南开大学
Nankai University

南　　开　　大　　学

计算机网络实验报告

Lab2

姓名：李雅帆

学号：2213041

年级：2022 级

专业：信息安全

2024 年 11 月 1 日

目录

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。
2. 通过浏览器访问 Web 服务器，获取自己编写的 HTML 文档，并显示 Web 页面。
3. 在获取 Web 页面的同时，使用 Wireshark 捕获与 Web 服务器的交互过程，设置过滤器使 Wireshark 仅显示 HTTP 报文，并详细说明 HTTP 交互过程。

二、 Web 服务器搭建

1. 安装 HBuilder

本实验借助 HBuilder 编写的 HTML 文档，完成 Web 页面的显示。

2. 运行服务器并通过浏览器访问 Web 页面

在 HBuilder 中，运行-运行到浏览器，选择你要使用的浏览器访问 Web 页面；也可以直接打开浏览器并访问 Web 服务器页面，即 http://127.0.0.1:8848/Web/index.html。

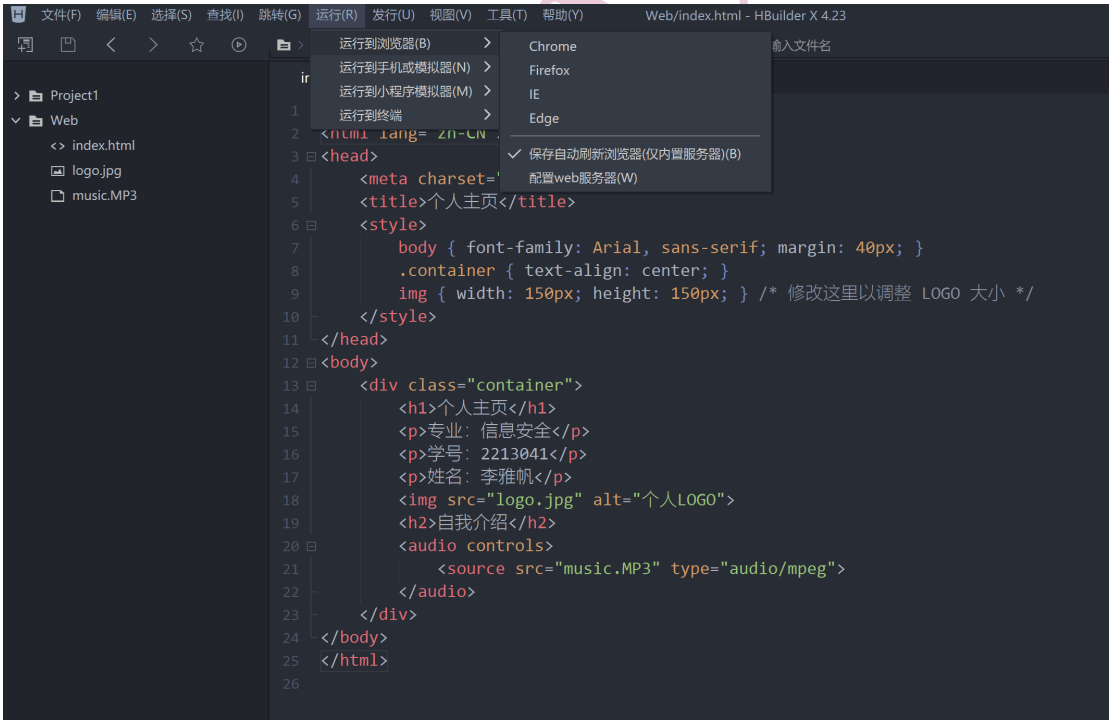

图 1: 运行服务器

图 2: Web 页面

三、 编写 Web 页面

1. HTML 代码

我编写的 HTML 文件代码如下:

HTML 文件

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <title>个人主页</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 40px; }
        .container { text-align: center; }
        img { width: 150px; height: 150px; }
    </style>
</head>
<body>
    <div class="container">
        <h1>个人主页</h1>
        <p>专业：信息安全</p>
        <p>学号：2213041</p>
        <p>姓名：李雅帆</p>
        <img src="logo.jpg" alt="个人LOGO">
```

```
        <h2>自我介绍</h2>
        <audio controls>
            <source src="music.MP3" type="audio/mpeg">
        </audio>
    </div>
</body>
</html>
```

2. HTML 结构说明

1. 文档声明和基本设置
   - `<!DOCTYPE html>`：指定 HTML5 文档类型。
   - `<html lang="zh-CN">`：声明页面的语言为简体中文（中国）。
   - `<meta charset="UTF-8">`：指定字符编码为 UTF-8，确保中文字符显示正确。
   - `<title> 个人主页 </title>`：设置网页的标题栏显示为“个人主页”。
2. 内嵌 CSS 样式
   - `body`：设置字体为 Arial，页面四周的边距为 40 像素。
   - `.container`：将页面内容居中对齐。
   - `img`：设置 LOGO 图片的宽度和高度均为 150 像素，允许在这里调整图片大小。
3. 页面主体内容
   - `<body>`：定义网页的主体部分。
   - `<div class="container">`：将主要内容放入一个居中的容器中。
   - `<h1> 个人主页 </h1>`：标题，显示为“个人主页”。
   - `<p>` 标签：显示专业、学号、姓名信息：
     - 专业：信息安全
     - 学号：2213041
     - 姓名：李雅帆
   - `<img src="logo.jpg" alt=" 个人 LOGO">`：嵌入一张图片作为 LOGO，来源文件为 `logo.jpg`，`alt` 属性提供图片的替代文本“个人 LOGO”。
   - `<h2> 自我介绍 </h2>`：二级标题，显示“自我介绍”。
   - `<audio controls>`：音频控件，包含播放/暂停等按钮。
   - `<source src="music.MP3" type="audio/mpeg">`:定义音频文件的路径为 `music.MP3`，类型为 `audio/mpeg`，支持 MP3 格式。

四、 Wireshark 捕获交互过程

(一) 基本信息

- 在使用 Wireshark 软件时，应选择适当的网络端口以进行数据包捕获。本次实验中开启的 Web 服务器地址为“127.0.0.1”（即本地回环地址），因此在 Wireshark 选择端口时，应选择“Adapter for loopback traffic capture”，即捕获本地回环（loopback）流量的适配器。

- 设置过滤器 (ip.dst == 127.0.0.1 or ip.src == 127.0.0.1) and (tcp.srcport == 8080 or tcp.dstport == 8080)

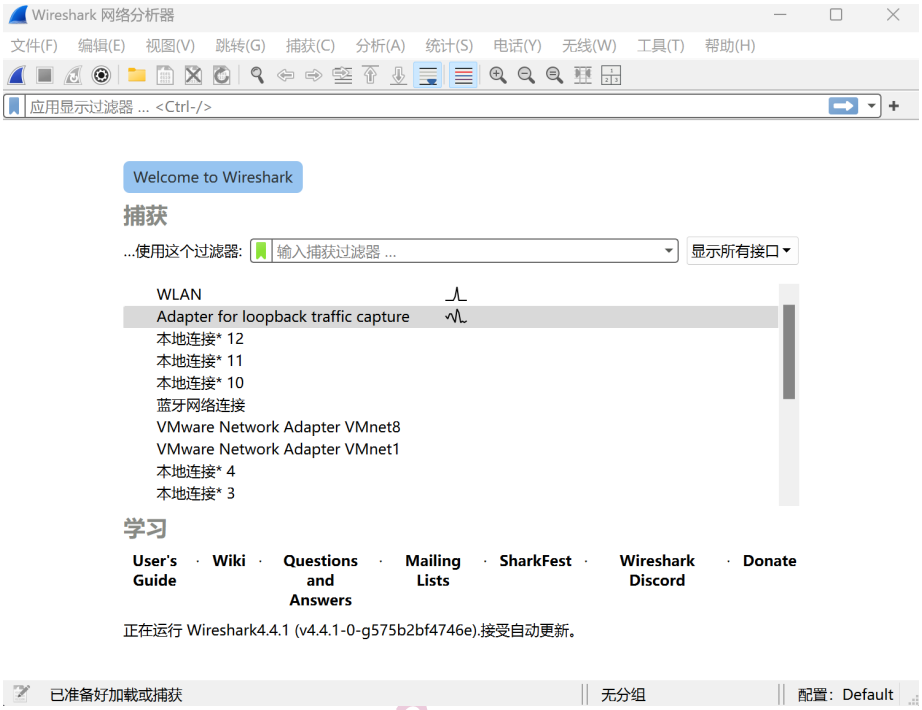

图 3

(二) 数据分析

在启动 Wireshark 进行数据捕获后，利用浏览器访问了上文搭建的网页。随后，可以通过分析捕获到的数据内容，进一步研究其之间的交互过程。经由过滤器进行筛选，主要对协议类型为 HTTP 的数据包进行分析。

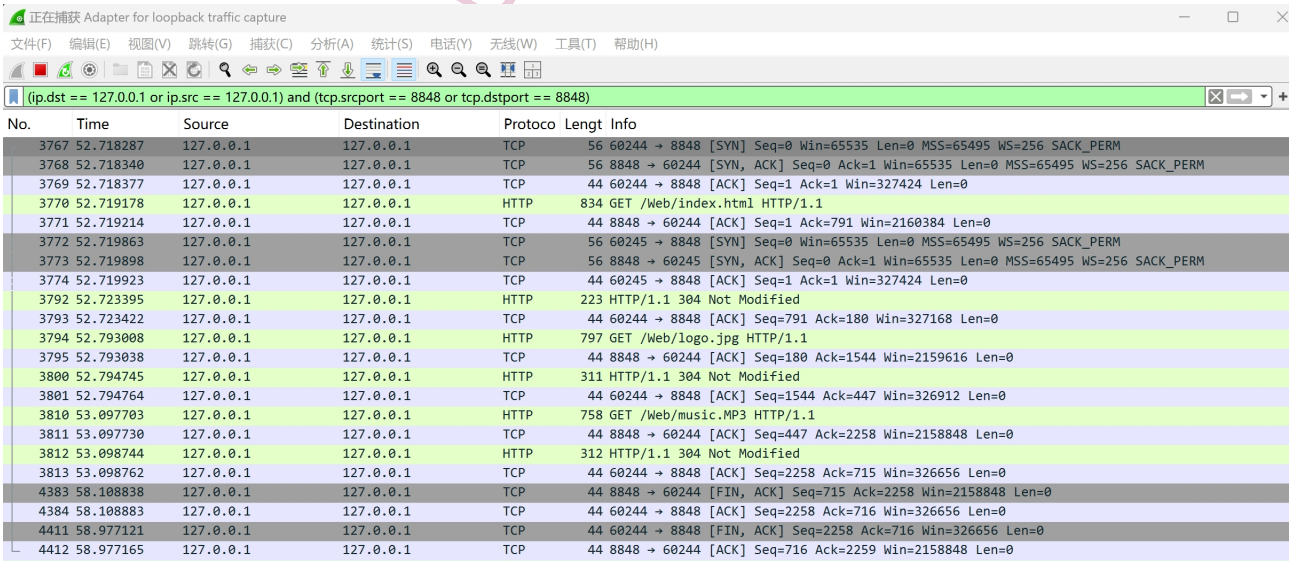

图 4: 捕获结果

1. TCP 三次握手建立连接

TCP 使用三次握手建立连接。

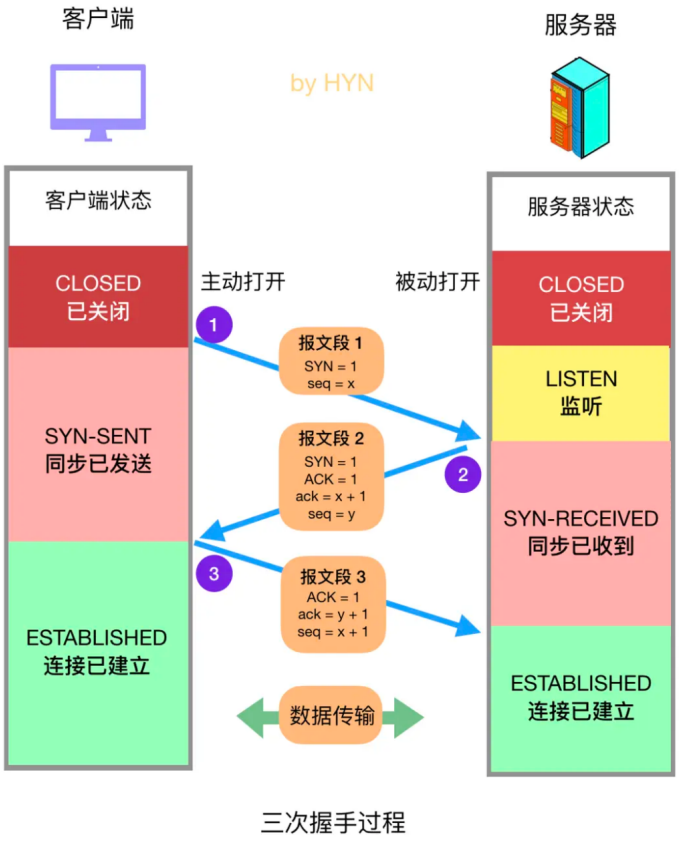

图 5

在 wireshark 中可以看到三次握手的过程。

3767	52.718287	127.0.0.1	127.0.0.1	TCP	56 60244 → 8848 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
3768	52.718340	127.0.0.1	127.0.0.1	TCP	56 8848 → 60244 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
3769	52.718377	127.0.0.1	127.0.0.1	TCP	44 60244 → 8848 [ACK] Seq=1 Ack=1 Win=327424 Len=0

图 6: 三次握手

在三次握手完成后，TCP 连接就被成功建立，之后客户端和服务器就可以开始双向数据传输。三次握手不仅确保了连接的可靠性，而且避免了过时的连接请求突然出现在网络中导致的潜在问题。这个三次握手的过程确保了两件事：

1. 双方都知道彼此有能力发送和接收数据（即都是活跃的）。
2. 双方都同步了彼此的初始序列号，这对于 TCP 的可靠数据传输机制来说非常重要。

- SYN (同步)
  - 客户端发送一个 TCP 数据包到服务器。在这个数据包的头部中，SYN 标志位被设置为 1，同时客户端会随机选择一个初始的序列号 J。
  - 数据包: Client -> Server [SYN, Seq=x]。
- SYN + ACK (同步 + 确认)

  - 服务器收到 SYN 数据包后,为了确认客户端的 SYN,会向客户端发送一个 SYN+ACK 数据包。这个数据包中，SYN 标志位和 ACK 标志位都被设置为 1。服务器也会选择一个自己的初始序列号 y 并且设置 ACK 的值为 x + 1 来确认客户端的序列号。
  - 数据包: Server -> Client [SYN, Seq=K, ACK, Ack=J+1]。

- ACK (确认)
  - 客户端收到服务器的 SYN+ACK 数据包后，会发送一个 ACK 数据包给服务器，确认服务器的 SYN。这个数据包的 ACK 值会被设置为 y + 1。
  - 数据包: Client -> Server [ACK, Ack=K+1]。

2. HTTP 请求报文

一个 HTTP 请求报文由请求行（request line)、请求头部（request header)、空行和请求数据 4 个部分构成。

- 请求行

  请求行数据格式由三个部分组成：请求方法、URI、HTTP 协议版本，他们之间用空格分隔。

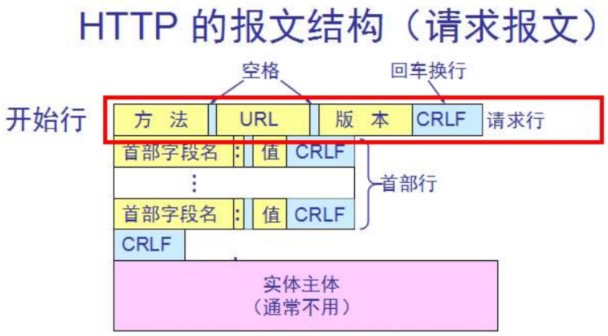

图 7: HTTP 请求报文

  - GET : HTTP 请求方法
  - URL ： 唯一资源定位符，描述了一个资源在网络上的位置
  - 版本号：HTTP 协议的版本 (最主流的是 1.1 版本)

- 请求头

  主要是用于说明请求源、连接类型、以及一些 Cookie 信息等。请求头部紧跟着请求行，该部分主要是用于描述请求正文。

  - 除第一行，剩下的数据中，直到遇见空行为止的数据，就是请求头，请求头是一个键值对结构的数据
  - 每个键值对都占一行
  - 键和值之间使用: + 空格隔开

- 请求正文

  一般用于存放 POST 请求类型的请求正文

- 空行

  请求头下面还有一个空行，表示请求头结束的标记

3. HTTP 响应报文

响应报文会将请求的结果返回给客户端。HTTP 响应报文由状态行（HTTP 版本、状态码(数字和原因短语)）、响应头部、空行和响应体 4 个部分构成。

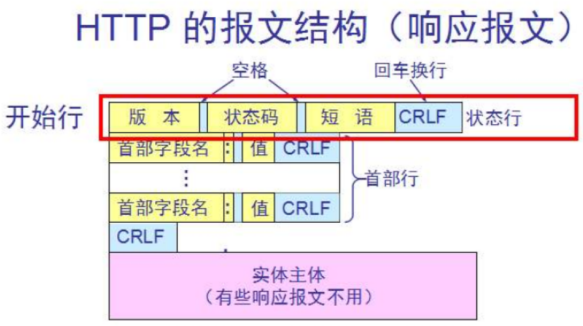

图 8: HTTP 响应报文

- 响应行

  响应的首行也是由三部分组成，但是含义和请求的首行有区别。

  – 版本号 (HTTP/1.1)：表示此响应使用的 HTTP 版本是 1.1。

  – 状态码 (200)：描述了请求的结果，是成功了还是失败了还是其他

  – 状态码解释 (ok)：对状态码使用文本又进一步的进行了解释

- 响应头

  – 除首行以外，下面的数据中，直到空行为止的这一部分的内容，就是响应头。

  – 响应头也都是键值对。

  – 键和值之间使用: + 空格隔开

- 空行

  响应头的结束标记。

- 正文

  空行下面的部分就是正文，也就是对请求的内容进行响应的数据，这个数据可能是多种格式的，可能是文本，图片，视频，音频等。

4. TCP 四次挥手关闭连接

TCP 使用四次挥手关闭连接，客户端和服务端分别释放连接。

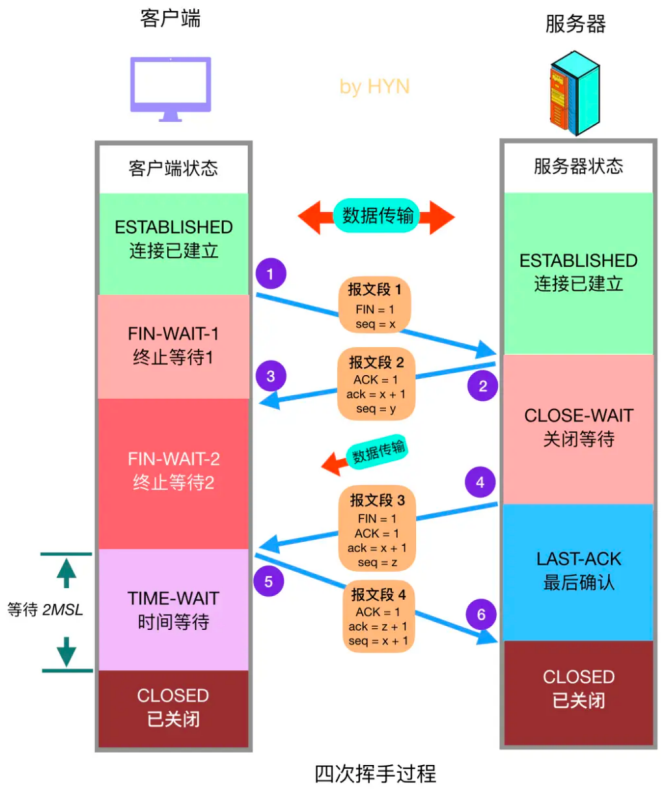

图 9

在 wireshark 中可以看到四次挥手的过程。

4383	58.108838	127.0.0.1	127.0.0.1	TCP	44 8848 → 60244 [FIN, ACK] Seq=715 Ack=2258 Win=2158848 Len=0
4384	58.108883	127.0.0.1	127.0.0.1	TCP	44 60244 → 8848 [ACK] Seq=2258 Ack=716 Win=326656 Len=0
4411	58.977121	127.0.0.1	127.0.0.1	TCP	44 60244 → 8848 [FIN, ACK] Seq=2258 Ack=716 Win=326656 Len=0
4412	58.977165	127.0.0.1	127.0.0.1	TCP	44 8848 → 60244 [ACK] Seq=716 Ack=2259 Win=2158848 Len=0

图 10: 四次挥手

主动方首先发送了一个 FIN, ACK 数据包，表示它已完成数据发送并希望关闭连接；被动方收到这个数据包后，发送了一个 ACK 数据包来确认；被动方随后也发送了一个 FIN, ACK 数据包，表示它也已完成数据发送并希望关闭连接；主动方收到这个数据包后，发送了一个 ACK 数据包来确认。

- FIN (结束)
  - 主动关闭方发送一个 TCP 数据包到对方。在这个数据包的头部中，FIN 标志位被设置为 1，表示主动方想要关闭连接。
  - 数据包: 主动方 → 被动方 [FIN, Seq = X]。
- ACK (确认)
  - 被动关闭方收到 FIN 数据包后，为了确认已收到主动方的终止请求，它会发送一个 ACK 数据包回去。
  - 数据包: 被动方 → 主动方 [ACK, Seq = Z, Ack = X + 1]。

- FIN (结束)
  - 被动关闭方在某个时刻后，也发送一个 FIN 数据包给主动方，表示它也准备好关闭这个连接了。
  - 数据包: 被动方 → 主动方 [FIN, Seq = Y]。
- ACK (确认)
  - 主动关闭方收到被动方的 FIN 数据包后，发送一个 ACK 数据包来确认。
  - 数据包: 主动方 → 被动方 [ACK, Seq = X + 1, Ack = Y + 1]。

五、 总结与思考

(一) 为何是三次握手而不是两次或四次握手?

TCP 的三次握手过程是为了确保双方都能可靠地建立连接，主要包括以下几点:

- **可靠性:** 第一次握手（SYN）由客户端发起，服务器回复（SYN-ACK）表示接收到了请求。第二次握手确认了服务器的存在和接收能力。第三次握手（ACK）由客户端发送，确保服务器已经准备好接收数据。这个过程确保双方都能互相确认，并且能处理丢包的情况。
- **状态同步:** 三次握手还允许双方在建立连接时同步各自的初始序列号，确保后续数据传输的顺序和可靠性。

如果是两次握手，可能会导致服务器没有确认客户端的状态，增加了连接不可靠的风险；而四次握手则会增加延迟，不必要地消耗资源。

(二) 为何是四次挥手而不是两次或三次挥手?

TCP 的四次挥手过程用于安全可靠地断开连接，主要包括以下几个步骤:

1. **主动关闭:** 客户端发送 FIN 包，请求断开连接。
2. **被动确认:** 服务器收到 FIN 包后，发送 ACK 确认，表示可以关闭连接。
3. **被动关闭:** 服务器发送 FIN 包，告知客户端它也准备关闭连接。
4. **最终确认:** 客户端收到 FIN 包后，发送 ACK 确认，完成断开过程。

四次挥手的原因包括:

- **数据传输的完整性:** 在 FIN 被发送后，主动关闭的一方仍然可以接收服务器发送的数据，因此需要确认双方的数据传输已经完成。
- **资源释放:** 在最后的 ACK 确认后，双方才能完全释放资源，这样可以避免数据丢失或连接状态不一致。

如果是两次挥手，可能会导致数据丢失的风险；三次挥手虽然可以达到部分确认，但仍然无法确保双方的数据状态完全一致。因此，四次挥手是更为安全和可靠的选择。