****

**《计算机网络LFTP》**

**实验报告**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学院名称** | **：** | 数据科学与计算机学院 | | | | | |
| **学生姓名1** | **：** | 张星 16340294 | | | | | |
| **学生姓名2** | **：** | 姚雪辉 15355119 | | | | | |
| **专业（班级）** | **：** | 16软件工程教务四班 | | | | | |
| **时间** | **：** | 2018 | 年 | 12 | 月 | 3 | 日 |

目录

[实验设计 2](#_Toc531719430)

[**LFTP未完成** 2](#_Toc531719431)

[**LFTP新开** 3](#_Toc531719432)

[**实现功能** 3](#_Toc531719433)

[使用说明 4](#_Toc531719434)

[测试结果 4](#_Toc531719435)

[代码模块 9](#_Toc531719436)

[使用说明 10](#_Toc531719437)

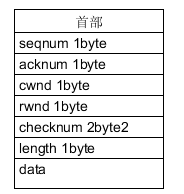
[实验分工 10](#_Toc531719438)

[README 10](#_Toc531719439)

# 实验设计

## **LFTP未完成**

我们的项目有两个部分，一个是LFTP未完成，一个是LFTP新开。一开始我们打算按照要求实现所有的要求，包括100%可靠的传输，流量控制，拥塞控制。具体的部分是新加头文件，虽然



UDP自身有校验和的部分，但是由于新加了首部的内容，还是做了校验和的部分，这样当接收方缓存满的时候，发送rwnd给发送方，然后发送方改成发送1bit的数据给接收方，直到接收方清空缓存再恢复，以此实现流量控制。通过cwnd的值，实现慢启动以及拥塞避免。但是由于项目启动过晚，以及多线程的编程问题（主要是计时器调试没法完成），没有办法完成，所以最后项目做了一半。改成停等协议，没有办法实现流水线，也就没有办法实现流量控制，以及拥塞控制，此部分的项目都保存在LFTP未完成文件夹中。虽然没有办法完成，但是部分文件还是可以沿用的，所以虽然是停等协议，还是多做了其他功能。

## **LFTP新开**

在LFTP新开的文件夹中是已经完成的完整的项目。但是采用了更为简单的停等协议，即发送方发包，接收方收到包并确认正确无误后，发一个正确的ACK，发送方收到ACK之后继续发包。我们用的是局域网测试，速度很快，但是没有丢包，所以手动设置了第二个数据包丢包，再截图里面会体现，其实如果用流水线，如果不用选择重传，而采用回退N步，不见得会比停等协议更加有效率，因为也是收到数据ACK之后，滑动窗口才会移动，所以虽然发了很多数据包，但是一旦顺序乱了，还是要从头开始重传，所以效率不是很好，这也是我们放弃流水线的一点原因。其中头文件还是沿用之前的头文件，所以虽然停等协议对seqnum没有要求，我们还是做了seqnum的处理，序号从1-127，而不是简单的0和1。由于Socket自带的setTimeOut函数，可以方便实现计时器，我们设置一秒为超时时间，超过一秒发送方未收到ACK。当然收到错误的数据包也会重传，我们沿用之前的首部数据，使用checksum，接收方首先将文件读入buff，然后将buff作为参数传入该类，根据传入参数计算出checksum并扩展buff，将checksum放到buff的最后，将新的数据流作为packet的参数发送给接收方。接收方接到数据包之后，同样使用该类解析数据，先取出buff最后的checksum，然后计算前面的byte数据，是否与该checksum相加后结果为65535，若不等，则出错，发给发送方NAK。Checksum的计算依据书上的理论，将数据分为16比特，循环相加，回卷，最后结果取反即为checksum。

## **实现功能**

以上所讨论的都是协议的本身，除此之外我们实现的功能有，根据老师的要求有9个，除了第六条（流量控制）和第七条（拥塞控制）没有实现，其他要求都实现了。

输入命令是根据文件查找和正则表达式判断的，其中命令的前三部分是正则表达式实现的，由于链接的形式很多，用正则表达式实在没有办法判断，所以就直接判定是IP地址才可以，至于第四部分的文件，如果是发送文件，则会在本地查找文件，不存在就会提示命令错误，所以如果命令输入错的离谱，就会多次提示命令错误。

另外也有提供连接功能，首先是客户端发送SYN给服务端，服务端的端口接受到之后，会给该客户端分配一个端口，并告诉客户端，并以此实现多客户端同时上传或下载的功能。多客户端的并发使用，我们采用了服务端多线程来实现。具体思路类似于TCP连接，客户端首先通过命令，将连接请求发送到服务端，服务端受到请求命令后记录客户端的IP地址和端口，新开一个线程，将地址和端口号传入线程，然后线程新建一个socket，使用不同的端口号，给发送端发确认信息。发送方收到后，则会往新的线程发消息，建立连接，此后就可以进行通信了，并且不会出现互相干扰的情况。

上传与下载功能，我们在客户端与服务端可以通过不同命令，来进入到相应模块来实现，此处两端的代码相似。需要注意的是，客户端选择上传功能时，我们先将文件名发送给服务端，以此作为服务端接收文件存储在磁盘的文件名，所以最好不要输入文件路径，或者本机即是服务端又是客户端，会出错。

# 使用说明

输入命令都在客户端完成，服务端无此功能。

若要在一台PC上开多个客户端，则需要更改socket端口号，编译后运行，以免出现冲突。

输入命令前需要确认服务端已在目标地址运行。

**命令：LFTP lsend XXX.XXX.XXX.XXX 2018.mp4**

若本地存在名为2018.mp4的文件，则向该IP地址上传该文件，服务端接收该文件，并保存为2018.mp4.若本地不存在该文件，则报错，提示该文件不存在。

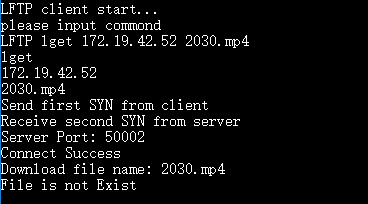
**命令：LFTP lget XXX.XXX.XXX.XXX 2018.mp4**

从服务端下载一个名为2018.mp4的文件，若文件不存在，则会发送一条消息提示客户端，否则正常下载，并保存为2018.mp4。

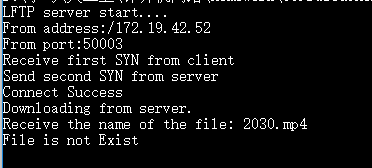
# 测试结果

首先测试客户端向服务端请求下载：服务端并无此文件，返回信息提示客户端。

客户端：

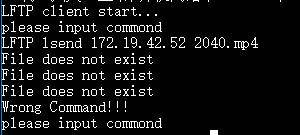


服务端：



接着测试从客户端向服务端发文件，此文件不存在，显示提示信息，重新输入命令。

客户端：有可能会出现多次报错的可能，是因为经过多层判断错误。

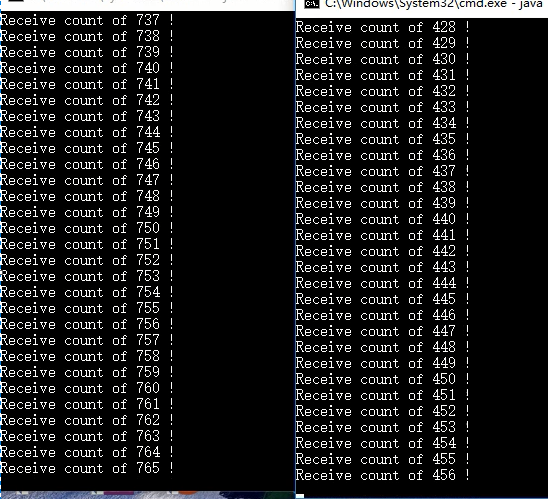


该IP地址不合法。

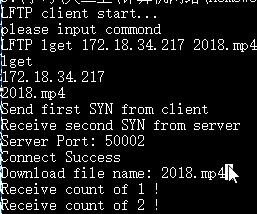
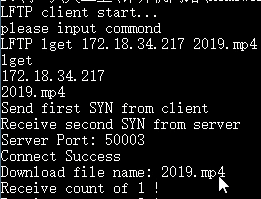


进行下载测试：首先在一台PC同时启动两个客户端，需要更改端口，重新编译才可正常工作。然后分别向服务端请求下载名为2018.mp4和2019.mp4的文件。

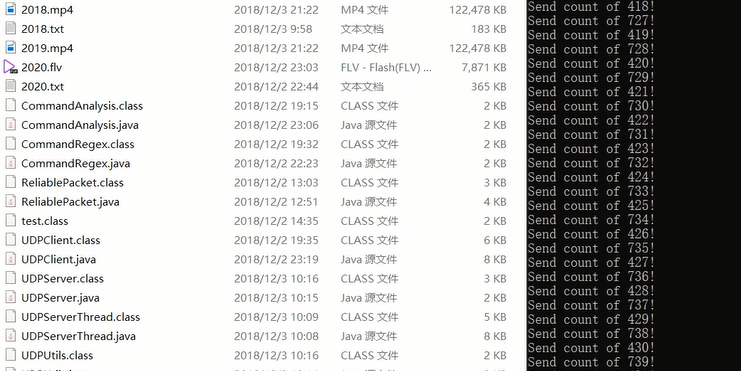
**（此过程Github中有视频，我们只是截取了一部分图片，完整过程可见Github.）**

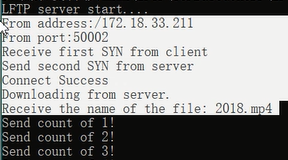
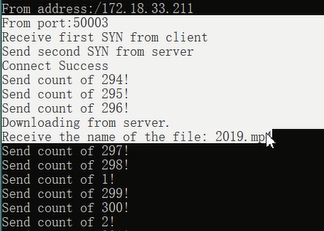


两者连接的服务端IP地址相同，端口号不同，这是因为处于不同的线程中。



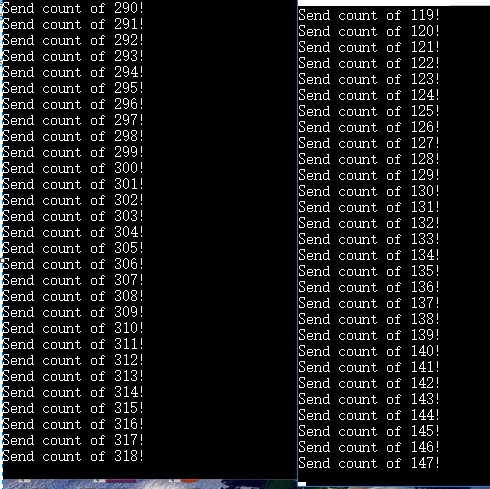
服务端收到请求，并开启两个不同线程进行传输，图中可看到交错顺序的序号，这是属于不同客户端的数据包。



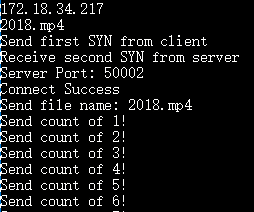
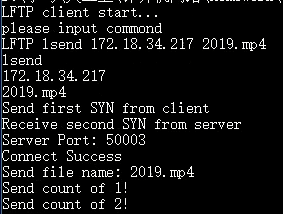


进行上传测试：同上，也是两个客户端程序同时向服务端上传文件，然后服务端进行接收。

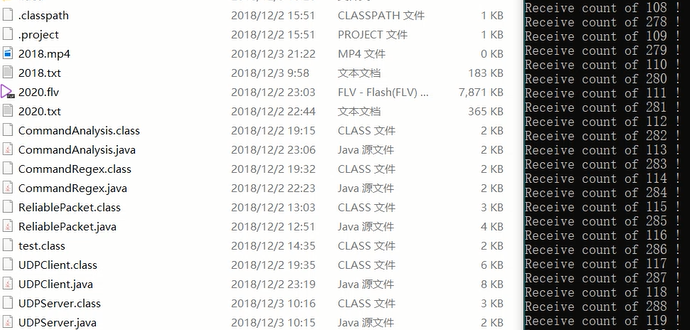
客户端：



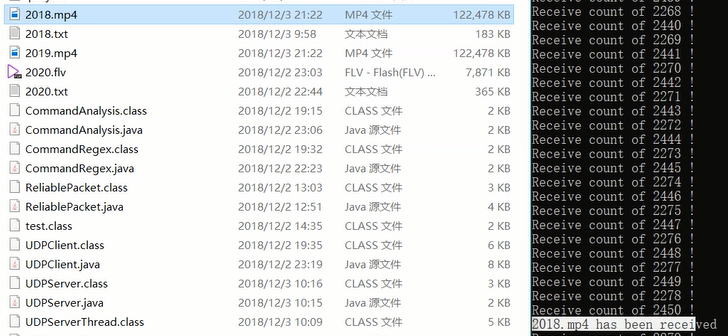
处于不同的端口。



服务端进行接收，可看到交替顺序的序号，是两个客户端同时在传输。

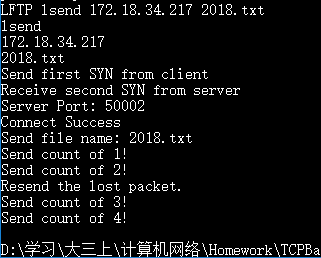


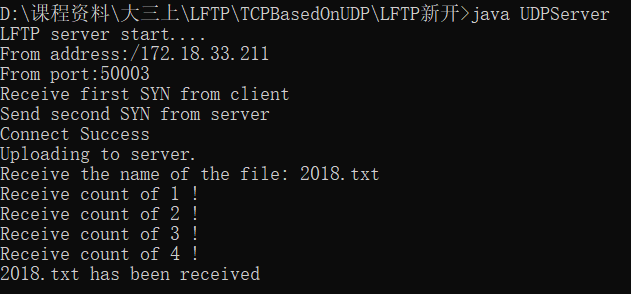
文件传输完毕，可以看到列表中出现了该文件。可以正常播放，播放过程详见GitHub视频。



丢包测试，由于我们是在校园网内进行测试，所以很难出现丢包的情况，所以我手动设置了一个丢包触发条件，进行测试。

如图：客户端上传是遇到了丢包，然后超时重传，服务端正常接收。





# 代码模块

本次实验我们采用JAVA语言。

ReliablePacket.java

该模块用于为UDP包生成检验和以及将sequenceNumber, rwnd, cwnd等信息插入数据流中，然后返回一个扩展的byte数组，依次作为参数发给另一端进行接收。另一端同样调用此类中的函数进行解析，将扩展的byte数组解析为原数据并进行数据包的校验。

UDPUtils.java

工具类，用于客户端与服务端都会用到的提示信息，以此作为当前连接状况的判断条件。

CommandRegex.java

用正则表达式判断命令输入正误。

CommandAnalysis.java

用文件查找判断命令正误，以及将命令拆分。

UDPServer.java

服务端主线程。

UDPServerThread.java

服务端子线程。

UDPClient.java

客户端代码

# 使用说明

客户端javac UDPClient.java && java UDPClient

服务端javac UDPServer.java && java UDPServer

# 实验分工

张星主要负责客户端部分，还包括首部，校验和，以及命令提示的编写。

姚雪辉主要负责服务端部分，还包括命令格式以及文件的判断编写。

# README

由于上传了视频，所以使得下载变得麻烦，所以直接上传了张星姚雪辉.zip，其中不包含视频，但是包含了所有的源代码截图以及实验报告，所以想要运行程序直接下载该文件即可。