CS4450

Computer Networks:
Architecture and Protocols

Lecture 7
Switched Ethernet

Spring 2018
Rachit Agarwal

Announcements

- We will have a “live” coding class on Thursday next week (02/22)
    - **Please bring your laptops**
    - We will learn how to implement sockets, etc.

- Problem Set 1 Solutions are out.

Goals for Today's Lecture

- **Wrap up Link layer**
- Finish CSMA/CD
- Understand how to build Link Layer on top of Physical Layer
- Switched Ethernet
- Spanning Tree Protocol

Recap: Link Layer — originally a broadcast channel

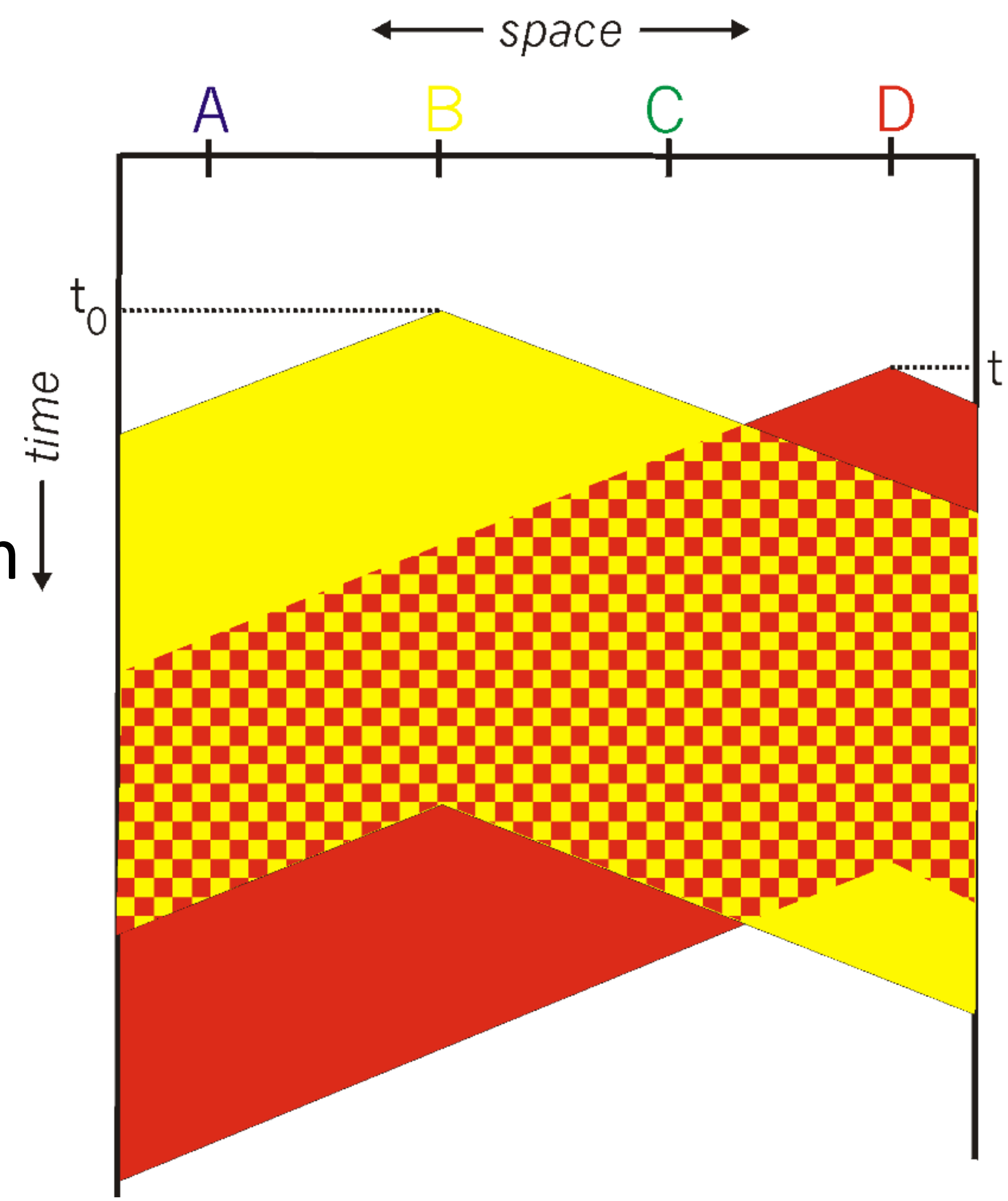
- **Sharing a broadcast channel**
  - Must avoid having multiple nodes speaking at once
    - Otherwise collisions lead to garbled data
  - Need distributed algorithm for sharing channel
    - Algorithm determines which node can transmit

- **Three classes of techniques**
  - **Frequency division multiplexing**:
    - Share frequencies (e.g., radio)
  - **Time division multiplexing**:
    - Take turns in transmitting (slotted ALOHA)
  - **Random access**:
    - Allow collisions, and then recover
    - **Carrier Sense Multiple Access (CSMA)**

CSMA/CD

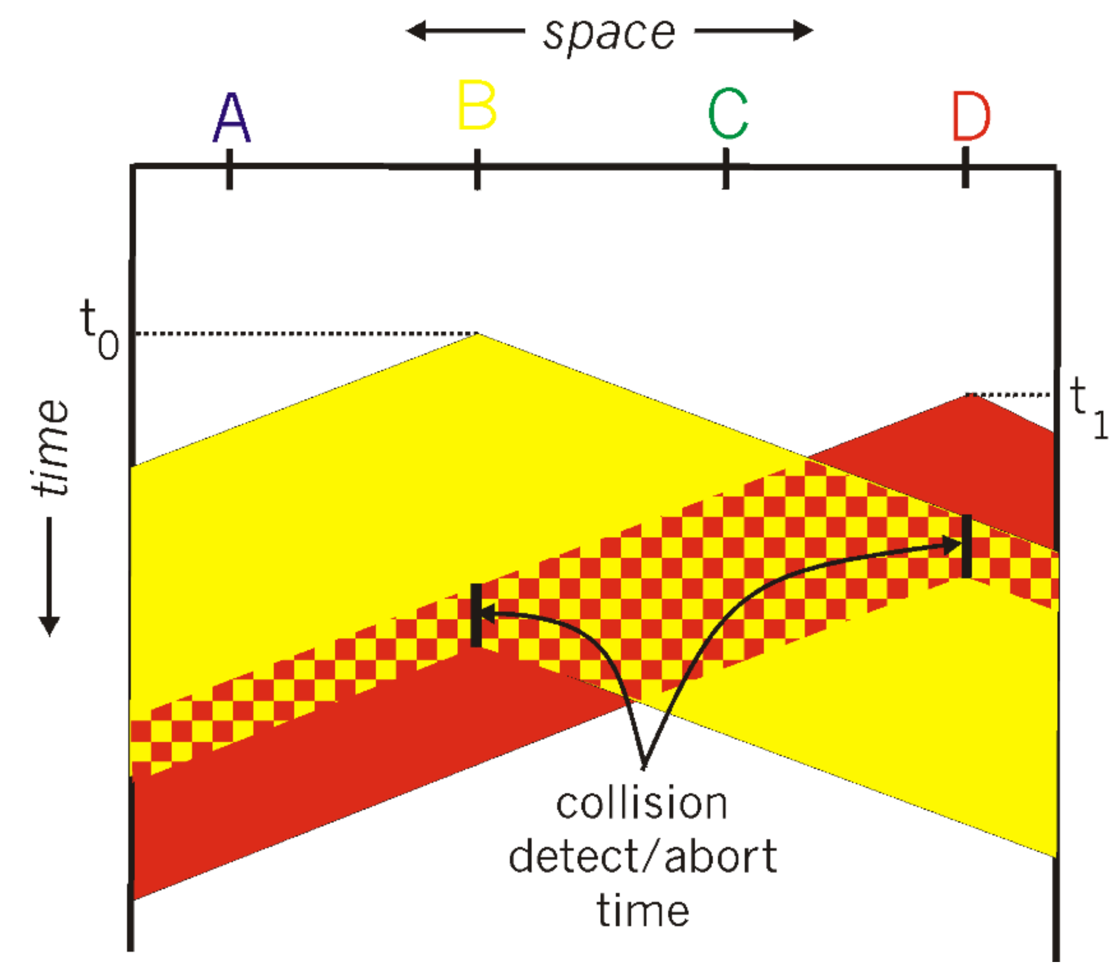
CSMA (Carrier Sense Multiple Access)

- CSMA: **listen** before transmit
  - If channel sensed idle: transmit entire frame
  - If channel sensed busy: defer transmission
- Does not necessarily eliminate collisions
  - Due to nonzero propagation delay
- Solution:
  - Include a **Collision Detection (CD)** m
  - If a collision detected
    - Retransmit

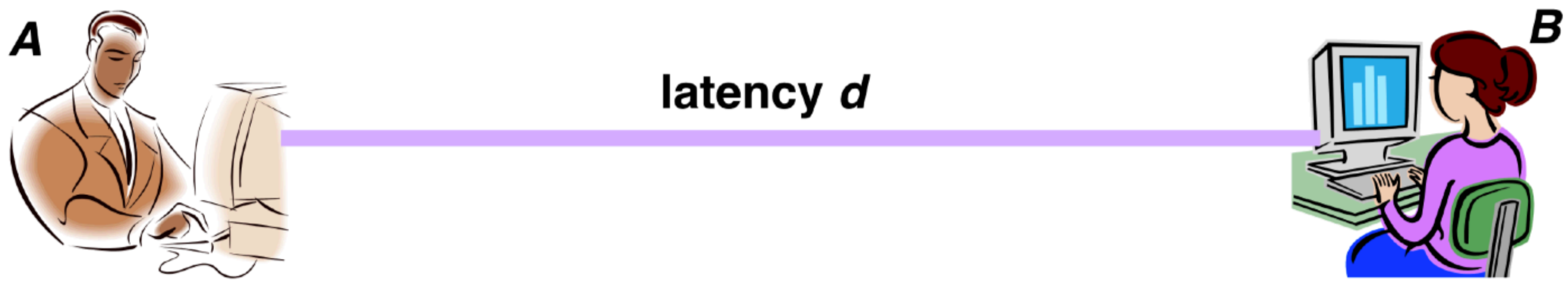


CSMA/CD (Collision Detection)

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



Limits on CSMA/CD Network Length

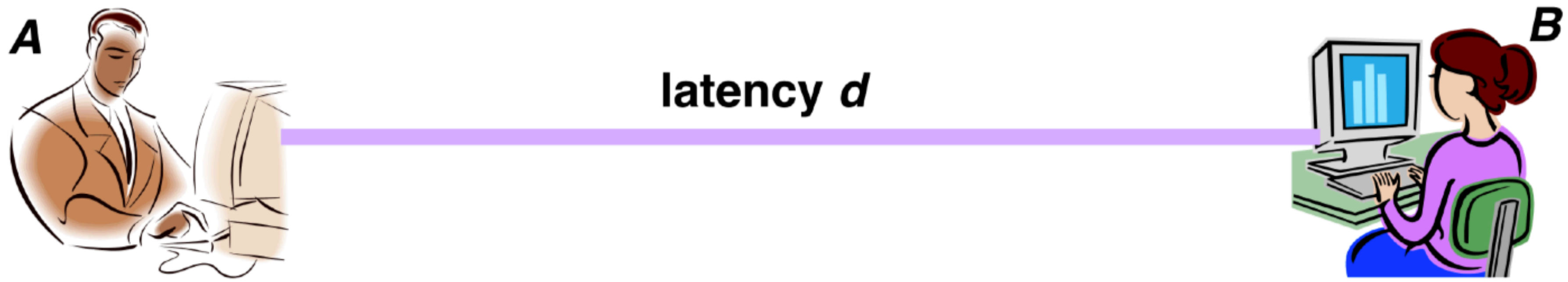


- **Latency depends on physical length of link**
  - Propagation delay

- **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, and sends jamming signal**
  - But A can’t see collision until **2d**
  - **A must have a frame size such that transmission time > 2d**
  - **Need transmission time > 2 * propagation delay**

Limits on CSMA/CD Network Length and Frame Size



- **Transmission time > 2 * propagation delay**
- **Imposes restrictions.**
  - **Example: consider 100 Mbps Ethernet**
  - **Suppose** minimum frame length: 512 bits (64 bytes)
    - Transmission time = 5.12 µsec
    - Thus, propagation delay < 2.56 µsec
    - Length < 2.56 µsec * speed of light
    - Length < 768m
- **What about 10Gbps Ethernet?**

Once a collision is detected …

- **When should the frame be resent?**

- Immediately?
    - Every NIC would start sending immediately
    - Collision again!

- Take turns?
    - Back to time division multiplexing

CSMA/CD in one slide!

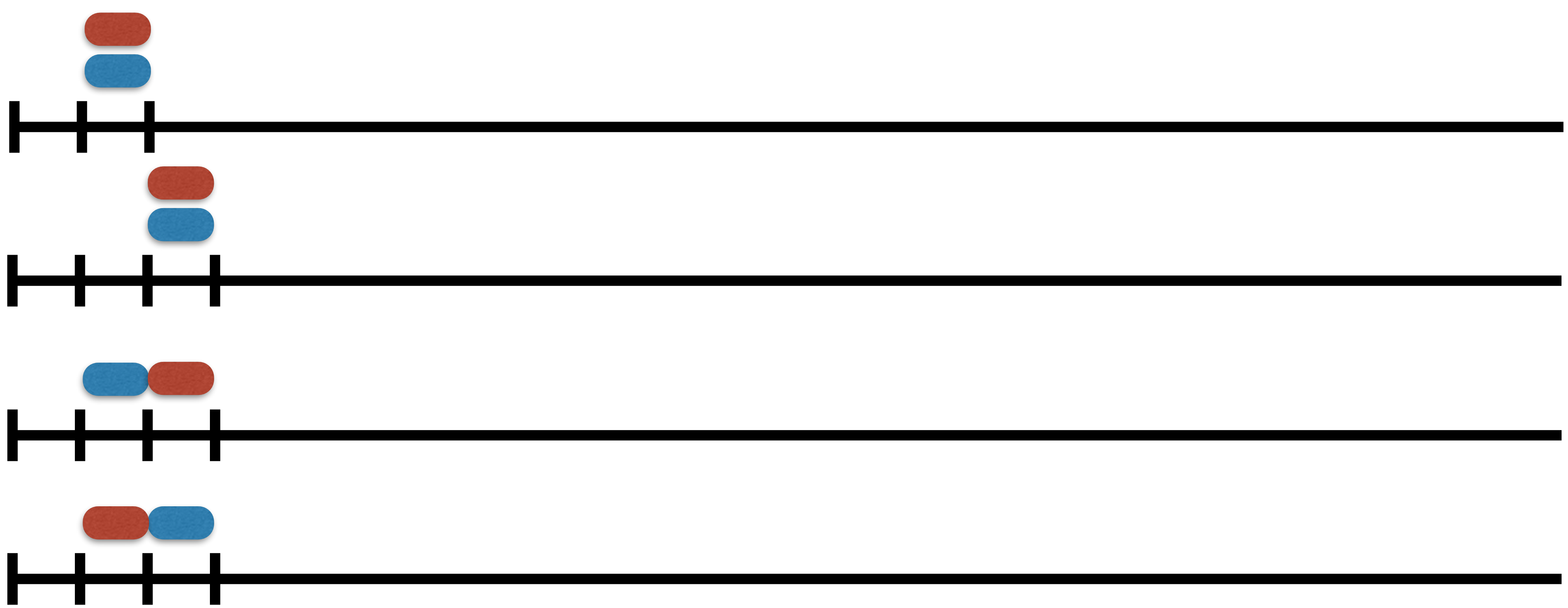
- **Carrier Sense: continuously listen to the channel**
  - If idle: start transmitting
  - If busy: wait until idle

- **Collision Detection: listen while transmitting**
  - No collision: transmission complete
  - Collision: abort transmission; send jam signal

- **Random access: exponential back off**
  - After collision, transmit after "waiting time"
  - After k collisions, choose "waiting time" from {0, …, 2^k-1)
  - Exponentially increasing waiting times
  - But also, exponentially larger success probability

CSMA/CD (Collision Detection): An example

Attempt 1: Suppose a collision happens



Attempt 2: Four possibilities

Success with Probability = 0.5

Group Exercise:

What is the success probability in attempt 3?

Answer: **0.75**

Performance of CSMA/CD: Why frames?

- **Time spent transmitting a packet (collision)**
  - Proportional to distance d; why?
- **Time spent transmitting a packet (no collision)**
  - Frame length p divided by bandwidth b
- **Rough estimate for efficiency (K some constant)**

$$E \sim \frac{\frac{p}{b}}{\frac{p}{b} + Kd}$$

- Observations:
  - For large packets, small distances, E ~ 1
  - **Right frame length depends on b, K, d: can't just use packets**
  - **As bandwidth increases, E decreases**
    - That is why high-speed LANs are switched

Link Layer on top of Physical Layer

Framing Frames

- Physical layer puts bits on a link

- But, data link layers at two hosts need to be able to exchange **frames**
  - Kind of an “interface” between **link layer and physical layer**
  - Implemented by the network adaptor

- **Framing problem**:
  - how does the link layer know **where each frame begins and ends?**

But first things first: We need source/destination 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

Back to 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 **1**s in the frame content
    - Receiver always removes a **0** appearing after five **1**s

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:

011111100111111011111101111100101111111

- Sender rule: five 1s -> insert a 0

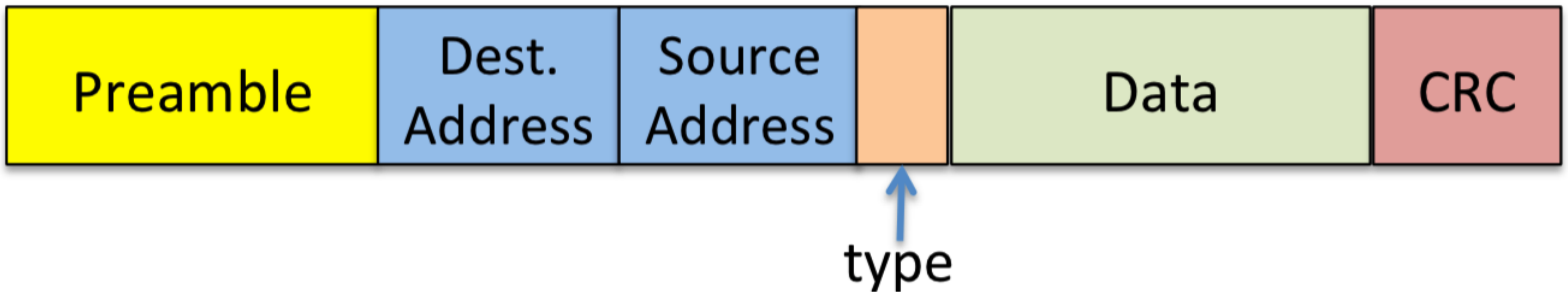
- After bit stuffing at the sender:

011111100111110101111100111110000101111111

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

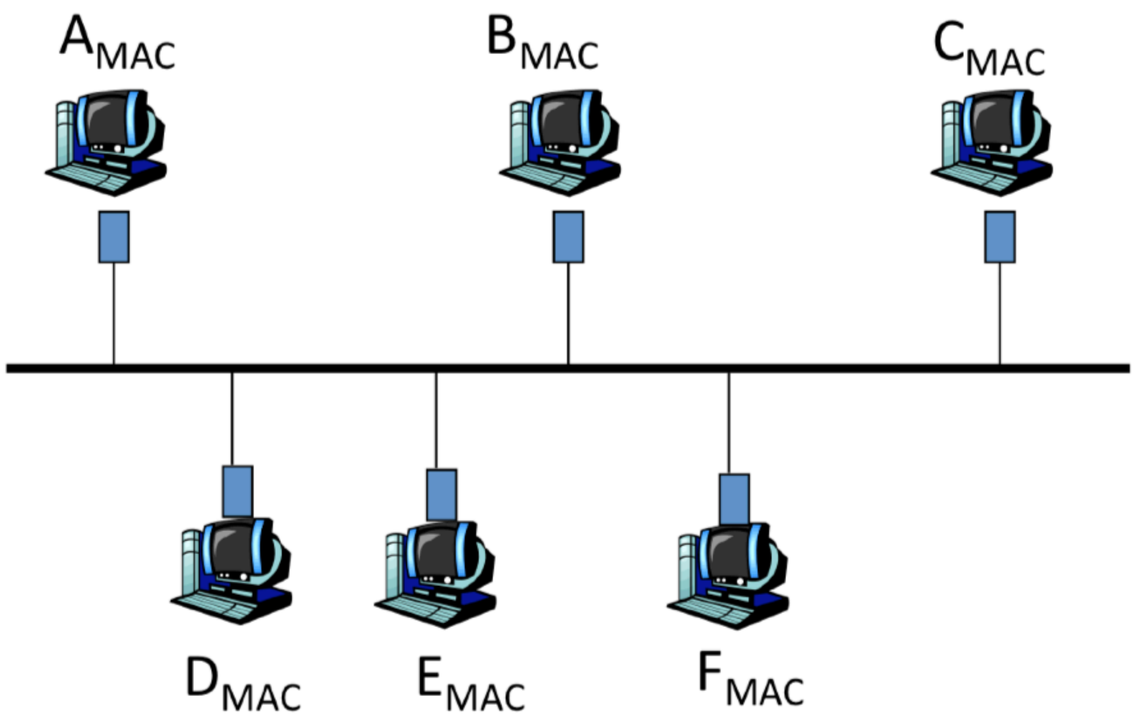
011111100111111011111101111100101111111

Ethernet “Frames”



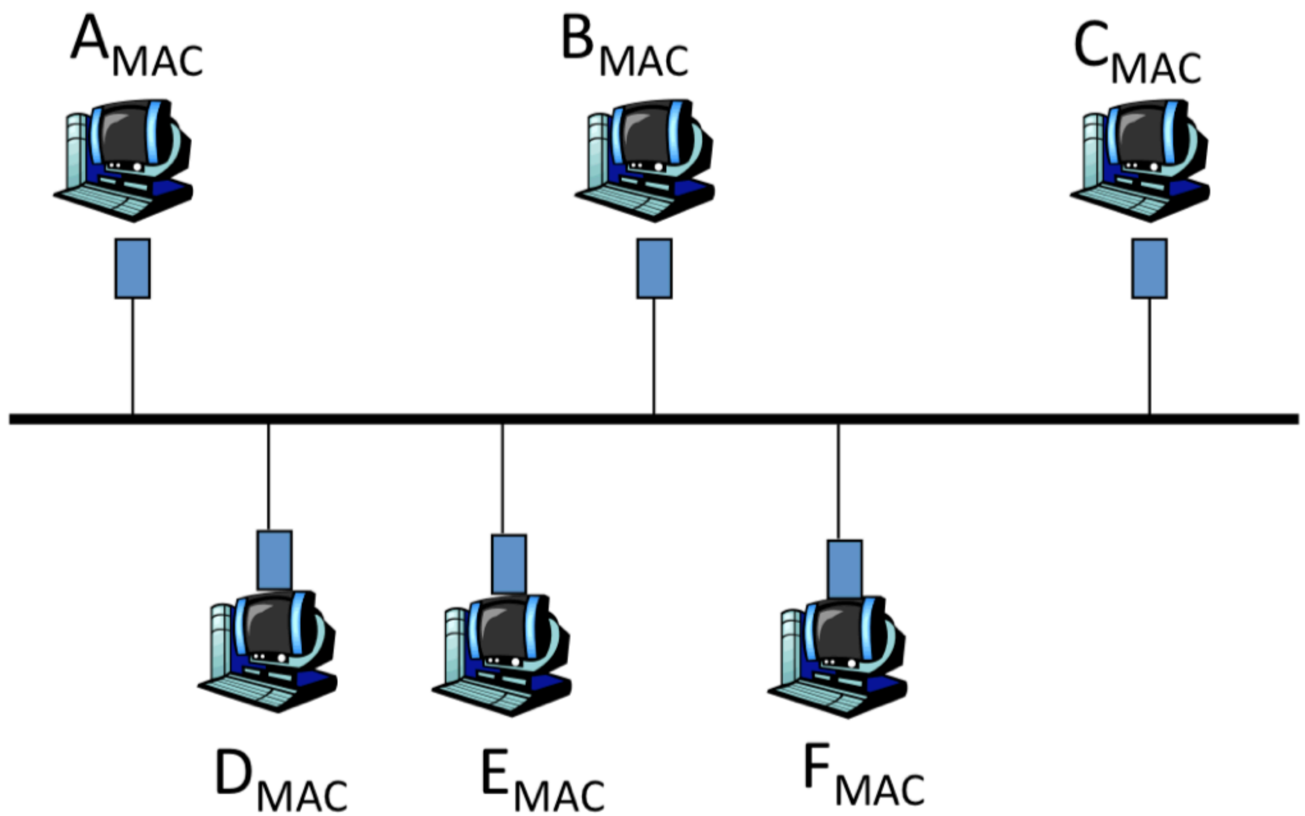
- **Preamble:**
  - 7 bytes for clock synchronization
  - 1 byte to indicate start of the frame
- **Addresses:** 6 + 6 bytes (MAC addresses)
- **Protocol type:** 2 bytes, indicating higher layer protocol (e.g., IP, Appletalk)
- **Data payload:** max 1500 bytes, minimum 46 bytes
- **CRC:** 4 bytes for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

Routing with Broadcast Ethernet



- Sender transmits onto a broadcast link
- Frame contains destination MAC addresses
- Each receiver's link layer passes the frame to the network layer **iff**:
  - destination address matches the receiver's MAC address
  - Or, the destination address 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 is good to go
  - No configuration by users or network operators
  - Broadcast as a means of bootstrapping communication

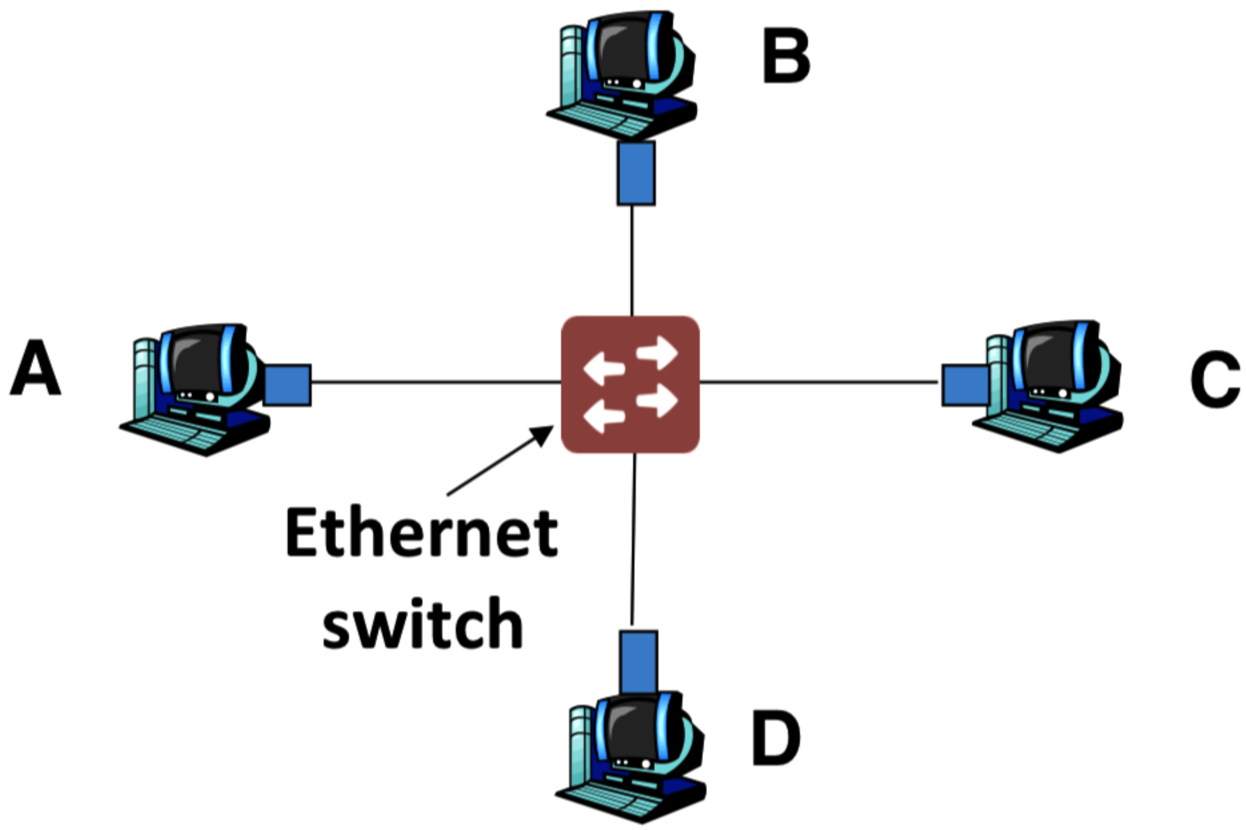
Evolution

- **Ethernet was invented as a broadcast technology**
  - Hosts share channel
  - Each packet received by all attached hosts
  - CSMA/CD for media 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)

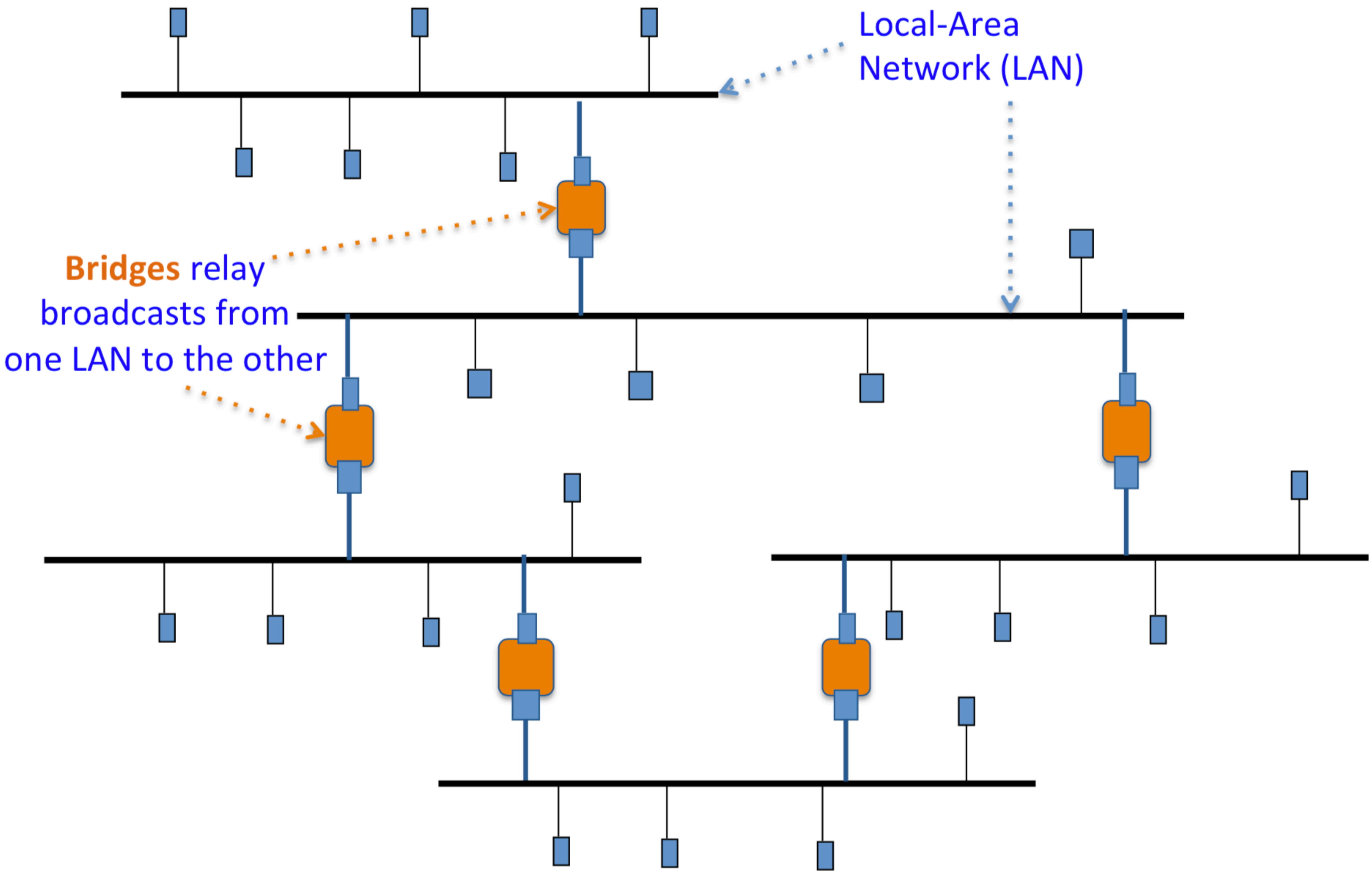
Switched Ethernet

Switched Ethernet

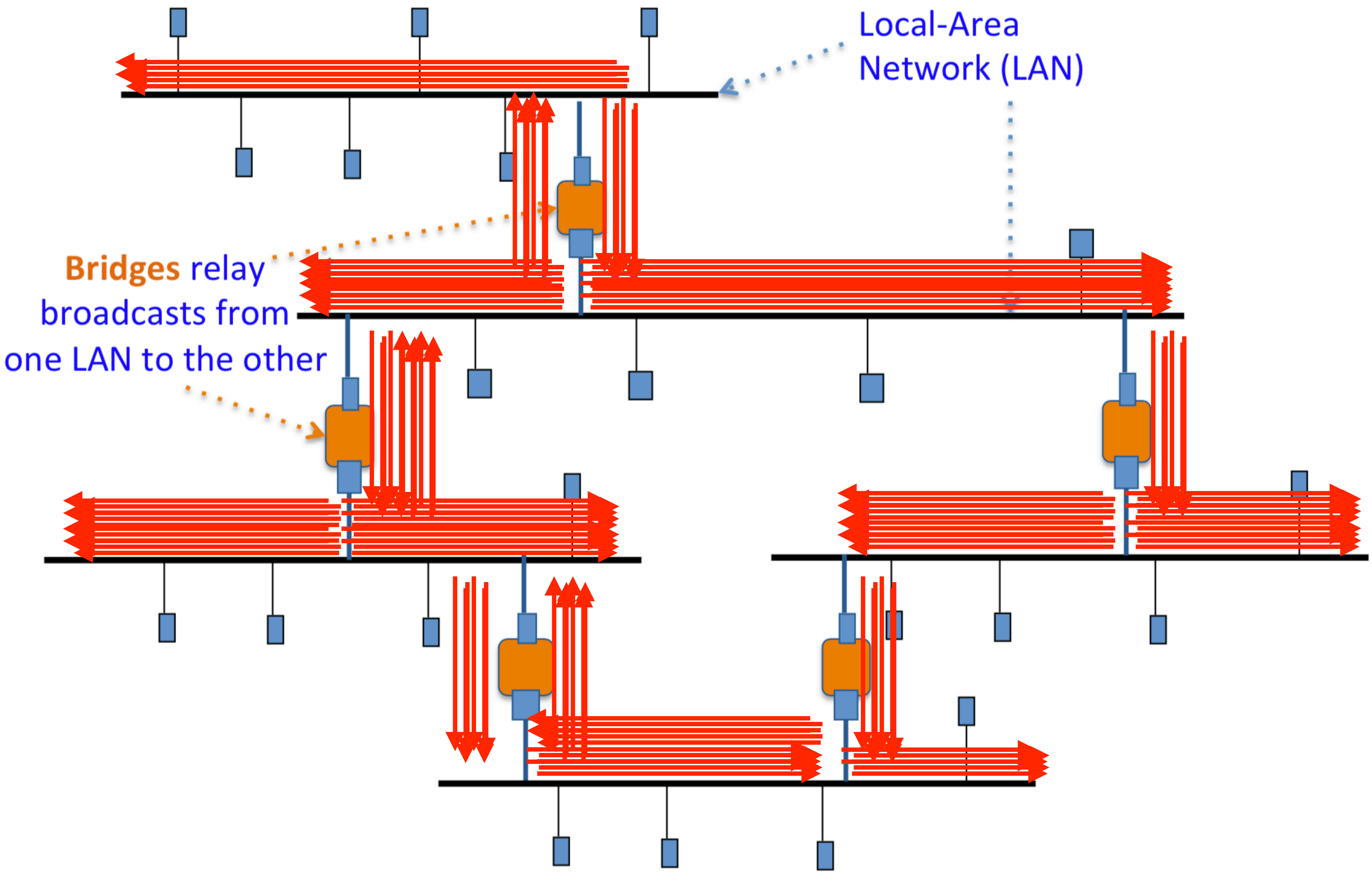


- Enables concurrent communication
    - Host A can talk to C, while B talks to D
    - No collisions -> no need for CSMA, CD
    - No constraints on link lengths or frame size

Routing in "Extended LANs"



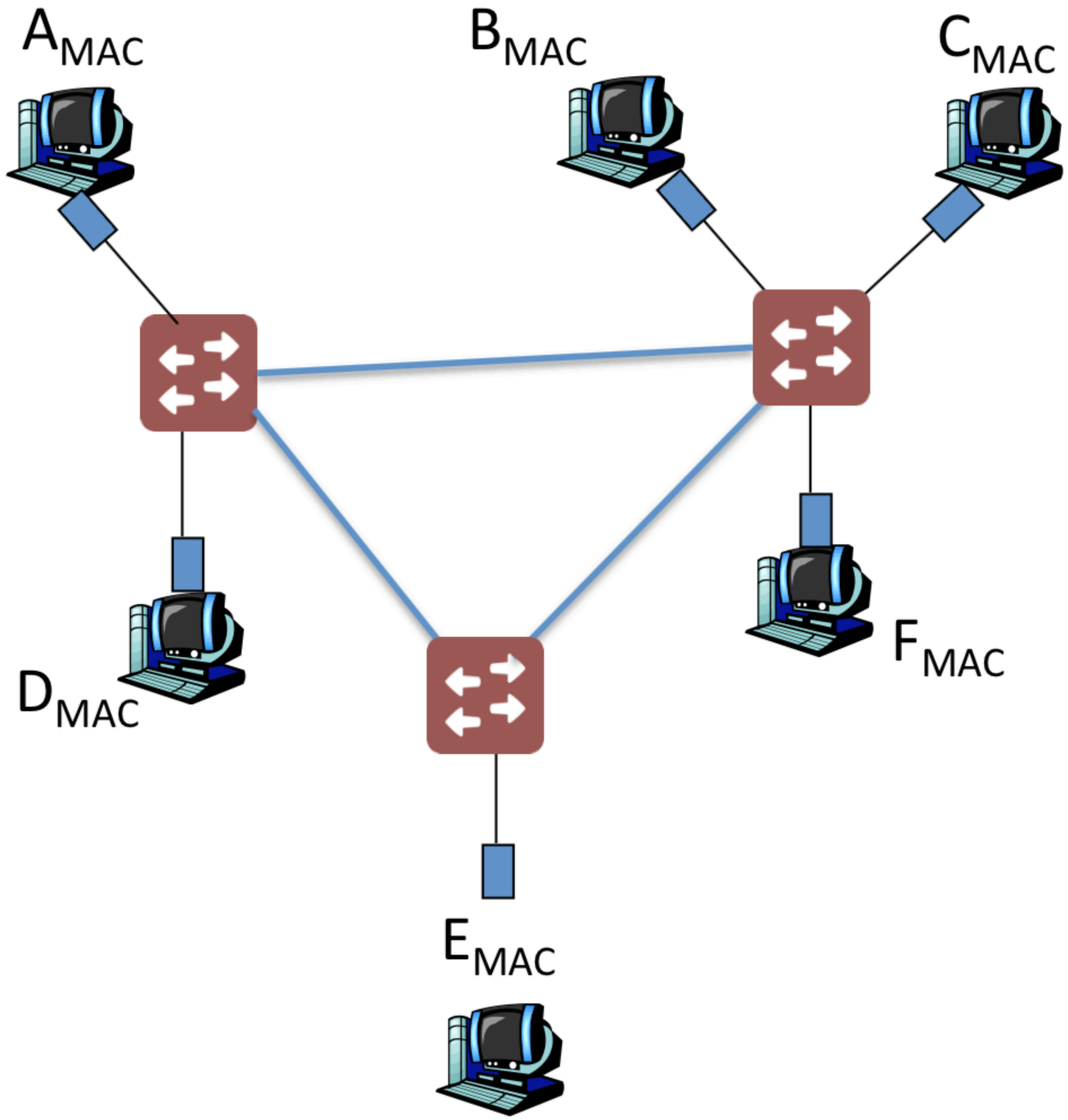
Naïvely Routing in "Extended LANs": Broadcast storm



How to avoid the Broadcast Storm Problem?

Get rid of the loops!

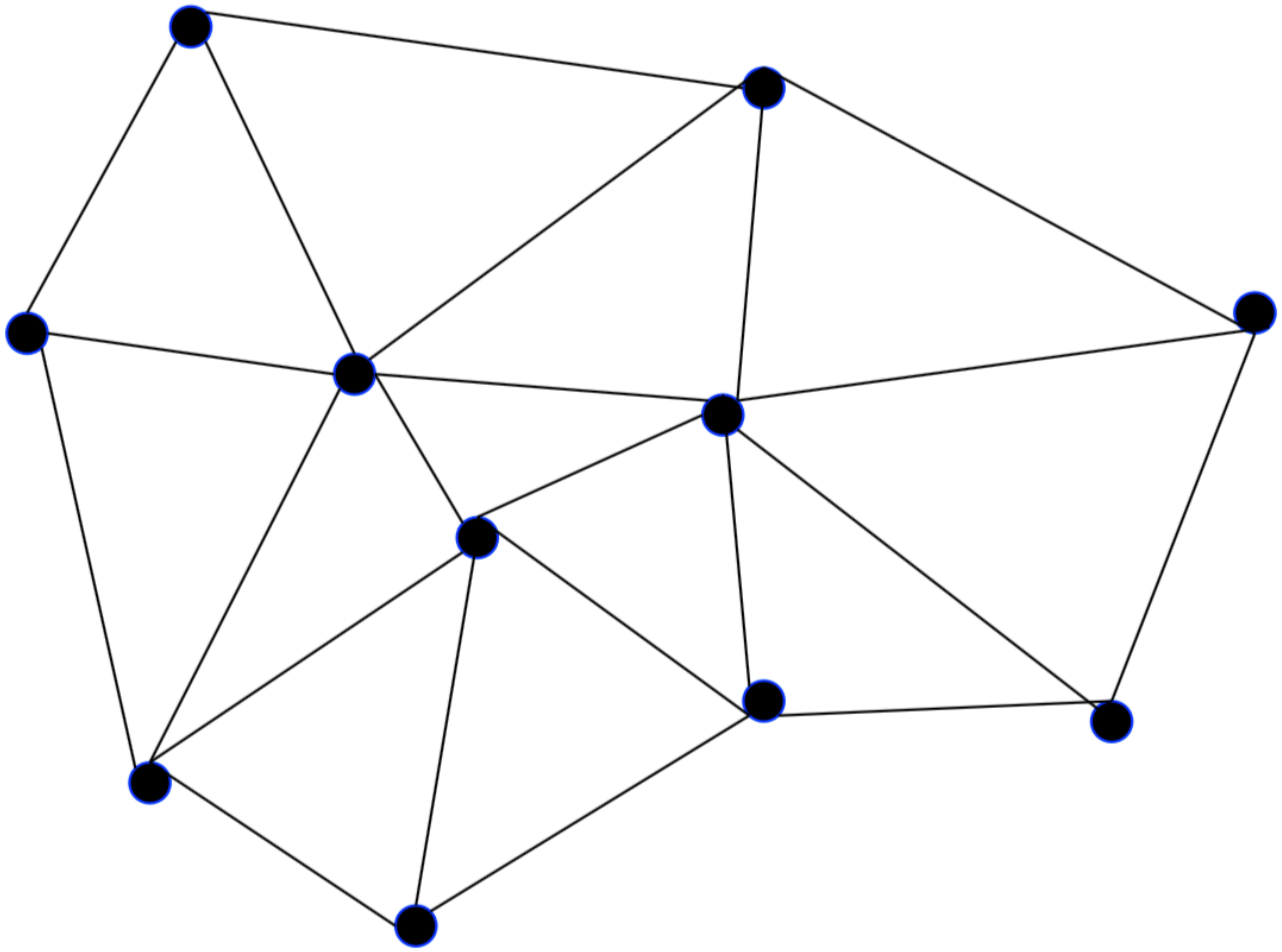
Lets get back to the graph representation!



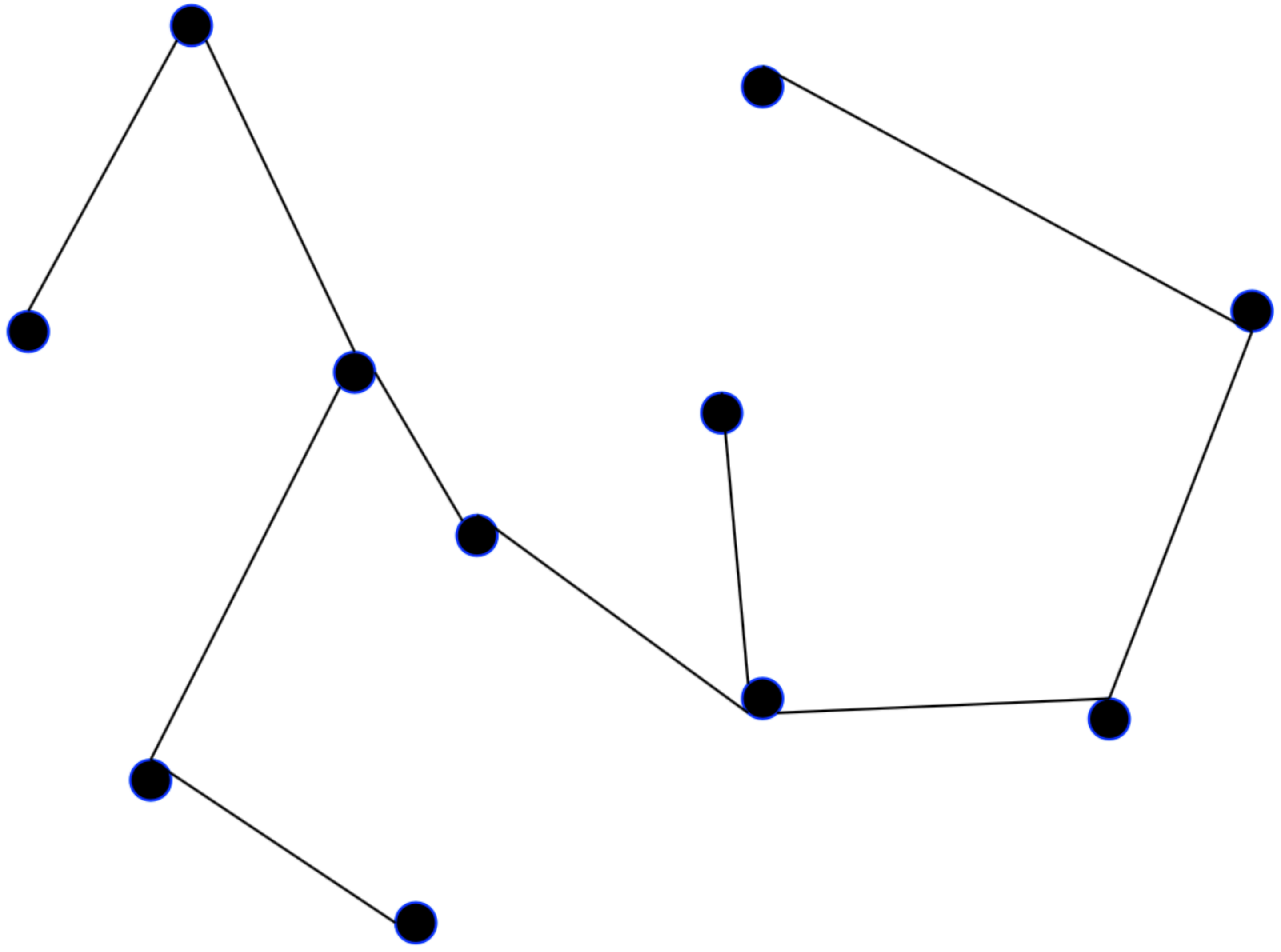
Easiest Way to Avoid Loops

- Use a topology where loop is impossible!
- Take arbitrary topology
- 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!

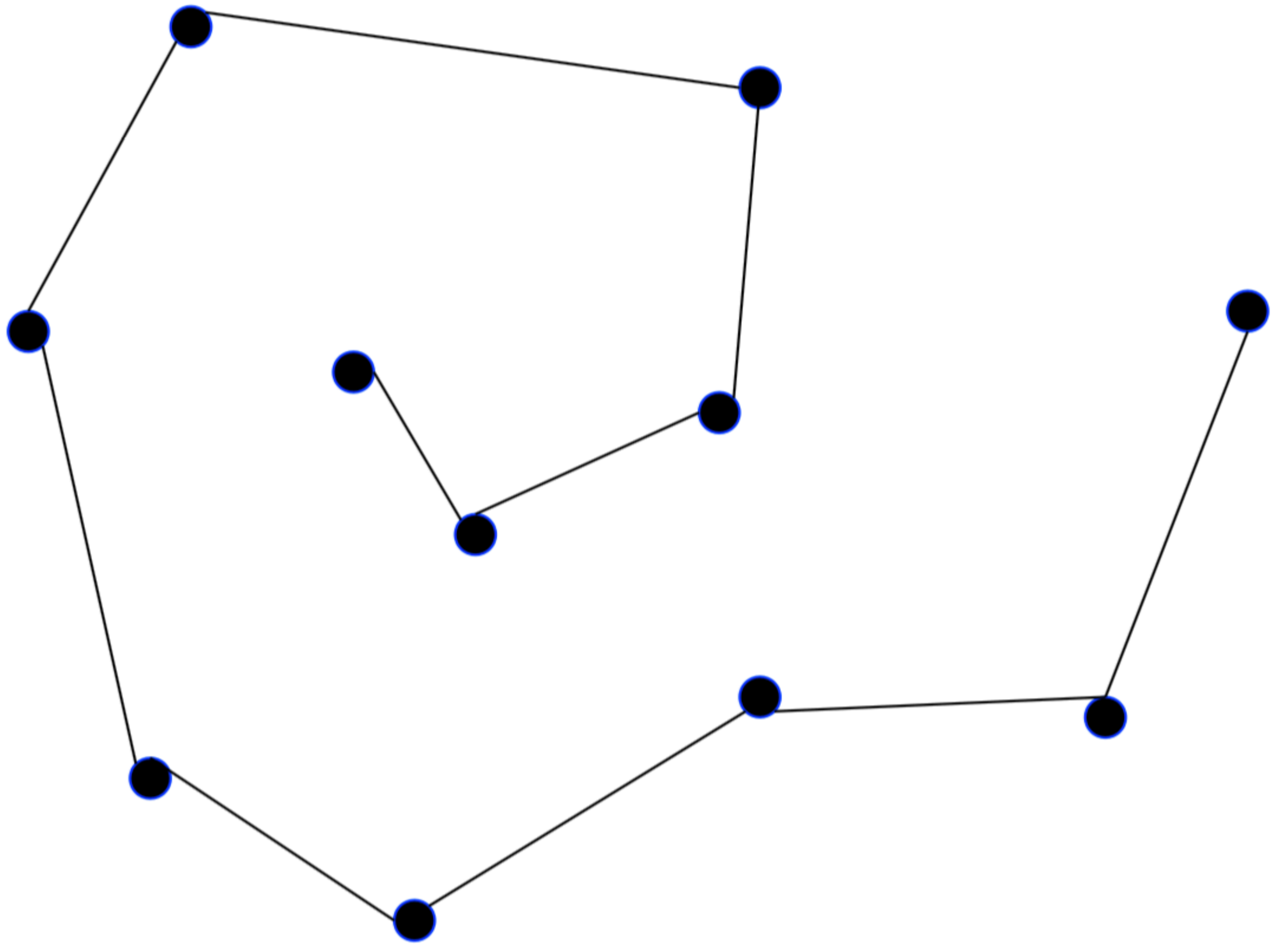
Consider Graph



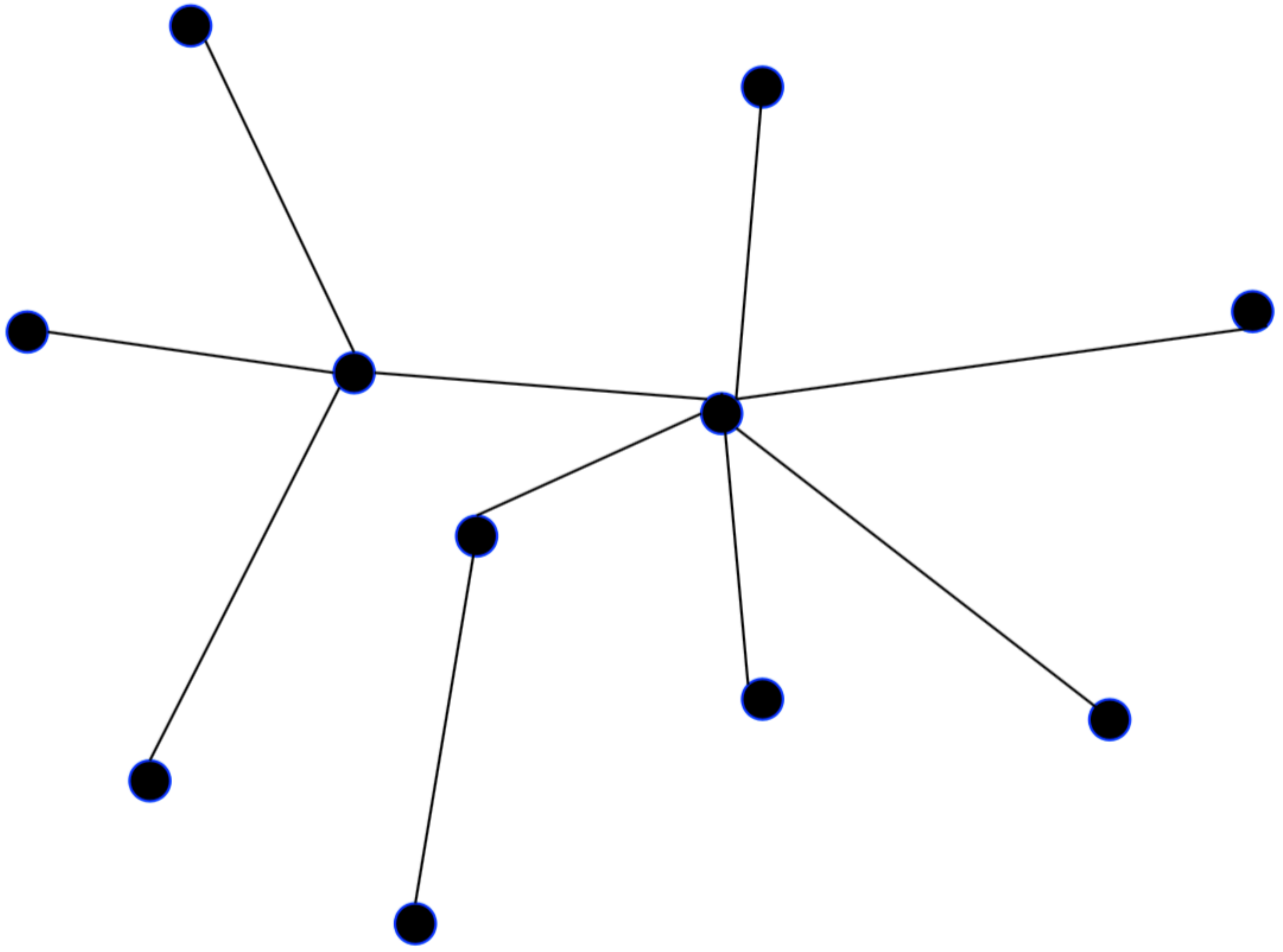
A Spanning Tree



Another Spanning Tree



Yet Another Spanning Tree



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 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 address) as root
  - Y in 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

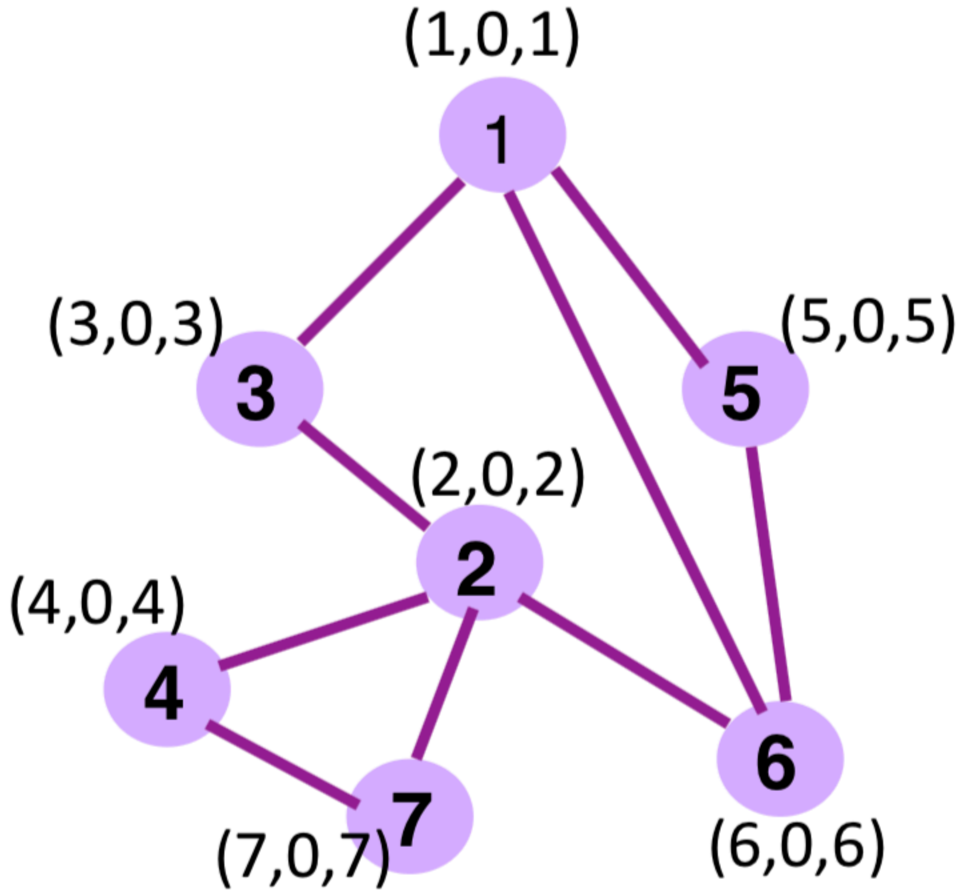
Steps in Spanning Tree Protocol

- 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

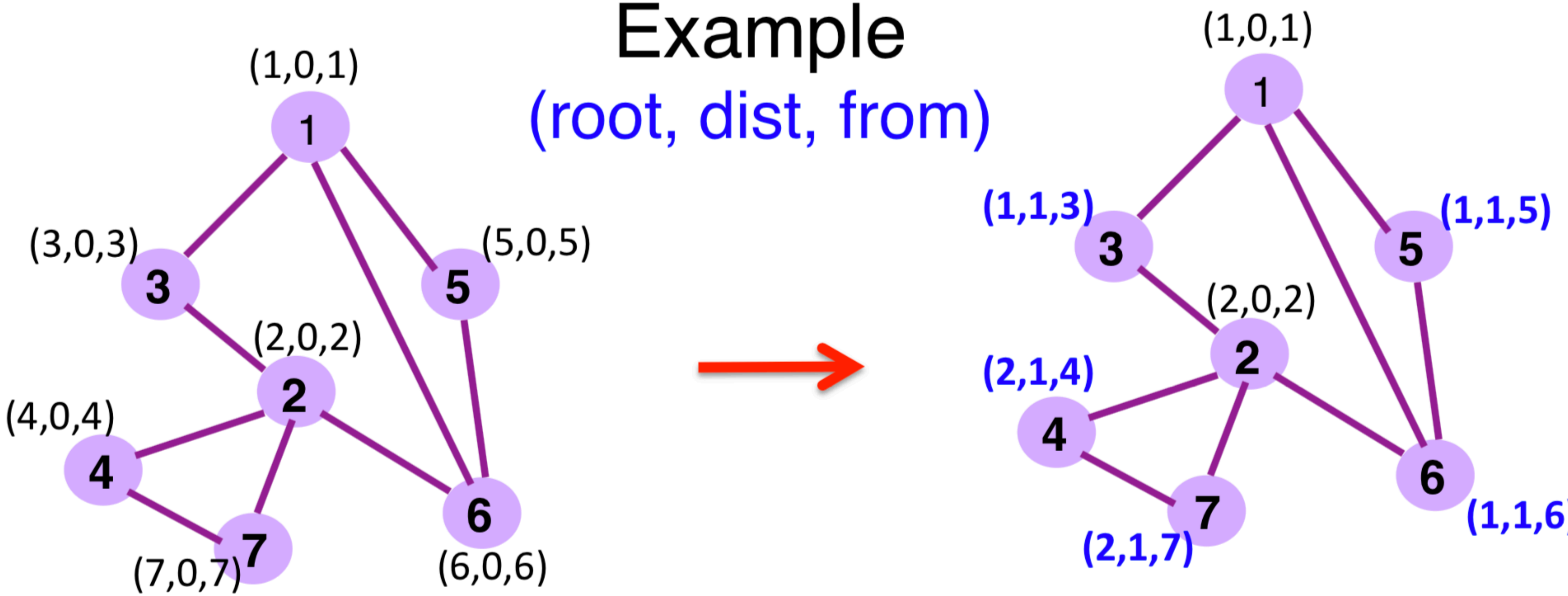
- Switches compute their distance from the root
  - Add 1 to the shortest distance received from a neighbor

- If root or shortest distance to it changed, send neighbors updated message (Y,d+1,X)


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

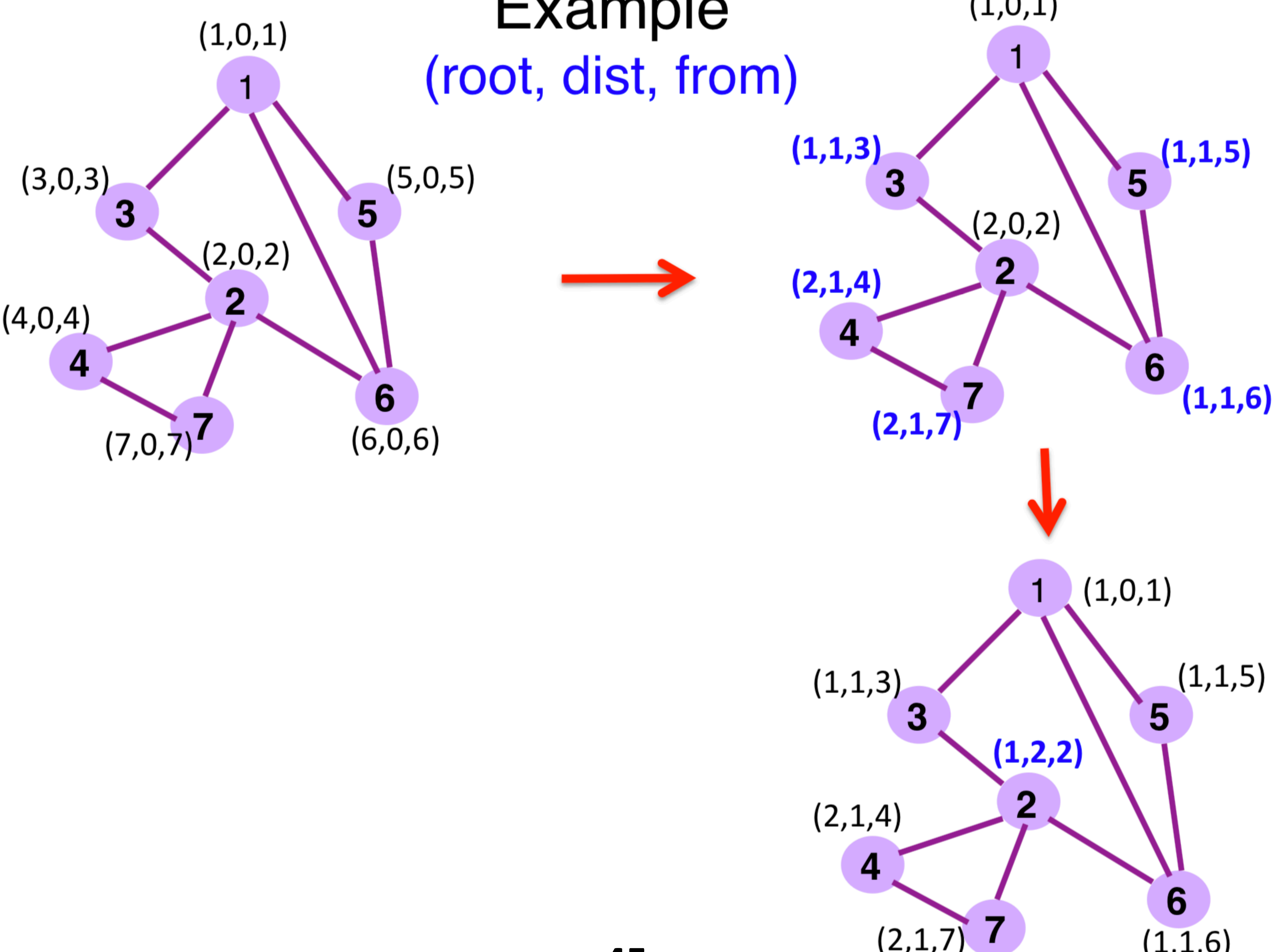
Example

(root, dist, from)



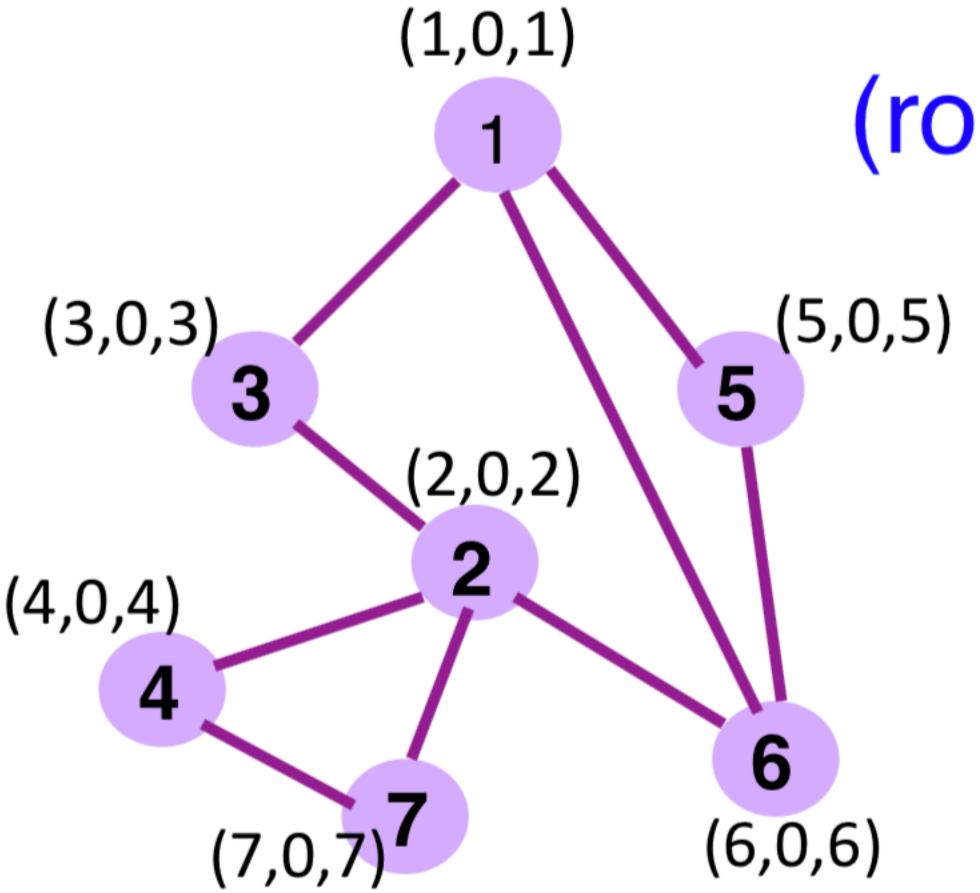
Example

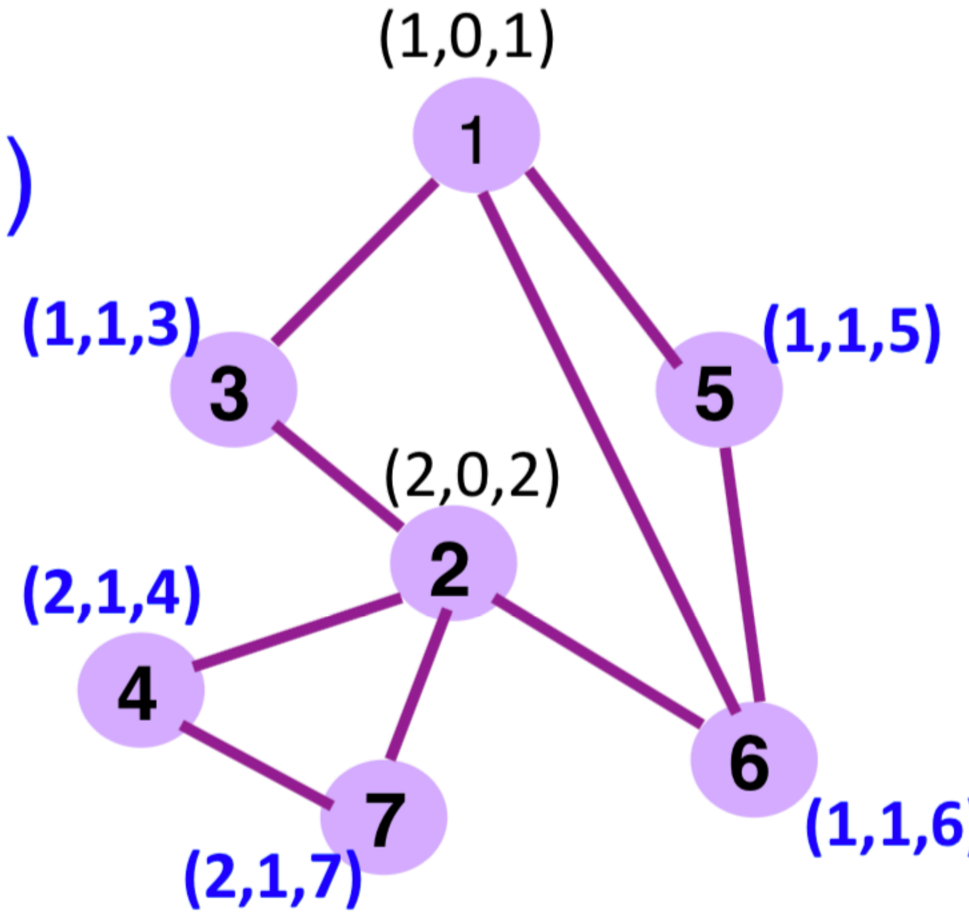
(root, dist, from)

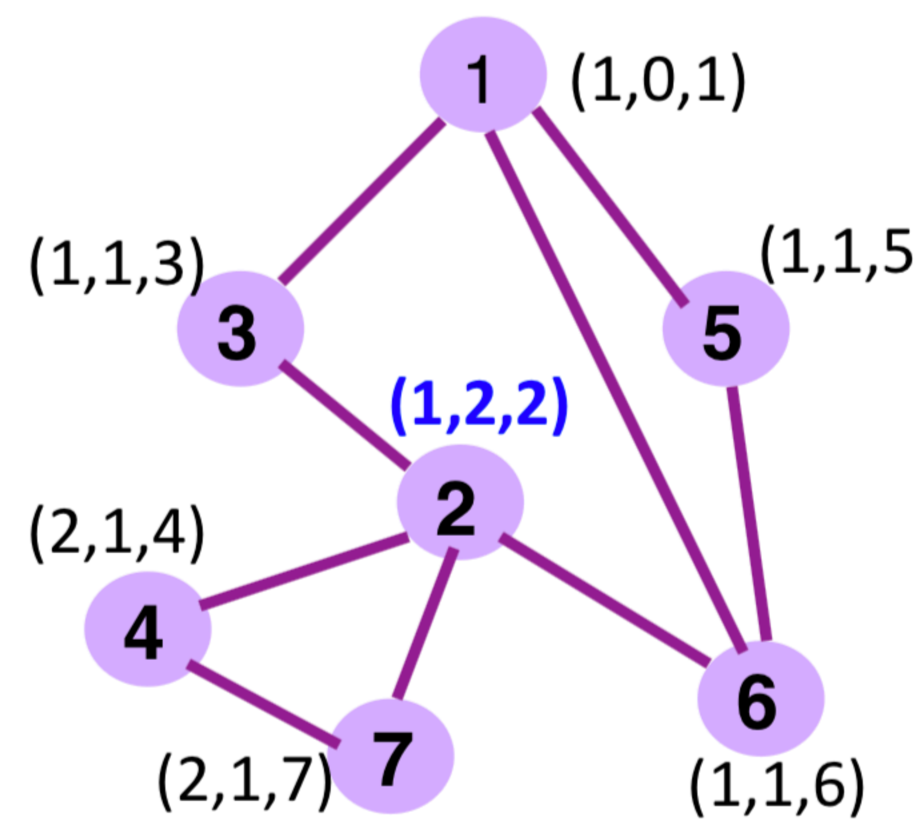


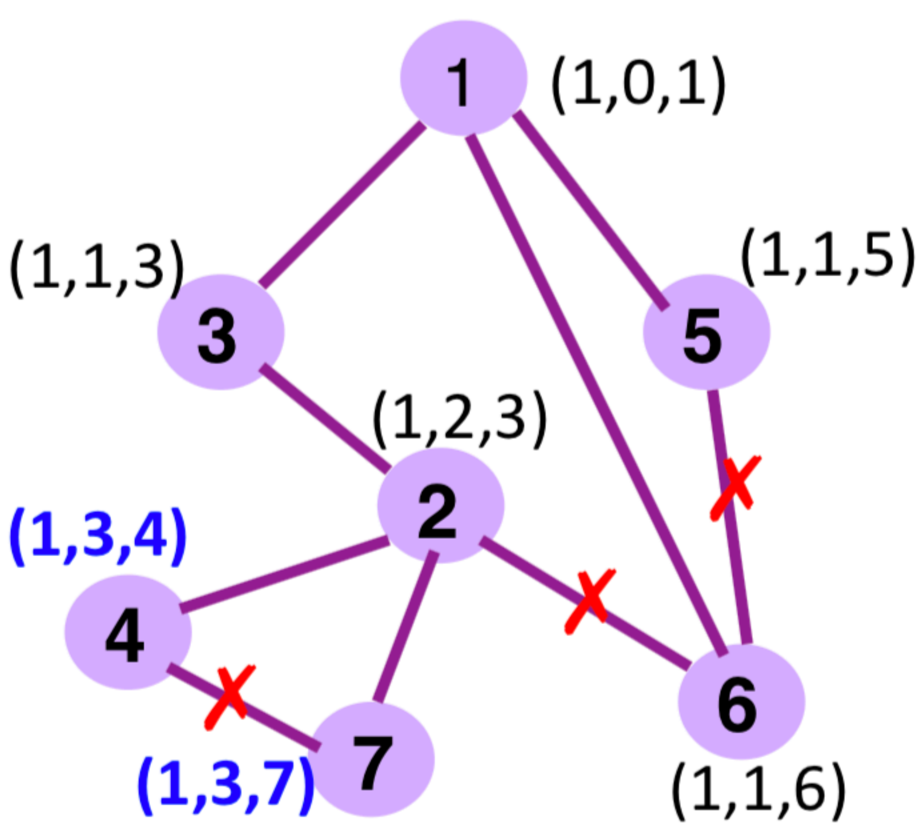
Example

(root, dist, from)



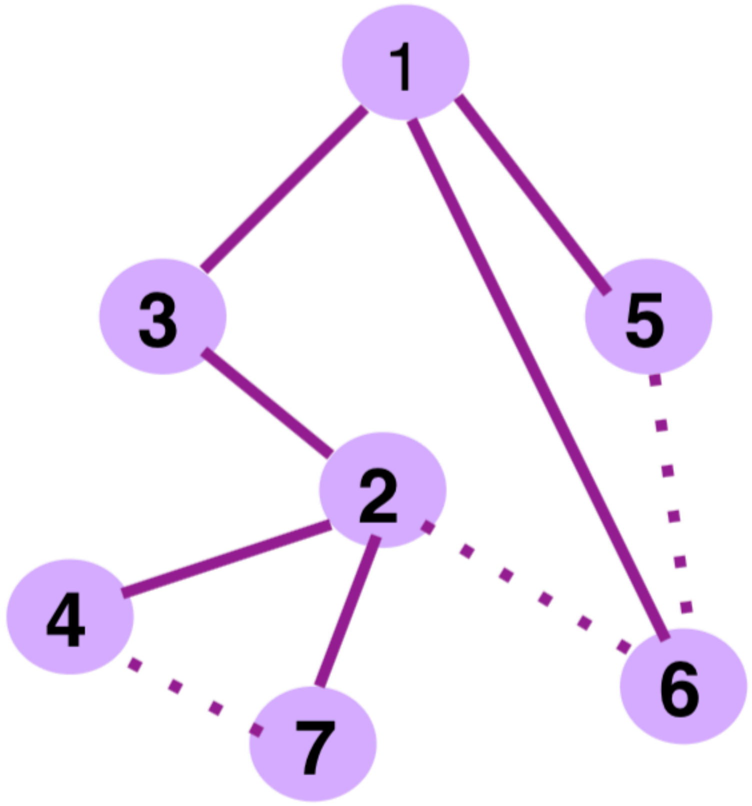






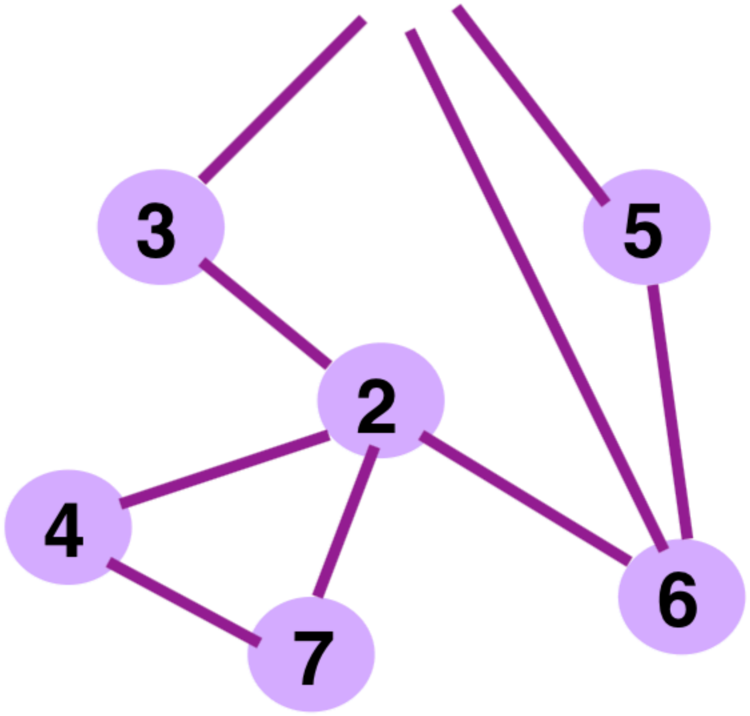
Links on Spanning Tree

- 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



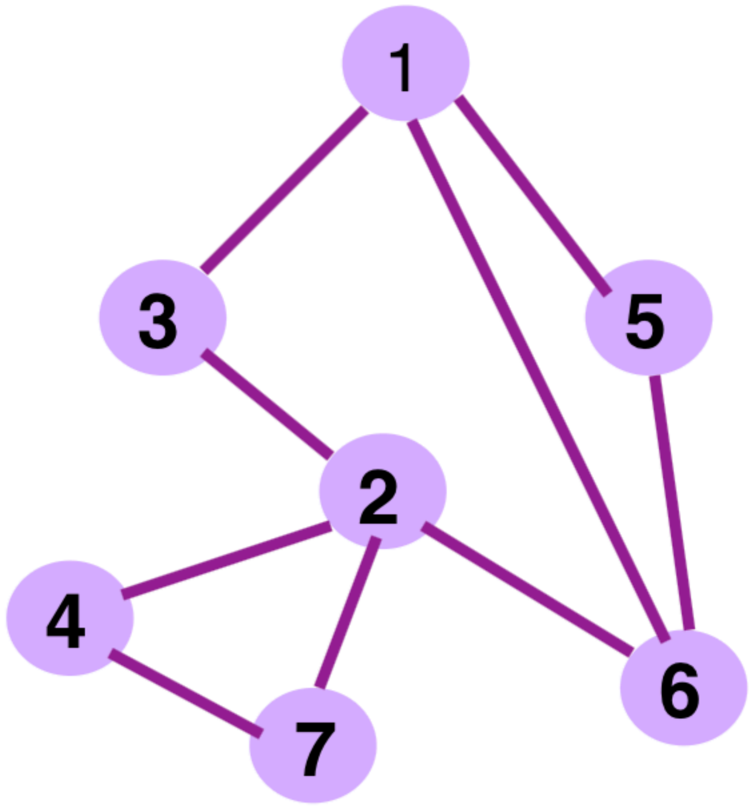
Robust Spanning Tree Algorithm

- Algorithm must react to failures
  - Failure of the root node
  - Failure of switches and links
- Root node sends periodic 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!

More details

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 it 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 its 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 receives (1,3,7) from 7
  - And realizes it is a longer path
  - So, prefers its own three-hop path
  - And removes 4-7 link from the tree