LFTP 项目报告

陈铭涛 16340024

陈慕远 16340025

LFTP 项目设计

### 项目简介

本次提交的LFTP 项目使用 Python 进行编写，实现了基于 UDP 的文件传输，实现了与 TCP 类似的连接状态和流量控制与拥塞控制。程序分为客户端与服务端，在命令行下使用。服务端支持多个客户端同时进行文件的上传或下载。服务器启动时默认监听在0.0.0.0:9999下，也可通过命令行参数指定监听的地址与端口号。客户端使用如下的三个命令分别进行服务器上文件列表打印，文件上传与文件下载：

python3 client.py ls serveraddr

python3 client.py lsend serveraddr filename

python3 client.py lget serveraddr filename

其中，serveraddr 的格式为 ip:port 或domain:port，支持使用域名作为服务器地址。文件上传时不能使用服务器上已存在的文件名。

#### 程序执行流程

程序实现了一个reliableUDP 模块，程序的应用部分调用该模块进行可靠的文件传输。

服务端的应用部分根据通信对象的 ip 地址和端口区分不同的连接所处的应用层逻辑状态，根据客户端发送的请求做出返回文件列表，接收文件以及发送文件的操作。当接收 reliableUDP 调用关闭连接的接口后删除通信对象的状态。

客户端的应用部分在启动后首先对用户所指定的各个参数进行分析，然后调用 reliableUDP 发送握手请求到指定的服务端，三次挥手完成后构成连接，应用将发送用户指定的请求命令到服务器，根据服务器的响应执行请求或打印出错误信息。在命令执行完成后，客户端程序将调用 reliableUDP 的关闭连接发送第一次挥手，待四次挥手完成后退出程序。

#### reliableUDP 模块设计简介

reliableUDP 模块在 LFTP 应用进行文件数据传输时使用，实现了基于 UDP 的可靠数据传输。

reliableUDP （简称 rUDP）在 发送和接收的数据包的表头结构与 TCP 的表头结构基本相同，在表头中实际被模块使用的字段为：源端口，目标端口，序列号码，确认号码，ACK，SYN，FIN，窗口大小，数据偏移量，以及校验和。其余字段大部分默认置为0。

rUDP中包含了 server 与 client 主要两个部分，在通讯过程中由客户端发起构建连接握手与释放连接挥手，当连接构建完成后服务器与客户端都可以向对方发送信息，使用双方各自的seq 号码标记发送数据包的序号以保证传输过程中的顺序可靠。

对每一个包，发送方会将不包含 checksum 的表头与数据合并计算出一个2字节的校验和作为简单的错误校验。校验和的计算对所有数据按16比特的组计算所有数据的1的补数和(1’s complement sum)。接收方对接收到的数据包含 checksum 位使用相同的方法进行计算，若计算获得的checksum 结果为全1则验证通过。在校验和的实现中，由于考虑到发送方使用的 socket 的 ip 地址与接收方所识别到的发送者地址不一定相同，因此没有加入实际 tcp 中计算 checksum 时使用的pseudo-header。

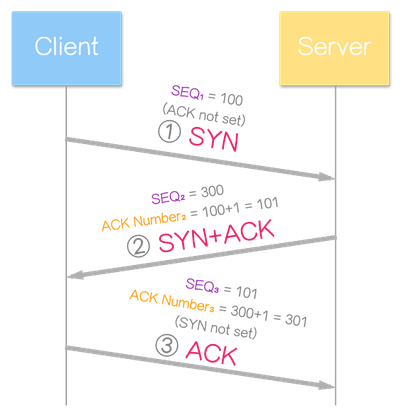
类似 TCP，对于一个连接服务端的状态分别有：CLOSED, LISTEN, SYN\_REVD, ESTABLISHED, CLOSE\_WAIT, LAST\_ACK。客户端的状态有: CLOSED, SYN\_SENT, ESTABLISHED, FIN\_WAIT\_1, FINWAIT\_2, TIME\_WAIT。状态在实现的 Python 代码中使用两个枚举类进行标识：

RecvStates = Enum('RecvStates', ('CLOSED', 'LISTEN', 'SYN\_REVD', 'ESTABLISHED', 'CLOSE\_WAIT', 'LAST\_ACK'))

SendStates = Enum('SendStates', ('CLOSED', 'SYN\_SENT', 'ESTABLISHED', 'FIN\_WAIT\_1', 'FIN\_WAIT\_2', 'TIME\_WAIT'))

对于每一个连接，rUDP 客户端将发送第一个握手消息，SYN 位为1，seq 序号为一个随机数，收到该握手消息后服务端将发送第二次握手消息，SYN 和 ACK位为1，ack 序号为收到的 seq 序号加1，seq 序号为另一个随机数。收到第二次握手的消息后，客户端将发送第三次握手的消息，ACK 位为1，ack 值为收到的 seq 序号加一。

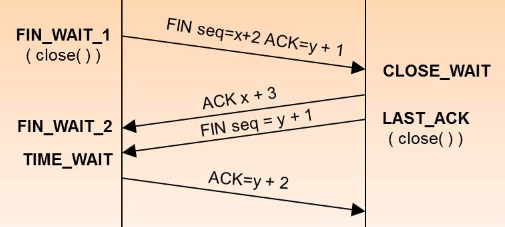
建立连接三次握手的过程如下：



（数值仅为说明，实际实现中 seq序号为一个随机值）

建立连接后客户端与服务端进入ESTABLISHED 状态，所有应用的数据传输在此状态下进行，双方第一次进行数据传输时的 seq 序号为其初始随机的 seq 序号下一，后续包的 seq 序号为该随机值加上累计传输的字节数。收到发送方传输的数据包后，若该数据包的 seq 值为接收方当前已 ack 到的数值，则返回的 ack 包的 ack 数值为此前的 ack 值加上收到的包的长度，并将数据放入接收窗口，否则返回的 ack 值不变。发送方对发送的每一个包绑定一个 timer，默认当数据包发出1秒后仍未收到大于其 seq 值的 ack 时进行重传。

在完成所有数据的传输后，客户端将发送第一次挥手信息，FIN 位为1，服务端接收到后发送第二次和第三次的挥手信息，客户端接收后返回第四次挥手信息结束连接。大致图示如下：



#### 流量控制

#### 拥塞控制

#### 应用模块

应用通过继承 rUDP 下的application 子模块中的 app 抽象类 来使用 rUDP。继承后的派生类中应当包含一个 rUDP 的客户端或服务端对象，在构造rUDP对象时将应用自身对象传入供 rUDP 进行调用。当 rUDP 连接接收到数据消息时，将把数据存入接收窗口并调用notify\_process\_data通知应用处理接收的数据。应用需要实现process\_data方法并在方法中调用 rUDP 的consume\_rcv\_buffer方法获取接收数据进行处理。当rUDP 接收到新的 ack 数据后将调用notify\_next\_move通知应用进行下一步操作（传输文件的下一部分，关闭连接等）。 在应用需要执行的操作结束后，调用 rUDP 的finish\_conn方法进行挥手结束连接。

#### 日志模块

rUDP 下包含 logger文件，在程序执行中应用与 rUDP 均调用该文件进行调试信息日志的输出，输出日志将存在程序执行目录下的lftp.log 文件下。

### 应用部分设计简介

LFTP 的应用部分分为 server 和 client，分别的使用方法如下：

1. 服务端：

python3 server.py [-h] [-p PORT] [-a ADDR] [-d DATADIR]

接收的各参数如下：

* -h 打印出程序参数帮助信息
* -p 程序监听的端口号
* -a 程序监听的 ip 地址
* -d 程序服务文件的根目录，客户端进行上传或下载时将对该目录下的文件进行操作，默认为程序的执行目录。

1. 客户端

python3 client.py [-h] command ServerAddr [filename]

各参数介绍如下：

* -h打印出程序参数帮助信息
* command 程序指令，支持的指令为列出文件ls, 下载文件lget, 上传文件lsend。
* ServerAddr 目标服务器地址，需指定ip 或域名地址加端口号
* Filename 当操作为上传或下载时需指定该参数，表示需要上传或下载的文件名

程序启动后，将首先使用argparser 对参数进行处理，若参数合法，服务端将调用 rUDP 开始对指定的地址进行监听。客户端将启动rUDP 对目标服务器进行连接的建立。建立后将发送对应的指令至服务器（客户端与服务器应用层交互的指令可参考LFTP Document文件）。服务器根据其实际数据目录下的情况返回对客户端指令的响应。若操作为上传或下载且服务器响应操作有效，则发送文件一方将首先使用一个数据包发送文件的大小，然后进行文件数据的传输。

在连接建立后的传输过程中，所有非 ack 的数据包大小均设为5k，每个包数据部分的前4个字节代表这个包的实际长度，对于长度不足5116字节的发送数据，应用部分进行发送前会对其末尾部分填充全0的 padding。

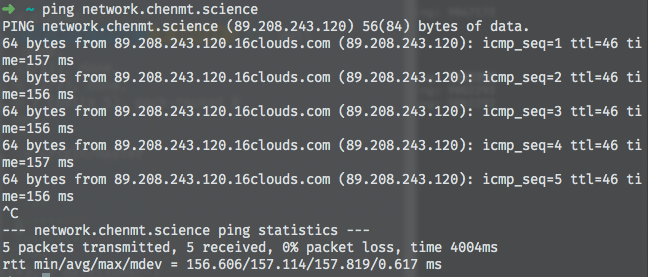
### 程序文档

对程序的各类与方法的介绍可参考 LFTP Document 文件。

程序测试

#### 测试环境

测试环境为分别位于美国洛杉矶和中国香港的两台服务器，两服务器之间的连接时延如下：



用于测试传输的文件为一个大小为100M 的文件：

