EECS 461 Midterm Exam
November 1998

Your Name:

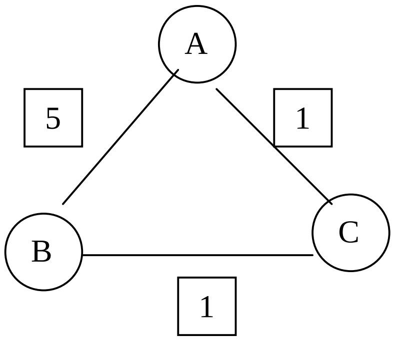
This is a **closed book** examination; you are permitted one 8"x11" page of notes. You have 50 minutes to answer as many questions as possible. The number in parentheses at the beginning of each question indicates the number of points given to the question; there are 40 points in all. We expect the exam to take roughly one point per minute.

Write all of your answers directly on this paper. Make your answers as concise as possible (you needn't cover every available nano-acre with writing). If there is something in a question that you believe is open to interpretation, then please go ahead and interpret **but** state your assumptions in your answer.

Problem 1: Routing (20 points)

A "routing loop" occurs if the next hop pointers in the network routing tables form a cycle for some destination. For example, for a packet destined for Z, the table at A says the next hop to get to Z is B, B's table says the next hop is C, and C's table says the next hop is back to A. In class, we illustrated how this might occur with distance vector routing.

(a) With link state routing, is it possible for a routing loop to occur *temporarily*? If so, give an example. If not, explain why not.



Yes, if the links form a loop. In the above example, if link A-C fails, then the table at C will point routes back to B (because it has seen the failure). However, B's table which has not got the update yet, will still point packets to A through C.

(b) With link state routing, is it possible for a routing loop to occur *permanently*? If so, give an example. If not, explain why not.

No, provided the system has a chance to settle down after each link state changes (a link is added or removed). If all nodes see the same topology and run the same algorithm to compute the best path, they will each compute a consistent set of shortest paths.

If a link fails intermittently, however, it is possible to get a loop indefinitely as two nodes won't ever have a consistent view.

Suppose we want to design a system that *never* has a routing loop. One approach is that whenever a link failure is detected, we queue packets wherever they are until all tables have been updated with the new information, and then we resume routing.

(c) Is it possible for a packet to *temporarily* travel in a loop, even if there is never a routing loop? If so, give an example, and give a way to prevent this from happening. If not, explain why not. (Hint: consider what happens to a packet queued for a failed link.)

Yes. In the example in part (a), a packet from B->A will go to C. If the link C->A then fails, it will have to backtrack back to B in order to get delivered.

The solution is to drop all packets in flight whenever there is a routing change.

(d) Is it possible for a packet to *permanently* travel in a loop (in other words, never reaches the destination), even if there is never a routing loop? If so, give an example, and give a way to prevent this from happening. If not, explain why not. (Hint: consider what happens to a packet queued for a link that works intermittently.)

Yes, if a link can fail and come up repeatedly. In this case, a packet gets sent to C; link C->A fails; it gets sent back to B; the link comes back; it gets sent to E again.

One solution is to prevent intermittent links – keep a link from adding itself back into the network if it has failed in the past.

Problem 2: Arbitration and Transport on a Satellite Link (20 points)

Like Ethernets, satellite links are a broadcast medium, requiring some contention resolution protocol. Unlike Ethernets, satellite links are high latency; one-way propagation delay can be as large as 250 milliseconds to a geosynchronous satellite. Further, they often only have a single sender and receiver (the satellite and the ground station).

All this can have a big impact on the design of both link arbitration and transport. For example, in our discussion of sliding window protocols in class, we assumed that the underlying link was full-duplex, capable of transmitting data and acks at the same time. In this problem, we examine what happens when the underlying link can only transmit in one direction at a time, along with what happens when it is high latency and only has a single sender and receiver.

(a) Suppose we use Ethernet-style carrier-sense, collision-detect to arbitrate access to a high latency link. Explain why this would be inefficient. Assuming no collisions occur, write a formula giving the link efficiency (what fraction of the link bandwidth is used transmitting useful data) in terms of the propagation delay (P), frame size (S), and link bandwidth (B). For a one-way propagation delay of 250 milliseconds and link bandwidth of 1 Gb/s, how big does the frame have to be for the link to reach 50% efficiency?

Since the latency is high, the probability of collision increases. Also, the sender has to wait a long time to be sure that there is no collision.

Minimum packet size on Ethernet = P * B

Assuming packet is larger than that;
Slot time = max (P + S/B, 2P) = P + S/B

Bandwidth achieved = S/(Slot Time) = S/(P+S/B)

Efficiency = $\dfrac{S}{(P+S/B)B} = \dfrac{S}{PB+S}$

S=0.5*(PB+S) => S=PB = 250 Mb

(b) It gets worse. Suppose a single sender and receiver connected to the same Ethernet are communicating using a sliding window protocol. Explain why we might see a collision on nearly *every* packet. (Recall that a collision occurs when both sender and receiver try to transmit at the same time.)

Each packet sent causes receiver to send back an ACK. With sliding window, sender will send another packet after the first, which will collide with the returning ACK. Same thing occurs for the next packet.

If there are only two hosts on the shared link, one simple approach is to use time division multiplexing (TDM) -- each side takes a turn transmitting.

(c) Since only one host can transmit at a time, suppose we use a stop and wait (alternating bit) protocol over a TDM link, with a maximum data frame size S and ack frame size A. Give a formula for the link efficiency of this scheme (ignore link errors).

Time to send data packet = T_{data} = P+S/B
Time to send ACK = T_{ack} = P +A/B
Amount of useful data sent = S
Bandwidth achieved = S/(T_{data} + T_{ack})

Efficiency = $\dfrac{S}{(P + S/B + P + A/B)B}$

(d) Suppose we modify the TDM scheme to allow one side to keep transmitting frames until its queue is empty. Why would a sliding window protocol be better than stop and wait in this case? With a maximum sliding window size of X, give a formula for link utilization in this new scheme. Detail any modifications you might want to make to TCP to work well in this environment.

With sliding window, sender will be able to send all its window, then receiver can ACK it all. This would require receiver to send a single ACK for the whole window rather than one ACK per packet.

Efficiency = $\dfrac{X.S}{(P + SX/B + P + A/B)B}$

TCP acks every packet that it receives. It will have to be modified to send a single ACK for a window full of packets.