



南开大学
Nankai University

南 开 大 学

计 算 机 学 院

计算机网络实验报告

配置 WEB 服务器 分析交互过程

汤清云 2013536

年级：2020 级

专业：计算机科学与技术

指导教师：张建忠 徐敬东

2022 年 10 月 27 日

摘要

本次实验从搭建 WEB 服务器开始，对 WEB 服务器和浏览器交互过程进行了仔细分析。结合 CSDN 有关资料，我学习了解了 TCP 协议的建立过程（即三次握手）和解除连接过程（四次挥手），并且在本实验文档中进行了详细说明。

关键词：WEB Wireshark HTML TCP HTTP

目录

一、 实验要求	1
二、 实验过程	1
（一） 搭建 web 服务器	1
（二） 制作 web 页面	3
（三） WireShark 捕捉过程	3
（四） 抓包分析	5
三、 实验总结	9

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。
- (3) 对捕获到的包进行分析说明。
- (4) 提交实验报告。

二、 实验过程

（一） 搭建 web 服务器

所选用平台为 windows，具体过程如下：[5]

1. 控制面板 → 程序和功能 → 启动或关闭程序 windows 功能，将【Internet Information Services】和【Internet Information Services 可承载的 Web 核心】两个选项下的所有子选项勾选，应用并确定。

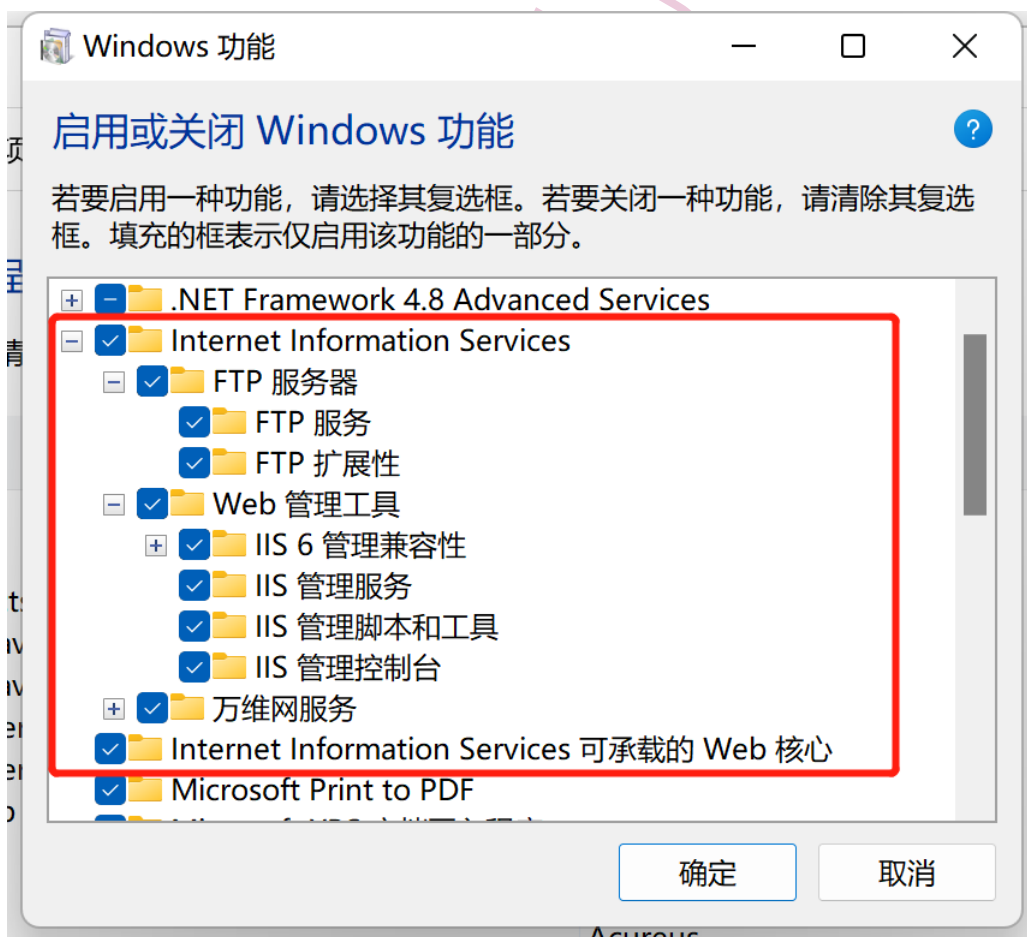


图 1: IIS 配置

2. 开 IIS 管理器，选择添加网站。【网站名称】可以任意填写，【物理路径】为登录此网站后所希望他人能访问的文件夹或文件的具体地址。【端口】设置为 80，【ip 地址】为本机地址。

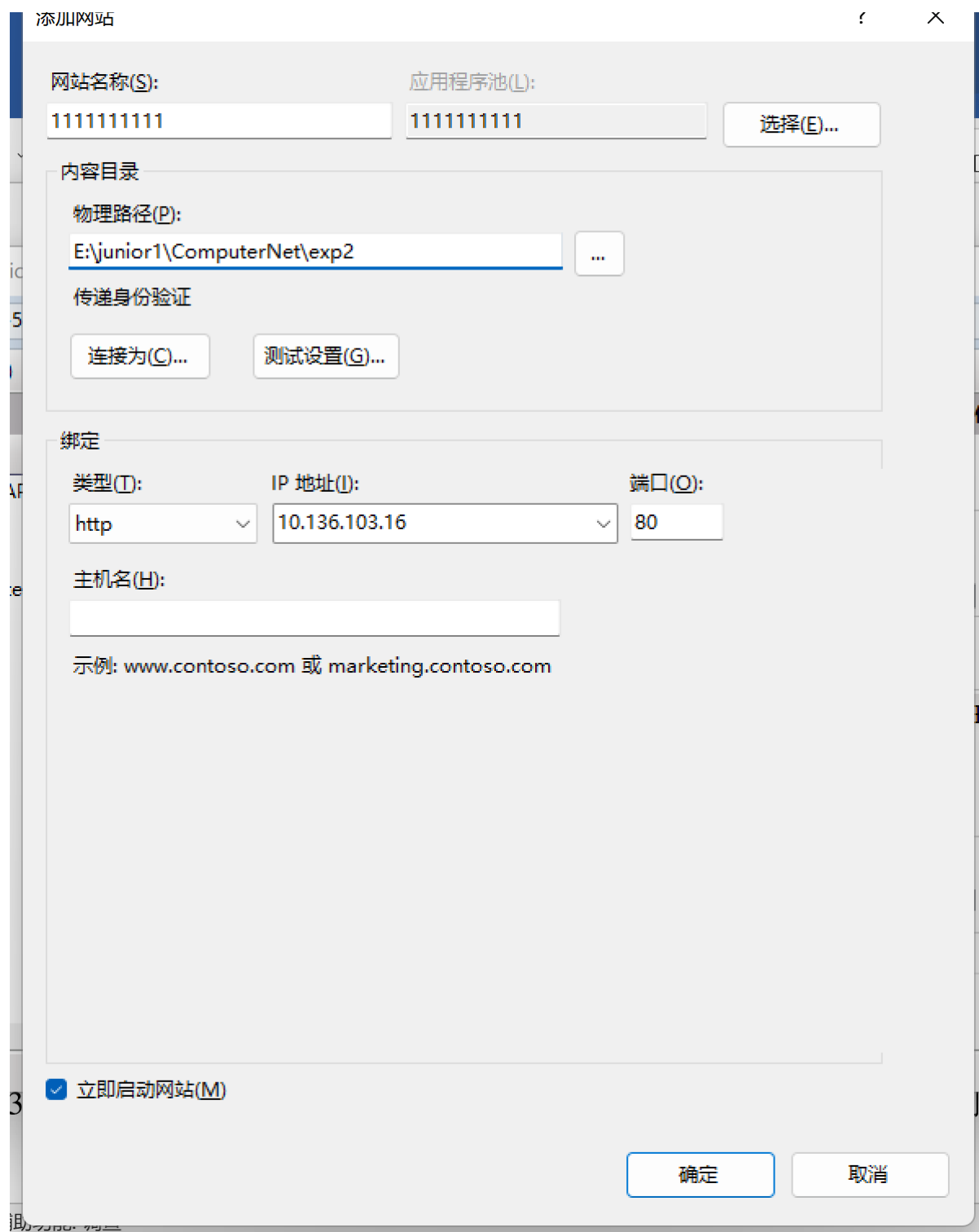


图 2: 配置网站

3. 网站建立成功后, 可以在任意浏览器输入 ip 地址, 查看是否可以正常登录。至此, web 服务器搭建成功。如下图



图 3: 登陆网站结果

(二) 制作 web 页面

在此我使用 Dreamweaver 建立了一个简单的网站，包含我的姓名、学号、学院、专业和一张图片作为自己的 logo。之后将其放入搭建网站的目录下。代码实现如下：

```

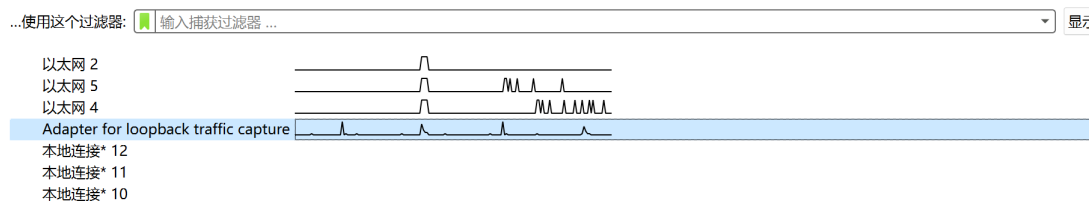
1      <!doctype html>
2      <html>
3      <head>
4      <meta charset="utf-8">
5      <title>ComputerNetLab2</title>
6      </head>
7      <body>
8          <center></center>
9          <center></center>
10         <center>2013536</center>
11         <center>汤清云</center>
12         <center>计算机学院</center>
13         <center>计算机科学与技术</center>
14         <center></
            center>
15     </body>
16 </html>

```

(三) WireShark 捕捉过程

1. 开启 wireshark 软件，由于此处为本机作为客户端去访问本机作为服务器搭建的网站，因此是一个 loopback，选择 Adapter for loopback traffic capture:

捕获



学习

User's Guide · Wiki · Questions and Answers · Mailing Lists

正在运行 Wireshark 4.0.0 (v4.0.0-0-g0cbe09cd796b), 接受自动更新。

图 4: Wireshark 选择

2. 使用浏览器登录界面并打开自建的 html 页面:



图 5: 访问自建 html 页面

3. 由于本次实验为 http 协议, 因此可以查看是否出现 http 协议相关数据流, 下图中 200 OK 说明抓包成功。

6	4.122836	TCP	127.0.0.1	127.0.0.1	44	49694 → 35600 [ACK] Seq=1 Ack=145 Win=8395 Len=0
7	4.830733	TCP	10.136.103.16	10.136.103.16	56	64877 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK
8	4.830829	TCP	10.136.103.16	10.136.103.16	56	80 → 64877 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495
9	4.830897	TCP	10.136.103.16	10.136.103.16	44	64877 → 80 [ACK] Seq=1 Ack=1 Win=327424 Len=0
10	4.831476	TCP	10.136.103.16	10.136.103.16	56	63481 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK
11	4.831565	TCP	10.136.103.16	10.136.103.16	56	80 → 63481 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495
12	4.831626	TCP	10.136.103.16	10.136.103.16	44	63481 → 80 [ACK] Seq=1 Ack=1 Win=327424 Len=0
13	4.840995	HTTP	10.136.103.16	10.136.103.16	502	GET / HTTP/1.1
14	4.841035	TCP	10.136.103.16	10.136.103.16	44	80 → 64877 [ACK] Seq=1 Ack=459 Win=2160640 Len=0
15	4.847511	HTTP	10.136.103.16	10.136.103.16	968	HTTP/1.1 200 OK (text/html)
16	4.847590	TCP	10.136.103.16	10.136.103.16	44	64877 → 80 [ACK] Seq=459 Ack=925 Win=326400 Len=0
17	5.261582	HTTP	10.136.103.16	10.136.103.16	444	GET /favicon.ico HTTP/1.1
18	5.261632	TCP	10.136.103.16	10.136.103.16	44	80 → 64877 [ACK] Seq=925 Ack=859 Win=2160384 Len=0
19	5.262506	HTTP	10.136.103.16	10.136.103.16	4979	HTTP/1.1 404 Not Found (text/html)
20	5.262581	TCP	10.136.103.16	10.136.103.16	44	64877 → 80 [ACK] Seq=859 Ack=5860 Win=321536 Len=0
21	7.616418	TCP	127.0.0.1	127.0.0.1	734	49694 → 35600 [PSH, ACK] Seq=1 Ack=145 Win=8395 Len=100

图 6: Wireshark 抓包成功

(四) 抓包分析

1. 结合我们所学的知识可知道, 在 TCP 协议下会有“三次握手”过程 [3]; 观察上图中所得到的数据包, 发现在 GET HTTP/1.1 之前出现了如下结果: 【SYN】 , 【SYN,ACK】 , 【ACK】, 即三次握手。

2. 对于第一次握手数据包, 客户端会发送一个 TCP 给服务器 [2], 其标志位为 SYN=1, 意为请求连接, 之后自身进入 SYN_SENT 状态; ACK=0, 表示当前没有接收到数据。Seq=0 意味着初始建立连接值为 0, 数据包的相对序列号从 0 开始, 表示当前还没有发送数据。点击该信息, wireshark 左下角会出现此数据包的具体信息。截图如下:

```
> Frame 7: 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: 10.136.103.16, Dst: 10.136.103.16
< Transmission Control Protocol, Src Port: 64877, Dst Port: 80, Seq: 0, Len: 0
  Source Port: 64877
  Destination Port: 80
  [Stream index: 2]
  [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 0]
  Sequence Number: 0 (relative sequence number)
  Sequence Number (raw): 1510585032
  [Next Sequence Number: 1 (relative sequence number)]
  Acknowledgment Number: 0
  Acknowledgment number (raw): 0
  1000 .... = Header Length: 32 bytes (8)
  < Flags: 0x002 (SYN)
  Window: 65535
  [Calculated window size: 65535]
  Checksum: 0x872c [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP),
  > [Timestamps]
```

图 7: 第一次握手

具体解释如下: [1]

第一行 Frame 7 指的是要发送的数据块, 其中, 所抓帧的序号为 7, 捕获字节数等于传送字节数: 56 字节;

第三行为 IPV4 协议, 也称网际协议, 是网络层; 源 IP 地址为 10.136.103.16; 目标 IP 地址为 10.136.103.16, 即一个回环。

展开部分为 TCP 协议, 也称传输控制协议, 是传输层; 源端口 (64877); 目标端口 (80); 序列号 (0); 长度为 0; 窗口大小为 65535

3. 对于第二次握手数据包 【SYN,ACK】, 实际上是服务器发回的确认包, 表示同意建立连接 SYN=1, 表示请求连接; ACK=1。Seq = 0 表示初始建立值为 0, 表示当前还没有发送数据; Ack = 1: 表示当前端成功接收的数据位数, 虽然此时客户端没有发送任何有效数据, 但是确认号还是被加 1, 因为包含 SYN 或 FIN 标志位。(不会对有效数据的计数产生影响, 因为含有 SYN 或 FIN 标志位的包并不携带有效数据)

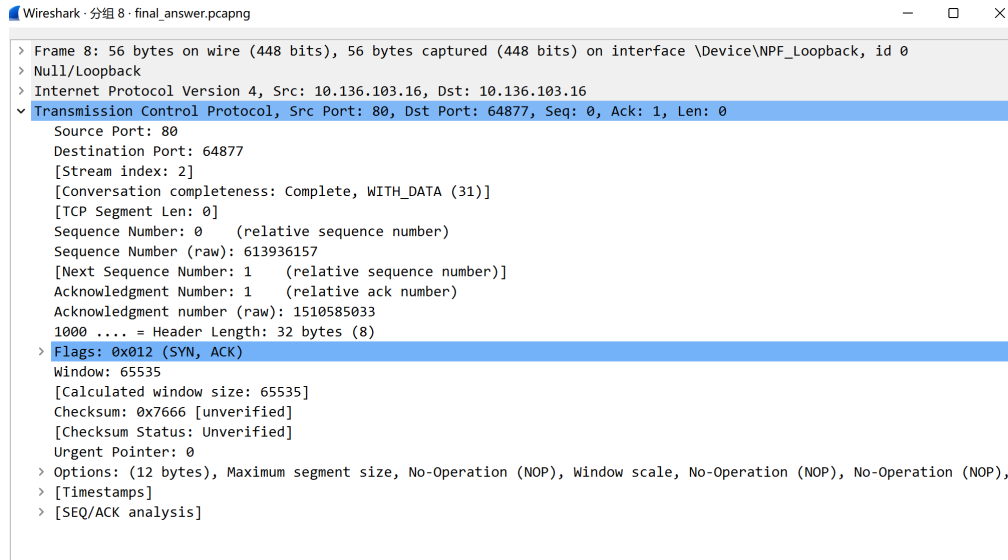


图 8: 第二次握手

4. 对于第三次握手数据包【ACK】，是客户端再次发送确认包 (ACK)，表示已经收到来自服务器的连接，SYN=0，ACK=1。Seq = 1：表示当前已经发送 1 个数据。Ack = 1：表示当前端成功接收的数据位数。

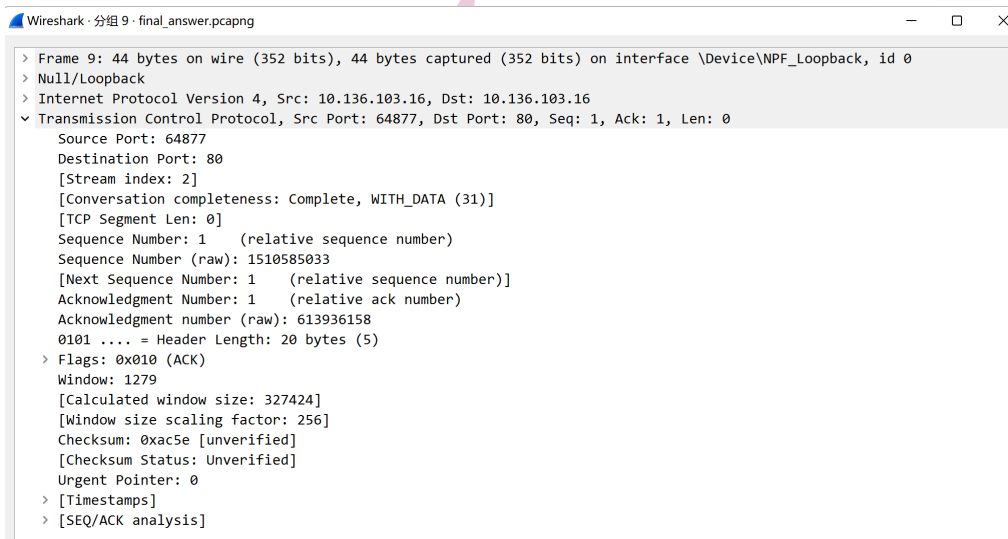


图 9: 第三次握手

5. 三次握手后，浏览器和目的主机才开始传输数据和进行 http 访问。首先浏览器向域名发出 GET 请求的报文 (HTTP 请求)。

16 4.847590	TCP	10.136.103.16	10.136.103.16	44 64877 → 80 [ACK] Seq=459 Ack=925 Win=326400 Len=0
17 5.261582	HTTP	10.136.103.16	10.136.103.16	444 GET /favicon.ico HTTP/1.1
18 5.261632	TCP	10.136.103.16	10.136.103.16	44 80 → 64877 [ACK] Seq=925 Ack=859 Win=2160384 Len=0
19 5.262506	HTTP	10.136.103.16	10.136.103.16	4979 HTTP/1.1 404 Not Found (text/html)
20 5.262581	TCP	10.136.103.16	10.136.103.16	44 64877 → 80 [ACK] Seq=859 Ack=5860 Win=321536 Len=0
420 12.233076	HTTP	10.136.103.16	10.136.103.16	628 GET /Lab2.html HTTP/1.1
421 12.233111	TCP	10.136.103.16	10.136.103.16	44 80 → 64877 [ACK] Seq=5860 Ack=1443 Win=2159872 Len=0
422 12.234902	HTTP	10.136.103.16	10.136.103.16	684 HTTP/1.1 200 OK (text/html)
423 12.234968	TCP	10.136.103.16	10.136.103.16	44 64877 → 80 [ACK] Seq=1443 Ack=6500 Win=2160640 Len=0
887 30.075065	TCP	10.136.103.16	10.136.103.16	44 64877 → 80 [FIN, ACK] Seq=1443 Ack=6500 Win=2160640 Le
888 30.075084	TCP	10.136.103.16	10.136.103.16	44 80 → 64877 [ACK] Seq=6500 Ack=1444 Win=2159872 Len=0
890 30.082543	TCP	10.136.103.16	10.136.103.16	44 80 → 64877 [FIN, ACK] Seq=6500 Ack=1444 Win=2159872 Le
891 30.082604	TCP	10.136.103.16	10.136.103.16	44 64877 → 80 [ACK] Seq=1444 Ack=6501 Win=2160640 Len=0

图 10: GET 请求报文

首先我们可以看到，此时使用的协议为 HTTP1.1，请求访问的界面是 Lab.html，请求访问方式为 GET。

理想状态下，TCP 连接一旦建立，在通信双方的任何一方关闭连接之前，此连接都会一直保持。捕捉到的发出 GET 请求的包具体内容如下：

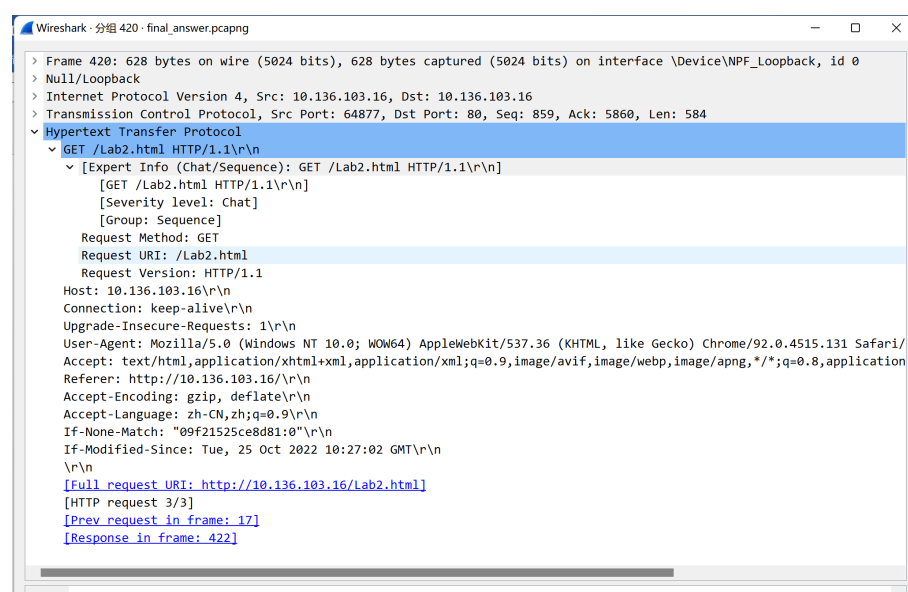


图 11: GET 具体信息

结合相关资料我们可以知道：

一个 HTTP 请求由四个部分组成：请求行、请求头部、空行、请求数据。

请求行即图中的下列代码，它由三部分组成：请求方法字段、URL 字段和 HTTP 协议版本字段，三个字段以空格分隔开。在本次实验中，请求方法为 GET，URL 字段即当前目录下的 Lab2.html 文件；HTTP 协议版本为 1.1

```
1 GET /Lab2.html HTTP/1.1\r\n
```

紧跟在请求行之后的是请求头部，浏览器在向服务器发送请求的时候必须指明请求类型（一般是 GET 或者 POST）。如有必要，浏览器还可以选择发送其他的请求头。大多数请求头并不是必需的，但 Content-Length 除外。对于 POST 请求来说 Content-Length 必须出现。而在上面截图的最后一部分可以看到一行空行，只有换行符，其意为告诉服务器请求头部到此截止。

空行之后是请求数据，由于此处为 GET 请求，故而没有数据（若为 POST，此处为要提交的数据）

6. 在请求被目的主机通过后，回应其封装好的 HTML 形式数据，称为“响应报文”，如下图。200 OK 表示客户端请求成功

18	5.261632	TCP	10.136.103.16	10.136.103.16	44	80 → 64877 [ACK] Seq=925 Ack=859 Win=2160
19	5.262506	HTTP	10.136.103.16	10.136.103.16	4979	HTTP/1.1 404 Not Found (text/html)
20	5.262581	TCP	10.136.103.16	10.136.103.16	44	64877 → 80 [ACK] Seq=859 Ack=5860 Win=321
420	12.233076	HTTP	10.136.103.16	10.136.103.16	628	GET /Lab2.html HTTP/1.1
421	12.233111	TCP	10.136.103.16	10.136.103.16	44	80 → 64877 [ACK] Seq=5860 Ack=1443 Win=21
422	12.234902	HTTP	10.136.103.16	10.136.103.16	684	HTTP/1.1 200 OK (text/html)
423	12.234968	TCP	10.136.103.16	10.136.103.16	44	64877 → 80 [ACK] Seq=1443 Ack=6500 Win=21
887	30.075065	TCP	10.136.103.16	10.136.103.16	44	64877 → 80 [FIN, ACK] Seq=1443 Ack=6500 W
888	30.075084	TCP	10.136.103.16	10.136.103.16	44	80 → 64877 [ACK] Seq=6500 Ack=1444 Win=21
890	30.082543	TCP	10.136.103.16	10.136.103.16	44	80 → 64877 [FIN, ACK] Seq=6500 Ack=1444 W
891	30.082604	TCP	10.136.103.16	10.136.103.16	44	64877 → 80 [ACK] Seq=1444 Ack=6501 Win=21

图 12: 响应报文

具体结果如下图:

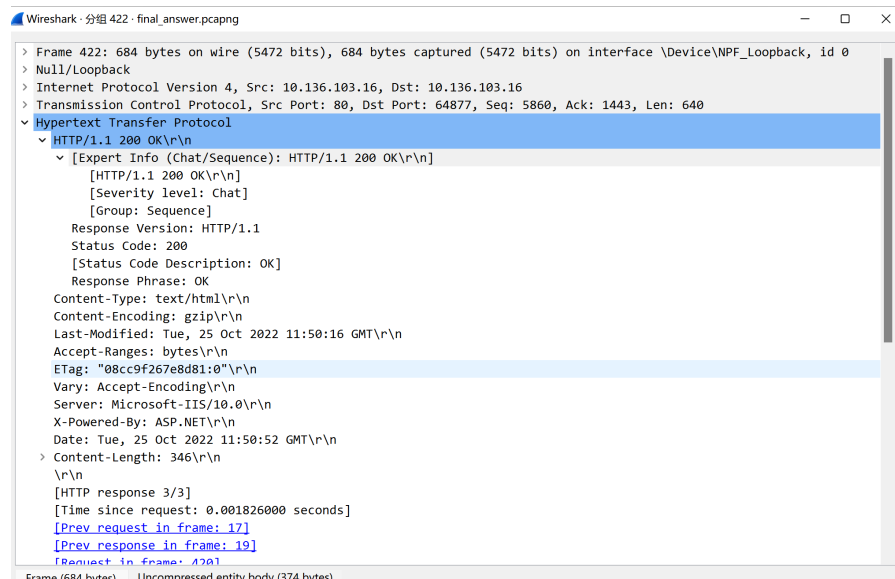


图 13: 响应报文具体信息

结合相关资料可以知道, 响应报文由状态行、响应头部、空行和响应体四部分组成。

响应行一般由协议版本、状态码及其描述组成。本实验中为 HTTP/1.1 200 OK。

响应头用于描述服务器的基本信息, 以及数据的描述, 服务器通过这些数据的描述信息, 可以通知客户端如何处理等一会儿它回送的数据。Content-Length 指明了返回的内容长度为 346。

与 HTTP 请求报文类似, 此处空行的意义为响应头部结束。

响应体就是响应的消息体, 本次实验为访问 html 界面, 故而返回所编写的 HTML 代码, 如下图所示。

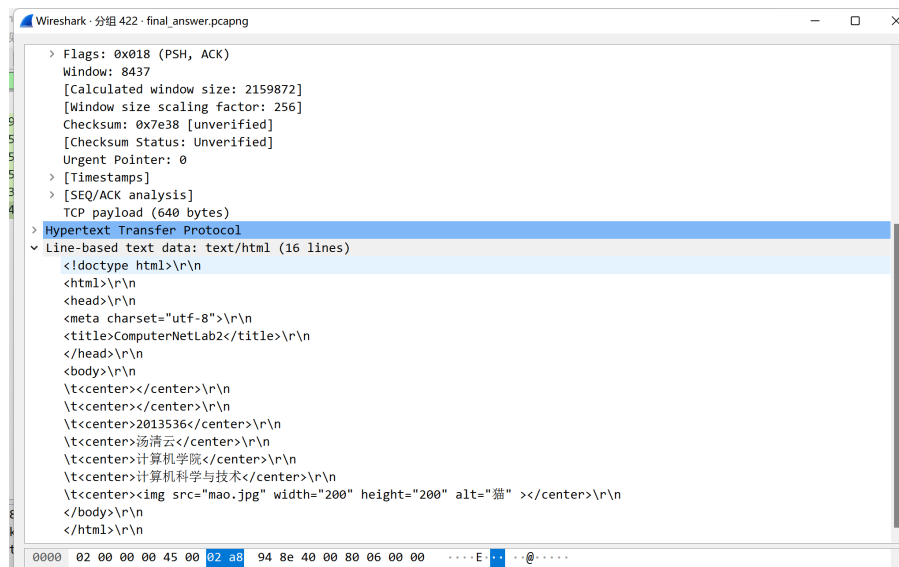


图 14: 响应报文数据

7. 之后关闭网站代表断开连接, TCP 使用“四次挥手”来关闭一个连接 [4]。具体如下图:

881 27.339093	TCP	127.0.0.1	127.0.0.1	45 50966 → 50911 [ACK] Seq=1 Ack=1 Win=8440 Len=1
882 27.399124	TCP	127.0.0.1	127.0.0.1	56 50911 → 50964 [ACK] Seq=1 Ack=2 Win=8440 Len=0 SLE=1 SRE=2
883 27.621124	TCP	127.0.0.1	127.0.0.1	45 50966 → 50911 [ACK] Seq=1 Ack=1 Win=8442 Len=1
884 27.621169	TCP	127.0.0.1	127.0.0.1	56 50911 → 50966 [ACK] Seq=1 Ack=2 Win=8439 Len=0 SLE=1 SRE=2
885 30.074941	TCP	10.136.103.16	10.136.103.16	44 63481 → 80 [FIN, ACK] Seq=1 Ack=1 Win=327424 Len=0
886 30.074977	TCP	10.136.103.16	10.136.103.16	44 80 → 63481 [ACK] Seq=1 Ack=2 Win=2161152 Len=0
887 30.075065	TCP	10.136.103.16	10.136.103.16	44 64877 → 80 [FIN, ACK] Seq=1443 Ack=6500 Win=2160640 Len=0
888 30.075084	TCP	10.136.103.16	10.136.103.16	44 80 → 64877 [ACK] Seq=6500 Ack=1444 Win=2159872 Len=0
889 30.082320	TCP	10.136.103.16	10.136.103.16	44 80 → 63481 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
890 30.082543	TCP	10.136.103.16	10.136.103.16	44 80 → 64877 [FIN, ACK] Seq=6500 Ack=1444 Win=2159872 Len=0
891 30.082604	TCP	10.136.103.16	10.136.103.16	44 64877 → 80 [ACK] Seq=1444 Ack=6501 Win=2160640 Len=0

me 7: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface 1/Loopback
 Ethernet Protocol Version 4, Src: 10.136.103.16, Dst: 10.136.103.16

图 15: 四次挥手

第一次挥手: 主动关闭方 (Client) 发送一个请求结束的报文【FIN,ACK】给 Server 以关闭 Client 到 Server 的数据传送, 并进入 FIN_WAIT1 状态。

第二次挥手: 被动关闭方 (Server) 接收到主动关闭方 (Client) 发送的【FIN,ACK】并发送【ACK】, 表示确认。确认序号 = 收到序号 + 1; 此时被动关闭方 (Server) 进入 CLOSE_WAIT 状态; 主动关闭方 (Client) 收到被动关闭方 (Server) 的【ACK】后, 进入 FIN_WAIT2 状态。

第三次挥手: 被动关闭方 (Server) 发送一个【FIN,ACK】以关闭 Server 到 Client 的数据传送, 并进入 LAST_ACK 状态。

第四次挥手: 主动关闭方 (Client) 收到被动关闭方 (Server) 发送的【FIN,ACK】并发送【ACK】, 此时主动关闭方 (Client) 进入 TIME_WAIT 状态, 经过 2MSL 时间后关闭连接; 被动关闭方收到主动关闭方的【ACK】后, 关闭连接。

三、实验总结

在这次实验中, 我学习了如何搭建 web 服务器以及服务器和浏览器的交互过程。对我之后计算机网络的学习很有帮助。

参考文献

- [1] Http 请求分析. <https://blog.csdn.net/w849593893/article/details/105724807>.
- [2] Tcp 报文分析. <https://blog.csdn.net/hebbely/article/details/54424823>.
- [3] 三次握手抓包分析. https://blog.csdn.net/qq_38937634/article/details/111773243.
- [4] 四次挥手. <https://blog.csdn.net/chun0801/article/details/94989287>.
- [5] 搭建 web 服务器. https://blog.csdn.net/qq_33413510/article/details/125447301.

NIKE