Problem 1

1. We can also remove padding and the length field.

Without a minimum packet size, we no longer require a length field. Ethernet originally requires a length field in case we need to pad a message that is shorter than the minimum packet size. Without a length field, we will not know which bits the padding and which bits are the actual data. However, Nethernet will not pad packets because there is no minimum packet size and does not require a length field.

1. Chart, line chart

   Description automatically generatedNo, this rule is no longer valid in Nethernet. In Ethernet, we must discard packets of length less than 64 bytes because the minimum packet size is 64 bytes. However, in Nethernet, there is no minimum packet size and packets of length less than 64 bytes are valid.
2. Consider stations A and B as shown in the figure to the right. A is trying to send a small packet (< 64 bytes) to B and B is trying to send a large packet (>> 64 bytes) to A. While B is sending the large packet, the small packet that A sends also arrives. According to the new rule given in the problem, a station waits and detects possible transmissions after sending the first bit **only when the packet size is less than 64 bytes**. Therefore, in this case, B would not detect any collisions if we did not implement normal means of error detection in Nethernet.

Diagram

Description automatically generated

1. We want all nodes on the network (even those not sending or receiving) to detect when collisions happen. Consider the scenario in the figure on the right. A is sending a small packet to B and B is sending a small packet to A at the same time. Both A and B will detect collisions because of the new rule of Nethernet (A and B both detect transmissions within the 51.2 usec period). On the other hand, there is no voltage spike at node C. Also, node C is not sending any packets, so it is not waiting for 51.2 usec. Thus, node C does not detect the collision.

Diagram

Description automatically generated

1. Consider the figure to the right where A is sending to B and C is sending to A at the same time. When A first sends to B, B will receive the clean packet and store it. It will not detect any collisions, so it will not expect retransmissions. However, A will detect a collision while waiting 51.2 usec after sending its small packet (it detects a signal from C). So, A will retransmit its message to B. B will receive the second packet, which is a duplicate.

Problem 2

1. Both bridges will map MAC address V to the upper interface. The packet will go to all three LANs (LANs 1, 2, 3). When B1 receives the packet, it will map V to upper interface and flood (i.e., send the packet to the other interface). So, the packet will be on LAN 2. When B2 receives the packet, it will map V to the upper interface and flood (i.e., send the packet to the lower interface). Then, the packet will be on LAN 3 and the victim V will receive the packet.
2. When S replies to V, the packet will not go to the real victim. B1 will receive S and will look at the destination MAC address V. It will look at its table and see that V is mapped to the upper interface. Thus, B1 will not transmit the message to the lower interface. To pick up the packet, H must be listening for packets whose destinations are MAC address V.
3. Below is the sequence of sends:
   1. **H sends a packet to S with forged source address V.** Both bridges will flood and send the packet to their lower interfaces. Both bridges will also map V to their upper interfaces.
      1. B1 bridging table: V 🡪 Up
      2. B2 bridging table: V 🡪 Up
   2. **The real victim V sends a packet to S**. Both bridges will flood and send the packet to their upper interfaces. Both bridges will relearn the entry for V and map it to their lower interfaces.
      1. B1 bridging table: V 🡪 Down
      2. B2 bridging table: V 🡪 Down
   3. **S sends a reply to V.** Both bridges will send the packet to their lower interfaces since a mapping for V exists. They will also learn S and map it to their upper interfaces. The reply from S will end up on all LANs.
      1. B1 bridging table: V 🡪 Down, S 🡪 Up
      2. B2 bridging table: V 🡪 Down, S 🡪 Up
   4. **Both H and V receive the reply since it is on all LANs.**
   5. **V sends a RESET to S.** Both bridges are mapping S to their upper interface, so the RESET message will end up on all LANs. Bridging tables are not changed.
   6. **S receives the RESET.**
4. To detect and report such attacks, a bridge could keep track of and look at how much time has elapsed between learning V was on LAN X and relearning that V is now on LAN Y. If it is impossible for a human to disconnect V from LAN X and connect it to LAN Y in that time period, then there is a high possibility of an attack.

Problem 3

1. When a bridge receives a message with source address SRC and destination address DST, it should first check if SRC is already learned in the bridging table. If so, it should check if the current interface the message is coming from differs from the one stored for SRC in the table. If so, then also keep track of and check the time since that interface was learned. If the time period is short enough, then instead of relearning the source address, drop the packet as the packet is circulating.

We keep a timer in case the endnode A is moved to a new LAN. If the time period is so short that it is impossible for a node to be moved that fast by a human, then we can suspect circulation.

In the given example, both B1 and B2 will first map A to the upper interface and then flood. Then, the counterclockwise message ends up on B2’s bottom LAN. Then, B2 will see that it is receiving 2 messages from the same source (A) from 2 different interfaces within a short span, so the packets must be circulating. Hence, B2 will drop this packet. Similarly, B1 will receive the clockwise message on its bottom LAN. B1 will see that it is receiving 2 messages from the same source (A) from 2 different interfaces within a short span and will also drop the packet. At this point, circulation stops.

1. Alyssa’s method assumes that both packets circulate at least one cycle so that at least one bridge maps the source to interface X and gets a packet with the same source from another interface Y in a short period of time. If one of the packets gets dropped soon enough (before reaching a bridge), then we will **NOT** get 2 messages with the same source from different interfaces in a short span of time. Thus, the other packet will indefinitely circulate around the network and will never be detected by any bridge.

Problem 4

1. Below is the sequence of events that occur:
   1. D broadcasts an ARP request on the LAN requesting A’s MAC address (given A’s IP address)
   2. E receives the ARP broadcast, sees that the IP address is different from its own, and forwards the packet due to the Berkeley implementation. So, there are now 2 packets on the upper LAN.
   3. The bridge receives both ARP broadcasts (the original and the forwarded message) and forwards them to the lower LAN.
      1. Bridging table: D 🡪 Up
   4. B and C receive the ARP request on the LAN since it is a broadcast. For each of the 2 messages, they will forward another message because the IP address on the message is different from their own. There are now 4 broadcast messages on the bottom interface.
   5. The 4 broadcast messages can be forwarded by the bridge to the upper interface, and each node there can forward the broadcast again. In the next iteration, the same process continues. This results in an exponential number of broadcasts in the network.
   6. A receives the 4 broadcast messages and will send 4 ARP replies to D that contains A’s MAC address. Since these ARP replies are destined to D, the messages will not get through the other nodes’ hardware and will be ignored (no forwarding).
   7. The bridge receives the ARP replies. It maps A’s MAC address (all 1s) to the lower interface. The destination (D) is stored in the bridging table, so it transmits the messages to the upper interface.
      1. Bridging table: D 🡪 Up, A 🡪 Down
   8. D receives the ARP replies on the LAN
   9. D puts the message packet with destination MAC address of all 1s on the data link
   10. All nodes on the data link (E, A, B, C) will read this packet since the MAC address of all 1s is the broadcast address. This will again cause the nodes to continue forwarding the message since the IP address on the message is that of A. Now, both the original ARP request (real broadcast) and the message to A (with destination of all 1s) are causing broadcast storms on the network. The number of broadcasts on the network increases exponentially with each iteration.

Overall, if there are a total of T nodes on the network that copy the broadcast for the first iteration, they each create another T copies on the second iteration, and so on. So, the number of broadcasts storming the network would be proportional to , where n is the number of iterations.

1. If the bridge is replaced by a router, then the problem gets a little better. D sends an ARP request to the router on the top link. Since the ARP request is directed toward the router, no other node will forward the ARP on the top LAN. However, the router will broadcast an ARP request on the bottom LAN with A’s IP address. This broadcast will be read by all nodes on the bottom LAN and forwarded due to the Berkeley implementation. The problem is now more localized to the nodes in the bottom LAN, which is at most M. On the first iteration, the ARP broadcast is forwarded by all M nodes, so there are M copies of the message. On the second iteration, each node on the LAN makes a copy of each of the M messages. This process continues and the number of messages grows exponentially. If there are n iterations, there will be copies on this LAN.

Assume n is the number of iterations that the packets are forwarded as per the Berkeley implementation. As seen in part a, if we use a bridge, there will be copies on the network. If we use a router, there will be copies on only one LAN.