**Assignment 1- Draft Design**

**Packet structure**:

struct packet{

struct header;

char[1024];

}

**Header structure**:

struct header{

int seq\_num;

int type;

}

**Required header types:**

1. syn (used by client to initiate connection)

2. wait (sent to client to indicate it is queued)

3. go/send (sent to client to begin transmission)

4. data (used by client to indicate a data packet)

5. finish (used by client to indicate end of file/ used by receiver to indicate the connection is successfully closed)

6. ack/nack (sent to client)

**Packet Buffer:**

struct packet\_buffer window\_buffer[WINDOW\_SIZE];

struct packet\_buffer \* window\_buffer

struct packet\_buffer{

int received;

int seq\_num;

void \* packet;

}

**Response packet from Receiver:**

struct ack\_payload{

struct header;

int ack;

int num\_nak;

int \* num\_nacks;

}

**Window Size:** 100Mbytes equivalent to 102400 packets of 1024 bytes each

**Sender Process(ncp):**

To initiate:

Sender sends syn packet to the receiver, sets timer, wait for response from receiver or timer fires

if timer fires:

resend syn

elif receive response:

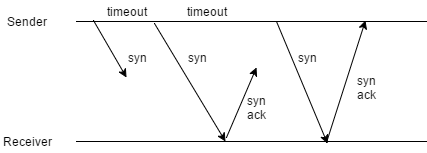
if response == wait:

wait(5 milliseconds)

resend

elif response == go:

senddata()



To send data:

Sender reads data from file (100Mbytes) into window\_buffer, craft into packets

send all packets in window, wait for response, set timer

if timer fires: resend the last window (or restart connection)

elif receive ack/nack from receiver

if naks:

1) resend missing packets

2) wait for ack/nack

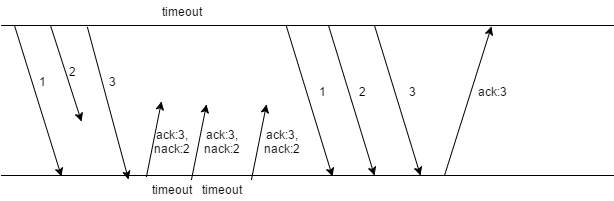
3) continue this until there are no nacks

else:

1) display the statistics

2) refill the buffers with the next window

3) repeat senddata()



To close connection:

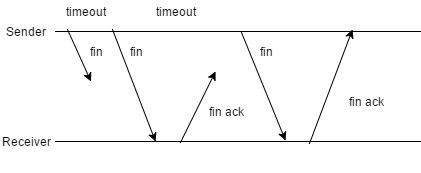
send finish header, set timer

if finish ack received:

shutdown

elif timer fires:

resend finish header



**Receiver:**

To establish connection:

Receives syn packet:

If connection is ongoing with the sender:

Discard the packet

Else if receiver is busy:

send wait packet

Else: send go packet, build connection

To receive data:

Queues all packets in a temporary buffer of size100Mbytes for processing:

if packet is already received: discard

elif seq\_num in window: copy packet into buffer with appropriate index into window buffer, mark seq\_num in window as received

check if all packets in window received:

if all received:

a) send ack

b) write buffer to file

c) update window

d) clear buffers

elif last packet in window received: send ack/nak

elif timer fires:

if received seq\_num in order:

a) send ack

b) write buffer to file

c) update window

d) clear buffers

else: send ack for highest number packet received and nacks for missing packets

To close connection: if receive fin packet: check if there is a connection with the sender and close connection; send finack