Phase 6
TCP

Group 4

Header Format

Packet Header (13 bytes):

1. First 2 bytes checksum.
2. Next byte is options by bit:
   - None, None, None, None, None, FIN ACK, FIN, SYN
3. Next 4 bytes is the sequence number.
4. Next 4 bytes is the ack number.
5. Flow control 2 bytes

Related Functions:

- `make_tcp_pkt(sequence, ackSeq = None, syn = False, fin = False, finAck = False, flow = None, data = bytearray())`
- `check_tcp(pkt, syn = False, fin = False, finAck = False)`

Three Way Handshake

- Follows a typical 3-way handshake protocol in which the client includes the header (info) packet in the third handshake message with the SYN bit unset.
- Both the client and server generate random starting sequence numbers from 0 to 1000.
- Header (info) packet contains the following:
  - Total number of packets in the transmission.
  - Name of the transmitted file.
  - Pkt length in bytes.
- Related functions:
  - `handshake_client(socket, addr, timeOut, headerPkt)`
  - `handshake_server(socket)`

Three Way Handshake

Receiver Output

```
Awaiting Connection...

First handshake message received with SYN option.

Second handshake message sent with SYN option.

Third handshake message received without SYN option and with header data packet.

Ack sent for third handshake message with header.

Sender random initial sequence number is: 598

Packet with sequence 598 received intact and stored.
Ack sent acknowledging packet with sequence: 598
Current free buffer space: 10370
```

Sender Output

```
Enter the name of the image you want to send (include file type extension): img2.jpg

The file img2.jpg contains 515505 bytes divided into 504, 1024 byte packets

Enter the name the image will be saved under (include file type extension): test.jpg

First handshake message with SYN option sent.

Second handshake message received with SYN option.

Third handshake message without SYN option sent including the first data packet (header).

Ack for third handshake message and header received.

Sender random initial sequence number is: 598

Packet with sequence number 598 sent
Currently in slow start mode
Window size is: 1
```

Flow Control

- The server maintains its own buffer array.
- The size of the buffer is currently 10370 bytes. (10x the message size).
- Server maintains a “buffIndex” variable which keeps track of the first empty buffer byte’s index.
- On every iteration where there is nothing to read from the socket the server pops a message from the buffer and calculates the remaining buffer space. Then it sends an ack including the buffer free space in the header.

Flow Control

- The client will not send a packet if the free buffer space in the last ack is smaller than the size of a packet.
- Client reduces free buffer space every time it sends a packet while its not receiving acks.
- Packets sent due to timeout are immune from the flow limitations. This allows the client to keep receiving buffer space updates even if all the sent packets or acks were lost/corrupted.
- Related Functions:
  - `send_gbn(clientSocket, addr, pktList, pktLength, sr=False)`
  - `receive_gbn(socket)`

Flow Control

Receiver Output

```
Packet with sequence 51901 received intact and stored.
Ack sent acknowledging packet with sequence: 30397
Current free buffer space: 1037

Packet with sequence 52925 received intact and stored.
Ack sent acknowledging packet with sequence: 30397
Current free buffer space: 1037

Packet with sequence 53949 received intact and stored.
Ack sent acknowledging packet with sequence: 30397
Current free buffer space: 2074

Packet with sequence 54973 received intact and stored.
Ack sent acknowledging packet with sequence: 30397
Current free buffer space: 3111

Packet with sequence 55997 received intact and stored.
Ack sent acknowledging packet with sequence: 30397
Current free buffer space: 4148

Packet with sequence 57021 received intact and stored.
Ack sent acknowledging packet with sequence: 30397
Current free buffer space: 5185
```

Congestion Control

- Slow Start:
  - Window size increases by 1 for every acked packet therefore the size of the window doubles every time a full window is acked.(multiplicative increase)
  - If timeout occurs, set ssthresh to N/2 and window size back to one (N=1).
  - Once the window size is equal or larger than ssthresh, switch to congestion control state.
- Congestion Control:
  - Window size increases by 1/N for every acked packet therefore the size of the window increases by one every time a full window is acked.(additive increase)
  - If timeout occurs, set ssthresh to N/2, window size back to one (N=1), and return to slow start state.
- Related Functions:
  - `send_gbn(clientSocket, addr, pktList, pktLength, sr=False)`

Congestion Control

Sender Output

```
Packet with sequence number 68377 sent
Currently in slow start mode
Window size is: 34


**TIMEOUT OCURRED IN GBN MODE, RESETTING NEXTSEQNUM TO BASE: 35609

Currently in slow start mode
Window size is: 1

Ack received for packet with sequence number: 34585
New Base is 35609 and Next Sequence to Send is 35609
Currently in slow start mode
Window size is: 2

Packet with sequence number 35609 sent
Currently in slow start mode
Window size is: 2

Packet with sequence number 36633 sent
Currently in slow start mode
Window size is: 2

Ack received for packet with sequence number: 35609
New Base is 36633 and Next Sequence to Send is 37657
Currently in slow start mode
Window size is: 3
```

Sender Output

```
Packet with sequence number 60185 sent
Currently in slow start mode
Window size is: 16

Ack received for packet with sequence number: 49945
New Base is 50969 and Next Sequence to Send is 61209
Currently in congestion avoidance mode
Window size is: 17

Packet with sequence number 61209 sent
Currently in congestion avoidance mode
Window size is: 17

Ack received for packet with sequence number: 50969
New Base is 51993 and Next Sequence to Send is 62233
Currently in congestion avoidance mode
Window size is: 17

Packet with sequence number 62233 sent
Currently in congestion avoidance mode
Window size is: 17

Packet with sequence number 63257 sent
Currently in congestion avoidance mode
Window size is: 17

Ack received for packet with sequence number: 51993
New Base is 53017 and Next Sequence to Send is 64281
Currently in congestion avoidance mode
Window size is: 17
```

Dynamic Timeout Calculation

- Every time a packet is sent, its send time is stored in an array.
- Every time an ack for a packet is received it compares the current time to the time that packet was sent and finds the time difference.
- This time difference is used to calculate a new timeout value using the following formula.

```
# Calculate the running RTT average
estimatedRTT = (1-alpha) * estimatedRTTPrev + alpha * sampleRTT

# Calculate the running average deviation from RTT
devRTT = (1-beta) * devRTTPrev + beta * abs(sampleRTT - estimatedRTT)

# Calculate safety factor
safetyFactor = 4 * devRTT

# Calcuilate the timeout interval for this iteration
timeoutInterval = estimatedRTT + safetyFactor
```

Dynamic Timeout Calculation

- Related Functions:
  - `send_gbn(clientSocket, addr, pktList, pktLength, sr=False)`
  - `time_out_update(sampleRTT, estimatedRTTPrev, devRTTPrev)`

Sender Output

```
Packet with sequence number 180498 sent
Currently in slow start mode
Window size is: 7

Current Timeout is: 0.018266597177324352

Ack received for packet with sequence number: 180498
New Base is 181522 and Next Sequence to Send is 181522
Currently in congestion avoidance mode
Window size is: 8

Packet with sequence number 181522 sent
Currently in congestion avoidance mode
Window size is: 8

Current Timeout is: 0.017173469785065138

Ack received for packet with sequence number: 181522
New Base is 182546 and Next Sequence to Send is 182546
Currently in congestion avoidance mode
Window size is: 8

Packet with sequence number 182546 sent
Currently in congestion avoidance mode
Window size is: 8

Current Timeout is: 0.015913033954378986
```

Connection Teardown

- The client begins by sending a message with the FIN option enabled to which the server responds with a message with the FINACK option enabled.
- Then the server sends its own message with FIN enabled to which the client responds with a FINACK message of its own.
- At this point both the client and server exit cleanly.
- Related functions:
  - `close_connection_client(socket, addr, seq, ackSeq, timeOut)`
  - `close_connection_server(socket, seq, timeOut)`

Connection Teardown

Receiver Output

```
Packet with sequence 515414 received intact and stored.
Ack sent acknowledging packet with sequence: 515414
Current free buffer space: 10961

Close connection first message with FIN option received.

Close connection second message with FINACK option sent.

Close connection third message with FIN option sent.

Close connection fourth message with FINACK option received.

File test.jpg received succesfully!

Transmission Time: 1.0637562274932861 seconds

Saved Succesfully!
```

Sender Output

```
Ack received for packet with sequence number: 515414
New Base is 516438 and Next Sequence to Send is 516438

Currently in congestion avoidance mode
Window size is: 21

Close connection first message with FIN option sent.

Close connection second message with FINACK option received.

Close connection third message with FIN option received.

Close connection fourth message with FINACK option sent.
```