­­­­CSE 3461

Lab Report II

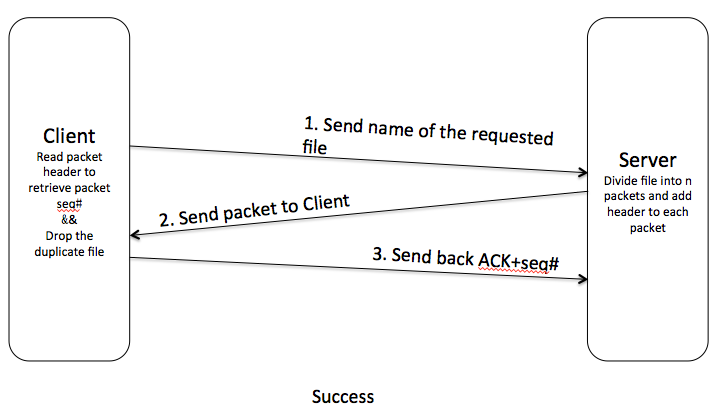
Professor: Chunyi Peng

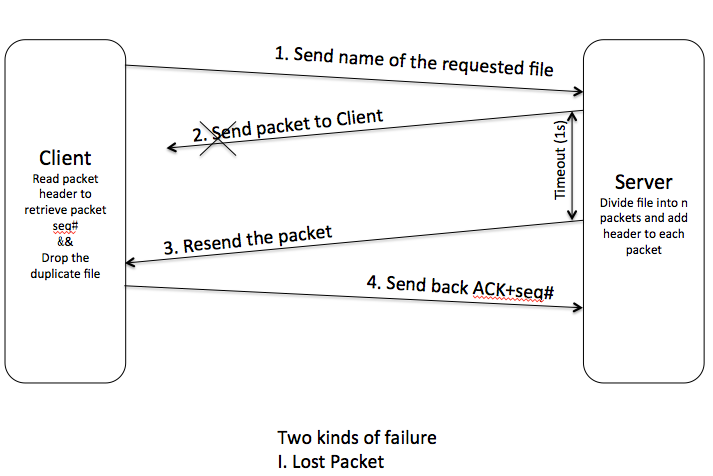
Jiasong Sun (200275947), Manlan Li (200301962)

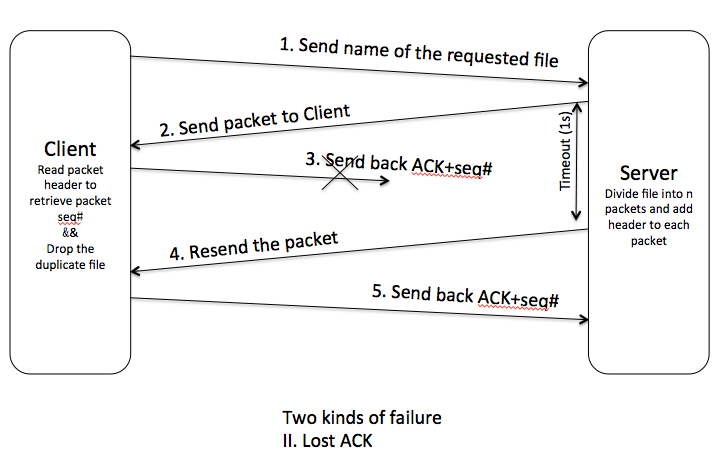
1. Abstract

This project requires us to implement UDP as well as reliable data transfer protocol (RDT 3.0) to perform the data transfer. Code in both sender side and receiver side are implemented.

1. Description







* 1. Sender Side Functions

//Read the requested file into buffer and send it to the client

void sendFileToClient(char \*fileName, int sd, int flags, struct sockaddr \*cliAddr, int cliLen, float probability);

//Get the size of file

int getFileSize(char \*filePath);

//Simulate packet lost; return 1 if loss packet; return 0 if not

int lostPacket(float pro);

//Waiting for ACK from client; if timeout or received ACK is not expected, return 1 to indicate we need to resend the packet. Return 1 if we need to resent packet; return 0, if we do not need to resent

int checkResendPacket(int sd, int timeout, char \*ackBuffer, int flags, struct sockaddr \*cliAddr, int \*cliLenAdd, int seqNum);

//Return 1 if it is expected ACK; return 0 if it is not expected ACK \*

int isExpectedACK(char \*ackBuffer, int seqNum);

//Return 1 if timer stops before timeout; return 0 if timeout

int startTimer(int sd, int timeout);

b) Client Side Functions

//This functon will receive the package from server and put file into a buffer

void receiveDataFromServer(int sd, int flags, int timeOut, char \*fileName, float probability);

//This funciton will return the index of a perticular char in the str

int indexOfCharInString(char \*str, int length, char ch);

//This funciton will parsing the header

int parseHeader(char \*packet, int headerLength, int \*seqNum, int \*pktMaxNum);

//This funciton will return length of header

int findHeaderLength(char \*packet, int length);

//This funciton will return Ack number to ackBuffer

void sendAck(int sd, int seqNum, int flags, const struct sockaddr \*echoServAddr,

int echoLen);

//This function simulates ACK lost; return 1 if ack lost; return 0 if not

int lostAck(float pro);

1. Difficulties

Problem

After receiving the last packet, the client will send back an ACK and close the port. However, this very last ACK client sends may get lost. But server doesn't know whether the packet is received or not. So the server will keep sending duplicated packet even thought the client side is already closed and client won’t send back another ack.

Solution

We set server resend packet maximum 16 times and after that the server will terminate it’s own port.

1. User Manual
   1. Compile:
      1. In your own Linux OS, open a terminal
      2. In the project folder, type *make* in the terminal, this should call Makefile, which is already included in the lab1 directory and **server** executable file will be generated;
      3. Type *cd receiver* enter receiver folder, type *make*. This should call Makefile again, which is already included in the lab1 directory and **client** executable file will be generated;
   2. Run:
      1. On your terminal, Type *./server port# pro*
      2. Open another terminal, in your receiver folder type *./client localhost port#* ***testFile*** *pro* testFile can be one of these three types: .html .jpeg (.jpg) .gif

(port# can be any arbitrary natural number outside the range of 0~1024, pro can be any arbitrary floating number between 0~1)

1. Sample Output

|  |  |
| --- | --- |
| Server | Client |
|  |  |