

实验报告及代码和设计评分细则

评 分 项 目		满分	得分	备注
文档格式（段落、行间距、缩进、图表、编号等）		10		
感想（含思政）		10		
意见和建议		10		
验收时间		10		
Socket 编程	代码可读性	10		
	注释	10		
	软件体系结构	30		
	问题描述及解决方案	10		
实验报告总分		100		
教师签名			日 期	

目 录

一、 实验概述	1
1.1 实验名称	1
1.2 实验目的	1
1.3 实验环境	1
1.4 实验内容	1
1.5 实验要求	1
二、 实验过程	1
三、 实验测试与分析	1
3.1 系统测试及结果说明	1
3.2 遇到的问题及解决方法	7
3.3 设计方案存在的不足	8
四、 实验总结	10
4.1 实验感想	10
4.2 意见和建议	10

Socket 编程实验

一、 实验概述

1.1 实验名称

Socket 编程实验。

1.2 实验目的

通过 socket 程序的编写、调试，了解计算机网络可靠传输协议，熟悉基于 UDP 协议的 socket 编程方法，掌握如何开发基于 TCP/UDP 的网络应用。

1.3 实验环境

操作系统：Windows/Linux

编程语言：C, C++

1.4 实验内容

完成一个 TFTP 协议客户端程序，实现一下要求：

- (1) 严格按照 TFTP 协议与标准 TFTP 服务器通信；
- (2) 能够实现两种不同的传输模式 netascii 和 octet；
- (3) 能够将文件上传到 TFTP 服务器；
- (4) 能够从 TFTP 服务器下载指定文件；
- (5) 能够向用户展现文件操作的结果：文件传输成功/传输失败；
- (6) 针对传输失败的文件，能够提示失败的具体原因；
- (7) 能够显示文件上传与下载的吞吐量；
- (8) 能够记录日志，对于用户操作、传输成功，传输失败，超时重传等行为记录日志；
- (9) 人机交互友好（图形界面/命令行界面均可）；
- (10) 额外功能的实现，将视具体情况予以一定加分。

1.5 实验要求

- (1) 必须基于 Socket 编程，不能直接借用任何现成的组件、封装的库等；
- (2) 提交实验设计报告和源代码；实验设计报告必须包括程序流程图，源代码必须加详细

注释。

- (3) 实验设计报告需提交纸质档和电子档，源代码、编译说明需提交电子档。
- (4) 基于自己的实验设计报告，通过实验课的上机试验，将源代码编译成功，运行演示给实验指导教师检查。

二、 实验过程

2.1 系统结构设计

2.1.1 模块框图

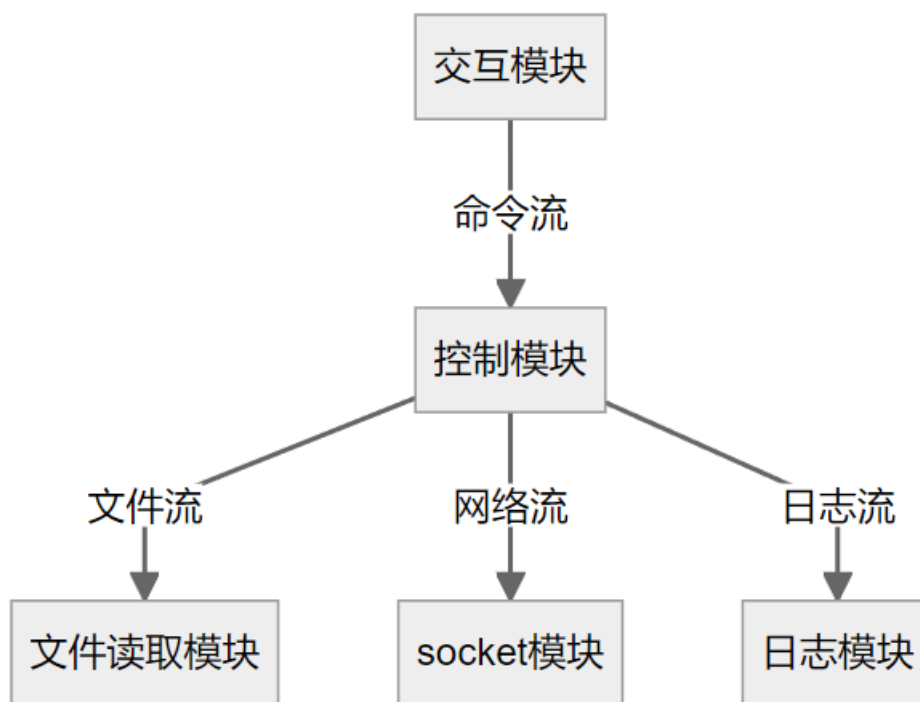


图 2-1-1 模块框图

2.1.2 模块功能说明

如图 2-1-1 所示，系统分为：命令行交互模块，文件读取模块，socket 模块，日志模块四大模块，功能如下：

1. **命令行交互模块**：从用户侧读取输入，包括文件名，传输方式等，然后交给控制模块，由控制模块返回信息再输出到控制台
2. **控制模块**：作为中间件，负责读入命令行的命令，以及根据命令控制 3 个模块进行处理
3. **文件读取模块**：主要是封装了读写文件的逻辑
4. **Socket 模块**：最主要的模块，负责与服务器进行通信，检测丢包操作等
5. **日志模块**：处理程序运行时产生的日志，以及程序关闭前输出到日志文件中

2.1.3 模块接口说明

文件读取模块，socket 模块和日志模块作为可抽象的公共接口，全部抽象为函数接口，每个函数的功能如下表 2-1-2 所示，函数之间的调用关系如图 2-1-2 所示

表 2-1-1 接口函数表格

模块类型	函数名	描述
文件读取模块	<code>bool upload(char* filename)</code>	上传文件到服务器
	<code>bool download(char* remoteFile, char* localFile)</code>	从服务器下载文件到本地
	<code>FILE* openLocalFile(const char* localFile, int choose)</code>	打开本地文件以供读取或写入（根据选择打开文件）
Socket 模块	<code>void setNonBlocking(SOCKET sockfd)</code>	将套接字设置为非阻塞模式
	<code>void initializeSockets()</code>	初始化套接字
	<code>void cleanupSockets()</code>	清理套接字
	<code>bool sendRequestPacket(int sock, tftpPacket *packet, sockaddr_in *serverAddr, int addr_len)</code>	发送 TFTP 请求数据包到服务器
	<code>bool waitForAcknowledgement(int sock, tftpPacket *rcv_packet, sockaddr_in *serverAddr, int *addr_len, unsigned short expectedBlock)</code>	等待接收 TFTP 确认数据包（ACK）
日志模块	<code>void logToFile(const char* message)</code>	记录消息到日志文件
	<code>void logError(const char *message)</code>	记录错误消息到日志文件

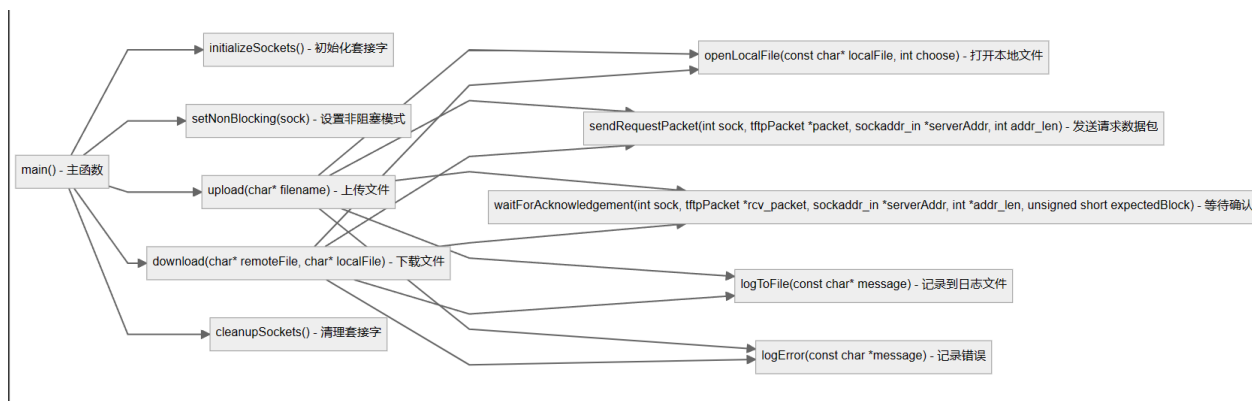


图 2-1-2 函数调用关系

2.1.4 数据处理流程

上传文件

在主程序 main() 中，通过调用 initializeSockets() 函数来初始化 Winsock，用户被要求输入服务器和客户端的 IP 地址。服务器端口固定为 69，客户端端口设置为 0（即自动分配）。创建客户端的 SOCKET 并通过 bind 函数与客户端端口进行绑定。随后，通过 ioctlsocket 函数设置该 SOCKET 为非阻塞模式。

用户输入操作序号和需要上传的文件名后，程序调用 upload() 函数来处理文件上传。

在 upload() 函数内，首先允许用户选择传输模式，即 netascii 或 octet。然后构建 WRQ（Write Request）数据包，该数据包的操作码为 2，文件名和传输模式也包含在这个数据包中。数据包通过 sendRequestPacket 函数发送。

一旦发送 WRQ 请求，程序将进入等待服务器 ACK（应答）的阶段。这里采用了一个轮询机制，即每隔 20ms 尝试接收 ACK，最多等待 3 秒。如果未接收到 ACK，该请求将重传，最多重传 3 次。如果仍未收到 ACK，该次传输将终止。

如果成功接收到 ACK，程序进入文件传输阶段。根据选择的传输模式，使用 fread 函数读取文件并构建数据块。对于每个数据块，最多重传 3 次，与服务器进行 ACK 确认的交互。

当所有数据块成功传输后，程序计算并显示传输的数据大小、耗时、速率和掉包率。

下载文件

下载文件的处理流程与上传文件类似。在主程序中，用户选择下载文件操作并输入文件名后，程序调用 download() 函数。

在 download() 函数内，用户选择传输模式（netascii 或 octet），构建 RRQ（Read Request）数据包并通过 sendRequestPacket 函数发送给服务器。

发送 RRQ 请求后，程序进入数据接收和 ACK 确认的循环。这里同样使用了一个轮询机制，每隔 20ms 尝试接收数据，最多等待 3 秒。对于接收到的每个数据块，程序会发送一个 ACK 包以确认其接收，并通过 fwrite 函数将数据写入本地文件。

当所有数据块接收并写入完成后，程序同样计算并显示传输的数据大小、耗时、速率和掉包率。

2.2 详细设计

2.2.1 文件上传

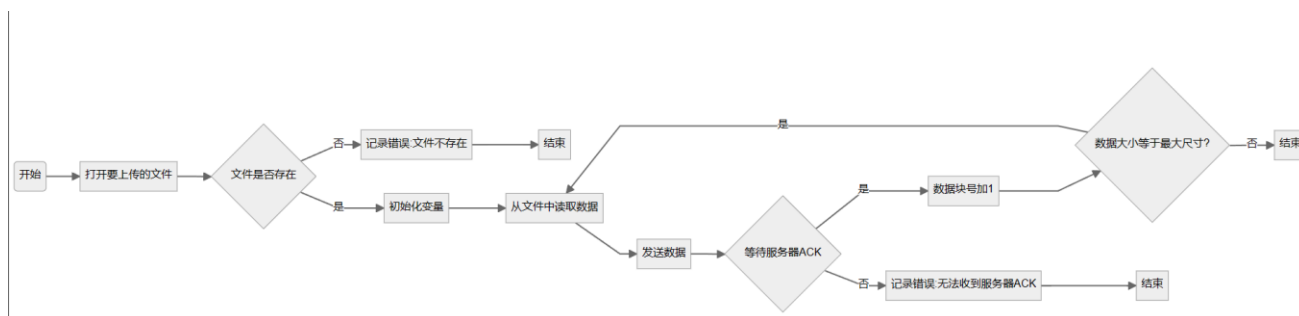


图 2-2-1 上传文件流程图

如前所述，这是文件上传功能的实现过程。用户首先被提示选择传输模式（netascii 或 octet）。随后，程序尝试发送写请求（WRQ）数据包到服务器。重传机制被置于 WRQ 发送和其他数据包发送环节，以确保数据的正确传输。

在成功发送 WRQ 后，程序会打开预先选定的文件并检查文件是否存在。不存在的话，程序会记录错误并终止。若文件存在，初始化各种变量和计数器，并开始读取文件。

文件数据被分块并发送到服务器，每次发送后，客户端都会等待服务器的确认报文（ACK）。如果在预定时间内接收到 ACK，客户端会更新数据块编号，并准备发送下一块数据。所有这些信息，包括收到的确认报文和发送的数据包，都会被记录在日志文件中。

如果客户端在预定时间内未收到服务器的 ACK，将触发重传机制。这个机制最多触发三次重传，超过这个限制则终止传输，并在日志文件中记录相关信息。

在数据传输过程中，如果发现发送的数据包大小少于最大数据包大小，说明文件已经传输完成，进程会正常结束。如果在任何环节接收到错误报文，错误信息会被记录在日志文件中，同时断开连接（伪连接）并关闭 socket。

2.2.2 文件下载

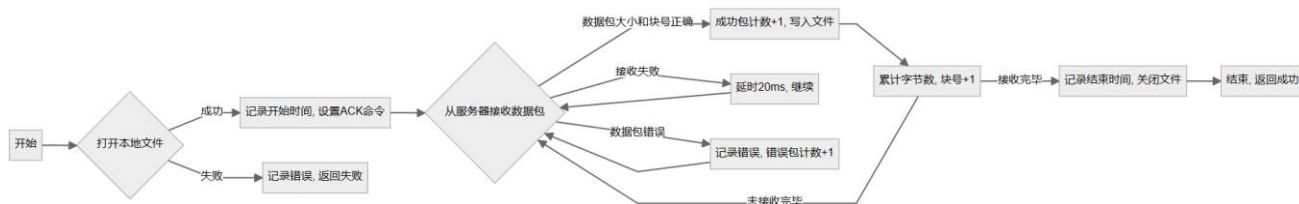


图 2-2-2 下载文件流程图

如上图所示：与上传文件同样，系统尝试对本地文件进行打开操作。如果成功，程序会记录当前的开始时间并设置 ACK 命令，进入数据接收的循环；如果失败，则会记录错误信息并返回失败状态。

一旦进入数据接收模式，程序从服务器接收数据包。这里有三种可能的结果：第一，如果数据包大小和块号都正确，那么成功的数据包计数会增加 1，并将数据写入本地文件；第二，

如果接收失败，则程序会延迟 20ms 后重新进入接收循环；第三，如果数据包出现错误，程序会记录错误并将错误包计数增加 1。

对于成功接收的数据包，程序会累计字节数并使块号加 1。这一循环会持续进行，直至全部数据接收完毕。在这一点上，程序会记录结束时间并关闭本地文件。

整个流程在接收所有数据包后终结，并返回成功状态。

2.3 代码实现

2.3.1 文件上传代码和注释

```

170 sendPacket.cmd = htons(hostshort: CMD_DATA); // 设置数据命令
171 start = clock(); // 记录开始上传的时间
172
173 do {
174     memset(Dst: sendPacket.data, Val: 0, Size: sizeof(sendPacket.data)); // 清零发送数据包的数据字段
175     sendPacket.block = htons(hostshort: block); // 设置数据块号
176     s_size = fread(Buffer: sendPacket.data, ElementSize: 1, ElementCount: DATA_SIZE, Stream: fp); // 从文件中读取数据到发送数据包的数据字段
177     transByte += s_size; // 累计已传输的字节
178
179     // 重传循环，最多重传PKT_MAX_RXMT次
180     for (rxmt = 0; rxmt < PKT_MAX_RXMT; rxmt++) {
181         if (sendRequestPacket(sock, packet: &sendPacket, serverAddr: &serverAddr, addr_len)) { // 发送数据包
182             spack++; // 成功包计数加1
183             if (waitForAcknowledgement(sock, rcv_packet: &rcv_packet, serverAddr: &serverAddr, addr_len: (int*)&addr_len, expectedBlock: block)) { // 等待
184                 break; // 接收到应答则跳出重传循环
185             } else {
186                 logError(message: "Can't receive ACK, Retransmission"); // 接收应答失败则记录错误，继续重传
187                 epack++; // 错误包计数加1
188                 continue;
189             }
190         }
191     }
192
193     if (rxmt >= PKT_MAX_RXMT) { // 如果重传次数超过最大值，则记录错误，关闭文件，并返回失败
194         logError(message: "Could not receive ACK from server.");
195         fclose(Stream: fp);
196         return false;
197     }
198
199     block++; // 数据块号加1
200 } while (s_size == DATA_SIZE); // 如果发送数据包的大小等于最大数据包大小，则继续发送下一个数据包
201
202 end = clock(); // 记录结束上传的时间
203 fclose(Stream: fp); // 关闭文件

```

图 2-3-1 文件上传代码和注释

2.3.2 文件下载代码和注释

```
263 start = clock(); // 记录开始下载的时间
264 sendPacket.cmd = htons(hostshort: CMD_ACK); // 设置ACK命令到发送数据包
265
266 int r_size; // 定义一个变量用于记录接收到的数据包的大小
267 do {
268     r_size = recvfrom(s: sock, buf: (char*)&rcv_packet, len: sizeof(tftpPacket), flags: 0, from: (struct sockaddr*)&serverAddr, fromlen: (int*)&addr_len);
269     // 如果接收到的数据包大小正确，并且命令是数据命令，并且数据块号正确，则处理接收到的数据包
270     if (r_size >= 4 && rcv_packet.cmd == htons(hostshort: CMD_DATA) && rcv_packet.block == htons(hostshort: block)) {
271         spack++; // 成功数据包计数加1
272         sendPacket.block = rcv_packet.block; // 设置接收到的数据块号到发送数据包
273         sendto(s: sock, buf: (char*)&sendPacket, len: sizeof(tftpPacket), flags: 0, to: (struct sockaddr*)&serverAddr, tolen: addr_len); // 发送ACK数据包到服务器
274         fwrite(Buffer: rcv_packet.data, ElementSize: 1, ElementCount: r_size - 4, Stream: fp); // 将接收到的数据写入到本地文件
275     } else if (r_size == -1) { // 如果接收数据包失败，则延时20毫秒后继续
276         Sleep(dwMilliseconds: 20);
277         continue;
278     } else { // 如果接收到的数据包有错误，则记录错误，并且错误数据包计数加1
279         epack++;
280         logError(message: "接收数据包错误");
281         continue;
282     }
283
284     transByte += (r_size - 4); // 累计接收到的数据字节数
285     block++; // 数据块号加1
286 } while (r_size == DATA_SIZE + 4); // 如果接收到的数据包大小等于最大数据包大小，则继续接收下一个数据包
287
288 end = clock(); // 记录结束下载的时间
289 fclose(Stream: fp); // 关闭本地文件
290
291 // 计算并显示下载的统计信息
292 double consumeTime = ((double)(end - start)) / CLK_TCK;
293 printf(Format: "文件大小: %.2f KB, 耗时: %.3f s\n", transByte / 1024, consumeTime);
294 printf(Format: "下载速度: %.2f kB/s\n", transByte / (1024 * consumeTime));
295 printf(Format: "丢包率: %.1f%%\n", epack / (epack + spack) * 100);
```

图 2-3-2 文件下载代码和注释

三、 实验测试与分析

3.1 系统测试及结果说明

1. 系统测试环境:

硬件环境:

AMD Ryzen 5 5625U with Radeon Graphics 2.30 GHz

Qualcomm FastConnect 6900 Wi-Fi 6E Dual Band Simultaneous (DBS) WiFi6E Network Adapter

软件环境:

Windows 11 Pro 22H2.1992

VSCode

gcc version 13.1.0 (MinGW-W64 x86_64-msvcrt-posix-seh, built by Brecht Sanders)

clumsy

tftp server

2. 系统测试方案

1. 不开启 clumsy 下载文件(ascii+bin)

2. 不开启 clumsy 上传文件(ascii+bin)

3. 开启 clumsy 下载文件

4. 开启 clumsy 上传文件

3. 系统测试过程和结果

1. 不开启 clumsy 下载文件(ascii+bin)

Tftp server 工作目录: D:\Misc\tftp, 目录下原有文件 big.txt, mid.txt, small.txt 和 pic.jpg

如下面的图所示, 分别用 ascii 和 bin 模式下载服务器上的 big.txt 文件(大小为 7.3KB), 采用 ascii 传输, 耗时 1s 左右, 下载速度 7KB/s; 采用 bin 模式传输耗时 2s, 速度 3.43KB/s

传输完毕后, 通过查看 tftp server 的 log 可知传输正确, 在 client 的工作目录下得到了传输的结果, 经过验证, 文件是一样的, 可知程序执行结果正确

```
E:\Code\CPP\TFTP
请输入服务器IP: 127.0.0.1
请输入客户端IP: 127.0.0.1
客户端socket创建成功!
请选择要进行的操作:
1.上传文件
2.下载文件
3.退出
2
请输入要下载的文件名:
big.txt
请输入要保存的文件名:
ascii.txt
mode:(1.netascii 2.octet)
1
Received: blockNum=1, dataSize=512
Received: blockNum=2, dataSize=512
Received: blockNum=3, dataSize=512
Received: blockNum=4, dataSize=512
Received: blockNum=5, dataSize=512
Received: blockNum=6, dataSize=512
Received: blockNum=7, dataSize=512
Received: blockNum=8, dataSize=512
Received: blockNum=9, dataSize=512
Received: blockNum=10, dataSize=512
Received: blockNum=11, dataSize=512
Received: blockNum=12, dataSize=512
Received: blockNum=13, dataSize=512
Received: blockNum=14, dataSize=512
Received: blockNum=15, dataSize=312
文件大小: 7.30 KB, 耗时: 1.011 s
下载速度: 7.23 kB/s
```

图 3-1-1 不开启 clumsy 下载文件(ascii)

```
E:\Code\CPP\TFTP
-----下载文件开始-----
请选择要进行的操作:
1.上传文件
2.下载文件
3.退出
2
请输入要下载的文件名:
big.txt
请输入要保存的文件名:
bin.txt
mode:(1.netascii 2.octet)
2
Received: blockNum=1, dataSize=512
Received: blockNum=2, dataSize=512
Received: blockNum=3, dataSize=512
Received: blockNum=4, dataSize=512
Received: blockNum=5, dataSize=512
Received: blockNum=6, dataSize=512
Received: blockNum=7, dataSize=512
Received: blockNum=8, dataSize=512
Received: blockNum=9, dataSize=512
Received: blockNum=10, dataSize=512
Received: blockNum=11, dataSize=512
Received: blockNum=12, dataSize=512
Received: blockNum=13, dataSize=512
Received: blockNum=14, dataSize=512
Received: blockNum=15, dataSize=312
文件大小: 7.30 KB, 耗时: 2.128 s
下载速度: 3.43 kB/s
丢包率 :0.0%
```

图 3-1-2 不开启 clumsy 下载文件(bin)

```
File <big.txt>: error 2 in system call createfile. The system cannot find the file specified.
Connection received from 127.0.0.1 on port 65360 [13/10 09:00:20.046]
Read request for file <big.txt>. Mode netascii [13/10 09:00:20.046]
Using local port 65361 [13/10 09:00:20.046]
<big.txt>: sent 15 blks, 7480 bytes in 0 s. 0 blk resent [13/10 09:00:20.492]
Connection received from 127.0.0.1 on port 65360 [13/10 09:02:06.602]
Read request for file <big.txt>. Mode octet [13/10 09:02:06.602]
Using local port 49201 [13/10 09:02:06.602]
<big.txt>: sent 15 blks, 7480 bytes in 1 s. 0 blk resent [13/10 09:02:07.024]
```

图 3-1-3 tftp server log

ascii.txt	2023/10/13 9:00	Text Document	8 KB
bin.txt	2023/10/13 9:02	Text Document	8 KB

图 3-1-4 client 下载的文件

2. 不开启 clumsy 上传文件(ascii+bin)

如下面的图所示，分别用 ascii 和 bin 模式下载服务器上的 big.txt 文件(大小为 7.3KB)，采用 ascii 传输，耗时 0.048s，上传速度 149KB/s，采用 bin 传输耗时 0.026s，速度 281KB/s

传输完毕后，通过查看 tftp server 的 log 可知传输正确，在 server 的工作目录下得到了上传的结果，经过验证，文件是一样的，可知程序执行结果正确

```
nu> E:\Code\CPP\TFTP
1
请输入要上传的文件名:
bin.txt
mode:(1.netascii 2.octet)
1
Sending the block:1
Sending the block:2
Sending the block:3
Sending the block:4
Sending the block:5
Sending the block:6
Sending the block:7
Sending the block:8
Sending the block:9
Sending the block:10
Sending the block:11
Sending the block:12
Sending the block:13
Sending the block:14
Sending the block:15
上传文件成功!
文件大小: 7.17 KB, 耗时: 0.048 s
上传速度: 149.41 KB/s
丢包率: 0.00%
-----上传文件开始-----

请选择要进行的操作:
1.上传文件
2.下载文件
3.退出
|
```

图 3-1-5 不开启 clumsy 上传文件(ascii)

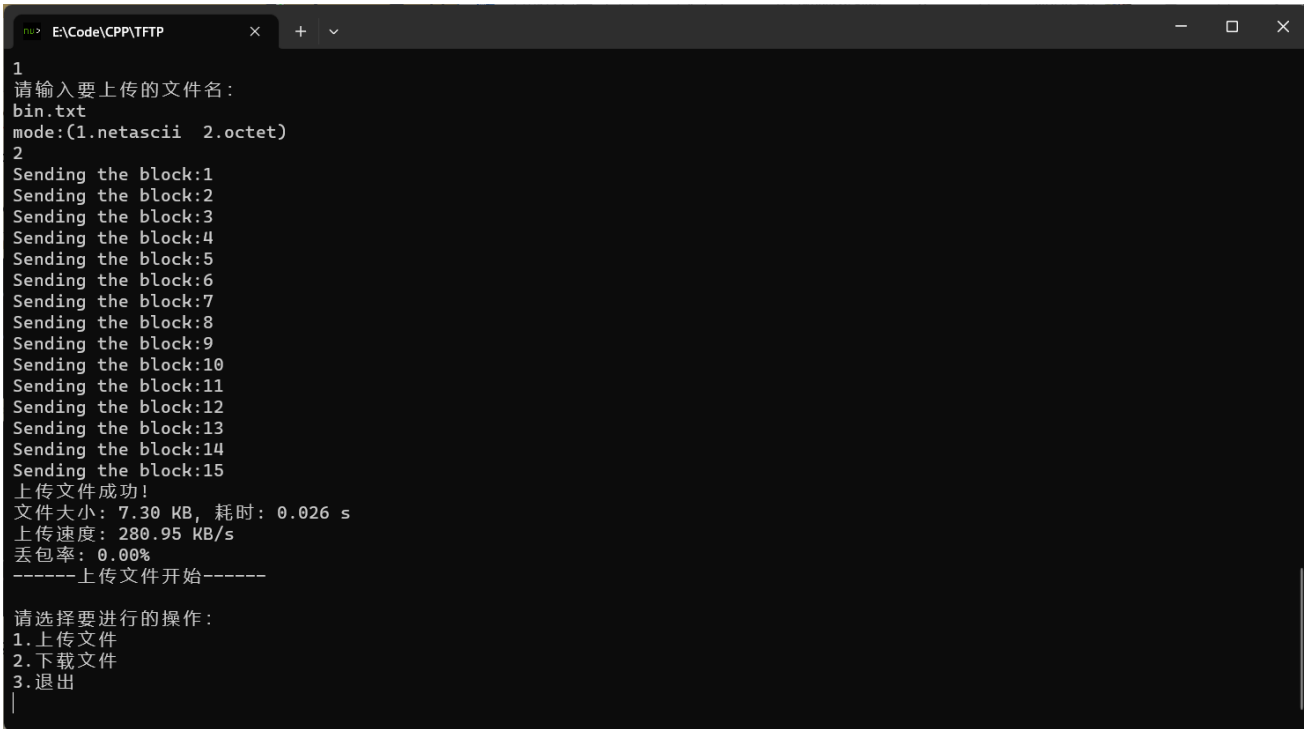


图 3-1-6 不开启 clumsy 上传文件(bin)

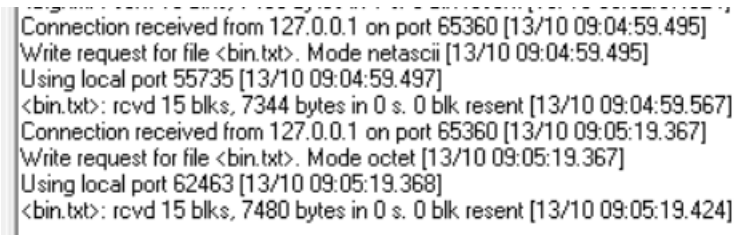


图 3-1-7 tftp server log

Data (D:) > Misc > tftp				Search tftp
Name	Date modified	Type	Size	
big.txt	2023/10/13 8:59	Text Document	8 KB	
mid.txt	2023/10/6 22:40	Text Document	3 KB	
pic.jpg	2023/2/11 11:30	JPG File	143 KB	
small.txt	2023/10/6 22:33	Text Document	1 KB	
bin.txt	2023/10/13 9:05	Text Document	8 KB	

图 3-1-8 tftp server 目录下的上传的文件

3. 开启 clumsy 下载文件

如图设置 clumsy 的 drop 选项，表示丢包率 10%

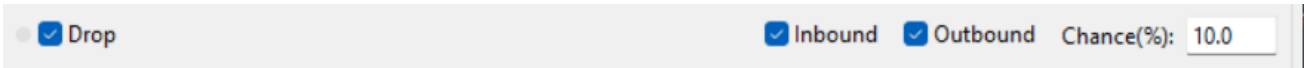
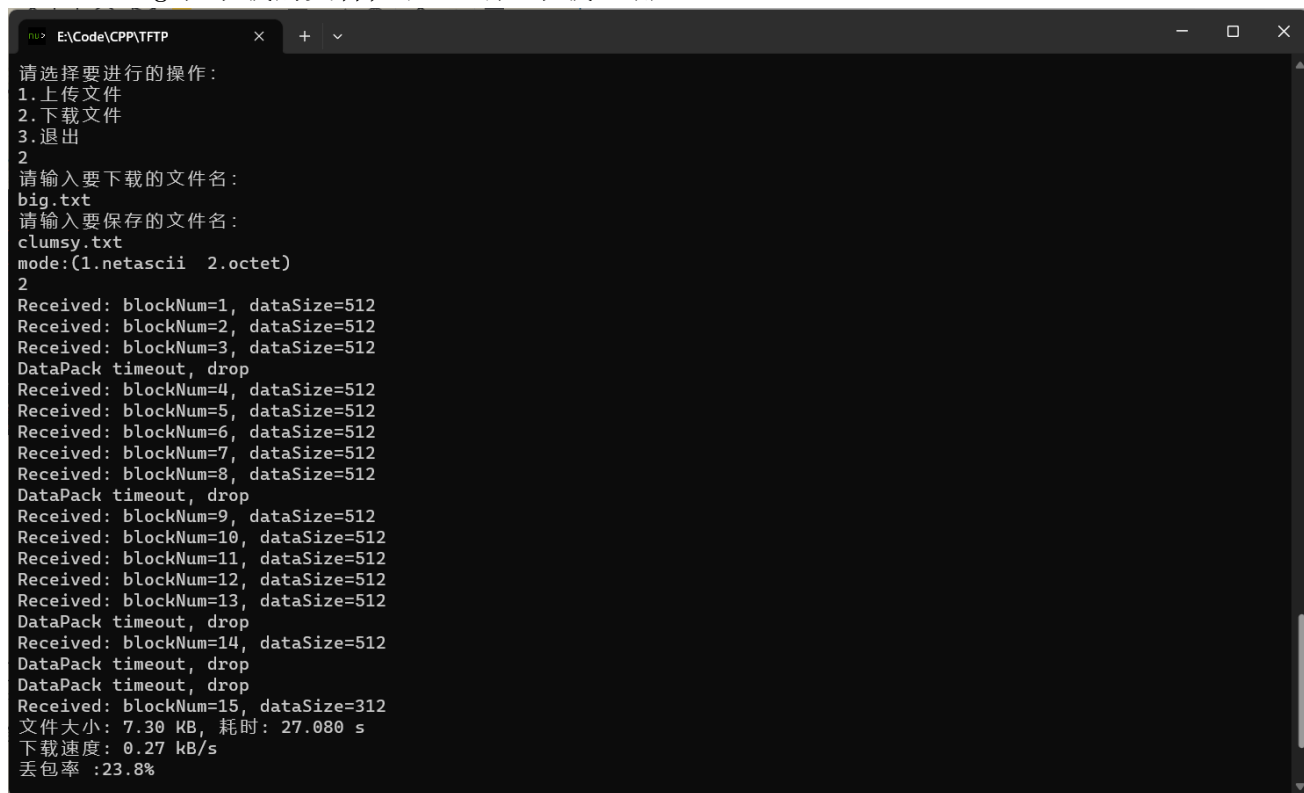


图 3-1-9 clumsy 设置

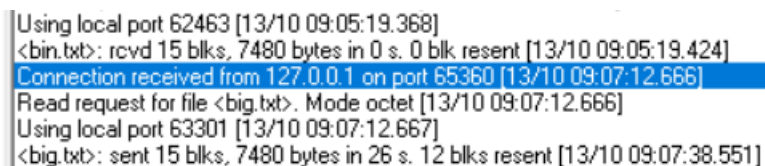
如图所示：

选择下载 big.txt，保存为 clumsy.txt，采用二进制传输，如图所示，clumsy 开启后出现了 5 次丢包，丢包后进行了重传，体现了可靠传输的特性，最后检查 tftp server log 和下载的文件，验证可知下载正确



```
E:\Code\CPP\TFTP
请选择要进行的操作：
1.上传文件
2.下载文件
3.退出
2
请输入要下载的文件名：
big.txt
请输入要保存的文件名：
clumsy.txt
mode:(1.netascii 2.octet)
2
Received: blockNum=1, dataSize=512
Received: blockNum=2, dataSize=512
Received: blockNum=3, dataSize=512
DataPack timeout, drop
Received: blockNum=4, dataSize=512
Received: blockNum=5, dataSize=512
Received: blockNum=6, dataSize=512
Received: blockNum=7, dataSize=512
Received: blockNum=8, dataSize=512
DataPack timeout, drop
Received: blockNum=9, dataSize=512
Received: blockNum=10, dataSize=512
Received: blockNum=11, dataSize=512
Received: blockNum=12, dataSize=512
Received: blockNum=13, dataSize=512
DataPack timeout, drop
Received: blockNum=14, dataSize=512
DataPack timeout, drop
DataPack timeout, drop
Received: blockNum=15, dataSize=312
文件大小：7.30 KB，耗时：27.080 s
下载速度：0.27 KB/s
丢包率：23.8%
```

图 3-1-10 开启 clumsy 下载文件



```
Using local port 62463 [13/10 09:05:19.368]
<bin.txt>: rcvd 15 blks, 7480 bytes in 0 s. 0 blk resent [13/10 09:05:19.424]
Connection received from 127.0.0.1 on port 65360 [13/10 09:07:12.666]
Read request for file <big.txt>. Mode octet [13/10 09:07:12.666]
Using local port 63301 [13/10 09:07:12.667]
<big.txt>: sent 15 blks, 7480 bytes in 26 s. 12 blks resent [13/10 09:07:38.551]
```

图 3-1-11 tftp server log

4. 开启 clumsy 上传文件

如图设置 clumsy 的 drop 选项，表示丢包率 10%

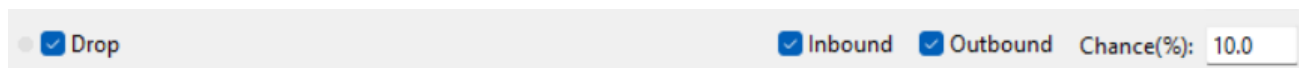
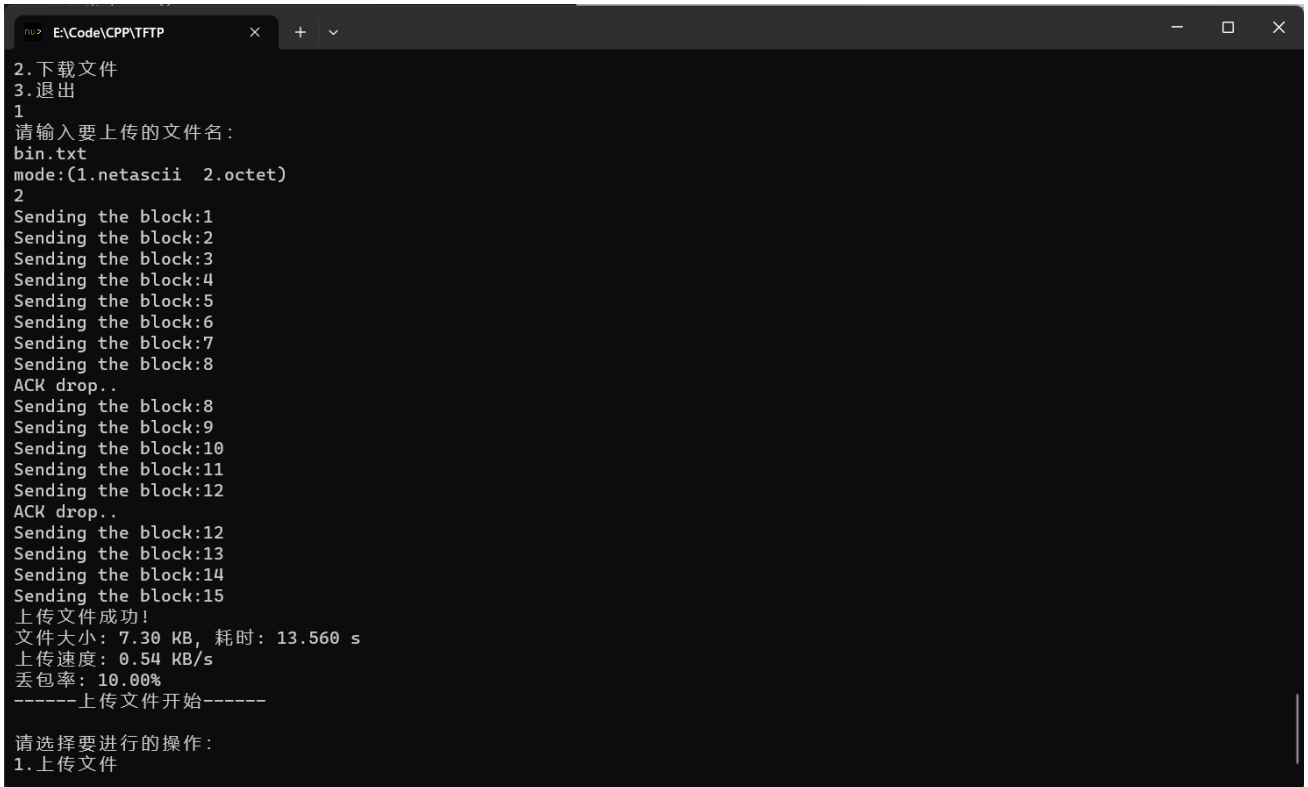


图 3-1-12 clumsy 设置

如图所示：

选择下载 big.txt，保存为 clumsy.txt，采用二进制传输，如图所示，clumsy 开启后出现了 2 次 ack drop，丢包后进行了重传，体现了可靠传输的特性，最后检查 tftp server log 和上传的文件，验证可知上传正确



```

E:\Code\CPP\TFTP
2. 下载文件
3. 退出
1
请输入要上传的文件名:
bin.txt
mode:(1.netascii 2.octet)
2
Sending the block:1
Sending the block:2
Sending the block:3
Sending the block:4
Sending the block:5
Sending the block:6
Sending the block:7
Sending the block:8
ACK drop..
Sending the block:8
Sending the block:9
Sending the block:10
Sending the block:11
Sending the block:12
ACK drop..
Sending the block:12
Sending the block:13
Sending the block:14
Sending the block:15
上传文件成功!
文件大小: 7.30 KB, 耗时: 13.560 s
上传速度: 0.54 KB/s
丢包率: 10.00%
-----上传文件开始-----

请选择要进行的操作:
1. 上传文件
```

图 3-1-13 开启 clumsy 上传文件

检查程序 log，如图所示为 ack 的日志，以及上传完之后的所有日志

```
tftp.log
6  WARNING: upload bin.txt, mode: octet, Can't receive ACK, Retransmission
7  Fri Oct 13 09:09:36 2023
8  WARNING: upload bin.txt, mode: octet, Can't receive ACK, Retransmission
9  Fri Oct 13 09:09:41 2023
10 WARNING: upload bin.txt, mode: octet, Can't receive ACK, Retransmission
11 Fri Oct 13 09:09:41 2023
12 ERROR: upload bin.txt, mode: octet, Could not receive ACK from server.
13 Fri Oct 13 09:10:52 2023
14 WARNING: upload bin.txt, mode: octet, Can't receive ACK, Retransmission
15 Fri Oct 13 09:10:57 2023
16 WARNING: upload bin.txt, mode: octet, Can't receive ACK, Retransmission
17 Fri Oct 13 09:11:02 2023
18 WARNING: upload bin.txt, mode: octet, Can't receive ACK, Retransmission
19 Fri Oct 13 09:11:02 2023
20 ERROR: upload bin.txt, mode: octet, Could not receive ACK from server.
21 Fri Oct 13 09:11:17 2023
22 WARNING: upload bin.txt, mode: octet, Can't receive ACK, Retransmission
23 Fri Oct 13 09:11:24 2023
24 WARNING: upload bin.txt, mode: octet, Can't receive ACK, Retransmission
25 Fri Oct 13 09:11:25 2023
26
27 INFO: upload bin.txt, mode: octet, size: 7.3 kB, consuming time: 13.5600 s
28
```

图 3-1-14 程序 log

3.2 遇到的问题及解决方法

1. Clang++的 fopen 函数不报错

使用 clang++编译的程序在测试 fopen 时不报错(实际上是报错方法与 g++有差异)，在文件判错的阶段出现了问题，一直得不到 fopen 不存在文件的时候的错误，经过折腾还是决定使用更加熟悉的 g++编译

2. g++ 编译程序的时候出现一大堆错误：

```

4-w64-mingw32/bin/ld.exe: C:\Users\KuJyo\AppData\Local\Temp\ccD8KEpp.o:main.cpp:(.text+0x1755): undefined reference to `
__imp_WSAStartup'
E:/SDK/scoop/Scoop/apps/mingw-winsys/13.1.0-16.0.5-11.0.0-r5/bin/./lib/gcc/x86_64-w64-mingw32/13.1.0/../../../../x86_64-
4-w64-mingw32/bin/ld.exe: C:\Users\KuJyo\AppData\Local\Temp\ccD8KEpp.o:main.cpp:(.text+0x179f): undefined reference to `
__imp_WSACleanup'
E:/SDK/scoop/Scoop/apps/mingw-winsys/13.1.0-16.0.5-11.0.0-r5/bin/./lib/gcc/x86_64-w64-mingw32/13.1.0/../../../../x86_64-
4-w64-mingw32/bin/ld.exe: C:\Users\KuJyo\AppData\Local\Temp\ccD8KEpp.o:main.cpp:(.text+0x182e): undefined reference to `
__imp_socket'
E:/SDK/scoop/Scoop/apps/mingw-winsys/13.1.0-16.0.5-11.0.0-r5/bin/./lib/gcc/x86_64-w64-mingw32/13.1.0/../../../../x86_64-
4-w64-mingw32/bin/ld.exe: C:\Users\KuJyo\AppData\Local\Temp\ccD8KEpp.o:main.cpp:(.text+0x1876): undefined reference to `
__imp_WSACleanup'
E:/SDK/scoop/Scoop/apps/mingw-winsys/13.1.0-16.0.5-11.0.0-r5/bin/./lib/gcc/x86_64-w64-mingw32/13.1.0/../../../../x86_64-
4-w64-mingw32/bin/ld.exe: C:\Users\KuJyo\AppData\Local\Temp\ccD8KEpp.o:main.cpp:(.text+0x18a6): undefined reference to `
__imp_htons'
E:/SDK/scoop/Scoop/apps/mingw-winsys/13.1.0-16.0.5-11.0.0-r5/bin/./lib/gcc/x86_64-w64-mingw32/13.1.0/../../../../x86_64-
4-w64-mingw32/bin/ld.exe: C:\Users\KuJyo\AppData\Local\Temp\ccD8KEpp.o:main.cpp:(.text+0x18bc): undefined reference to `
__imp_inet_addr'
E:/SDK/scoop/Scoop/apps/mingw-winsys/13.1.0-16.0.5-11.0.0-r5/bin/./lib/gcc/x86_64-w64-mingw32/13.1.0/../../../../x86_64-
4-w64-mingw32/bin/ld.exe: C:\Users\KuJyo\AppData\Local\Temp\ccD8KEpp.o:main.cpp:(.text+0x18e2): undefined reference to `
__imp_bind'
E:/SDK/scoop/Scoop/apps/mingw-winsys/13.1.0-16.0.5-11.0.0-r5/bin/./lib/gcc/x86_64-w64-mingw32/13.1.0/../../../../x86_64-
4-w64-mingw32/bin/ld.exe: C:\Users\KuJyo\AppData\Local\Temp\ccD8KEpp.o:main.cpp:(.text+0x1913): undefined reference to `
__imp_closesocket'
E:/SDK/scoop/Scoop/apps/mingw-winsys/13.1.0-16.0.5-11.0.0-r5/bin/./lib/gcc/x86_64-w64-mingw32/13.1.0/../../../../x86_64-
4-w64-mingw32/bin/ld.exe: C:\Users\KuJyo\AppData\Local\Temp\ccD8KEpp.o:main.cpp:(.text+0x191c): undefined reference to `
__imp_WSACleanup'
E:/SDK/scoop/Scoop/apps/mingw-winsys/13.1.0-16.0.5-11.0.0-r5/bin/./lib/gcc/x86_64-w64-mingw32/13.1.0/../../../../x86_64-
4-w64-mingw32/bin/ld.exe: C:\Users\KuJyo\AppData\Local\Temp\ccD8KEpp.o:main.cpp:(.text+0x194b): undefined reference to `
__imp_htons'
E:/SDK/scoop/Scoop/apps/mingw-winsys/13.1.0-16.0.5-11.0.0-r5/bin/./lib/gcc/x86_64-w64-mingw32/13.1.0/../../../../x86_64-
4-w64-mingw32/bin/ld.exe: C:\Users\KuJyo\AppData\Local\Temp\ccD8KEpp.o:main.cpp:(.text+0x1962): undefined reference to `
__imp_inet_addr'
collect2.exe: error: ld returned 1 exit status

```

图 3-2-1 问题 1

经过查看资料和同学指点，才发现是忘记了链接文件库，需要编译参数”

```
g++ main.cpp -o main.exe -lwsck32
```

3. tftp server 一直无法 bind

经查看是 tftp server 选错了网卡，tftp server 默认不是 127.0.0.1 的网卡，而是形如 10.12.173.91 的外网网卡（当然也可以改成这个 ip，不过既然是本地测试，那就不需要走一遍外网网卡）

4. 对有些函数的用法不太明确，比如说 sendto，recvfrom 等，通过查阅资料 and 不断尝试才明白用法

3.3 设计方案存在的不足

1. **发送数据包是单线程的：**就实际测试来看，单线程的停等协议发送数据包速度还是比较慢的，稍微大一点的文件（1MB）传送起来就特别慢，大部分时间耗在了等待数据包中，没有实现滑动窗口或者异步发送等待的操作
2. **错误提示不够明确：**对于传输的错误不太明确，以及丢包率的计算也还需要改进，与 tftp 并不是完全吻合
3. **功能单一：**只有上传和下载的功能，要成为一个可用的协议最好还是要加上一些列出文件，取文件头信息的命令
4. **边界情况处理不太明确：**例如同名文件处理，下载文件的文件在服务器上不存在等边界情况

基本没有处理，当然作为一个协议层面上来说似乎不是必要的

5. **数据块的探测伸缩：**每一个数据包固定了是512字节，如果探测IP帧的大小进而进行伸缩控制就最好，一方面避免IP层进行分片，另一方面充分利用网络带宽
6. **认证与安全功能：**作为一个应用层网络协议，可以进行一些basic auth的操作，以及增加一些密钥交换的功能(传输层安全)

四、 实验总结

4.1 实验感想

计网实验从实验一的计算机网络组网实验到实验二的仿真软件组网实验，带我们体验了组网的乐趣，自制网线比较有趣，第一次了解到网线的水晶头的细节结构，看到自己制作的网线在测试机上全部跑通也十分具有成就感。

之后的华为设备组网实验，先是在 eNSP 上组建虚拟网，不过这个软件要求太苛刻，自己的电脑怎么都装不上，还是特别折腾，只能在教室电脑上进行实验，然后在实物上实验，一睹交换机真容让我感到非常新奇。实验中与同学一起合作，不断猜想错误，以及参考实验手册和 ppt，最后实现了各主机互 ping，成功组网的感觉让人欢呼。

利用 eNSP 实现了校园网的虚拟组网，主要是考虑 IP 的分配问题，以及网段的划分，计算比较繁琐，最后以宏观视角看到了客户端与服务器的通信过程，收获颇丰！

在实验三套接字编程的过程中，我完成了一个符合 TFTP 协议要求的客户端程序，它可以与标准的 TFTP 服务器通信，实现文件的上传和下载，同时也能够友好地与用户交互，展示文件操作的结果和传输的吞吐量，虽然只有上传和下载功能，但是自己动手完全基于 socket 编程实现的感觉是不一样的！

在实验的过程中，遭遇了不少的困难和问题。记得在配置 TFTP 服务器时，遇到了一个令人头疼的问题，TFTP server 一直无法 bind。经过一番查找，发现是因为 TFTP 服务器默认选择了外网网卡，而不是 127.0.0.1 的本地网卡。于是我调整了网络设置，使其绑定到了正确的网卡，解决了这个问题。这个过程虽然耗时，但却让我学到了很多实际操作中可能遇到的网络问题，也让我明白了实验不仅仅是代码的编写，更是对网络环境的理解和掌握。

另外，在实验的编程过程中，我对一些函数的用法感到困惑，比如 sendto 和 recvfrom 等函数。课堂上没有教过的知识，还需要自己动手实践。于是我查阅了大量的资料，不断地尝试和调试，终于明白了这些函数的用法和实际应用。这个过程虽然艰辛，但收获颇丰，让我感受到了实践的力量，也让我明白了理论知识的重要性。

此次实验也让我体会到了我国网络安全和发展的重要性。在网络高速发展的今天，我们应该不断提高自己，学好网络知识，为我国的网络安全做出贡献。老师的耐心指导和同学的帮助，让我深刻体会到了集体的力量。通过这次实验，我不仅仅学到了专业知识，更学会了如何在团队中协作，如何面对困难不退缩。感谢老师的悉心教导和同学的帮助，让我在困境中不断进步，实现了自我超越！

4.2 意见和建议

1. **eNSP 软件问题：**而且与 WSL(windows sub linux) (子系统)有冲突(hyper-v)，还有虚拟化选项也经常出问题，软件也经常卡死，可能是版本太老，现用的版本是 win7 的，建议更新一下到 win10 以上能用的版本，或者用其他方法模拟

2. 实现 tftp client 的代码其实不算多，可以适当增加一些功能

3. 可以尝试考虑迁入分级通关(vmcourse)中，通过代码评测机制，逐步完成实验，一方面不用把验收搞的太麻烦，另一方面还能让学生自己查错解决问题，而且也能降低实验的难

度阶梯，一步步完成使实验更加清晰且平滑

原创性声明

本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。

已阅读并同意以下内容。

判定为不合格的一些情形：

- （1） 请人代做或冒名顶替者；
- （2） 替人做且不听劝告者；
- （3） 实验报告内容抄袭或雷同者；
- （4） 实验报告内容与实际实验内容不一致者；
- （5） 实验代码抄袭者。

作者签名：

以下内容皆不打印

附件 I 实验报告规范化要求

目 录 (黑体小 2 号加粗居中)

1	实验	1
1.1		1
1.2		3
1.2.1		7
1.3		10
	
	
	
3	实验	20
3.1		20
3.2		23
3.2.1		25
3.3		30
	
	
4	感想和建议	40
附录		45

(章为宋体小 4 号加粗，其余宋体小 4 号，字母、阿拉伯数字为 Time New Roman 小 4 号)

• • • • •

• • • • •

• • • • •

• • • • •

• • • • •

• • • • •

XX

$$E_2 = A_2 \sin(2\pi f_2 t + \varphi_{02} + \varphi_{path2}) \quad (3-2)$$

[illegible]表 3-1 ☐ × × × × × × × × × ×

	x x x	x x x	x x x	x x x
x x				
x x x				
x x	x x	x x		x x
x x x				
x x	x x	x x		x x
x x x				
x x	x x	x x		x x
x x x				
x x	x x	x x		x x

(表标题：位于表格上方，黑体小 4 号，字母、阿拉伯数字为 Time New Roman 小 4 号，表内容：宋体 5 号，字母、阿拉伯数字为 Time New Roman 5 号)

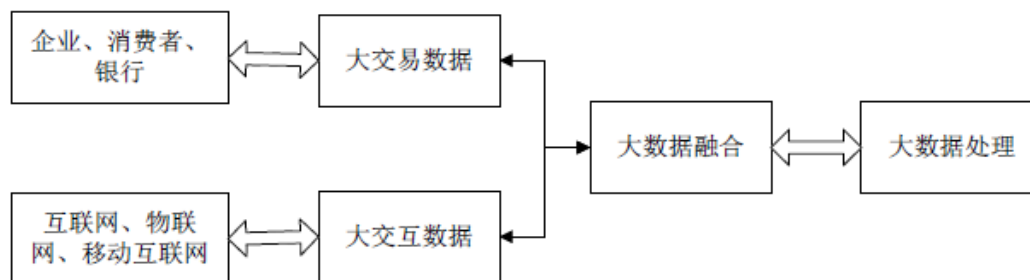
[illegible]

图 3-1 $\square \times \times \times \times \times \times \times \times \times \times$

(图标题：位于图下方，黑体小 4 号，字母、阿拉伯数字为 Time New Roman 小 4 号)

• • • • •

-----章与章之间插入分页符-----

附件 II 有关计算机网络实验检查的一些说明

一、检查方式

采用多样性开放式检查方式，包括：（1）现场教师检查；（2）课后助教检查。

详细说明如下：

（1）现场教师检查

包括实验课课内检查和实验机房预约检查。若实验课内检查不完的情况下，可预约检查，需向教学实验中心预约实验机房并同时预约指导教师或实验助教。

适用：局域网组网实验

（2）课后助教检查

此方式需先预约实验助教，需和助教预约合适的时间和地点。

适用：SOCKET 编程实验和综合布线实验

助教联系方式

班级	指导老师	助教姓名	QQ 号
网安 1901	王美珍	王莉莉	1456059136
网安 1902	高鹏毅	叶君妍	2239308357
网安 1903	肖凌	马林威	1609586673
网安 1904	李升辉	白康男	1659516895
信安 1901	陈凯	张洪涛	1101944844
信安 1902	梅松	李云云	3384109516
信安 1903	赵贻竹	白无暇	1318488173
信安 1904	张云鹤	熊博涛	465272324

无论是哪种方式，需能够充分说明设计的思路和实现方法。

二、检查要求

（1）SOCKET 编程：仅 C 和 C++ 语言，使用给定的服务器和网络质量控制软件，根据实验检查点及相应分值得分。

（2）局域网组网实验：仅检查实物配置，据实验检查点及相应分值得分。

（3）综合布线实验：据实验检查点及相应分值得分。

三、实验报告

实验报告主要体现无法在实际操作过程中展现的内容，包括设计理念、设计思路、设计方

案、实验过程中遇到的问题描述分析和解答、经验和心得体会、建议和总结。

实验报告内容**仅需撰写 SOCKET 编程部分**，但总结和感想部分则需包含三组实验。

实验报告禁止粘贴代码，若需展示代码，需使用流程图、状态机或伪代码表示。实验报告尽量只贴关键的图，无需贴大量的操作步骤图。

实验报告需严格按照指定的文档格式撰写，需提交 pdf 电子版本和纸质版本。

四、程序和设计文件

（1）程序

程序包括源码和编译后可执行文件。编译后的可执行文件要求打包所有的资源文件，可直接运行；**源码需有充分的注释**。

（2）设计文件

需包含所有的配置命令，打开后能够直接在平台上进行运行，并能展现出与实际操作演示一致的运行过程和结果。

五、最终提交的文件

（1）程序（源码+可执行程序）[电子版]；

（2）实验报告（pdf 格式）[电子版]；

（3）实验报告[纸质版]。

提交的电子版文件需压缩成一个文件包，命名方式：班级-学号-姓名。如：信安2001-U20010101-张三

六、检查时间节点和人员

SOCKET 编程实验需在第三次实验课（第十三周，信安专业在 11.29 日之前，网安专业在 12 月 1 日之前）结束前完成检查；

局域网组网实验需在第四次实验课（第十四周，信安专业在 12.6 日之前，网安专业在 12 月 8 日之前）结束前完成检查；

综合布线实验需在第十四周周末（12 月 10 日下午 6 点）前完成检查。