**华中科技大学**

**网络空间安全学院**

**《计算机通信与网络》课程设计报告**

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](#_Toc514148436)** [1](#_Toc514148436)

[1.1 实验名称 1](#_Toc1534750222)

[1.2 实验目的 1](#_Toc1918171767)

[1.3 实验环境 1](#_Toc1025734270)

[1.4 实验内容 1](#_Toc773033624)

**[二、 实验过程](#_Toc1552607260)** [2](#_Toc1552607260)

[2.1 系统结构设计 2](#_Toc513937138)

[2.2 详细设计 3](#_Toc920717838)

[2.3 代码实现 5](#_Toc836515570)

**[三、 实验测试与分析](#_Toc1353238939)** [7](#_Toc1353238939)

[3.1 系统测试及结果说明 7](#_Toc104094165)

[3.2 遇到的问题及解决方法 12](#_Toc1647536329)

[3.3 设计方案存在的不足 12](#_Toc914222897)

**[四、 实验总结](#_Toc499897395)** [13](#_Toc499897395)

[4.1 实验感想 13](#_Toc1919238312)

[4.2 意见和建议 13](#_Toc1188521587)

**Socket编程实验**

## **实验概述**

## **实验名称**

Socket编程实验。

## **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。

## **实验环境**

操作系统：Arch Linux 6.5.5-arch1-1

编程语言：C，C++，Python，PyQt6

## **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）；

## **实验过程**

## **系统结构设计**

1. **模块框图**

程序结构自下而上分为四层结构，分别为socket层、tftp协议层和GUI层（如图2-1所示）

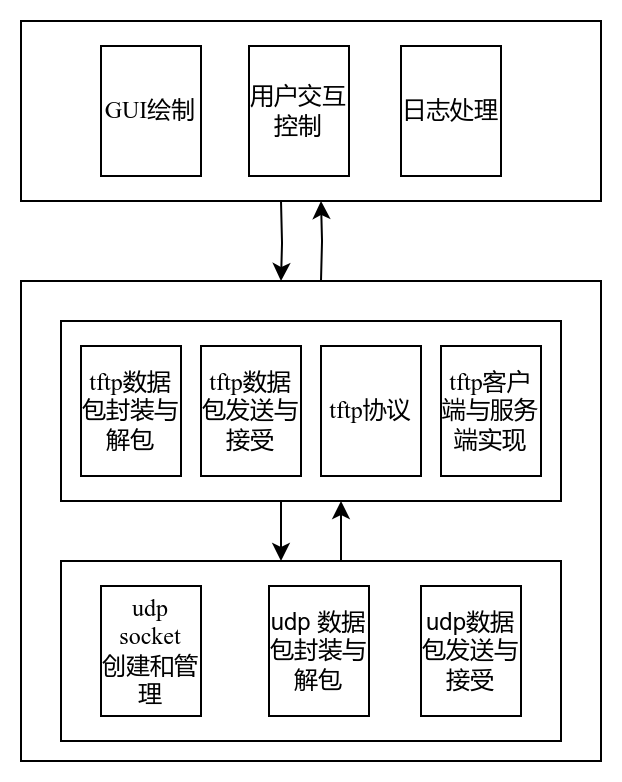


图2-1 系统结构框图

1. **模块功能说明**

程序的socket层模块负责udp socket的创建和管理，管理虚拟通信连接，负责将比特流数据封装为udp数据包以及将udp数据包解包为比特流数据，使用socket套接字发送和接收udp数据包。

tftp协议层涉及与底层socket层进行通信，tftp协议层将数据封装为tftp数据包，将tftp数据包以比特流数据传输给scoket层，以及将socket层接收的比特流数据包解析为tftp数据包，并解析得到数据，tftp协议层实现tftp客户端与服务端的通信协议，客户端协议实现文件的发送与接收，服务端协议实现等待客户端连接、处理客户端请求、文件的发送与接收、多线程并发等。

GUI层实现程序界面的绘制、提供用户交互、以及日志的记录等功能。

socket层与tftp层使用c语言实现，tftp层可直接调用socket层的函数，编译后封装为动态链接库，再使用Python的ctypes模块封装为Python库，GUI层使用Python以及PyQt实现，通过封装的tftp库调用动态链接库中的函数。

1. **模块之间的接口说明**

socket层中主要通信结构体为udp\_socket、udp\_peer，udp\_socket结构体储存socket端口号、绑定的ip地址、ip协议版本号，udp\_peer结构体储存socket通信时的对端信息。模块实现的主要函数有socket的初始化init\_udp\_socket，udp数据包的创建create\_msg，udp数据包的发送与接收send\_msg、recv\_msg。

tftp协议层的数据结构与函数实现较多，主要的数据结构有packet、tftp\_addr、tftp\_log，packet是tftp协议层内的数据包，根据数据包协议可分为rrq\_packet、wrq\_packet、data\_packet、ack\_packet和err\_packet。函数实现有数据包的创建create\_rrq\_packet，tftp数据包与udp数据包的转换packet2msg和msg2packet，数据包的发送与接收send\_packet、recv\_packet，以及tftp层的日志记录logger。

tftp客户端使用tftp协议层实现文件的发送与接收，实现的功能函数有tftp\_get和tftp\_put。

tftp服务端的数据结构体有tftp\_client和tftp\_server，分别用于存储连接的客户端信息和服务端信息，服务端的主要函数实现有初始化客户端连接init\_client，初始化服务器init\_server，文件发送线程send\_thread，文件接收线程recv\_thread，服务器主线程tftp\_server。

## **详细设计**

1. **核心函数流程**

程序的核心层为tftp协议层，主要包括数据的发送与和接收，发送数据时，将数据分割为512字节的数据块，交由send\_data\_block发送，发送后自动转换到接收状态，等待对该数据块的确认，确认号正确进行下一个块的发送，确认号错误或确认超时则进行重发，当错误次数大于设定的重传次数时，错误返回，数据的发送过程如图2-2所示。

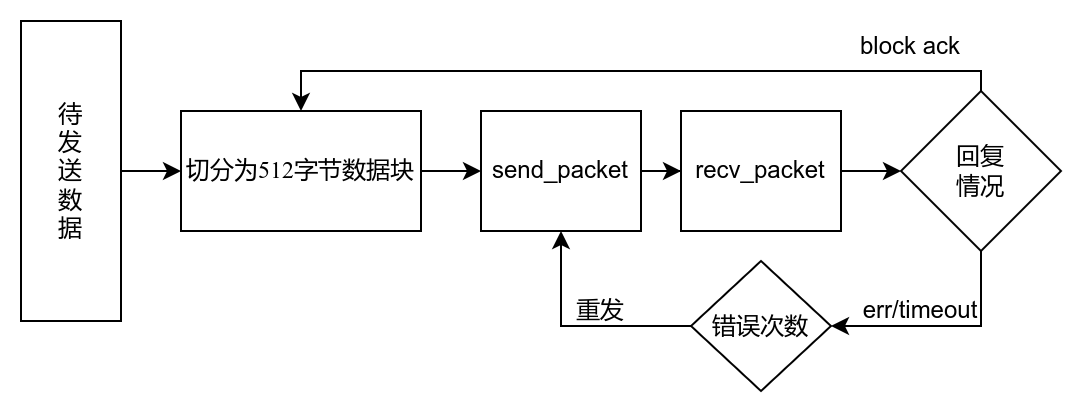


图2-2 数据发送流程

数据接收时，使用recv\_data\_block接收数据块，将数据块返回给recv\_data拼接得到完整数据，接收数据按照接收到的数据长度判断是否完成接收，当接收的长度不足512字节时接收完成，数据块接收后对该数据块号发送确认数据包，并转换到下一个数据包的等待，若数据包的块号是顺序接收的下一个块号，则保留该数据包并进行确认，如果数据包块号错误或超时，则重新发送ack并等待接收，当错误次数大于设定重传次数时，错误返回，数据的接收过程如图2-3所示。

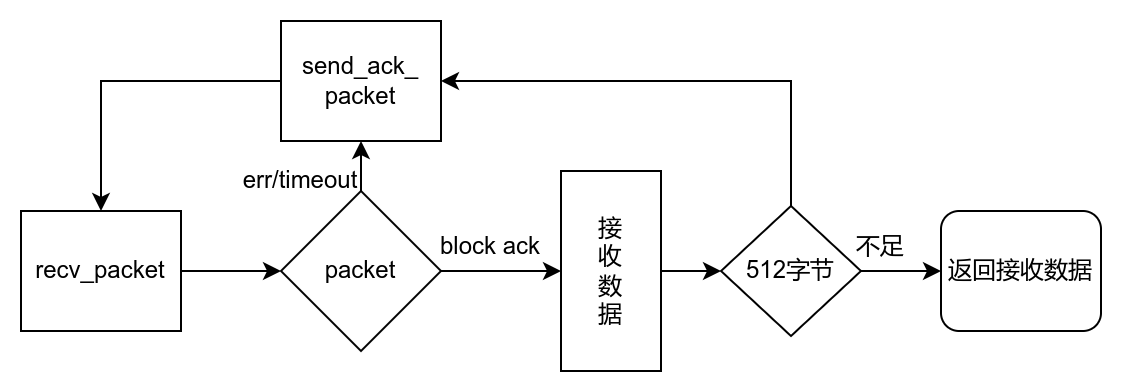


图2-3 数据接收流程

tftp客户端实现中，使用tftp协议层实现的send\_data和recv\_data进行发送和接收数据，除了数据的接收和发送，还需要进行与与服务器连接发送请求，客户端首先初始化socket,尝试向服务器端发送数据时，发送wrq请求数据包，等待服务器进行回复，当接收到服务器回复的ack block为0的数据包时，数据发送过程开始，使用send\_data发送，当接收到错误信息或者等待ack超时，客户端重发wrq请求数据包，若重发次数大于设定重发次数，则发送失败，文件发送过程如图2-4所示。

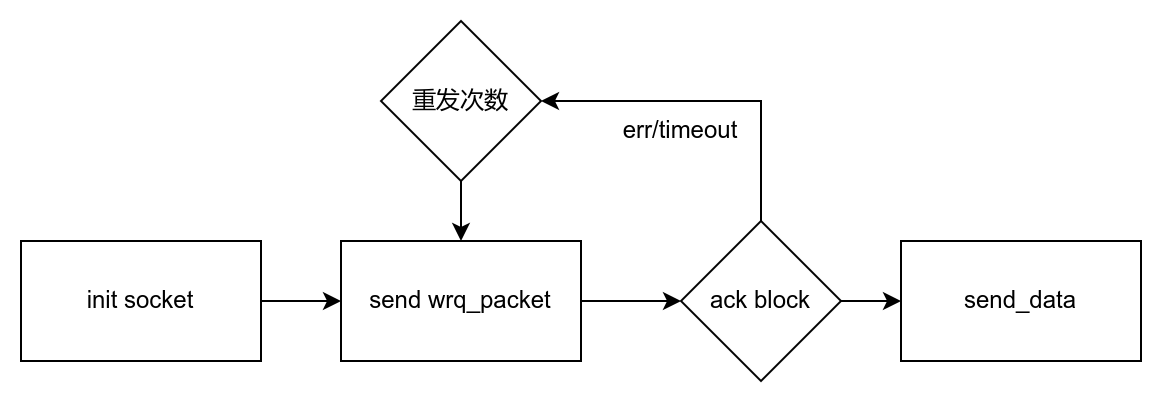


图2-4 客户端文件发送流程

tftp客户端文件接收与文件发送类似，首先初始化init socket端口，发送rrq请求数据包，等待接收block为1的data数据包，若接收成功，则将数据包重新放回缓冲区，转入recv\_data接收完整数据，若接收到错误信息或等待超时，则重新发送rrq请求数据包，若重发次数大于设定重发次数，则接收失败，文件接收过程如图2-5所示。

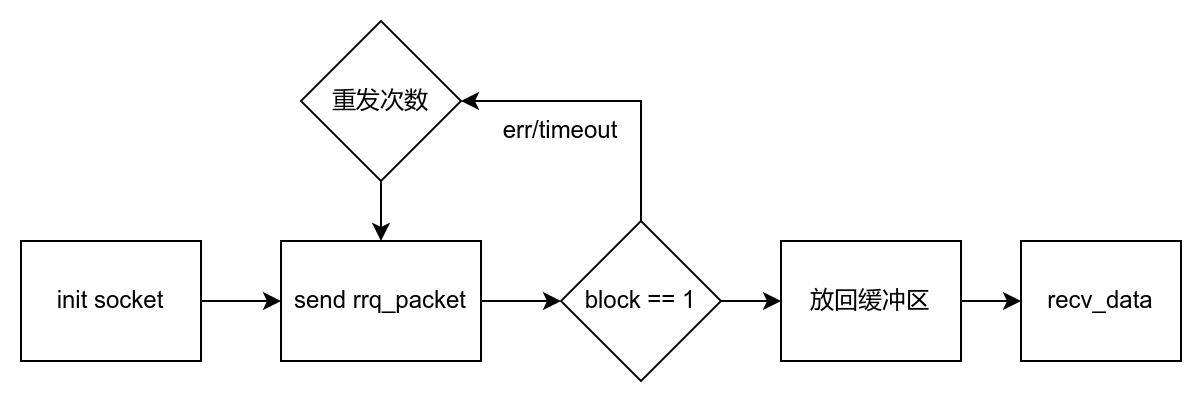


图2-5 客户端文件接收流程

tftp进行文件传输过程时需要对传输的日志、错误信息进行记录，有logger实现，logger包含了格式化日志记录，时间记录等功能，能够对数据传输的时间、传输速度进行计算。

为方便图形界面对tftp协议实现的调用，使用ctypes模块将tftp协议封装为TftpClient类和TftpServer类，在TftpClient类中实现对底层api的调用。

图形界面部分主要包含Ui\_TabClient类、Ui\_TabServer类、Ui\_TabLog类，分别对应客户端界面、服务器端界面和日志界面，在Ui\_TabClient类和Ui\_TabServer类中利用tftp协议层中的logger日志，可以得到文件的传输情况、传输速率等，以及服务器的连接情况，连接的客户端数目、客户端传输的文件情况、以及每个客户端的传输速率。

1. **核心数据结构**

程序的核心数据结构主要由socket层中的upd\_socket，储存socket信息，udp\_peer，储存数据传输的对端信息，tftp层中主要由packet结构，用于储存协议数据包，packet\_link作为接收数据包缓冲区，使用链表实现，当接收到packet数据包后添加到链表末尾，处理数据包时从链表依次读取，tftp\_addr结构体用于保存tftp传输中的本地地址端口信息和远程地址端口信息、ip协议版本等数据，tftp\_log是logger功能的结构体，用于储存logger的文件信息，logger状态，是否输出到标准输出等。

封装为Python模块后，Python模块中的数据结构由TftpClient类和TftpServer类实现，所有的数据储存在实例化后的对象属性中，包括本地、远程地址信息、端口信息、socket信息、传输对端信息等，方便上层模块的直接调用。

## **代码实现**

程序的核心代码分为c语言部分和Python语言部分，c语言部分实现的内容包括socket协议的封装udp.c，tftp协议的实现tftp.c，tftp协议的文件操作tftp\_file.c，tftp客户端实现tftp\_client.c，tftp协议服务器端实现tftp\_server.c。

tftp协议的核心实现在tftp.c中，包括数据包的创建、数据包的转换、数据包的发送接收等内容，实现了RRQ、WRQ、DATA、ACK、ERR五种tftp数据包，并且能够将数据包与socket层的udp消息数据包使用packet2msg和msg2packet进行相互转换，使用send\_packet和recv\_packet调用socket层的send\_msg和recv\_msg实现tftp数据包的接收与发送。按照tftp协议，将数据分为512字节的块进行传输，在send\_data\_block和recvb\_data\_block中实现协议的传输细节，错误处理等。

tftp的客户端利用tftp协议层实现的api，实现了文件发送功能tftp\_put和文件接收功能tftp\_get，位于tftp\_client.c中，根据tftp协议规定的传输流程进行服务器请求和数据发送接收。

tftp的服务器端同样利用tftp协议层实现的api，不同与客户端，服务器端需要实现多客户端的请求处理，使用多线程完成并发操作，满足多个客户端同时的文件传输请求，主要包括主线程server\_main用于接收客户端请求并初始化客户端连接，为客户端创建文件传输线程，send\_thread是文件的发送线程，向该线程对应的客户端发送文件，recv\_thread是文件接收线程，从该线程对应的客户端接收文件，服务器端的核心代码实现位于tftp\_server.c中。

为了方便图形画界面的调用，编译时将tftp层和socket层的代码编译为动态链接库，使用Python的ctypes模块进行api调用，将底层api封装为TftpClient类和TftpServer类，对底层api进行抽象，避免在图形化界面中直接调用底层api。模块的封装位于tftp.py文件中，封装类为cpytftp。

GUI界面的实现使用PyQt实现，将客户端、服务器端、日志记录分为不用的Tab进行实现，在Tab实现中将程序的界面和逻辑部分进行分离，使用多线程完成tftp库中的api调用，避免GUI界面主线程进入函数调用等待界面，日志记录也有子线程单独实现，提高GUI界面的流畅性和保证程序的并行处理能力。

## **实验测试与分析**

## **系统测试及结果说明**

测试环境如下：

操作系统：Linux archlinux 6.5.5-arch1-1

编译器：GNU GCC 13.2.10 GNU Make 4.4.1

调试器：GNU gdb (GDB) 13.2

Python解释器：Python 3.11.5

Python模块：PyQt6 6.5.2 QDarkStyle 3.1

创建Python venv环境并按照依赖模块，编译cpytftp库（如图3-1所示）



图3-1 程序安装编译

运行run.sh脚本打开程序界面（如图3-2所示）



图3-2 程序主界面

界面左下角为界面选择，依次为客户端界面、客户端日志界面、服务器界面、服务器日志界面。

客户端界面（如图3-2所示）右上角为服务器信息输入位置，右下角为文件和传输选项设置，左侧上半部分为服务器文件目录信息，下半部分为传输速率显示。

客户端日志界面（如图3-3所示）主要显示区域用于显示传输详细日志，右侧按钮用于保存日志到文件和清空日志。

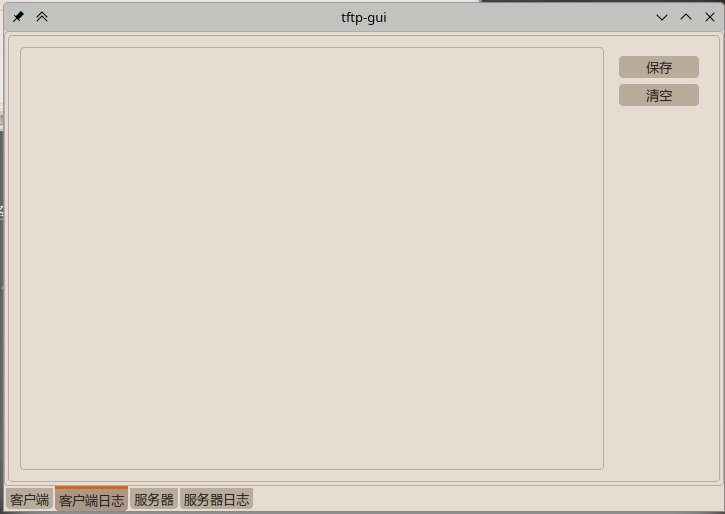


图3-3 客户端日志界面

服务器界面（如图3-4所示）左侧为正在进行的传输文件信息，传输速率显示，右侧上半部分为服务器参数设置，包括监听地址、监听端口、目录、目录文件列表以及是否运行上传的设置，右侧下半部分显示当前的连接情况。

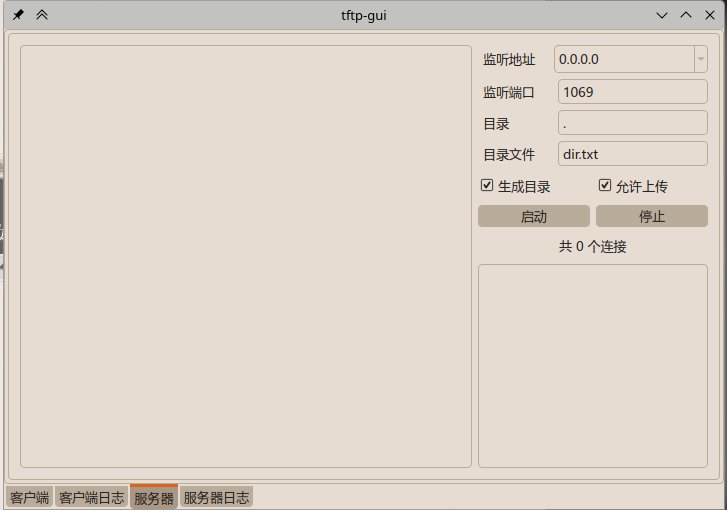


图3-4 服务器界面

服务器日志界面与客户端日志界面相同，包括日志显示区域和日志保存清空按钮。

输入服务端进行连接测试（如图3-5所示），自动获取服务器目录下的dir.txt文件，展示服务器目录下的文件信息。



图3-5 服务器连接测试

使用OCTET模式进行下载测试（如图3-6所示）。程序能够正常进行下载，并显示当前的传输进度和传输速率。

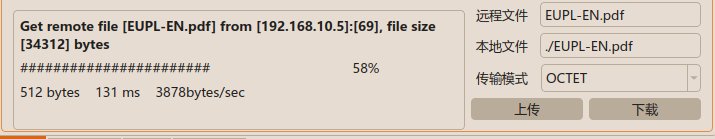


图3-6 OCTET模式下载测试

使用NETASCII模式进行上传测试（如图3-7所示）。程序能够正常进行上传，并显示传输进度和传输速率，传输完成后可以显示整个文件的传输时间和传输平均速度。

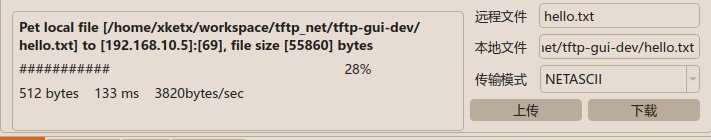




图3-7 NETASCII模式上传测试

传输过程日志记录如图3-8所示。可以显示每一个传输块的块号、数据块大小、传输状态、以及传输用时等信息。

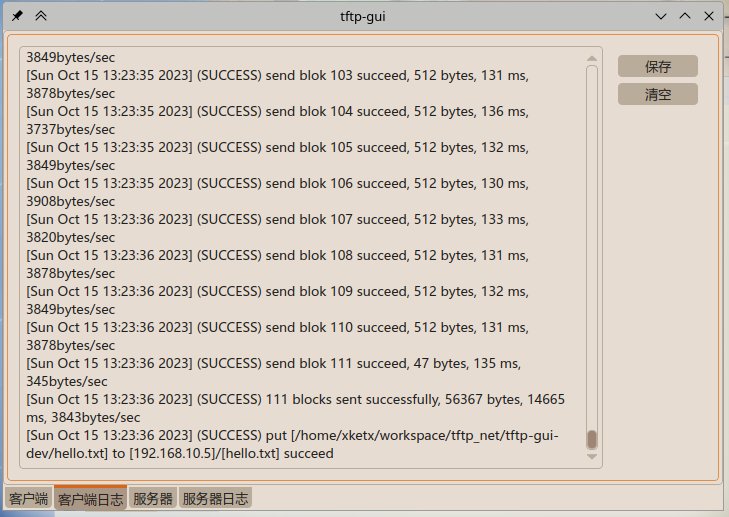


图3-8 传输日志记录

当传输出现错误，例如服务器不存在（如图3-9所示），或者发生丢包（如图3-10所示）等情况时，都能够按照预期形式停止传输或进行重传操作。

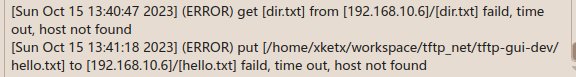


图3-9 服务器不存在错误

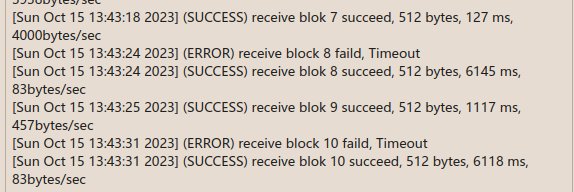
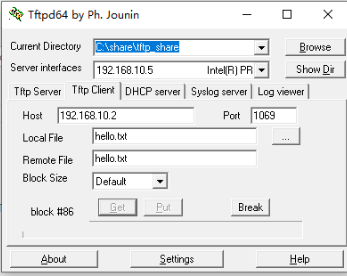


图3-10 传输丢包错误

对服务器功能进行测试，使用tftpd64作为客户端下载服务器文件（如图3-11所示），上传文件到服务器（如图3-12所示），经测试，服务器端上传下载功能均工作正常，服务器端能够显示连接情况和文件传输情况。



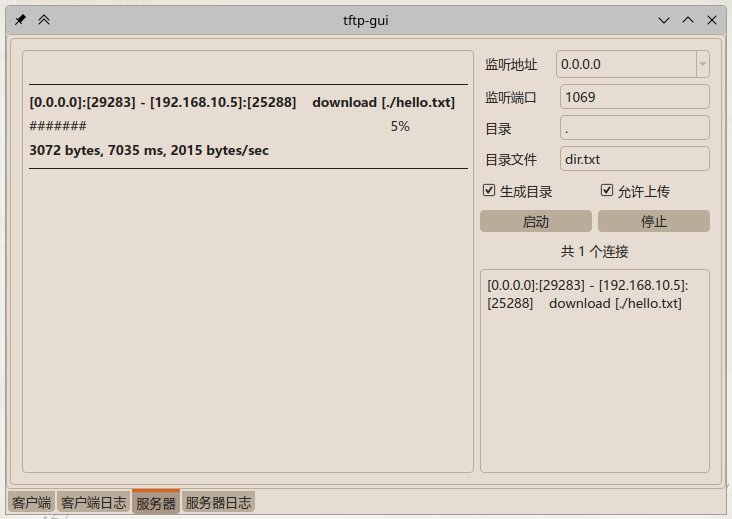


图3-11 服务器端下载文件

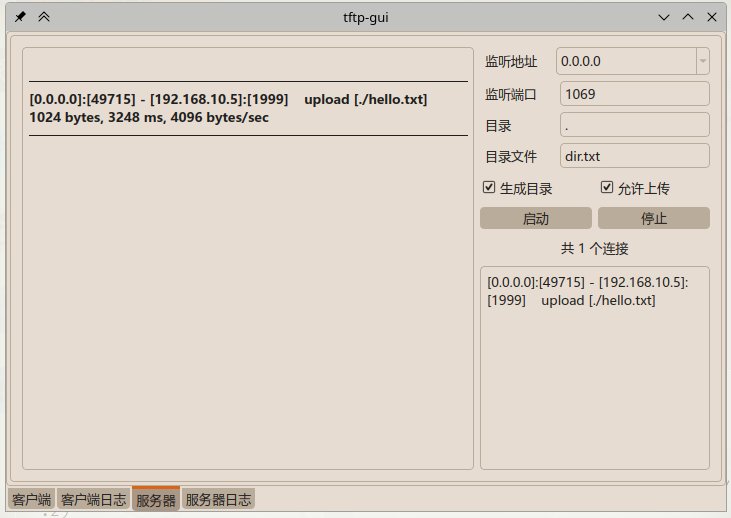


图3-12 服务器端上传文件

## **遇到的问题及解决方法**

在socket层中，实现了socket套接字和初始化和回收，但是在回收时直接free了udp\_socket结构体数据，忽略了需要关闭socket套接字，导致端口占用，再次尝试初始化socket套接字时出现错误，以至于后续调用产生Segement Fault，经过大量调试工作才找到错误的原因，修改回收socket套接字的函数，添加了close socket套接字的部分，程序恢复正常，能够重复初始化套接字。

实现NETASCII模式时，由于需要将文本数据在CRLF/LF之间进行转换，转换后数据块的长度可能不足512字节或者超过512字节，也可能存在CRLF符号在两个相邻的数据块中，导致不能转换所有的CRLF/LF，并且发送提前停止。针对该情况，重新实现了tftp模块中的文件读写功能，在文件读写时对CRLF/LF进行转换，读取的文件内容放入缓冲区，输出时从缓冲区读取，写入文件时，将数据先写入缓冲区，并完成CRLF/LF的转换，再写入文件，使用该机制后，tftp每次能够正常从缓冲区读取512字节，不会导致传输提前停止，并且能够将多个数据块中的CRLF/LF进行替换。

程序打包时，使用pyinstaller工具将所有的模块与python代码进行打包封装，但是打包后得到的可执行文件无法正常加载c语言层编译的so文件，打包后不能够脱离开发环境运行，查阅相关文档后发现，需要使用--add-data参数将自己实现的cpytftp模块路径添加到打包的可执行文件中，重新打包后，程序能够脱离开发环境正常运行。

## **设计方案存在的不足**

程序的日志记录在c语言层面实现，使用临时文件记录日志，GUI层通过读取临时文件获取日志信息，通过这种方式进行日志传输性能受限于IO性能，程序有额外的文件读写开销，当传输速率较快时，日志产生速度较高，占用过多的IO资源，应当将日志传输方式进行修改，例如建立日志缓冲区，c语言层将日志写入日志缓冲区，GUI层从日志缓冲区读取日志。

tftp传输过程中的错误反馈在c语言层面实现，但是Python层未对所有的错误情况实现相应的接口，导致发生错误时，虽然底层能够正确处理，但是用户界面不能展示所有的错误信息。

程序虽然实现了服务器端的功能，并且理论上支持多个客户端连接，有并行传输的能力，但是未进行足够的测试，不能确保服务器端的稳定性，应该使用更多的客户端进行连接，测试在多个客户端情况下的文件传输能力。

## **实验总结**

## **实验感想**

本次计算机网络实验包含了三个环节，真实设备组网、仿真组网和socket套接字编程。在真实设备组网中，学习了网线水晶头制作、网络交换机、路由器、无线AP等网络设备的配置，通过配置路由器，学习了RIP协议和OSPF协议的配置，掌握了网络配置的基础技能。

在仿真组网中，使用eNSP软件创建虚拟网络，实现了简单的校园网网络配置，能够让拥有数十个设备的网络正常运行，以及实现设备的跨网络访问，通过配置ACL实现网络间的访问控制，以及配置虚拟AC和AP使有线网络和无线网络能够相互访问。

socket套接字编程实验中通过实现文件传输协议tftp，学习了如何使用底层socket套接字构建网络应用程序，了解了如何使用udp协议实现文件的传输，在udp协议不提供可靠传输的情况下，通过错误反馈、超时重传等机制使文件能够在网络丢包、乱序等情况下正确传输。除了学习socket套接字的使用，还尝试了分层次分模块开发程序，将程序的开发按层次进行，提高程序开发效率以及提高了代码的可移植性，在程序开发时，还学习了使用Python和PyQt开发简单的GUI界面，学习了使用多线程方式处理界面逻辑，掌握了基础的图形化程序开发能力。

## **意见和建议**

试验包含真实设备组网和仿真组网部分，仿真组网虽然可以使用更多的网络设备，构建更复杂的网络环境，但是相较于真实设备组网，仿真组网不够直观，不能很好地体现网络的工作情况，真实设备组网能够提高学生的动手解决问题的能力，因此可以增加真实设备组网的内容，例如增加防火墙设备的配置，增加组网规模等。

|  |
| --- |
| 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：** |