

Lecture 8:

Networks of Links

CSE 123: Computer Networks

Aaron Schulman

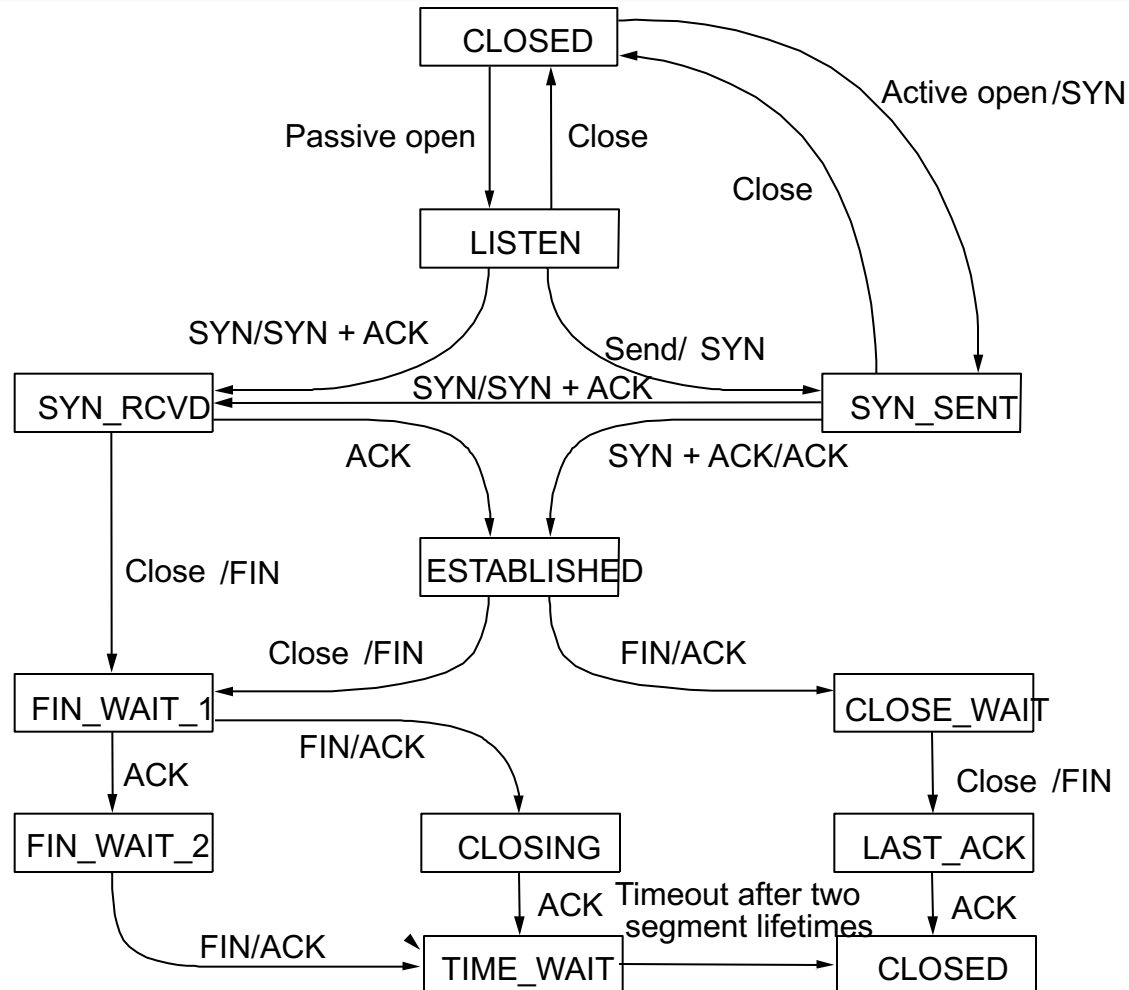




Lecture 8 Overview

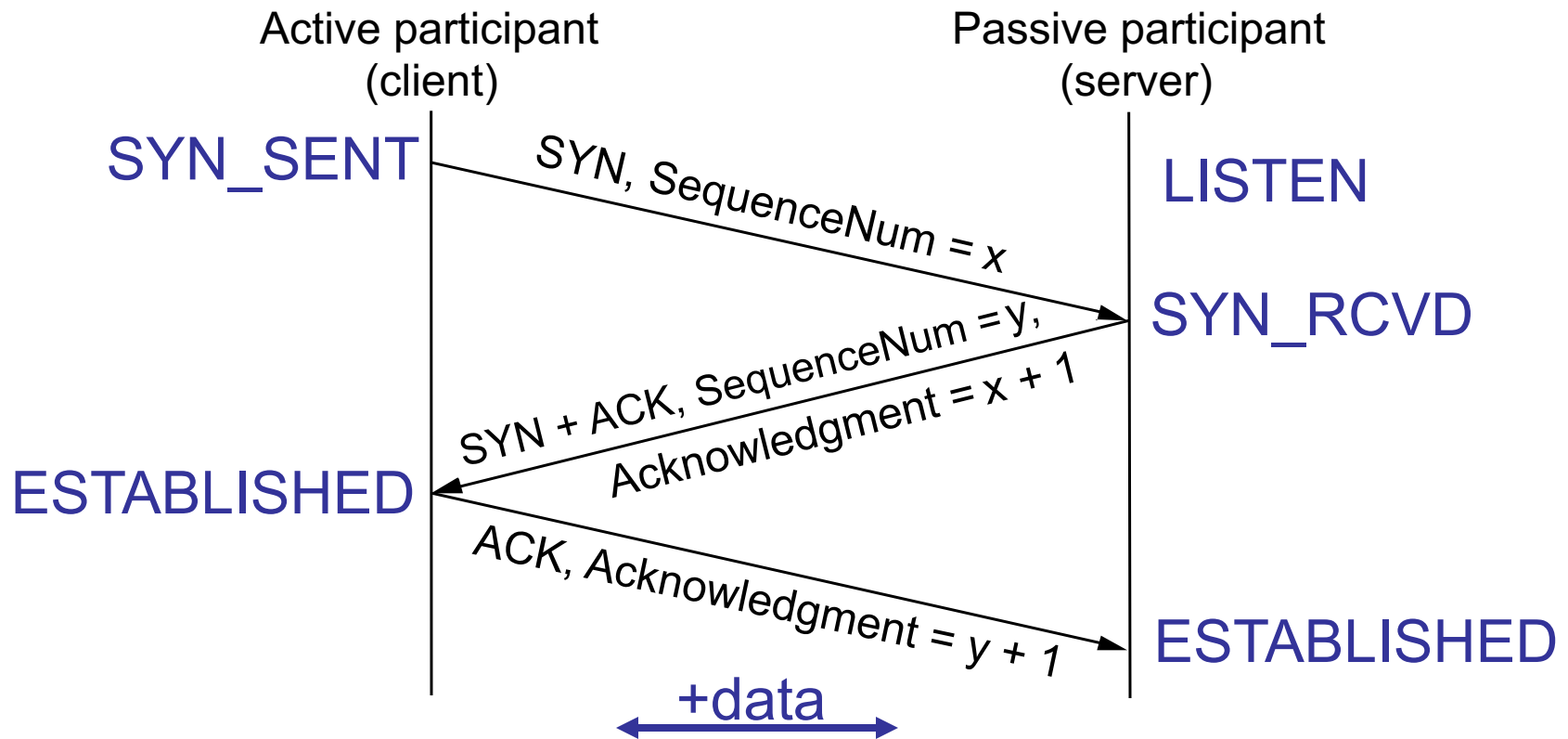
- Finish talking about TCP states
- Networks of Links
 - ◆ Hubs & Bridging & switching
- Spanning Tree review

TCP State Transitions





Connection, with States

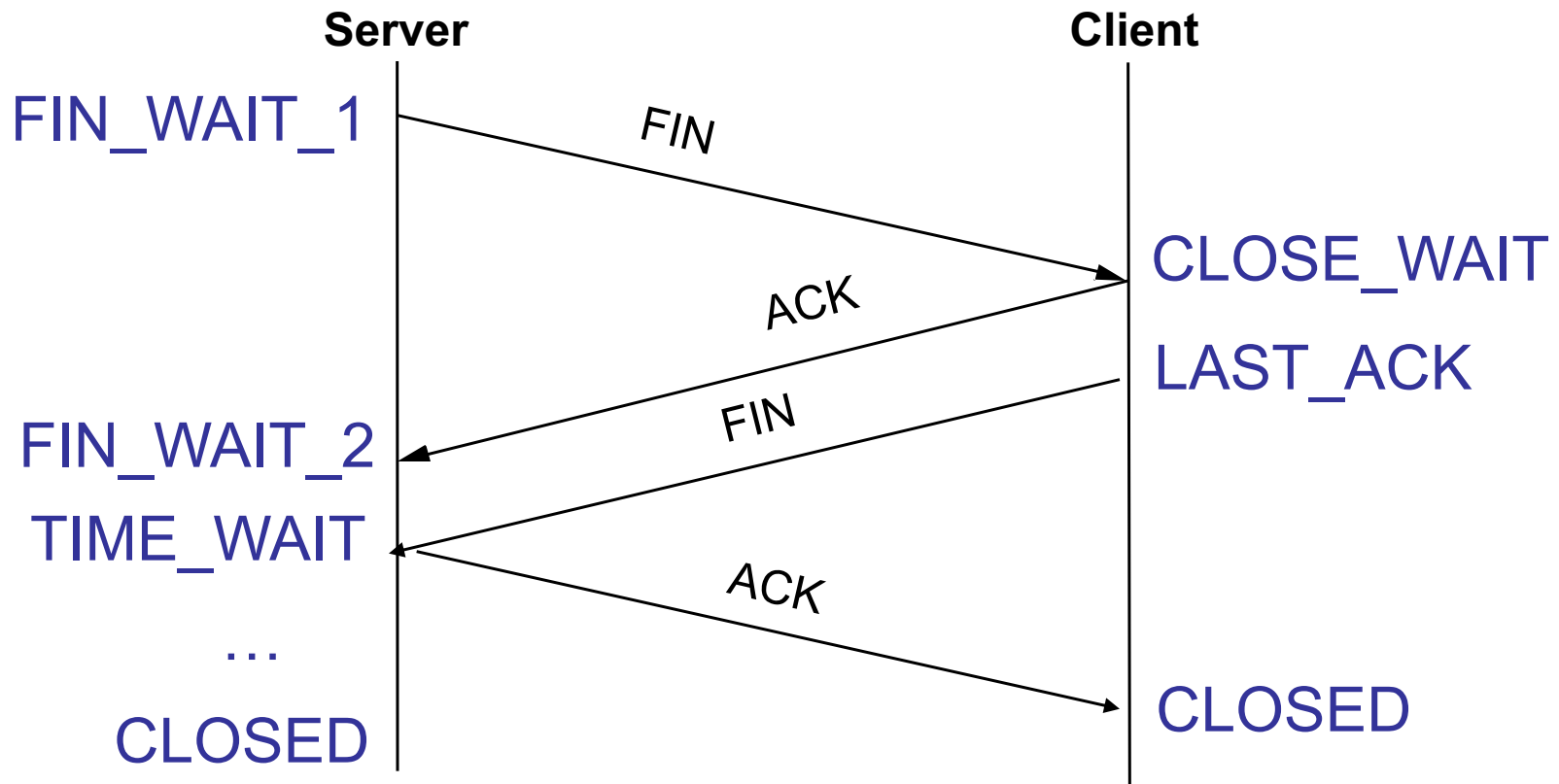




Connection Teardown

- Orderly release by sender and receiver when done
 - ◆ Delivers all pending data and “hangs up”
- Cleans up state in sender and receiver
- TCP provides a “symmetric” close
 - ◆ Both sides shutdown independently, why? You don’t know when the other side will be done!

TCP Connection Teardown





The TIME_WAIT State

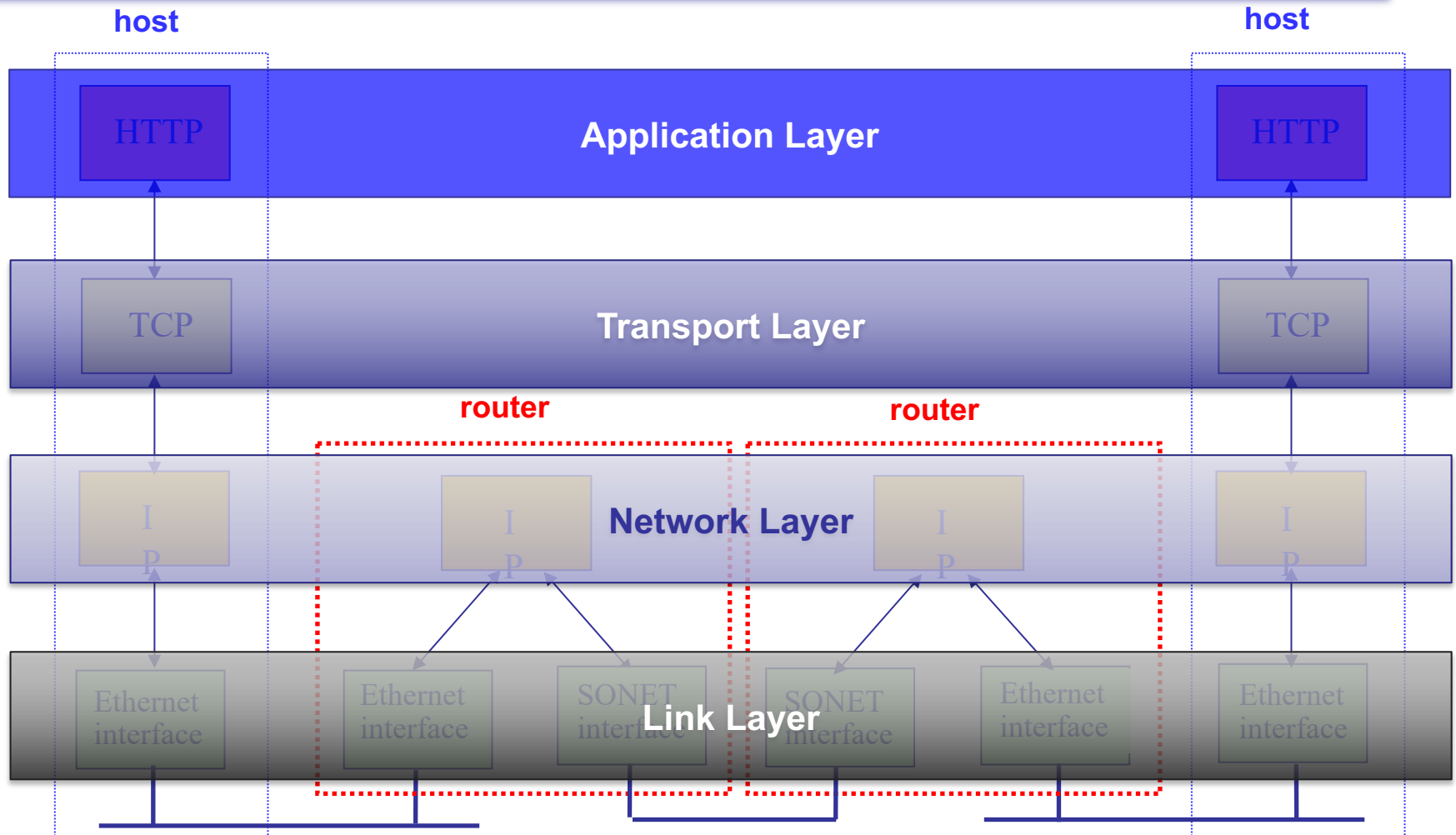
- We wait $2 \times \text{MSL}$ (maximum segment lifetime) before completing the close
- Maximum segment lifetime is usually 60 seconds (longest period of time packet can be stuck in network)
 - ◆ Why?
- ACK might have been lost and so FIN will be resent
 - ◆ Could interfere with a subsequent connection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

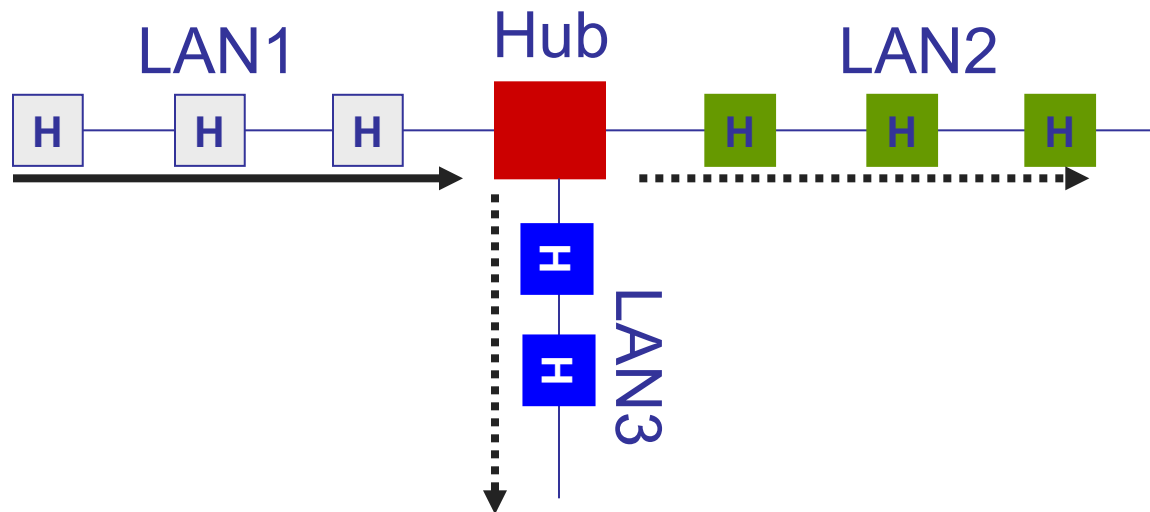
TCP/IP Protocol Stack





Hubs/Repeaters

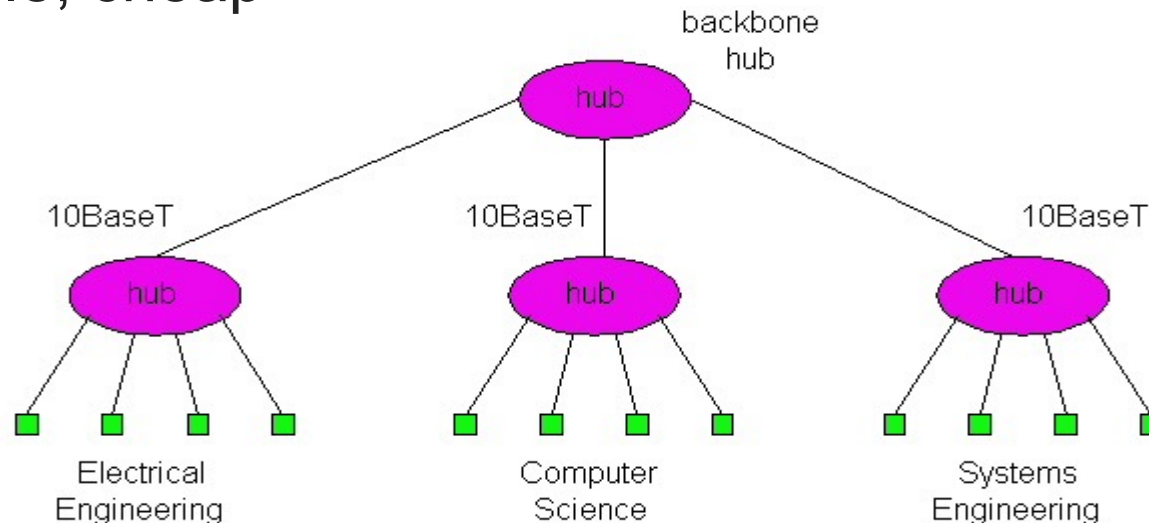
- Physical layer device
 - ◆ One “port” for each LAN
 - ◆ **Repeat** received *bits* on one port out *all* other ports





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 Shared Bus

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

Bridges

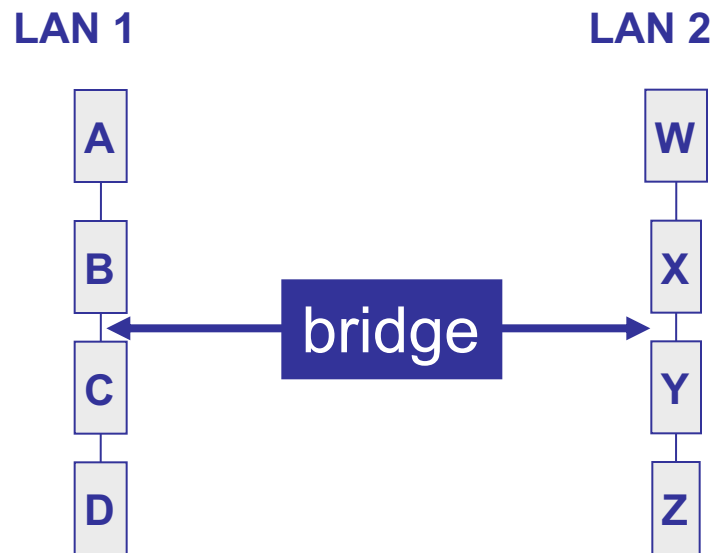


- Store and forward device
 - ◆ Data-link layer device
 - ◆ Buffers entire packet and *then* rebroadcasts it on other ports
 - ◆ Regenerates transmission so network can be longer distance
- Creates *separate* Collision Domains (CD)
 - ◆ Senses line for access to each LAN (acts like a host)
 - ◆ Can accommodate 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

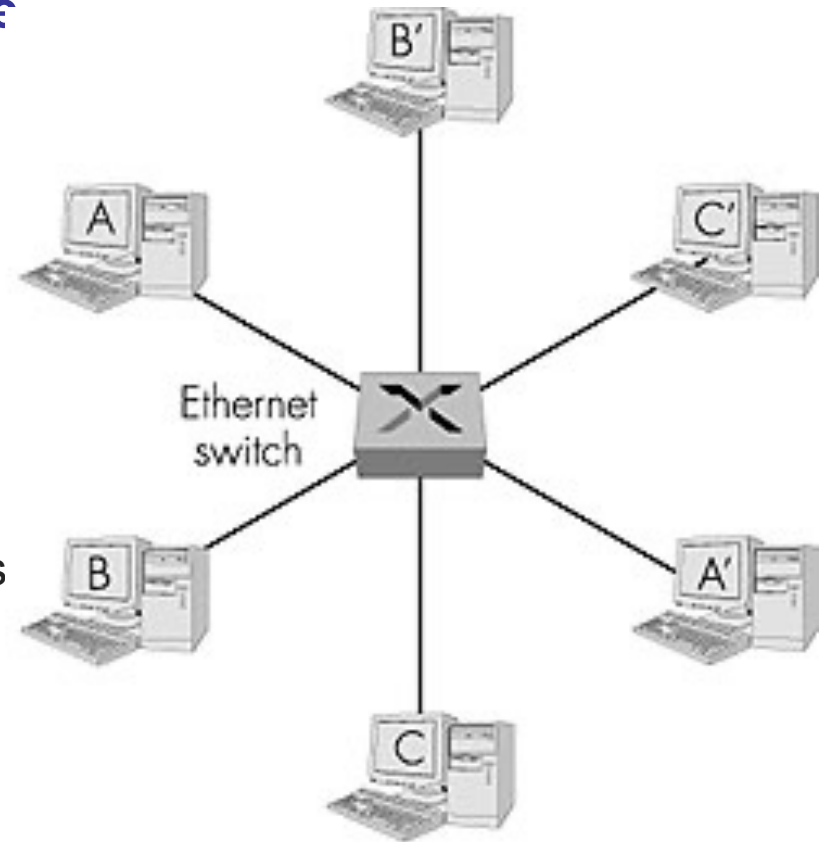


Switched Ethernet

(Common type of bridging)



- 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 sharing of links
 - ◆ Each link is its own collision domain
 - ◆ Collision detection largely irrelevant



Forwarding Tables in Bridges



- Need to know “destination” of frame
 - ◆ Destination “MAC” address in frame header (48bit in Ethernet)
- Need know which destinations are on which LANs
 - ◆ One approach: statically configure tables by hand
 - » Table, mapping address to output port (i.e. LAN)
 - ◆ But we’d prefer something automatic and dynamic...
- Simple algorithm for static forwarding table:
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*/



For Next Time

- Read 3.2.5 in P&D
- Homework 2 out today
- Start on Project 2