

Switched Ethernet

Announcements

- HW3 will be pushed out 2-3 days
 - Both release and due dates
 - Will post on piazza when we release
- Get started on project#3!

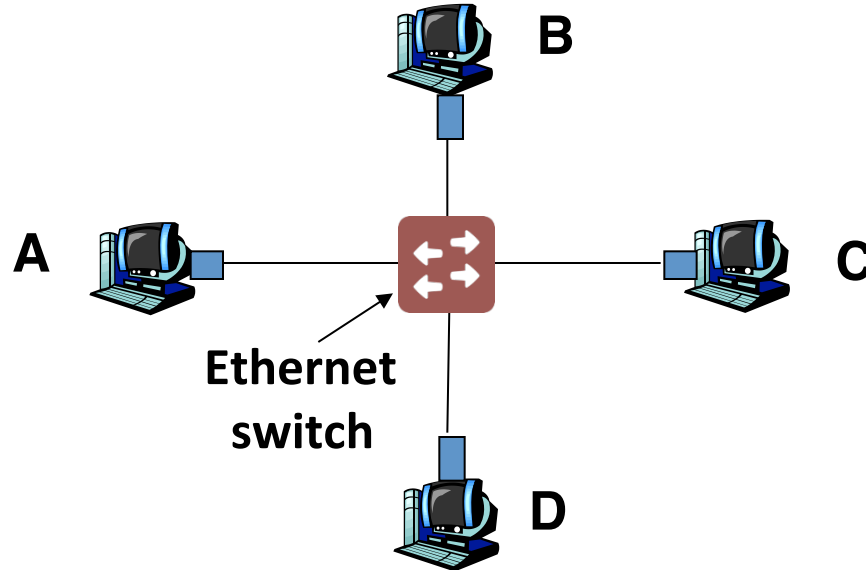
Last Time: Broadcast Ethernet

- Ethernet as a shared medium
- How hosts communicate using CSMA / CD
 - Carrier sense: wait for link to be idle
 - Collision detection: listen while transmitting
- Today: “Switched” Ethernet

Broadcast vs. Switched Ethernet

- Ethernet was invented as a broadcast technology
 - Each packet received by all attached hosts
 - CSMA/CD for media access control
- Current Ethernets are “switched”
 - Point-to-point links between switches and to hosts
 - No sharing, no CSMA/CD

Why Switched Ethernet?



- Enables concurrent communication
 - Host A can talk to C, while B talks to D
 - No collisions → no need for CSMA, CD
 - No constraints on link lengths, *etc.*

The evolution of Ethernet

- From the shared media coax cables to dedicated links
- From 3 Mbit/s experimental Ethernet to 100 Gbit/s
- From electrical signaling to optical
- **Changed almost everything except the frame format**
- **Lesson:** the right **interface** can accommodate many changes
 - Evolve the implementation while maintaining the interface (backward compatibility)

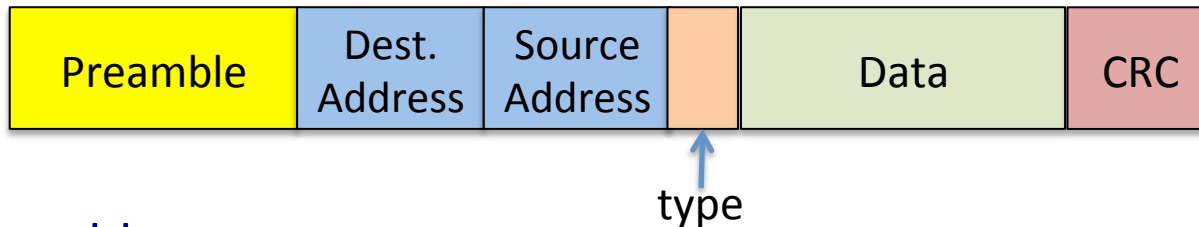
Topics

- Frames and framing
- Addressing
- Routing
- Forwarding

- Discovery: Bootstrapping end-to-end commn.

Ethernet “Frames”

- Encapsulates IP datagram



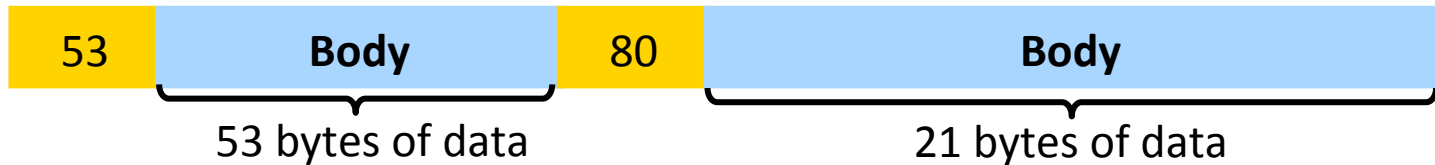
- **Preamble:**
 - 7 bytes for clock synchronization
 - 1 byte to indicate start of frame
- **Addresses:** 6 bytes
- **Type:** 2 bytes, indicating higher-layer protocol (e.g., IP, Appletalk)
- **Data payload:** max 1500 bytes, minimum 46 bytes
- **CRC:** 4 bytes for error detection

Framing frames

- Physical layer puts bits on a link
- But, two hosts connected on the same physical medium need to be able to exchange **frames**
 - Service provided by the **link layer**
 - Implemented by the network adaptor
- **Framing problem:** how does the link layer determine **where each frame begins and ends?**

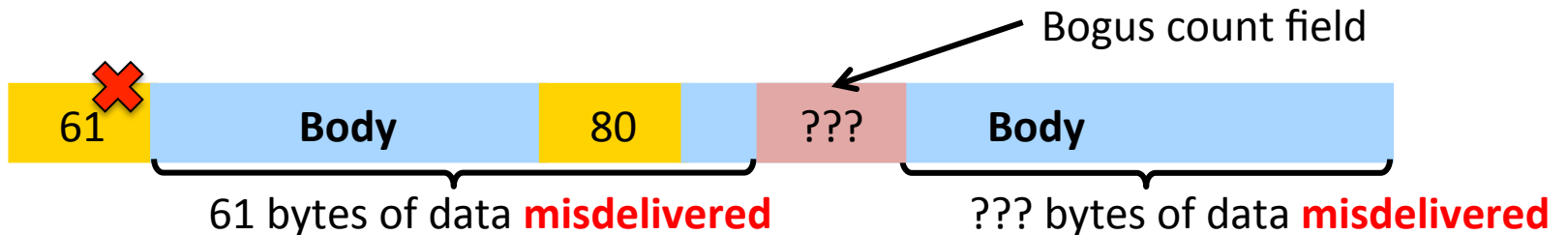
Simple approach: count bytes

- Sender includes number of bytes in header



- Receiver extracts this number of bytes of body

- But what if the Count field is corrupted?



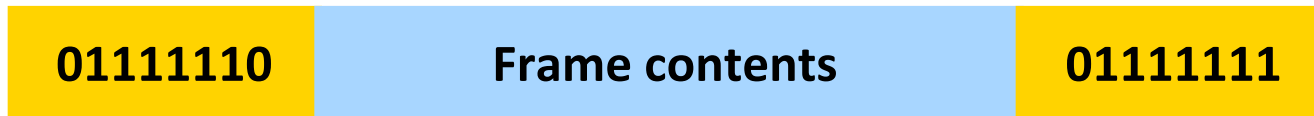
- L2 will frame the wrong bytes → a **framing error**
- CRC tells you to discard this frame, but what about the next one?

Desynchronization

- Once framing on a link is desynchronized, **it can stay that way**
- Need a method to **resynchronize**

Framing with sentinel bits

- Delineate frame with special “sentinel” **bit** pattern
 - *e.g.*, 01111110 \Rightarrow start, 01111111 \Rightarrow end



- **Problem:** what if sentinel occurs within frame?
- Solution: **bit stuffing**
 - **Sender** always inserts a **0** after five **1s** in the frame contents
 - **Receiver** always removes a **0** appearing after five **1s**

When receiver sees five 1s...



- If next bit **0**, remove it, and begin counting again
 - Because this must be a stuffed bit; we can't be at beginning/end of frame (those had six or seven **1s**)
- If next bit **1** (*i.e.*, we've seen six **1s**) then:
 - If following bit is **0**, this is start of frame
 - Because the receiver has seen **01111110**
 - If following bit is **1**, this is end of frame
 - Because the receiver has seen **01111111**

Example: sentinel bits

- Original data, including start/end of frame:

01111110011111101111101111100101111111

- Sender rule: five 1s → insert a 0

– After bit stuffing at the sender:

01111110011111010111110011111000101111111

- Receiver rule: five 1s and next bit 0 → remove 0

01111110011111101111101111100101111111

Outline

- Frames and framing
- Addressing
- Routing
- Forwarding

- Discovery: Bootstrapping end-to-end commn.

Medium Access Control Address

- MAC address
 - Numerical address associated with a network adapter
 - Flat name space of 48 bits (e.g., **00-15-C5-49-04-A9** in HEX)
 - Unique, hard-coded in the adapter when it is built
- Hierarchical Allocation
 - **Blocks**: assigned to vendors (e.g., Dell) by the IEEE
 - First 24 bits (e.g., **00-15-C5-**-**-****)
 - **Adapter**: assigned by the vendor from its block
 - Last 24 bits

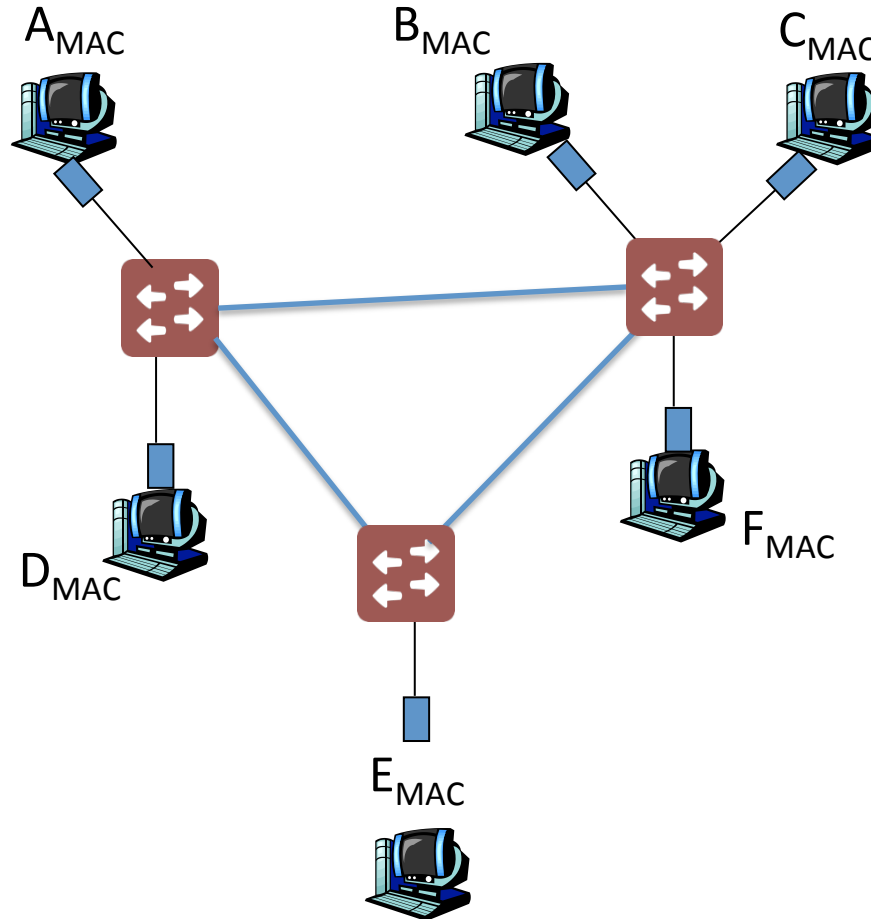
MAC Address vs. IP Address

- MAC addresses (used in link-layer)
 - **Hard-coded** when adapter is built
 - **Flat** name space of 48 bits (e.g., 00-0E-9B-6E-49-76)
 - Like a social security number
 - Portable, and can stay the same as the host moves
 - Used to get packet between interfaces on same network
- IP addresses
 - **Configured**, or learned dynamically
 - **Hierarchical** name space of 32 bits (e.g., 12.178.66.9)
 - Like a postal mailing address
 - Not portable, and depends on where the host is attached
 - Used to get a packet to destination IP subnet

Outline

- Frames and framing
- MAC addresses
- Routing
- Forwarding

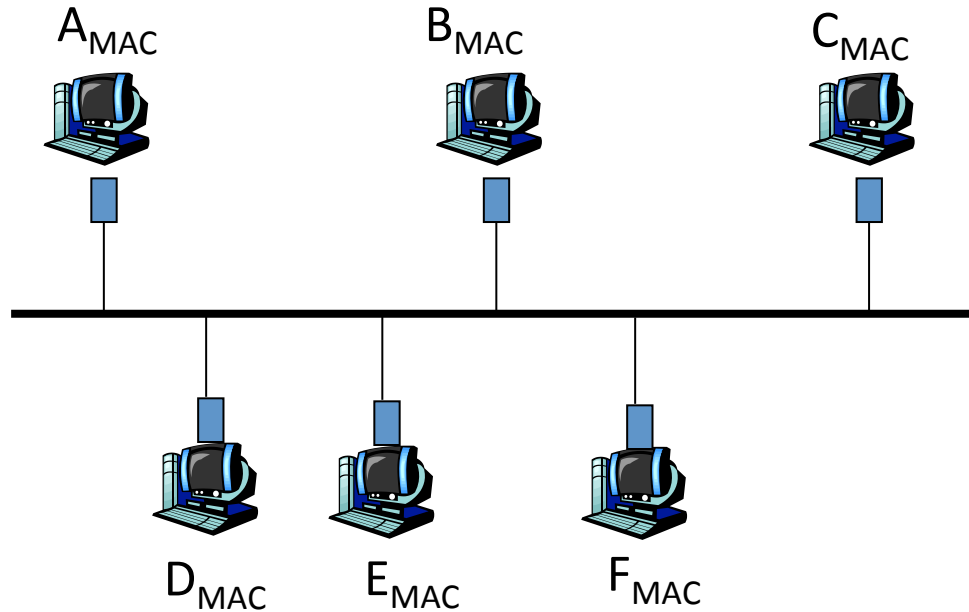
Routing with Switched Ethernet?



Why does Ethernet not use LS/DV?

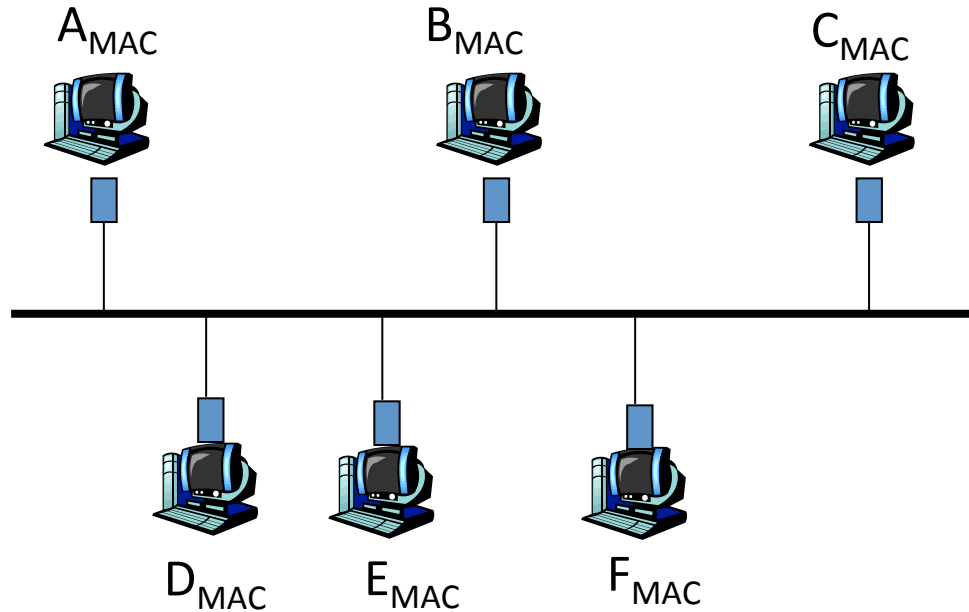
- Concerns over scalability
 - Flat MAC addresses cannot be aggregated like IP addresses
- Legacy

“Routing” with broadcast Ethernet



- Sender transmits frame onto broadcast link
- Frame contains destination MAC address
- Each receiver's link layer passes the frame to the network layer:
 - If destination address matches the receiver's MAC address
 - Or if the destination addr. is the broadcast MAC address (ff:ff:ff:ff:ff:ff)

“Routing” with broadcast Ethernet

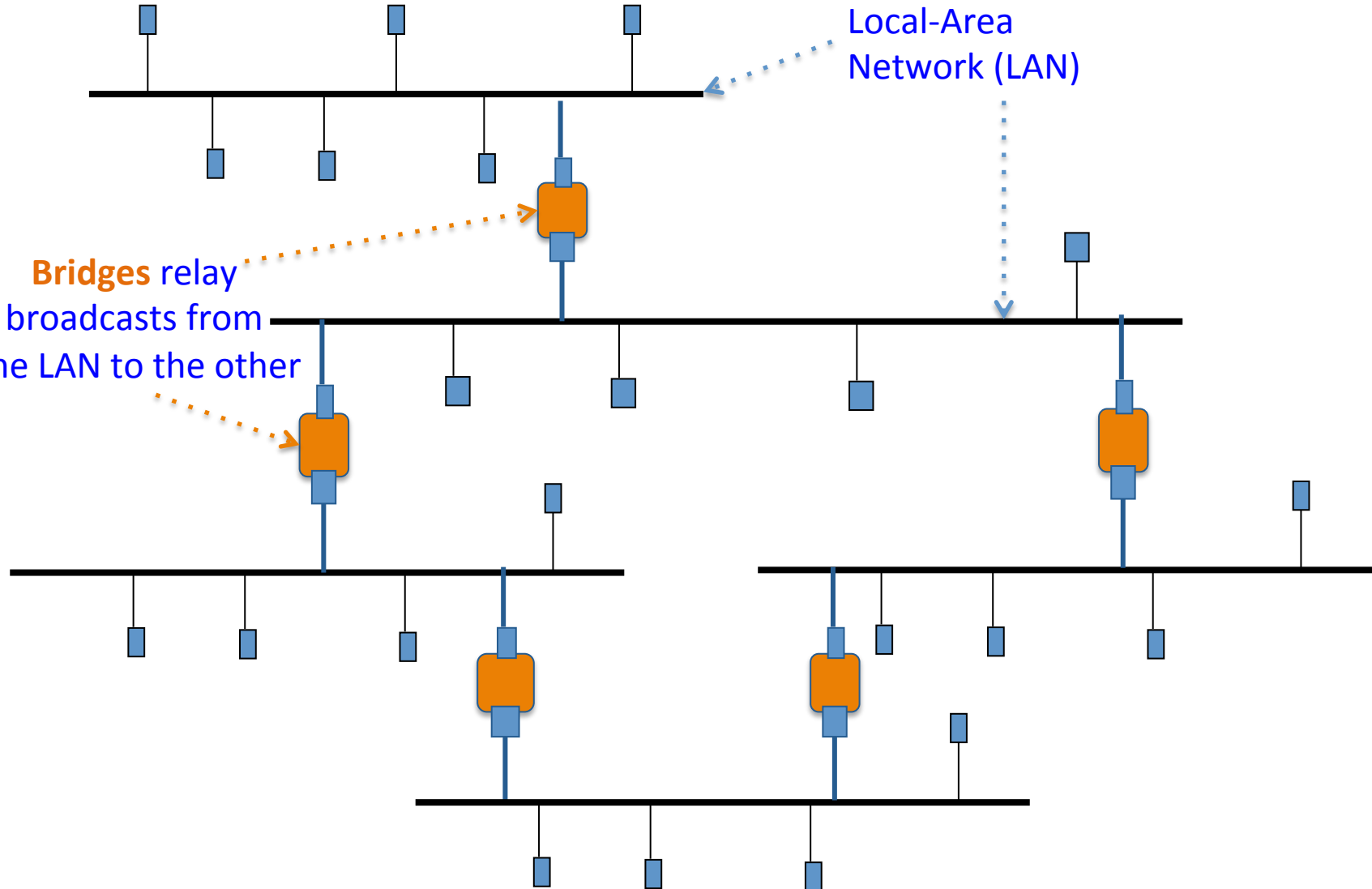


- Ethernet is ‘plug-n-play’
 - A new host plugs into the Ethernet and is good to go
 - No configuration by users or network operators
 - Broadcast as a means of bootstrapping communication

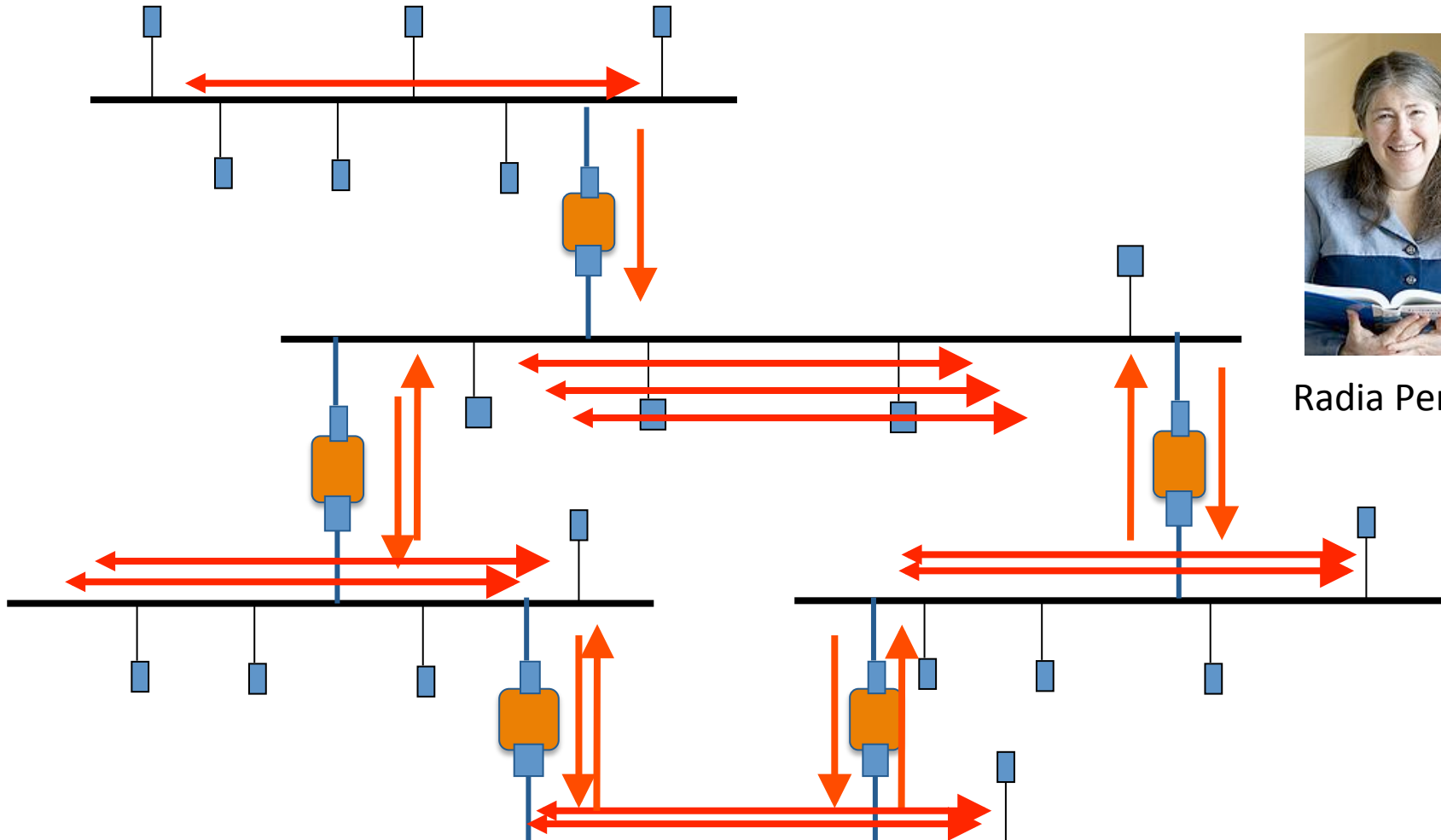
Why does Ethernet not use LS/DV?

- Concerns over scalability
 - Flat MAC addresses cannot be aggregated like IP addresses
- Legacy
 - Backward compatibility with broadcast Ethernet
 - Desire to maintain Ethernet's plug-n-play behavior
 - How broadcast Ethernet evolved

Routing in “Extended LANs”



The “Broadcast Storm” Problem



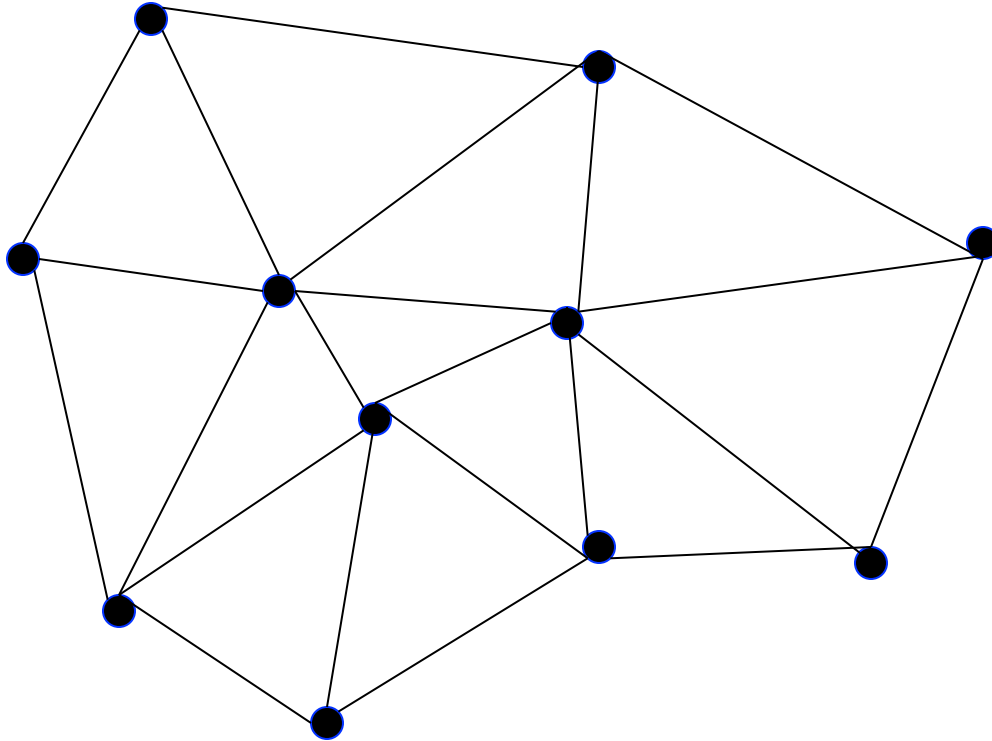
Radia Perlman

Perlman's idea: eliminate loops in the topology

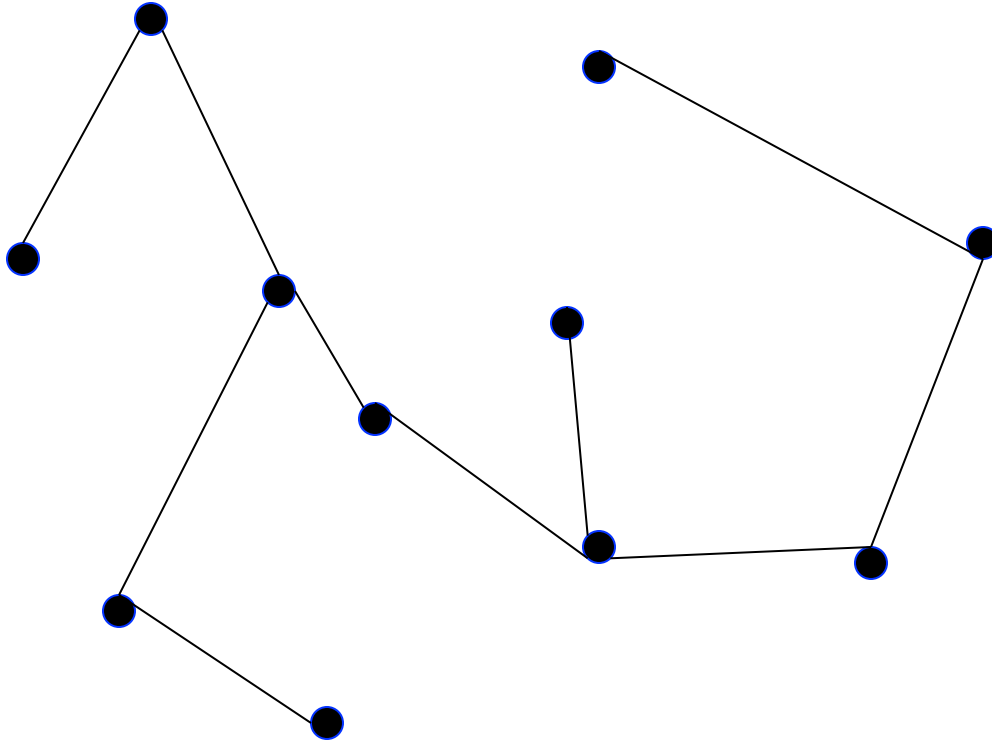
Easiest Way to Avoid Loops

- Use a topology where loops are impossible!
- Take arbitrary topology
- Build **spanning tree**
 - Sub-graph that includes all vertices but *contains no cycles*
 - Links not in the spanning tree are not used to forward frames
- Only one path to destinations on spanning trees
 - So don't have to worry about loops!

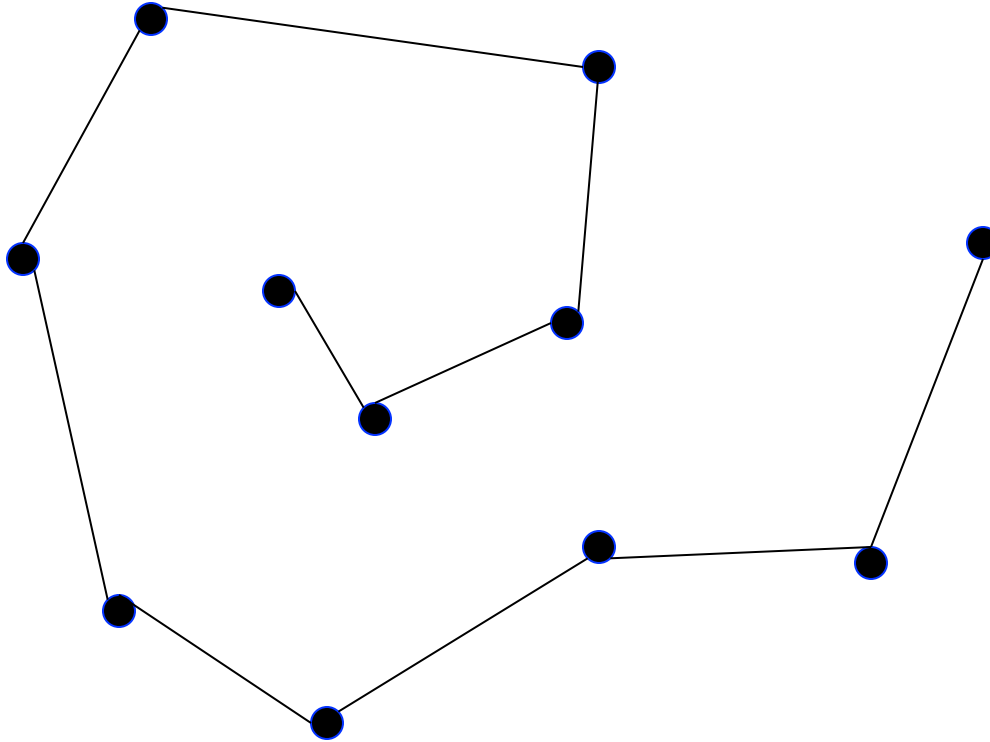
Consider graph



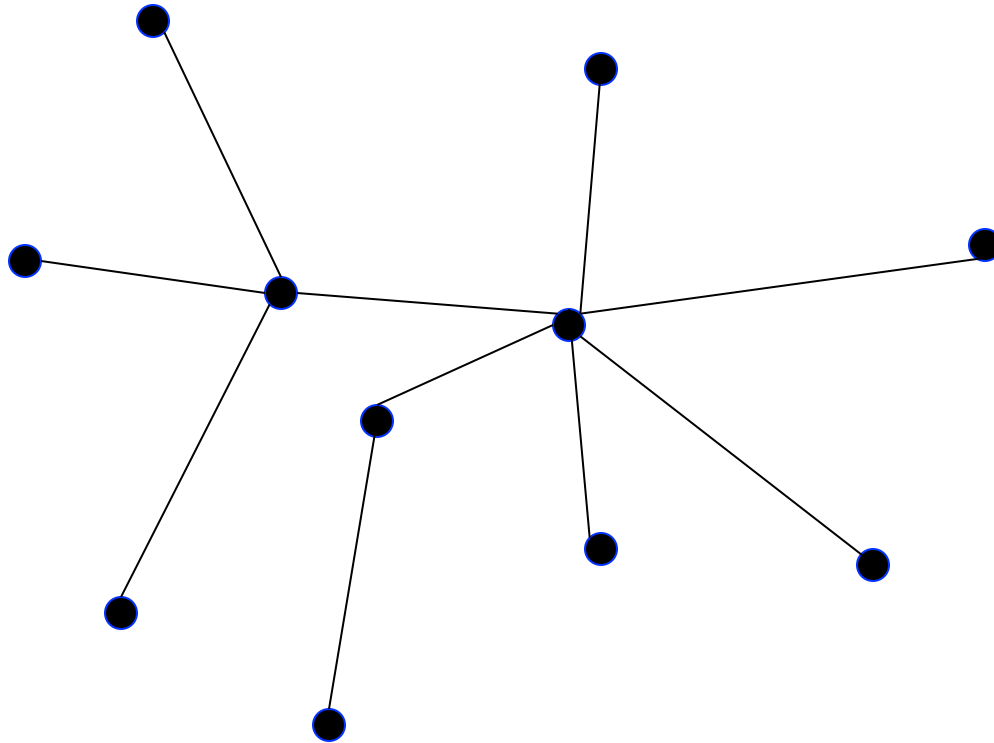
A Spanning Tree



Another Spanning Tree



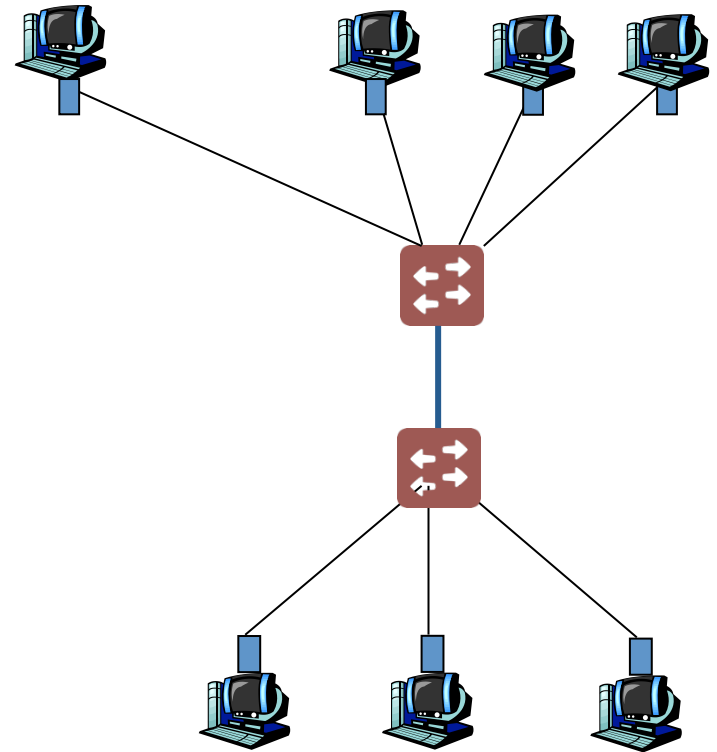
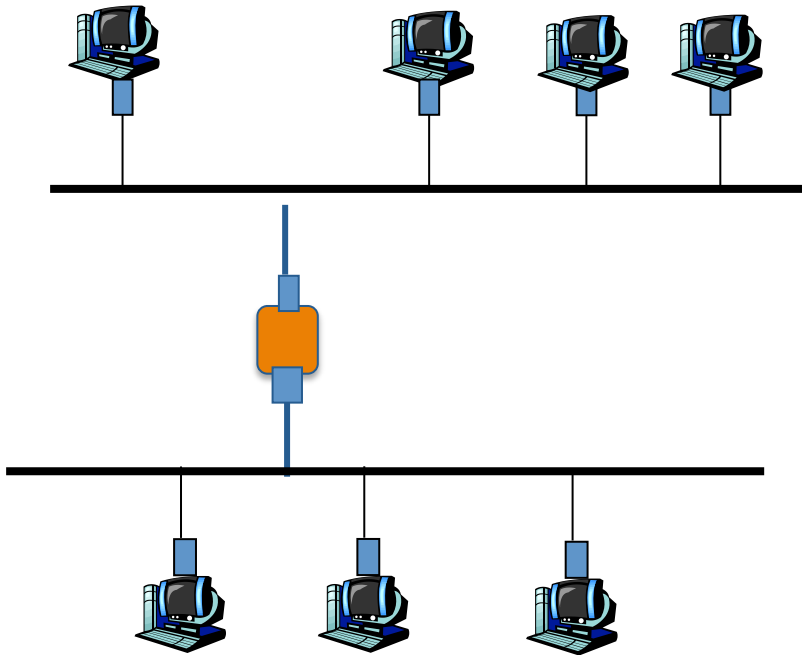
Yet Another Spanning Tree



Spanning Tree Protocol (Perlman'85)

- Protocol by which bridges construct a spanning tree
- Nice properties
 - Zero configuration (by operators or users)
 - Self healing
- Still used today

From Extended LANs to Switched Ethernet



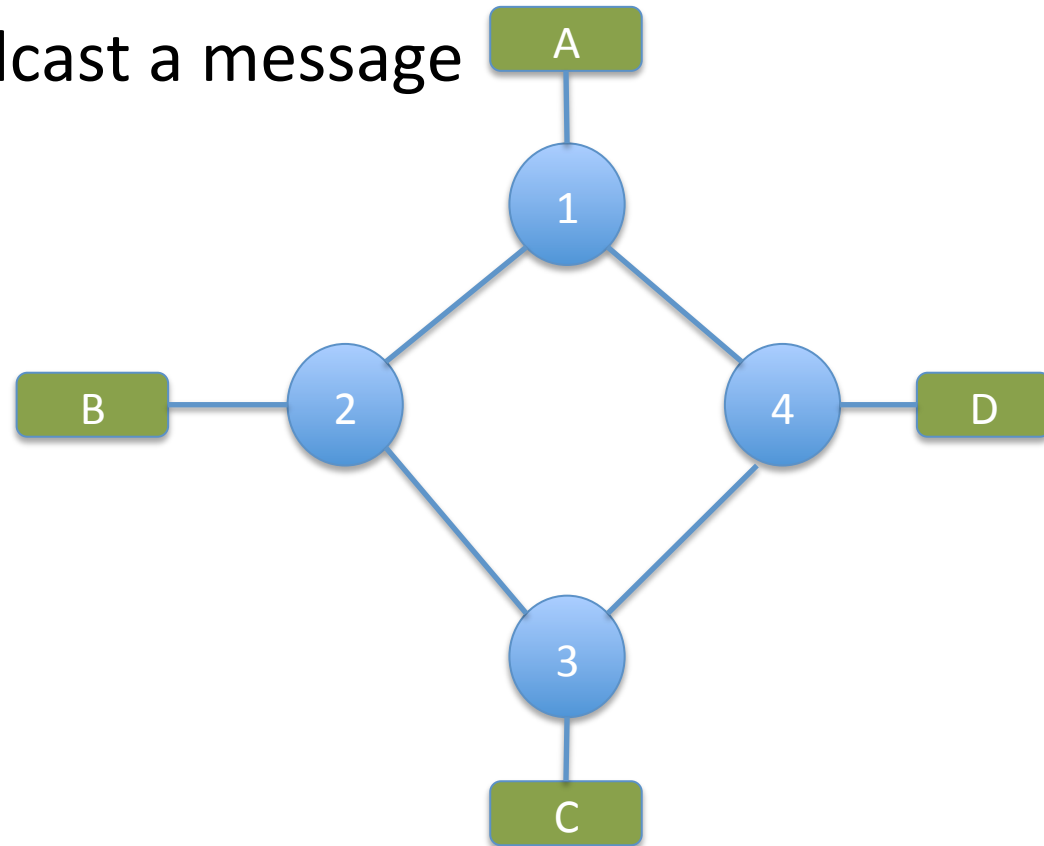
Switched Ethernet

- Constraints (for backward compatibility)
 - No changes to end-hosts
 - Maintain plug-n-play aspect
- Earlier Ethernet achieved plug-n-play by leveraging a broadcast medium
 - Can we do the same in a switched topology?

Flooding (still) leads to loops

Example: A wants to broadcast a message

- A sends packet to 1
- 1 Floods to 2 and 4
- 2 Floods to B and 3
- 4 Floods to D and 3
- 3 Floods packet from 2 to C and 4
- 3 Floods packet from 4 to C and 2
- 4 Floods packet from 3 to D and 1
- 2 Floods packet from 3 to B and 1
- 1 Floods packet from 2 to A and 4
- 1 Floods packet from 4 to B and 2
-

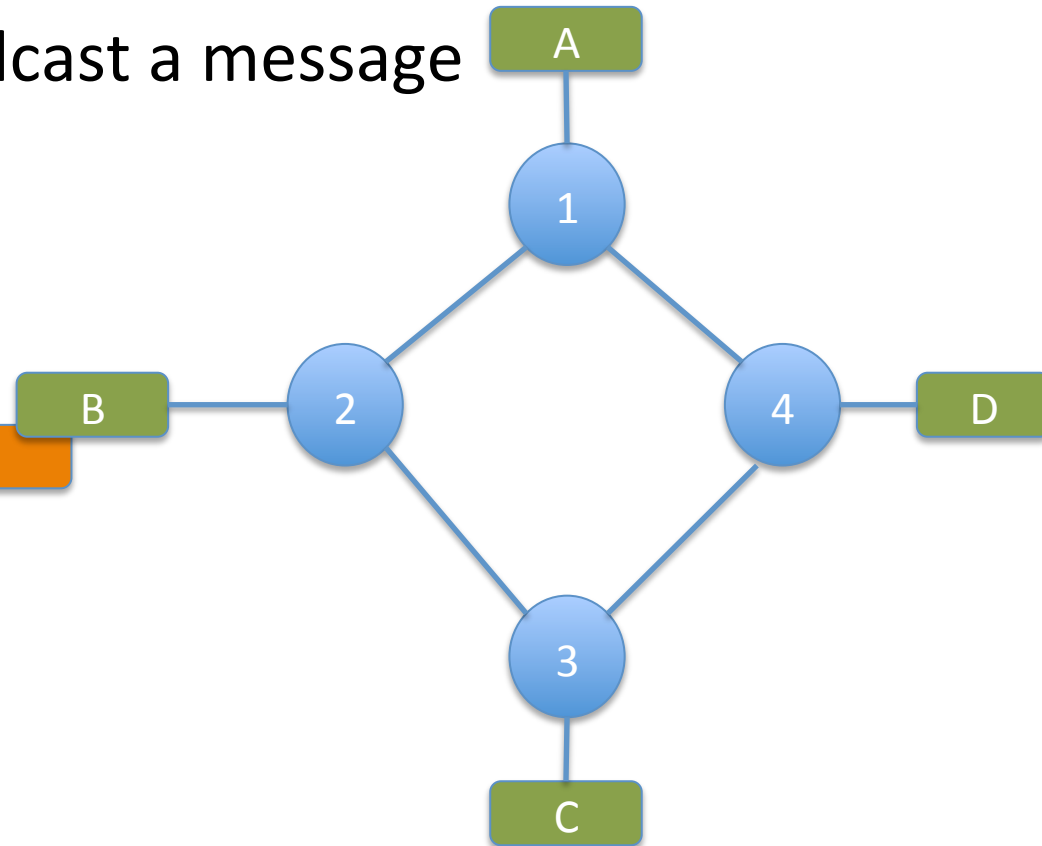


- A “broadcast storm” if the network contains a cycle of switches

A Rejected Solution

Example: A wants to broadcast a message

- A sends packet to 1
- 1 Floods to 2 and 4
- 2 Floods to B and 3
- 4 Floods to D and 3
- 3 Floods packet from 2 to C and 4
- 3 Floods packet from 4 to C and 2
- done!



Can we have 3 remember that it has already received and flooded the message?

Spanning Tree Approach

- Take arbitrary topology
- Pick subset of links that form a spanning tree

You: Design a Spanning Tree Algorithm

Take 5 minutes, design with your neighbors,
report back

- Goals
 - Distributed
 - Self-configuring
 - Must adapt when failures occur
 - But don't worry about that on first try...

Algorithm Has Two Aspects

- Pick a root:
 - Destination to which shortest paths go
 - Pick the one with the smallest identifier (MAC addr.)
- Compute shortest paths to the root
 - No shortest path can have a cycle
 - Only keep the links on shortest-paths
 - Break ties in some way (so we only keep one shortest path from each node)
- Ethernet's spanning tree construction does both with a single algorithm

Breaking Ties

- When there are multiple shortest paths to the root, choose the path that uses the neighbor switch with the lower ID.
- One could use any tiebreaking system, but this is an easy one to remember and implement

Constructing a Spanning Tree

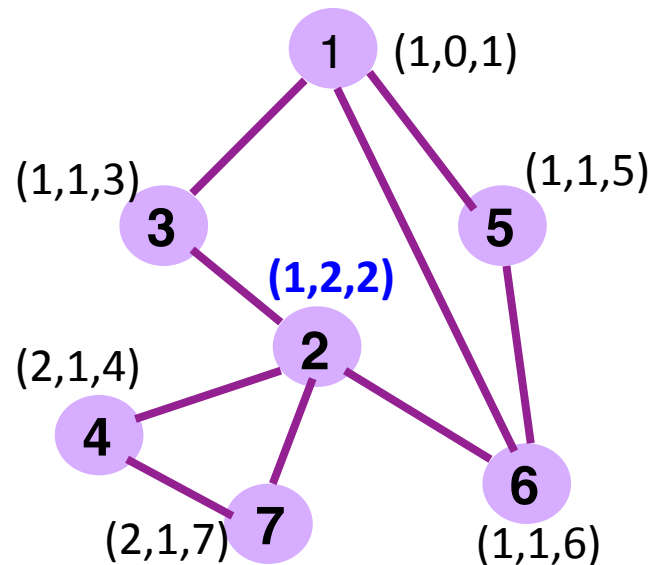
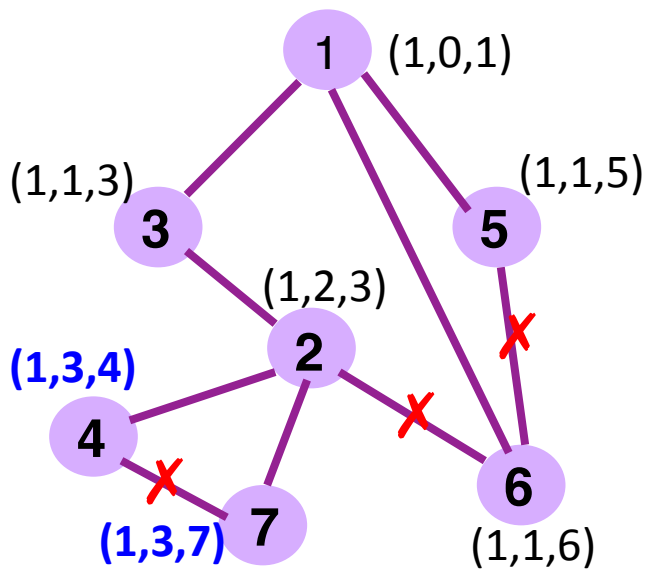
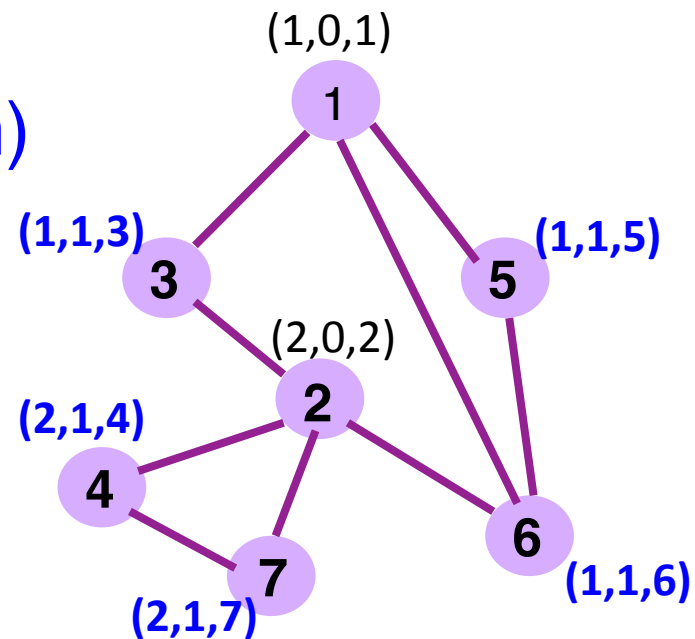
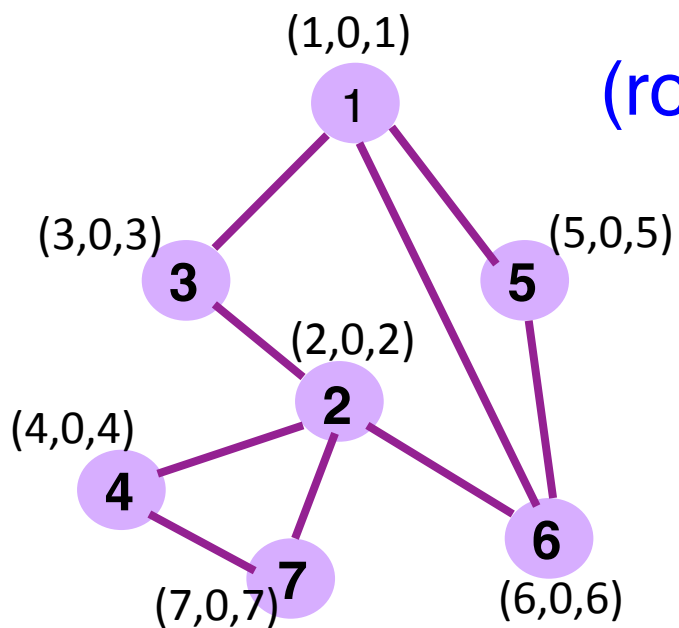
- **Messages (Y, d, X)**
 - From node X
 - Proposing Y as the root
 - And advertising a distance d to Y
- Switches elect the node with smallest identifier (MAC address) as root
 - **Y** in the messages
- Each switch determines if a link is on its shortest path to the root; excludes it from the tree if not
 - **d** to **Y** in the message

Steps in Spanning Tree Algorithm

- Initially, each switch proposes itself as the root
 - I.e., switch X announces $(X, 0, X)$ to its neighbors
- Switches update their view of the root
 - Upon receiving message (Y, d, Z) from Z , check Y 's id
 - If Y 's id $<$ current root: set root = Y
- Switches compute their distance from the root
 - Add 1 to the shortest distance received from a neighbor
- If root or shortest distance to it **changed**, send neighbors updated message $(Y, d+1, X)$

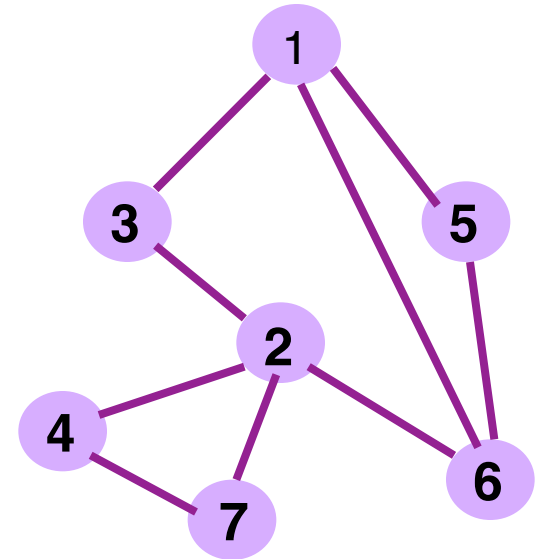
Example

(root, dist, from)



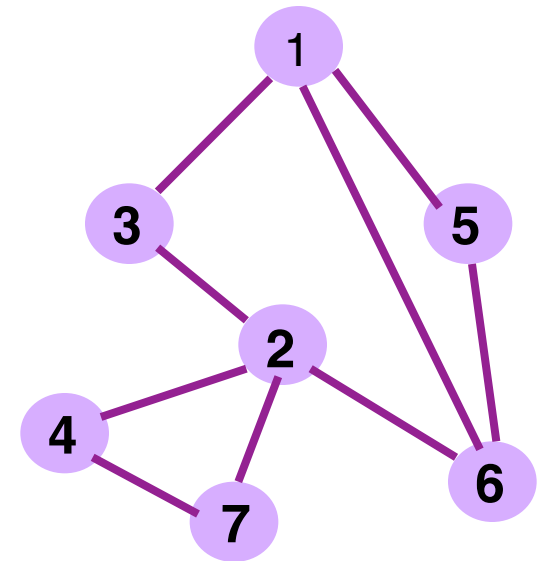
Example From Switch #4's Viewpoint

- Switch #4 thinks it is the root
 - Sends (4, 0, 4) message to 2 and 7
- Then, switch #4 hears from #2
 - Receives (2, 0, 2) message from 2
 - ... and thinks that #2 is the root
 - And realizes it is just one hop away
- Then, switch #4 hears from #7
 - Receives (2, 1, 7) from 7
 - And realizes this is a longer path
 - So, prefers its own one-hop path
 - And removes 4-7 link from the tree



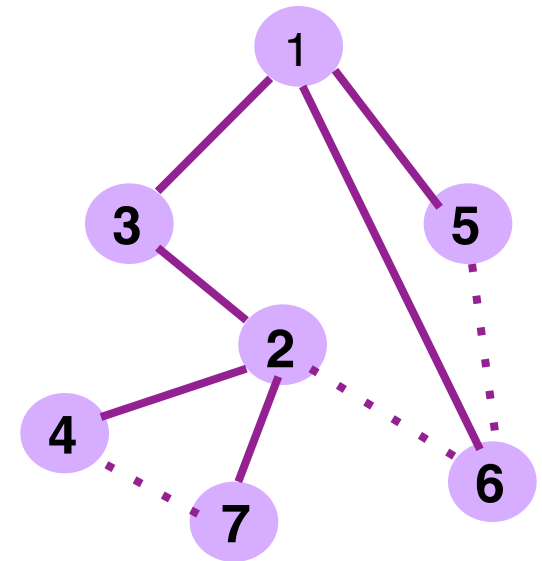
Example From Switch #4' s Viewpoint

- Switch #2 hears about switch #1
 - Switch 2 hears (1, 1, 3) from 3
 - Switch 2 starts treating 1 as root
 - And sends (1, 2, 2) to neighbors
- Switch #4 hears from switch #2
 - Switch 4 starts treating 1 as root
 - And sends (1, 3, 4) to neighbors
- Switch #4 hears from switch #7
 - Switch 4 receives (1, 3, 7) from 7
 - And realizes this is a longer path
 - So, prefers its own three-hop path
 - And removes 4-7 link from the tree



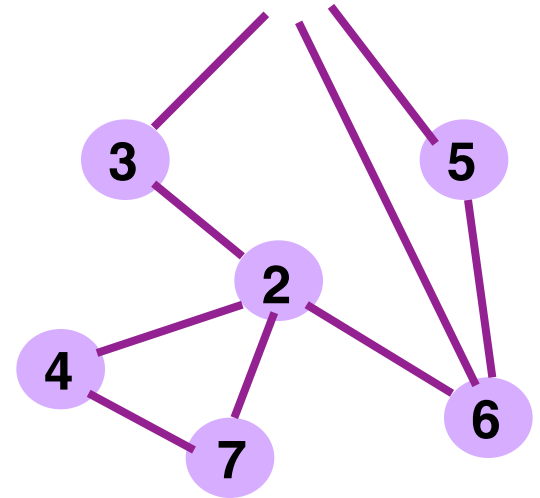
Links on spanning tree

- 3-1
- 5-1
- 6-1
- 2-3
- 4-2
- 7-2



Now which ones are on the spanning tree?

- 2 is new root
- 3-2
- 6-2
- 4-2
- 7-2
- 5-6



Robust Spanning Tree Algorithm

- Algorithm must react to **failures**
 - Failure of the root node
 - Failure of other switches and links
- Root switch sends periodic root announcement messages
 - Other switches continue forwarding messages
- Detecting failures through timeout (**soft state**)
 - If no word from root, *time out and claim to be the root!*

Outline

- Frames and framing
- MAC addresses
- Routing
- Forwarding

- Discovery: Bootstrapping end-to-end commn.