

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 李欣宇

班 级 信安 1901 班

学 号 U201911658

联系方式 19963384224

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	1
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.....	2
二、 实验过程.....	3
2.1 系统结构设计.....	3
2.1.1 系统模块.....	3
2.1.2 数据处理流程.....	4
2.2 详细设计.....	5
2.3 代码实现.....	7
三、 实验测试与分析.....	10
3.1 系统测试及结果说明.....	10
3.1.1 测试环境.....	10
3.1.2 测试方案.....	10
3.1.3 测试过程.....	10
3.2 遇到的问题及解决方法.....	16
3.3 设计方案存在的不足.....	17
四、 实验总结.....	18
4.1 实验感想.....	18
4.2 意见和建议.....	18

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 系统模块

模块划分与功能

在 Socket 实验中根据系统所需要满足的功能分为两大部分，分别是上传文件功能和下载文件功能。在代码中分别使用 upload 和 download 两个函数实现。

upload 函数中主要是使用 sendto 函数发送数据包，使用 recvfrom 函数接收 ACK 确认包来实现将文件上传服务器的功能。

download 函数中主要是用 recvfrom 函数接收从服务器发来的数据包，使用 sendto 发送 ACK 确认包，具体系统模块框图如下图 2.1 所示。

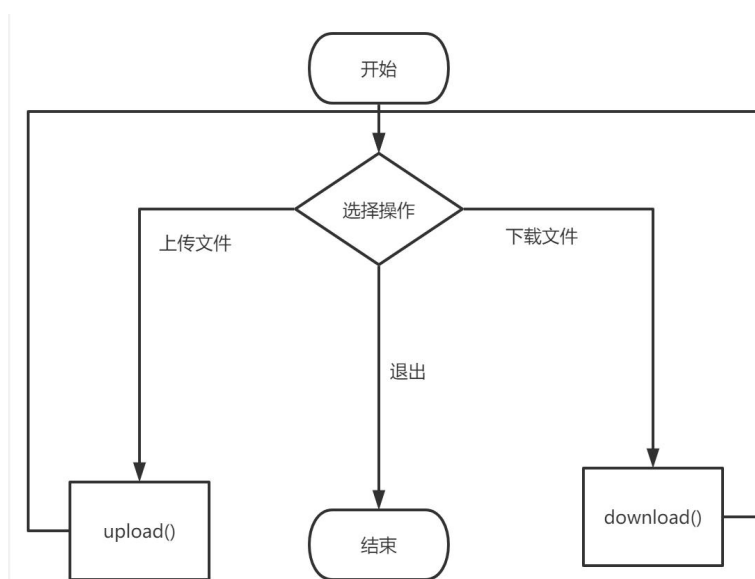


图 2-1 系统模块框图

模块关系

对于 upload 和 download 两大模块从服务器下载文件和向服务器上传文件，这两个大模块是并列的，但其中封装调用的内部函数是有重叠的，两个模块都要先向服务器的 69 号端口发起连接（伪连接）请求，而后收到请求响应后再执行对应的操作。

两个模块下的基本功能划分如下图 2.2 所示

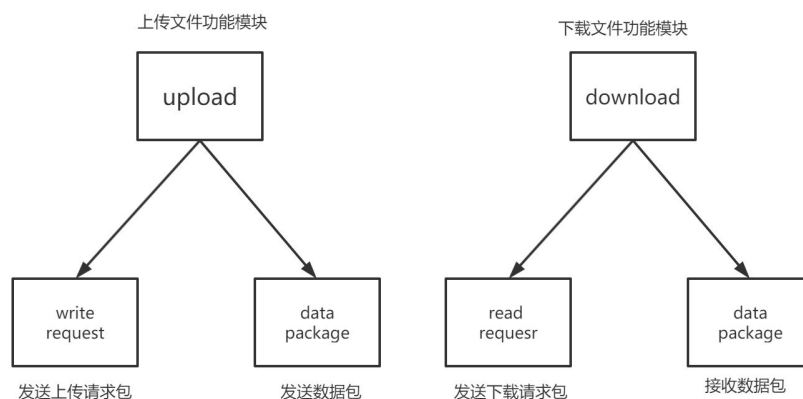


图 2-2 模块关系

接口函数

本程序只是用了三个接口函数，函数内的具体设计将在 2.2 节进行阐述，以下表格就是核心函数的参数、返回值和功能：

表 2-1 接口函数

函数名	输入参数	返回值	功能
setnonblocking	sockfd: 创建后的 socket 的标识字	无	设置 socket 传输为非阻塞模式
upload	filename: 要上传的文件名	布尔型: 是否上传成功	从本地上传文件到服务器
download	remotefile: 服务器上的文件名 loaclname: 下载到本地后重命名的文件名	布尔型: 是否下载成功	从服务器下载文件到本地

2.1.2 数据处理流程

上传数据流程

首先在主程序中使用 WSASStartup 函数将 Winsock 初始化，输入服务器和客户端的 IP（DEBUG 模式下均默认使用本地 IP），服务器端口为 69，客户端端口设置为 0，自动分配，然后创建客户端 socket，使用 bind 函数将 socket 与客户端自动分配的端口绑定，使用 ioctlsocket 函数指定使用非阻塞模式，然后输入对应操作序号和要上传的文件名后进入 upload 函数模块中。

在 upload 函数中，首先选择传输使用的模式 netascii 或者 octet，编写 WRQ 请求包，操作码为 2，文件名和传输模式均依次写入 sendPacket 包内，发送 WRQ 请求包，发送后每隔 20ms 进行接收 ACK 包操作，如果未接收到继续等待 20ms 再接收，直到等待 3s 后仍未接收到或者是成功接收到 ACK 包结束当前操作，如果是 3s 后仍未接收到则会选择重传 WRQ 包，如果重传 3 次仍未获取到 ACK 包则结束该次传输，如果成功接收到 ACK 包则进入上传文件环节。

在传输文件环节中，如果是使用 `netascii` 模式则使用 `r` 读取文件，`octet` 模式则使用 `rb` 读取文件，开始计时，进入循环，写入块号、使用 `fread` 将数据读入数据包的数据区，每个包都最多进行 3 次重传，发送包后每隔 20ms 接收 ACK 包一次，3s 后未接收到则进行超时重传操作，接收到后根据 ACK 反馈的块编号决定是否重传或进行下一块数据包的传输。所有数据包发送成功后停止计时，进行传输数据大小、传输时间、传输速率、掉包率的计算，并输出在终端中，给予用户反馈，至此结束本次传输。

下载数据流程

与上传数据类似，主程序中操作相同，输入下载文件操作序号并输入文件名后进入 `download` 模块，。

在 `download` 函数中，选择传输使用的模式 `netascii` 或者 `octet`，然后编写 RRQ 请求包，操作码为 1，文件名和传输模式写入其中，发送给服务器，发送成功后就进入接收数据发送 ACK 的下载文件环节。

在下载文件环节中，如果是使用 `netascii` 模式则使用 `w` 写入文件，`octet` 模式则使用 `wb` 写入文件。计时开始，进入循环，循环正常结束条件为接收到的文件数据大小与要下载的文件大小相同，每隔 20ms 尝试接收数据包，最多等待 3s，3s 后未接收到则发送重复的 ACK 包给服务器，最多重发 3 次 ACK，若 3 次后仍未接收到数据包则断开连接，终止传输，若接收到 ACK 包，且数据包与需求的包块号相等，则发送 ACK 给服务器，表明已收到，并将接收到的数据使用 `fwrite` 写入文件，直到数据传输完成，停止计时，计算文件大小、文件传输时间、传输速率、掉包率，并输出到终端据给予用户反馈，至此下载传输完成。

2.2 详细设计

文件上传

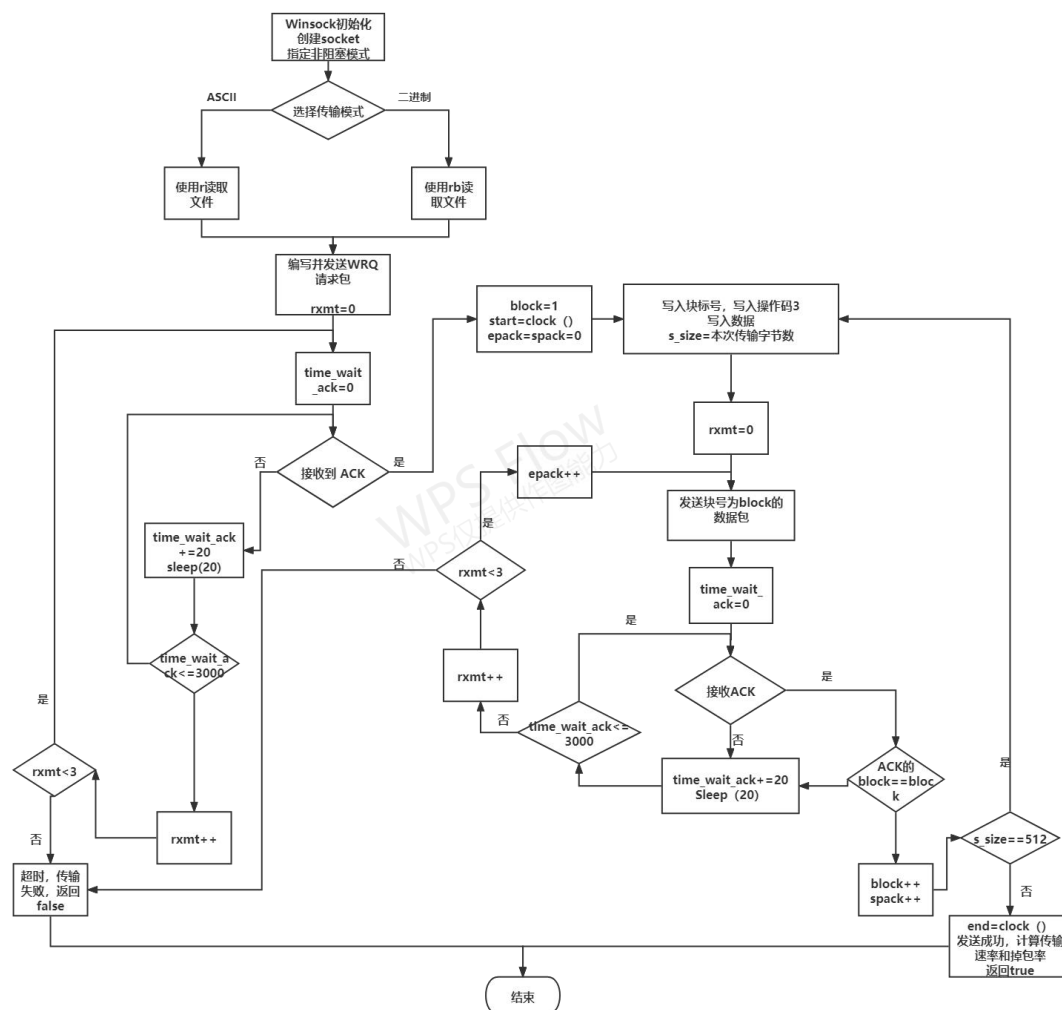


图 2-3 upload 函数流程图

说明：

上图 2.3 即为上传文件的整个代码实现过程，在用户进行功能选择前初始化 Winsock 并根据地址设置客户端的 socket，同时使用 bind() 完成 socket 的绑定。客户端向服务器发起写请求数据包，用户根据服务器返回的数据包进行相应的操作。

如果用户在设定时间内接收到服务器端的确认报文，那么就生成下一块数据包并发送给服务器，同时将接收到的确认报文信息和发送的数据包信息写日志文件中，更新数据吞吐量。

如果用户在设定时间内还未接收到服务器的返回报文（确认报文或错误报文），那么进行重传机制，将上一次发送的数据包再次发送给服务器，同时在日志文件中记录进行的动作，重传机制设定最多重传三次，否则将终止传输。

如果接收到错误包，那么打印出错误信息并将错误信息记录在日志中，同时断开连接（伪连接），关闭 socket。

文件下载

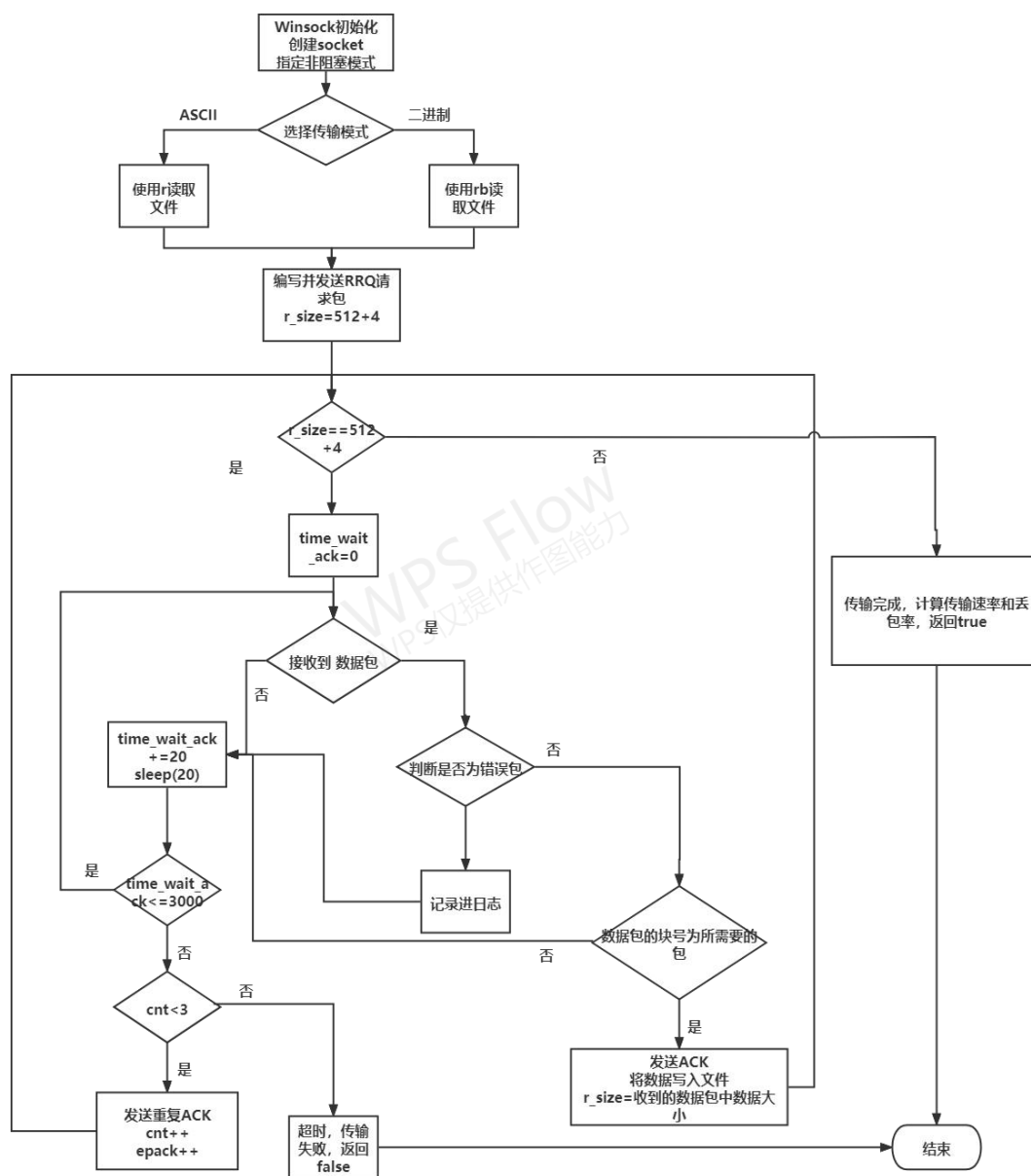


图 2-4 download 函数流程图

说明：

下载文件的代码实现流程图如上图 2.4 所示，下载文件和上传文件一样，对于从服务器端接收的数据包进行解析，分析数据包的类型做相应的动作即可，唯一不同之处在于在成功发送请求包后不会等待来自服务器 ACK，而是直接进入等待接收数据包的环节。

2.3 代码实现

上传文件（upload）伪代码

```

1  bool upload (char* filename){
2      /*变量定义与初始化*/
3      rxmt=0; //同一个数据包重传次数的循环控制变量, rxmt<3
4      spack=0,epack=0;//spack为成功传输包的数量 epack为传输失败（丢包）的数量
5      start,end;//start为开始传输的时间, end为结束传输的时间
6      transByte=0;//记录传输的字节数
7      time_wait_ack;//time_wait_ack为等待ACK包的时间控制变量, <3000ms
8      r_size;//r_size为本次接收的数据包的大小, 小于4说明为错误包, 大于等于4才可能是ACK包
9      s_size;//s_size为本次发送的数据包的大小, 默认最后一个数据包小于512字节
10     block=1;//块号
11     choose;//传输模式, =1为ascii =2为二进制
12     sendPacket,rcv packet;//前者为发送包, 后者为接收包
13
14     /*发送WRQ请求包*/
15     sendPacket写入操作码2和文件名、传输方式
16     for rxmt=0 to 2
17         rxmt++
18         发送WRQ请求包
19         发送失败:
20             epack++//发送失败的包累加
21             continue 结束本次循环
22         成功发送:
23             spack++ //发送成功的包累加
24             for time_wait_ack=0 to 3000
25                 time_wait_ack+=20 //每隔20ms尝试接收ACK
26                 接收到来自服务器的包
27                 是坏包:
28                     输出接受包异常, 继续等待包
29                 是ACK包且ACK确认的是上次发送的数据包块号:
30                     break//结束time_wait_ack控制的循环
31             未接收到包
32                 sleep(20)//等待20ms再尝试接收
33             在3s内即上述time_wait_ack循环中已经成功接收到ACK包则break
34             否则:
35                 epack++//丢包数累加
36                 相关信息输入到日志中
37     上述rxmt控制的循环中未成功接收到ACK:
38     将错误信息写入日志
39     return false //结束传输过程, 关闭socket
40
41     /*数据传输*/
42     根据传输方式采用r/rb读取文件
43     如果读取文件失败:
44         将错误信息写入日志
45         return false //结束传输过程, 关闭socket
46     start=clock() //开始计时
47     do{
48         sendPacket写入块编号, 读入数据
49         s_size更新为本包的数据大小
50         transByte+=s_size
51         for rxmt=0 to 2
52             rxmt++
53             发送数据包
54             发送失败:
55                 epack++//发送失败的包累加
56                 continue 结束本次循环
57             成功发送:
58                 spack++ //发送成功的包累加
59                 for time_wait_ack=0 to 3000
60                     time_wait_ack+=20 //每隔20ms尝试接收ACK
61                     接收到来自服务器的包
62                     是坏包:
63                         输出接受包异常, 继续等待包
64                     是ACK包且ACK确认的是上次发送的数据包块号:
65                         break//结束time_wait_ack控制的循环
66                 未接收到包
67                     sleep(20)//等待20ms再尝试接收
68                 在3s内即上述time_wait_ack循环中已经成功接收到ACK包则break
69                 否则:
70                     epack++//丢包数累加
71                     相关信息输入到日志
72     上述rxmt控制的循环中未成功接收到ACK:
73     将错误信息写入日志
74     return false //结束传输过程, 关闭socket
75     block++ //块号增加, 即将进入下一块的传输
76     while(s_size==512)
77     end=clock()//结束计时
78     //如果刚传输的包中数据仍未512说明非最后一个包,
79     //不等于512说明是最后一个包, 则完成传输, 跳出循环
80
81     /*本次传输信息的输出*/
82     传输时间=(end-start)/CLK_TCK
83     文件大小为transByte/1024 kB
84     传输速率为transByte/1024/传输时间 kB/s_size
85     掉包率=epack/(epack+spack) *100 %
86     return true
87

```

图 2-5 upload 函数伪代码

下载文件（download）伪代码

```

1  bool download(char* remotefile,char* localfile)
2      /*变量定义与初始化*/
3      cnt=0;//重传次数控制变量，<3
4      spack=0,epack=0;//spack为成功传输包的数量 epack为传输失败（丢包）的数量
5      start,end;//start为开始传输的时间，end为结束传输的时间
6      transByte=0;//记录传输的字节数
7      time_wait_data;//time_wait_data为等待数据包的时间控制变量，<3000ms
8      r_size;//r_size为本次接收的数据包的大小
9      block=1;//块号
10     choose;//传输模式，=1为ascii =2为二进制
11     sendPacket,rcv_packet,ack;//前者为发送包，后者为接收包
12
13     /*发送RRQ请求包*/
14     start=clock();//开始计时
15     sendPacket写入操作码1和文件名、传输模式
16     发送sendPacket包
17         发送失败：
18             epack++
19             重新发送，最多重新发送3次，3次后均失败则直接终止传输 return false
20         发送成功：
21             spack++
22     /* 接收数据包 */
23     根据传输模式使用w/wb写入文件
24     如果打开文件失败则将错误写入日志，终止传输 return false
25     do{
26         for time_wait_data=0 to 3000
27             time_wait_data+=20 //每隔20ms尝试接收数据包
28             接收到来自服务器的包
29             是坏包：
30                 输出接受包异常，继续等待包
31             是操作码为data包且该包的block为需要接收的block
32                 cnt=0
33                 spack++ //成功的包累加
34                 编写ACK确认包发送服务器
35                 对接收到的数据写入文件中
36                 break//结束time_wait_data控制的循环
37             未接收到包
38                 sleep(20)//等待20ms再尝试接收
39             3s内未接收到数据包，且是第一个数据包未接受到则直接break退出do循环
40             3s内未接收到数据包，且非第一个数据包并且cnt<3：
41                 cnt++//即进行第一次重传
42                 重传上一个block的ACK确认包给服务器
43                 continue; //退出本次循环，进行等待接收ACK
44             3s内未接收到数据包且cnt==3： //意味着三次重传ACK后均未接收到新的数据包
45                 连接超时，记录相关错误信息在日志中
46                 终止传输
47                 return false
48             transByte+=(r_size-4) //-4 是减去前面的操作码占用的字节，获得数据块大小
49             block++; //进入下一块的等待中
50     while(r_size==512+4) //默认最后一个数据包的大小是小于512+4的，如果等于说明传输未完成，继续循环
51     end=clock()//结束计时
52
53     /*本次传输信息的输出*/
54     传输时间=(end-start)/CLK_TCK
55     文件大小为transByte/1024 kB
56     传输速率为transByte/1024/传输时间 kB/s_size
57     掉包率=epack/(epack+spack) *100 %
58     return true

```

图 2-6 download 函数伪代码

三、 实验测试与分析

3.1 系统测试及结果说明

3.1.1 测试环境

windows 10

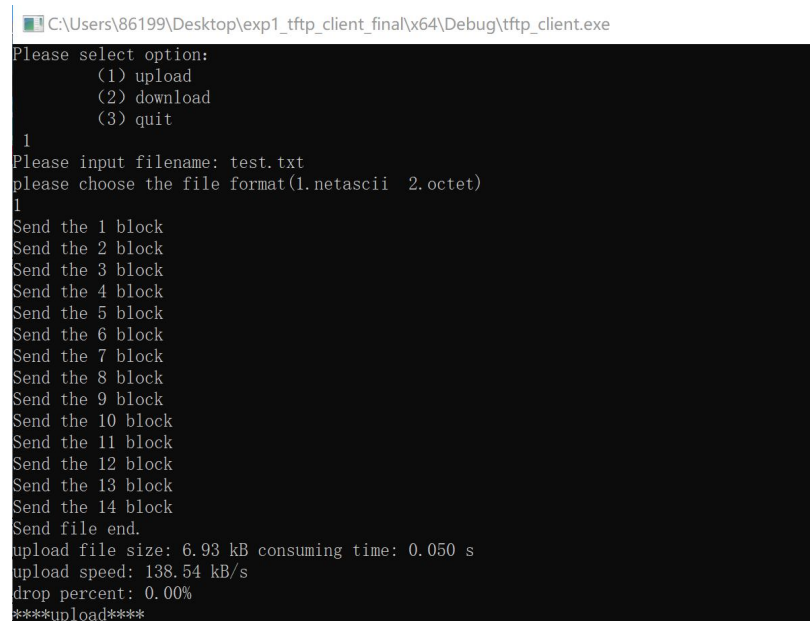
vs2019 x64

3.1.2 测试方案

1. 不开启 clumsy 应用上传文件（netascii）
2. 不开启 clumsy 应用上传文件（octet）
3. 不开启 clumsy 应用下载文件（netascii）
4. 不开启 clumsy 应用下载文件（octet）
5. 开启 clumsy 应用上传文件
6. 开启 clumsy 应用下载文件

3.1.3 测试过程

1. 不开启 clumsy 应用上传文件（netascii）



```
C:\Users\86199\Desktop\exp1_tftp_client_final\x64\Debug\tftp_client.exe
Please select option:
    (1) upload
    (2) download
    (3) quit
1
Please input filename: test.txt
please choose the file format(1.netascii 2.octet)
1
Send the 1 block
Send the 2 block
Send the 3 block
Send the 4 block
Send the 5 block
Send the 6 block
Send the 7 block
Send the 8 block
Send the 9 block
Send the 10 block
Send the 11 block
Send the 12 block
Send the 13 block
Send the 14 block
Send file end.
upload file size: 6.93 kB consuming time: 0.050 s
upload speed: 138.54 kB/s
drop percent: 0.00%
***upload***
```

图 2-7 upload（netascii）

说明：选择 1（upload）功能，上传 test.txt 文件，模式为 netascii，显示成功传输，共 14 个包

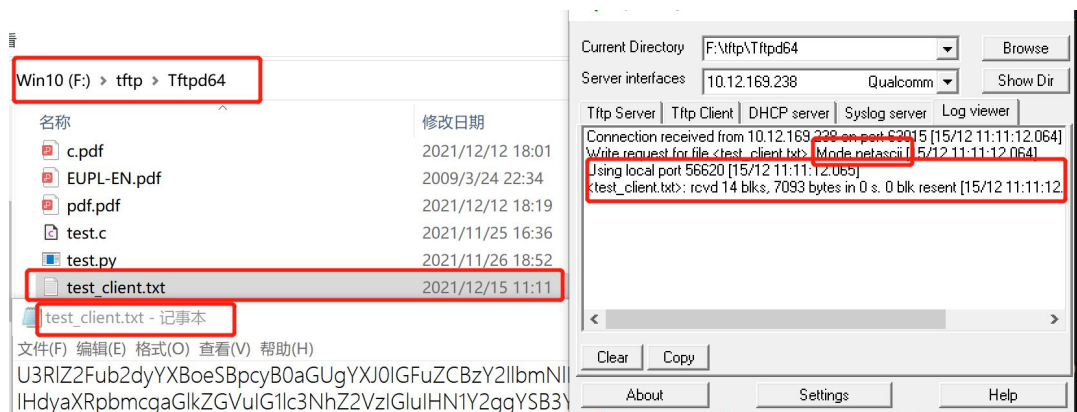


图 2-8 upload（netascii）查看传输的文件

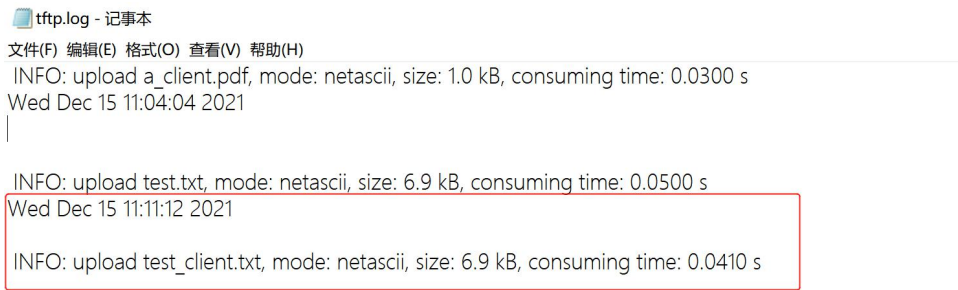


图 2-9 upload（netascii）日志记录

2. 不开启 clumsy 应用上传文件（octet）

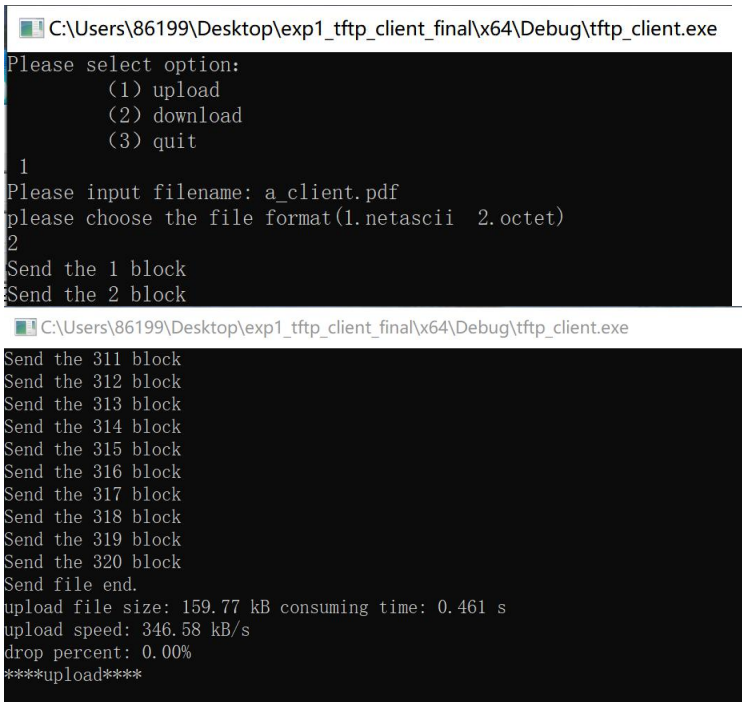


图 2-10 upload（octet）

说明：选择 1（upload）功能，上传 a_client.pdf 文件，模式为 octet，显示成功传输，共 320 个包

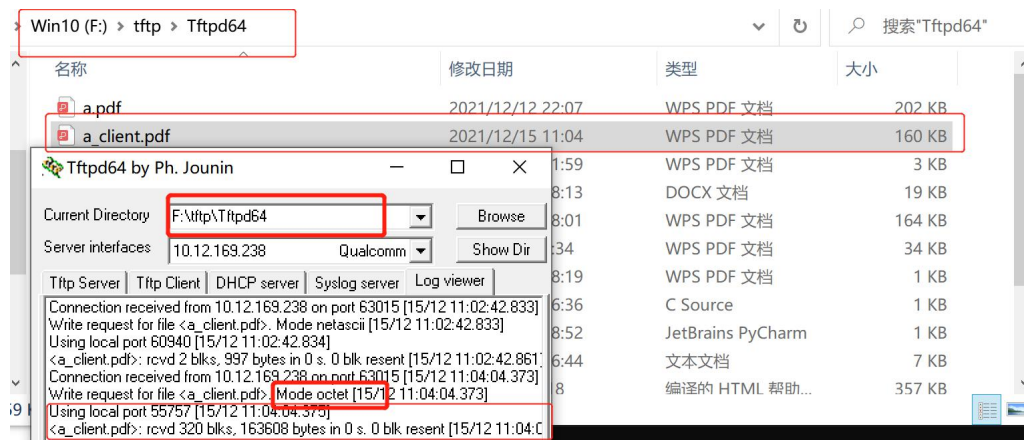


图 2-11 upload（octet）查看文件

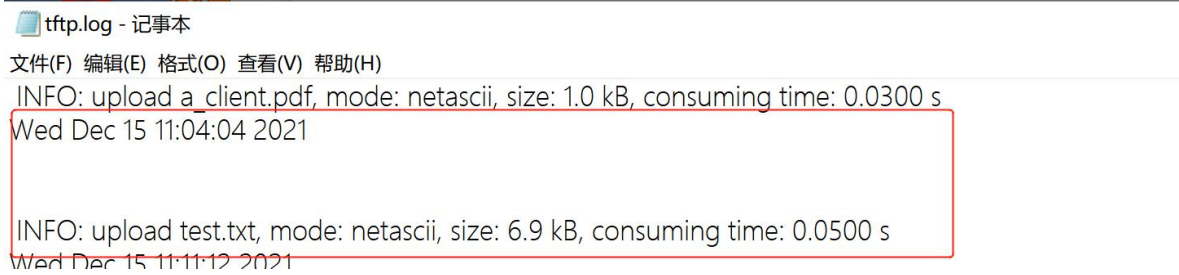


图 2-12 upload（octet）日志记录

3. 不开启 clumsy 应用下载文件（netascii）

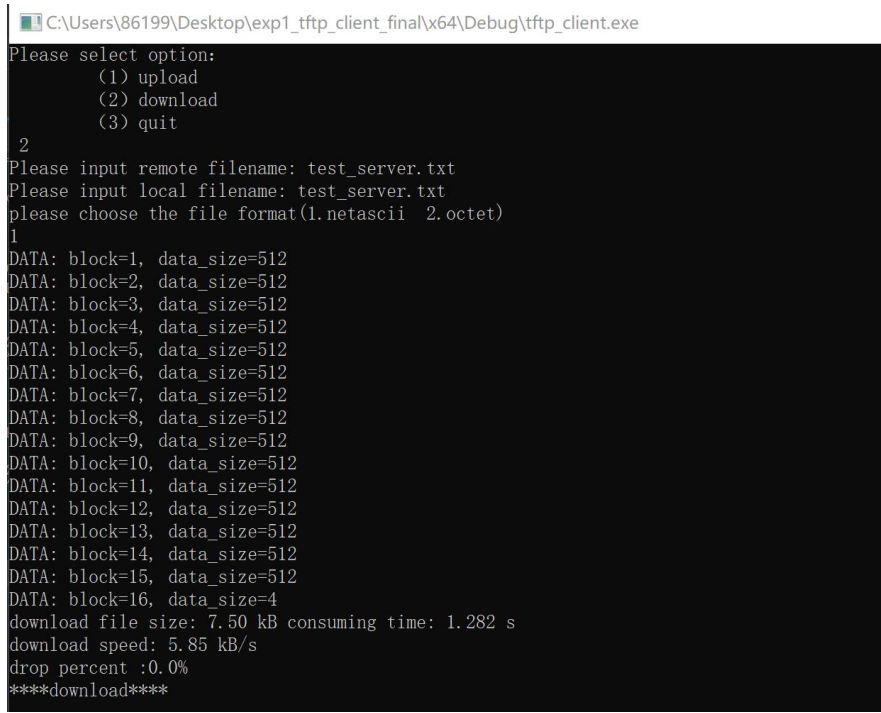


图 2-13 download（netascii）

说明：选择 2（download）功能，下载 test_server.txt 文件，模式为 netascii，显示成功传输，共 16 个包

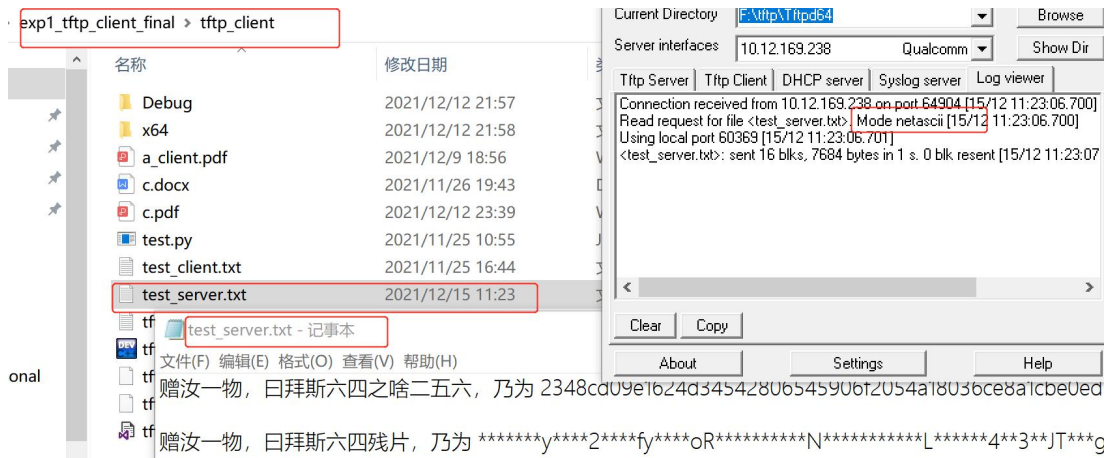


图 2-14 download（netascii）查看文件

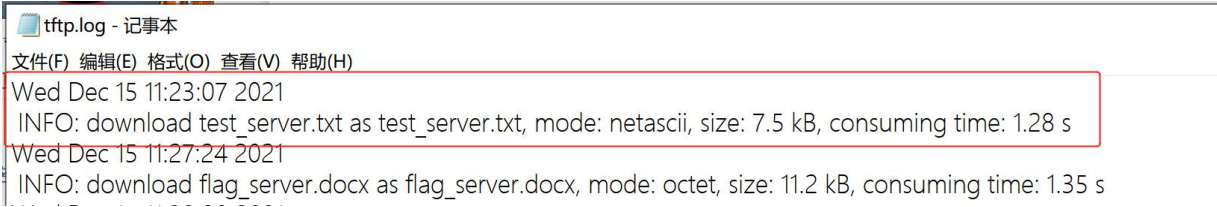


图 2-15 download（netascii）日志记录

4. 不开启 clumsy 应用下载文件（octet）

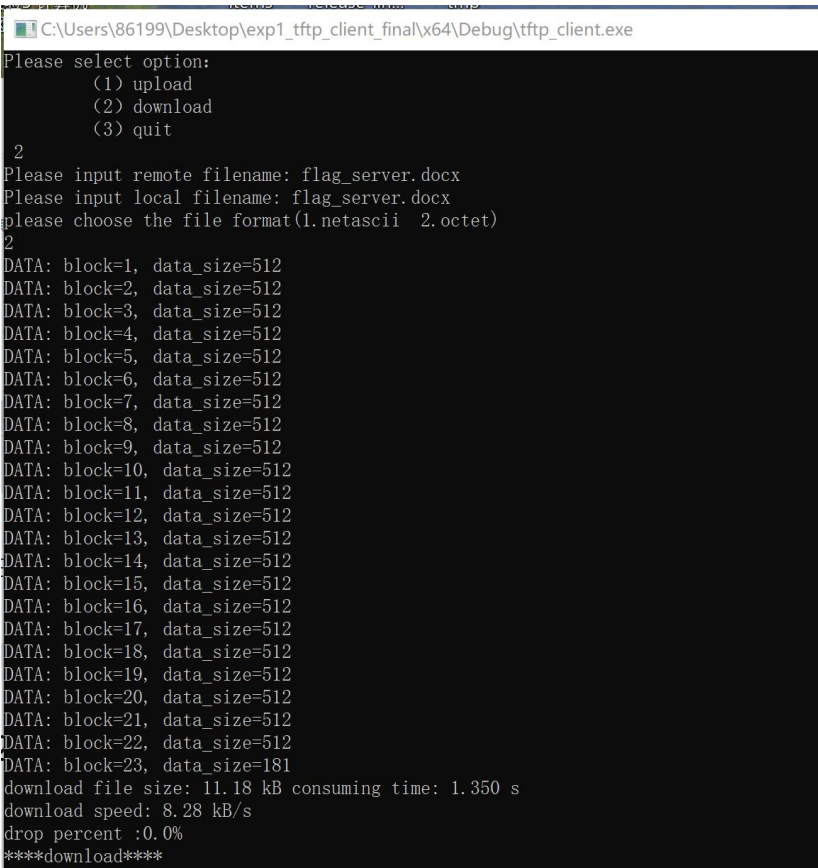


图 2-16 download（octet）

说明：选择 2（download）功能，下载 flag_server.docx 文件，模式为 octet，显示成功传输，共 23 个包

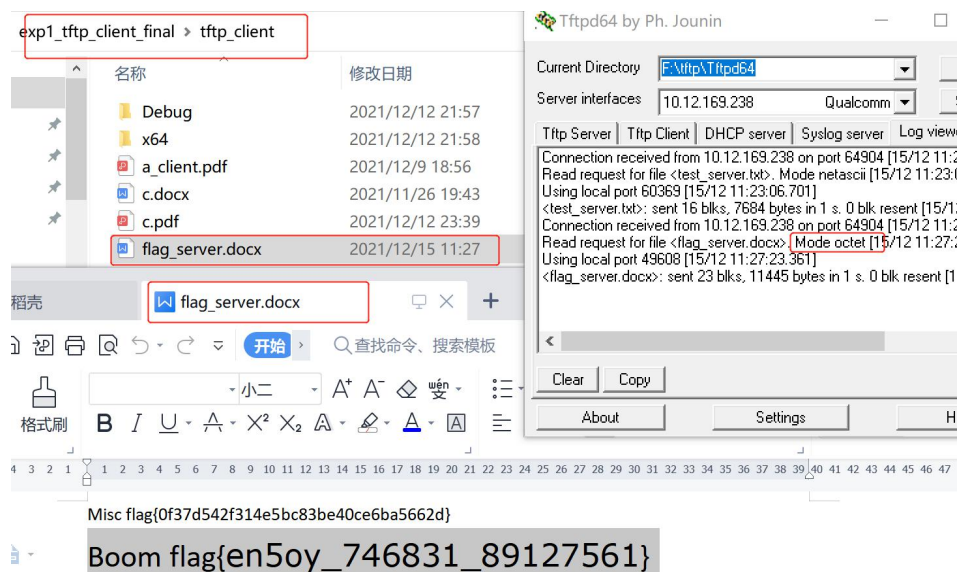


图 2-17 download (octet) 查看文件

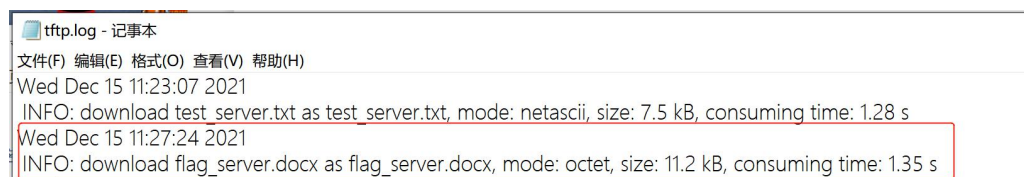


图 2-18 download (octet) 日志记录

5. 开启 clumsy 应用上传文件

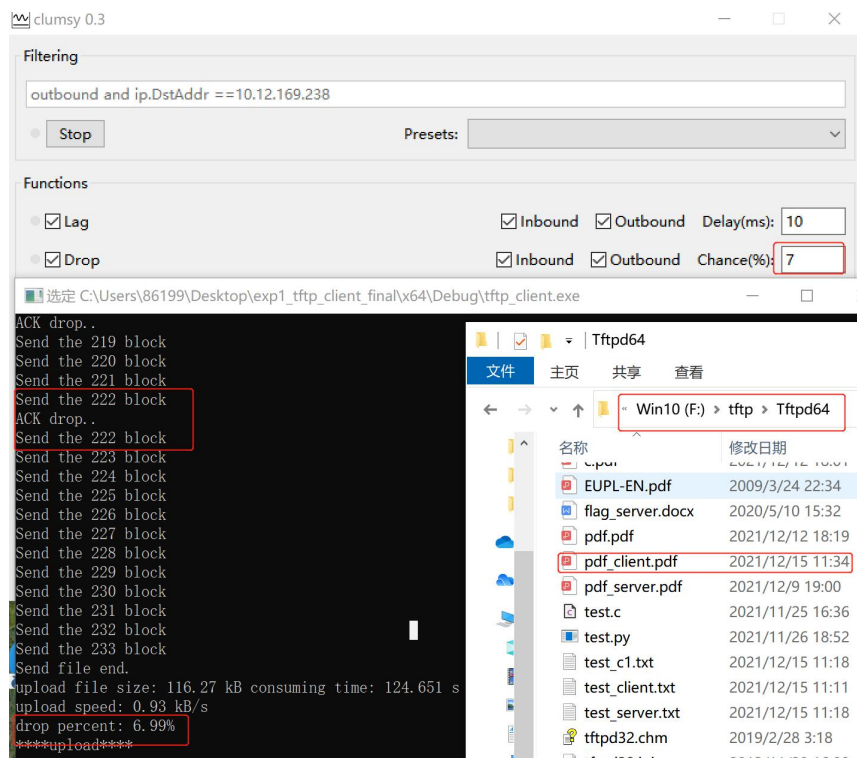
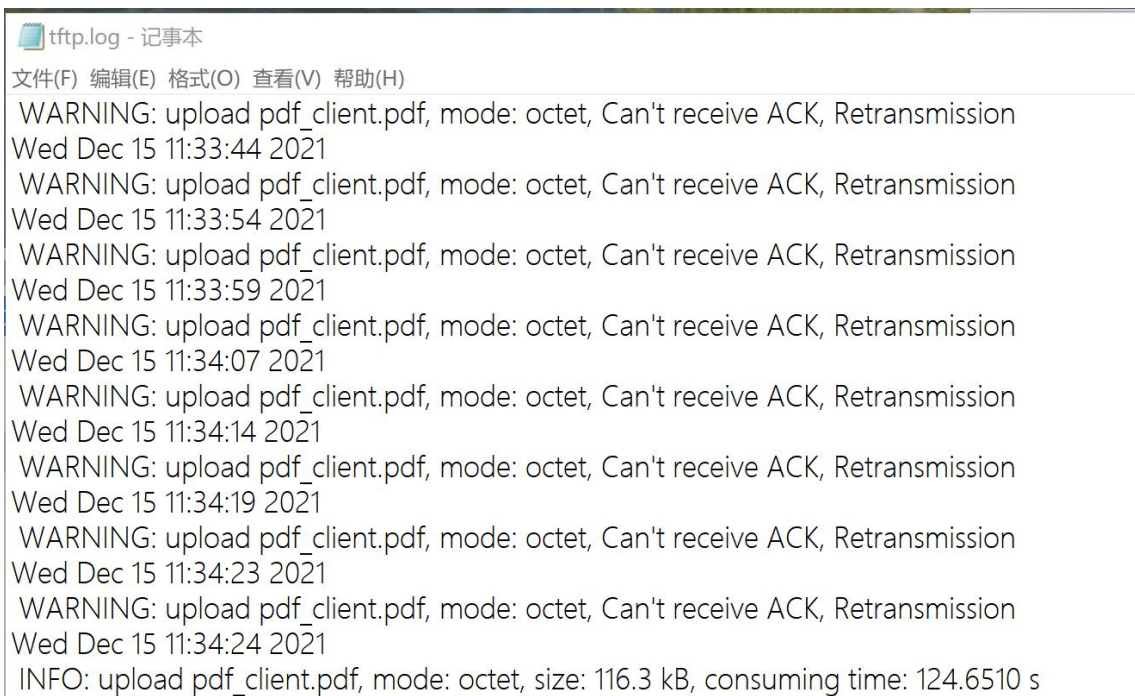


图 2-19 开启延时和掉包后 upload

说明：在 clumsy 中开启的掉包率为 7%，代码运行后计算为 6.99%，在运行中可见 ACK drop... 提示，说明该包的 ACK 包未接收到，又重发该数据包（图中为 222 块号），开启延时也能看到传输速率明显降低

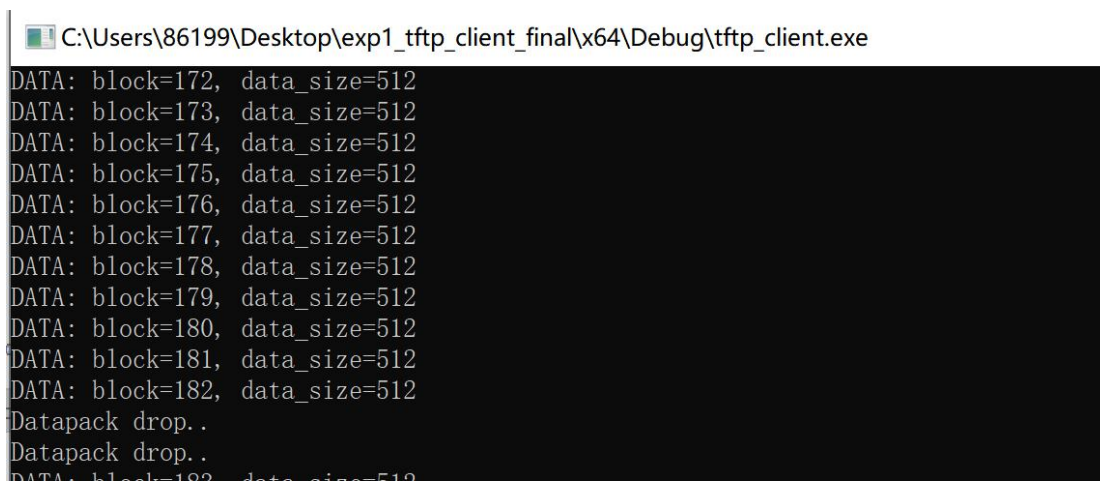


```
tftp.log - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
WARNING: upload pdf_client.pdf, mode: octet, Can't receive ACK, Retransmission
Wed Dec 15 11:33:44 2021
WARNING: upload pdf_client.pdf, mode: octet, Can't receive ACK, Retransmission
Wed Dec 15 11:33:54 2021
WARNING: upload pdf_client.pdf, mode: octet, Can't receive ACK, Retransmission
Wed Dec 15 11:33:59 2021
WARNING: upload pdf_client.pdf, mode: octet, Can't receive ACK, Retransmission
Wed Dec 15 11:34:07 2021
WARNING: upload pdf_client.pdf, mode: octet, Can't receive ACK, Retransmission
Wed Dec 15 11:34:14 2021
WARNING: upload pdf_client.pdf, mode: octet, Can't receive ACK, Retransmission
Wed Dec 15 11:34:19 2021
WARNING: upload pdf_client.pdf, mode: octet, Can't receive ACK, Retransmission
Wed Dec 15 11:34:23 2021
WARNING: upload pdf_client.pdf, mode: octet, Can't receive ACK, Retransmission
Wed Dec 15 11:34:24 2021
INFO: upload pdf_client.pdf, mode: octet, size: 116.3 kB, consuming time: 124.6510 s
```

图 2-20 开启延时和掉包后 upload 的日志记录

说明：日志中记录了 ACK 未接收到后重发数据包的情况

6. 开启 clumsy 应用下载文件



```
C:\Users\86199\Desktop\exp1_tftp_client_final\x64\Debug\tftp_client.exe
DATA: block=172, data_size=512
DATA: block=173, data_size=512
DATA: block=174, data_size=512
DATA: block=175, data_size=512
DATA: block=176, data_size=512
DATA: block=177, data_size=512
DATA: block=178, data_size=512
DATA: block=179, data_size=512
DATA: block=180, data_size=512
DATA: block=181, data_size=512
DATA: block=182, data_size=512
Datapack drop..
Datapack drop..
DATA: block=183, data_size=512
```

图 2-21 开启延时和掉包后 download 中两次掉包记录

说明：传输过程中出现了某一个块号数据包两个 3s 内未接收到的情况

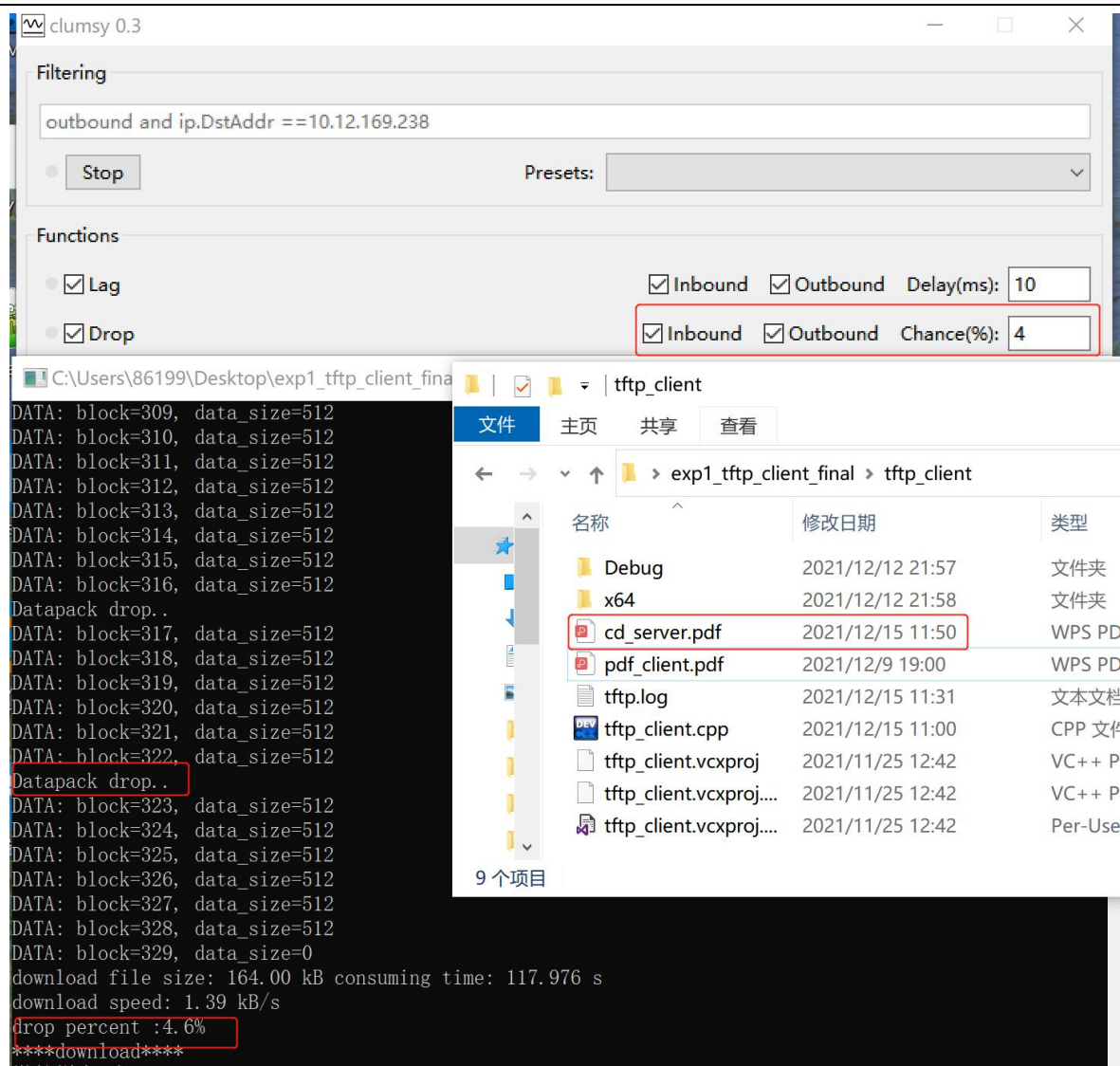


图 2-22 开启延时和掉包后 download

说明: clumsy 掉包率设置成 4%, 程序运行检测到的是 4.6%, 传输过程可见 Datapack drop... 的提示

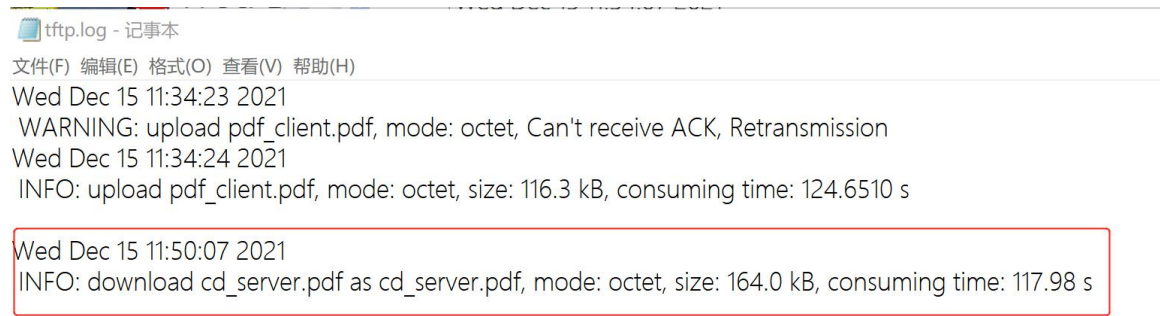


图 2-23 开启延时和掉包后 download 的日志记录

3.2 遇到的问题及解决方法

1. 在丢包重发问题上起初并没有什么思路，后来受老师提示的非阻塞模式和同学帮助想到可以每隔 20ms 进行一次尝试接收，最多等待 3s，若还未接收到 ACK 或 data 包则进行重发，根据理论课上学习到的 3 个冗余 ACK，选择了最多重发 3 次，并在此期间记录丢包数，一并解决了丢包率计算问题。

2. 另外一个问题是对接口函数参数的具体用法不理解起初写 download 函数时，始终无法收到服务器的回应，后来通过查阅资料发现 sendto 函数的第二个参数一定要是实际发送给服务器的报文大小，而不是数组的大小。

3.3 设计方案存在的不足

1. 丢包率的计算上我仍认为有瑕疵，与 clumsy 软件给出的数据以及 tftp64 软件给出的数据还是有所出入，但是原因暂未找到

2. 对于在处理过程中会出现的一些 error 处理不尽详实，传输过程中的错误仅大体给出了提示，但是具体错误并未给出

四、 实验总结

4.1 实验感想

socket 实验中对 Winsock 中内置的几个 API 函数有了比较深刻的了解，关键是在此之前从未接触过这一方面知识，在认真学习了指导书后有了大体思路，但是具体实现上还是困难重重，主要是虽然相关理论课已经学习过，但是理解不到位，tftp 的实现原理还是有些困惑的，在实验的过程中查阅了很多的资料，基本是遍调试边摸索，完成的初稿只实现了基本功能，但是对于丢包率的计算还是有困难，因为对丢包的这个概念还是拿捏不准，最后在同学的帮助下完成了终稿，虽然代码还是很短小的，但是整个过程还是非常艰难的，但是值得庆幸的是我完成任务后回过头复盘自己的思路发现自己对 tftp 的理解和其他各种传输协议有了更深的了解，于我而言，这次实验是成功的。

实物实验是最有意思的，第一次课自制网线了解了相关知识，第二次实验中与同伴进行实物的连接，做之前觉得挺难的，老师刚刚讲完相关理论课很多原理还不理解，但是对照老师的实验指导书，加上与同伴的交流，历经 4 个小时也是成功实现了各主机互 ping，最有成就感的时候还是当自己的笔记本连上了局域网，ping 通实验室主机的时候，非常有趣又实用，更进一步加深了相关知识的理解。

第三个实验由于检查时间比较靠前，完成的仓促，多亏还有老师的指导视频，一点点琢磨每一步的用意，vlan 的配置，ip 的划分还有静态路由的设置，这个实验收获最大的就是对交换机的 vlan 划分和静态路由的配置，也学习到了查看路由表。

其实三次实验都是具有挑战性的，做的时候觉得真的很难，做完后回头复盘又有柳暗花明又一村的感觉，同时也是真的学到了很多实用的知识，理论知识指导我完成实践，通过实践又融会贯通了理论知识，我想这才是实验的意义，最后要特别感谢老师提供的实验指导书和指导视频，以及耐心的教导和讲解！

4.2 意见和建议

1. 建议实物实验开始前老师着重强调一下密码修改问题，某些同学乱改密码，在重置密码上真的浪费了非常多的时间。

2. 能否考虑换一个软件代替 eNSP，这个软件体验感极差，经常会卡死和出现一些未知错误，有时候还未保存就卡死了，一切又要从头开始，有时候保存了也会不完整，重新打开还有修改一些东西。

3. 另外感觉第三个实验检查时间有些早，做起来有些仓促。

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：李欣宇