南开大学

Nankai University

南　　开　　大　　学

计　算　机　学　院

计算机网络实验报告

实验 3：配置 Web 服务器 捕获 HTTP 报文并分析

贾景顺

年级：2022 级

专业：计算机科学与技术

指导教师：张建忠　徐敬东

2026 年 1 月 3 日

摘要

本实验旨在通过搭建 Web 服务器与抓包分析，深入理解应用层 HTTP 协议的工作机制及其在 TCP/IP 协议栈中的封装过程。实验首先在 Windows 11 环境下利用 Internet Information Services (IIS) 构建了 Web 服务器，并设计发布了一个包含个人信息、六幅展示图像、背景音乐及动态交互特效的 HTML5 网页。随后，使用 Wireshark 网络协议分析工具捕获了 Edge 浏览器与 Web 服务器之间的完整交互数据报文。通过对核心 HTTP 请求与响应报文的详细解码，逐层分析了从数据链路层、互连层、传输层到应用层的封装结构，并结合网页加载过程还原了包含主文档解析及多媒体资源并发请求在内的完整 HTTP 交互流程。实验结果清晰地展示了 Web 资源的请求响应模式，验证了 TCP/IP 协议栈的分层协作原理，加深了对计算机网络体系结构的直观认识。

摘要： TTP 协议；IIS 服务器；Wireshark；报文封装；TCP/IP 协议栈实验总结

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，包含简单文本信息（包含专业、学号、姓名）和 6 幅图像。

(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) 分析两个报文的封装层次，包括数据链路层、互连层、传输层、应用层。

(4) 对整个 HTTP 交互过程进行详细说明，使用 Wireshark 过滤器使其只显示 HTTP 协议。

(5) 提交 HTML 文档、Wireshark 捕获文件和实验报告

注：页面不要太复杂，包含所要求的基本信息即可。使用 HTTP，不使用 HTTPS。

二、 实验环境

本次实验在 x86-64 架构物理机中进行。

使用 Internet Information Services（IIS）进行服务器搭建。

使用 Wireshark 工具，捕获交互过程。

使用 Google 浏览器连接服务器。

相关系统及应用程序版本如下：

表 1: 实验环境配置

系统或程序	版本
Windows 及 IIS	Windows 11, 24H2
Wireshark	4.6.2
Edge 浏览器	版本 143.0.3650.96

三、 WEB 服务器搭建

在系统设置中启用 Internet Information Services，并勾选 Web 管理工具以及静态内容。

打开 IIS。创建网站 NetWorkLab，设置物理路径为项目根路径，并完成 IP 地址和端口的设定：

图 1: Web 服务器设置

四、 页面设计

4.1 内容介绍

本网页设计包含以下五个核心部分，通过 HTML、CSS 和 JavaScript 共同实现：

1. **欢迎封面**
   - **功能**：网页加载时提示“点击屏幕任意处开始”。目的是引导用户点击，从而避开浏览器禁止自动播放音频的限制。
   - **实现**：使用 `<div>` 标签覆盖全屏，设置层级 `z-index` 为最高。通过 JavaScript 的 `onclick` 事件监听点击，点击后隐藏封面并触发音乐播放。
2. **背景音乐播放**
   - **功能**：进入主页后自动播放 MP3 欢迎播报，播放完毕后自动停止（不循环）。
   - **实现**：使用 HTML5 的 `<audio>` 标签加载 `bgm.mp3` 文件。在封面的点击事件中调用 `.play()` 函数启动播放。
3. **头像与个人信息**
   - **功能**：展示专业、学号、姓名等基本信息，左上角悬挂一张圆形头像。
   - **实现**：使用 CSS 的 `position: absolute` 将头像定位到容器左上角外部，利用 `border-radius: 50%` 实现圆形裁剪，并添加 `hover` 伪类实现动画。
4. **图片展示**
   - **功能**：整齐展示 6 张图片，鼠标指向图片时会有上浮和发光效果。

- **实现**：使用 Flex 弹性布局 (`display: flex`) 让图片自动换行居中。使用 CSS 的 `object-fit: cover` 保证图片大小统一且不变形。

5. **星星下落特效**

- **功能**：屏幕背景中有星星不断随机生成并缓缓下落。
- **实现**：使用 JavaScript 的 `setInterval` 定时器每隔 150 毫秒生成一个星星元素，随机分配颜色和位置。配合 CSS 的 `@keyframes` 关键帧定义下落动画。

4.2 源码展示

index.html

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <title>欢迎来到狗修金的计网作业网站</title>
    <style>
        body {
            font-family: 'Microsoft YaHei', 'Segoe UI', sans-serif;
            background: linear-gradient(135deg, #fdfbfb 0%, #ebedee 100%);
            margin: 0;
            padding: 0;
            color: #333;
            overflow-x: hidden;
            min-height: 100vh;
            position: relative;
        }

        #welcome-screen {
            position: fixed;
            top: 0;
            left: 0;
            width: 100%;
            height: 100%;
            background: rgba(0, 0, 0, 0.85);
            color: white;
            z-index: 9999;
            display: flex;
            flex-direction: column;
            justify-content: center;
```

```
30            align-items: center;
31            cursor: pointer;
32            transition: opacity 0.8s;
33        }
34
35        #welcome-screen h1 {
36            color: white;
37            border: none;
38            font-size: 40px;
39            text-shadow: 0 0 20px #ff69b4;
40            animation: pulse 1.5s infinite;
41        }
42
43        @keyframes pulse {
44            0% { transform: scale(1); }
45            50% { transform: scale(1.1); }
46            100% { transform: scale(1); }
47        }
48
49        .container {
50            position: relative;
51            max-width: 800px;
52            margin: 80px auto 40px auto;
53            background-color: rgba(255, 255, 255, 0.9);
54            padding: 40px;
55            border-radius: 20px;
56            box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1);
57            z-index: 10;
58            opacity: 0;
59            transition: opacity 1s;
60        }
61
62        .my-avatar {
63            position: absolute;
64            top: -50px;
65            left: -50px;
66            width: 120px;
67            height: 120px;
68            border-radius: 50%;
69            border: 6px solid white;
70            box-shadow: 0 5px 15px rgba(0,0,0,0.2);
71            transition: transform 0.5s cubic-bezier(0.175,
                  0.885, 0.32, 1.275);
72            z-index: 20;
73            cursor: pointer;
74        }
75
76        .my-avatar:hover {
77            transform: rotate(360deg) scale(1.1);
78        }
79
80        .info-card {
81            background-color: #fff0f5;
```

```
            border-left: 5px solid #ff69b4;
            padding: 15px 25px;
            margin: 20px 0;
            border-radius: 8px;
        }

        h1 {
            text-align: center;
            color: #555;
            border-bottom: 2px dashed #ffb6c1;
            padding-bottom: 10px;
            margin-top: 30px;
        }

        .image-gallery {
            display: flex;
            flex-wrap: wrap;
            justify-content: center;
            gap: 20px;
            margin-top: 30px;
        }

        .image-gallery img {
            width: 220px;
            height: 160px;
            object-fit: cover;
            border-radius: 12px;
            box-shadow: 0 4px 10px rgba(0,0,0,0.1);
            transition: all 0.3s ease;
            cursor: pointer;
        }

        .image-gallery img:hover {
            transform: translateY(-10px) scale(1.05);
            box-shadow: 0 15px 30px rgba(255, 182, 193, 0.6);
            z-index: 15;
        }

        .star {
            position: fixed;
            top: -10px;
            user-select: none;
            pointer-events: none;
            z-index: 5;
            animation: fall linear infinite;
        }

        @keyframes fall {
            0% { transform: translateY(-10vh) rotate(0deg);
                 opacity: 1; }
            100% { transform: translateY(110vh) rotate(360deg);
                   opacity: 0; }
        }
```

```
133
134         .footer {
135             text-align: center;
136             margin-top: 40px;
137             color: #999;
138             font-size: 12px;
139         }
140     </style>
141 </head>
142 <body>
143     <div id="welcome-screen" onclick="enterSite()">
144         <h1>  点击进入丛雨の私密空间  </h1>
145         <p>Click anywhere to enter</p>
146     </div>
147 
148     <audio id="bgm">
149         <source src="ciallo.mp3" type="audio/mpeg">
150     </audio>
151     <div class="container" id="main-content">
152         <img src="logo.jpg" alt="头像" class="my-avatar">
153 
154         <h1>欢迎来到狗修金的计网作业网站</h1>
155         <div class="info-card">
156             <p><strong>专业: </strong>计算机科学与技术</p>
157             <p><strong>学号: </strong>2211312minor</p>
158             <p><strong>姓名: </strong>贾景顺</p>
159         </div>
160 
161         <h1>Ciallo～( ・  < ) </h1>
162 
163         <div class="image-gallery">
164             <img src="1.jpg">
165             <img src="2.jpg">
166             <img src="3.jpg">
167             <img src="4.jpg">
168             <img src="5.jpg">
169             <img src="6.jpg">
170         </div>
171 
172         <div class="footer">
173             Computer Network Experiment © 2026
174         </div>
175     </div>
176 
177     <script>
178 
179         function enterSite() {
180             var welcome = document.getElementById('welcome-
                screen');
181             var mainContent = document.getElementById('main-
                content');
182             var music = document.getElementById('bgm');
```

```
            music.play().catch(function(error) {
                console.log("音乐播放失败，可能文件路径不对:",
                    error);
            });

            welcome.style.opacity = '0';
            setTimeout(function(){
                welcome.style.display = 'none';
            }, 800);

            mainContent.style.opacity = '1';

            setInterval(createStar, 150);
        }

        function createStar() {
            const star = document.createElement('div');
            star.classList.add('star');

            const shapes = ['', '', '', ''];
            star.innerHTML = shapes[Math.floor(Math.random() *
                shapes.length)];

            const colors = ['#FF69B4', '#87CEFA', '#FFA500',
                '#9370DB', '#32CD32'];
            star.style.color = colors[Math.floor(Math.random()
                * colors.length)];

            star.style.left = Math.random() * 100 + 'vw';
            const size = Math.random() * 200 + 10;
            star.style.fontSize = size + 'px';
            const duration = Math.random() * 3 + 2;
            star.style.animationDuration = duration + 's';

            document.body.appendChild(star);
            setTimeout(() => {
                star.remove();
            }, duration * 1000);
        }
    </script>
</body>
</html>
    }
```

4.3 界面展示

最终实现效果如下:

图 2: 界面展示

五、 Wireshark 抓包

5.1 抓包流程

抓包使用 Wireshark 程序实现，在启用前使用 Edge 的无痕隐私模式防止浏览 Cookie 对实验存在影响。

打开 Wireshark，使用 Adapter for loopback traffic capture 抓包（用于捕获本机自己的网络）。

在浏览器中输入“http://192.168.2.6/index.html”进入网站，音频播放完成后，退出网站，并关闭捕获查看结果，保存为 capture.pcapng。随后利用过滤器进行“HTTP”及“tcp.port == 80 || udp.port == 80”分析具体流程

5.2 报文的封装结构

5.2.1 请求报文

选取 GET /index.html HTTP/1.1 该报文作为典型的请求报文进行分析，具体信息如下图所示：

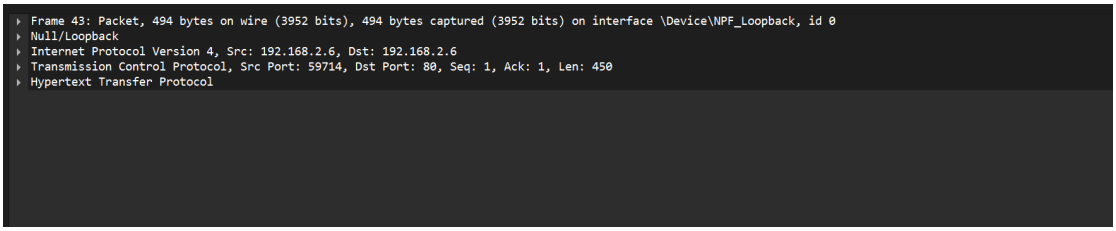

图 3: 请求报文

该报文的主要结构如下：

1. 数据链路层：这是报文的最底层，但因为使用的是 Loopback（环回）接口，数据没有流经真实的物理网卡，因此这里没有源 MAC 和目的 MAC 地址，而是使用了 Null/Loopback 协议头。它定义了数据在链路上的传输格式。在这个特定环境下，它标识这是一个用于本机内部通信的环回帧。

2. 互连层：从 Internet Protocol Version 4, Src: 192.168.2.6, Dst: 192.168.2.6 可以看出，使用 IPv4 协议，源 IP、目的 IP 均为 192.168.2.6。负责逻辑寻址和路由。这一层头部包裹了上层数据，确立了通信的起点和终点都是本机。

3. 传输层：使用 TCP 协议。源端口 59714 是浏览器随机分配的一个临时端口（Ephemeral Port），用于接收服务器回传的数据，目的端口为前面设置的 80。Seq: 1, Ack: 1 表明这是在 TCP 三次握手建立连接之后，发送的第一个携带真实数据的报文。该层的封装提供端到端的可靠传输服务，通过端口号区分具体的应用程序。

4. 应用层：使用 HTTP 协议，是最顶层的数据，包含了具体的业务逻辑，请求服务器发送 index.html 这个文件。

5.2.2 响应报文

选取 HTTP/1.1 200 OK (text/html) 该报文作为典型的请求报文进行分析，具体信息如下图所示：

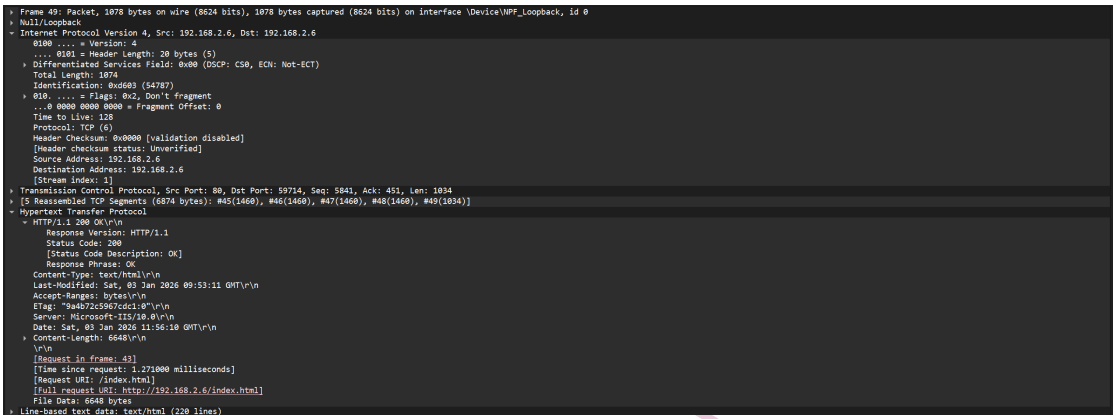

图 4: 响应报文

该报文的主要结构如下：

1. 数据链路层：这是报文的最底层，但因为使用的是 Loopback（环回）接口，数据没有流经真实的物理网卡，因此这里没有源 MAC 和目的 MAC 地址，而是使用了 Null/Loopback 协议头。捕获长度为 1078 字节，表明这是一个携带了大量数据的实际负载帧，它负责将上层的 IP 数据报在本机的网络协议栈内部进行逻辑上的点对点传递。

2. 互连层：从 Internet Protocol Version 4, Src: 192.168.2.6, Dst: 192.168.2.6 可以看出，使用 IPv4 协议，源 IP、目的 IP 均为 192.168.2.6。负责逻辑寻址和路由。IP 头部中的 Protocol 字段值为 6，标识其承载的上层协议为 TCP。此外，Time to Live (TTL) 值为 128，这是 Windows 系统默认的 TTL 值，侧面验证了服务器运行在 Windows 环境下。

3. 传输层：使用 TCP 协议。源端口（Src Port）为 80，这是标准的 HTTP 服务端口，表明数据来自 IIS 服务器；目的端口（Dst Port）为 59714，与之前请求报文（Frame 43）的源端口一致，确保数据能准确送达发起请求

的浏览器进程。值得注意的是，该层显示了“Reassembled TCP Segments”信息，说明由于 HTML 文件较大（6648 字节），超过了单个数据包的限制，因此被 TCP 协议切分成了多个段（从 Frame 45 到 Frame 49）进行传输，而当前分析的 Frame 49 是这一系列分段重组后的最终携带者，总负载长度达到了 6874 字节。

4. 应用层：应用层展示了 HTTP 协议的响应详情，状态行“HTTP/1.1 200 OK”明确表示服务器已成功处理请求并返回了资源。响应头中包含了关键元数据：“Server: Microsoft-IIS/10.0”表明了服务器软件的版本；“Content-Type: text/html”告知浏览器返回的内容是 HTML 网页文档；“Content-Length: 6648”指明了实体数据的具体大小。报文的最末端包含了“Line-based text data”，这即是实际的 HTML 源代码（即您编写的 index.html），浏览器接收到这部分数据后，便开始解析并渲染出带有文字、头像和背景音乐的网页界面。

5.3 HTTP 交互过程

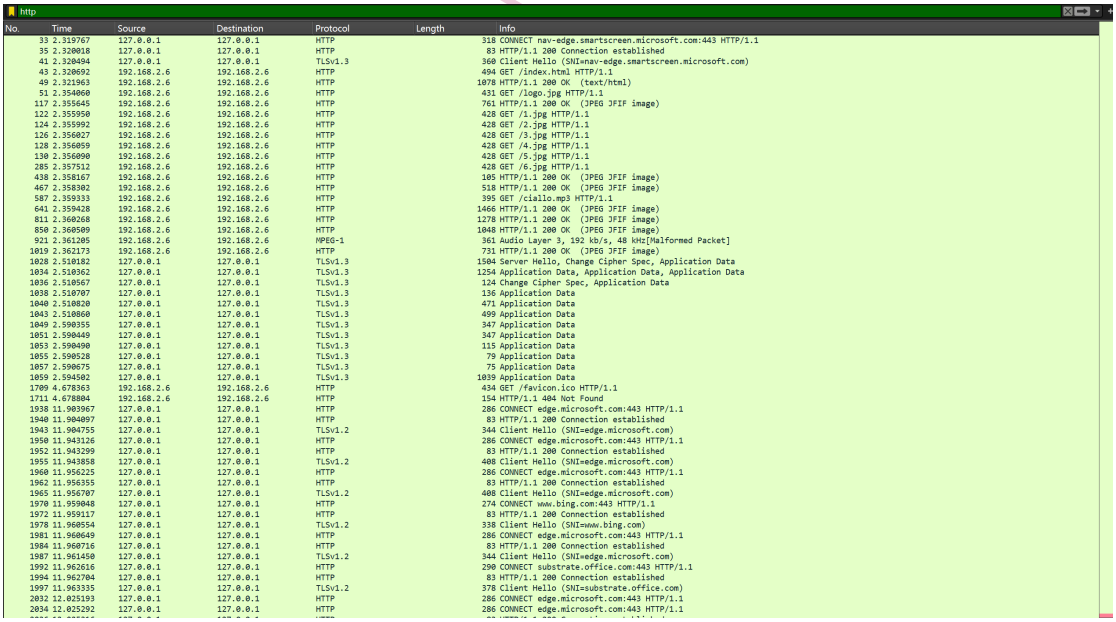

图 5: Wireshark 捕获结果

使用 Wireshark 捕获并使用 HTTP 筛选后的结果如上图所示整个 HTTP 交互过程始于客户端浏览器向服务器发起的连接请求。在 TCP 三次握手成功建立连接后，浏览器作为客户端向服务器（192.168.2.6）的 80 端口发送了第一个 HTTP 请求报文，方法为 GET，请求的资源路径为 /index.html。服务器接收到请求后进行处理，返回了一个状态码为 200 OK 的响应报文，该报文中包含了 Content-Type 为 text/html 的头部信息以及网页的 HTML 源代码主体。

浏览器接收到主文档（HTML）后立即开始进行解析（Parsing）。在解析过程中，浏览器发现文档中引用了多个外部资源，因此自动发起了新一轮的并发请求。具体而言，浏览器识别到了 <img> 标签引用的头像文件 avatar.jpg 和 6 幅展示图片（1.jpg 至 6.jpg），以及通过 JavaScript 和 <audio> 标签触发的背景音乐文件 bgm.mp3。为了提高页面加载速度，浏览器复用了已建立的 TCP 连接（基于 HTTP/1.1 的 Keep-Alive 特性）或建立了新的并行连接，连续向服务器发送了针对这些静态资源的 GET 请求。

服务器对这些后续请求逐一进行了响应。对于图片资源请求，服务器返回了 Content-Type 为 image/jpeg 的 200 OK 响应，报文主体中携带了图片的二进制数据；对于音频资源请求，服务器返回了 Content-Type 为 audio/mpeg 的响应，确保音频流能够被浏览器解码播放。在此过程中，Wireshark 捕获列表显示了一系列交替出现的 GET 请求行和 200 OK 响应行。当所有引用的资源文件都被成功接收后，浏览器将它们渲染在对应的位置并开始播放音乐，至此网页加载完成。随后，在经过一段空闲时间或用户关闭页面后，客户端与服务器通过 TCP 四次挥手过程断开连接，结束了本次 HTTP 会话

另外在捕获过程中，观察到了 TLSv1.2 的 Client Hello /Application Data 报文。这是浏览器尝试建立 HTTPS 安全连接或后台服务通信产生的流量。由于本次实验仅针对 HTTP 明文协议进行分析，且 Web 服务器未配置 SSL 证书，故该加密握手过程不属于本次实验的核心交互范围，予以忽略。

六、 总结

通过本次计算机网络实验，我成功完成了基于 IIS 的 Web 服务器搭建与配置，并实现了包含多媒体元素的网页发布。在利用 Wireshark 进行协议分析的过程中，我不仅掌握了网络抓包工具的过滤器使用与环回流量捕获技巧，更通过对实际数据流的观察，直观地理解了抽象的网络协议分层概念。我深入分析了 HTTP 报文在不同协议层中的具体封装形式，明确了源 IP 与目的 IP 的寻址逻辑、TCP 端口的复用机制以及 HTTP 状态码的实际含义。此外，通过观察浏览器对 HTML、图片及音频文件的加载过程，我深刻体会到了浏览器解析文档后发起并发请求的工作模式以及 TCP 协议在底层提供的数据重组与可靠传输服务。总体而言，本次实验有效地将理论知识与工程实践相结合，验证了 HTTP 协议简单快速、灵活无状态的核心特征，为理解网络通信原理奠定了坚实基础。