CMPT 371: Midterm 1 Solutions

July 6, 2006

1 Short-Answer Questions

1. Name the five layers of the network protocol stack we have been covering in top-down order.

   Solution: Application, Transport, Network, Link, Physical

2. True or false:
   (a) SMTP is used to download mail from a mail server to a local host.
   (b) UDP provides a guarantee of delivery.
   (c) One DNS record might appear in multiple DNS servers.
   (d) Losing a segment does not cause the timeout value to change in TCP.

   Solution: FFTT

3. Who inserts records into a DNS database?

   Solution: An official domain name registrar, which are the organizations that sell domain names.

4. Consider two nodes x and y connected by a link L. Node x is forwarding a packet to node y.
   (a) Does the transmission delay at node x depend on (i) the size of the packet, (ii) the length of the link, or (iii) both?
   (b) Does the propagation delay at node x depend on (i) the size of the packet, (ii) the length of the link, or (iii) both?

   Solution: (a): (i); (b): (ii)

5. What does AIMD stand for? Briefly describe what it means.

   Solution: It stands for "Arithmetic Increase, Multiplicative Decrease." It refers to the way TCP congestion control adjusts the sending rate of the sender. When there is no loss, the sender's window increases by 1 MSS every RTT. As soon as a loss is detected, the sender's window size is cut in half.

6. Briefly describe the difference between Go-Back-N and selective repeat.

   Solution: Both are types of reliable data transfer pipelining. In GBN, there is a timer associated with the first sent, but not acknowledged packet (the sendbase). On timeout, all sent but not acknowledged packets (there can be at most N of them) are resent. The sender discards out-of-order ACKs and the receiver discards out-of-order packets. In selective repeat, there is a timer associated with each sent, but not acknowledged packet (again there can be N of them). If one of the timers times out, then only that packet is resent. The sender keeps track of out-of-order ACKs and the receiver keeps track of out-of-order packets.

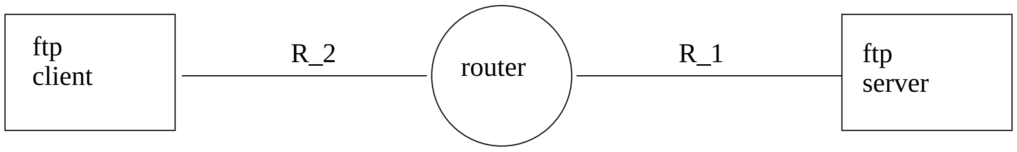
2 Longer Questions

1. Recall that ftp is a protocol for transfering files. To invoke it, the user types "ftp hostname" where hostname is the full name (including domain name) of an ftp server. The server responds by presenting a prompt to the client (assume the client doesn't need to log in). The client can then get a file by typing "get filename."

   (a) How many packets does the client send out before the server starts sending the file (hint: the client may not send all its packets to the ftp server)? Explain your answer and state any assumptions you are making.

   Solution: Let's assume that the DNS entry for the server is stored in the client's local DNS server. Therefore, the client must send out one (UDP) packet to the local DNS server. Then, it must initiate a TCP connection with the ftp server (2 packets), and finally make the request for the file (1 packet). So the total is 4 packets. This is basically a minimum number. The DNS query might take more packets and, since ftp uses two TCP connections, there might be more packets because of that. Anything along those lines should have received credit.

   (b) Assume there is one router and two links between the ftp server and client (see picture below). The first link has transmission rate R_1 and the second link has transmission rate R_2. Assume the file gets broken into three packets each of size L. We will ignore all propagation and processing delays. How long does it take from when the server starts sending the file until the client has received the whole file if $R_2 \geq R_1$? What if $R_2 < R_1$? In the second case, how long does the second packet spend in the router's queue?



   Solution: This question implicitly assumes that the sender is pipelining, since, of course, ftp uses TCP and that is what TCP does in reality. Also, there would be no distinction between the two cases if there were no pipelining.

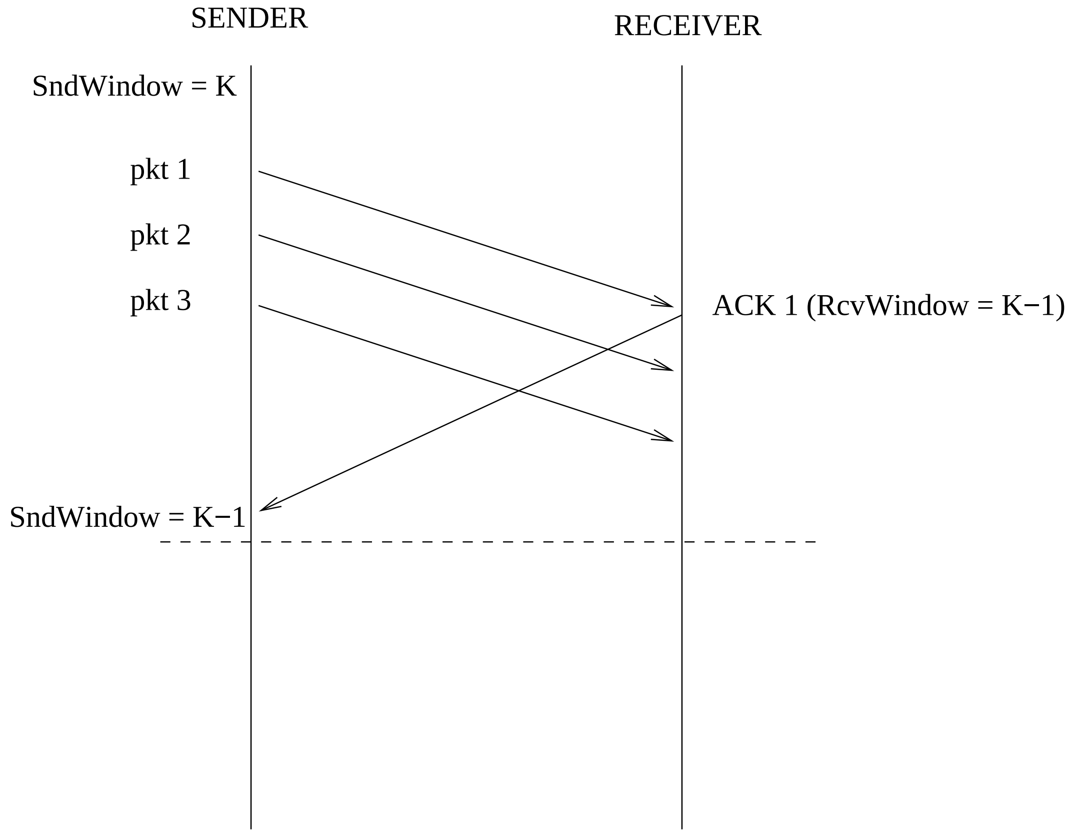
   Start with the case $R_2 \geq R_1$: the third packet reaches the router at time $3L/R_1$. Then it's just a matter of transmitting that packet to the client, which takes L/R_2. So, in total, $3L/R_1 + L/R_2$. **Note:** If points were deducted because you added the term L/R_2, that was my fault. You are entitled to a re-mark.

   In the second case, the argument is similar. At time L/R_1, the first packet is at the router. At time $L/R_1 + L/R_2$, the first packet is at the client and the second packet is ready to go at the router. So at time $L/R_1 + 3L/R_2$, the third packet arrives at the client. Again, if points were taken off for the L/R_1 term, you are entitled to have them back.

   Finally, to calculate the queuing time of the second packet at the router. We said that at time $L/R_1 + L/R_2$, the second packet is ready to be transmitted by the router. On the other hand, the second packet arrives at the router at time $2L/R_1$. So it waits for $L/R_1 + L/R_2 - 2L/R_1 = L/R_2 - L/R_1$.

2. Recall how TCP flow control works: whenever the receiver sends an ACK, it includes a parameter `RcvWindow`, the amount of space it has left in its buffer. When the sender receives this ACK, it sets `SndWindow`, the size of its send window, to `RcvWindow`. Assume all packets have size 1 and that the receiver's buffer has size K and that it never empties its buffer.

   (a) Starting from the dotted line in the following diagram, how many packets is the sender allowed to send before it receives another ACK?



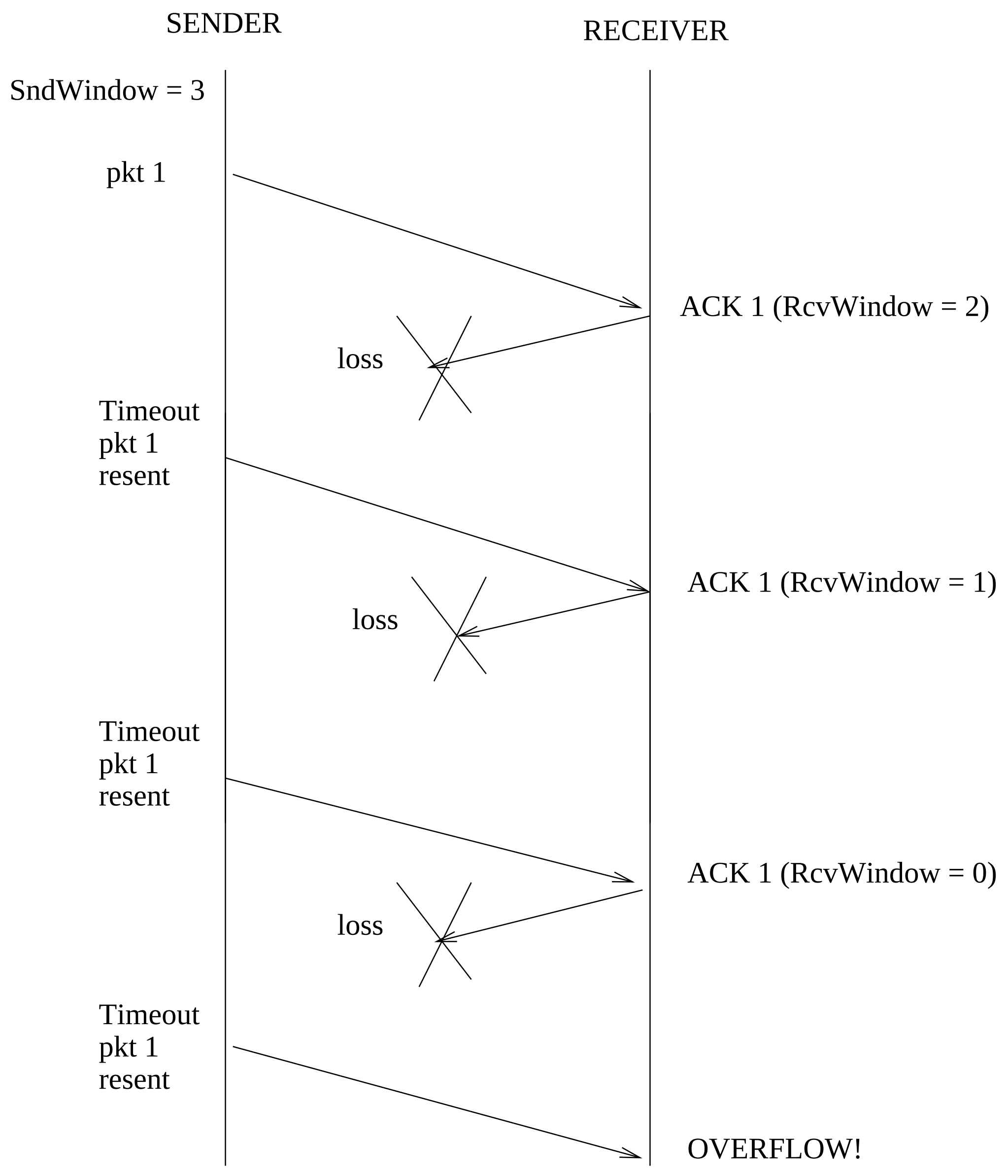
   Solution: The sender's window has size $K - 1$ at the dotted line, but the sender has sent 2 packets which have not been acknowledged, so it can send $K - 3$ packets before it gets another acknowledgement.

   (b) Assuming no packet loss, corruption or reordering, explain why it is never possible for the sender to overflow the receiver's buffer.

   Solution: The problem here is that it might take a lot of time for an ACK to get from the receiver to the sender. Therefore, the value of `RcvWindow` that the receiver includes in an ACK might be outdated by the time it gets to the sender. Of course, this is not really a problem because when the sender receives an ACK and sets `SendWindow` to `RcvWindow` it counts those packets that have been sent but not acknowledged as part of the send window. Hence, the number of packets sent starting at the time that the receiver's buffer shrinks to size, say, ℓ, is at most ℓ.

(c) Assume packets can be lost and that the receiver is stupid and doesn't check whether a received packet is a duplicate of a previous packet before it puts the packet in its buffer. Draw a diagram like the one above that shows that the sender can overflow the receiver's buffer (you can pick a value for K).

Solution:
K = 3


SENDER
RECEIVER
SndWindow = 3
pkt 1
ACK 1 (RcvWindow = 2)
loss
Timeout
pkt 1
resent
ACK 1 (RcvWindow = 1)
loss
Timeout
pkt 1
resent
ACK 1 (RcvWindow = 0)
loss
Timeout
pkt 1
resent
OVERFLOW!