

华中科技大学
网络空间安全学院

《计算机通信与网络》实验报告

姓 名 _____

班 级 _____

学 号 _____

联系方式 _____

分 数 _____

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	4
3.3 设计方案存在的不足	5
四、 实验总结	1
4.1 实验感想	1
4.2 意见和建议	2

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

该程序使用 Qt 进行人机交互图形界面开发，在 Qt 使用 Qt Widgets Application 创建一个 Qmainwindow 的类，然后在 vs2022 用拓展 qt 插件打开.pro 文件，其中包含“mainwindow.h”头文件，“mainwindow.cpp”和“main.cpp”源文件以及“mainwindow.ui”界面文件等。

接下来对文件的内容和功能分别进行介绍：

mainwindow.h 在 Qt 框架中定义了一个继承自 QMainWindow 的名为 MainWindow 的类，它承担了整个应用程序主窗口界面的逻辑和功能。其主要组成部分为预处理器指令、Qt 库相关的包含、标准 C++/C 库的包含、Windows 套接字的包含、MainWindow 类的定义、TFTP 数据包结构的定义、常量定义和声明语句等。主窗口通常作为应用程序的主要接口，展示并且让用户与应用程序互动。

mainwindow.h 作为搭建用户界面和实现主窗口功能的基础框架。而其对应的源文件 mainwindow.cpp 则负责具体实现该文件中声明的各个函数和类的行为。其包含主要操作函数，并且实现 TFTP 协议中的文件上传(WRQ)和下载(RRQ)功能，上传下载时可以选择不同模式，支持超时重传并且可以记录日志文件等功能。接下来的内容也将以其为主体展开，对该源文件的代码内容进行更加详细地解释以及对编写的思路进行更加深入地阐述。

main.cpp 负责程序的启动和设置初期的应用程序环境，创建主窗口并进入 Qt 的事件驱动模型中，直到用户关闭主窗口或者程序以其他方式终止。

mainwindow.ui 是 Qt ui 界面文件，用来进行界面设计，可以直接在 qtcreeator 进行设计也可以通过记事本进行文本编辑，结合上学期计算机基础课程所学习的 html 设计的知识可以完成界面的设计。

2.1.1 模块框图

界面主要有用户操作模块和信息显示界面，主要模块包括初始化模块，输入读取模块，上传模块，下载模块，日志记录模块。工作大致流程如下图所示：

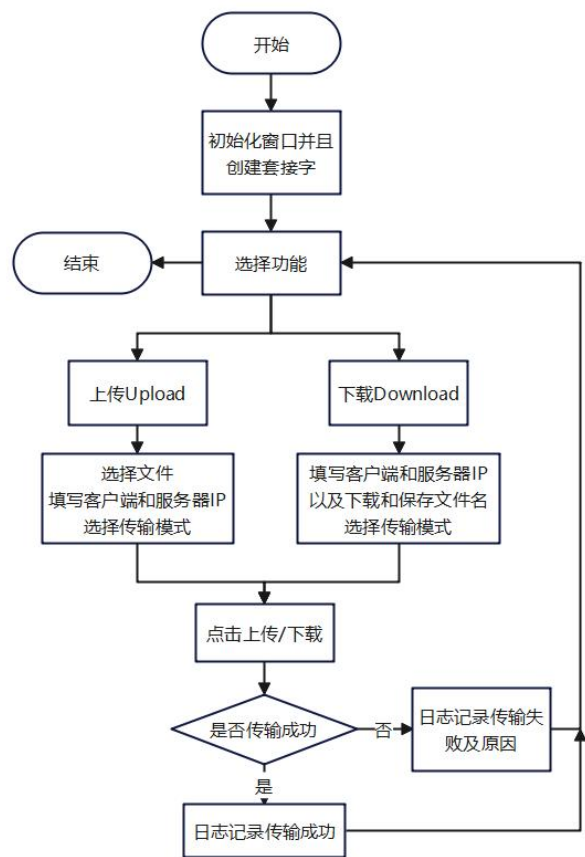


图 2-1 模块框图

2.1.2 模块功能说明

各个模块都有其特殊的功能，共同作用才能完成相关操作。其功能如表 2-1 所示。

表 2-1 模块功能简要说明

模块名称	主要功能
初始化模块	创建套接字，为其他模块的实现提供初始条件
输入读取模块	读取用户输入，选择上传或下载的文件，以及传输方式
上传模块	能够将文件上传到 TFTP 服务器
下载模块	能够从 TFTP 服务器下载指定文件
日志记录模块	记录时间、用户操作、传输成功，传输失败，超时重传等行为

2.1.3 模块之间的接口说明

(1) 初始化模块 (MainWindow 构造函数和 initUI 方法):

接口函数:

MainWindow::MainWindow(QWidget* parent = nullptr): 主窗口的构造函数，负责初始化

UI 界面和成员变量。

`void MainWindow::initUI()`: 负责设置初始布局和组件的默认值。

接口数据结构:

`Ui::MainWindow* ui`: 自动生成的 UI 类, 包含所有界面组件的实例。

(2) 输入读取模块:

接口函数:

`void MainWindow::on_PathChoose_pressed()`: 用于处理 ‘路径选择’ 按钮按下事件。

`void MainWindow::on_uploadButton_pressed()`: 用于处理 ‘上传’ 按钮按下事件, 启动上传流程。

`void MainWindow::on_downloadButton_pressed()`: 用于处理 ‘下载’ 按钮按下事件, 启动下载流程。

接口数据结构 (用户输入数据并通过 UI 组件获取):

`QLineEdit *PathShow`: 显示用户选择的文件路径。

`QLineEdit *uploadLocalIP`: 用于输入上传操作中本地 IP 的文本输入框。

`QLineEdit *uploadServerIP`: 用于输入上传操作中服务器 IP 的文本输入框。

`QLineEdit *downloadLocalIP`: 用于输入下载操作中本地 IP 的文本输入框。

`QLineEdit *downloadServerIP`: 用于输入下载操作中服务器 IP 的文本输入框。

`QComboBox *uploadMode`: 上传文件的传输模式选择框。

`QComboBox *downloadMode`: 下载文件的传输模式选择框。

(3) 日志记录模块:

接口函数:

`sprintf(logBuf, “日志内容”, 对应输入内容)` 以及 `fwrite(logBuf, 1, strlen(logBuf), logPointer)`:

负责将事件或者操作日志记录到日志文件中。

接口数据结构:

`FILE* logPointer`: 用于操作日志文件的文件指针。

`char logBuf[MAX_DATA_SIZE]`: 用于格式化日志信息的缓冲区。

(4) 上传/下载模块:

接口函数:

`on_uploadButton_pressed` 和 `on_downloadButton_pressed` 函数充当了用户操作模块与上

传/下载模块的接口，包括设置网络通信和文件传输以及处理网络通信事件，如接收 ACK 或数据包等，同时也调用日志记录模块的方法以写入操作日志。

接口数据结构：

tftpData: 上传/下载数据包结构

2.1.4 数据处理流程

初始化主界面，创建标签页，布局界面元素。等待用户输入和选择操作（上传或下载）。上传界面需选择文件并输入 IP，下载界面需输入文件名和 IP，在用户触发上传或下载按钮后，验证输入字段。根据用户选择的传输模式，以正确的方式打开文件（文本模式或二进制模式）。建立网络通信，发送请求给服务器（WRQ 或 RRQ），并捕捉服务器的应答。通过非阻塞 I/O 进行文件的发送或接收。收到数据后，进行必要的 ACK 发送和接收，处理可能的错误。在操作完成时，将文件下载的吞吐量，下载时间，丢包率等信息显示到 UI，及结果信息写入日志文件。在发生错误时，显示错误信息，可能的重新尝试，或终止操作。

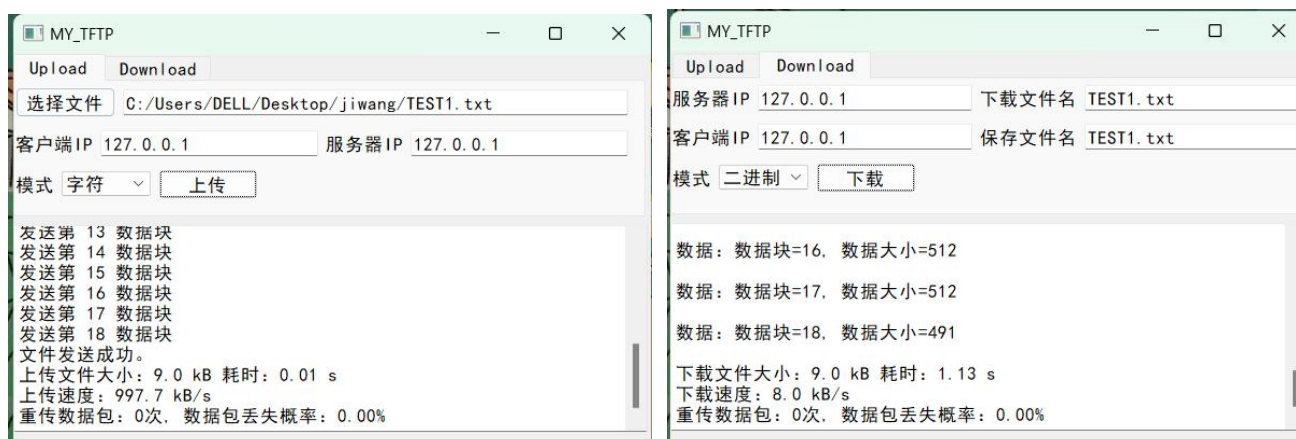


图 2-2 上传和下载的人机交互图形界面

2.2 详细设计

首先定义相关参数，如服务器和客户端的 IP 地址结构、客户端的 socket 描述符、非阻塞模式的选项、日志文件指针、传输字节和耗时统计等，然后用主窗口构造函数和析构函数，并初始化 UI 用户界面，选择文件路径，最后是核心函数——上传与下载函数，两个函数均利用套接字与服务器连接。接下来对上传下载函数流程和数据结构进行更加详细地说明。

2.2.1 上传函数流程

函数 on_uploadButton_pressed() 是 MainWindow 类中的一个槽函数，主要设计用于处理用户点击上传按钮时的逻辑。这个函数主要执行拟定的上传协议功能，能够将文件上传到 TFTP 服务器，以下是详细设计的流程：

首先使用 `ui->output->clear();` 清空用于显示信息的 `QTextBrowser` 组件，通过 `ui` 中的组件获取用户输入的文件路径、服务器 IP 地址、以及客户端 IP 地址，从完整的文件路径提取文件名并打开或创建日志文件。接着初始化服务器和客户端的 `sockaddr_in` 结构体，创建 UDP socket 并设置为非阻塞模式，将 `socket` 绑定到客户端的地址信息上。然后根据用户选择的传输模式，准备一个写请求(WRQ)数据包，并发送到服务器，循环等待服务器的确认(ACK)。成功接收到 ACK 后，继续上传流程；如果接收超时，则记录错误日志、输出错误信息并结束上传流程。根据用户选择的传输模式，以正确的模式打开文件。循环读取文件内容，分多个数据包上传，并等待服务器的 ACK。文件上传完毕后，计算和显示上传文件的大小、耗时和速度，记录到日志文件，并输出成功信息。最后关闭文件和日志文件指针。

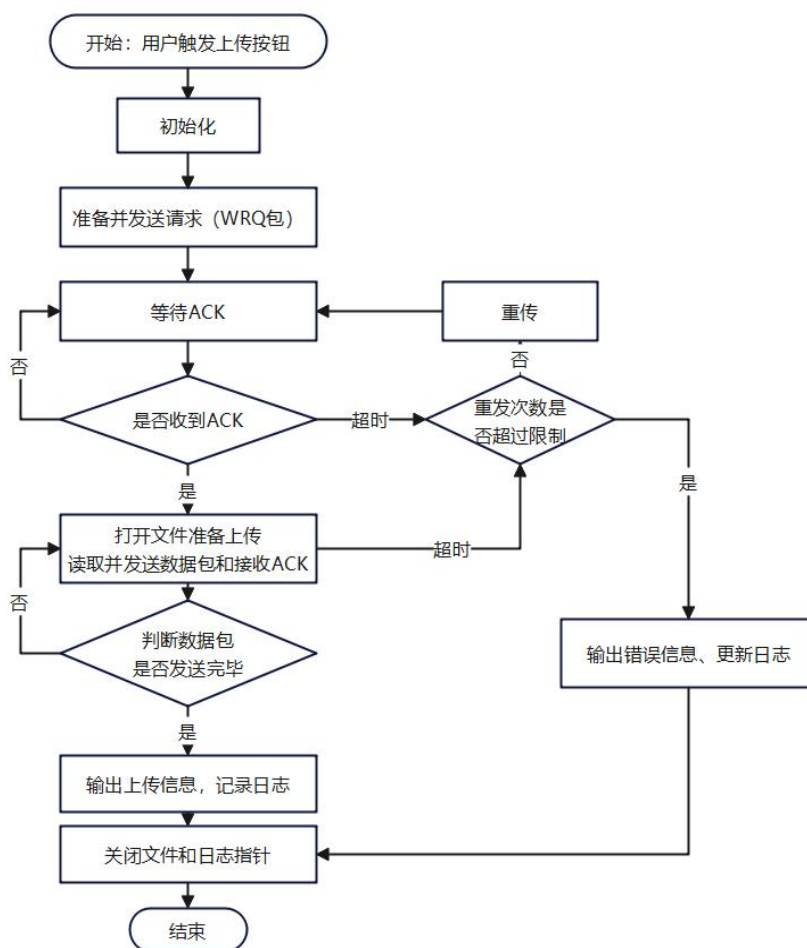


图 2-3 上传函数流程图

2.2.2 下载函数流程

函数 `on_downloadButton_pressed()` 是在 Qt 应用程序中 `MainWindow` 类的一个成员函数，用于处理当用户点击界面上的下载按钮时所触发的一系列操作。这个函数主要执行拟定的下载协议功能，为用户从 TFTP 服务器下载文件，以下是详细设计的流程：

先清空之前可能存在的下载信息输出，为显示新的下载信息做准备。获取用户在界面上输入的服务器文件名（要下载的文件）、本地文件名（文件下载后保存的位置）、服务器 IP 地址以及客户端 IP 地址。打开日志文件 `tftp.log` 以记录下载过程中的相关事件。设置服务器地址，包括 IP 地址和 TFTP 标准端口号（69）。为客户端设置动态分配的端口和 IP 地址。创建 UDP socket 并设置为非阻塞模式，以便进行异步网络通信。把创建的 socket 和客户端 IP:端口绑定在一起。向服务器发送文件读取请求（RRQ），请求下载文件。打开（或创建）一个本地文件，将下载的数据写入该文件。使用 `recvfrom` 函数循环接收服务器发送的数据块，并对每个数据块发送确认响应（ACK）。下载完成后计算下载耗时，在信息输出区显示下载文件的大小、耗时和下载速度，如果中途发生超时或其他错误，记录错误，并提前结束下载。下载完成或发生错误时，更新日志文件，并关闭文件指针。

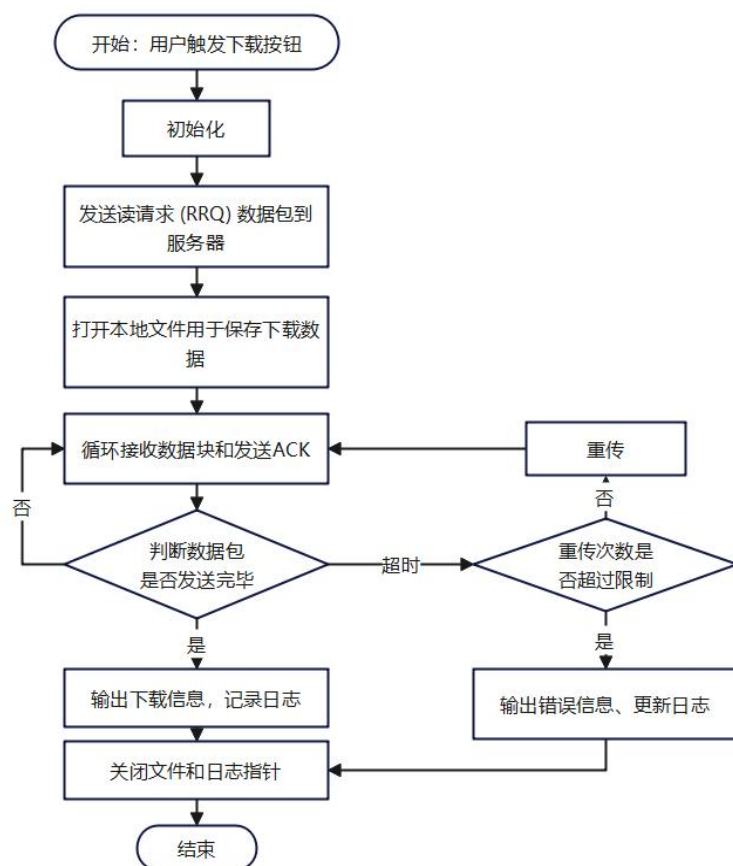


图 2-4 下载函数流程图

2.2.3 核心数据结构

(1) UI 组件：

QTextBrowser *output: 显示执行过程和结果信息的文本框。

QLineEdit *PathShow: 显示选中文件的路径。

(2) 网络通讯结构（全局变量）：

sockaddr_in serverAddress: 存储服务器的地址和端口信息。

sockaddr_in clientAddress: 存储客户端的地址和端口信息。

SOCKET sock: 客户端的 socket 描述符。

sockaddr_in sender: 存储服务器发送数据时的地址信息。

(3) 日志记录（全局/局部变量）：

FILE* logPointer: 指向日志文件的文件指针。

char logBuf[MAX_DATA_SIZE]: 存储日志信息的缓冲区。

FILE* fp: 指向上传/下载文件的文件指针。

(4) TFTP 数据包结构 (tftpData):

```
struct tftpData {  
    unsigned short code;    // 操作码  
    union {  
        unsigned short block; // 数据块编号  
        char filename[2];     // WRQ/RRQ 时用来存储文件名和传输模式  
    };  
    char data[DATA_SIZE];    // 存储文件数据  
};
```

(5) 计时变量：

clock_t start, end: 开始和结束时间，用于计算下载耗时。

(6) 传输统计和控制变量：

int timeWaitingForData, receiveSize, chooseMode, resent: 用于控制接收等待、重传次数、选择的传输模式等。

2.3 代码实现

2.3.1 Ui 交互界面实现

(1) 用户操作模块：

上传/下载选择：提供两个标签页(QTabWidget)：一个用于上传(Upload)，另一个用于下载(Download)。每个标签页都包含必要的输入字段和按钮，供用户选择上传或下载操作。

IP 地址输入：在 Upload 和 Download 标签页中，提供 QLineEdit 组件允许用户输入或粘贴

服务器和本地 IP 地址。

文件选择：提供一个按钮(QPushButton)，用户点击后，弹出一个文件选择对话框(QFileDialog)，用于选择需要上传或下载的文件。用户选择文件后，所选文件路径显示在一个文本框(QLineEdit)中。

传输模式选择：提供一个下拉菜单(QComboBox)，让用户选择 Netascii 或 Octet 传输模式。

开始操作：启动上传或下载操作的按钮，用户点击该按钮(QPushButton)，程序根据用户的输入来初始化和执行相应的上传或下载操作。传输操作使用非阻塞模式，不干扰界面的响应。

非阻塞网络通信：使用异步网络 API(QSocketNotifier, QUdpSocket 或 Winsock 的 select)来监听和发送网络数据，确保 UI 在网络操作时保持响应。

(2) 信息显示模块：

操作结果显示：一个文本浏览区域(QTextBrowser)显示程序的实时状态，包括操作成功或失败的指示。上传或下载过程中的每个步骤，如连接建立、数据包传输等，都会实时更新到该区域。

详细信息写入：下载或上传操作完成后，会在信息显示区域中显示操作总结，包括传输的文件大小、耗时和数据吞吐量。计算并显示上传或下载的丢包率。相关信息同时也写入日志文件中，便于记录和后续分析。

日志文件管理：创建一个日志文件，用于记录每次上传或下载的结果和详细信息。使用标准文件写入操作，记录操作的开始、进行和结果。

2.3.2 套接字初始化

这段代码设置了 TFTP 客户端与服务器间通信所需的网络地址，并创建了一个用于 UDP 通信的客户端套接字。设置服务器地址 serverAddress 和客户端地址 clientAddress 使用 IPv4 地址族 (AF_INET)，服务器端口号被设置为 TFTP 的默认端口 69，使用 htons()函数确保端口号以网络字节序存储，客户端的端口号设置为 0，意味着操作系统将自动分配一个端口号，两者 IP 地址由字符串形式转化为网络字节序的整数并进行存储。通过 socket()函数创建一个 UDP 套接字，该套接字支持 IPv4 地址族，数据报服务为 SOCK_DGRAM。使用 ioctlsocket()，将 sock 套接字设置为非阻塞模式。&Opt 是一个配置选项，使得 recvfrom()等函数在没有数据时立刻返回，不会挂起程序执行。再检查套接字是否创建成功，如果 sock 等于 INVALID_SOCKET,表示套接字创建失败，显示错误信息并返回。否则，显示套接字创建成功的信息。然后绑定套接字到客户端地址，使用 bind()函数将之前创建好的 UDP 套接字与客户端的地址信息

clientAddress 绑定。这确保了客户端从特定的 IP 地址和端口发送和接收数据。

通过这段代码，程序设置好了网络通信的基础环境，创建了用于后续发送和接收数据的 UDP 套接字，并准备好进行 TFTP 数据传输。

```
// 设置服务器地址并使用指定的IP地址和TFTP默认端口69
serverAddress.sin_family = AF_INET;
serverAddress.sin_port = htons(69);
serverAddress.sin_addr.S_un.S_addr = inet_addr(serverIP);
// 设置客户端地址，并使用指定的IP地址和动态分配的端口
clientAddress.sin_family = AF_INET;
clientAddress.sin_port = htons(0);
clientAddress.sin_addr.S_un.S_addr = inet_addr(clientIP);

sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP); // 创建客户端的UDP socket
ioctlsocket(sock, FIONBIO, &opt); // 设置socket为非阻塞模式
if (sock == INVALID_SOCKET) // 创建失败
{
    ui->output->append("客户端socket创建失败！");
    fclose(logPointer);
    return;
}
ui->output->append("客户端socket创建成功！");
// 将创建的socket绑定到客户端地址
bind(sock, (LPSOCKADDR)&clientAddress, sizeof(clientAddress));
```

图 2-5 套接字创建初始化代码段

2.3.3 上传文件发送 WRQ（下载文件发送 RRQ 同理）

这段代码设置了 TFTP 写请求（WRQ）数据包的内容，并将其发送到了服务器。在发送数据包之前，代码首先确定了文件名和所需的传输模式。通过用户界面上传模式选择器 uploadMode 获取用户所选传输模式的索引，根据索引使用 sprintf 函数将文件名和传输模式串联起来，并以空字符'\0'结束，以构造出 TFTP 协议中规定的特定格式；这里'netascii'对应文本模式传输而'octet'对应二进制模式传输。然后使用 sendto 函数将格式化好的 WRQ 数据包发送给服务器，sendto 函数指定了数据包的大小和目的地址，地址来自之前定义好的服务器 socket 地址结构体 serverAddress。数据包发送后，程序会进入等待状态，循环检查是否收到了服务器的 ACK 确认响应，从而开始文件传输；如果在指定超时时间 PKT_RCV_TIMEOUT 内未收到响应，则记录错误信息并停止进程。

```
tftpData sendPacket, recievePacket; // 向服务器发送写请求(WRQ)的包
// 构造用于发送的WRQ包
sendPacket.code = htons(CMD_WRQ);
chooseMode = ui->uploadMode->currentIndex(); // 写入文件名和传输模式
sprintf(buf, "chooseMode=%d", chooseMode);
ui->output->append(buf);
if (chooseMode == 0)
    sprintf(sendPacket.filename, "%s%c%s%c", filename, 0, "netascii", 0);
else
    sprintf(sendPacket.filename, "%s%c%s%c", filename, 0, "octet", 0);

sendto(sock, (char*)&sendPacket, sizeof(tftpData), 0, (struct sockaddr*)&serverAddress, lenOfAddress);
```

图 2-6 上传文件发送 WRQ 代码段

2.3.4 上传文件发送数据并等待 ACK

这段代码是 TFTP 客户端上传功能的核心部分，负责将文件的数据块逐一发送给 TFTP 服务器，并等待服务器的数据确认(ACK)。在外部循环中，对于每个文件数据块，最多尝试发起 3 次（由 PKT_MAX_RXMT 控制）发送操作。通过调用 sendto()函数，指定要发送的数据包的内容、大小和目标服务器地址来发送数据包，并在用户界面显示已发送数据块的编号。然后，进入内部循环，等待并使用 recvfrom()函数来接收服务器的 ACK 包，设定最长等待时间为 3 秒（PKT_RCV_TIMEOUT）。收到服务器针对当前数据块的正确 ACK 后，内部循环结束，表明当前数据块发送成功，随即跳到下一个数据块。如果在设定的超时时间内没有收到 ACK，则记录警告信息，并重试发送当前数据块。连续超过 3 次发送尝试失败后，将错误信息记录到日志，显示上传失败的提示，并退出上传过程。

过程的部分伪码如下：

```
1. 定义变量 rxmt 为 0
2. // 尝试最多 PKT_MAX_RXMT 次发送数据包并等待 ACK
3. 当 rxmt 小于 PKT_MAX_RXMT 时：
4.
5.     // 发送数据包到服务器
6.     发送数据包 sendPacket 到地址 sender
7.     输出 "发送第 [block] 数据块"
8.
9.     // 初始化等待ACK 的累计时间
10.    定义变量 time_wait_ack 为 0
11.
12.    // 等待ACK 应答，最多等待PKT_RCV_TIMEOUT
13.    当 time_wait_ack 小于 PKT_RCV_TIMEOUT 时：
14.
15.        // 尝试接收服务器的ACK 包
16.        接收数据到 recievePacket 并记录接收到的数据大小为 receiveSize
17.
18.        // 检查接收到的包是否异常小
19.        如果 receiveSize 大于 0 且小于 4：
20.            输出 "收到的数据包异常：receiveSize=[receiveSize]"
21.
22.        // 检查是否接收到预期的ACK 包
23.        如果 receiveSize 大于或等
于 4 且 recievePacket.code 是 ACK 且 recievePacket.block 与当前块号相同：
24.            跳出接收循环
25.
26.        // 暂停 20 毫秒后再次检查
27.        暂停 20 毫秒
```

```
28.          时间累加 20 毫秒
29.
30.          // 如果在超时时间内收到ACK，跳出发送循环
31.          如果 time_wait_ack 小于 PKT_RCV_TIMEOUT:
32.              跳出发送循环
33.          否则:
34.              // 未收到ACK，打印警告并记录重发
35.              获取当前时间并格式化
36.              输出 "警告: 上传 [filename], 模式: [chooseMode], 未收到 ACK, 尝试
重发"
37.              写入日志
38.              重发计数增加
39.              继续发送循环
40.
41.          // 重发次数增加
42.          rxmt 自增
43.
44.          // 检查是否达到最大重试次数
45.          如果 rxmt 大于或等于 PKT_MAX_RXMT:
46.              输出 "无法从服务器接收确认。"
47.          关闭文件指针 fp
```

每个数据块发送成功后，准备发送后续的文件数据块，直到文件的全部数据被成功上传，整个过程最终计算并展示上传的总耗时、文件大小及传输速度等信息。

而 TFTP 下载功能的大部分代码与上传相似，这边就不再赘述。TFTP 在下载文件时的数据包接收与确认的过程，需要先检查第一个数据包长度是否等于 516 字节，如果是则继续接收后续数据包如果数据包未满足，说明传输完成，退出循环。最后计算下载耗时等信息。

三、实验测试与分析

3.1 系统测试及结果说明

3.1.1 软件环境配置

在同一台机器上进行服务器与客户端之间的连接传输，系统测试环境如下：

处理器：12th Gen Intel(R) Core(TM) i5-1240P 1.70 GHz

系统类型：64 位操作系统，基于 x64 的处理器

Windows 版本：Windows 11 家庭中文版

IDE：Visual Studio 2022, Qt5.12.6, Qt Creator 4.10.2

3.1.2 文件上传测试

首先是在正常情况下上传测试，选择 octet 模式，点击上传，在底部可以查看传输过程所用时间，传输大小，平均上传速度和丢包率等信息。

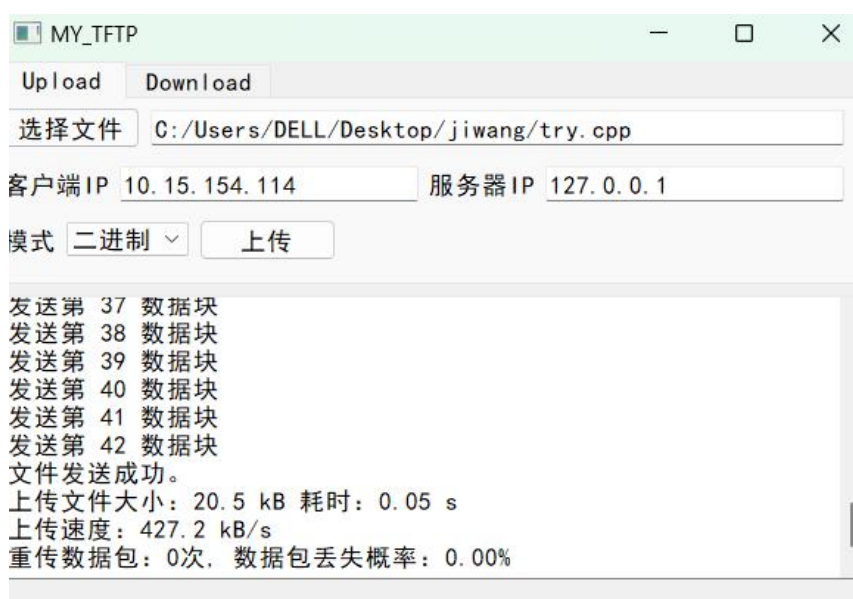


图 3-1 正常情况下上传界面

在开启丢包延迟软件的情况下进行测试，并选择 netascii 模式，点击上传，交互界面功能正常，显示发送成功之后观察服务器所在文件夹，可看到文件已经上传完成。



图 3-2 上传界面



图 3-3 上传成功

打开日志，可以看到日志中保存了上传信息。



图 3-4 上传成功记录

上面正常情况下的测试使用二进制模式上传文件，结果表示可以成功传输。故再次条件下，调高延迟和丢包率，使得文件传输失败，观察显示信息与日志，结果如下图，说明功能正常。

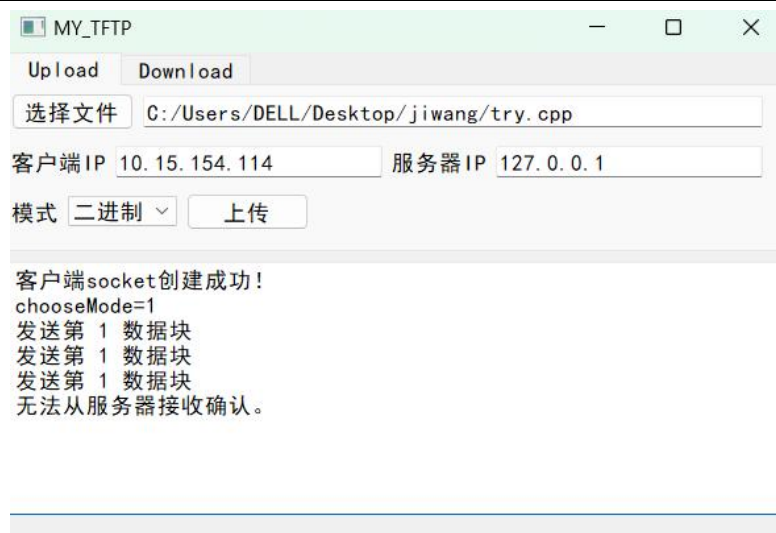


图 3-5 上传失败的信息

```
Wed Dec 20 09:17:50 2023
警告: 上传 try.cpp, 模式: octet, 未收到ACK, 尝试重发
Wed Dec 20 09:17:54 2023
警告: 上传 try.cpp, 模式: octet, 未收到ACK, 尝试重发
Wed Dec 20 09:17:59 2023
警告: 上传 try.cpp, 模式: octet, 未收到ACK, 尝试重发
Wed Dec 20 09:17:59 2023
错误: 上传 try.cpp, 模式: octet, 等待ACK超时
```

图 3-6 日志成功记录

3.1.3 文件下载测试

选择服务器文件夹中的文件作为下载文件，保存文件名包含绝对路径，便可以将下载的文件保存到目的路径之下，并可以改变保存的文件名称。直接在开启丢包延迟软件的情况下进行测试，使用字符模式将上传的文本文件下载到一个文件夹中，结果如图所示。



图 3-7 绝对路径下载





 ui_mainwindow.h	2023/12/19 16:12	C++ Header file	13 KB
 workcp.cpp	2023/12/18 11:10	C++ Source file	1 KB
 EEcp.pdf	2023/12/20 9:44	WPS PDF 文档	34 KB
 t1cp.txt	2023/12/20 9:37	文本文档	0 KB

图 3-8 下载成功

观察日志，可看到下载情况被正确记载，同时记录了错误原因。

```
Wed Dec 20 09:43:46 2023
警告: 下载 EUPL-EN.pdf 保存为 EEcp.pdf, 模式: netascii, 未能接收到数据块 #5, 重发
Wed Dec 20 09:43:50 2023
警告: 下载 EUPL-EN.pdf 保存为 EEcp.pdf, 模式: netascii, 未能接收到数据块 #8, 重发
Wed Dec 20 09:43:54 2023
警告: 下载 EUPL-EN.pdf 保存为 EEcp.pdf, 模式: netascii, 未能接收到数据块 #8, 重发
Wed Dec 20 09:44:00 2023
警告: 下载 EUPL-EN.pdf 保存为 EEcp.pdf, 模式: netascii, 未能接收到数据块 #41, 重发
Wed Dec 20 09:44:04 2023
警告: 下载 EUPL-EN.pdf 保存为 EEcp.pdf, 模式: netascii, 未能接收到数据块 #43, 重发
Wed Dec 20 09:44:13 2023
警告: 下载 EUPL-EN.pdf 保存为 EEcp.pdf, 模式: netascii, 未能接收到数据块 #60, 重发
Wed Dec 20 09:44:17 2023
警告: 下载 EUPL-EN.pdf 保存为 EEcp.pdf, 模式: netascii, 未能接收到数据块 #62, 重发
Wed Dec 20 09:44:21 2023
信息: 下载 EUPL-EN.pdf 保存为 EEcp.pdf, 模式: netascii, 文件大小: 33.5 kB, 耗时: 38.64 s, 下载速度: 0.9 kB/s
```

图 3-9 日志记录成功

经测试，系统各项功能可以正常运行，符合需求，达到了预期的设计目标。

3.2 遇到的问题及解决方法

在本次实验中走了不少弯路也遇到了一些问题，网上的资料很繁杂而且时间跨度大以及有的内容专限于一些固定的版本，在选择的过程中很繁琐很容易被误导，所以信息的正确检索能力也是很重要的。尤其是在调试 qt 人机交互界面的时候多次已经想放弃，但是最后又调整心态坚持了下来。

首先是 socket 编程上手难度较大，虽然老师给了一个范例但是因为其使用了大量的封装好的函数以及一些不太常用的语法，经过在网上查阅资料以及询问老师和同学，花了较长的时间完成对代码的理解，才上手开始自己改写和使用。同时因为机器状态问题，需合理设置等待响应的的时间以及重传次数，让客户端能适应更复杂的链路环境。

然后就是 qt 人机交互界面环境的配置和设计，网上大部分都是 win10 和 vs2019 的配置教程，一开始尝试用 vscode 中途与教程出现了较大偏差无法解决，后面放弃选择了 vs2022，而 qt 的版本也繁杂，最终选择了老版，而组合使用的时候一开始也出现了问题，经过长时间 debug，最后运行出现了 lnk2019 和 lnk2001 的问题无法解决故而放弃，后面尝试从 qt 先创建文件再在 vs 用扩展 qt 工具打开最终成功运行代码。而因为 Qt 默认的编码是 unicode，无法正确显示中

文，所以查阅了很多资料多次尝试最终发现使用`#pragma execution_character_set("utf-8")`可以实现本地字符集与 Unicode 之间的转换，解决了汉语显示乱码等问题。而且 ui 界面很多函数的理解和运用也对编写代码造成了一定难度，通过大范围查找资料进行了解和学习，一些使用频率较高的函数我也进行了注释，增强了代码的可读性。

3.3 设计方案存在的不足

1. 传输效率问题

在出现高延迟和丢包的情况下，目前的设计需要等待 ACK 确认才能传输下一个数据包，这导致传输效率大幅下降。尽管这能保障数据的正确性，但是在网络环境不佳时速度会明显降低。可以考虑采用窗口机制或其他传输优化策略，以提高在复杂网络环境下的数据传输能力。

2. 功能性问题

（1）程序只支持单个文件的上传或下载，缺乏批处理功能。这要求用户在处理多个文件传输时必须手动一一操作，效率低下。实际应用中常需要批量处理文件，提升程序的自动化处理能力是必要的。

（2）设计方案较为简单，仅支持最基本的文件传输功能。对于更多实际需求如并发传输、大文件传输等功能不支持，限制了其在实际应用场景中的使用范围。

3. 安全性问题

（1）方案中忽略了身份认证环节，这在实际应用中可能诱发用户身份盗用等安全问题。增加密钥验证或用户登录的步骤，可以大幅提高安全性，确保只有授权用户才能进行文件的上传和下载。

（2）对于错误的处理不够完善，仅在文件路径为空时进行了验证，而缺乏对路径合法性的判断，同样对于输入的 IP 地址的验证也不足。这表示程序可能需要更加健壮的输入验证机制，以确保操作的准确性和安全性。

4. 用户友好度问题

程序的界面设计在美观性和功能明确性方面也有提升的空间，应考虑提供更为直观和简洁的操作界面，以减少用户使用中的困惑和操作错误。

综上所述，设计方案需要在传输效率的优化、功能性的扩展、安全性的增强以及用户友好度的改善等多个方面进行全面的考量和提升。通过这些细化点的改进，可以让设计方案更加贴合实际应用需求，并提升其在实际场景中的使用价值。

四、 实验总结

4.1 实验感想

4.1.1 Socket 编程实验体会

通过这次的套接字编程实验让我深刻体会到计算机网络编程的实际应用，同时强化了我对传输层协议特别是 UDP 及其非连接性质的理解。通过编写和测试 TFTP 协议的客户端动作，我获得了宝贵的实战经验，并学会了处理网络应用开发中的各种情况，如超时重传、非阻塞 I/O 等。实验过程中，认识到在网络编程中常常需要处理的各种意外情况，比如网络延迟、数据传输中的丢包等，所以需要更加细致地编写代码，并在我的程序中实现了较为健壮的错误处理和异常管理。特别是在实现非阻塞套接字和超时重传机制时，我对编程的每个细节都进行了仔细地思考和调试，确保了上传和下载功能的可靠运行。同时对 Qt 人机画面交互环境的配置和代码的编写有了更深入的了解，也是一次不断探索的过程，提高了自己对不同 IDE 的跨平台使用的适应能力以及对庞杂信息的检索能力。

4.1.2 组网实验体会

通过这次 Cisco Packet Tracer 的组网实验，我深刻理解并实践了网络基础知识的重要性。这个实验不仅加强了我对 IP 协议、网络层和数据链路层协议工作原理的理解，还巩固了我对 IP 地址规划方法、路由协议和交换机配置的掌握。通过为不同实验室和数据中心分配 IP 地址，我学习了如何在一个复杂的网络环境中进行精确的网络规划。此外，实验中 VLAN 的划分原理和访问控制配置方法的实践，让我更清晰地认识到了网络划分的安全意义，以及如何通过配置来实现不同网络层级间的隔离。制定合理的 VLAN 划分，不仅提高了对数据安全的保护，还提升了网络的效率。通过建立 ftp 服务器和配置相关客户端进行文件上传下载，我对实际网络应用服务有了更加实践的了解。而进阶任务中引入的无线网络环境配置和对 DNS、HTTP 服务器的建立与测试，让我对现代网络中常见的无线连接以及域名系统的配置与使用有了真实的体验。

总体而言，这两个实验是一次非常有收获的经历，它不仅加深了我对理论知识的理解也提高了我的编程技能，而且让我对网络通讯有了更深刻的认识，提升了解决实际网络问题的能力，这将大大有助于我将来在网络工程和网络安全方面的职业发展。此外，编写实验报告的过程也提升了我的文档撰写和整理能力，这对于我的学术和职业发展同样重要。

专业课程的学习与实验是为了更好地服务国家网络空间安全的大局，以科技强国为己任，肩负起新时代网络空间安全的使命。本次实验让我更加清晰地认识到，未来不论是从事网络安全的科研工作，还是网络安全的防御与管理工，都需要将这次实验中学习到的知识和技能，转化为维护网络安全的实际能力，为实现网络强国战略、保障人民网络安全福祉、维护国家网络空间主权和安全利益而努力。

4.2 意见和建议

首先非常感谢课题组老师们的辛勤付出和对课程的充分准备，我在这次实验中收获颇丰。特别是xxx老师每一节实验课都在教室流动，方便我们随时提出问题并及时给予帮助和解答，老师们辛苦了！

基于我在本次实验课程中遇到的问题，我有以下一些小提议：

- (1) 套接字编程对于大部分同学入门的难度较大，可能老师本意即培养我们搜集和筛选资料的能力，但是网上的资料过于繁杂，造成很多不必要的时间的浪费。其实资料库里的资料挺丰富的，可以感受到老师们的用心，但是有的资料还是有些笼统，对于可使用性和针对性还可以有一定的优化。
- (2) 组网实验的难度与套接字相差稍大，组网实验做起来还是比较顺利的，可以让学有余力的同学多进行一些其他内容的尝试。

再次感谢老师们的教导和同学们对我的帮助，我得以顺利地完成本次实验。也希望我们这门课程能逐步发展得更加完善，让学弟学妹们能更加有所受益。

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：