南開大學
Nankai University

计算机学院

计算机网络实验报告

实验二　Web 服务器抓包分析

姓名：赵康明

学号：2110937

专业：计算机科学与技术

2023 年 11 月 3 日

目录

1 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）、自己的 LOGO、自我介绍的音频信息。页面不要太复杂，包含要求的基本信息即可。
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. 使用 HTTP，不要使用 HTTPS。
4. 提交实验报告。

2 Web 服务器搭建

2.1 框架代码

本次网页框架选取的是 python 语言中的 Flask 框架，选定 host 主机为本机（缺省），监听端口号指定为 8000，即设置为回环地址，将数据包从本机发送至本机。框架代码如下：

```python
from flask import Flask, render_template

app = Flask(__name__, static_folder='static')

# 定义一个简单的路由，用于渲染网页
@app.route('/')
def hello_world():
    return render_template('index.html')

if __name__ == '__main__':
    app.run(port=8000)
```

2.2 网页代码

编写网页代码如下，涉及到了个人有关信息的文本、背景图片的设置以及音频的插入：

```html
<!DOCTYPE html>
<html>
<head>
    <title>赵康明的主页</title>
    <style>
        body {
            background-image: url('/static/cover.jpeg');
            background-size: cover;
            background-repeat: no-repeat;
```

```
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
            position: relative;
        }
        h1, p {
            text-align: center; /* 将文本居中对齐 */
            font-family: "楷体"; /* 设置字体为楷体 */
            color: white; /* 设置文字颜色为白色 */
            font-size: 40px; /* 调整字体大小 */
        }
        #logo {
            position: absolute;
            top: 20px;
            left: 20px;
            width: 100px; /* 调整Logo的宽度 */
        }
        #playButton {
            position: absolute;
            top: 20px;
            right: 20px;
            background-color: #3498db;
            color: white;
            border: none;
            padding: 10px 20px;
            cursor: pointer;
        }
    </style>
</head>
<body>
    <img id="logo" src="/static/logo.png" alt="Logo">
    <p>学号: 2110937</p>
    <p>姓名: 赵康明</p>
    <p>专业: 计算机科学与技术</p>

    <!-- 添加JavaScript代码来播放音乐 -->
    <audio id="music" src="/static/bgm.mp3"></audio>
    <button id="playButton">播放音乐</button>

    <script>
```

```
        var music = document.getElementById("music");
        var playButton = document.getElementById("playButton");
        var isPlaying = false;

        playButton.addEventListener("click", function() {
            if (isPlaying) {
                music.pause();
                playButton.textContent = "播放音乐";
            } else {
                music.play();
                playButton.textContent = "暂停音乐";
            }
            isPlaying = !isPlaying;
        });
    </script>
</body>
</html>
```

当运行我们的 main.py 后，可以通过访问 http://127.0.0.1:8000 访问我们的页面，运行效果如下：

图 2.1: 代码运行图

网页效果展示：当运行我们的 main.py 后，可以通过访问 http://127.0.0.1:8000 访问我们的页面，左上角为个人 logo，右上角为播放音乐按钮，中间为个人信息文本，运行效果如下：

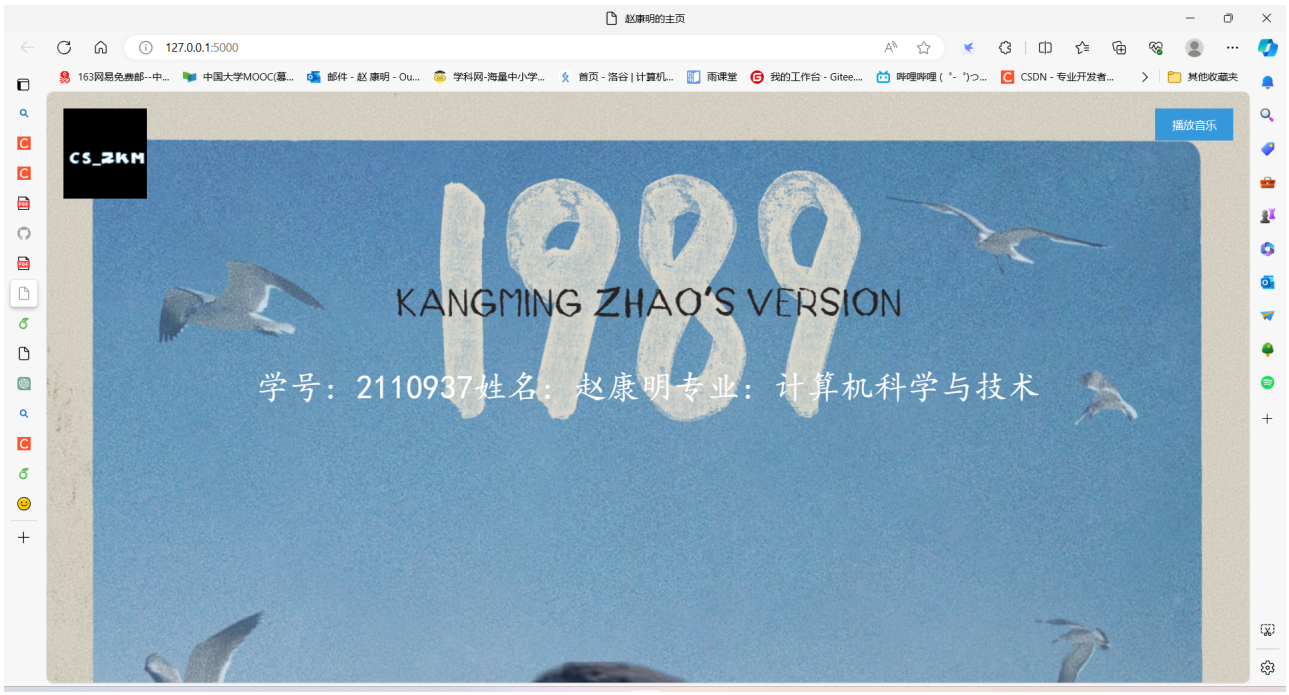

图 2.2: 网页实现效果图

3 WireShark 抓包过程及其分析

3.1 WireShark 启动

首先需要我们启动 WireShark 软件，然后再运行 main.py 文件进行访问。注意本次发送端和接收端均为本机，windows 系统没有提供本地回环网络的接口，用 wireshark 监控网络的话只能看到经过网卡的流量，看不到访问 localhost 的流量，因为 wireshark 在 windows 系统上默认使用的是 WinPcap 来抓包的，需要我们在安装 WireShark 的基础上再安装一个 Npcap 替换 WinPcap 进行抓包。

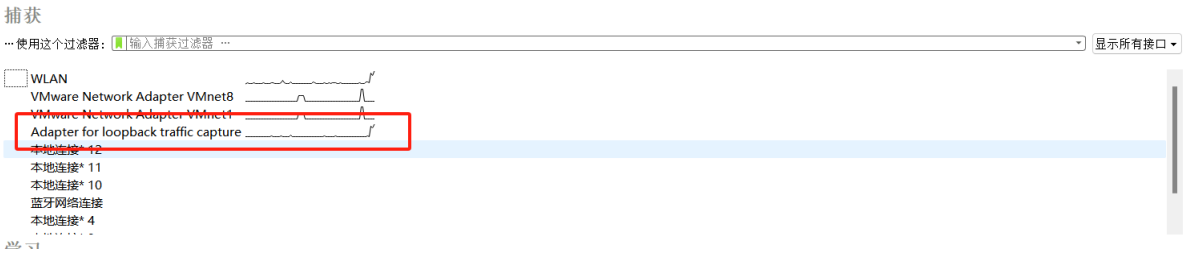

图 3.3: wireshark

此时我们应当选择图中红圈处的 **Adapter for loopback traffic capture** 进行抓包分析。开始抓包后我们运行代码，启动服务器，并在捕获过滤器中使用 **tcp.port==8000** 进行过滤，得到数据包，以便于下一步分析。

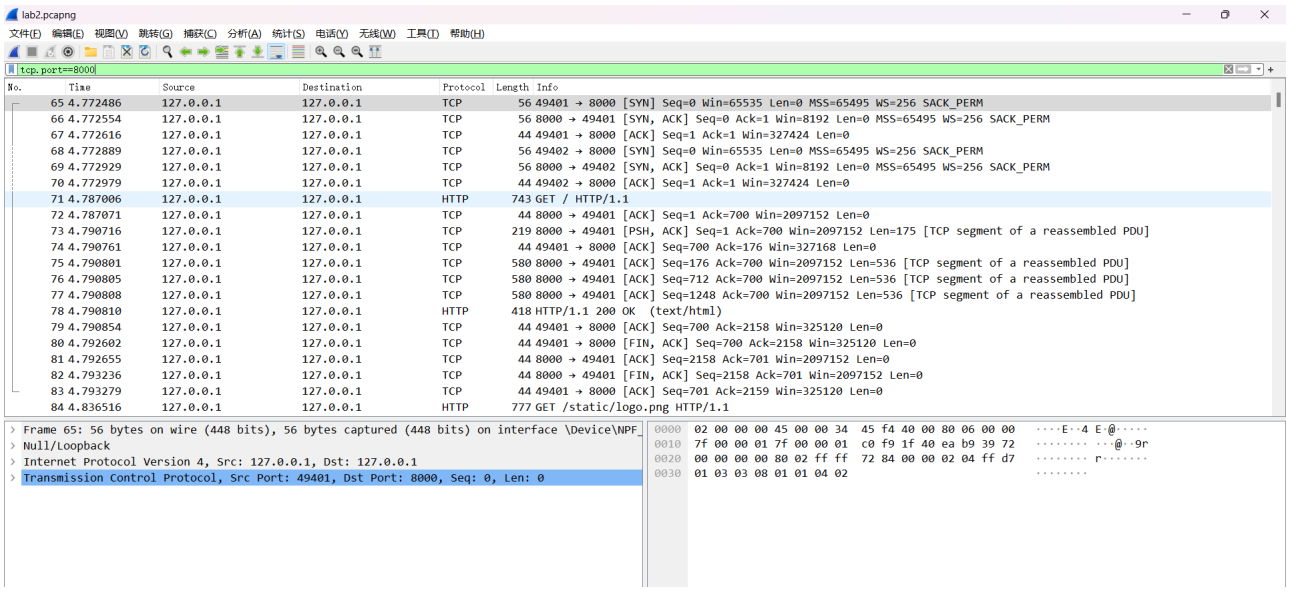

图 3.4: 数据包捕获结果

3.2　三次握手

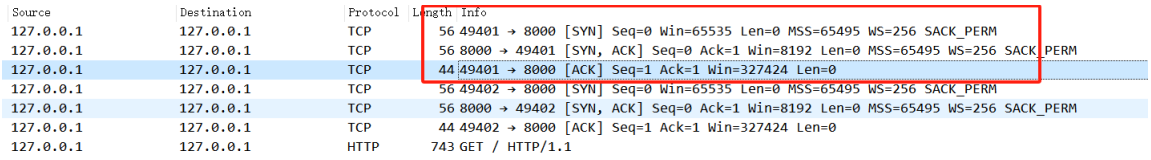

图 3.5: 三次握手

1. 第一次握手：客户端发送序列号 Seq=0 的 SYN 报文到服务器，表示请求建立连接。

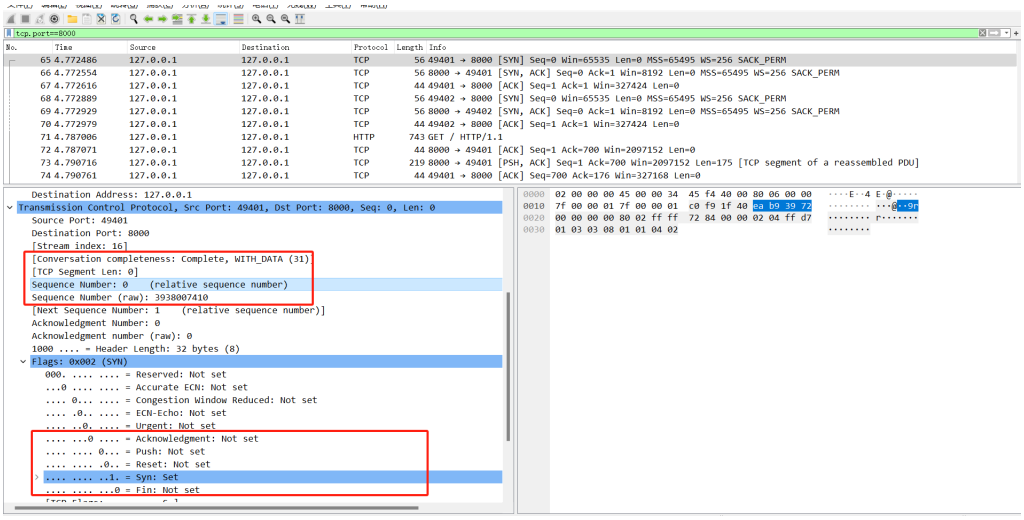

图 3.6: 第一次握手

2. 第二次握手：服务器收到客户端的 SYN 报文，发送序列号 Seq=0、确认号 Ack=1 的 SYN, ACK 报文到客户端，表示同意建立连接。

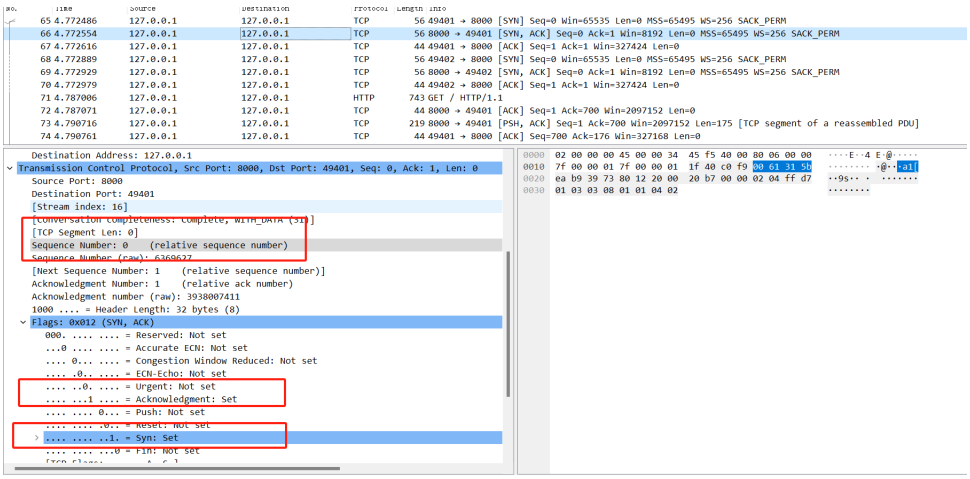

图 3.7: 第二次握手

3. 第三次握手：客户端收到服务器的 SYN, ACK 报文，发送序列号 Seq=1、确认号 Ack=1 的 ACK 报文到服务器，表示确认建立连接。

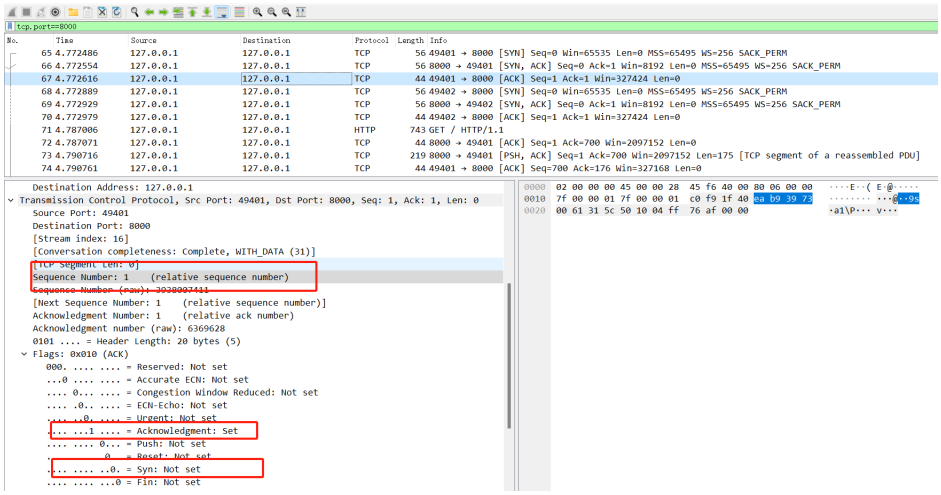

图 3.8: 第三次握手

由图中我们可以发现在 GET 请求之前，有两个端口都进行了三次握手连接，经过查阅资料，是为多线程考虑，创建两个线程，独立接收文件，使得效率更高。所以此处出现了两个客户端端口进行三次握手连接。连接完毕后发出 GET 请求

3.3 Http 请求报文

请求报文如下：

```
Hypertext Transfer Protocol
GET / HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
    Request Method: GET
    Request URI: /
```

```
    Request Version: HTTP/1.1
  Host: 127.0.0.1:8000\r\n
  Connection: keep-alive\r\n
  sec-ch-ua: "Chromium";v="118", "Microsoft Edge";v="118",
      "Not=A?Brand";v="99"\r\n
  sec-ch-ua-mobile: ?0\r\n
  sec-ch-ua-platform: "Windows"\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/118.0.0.0 Safari/537.36 Edg/118.0.2088.69\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=0.7\r\n
  Sec-Fetch-Site: none\r\n
  Sec-Fetch-Mode: navigate\r\n
  Sec-Fetch-User: ?1\r\n
  Sec-Fetch-Dest: document\r\n
  Accept-Encoding: gzip, deflate, br\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
```

3.3.1 请求行

请求行主要由三部分组成：请求方法、URL 和 HTTP 协议版本。

- 请求方法：GET, POST, HEAD, PUT, DELETE。此处为 GET。
- 请求 UR:：在本实验中，由于是同一台主机，于是省略了 URL。
- http 版本：在本实验中是 HTTP/1.1。

3.3.2 请求头

- Host: 127.0.0.1:8000
  目标主机的 IP 地址和端口号，指示请求要发送到哪个主机和端口。
- Connection: keep-alive
  指示客户端希望保持连接活动状态，以便在同一连接上发送多个请求和响应。
- sec-ch-ua: ”Chromium”;v=”118”, ”Microsoft Edge”;v=”118”, ”Not=A?Brand”;v=”99”
  用户代理字符串，提供有关浏览器的信息，包括名称和版本。
- sec-ch-ua-mobile: ?0
  用户代理字符串的移动版本信息，表示浏览器不是移动版本。
- sec-ch-ua-platform: ”Windows”
  用户代理字符串的平台信息，表明浏览器在 Windows 操作系统上运行。

- Upgrade-Insecure-Requests: 1
  指示浏览器希望在 HTTPS 上进行连接，以提高安全性。
- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 Edg/118.0.2088.69
  用户代理字符串，提供了有关浏览器的详细信息，包括浏览器的名称、版本、操作系统和渲染引擎等。
- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
  image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
  指示客户端可以接受的响应类型，例如 HTML、XML 和图像等。
- Sec-Fetch-Site: none
  指示请求的来源，‘none‘ 表示没有引用来源。
- Sec-Fetch-Mode: navigate
  指示请求的类型，‘navigate‘ 表示导航请求，通常是用户输入的 URL。
- Sec-Fetch-User: ?1
  指示用户代理是否包含有关用户的信息，‘?1‘ 表示包含了一些信息。
- Sec-Fetch-Dest: document
  指示请求的目标类型，‘document‘ 表示请求的是文档类型的资源。
- Accept-Encoding: gzip, deflate, br
  指定客户端接受的内容编码方法，例如 gzip、deflate 和 br（Brotli）。
- Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
  指定客户端所接受的语言首选项，以便服务器可以选择最适合的语言进行响应。

3.3.3 请求体

GET 请求通常用于请求资源，例如获取网页、图像、文档等，而请求参数通常用于向服务器传递查询条件。GET 请求的数据通常不包含在请求体中，而是包含在 URL 中，因此没有请求体。

3.4 Http 响应报文

响应报文如下：

```
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
    [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
    Response Version: HTTP/1.1
    Status Code: 200
    [Status Code Description: OK]
    Response Phrase: OK
Server: Werkzeug/3.0.1 Python/3.10.7\r\n
Date: Sun, 29 Oct 2023 13:12:17 GMT\r\n
```

```
Content-Type: text/html; charset=utf-8\r\n
Content-Length: 1982\r\n
Connection: close\r\n
\r\n
[HTTP response 1/1]
[Time since request: 0.003804000 seconds]
[Request in frame: 71]
[Request URI: http://127.0.0.1:8000/]
File Data: 1982 bytes
```

3.4.1 响应行

响应行包括：Http 协议版本和状态码。

- Http:HTTP 协议版本为 1.1
- 状态码：200 请求成功

3.4.2 响应头

- **Server: Werkzeug/3.0.1 Python/3.10.7**
  服务器的标识信息，指示服务器正在使用 Werkzeug 框架的版本 3.0.1 和 Python 的版本 3.10.7 来处理请求。
- **Date: Sun, 29 Oct 2023 13:12:17 GMT**
  响应生成的日期和时间，以 GMT 时区表示，指示响应是在 2023 年 10 月 29 日 13:12:17（格林威治标准时间）生成的。
- **Content-Type: text/html; charset=utf-8**
  响应体的内容类型，表明响应的内容是 HTML 文本，并使用 UTF-8 字符集编码。
- **Content-Length: 1982**
  响应体的长度。
- **Connection: close**
  指示连接将被关闭。
- **[HTTP response 1/1]**
  HTTP 响应相关的信息。
- **[Time since request: 0.003804000 seconds]**
  请求的响应时间相关的信息。
- **[Request in frame: 71]**
  请求帧。
- **[Request URI: http://127.0.0.1:8000/]**
  这是请求的 URI，指示请求的目标 URL。

3.4.3 响应体

响应体为所请求的数据，本次实验即为编写 html 网页内容。

3.5 四次挥手

四次挥手过程：

1. 第一次挥手：客户端（40920）发送 FIN，ACK 报文至服务端（8000），请求关闭客户端到服务端之间的连接。
2. 第二次挥手：服务端收到 FIN，ACK 报文，ACK=Seq+1，将 ACK 发回客户端，表示确认收到终止连接的请求。
3. 第三次挥手：服务端发送一个 FIN，ACK 报文给客户端，请求关闭服务端到客户端的连接。
4. 第四次挥手：客户端收到 Fin，ACK 报文后，ACK=Seq+1，发送 ACK 报文给服务端，表示确认收到终止连接的请求，双方终止连接。

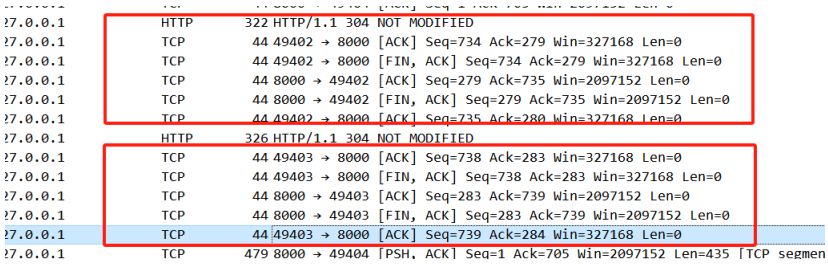

图 3.9: 四次挥手

经历完以上四次挥手，客户端与服务端断开连接，访问结束。

4 实验总结与思考

本次实验使用了 WireShark 进行数据抓包分析，实验较为简单，深刻了解了三次握手建立连接和四次挥手断开连接的详细过程，同时还重点学习了 Http 请求报文和响应报文的有关内容。在此基础上还了解了浏览器会采用多线程进行收发消息提高效率的这一举措。