­­­­CSE 3461

Lab Report II

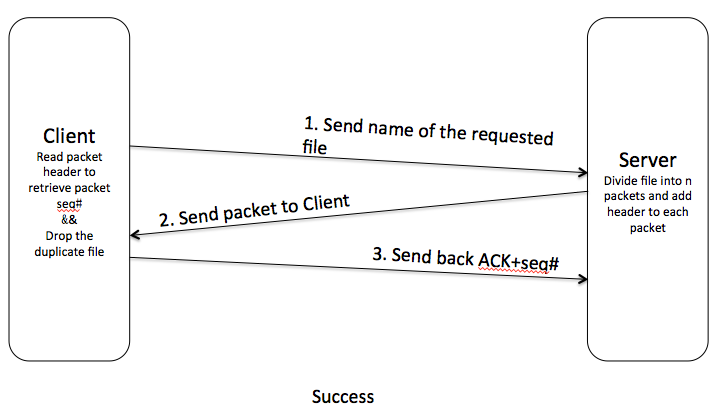
Professor: Chunyi Peng

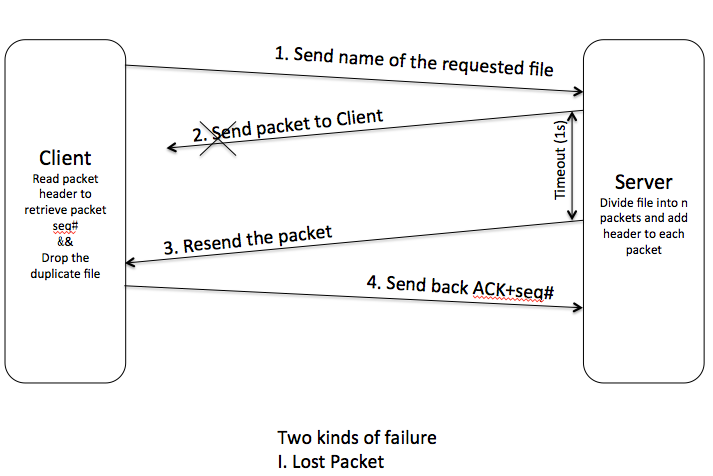
Jiasong Sun (200275947), Manlan Li (200301962)

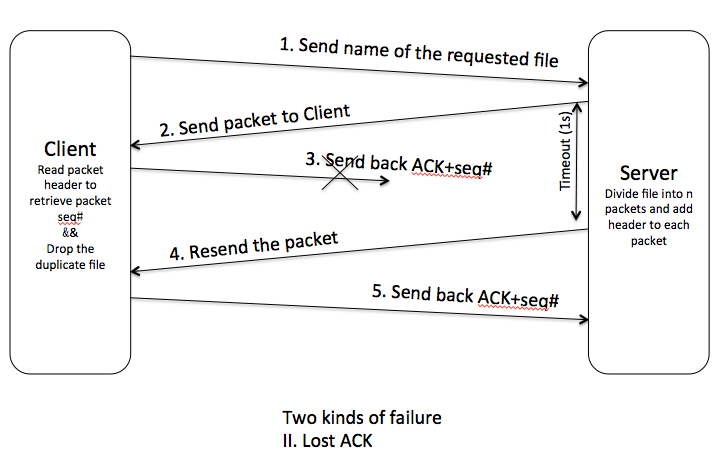
1. Abstract

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

/\*

\* Function: sendFileToClient

\* --------------------

\* 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);

/\*

\* Function: getFileSize

\* --------------------

\* Get the size of file

\*

\* \*filePath: the path of specific file

\*/

int getFileSize(char \*filePath);

/\*

\* Function: lostPacket

\* --------------------

\* Simulate packet lost; return 1 if loss packet; return 0 if not

\*

\* pro: ??

\*/

int lostPacket(float pro);

/\*

\* Function: checkResendPacket

\* --------------------

\* 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);

/\*

\* Function: isExpectedACK

\* --------------------

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

\*

\*

\*/

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

/\*

\* Function: startTimer

\* --------------------

\* Return 1 if timer stops before timeout; return 0 if timeout

\*

\*

\*/

int startTimer(int sd, int timeout);

b) Client Side Functions

/\*

\* Function: receiveDataFromServer

\* --------------------

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

\*

\* sd: ?

\* flags: a mark to see whether the package is lost or not

\* timeOut: timeout period of pkg losing

\* \*fileName: name of file to be sent

\* probability: the probability of package losing

\*

\*/

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

/\*

\* Function: indexOfCharInString

\* --------------------

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

\*

\* \*str: String which may contain ch

\* length: length of str

\* ch: the perticular char to be found

\*

\*/

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

/\*

\* Function: parseHeader

\* --------------------

\* This funciton will parsing the header

\*

\* \*packet: String of packet content

\* length: length of packet

\*

\*/

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

/\*

\* Function: findHeaderLength

\* --------------------

\* This funciton will return length of header

\*

\* \*packet: String of packet content

\* length: length of packet

\*

\*/

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

/\*

\* Function: sendAck

\* --------------------

\* This funciton will return Ack number to ackBuffer

\*

\* sd: ?

\* flags: a mark to report if function works

\* seqNum:

\* sockaddr \*echoServAddr:

\* echoLen:

\*

\*/

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

int echoLen);

/\*

\* Function: lostAck

\* --------------------

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

\*

\* pro:

\*

\*/

int lostAck(float pro);

1. Difficulties
2. after client sent if final ack might be lost 🡪 server doesn't know whether the pkg is received or not 🡪 keep sending duplicated pkg. 🡪but client port is closed 🡪 wont send another ack 🡪 server send at most 16 pkg again 🡪 server terminate its port
3. 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 |
|  |  |