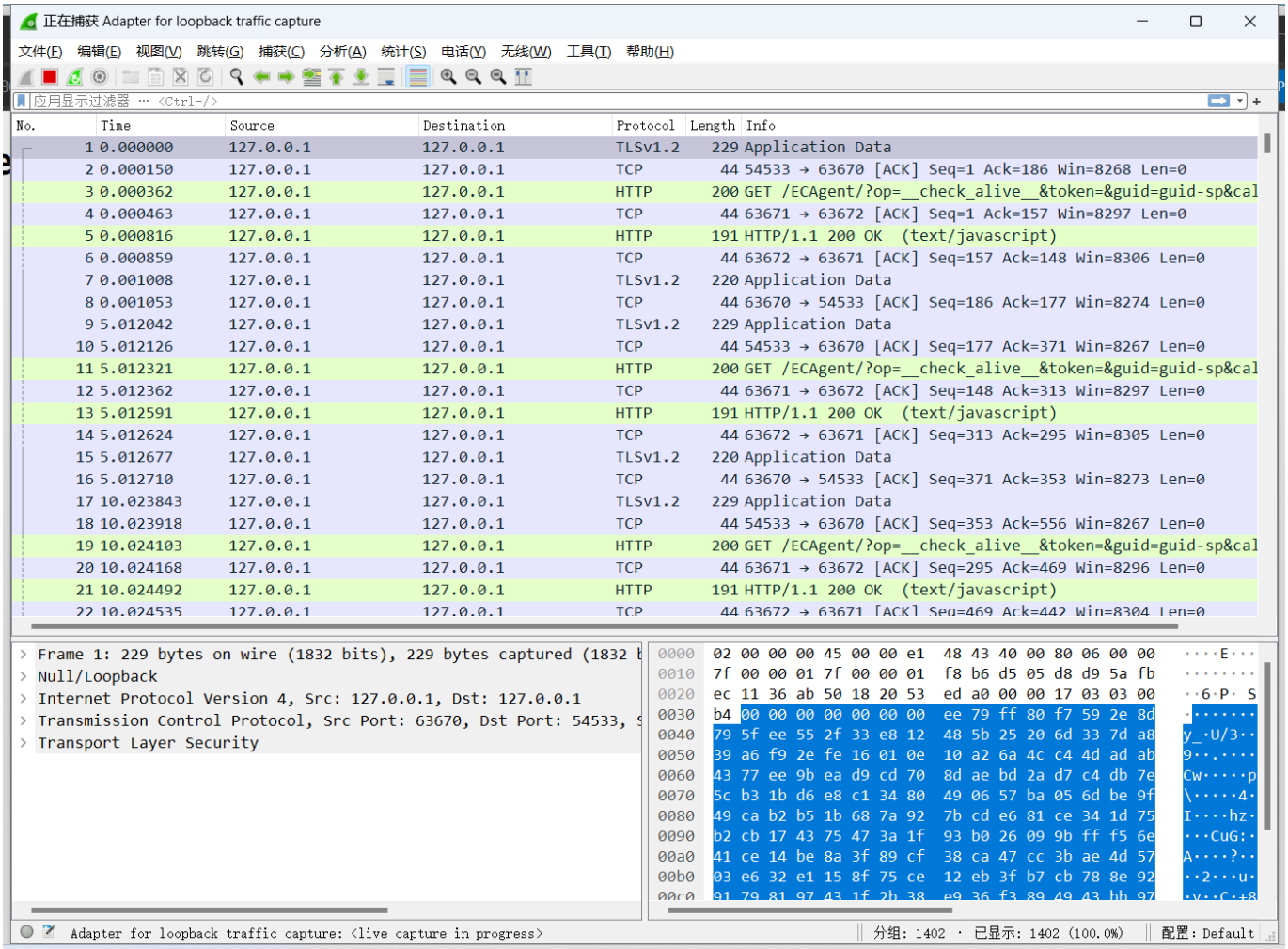


接下来我们选择VMnet8，之后选择编辑捕获过滤器，在Filter Expression里选择输入host 192.168.130.1 and port 8080



在本次的实验中没有去租借服务器，客户端和服务器都是在本机上运行的，所以选择自己的过滤器，并且选择adapter这个选项去进行数据报的嗅探。进去之后可以看到如下界面：


文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
应用显示过滤器 … <Ctrl-/>
Welcome to Wireshark
打开
D:\dasanshang\jisuanjiwangluo\shiyaner\1.pcapng (8124 KB)
捕获
…使用这个过滤器: host 127.0.0.1 and port 8080
显示所有接口
WLAN
Adapter for loopback traffic capture
本地连接* 12
本地连接* 11
本地连接* 10
蓝牙网络连接
VMware Network Adapt
VMware Network Adapt
VMware Network Adapt
本地连接* 4
本地连接* 3
以太网 3
以太网
USBPcap1
USBPcap2
POS: (649, 479)
RGB: (229,243,255)
学习
User's Guide · Wiki · Questions and Answers · Mailing Lists · SharkFest · Wireshark Discord · Donate
正在运行 Wireshark4.0.10 (v4.0.10-0-gf5c7c25a81eb).接受自动更新。



基于此界面来进行本次关于网络传输数据报的分析。

(2) `HTTP` 数据分析

本次的网页设计采用的是`HTTP`协议进行传输，HTTP协议（Hyper Text Transfer Protocol，超文本传输协议）,是用于从万维网（WWW:World Wide Web ）服务器传输超文本到本地浏览器的传送协议。HTTP基于TCP/IP通信协议来传递数据。HTTP基于客户端/服务端（C/S）架构模型，通过一个可靠的链接来交换信息，是一个无状态的请求/响应协议。

> HTTP的特点：

- HTTP是无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。
- HTTP是媒体独立的：只要客户端和服务器知道如何处理的数据内容，任何类型的数据都可以通过HTTP发送。客户端以及服务器指定使用适合的MIME-type内容类型。
- HTTP是无状态：无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

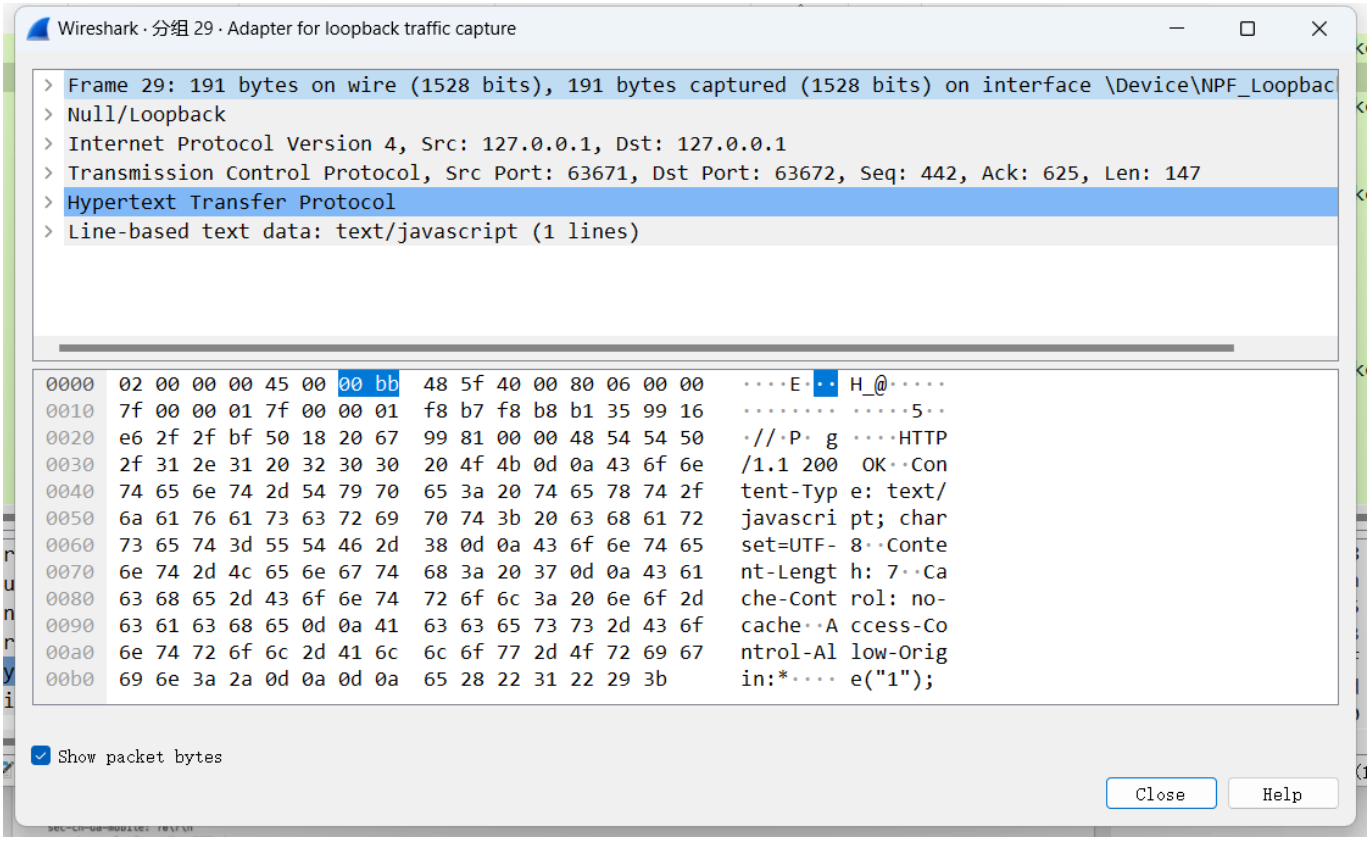
首先在过滤器中输入`'http'`进行筛选，可以得到如下的界面：

No.	Time	Source	Destination	Protocol	Length	Info
27	15.031314	127.0.0.1	127.0.0.1	HTTP	200	GET /ECAgent/?op=__check_alive__&token=&guid=guid-sp&cal
29	15.031833	127.0.0.1	127.0.0.1	HTTP	191	HTTP/1.1 200 OK (text/javascript)
58	20.045659	127.0.0.1	127.0.0.1	HTTP	200	GET /ECAgent/?op=__check_alive__&token=&guid=guid-sp&cal
60	20.046057	127.0.0.1	127.0.0.1	HTTP	191	HTTP/1.1 200 OK (text/javascript)
114	21.883821	10.130.27.249	23.216.147.61	HTTP	198	GET /connecttest.txt HTTP/1.1
387	25.101278	127.0.0.1	127.0.0.1	HTTP	200	GET /ECAgent/?op=__check_alive__&token=&guid=guid-sp&cal
389	25.101656	127.0.0.1	127.0.0.1	HTTP	191	HTTP/1.1 200 OK (text/javascript)
457	25.610862	10.130.27.249	183.47.124.53	HTTP	951	POST /mmtls/0000299b HTTP/1.1
467	25.706715	10.130.27.249	23.216.147.61	HTTP	198	GET /connecttest.txt HTTP/1.1
469	25.714913	10.130.27.249	123.151.48.208	HTTP	956	POST /mmtls/0000299b HTTP/1.1
486	25.923284	10.130.27.249	183.60.8.218	HTTP	986	POST /mmtls/0000299b HTTP/1.1
608	30.142765	127.0.0.1	127.0.0.1	HTTP	200	GET /ECAgent/?op=__check_alive__&token=&guid=guid-sp&cal
610	30.143101	127.0.0.1	127.0.0.1	HTTP	191	HTTP/1.1 200 OK (text/javascript)
674	30.583554	10.130.27.249	123.151.48.193	HTTP	940	POST /mmtls/000029ac HTTP/1.1
908	33.489186	10.130.27.249	222.30.51.134	HTTP	208	GET / HTTP/1.1
930	33.675359	10.130.27.249	23.32.57.240	HTTP	198	GET /connecttest.txt HTTP/1.1

从这张图中可以看出，node框架采用的是http1.1协议来向本机服务器发起一个get请求，然后在服务器收到此get请求后将其所请求的数据返还给客户端，接下来对请求和返还的具体数据报信息进行分析。

> 访问页面的请求

点击第一行的数据报，可以进入到其具体信息的界面，概览信息如下图所示：



由理论课上的知识可以得知，`HTTP`请求报文由3部分组成（**请求行+请求头+请求体**）：

请求行：

请求行分为三部分：

- 请求方法：GET和POST是最常见的HTTP方法，除此以外还包括DELETE、HEAD、OPTIONS、PUT、TRACE
- 请求对应的URL地址，它和报文头的Host属性组成完整的请求URL
- HTTP协议名称及版本号

他们之间以空格分隔，每一行都以\r\n结尾，可以看出本次是GET请求，以HTTP1.1请求当前页面。

请求头：

HTTP的报文头包含若干个属性，格式为“属性名:属性值”，服务端据此获取客户端的信息。与缓存相关的规则信息，均包含在header中。

常见的请求头有以下几种：

- Accept：请求报文可通过一个“Accept”报文头属性告诉服务端客户端接受什么类型的响应，Accept属性的值可以为一个或多个MIME类型的值（描述消息内容类型的因特网标准， 消息能包含文本、图像、音频、视频以及其他应用程序专用的数据）
- Referer：表示这个请求是传输过来的URL，如果通过google搜索出一个商家的广告页面，这个请求报文的Referer报文头属性值就是http://www.google.com
- Cookie：客户端的Cookie就是通过这个报文头属性传给服务端
- Cache-Control：对缓存进行控制

本次可看出有Host、connection、Accept等键值对。

请求体：

报文体将一个页面表单中的组件值通过param1=value1&param2=value2的键值对形式编码成一个格式化串，它承载多个请求参数的数据。

> 服务器返回的数据

点击与GET交互的下一行数据，可以观察到如下两个页面：

```
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 187
  Identification: 0x4867 (18535)
> 010. .... = Flags: 0x2, Don't fragment
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 128
  Protocol: TCP (6)
  Header Checksum: 0x0000 [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 127.0.0.1
  Destination Address: 127.0.0.1
Transmission Control Protocol, Src Port: 63671, Dst Port: 63672, Seq: 589, Ack: 781, Len: 147
  Source Port: 63671
  Destination Port: 63672
  [Stream index: 1]
  [Conversation completeness: Incomplete (28)]
  [TCP Segment Len: 147]
  Sequence Number: 589    (relative sequence number)
  Sequence Number (raw): 2973079977
  [Next Sequence Number: 736    (relative sequence number)]
  Acknowledgment Number: 781    (relative ack number)
  Acknowledgment number (raw): 3861852251
  0101 .... = Header Length: 20 bytes (5)
> Flags: 0x018 (PSH, ACK)
  Window: 8295
  [Calculated window size: 8295]
  [Window size scaling factor: -1 (unknown)]
  Checksum: 0x9852 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
> [Timestamps]
> [SEQ/ACK analysis]
```

```
   [Window size scaling factor: -1 (unknown)]
   Checksum: 0x9852 [unverified]
   [Checksum Status: Unverified]
   Urgent Pointer: 0
 > [Timestamps]
 > [SEQ/ACK analysis]
   TCP payload (147 bytes)
Hypertext Transfer Protocol
 > HTTP/1.1 200 OK\r\n
   Content-Type: text/javascript; charset=UTF-8\r\n
 > Content-Length: 7\r\n
   Cache-Control: no-cache\r\n
   Access-Control-Allow-Origin:*\r\n
   \r\n
   [HTTP response 5/5]
   [Time since request: 0.000398000 seconds]
   [Prev request in frame: 27]
   [Prev response in frame: 29]
   [Request in frame: 58]
   [Request URI: http://127.0.0.1:54533/ECAgent/?op=__check_alive__&token=&guid=guid-sp&callback
   File Data: 7 bytes
Line-based text data: text/javascript (1 lines)
   e("1");
```

```
000  02 00 00 00 45 00 00 bb  48 67 40 00 80 06 00 00   ····E··· Hg@·····
010  7f 00 00 01 7f 00 00 01  f8 b7 f8 b8 b1 35 99 a9   ········ ·····5··
020  e6 2f 30 5b 50 18 20 67  98 52 00 00 48 54 54 50   ·/0[P· g ·R··HTTP
030  2f 31 2e 31 20 32 30 30  20 4f 4b 0d 0a 43 6f 6e   /1.1 200  OK··Con
040  74 65 6e 74 2d 54 79 70  65 3a 20 74 65 78 74 2f   tent-Typ e: text/
050  6a 61 76 61 73 63 72 69  70 74 3b 20 63 68 61 72   javascri pt; char
060  73 65 74 3d 55 54 46 2d  38 0d 0a 43 6f 6e 74 65   set=UTF- 8··Conte
070  6e 74 2d 4c 65 6e 67 74  68 3a 20 37 0d 0a 43 61   nt-Lengt h: 7··Ca
080  63 68 65 2d 43 6f 6e 74  72 6f 6c 3a 20 6e 6f 2d   che-Cont rol: no-
090  63 61 63 68 65 0d 0a 41  63 63 65 73 73 2d 43 6f   cache··A ccess-Co
0a0  6e 74 72 6f 6c 2d 41 6c  6c 6f 77 2d 4f 72 69 67   ntrol-Al low-Orig
0b0  69 6e 3a 2a 0d 0a 0d 0a  65 28 22 31 22 29 3b      in:*···· e("1");
```

由理论课上的知识可以得知，HTTP的响应报文也由三部分组成（**响应行+响应头+响应体**）：

响应行：

响应行由两部分组成：

- 报文协议及版本
- 状态码及状态描述

本次的版本为HTTP1.1并且一切正常。

常见的状态码及其描述：

- 200 (OK): 找到了该资源，并且一切正常。

- 302/307：临时重定向，指出请求的文档已被临时移动到别处, 此文档的新的url在location响应头中给出
- 304 (NOT MODIFIED): 该资源在上次请求之后没有任何修改。这通常用于浏览器的缓存机制。
- 401 (UNAUTHORIZED): 客户端无权访问该资源。这通常会使得浏览器要求用户输入用户名和密码，以登录到服务器。
- 403 (FORBIDDEN): 客户端未能获得授权。这通常是在401之后输入了不正确的用户名或密码。
- 404 (NOT FOUND): 在指定的位置不存在所申请的资源。

响应头：

响应报文头与请求头类似，也是由多个属性组成。

响应体：

响应报文体是我们真正想获得的东西，响应体为所请求的数据（可以为html文档内容或图片等各种格式数据），本次即为html文档。

(3) `TCP` 三次握手

TCP的三次握手和四次挥手是建立和终止TCP连接的过程。

1. 三次握手（Three-way Handshake）：

   第一步：客户端发送一个SYN（同步）报文给服务器端，表示请求建立连接，并选择一个初始的序列号。

   第二步：服务器端接收到SYN报文后，回复一个SYN+ACK（同步+确认）报文给客户端，表示同意建立连接，并为连接分配资源，确认号是客户端的序列号加一。

   第三步：客户端收到服务器端的SYN+ACK报文后，向服务器端发送一个ACK（确认）报文，表示连接建立成功。服务器端收到ACK报文后，连接建立完成。

   这样，双方完成了三次握手，建立了可靠的TCP连接，可以开始传输数据。

2. 四次挥手（Four-way Handshake）：

   第一步：当客户端要关闭连接时，发送一个FIN（结束）报文给服务器端。

   第二步：服务器端接收到FIN报文后，返回一个ACK报文作为确认，表示收到客户端的关闭请求。

   第三步：服务器端向客户端发送一个FIN报文，表示服务器端也准备关闭连接。

第四步：客户端接收到服务器端的FIN报文后，发送一个ACK报文作为确认，表示双方都同意关闭连接。

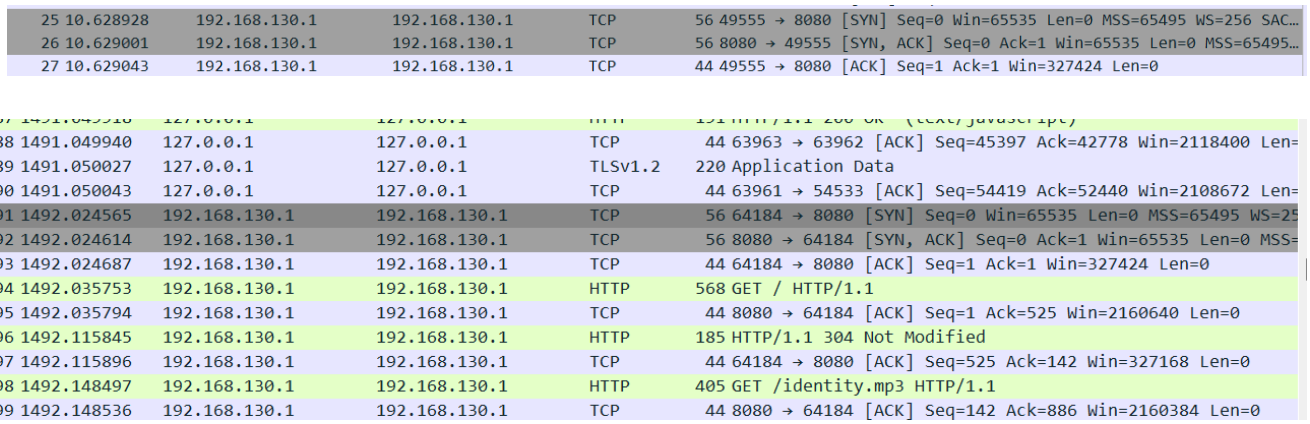
这样，双方完成了四次挥手，连接断开。

通过三次握手和四次挥手，TCP实现了可靠的连接建立和连接断开过程，确保数据的可靠传输和资源的及时释放。

TCP协议提供的是按序、可靠的服务，是一种面向连接的传输方式，即其发送数据之前发送方和接收放需要握手，断开链接时需要四次挥手。

> 三次握手过程

1. 第一次握手：客户端发送syn包(seq=x)到服务器，并进入SYN_SEND状态，等待服务器确认
2. 第二次握手：服务器收到syn包，必须确认客户的SYN(ack=x+1)，同时自己也发送一个SYN包(seq=y)，即SYN+ACK包，此时服务器进入SYN_RECV状态
3. 第三次握手：客户端收到服务器的SYN+ACK包，向服务器发送确认包ACK(ack=y+1)，此包发送完毕，客户端和服务器进入ESTABLISHED状态，完成三次握手握手过程中传送的包里不包含数据，三次握手完毕后，客户端与服务器才正式开始传送数据。理想状态下，TCP连接一旦建立，在通信双方中的任何一方主动关闭连接之前，TCP连接都将被一直保持下去。



25	10.628928	192.168.130.1	192.168.130.1	TCP	56 49555 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SAC…
26	10.629001	192.168.130.1	192.168.130.1	TCP	56 8080 → 49555 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495…
27	10.629043	192.168.130.1	192.168.130.1	TCP	44 49555 → 8080 [ACK] Seq=1 Ack=1 Win=327424 Len=0
4188	1491.049940	127.0.0.1	127.0.0.1	TCP	44 63963 → 63962 [ACK] Seq=45397 Ack=42778 Win=2118400 Len=
4189	1491.050027	127.0.0.1	127.0.0.1	TLSv1.2	220 Application Data
4190	1491.050043	127.0.0.1	127.0.0.1	TCP	44 63961 → 54533 [ACK] Seq=54419 Ack=52440 Win=2108672 Len=
4191	1492.024565	192.168.130.1	192.168.130.1	TCP	56 64184 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=25
4192	1492.024614	192.168.130.1	192.168.130.1	TCP	56 8080 → 64184 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=
4193	1492.024687	192.168.130.1	192.168.130.1	TCP	44 64184 → 8080 [ACK] Seq=1 Ack=1 Win=327424 Len=0
4194	1492.035753	192.168.130.1	192.168.130.1	HTTP	568 GET / HTTP/1.1
4195	1492.035794	192.168.130.1	192.168.130.1	TCP	44 8080 → 64184 [ACK] Seq=1 Ack=525 Win=2160640 Len=0
4196	1492.115845	192.168.130.1	192.168.130.1	HTTP	185 HTTP/1.1 304 Not Modified
4197	1492.115896	192.168.130.1	192.168.130.1	TCP	44 64184 → 8080 [ACK] Seq=525 Ack=142 Win=327168 Len=0
4198	1492.148497	192.168.130.1	192.168.130.1	HTTP	405 GET /identity.mp3 HTTP/1.1
4199	1492.148536	192.168.130.1	192.168.130.1	TCP	44 8080 → 64184 [ACK] Seq=142 Ack=886 Win=2160384 Len=0

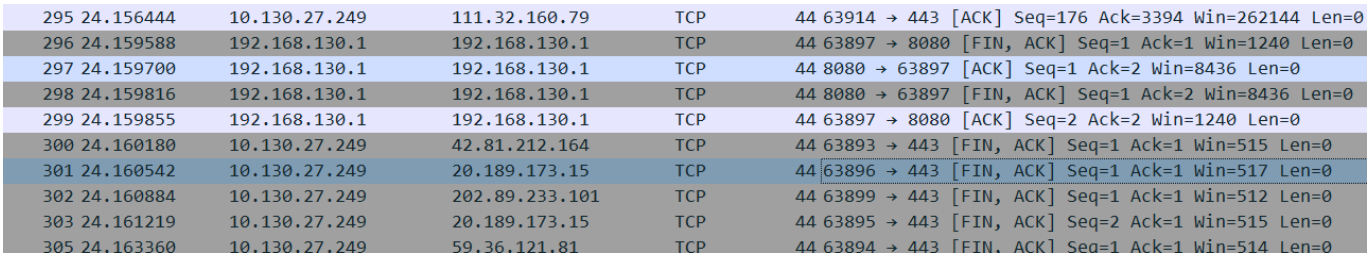
- 如上图所示，即为tcp的三次握手过程。首先由64184向8080端口发送了一条syn包，并令seq=x=0（本机初始序列号），报文长度为0，滑动窗口为65535，最大窗口长度为16344，窗口扩大因子为64，此时浏览进入了SYN_SEND状态，这是第一次握手
- 然后服务器8080端口接收到了浏览器发送的syn包后，确认客户的syn，使ack=x+1=1，seq=y=0，对序列号为1 (ack）之前的报文进行确认，同时向客户发送一个（syn,ack）包，其中报文长度为0，滑动窗口为65535，最大窗口长度为16344，这是第二次握手
- 客户端接受到了服务器发送的(syn，ack） 包后，客户端向服务器返回确认包，其中ack=y+1=1，滑动窗口为408256，报文长度为0，这是第三次握手。

至此三次握手完成，服务器和客户端成功建立了联系。

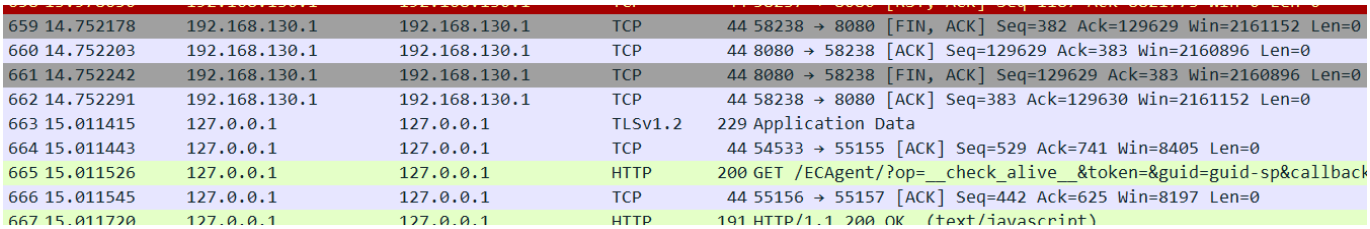
(4) TCP 四次挥手

> 四次挥手过程

1. 第一次挥手：Client发送一个FIN，用来关闭Client到Server的数据传送，Client进入FIN_WAIT_1状态。
2. 第二次挥手：Server收到FIN后，发送一个ACK给Client，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序号），Server进入CLOSE_WAIT状态。
3. 第三次挥手：Server发送一个FIN，用来关闭Server到Client的数据传送，Server进入LAST ACK状态。
4. 第四次挥手：Client收到FIN后，Client进入TIME_WAIT状态，接着发送一个ACK给Server，确认序号为收到序号+1，Server进入CLOSED状态，完成四次挥手。



295	24.156444	10.130.27.249	111.32.160.79	TCP	44 63914 → 443 [ACK] Seq=176 Ack=3394 Win=262144 Len=0
296	24.159588	192.168.130.1	192.168.130.1	TCP	44 63897 → 8080 [FIN, ACK] Seq=1 Ack=1 Win=1240 Len=0
297	24.159700	192.168.130.1	192.168.130.1	TCP	44 8080 → 63897 [ACK] Seq=1 Ack=2 Win=8436 Len=0
298	24.159816	192.168.130.1	192.168.130.1	TCP	44 8080 → 63897 [FIN, ACK] Seq=1 Ack=2 Win=8436 Len=0
299	24.159855	192.168.130.1	192.168.130.1	TCP	44 63897 → 8080 [ACK] Seq=2 Ack=2 Win=1240 Len=0
300	24.160180	10.130.27.249	42.81.212.164	TCP	44 63893 → 443 [FIN, ACK] Seq=1 Ack=1 Win=515 Len=0
301	24.160542	10.130.27.249	20.189.173.15	TCP	44 63896 → 443 [FIN, ACK] Seq=1 Ack=1 Win=517 Len=0
302	24.160884	10.130.27.249	202.89.233.101	TCP	44 63899 → 443 [FIN, ACK] Seq=1 Ack=1 Win=512 Len=0
303	24.161219	10.130.27.249	20.189.173.15	TCP	44 63895 → 443 [FIN, ACK] Seq=2 Ack=1 Win=515 Len=0
305	24.163360	10.130.27.249	59.36.121.81	TCP	44 63894 → 443 [FIN, ACK] Seq=1 Ack=1 Win=514 Len=0



659	14.752178	192.168.130.1	192.168.130.1	TCP	44 58238 → 8080 [FIN, ACK] Seq=382 Ack=129629 Win=2161152 Len=0
660	14.752203	192.168.130.1	192.168.130.1	TCP	44 8080 → 58238 [ACK] Seq=129629 Ack=383 Win=2160896 Len=0
661	14.752242	192.168.130.1	192.168.130.1	TCP	44 8080 → 58238 [FIN, ACK] Seq=129629 Ack=383 Win=2160896 Len=0
662	14.752291	192.168.130.1	192.168.130.1	TCP	44 58238 → 8080 [ACK] Seq=383 Ack=129630 Win=2161152 Len=0
663	15.011415	127.0.0.1	127.0.0.1	TLSv1.2	229 Application Data
664	15.011443	127.0.0.1	127.0.0.1	TCP	44 54533 → 55155 [ACK] Seq=529 Ack=741 Win=8405 Len=0
665	15.011526	127.0.0.1	127.0.0.1	HTTP	200 GET /ECAgent/?op=__check_alive__&token=&guid=guid-sp&callback
666	15.011545	127.0.0.1	127.0.0.1	TCP	44 55156 → 55157 [ACK] Seq=442 Ack=625 Win=8197 Len=0
667	15.011720	127.0.0.1	127.0.0.1	HTTP	191 HTTP/1.1 200 OK (text/javascript)

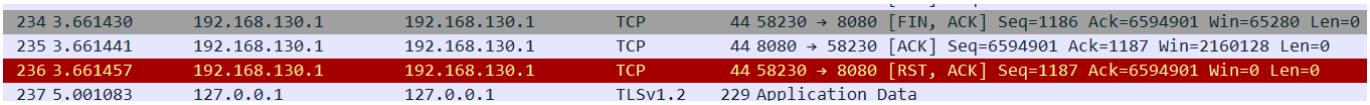
- 如上图所示，即为tcp四次挥手过程。首先，8080向63897端口发送了一条请求结束的报文(fin，ack)，报文序列号为seq=×=176，ack=y=3394，报文长度为0，这是第一次挥手
- 然后63897端口接收到了浏览器发送的rin包后，确认其报文序列号为seq=y=176,ack=x+1=3393+1=3394，这是第二次挥手
- 然后服务器63897端口再向8080发送一个fin包，其内容与前一个基本相同，用来关闭服务器端到客户端的数据传送，这是第三次挥手
- 8080端口对收到的fin包进行确认，其报文序列号为seq=y=176,ack=3393+1=3394，客户端进入关闭状态，这是第四次挥手。

至此tcp的四次挥手结束，客户端和服务器断开连接，访问结束。

四、问题与思考

(1)问题一

在TCP的四次挥手过程中，有时候会出现RST（重置）报文的情况。RST报文是一种TCP报文，用于立即中止连接，无需经过正常的挥手过程。



RST报文的出现可能是由于以下情况：

1. 异常连接终止：当某一方在接收到对方的FIN报文后，发现自己没有对应的连接或者已经关闭了连接，但是对方仍然发送数据。这种情况下，该方可以发送RST报文来中止连接，以表示连接异常终止。

2. 攻击性行为：RST报文也可以被用作网络攻击的一种手段。攻击者可以发送伪造的RST报文，以迫使另一端中止连接，从而中断通信或者进行其他恶意操作。

需要注意的是，正常的四次挥手过程中并不一定会出现RST报文。RST报文的出现通常是在异常情况下才会发生，用于迅速中止连接。

(2)问题二

本次设计使用的是http协议，但是老师在上课中提到了这个协议并不安全，并且已经有https协议来代替，所以http和https的区别是什么呢?

在查阅网上的相关资料后得知，主要有以下几点区别：

1. HTTPS协议需要到 CA （Certificate Authority，证书颁发机构）申请证书，一般免费证书较少，因而需要一定费用
2. HTTP 是超文本传输协议，信息是明文传输，HTTPS 则是具有安全性的 SSL 加密传输协议
3. HTTP 和 HTTPS 使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443
4. HTTP 的连接很简单，是无状态的。HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议，比 HTTP 协议安全。(无状态的意思是其数据包的发送、传输和接收都是相互独立的。无连接的意思是指通信双方都不长久的维持对方的任何信息。)

所以https可以保证数据传输的机密性和防止重放攻击，是现行架构下最安全的解决方案，虽然不是绝对安全，但它大幅增加了中间人攻击的成本。

但https也有一些缺点，比如握手阶段比较费时、增加数据开销、加密范围也比较有限等，所以使用协议时需要综合考虑。

五、总结与展望

(1)总结

本次实验是计算机网络的第二次实验，这一次的实验主要是对网络数据报进行分析，在本次实验中首先设置了一个简单的web服务器，并对http请求做了分析，接下来对tcp三次握手和四次挥手也做了比较详细的分析，最后也在网上查阅了相关资料，对http和https的区别有了更进一步的认识。

(2)展望

本次实验让我对网络方面的东西产生了更大的兴趣，由于本学期也选上了`网络技术与应用`这门课，感觉这两门课所用的东西是相辅相成的，希望自己可以结合运用，在本学期得到更好的发展。