CS 118, Fall 2022
Homework 2
Due Nov. 9, 2022 at 4pm

1 Ethernet, Min Packet Sizes, and Semi-Reliability

At Interop 2022, a leading trade show, two members of the 2022 CS 118 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 nor 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 μsec after its first bit is sent; if A detects any transmission during this period, A detects a collision, and does the usual retransmission.

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.

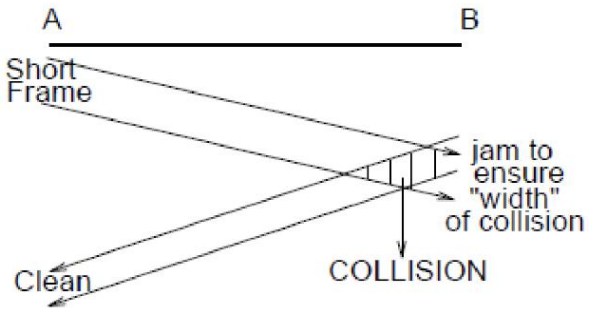

Figure 1: Short frame collision

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

Solution:

Since there is no minimum packet size, we no longer need padding, since we can now send any size of packet we want. From this, we also notice we don't need the length field in the Ethernet header, as there is no extra padding that needs to be removed by the receiver.

(b) Receivers normally discard runt packets of size less than 64 bytes in normal Ethernet. Is this rule still valid for Nethernet? Explain.

Solution:

This rule is no longer valid for Nethernet. We were able to do this for Ethernet since meaningful packets must be a minimum size of 64 bytes. With Nethernet, we no longer have this guarantee due to the lack of a mininum packet size, so we must consider these smaller packets.

(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) in addition to this new mechanism. Explain with an example why this is still needed so that all stations can detect a collision.

Solution:

Assume A sends a small packet and B sends a small packet that collides with A's. In this situation, we know that A will receive a transmission within the 51.2 μsec waiting period, which tells A that a collision occurred. However, assume there is another station C. Since C never sent anything, it doesn't interpret this transmission as a collision. As a result, we need the average voltage mechanism for these other stations to perform collision detection.

(d) Suppose we use the mechanism in `(c)` (detection via increased voltage) 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.

Solution:

Assume the same situation described in my answer to part `(c)`. Now assume that station C receives the transmission from B at a time when it is no longer colliding with A's transmission. Since C has not sent anything *and* there is no rise in voltage, C will have missed the collision.

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

Solution:

Taking the situation described in my answer to part `(d)`, we now assume that B was sending to C. Since C missed the collision between A and B's packets, it accepts B's transmission. However, B knows it collided with A since it received A's transmission within the 51.2 μsec window. As a result, B re-transmits the same packet. Once again, C will receive a packet without any collisions and accept the duplicate.

2 Hacking Bridges

Bridge learning can be affected by wrong (or hacked) behavior of stations. In `Figure 2` below, assume that a hacker with MAC address H wishes to impersonate a victim with MAC address V when talking to a server with MAC address S. **Assume all bridge tables are initially empty.**

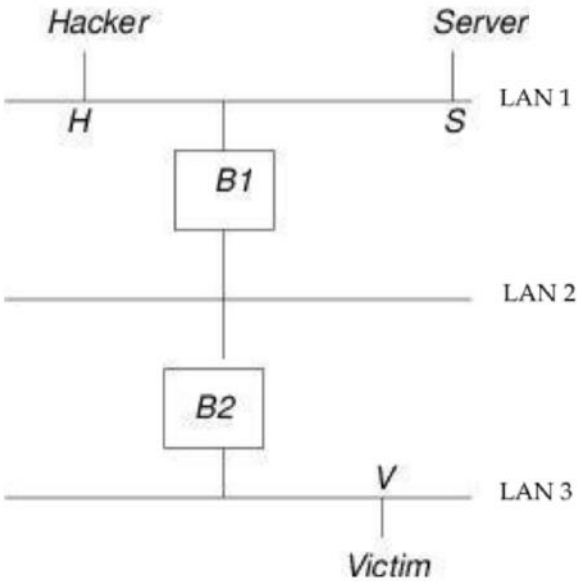

Figure 2: Bridge architecture

(a) Suppose H starts by sending an Ethernet packet to S with a forged MAC source address V. What do the bridges learn? Which LANs does this packet go to?

Solution:

$B1$ will learn that V is accessible through its upper port and $B2$ will learn that V is accessible through its upper port. Since neither bridge knows where S is, this packet will go to `LAN 1`, `LAN 2`, and `LAN 3`.

(b) When S replies to V will the packet go to the real victim? What must H do to pick up the packet?

Solution:

The packet will never go to the real victim, as $B1$ thinks V is accessible through its upper port, so it will discard the packet, causing the packet to only be transmitted on `LAN 1`. In order for H to pick up the packet, it must listen for packets with destination MAC address V.

(c) Assume that if the real victim V ever gets an unsolicited packet from S, it will send a `RESET` packet to S that will stop the cozy communication that H is having with S. Describe a scenario (i.e., the sending of some packets) that causes a packet from S to reach the real victim, spoiling the hacker's efforts. Assume that H keeps sending so the bridges never time out their entry for V. Describe the events carefully for more points.

Solution:

Assume V sends a packet to S while H and S are communicating. As this packet travels through the network, it causes $B1$ and $B2$ to change their learning so that V is now accessible through the bottom port of each bridge. As long as H doesn't send anything afterwards to rewrite the bridges' learning again, the next time S sends to V, the bridges will direct the packet downwards to `LAN 3`, where V will receive it and send the `RESET` packet to S.

(d) What could a bridge do to detect and report possible attacks such as this?

Solution:

When the bridge learns which LAN a station is on, the bridge can pass on a message to all other ports stating that it has learned that station. Since these architectures are trees (non-cyclic), if that message is ever received by the station that the bridge "learned", that means that the bridge had learned an incorrect or hacked MAC address. For instance, in the example above, when $B1$ learns that V is on the upper port, it can pass on a message that it has learned V's location to the lower port. This message will eventually reach V, which should be impossible since V was supposed to be accessible through the upper port. V can then send a message back to inform $B1$ that something went wrong and $B1$ can drop the bridge table entry.

3 Bridging and Loops

Even the Spanning Tree algorithm cannot prevent temporary loops (for example if two separate LANs are connected by a bridge, and then someone plugs together two separate LANs using a repeater). Eventually, the loop will be broken, and the right bridges will turn off, but packets can circulate at very high speeds till the loop is broken.

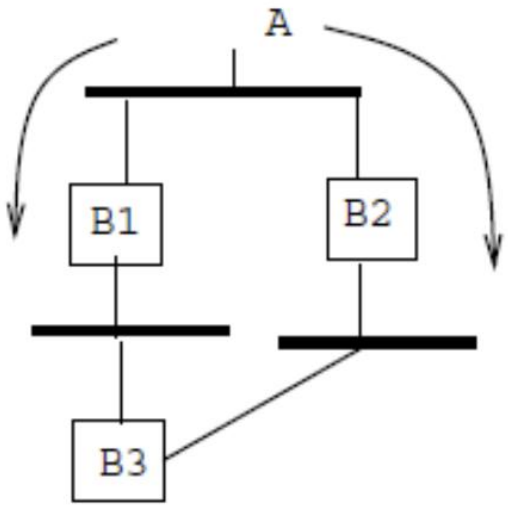

Figure 3: Temporary bridge loop

Alyssa P. Hacker has thought of a way to improve the situation during temporary loops. Consider a temporary loop of bridges shown in `Figure 3` above and assume that bridges $B1$, $B2$, and $B3$ all think they are ON. Suppose A sends a multicast packet. Both $B1$ and $B2$ pick up the packet, and so the packet will circulate in *two* directions, both clockwise and anti-clockwise. This will also happen if the destination is unknown.

(a) Based on Alyssa's observation, how could you modify the bridge learning and forwarding to prevent multicast and unknown destination packets from circulating continuously in a temporary loop?

Solution:

You could implement some sort of table that tracks if the bridge has seen a given packet and forwarded in a certain direction. If that bridge encounters the same packet going in the same direction at some point later, the bridge knows it has already forwarded that packet in that direction, allowing the bridge to safely discard it.

(b) Alyssa's method is sound if there is no packet loss. What goes wrong with her method if packets can be dropped?

Solution:

If a packet is dropped, then the bridge will need to forward the retransmission. In my solution from `(a)`, I assumed that the same packet moving in the same direction must be a looping packet. However, since a retransmission is just a copy of a packet, bridges will see it as the same packet moving in the same direction. As a result, if we use the solution from `(a)`, retransmissions will be dropped as if they were looping packets.

4 IP Broadcast Storms, Bridges versus Routers

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 `Figure 4` 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/prefix. 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!)

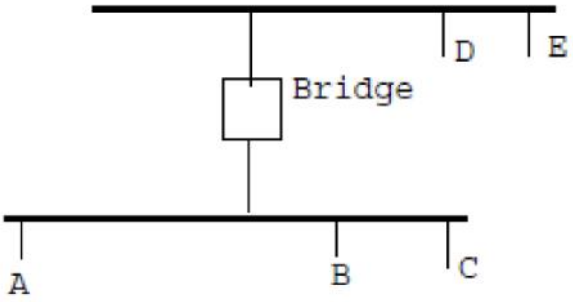

Figure 4: 2 LANs connected by a bridge

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

Solution:

I assume that when an endnode forwards a packet, the packet retains the original sender's information

- Since D does not have A's MAC address, it must broadcast an ARP request, detailing its MAC address and IP address as the sender, and A's IP address as the target
- The ARP request will cross the bridge, teaching the bridge that D is on the top LAN
- As the request reaches endnodes B, C, and E, the endnodes will see that they aren't the target and attempt to forward the request
- Each time an endnode forwards a request, an additional broadcast is sent to the network, which will also reach non-target endnodes, causing an exponential increase in the number of packets in the network

- A will receive the ARP request, format an ARP response to send to D, and send it out (this happens for each of the re-broadscasts as well)
- The ARP responses will cross the bridge, teaching the bridge that A is on the bottom LAN
- D will receives the ARP response and learn that A has a MAC address of all `1`s
- Since the bridge knows the location of A now, any duplicated ARP requests from the bottom LAN will stay on the bottom LAN
- The same duplication issue will occur with any future packets D sends out to A, since it incorrectly learned that A's MAC address was the broadcast address

(b) Suppose the bridge is replaced by an IP router. (Of course, the masks at the nodes must be changed so that there are now two masks, one for each LAN. Note a mask is just a bitmap of `1`'s in the most significant bits that tells you how long the prefix for that subnet is). The problem does not disappear, but it does get a little better. Explain as precisely as you can the improvement using two parameters: T, the total number of endnodes in the network of Ethernets, and M, the maximum number of endnodes in a single Ethernet.

Solution:

When using a bridge as described in `(a)`, we initially have no idea where A is, since the entire point of the ARP request is to figure out what A's MAC address is. As a result, when a broadcast is sent out, there are $T-2$ endnodes (all T endnodes minus the sender and receiver) that will receive the broadcast and duplicate it, with each duplicate reaching $T-2$ endnodes as well. This means there are $T-1$ endnodes that are constantly receiving and forwarding duplicate packets.

When using an IP router, we know that the router itself has an ARP table that maps IP addresses to the correct MAC address. As a result, once the broadcast goes through the IP router, it will be redirected directly to the receiver, avoiding the duplicate packets from the endnodes on the same LAN as the receiver. Therefore, only endnodes on the top LAN will end up infinitely receiving and forwarding. Assuming there are M endnodes on the top LAN, there would only be at most M endnodes caught in the loop of receiving and forwarding, unlike the $T-1$ if we used a bridge.