Final Exam

Computer Networks

Fall 2014

Prof. Cheng-Fu Chou

Question 1: Transport Layer(20%)

Consider the following plot of TCP window size as a function of time for two TCP connections A and B. In this problem we will suppose that both TCP senders are sending large files. We also assume that the packet loss events are independent in connection A and B.

a) (5%) Considering the above values of congestion window (CongWin) for these connections, can we identify the type of TCP connections (Reno or Tahoe) that have been used by connection A and B? Justify your answers.

Ans: Considering the different changes of CongWin in the 6th and 12th transmission rounds, connection A uses TCP Reno, whereas we cannot say that connection B uses TCP Reno or Tahoe.

b) (5%) What are the values of the Threshold parameter between the 1st and the 14th transmission rounds for each connection?

Ans: Connection A: The value of Threshold is 8 between the first and the sixth transmission round. It is 5 between the sixth and the fourteenth transmission round.

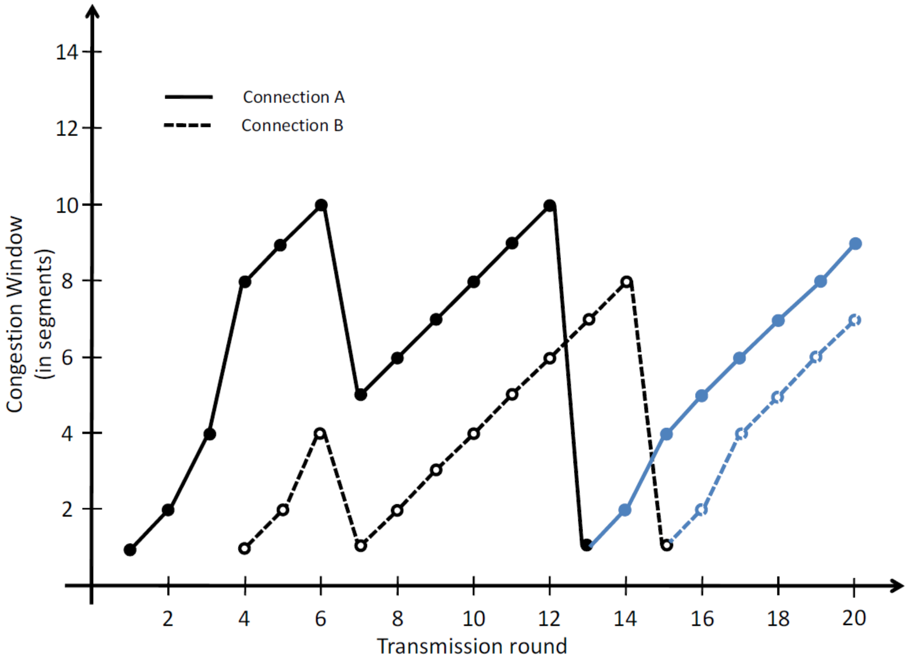
Connection B: With the above plot we cannot identify the exact value of Threshold for connection B between the first and the sixth transmission round. It could have any value larger than 4. From the sixth to the fourteenth transmission round, it is 2 and at the fourteenth transmission round it is 4.

c) (5%) At the 12th transmission round for connection A, is segment loss detected by a triple duplicate ACK or by timeout? Justify your answer.

Ans: It is detected by timeout, because CongWin has dropped to 1 at the 13th transmission round.

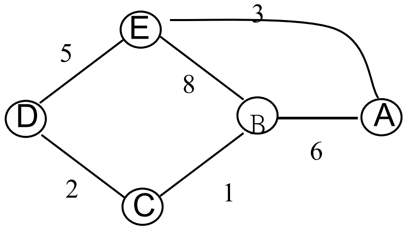
d) (5%) Assume that the segment size is 1460 bytes and that a total of 87600 bytes have been successfully transmitted over connection A before the 13th transmission round. At which transmission round the cumulative amount of the successful transmitted data is equal to 163520 bytes? Again we assume that there is neither timeout nor duplicate ACK after the 13th transmission round.

Ans: 87600 is equal to 87600/1460 = 60 segments. We would like to know at which transmission round the 163520/1460 = 112 segment will be transmitted. Thus we have to find x such that: 1 + 2 + 4 + 5 + 6 + 7 + 8 + … + x = 112 - 60 = 52, x(x + 1)/2 - 3 = 52, x = 10. This means that in the 21nd transmission round 163520 bytes will be transmitted.



Question 2: Routing Algorithms(30%)

Consider the network in the figure below. The numbers on links between the nodes represent the costs corresponding to these links. Assume that nodes initially know only the costs to their neighbors.



a) (10%) Using the distance-vector algorithm, show the distance tables at node E. Assume that the algorithm works in a synchronous manner, where all nodes simultaneously receive distance vectors from their neighbors, compute their new distance vectors, and inform their neighbors if their distance vectors have changed.

cost to

		A	B	C	D	E
	A	∞	∞	∞	∞	∞
from	B	∞	∞	∞	∞	∞
	D	∞	∞	∞	∞	∞
	E	3	9	∞	5	0

cost to

		A	B	C	D	E
	A	0	7	∞	∞	3
from	B	7	0	1	∞	9
	D	∞	∞	2	0	5
	E	3	9	7	5	0

cost to

		A	B	C	D	E
	A	0	7	8	8	3
from	B	7	0	1	3	9
	D	8	3	2	0	5
	E	3	8	7	5	0

cost to

		A	B	C	D	E
	A	0	7	8	8	3
from	B	7	0	1	3	8
	D	8	3	2	0	5
	E	3	8	7	5	0

b) (5%) Create a routing loop between the nodes B and C by changing the cost of the link between the nodes C and D. What is the minimum change in link cost that creates the routing loop? What is this problem alternatively called?

Increase the link cost to at least 4. Count-to-infinity problem.

c) (5%) How does RIP solve this problem? If RIP were used for routing in the above network,what is the finite number that would play the role of ∞?

Using poisoned reverse. 16.

d) (5%) If OSPF were used in the above network, how would it handle the routing loop? How do nodes learn the link costs in OSPF?

OSPF uses link-state routing, a global routing algorithm. Hence, the problem does not arise. Linkstate broadcast.

e) (5%) How does BGP solve this problem?

The AS-PATH attribute.

f) (5%) Assume the IP addresses of the 5 nodes A, B, C, D, and E are 130.132.5.32, 130.132.5.33, . . . , 130.132.5.36. Assume that the network in Fig. 2 is an autonomous system in the Internet with AS number 0. Node A is the BGP gateway of the AS. If A announces 130.132.5.0/28 as the prefix of the network, is it valid? If no, please propose a valid one. Please note that this AS should be assigned as few IP addresses as possible.

No. 130.132.5.32/29

Question 3: UniCast & MultiCast (15%)

Consider destinations connected to a single source by a binary tree of routers as shown below (the source is the node at the top). Each time a packet (or copy of a packet) is sent over a single link, it incurs a unit of cost. In a single time step, a node can receive all transmitted broadcast packets from its neighbors, duplicate the packets, and send them to all of its neighbors (except to the node that sent a given packet). At the next time step, neighboring nodes can receive, duplicate, and forward these packets, and so on.

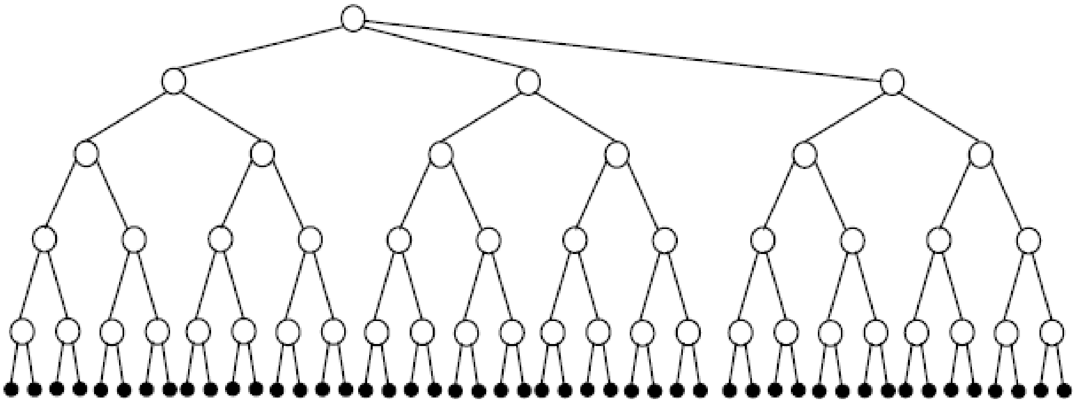


Figure 1: A computer network.

a) Assume that uncontrolled flooding is used to provide broadcast in this network. At time step k, how many copies of the broadcast packet will be transmitted, assuming that during time step 1, a single broadcast packet is transmitted by the source node to its three neighbors?

After 1 step, 3 copies are transmitted, after 2 steps, 6 copies are transmitted, after 3 steps, 12 copies are transmitted, and so on. $3 \times 2^{k-1}$ copies will be transmitted at step k.

b) Assuming there are only 48 destinations (as shown in the figure), what is the cost of sending a broadcast packet using N-way-unicast?

With N-way-unicast, the source unicasts a copy to each of the 48 destinations over a path with 5 hops. The cost is therefore 5*48 = 240.

c) Assuming there are 48 destinations, what is the cost of sending a broadcast packet using spanning-tree broadcast?

With spanning-tree broadcast, a copy of the message is forwarded over each link exactly once.

The cost is therefore 3+6+12+24+48 = 93.

Question 4: Reliable Broadcast Channel(15%)

Consider a scenario in which a host, A, wants to simultaneously send messages to hosts B and C. A is connected to B and C via a broadcast channel --- a packet sent by A is carried by the channel to both B and C Suppose that broadcast channel connecting A, B, and C can independently lose and corrupt messages (and so, for example , a message sent from A might be correctly received by B, but not by C.) Design a stop-and-wait-like error-control protocol for reliably transferring a packet from A to B and C, such that A will not get new data from the upper layer until it knows that both B and C have correctly received the current packet. Give FSM descriptions of A and B. (Hint: The FSM for C should be essentially the same as for B.)

This problem is a variation on the simple stop and wait protocol (rdt3.0). Because the channel may lose messages and because the sender may resend a message that one of the receivers has already received (either because of a premature timeout or because the other receiver has yet to receive the data correctly), sequence numbers are needed. As in rdt3.0, a 0-bit sequence number will suffice here.

The sender and receiver FSM are shown in Figure 3. In this problem, the sender state indicates whether the sender has received an ACK from B (only), from C (only) or from neither C nor B. The receiver state indicates which sequence number the receiver is waiting for.

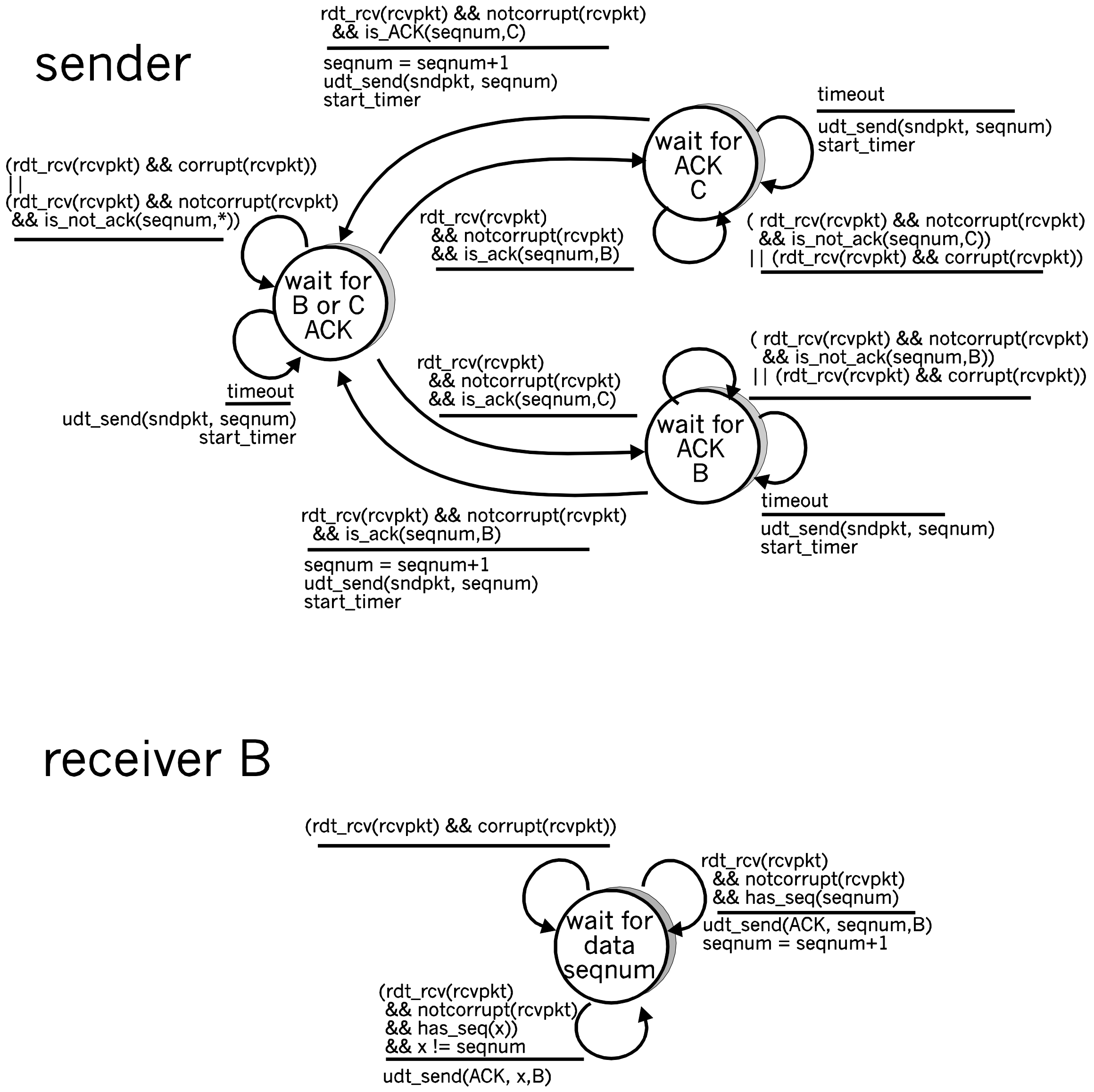

sender
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && is_ACK(seqnum,C)
seqnum = seqnum+1
udt_send(sndpkt, seqnum)
start_timer
timeout
udt_send(sndpkt, seqnum)
start_timer
(rdt_rcv(rcvpkt) && corrupt(rcvpkt))
| |
(rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)
&& is_not_ack(seqnum,*))
wait for ACK C
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && is_ack(seqnum,B)
(rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && is_not_ack(seqnum,C))
| | (rdt_rcv(rcvpkt) && corrupt(rcvpkt))
wait for B or C ACK
timeout
udt_send(sndpkt, seqnum)
start_timer
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && is_ack(seqnum,C)
(rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && is_not_ack(seqnum,B))
| | (rdt_rcv(rcvpkt) && corrupt(rcvpkt))
wait for ACK B
timeout
udt_send(sndpkt, seqnum)
start_timer
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && is_ack(seqnum,B)
seqnum = seqnum+1
udt_send(sndpkt, seqnum)
start_timer
receiver B
(rdt_rcv(rcvpkt) && corrupt(rcvpkt))
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && has_seq(seqnum)
udt_send(ACK, seqnum,B)
seqnum = seqnum+1
wait for data seqnum
(rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && has_seq(x)) && x != seqnum
udt_send(ACK, x,B)

Figure 3. Sender and receiver FSMs.

Add two more problems from HW