**Ashwin Suresh and Sagar Shah**

**Purpose**

This project is an implementation of go-back-N protocol that uses UDP as the underlying transport protocol. sock352.py contains all the methods and classes that are called by the client and the server classes. We also now implement a buffer to receive data from that instead of the file that is sent. This is an implementation of sliding buffer, only allowing to send enough data that will fit the buffer.

**Methods**

Init

Initializes the UDP socket with the IP address. If UDPportTx not defined, use same port for both receiving and transmitting. Finally, binds server to a specific receiving port to set constant address and sets the timeout to 0.2 seconds.

Bind

Binds the client to the server

Connect

Create a random sequence number and create a packet header with that number. Connect once Ack has been received, and perform a triple handshake. Finally, update the sequence number for next.

Listen

Listens for connection.

Accept

Call packetRead until we get new connection. Then, while connection is not established, server will constantly look for new packets every .2 sec. once flag = a SYN bit, then the current seqNo is first seqNo of first packet sent. Then, create another header and send an ACK to the client. Finally, return socket and address of client. Also create the window slider thread with a buffer lock.

Close

If client, call client close. Else, call server close.

ClientClose

Sets the timeout, and wait for ack. Then, send packet w terminating sequence number, and once receives ack from the other one, client closes with a termination connection handshake.

ServerClose

Does same thing for clientClose, but for server, also with a terminating handshake. It also sends the signal to recvToBuffer to terminate its endless loop.

Send

Starts two threads for sendData and ackData, then join and return. We noticed this worked for a lot of large files, but not all large files. We think this might be do to complications in threading.

SendData

Break up the message into packets of 31000 bytes each, create a lock, and send the data packet by packet. We only do this if there is enough space in the buffer to allow data to be sent.

AckData

If a packet is dropped, create a lock to go back the last sent ack and resend. If not, send the ack packet with the correct sequence number. Once data is acknowledged, update the slider with the appropriate space available.

Recv

Method for server receiving packet from client (nbytes is number of bytes we're receiving). Copy up to nbytes into a buffer, and return buffer if there is data. Data will only be copied if there is enough space in the buffer. Then, we only send totalData.

recvToBuffer

Check incoming packets until we receive correct seq num. Send ack w the seqNo from the client's packet back to client. If the buffer does not have space, then it will wait until there is space after more data is received. Once all packets have been sent, return all the bytes into savedFile.

HeaderMake

Used to create the header in the proper format before we send it. Create struct and update it with the five flags that can be modified.

PacketRead

Used to interpret the packet header. Get packet from the sender address. Fill in packet with header for the first 40 bytes, and data for the rest. For header specifically, unpack so we can see the relevant info we need in an array. Check the flags, and return necessary header information for each relevant flag. Finally, return the correct header for the relevant flag.