

Lecture 8:

Networks of links

CSE 123: Computer Networks

Aaron Schulman



Project 1 due today (not yesterday)



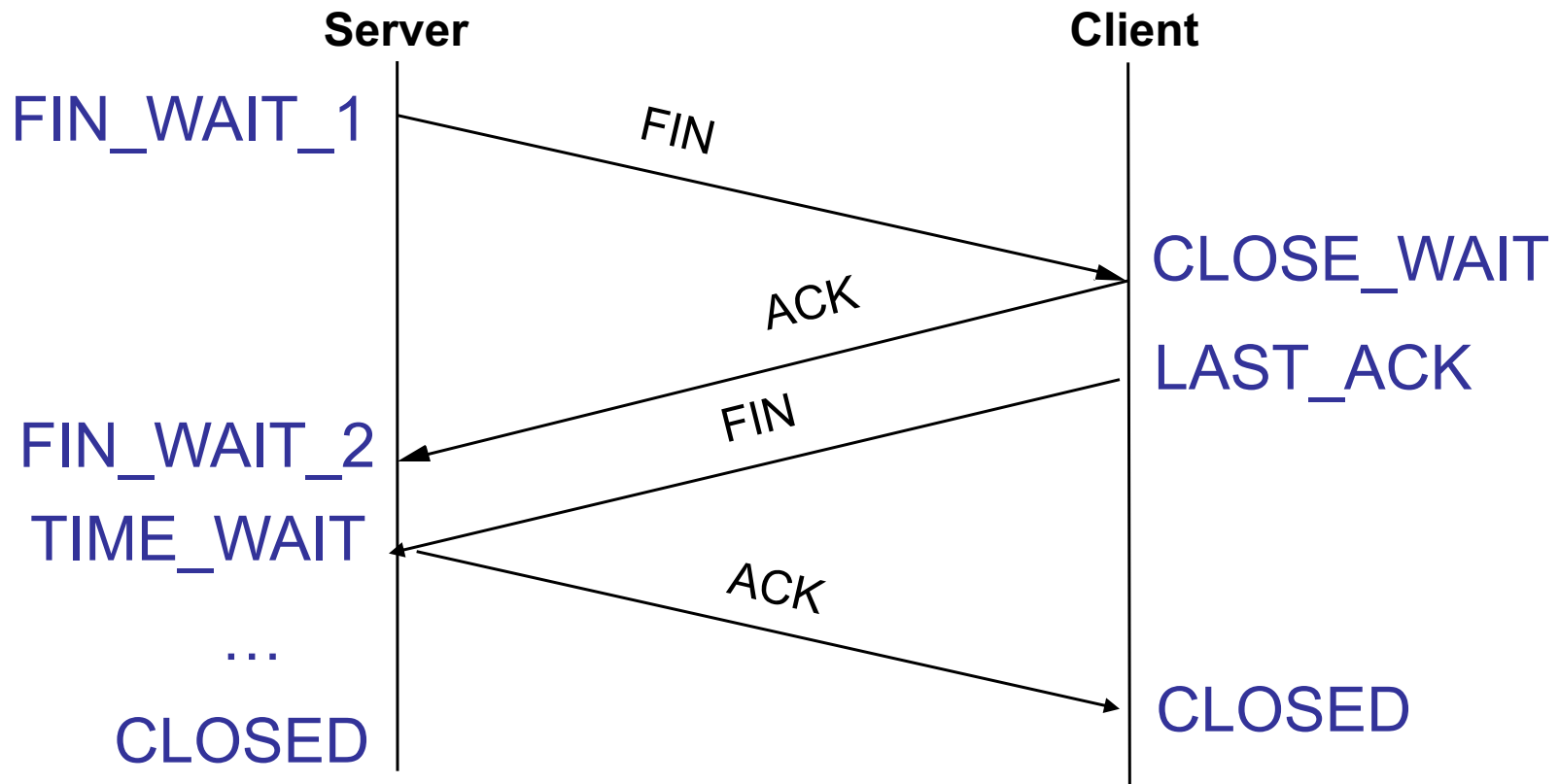
UCSD CSE



Lecture 8 Overview

- Finish closing TCP connections
- Bridging & switching (link layer)
 - ◆ Learning bridges
 - ◆ Spanning Tree

TCP Connection Teardown





The TIME_WAIT State

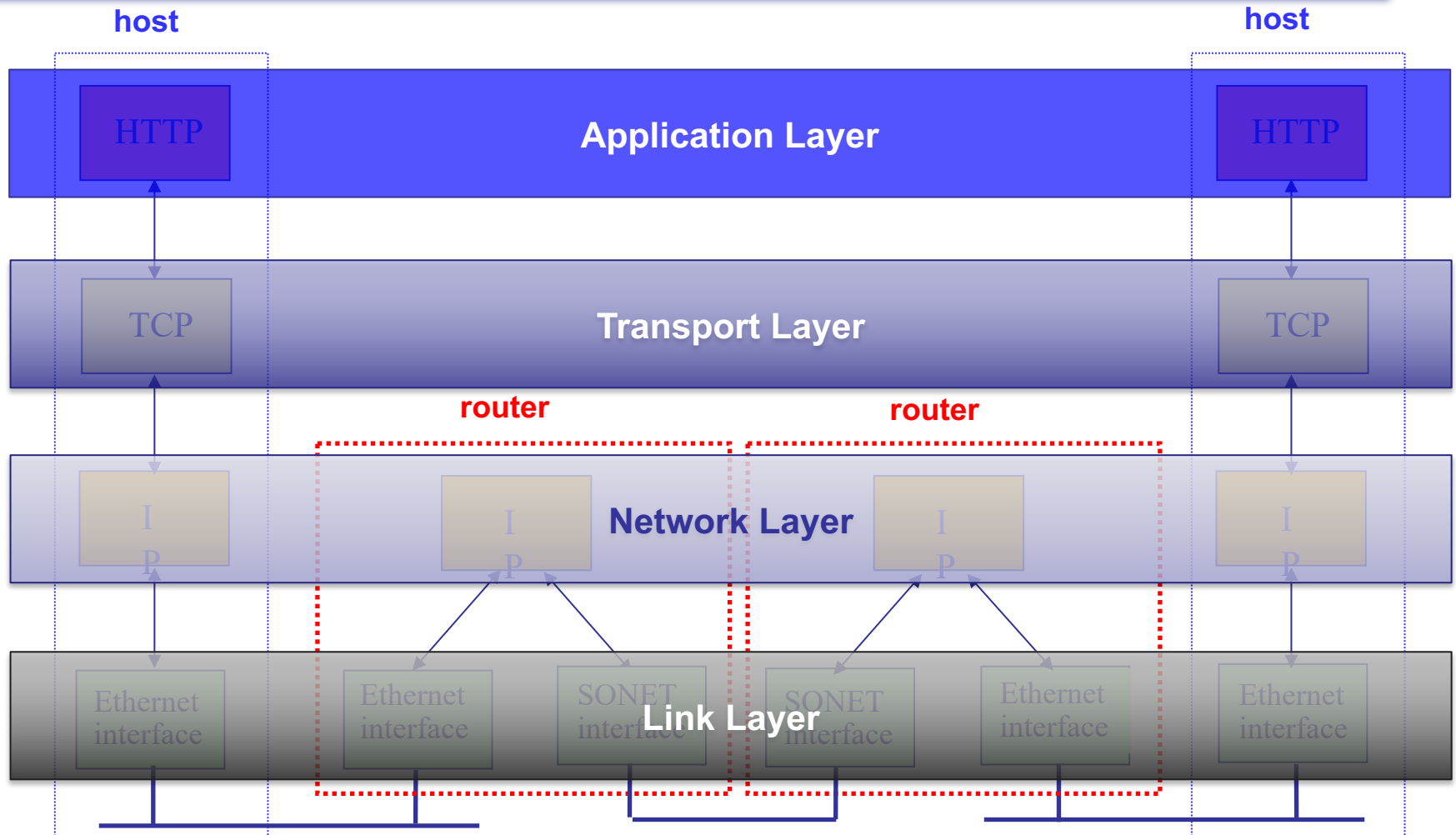
- We wait $2*MSL$ (maximum segment lifetime) before completing the close
- Maximum segment lifetime is usually 60 seconds (longest period of time packet can be stuck in network)
 - ◆ Why?
- ACK might have been lost and so FIN will be resent
 - ◆ Could interfere with a subsequent connection
- Real life: Abortive close
 - ◆ Don't wait for $2*MSL$, simply send Reset packet (RST)
 - ◆ Why?



Lots of Icky Details

- Window probes
 - Silly Window Syndrome
 - Nagle's algorithm
 - PAWS
 - Etc...
-
- Steven's books "TCP/IP Illustrated (vol 1,2)" is a great source of information on this

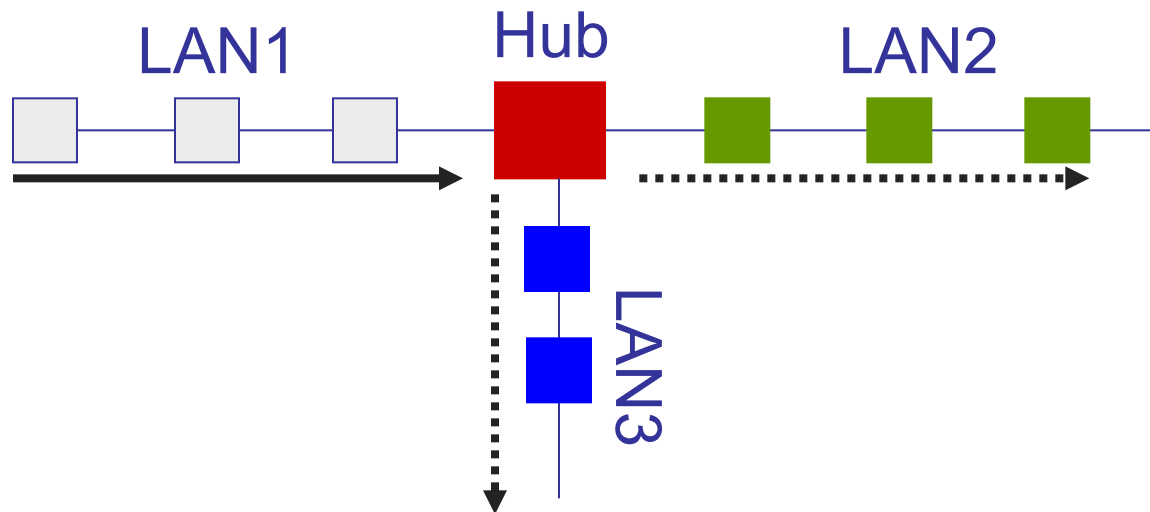
TCP/IP Protocol Stack





Hubs/Repeaters

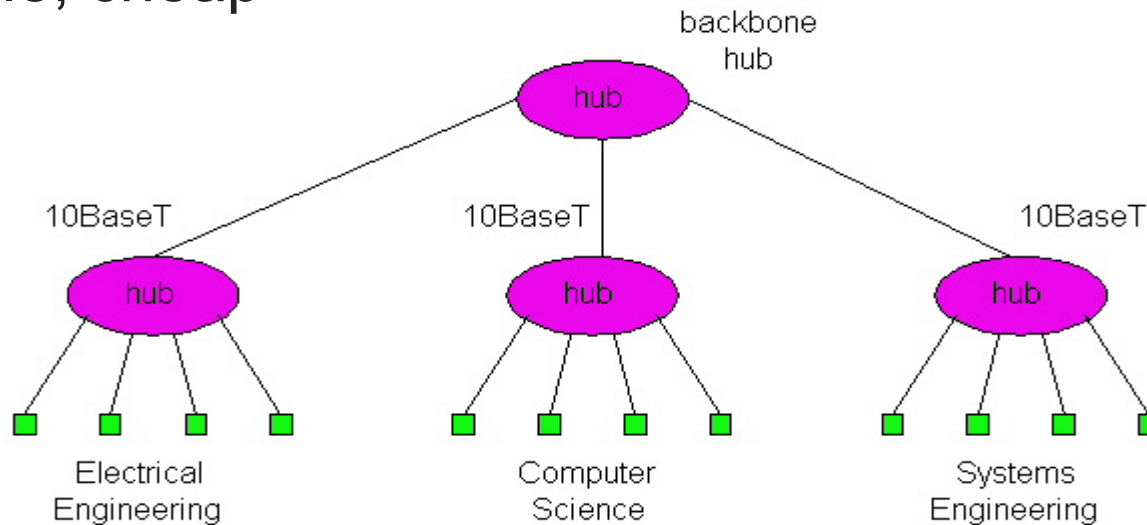
- Physical layer device
 - ◆ One “port” for each LAN
 - ◆ **Repeat** received *bits* on one port out *all* other ports





Hub Advantages

- Hubs can be arranged into hierarchies
 - ◆ Ethernet: up to four hubs between any pair of nodes
- Most of LAN continues to operate if “leaf” hub dies
- Simple, cheap





Still One Big Bus

- Single **collision domain**
 - ◆ No improvement in max throughput
 - ◆ Average throughput $<$ as # of nodes increases
 - ◆ Why?
- Still limited in distance and number of hosts
 - ◆ Collision detection requirements
 - ◆ Synchronization requirements
- Requires performance homogeneity
 - ◆ Can't connect both 1 Gbps and 100 Mbps devices

Bridges

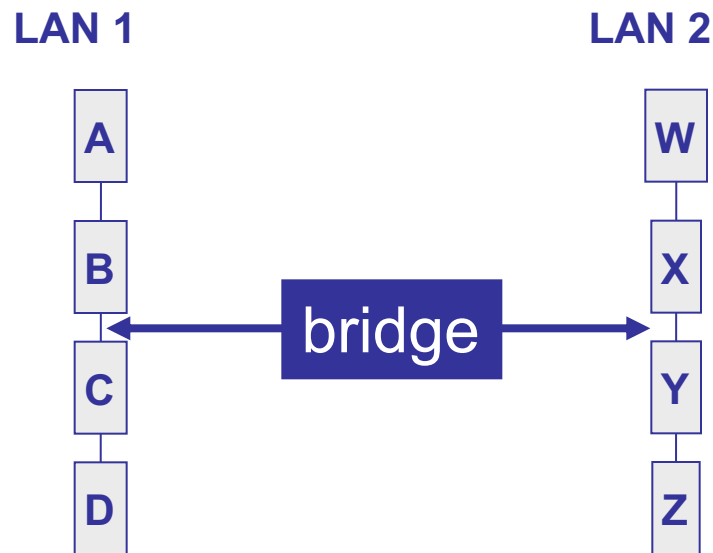


- Store and forward device
 - ◆ Data-link layer device
 - ◆ Buffers entire packet and *then* rebroadcasts it on other ports
- Creates *separate* Collision Domains (CD)
 - ◆ Senses line for access to each LAN (acts like a host)
 - ◆ Can accommodate different speed interfaces (issues?)
 - ◆ Separate CDs improves throughput (why?)
- Can significantly improve performance
 - ◆ Not all frames go everywhere. (Why did they with a hub?)



Selective Forwarding

- Only rebroadcast a frame to the LAN where its destination resides
 - ◆ If *A* sends packet to *X*, then bridge must forward frame
 - ◆ If *A* sends packet to *B*, then bridge shouldn't





Forwarding Tables

- Need to know “destination” of frame
 - ◆ Destination address in frame header (48bit in Ethernet)
- Need know which destinations are on which LANs
 - ◆ One approach: statically configured by hand
 - » Table, mapping address to output port (i.e. LAN)
 - ◆ But we’d prefer something automatic and dynamic...
- Simple algorithm:
Receive frame *f* on port *q*
Lookup *f.dest* for output port /* know where to send it? */
If *f.dest* found
then if output port is *q* then drop /* *already delivered* */
 else forward *f* on output port;
else flood *f*;
/* forward on all ports but the one where frame arrived*/



Learning Bridges

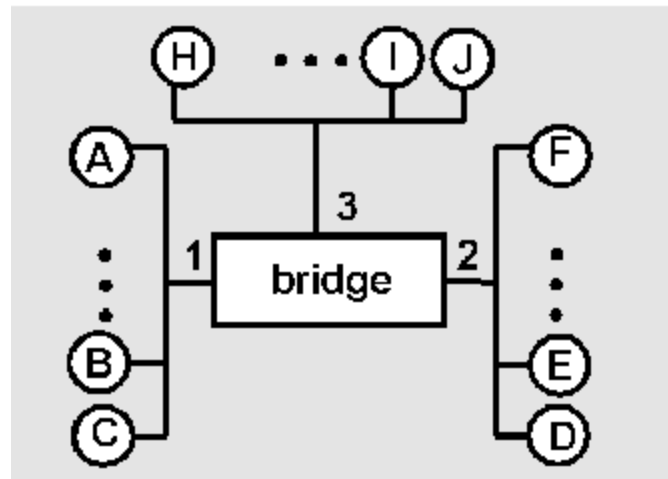
- Eliminate manual configuration by learning which addresses are on which LANs
- Basic approach
 - ◆ If a frame arrives on a port, then associate its **source** address with that port
 - ◆ As each host transmits, the table becomes accurate
- What if a node moves? Table aging
 - ◆ Associate a timestamp with each table entry
 - ◆ Refresh timestamp for each new packet with same source
 - ◆ If entry gets too stale, remove it

Host	Port
A	1
B	1
C	1
D	1
W	2
X	2
Y	3
Z	2



Learning Example

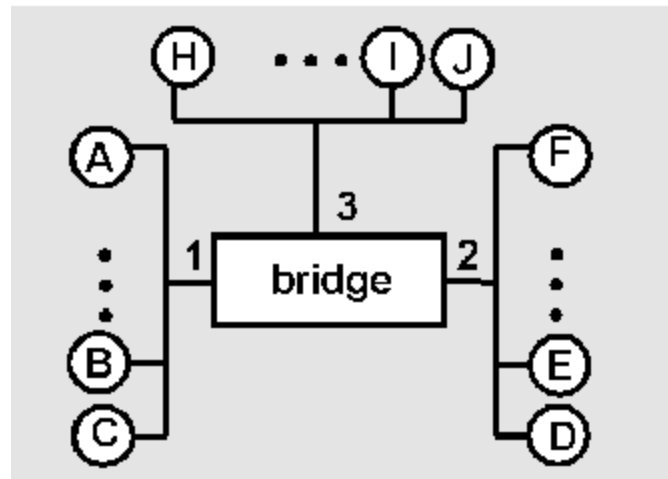
Suppose *C* sends frame to *D* and *D* replies back with frame to *C*



- C sends frame, bridge has no info about D, so floods to both LANs
 - ◆ bridge notes that C is on port 1
 - ◆ frame ignored on upper LAN
 - ◆ frame received by D



Learning Example



- *D* generates reply to *C*, sends
 - ♦ bridge sees frame from *D*
 - ♦ bridge notes that *D* is on port 2
 - ♦ bridge knows *C* on port 1, so **selectively** forwards frame via port 1



Learning Bridges Recap

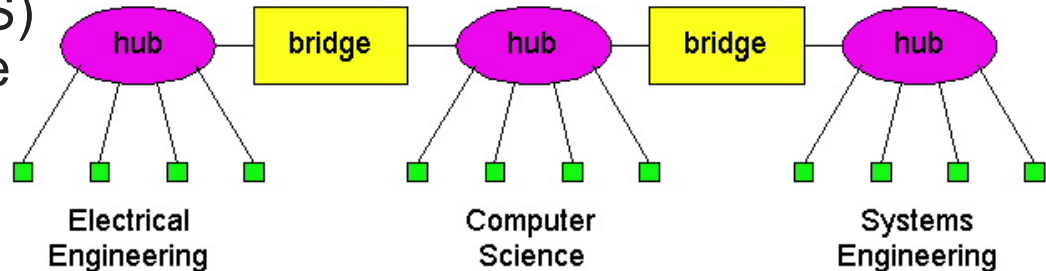
- Each bridge keeps a list mapping link-layer *destination* address to port number
 - ◆ (what are the directions to this destination?)
- This list is populated by looking at the *source* address of each packet it receives on a given port and entering those values in the table
 - ◆ (if a packet from *A* came from port *x*, then packets to *A* should be sent on port *x*)
- If a packet arrives with a destination address not in the table, then send on all ports (except the one it came on)
- Simple, automatic, self healing



Network Topology

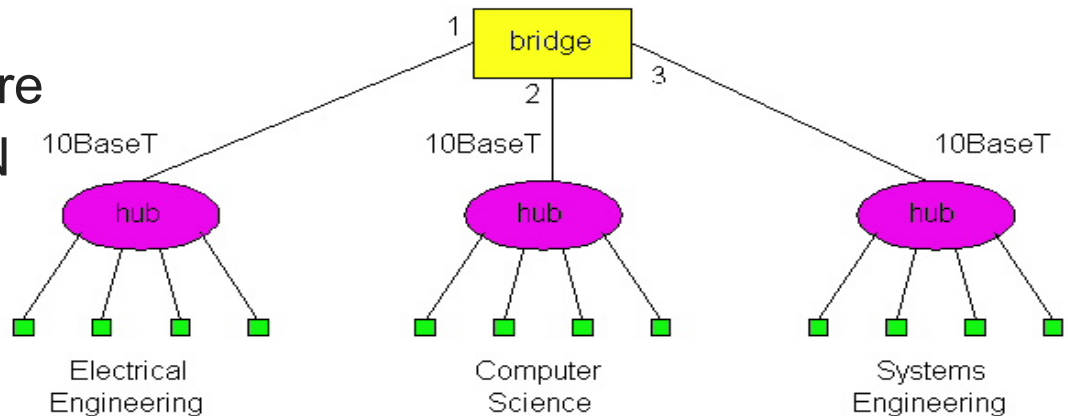
- Linear organization

- ◆ Inter-bridge hubs (e.g. CS) are single points of failure
- ◆ Unnecessary transit (e.g. EE $\leftrightarrow$ SE must traverse CS)



- Backbone/tree

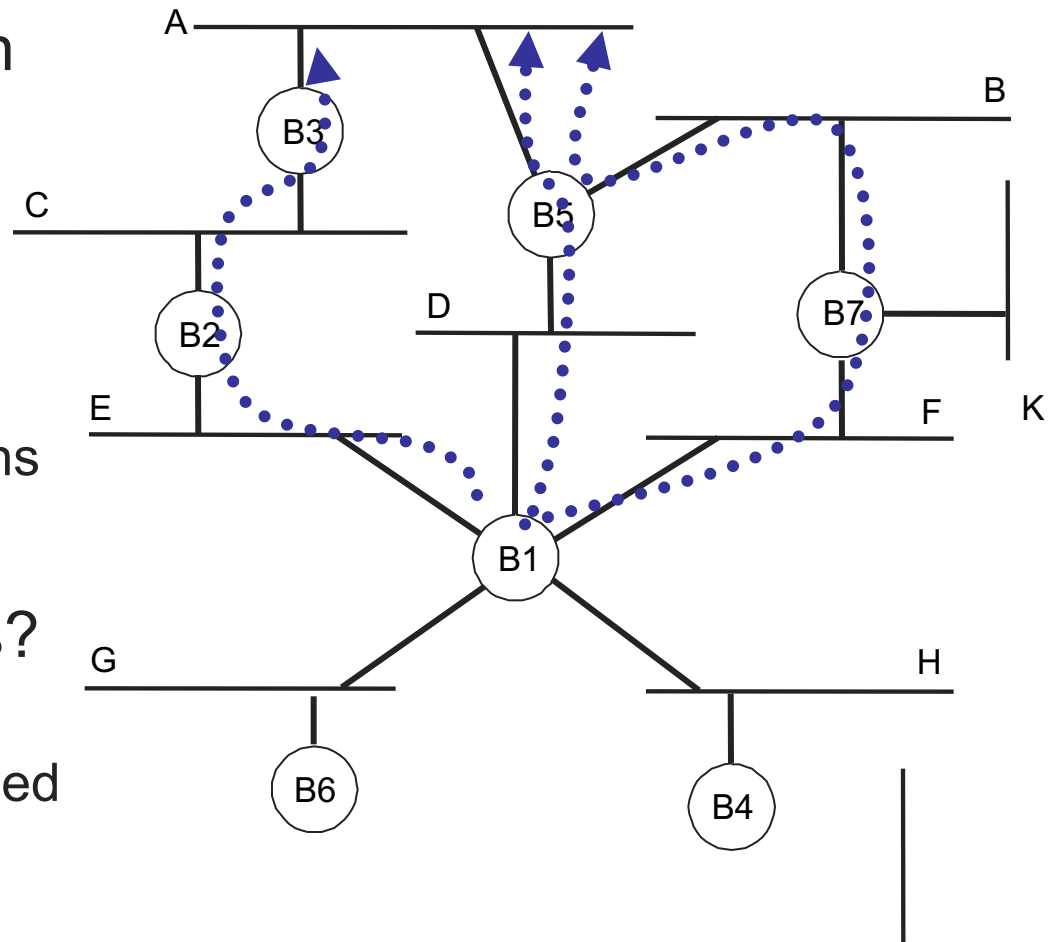
- ◆ Can survive LAN failure
- ◆ Manages all inter-LAN communication
- ◆ Requires more ports





An Issue: Cycles

- Learning works well in tree topologies
- But trees are fragile
 - ◆ Net admins like redundant/backup paths
- How to handle Cycles?
 - ◆ Where should *B1* forward packets destined for LAN *A* (3 copies of packets go to *A*!)





-
- The diagram illustrates a quantum circuit with 8 qubits (A-H) and 7 gates (B1-B7). The circuit is structured as follows:
- Qubits:** A, B, C, D, E, F, G, H.
 - Gates:** B1, B2, B3, B4, B5, B6, B7.
 - Connections:**
 - Gate B1 connects qubits A, B, C, and D.
 - Gate B2 connects qubits C and E.
 - Gate B3 connects qubits A and C.
 - Gate B4 connects qubits G and H.
 - Gate B5 connects qubits A, B, and D.
 - Gate B6 connects qubits G and I.
 - Gate B7 connects qubits B and F.
 - Annotations:**
 - A blue dashed arrow points from qubit A towards gate B5.
 - Red dashed lines connect qubits A, C, G, and I to gates B3, B2, B6, and B4 respectively.



Spanning Tree Algorithm

- Each bridge sends periodic configuration messages
 - ◆ (RootID, Distance to Root, BridgeID)
 - ◆ All nodes think they are root initially
- Each bridge updates route/Root upon receipt
 - ◆ Smaller root address is better
 - ◆ Select port with lowest cost to root as “root port”
 - ◆ To break ties, bridge with smaller address is better
- Rebroadcast new config to ports for which we’re “best”
 - ◆ Don’t bother sending config to LANs with better options
 - ◆ Add 1 to distance, send new configs on ports that haven’t told us about a shorter path to the root
- Only forward packets on ports for which we’re on the shortest path to root (prunes edges to form tree)

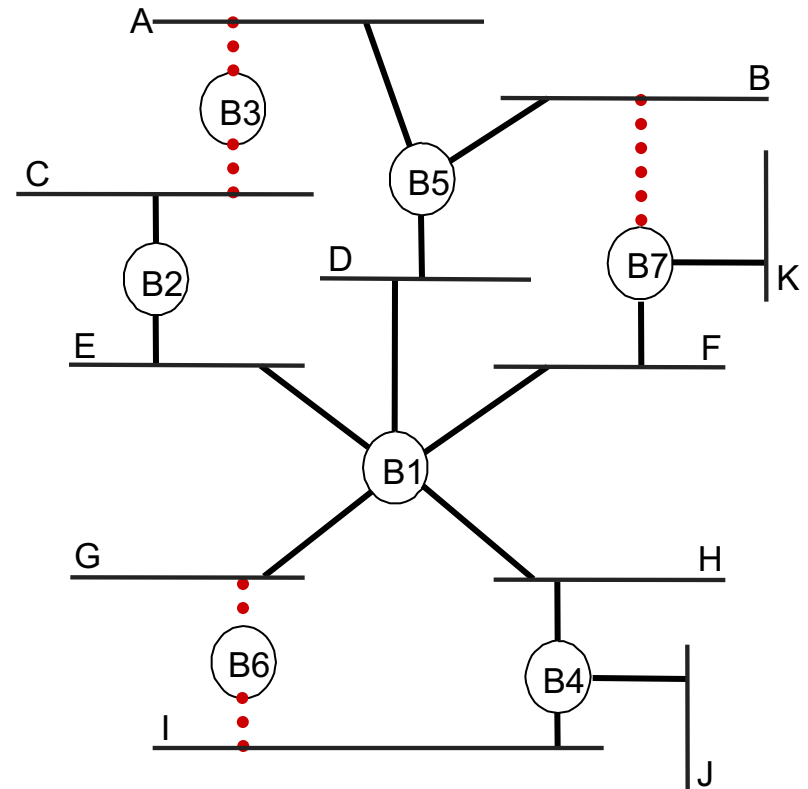
Spanning Tree Example



- Sample messages to and from B3:

1. B3 sends (B3, 0, B3) to B2 [LAN C] and B5 [LAN A]
2. B3 receives (B2, 0, B2) and (B5, 0, B5) and accepts B2 as root
3. B3 sends (B2, 1, B3) to B5
4. B3 receives (B1, 1, B2) and (B1, 1, B5) and accepts B1 as root
5. B3 wants to send (B1, 2, B3) but doesn't as its nowhere "best"
6. B3 receives (B1, 1, B2) [LAN C] and (B1, 1, B5) [LAN A] again and again...

B3 turns off forwarding to LAN A and C





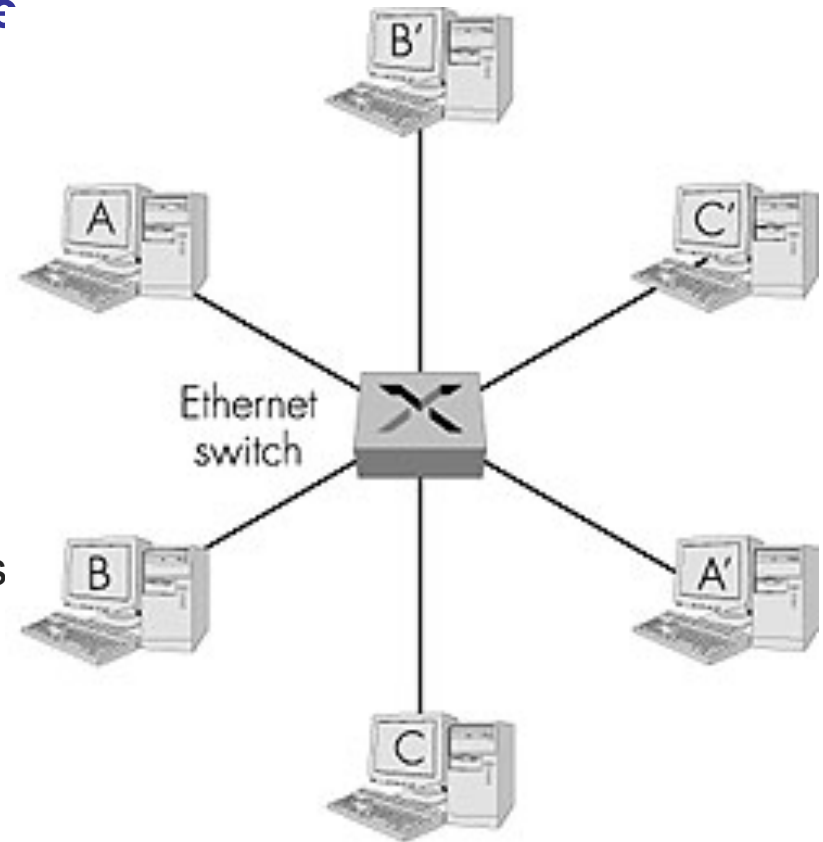
Important Details

- What if root bridge fails?
 - ◆ Age configuration info
 - » If not refreshed for MaxAge seconds then delete root and recalculate spanning tree
 - » If config message is received with a more recent age, then recalculate spanning tree
 - ◆ Applies to all bridges (not just root)
- Temporary loops
 - ◆ When topology changes, takes a bit for new configuration messages to spread through the system
 - ◆ Don't start forwarding packets immediately -> wait some time for convergence



Switched Ethernet

- Hosts **directly connected to a bridge**
 - ◆ learning + spanning tree protocol
- Switch supports parallel forwarding
 - ◆ A-to-B and A'-to-B' simultaneously
 - ◆ Generally full duplex as well
- **Switch** backplane capacity varies
 - ◆ Ideally, **nonblocking**
 - ◆ I.e., can run at full line rate on all ports
- No longer any shared bus
 - ◆ Each link is its own collision domain
 - ◆ Collision detection largely irrelevant





Layer-2 Forwarding

- Create spanning tree across LANs
 - ◆ Learn which ports to use to reach which addresses
- Benefits
 - ◆ Higher link bandwidth (point-to-point links)
 - ◆ Higher aggregate throughput (parallel communication)
 - ◆ Improved fault tolerance (redundant paths)
- Limitations
 - ◆ Requires homogeneous link layer (e.g. all Ethernet)
 - ◆ Harder to control forwarding topology
- What if we want to connect different link layers?



For Next Time

- Read 3.2.5-6