

Solution to Question 1: ``Quickies'' (25 points, 15 minutes)

- What is meant by the term *statistical multiplexing*?
Answer: In statistical multiplexing, data from multiple users (senders) is sent over a link. If bandwidth is not used by one user, it is then free to be used by other users. Thus, senders share the link bandwidth, with no user having all of the link bandwidth allocated to it
- What are two key differences between a *datagram network* and a *virtual-circuit network*?
Answer: In a VC network, routers have *state* for each VC that passes through the router; a datagram network is stateless. In a VC network, the network-layer packet carries a VC id rather than a source/destination address. In a VC network, all traffic between a given source/destination pair follows the same path; in a datagram network, different packets can take a different route.
- Consider an http client that wants to retrieve a WWW document at a given URL. The IP address of the http server is initially unknown. The WWW object at the URL has one embedded GIF image that resides at the same server as the original object.
 - What transport and application layer protocols besides http are needed in this scenario?
Answer: DNS is needed to determine the 32-bit IP address of the server. TCP is used to carry the HTTP request; UDP is used to transport the DNS messages.
 - Suppose that the time needed to contact and receive a reply from any server (for any protocol) is RTT. How many RTT's are needed from when the user first enters the URL until the complete document is displayed? Assume that non-persistent http is used. Consider the delays of all protocols in your answer, not just those of http.
Answer: one RTT to do DNS, one RTT set up 1st TCP connection to WWW server; one RTT to get object, one object to set up 2nd TCP connection to WWW server, one RTT to get GIF image
- How are ports numbers used by TCP and/or UDP in demultiplexing incoming segments?
Answer: the receiver looks at the sender IP address, the sender port number and the receiver port number to determine which socket the arriving message should be demultiplexed to.
- Which protocol – Go-Back-N or Selective-Repeat - makes more efficient use of network bandwidth? Why?
Answer: Selective repeat makes more efficient use of network bandwidth since it only retransmits those messages lost at the receiver (or prematurely timed out). In go-back-N, the sender retransmits the first lost (or prematurely timed out) message as well as all following messages (without regard to whether or not they have been received).

Question 2: Transport Layer Potpourri (24 points, 20 minutes)

- Suppose the sender and receiver in a pipelined reliable data transfer protocol have a window of size N . Suppose the sequence number of the segment at the base of the window at the receiver is x . What are the possible range of sequence numbers in the sender's window? Justify your answer.

Answer: If the receiver window base is x , then it has acked the previous N packets. However, those ACKs may not have been received by the sender. Hence the sender window could still be $[x-N+1, x]$. On the other hand, if all of the ACKs have been received at the sender, then the sender's window would be $[x, x+N-1]$. Hence the range is $[x-N+1, x+N-1]$.

- Suppose that a reliable data transfer protocol such as TCP has an RTT estimate that is far too small. What is the consequence? Now suppose the RTT estimate is far too large. What is the consequence?

Answer: if the RTT estimate is far too small, the retransmission timer will be too small and will timeout prematurely – resulting in the retransmission of packets that may not be lost. If the RTT estimate is far too large, the retransmission timer will be too large. In this case, the packet will not be transmitted until a long time after it has been lost – resulting in long error recovery delays.

- What is the role of the threshold in TCP congestion control? How does it change over time?

Answer: When the TCP window, W , is larger than the threshold, TCP is in congestion avoidance and grows the window slowly (at the rate of $1/W$ per packet acknowledged, or equivalently, at the rate of 1 packet per RTT). Note that on packet loss the threshold value is halved.

- Consider two TCP senders. They are at different sending hosts and go to different destinations, but pass through a *common* bottleneck link (that is the only bottleneck link on either of their paths). What does it mean to say that TCP provides fair sharing of bandwidth at the bottleneck link? Suppose the RTTs of the two connections are very different. Is TCP “fair” in this case? Justify your answer.

Answer: TCP is fair is that in the long term, each receiver will eventually see the same throughput at that link. As noted in the answer above, the TCP window size grows at a rate that is proportional to the RTT. If the RTT's are different, then the two TCP senders will grow their windows at different rates (even if they start with the same window size).

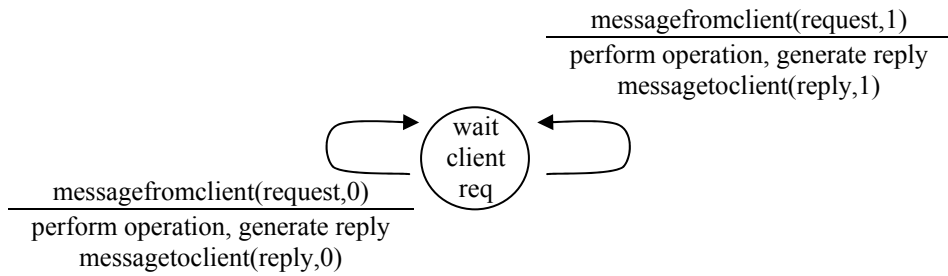
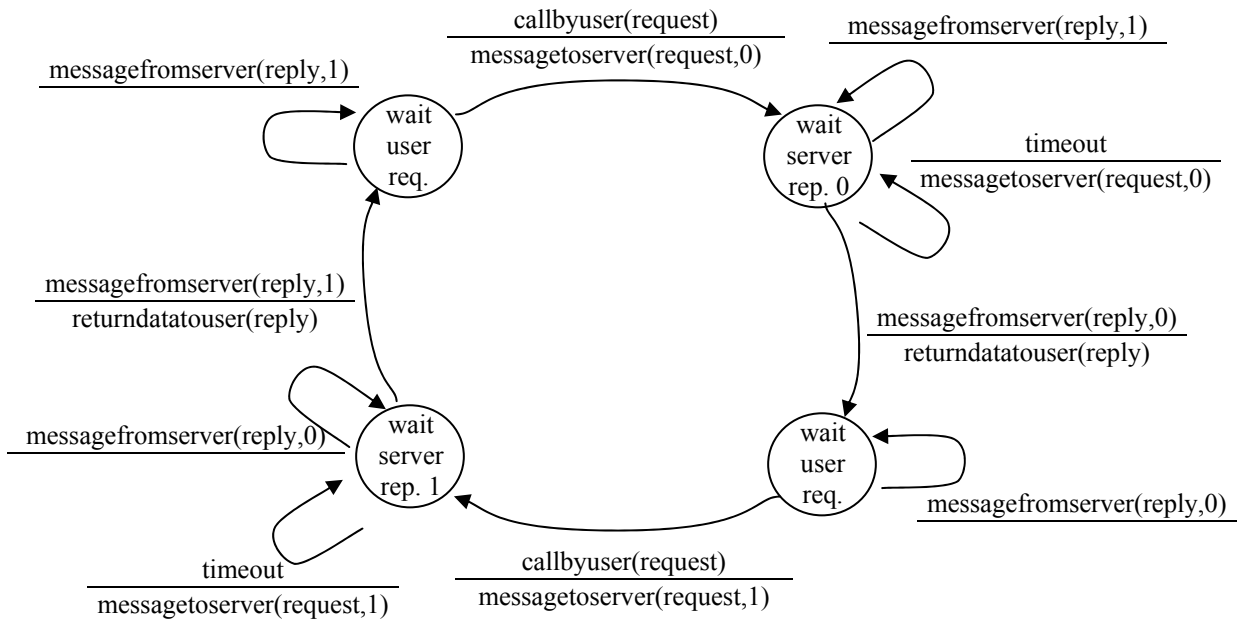
Question 3: A distributed transaction processing system (26 points, 20 minutes)

Consider a distributed transaction processing system of a client and a remote server. The client receives transaction requests from local users. These transaction requests must be communicated to the server, which will execute the transaction request and return the result of the transaction request. (You can think of a transaction as requesting an account balance from the server database, and the response containing the account balance. You can think of the transaction as being read-only – that is, executing a transaction will not change the state of the server database). *The client and server communicate over a medium that can lose and delay messages; the maximum delay in the medium is not known.* The medium will *not* corrupt or reorder messages.

The client should receive requests from local users (via the event *callbyuser(request)*) and return results to users (via the event *returndatatouser(data)*) in the order in which the requests were generated. The server receives messages from the client via the *messagefromclient(clientmsg)* event and sends messages to the client via the *mesagetoclient(servermsg)* event, where *clientmsg* and *servermsg* are messages (that you define) sent from the client and server, respectively. The client receives messages from the server via the *messagefromserver(servermsg)* event.

Give a FSM description of the client and server. Describe the format of the messages sent from client-to-server and from server-to-client. Your protocol should be minimalist in the sense that it should not contain any functionality that is not strictly needed to meet the above requirements.

Answer: the solution to this problem is essentially the stop and wait protocol. We need a timer because messages can be lost and the maximum delay is not bounded. The client will have the timer and will send copies of the previous request on timeout; the server does not need a time. Since there may be premature timeouts at the client, there will be duplicate messages, and the client will need to use the sequence to make sure that it returns the data to the user only once. A checksum is not needed since messages can not be corrupted. The FSMs are shown on the following page.

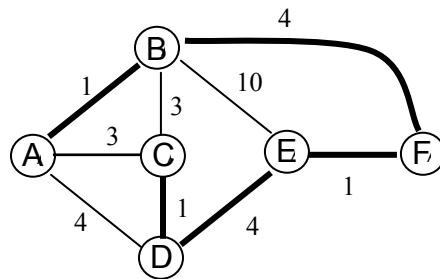


Question 4: Routing Algorithms (25 points, 15 minutes)

Consider the network shown below.

- Show the operation of Dijkstra's (Link State) algorithm for computing the least cost path from F (the *rightmost* node in the figure below!) to all destinations. Also, explicitly list the shortest path routes from E to all destinations that are the result of the algorithm's computation.

Show the distance table that would be computed by the distance vector algorithm in B. Note: you do not have to run the distance vector algorithm; you should be able to compute the table by inspection.



<u>N</u>	<u>A</u>	<u>B</u>	<u>C</u>	<u>D</u>	<u>E</u>
F	infty	4,F	infty	infty	1,F
FE	infty	4,F	infty	5,E	
FEB	5,B		7,B	5,E	
FEBD	5,B		6,D		
FEBDA			6,D		
FEBDAC					

The distance table in B is shown below.

		via			
		A	C	E	F
dest	A	1	6	16	9
	C	4	3	15	10
	D	5	4	14	9
	E	7	8	10	5
	F	6	9	11	4