华 中 科 技 大 学

网络空间安全学院

本科：《计算机通信与网络实验》

实验报告

姓　　名 _____

班　　级 _____

学　　号 _____

联系方式 __

分　　数 _____

评 分 人 _____

报告要求

1. 报告不可以抄袭，发现雷同者记为 **0** 分。

2. 报告中不可以只粘贴大段代码，应是文字与图、表结合的，需要说明流程的时候，也应该用流程图或者伪代码来说明；如果发现有大段代码粘贴者，报告需重写。

3. 报告格式严格按照要求规范，并作为评分标准。

课程目标评价标准

（1）课程目标 1 的实验报告评价标准

实验报告评价标准			
优秀	良好	及格	不及格
报告内容完整，涵盖所有实验目标和要求。实验设计创新，能够展示对可靠数据传输机制的深入理解。编程实现正确，能使用伪代码方式表述，易于理解。实验结果准确，能够通过数据和图表清楚展示传输效率和可靠性。分析深入，能够对实验结果进行详细分析，并提出改进方案。报告格式规范，语言表达清晰，无语法错误。	报告内容较为完整，基本涵盖实验目标和要求。实验设计合理，能够较好地展示对可靠数据传输机制的理解。编程实现基本正确，能较为清晰的使用伪代码方式表述。实验结果较为准确，能够通过数据和图表展示传输效率和可靠性。分析较为深入，能够对实验结果进行分析，并提出一些改进方案。报告格式较为规范，语言表达较为清晰，有少量语法错误。	报告内容基本完整，部分涵盖实验目标和要求。实验设计一般，能够展示对可靠数据传输机制的基本理解。能使用伪代码描述，但不够清晰。实验结果基本准确，但数据和图表展示不够清楚。分析一般，能够对实验结果进行一些分析，但改进方案不够具体。报告格式基本规范，语言表达基本清晰，有较多语法错误。	报告内容不完整，未能涵盖实验目标和要求。实验设计不合理，未能展示对可靠数据传输机制的理解。编程实现存在较多问题，不能使用伪代码描述。实验结果不准确，数据和图表展示不清楚，无法有效展示传输效率和可靠性。分析浅显，未能对实验结果进行有效分析，缺乏改进方案。报告格式不规范，语言表达不清晰，有大量语法错误。

（2）课程目标 2 的实验报告评价标准

实验报告评价标准			
优秀	良好	及格	不及格
能够正确分析、设计计算机网络关键问题的解决方案，正确掌握可靠数据传输等关键问题的设计原理。	能够分析、设计计算机网络关键问题的解决方案，绝大部分完成可靠数据传输等关键问题的设计，但方案存在不影响功能的瑕疵。	能够设计计算机网络关键问题的解决方案，能够部分完成对可靠数据传输等关键问题的设计，但方案存在部分影响功能的瑕疵。	不能正确设计计算机网络关键问题的解决方案；不能完成可靠数据传输等关键问题的方案设计。

（3）课程目标 3 的实验报告评价标准

实验报告评价标准			
优秀	良好	及格	不及格
能够正确掌握计算机网络体系结构和 核心功能，理解当前网络体系和协议栈存在的不足和问题，并能够正确理解这些不足和问题。	能够绝大部分掌握计算机网络体系结构和核心功能，了解当前网络体系和协议栈存在不足和问题，并能够理解这些不足和问题对社会可持续发展的影响。	能够部分掌握计算机网络体系结构和核心功能，了解当前网络体系和协议栈存在的不足和问题，并能够理解这些不足和问题对社会可持续发展的影响。	不能正确掌握计算机网络体系结构和核心功能，不了解当前网络体系和协议栈存在的不足和问题了，不能对网络体系中的问题给出解决思路。

报告评分表

评 分 项 目		满分	得分	评分标准
Socket 编程	系统设计（目标 1）	15		15-11：能够给出明确需求，系统功能完整、正确和适当。 10-6：阐述的需求不够完整，系统的设计不够完整、恰当和准确。 5-0：需求不够明确，系统设计不够完整、正确和恰当。
	详细设计（目标 2）	15		15-11：函数和数据结构描述完整，关系清晰，流程设计正确规范。 10-6：函数和数据结构描述基本完整，流程设计基本正确。 5-0：函数和数据结构描述不完整，流程设计有错误。
	代码实现（目标 1）	10		10-8：代码能够实现设计的功能要求，考虑错误处理和边界条件。有充分的注释，代码格式规范。 7-5：错误处理和边界条件的考虑不足，有简单注释，格式较为规范。 4-0：代码未能完全实现功能要求，缺少错误处理和边界条件的考虑。注释不足，代码格式不够规范。
	测试及结果分析（目标 1、3）	20		20-14：测试方法科学、完整，结果分析准确完备。 13-8：测试方法描述基本正确、完整，结果分析准确完备。 7-0：测试仅针对数据集的通过性进行描述。
	问题描述及解决方案（目标 2、3）	10		10-8：遇到的问题及解决方案真实具体 7-5：遇到描述不够详细，解决方案不够具体 4-0：没有写什么内容。
感想（含思政）（目标 3）		10		10-8：感想真实具体。 7-5：感想比较空洞。 4-0：没有写什么感想。
遇到的问题和解决方案，及意见和建议（目标 3）		10		10-8：意见和建议有的放矢。 7-5：意见和建议不够明确。 4-0：没有写什么内容。
文档格式（文字字体字号、行间距、缩进按模板要求，图、表清晰标号正确）（目标 1）		10		10-8：格式规范美观，满足要求。 7-5：基本满足要求。 4-0：格式较为混乱。
总分		**100**		
教师签名			日期	

目　　录

1 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，实现以下要求：

（1）严格按照 TFTP 协议与标准 TFTP 客户端通信；

（2）能够使用标准 TFTP 客户端将文件上传到 TFTP 服务器；

（3）能够使用标准 TFTP 客户端从 TFTP 服务器下载指定文件；

（4）能够向用户展现文件操作的结果：文件传输成功/传输失败；

（5）针对传输失败的文件，能够提示失败的具体原因；

（6）能够显示文件上传与下载的吞吐量；

（7）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；

（8）人机交互友好（图形界面/命令行界面均可）；

（9）额外功能的实现，将视具体情况予以一定加分（总分不超过 100 分）。

1.5 实验要求

（1）必须基于 Socket 编程，不能直接使用任何现成的组件、封装的库等；

（2）提交实验设计报告和源代码；实验设计报告应按照实验报告模板，核心内容须包括程序流程图，源代码必须加详细注释。

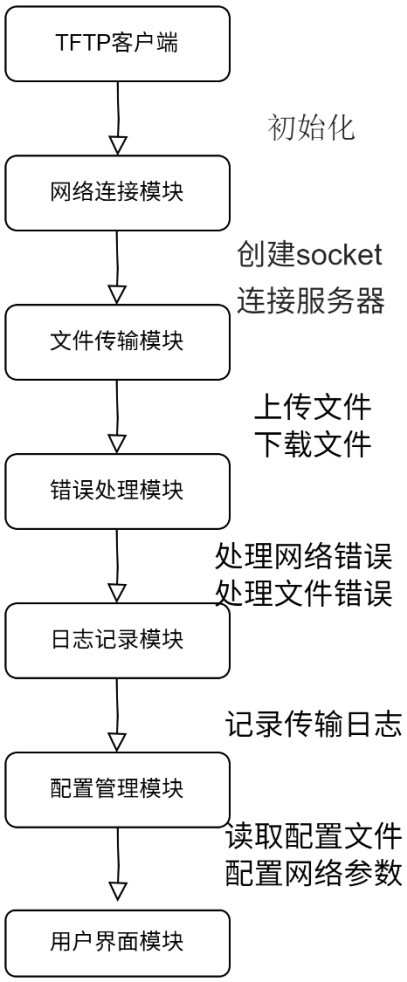
（3）实验设计报告、源代码、编译说明等文档提交按照课程组发布的实验资料提交说明要求进行提交。

（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。

2 实验过程

2.1 系统结构设计

2.1.1 模块框图



2.1.2 模块功能说明

TFTP 客户端：主模块，负责协调各个模块的工作，提供用户界面。

网络连接模块：负责创建 socket 并连接到 TFTP 服务器。

文件传输模块：负责文件的上传和下载。

错误处理模块：负责处理网络和文件操作中的错误。

日志记录模块：负责记录文件传输的日志。

配置管理模块：负责读取配置文件并配置网络参数。

用户界面模块：提供用户操作界面，接收用户输入并显示传输状态。

2.1.3 模块之间的接口说明

网络连接模块：

init_network()：初始化网络连接。

connect_server()：连接到 TFTP 服务器。

文件传输模块：

upload_file(char* filename)：上传文件。

download_file(char* remoteFile, char* localFile)：下载文件。

错误处理模块：

handle_network_error(int errorCode)：处理网络错误。

handle_file_error(int errorCode)：处理文件错误。

日志记录模块：

log_transfer_status(char* message)：记录传输状态。

配置管理模块：

read_config_file()：读取配置文件。

configure_network()：配置网络参数。

用户界面模块：

display_user_interface()：显示用户界面。

get_user_input()：获取用户输入。

2.1.4 数据处理流程

TFTP 客户端启动后初始化网络环境，创建 UDP 套接字并配置地址。用户可选择上传或下载操作：上传时，客户端发送写请求(WRQ)，收到确认后将文件分成 512 字节的数据块逐块发送，每块都需等待服务器确认，超时则重传；下载时，客户端发送读请求(RRQ)，然后接收服务器发来的数据块，写入本地文件并发送确认。

整个过程采用非阻塞模式，实时记录传输统计信息，并将重要事件写入日志。系统会对网络错误、文件操作异常等进行处理，确保传输可靠性。当收到小于 512 字节的数据块时，表示传输完成。

2.2 详细设计

2.2.1 核心函数流程

2.2.1.1 上传文件流程

1）用户选择上传文件，并选择文件格式（netascii 或 octet）。

2）客户端发送 WRQ 请求到服务器。

3）服务器返回 ACK 确认。

4）客户端开始发送文件数据，每个数据包包含 512 字节数据。

5）服务器接收数据并返回 ACK 确认。

6）重复步骤 4 和 5，直到文件传输完成。

2.2.1.2 下载文件流程

1）用户选择下载文件，并指定本地文件名。

2）客户端发送 RRQ 请求到服务器。

3）服务器返回 DATA 数据包。

4）客户端接收数据并返回 ACK 确认。

5）重复步骤 3 和 4，直到文件传输完成。

2.2.2 核心数据结构

2.2.2.1 tftpPacket 数据包结构体

```
struct tftpPacket {
    unsigned short cmd;       // 操作码 (2 字节)
    union {
        unsigned short code;      // 传输模式
        unsigned short block;     // 块编号
        char filename[2];         // 文件名
    };
    char data[DATA_SIZE];         // 数据区 (512 字节)
};
```

2.2.2.2 sockaddr_in 结构体

```
//网络地址结构
struct sockaddr_in {
```

```
    short sin_family;           // 地址族
    unsigned short sin_port;    // 端口号
    struct in_addr sin_addr;    // IP 地址
    char sin_zero[8];           // 填充
};
// 全局变量
SOCKET sock;                    // 客户端 socket
sockaddr_in serverAddr;         // 服务器地址
sockaddr_in clientAddr;         // 客户端地址
```

2.2.2.3 传输控制参数

```
// 传输控制常量
#define DATA_SIZE 512           // 数据块大小
#define PKT_MAX_RXMT 3          // 最大重传次数
#define PKT_RCV_TIMEOUT 3000    // 超时时间(ms)
// 传输状态变量
double transByte;               // 传输字节数
double consumeTime;             // 消耗时间
```

2.2.2.4 日志记录结构

```
// 日志相关
FILE* logFp;                    // 日志文件指针
char logBuf[512];               // 日志缓冲区
time_t rawTime;                 // 时间戳
struct tm* info;                // 时间信息
```

2.3 代码实现

2.3.1 上传文件核心逻辑

```
    do {
        memset(sendPacket.data, 0, sizeof(sendPacket.data));
```

```
        sendPacket.block = htons(block);       // 设置块编号
        s_size = fread(sendPacket.data, 1, DATA_SIZE, fp);   // 读取文件
        transByte += s_size;

        for (rxmt = 0; rxmt < PKT_MAX_RXMT; rxmt++) {    // 重传机制
            if (sendto(sock, (char*)&sendPacket, s_size + 4, 0,
                        (struct sockaddr*)&serverAddr, addr_len) != SOCKET_ERROR) {
                if (waitForAck(block)) goto next_block;   // 等待确认
            }
        }
    } while (s_size == DATA_SIZE);   // 文件未结束则继续
```

设计说明：上传时采用分块传输策略，每块固定大小来达到提高效率的目的；采用停等协议确保可靠传输；集成重传机制处理网络丢包情况

2.3.2 下载文件关键部分

```
for (time_wait_data = 0; time_wait_data < PKT_RCV_TIMEOUT; time_wait_data += 20) {
    r_size = recvfrom(sock, (char*)&rcv_packet, sizeof(tftpPacket), 0,
                        (struct sockaddr*)&serverAddr, (int*)&addr_len);
    if (r_size >= 4 && rcv_packet.cmd == htons(CMD_DATA) &&
        rcv_packet.block == htons(block)) {
        fwrite(rcv_packet.data, 1, r_size - 4, fp);   // 写入文件
        sendAck(block);   // 发送确认
        break;
    }
    Sleep(20);   // 非阻塞等待
}
```

设计说明：使用非阻塞模式提高响应性；实现超时检测机制避免无限等待；采用即时确认策略减少传输延迟

2.3.3 性能统计模块

```
void logTransferStats(const char* operation, const char* filename) {
    double speed = transByte / (1024 * consumeTime);   // 计算传输速率
    double dropRate = epack / (epack + spack) * 100;   // 计算丢包率

    sprintf(logBuf, "%s %s: size=%.2fKB, time=%.3fs, speed=%.2fKB/s, drop=%.2f%%\n",
            asctime(info), operation, transByte/1024, consumeTime, speed, dropRate);
    fwrite(logBuf, 1, strlen(logBuf), logFp);   // 写入日志
}
```

设计说明：实现完整的性能监控系统；提供详细的传输统计信息；支持日志记录便于问题分析

2.3.4 错误处理机制

```
void handleError(int errorType, const char* context) {
    time(&rawTime);
    info = localtime(&rawTime);

    switch(errorType) {
        case TIMEOUT_ERR: retransmit(); break;
        case FILE_ERR: cleanupFile(); break;
        case NETWORK_ERR: resetConnection(); break;
    }
    logError(errorType, context);   // 记录错误信息
}
```

设计说明：采用分层的错误处理；策略实现错误恢复机制；提供详细的错误日志支持

3 测试与分析

3.1 系统测试及结果说明

3.1.1 测试环境

客户端：操作系统：Windows 11

编译环境：Visual Studio 2022

开发语言：C++

服务器：TFTP 服务器软件：Tftpd64

版本：4.64

3.1.2 测试案例

3.1.2.1 正常上传文件

输入服务器和客户端 ip 地址和设定服务器文件存储位置后，选择 option 1，将 client 文件夹的 a.jpg 上传到服务器。

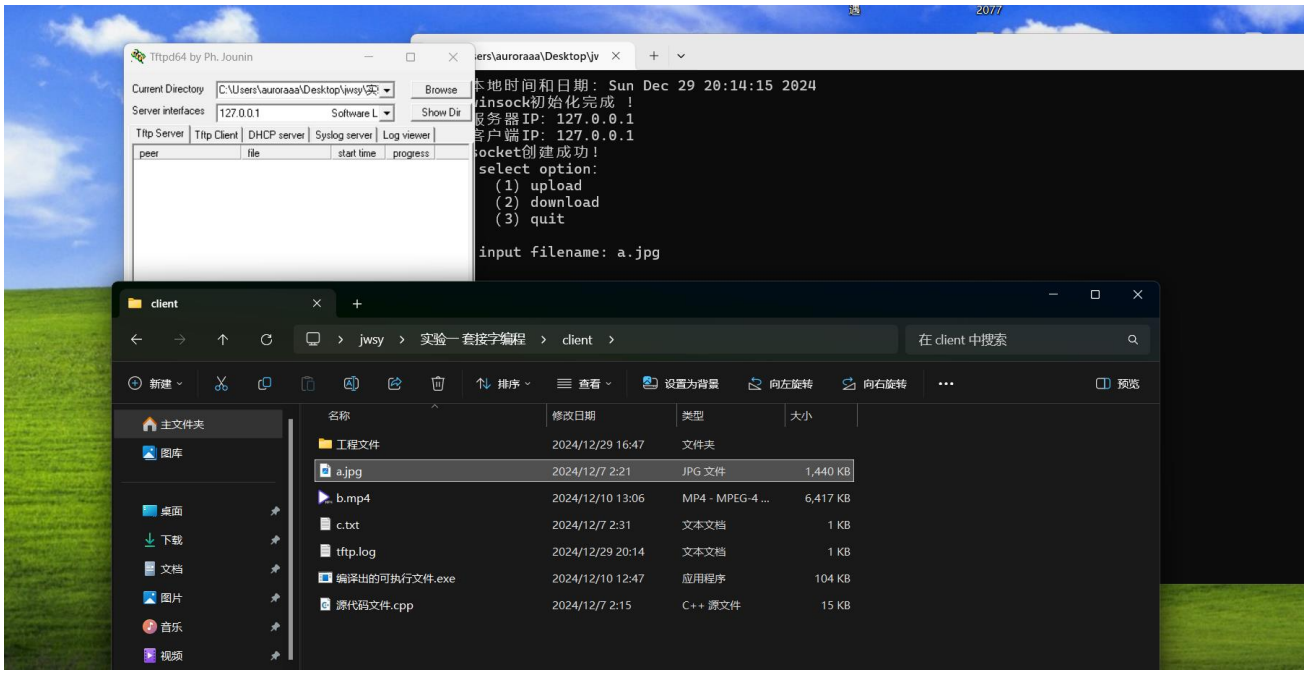


图 1 上传前截图

上传成功，并在服务器路径成功查看 a.jpg 的内容，同时显示上传文件大小，花费时间，上传速度和丢包率。

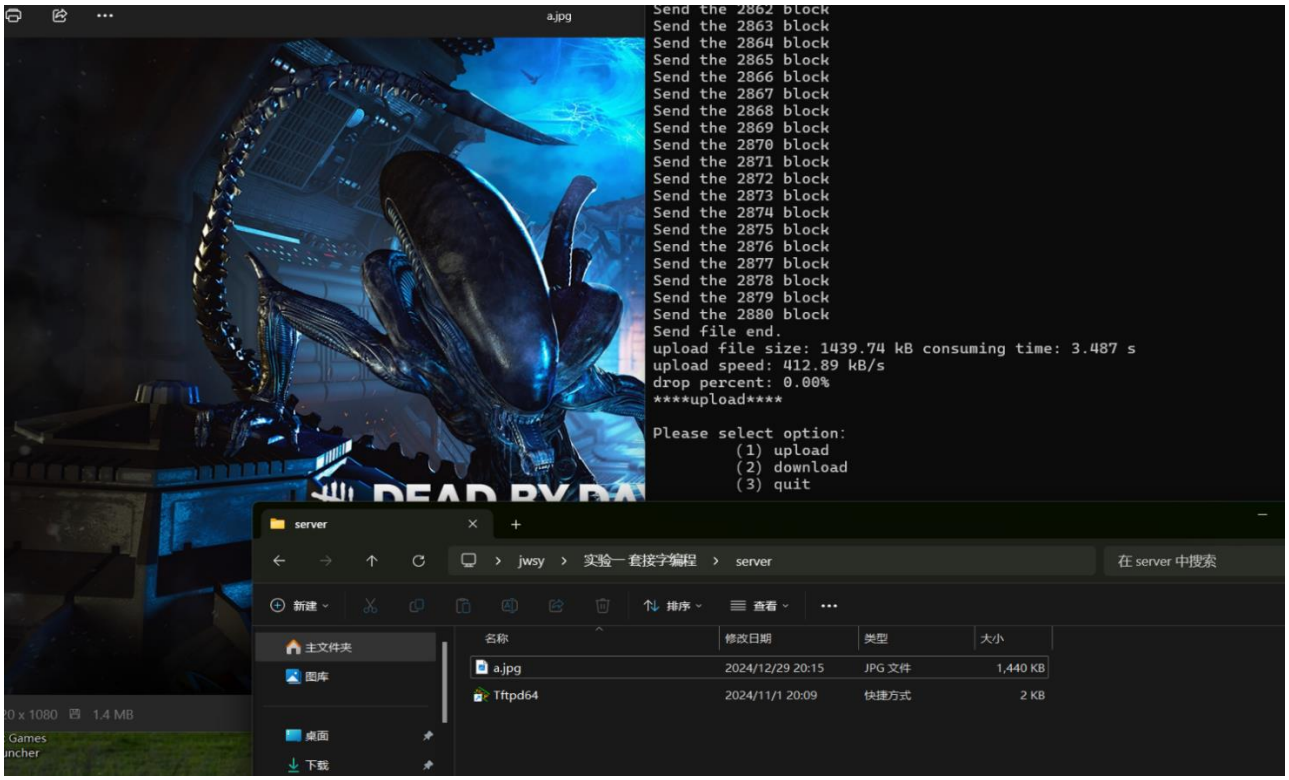


图 2 上传成功截图

3.1.2.2 正常下载文件

选择 option 2，将 server 文件夹的 b.mp4 下载到客户端并命名为 b1.mp4。

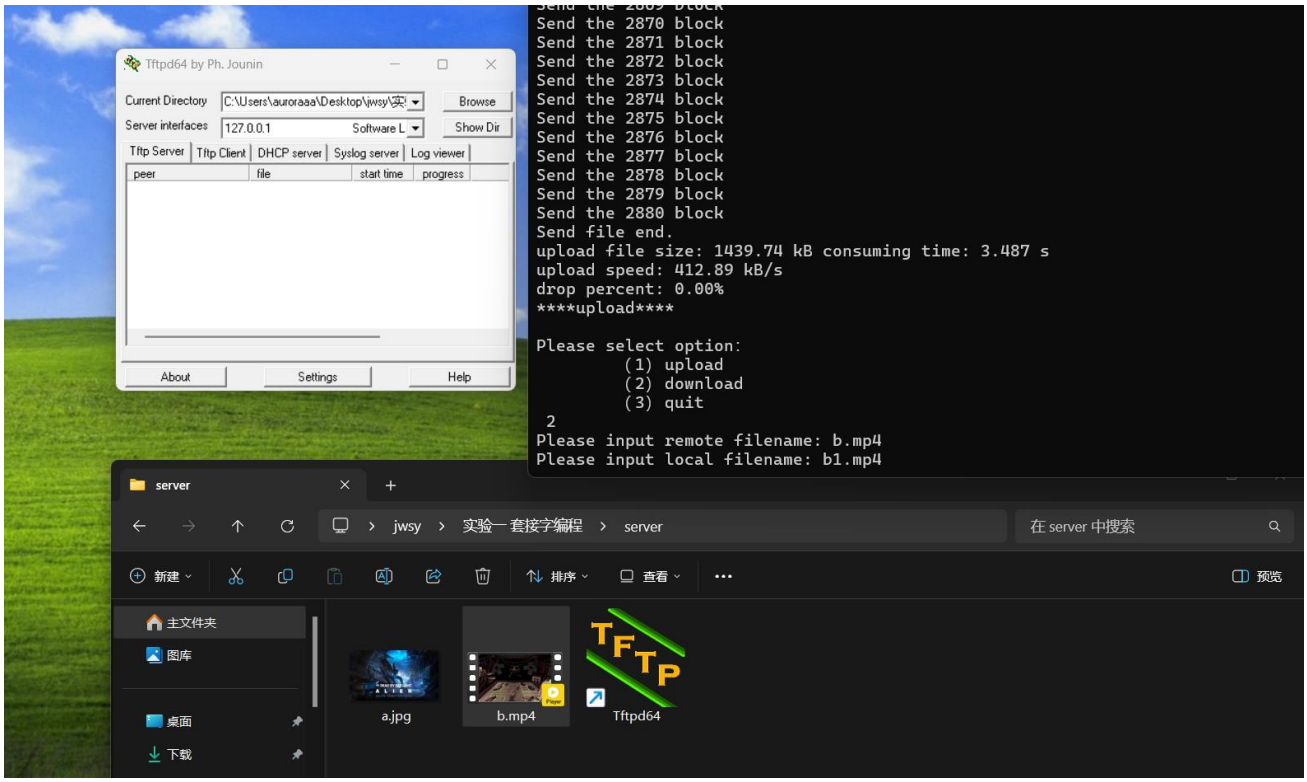


图 3 下载前截图

下载成功，并在服务器路径成功查看 b1.mp4 的内容，同时显示下载文件大小，花费时间，下载速度和丢包率。

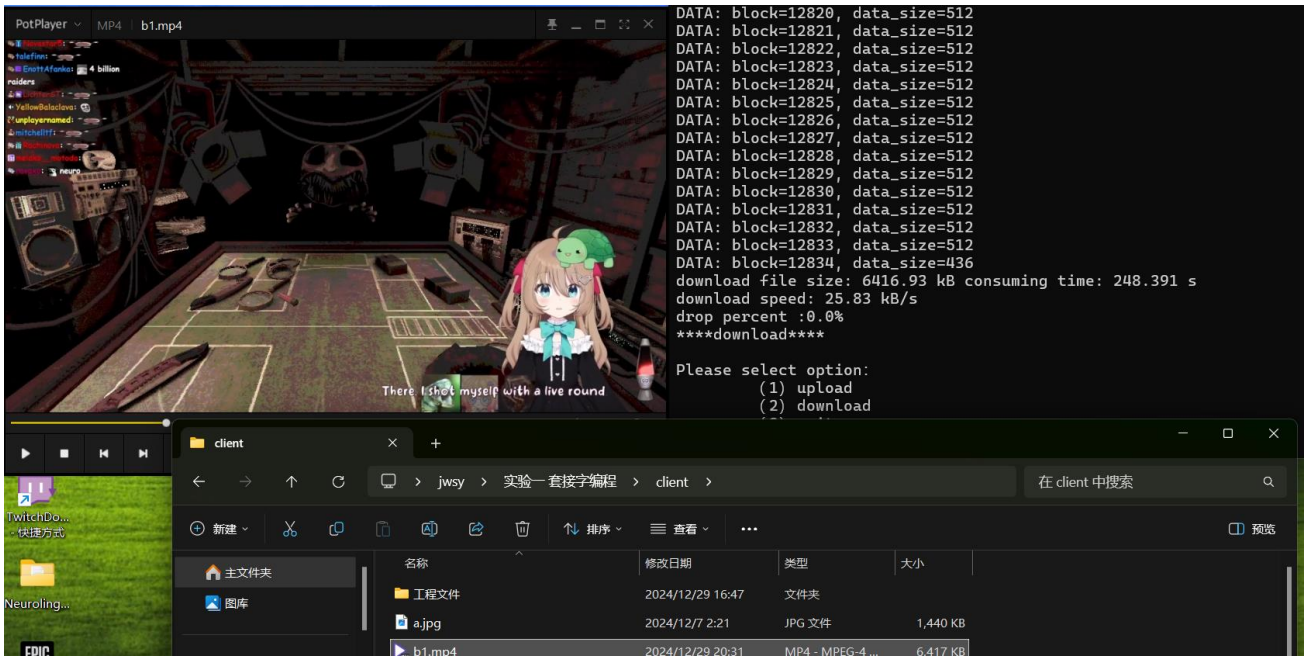


图 4 下载成功截图

3.1.2.3 异常环境测试

用 clumsy 软件模拟丢包和网络延迟的情况，同是上传 a.jpg，上传速度减少，但仍然传输成功，体现了程序实现的传输功能的抗干扰能力。

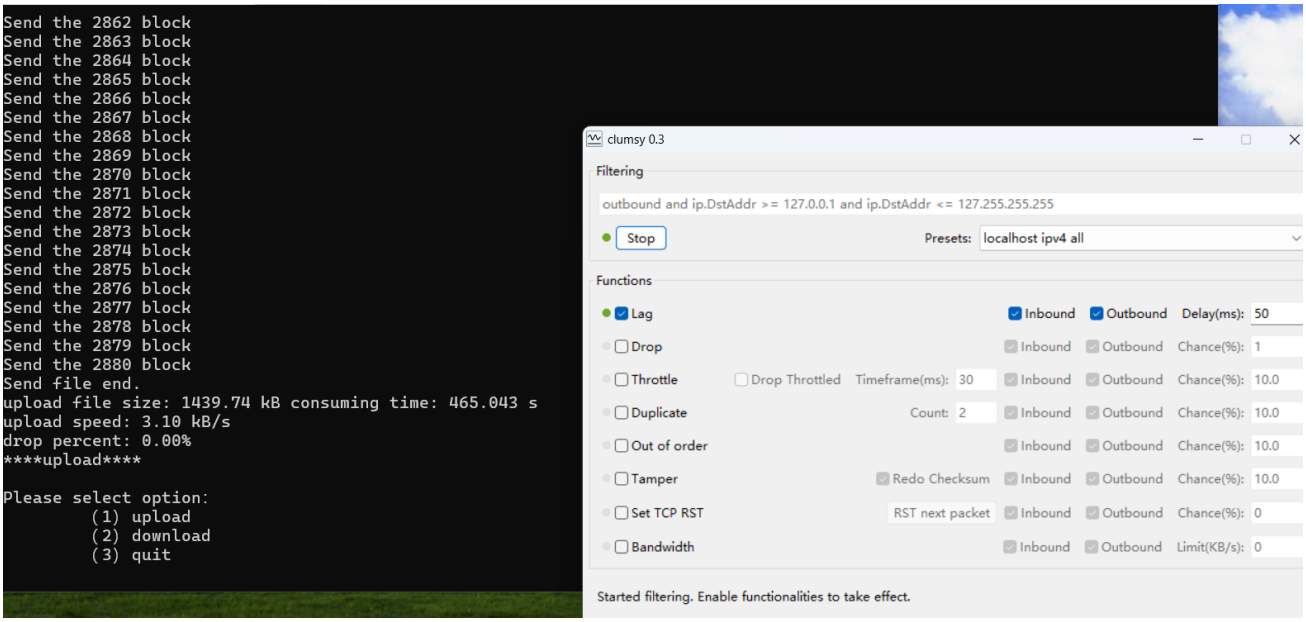


图 5 在 lag=50ms 的情况下上传 a.jpg

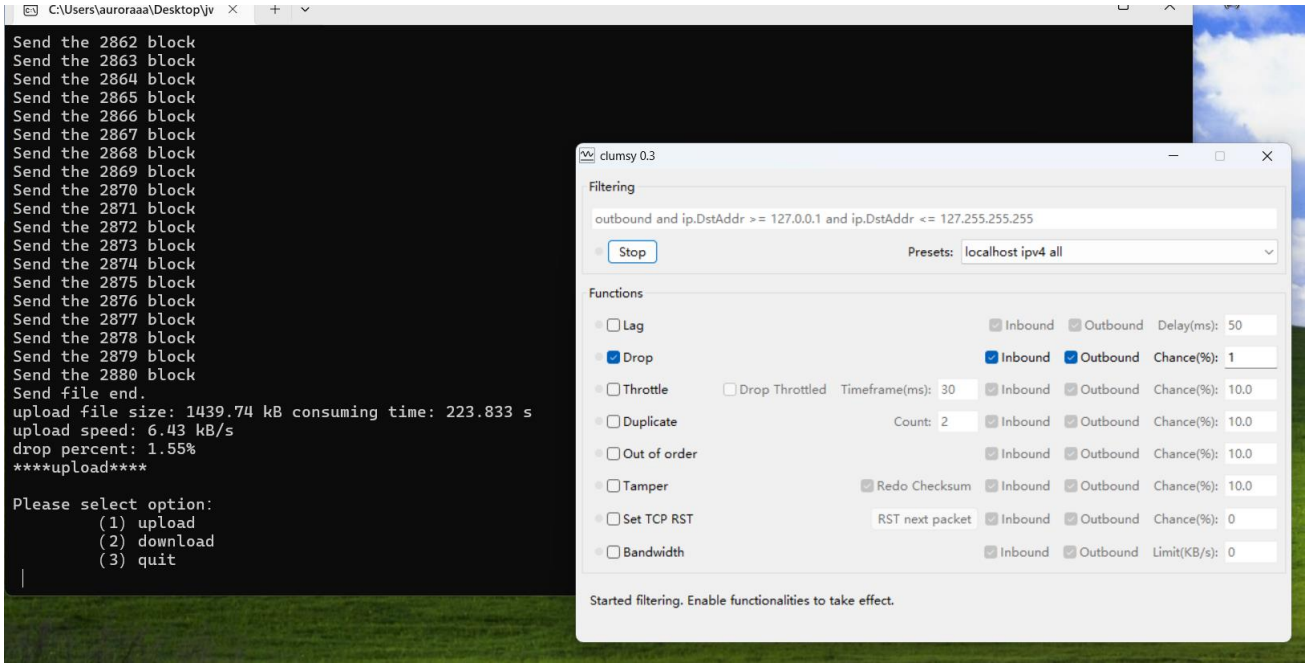


图 6 在丢包率为 1%的情况下上传 a.jpg

3.1.2.4 错误信息显示

在上传文件时选择并不存在的文件，程序会显示错误信息并要求重新尝试上传。

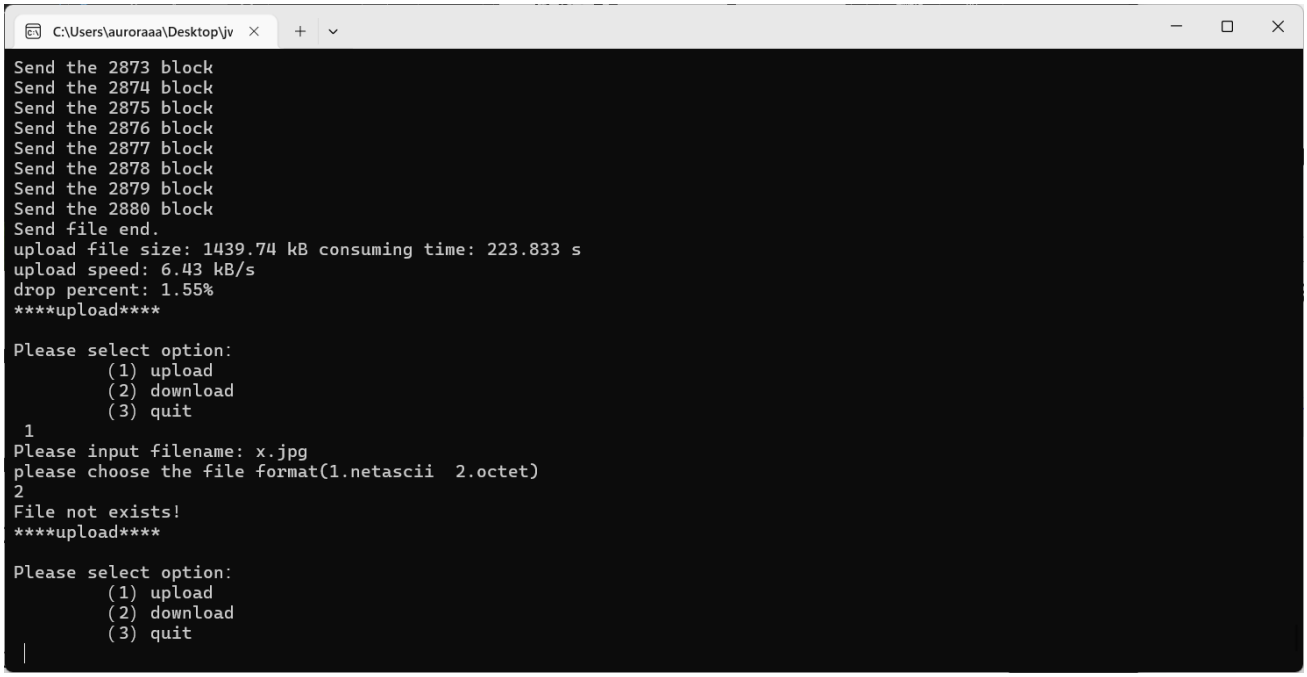


图 7 错误信息显示

3.1.2.5 日志显示

在传输文件后 tftp 服务器会自动记录传输日志。

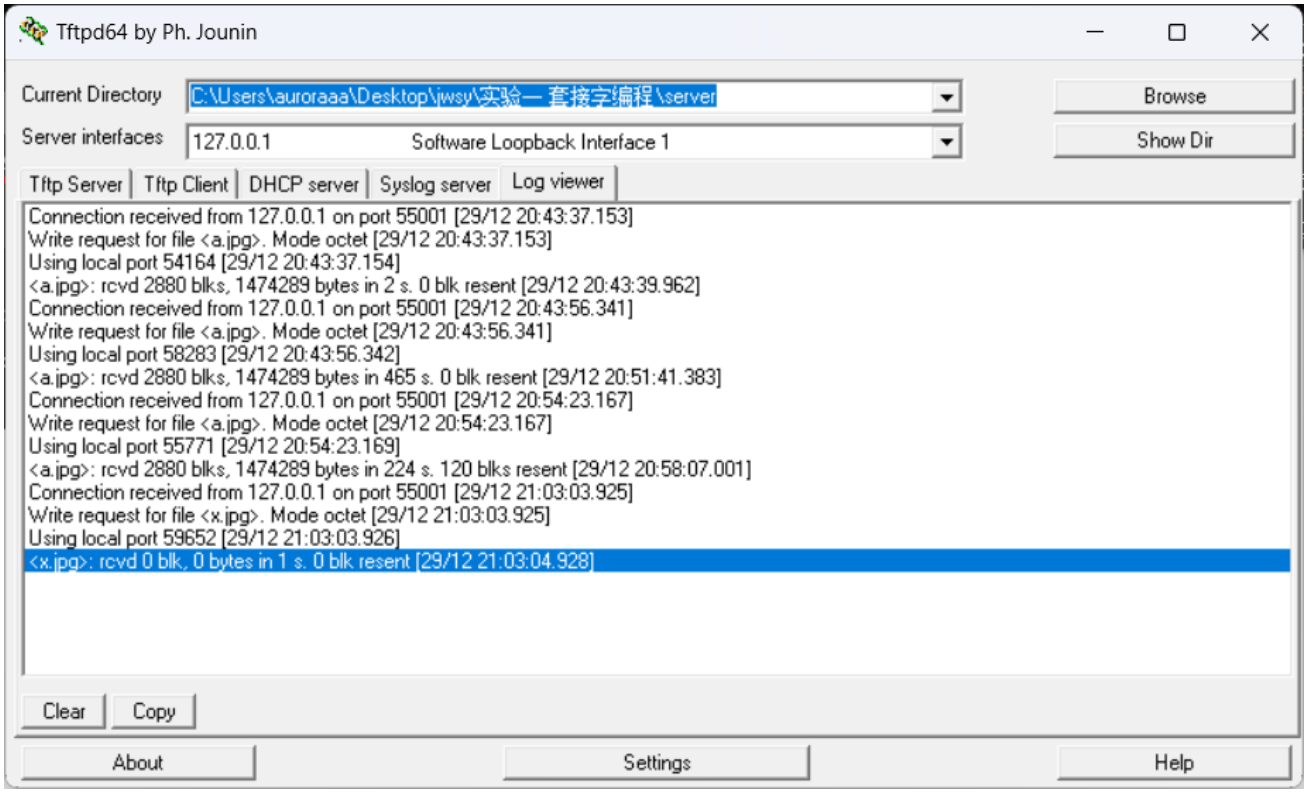


图 8 日志信息显示

3.2 遇到的问题及解决方法

首先是端口占用问题，当其他程序已经占用了指定端口时，可以将端口号设为 0 来实现动态分配，从而避免冲突。

其次，网络传输中经常会出现数据包丢失的情况，为此设计了一个最多重试三次的超时重传机制来保证传输的可靠性。对于大文件传输容易中断的问题，实现了断点续传功能，通过记录传输位置来支持传输中断后的继续传输。

在文件操作方面增加了文件访问权限的检查机制而且提供明确的错误提示信息。

3.3 设计方案存在的不足

当前 TFTP 客户端的设计方案仍存在一些值得改进的地方。

由于采用了停等机制（每发送一个数据包就等待确认），传输效率相对较低。根据课上学到的可靠数据传输机制，可以考虑实现滑动窗口机制，允许同时发送多个数据包，这样能显著提高传输速度。同时，固定的 512 字节数据块大小可能并不适合所有网络环境，可以考虑根据网络状况动态调整块大小。

目前的实现缺少文件校验机制，无法保证传输文件的完整性。我们应该添加校验，在传输完成后验证文件是否正确。此外，对于大文件传输的断点续传功能实现得较为简单，无法保存传输进度，可以增加传输进度保存和恢复的功能。以及未能实现多进程传输，无法同时传输多个文件。

当前的命令行界面较为简陋，缺乏图形界面和传输进度显示。同时，错误提示信息不够友好，可以提供更详细的错误描述和解决建议。

未能实现数据加密和用户认证机制，没有做到用户文件管理，这在实际应用中可能存在安全隐患和可操作性的问题。

4 实验总结

4.1 实验感想

4.1.1 Socket 编程实验感想

编写这个 TFTP 客户端让我对网络编程有了更深的认识。最初觉得文件传输很简单，但实际开发中遇到了很多问题。比如数据包老是莫名丢失，调试了很久才发现是超时重传机制没设计好；还有一次测试大文件传输时，程序总是中途断开，后来才知道要处理网络异常情况。

跨平台测试时，文件传过去就是乱码，原来是忘记考虑字节序的问题。这些问题一个个解决下来，让我明白了网络编程不仅要考虑基本功能，更要注意各种异常情况的处理。

虽然程序最后能正常运行了，但还有很多可以改进的地方，比如传输速度还不够快，用户界面也比较简陋。不过通过这次实验，我学到了很多实用的编程技巧，这些经验对我来说很宝贵。

4.1.2 局域网组网实验感想

这次 GNS3 组网实验让我对网络配置有了直观认识。刚开始时总是连不通，排查后发现是 IP 地址和子网掩码配错了；路由表也折腾了好久，开始没理解静态路由和动态路由的区别，后来通过反复测试才搞明白。

最麻烦的是 VLAN 配置，一开始完全不知道 trunk 和 access 端口怎么设置，导致不同 VLAN 之间老是通不了。还有 ACL 访问控制列表的配置也很容易出错，好几次都把自己给锁在外面了。

虽然遇到不少问题，但每解决一个问题都特别有成就感。通过这次实验，我终于理解了课本上那些网络概念在实际中是怎么用的。不过还是觉得命令行配置挺麻烦的，要是有图形界面就好了。

4.1.3 综合组网实验感想

综合组网实验则更加接近实际生活的运用，在与基础实验相同的配置 pc 的 ip 地址，配置交换机和端口模式和 vlan，以及路由器的路由表之后，我遇到了服务器经常断连和内存不足的情况，而导致交换机和路由器反复重启的问题，挫败实验进程。

在能够实现 pc 间的访问控制后，ftp 服务的 nano 功能花费了我很多时间熟悉，启发我进

一步学习 shell 操作的知识。http 服务中总是无法在浏览器打开自己设定的页面，最后通过修改 http 的 nginx 配置实现的，这让我进一步对学习过程有更深的理解。

总之组网实验我遇到了大量问题，但是都通过网上查询，自我摸索和询问同学中解决了，这种学习成长的过程让我受益匪浅。

4.2 意见和建议

在本次计网实验的套接字编程实验和 gns3 组网实验我遇到了大量的新知识并难以短时间内解决消化，希望在之后的课程中可以加强对这些知识的指导。

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