

The Link Layer: Ethernet

CPSC 433/533, Spring 2021

Anurag Khandelwal

Outline

- Frames and framing
- Addressing
- Routing
- Forwarding
- Discovery: Bootstrapping end-to-end communication

Medium Access Control Address

- MAC address
 - Numerical address associated with a network adapter
 - Flat namespace of 48 bits (e.g., 00-15-C5-49-04-A9 in HEX)
- 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 (e.g., 00-15-C5-49-04-A9)



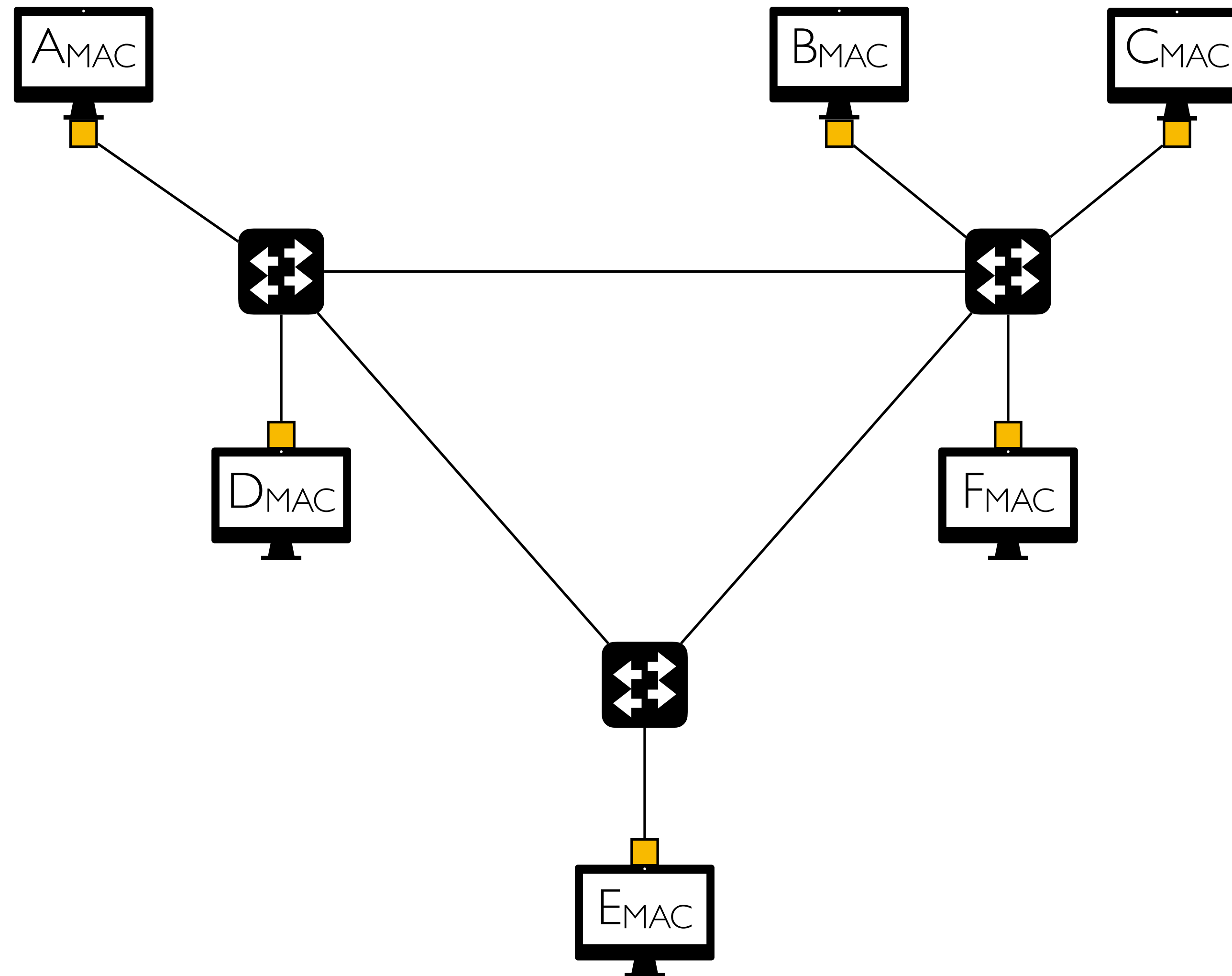
MAC address vs. IP address

- **MAC addresses (used in link layer)**
 - *Hard-coded* when adapter is built
 - *Flat* namespace of 48 bits (e.g., 00-15-C5-49-04-A9)
 - Like a social security number
 - Portable and can stay the same as the host moves
 - Used to get packets between interfaces on the same network
- **IP addresses**
 - *Configured*, or learned dynamically
 - *Hierarchical* namespace 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
- Addressing
- Routing
- Forwarding
- Discovery: Bootstrapping end-to-end communication

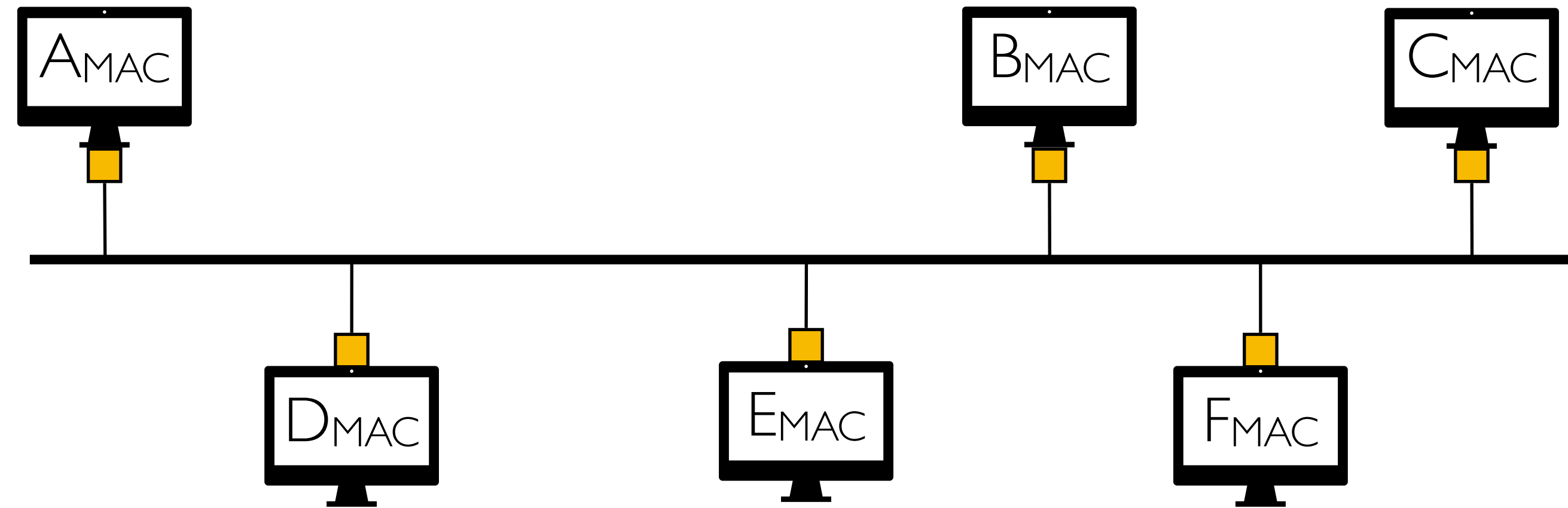
Routing with Switched Ethernet?



Why does Ethernet not use LS/DV?

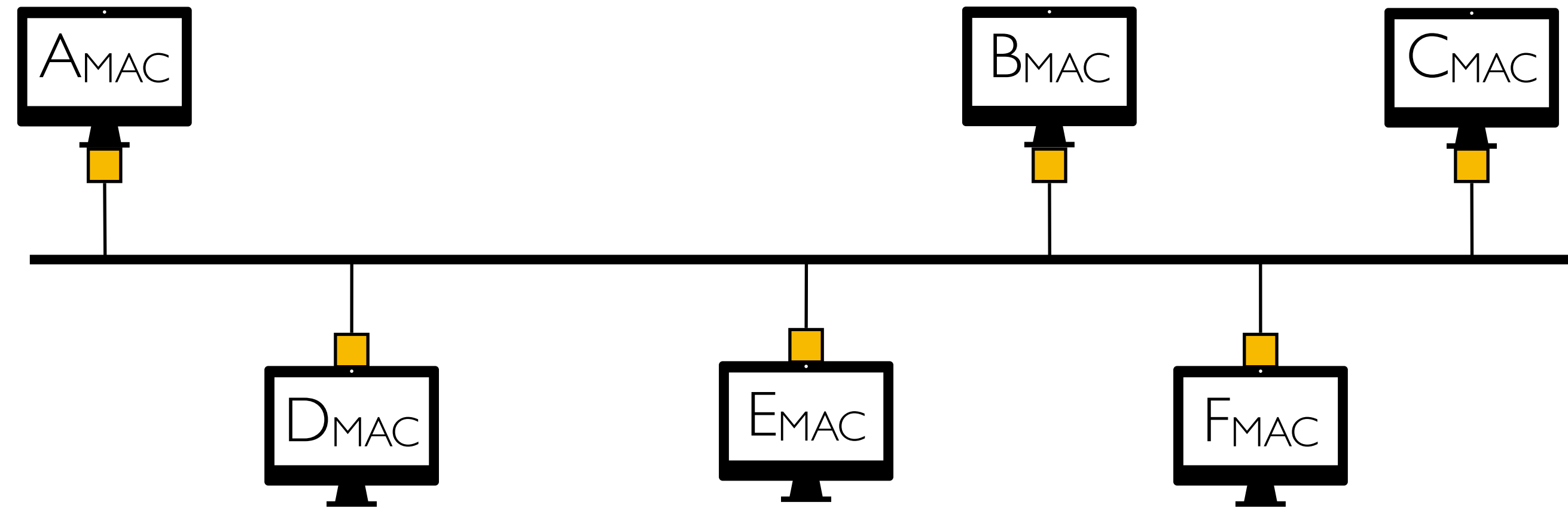
- Concerns over scalability
 - Flat MAC addresses cannot be aggregated like IP addresses
- Legacy

“Routing” with broadcast ethernet



- Sender transmits frame on to 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 address is the broadcast MAC address (ff:ff:ff:ff:ff:ff)

“Routing” with broadcast ethernet

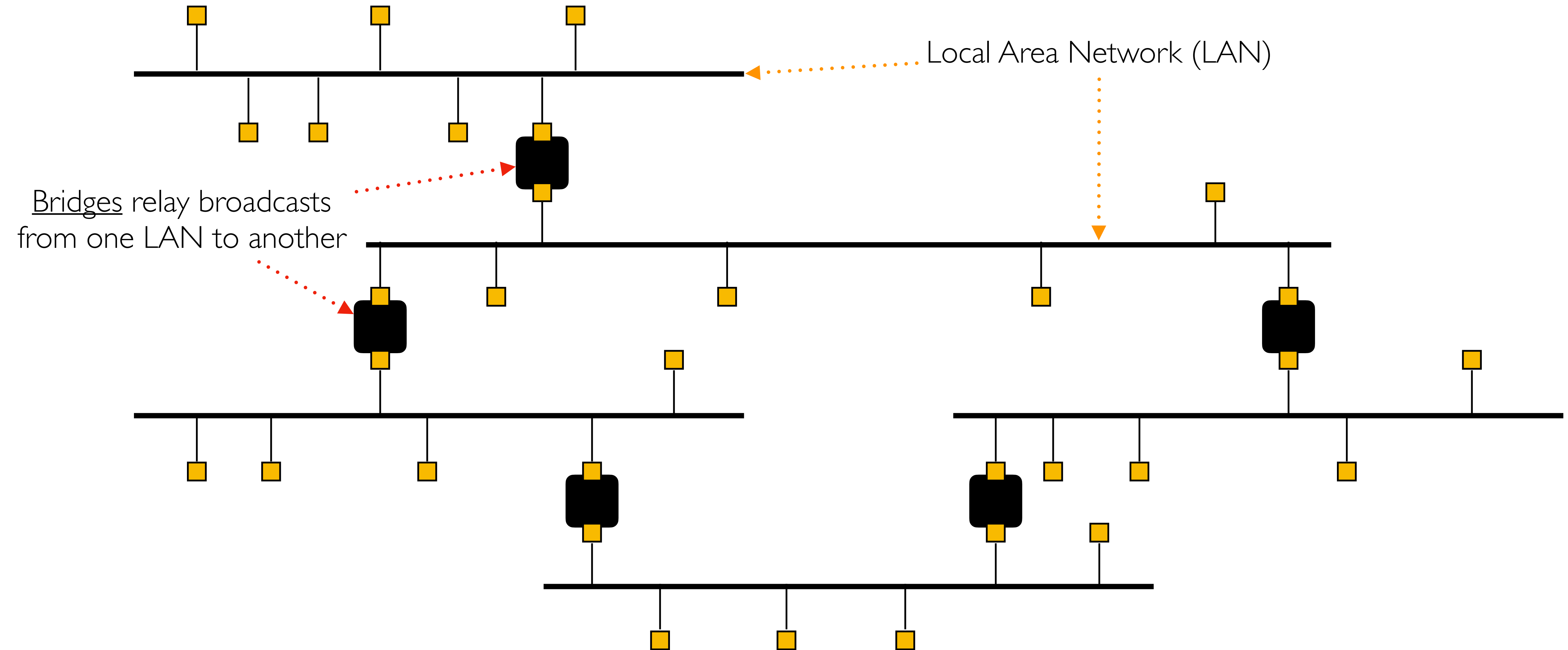


- Ethernet is ‘plug-n-play’
 - A new host plugs into the Ethernet and is good to go
 - No configuration by users or network operators
 - Broadcast as a means of bootstrapping communication

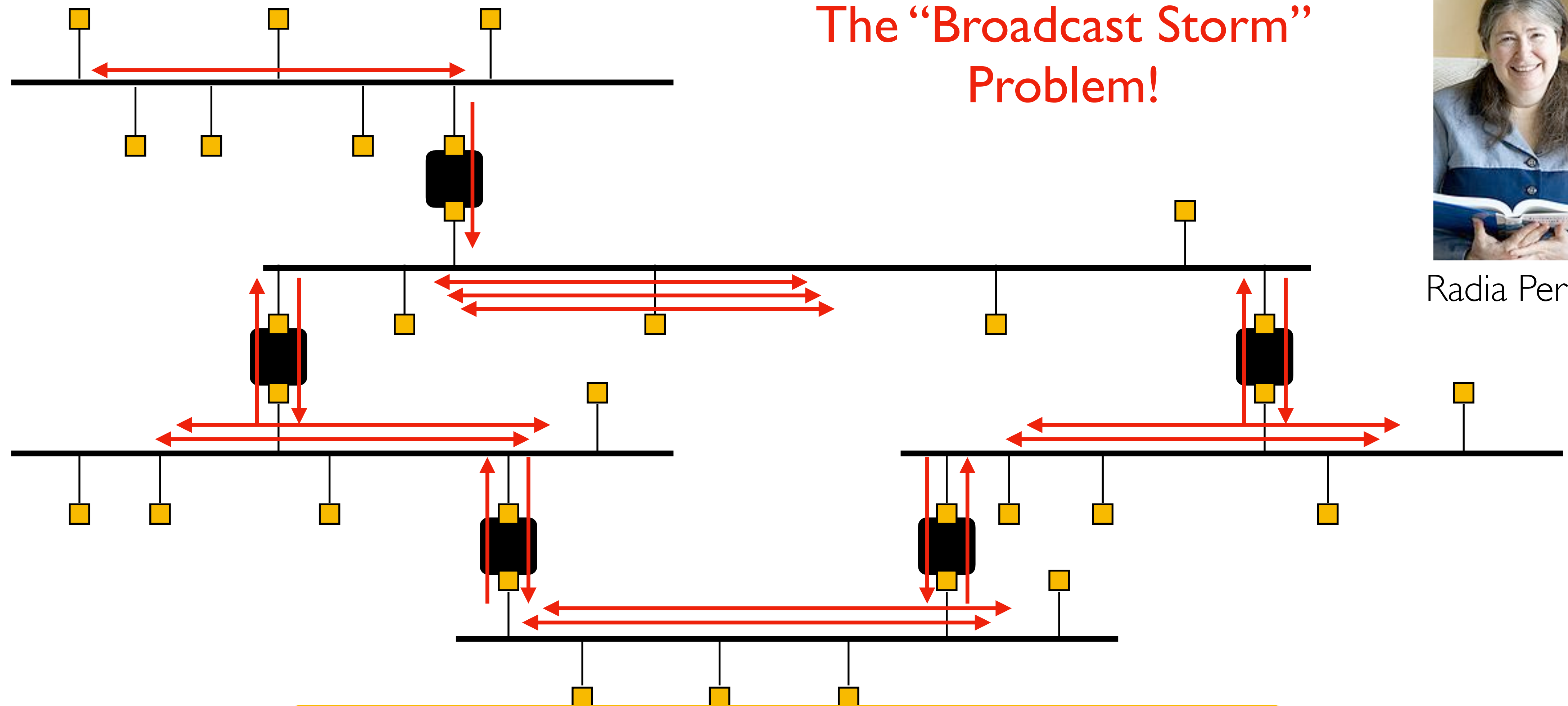
Why does Ethernet not use LS/DV?

- **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

Some History: Routing in “Extended LANs”



Some History: Routing in “Extended LANs”



The “Broadcast Storm”
Problem!



Radia Perlman

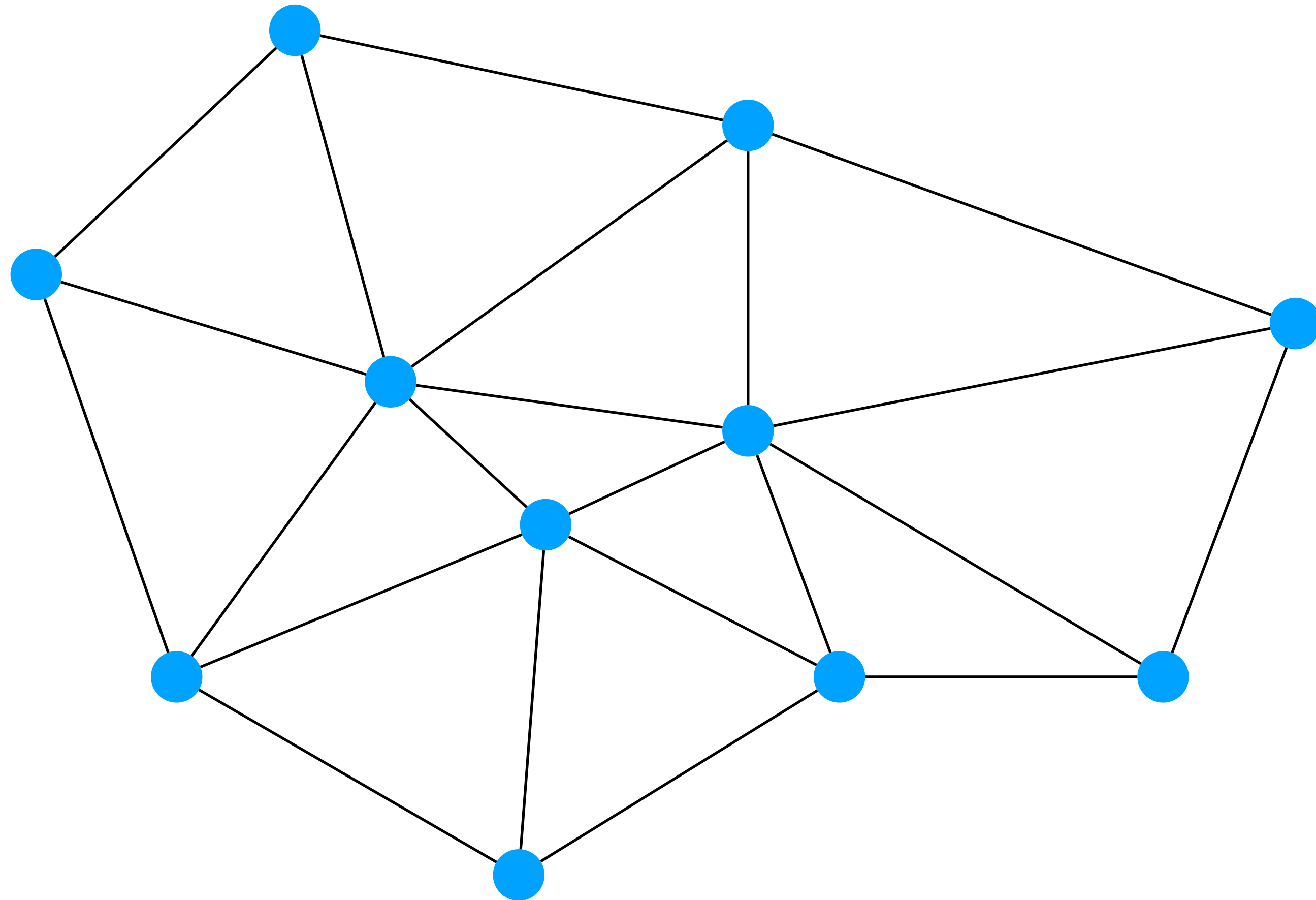
Perlman’s Idea: eliminate loops in the topology

Loops!

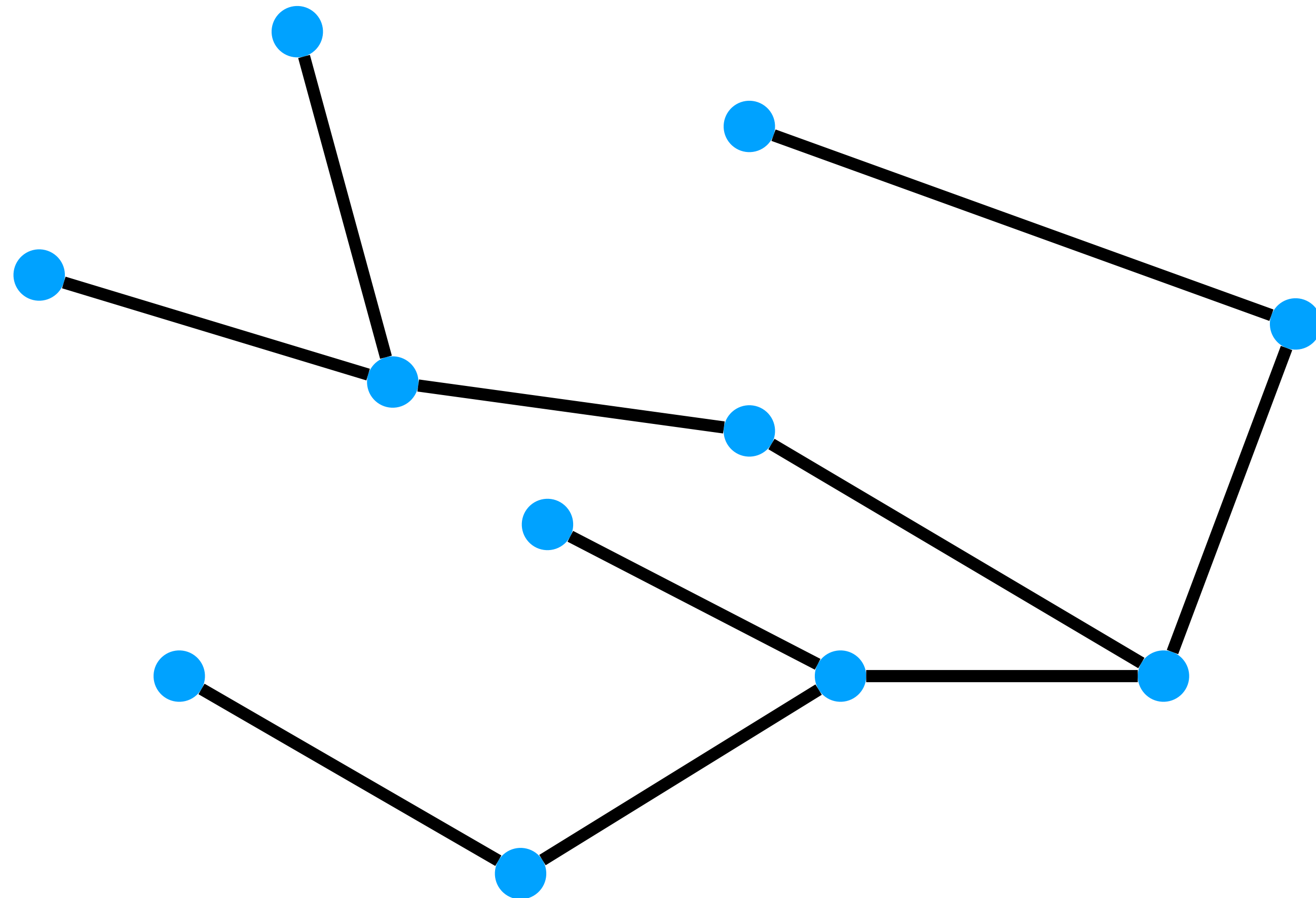
Easiest Way to Avoid Loops

- 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 destination on spanning trees**
 - So don't have to worry about loops!

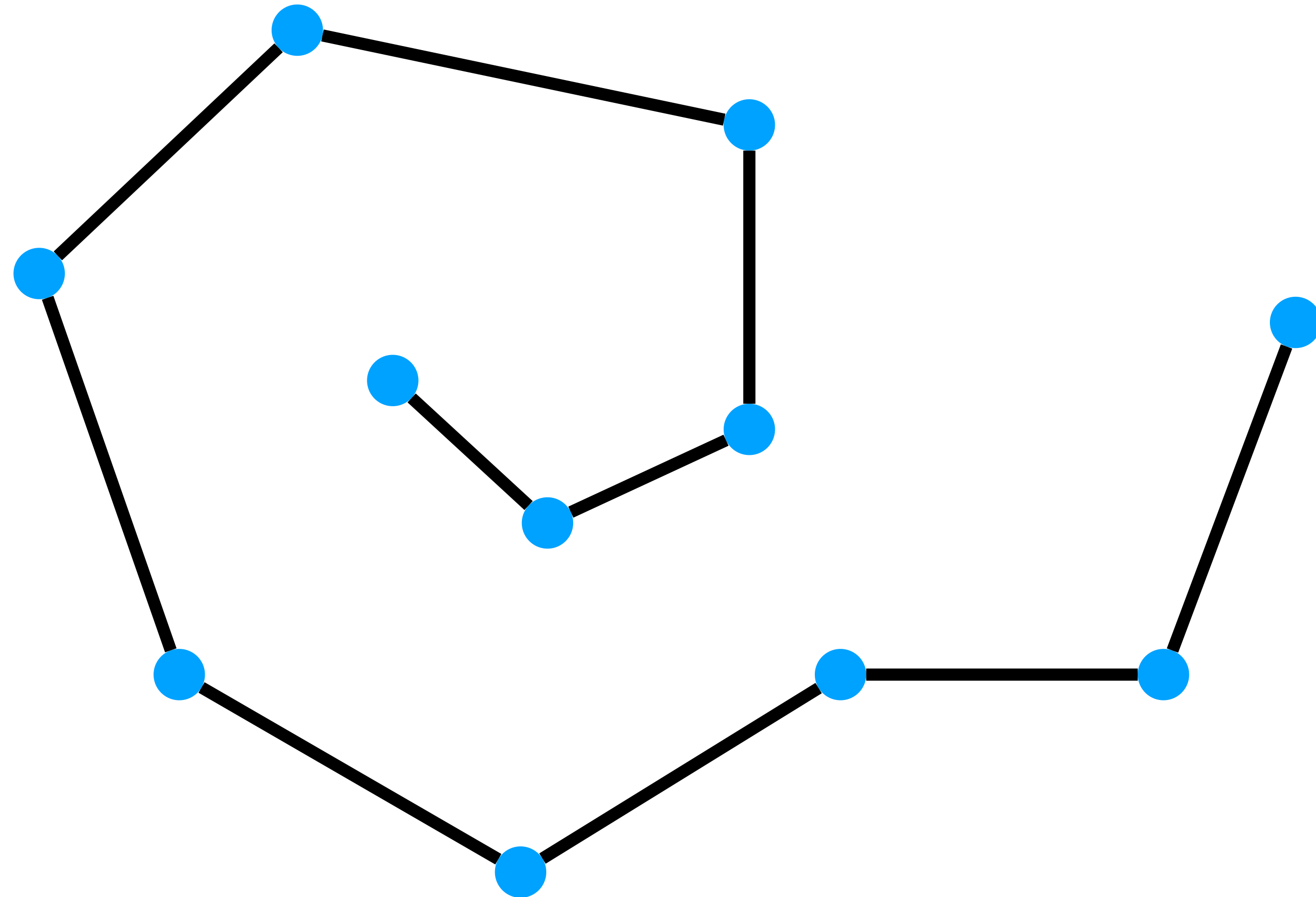
Consider Graph



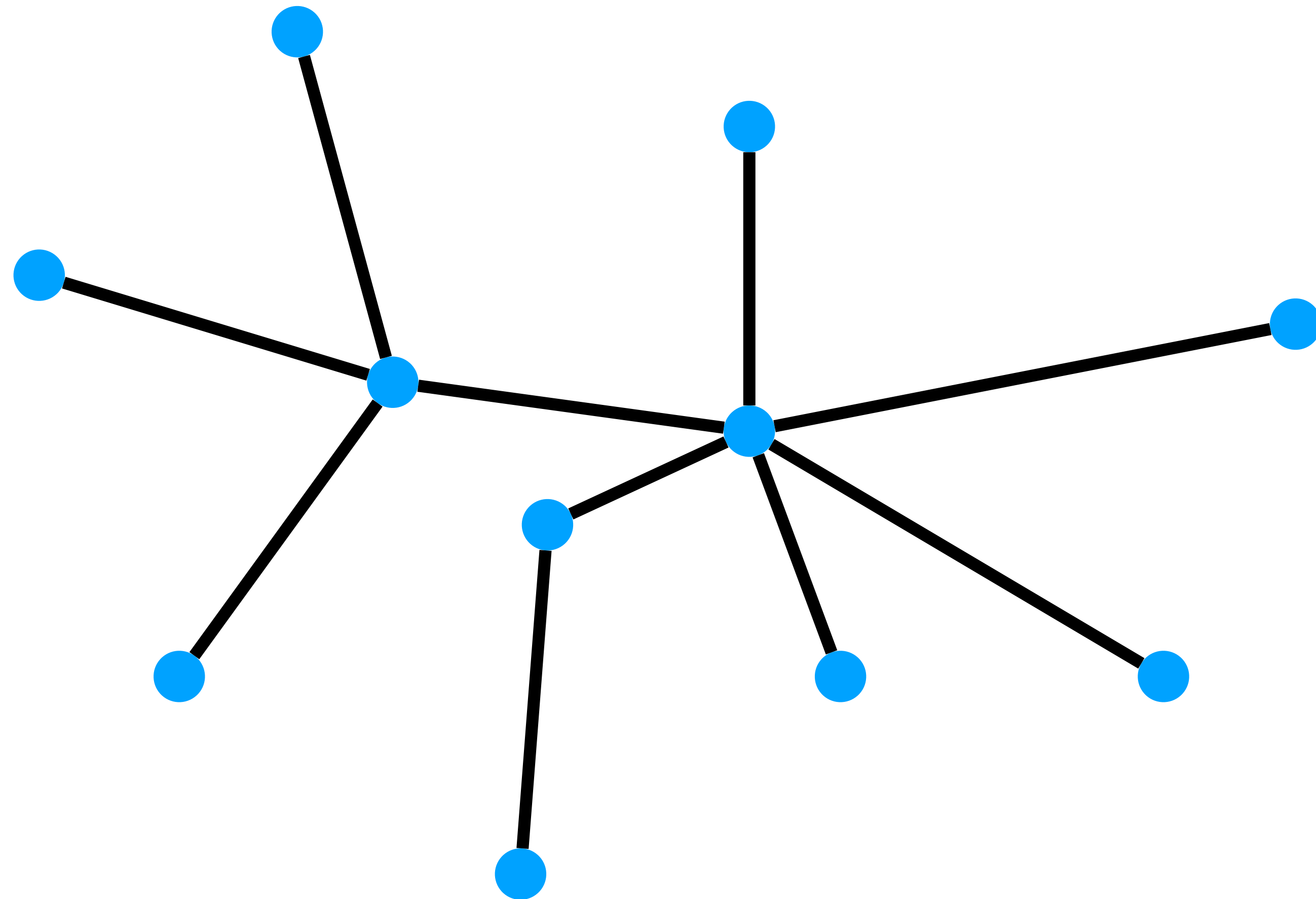
A Spanning Tree



Another Spanning Tree



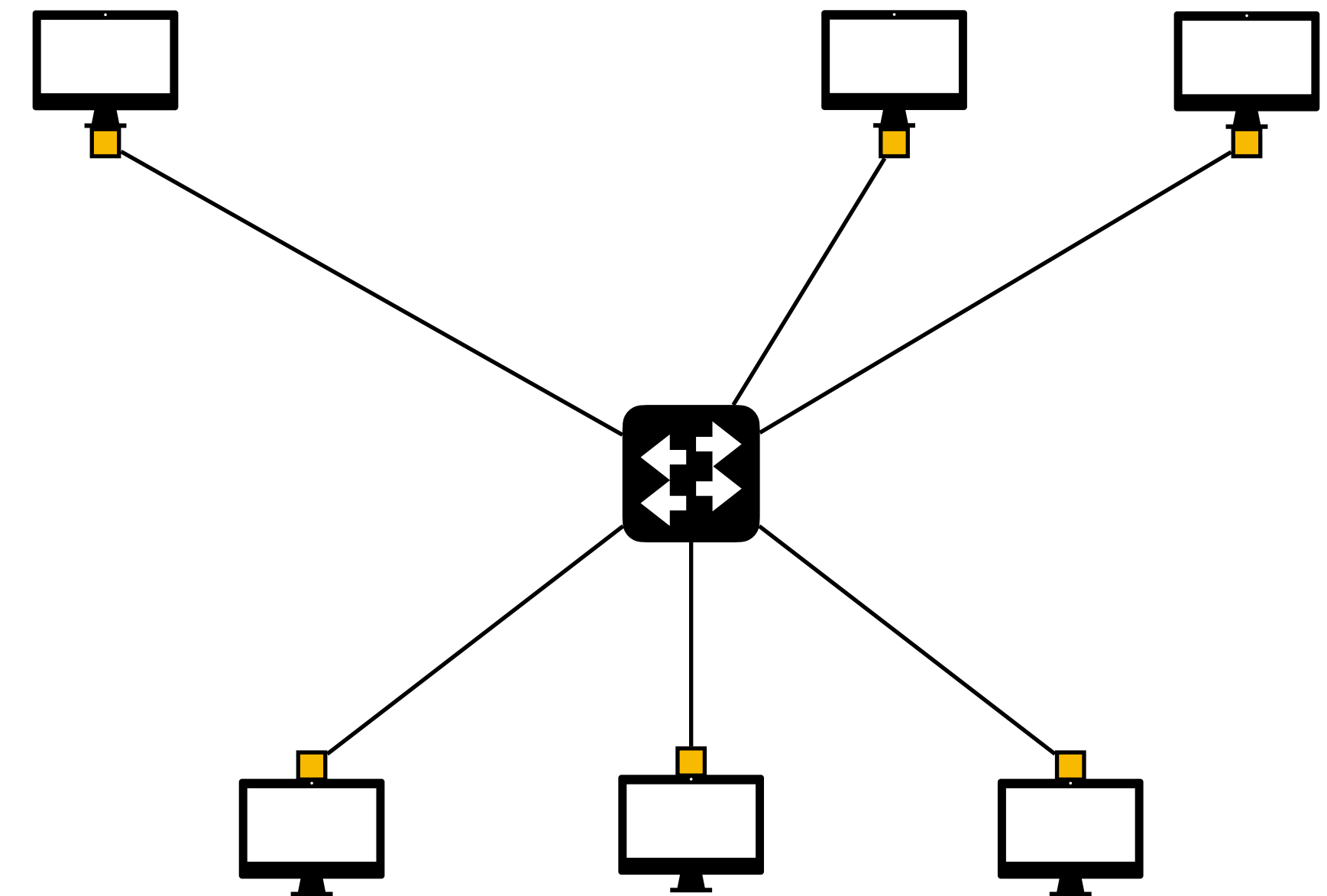
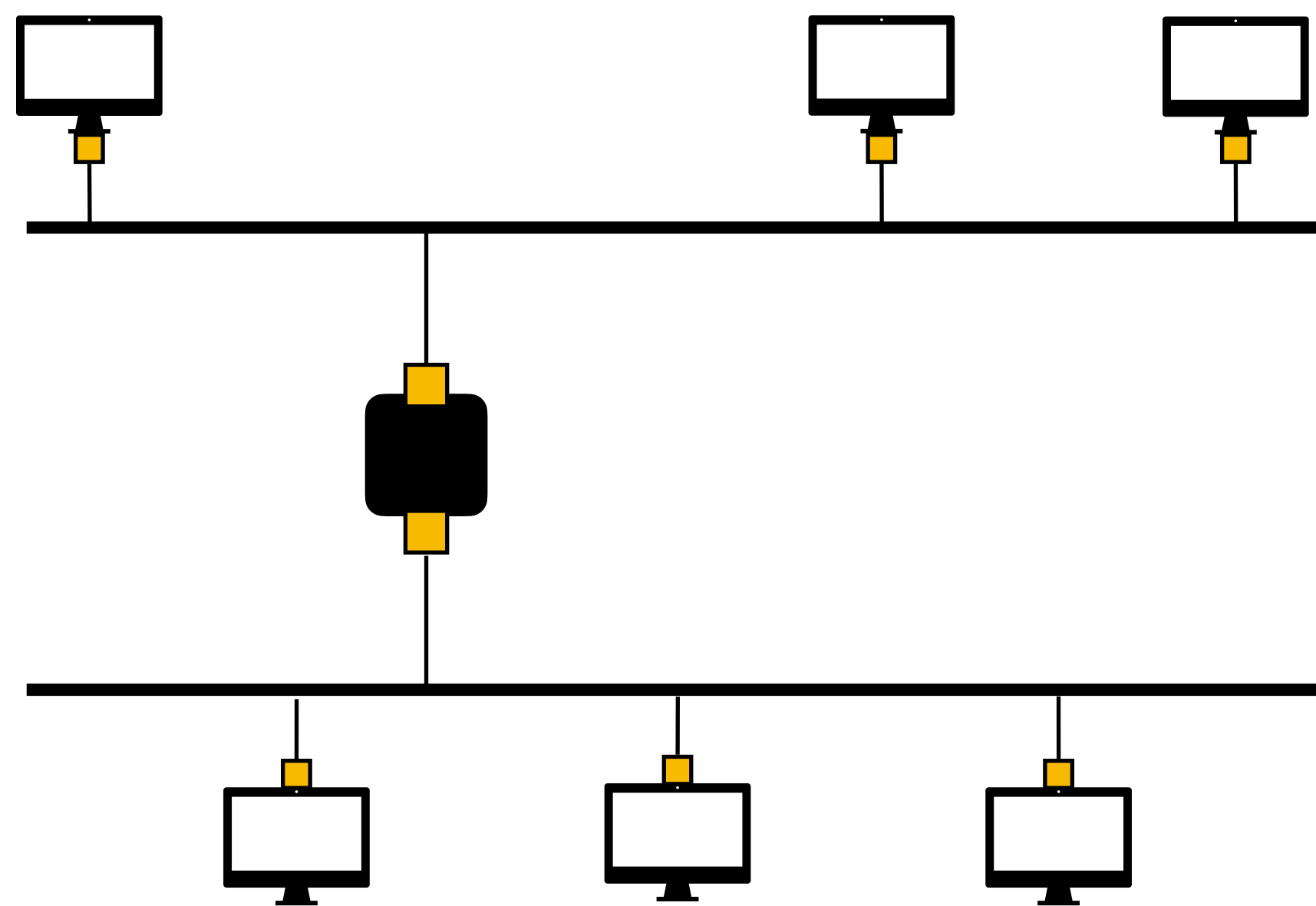
Yet Another Spanning Tree



Some History: 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

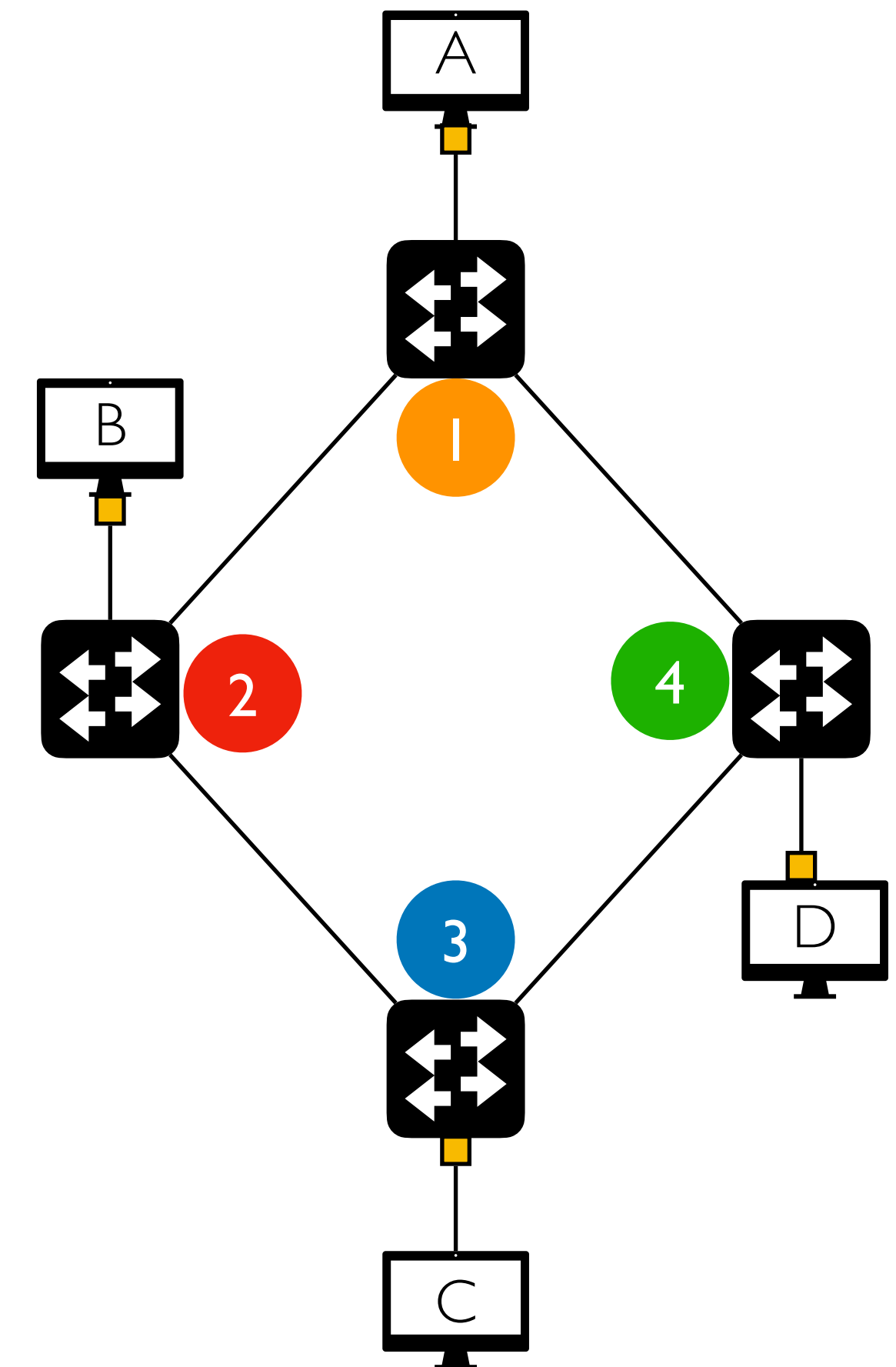
- Constraints (for backward compatibility)
 - No changes to end-hosts
 - Maintain plug-n-play aspect
- Earlier Ethernet achieved plug-n-play by leveraging a broadcast medium
 - *Can we do the same in a switched topology?*

“Broadcast” packets on all interfaces except the interface you receive the packet from

Flooding with loops (still) leads to Broadcast Storm!

Example: A wants to broadcast a message

- A sends packet to I
- I Floods to 2 and 4
- 2 Floods to B and 3, 4 Floods to D and 3
- 3 Floods packet from 2 to C & 4, 3 Floods packet from 4 to C & 2
- 4 Floods packet from 3 to D & I, 2 Floods packet from 3 to B & I
- I Floods packet from 2 to A & 4, I Floods packet from 4 to B & 2
- ...
- A “broadcast storm” if the network contains a cycle of switches



Questions?

Spanning Tree Approach

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

Algorithm Has Two Aspects

- **Pick a root:**
 - Destination to which shortest paths go
 - Pick the one with the smallest identifier (MAC address)
- **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 (simultaneously) 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, but this is an easy one to remember and implement

Constructing a Spanning Tree

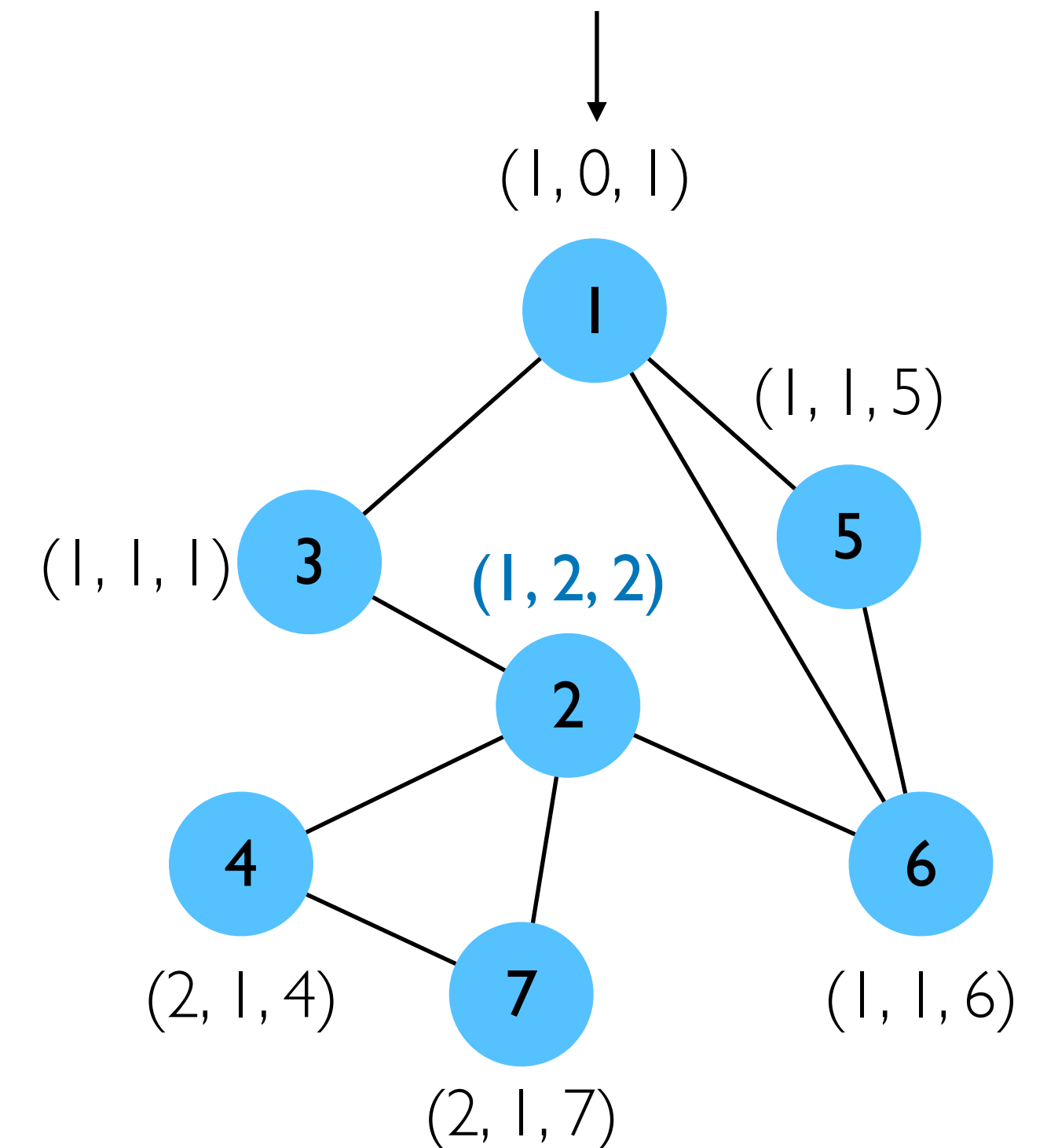
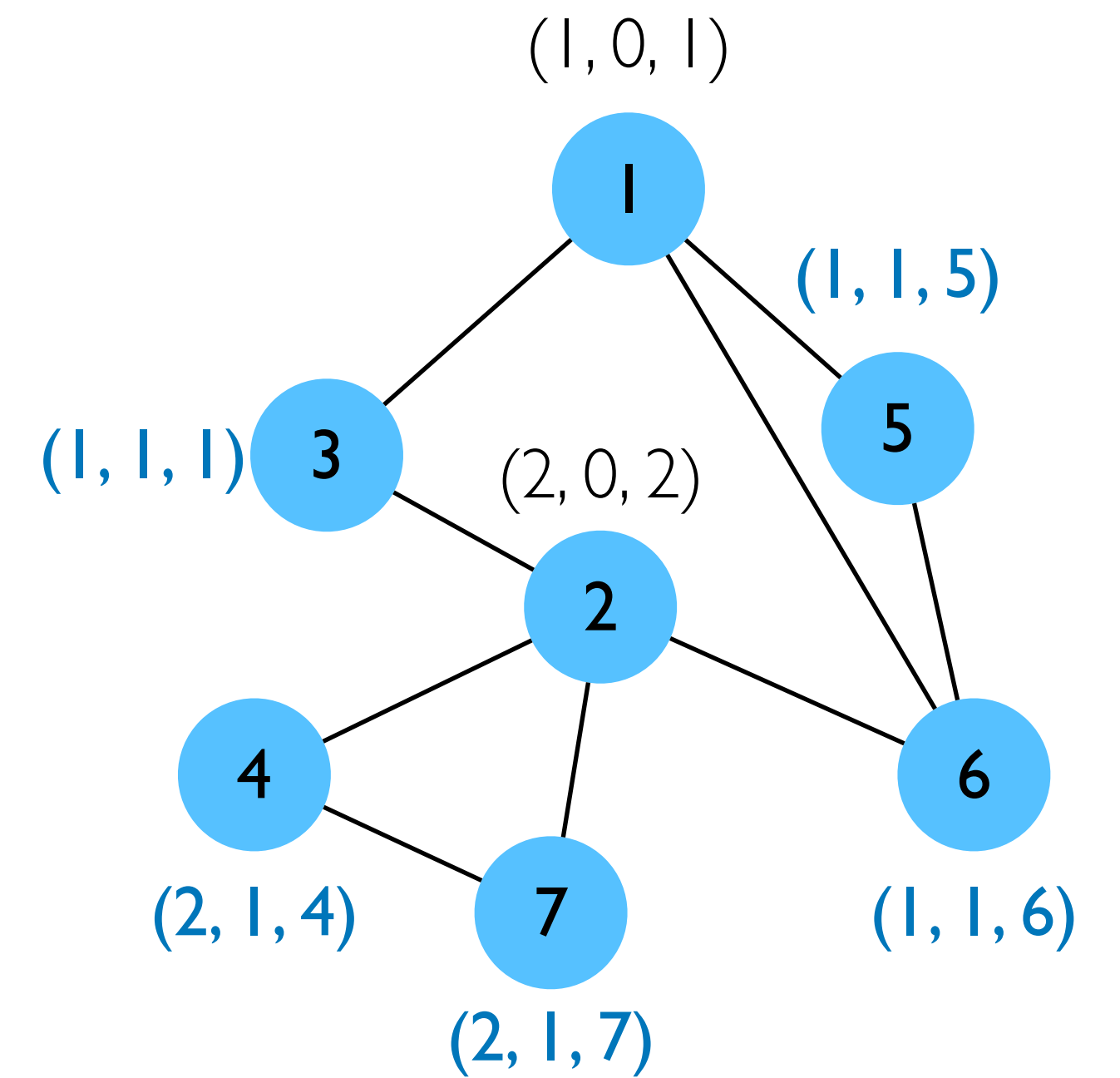
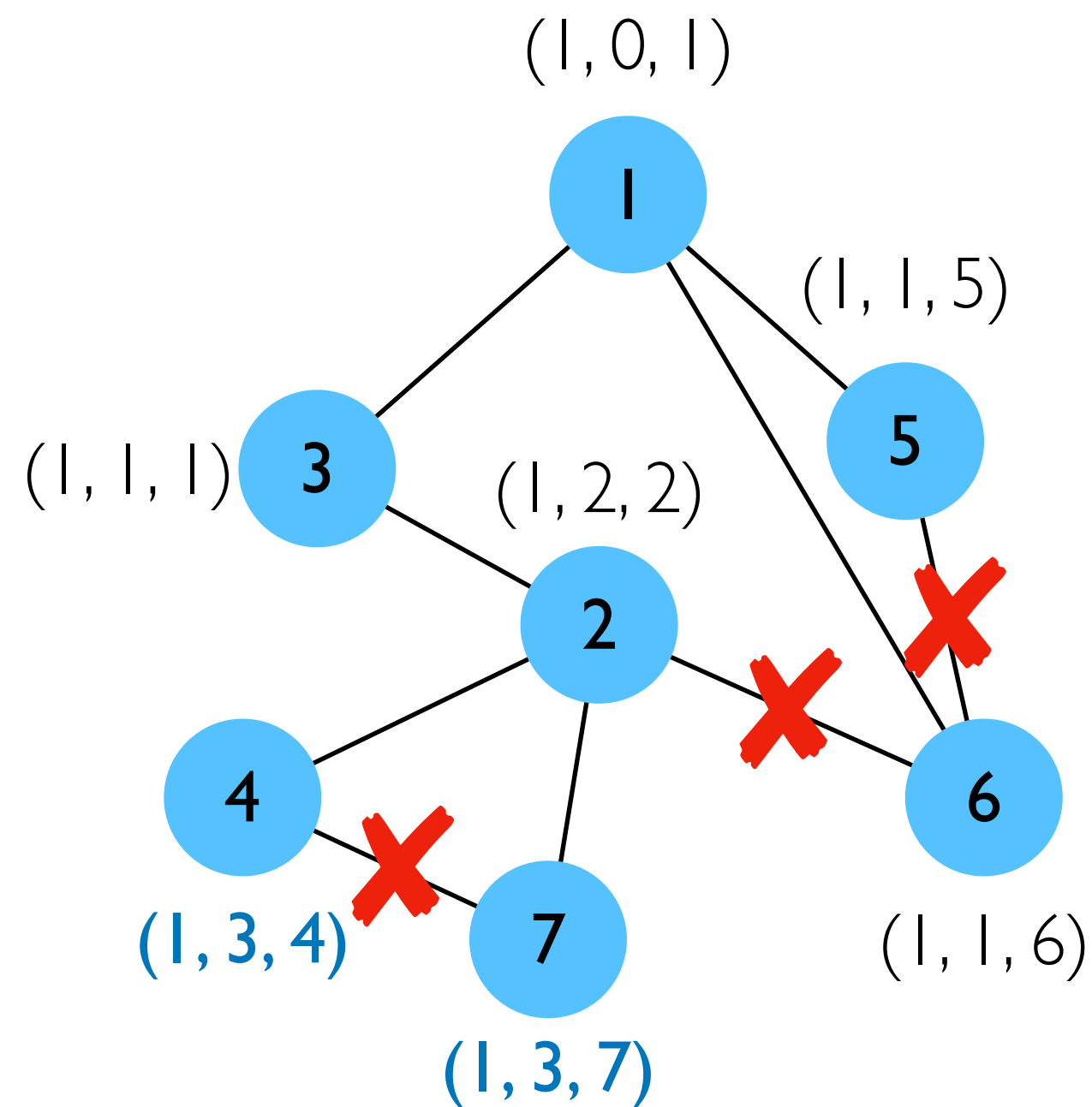
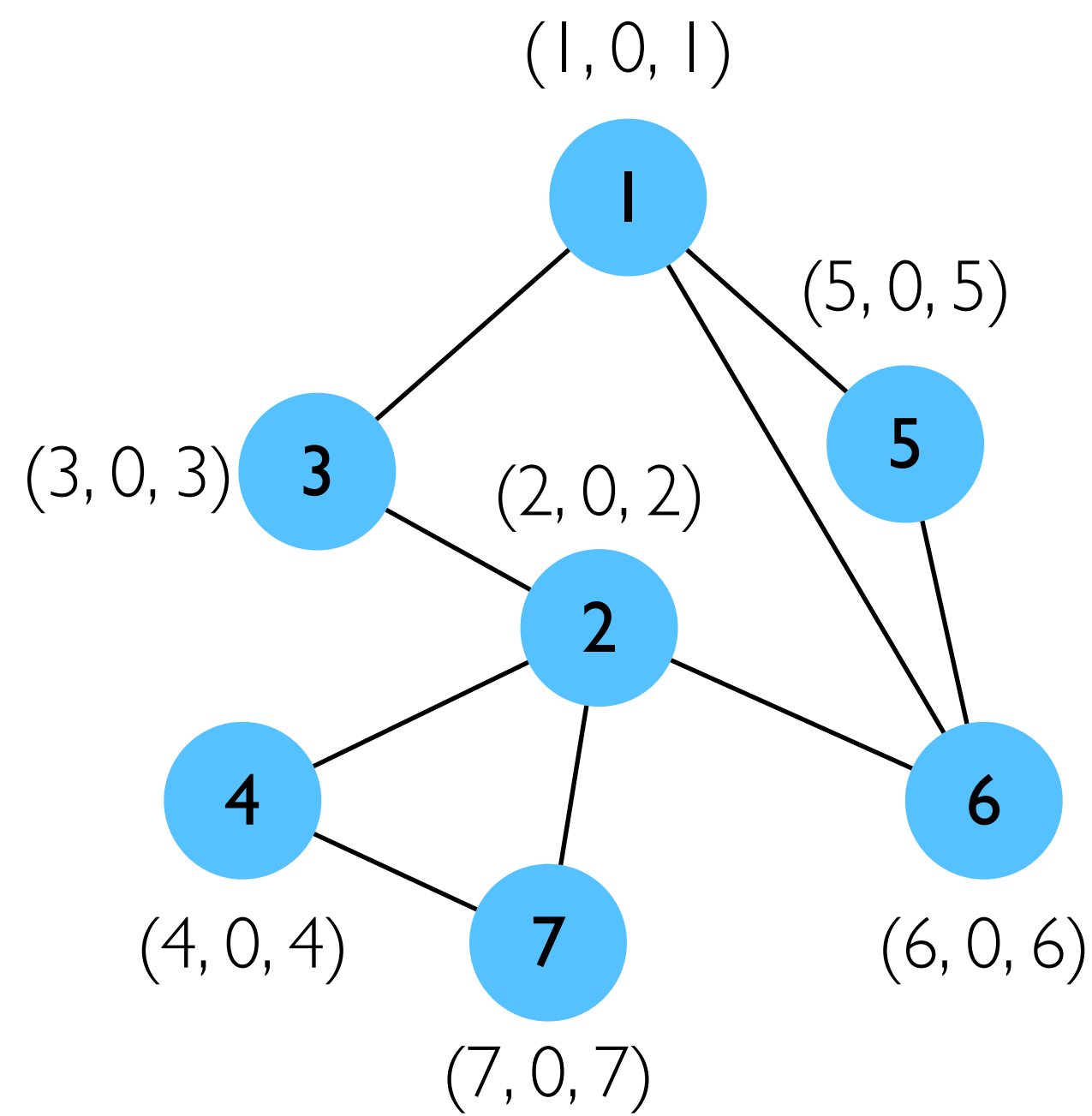
- **Message(Y, d, X) \leftarrow (Root, Distance, From)**
 - From node X
 - Proposing Y as the root
 - And advertising a distance d to Y
- **Switches elect the node with the smallest identifier (MAC address) as root**
 - Y in the messages
- **Each switch determines if a link is on its shortest path to 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 has changed, send neighbors updated message $(Y, d+1, X)$

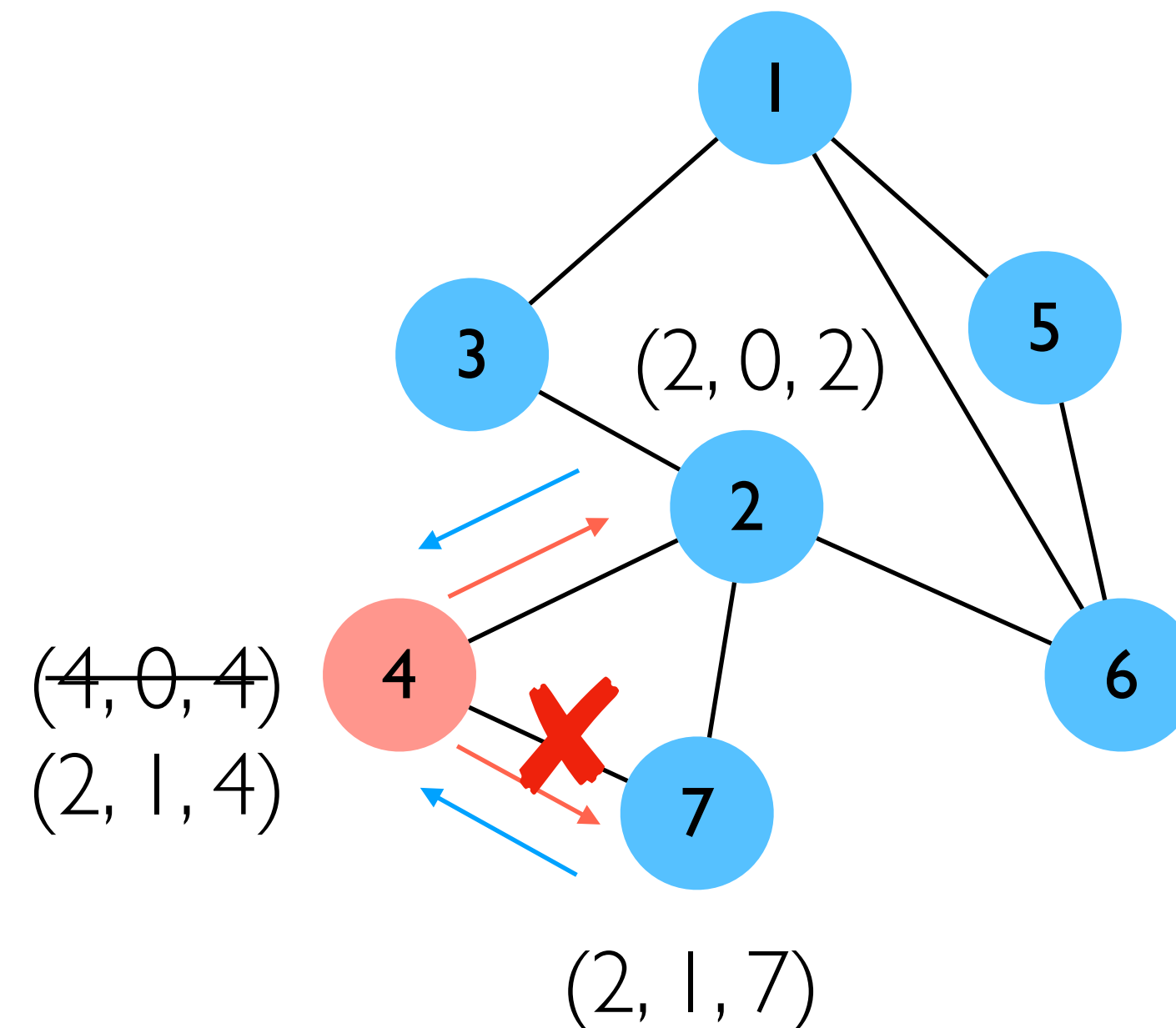
Example

(Root, dist, from)



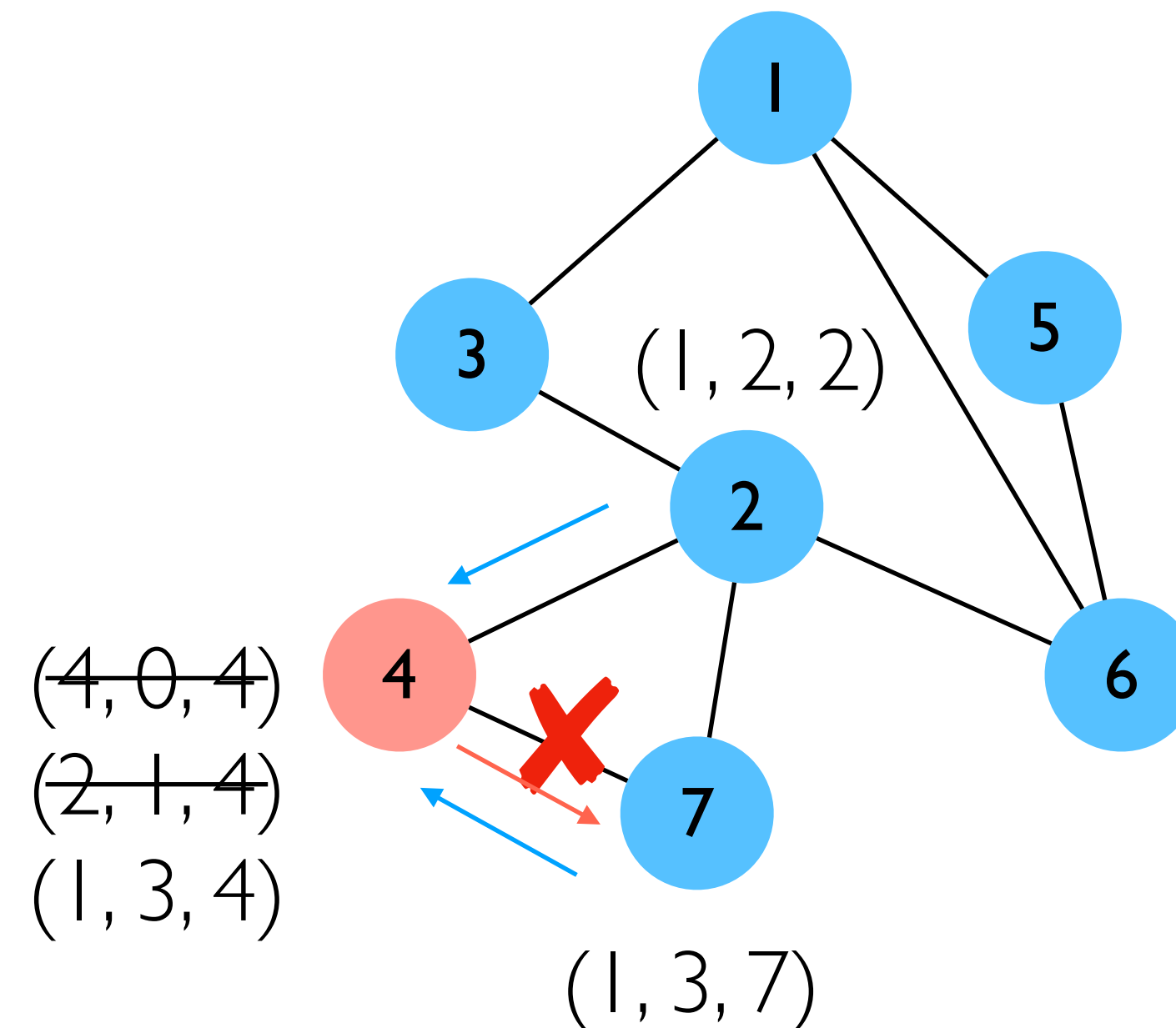
Example From Switch #4's Viewpoint

- 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



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, *timeout and claim to be the root!*

Questions?

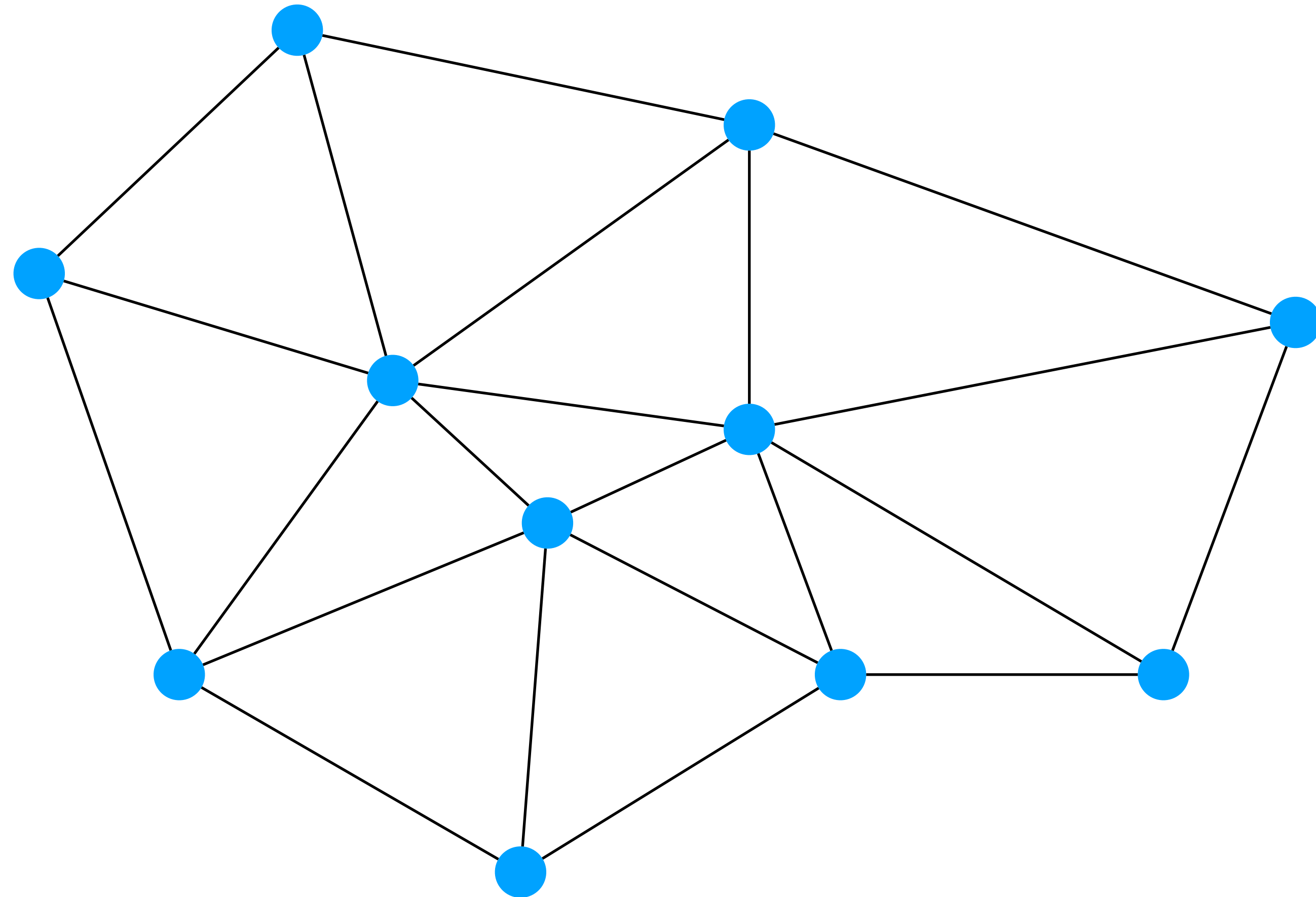
Outline

- Frames and framing
- Addressing
- Routing
- **Forwarding**
- **Discovery: Bootstrapping end-to-end communication**

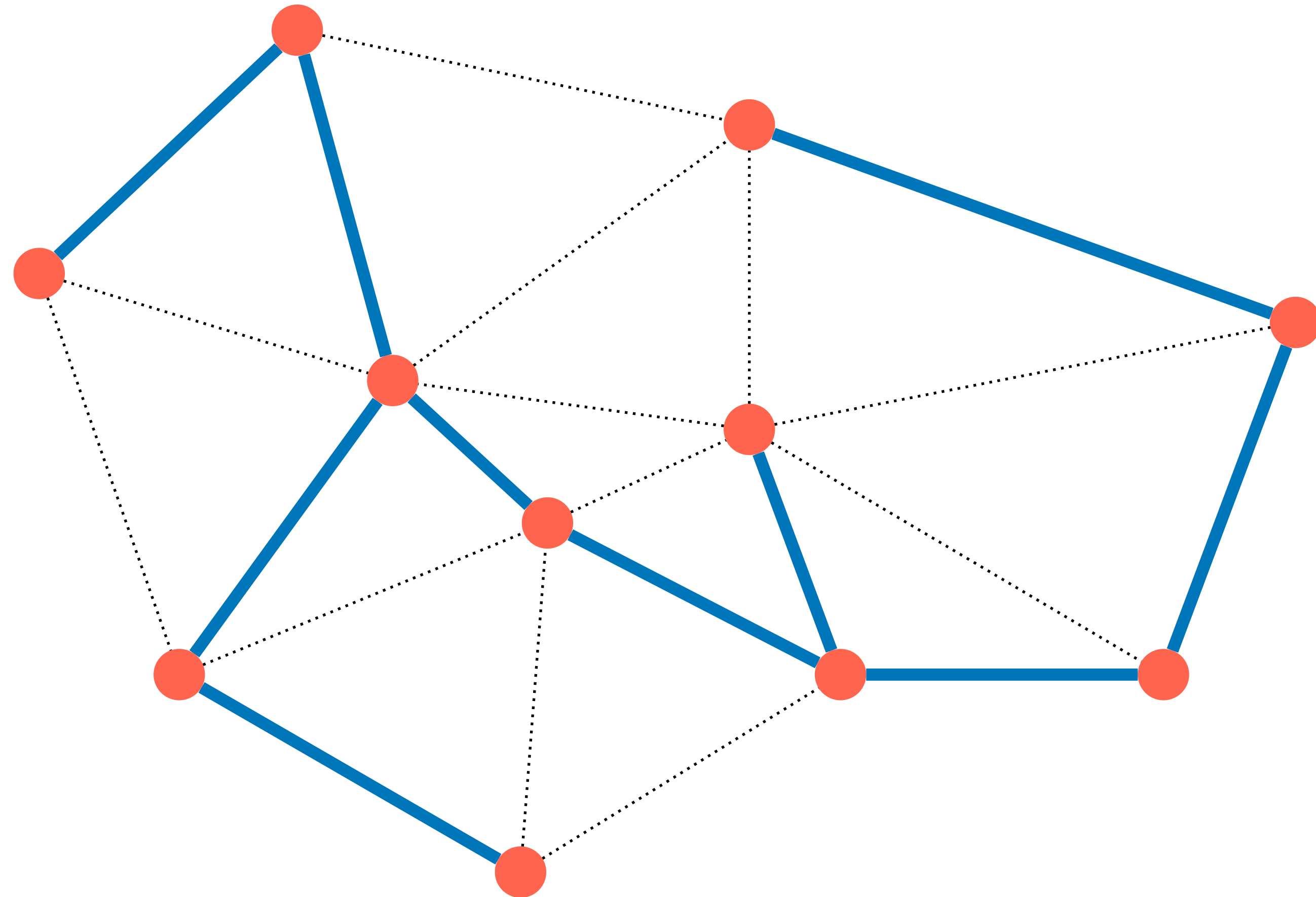
Flooding on a Spanning Tree

- Switches flood using the following rule:
 - *(Ignoring all ports not on the spanning tree!)*
 - Originating switch sends packet out on all ports
 - When a packet arrives on one incoming port, send it out on all ports other than the incoming port

Flooding on Spanning Tree



Flooding on Spanning Tree



But isn't flooding wasteful?

- Yes, but we can use it to bootstrap more efficient forwarding
- Idea: watch the packets going by, and *learn* from them
 - If node A sees a packet from node B come in on a port, ***it knows what port to use to reach B!***
 - Works because there's only one path to B

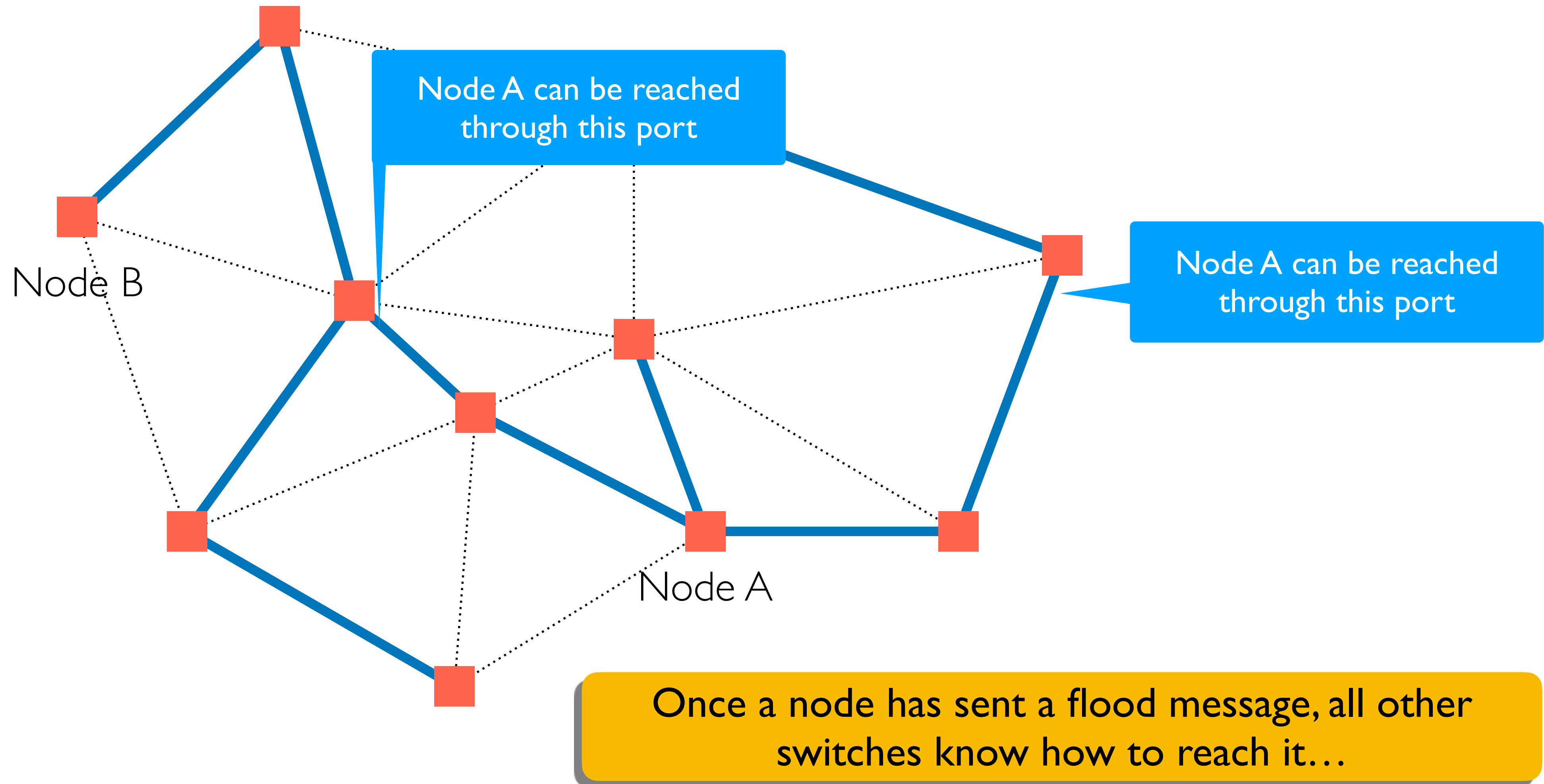
Nodes can “learn” routes

- Switch learns how to reach nodes by remembering where flooding packets came from
 - If flood packet from Node A entered switch on port 4, then switch uses port 4 to send to Node A

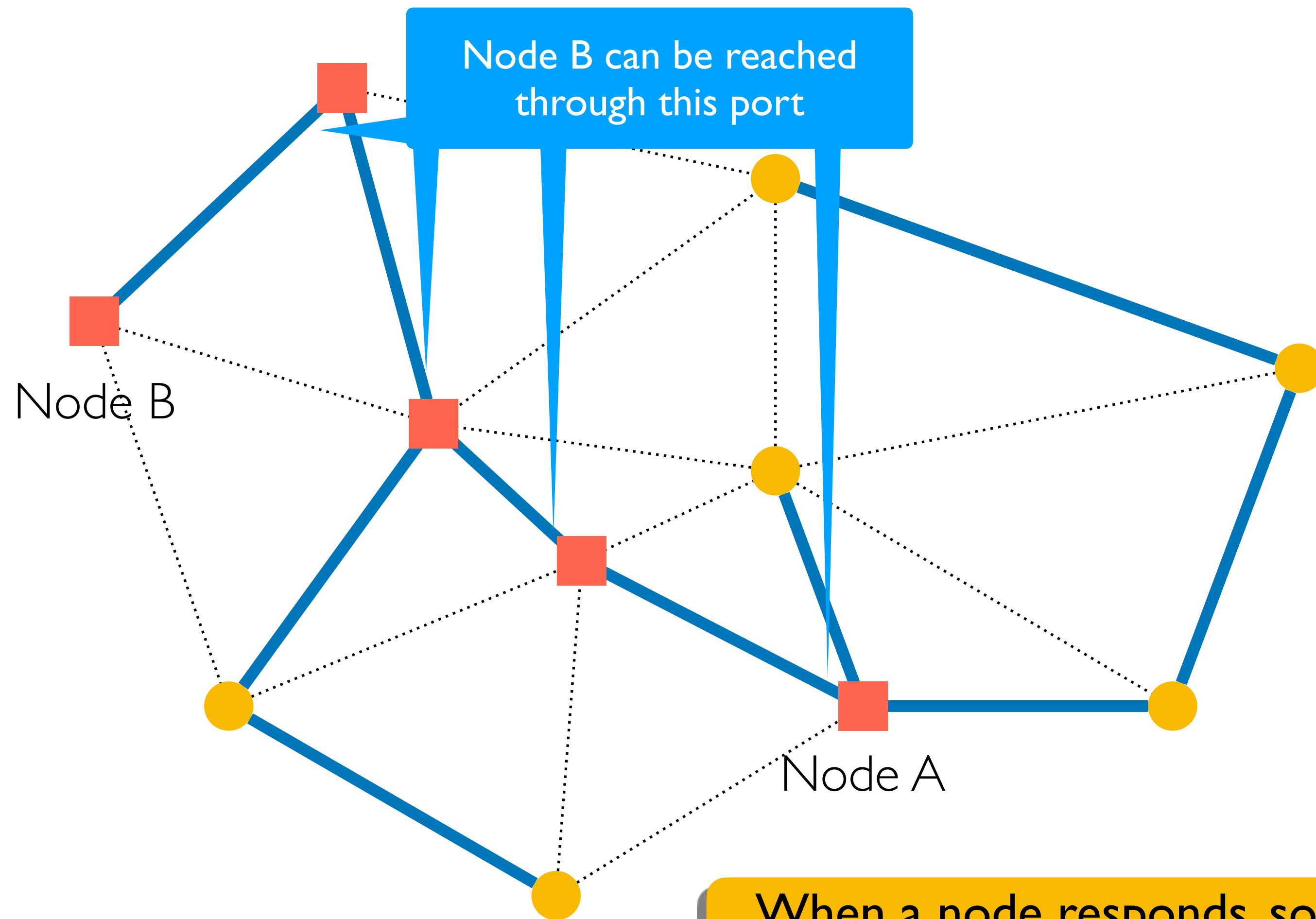
General Approach

- Flood first packet to node you are trying to reach
- All switches learn where you are
- When destination responds, some switches learn where it is...
 - Only some switches, because packet to you follows direct path, and is not flooded

Learning from Flood Packets



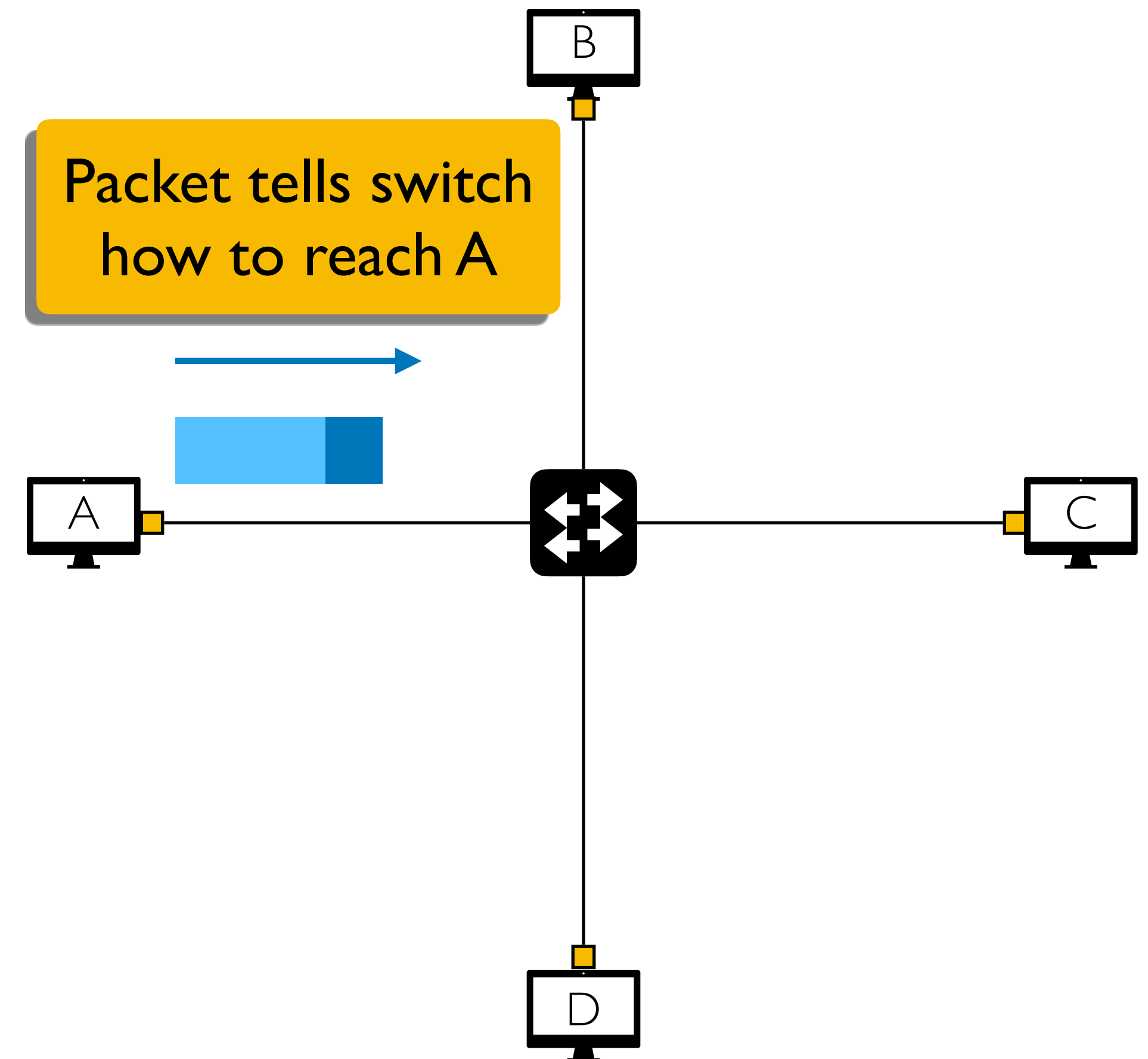
Learning from Flood Packets



Ethernet switches are “self learning”

When a packet arrives:

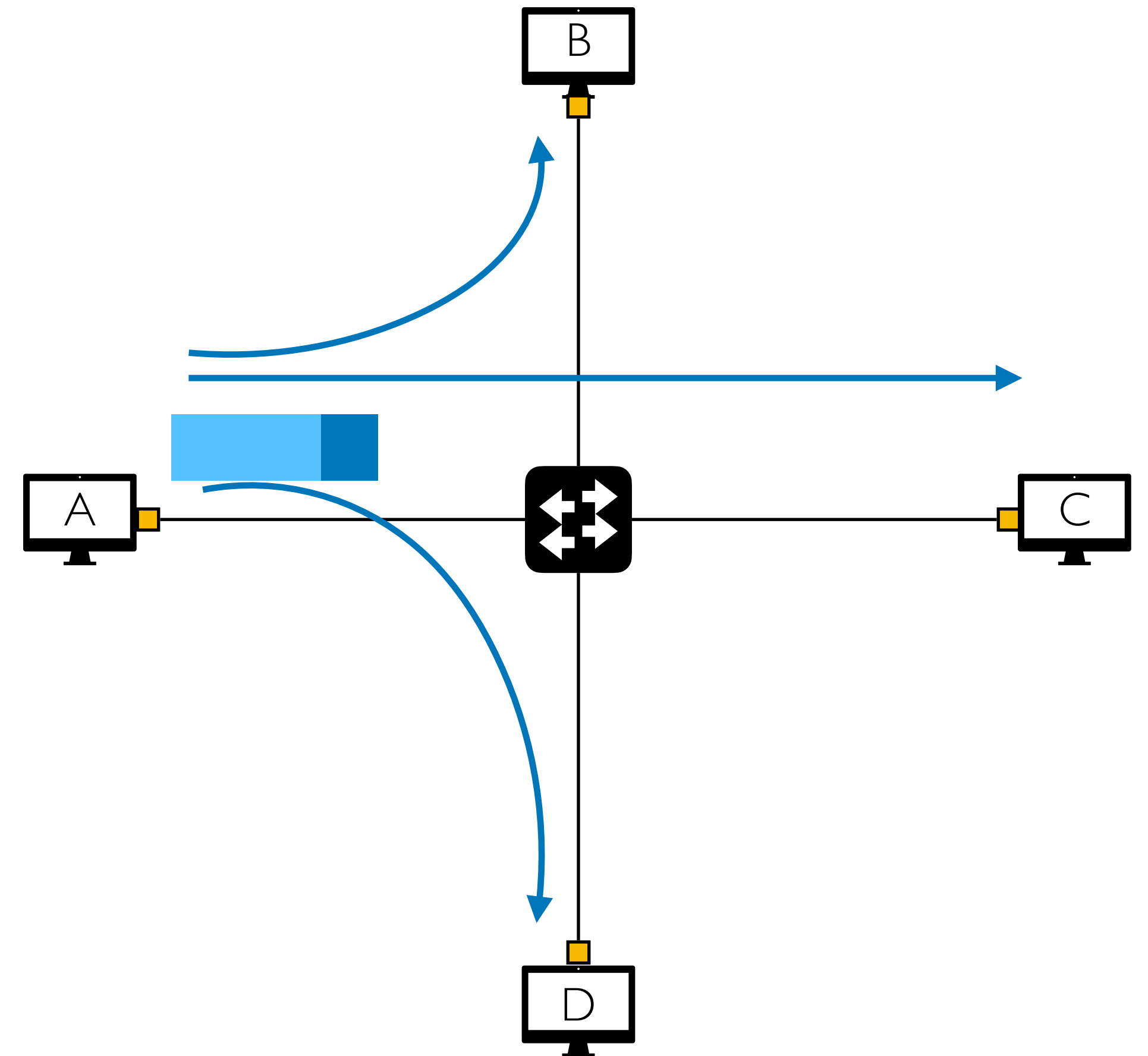
- Inspect *source* MAC address, associate with incoming port
- Store mapping in the switch table
- Use *time-to-live* field to eventually forget mapping



Self Learning: Handling Misses

When a packet arrives with unfamiliar destination:

- Forward packet out all other ports
- Response may teach switch about that destination



Summary of Learning Approach

- Avoids loop by restricting to spanning tree
 - This makes flooding possible
- Flooding allows packet to reach destination
 - And in the process switches learn how to reach the source of flood
- No route “computation”
 - Forwarding entries a consequence of traffic pattern

Questions?

Contrast: IP vs. Ethernet

- **IP**
 - Packets forwarded on all links
 - Aggregable addresses
 - Routing protocol computes loop-free *paths*
 - Forwarding table computed by routing protocol
- **Ethernet**
 - Packets forwarded on subset of links (spanning tree)
 - Flat addresses
 - “Routing” protocol computes loop-free *topology*
 - Forwarding table derived from data packets (+SPT floods)

Strengths of Ethernet's Approach

- Plug-n-Play
- Simple

Weaknesses of This Approach

- **Much of the network bandwidth goes unused**
 - Forwarding is only over the spanning tree
- **Delay in reestablishing spanning tree**
 - Network is “down” until spanning tree rebuilt
 - And rebuilt spanning tree might be quite different
- **Slow to react to host movement**
 - Entries must time out
- **Poor predictability**
 - Location of root and traffic patterns determines forwarding efficiency

Outline

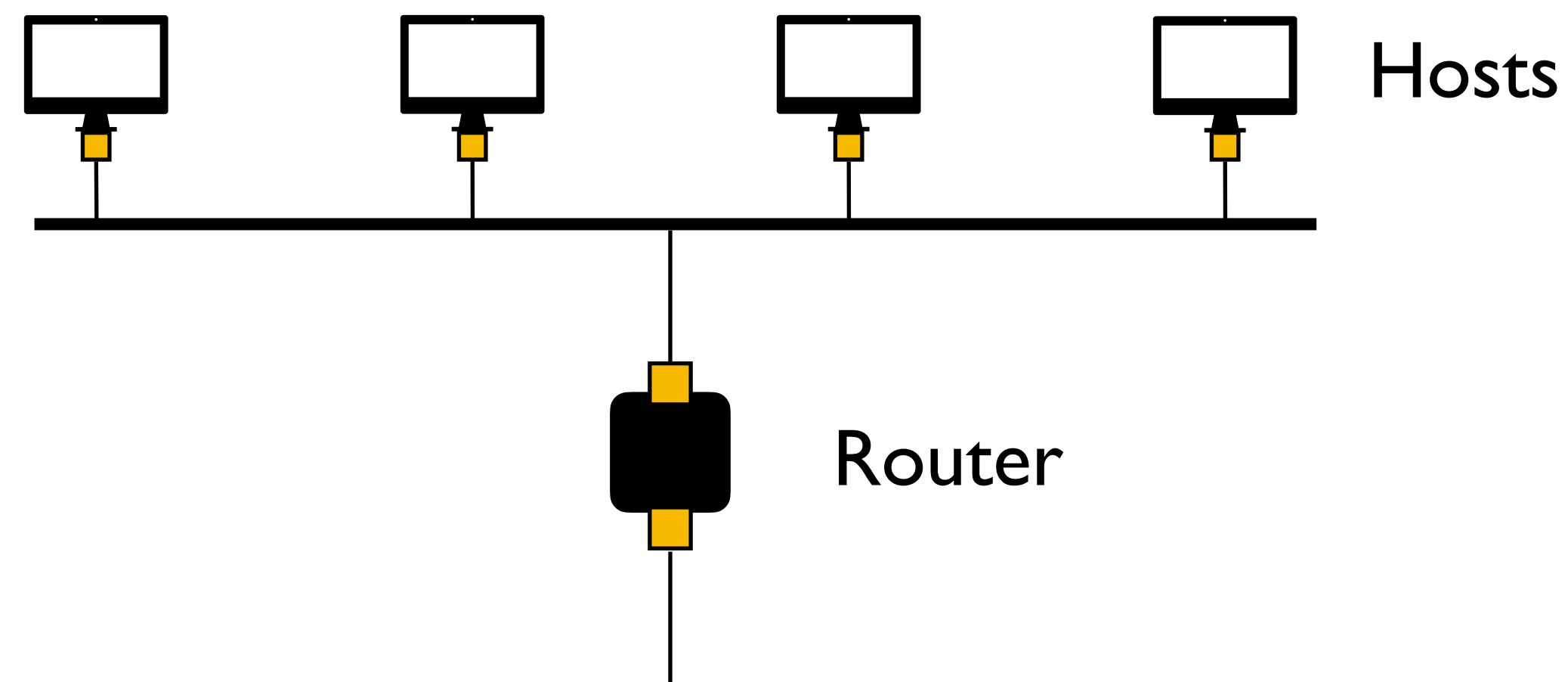
- Frames and framing
- Addressing
- Routing
- Forwarding
- **Discovery: Bootstrapping end-to-end communication**

Discovery

- A host is “born” knowing only its MAC address
- Must discover lots of information before it can communicate with a remote host B
 - What is my IP address?
 - What is B's IP address?
 - What is B's MAC address? (if B is local)
 - What is my first-hop router's MAC address? (if B is not local)
 - ...

ARP & DHCP

- Link layer discovery protocols
 - ARP → Address Resolution Protocol
 - DHCP → Dynamic Host Configuration Protocol
 - Confined to a single Local Area Network (LAN)
 - Rely on broadcast capability



ARP & DHCP

- Link layer discovery protocols
- Serve two functions
 - Discovery of local end-hosts
 - For communication between hosts on the same LAN

ARP & DHCP

- Link layer discovery protocols
- Serve two functions
 - Discovery of local end-hosts
 - Bootstrap communication with remote hosts
 - What's my IP address?
 - Who/where is my local DNS server?
 - Who/where is my first hop router?

Questions?

DHCP

- “Dynamic Host Configuration Protocol”
 - Defined in RFC 2131
- **A host uses DHCP to discover**
 - Its own IP address
 - Its netmask
 - IP address(es) for its local DNS name server(s)
 - IP address(es) for its first-hop “default” router(s)

DHCP: Operation

- I. One or more local DHCP servers maintain required information
 - IP address pool, netmask, DNS servers, etc.
 - Application that listens on UDP port 67

DHCP: Operation

1. One or more local DHCP servers maintain required information
2. Client *broadcasts* a DHCP discovery message
 - L2 broadcast, to MAC address FF:FF:FF:FF:FF:FF

DHCP: Operation

1. One or more local DHCP servers maintain required information
2. Client *broadcasts* a DHCP discovery message
3. One or more DHCP servers responds with a DHCP “offer” message
 - Proposed IP address for client, lease time
 - Other parameters

DHCP: Operation

1. One or more local DHCP servers maintain required information
2. Client *broadcasts* a DHCP discovery message
3. One or more DHCP servers responds with a DHCP “offer” message
4. Client *broadcasts* a DHCP request message
 - Specifies which offer it wants
 - Echoes accepted parameters
 - Other DHCP servers learn they were not chosen

DHCP: Operation

1. One or more local DHCP servers maintain required information
2. Client *broadcasts* a DHCP discovery message
3. One or more DHCP servers responds with a DHCP “offer” message
4. Client *broadcasts* a DHCP request message
5. Selected DHCP server responds with an ACK

DHCP: Operation

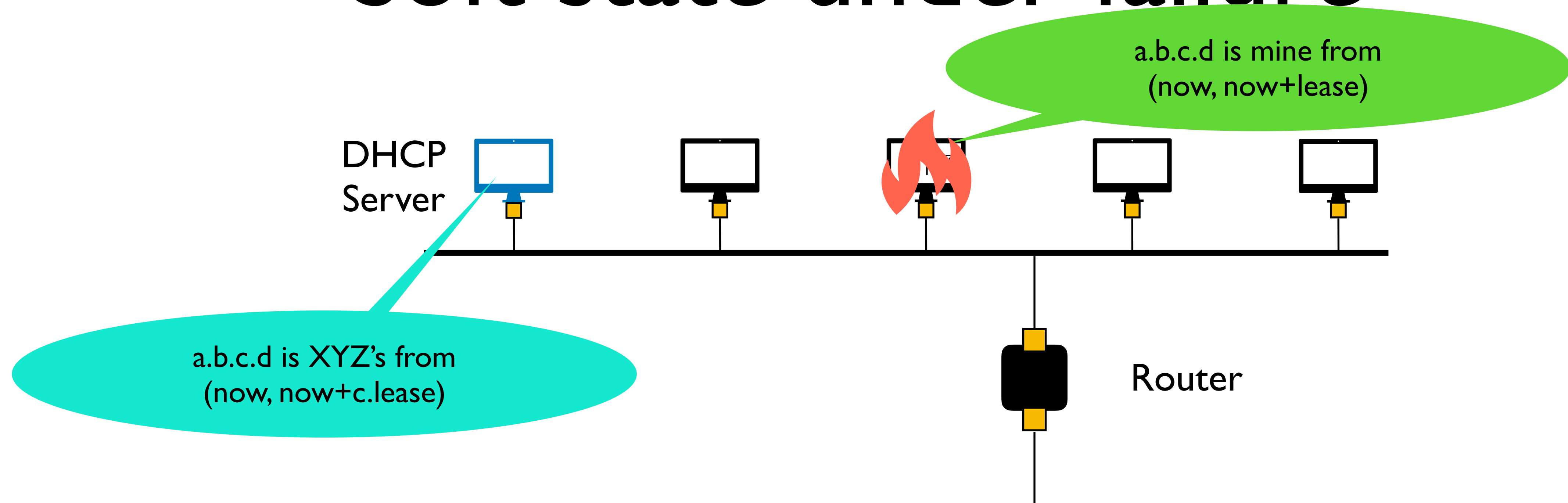
1. One or more local DHCP servers maintain required information
2. Client *broadcasts* a DHCP **discovery** message
3. One or more DHCP servers responds with a DHCP “**offer**” message
4. Client *broadcasts* a DHCP **request** message
5. Selected DHCP server responds with an **ACK**

DHCP “relay agents” used when the DHCP server isn’t on the same broadcast domain — see text!

DHCP uses “soft state”

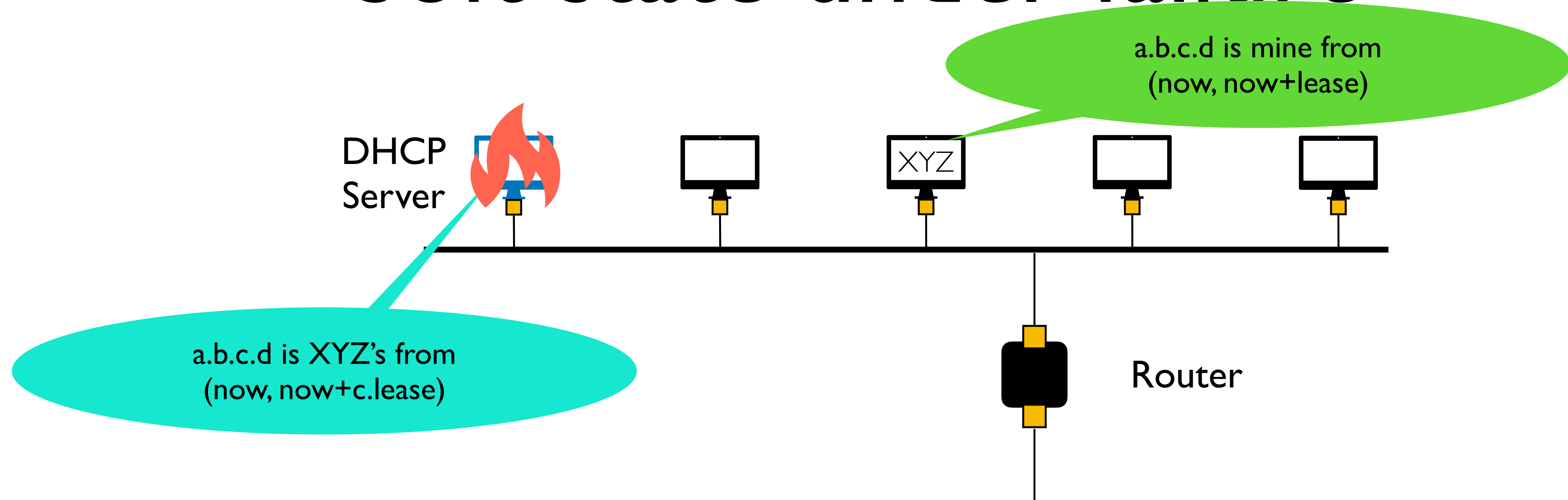
- **Soft State:** if not refreshed, state is forgotten
 - *Hard state:* allocation is **deliberately** returned/withdrawn
 - e.g., used to track address allocation in DHCP
- **Implementation:**
 - Address allocations are associated with a lease period
 - Server: sets a timer associated with the record of allocation
 - Client: must request a renewal before lease period expires
 - Server: resets timer when a renewal arrives; sends ACK
 - Server: reclaims allocated address when timer expires
- **Simple, yet robust under failure**
 - State always fixes itself within (small constant of) lease time

Soft state under failure



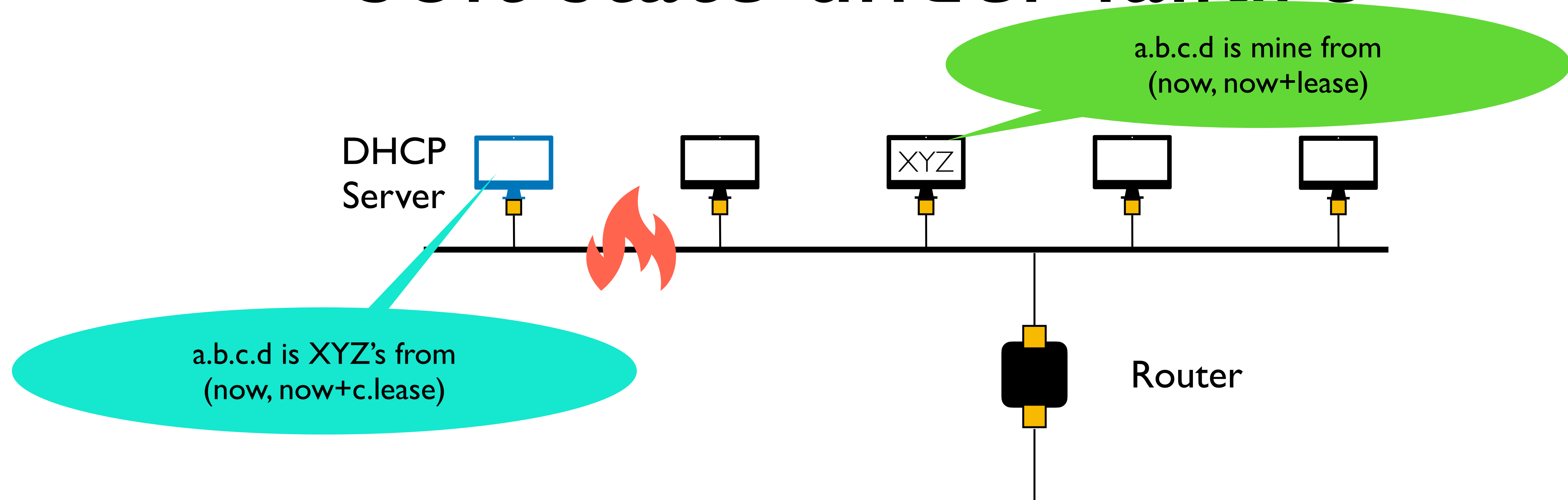
- What happens when host XYZ fails?
 - Renewals from XYZ stop
 - DHCP server reclaims a.b.c.d after $O(\text{lease period})$

Soft state under failure



- What happens when host DHCP Server fails?
 - ACKs from DHCP Server stop
 - XYZ releases address after $O(\text{lease period})$; send new request
 - A new DHCP server can respond and we're back on track in $\sim \text{lease time}$

Soft state under failure



- What happens if the network fails?
 - Neither renewals nor ACKs get through
 - Independently: XYZ releases address; DHCP server reclaims it

Questions?