南開大學
Nankai University

计算机学院

计算机网络实验报告

配置 Web 服务器，编写简单页面，分析交互过程

姓名：张铭徐

学号：2113615

专业：计算机科学与技术

2023 年 10 月 30 日

目录

1 实验内容介绍

1.1 实验要求

本次实现我们拟实现搭建一个 Web 服务器，制作一个简单的网页，分析我们的客户端与服务器的交互过程，具体而言，要求如下：

- 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。
- 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。
- 使用 HTTP，不要使用 HTTPS。

1.2 实验背景知识介绍

在要求中，我们需要使用 HTTP 协议，而不可以使用 HTTPS 协议，在考虑原因之前，我们不妨先来考虑 HTTP 协议和 HTTPS 协议之间的区别：

1. **安全性**：
   - **HTTP**：HTTP 代表超文本传输协议，它是一个非加密的协议，这意味着传输的数据不是安全的，容易被中间人攻击者截取，在我们本次实验中，我们就需要对数据包进行捕获抓取。
   - **HTTPS**：HTTPS 代表安全超文本传输协议。它是一个加密的协议，使用 SSL（安全套接层）或 TLS（传输层安全）对数据进行加密，从而保护用户的数据免受中间人攻击者的侵害。
2. **端口号**：
   - **HTTP**：默认使用端口 80。
   - **HTTPS**：默认使用端口 443。
3. **证书**：
   - **HTTP**：不需要任何证书。
   - **HTTPS**：需要 SSL 证书。网站管理员必须从认证机构处购买此证书。
4. **URL 显示**：
   - **HTTP**：URL 以 “http://” 开头。
   - **HTTPS**：URL 以 “https://” 开头。
5. **性能**：
   - **HTTP**：因为没有加密和解密的过程，所以 HTTP 的性能比 HTTPS 稍好。
   - **HTTPS**：因为需要加密和解密数据，所以 HTTPS 的性能略低于 HTTP。但随着现代硬件和软件优化，这种性能差异在实际应用中往往可以忽略不计。

那么我们从上面两者的区别的讨论中可以看出，事实上，HTTP 和 HTTPS 最本质的区别就是在于是否对传输数据进行加密，我们本次实验不希望对数据进行加密，因为一旦加密我们就很难对数据包进行捕获了，就算可以捕获也需要一些时间解密，相对而言较为繁琐，所以本次实验我们采取端口号为 80 的 HTTP 协议进行网站的搭建。

接下来我们考虑我们需要抓取什么内容，事实上，本次实验的目的是分析客户端与服务器的交互过程，由于我们是在访问一个网页，在分析交互过程之前，我们首先回顾一下 TCP/IP 体系结构的层次结构。TCP/IP 体系结构，也称为 Internet 协议套件，是一个层次化的网络通信模型。其通常被描述为包含四层，每一层都有其特定的职责和功能。

- 应用层：作用：这一层为用户和进程提供了网络服务，并定义了与其他应用程序的通信规则。功能：提供端到端的用户接口。定义了与其他应用程序的通信规则。常见协议和应用：HTTP, FTP, SMTP, DNS, DHCP 等。
- 传输层：作用：确保端到端的通信可靠性和效率。功能：流量控制：确保数据的快速和可靠传输。错误检测和纠正。数据分段与重组。常见协议：TCP (提供可靠传输) 和 UDP (提供无连接的快速传输)。
- 网络层：作用：负责数据包的发送和路由，包括它们从源到目的地的整个路径。功能：数据包路由：选择数据包从源到目标的最佳路径。IP 地址分配和处理。数据包分片与重组。常见协议：IP, ICMP, ARP 。
- 链路层：作用：负责与网络的物理硬件和媒介交互。功能：帧的创建、传输和接收。MAC 地址处理：这是物理地址，用于标识网络上的设备。错误检测。常见技术和协议：Ethernet, Wi-Fi, PPP 等。

那么事实上，我们客户端与服务器的交互，或者具体到本次实验中，我们在访问一个 Web 页面的时候，事实上是在以本机作为客户端，与网页的服务器端进行数据传输的过程，那么既然要数据传输，我们就需要先建立连接，在 TCP/IP 体系结构中，在建立连接时，是通过 TCP 三次握手实现的，在说三次握手之前，我们先来看一下 TCP 首部格式：

图 1.1: TCP 首部格式

在这部分中有一些状态位是我们后面需要用到的，我们在这里对其进行一些说明：

- 确认标志位 ACK：取值为 1 时确认号字段才有效；取值为 0 时确认号字段无效。TCP 规定，在连接建立后所有传送的 TCP 报文段都必须把 ACK 置 1。
- 同步标志位 SYN：在 TCP 连接建立时用来同步序号。
- 复位标志位 RST：用来复位 TCP 连接。
- 结束标志位 FIN：标志着发送端已经完成了数据发送，并希望关闭连接。

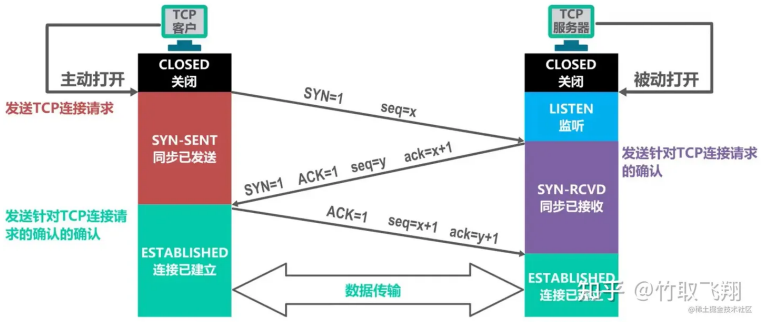


图 1.2: TCP 三次握手

接下来我们就可以考虑如何建立 TCP 的连接：TCP 建立连接的过程叫做握手，握手需要在客户和服务器之间交换三个 TCP 报文段，称之为三报文握手，采用三报文握手主要是为了防止已失效的连接请求报文段突然又传送到了，因而产生错误。

实际上，TCP 连接的过程需要回答并理解以下三个问题：

- 使 TCP 双方能够确知对方的存在。
- 使 TCP 双方能够协商一些参数（最大窗口值是否使用窗口扩大选项和时间戳选项，以及服务质量等）。
- 使 TCP 双方能够对运输实体资源（例如缓存大小连接表中的项目等）进行分配。

三次握手所传递的报文，实际上就是在解决上述三个问题的过程，下图展示了 TCP 三次握手的原理：如图1.2所示：

我们考虑：在第一次握手时，客户端向服务器端发送一个包，里面包含 SYN=1，意为我们想要建立连接；然后服务器收到了客户端的包，发送一个 SYN=1 和 ACK=1，意为服务器端同意与客户端建立连接，然后在客户端收到该报文后，又发送了一个数据包给服务器端，包含一个 ACK=1，这就意味着客户端确认收到服务器端同意连接的信号。

在经过三次握手后，客户端和主机端分别感知到了对方的存在，并且同意建立了连接，那么接下来就可以进入到数据传输阶段了。在数据传输阶段过后，我们想要关闭客户端和主机端的连接，就需要 TCP 四次挥手：四次挥手如下图1.3所示：

我们对四次挥手的过程做以说明：首先客户端想要对服务器端断开连接，那么就需要先告诉服务器端，我要结束连接，在报文层面体现的就是会给服务器端发送一个 FIN=1 的报文，然后服务器端收到了请求关闭连接的请求后，发送一个 ACK，意味着我已经收到你的请求了，然后服务器端给客户端发送一个 FIN，意为我现在要关闭连接了，是否收到这条消息？然后客户端收到消息后，给服务端发送一个 ACK 意为我已经准备好了！随时可以关闭。

事实上，我们称之为四次挥手，但是实际上，我们可以将服务器的 ACK 和 FIN 报文放在同一个数据包中进行传输，以便于减少传输延迟，为什么要分开描述这两个步骤呢？主要是为了概念上的清晰。描述四次挥手可以更清楚地展示连接终止的双向性：客户端希望终止它到服务器的连接，而服务器也希望终止它到客户端的连接。

那么本次实验中，我们将会详细的对建立连接，终止连接，数据包传输这三个过程进行详细的捕获分析，以全面的验证客户端与服务器端进行数据交互的完整过程。

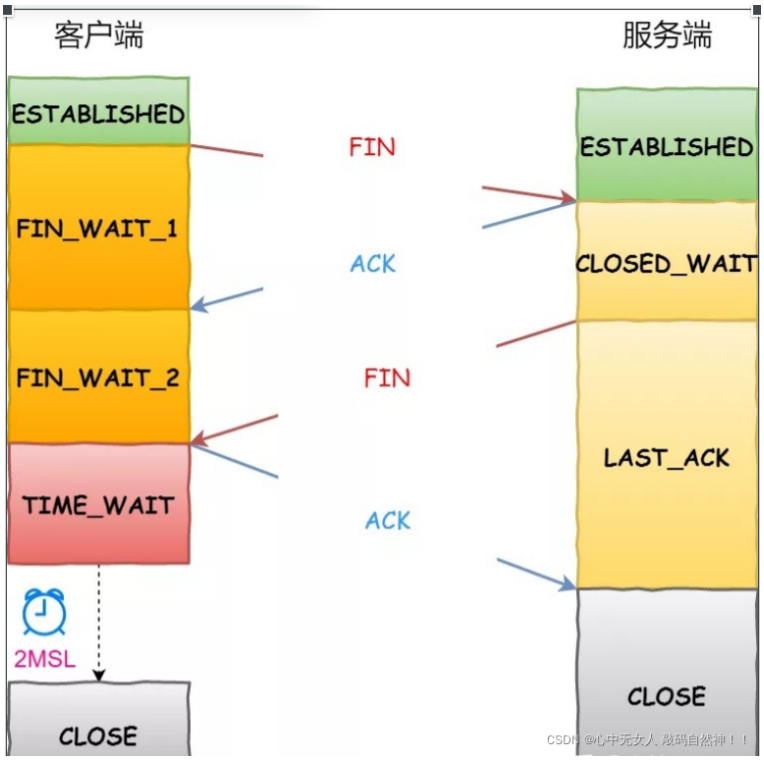


图 1.3: TCP 四次挥手

1.3 实验平台介绍

本次实验我们在 Windows 系统下进行实验，我们拟采用 PHP_Study_Pro 进行网页的搭建，我们下载软件后，启动 Apache 服务后，将网页的源码放入其 WWW 目录下，就可以使用在浏览器中访问 Localhost:80 对服务器进行访问；抓包过程我们使用 Wireshark 进行实验。

2 TCP 握手阶段捕获分析

首先我们需要注意的是，我们是在本机端新建了一个服务器，那么我们实际上访问 localhost:80 就不是在进行互联网上的数据交互，而流量通常在所谓的“回环接口”（loopback interface）上流转，所以我们在捕获数据包的时候需要选择 loopback 进行捕获。

我们打开 wireshark 软件，开始捕获数据包，并在浏览器中访问 localhost:80，然后终止捕获数据包，如图2.4所示；而由于我们目前捕获的数据包是所有的流量包，而我们只想要捕获与固定端口进行交互的数据，所以我们可以在过滤器中输入：tcp.port==80，这样我们就找到了固定端口的数据包。我们可以看到如图2.5所示的三次握手的数据包：

- 客户端：58673 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=65475 WS=256 SACK_PERM: 客户端发起连接请求，发送带有 SYN 标志的包到服务器的 80 端口。序列号为 0，窗口大小为 65535，最大段大小（MSS）为 65475，窗口缩放因子（WS）为 256，支持选择性确认（SACK_PERM）。
- 服务器：80 → 58673 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65475 WS=256 SACK_PERM: 服务器接收到客户端的 SYN 包，并响应带有 SYN 和 ACK 标志的包。序列号为 0，确认号为 1（表示期望接收客户端的下一个数据包的序列号为 1），窗口大小、MSS、WS 和 SACK_PERM 的值与客户端发送的值相同。

- 客户端：58673 → 80 [ACK] Seq=1 Ack=1 Win=327168 Len=0：客户端收到服务器的 SYN, ACK 包，并发送了一个 ACK 包作为确认。序列号为 1，确认号也为 1，窗口大小为 327168。

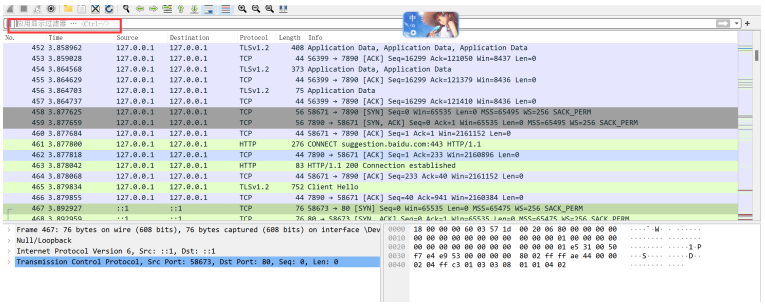


图 2.4: 数据包

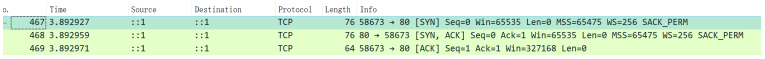


图 2.5: TCP 三次握手

3 TCP 挥手阶段捕获分析

我们继续对捕获的数据包分析，我们可以找到 TCP 四次挥手的数据包：如图3.6所示，我们对这部分进行说明分析：

- 服务器 80 → 58673 [FIN, ACK] Seq=233 Ack=813 Win=2159616 Len=0：服务器想要终止连接，所以它发送了一个带有 FIN 标志的包。同时，该包还带有 ACK 标志，确认了之前从客户端收到的数据。
- 客户端 58673 → 80 [ACK] Seq=813 Ack=234 Win=326912 Len=0：客户端收到服务器的 FIN 包，并发送了一个 ACK 包作为确认。这是对服务器的 FIN 的响应。
- 客户端 58673 → 80 [FIN, ACK] Seq=813 Ack=234 Win=2160640 Len=0：现在，客户端也想要终止连接，所以它发送了一个带有 FIN 标志的包。这不是对之前服务器的 FIN 的回应，而是客户端自己发起的连接终止请求。此包同样也带有 ACK 标志，但这不是新的确认，而是沿用之前的确认。
- 服务器 80 → 58673 [ACK] Seq=234 Ack=814 Win=2159616 Len=0：现在，客户端也想要终止连接，所以它发送了一个带有 FIN 标志的包。这不是对之前服务器的 FIN 的回应，而是客户端自己发起的连接终止请求。此包同样也带有 ACK 标志，但这不是新的确认，而是沿用之前的确认。

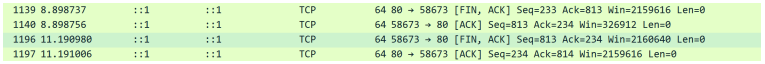


图 3.6: TCP 四次挥手捕获数据包

这部分就是一个很典型的四次挥手，但是值得注意的是，我们需要避免一个误区：事实上客户端和服务器端都可以主动终止连接，在我们这里，就是经典的服务器端在完成数据传输后主动关闭连接的实例。

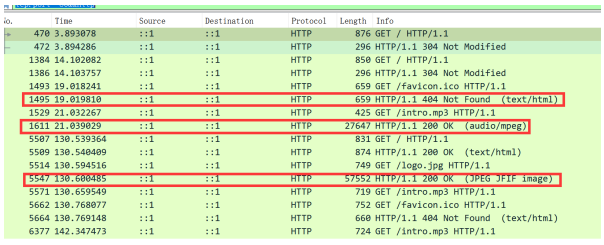
4 HTTP 协议数据传输过程捕获分析

我们使用的 http 如下：我们可以看到，包括了三种类型的数据，分别是文本信息，图片信息，音频信息，我们可以通过捕获数据包对这三种类型的数据分别抓取。

网站源码

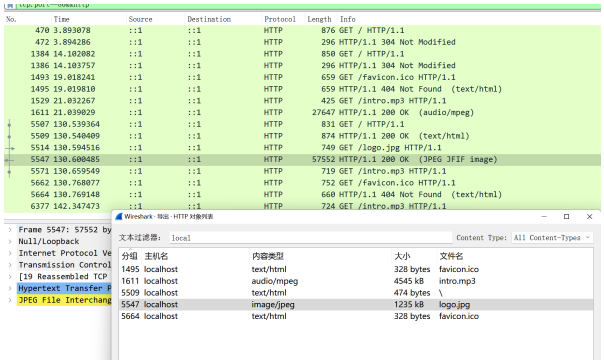
```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>我的简介</title>
</head>
<body>
    <h1>专业：计算机科学与技术</h1>
    <p>学号：2113615</p>
    <p>姓名：张铭徐</p>
    <img src="logo.jpg" alt="我的LOGO" width="100">
    <audio controls>
        <source src="intro.mp3" type="audio/mp3">
        您的浏览器不支持音频播放。
    </audio>
</body>
</html>
```

我们在 Wireshark 的过滤器中输入 tcp.port==80&&http，这部分意为我们要捕获 http 协议，并且端口号为 80 的数据包，结果如下所示：



No.	Time	Source	Destination	Protocol	Length	Info
470	3.893078	::1	::1	HTTP	876	GET / HTTP/1.1
472	3.894286	::1	::1	HTTP	296	HTTP/1.1 304 Not Modified
1384	14.102082	::1	::1	HTTP	850	GET / HTTP/1.1
1386	14.103757	::1	::1	HTTP	296	HTTP/1.1 304 Not Modified
1493	19.018241	::1	::1	HTTP	659	GET /favicon.ico HTTP/1.1
1495	19.019810	::1	::1	HTTP	659	HTTP/1.1 404 Not Found (text/html)
1529	21.032267	::1	::1	HTTP	425	GET /intro.mp3 HTTP/1.1
1611	21.039029	::1	::1	HTTP	27647	HTTP/1.1 200 OK (audio/mpeg)
5507	130.539364	::1	::1	HTTP	831	GET / HTTP/1.1
5509	130.540409	::1	::1	HTTP	874	HTTP/1.1 200 OK (text/html)
5514	130.594516	::1	::1	HTTP	749	GET /logo.jpg HTTP/1.1
5547	130.600485	::1	::1	HTTP	57552	HTTP/1.1 200 OK (JPEG JFIF image)
5571	130.659549	::1	::1	HTTP	719	GET /intro.mp3 HTTP/1.1
5662	130.768077	::1	::1	HTTP	752	GET /favicon.ico HTTP/1.1
5664	130.769148	::1	::1	HTTP	660	HTTP/1.1 404 Not Found (text/html)
6377	142.347473	::1	::1	HTTP	724	GET /intro.mp3 HTTP/1.1

图 4.7: 所有 http 协议数据包



No.	Time	Source	Destination	Protocol	Length	Info
470	3.893078	::1	::1	HTTP	876	GET / HTTP/1.1
472	3.894286	::1	::1	HTTP	296	HTTP/1.1 304 Not Modified
1384	14.102082	::1	::1	HTTP	850	GET / HTTP/1.1
1386	14.103757	::1	::1	HTTP	296	HTTP/1.1 304 Not Modified
1493	19.018241	::1	::1	HTTP	659	GET /favicon.ico HTTP/1.1
1495	19.019810	::1	::1	HTTP	659	HTTP/1.1 404 Not Found (text/html)
1529	21.032267	::1	::1	HTTP	425	GET /intro.mp3 HTTP/1.1
1611	21.039029	::1	::1	HTTP	27647	HTTP/1.1 200 OK (audio/mpeg)
5507	130.539364	::1	::1	HTTP	831	GET / HTTP/1.1
5509	130.540409	::1	::1	HTTP	874	HTTP/1.1 200 OK (text/html)
5514	130.594516	::1	::1	HTTP	749	GET /logo.jpg HTTP/1.1
5547	130.600485	::1	::1	HTTP	57552	HTTP/1.1 200 OK (JPEG JFIF image)
5571	130.659549	::1	::1	HTTP	719	GET /intro.mp3 HTTP/1.1
5662	130.768077	::1	::1	HTTP	752	GET /favicon.ico HTTP/1.1
5664	130.769148	::1	::1	HTTP	660	HTTP/1.1 404 Not Found (text/html)
6377	142.347473	::1	::1	HTTP	724	GET /intro.mp3 HTTP/1.1

Wireshark · 导出 · HTTP 对象列表

分组	主机名	内容类型	大小	文件名
1495	localhost	text/html	328 bytes	favicon.ico
1611	localhost	audio/mpeg	4545 kB	intro.mp3
5509	localhost	text/html	474 bytes	\
5547	localhost	image/jpeg	1235 kB	logo.jpg
5664	localhost	text/html	328 bytes	favicon.ico

图 4.8: 导出对象

我们可以看到：在 http 协议中，我们捕获到了文本信息，图像信息，音频文件这三个数据包，我们同样可以进一步的导出这部分的数据，在导出对象中选中这些数据，即可导出，如下图4.8所示：我们就可以对捕获到了各个文件进行导出，我们可以选中对象，然后在 save 中保存，然后就可以在本地打开捕获到的数据包了。

5 实验总结

本次实验旨在探究客户端和服务器端的数据传输过程，我们通过实际的网络抓包工具 Wireshark 对本地 localhost 的 80 端口进行观察，分析其数据包的传输过程，从而深入了解 TCP 连接和终止的详细过程。

我们捕获了 TCP 三次握手的连接建立，四次挥手的连接终止，以及中间的数据传输过程的详细数据包，值得一提的是，我们对于数据传输过程而言，捕获了网站中包含的文本，图片，音频三种数据，并可以保存在本地进行查看。总体来看，本次实验加深了我们对于 TCP/IP 体系结构的理解，并通过实际操作对 HTTP 协议的不安全性有了更深的理解，总体来看，本次实验圆满成功!