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　李　涵

班　　级　网安 1903 班

学　　号　U201910388

联系方式　13797390520

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

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

根据要求的功能，我主要将本次实验的程序结构分为以下 5 个模块：

(1) 初始化模块：完成程序所需的变量与资源文件的初始化。

(2) 用户交互模块：生成用户交互界面，提示用户选择操作，并输入上传/下载的文件名，文件类型，然后调用对应模块，最后完成收尾资源回收工作。

(3) 文件上传模块：自定义函数，由用户交互模块获取需上传的文件名作为参数，让用户选择上传格式，以相应方式连接 TFTP 服务器上传文件。

(4) 文件下载模块：自定义函数，由用户交互模块获取需下载的文件名作为参数，让用户选择下载格式，以相应方式连接 TFTP 服务器下载文件。

(5) 错误反馈包解析模块：接收到服务器的错误反馈数据报时，以数据包中的错误码字段为参数，调用此函数解析为相应的错误信息，打印反馈给用户，同时记录日志。

各个模块结构的具体调用流程如下：

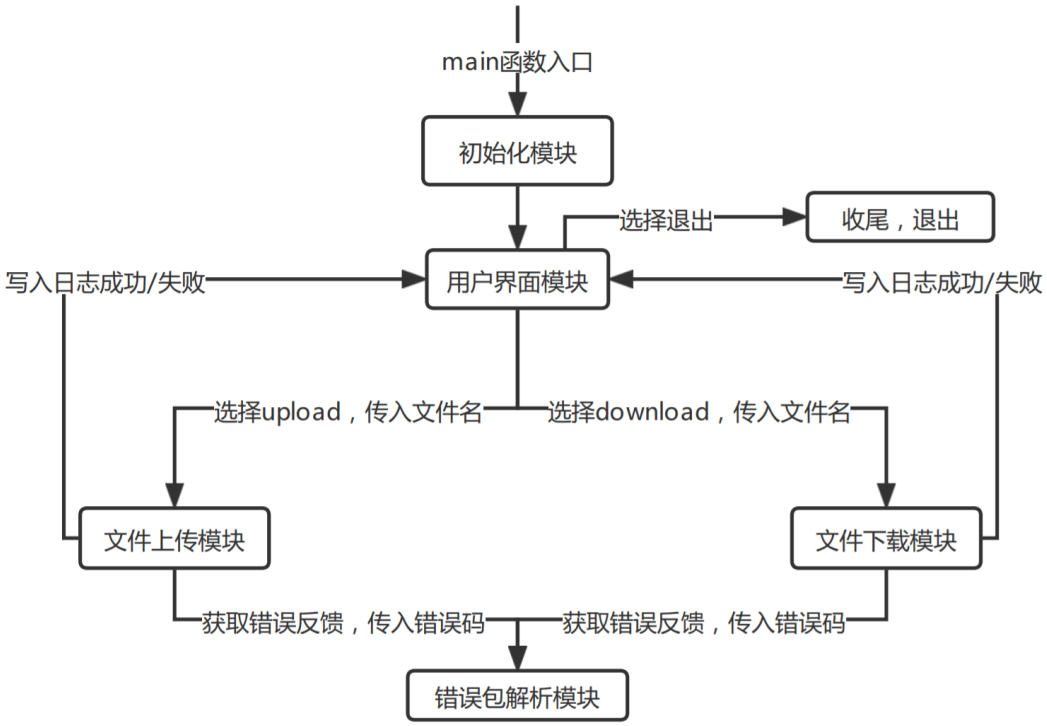

图 2.1 系统结构设计模块调用图

2.2 详细设计

（一） 核心数据结构

(1) TFTP 数据包结构体

定长的数据包易于处理，所以 TFTP 数据包结构体采用定长 516 字节数据，它根据操作码 op 的值分为以下 5 种类型：

1. op = 1 读文件请求包 RRQ，此时中间 2 字节与 data 的前一部分作为可变的 filename 字段，表示要读取或上传的文件名，以 0 结尾；其后的 data 段为可变的传输数据类型二进制 octet 或字符串 netascii，也以 0 结尾。
2. op = 2 写文件请求包 WRQ，后面的格式同请求包。
3. op = 3 文件数据包 DATA，将中间 2 字节作为块号，表示数据包的顺序，data 段为具体数据。
4. op = 4 回应包 ACK，将中间 2 字节作为 ACK 序号，表示对应的数据包块号成功接收，此时 data 段无意义。
5. op = 5 错误信息包 ERROR，将中间 2 字节作为错误码，表示错误类型，data 段为可变的字符串形式的错误描述信息。

综上，中间 2 字节可以有三种解析形式，当 op = 1/2 时，作为 2 个 char 类型，当 op = 3/4 时作为 short 类型，op = 5 时也是 short 类型。而 data 段为可变长字符串，解析格式都是 char 类型，因此可以设定如下数据结构：

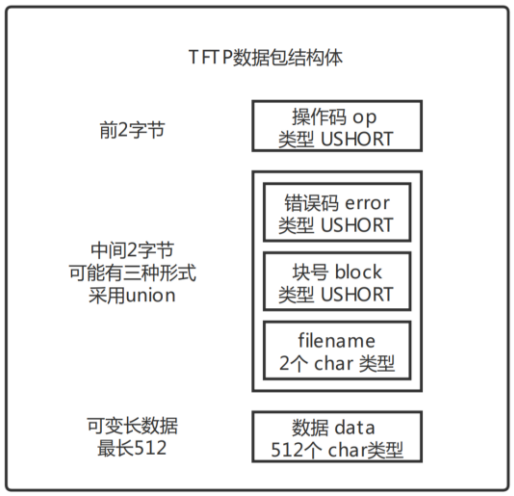

图 2.2 TFTP 数据包结构体

(2) SOCKET

SOCKET 类型本质上是一个 int 类型，是系统调用创建套接字时返回的句柄，对应一个具体套接字，我采用了 socket(AF_INET, SOCK_DGRAM, 0) 函数创建了一个套接字，并用 SOCKET 类型的 client_socket 接收来标注与服务器通信的接口。其中 SOCK_DGRAM 参数代表本套接字用来传输 UDP 数据报。

(3) sockaddr_in

该数据类型是一个结构体，用来指示 recvfrom 与 sendto 函数接收或发送的接口地址。在这里我定义了 sockaddr_in 类型的 server_addr 来保存服务器的接口地址。

其中成员变量 sin_family = AF_INET 地址族；sin_addr.s_addr = inet_addr("127.0.0.1") 表示 IP 地址为本机地址，inet_addr 将字符串转换为网络字节序二进制；sin_port = htons(69)，表示端口为 69，是 UDP TFTP 的默认端口，htons 将整型转变成网络字节序二进制。

（二） 核心函数流程

(1) 初始化模块：位于 main 函数入口。

a) 初始化日志文件，若已存在则定位到末尾，不存在则创建；

b) 初始化 winsock，并判断 winsock 版本的正确性；

c) 使用 sockaddr_in 结构体设置目标服务器 IP 地址和端口，作为信息交流接口；

d) 调用 WinAPI 创建 socket，并设置为费阻塞模式。

(2) 用户交互模块：位于 main 函数初始化后。

a) 打印用户界面，让用户选择操作；

b) 让用户输入想要上传/下载的文件名；

c) 以该文件名为参数调用文件上传/下载函数；

d) 结束后回收资源，关闭 socket、日志文件，清理 winsock。

(3) 文件上传模块：自定义函数。

a) 让用户选择上传格式 netascii 或 octet；

b) 用相应的格式（二进制或文本）打开想要上传的文件，并获取文件大小；

c) 构建 WRQ 报文并发送；

d) 等待接收 ACK 0 报文最多 2 秒，若未收到则重传请求包最多 3 次；

e) 开始上传数据包，每个 ACK 最多等待 2 秒，每个数据包最多超时重传 3 次；

f) 若收到非顺序 ACK 则忽略并写入日志，若收到错误反馈则调用错误反馈包解析模块，若有超时则写入日志并失败返回；

g) 计算总时延与传输速度，反馈给用户，写入日志。

(4) 文件下载模块：自定义函数。

a) 让用户选择下载格式 netascii 或 octet；

b) 用相应的格式（二进制或文本），以想要下载的文件名，创建/覆盖并打开文件；

c) 构建 RRQ 报文并发送；

d) 等待接收每个文件数据报文最多 2 秒，若未收到则重传 ACK 最多 3 次；若等待的是 block 1，则重传 RRQ 请求包；

e) 若收到非顺序 block 的数据报则忽略，并重传该 ACK，写入日志，若收到错误反馈则调用错误反馈包解析模块，若有超时则写入日志并失败返回；

f) 计算总时延与传输速度，反馈给用户，写入日志。

(5) 错误反馈包解析模块：自定义函数。

接收到服务器的错误反馈数据报时（操作码为 5），以数据包中的错误码字段为参数，调用此函数解析为相应的错误信息，打印反馈给用户，同时记录日志。

错误码	表示的意思
0	未定义 Not defined, see error message
1	文件未找到 File not found
2	访问被拒绝 Access violation
3	磁盘满或超出可分配空间 Disk full or allocation exceeded
4	非法的 TFTP 操作 Illegal TFTP operation
5	未知的传输 ID Unknown transfer ID
6	文件已经存在 File already exists
7	没有该用户 No such user

图 2.3 错误反馈码解析表

2.3 代码实现

(1) TFTP 数据包结构体

```
//TFTP数据包结构体
struct tftp_pack{
    USHORT op;//操作码
    union{
        USHORT error;//错误码
        USHORT block;//data块号
        char filename[2];//承接下面的data
    };
    char data[DATA_SIZE];// 数据或错误描述
};
```

图 2.4 TFTP 数据包结构体

(2) 初始化模块

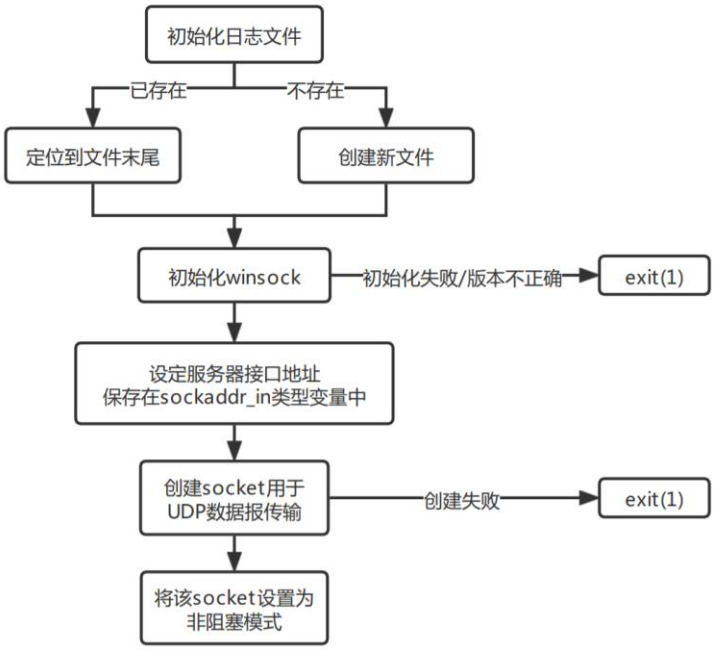

图 2.5 初始化模块流程图

其中服务器接口地址采用如下方式设定，成员变量 sin_family = AF_INET 地址族；sin_addr.s_addr = inet_addr("127.0.0.1") 表示 IP 地址设置为本机地址，inet_addr 将字符串转换为网络字节序二进制；sin_port = htons(69)表示端口为 69，是 UDP TFTP 的默认端口，htons 将整型转变成网络字节序二进制。

```
//服务端地址
memset(&server_addr, _Val: 0, sizeof(server_addr));
server_addr.sin_family = AF_INET;//AF_INET:TCP/IP IPv4
server_addr.sin_addr.s_addr = inet_addr( cp: "127.0.0.1");
server_addr.sin_port = htons( hostshort: 69);//设置端口 69
```

图 2.6 服务器接口地址设置

```
//创建 socket
client_socket = socket(AF_INET, SOCK_DGRAM, protocol: 0);//SOCK_DGRAM:UDP数据报
if (client_socket < 0) {     //套接字创建成功时，返回套接字，失败返回-1
    perror( _ErrMsg: "Create Socket Failed:");
    exit( _Code: 1);
}
//非阻塞模式,当创建一个套接口时，它就处于阻塞模式
unsigned long Opt = 1;//如允许非阻塞模式则非零，如禁止非阻塞模式则为零
ioctlsocket(client_socket, cmd: FIONBIO, &Opt);
printf( format: "Client's Socket created!\n\n");
```

图 2.7 socket 采用 WinAPI 创建

(3) 用户交互模块

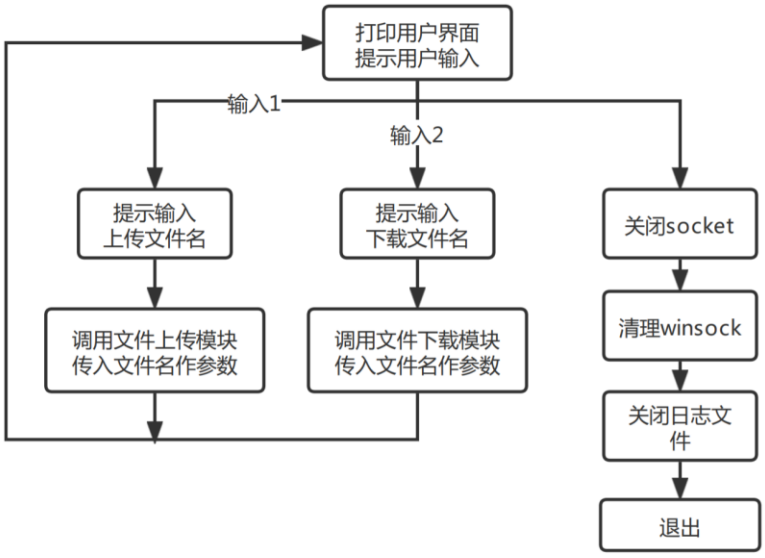

图 2.8 用户交互模块流程图

```
==========------TFTP Client------=======
Operate:
1. Download a file from the server
2. Upload a file to the server
3. Quit
```

图 2.9 用户选择界面

(4) 文件上传模块

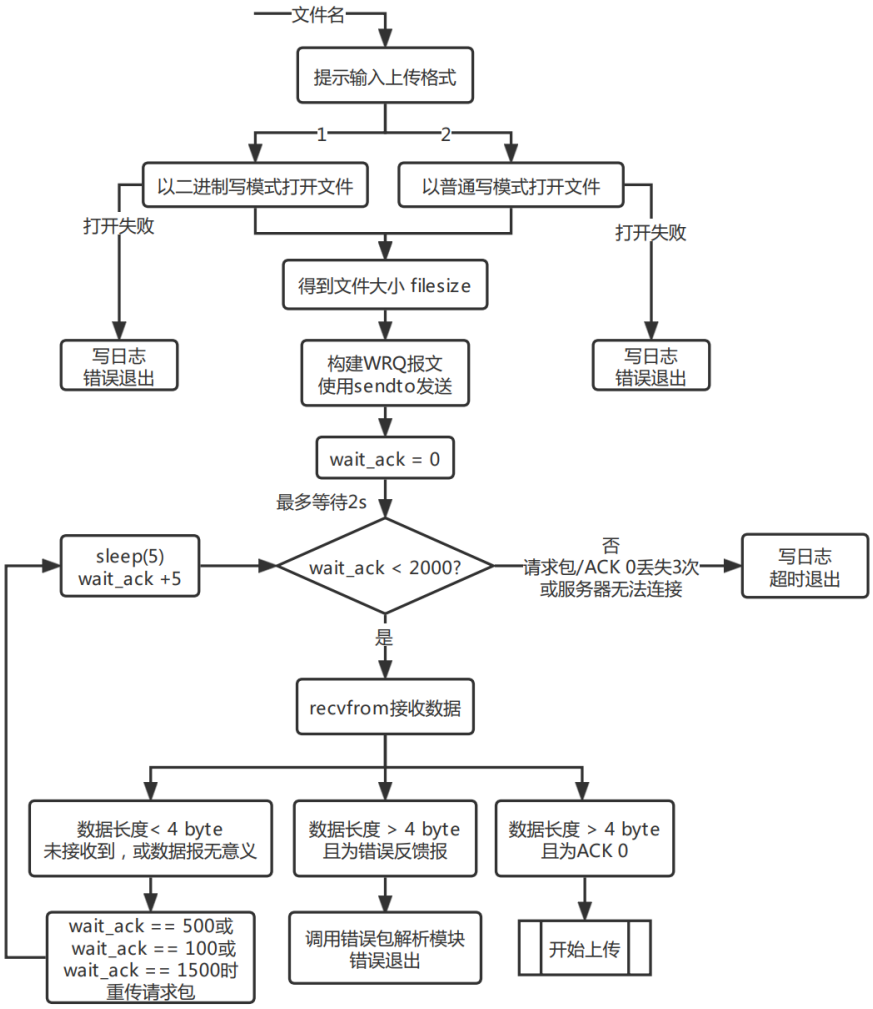

图 2.10 文件上传写请求发送流程图

文件上传首先需要打开待上传文件，然后获取文件大小方便最后计算速度。

之后需要发送 WRQ 报文，等待服务器相应 ACK 0 之后才能正式开始传输文件。由于 WRQ 报文和 ACK 0 都有可能丢失，因此设定在 2s 的超时时限内，若没收到 ACK 0 则每隔 0.5s 重新发送写请求包。由于采用了非阻塞状态，每次调用 recvfrom 时都立即返回，因此需要 sleep 5 毫秒。

由于服务器采用 69 号端口接收 WRQ 报文，但是会用随机空闲端口来接收后续的文件，因此需要根据 ACK 0 的报文信息重新设置之后发送文件的端口。

在成功收到 ACK 0 报文后，开始文件数据传输，同时开始计时：

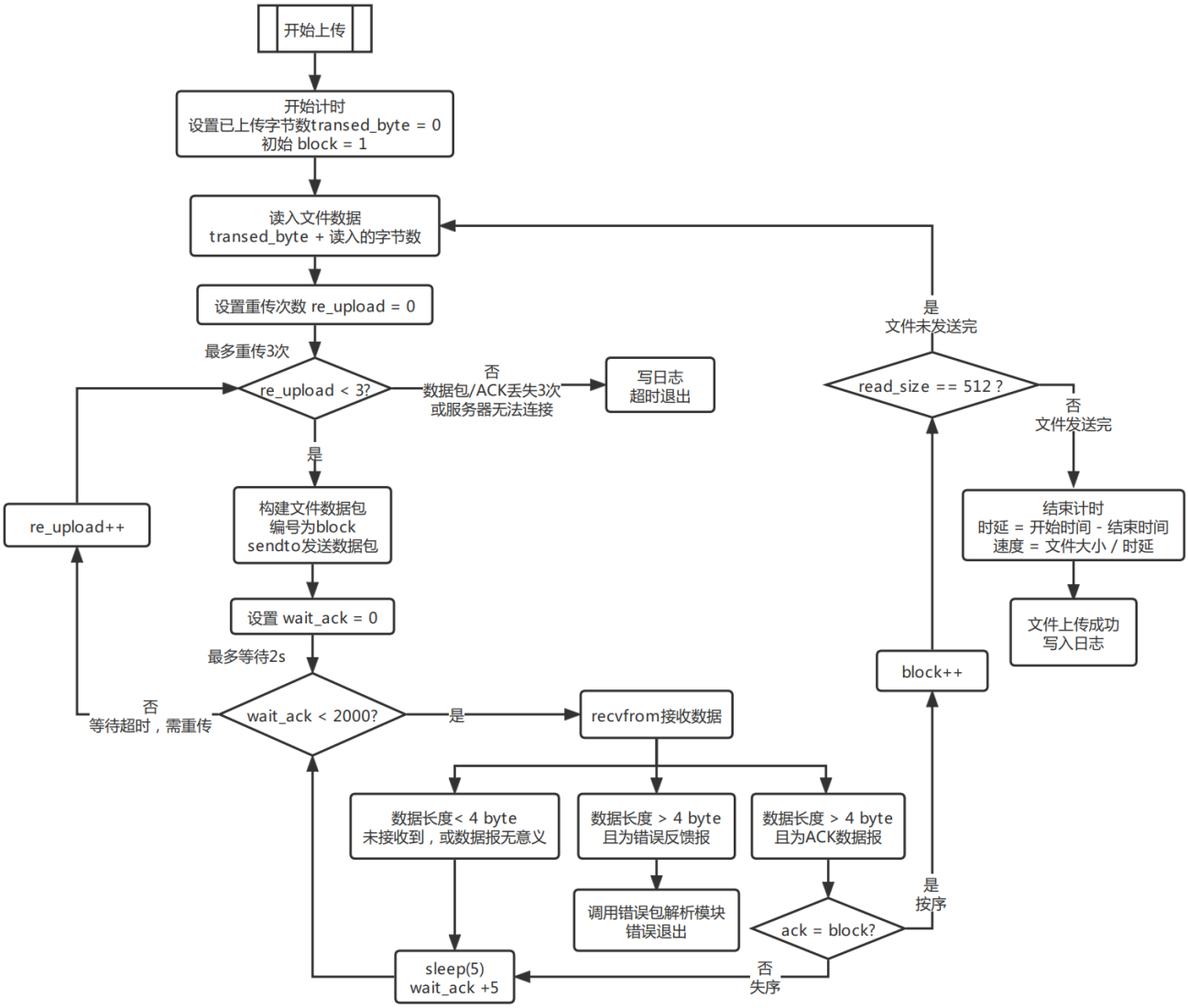

图 2.13 文件数据上传流程图

写入块编号不能直接采用 short 类型写入，需要用 htons 函数转换为网络字节序。

read_size 为最多从文件中读取 DATA_SIZE(512 字节)的数据大小，因此如果 read_size == 512 则文件未读取完，若小于 512 则说明文件上传了最后一块数据，则终止上传，服务器收到小于 512 字节的数据包也会明白上传完毕。

同时每个数据包最多重传 3 次，每次重传都用最多 2s 等待接收对应的 ACK。接收到服务器传来的数据包后的解析，只有当顺序接收到正确 ACK 才停止，否则一直循环到超时。

成功上传后，停止计时，时延=结束时间-开始时间，速度=文件大小/时延。

(5) 文件下载模块

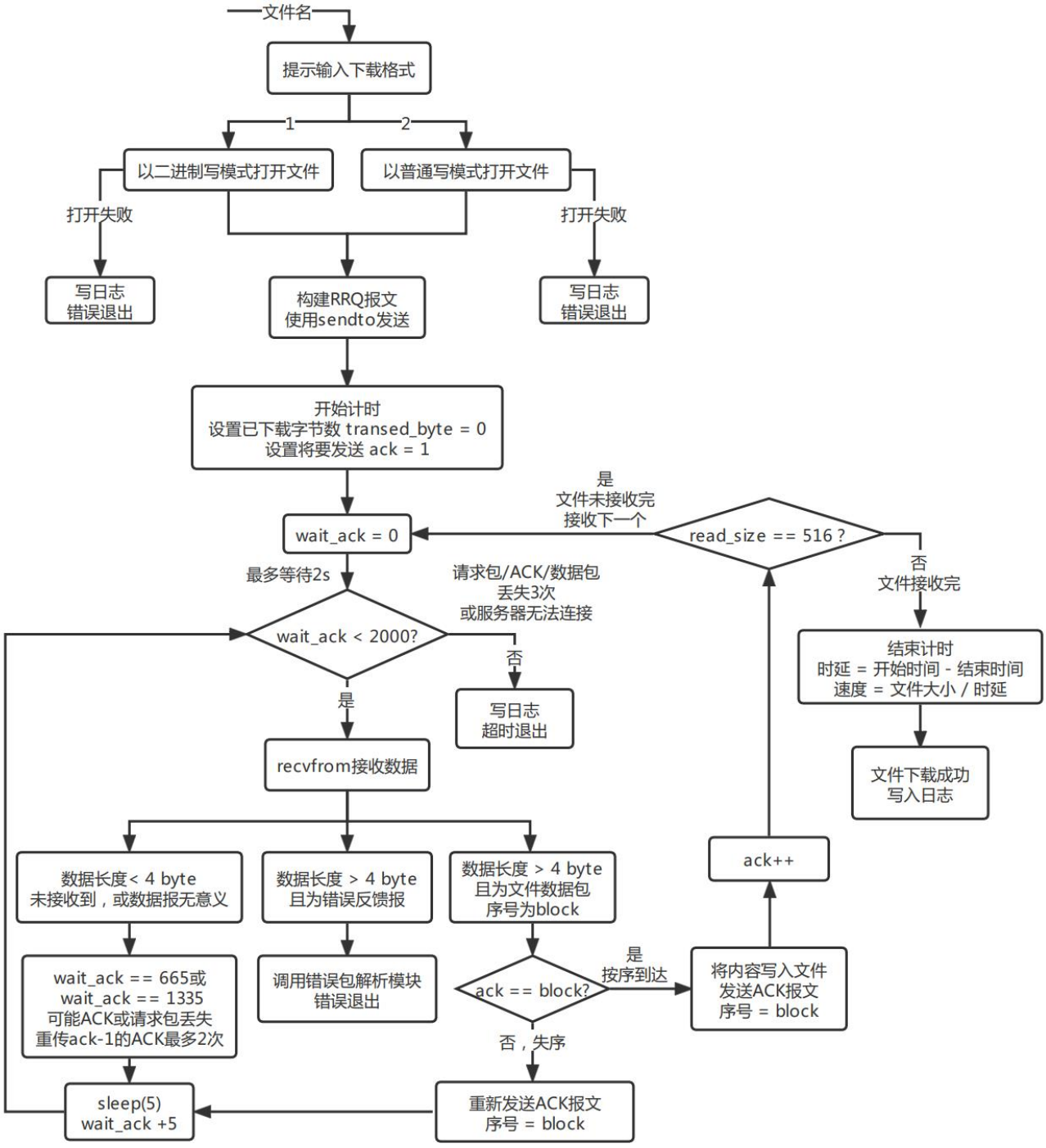

图 2.17 文件下载模块流程图

文件下载模块首先需要向服务器发送一个 RRQ 请求报文，然后等待服务器的相应。服务器会换端口来直接发送 block 1 的文件数据包来响应，因此还是需要保存该接口地址，以后所有的 ACK 报文都向该地址发送，与上传模块一样。

接收到的是文件数据包，当解析到操作码为 3 的报文时，需要首先判断是否按序，没按照顺序则丢弃并写入日志，同时重传该冗余报文的 ACK；如果是按序报文，则不仅需要发送 ACK，同时写入创建的文件，由于接收到的报文前 4 字节不是文件数据，因此写入长度要是接收到的长度减 4。

由于传输的 ACK 或数据包可能丢失，因此设定最多重传 2 次，若是请求包丢失则重传 RRQ。

最后下载成功计算时延与速度，并写入日志，代码与上传模块类似。

(6) 错误反馈包解析模块

在解析过程中，若遇到操作码为 5 的报文则解析其第 2 和 3 个字节，采用 ntohs 函数将网络字节序转为主机字节序后，采用 short 类型传入该模块，调用 switch(){}结构选择打印对应的错误报告字符串，并记录日志。

三、 实验测试与分析

3.1 系统测试及结果说明

测试环境：Windows11 系统，采用 clumsy 软件模拟丢包

TFTP 服务器软件：tftpd64

TFTP 服务器接口：IP 127.0.0.1，端口号 69

测试文件：大小为 128KB 的 jfif 图片，165KB 的 txt 文本

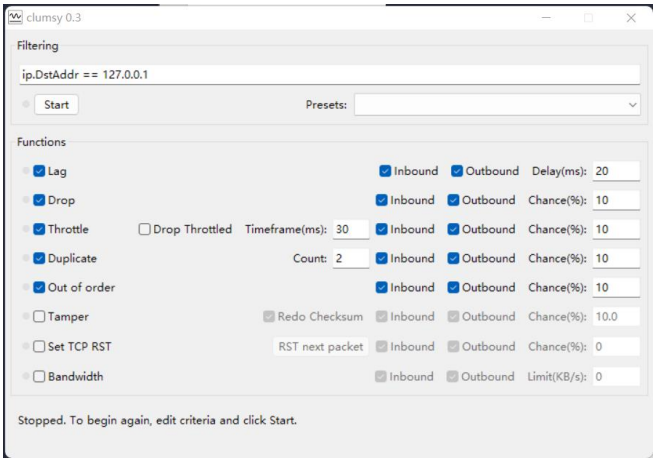

图 3.1 开启丢包干扰

```
=======------TFTP Client------=======
Operate:
1. Download a file from the server
2. Upload a file to the server
3. Quit
1
Input the name of the file you want to download:
5.txt
Choose a format: 1.netascii  2.octet
1
Download 5.txt: received data ack1.
Download 5.txt: received data ack2.
```

图 3.2 设定以 netascii 模式下载 5.txt

```
Download 5.txt: ignored data ack327.
Download 5.txt: resending ACK of ack327.
Download 5.txt: received data ack328.
Download 5.txt: received data ack329.
Download 5.txt: received data ack330.
Download successfully.
Wed Jan 05 01:28:43 2022
Download 5.txt, mode: netascii, size: 164.53kB, time consumed: 42.947s, speed: 3.83kB/s.
```

图 3.3 能够重传，下载成功

```
==========------TFTP Client------========
Operate:
1. Download a file from the server
2. Upload a file to the server
3. Quit
2
Input the name of the file you want to upload:
1.jfif
Choose a format: 1.netascii  2.octet
2
Upload 1.jfif: Sent the block1.
Upload 1.jfif: Sent the block2.
Upload 1.jfif: didn't receive ACK of block2, resending.
```

图 3.4 设定以 octet 模式上传 1.jfif

```
Upload 1.jfif: Sent the block253.
Upload 1.jfif: Sent the block254.
Upload 1.jfif: Sent the block255.
Upload successfully.
Wed Jan 05 01:38:04 2022
Uploaded 1.jfif, mode: octet, size: 127.09kB, time consumed: 74.378s, speed: 1.71kB/s.
```

图 3.5 能够重传，上传成功

```
==========------TFTP Client------========
Operate:
1. Download a file from the server
2. Upload a file to the server
3. Quit
1
Input the name of the file you want to download:
1234423.hahah
Choose a format: 1.netascii  2.octet
2
Wed Jan 05 01:40:07 2022
ERROR 1: File not found.
```

图 3.6 请求下载不存在的文件，可以接受并解析错误反馈包

```
Upload 1.jfif: ignored data block215.
Upload 1.jfif: didn't receive ACK of block216, resending.
Upload 1.jfif: ignored data block234.
Upload 1.jfif: ignored data block234.
Upload 1.jfif: didn't receive ACK of block235, resending.
Wed Jan 05 01:38:04 2022
Uploaded 1.jfif, mode: octet, size: 127.09kB, time consumed: 74.378s, speed: 1.71kB/s.

Wed Jan 05 01:40:07 2022
ERROR 1: File not found.
```

图 3.7 日志文件写入

3.2 遇到的问题及解决方法

(1) TFTP 数据包结构体的设置，如何才能让第 2、3 个字节能够采用不同的解析方式。采用 union 结构可以完美解决。

(2) recvfrom 函数默认为阻塞态，如果中间服务器的数据包丢失，则客户端会直接卡死在这里，这显然无法完成程序要求，因此最后查询资料需要更改为非阻塞态。

(3) 刚开始忘了服务器总是采用 69 号端口来接收请求包，而另设置一个随机空闲端口来与客户端传输其他数据，因此总是传输失败，怎么 debug 也没用，苦恼了很久。之后才想起来服务器的端口更变，需要更新服务器接口地址才能继续传输。

(4) 请求包/第一个应答包丢失。一开始没有设置请求包重传，在开启链路干扰软件后，常常完全一开始就连接不上服务器，后来才想到，如果 WQR 或者 RRQ 或第一个服务器应答包丢失，那会直接断连，设置了请求包重传后才终于解决。

(5) 一开始不太理解 netascii 与 octet 模式的区别，用 netascii 去传图片，然后图片文件总是不完整，只能传一小段就提示上传/下载成功了。后来才了解到 netascii 仅适合传输文本文件，其他文件格式还是得用二进制。在此基础上我还加入了文件大小比对，来判断文件是否已经完整传输。

(6) 错误反馈包解析时，一开始是直接传入接收到的数据包第 2、3 个字节，采用 short 类型解析，总是没法正确的和数字 0~7 比对，后来调试了很久才发现它是网络字节序，需要用 ntohs 函数转换成主机字节序之后才能正确得到数字 0~7。

(7) 开启链路干扰后，下载时有时会收到 0 号错误反馈包：ERROR 0: Not defined, see error message.让我摸不着头脑，后来发现是服务器达到重传上限制后发送，并停止数据传输。

3.3 设计方案存在的不足

(1) 在下载不存在的软件时也会创建一个空文件，没有在判断后及时删除。

(2) 时间控制不太准确，只将 Sleep 函数运行的时间算在了时间控制里，没有考虑代码运行时间，因此实际最大等待时间是会超过设定的 2s 的，但是问题不大。

(3) 采用每 5 毫秒进行一次 recvfrom 函数调用，由于是非阻塞态所以常常会返回空值，然后判断后进行下一次循环接收，这样带来额外性能开销。

(4) 等待时间过短，2s 最大等待时间在链路干扰过高时常常断连，不过这个很好调整。

四、 实验总结

4.1 实验感想

(1) 实验一

本次实验对我来说比较难，由于此前完全不熟悉套接字编程，我花了很长时间才完成整个程序。

在学习理论知识的时候对 socket 认识不是很深，只知道套接字是连接应用层和运输层的接口，对我来说这是一个很抽象的概念，同时我也一直对运输层如何根据数据报中的端口来交付给上层应用比较疑惑，而且我一直以为“端口”是物理真实存在的……完整地完成这次实验下来，终于解答了我的诸多疑惑。

首先“端口”就是一个标记接口编号，运输层根据这个编号将数据交付给不同的 socket，而 socket 只能绑定一个端口、一个进程，因此说它是一个“接口”，使用进程创建，然后绑定一个端口号，剩下的全都交给操作系统自动完成了，因此可以很方便地通过编程来完成网络交流，不用在意实现细节。

然后就是各种 WindowsAPI 函数的熟悉过程。一开始看到这些奇奇怪怪的 API 函数和参数就头大，通过大量查找资料，才逐渐学会他们的用法，并了解各个参数的含义、各个数据类型的作用、网络字节序与主机字节序的转换等等，整个过程非常枯燥，但还是能感受到这些系统实现的精妙之处，最终能够实现我的程序，还是让我很有成就感。

认识了 API 函数，再就是逻辑实现。在学习理论知识的时候，各种重传什么的，感觉挺简单，实际自己写起算法来，才知道需要考虑的东西太多太多，稍有遗漏之处，整个程序甚至很难 debug。总之我花费了很长时间不断完善我的程序，最终提交的结果至少我自己是比较满意了。

总的来说，这次基本从 0 基础开始让我学习了 socket 底层编程与可靠信道传输的知识，不仅是对课堂理论知识的巩固，更有大量扩展，让我印象深刻，查询了很多资料，也学习到了很多。

(2) 实验二

实验二并不太难，它让我了解了 VLAN 的概念，并让我了解了真正的组网过程，同时实机操作更锻炼了我们动手能力。

一开始我们在 eNSP 的虚拟设备上学习模拟组网实验，刚刚接触 VLAN 这个概念，而且对路由器和交换机转发表的构建了解还不深，又碰上了三层交换机这种新知识，因此手足无措。于是我在网上查询了许多 VLAN 有关的知识，完整地理解了 VLAN 的通信过程和三层交换机完成的工作，最后才能理解这个实验要我们完成的全部内容和逻辑。

在实物组网的配置过程和虚拟设备上几乎相同，唯一要注意的就是网线的连接方式，最后配置 WLAN 的时候有些麻烦，不过由于我此前配置过自己的 WIFI，总的来说完成得还是比较顺利。

通过此实验，对组网配置过程有了基本了解，同时学习了 VLAN 这一重要概念。

(3) 实验三

实验三是实验二的扩展，让我们实现一个综合组网的过程。

有实验二的学习基础，我对 VLAN 以及各种设备的具体配置流程不再那么陌生，本次实验主要在于组网的逻辑。

首先是让我们划分地址块，在学习完 Ipv4 部分的理论知识后，我对 IP 的具体划分还是不太熟悉，通过此实验我彻底弄懂了整个过程，同时这也是考试的终点内容之一，非常有收获。

然后是路由器和交换机转发表的建立过程。路由器需要我们先配置 OSPF，然后等待所有路由节点交换 OSPF 信息即可完成转发表的建立，不建立路由器的转发表则所有网络节点无法相互访问。而交换机的转发表是自动建立的，当收到数据时，直接将该 IP 与端口列入自己转发表，因此一开始 ping 不通的设备过一会就能 ping 通了。

最后 VLAN 的配置过程，已经在实验二中向我们介绍了，所以不太难。

整个流程让我彻底明白了组网的过程，有些复杂，中间出了许多错误，但是只要完成了连接流程，理解了组网逻辑，多花时间还是能够顺利完成。

4.2 意见和建议

对于第一个实验，我还是希望老师能够讲一下常用的 recv 接收方式，一开始我对 recv 的循环接收过程比较迷惑，查询了许多网页才了解到采用非阻塞接收并设置等待这种方式。我认为了解这个内容可以帮助我们更快入门。

实验二我希望任务书更加详细一点，由于大家都是第一次接触到实物组网的过程等，以及不理解 VLAN 是什么以及实现方式，希望这些能容能在以后的任务书里有更详细的介绍。

实验三主要是环境配置，我的电脑无论如何也装不上 eNSP 这个软件的环境，查了很久的资料通过方式也没有解决，放在 win10 虚拟机里也装不上，非常玄学，最后还是在 win7 的虚拟机里成功运行的。没法运行软件让我一直很着急，很多同学也是这样的情况，浪费很多时间。所以我希望老师能够延长课内学时让我们用实验室的电脑，或者发布一个配置好的虚拟机等方式让我们运行。

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：