Computer Networks Midterm Project Design And Testing Documents

Academy: School of Data and Computer Science

Member：梁俊华 16340129 [liangjh45@mail2.sysu.edu.cn](mailto:liangjh45@mail2.sysu.edu.cn)

梁育诚 16340133 [jacky14.liang@gmail.com](mailto:jacky14.liang@gmail.com)

Project Name：LFTP

一、项目介绍

LFTP是一个基于C-S架构的文件传输软件，支持客户端与服务器之间的大文件可靠传输服务。客户端发出传输指令，指明传输或接受文件，然后服务器处理用户请求，进行相应的操作。与此同时，LFTP还支持多用户的并发操作。

二、项目设计

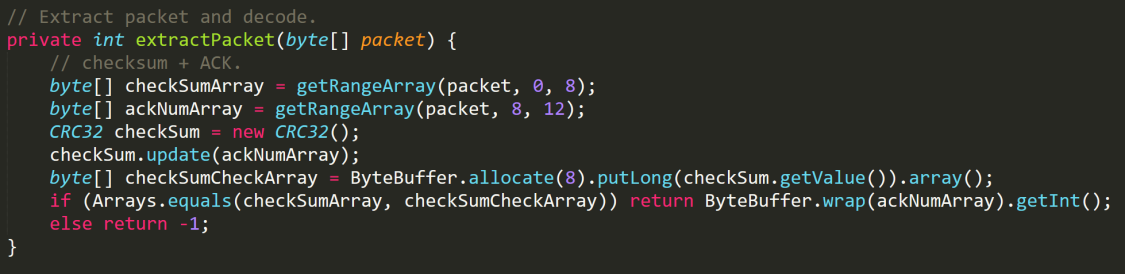
本项目的设计是基于作业要求的，根据每个不同的要求设计对应的功能。从总体上，我们使用Java来实现本项目，因为Java具有良好的多线程编程方式，同时可以灵活支持UDP传输协议等，网络编程的API也非常丰富。

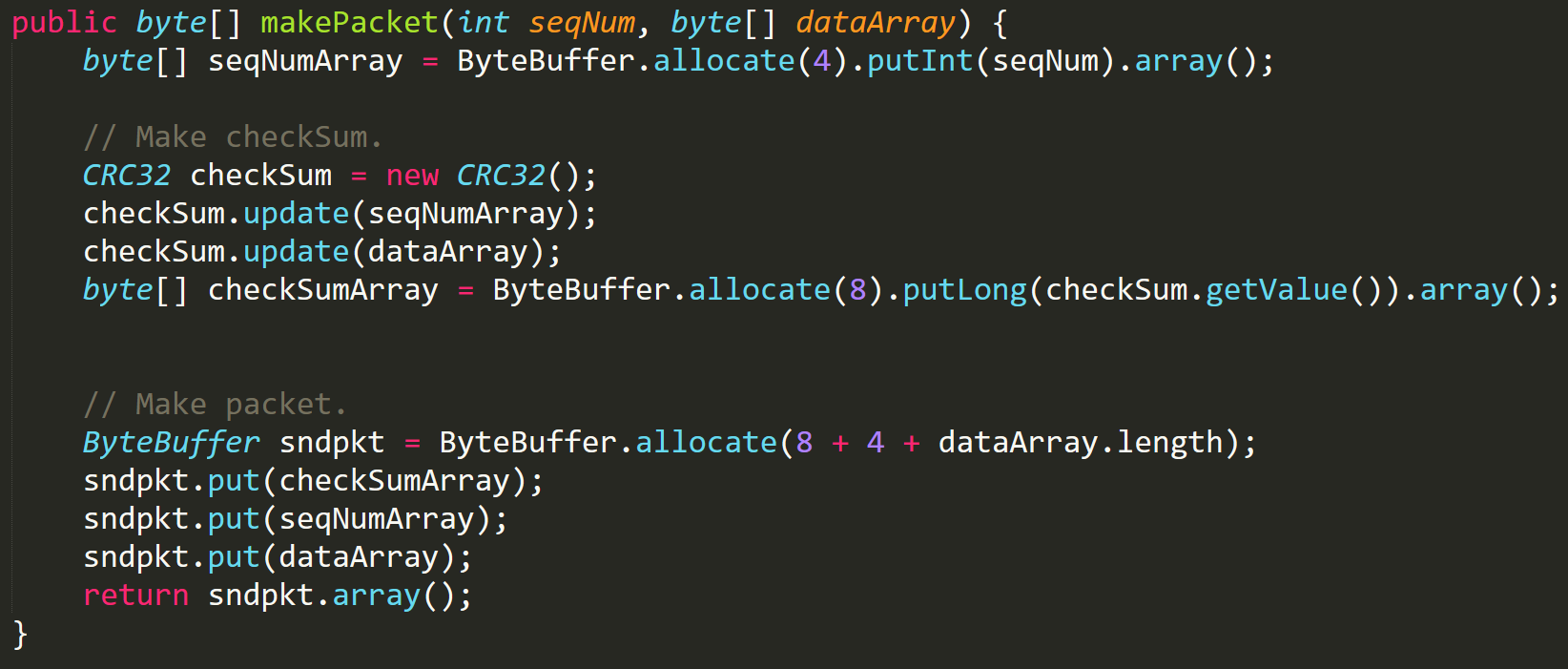
由于使用C-S架构，因此直接实现两个类LFTPClient和LFTPServer即可，运行时分别运行两个端程序即可，服务器是一直开着的，客户端可以有多个。

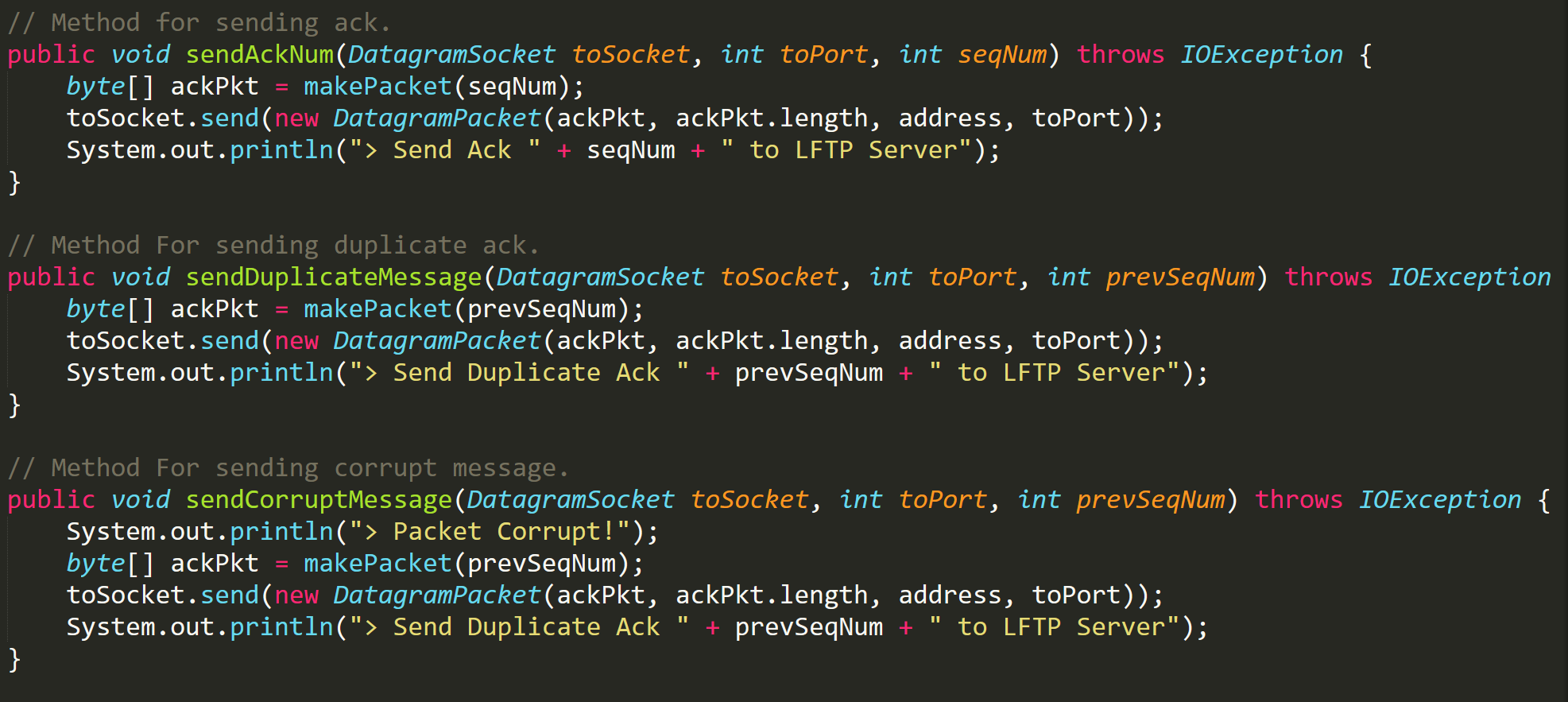
本项目的传输协议要求使用UDP来实现可靠传输功能，在Java中，TCP使用的是流（InputStream或OutputStream），UDP使用的则是包（Datagram），在发送方我们需要对数据包进行分割，包装成Datagram，然后进行发送，在接收方对数据包进行检验和发送ACK包以确认传输到达。

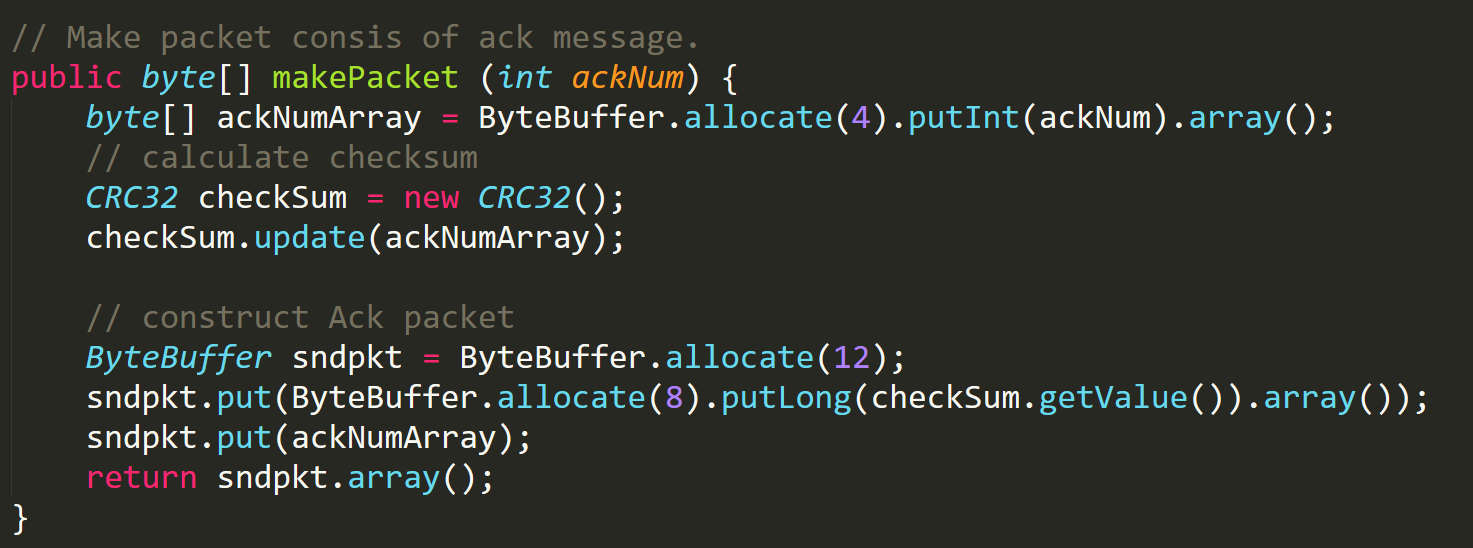
三、实现过程

1. 首先实现文件传输功能。我们需要对源数据读入，然后切割，划分成每个datagram，向头部添加checksum（使用CRC）、seqNum等信息，其中checksum占8个字节，seqNum占4个字节，头部信息共12个字节。然后client向服务器默认端口号（30000）来发起传输请求，服务器应答后，返回一个ACK来确认可以传输，然后双方就开始正式的数据传输了。注意，第一个数据包传输的是文件名、大小等信息。

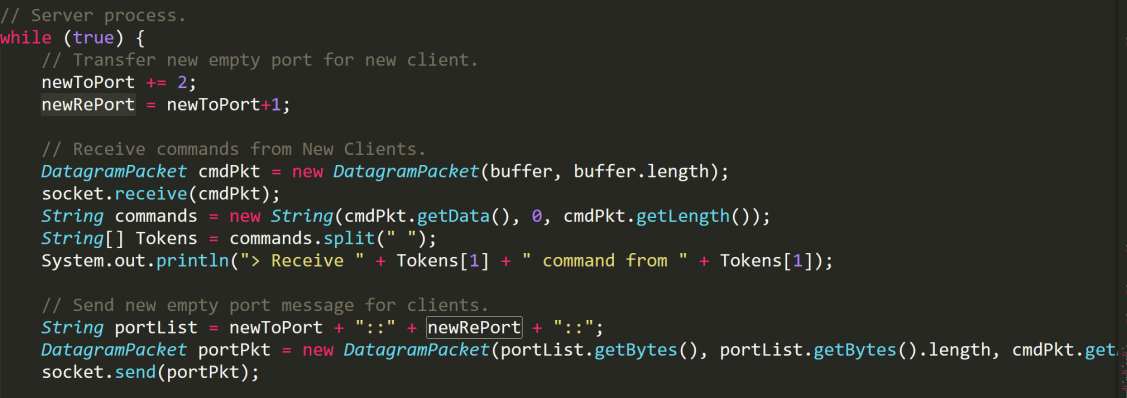


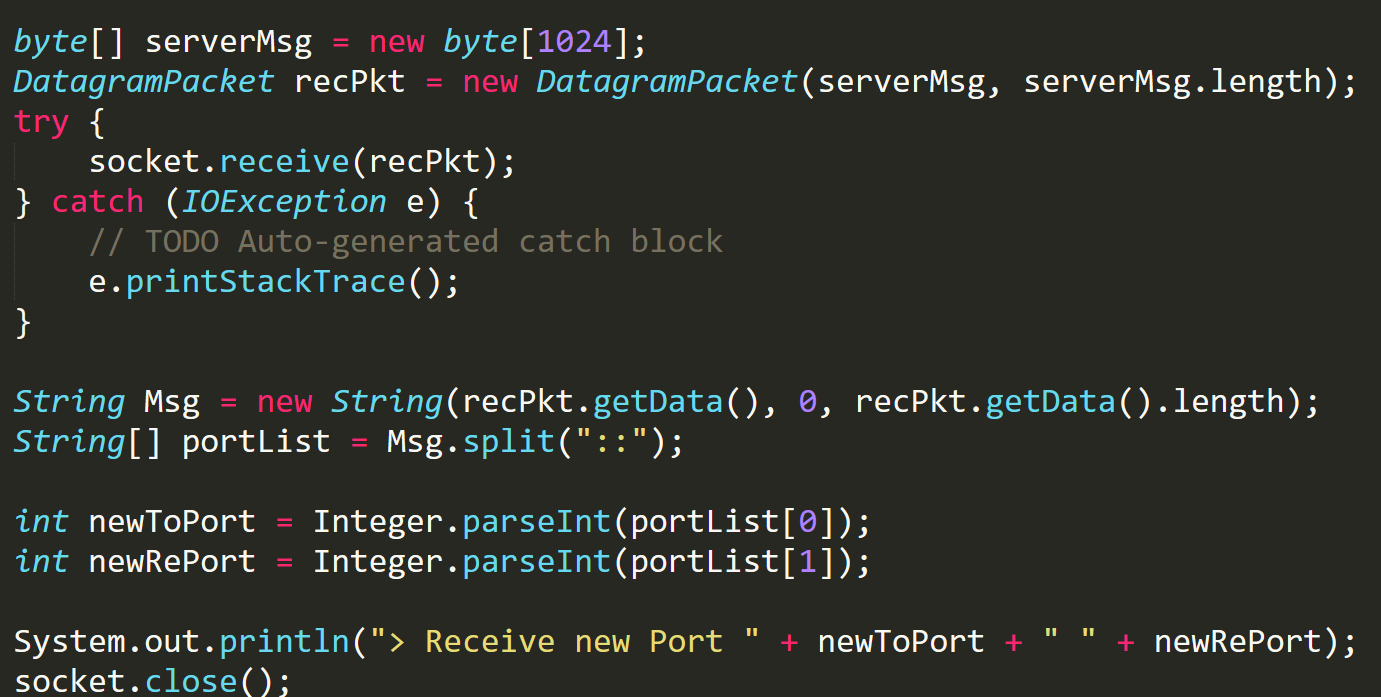




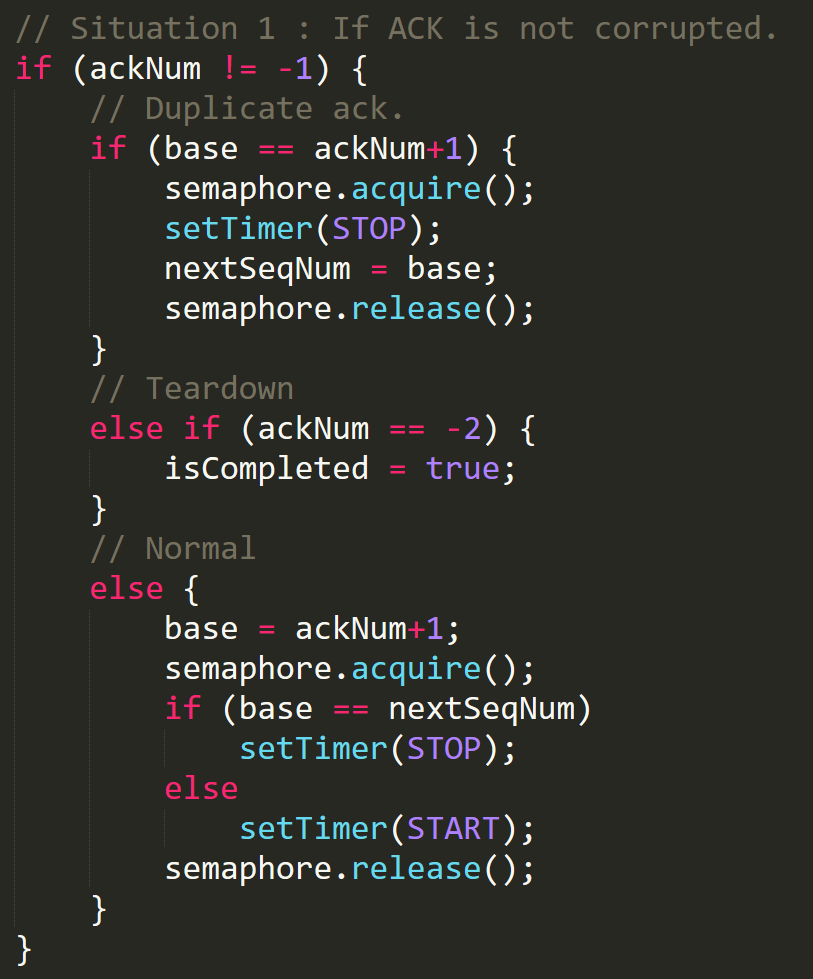


2. 完成第一步的数据传输后，就要增加多线程了。实现多线程会导致一个很严重的问题，就是端口被占用的情况。以客户端为例，如果服务器只提供一个端口，那么就不能够同时使用这个端口进行数据的接受与ACK包的发送。多线程情况下更加会有这个问题，多个进程同时访问服务器更加会有这个问题。经过思考，我们的解决办法是，每次建立一个传输连接时，服务器有义务向客户端提供两个端口（一个用于接收数据，一个用于发送确认的ACK包）。在每次接收到客户端命令后，这两个端口都按照一定规律自增，以确保不会重复。这样的话，客户端就可以同时向服务器发送数据，并接受服务器返回的ACK包。



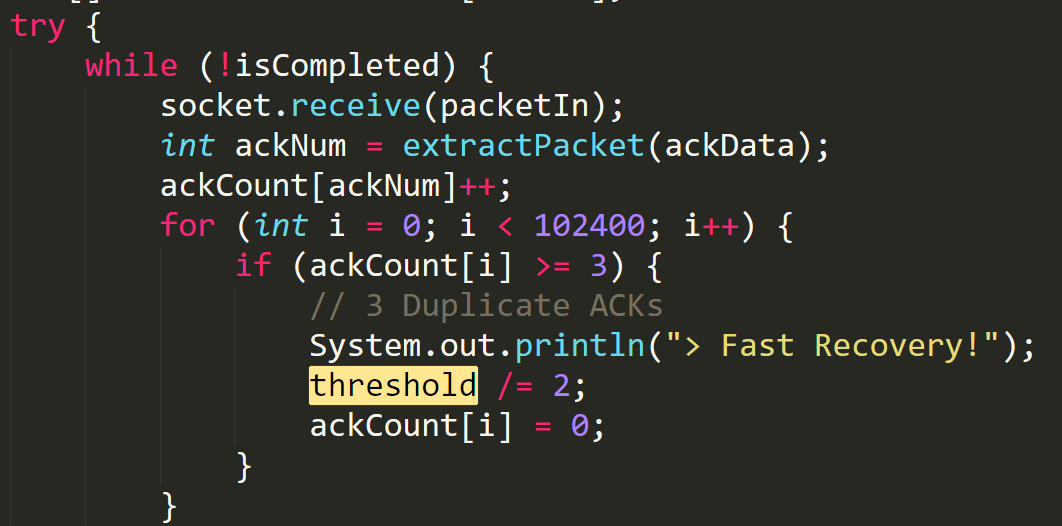


3. 在实现了基本的多线程文件传输后，就可以在这个基础上逐步添加一些TCP的特性了。先来看看流控制（Flow Control）。TCP的流控制是通过窗口滑动机制实现的。在Server端维护一个窗口，记录base（已ACK的seqNum）、nextSeqNum（当前发送包的seqNum），窗口大小为10，即服务器如果得不到最近发送的10个包的确认，它将停止发送。由于这里存在多线程，因此对窗口的滑动操作需要引入同步互斥机制，以确保两个不同的进程不能同时滑动窗口。Java提供了Semaphore信号量，我们只需要在判断阻塞的时候添加一个互斥锁，保证其他线程不能发送或接受包，以滑动窗口。在完成操作后，记得要释放掉这个锁。



4. 显示debug信息。传输开始时，服务器与客户端返回命令信息和文件信息，传输过程中，服务端与客户端反馈seqNum与ACK信息。以确认packet的发送与成功接受。

5. 拥塞控制。拥塞控制方面我们实现了快速恢复机制。使用一个变量threshold来限制windowsize，当服务器收到三个连续的duplicate ACK后，就启动快速恢复。



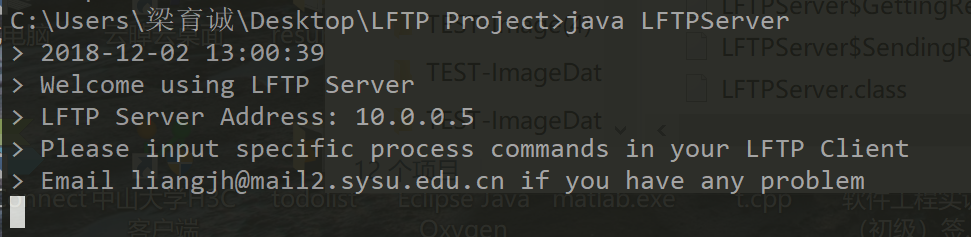
四、测试结果

测试分为本机测试与跨IP测试，本机测试即是用127.0.0.1作为服务器和客户端，在一台电脑上传输和下载文件，跨IP测试使用两个台电脑测试，一台电脑运行客户端，另一台电脑运行服务端，然后进行文件传输与下载测试。先进行本机测试，再进行跨IP测试。

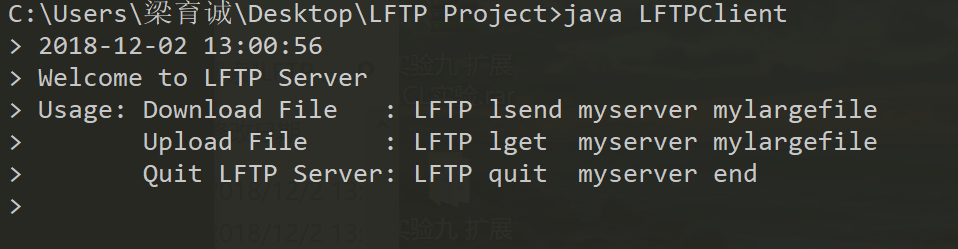
本机测试：

开始界面：

服务器

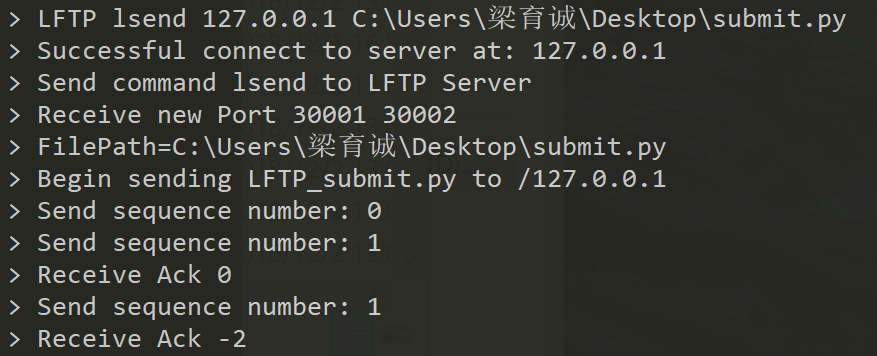


客户端

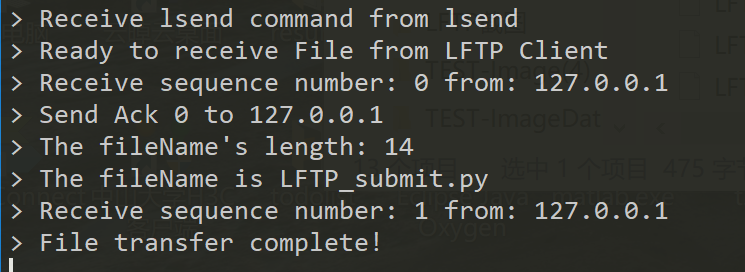


客户端向服务器传输python代码文件：

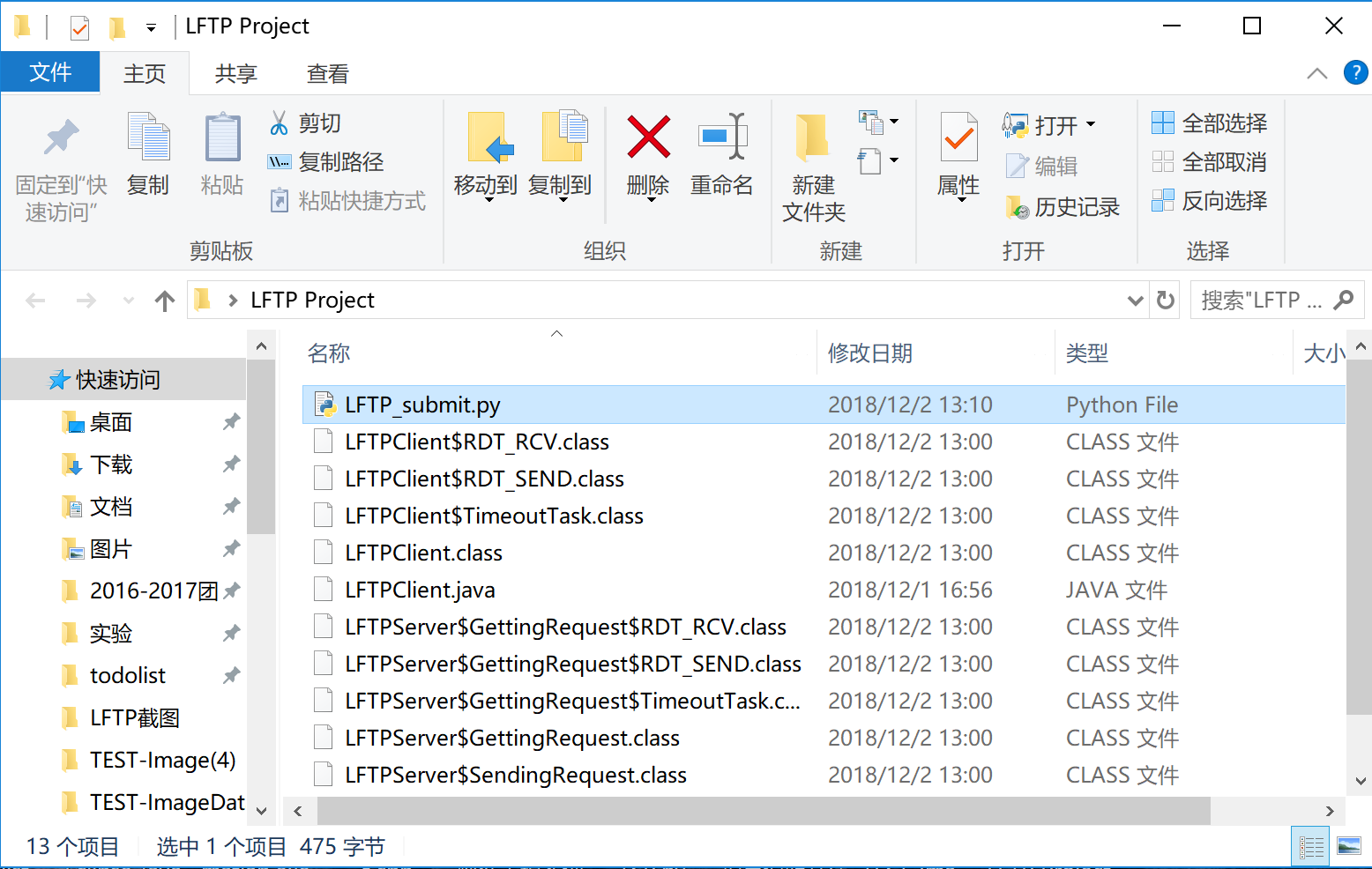
客户端console



服务器console

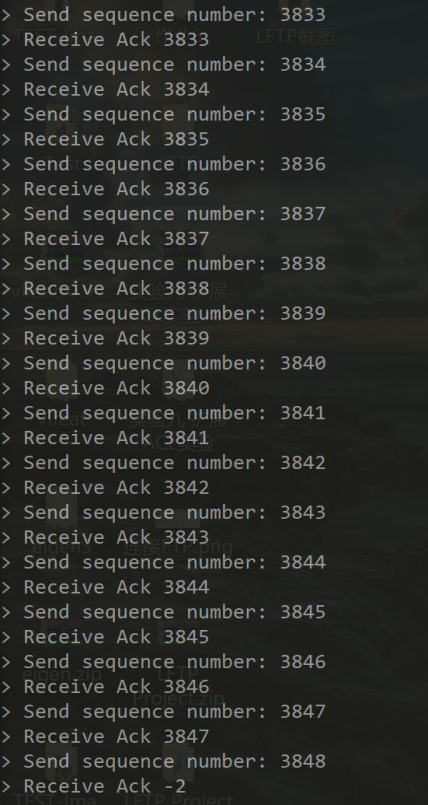


服务器文件夹

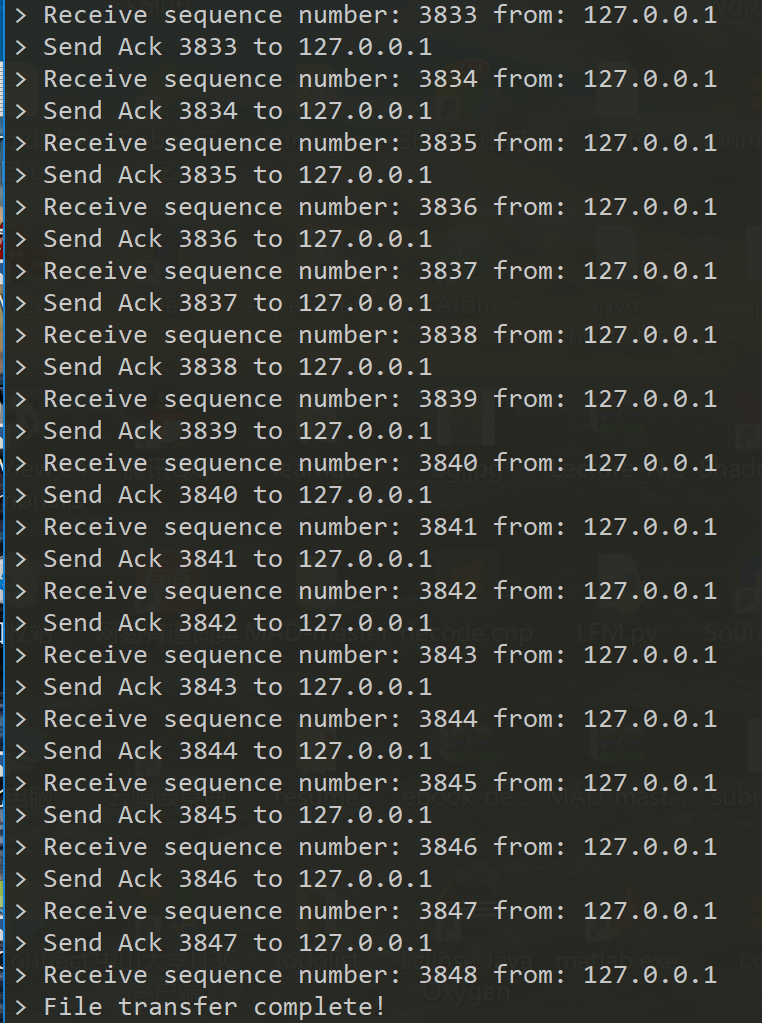


客户端向服务器传输MP3文件：

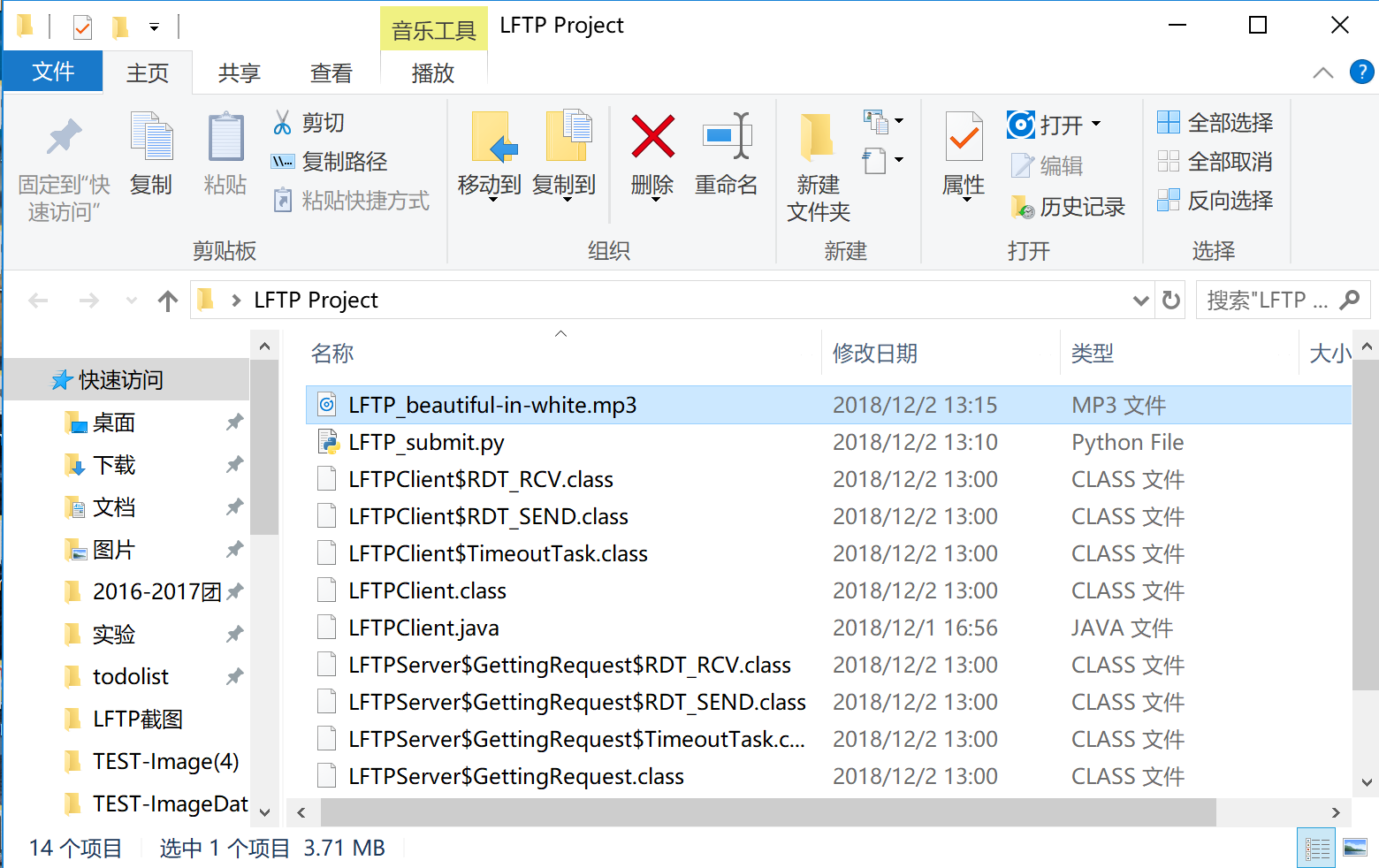
客户端console（部分）



服务器console（部分）

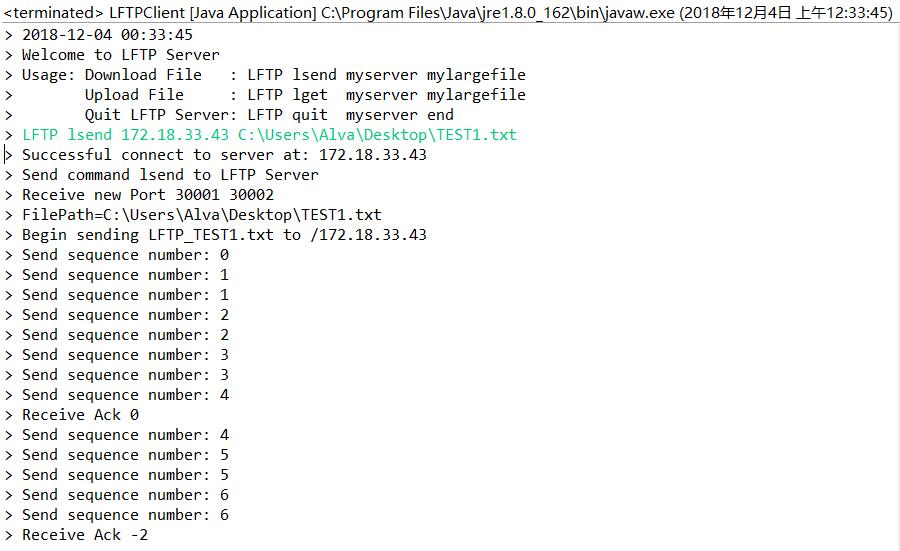


服务器文件夹



跨IP测试：

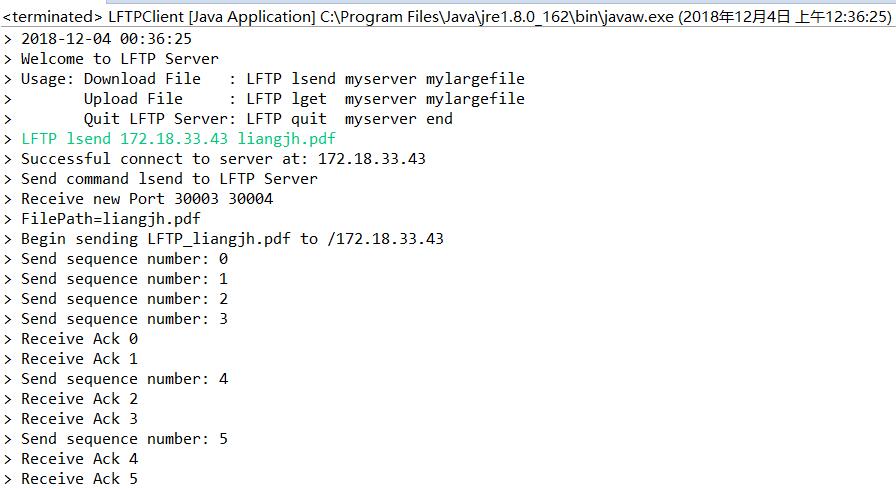
客户端发送txt：



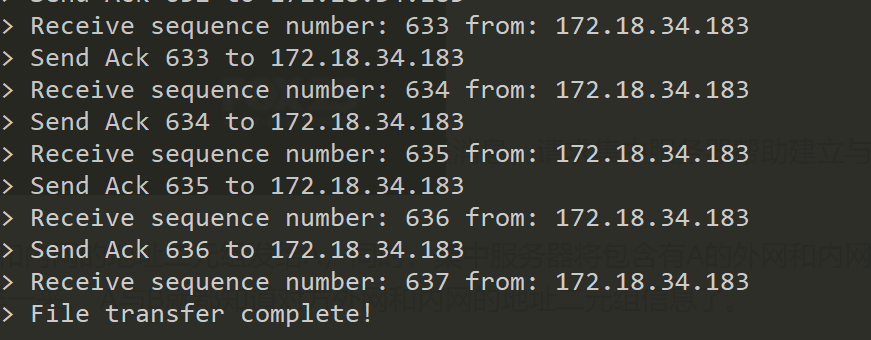
服务器接受txt：



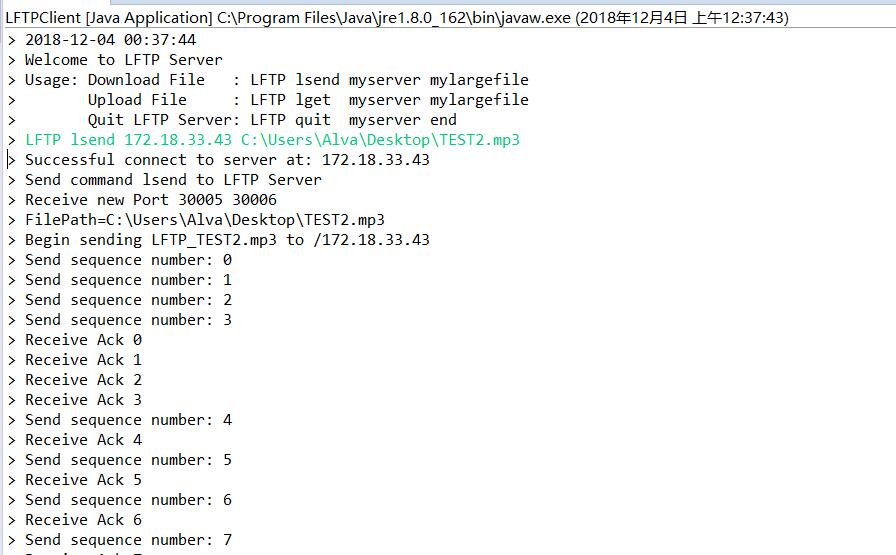
客户端发送pdf：



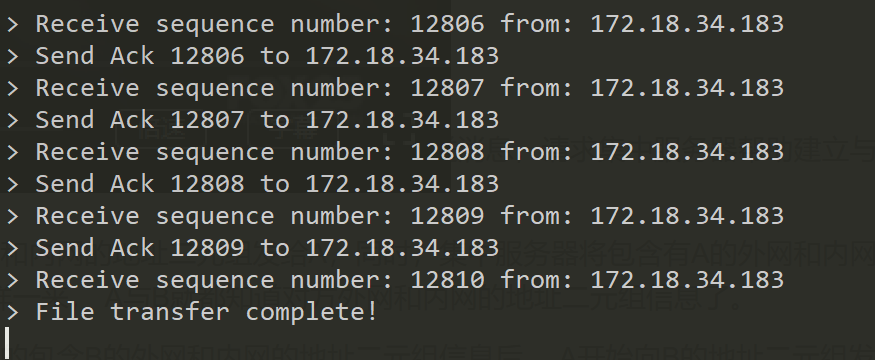
服务器接受pdf：



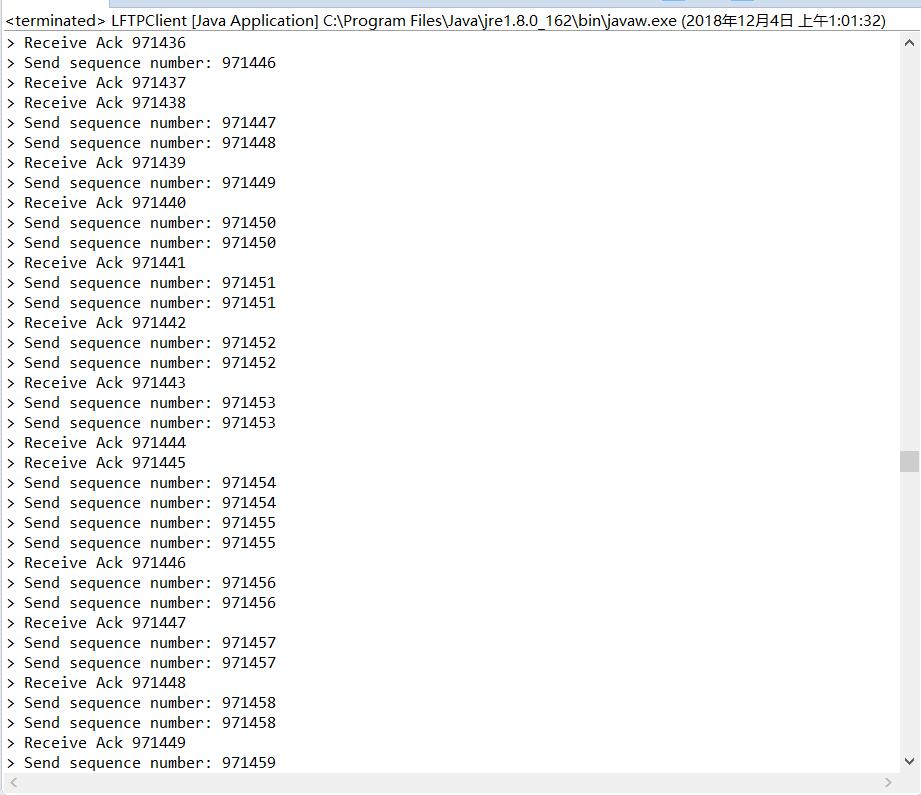
客户端发送mp3：



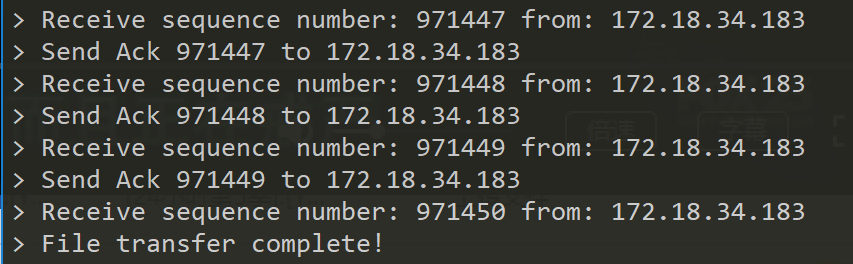
服务器接受mp3：



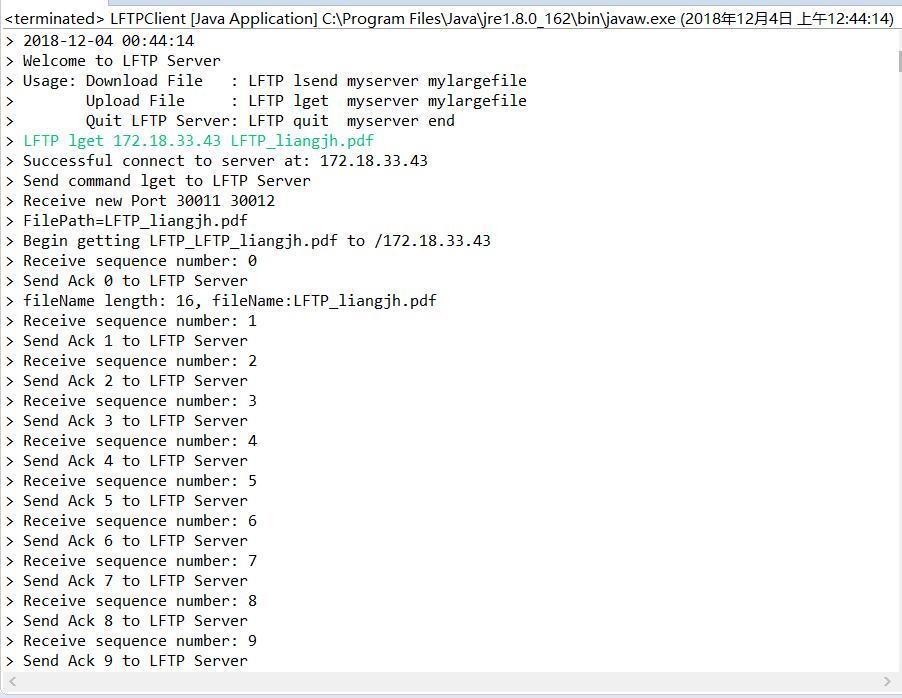
客户端发送mp4：



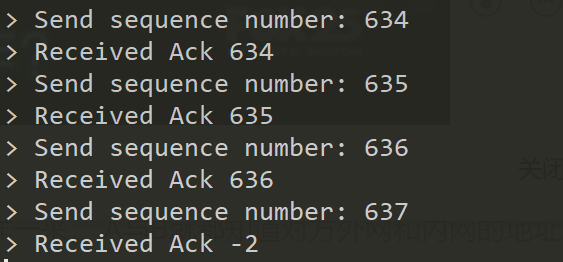
服务器接受mp4：



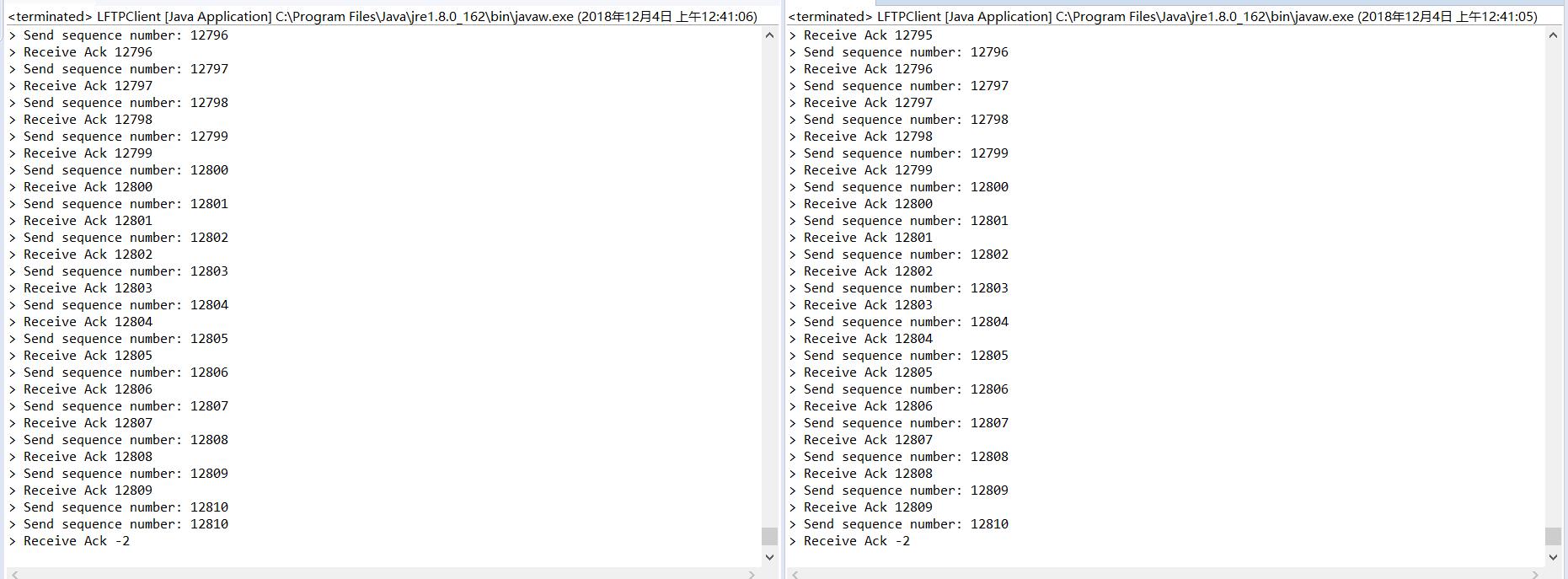
客户端下载pdf：

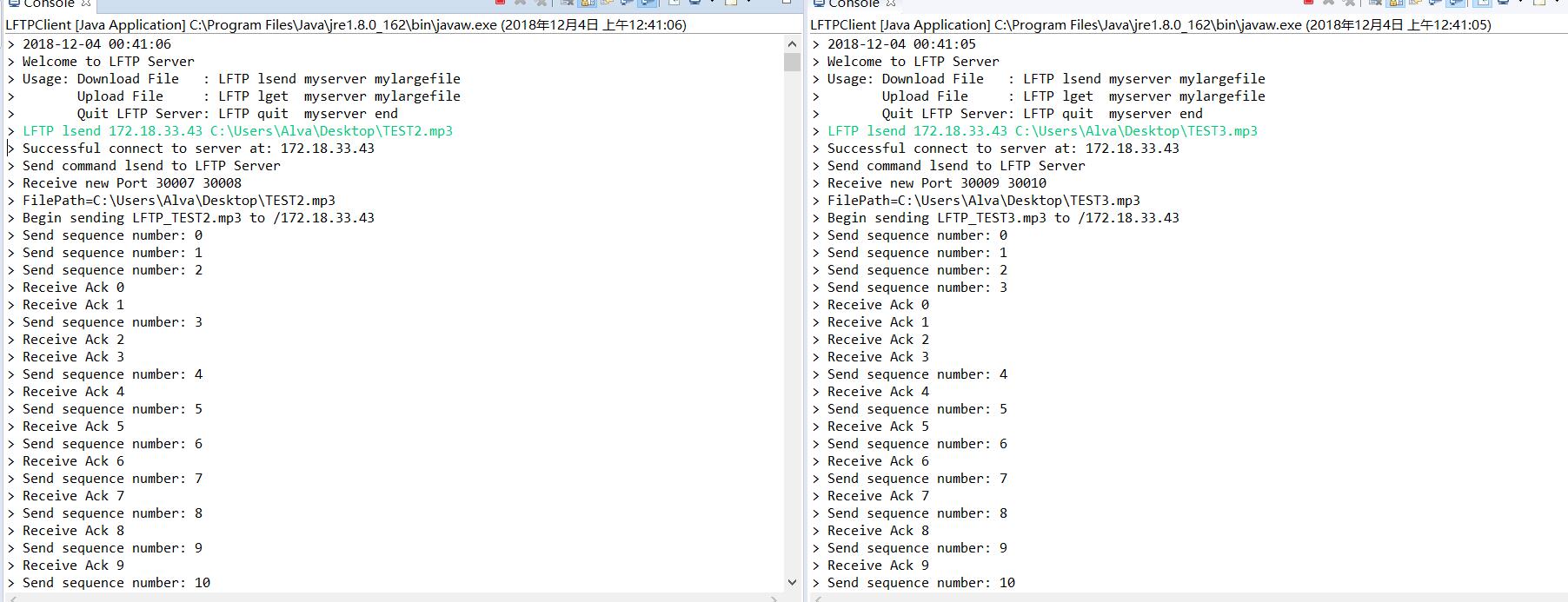


服务器发送pdf：

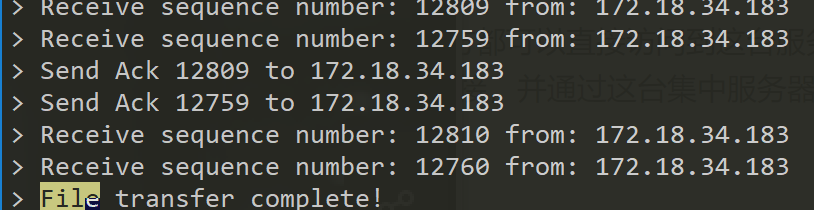


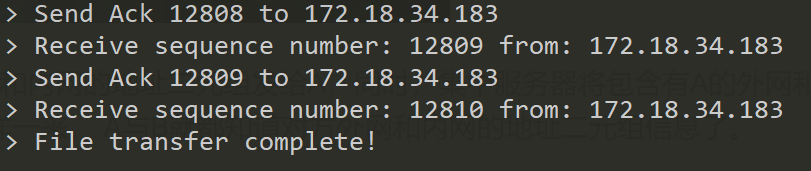
客户端并行发送两个文件：



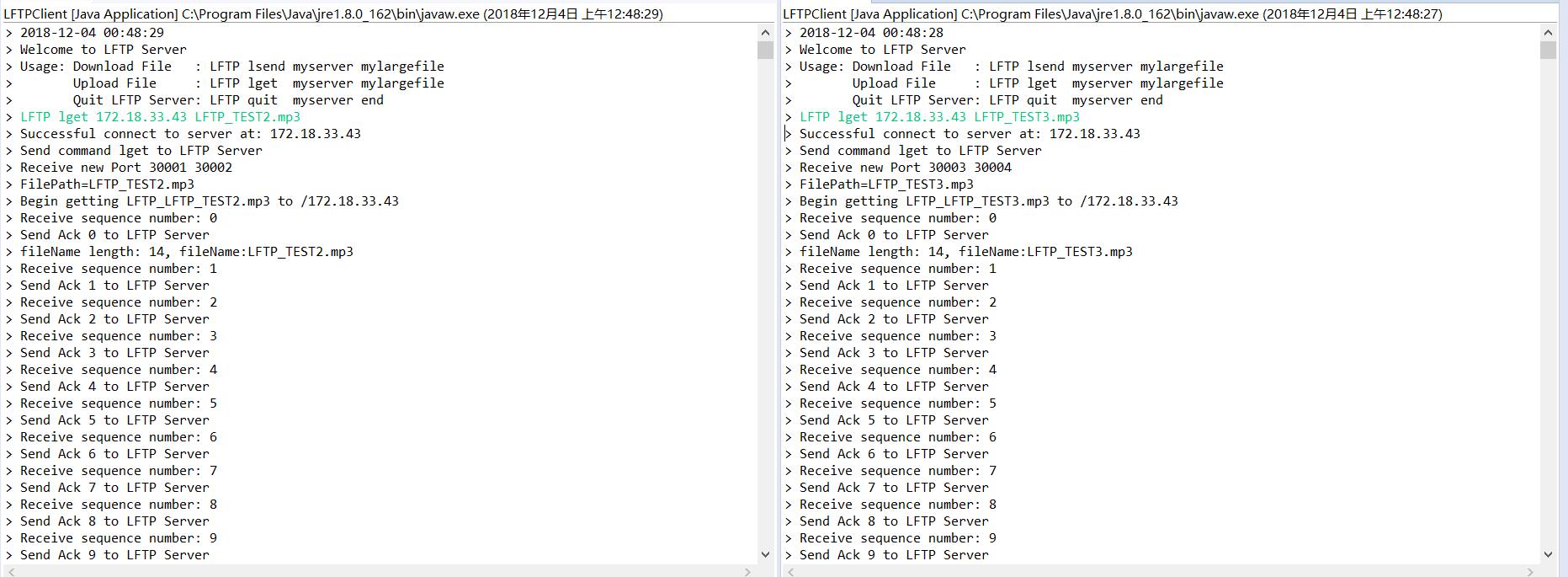


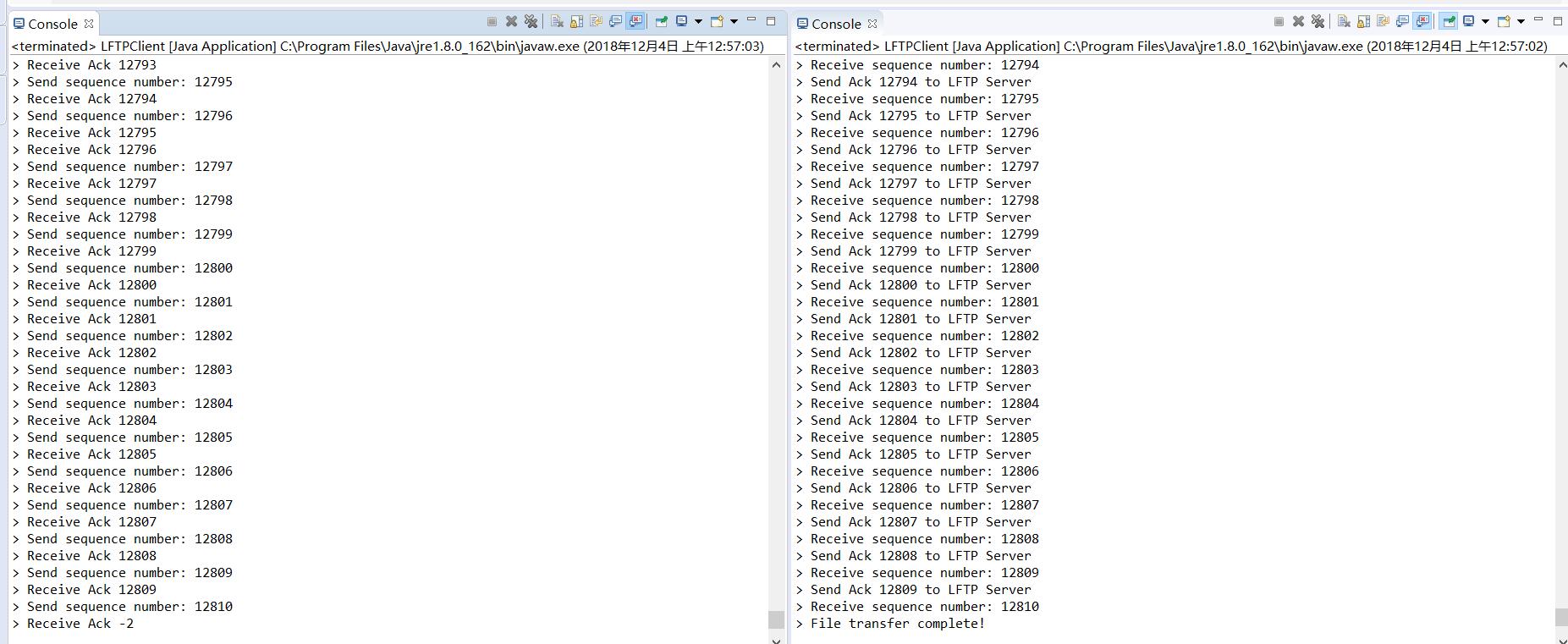
服务器并行接受两个文件：



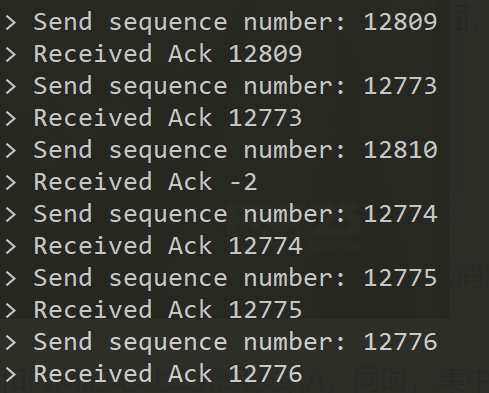


客户端并行下载两个文件：

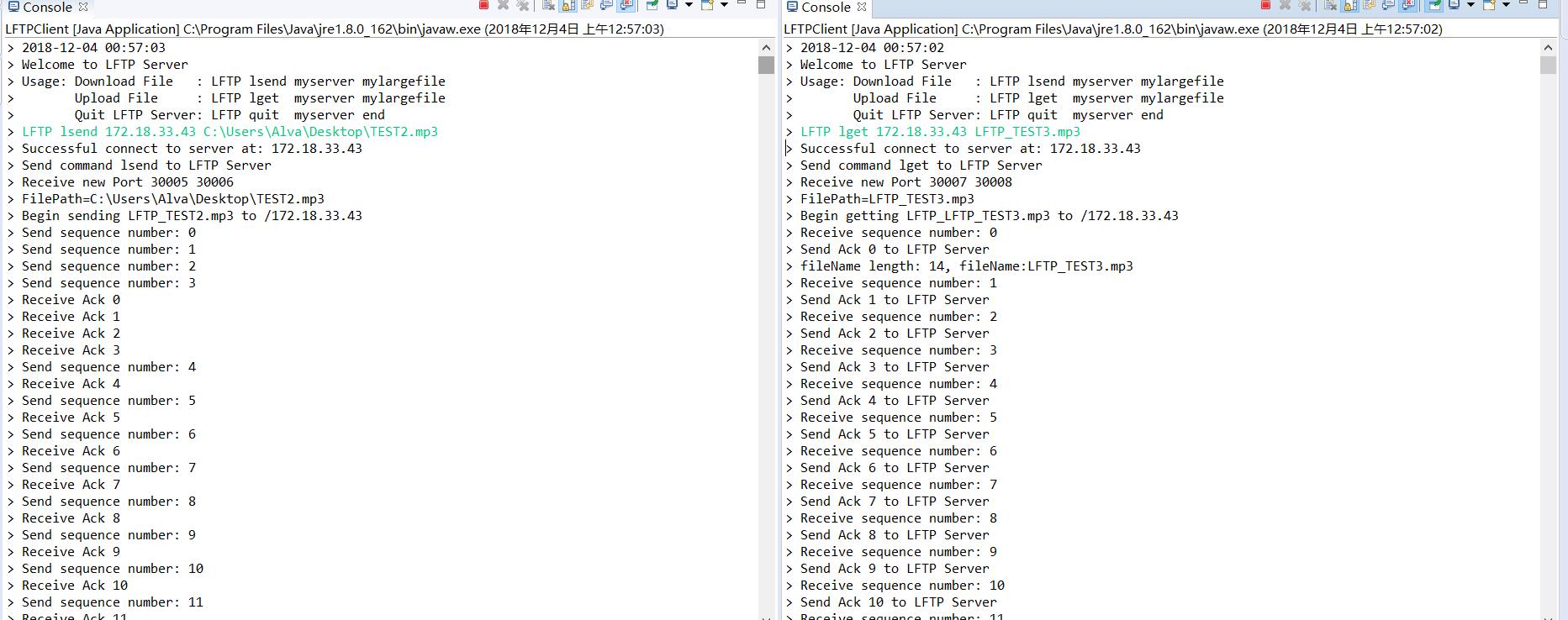


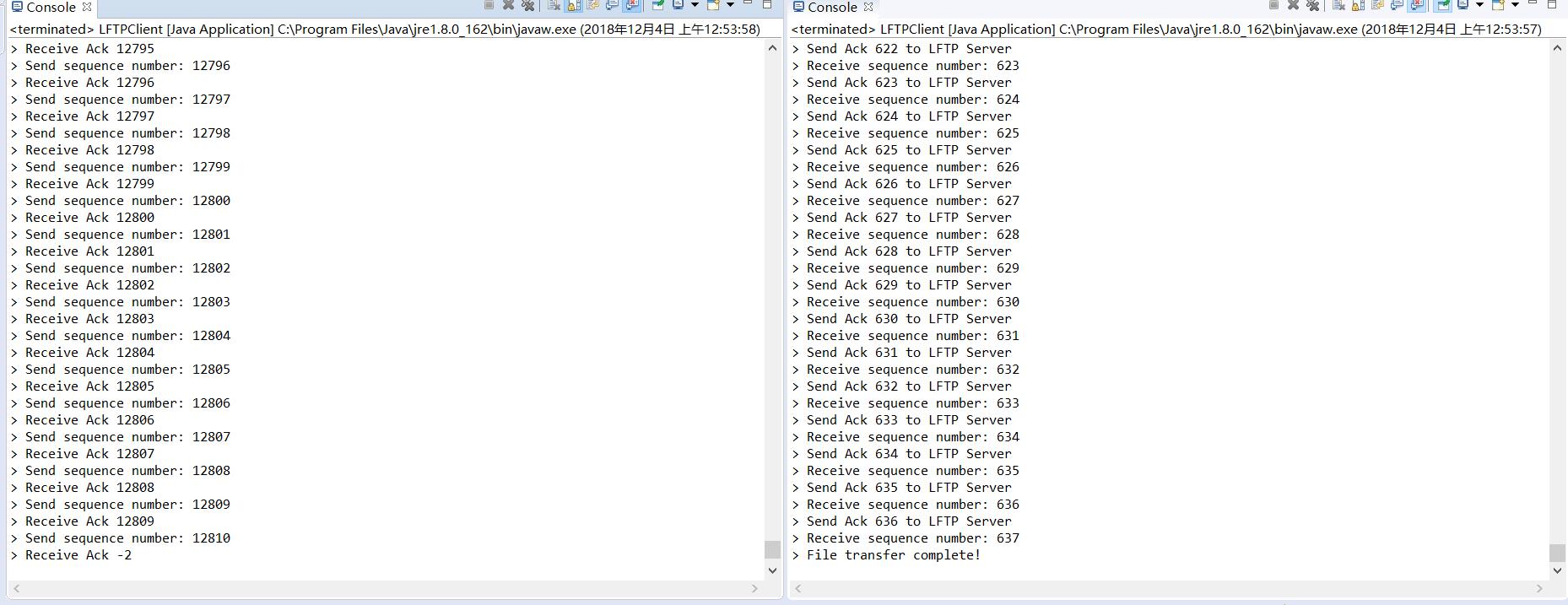


服务器并行发送两个文件：

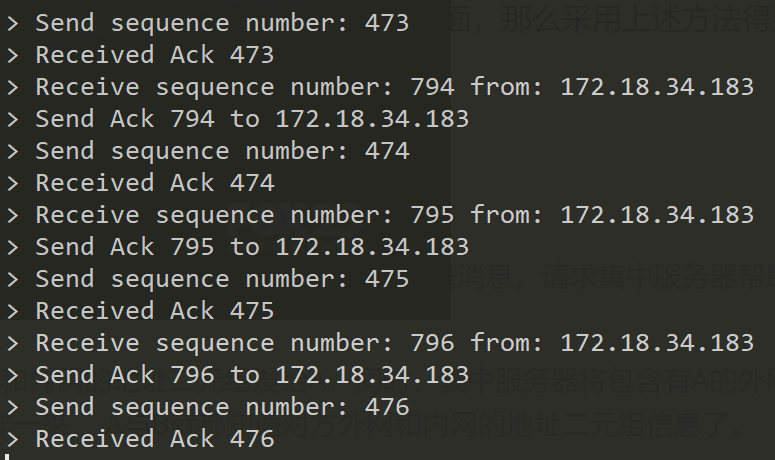


客户端发送下载并行（即发送一个文件、同时下载一个文件）：

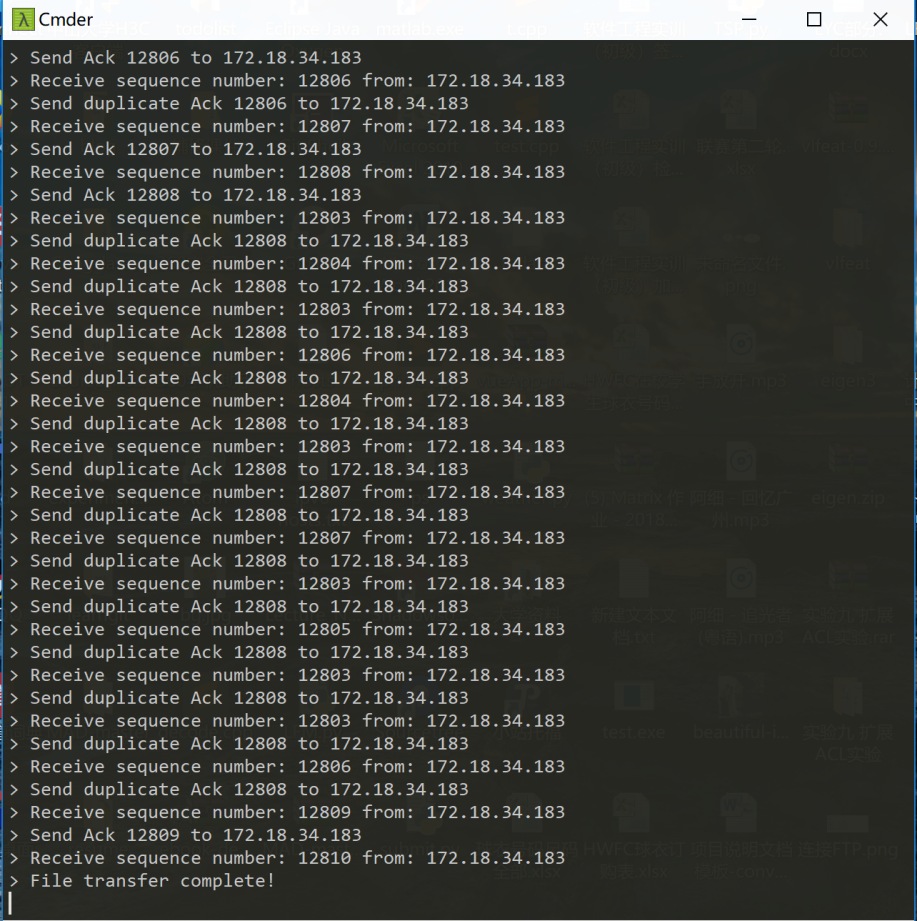




客户端发送接收并行（即接受一个文件、同时发送一个文件）：



丢包重传处理（随机丢包测试）：



测试部分分析：测试部分按照计划分为本机测试与跨IP测试。先传较小的文件如txt、pdf等，然后再传mp3、mp4等音频大文件。在服务器查看文件大小，以及打开文件确认文件没有损坏。然后是各自在命令行中截取部分代码进行分析，主要看发送（或接受）与ACK包的号码之间的关系，这样基本就可以确认程序能够正确运行了。

五、难点与分析

1. 首先是实现大文件传输。一开始的时候无法传出大于1M的文件，怀疑是数据丢包了。后来将数据合理分片后，逐个传输就可以解决了。每个数据包都要有自己的seqNum和checksum，在接收的时候要确认，按照课本上的自动机模型进行编程。

2. 封装数据包的时候，遇到了文本类型转换的问题。我们都知道，数据传输是以byte类型来传的，一开始的时候我们想在包里面手动添加头部信息再用split函数来获取，但是这样存在一个string和byte之间的转换，很容易出错，后来直接对位进行操作，因为头部信息在包中的位置是固定的，也就是说在byte[]数组中的下标是确定的，我们可以直接通过截取下标来获取头部信息。

3. 然后就是端口问题。这是课本上没有提到过的，于是上网找了很久资料，发现是端口占用的问题。解决办法也很简单，就是每次服务器提供两个自己的空闲端口给客户端进行收发数据。每一个线程（连接）就和服务器独有的两个端口号来进行数据传输，互相不干扰。

4. 最后是地址问题。因为开发的时候，两个人都是分开在自己电脑上进行本地测试，对服务器、客户端的地址都不在意，后来到跨IP测试的时候就发现文件无法传输。一开始找了很久都找不到原因，后来仔细思考了一下，既然本地测试是没问题的，说明数据传输的部分应该是没问题的，唯一不同的是连接的地址改变了，因此在许多地方进行了输出，发现确实是服务端发送的地址错了，错用了本机的IP地址，修改后程序就能够正常工作了。

六、总结

这次实验应该是第一次的网络编程，Java这门语言在网络编程这方面还是比较好用。通过实现一个传输大文件的LFTP软件，让我们更加清晰地了解TCP的工作原理与机制，知道如何在一个不可靠的传输协议基础上，通过添加一系列的机制来确保实现可靠的传输服务。整个开发过程比较漫长，也遇到了很多bug，下面有几点总结：第一，一定要先构思好整个应用的架构，不能盲目下手，要在理论上自己演示一遍确认没问题再开始编程，否则程序出错将很难找到问题所在；第二，分模块实现，避免将代码都写在一个函数中。这次实验牵涉到很多关于packet的操作，包括封装和提取，这些代码很多都是大同小异的，因此可以提取出来，精简代码，也使得自己在debug时能够看的清楚；第三，输出seqNum和ACK。这个debug信息可以帮助我们看到整个程序的工作流程，问题是出在服务端还是在客户端，通过观察发送与接受的seqNum就可以很快锁定出错的地方了；第四，先确保连通性，再确认正确性。网络编程中我们最关心的问题就是是否连通和是否准确，因此我们编程也是按照这个思路进行的。首先要实现传输，再在传输成功的基础上确保数据不丢失。如果顺序倒转的话，会发现写了很多代码但是文件根本传不了，无从debug。

总的来说这个项目让我们对上半学期学到的课本的知识有了更深的了解，亲身实践过才能更好地体会到TCP的工作原理。在忙碌的学期中能完成这样一个LFTP项目，我们都比较满意成果，也希望在日后的学习中，能够将更多的课本知识应用到实际当中。