Lecture 4: Spanning Tree Routing

CS 356: Introduction to Computer Networks

Instructor: Venkat Arun

P&D Chapter 3.2

Borrows some images from “Computer Networks: A Systems Approach” by Peterson and Davie

Recap

There are three types of addresses commonly used on the internet. We build up the higher layers of addressing from the lower layers

- Ethernet addresses: Completely flat. Only guarantee* is that the addresses are unique
- IP addresses: Hierarchical address. Depends on *where* the device is on the internet
- Domain name: Human interpretable. DNS maps domain name to IP address. For performance optimization, mapping may depend on where the user is

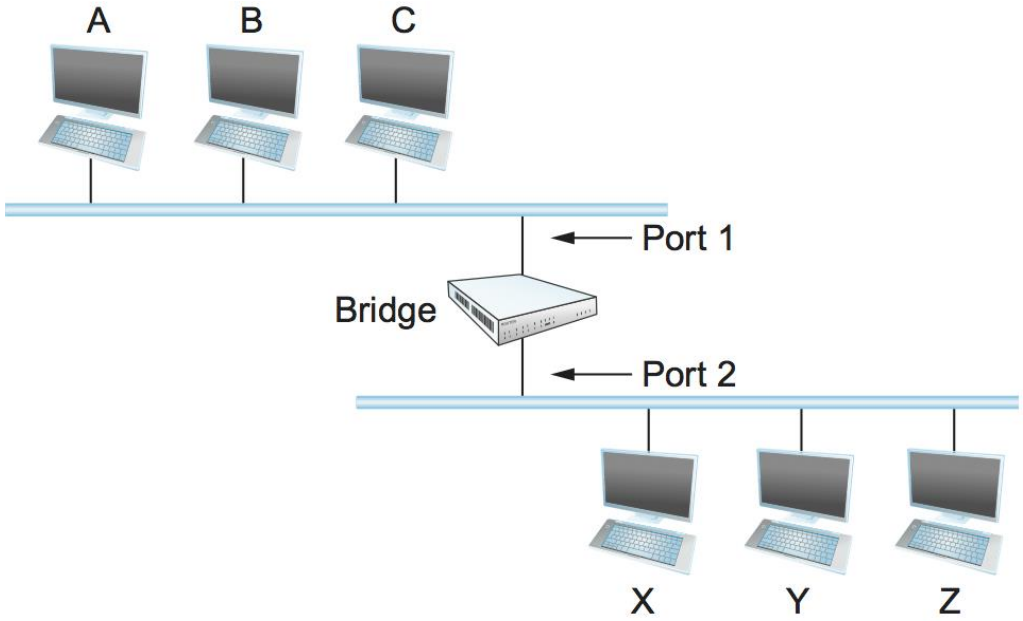
*terms and conditions apply ☺

Simple ethernet bridge

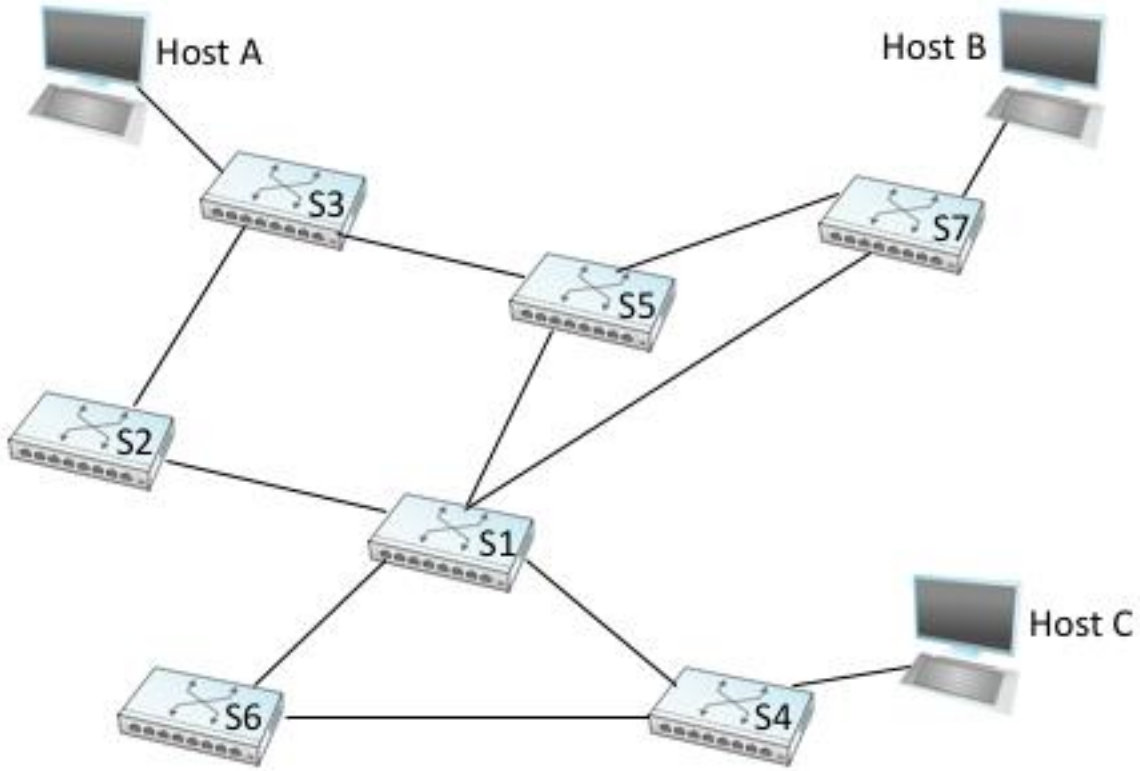
```
table = {}
def pkt_recv(pkt):
    # Update the table
    if pkt.src_addr not in table:
        # Do not confuse port of the bridge with TCP/UDP port number
        table[pkt.src_addr] = pkt.src_port

    # Forward the packet
    if pkt.dst_addr in table:
        forward(table[pkt.dst_addr], pkt)
    else:
        for port in all_ports:
            forward(port, pkt)

def forward(port, pkt):
    if port != pkt.src_port:
        # Send pkt on port
```



This technique fails when there are cycles

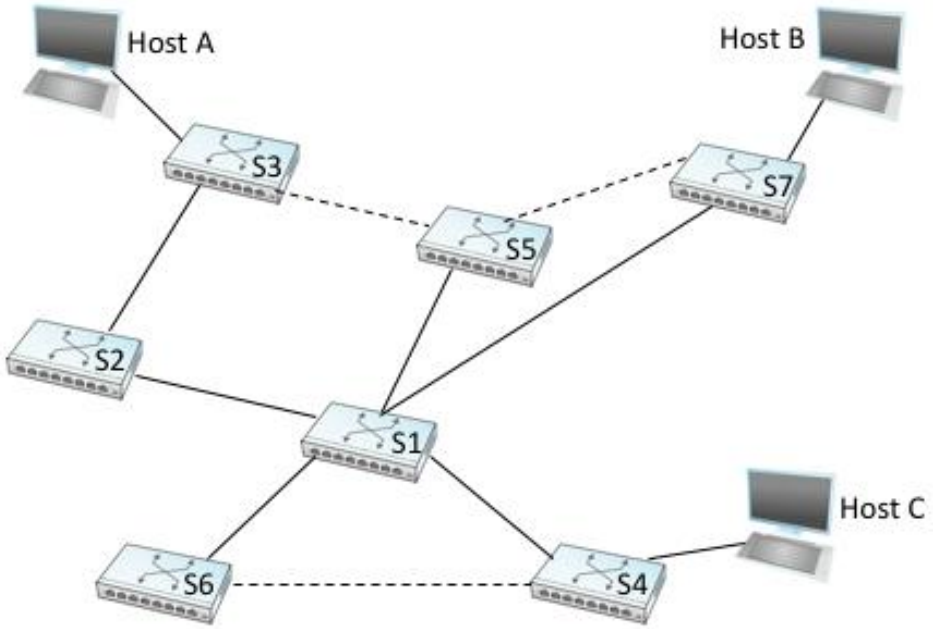


Suppose `table` is empty for all the switches. When host A sends a packet to B, the packets will circulate forever inside the network on *all* the switches because they forward it indefinitely

Quick and dirty fixes

- Remember all packets that a switch sent. Do not forward the same packet twice
  - If an end host sends identical packets, say to retransmit information, the network will refuse to forward the second one
  - Switches need a *lot* of *fast* memory
  - Q: How does this compare to updating the routing table?
- Create a “time to live (TTL)” field in the packet. Switches decrement it each time. If it reaches 0, they do not forward
  - Much lower overhead
  - Avoids infinite circulation, but packets can still circulate
  - IP headers have a TTL field, but it is used as a safety mechanism, not the first line of defense
  - Note: we are talking of ethernet here. Ethernet bridges ignore IP headers

Proper fix: switches should enough links that cycles no longer exist



Goal: All nodes should be connected, but there should be no cycles => tree

Q: Should the network admin just avoid adding those extra links in the first place then?

Q: How would you do this?

Three approaches

- Have a centralized server run a routing algorithm (here, spanning tree) and tell everyone which links to turn on
  - Essential idea behind Software Defined Networking (SDN)
  - Problems?
  - Single point of failure. Plus, what happens if we cannot reach this centralized server?
- Design a distributed algorithm where each switch operates independently and yet the system converges to a good answer
  - Self repairing and more resilient to failures
  - Algorithms harder to design, but are a one-time effort
  - Today, it is the fallback option when SDN fails. SDN is optional, self-repairing networks are not
- Cop out: get a human to fix issues
  - Sometimes needed even today

Distributed spanning tree algorithm

Strategy:

- All switches pick a common “root” switch

  Whichever has the lowest ethernet address (assumes unique addresses)

- All other switches select the shortest path to that root switch. Any links not on the shortest path of at least one switch are removed

  If two paths have the same length, pick hop with lower ethernet address

- If information is stale, timeout so that if a link fails, we pick a different tree

PS: Distributed algorithms are always hard

Logic at each switch

- If I think I am the root switch:
  - Tell all my neighbors that I have a path to the root (me) with 0 hops
  - If someone advertises a path to a root with address smaller than me, stop thinking that I am the root switch and start thinking they are the root switch
- If I think someone else (say X) is the root switch
  - Keep track of the shortest path to X that anyone has advertised
  - Tell all my neighbors about this shortest path to X
  - If someone else advertises a path to X that is shorter than mine, update my idea of a shortest path. Re-advertise

Logic at each switch: handling failures

- If I think I am the root switch:
  - Tell all my neighbors that I have a path to the root (me) with 0 hops
  - If someone advertises a path to a root with address smaller than me, stop thinking that I am the root switch and start thinking they are the root switch
  - Periodically advertise myself to my neighbors to assure them I'm "alive"
- If I think someone else (say X) is the root switch
  - Keep track of the shortest path to X that anyone has advertised
  - Tell all my neighbors about this shortest path to X
  - If someone else advertises a path to X that is shorter than mine, update my idea of a shortest path. Re-advertise
  - Upon receiving a periodic message, create corresponding update to other neighbors
  - If I haven't heard from the root in a while, start thinking that I'm root again

Distributed spanning tree algorithm

- Is this guaranteed to produce a tree?
- Will the tree be “optimal”? What does “optimal” mean?
- Does the best routing scheme always lead to a tree?