Midterm Exam SOLUTION

CMPSCI 591 and 453 / ST550A: Computer Networks
Spring 2002
Prof. Jim Kurose

Question 1: ``Quickies'' Answer each of the following questions *briefly, i.e., in at most a few sentences.*

a) Suppose all of the network sources send data at a constant bit rate. Would packet-switching or circuit-switching be more desirable in this case? Why?
   Either packet switching or circuit switching would work well here. Both would utilize network resources equally well. Circuit switching would utilize resources well because the traffic is constant rate, and thus a circuit would always be busy carrying traffic.
   Now suppose that all of the network sources are bursty – that they only occasionally have data to send. Would packet-switching or circuit switching be more desirable in this case? Why?
   Packet switching is more desirable because it provides statistical multiplexing, and can support more network sources with only a small probability of congestion.

b) What does it mean for HTTP1.1 to be "persistent"? What is the value of HTTP1.1 being persistent, as opposed to being non-persistent, as in HTTP 1.0?
   HTTP1.1 keeps a single TCP connection open while transferring multiple objects over the connection. The advantage of re-usuing the single connection is that one does not have to set up multiple connections serially, with the ensuing startup delays due to the TCP triple handshake.

c) In DNS, what is the role of (a) the local DNS server, (b) a root DNS server, (c) the authoritative DNS server?
   The local DSN server receives DNS queries from a local process, and either contacts other DNS servers to resolve a name or returns the IP address immediately if it is in its cache. A root DNS server can always be contacted to resolve any addresses; it will either have or know how to find the name-to-address translation for any name. The authoritative DNS server is the sole location in the Internet that is responsible for defining and storing (always) the name-to-address mapping of names for which it is the authoritative server.

d) What is the purpose of the connection-oriented welcoming socket, which the server uses to perform an *accept()*? Once the *accept()* is done, does the server use the welcoming socket to communicate back to the client? Explain.
   The welcoming socket is used in a server to wait for incoming client connection requests in connection-oriented (TCP-based) communication. A new socket is created on return from the accept() and it is this new socket that is then used by the server to communicate back to the client.

e) What is meant by *demultiplexing* a protocol data unit up to an upper-level protocol?
   Demultiplexing refers to passing an incoming protocol data unit (PDU) up to the appropriate higher level protocol. The PDU will typically have a field that will indicate the upper layer protocol to which the PDUs payload should be passed.

f) Briefly describe TCP's slowstart algorithm. What causes the TCP slowstart algorithm to end, and TCP congestion avoidance to begin?
   TCP's congestion window is increased by one (one segment's worth of bytes) for each ACK received. This causes the number of segments sent per RTT to increase exponentially. Slowstart ends when a threshold window size is reached.

Question 2: A reliable data transfer protocol

In class, we have seen a number of mechanisms used to provide for reliable data transfer:

- checksum
- ACKs
- timers
- sequence numbering

Consider a sender and receiver that are connected by a sender-to-receiver channel that can corrupt and lose packets. The receiver-to-sender channel is perfect (i.e., it will not lose or corrupt packets). The delay on each channel is known to always be less than some maximum value, *d*. Neither channel will reorder packets. [Note: re-read the channel properties just described and make sure you understand them!] You are to design a reliable data transfer protocol for this scenario *using only those mechanisms (among the four listed above) that are absolutely required*. That is, if you use a mechanism that is not required, your answer will not be given full credit, even if the protocol works. Your protocol should be as simple as possible but have the functionality to reliably deliver data under the stated assumptions. Your solution need not be efficient; it must work correctly.

a) Draw the sender and receiver FSMs.

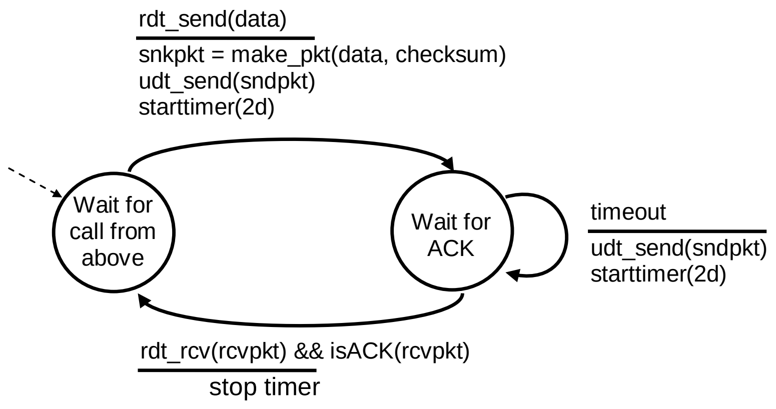
b) For each of the mechanisms (from among the four list above) that you use in your protocol, explain the role/purpose of the mechanism and why can't you get by without it? [Note: I do not mean to imply here that your protocol will use all four mechanisms above (maybe it does; maybe it does not). I am asking you to explain why you need the mechanisms that you have chosen to include in your protocol]

Because the sender-to-receiver channel can corrupt packets, the data-sent on the sender-to-receiver channel will need a checksum to detect bit errors. Because the sender-to-receiver channel can lose packets, we will need to have a timer to timeout and retransmit packets that have not been received by the receiver. The receiver will need to indicate which packets it has received by using an ACK message; if a packet is not received or is received corrupted, no ACK is sent. Because the maximum delay of the channel is bounded at d, the sender can set its timeout value to 2d, and thus only retransmit when it is certain that a retransmission is needed (and expected by the receiver). Thus there is no need for sequence numbers, since there will be no unneeded (and unexpected at the receiver) retransmissions.

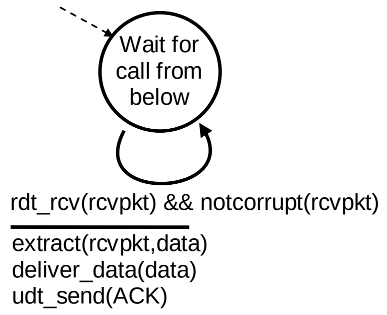
The FSM for the sender and receiver are shown below.

As an example of the mistakes in answers to watch out for:

- *Do not need to use sequence numbers*
- *Do not need checksums on receiver-to-sender channel*
- *Setting timer to 2d, and appropriately stopping and restarting channel*
- *Separate FSMs are needed for the sender and the receiver!*
- *Do not need NAKs (absence of ACK serves as a NAK)*



(a) sending side



(b) receiving side

Problem 3: Congestion Control and TCP (23 points, 20 minutes)

a) What is the difference between congestion control and flow control?

   Flow control is about matching the speed of a sender to the capabilities of the receiver. Congestion occurs when senders overutilize the resources within the network.

b) It is said that a TCP connection "probes" the network path it uses for available bandwidth. What is meant by that?

   TCP keeps increasing its send rate (by increasing its window size) until loss occurs (at which point congestion has set in). TCP then sets its rate lower but again begins increasing its send rate to again determine the point at which congestion sets in. In this sense, TCP is contantly probing the network to see how much bandwidth it can use.

c) Suppose that in TCP, the sender window is of size N, the base of the window is at sequence number x, and that the sender has just sent a complete window's worth of segments. Let RTT be the sender-to-receiver-to-sender round trip time, and let MSS be the segment size.

   i. Is it possible that there are ACK segments in the receiver-to-sender channel for segments with sequence numbers lower than x? Justify your answer.

      It is possible. Suppose that the window size is N=1. The sender sends packet x-1, which is delayed and so it timeouts and retransmits x-1. There are now two copies of x-1 in the network. The receiver receives the first copy of x-1 and ACKs. The receiver then receives the 2nd copy of x-1 and ACKs. The sender receives the first ACK and sets it window base to x. At this point, there is still an ACK for x-1 propagating back to the sender.

   ii. Assuming no loss, what is the throughput (in packets/sec) of the sender-to-receiver connection?

      Assume that N is measured in segments. The sender can thus send N segments, each of size MSS bytes every RTT secs. The throughput is this $N \cdot MSS/RTT$.

   iii. Suppose TCP is in its congestion avoidance phase. Assuming no loss, what will the window size be after the N segments are ACKed?

      N+1

d) Consider the use of TCP over a wireless channel, which is much more prone to bit-level errors than a wire or optical fiber. Comment on how well suited TCP's (i) error detection mechanism, and (ii) its window-based congestion control algorithm are for such a bit-error-prone environment.

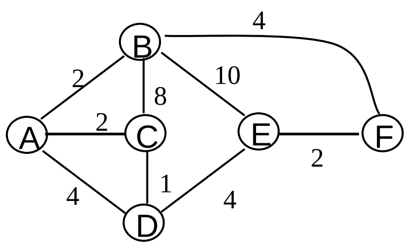
   TCPs checksum method is not very strong (compared to other techniques such as CRC) in detecting bit level errors.

   TCPs window based congestion control algorithm interprets the absence of an ACK as an indication of congestion (buffer overflow) in the network. If the receiver is not sending ACKs because of bit level corruption (which is not related to congestion control), then the sender will incorrectly interpret the absence of ACKs as a congestion indication and will unnecessarily decrease its window. Thus the window based congestion control algorithm is not well suited for an environment in which packets can be lost due to bit level errors.

Question 4: Routing Algorithms

a) Consider the network shown below. Show the operation of Dijkstra's (Link State) algorithm for computing the least cost path from **D** to all destinations.

N	D(A),p(A)	D(B),p(B)	D(C),p(C)	D(E),p(E)	D(F),p(F)
D	4,A	infty	1,C	4,E	infty
DC	3,C	9,C		4,E	infty
DCA		5,A		4,E	infty
DCAE		5,A			6,E
DCAEB					6,E
DCAEBF					



b) Consider a node Z, which has only two neighbors – X and Y. The link cost from Z to X is 2 and the link cost from Z to Y is 3. Suppose X and Y have the distance tables shown below, which they send to Z.

D^X	s_1	s_2	s_3
f	13	$\textcircled{2}$	4
g	$\textcircled{5}$	7	9
…			

D^Y	t_1	t_2	t_3
f	6	$\textcircled{5}$	8
g	$\textcircled{2}$	9	7
…			

Complete the following table in node Z after it receives the distance tables from its neighbors X and Y:

D^Z	X	Y
f	4	8
g	7	5
…		

$D^Z(f,X) = c(Z,X) + mincost(X,f) = 2 + 2 = 4$
$D^Z(g,X) = c(Z,X) + mincost(X,g) = 2 + 5 = 7$
$D^Z(f,Y) = c(Z,Y) + mincost(Y,f) = 3 + 5 = 8$
$D^Z(g,Y) = c(Z,Y) + mincost(Y,g) = 3 + 2 = 5$