

CS4450

Computer Networks: Architecture and Protocols

Lecture 7

“Why” Frames

“Why” Switched Ethernet

Spanning Tree Protocol

Rachit Agarwal



Life Lessons

- **Life is full of important people and events**
 - **YOU**, my PhD students, family, colleagues, deadlines, friends, me ...
- **Sharing life across people is like sharing networks between users**
 - Delays mostly due to just transmission and propagation;
 - My meetings, sleep (rare, but happens)
 - When $\#incoming\text{-packets} > capacity$, queueing delay is inevitable;
 - If $\#emails > \text{what I can handle}$, queueing delay (current status)
 - Sometimes failures happen — requires retransmitting packets
- **Last week was one of those for me**
 - Queueing delay at my inbox (too many emails to handle)
 - **I am reducing the queue sizes ...**
 - **Help me!**

Announcements

- Please give your TAs more work to do
 - **I am happy to receive emails**
 - **Please cc the lead TA on emails: Daniel (da462@)**
- Problem Set 1 solutions are out (on Piazza)

Goals for Today's Lecture

- **Dive deep into Link layer design**
 - Finish CSMA/CD
 - Why Frames? — Implementing Link Layer on top of Physical Layer
 - Why Switched Ethernet? — Understanding scalability problems
- Switched Ethernet: **THE Spanning Tree Protocol**

Recap from last lecture

Recap: Network Design Goals

- **Build something that works**
- Connect existing networks
- Robust in face of failures
- Support multiple types of delivery service
- Accommodate a variety of networks
- Allow distributed management
- Easy host attachment
- Cost effective
- Allow resource accountability

Recap: Some missing goals

- **Performance**
- Security
 - Resilience to attacks (denial-of-service)
 - Endpoint security
 - Tracking down misbehaving users
- Privacy
- Availability
- Resource sharing (fairness, etc.)
- ISP-level concerns
 - Economic issues of interconnection

Recap: Data Link Layer

- **Communication Medium**
 - **Point-to-point**
 - The high-level ideas discussed so far were for point-to-point
 - **Broadcast**
 - Original design of Link layer protocols
 - More recent versions have moved to point-to-point
 - We will discuss why so!
- **Network Adapters (e.g., NIC — network interface card)**
 - The hardware that connects a machine to the network
 - Has a “name” — MAC (Medium access control) address



Recap: Sharing a broadcast channel

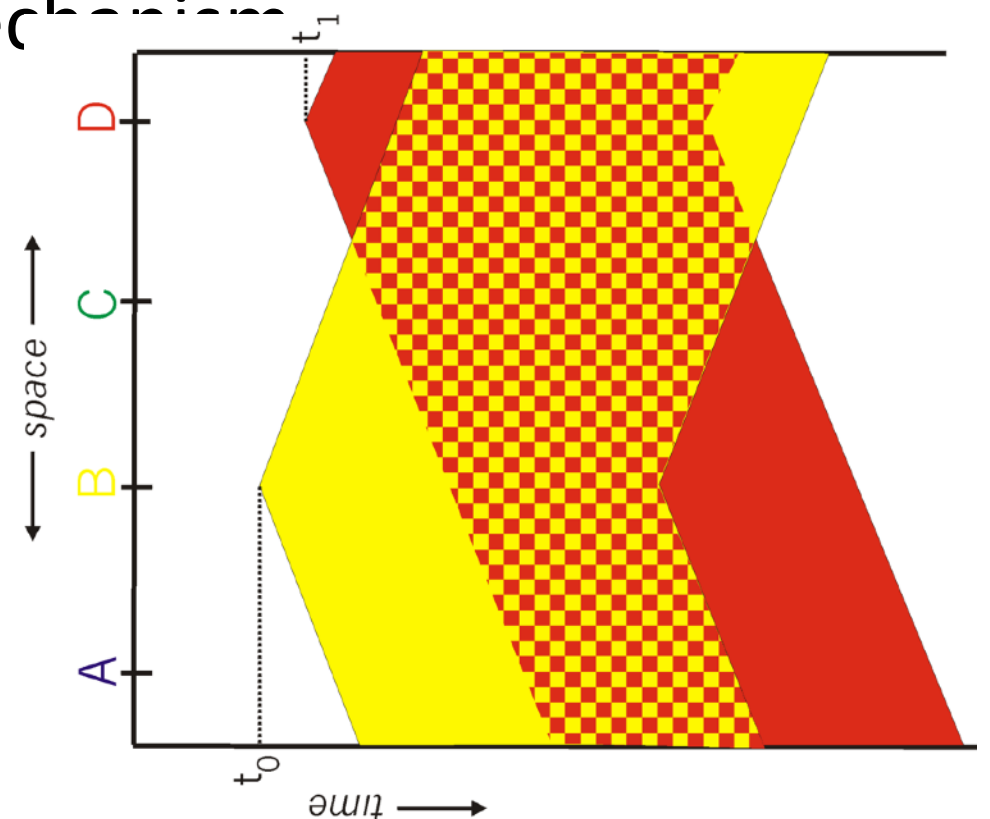
- **Context: a shared broadcast channel**
 - Must avoid/handle having multiple sources speaking at once
 - Otherwise collisions lead to garbled data
 - Need **distributed algorithm** for sharing channel
 - Algorithm determines **when** and **which** source can transmit
- **Three classes of techniques**
 - **Frequency-division multiple access**: coordinated sharing in space
 - **Time-division multiple access**: coordinated sharing in time
 - **Random access**: uncoordinated sharing
 - Detect collisions, and if needed, recover from collisions
 - **Carrier Sense Multiple Access (CSMA)**

Recap: CSMA (Carrier Sense Multiple Access)

- CSMA: **listen** before transmit
 - If channel sensed idle: transmit entire frame
 - If channel sensed busy: defer transmission
- Does this eliminate all collisions?
 - **No**, because of nonzero propagation delay

- Solution:

- Include a **Collision Detection (CD)** mechanism
- If a collision detected
 - Retransmit
 - **When to retransmit?**



Questions?

CSMA/CD

Once a collision is detected ...

- **When should the frame be retransmitted?**
- Immediately?
 - Every NIC would start sending immediately
 - Collision again!
- Take turns?
 - Back to time division multiplexing
 - Problem?
 - Underutilization

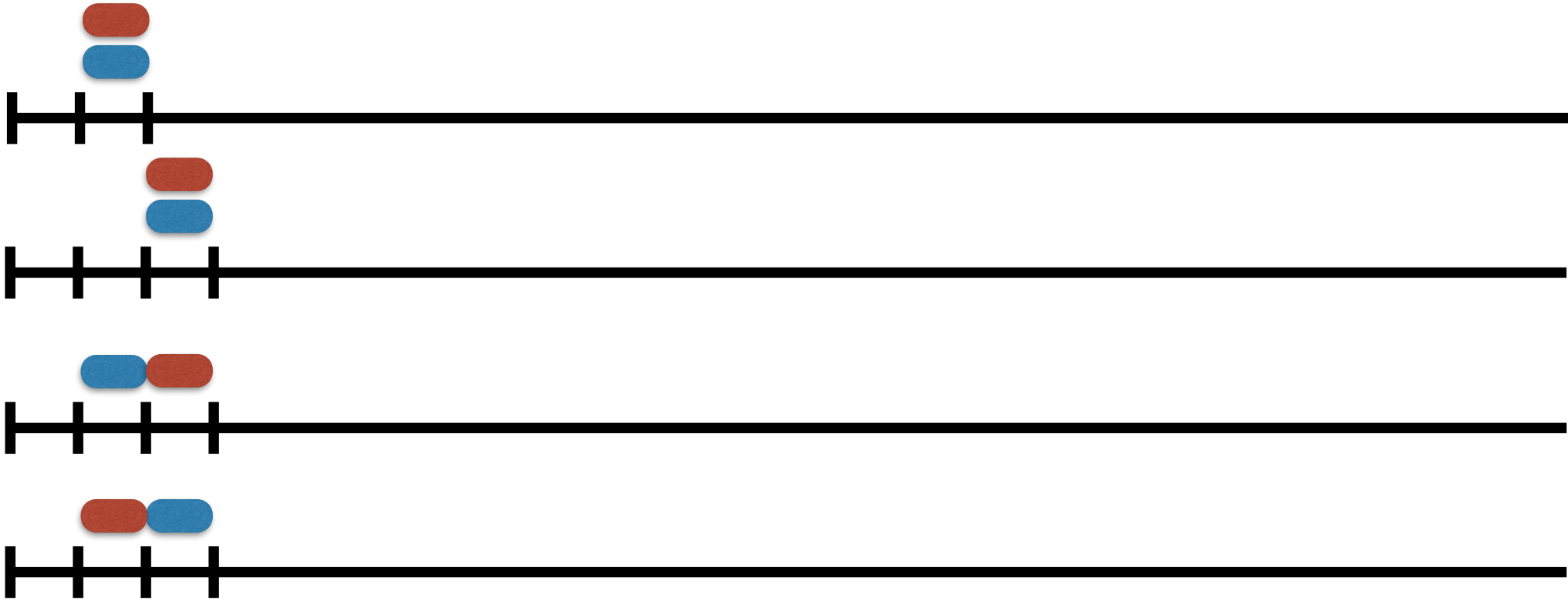
CSMA/CD in one slide!

- **Carrier Sense: continuously listen to the channel**
 - If idle: start transmitting
 - If busy: wait until idle
- **Collision Detection: listen while transmitting**
 - No collision: transmission complete
 - Collision: abort transmission
- **When to retransmit?: exponential back off**
 - After collision, transmit after “waiting time”
 - After k collisions, choose “waiting time” from $\{0, \dots, 2^k - 1\}$
 - Exponentially increasing waiting times
 - But also, exponentially larger success probability

Exponential Back-off: An example



Attempt 1: Suppose a collision happens



Attempt 2: Four possibilities

Success with Probability = 0.5

Questions?

Group Exercise:

What is the success probability in attempt 3?

Answer: 0.75

Why Frames?

(Layering: Link Layer on top of Physical Layer)

Building Link Layer on top of Physical Layer

- Physical layer sends/receives bits on a link, and forwards to link layer
- View at the destination side physical layer:

01010110011111101111101111100101000111

- Challenge: how to take the above bits and convert to:

01010110011111101111101111100101000111

- **Problem:** how does the link layer separate data into correct “chunks”?
 - Chunks belonging to different applications
- Data link layer **interfaces** with **physical layer** using **frames**
 - Implemented by the network adaptor
 - **Finally: What are these frames?**

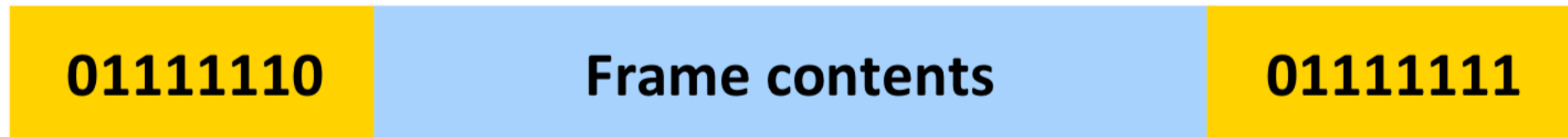


Frames



Identifying start/end of frames: Sentinel Bits

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



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

When Receiver sees five 1s...



- If next bit is 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/seven 1s)
- If next bit is 1 (i.e., we have six 1s) then:
 - If following bit is 0, this is the start of the frame
 - Because the receiver has seen 01111110
 - If following bit is 1, this is the end of the 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

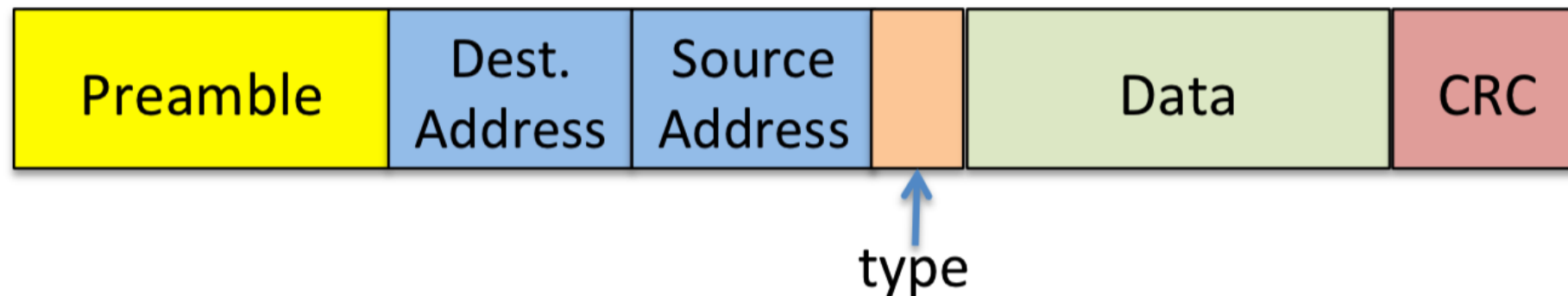
- After bit stuffing at the sender:

01111110011111010111110011111000101111111

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

01111110011111101111101111100101111111

Ethernet “Frames”



- **Preamble:**
 - 7 bytes for clock synchronization
 - 1 byte to indicate start of the frame
- **Names:** 6 + 6 bytes (MAC names/addresses)
- **Protocol type:** 2 bytes, indicating higher layer protocol (e.g., IP)
- **Data payload:** max 1500 bytes, minimum 46 bytes
- **CRC:** 4 bytes for error detection

What about source/destination Addresses?

- **Frames are at Layer-2**
 - Thus, use Layer-2 addresses (MAC names/addresses)
- **MAC address**
 - Numerical address associated with the network adapter
 - Flat namespace of 6 bytes (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 IEEE
 - First 24 bits (e.g., **00-15-C5-**-**-****)
 - **Adapter**: assigned by the vendor from its block
 - Last 24 bits

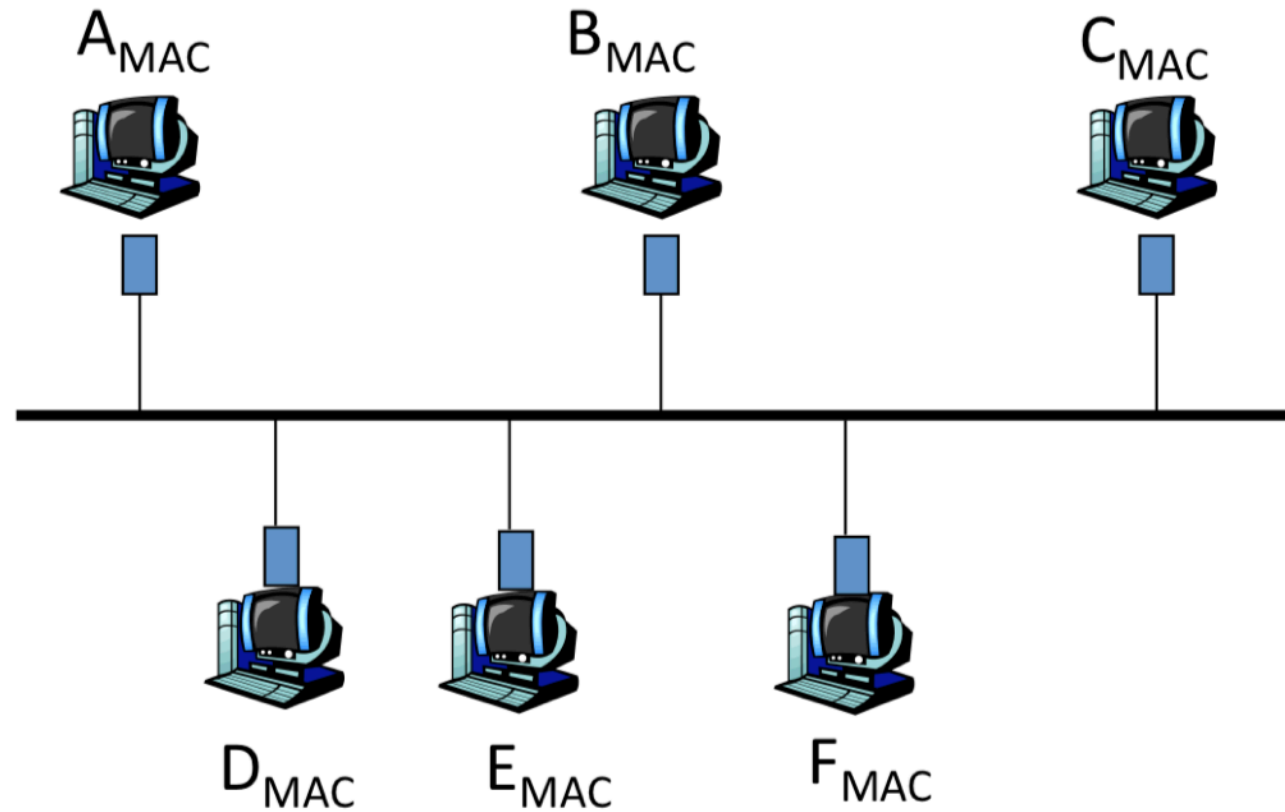
Questions?

**Putting it all together
(Traditional Ethernet)**

Traditional Ethernet

- **(Source) Link layer receives data from the network layer (more later)**
- **(Source) Link layer divides data into frames**
 - How does it know source/destination MAC names?
 - Source name is easy ... destination name is tricky (more later)
- **(Source) Link layer passes the frame to physical layer**
 - Frames up the frames (using sentinel bits)
 - **And broadcasts on the broadcast Ethernet**
- **(EACH) physical layer regenerates the frame...**
 - And sends it up to the (destination) link layer
 - Which sends the data to the network layer **If and only if:**
 - destination name matches the receiver's MAC name
 - Or, the destination name is the broadcast address (FF:FF:FF:FF:FF:FF)

Traditional Ethernet



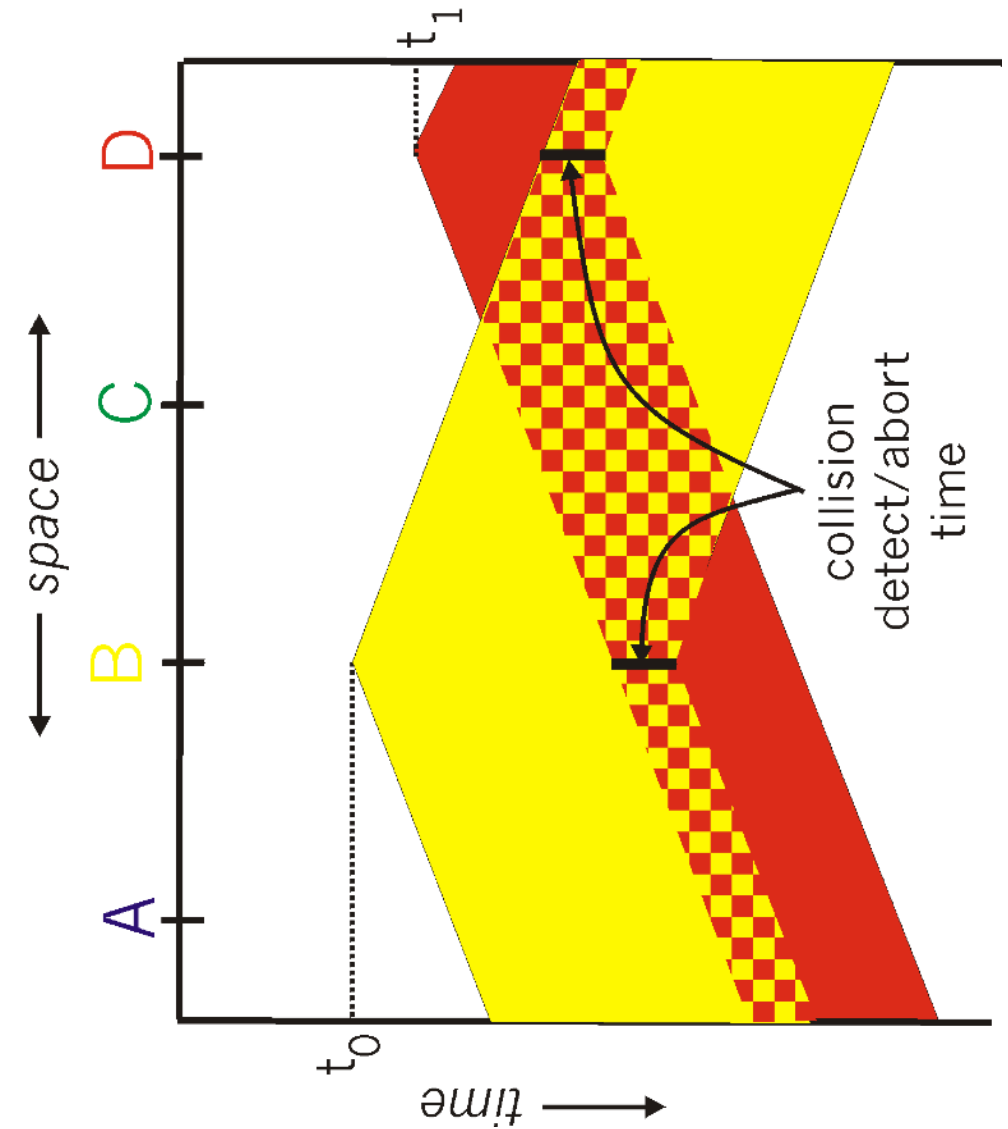
- Ethernet is “plug-n’play”
 - A new host plugs into the Ethernet is good to go
 - No configuration by users or network operators
 - Broadcast as a means of bootstrapping communication

Questions?

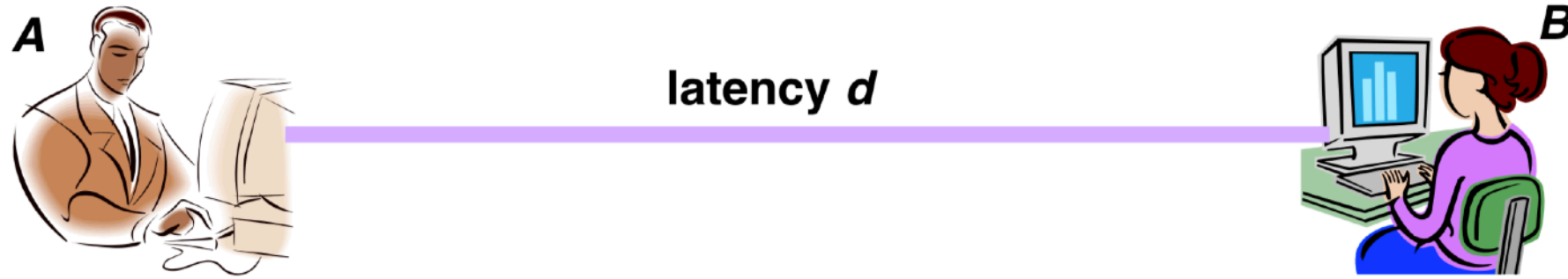
WHY Switched Ethernet?

Collision Detection limits Ethernet scalability

- **B** and **D** can tell that collision occurred
- However, need restrictions on
 - **Minimum frame size**
 - **Maximum distance**

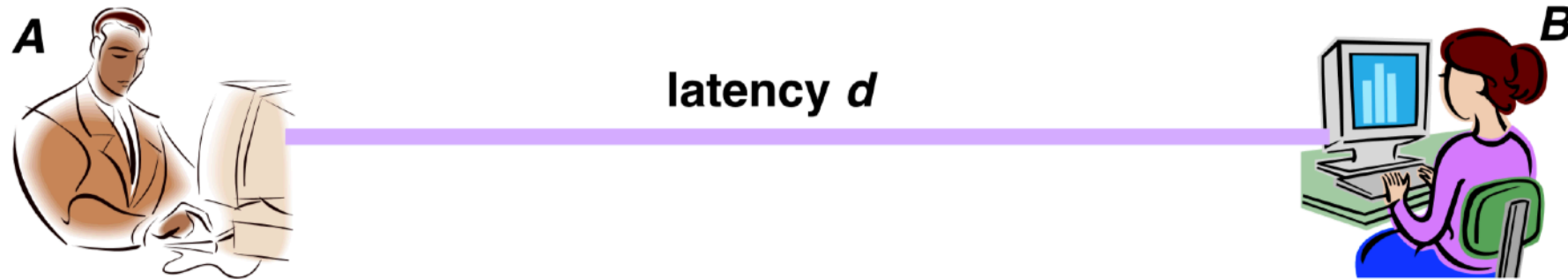


Limits on Traditional Ethernet Scalability



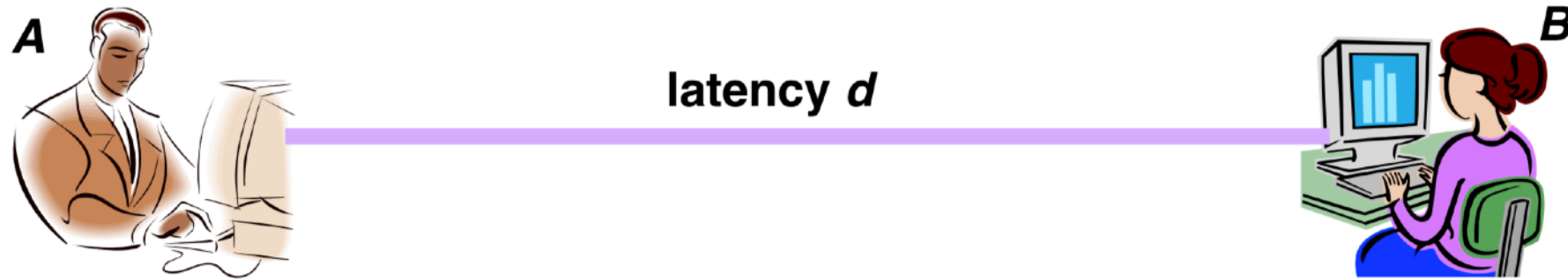
- **Latency depends on physical length of link**
 - Propagation delay
- **Suppose A sends a packet at time 0**
 - B sees an idle line at all times before d
 - ... so B happily starts transmitting a packet
- **B detects a collision at time d**
 - But A can't see collision until $2d$
 - A must have a frame size such that transmission time $> 2d$
 - Need **transmission time $> 2 * \text{propagation delay}$**

Limits on Traditional Ethernet Scalability



- **Transmission time $> 2 * \text{propagation delay}$**
- **Requires either very large frames (underutilization) or small scale.**
 - **Example: consider 100 Mbps Ethernet**
 - **Suppose** minimum frame length: 512 bits (64 bytes)
 - Transmission time = 5.12 μsec
 - Thus, propagation delay $< 2.56 \mu\text{sec}$
 - Length $< 2.56 \mu\text{sec} * \text{speed of light}$
 - Length $< 768\text{m}$
- **Cannot scale beyond $\sim 76.8\text{m}$ for 1Gbps and beyond $\sim 7.68\text{m}$ for 10Gbps**

Limits on Traditional Ethernet Scalability



- **Transmission time $> 2 * \text{propagation delay}$**
- **Cannot scale beyond $\sim 76.8\text{m}$ for 1Gbps and beyond $\sim 7.68\text{m}$ for 10Gbps**
- **This is WHY modern Ethernet networks are “switched”**

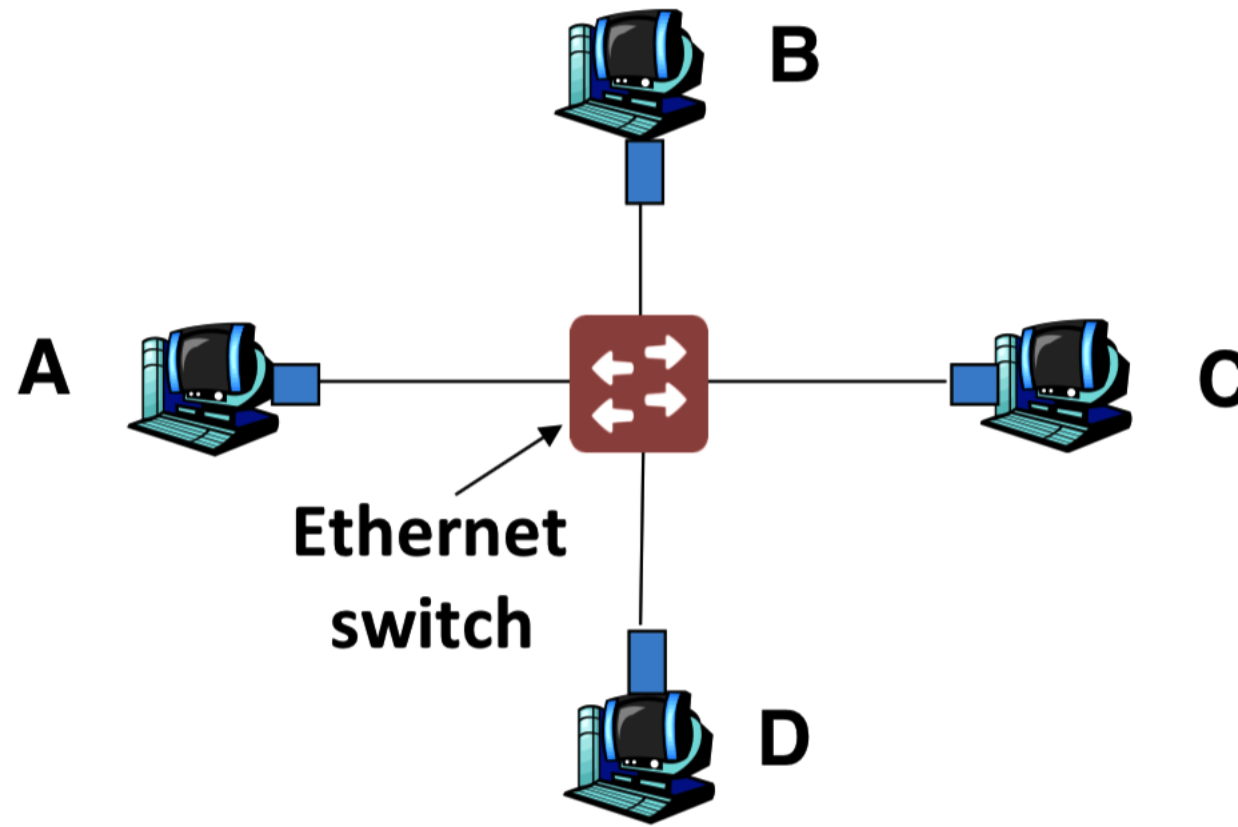
Evolution

- **Ethernet was invented as a broadcast technology**
 - Hosts share channel
 - Each packet received by all attached hosts
 - CSMA/CD for access control
- **Current Ethernets are “switched”**
 - Point-to-point medium between switches;
 - Point-to-point medium between each host and switch
 - Sharing only when needed (using CSMA/CD)

Questions?

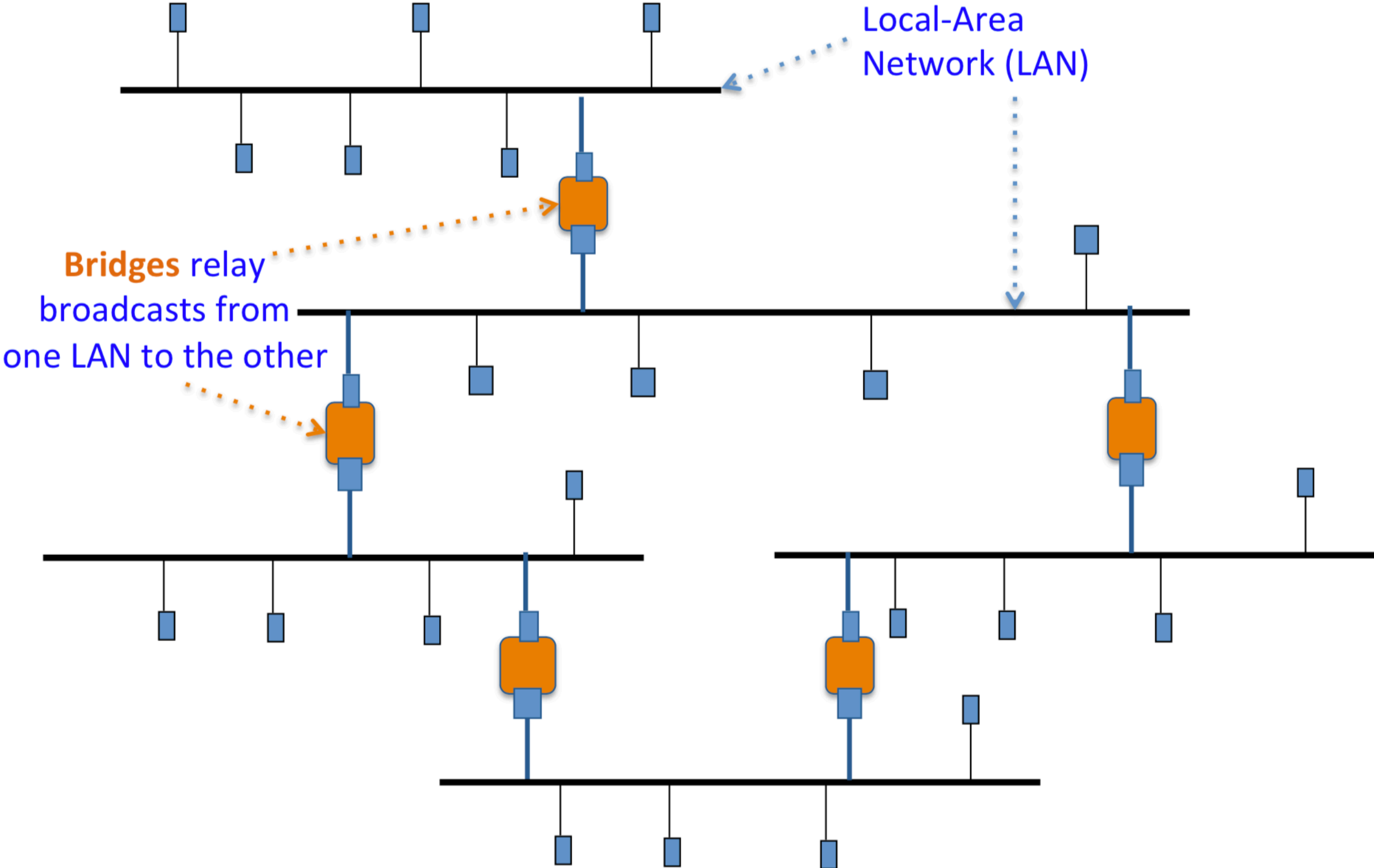
Switched Ethernet

Switched Ethernet

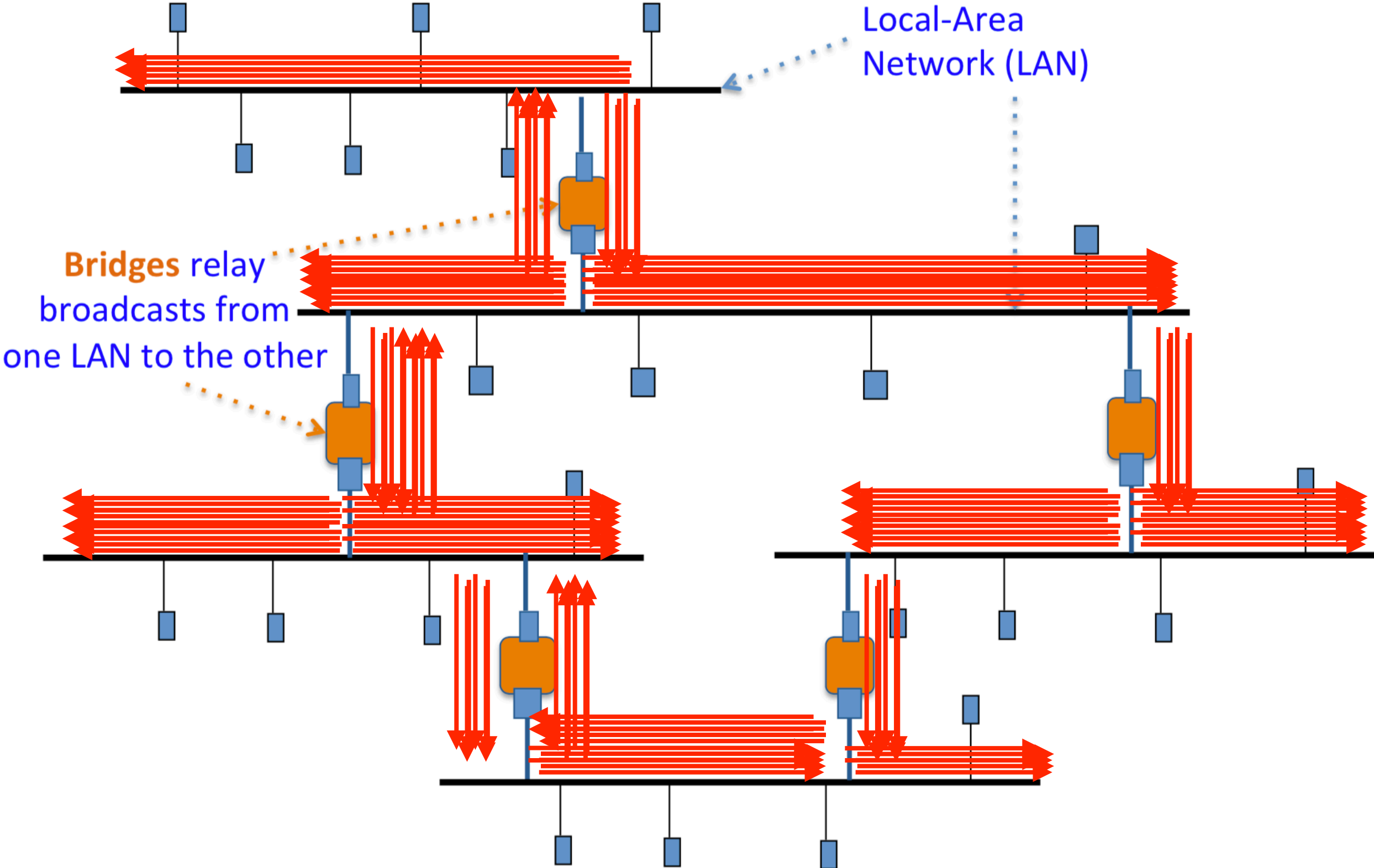


- Enables concurrent communication
 - Host A can talk to C, while B talks to D
 - No collisions -> no need for CSMA, CD
 - No constraints on link lengths or frame size

Routing in Switched Ethernet (Extended LANs)



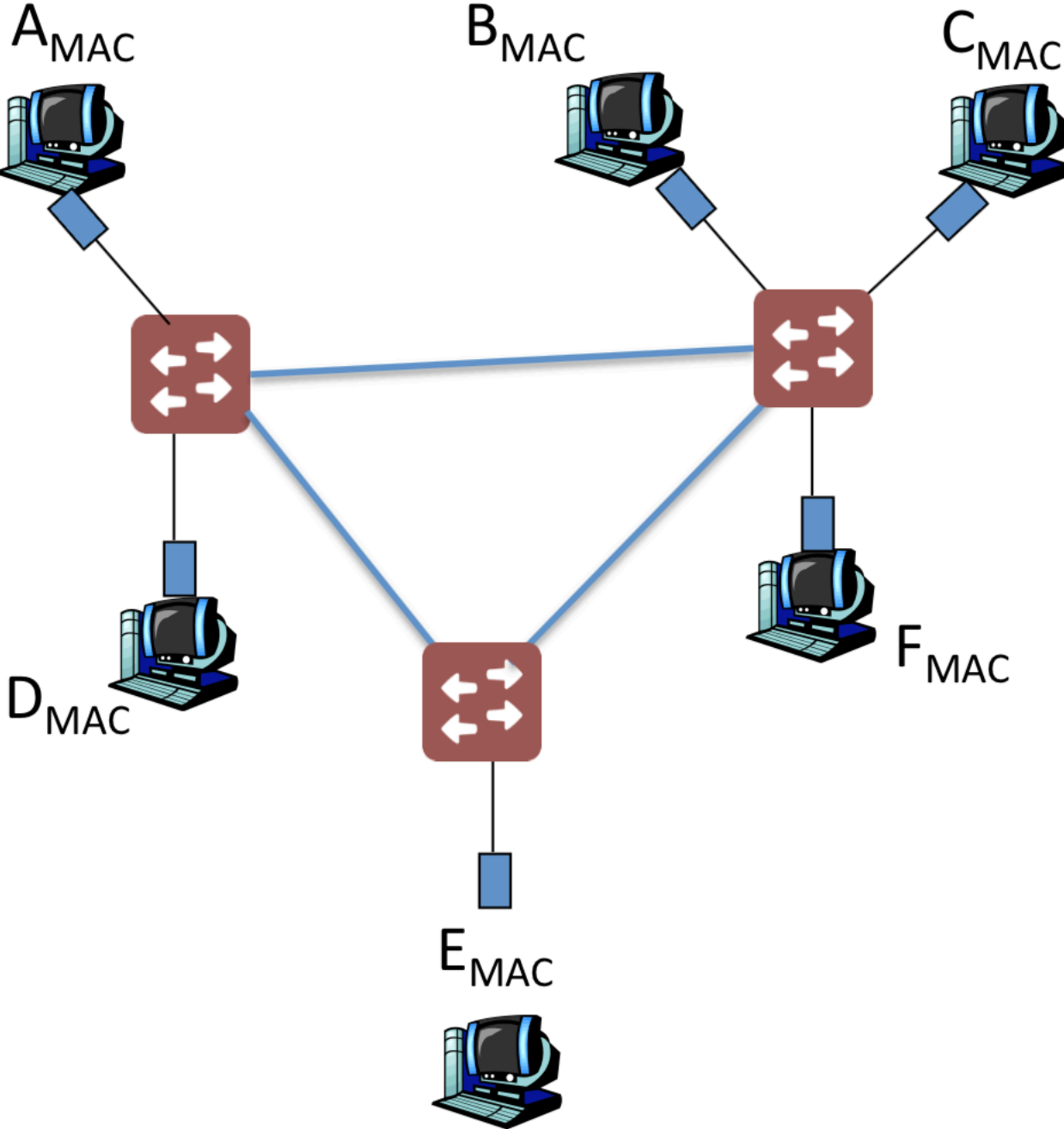
Naïvely Routing in "Extended LANs": Broadcast storm



How to avoid the Broadcast Storm Problem?

Get rid of the loops!

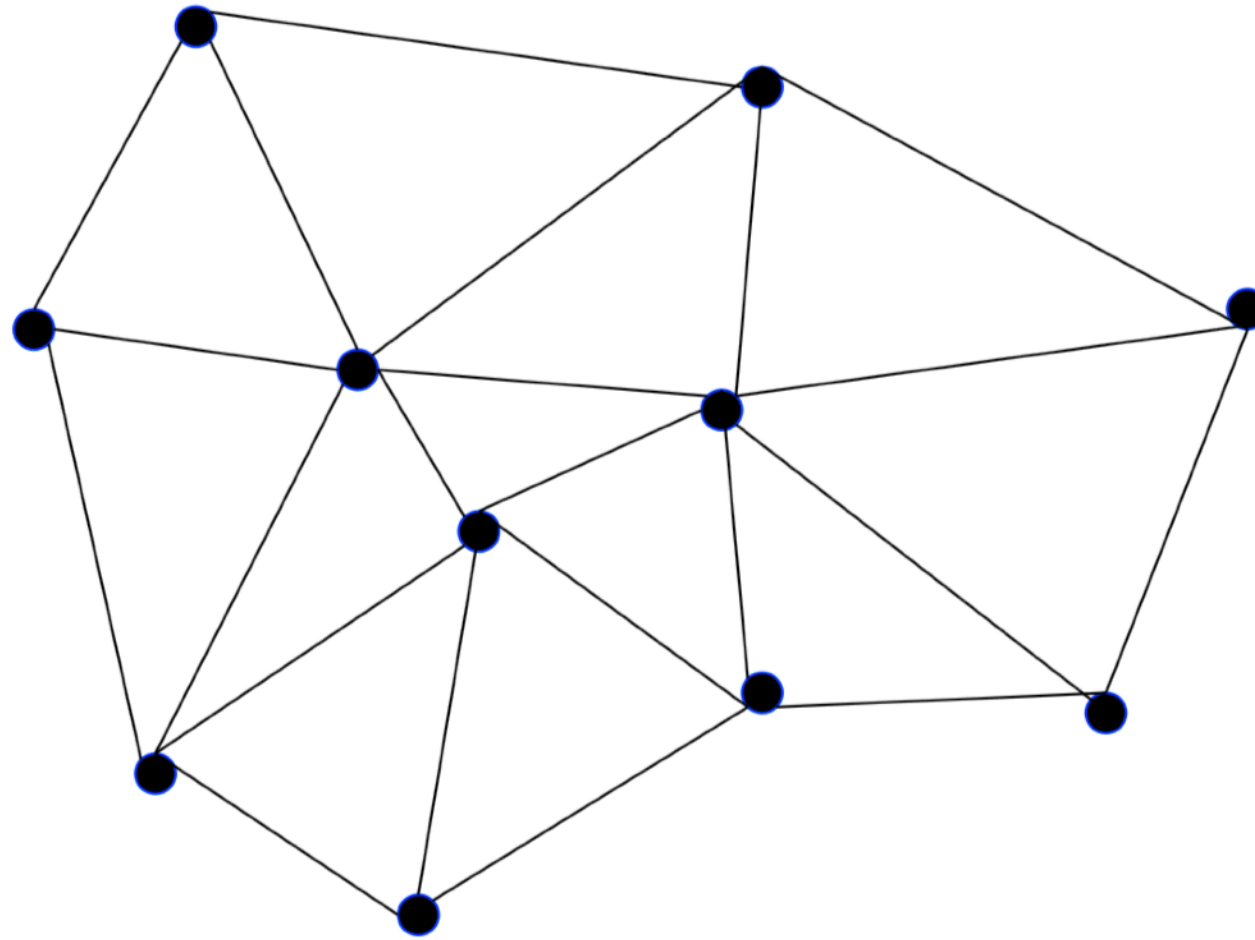
Lets get back to the graph representation!



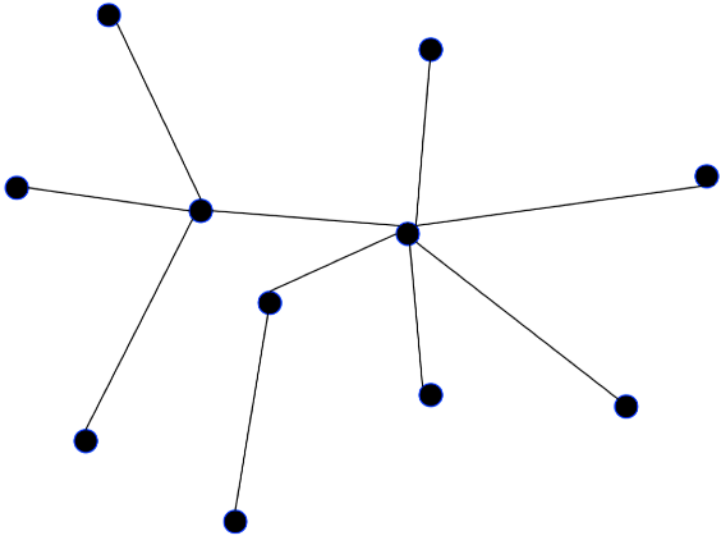
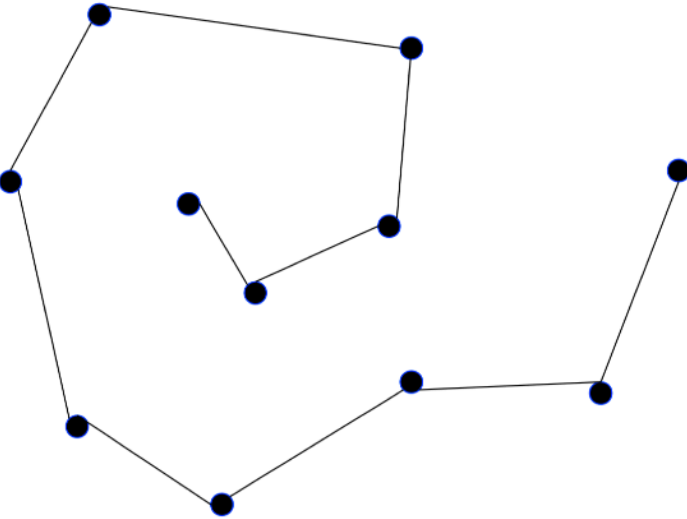
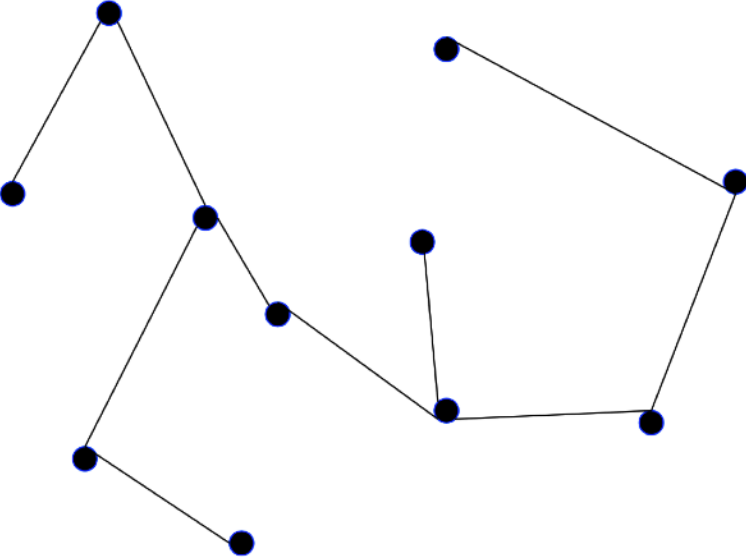
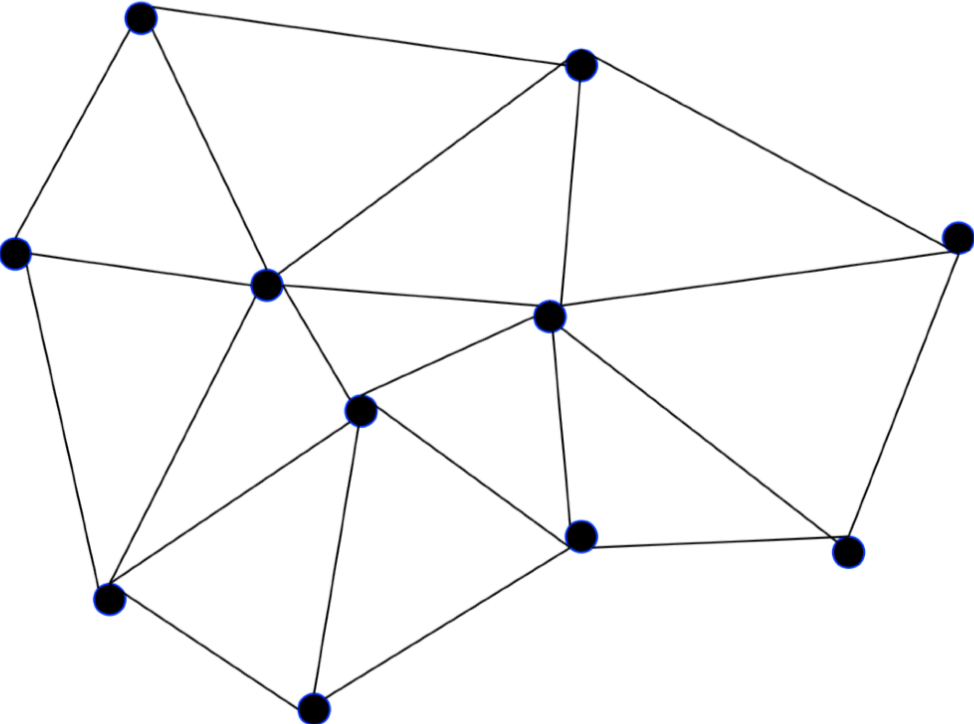
Easiest Way to Avoid Loops

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

Consider Graph



Multiple Spanning Trees



Questions?

Spanning Tree Protocol

- Protocol by which bridges construct a spanning tree
- Nice properties
 - Zero configuration (by operators or users)
 - Self healing
- Still used today
- Constraints for backwards 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 for a switched topology?

Spanning Tree Approach

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

Group Exercise:

Design a Spanning Tree Protocol

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 the shortest paths go
 - Pick the one with the smallest identifier (MAC name/address)
- Compute the shortest paths to the root
 - No shortest path can have a cycle
 - Only keep the links on the shortest path
 - 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 tie breaking system**
 - This is just an easy one to remember and implement

Constructing a Spanning Tree

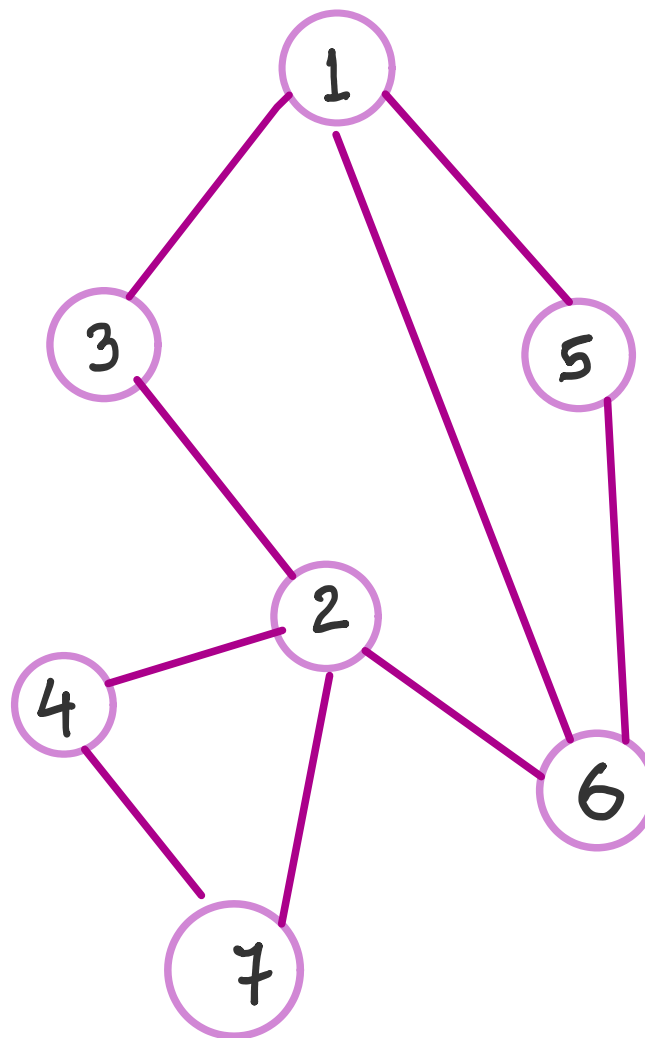
- Messages (Y,d,X)
 - From node X
 - Proposing Y as the root
 - And advertising a distance d to Y
- Switches elect the node with smallest identifier (MAC name) as root
 - Y in messages
- Each switch determines if a link is on its shortest path to the root
 - If not, excludes it from the tree
 - d to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

Steps in Spanning Tree Protocol

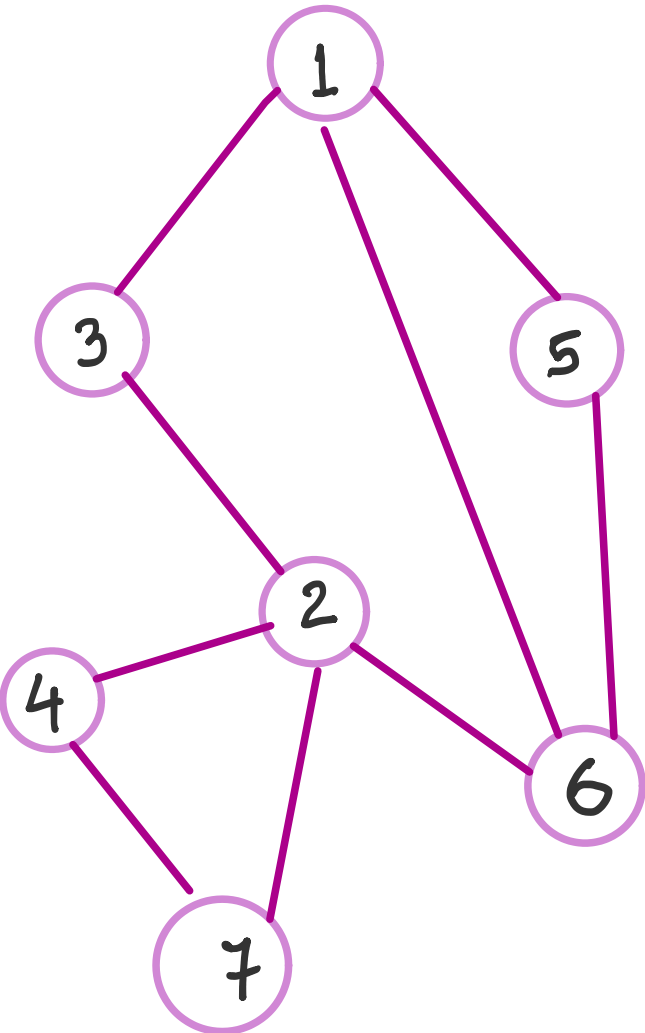
- Messages (Y,d,X)
 - For root Y ; From node X ; advertising a distance d to Y
- Initially each switch proposes itself as the root
 - that is, switch X announces $(X,0,X)$ to its neighbors
- Switches update their view
 - Upon receiving message (Y,d,Y) from Z , check Y 's id
 - If Y 's id $<$ current root: set root = Y
- Switches compute their distance from the root
 - Add 1 to the shortest distance received from a neighbor
- If root **changed** OR shortest distance to the root **changed**, send all neighbors updated message $(Y,d+1,X)$

Group Exercise:

Lets run the Spanning Tree Protocol on this example

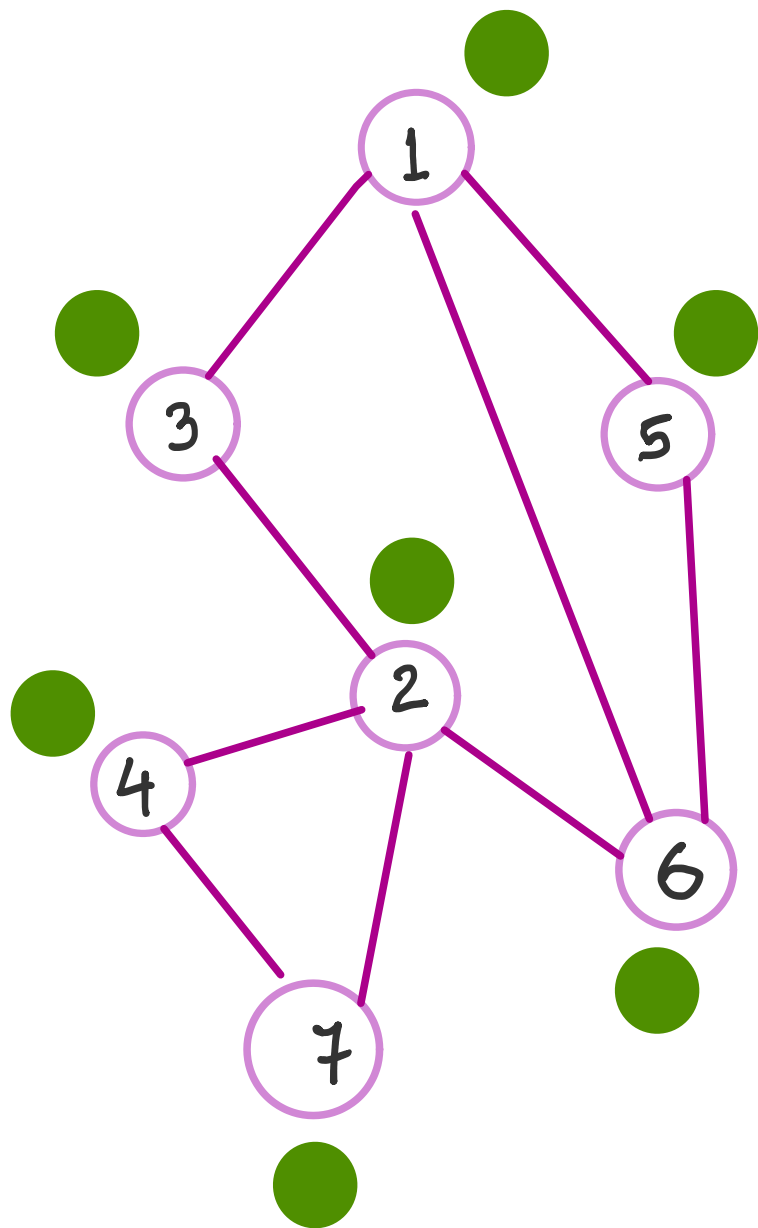


Round 1



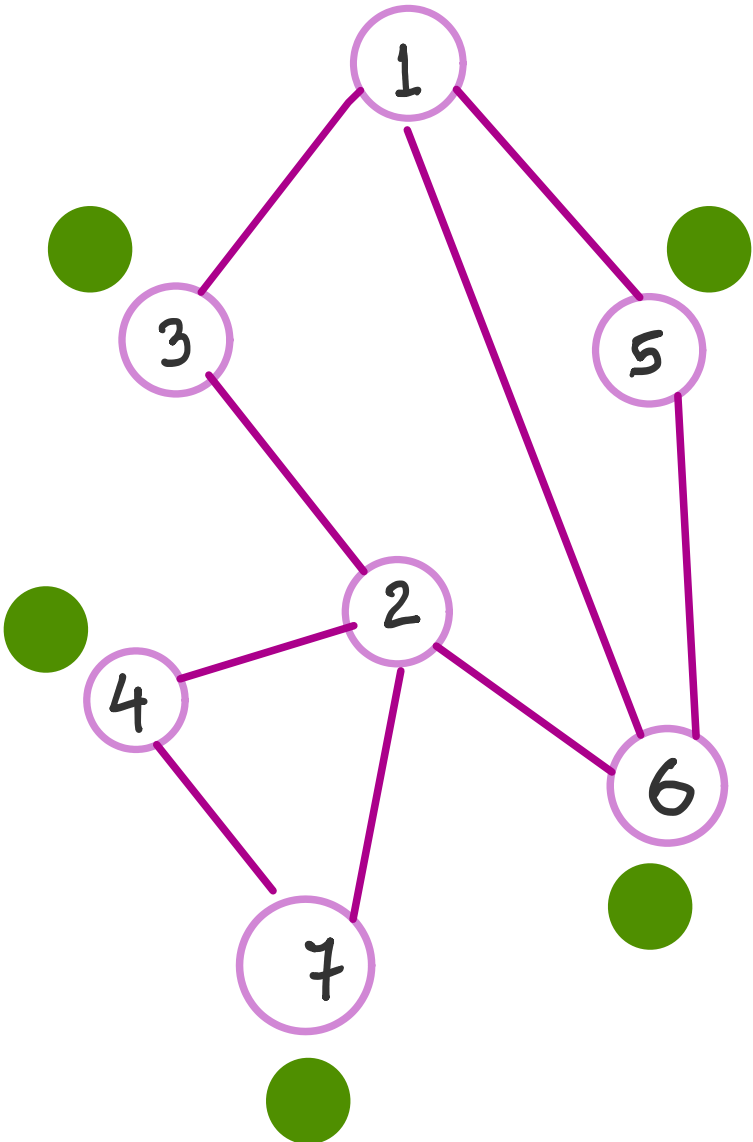
	Receive	Send
1		(1, 0, 1)
2		(2, 0, 2)
3		(3, 0, 3)
4		(4, 0, 4)
5		(5, 0, 5)
6		(6, 0, 6)
7		(7, 0, 7)

Round 2



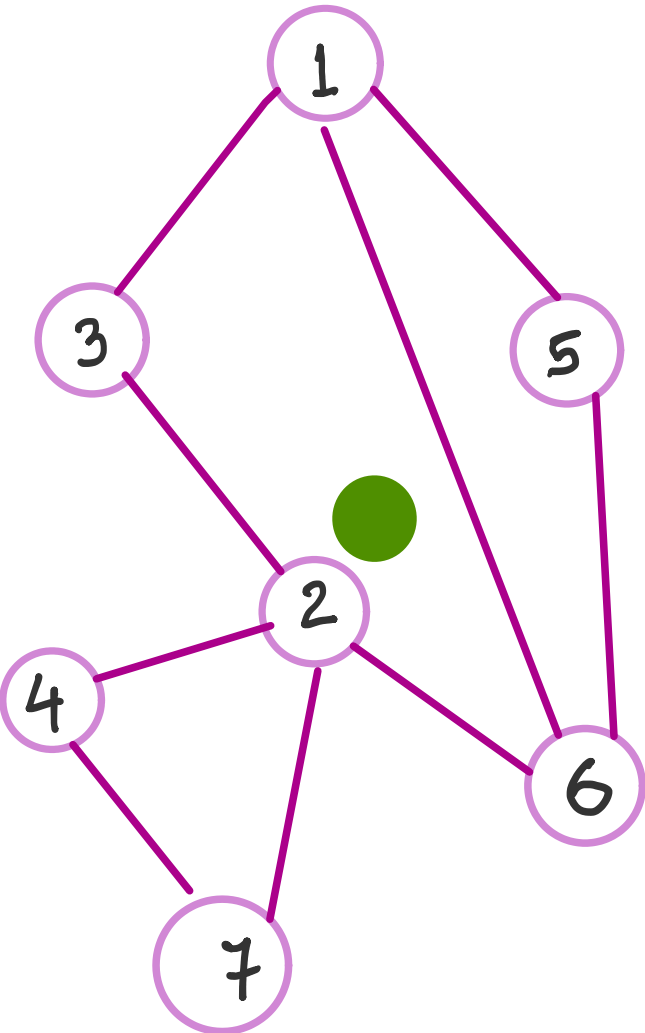
	Receive	Send
1 (1, 0, 1)	(3, 0, 3), (5, 0, 5), (6, 0, 6)	
2 (2, 0, 2)	(3, 0, 3), (4, 0, 4), (6, 0, 6), (7, 0, 7)	
3 (3, 0, 3)	(1, 0, 1), (2, 0, 2)	(1, 1, 3)
4 (4, 0, 4)	(2, 0, 2), (7, 0, 7)	(2, 1, 4)
5 (5, 0, 5)	(1, 0, 1), (6, 0, 6)	(1, 1, 5)
6 (6, 0, 6)	(1, 0, 1), (2, 0, 2), (5, 0, 5)	(1, 1, 6)
7 (7, 0, 7)	(2, 0, 2), (4, 0, 4)	(2, 1, 7)

Round 3



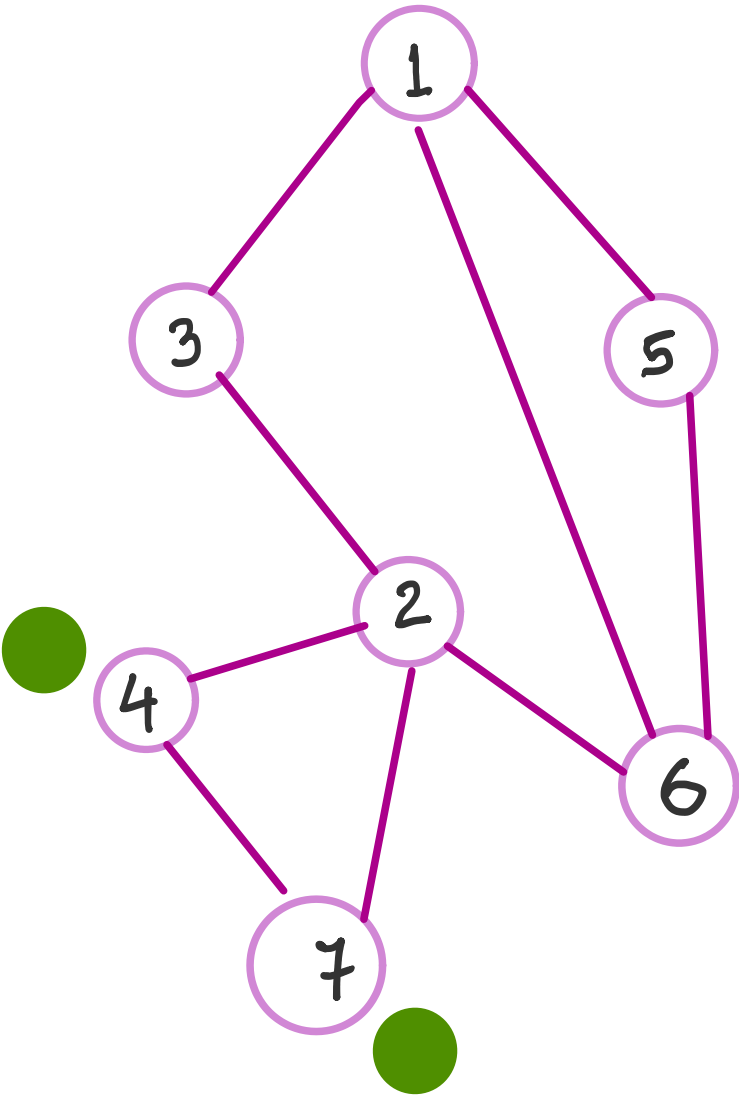
	Receive	Send
1 (1, 0, 1)	(1, 1, 3), (1, 1, 5), (1, 1, 6)	
2 (2, 0, 2)	(1, 1, 3), (2, 1, 4), (1, 1, 6), (2, 1, 7)	(1, 2, 2)
3 (1, 1, 3)		
4 (2, 1, 4)	(2, 1, 7)	
5 (1, 1, 5)	(1, 1, 6)	
6 (1, 1, 6)	(1, 1, 5)	
7 (2, 1, 7)	(2, 1, 4)	

Round 4



	Receive	Send
1 (1, 0, 1)		
2 (1, 2, 2)		
3 (1, 1, 3)	(1, 2, 2)	
4 (2, 1, 4)	(1, 2, 2)	(1, 3, 4)
5 (1, 1, 5)		
6 (1, 1, 6)	(1, 2, 2)	
7 (2, 1, 7)	(1, 2, 2)	(1, 3, 7)

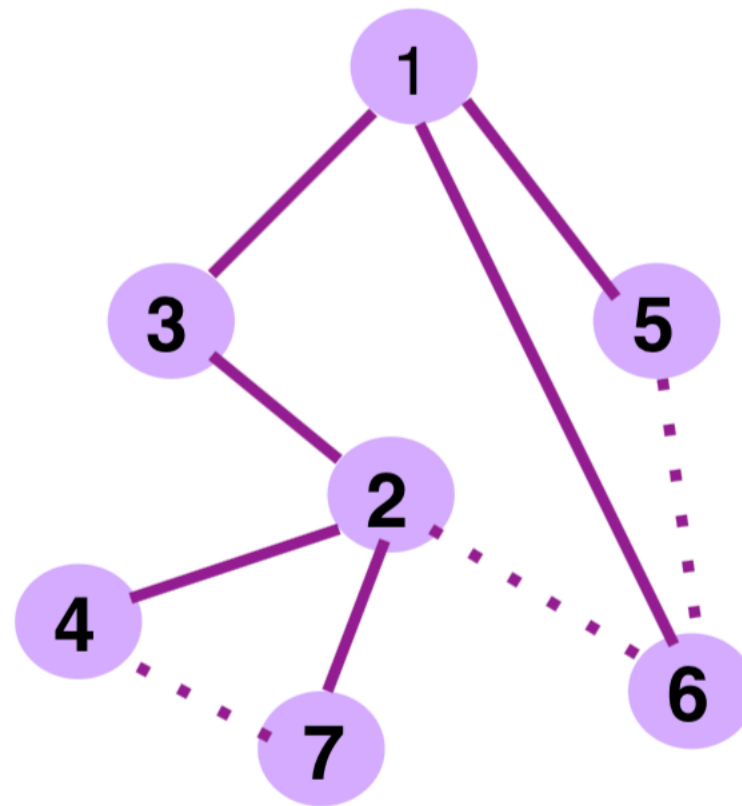
Round 5



	Receive	Send
1 (1, 0, 1)		
2 (1, 2, 2)	(1, 3, 4), (1, 3, 7)	
3 (1, 1, 3)		
4 (1, 3, 4)	(1, 3, 7)	
5 (1, 1, 5)		
6 (1, 1, 6)		
7 (1, 3, 7)	(1, 3, 4)	

After Round 5: We have our Spanning Tree

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



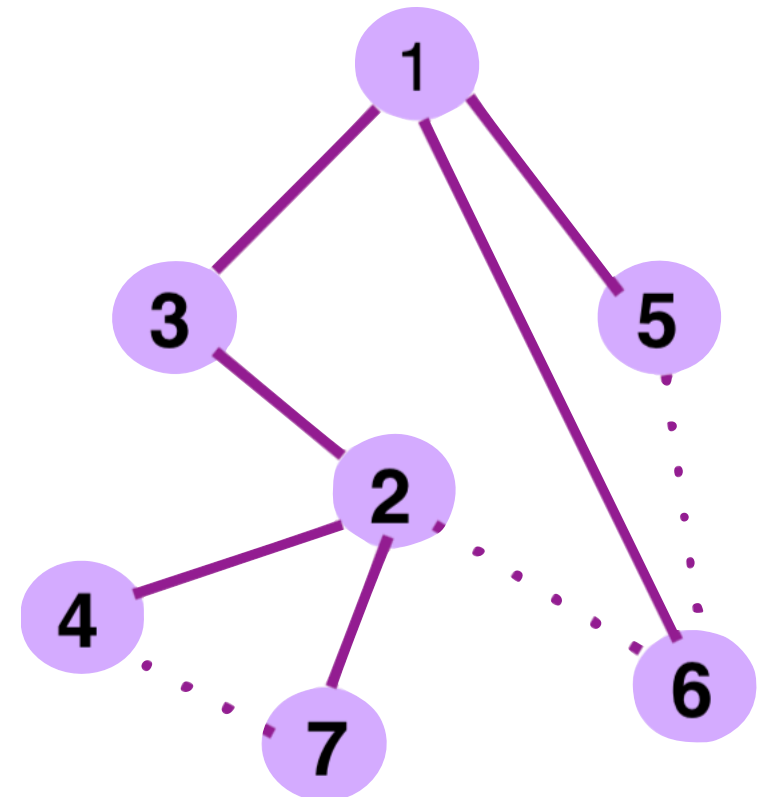
Questions?

Spanning Tree Protocol ++ (incorporating failures)

- Protocol must react to **failures**
 - Failure of the root node
 - Failure of switches and links
- **Root node sends periodic announcement messages**
 - Few possible implementations, but this is simple to understand
 - Other switches continue forwarding messages
- Detecting failures through timeout (**soft state**)
 - If no word from root, time out and send a $(Y, 0, Y)$ message to all neighbors (in the graph)!
- **If multiple messages with a new root received, send message (Y, d, X) to the neighbor sending the message**

Suppose link 2-4 fails

- 4 will send $(4, 0, 4)$ to all its neighbors
 - 4 will stop receiving announcement messages from the root
 - Why?
- At some point, 7 will respond with $(1, 3, 7)$
- 4 will now update to $(1, 4, 4)$ and send update message
- New spanning tree!



Questions?

The end of Link Layer

And the beginning of network layer :-D

