

Solutions to the Midterm Exam

Problem 1

- (a) transport layer and application layer
- (b) The transmission delay is the delay to transmit all the packet's bits into the link, and the propagation delay is the time a packet needs to traverse a link.
- (c) A network application consists of many components such as a user agent, a protocol, etc. whereas an application layer protocol is just a part of a network application, namely the one that is responsible for the exchange of messages.
- (d) HTTP (for example)
- (e) HTTP: 80, SMTP: 25, Telnet: 23 (for example)
- (f) Yes.
- (g) Pipelining saves a round-trip time that would have otherwise been wasted by waiting for an entire object to arrive before requesting the next one.
- (h) By attaching source and destination port numbers to each message. This enables it to deliver the messages to the right processes.
- (i) The third.
- (j) To make sure that a simple reordering of a pair of packets in the channel (which would cause 2 duplicate acknowledgements) can be handled without retransmissions.

Problem 2

The delay is

$$\left(\frac{S+h}{R}\right)\left(\frac{F}{S} + \ell - 1\right) = \frac{1}{R}\left(F + (\ell - 1)(S + h) + \frac{Fh}{S}\right). \quad (1 \text{ point})$$

Taking the derivative of this function and setting it to 0 gives

$$(\ell - 1) - \frac{Fh}{S^2} = 0 \quad \Rightarrow \quad S = \sqrt{\frac{Fh}{\ell - 1}}. \quad (1 \text{ point})$$

Taking the second derivative gives $2Fh/S^3$, which is greater than 0 and therefore $S = \sqrt{Fh/(\ell - 1)}$ minimizes the delay. (1 point)

Problem 3

- (a) Minimum possible timeout interval: $2d + t$.
- (b) Average time needed by sender:

$$(2d + t) + \frac{1 - p}{p} \cdot (2d + t) = \frac{2d + t}{p} \quad (1 \text{ point})$$

- (c) $(2d + t)/p = 3/0.8 = 3.75$. Since $3.75 < 4$, the transport layer can be expected to keep up with transmission rate of application layer.

Problem 4

- (a) When the receiver is in the state “wait for 1 from below” and it receives a packet whose sequence number is 0, it will simply send a NAK to the sender. The sender, who is still in the state “wait for ACK or NAK 0” (because the previous ACKs from the receiver got lost), receives this NAK and will simply retransmit the sequence 0 packet without changing the state. Hence, the receiver can never receive a sequence 1 packet, and it forever stays in that state, while the sender will forever stay in its state too. (2 points)
- (b) To correct this flaw, we can change the receiver side by replacing in state “wait for 1 from below” the self-loop event with the two self-loop events

- `rdt_rcv(rcvpkt) && corrupt(rcvpkt): compute chksum,
make_pkt(sndpkt,NAK,chksum), udt_send(sndpkt)` (1 point)
- `rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && has_seq0(rcvpkt): compute chksum,
make_pkt(sndpkt,ACK,chksum), udt_send(sndpkt)` (1 point)

Problem 5

$$\begin{aligned} & 1011010011101000 \\ + & 0110111011000111 \\ = & 0010001110101111 \quad \text{plus an overflow bit} \end{aligned}$$

which gives, when added back to the rightmost bit, 0010001110110000. (1 point)

$$\begin{aligned} & 0010001110110000 \\ + & 1110011100111000 \\ = & 0000101011101000 \quad \text{plus an overflow bit} \end{aligned}$$

which gives, when added back to the rightmost bit, 0000101011101001. (1 point)

The 1's complement finally results in 1111010100010110. (1 point)

Problem 6

- (a) In order to avoid problems with the sequence numbers, we want to avoid having the leading edge of the receiver's window (i.e. the one with the highest sequence number) wrap around in the sequence number space and overlap with the trailing edge (i.e. the one with the lowest sequence number) in the sender's window. Since for both protocols the receiver window can be by $w - 1$ numbers ahead of the sender window, both protocols need a sequence number space that is at least twice as large as the window size, $k \geq 2w$.
- (b) Go-Back-N uses cumulative acknowledgements, whereas Selected Repeat uses acknowledgements for individual packets.
- (c) TCP uses cumulative acknowledgements.
- (d) No.
- (e) $C \rightarrow S: SYN = ACK = 1, seq = client_sn, ack = server_sn$
 $C \leftarrow S: SYN = ACK = 1, seq = server_sn, ack = client_sn + 1$
 $C \rightarrow S: ACK = 1, seq = client_sn + 1, ack = server_sn + 1$

Problem 7

It takes the slow start phase k ms to go from 1 to 2^{k-1} segments per ms, which is the first time the threshold is exceeded. Thus, the number of segments sent in the slow start phase is

$$\sum_{i=0}^{k-1} 2^i = 2^k - 1 \quad (1 \text{ point})$$

Afterwards, the number of segments sent in 1 ms just increases by one after each ms. Hence, it takes 2^{k-1} ms to go from $2^{k-1} + 1$ to 2^k segments per ms. During that time, a total number of

$$\sum_{i=1}^{2^{k-1}} (2^{k-1} + i) = 2^{k-1} \cdot 2^{k-1} + \sum_{i=1}^{2^{k-1}} i = 2^{2(k-1)} + \frac{2^{k-1}(2^{k-1} + 1)}{2} \quad (1 \text{ point})$$

segments are transmitted. Both phases together cover $k + 2^{k-1}$ time steps. Hence, the average transmission rate is

$$\frac{(2^k - 1) + 2^{2(k-1)} + 2^{k-2}(2^{k-1} + 1)}{k + 2^{k-1}} \approx \frac{3}{2} \cdot 2^{k-1} \quad (1 \text{ point})$$

which is actually quite close to the maximum possible rate of 2^k .