EECS 122: Introduction to Communication Networks

Homework 6 Solutions

Solution 1.

a) One station is always active and always has data to send. It sends data for time H, then releases the token, which takes time L to return, and the process repeats. Therefore, the efficiency is $\frac{H}{H+L}$.

b) From part (a), the larger H is, the higher the efficiency. As H tends to ∞ (infinity), the efficiency tends to 1. Therefore, the "best" H is ∞.

c) The worst case is when N stations are active and each station holds the token for the maximum allowed time H. From the time a station releases the token, the token spends time L propagating around the ring, and time H stopped at each station along the way, for a total of $L+(N-1)H$ until the first station receives the token and can transmit again.

Solution 2. As explained in the textbook, the average token rotation time is TTRT (more precisely, the time taken for a station to receive the token n times cannot be more than $(n+1)$TTRT, so the average token rotation time is $\frac{n+1}{n}$TTRT, which approaches TTRT as $n \to \infty$). During each token rotation, the token spends time L propagating between stations, leaving time $\text{TTRT} - L$ during which it is being held by transmitting stations. Therefore, the efficiency is $\frac{\text{TTRT}-L}{\text{TTRT}}$. The textbook shows that the worst-case access time is 2TTRT.

a) The efficiency is maximized (to 1) as TTRT $\to \infty$. However, the worst-case access time (2TTRT) also tends to infinity.

b) TTRT cannot be smaller than L. Therefore, $\text{TTRT} = L$ minimizes the worst-case access time to $2L$. However, for $\text{TTRT} = L$, the efficiency is 0.

Solution 3. The 500 byte frame will be classified as synchronous traffic in FDDI terms. Each time a station gets the token, it will be allowed to send one such frame. Since the bit rate of FDDI is 100 Mbps, the time needed to send such a frame is

$$500 \text{ bytes} \cdot \frac{8 \text{ bits}}{\text{byte}} \cdot \frac{\text{second}}{10^8 \text{ bits}} = 40 \ \mu\text{s}$$

a) Say the number of stations is N. Each station keeps the token for at least 40 μs, so the token rotation time is $200 \ \mu\text{s} + N \cdot 40 \ \mu\text{s}$. This has to be at most 1 ms $= 1000 \ \mu$s. Therefore, the maximum N is 20.

b) Each station now holds the token for 40 μs to transmit the frame, plus another 200 μs waiting for the last bit to come back. So for N stations, the token rotation time is 200 μs $+ N \cdot 240$ μs, which again must be at most 1000 μs. Therefore, the maximum N is 3. This illustrates how badly delayed token release can degrade performance.

Solution 4. Figure **??** presents a scenario where the hidden-terminal problem is not solved by the RTS/CTS mechanism. The upper part of the figure shows the connectivity graph: there are 5 stations: A, B, C, D, and E. In the lower part of the figure, the transmissions of each station are shown. The problem starts when station C cannot "hear" the CTS from B to A: this is because C is within range of both B and D, and is "drowned" by the data transmission of D and the CTS transmission of B, so that it doesn't receive anything at all. Also, C cannot hear the data sent from A to B, since A and C are not in range. Therefore, when C "wakes up" (C heard the RTS from D to E, so decided to wait for the whole transmission from D to E) and wants to send a packet, say to D (it could be any other station), it "drowns" B when it sends an RTS to D. So, A's data to B gets corrupted, although their RTS/CTS handshake got through without problems.

Because of situations like the above, RTS/CTS is usually accompanied by an ACK from the receiver to the sender, after data has been transmitted. In the above case, B will not send an ACK back to A, so A will know that it's data didn't get through and try again.

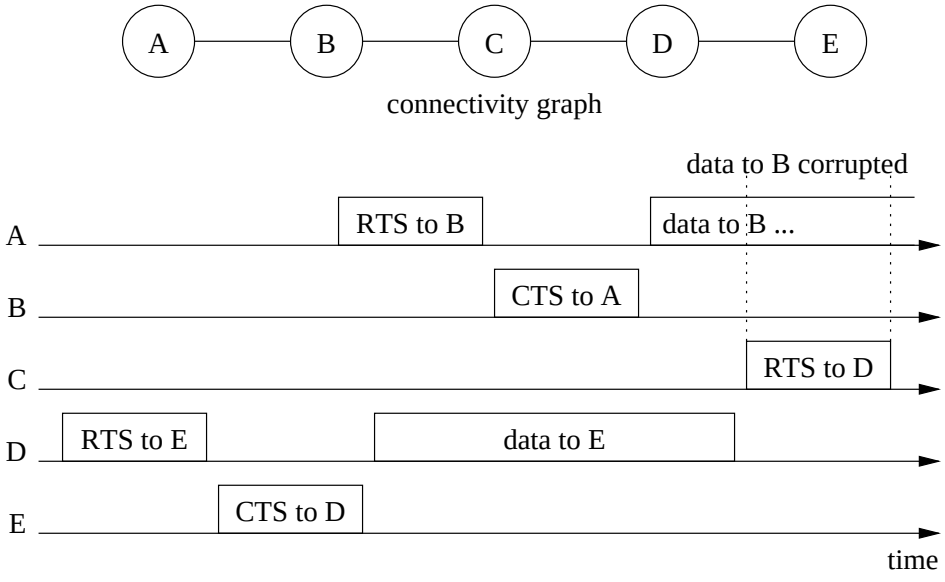

Figure 1: Illustration for solution 4.

Solution 5. This is a difficult problem. A group from ERL (Electronics Research Lab) and PATH, call the *Web Over Wireless* (WOW) group, is developing such a protocol.

The basic idea is that, for the full-connectivity case, an already existing standard for wire-line networks, the IEEE 802.4 Token-bus standard, can be used. This standard builds a logical token ring over a multiple-access broadcasting medium such as a twisted pair cable that every node is connected to. Assuming a set of radio stations all within range of each other does not change the nature of the medium, so the same protocol can be used. This protocol includes mechanisms for replacing lost tokens, detecting and removing duplicate tokens, adapting to node failures, and so on.

Relaxing the full-connectivity assumption turns out to be tricky. The Token-bus standard does not work anymore. Still, there are ways to work around the problem and come up with a token-ring protocol that is both robust and can also adapt to changes in the environment. The description of such a protocol is tedious, so it is not given here. You can download technical reports from the WOW web page: http://path5.eecs.berkeley.edu/woverw/

Solution 6. When a 1024-byte IP packet is sent using AAL5, the CS sublayer first augments it with 4 bytes of header and 8 bytes of trailer, then pads it to a multiple of 48 bytes, bringing the number of bytes to $1056 = 22 \cdot 48$. Then the SAR sublayer splits it into 48-byte cell payloads, and the ATM layer adds a 5-byte header to each one to form the ATM cells. So each packet ultimately appears on the links as $(22 \text{ cells})(53 \text{ bytes/cell})(8 \text{ bits/byte}) = 9328$ bits. A packet arrives successfully if every bit successfully traverses all twenty links. There are $9328 \cdot 20 = 186,560$ opportunities for a bit error, each with probability 10^{-10}. The fraction of packets that get lost is the probability that an error happens, which is one minus the probability that no errors happen: $1 - (1 - 10^{-10})^{186,560} \doteq 1.87 \times 10^{-5}$.

(Actually, if a bit of a cell payload gets inverted on one link, and inverted again on another link, the second error cancels out the first, so not every packet that experiences errors fails to arrive intact. But even for a bit that gets inverted on the first link, the probability that the very same bit will get inverted again on any of the remaining nineteen is $1 - (1 - 10^{-10})^{19} \doteq 1.9 \times 10^{-9}$, so our method was accurate to within two parts per billion, and presumably we don't need to know the cell loss rate with nine-digit precision, and presumably we didn't know the bit error rate with nine-digit precision to begin with.)