**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 4 problems.

1. **Ethernet, Min Packet Sizes, and Semi-Reliability**: At Interop 2017, a leading trade show, two members of the 2017 cs118 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 Figure 1 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.
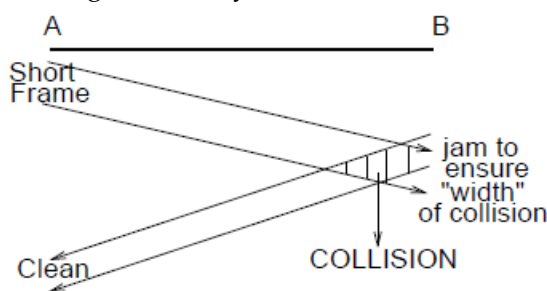


Figure 1:

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.

- **2. 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 *INT RO* 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 *INT RO* 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.

3. **IP Broadcast Storms, Bridges versus Routers:** (Adapted from Perlman's book) A broadcast storm is an event that causes a flurry of messages. One implementation that caused broadcast storms was the Berkeley UNIX endnode IP implementation. In this implementation, an endnode attempts to forward a packet that it mysteriously receives with a network layer (IP) address that is different from itself. This is what you would do if you found a neighbor's letter wrongly placed in your mailbox. However, this seemingly helpful policy can cause problems.

Consider the figure below which shows 2 LANs connected by a bridge, with several IP endnodes on each LAN. There are no IP routers. All IP endnodes are configured with the same mask and so can tell that they have the same net number. Suppose IP endnode A is incorrectly configured and incorrectly thinks its data link address is all 1's. The data link address of all 1's is the broadcast address: any packet sent to such an address is received by all stations on a LAN (it is the ultimate multicast address!).
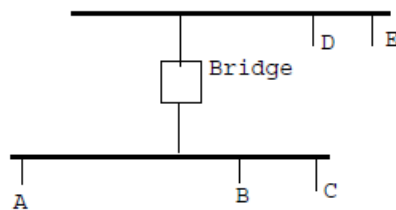


What happens when another IP endnode B decides to send a packet to IP endnode A.? Assume that B initially does not have A's data link address in its cache, and so must do the ARP protocol. Give the sequence of events.

Suppose bridge B is replaced by an IP router. (Of course, the masks at the nodes have to be changed so that there are now two masks, one for each LAN.) The problem does not disappear but it does get a little better. Explain.

4. **Modifying Routing to Avoid Fragmentation**: In order to avoid fragmentation, we may need to compute the minimum of the maximum packet sizes of all links on the best route to each destination D. For example in

the figure below, if the best route from S to D is through R1, the minimum of the maximum packet sizes is 250 bytes because the route uses the link from R1 to D.

In distance vector routing, a router R computes its own distance, Distance(D, R) to a destination D using the distances sent by its neighbors as follows: Distance(D, R) = Minimum across all neighbors N of Distance(D, N ) + Distance(R, N ). How would you modify this protocol to also compute the minimum max packet size on the shortest distance route to D?