Directions: Some of the problems appear long but they are not too hard when you understand them. Each problem is meant to teach you something. We will pick 2 out of the first 3 problems.

• 1, Reliable Initialization via Alternating Bit: We know that we can reliably initialize a Data Link if the sender keeps a crash counter in non-volatile memory that is incremented every time the sender restarts. All RESET requests are tagged with the crash counter and RESET-ACKs are only accepted as valid if they contain the same number as the current crash counter. However, this requires us to keep a large integer (say 64 bits) counter to make sure the counter never wraps around. Recall, however, we had the same problem with stop-and-wait and we saw that we could get away with a single bit instead of a large sequence number as long as the physical channel is FIFO. So suppose the sender keeps a single bit called a crash bit in non-volatile memory and we assume a FIFO physical channel.

  – Suppose we increment (using mod 2 arithmetic) the crash bit everytime we restart, just as we incremented the crash counter. Assume the rest of the scheme stays just the same as the crash counter scheme. Show that this modification does not work — i.e., show a counterexample in which data sent after the last crash is lost despite the fact that the physical link is FIFO.

  – Describe a scheme using a crash bit that does work. (Hint: recall when the alternating bit protocol increments its sequence number/counter.) Informally argue why your scheme works.

Make sure you realize that when the sender is expecting a RESET-ACK(0), it cannot be fooled by a RESET-ACK(1) or by an ACK(0) or ACK(1). Data acks can not substitute for RESET-ACKs. Yet, you must somehow find a situation where the sender gets an old RESET-ACK followed by an old ACK to cause a problem. Hint: Consider crashing the sender multiple times in a row.

• 2. Ethernet, Min Packet Sizes, and Semi-Reliability, : At Interop 97, a leading trade show, two members of the 2001 CS 123 class have unveiled their new version of the Ethernet. Their product, Nethernet, is identical to standard Ethernet except that it no longer requires a minimum packet size. Recall the figure below that we used to justify the minimum packet size. The problem is that if A and B sent small frames, they might collide in the middle of the wire and yet neither A or B would detect the collision. To fix the problem, Nethernet adds the following rule: if a station like A sends a short packet of size less than 64 bytes, A must wait for at least 51.2 usec after its first bit is sent; if A detects any transmission during this period, A detects a collision, and does the usual retransmission.

  – a) If Nethernet requires no min packet size, what additional features of the normal Ethernet protocol can be removed as well?

  – b) Receivers normally discard runt packets of size less than 64 bytes in normal Ethernet. Is this rule still valid for Nethernet? Explain.
c) Nethernet also requires the normal means of detecting collisions (i.e., more than one signal at the same point is detected by an increase in voltage) as well as the new mechanism? Explain with an example why this is needed so that all stations can detect a collision.

d) Suppose we use the mechanism in c) as well as the new Nethernet mechanism to detect collisions. Show using an example that it is still possible for some station to not detect collisions.

e) Use the results of b) and d) to show that Nethernet collisions can result in duplicate packets being received by a receiver.

Do not forget that when A is sending the receiver need not be B. It could be any station C that is anywhere on the Ethernet to cause problems.

3. Bridging: It often happens that a node knows the higher layer address of another node and needs to know its Data Link address. Suppose someone builds an “introduction” service to do this as follows. An intro server has a well known multicast address, say INTRO. A node X that wishes to know the Data Link address corresponding to higher level address H, sends a LAN frame with destination address INTRO and with its own source address X, but with H in the data portion of the frame. When the server gets the frame, it looks up the Data Link address corresponding to H (say Y). It then forwards the original frame to Y by changing the destination address from INTRO to Y. When Y gets the frame, Y knows X’s address from the source address and so can send a reply directly to X. When X gets such a reply, X also knows the Data Link address of Y.

This protocol works fine on a LAN. But it can fail in an Extended LAN with transparent bridges.

1. Describe a topology and a scenario in which this protocol fails.

2. (How can you fix the introduction protocol to work in an Extended LAN?)

Some clarifications: An extended LAN is two or more Ethernets connected with a bridge. So draw 2 Ethernets connected by a bridge. Next place, stations X, Y, and INTRO wherever you like to cause a problem. For instance, you can put them all on one side of the bridge, or any pair of them on one side and the remaining station on the other side. They key fact you must exploit is that “Bridges forward based on *destination* addresses and learn based on *source addresses*”. Carefully go through the INTRO protocol to see what may cause bridging to fail, drawing the protocol headers containing source and destination addresses.