EECS 122: Introduction to Communication Networks

Midterm 2 Solutions

Problem 1. A source process and sink process are running the stop-and-wait protocol on two hosts connected by a 16 Mbps 802.5 token ring (and neither host is using the ring for anything else). The maximum token-holding time is 2 ms, the ring latency is ≤ 0.3 ms, and the number of hosts on the ring is ≤ 250. Data frames are 800 bytes, ack frames are 200 bytes. Assume the token size is negligible (zero), and so is processing time on the hosts. Assume the ring can lose frames, but never the token. When the source sends a packet, it sets a retransmission timer and passes the packet to a lower layer, which then takes care of sending the packet onto the ring.

a) (2 points) The protocol is correct for what values of the retransmission timeout?

Stop-and-wait is correct only if the timeout is greater than any possible round-trip time (RTT), so we must find the maximum possible RTT.

When the source sends a message, by handing it down to a lower layer, that lower layer must then wait to obtain the token. The longest it might have to wait is when the token has just gone by, and must go all the way around again. That takes one ring latency (L), plus up to one token-holding-time (THT) for each station that might capture the token along the way (up to 248 stations, because the source doesn't wait for itself, and the sink has no reason to capture the token).

Once the source has the token, it sends the data frame (requiring a transmission time of $\frac{800 \text{ bytes}}{16 \text{ Mbps}}$) immediately followed by the token.

We must then wait for the token to get to the sink so it can send the ack frame. (By the time the token arrives, the data must have already arrived, because it was sent before the token). This requires the propagation delay from the source to the sink, plus up to one THT for each station that might capture the token along the way (again, up to 248 stations).

Then the sink sends the ack frame, requiring a transmission time of $\frac{200 \text{ bytes}}{16 \text{ Mbps}}$, and we must wait for it to reach the source, requiring the propagation delay from the sink to the source. The two propagation delays, from source to sink and sink to source, add up to L.

In total, we have waited up to $496 \text{ THT} + 2L + \frac{1000 \text{ bytes}}{16 \text{ Mbps}} = 992 \text{ ms} + 0.6 \text{ ms} + 0.5 \text{ ms} = 993.1 \text{ ms}$.

So the protocol is correct for all timeout values > 993.1 ms.

This problem was graded as follows: A half point was given for indicating that the protocol is correct for all timeout values greater than the round-trip time (we also accepted $\geq$). A half point was given for recognizing that there are up to two token rotations (media access times) per round-trip time (one before the data is sent, and one after the data is sent and before the ack is sent). One point was given for taking into account all three components of the round-trip time: transmission time (for one data frame and one ack frame, not 250 of them—token rings are not store-and-forward), propagation delay (one or two ring latencies, not 250 of them), and token holding times (248 or 249 or 250 of them per token rotation, not one or two). If only two of the three components were included, that was worth a half point. Expressions appearing on the page did not count if they were not actually used to derive the final answer. Arithmetic errors were ignored.

b) (2 points) If there are no active stations besides the source and sink, and the actual ring latency is 0.1 ms, and there are no transmission errors, what is the efficiency of the protocol? Remember that an efficiency of 100% means the protocol can deliver data as fast as the network (LAN in this case) will allow. The network may have its own inefficiency (compared to the rate of the physical layer), but that is not considered an inefficiency of the retransmission protocol.

The efficiency of the protocol is its throughput divided by the rate of the channel (LAN).

The throughput of the protocol is one packet per RTT. The round-trip time is the transmission time of a data frame, plus the propagation time from source to sink, plus the transmission time of an ack frame, plus the propagation time from sink to source, for a total of $L + \frac{1000 \text{ bytes}}{16 \text{ Mbps}} = 0.1 \text{ ms} + 0.5 \text{ ms} = 0.6 \text{ ms}$.

To obtain the maximum channel rate, imagine that the source never waits for acks before sending, and the channel never loses packets. In this scenario, the source sends as many packets as it can for one THT, then lets the token go around the ring before repeating. In one THT the source can send $2 \text{ ms} \cdot 16 \text{ Mbps} \cdot \frac{1 \text{ packet}}{800 \text{ bytes}} = 5$ packets, and it does this every $\text{THT} + L = 2.1$ ms.

So the efficiency of the stop-and-wait protocol is:

$$\frac{1 \text{ packet}/0.6 \text{ ms}}{5 \text{ packets}/2.1 \text{ ms}} = \frac{2.1}{5 \cdot 0.6} = 70\%$$

An alternate expression for the efficiency of the stop-and-wait protocol is the overall efficiency (of stop-and-wait-on-token-ring compared to the physical layer) divided by the efficiency of the token ring:

$$\frac{\frac{800 \text{ bytes}/16 \text{ Mbps}}{\text{RTT}}}{\frac{\text{THT}}{\text{THT}+L}} = \frac{0.4 \text{ ms}/0.6 \text{ ms}}{2 \text{ ms}/2.1 \text{ ms}} = \frac{0.4 \cdot 2.1}{0.6 \cdot 2} = 70\%$$

This problem was graded as follows: For either approach (throughput divided by channel rate, or overall efficiency divided by token ring efficiency), one point was given for the numerator and one point for the denominator. For the numerator, a half point was given for expressing throughput as one packet per RTT, or expressing overall efficiency as one data frame transmission time per RTT; the other half point was for correct computation of RTT. For the denominator, a half point was for understanding that the channel rate is less than 16 Mbps (or that the token ring efficiency is less than 1) because of the overhead of the ring latency; the other half point was for correct computation of the denominator.

Problem 2. Suppose a source forms five-bit codewords from two-bit messages by appending three check bits as follows:

message:	00	01	10	11
check bits:	000	011	111	100

a) (2 points) How many bit inversions per codeword can a receiver always *detect*?

Every pair of codewords differ by at least three bits, and some differ by exactly three bits. Although three bit inversions could turn one codeword into another, two cannot, so the receiver can always detect up to two inversions per codeword.

In terms of Hamming distance, the closest pair of codewords are separated by a distance of three, so the receiver can always detect errors as long as there are less than three of them per codeword (so up to two).

A correct answer with no explanation and no work shown received 1.5 points.

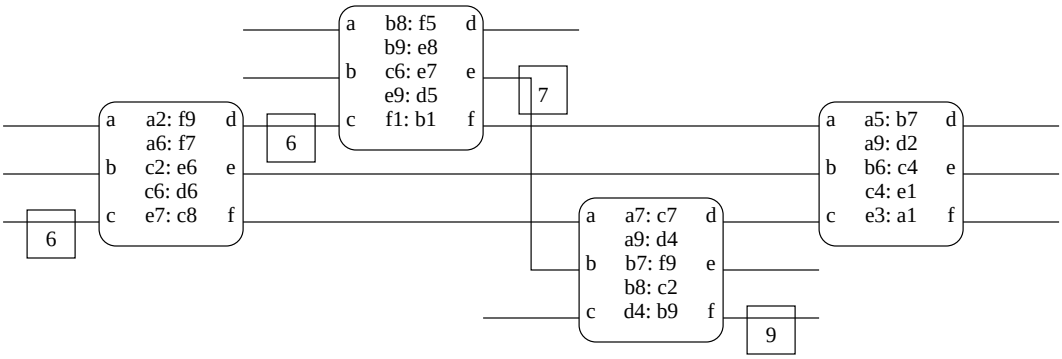
b) (2 points) How many bit inversions per codeword can a receiver always *correct*?

Every pair of codewords differ by at least three bits, and some differ by exactly three bits. Inverting two bits could move the message closer to some other codeword, causing the receiver to guess the wrong codeword, but inverting only one bit always leaves the message closer to the original codeword, so the receiver can always detect up to one inversion per codeword.

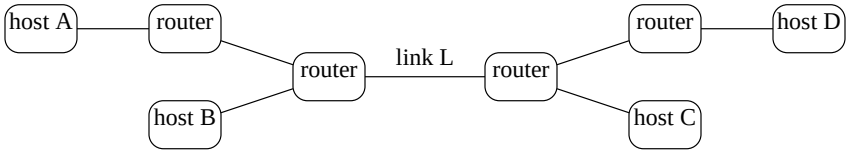
In terms of Hamming distance, the closest pair of codewords are separated by a distance of three, so the receiver can always correct errors as long as there are less than 3/2 of them per codeword (so up to one).

A correct answer with no explanation and no work shown received 1.5 points.

Problem 3. (2 points) Each ATM switch below has six inputs (a–f) and six outputs (also a–f; the links are bidirectional). The virtual circuit tables show inputs on the left and outputs on the right. A cell containing VCI 6 is entering input c of one of the switches. Draw the cell (including its VCI) on every link it traverses.



Problem 4. In the figure below, there is a transport layer connection between hosts A and D, and another between hosts B and C, with the data flowing left-to-right in both cases. Suppose all packets are the same size (both data and acks), and every link has a data rate of 10 packets/second. Assume the propagation delays, queueing delays, and processing time are all negligible (zero). Because of the inefficiency of the protocol, the links are not fully utilized; in particular, the left-to-right traffic on link L is only 6 packets/second on average.



a) (2 points) If the transport layers use a congestion control algorithm that allocates bandwidth fairly, what is the average throughput of the A-D connection? The B-C connection?

The total throughput is 6 packets/second, so allocating bandwidth fairly to the two connections gives 3 packets/second to each one.

b) (2 points) What network resource does TCP allocate (approximately) fairly?

Storage (the amount of data in the routers and on the links at any instant).

c) (2 points) If the transport layers are TCP, what is the average throughput of the A-D connection (approximately)? The B-C connection?

TCP gives approximately the same window W to each competing connection. A connection with a window of W has a throughput of W/RTT. The propagation delays and queuing delays are zero, so the round-trip time is composed of the packet transmission time at each hop. Every link has a data rate of 10 packets/second, so the packet transmission time is 0.1 seconds. For the A-D connection, a round-trip is ten hops, so the RTT is 1 second. For the B-C connection, a round-trip is six hops, so the RTT is 0.6 seconds. The total throughput is 6 packets/second, so:

$$\frac{W}{1 \text{ second}} + \frac{W}{0.6 \text{ seconds}} = \frac{6 \text{ packets}}{\text{second}}$$

$$W = 2.25 \text{ packets}$$

The throughput of the A-D connection is $W/1 \text{ second} = 2.25 \text{ packets/second}$, and the throughput of the B-C connection is $W/0.6 \text{ seconds} = 3.75 \text{ packets/second}$.