P3)

```
  01010011
+ 01100110
----------
  10111001
+ 01110100
----------
  00101110
```

One's Complement = 11010001

The reciever will add all the 3 numbers together and then will the checksom to it as well. If this sum contains a zero, the receiver knows that there has to have been an error somewhere along communication.

This can detet all 1-bit errors, but 2-bit errors may go undetected.

P6)

Suppose the sender is waiting for call 1 from above and the receiver is waiting for 1 from below.

The sender sends a packet with seq. no. 1 and transitions to "Wait for ACK or NAK 1". waiting for an ACK or NAK.

Suppose now the receiver receives the packet with seq. no. 1 correctly, sends an ACK, and transitions to state "Wat for 0 from below", waiting for a data packet with seq. no. 0. But however the ACK is corrupted.

When rdt2.1 sender gets the corrupted ACK, it resends the packet with sequence number 1. However, the receiver is waiting for a packet with sequence nomber 0 and always sends a NAK when it doesn't get a packet with seq no 0. Thus they get stuck in a loop and neither will progress from that state.

P12)

The protocol would still work, since a retransmission would be what would happen if the packet received with errors has actually been lost.

~~To get a more subtle issue behind this~~

P14) In a NAK-only protocol, the loss of packet x is only detected by the receiver when packet x+1 is received. That is, the receivers receives x-1 and then x+1, only x+1 is required does the receiver realize that x was missed. If there is a long delay between the transmission of x and the transmission of x+1.

On the other hand, if data is being sent often then recovery under a NAK-only scheme could happen quickly.

P16) Yes. This causes the sender to send a number of pipelined data into the channel.

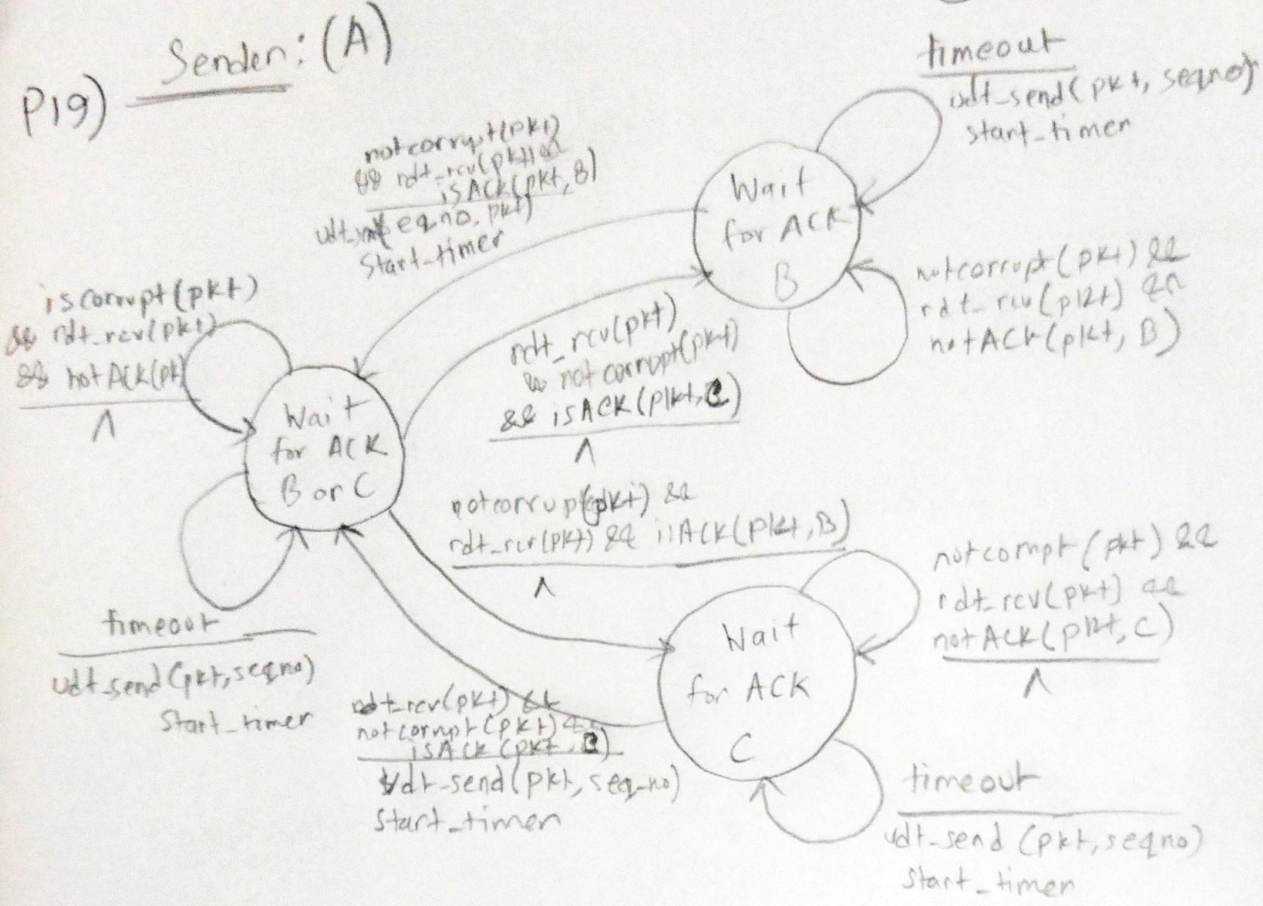
Yes. Here is one potential problem. If data segments are lost in the channel, then the sender of rdt 3.0 won't re-send those segments, unless there are some additional mechanism in the application to recover from loss.

P19) This is basically the same as rdt 3.0 but having 2 recievers instead of 1.

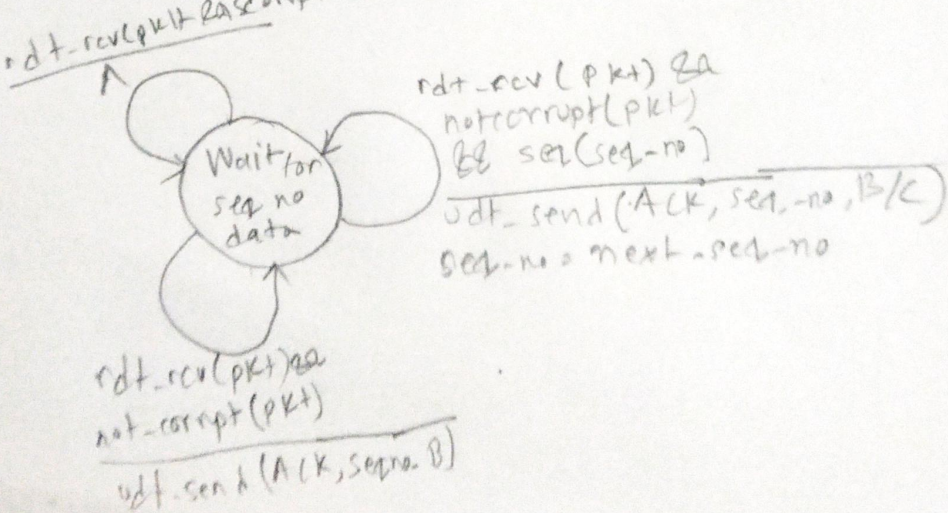
Here we have total of 3 states where we are waiting for the ACK of B or C then if we get the ACK for either B or C then we move to the state where we haven't gotten any ACK yet, either B or C.

Continued.

P19) Sender: (A)



Receiver: (B and C)



P26) (a) $2^{32} = 4.2$ Gigabytes

(b) Segments $= \frac{2^{32}}{536} \approx 8012999$

1 header is 66 bytes so

total headers $= 66 \times 8012999 = 528,857,934$ bytes

$\therefore$ total transmission $= 2^{32} + 528,857,934$ bytes

$= 4823825230$ bytes

on a 155 Mbps link:

time $= \frac{4823825230 \times 8}{155 \times 10^6} \approx 248.98$

≈ 249 secs

P27) (a) Seq no. = 207.

Source port = 302

Destination port = 80

(b) ACK no. = 207

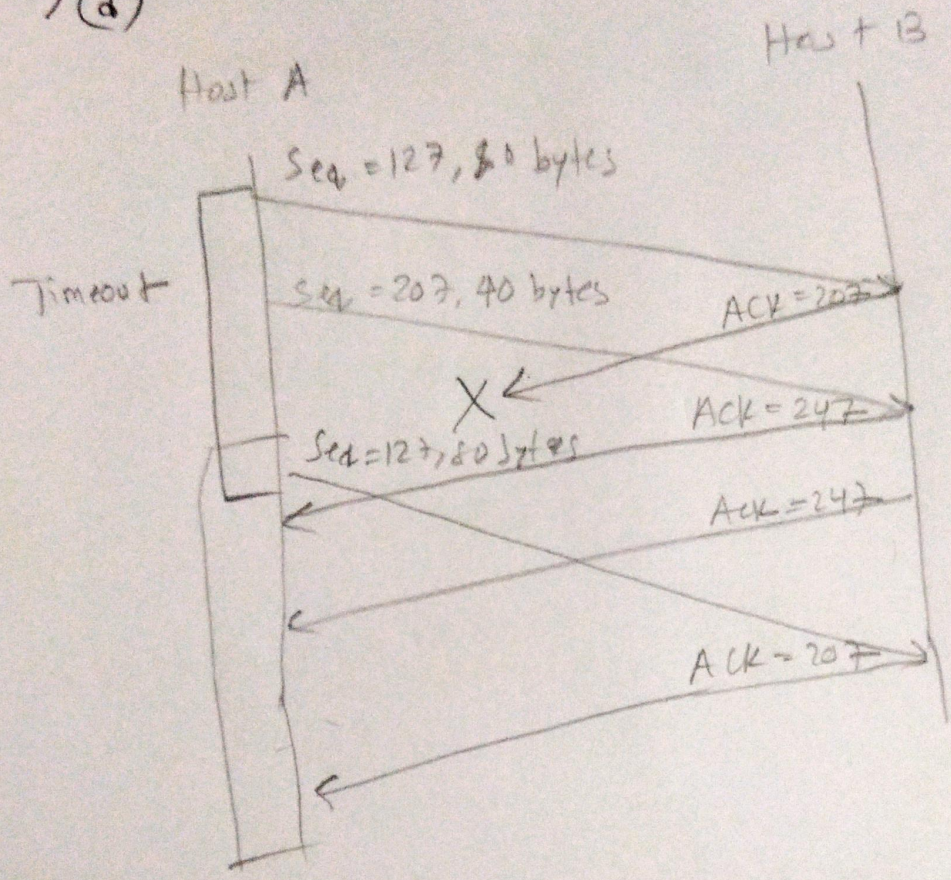
Source port = 80

dest. port = 302

(c) Ack. no. = 127

Continued

P26) (d)



P40) (a) [1, 6] and [23, 26]

(b) [6, 16] and [17, 22]

(c) Triple duplicate ACK.

(d) Timeout, since window size is 1

(e) 32

(f) 21

(g) $\lfloor \frac{29}{2} \rfloor = 14$

(h) 7th transmission round

(Continued.)

P40) (i) window size = 7
threshold = 4

(j) threshold = 21
window size = 1

(k) 1 + 2 + 4 + 8 + 16 + 21 = 52 packets

P44) (a) 12 - 6 = 6 RTTs

(b) $\frac{6+7+8+9+10+11}{6} = \frac{51}{6} = 8.5$ MSS/RTT

P48) (a) Total number of packets $= \sum_{n=0}^{W/2} \left(\frac{W}{2} + n\right)$

$= \frac{W}{2} \times \frac{W}{2} + \frac{\frac{W}{2}\left(\frac{W}{2}+1\right)}{2}$

$= \frac{3W^2}{8} + \frac{3}{4}W$

Loss Rate $= \frac{1}{\frac{3W^2}{8} + \frac{3W}{4}}$

P47) (b) for Large enough W

$$W^2 >> W \implies L \approx \frac{8}{3W^2}$$

or, $$W = \sqrt{\frac{8}{3L}}$$

$$\text{Throughput} = \frac{3}{4}\sqrt{\frac{8}{3L}} \cdot \frac{MSS}{RTT}$$

$$= \frac{1.22 \cdot MSS}{RTT \cdot \sqrt{L}}$$

P48) (a) $$\frac{W \times 1500 \times 8}{150 \times 10^{-3}} = 10 \times 10^6$$

$$\therefore W = 125 \text{ segments}$$

(b) $$\left\lceil \frac{\left(\frac{W}{2} + W\right)}{2} \right\rceil = \left\lceil \frac{3}{4} W \right\rceil = \left\lceil \frac{3}{4} \times 125 \right\rceil = 94$$

$$\text{Average Throughput} = \frac{94 \times 1500 \times 8}{150 \times 10^{-3}}$$

$$= 7.52 \text{ Mbps}$$

(c) $$W - \frac{W}{2} = 125 - 62 = 63$$

1 window increase = 1 RTT

$$\therefore 63 \times 150 \times 10^{-3} = 9.45 \text{ secs.}$$