Ethernet

CS/ECE 438: Spring 2014

Instructor: Matthew Caesar

http://courses.engr.illinois.edu/cs438/

Some History

- Ethernet was invented as a broadcast technology
  - Each packet received by all attached hosts
- Easy to set up, cheap to build
  - But hosts had to share channel (multiple access)
- Current Ethernets are “switched”
  - No sharing
- But need spanning tree to route on switches
  - Everyone hates spanning tree, trying to eliminate it

Today

- Study two algorithms that are “endangered”
  - But both important conceptually!
- Spanning Tree
  - Still used, but alternatives being developed
- Multiple Access in wired media (largely extinct)
  - Rarely used, but useful background for wireless

Ethernet: Key Concepts

Overview of Ethernet

- Dominant wired LAN technology
  - Pretty much obsoleted token ring, optical LANs, ATM
- Defines a spectrum of techniques
  - Physical wiring, contention resolution (CSMA/CD), framing, encoding, devices (hubs/switches/bridges), forwarding, addressing

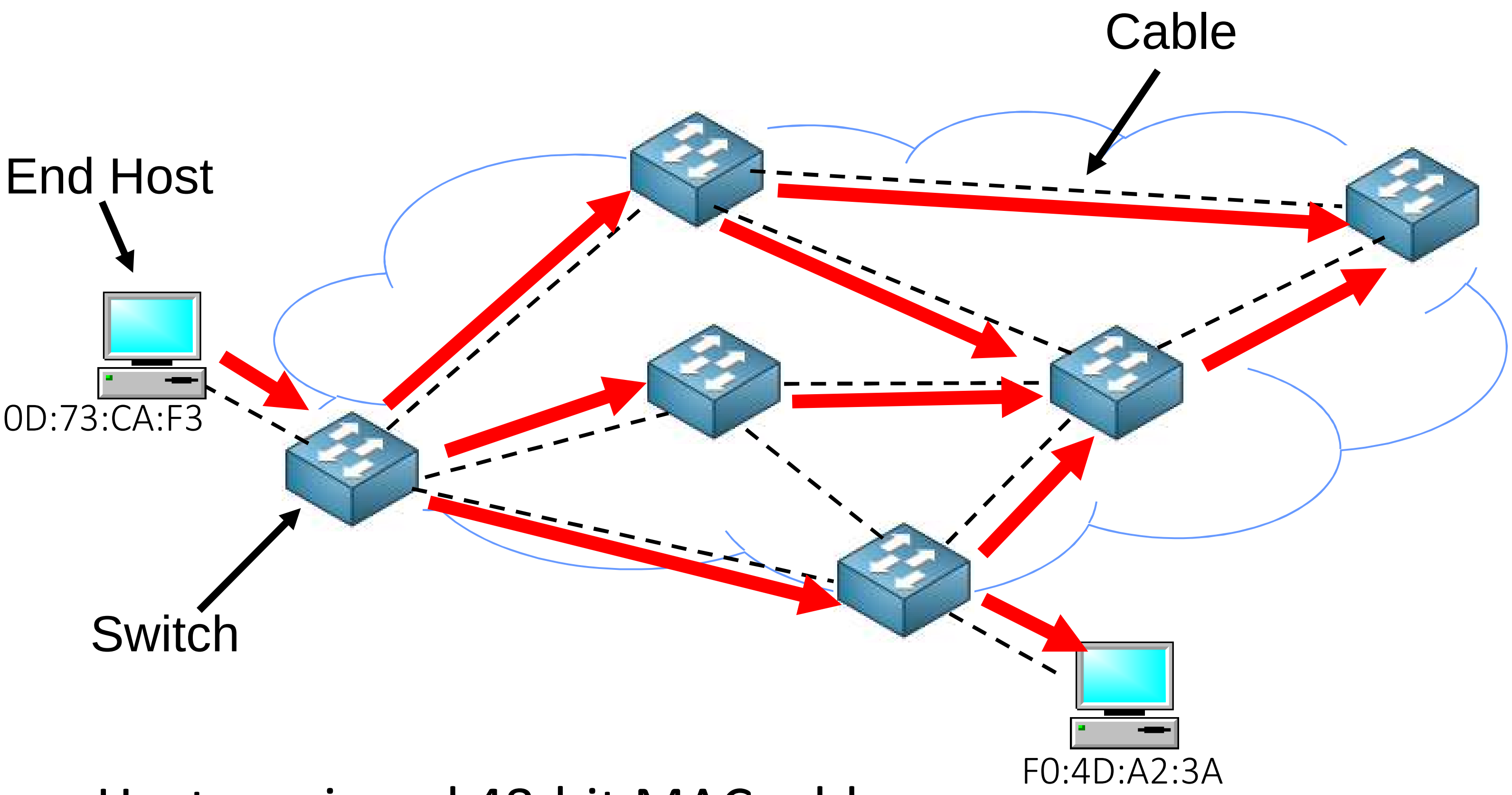
Overview of Ethernet

- Ethernet uses CSMA/CD
  - Carrier sense, collision detection, random access
- Limitations on Ethernet length
  - Need to ensure collisions are detected before sender is done transmitting a packet
- Frame structure
  - Preamble for synchronization

Overview of Ethernet

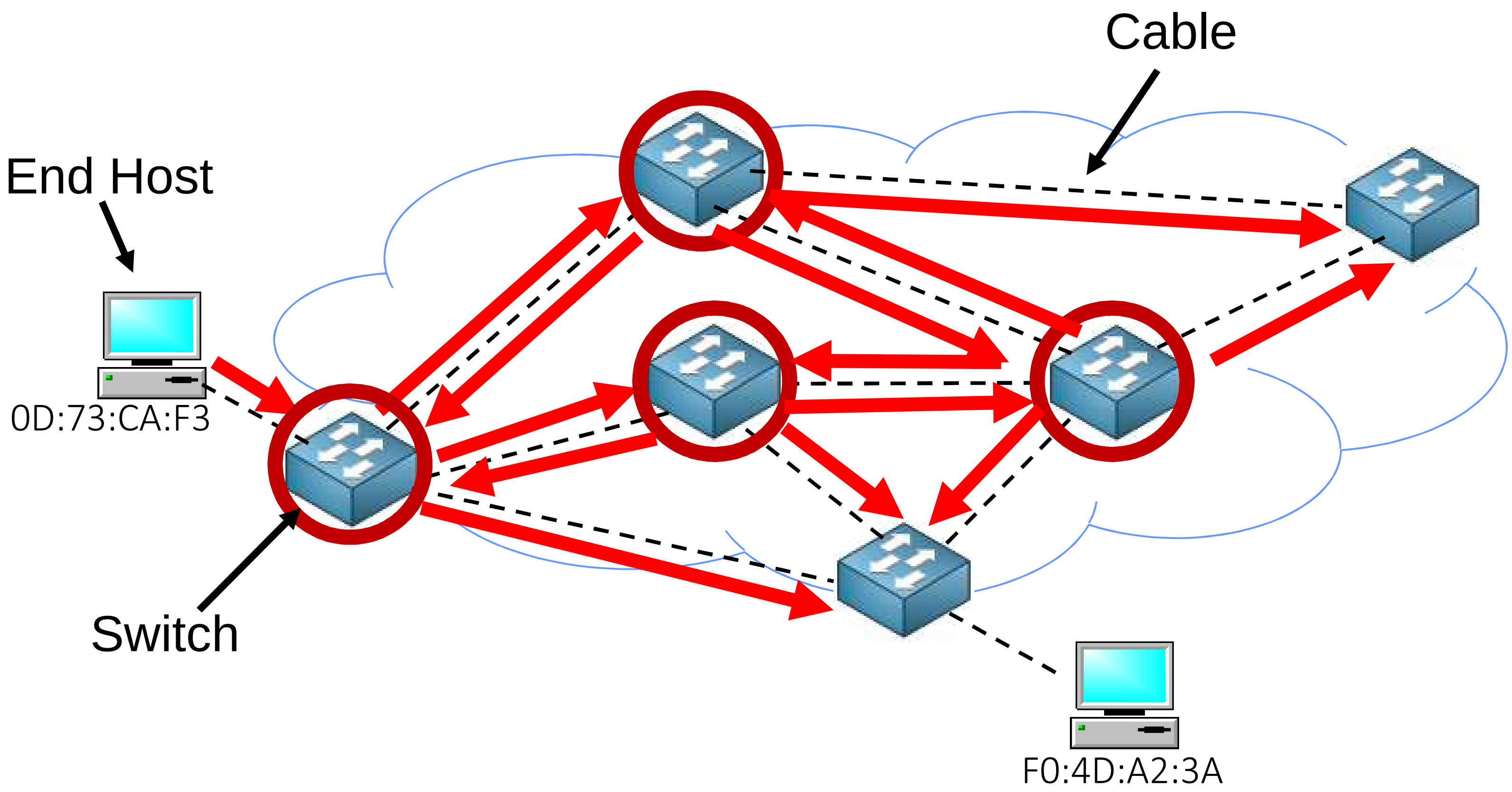
- Device types
  - Hubs: physical layer repeaters (obsolete?)
  - Switch: store and forward, breaks subnet into isolated LAN segments, learning
- Semantics: Unreliable, Connectionless
- Benefits: easy to administer and maintain, plug-and-play
- Downsides: scaling, security

Ethernet Forwarding



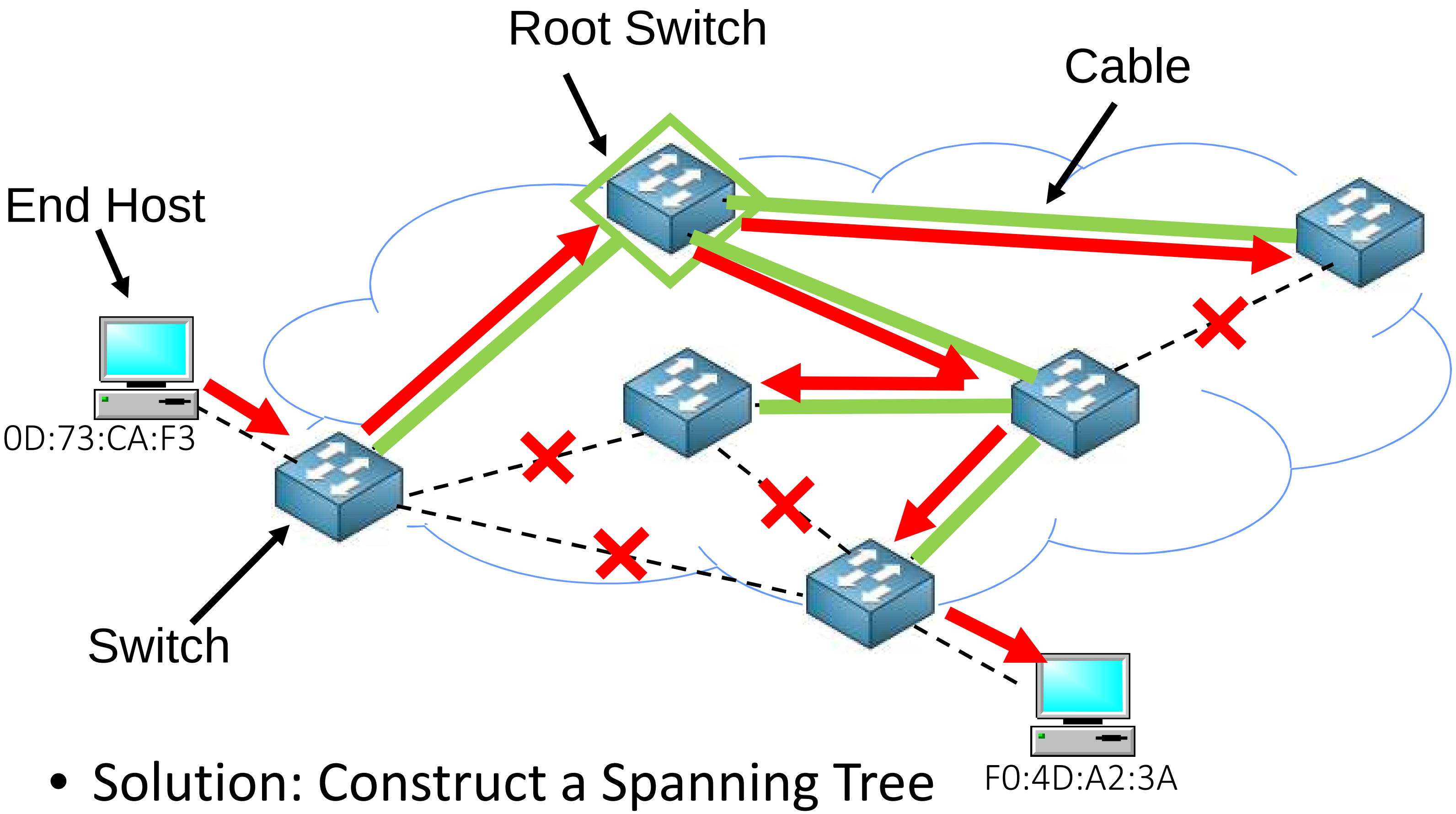
- Hosts assigned 48-bit MAC addresses
- Forwarding by “flooding”

Ethernet Forwarding



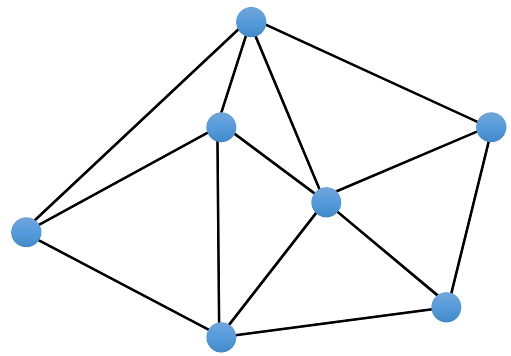
- How to flood with stateless switches?

Ethernet Forwarding



- Solution: Construct a Spanning Tree
  - Elect a “root” switch
  - Root-facing ports are active, others disabled

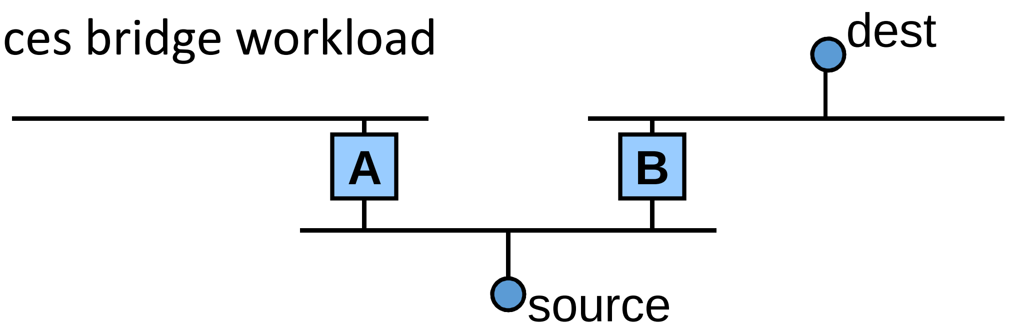
Avoiding Flooding



- Flooding packets throughout network introduces problems
  - Scalability, privacy, resource isolation, lack of access control
- Scalability requirement is growing very fast
  - Large enterprises: 50k end hosts
  - Data centers: 100k servers, 5k switches
  - Metro-area Ethernet: over 1M subscribers

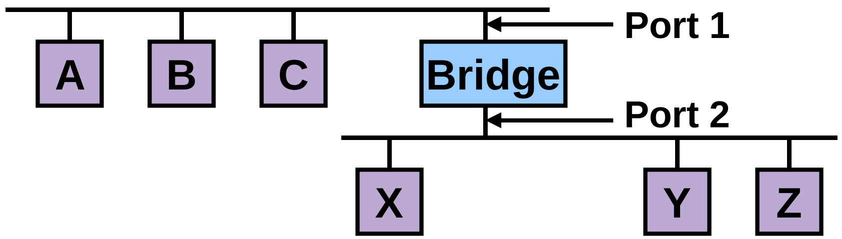
Avoiding Flooding

- Suppose source sends a frame to a destination
  - Which LANs should a frame be forwarded on?
- Trivial algorithm
  - Forward all frames on all (other) LAN’s
  - Potentially heavy traffic and processing overhead
- Optimize by using address information
  - “Learn” which hosts live on which LAN
  - Maintain forwarding table
  - Only forward when necessary
  - Reduces bridge workload



Learning Bridges

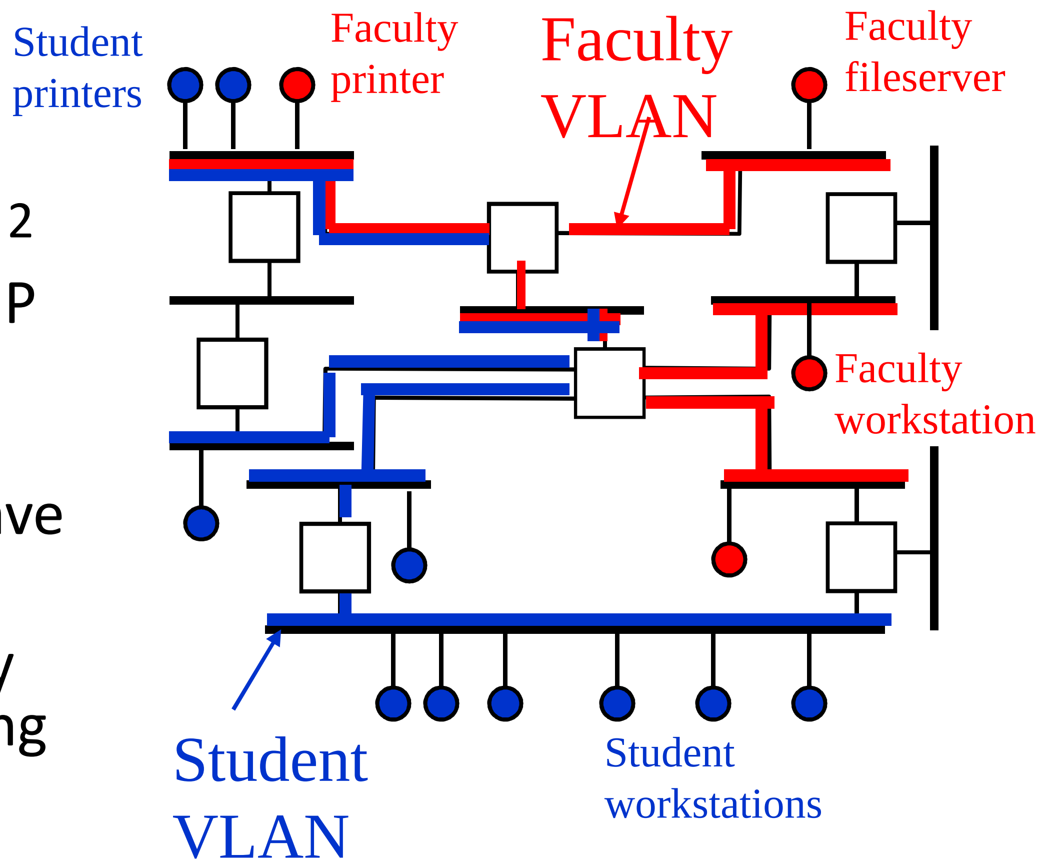
- Bridge learns table entries based on source address
  - When receive frame from A on port 1
    add A to list of hosts on port 1
  - Time out entries to allow movement of hosts
- Table is an "optimization", meaning it helps performance but is not mandatory
- Always forward broadcast frames



Host	Port
A	1
B	1
C	1
X	2
Y	2
Z	2

Scaling Ethernet with VLANs

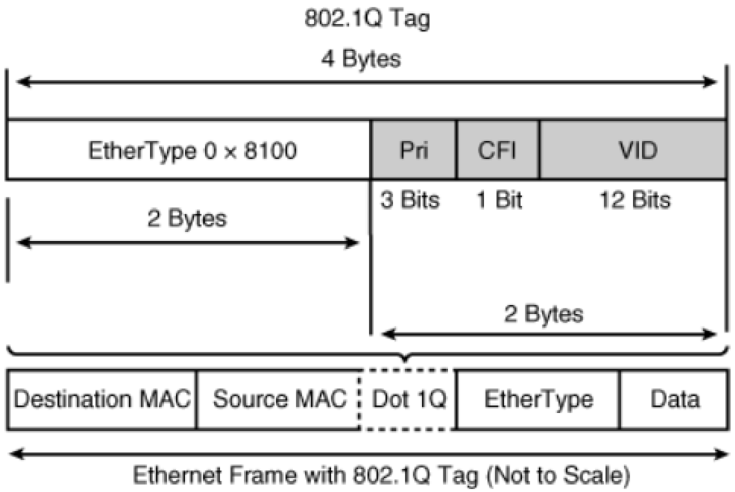
- Divide up hosts into logical groups called **VLANs**
  - VLANs isolate traffic at layer 2
- Each VLAN corresponds to IP subnet, single broadcast domain
- Ethernet packet headers have VLAN tag
- Bridges forward packet only on subnets on corresponding VLAN

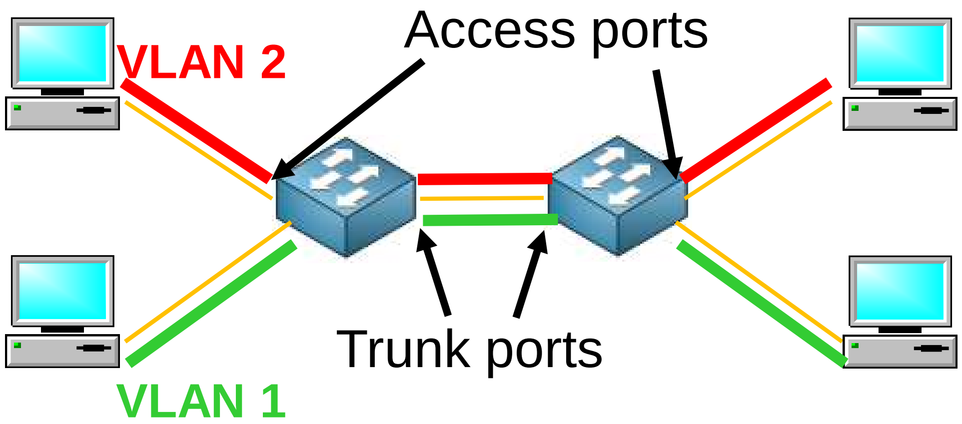


Virtual LANs

- Downsides of VLANs
  - Are manually configured, complicates network management
  - Hard to seamlessly migrate across VLAN boundaries due to addressing restrictions
- Upsides of VLANs
  - Limits scope of broadcasts
  - Logical separation improves isolation, security
  - Can change virtual topology without changing physical topology
    - E.g., used in data centers for VM migration

How VLANs are implemented

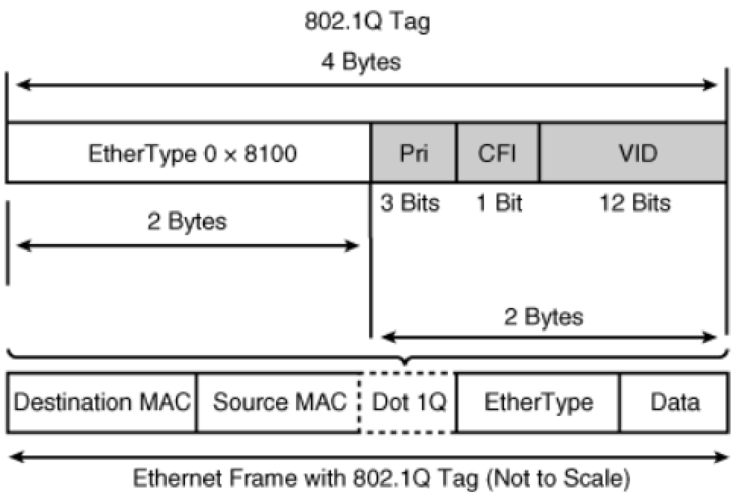


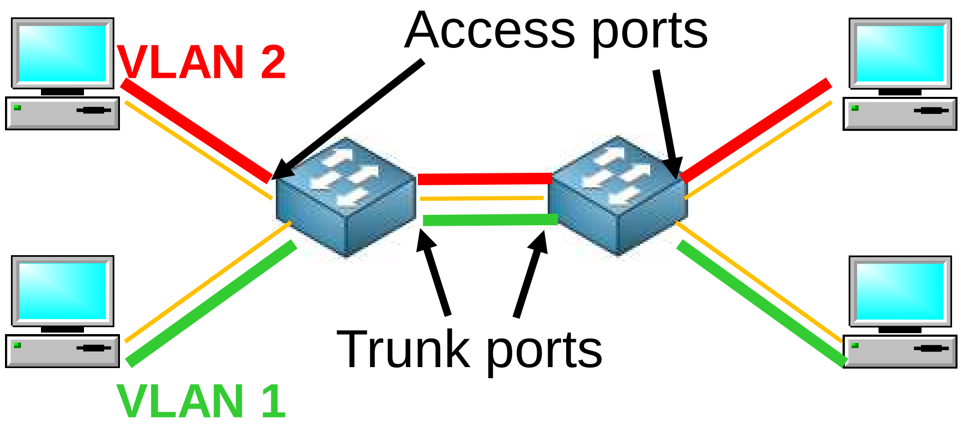


- Packets are annotated with 12-bit VLAN tags
  - Up to 4096 VLANs can be encapsulated within a single VLAN ID

- LAN switches can configure ports as access ports or trunk ports
  - Access ports append tags on packets
  - VLAN membership almost always statically encoded in access switch's configuration file
  - Trunk ports can multiplex several VLANs

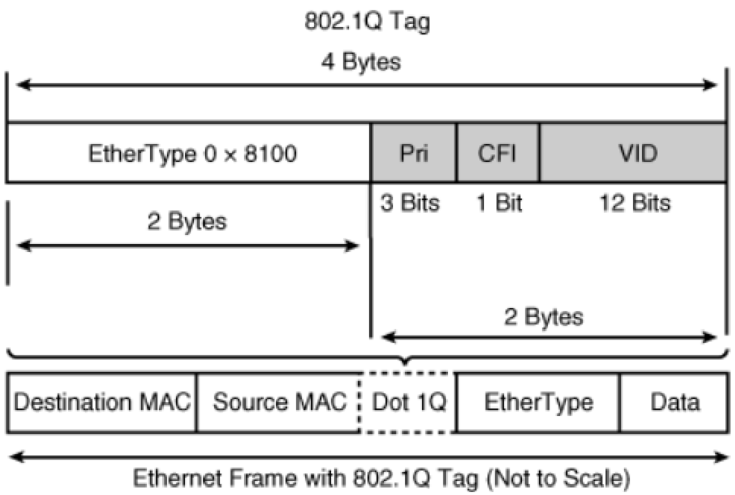
How VLANs are implemented

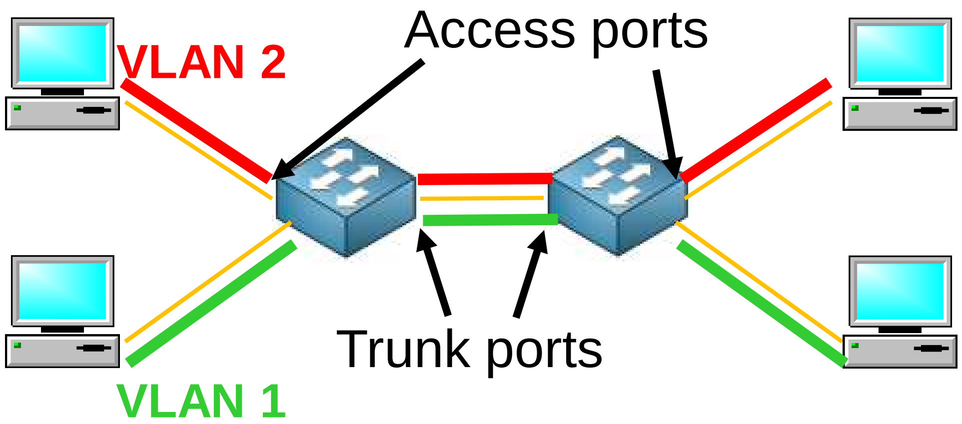




- 802.1Q (VLAN spec) defines a few other fields too
  - Ethertype of 0x8100 instructs switch to decode next 2 bytes as VLAN header
  - 3 bits of priority (like IP ToS)
  - 1 bit for compatibility with token ring
- What if 4096 VLANs isn't enough?
  - QinQ (802.1ad) – can encapsulate VLANs within VLANs by stacking VLAN tags
  - Up to 4096 VLANs can be multiplexed within a single VLAN ID→ 4096^2 combinations

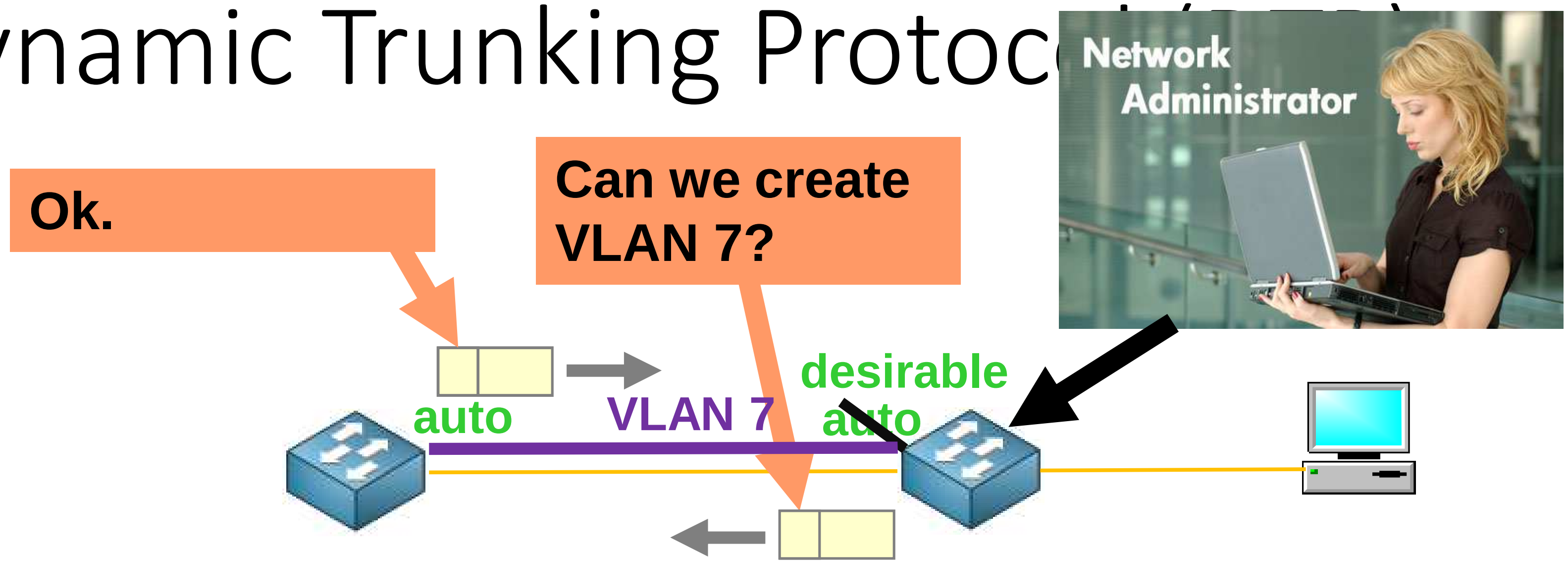
How VLANs are implemented





- Native mode
  - IEEE likes to make specs that are backwards compatible
  - 802.1Q allows trunk ports to carry both tagged and untagged frames
  - Frames with no tags are said to be part of the switch's native VLAN

Dynamic Trunking Protoc



- Protocol to automate certain aspects of VLAN configuration
  - Determines whether two connected switches want to create a trunk
  - Automatically sets parameters such as encapsulation and VLAN range
- DTP transitions port through a set of states
  - Auto (port is willing to be trunked), On/Off (permanently forces link into/from trunking, even if neighbor disagrees), Desirable (attempts to make port a trunk; pursues agreement with neighbor)

Medium Access Control Address

- MAC address
  - Numerical address associated with an adapted
  - 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

- Broadcast address (FF-FF-FF-FF-FF-FF)
  - Send the frame to *all* adapters

MAC Address vs. IP Address

- MAC addresses (used in link-layer)
  - Hard-coded in read-only memory when adapter is built
  - Like a social security number
  - Flat name space of 48 bits (e.g., 00-0E-9B-6E-49-76)
  - 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
  - Like a postal mailing address
  - Hierarchical name space of 32 bits (e.g., 12.178.66.9)
  - Not portable, and depends on where the host is attached
  - Used to get a packet to destination IP subnet

Naming

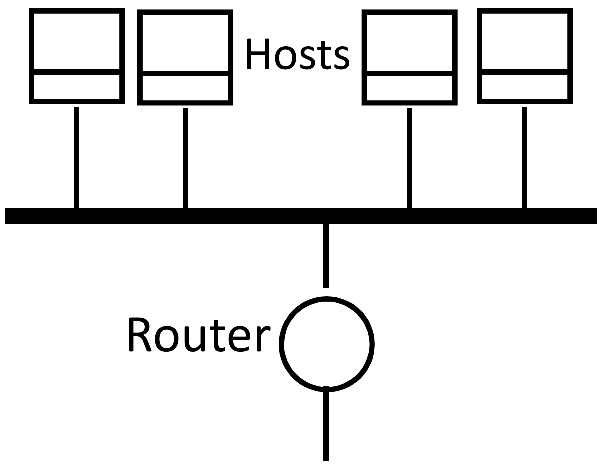
- Application layer: URLs and domain names
  - names “resources” -- hosts, content, program
  - (*recall: mixes the what and where of an object*)
- Network layer: IP addresses
  - host’s network location
- Link layer: MAC addresses
  - host identifier
- Use all three for end-to-end communication!

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? (remote)
  - what is B’s MAC address? (if B is local)
  - what is my first-hop router’s address? (if B is not local)
  - …

ARP and DHCP

- Link layer discovery protocols
  - “Address Resolution Protocol”, “Dynamic Host Configuration Protocol”
  - confined to a single local-area network (LAN)
  - rely on broadcast capability of a LAN



ARP and DHCP

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

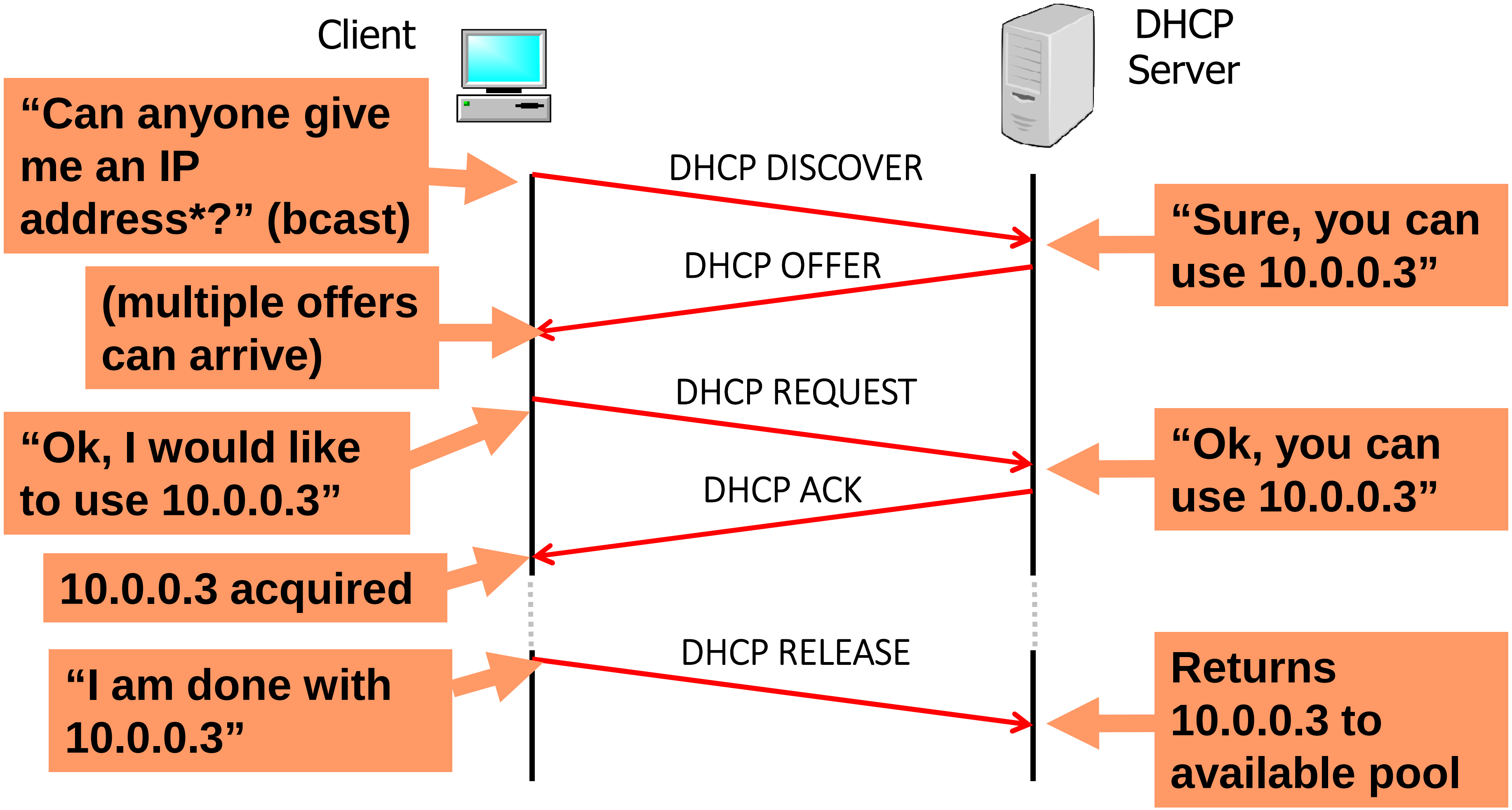
ARP and DHCP

- 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?

Dynamic Host Configuration Protocol (DHCP)

- Automatically configure hosts
  - Assign IP addresses, DNS server, default gateway, etc.
  - Client listen on UDP port 68, servers on 67
- Very common LAN protocol
  - Rare to find a device that doesn't support it
- Address is assigned for a lease time

Dynamic Host Configuration Protocol (DHCP)



*and other config information

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 DNS name server(s)
  - IP address(es) for its first-hop “default” router(s)

DHCP: operation

1. 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 “relay agents” used when the DHCP server isn’t on the same broadcast domain -- see text)

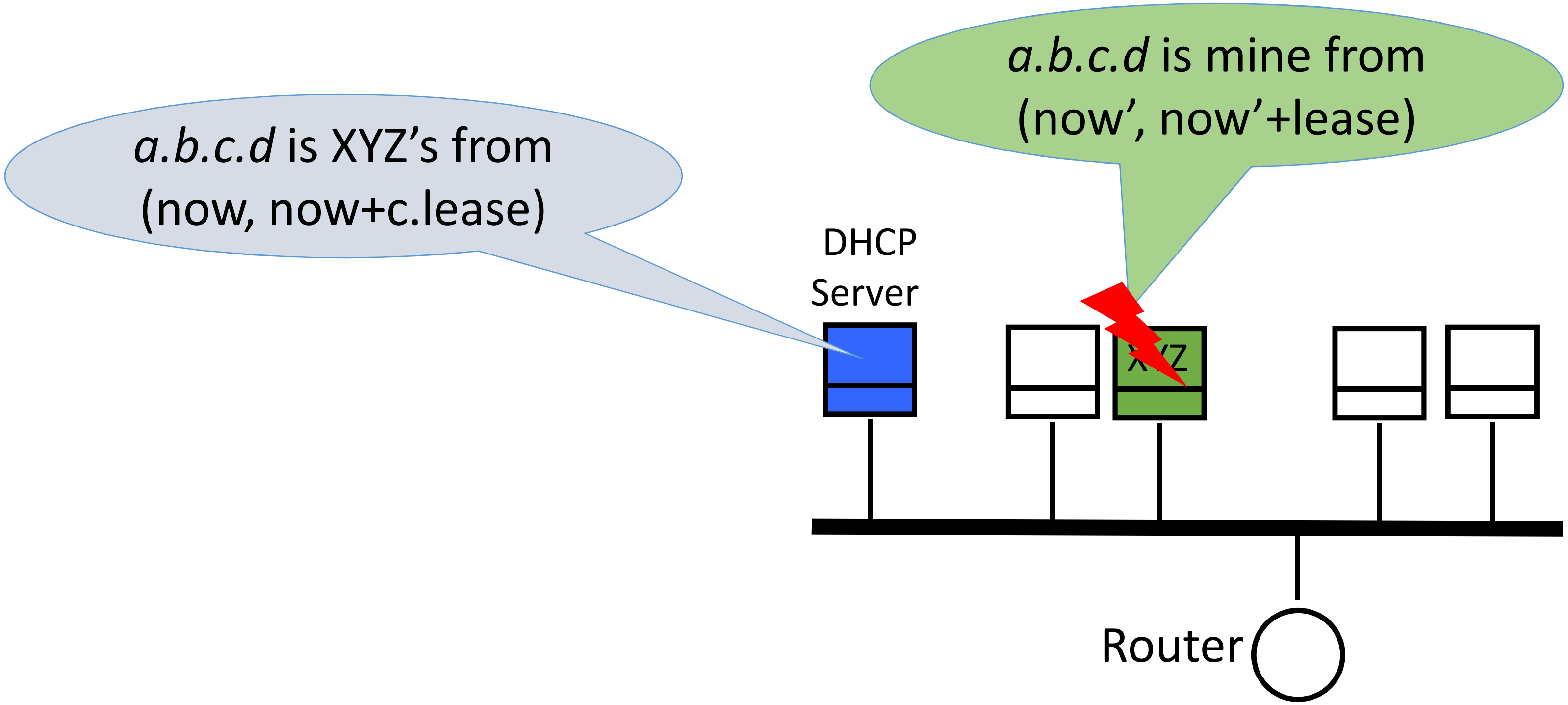
DHCP uses "soft state"

- Soft state: if not refreshed, state is forgotten
  - hard state: allocation is deliberately returned/withdrawn
  - 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 refresh before lease period expires
  - server: resets timer when a refresh arrives; sends ACK
  - server: reclaims allocated address when timer expires

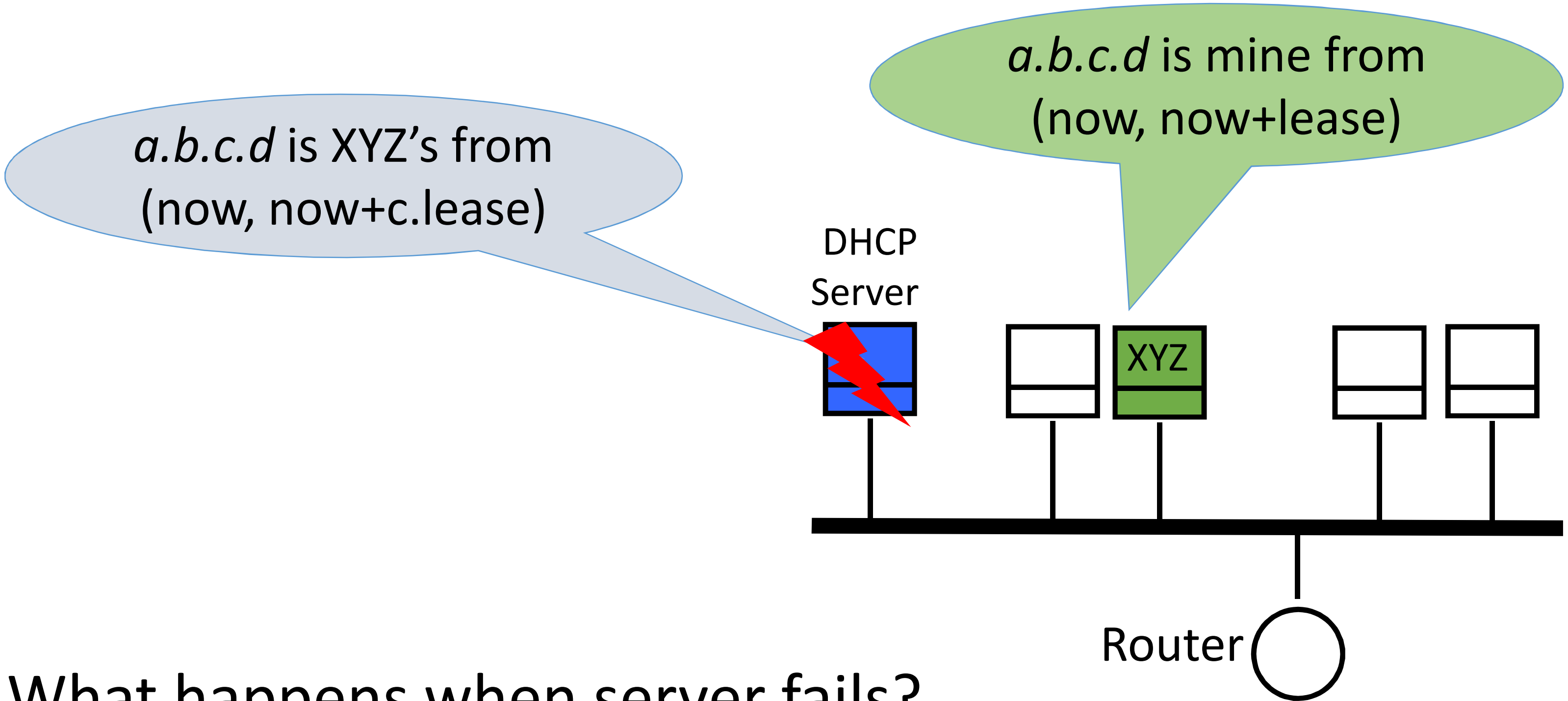
- Simple, yet robust under failure
  - state always fixes itself in (small constant of) lease time

Soft state under failure



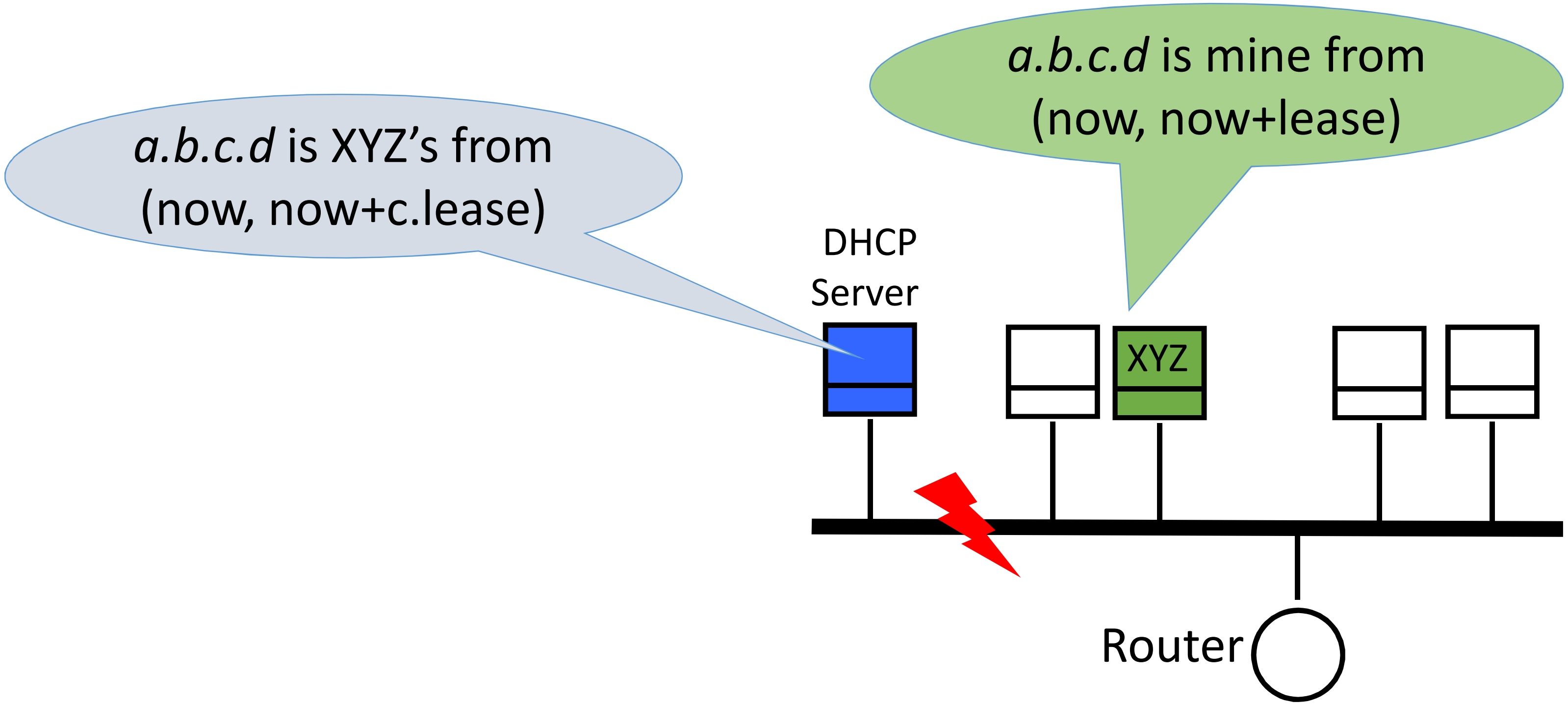
- What happens when host XYZ fails?
  - refreshes from XYZ stop
  - server reclaims *a.b.c.d* after O(lease period)

Soft state under failure



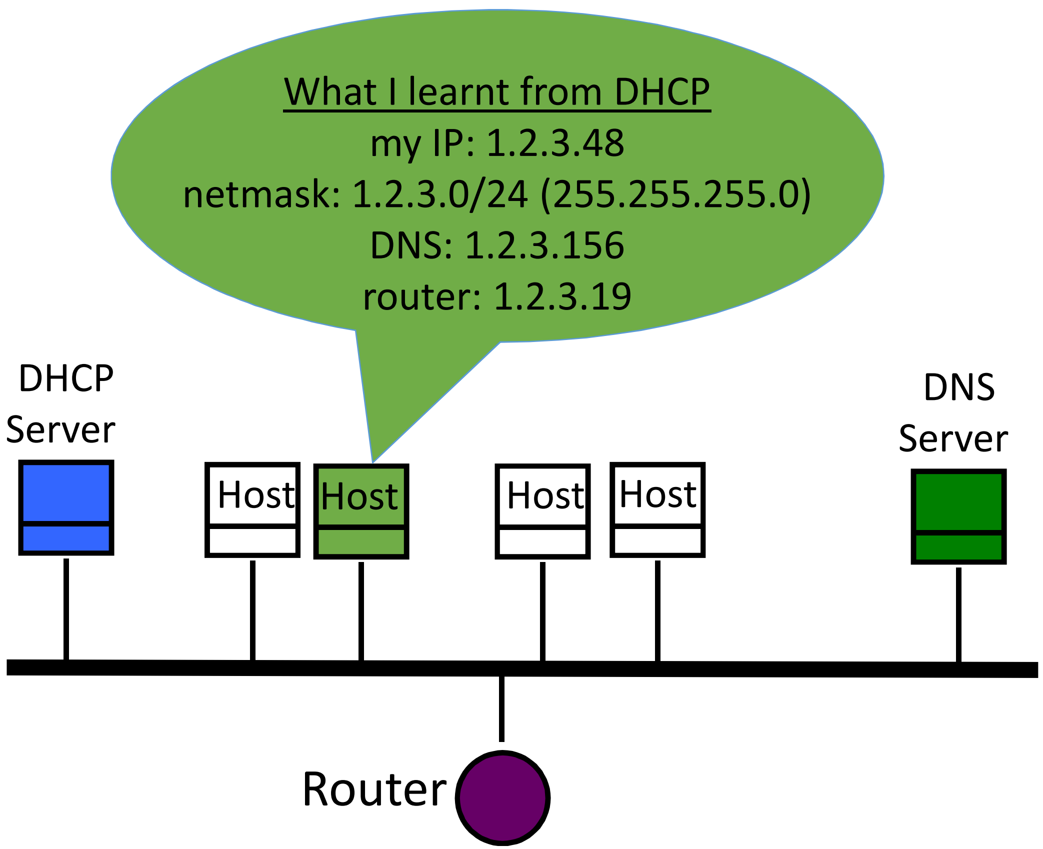
- What happens when server fails?
  - ACKs from server stop
  - XYZ releases address after O(lease period); send new request
  - A new DHCP server can come up from a `cold start' and we're back on track in ~lease time

Soft state under failure

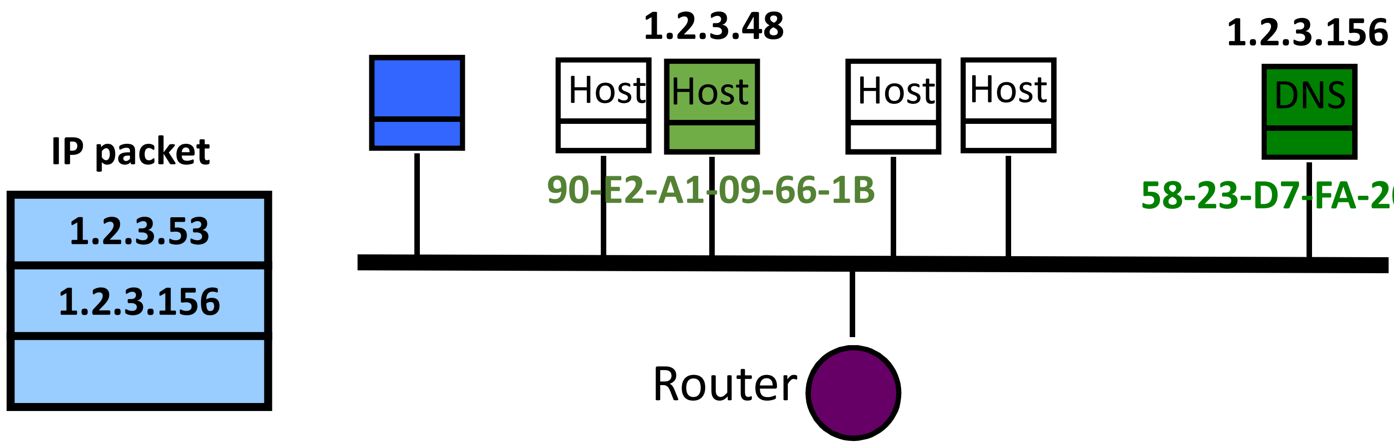


- What happens if the network fails?
  - refreshes and ACKs don't get through
  - XYZ release address; DHCP server reclaims it

Are we there yet?



Sending Packets Over Link-Layer



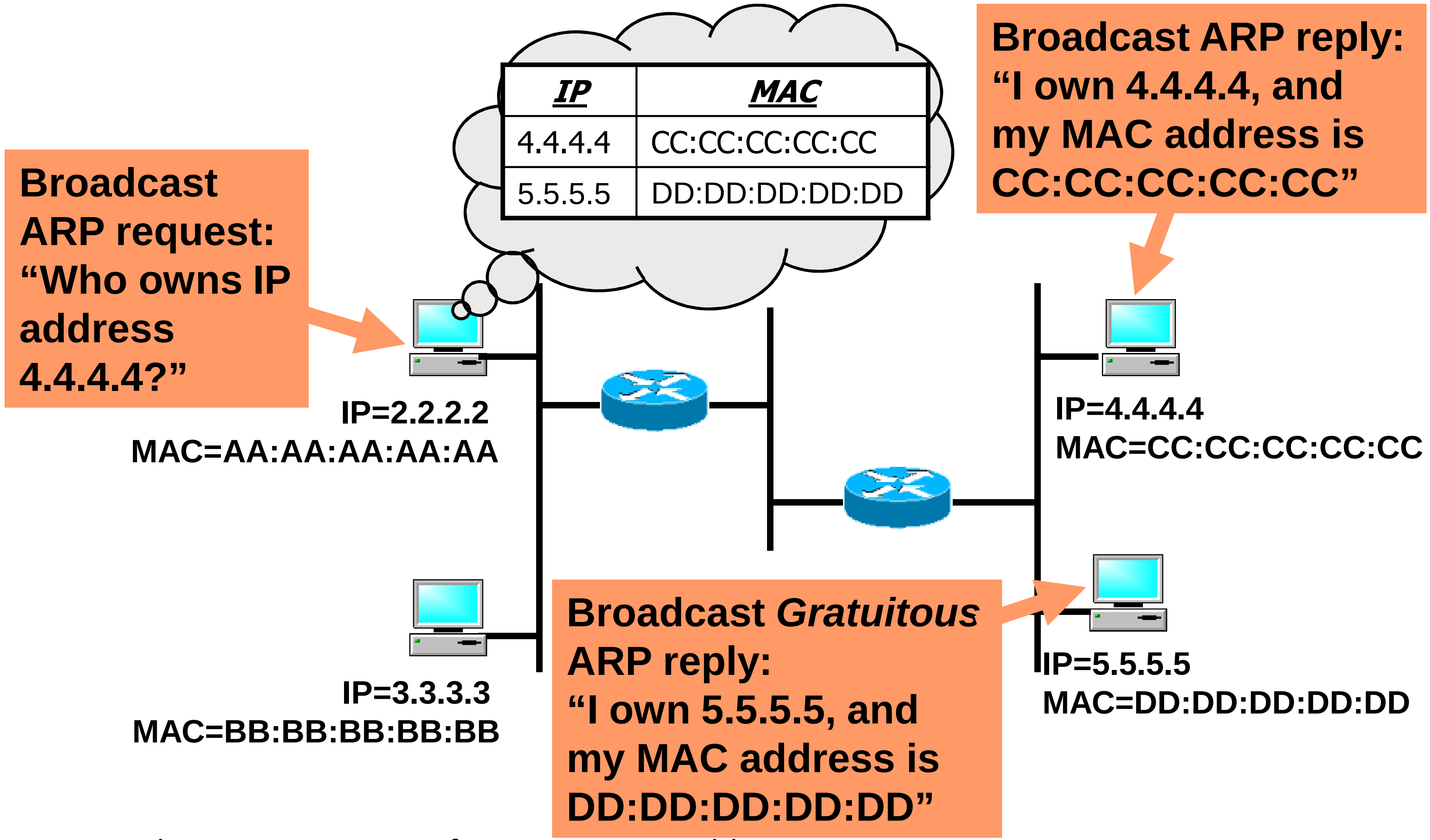
- Link layer only understands MAC addresses
  - Translate the destination IP address to MAC address
  - Encapsulate the IP packet inside a link-level frame

ARP: Address Resolution Protocol

- Every host maintains an ARP table
  - list of (IP address → MAC address) pairs
- Consult the table when sending a packet
  - Map destination IP address to destination MAC address
  - Encapsulate the (IP) data packet with MAC header; transmit
- But: what if IP address not in the table?
  - Sender broadcasts: "**Who has IP address 1.2.3.156**?"
  - Receiver responds: "**MAC address 58-23-D7-FA-20-B0**"
  - Sender caches result in its ARP table

Address Resolution Protocol (ARP)

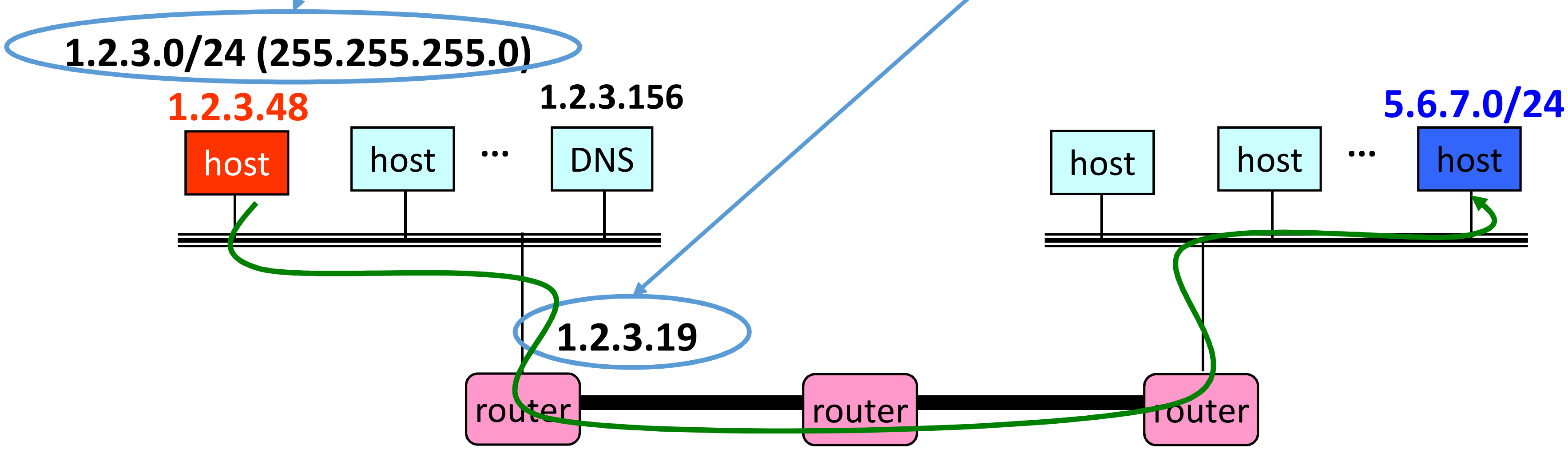
- Networked applications are programmed to deal with IP addresses
- But Ethernet forwards to MAC address
- How can OS know the MAC address corresponding to a given IP address?
- Solution: Address Resolution Protocol
  - Broadcasts ARP request for MAC address owning a given IP address



- ARP: determine mapping from IP to MAC address
- What if IP address not on subnet?
  - Each host configured with “default gateway”, use ARP to resolve its IP address
- Gratuitous ARP: tell network your IP to MAC mapping
  - Used to detect IP conflicts, IP address changes; update other machines’ ARP tables, update bridges’ learned information

What if the destination is remote?

- Look up the MAC address of the first hop router
  - 1.2.3.48 uses ARP to find MAC address for first-hop router **1.2.3.19** rather than ultimate destination IP address
- How does the red host know the destination is not local?
  - Uses netmask (discovered via DHCP)
- How does the red host know about 1.2.3.19?
  - Also DHCP



Security Analysis of ARP

- Impersonation
  - Any node that hears request can answer …
  - … and can say whatever they want

- Actual legit receiver never sees a problem
  - Because even though later packets carry its IP address, its NIC doesn’t capture them since not its MAC address

Steps in Sending a Packet

What do hosts need to know?

And how do they find out?

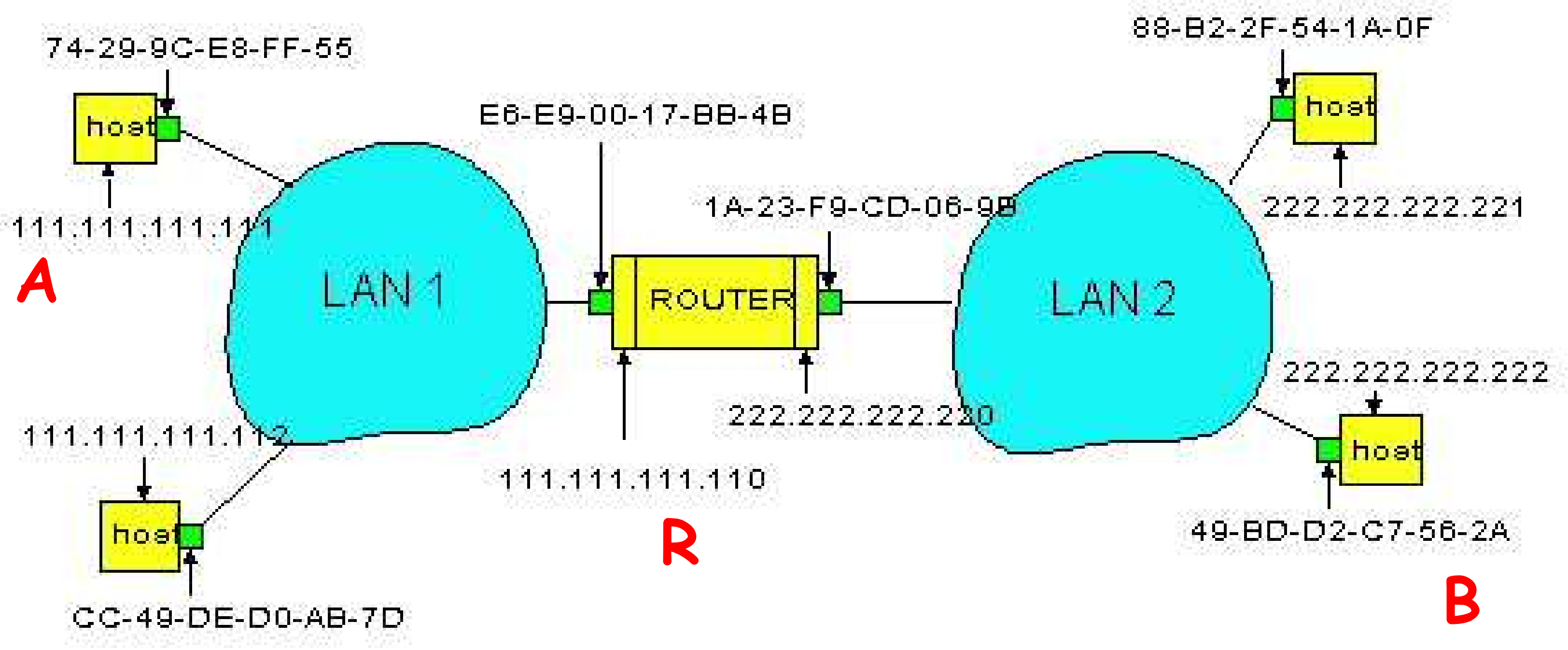
Steps in reaching a Host

- First look up destination’s IP address
- Need to know where local DNS server is
  - **DHCP**
- Also needs to know its own IP address
  - **DHCP**

Sending a Packet

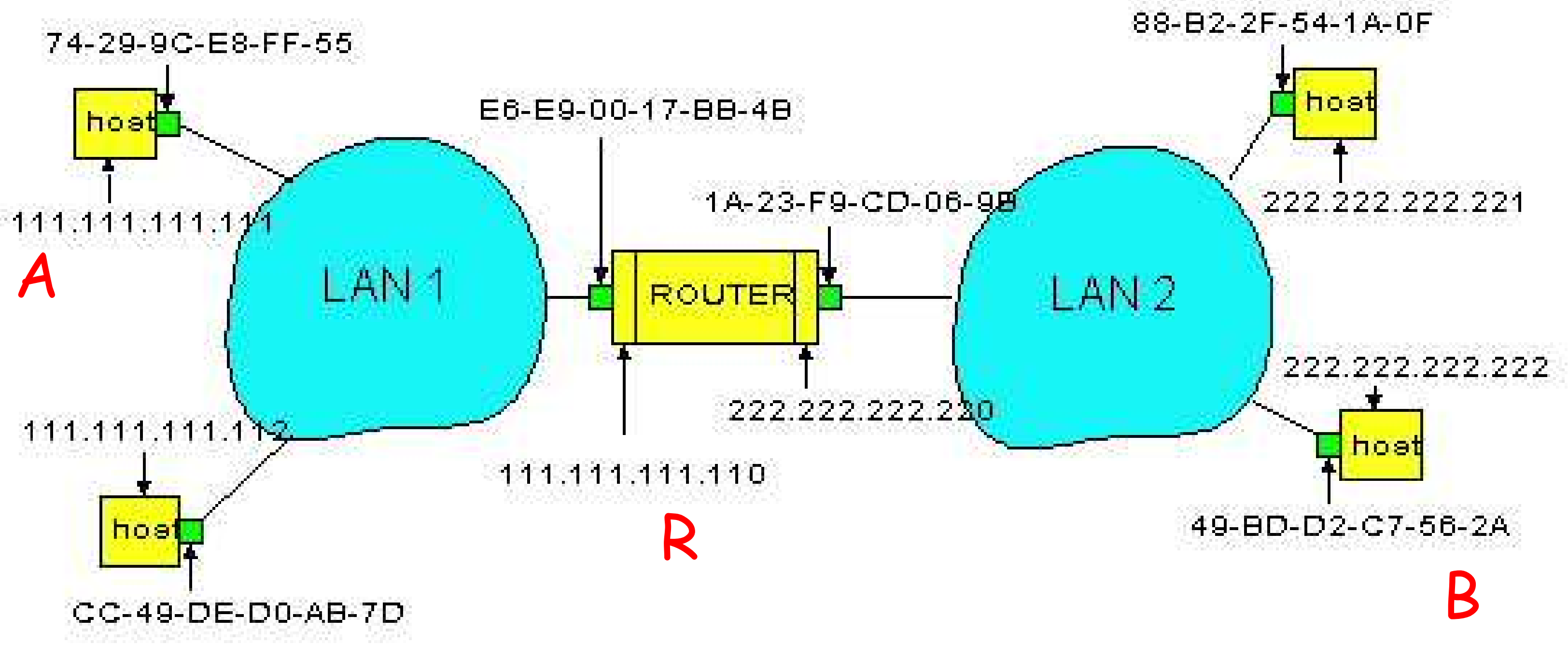
- On same subnet:
  - Use MAC address of destination.
  - ***ARP***
- On some other subnet:
  - Use MAC address of first-hop router.
  - ***DHCP + ARP***
- And how can a host tell whether destination is on same or other subnet?
  - Use the netmask
  - ***DHCP***

Example: A Sending a Packet to B



How does host A send an IP packet to host B?

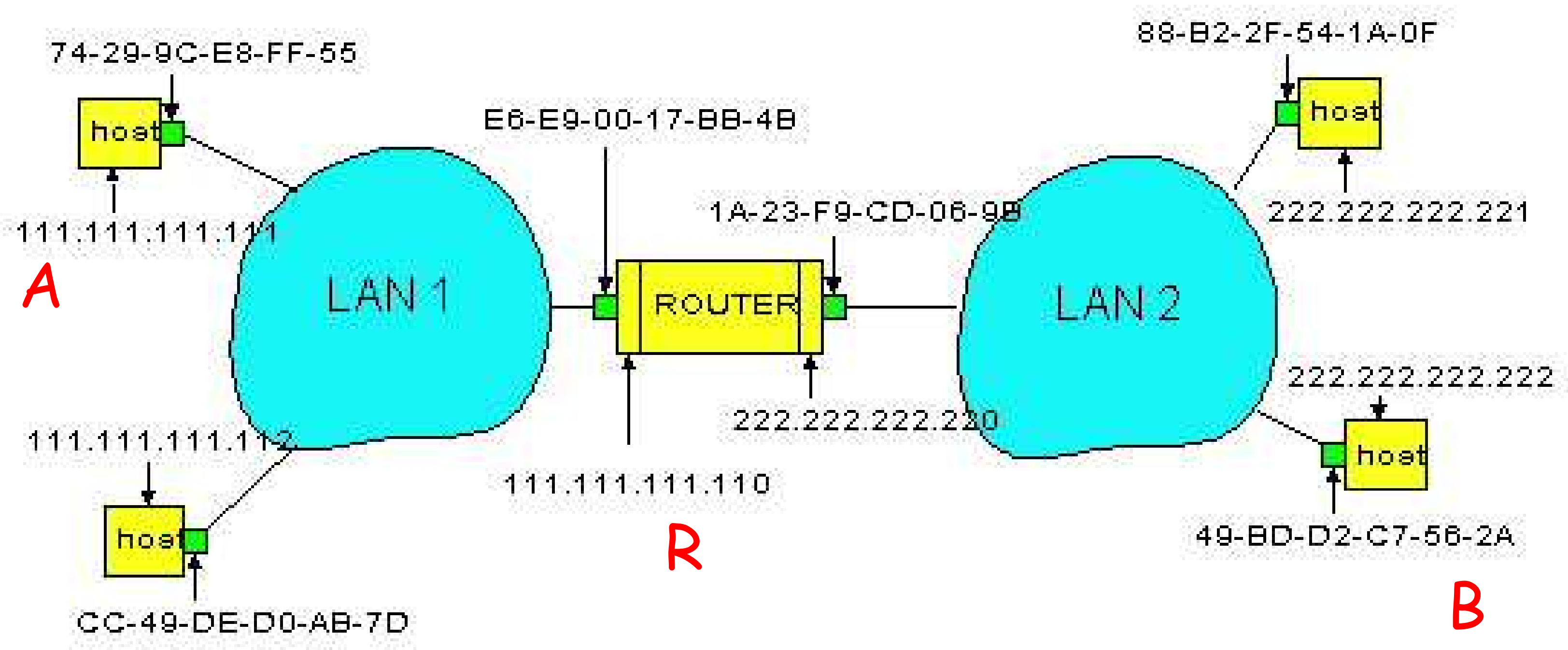
Example: A Sending a Packet to B



1. A sends packet to R.
2. R sends packet to B.

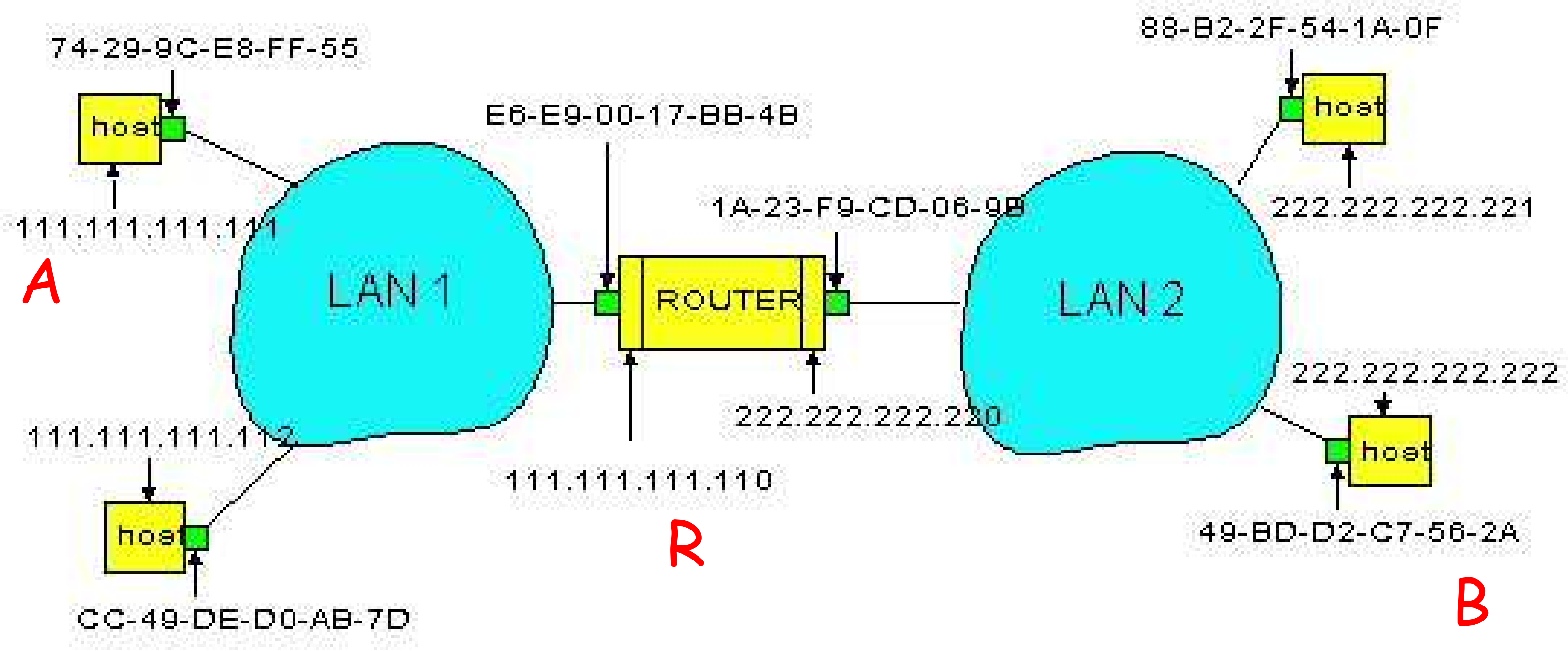
Host A Decides to Send Through R

- Host A constructs an IP packet to send to B
  - Source 111.111.111.111, destination 222.222.222.222
- Host A has a gateway router R
  - Used to reach destinations outside of 111.111.111.0/24
  - Address 111.111.111.110 for R learned via DHCP



Host A Sends Packet Through R

- Host A learns the MAC address of R's interface
  - ARP request: broadcast request for 111.111.111.110
  - ARP response: R responds with E6-E9-00-17-BB-4B
- Host A encapsulates the packet and sends to R

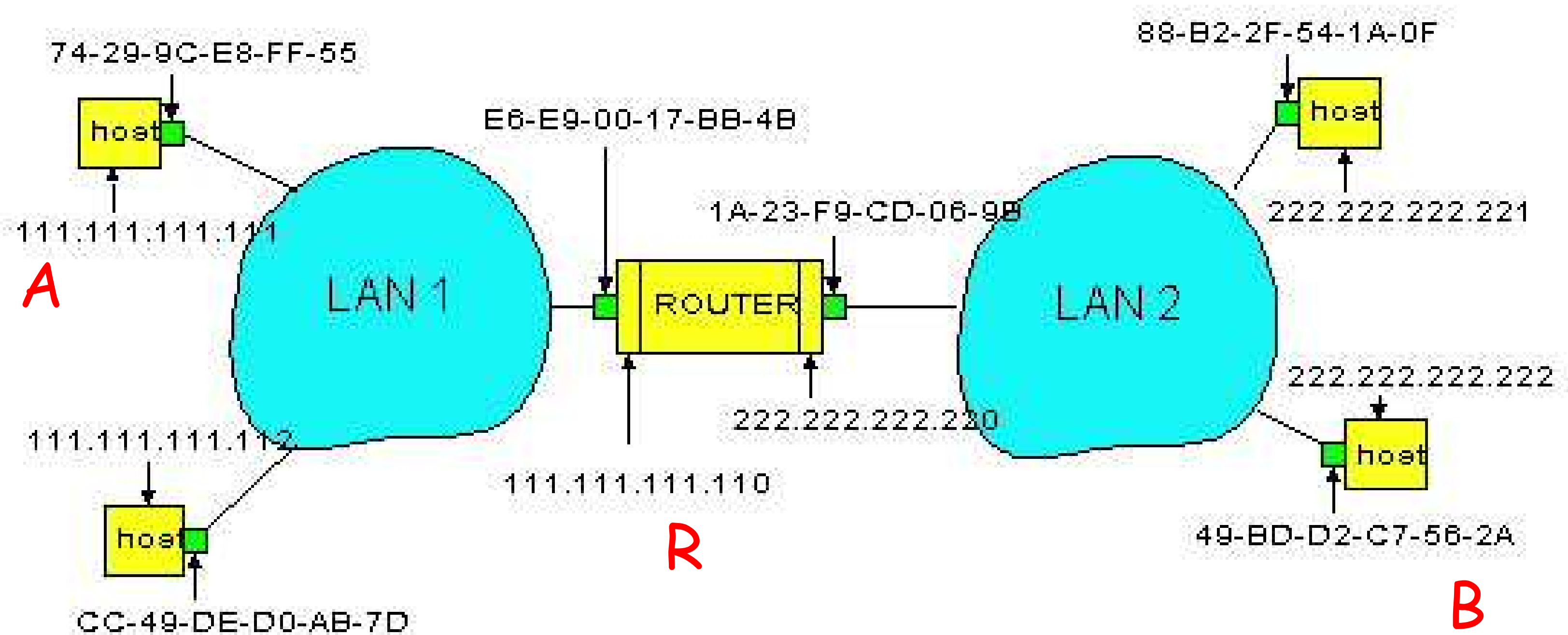


ket

Two points:

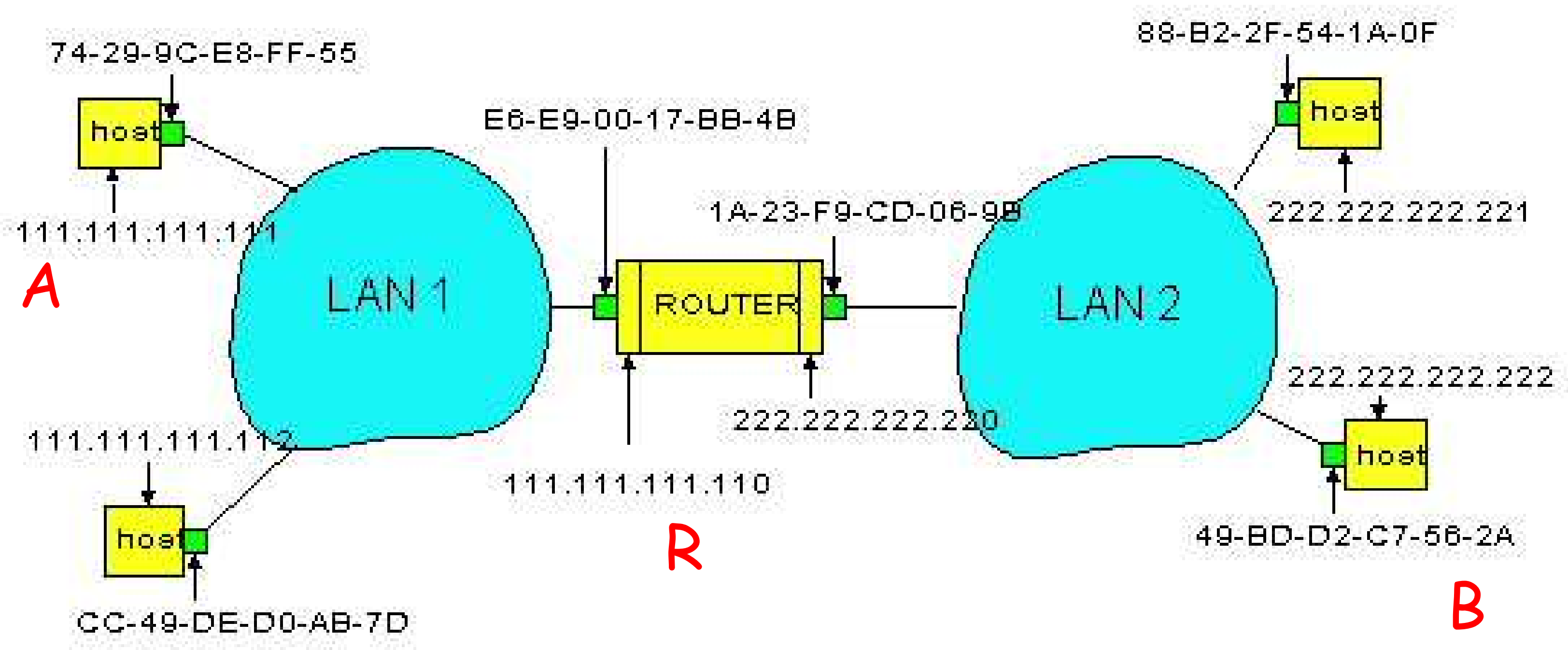
- Routing table points to this port
- Destination address is within mask of port's address (i.e., local)

-
  - R sees ... is destined to 222.222.222.222
- Router R consults ... forwarding table
  - Packet matches 222.222.222.0/24 via other adapter (port)



R Sends Packet to B

- Router R's learns the MAC address of host B
  - ARP request: broadcast request for 222.222.222.222
  - ARP response: B responds with 49-BD-D2-C7-56-2A
- Router R encapsulates the packet and sends to B



Key Ideas in Both ARP and DHCP

- Broadcasting: used for initial bootstrap
- Caching: remember the past for a while
  - Store the information you learn to reduce overhead
  - Remember your own address & other host's addresses
  - *Key optimization for performance*
- Soft state: eventually forget the past
  - Associate a time-to-live field with the information
  - … and either refresh or discard the information
  - *Key for robustness*

Discovery mechanisms

We've seen two broad approaches

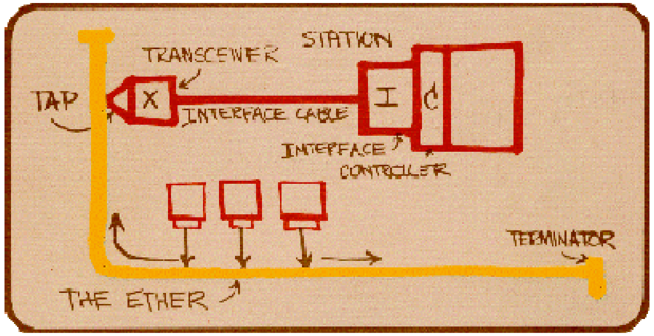
- Broadcast (ARP, DHCP)
  - flooding doesn't scale
  - no centralized point of failure
  - zero configuration
- Directory service (DNS)
  - no flooding
  - root of the directory is vulnerable (caching is key)
  - needs configuration to bootstrap (local, root servers, *etc.*)

Can we get the best of both?

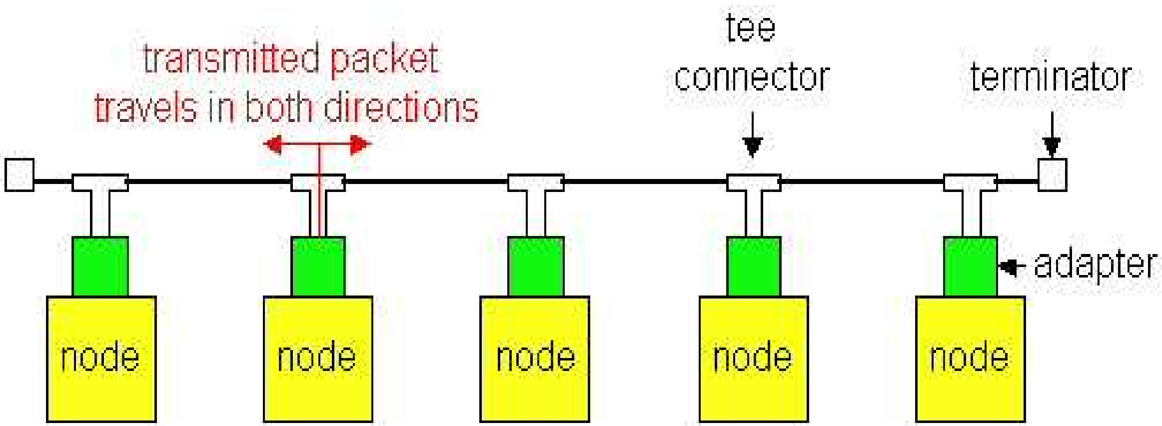
- Internet-scale yet zero config?

Ethernet

Ethernet



- Bob Metcalfe, Xerox PARC, visits Hawaii and gets an idea!
- Shared wired medium
  - coax cable



Evolution

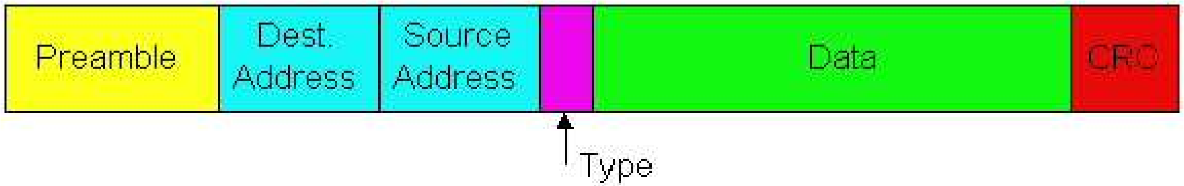
- Ethernet was invented as a broadcast technology
  - Hosts share channel
  - Each packet received by all attached hosts
  - CSMA/CD for media access control
- Current Ethernets are "switched"
  - Point-to-point links between switches; between a host and switch
  - No sharing, no CSMA/CD
  - (Next lecture) uses "self learning" and "spanning tree" algorithms for routing

Ethernet: CSMA/CD Protocol

- **Carrier sense**: wait for link to be idle
- **Collision detection**: listen while transmitting
  - No collision: transmission is complete
  - Collision: abort transmission & send **jam** signal
- **Random access**: binary exponential back-off
  - After collision, wait a random time before trying again
  - After m^{th} collision, choose K randomly from {0, …, 2^m-1}
  - … and wait for K*512 bit times before trying again
    - If transmission occurring when ready to send, wait until end of transmission (CSMA)

Ethernet Frame Structure

- Encapsulates IP datagram

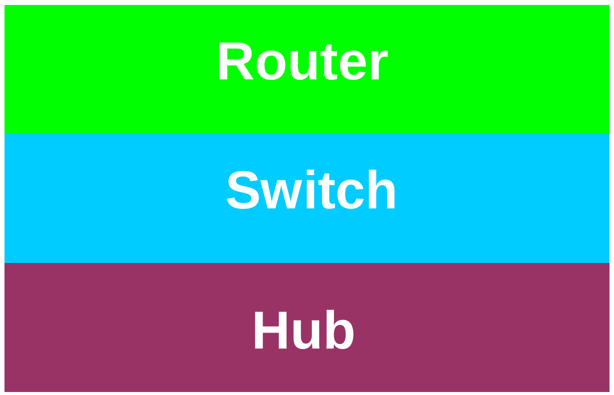


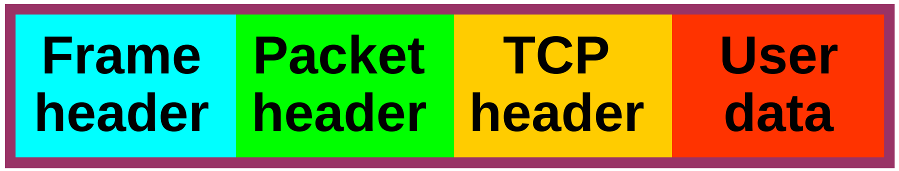
- Preamble: 7 bytes with a particular pattern used to synchronize receiver, sender clock rates
- Addresses: 6 bytes: frame is received by all adapters on a LAN and dropped if address does not match
- Type: 2 bytes, indicating higher-layer protocol (e.g., IP, Appletalk)
- CRC: 4 bytes for error detection
- Data payload: maximum 1500 bytes, minimum 46 bytes

Routing with Switches

Shuttling Data at Different Layers

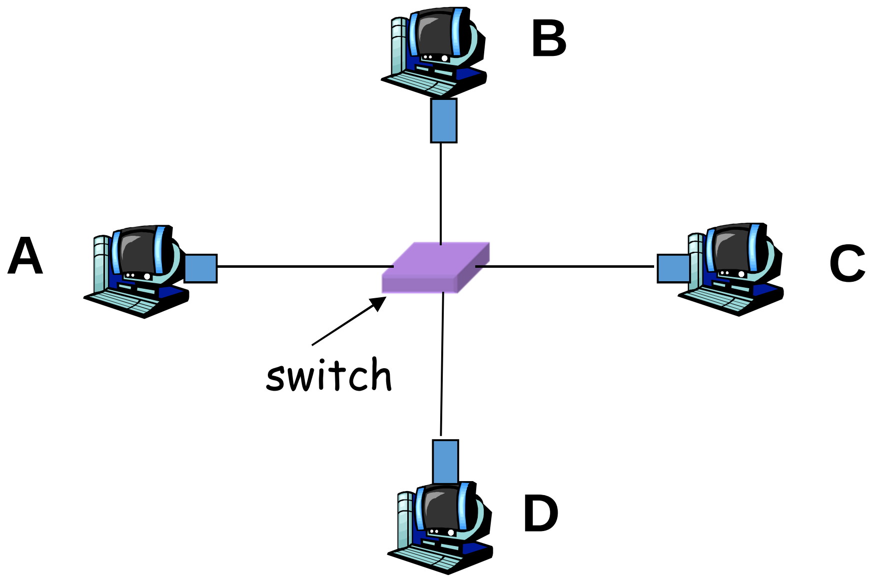
- Different devices switch different things
  - Physical layer: electrical signals or bits (hubs)
  - Link layer: frames (switches)
  - Network layer: packets (routers)





Switches Enable Concurrent Communication

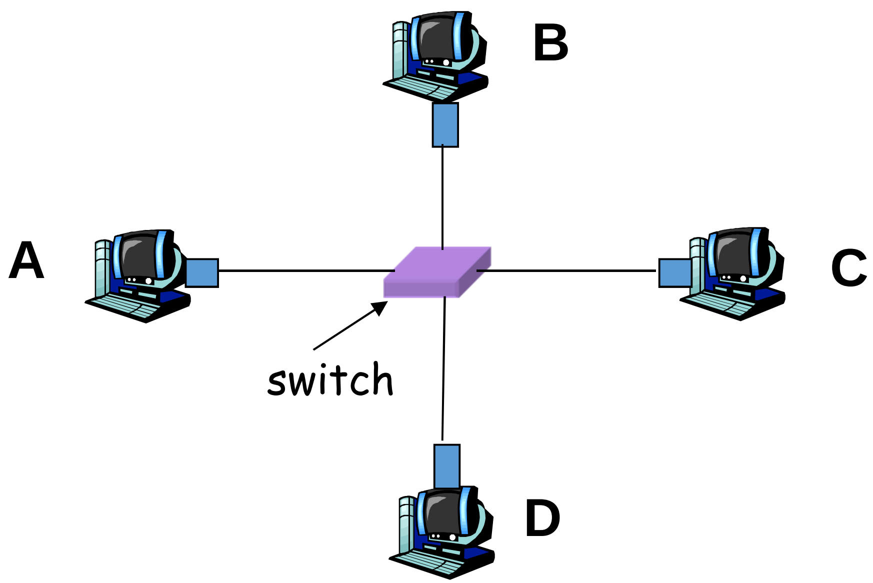
- Host A can talk to C, while B talks to D



- Completely avoids collisions (if hosts directly attached)
  - No need for all material we discuss later in lecture
  - Change in nature of multiple access, but same framing
    - ***Key to the success of ethernet!***

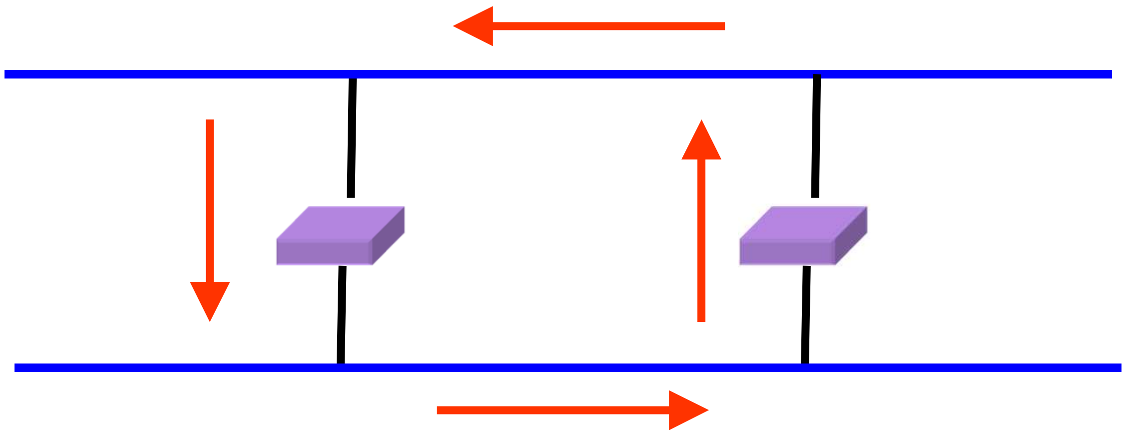
Self Learning

- Maps destination MAC to outgoing interface
- Construct switch table automatically
- Floods when does not have entry in table



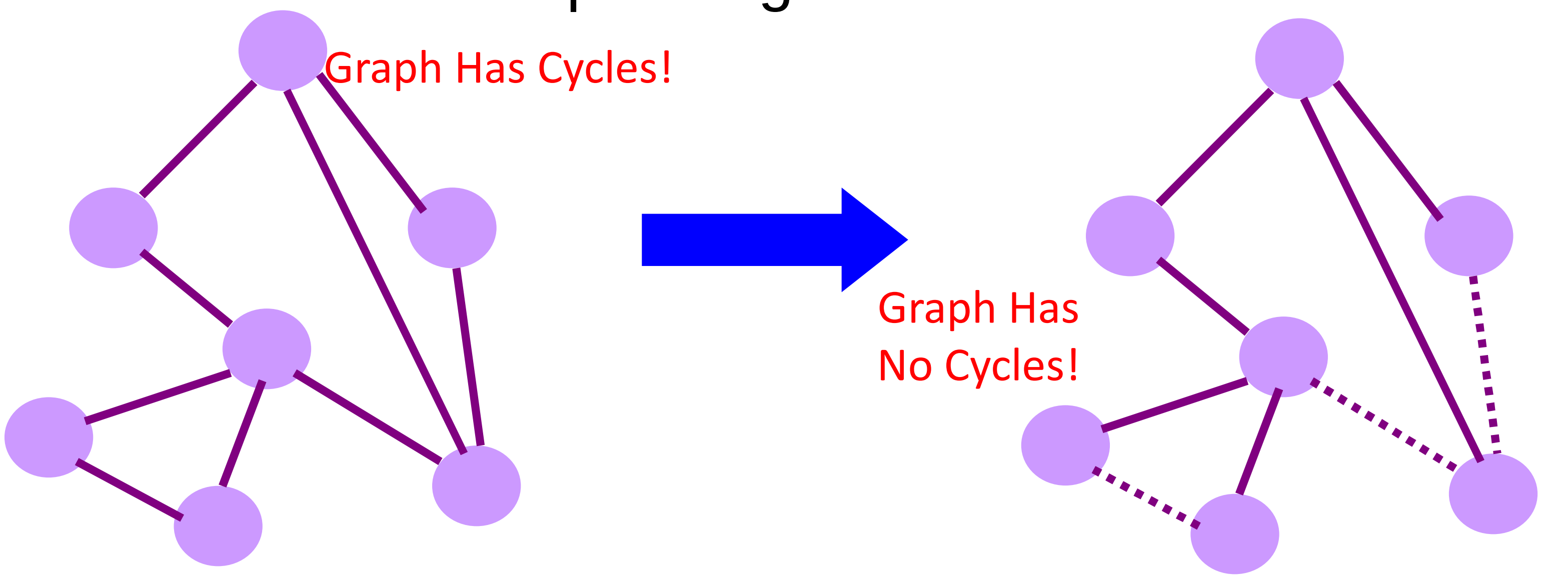
Flooding Can Lead to Loops

- Flooding can lead to forwarding loops
  - E.g., if the network contains a cycle of switches
  - "Broadcast storm"



Solution: Spanning Trees

- Ensure the forwarding topology has no loops
  - Avoid using some of the links when flooding
  - … to prevent loop from forming
- Spanning tree
  - Sub-graph that covers all vertices but *contains no cycles*
  - Links not in the spanning tree do not forward frames



You: Design a Spanning Tree Algorithm

- Distributed
- No global information
- Neighbors can exchange information
- Must adapt when failures occur
  - But don't worry about that on first try…

- Take 5 minutes, break into groups, report back

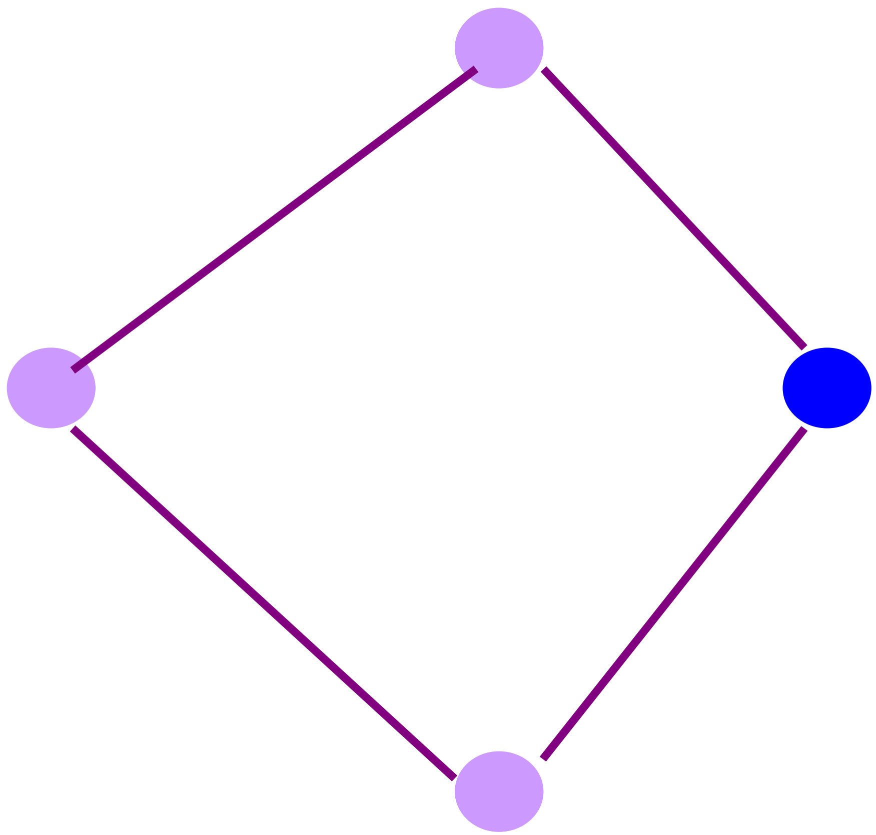
What Do We Know?

- Shortest paths to (or from) a node form a tree
  - No shortest path can have a cycle

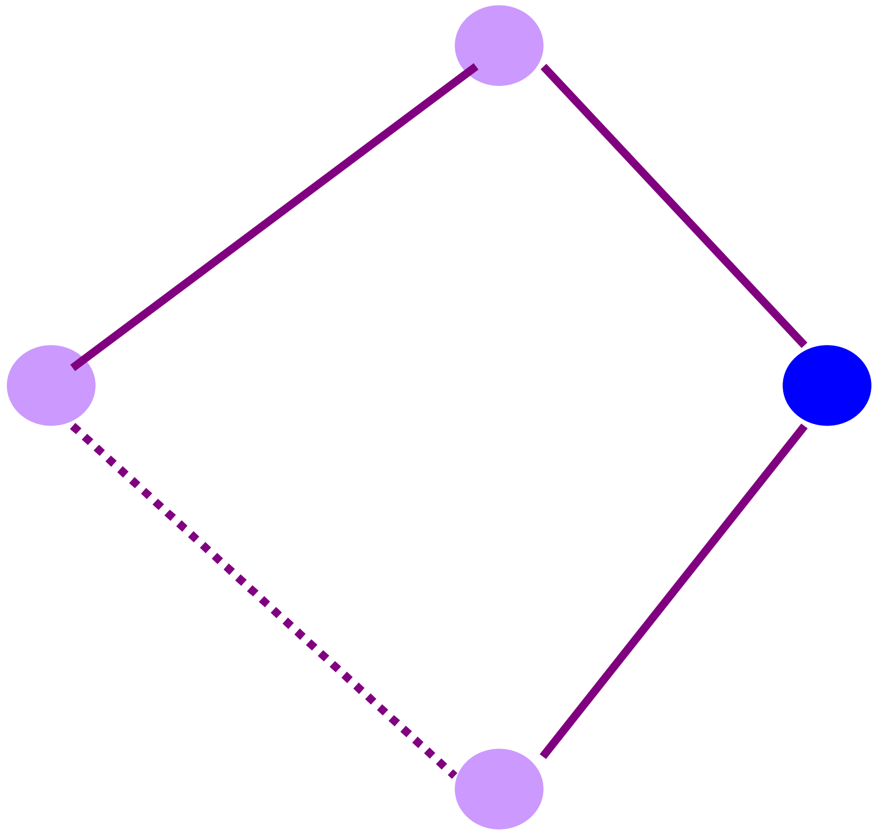
- But we must limit each node to one outgoing port towards destination
  - Why?

- Because this is not a directed graph!

Two Shortest Paths Create Cycle!



Must only choose one



Algorithm Has Two Aspects

- Pick a root:
  - This will be the destination to which all shortest paths go
  - **Pick the one with the smallest identifier (MAC add.)**

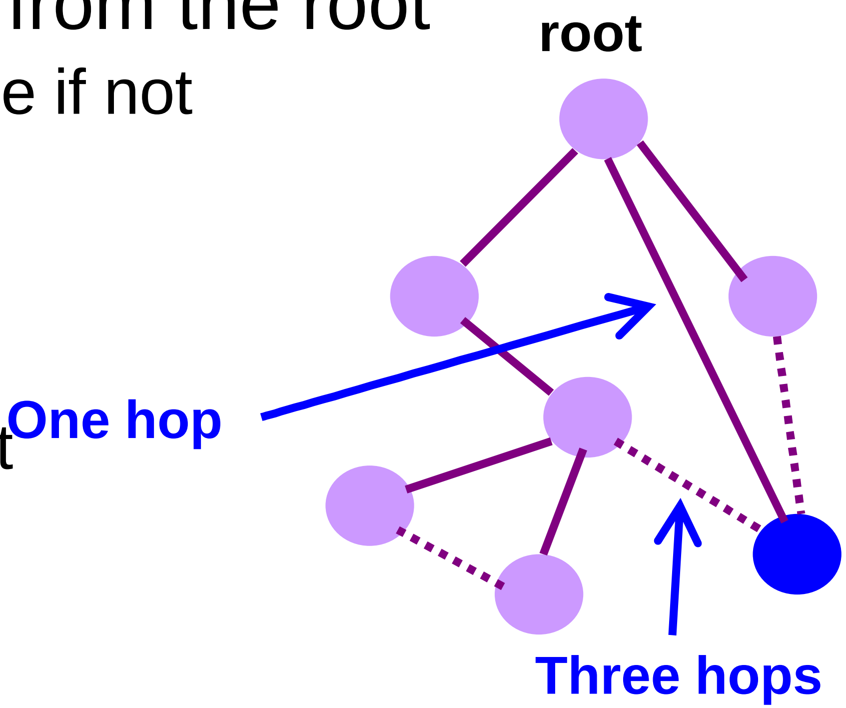
- Compute shortest paths to the root
  - Only keep the links on shortest-paths
  - Break ties in some way, so only keep one shortest path from each node

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
- In homeworks and test, remember this.

Constructing a Spanning Tree

- Switches need to elect a root
  - The switch w/ smallest identifier (MAC addr)
- Each switch determines if each interface is on the shortest path from the root
  - Excludes it from the tree if not
- **Messages (Y, d, X)**
  - From node X
  - Proposing Y as the root
  - And the distance is d

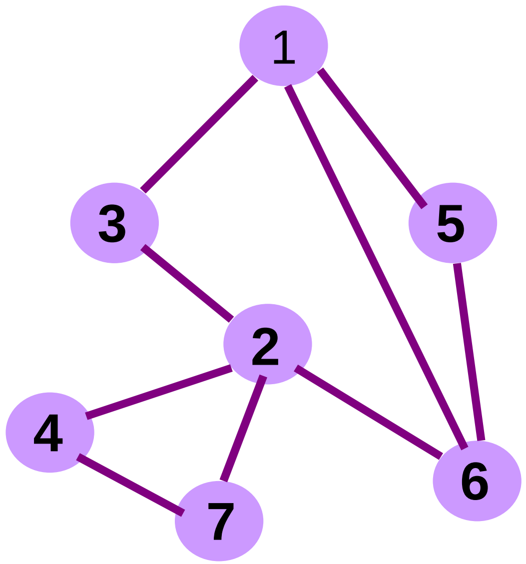


Steps in Spanning Tree Algorithm

- Initially, each switch proposes itself as the root
  - Switch sends a message out every interface
  - … proposing itself as the root with distance 0
  - Example: switch X announces (X, 0, X)
- Switches update their view of the root
  - Upon receiving message (Y, d, Z) from Z, check Y's id
  - If new id smaller, start viewing that switch as root
- Switches compute their distance from the root
  - Add 1 to the distance received from a neighbor
  - Identify interfaces not on shortest path to the root
  - … and exclude them from the spanning tree
- If root or shortest distance to it changed, "flood" updated message (Y, d+1, X)

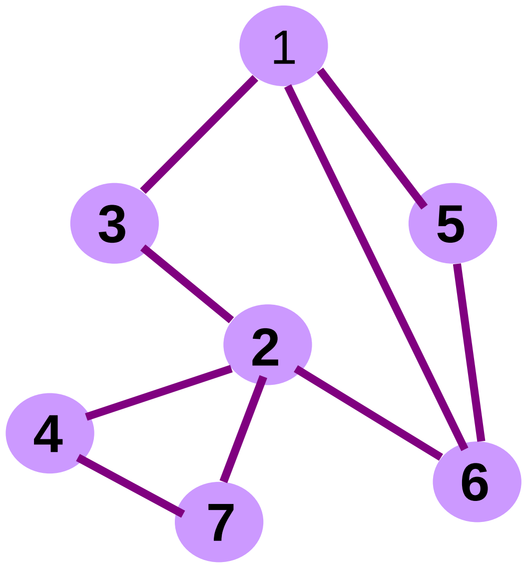
Example From Switch #4's Viewpoint

- 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



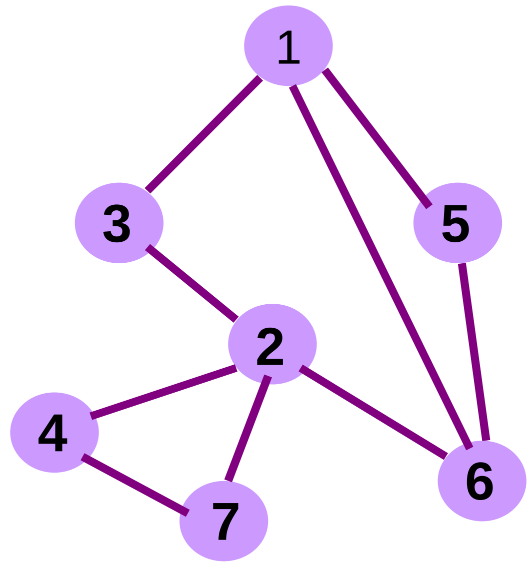
Example From Switch #4's Viewpoint

- 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



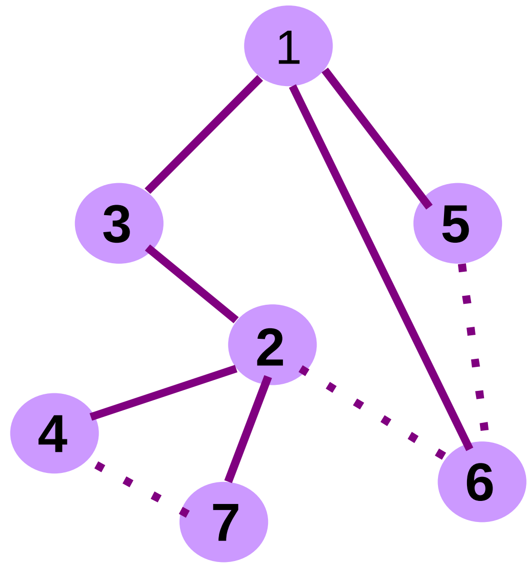
Which links are on spanning tree?

- Take a few minutes, work this out
- 3-1?
- 5-1?
- 6-1?
- 2-6?
- 2-3?



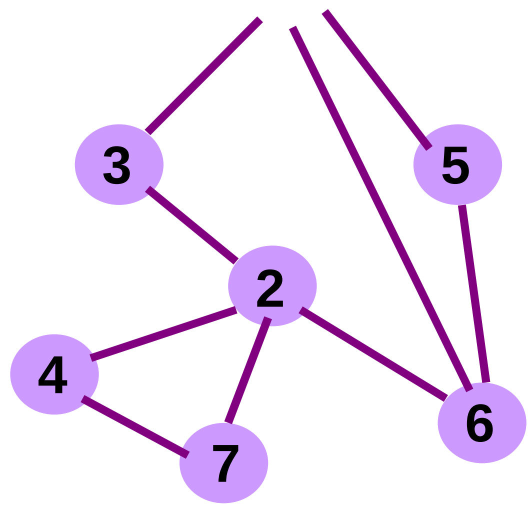
Links on spanning tree

- 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
    - Need to elect a new root, with the next lowest identifier
  - Failure of other switches and links
    - Need to recompute the spanning tree
- Root switch continues sending messages
  - Periodically reannouncing itself as the root (1, 0, 1)
  - Other switches continue forwarding messages
- Detecting failures through timeout (soft state)
  - If no word from root*, time out and claim to be the root!*

Why do people hate spanning tree?

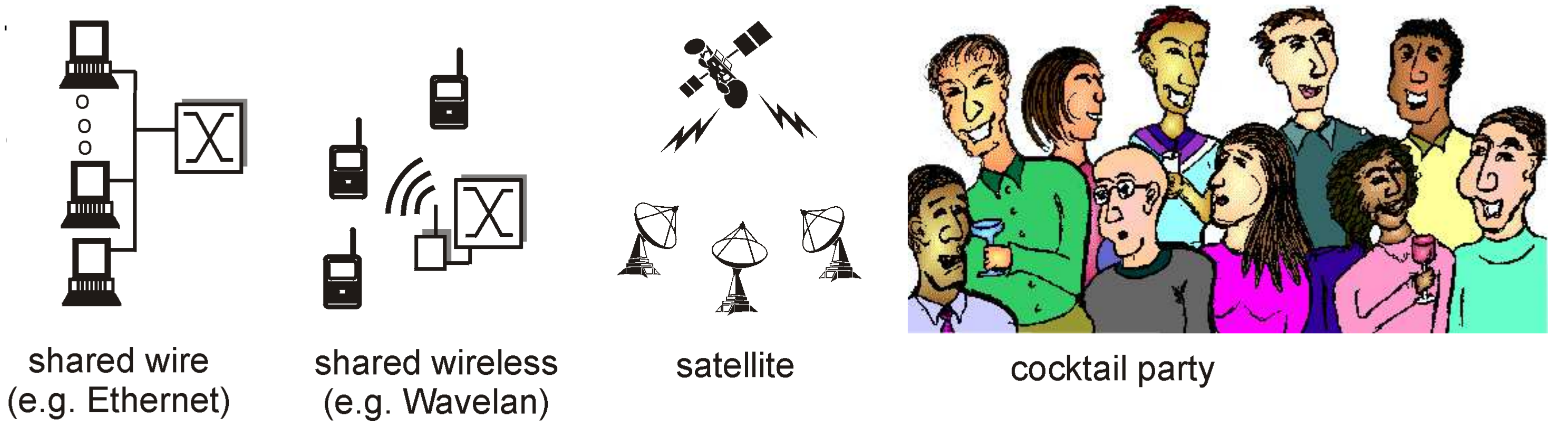
- Delay in reestablishing spanning tree
  - Network is “down” until spanning tree rebuilt
  - Work on rapid spanning tree algorithms…
    - And multiple spanning trees

- Much of the network bandwidth goes unused
  - Forwarding is only over the spanning tree
  - *Why did you bother with all those other links?*

Broadcast
vs Point-to-Point

Point-to-Point vs. Broadcast Media

- Point-to-point: dedicated pairwise communication
  - Long-distance fiber link
  - Point-to-point link between Ethernet switch and host
- Broadcast: shared wire or medium



Multiple Access Algorithm

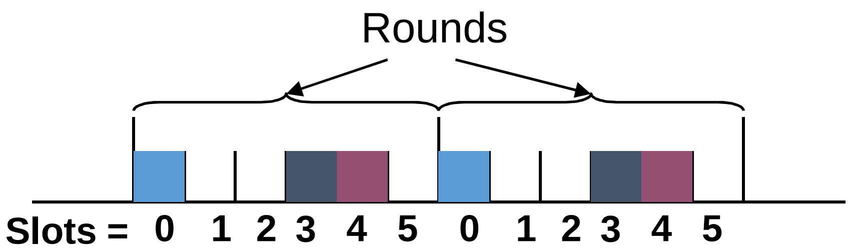
- Single shared broadcast channel
  - Must avoid having multiple nodes speaking at once
  - Otherwise, collisions lead to garbled data
  - Need distributed algorithm for sharing the channel
  - Algorithm determines which node can transmit

- Classes of techniques
  - Channel partitioning: divide channel into pieces
  - Taking turns: scheme for trading off who gets to transmit
  - Random access: allow collisions, and then recover

Channel Partitioning: TDMA

TDMA: Time Division Multiple Access

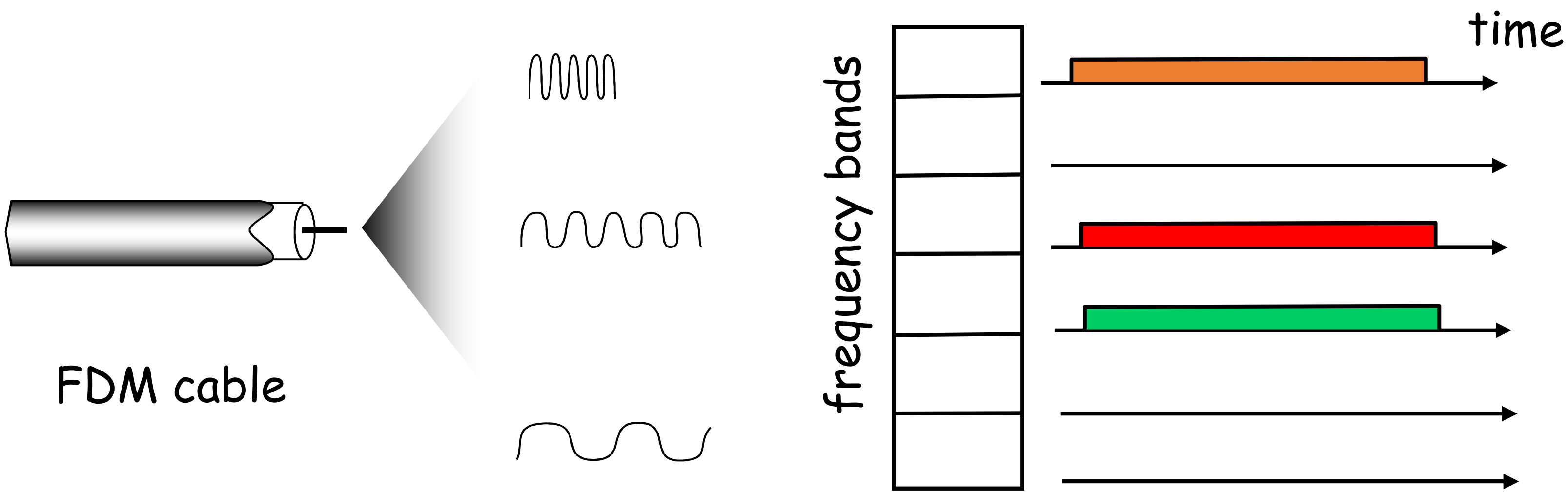
- Access to channel in "rounds"
  - Each station gets fixed length slot in each round
- Time-slot length is packet transmission time
  - *Unused slots go idle*
- Example: 6-station LAN with slots 0, 3, and 4



Channel Partitioning: FDMA

FDMA: Frequency Division Multiple Access

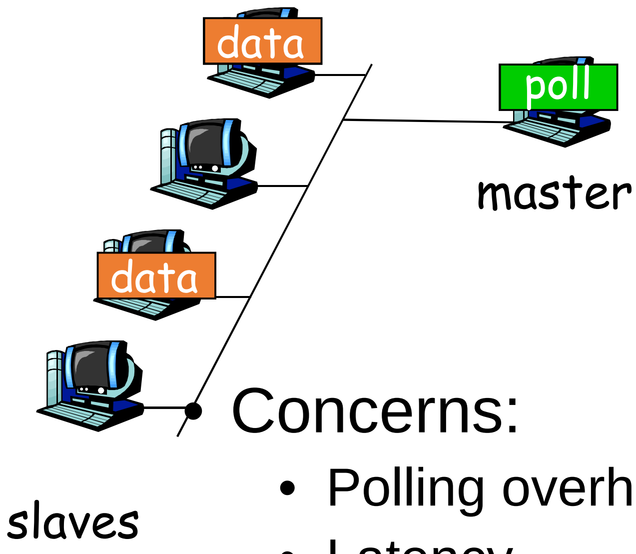
- Channel spectrum divided into frequency bands
- Each station assigned fixed frequency band
- Unused transmission time in frequency bands go idle
- Example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



"Taking Turns" MAC protocols

Polling

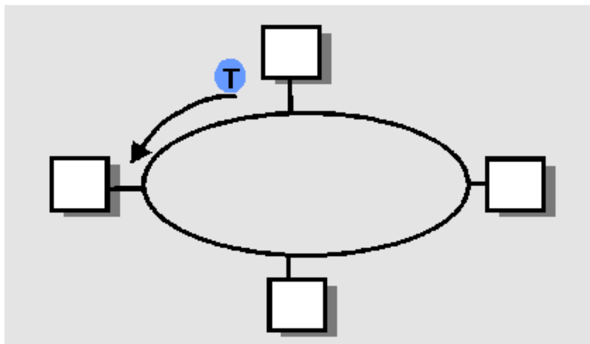
- Master node "invites" slave nodes to transmit in turn



- Concerns:
  - Polling overhead
  - Latency
  - Single point of failure (master)

Token passing

- Control token passed from one node to next sequentially
- Node must have token to send
- Concerns:
  - Token overhead
  - Latency
  - At mercy of any node



None of these are the “Internet way”…

- Why not?

- What’s wrong with
  - TDMA
  - FDMA
  - Polling
  - Token passing

- Turn to random access
  - Optimize for the common case (no collision)
  - Don’t avoid collisions, just recover from them….
  - **Sound familiar?**

Random Access MAC Protocols

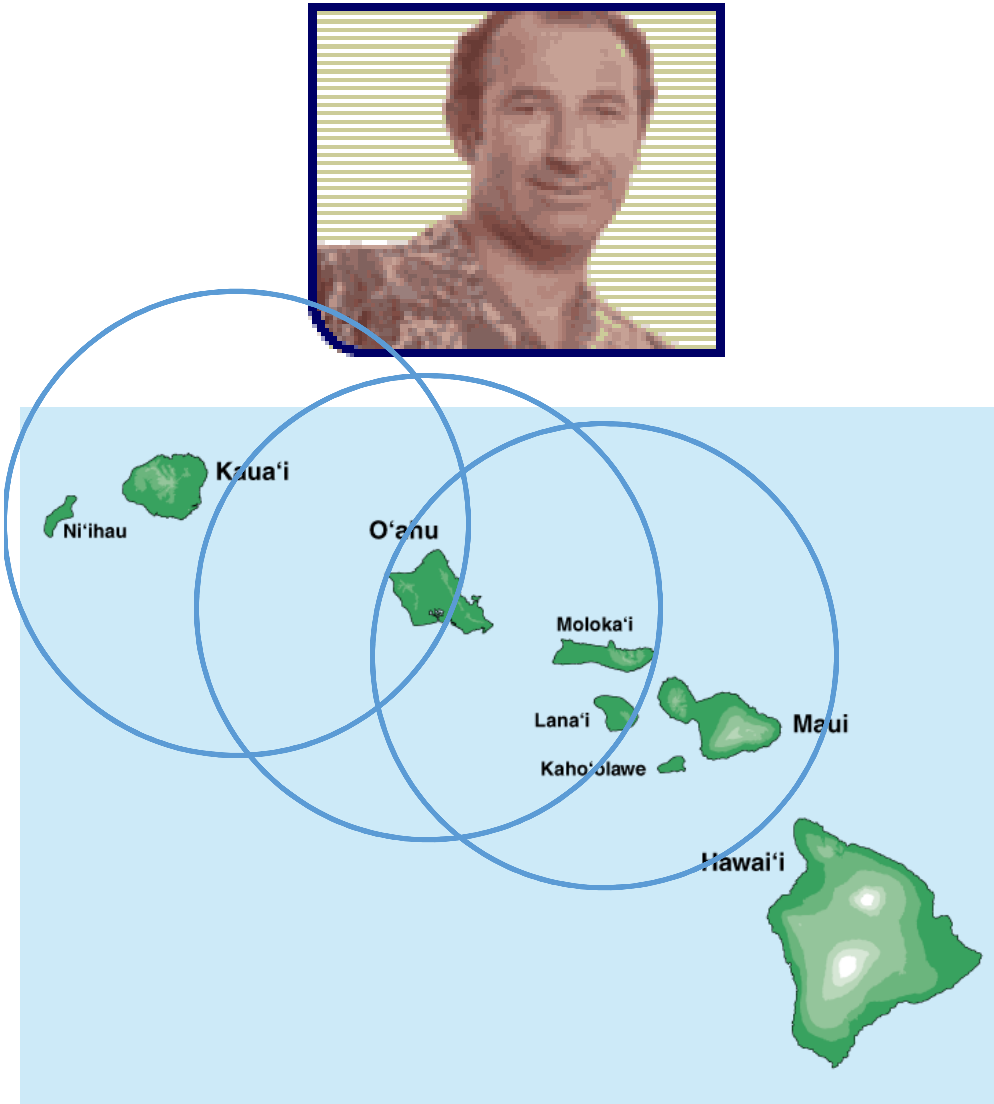
Random Access MAC Protocols

- When node has packet to send
  - Transmit at full channel data rate
  - No *a priori* coordination among nodes
- Two or more transmitting nodes $\Rightarrow$ collision
  - Data lost
- Random access MAC protocol specifies:
  - How to detect collisions
  - How to recover from collisions
- Examples
  - ALOHA and Slotted ALOHA
  - CSMA, CSMA/CD, CSMA/CA (wireless, covered later)

Key Ideas of Random Access

- Carrier sense
  - *Listen before speaking, and don't interrupt*
  - Checking if someone else is already sending data
  - … and waiting till the other node is done
- Collision detection
  - *If someone else starts talking at the same time, stop*
    - ***But make sure everyone knows there was a collision!***
  - Realizing when two nodes are transmitting at once
  - …by detecting that the data on the wire is garbled
- Randomness
  - *Don't start talking again right away*
  - Waiting for a random time before trying again

Where it all Started: AlohaNet



- Norm Abramson left Stanford in 1970
- ***So he could surf!***
- Set up first data communication system for Hawaiian islands
- Hub at U. Hawaii, Oahu
- Had two radio channels:
  - Random access:
    - Sites sending data
  - Broadcast:
    - Hub rebroadcasting data

Aloha Signaling

- Two channels: random access, broadcast
- Sites send packets to hub (random)
  - If received, hub sends ACK (random)
  - If not received (due to collision), site resends
- Hub sends packets to all sites (broadcast)
  - Sites can receive even if they are also sending
- Questions:
  - When do you resend? Resend with probability p
  - How does this perform? Need a clean model….

Slotted ALOHA

Assumptions

- All frames same size
- Time divided into equal slots (time to transmit a frame)
- Nodes are synchronized
- Nodes begin to transmit frames only at start of slots
- If multiple nodes transmit, nodes detect collision

Operation

- When node gets fresh data, transmits in next slot
- No collision: success!
- Collision: node retransmits with probability **p** until success

Slot-by-Slot Example

Efficiency of Slotted Aloha

- Suppose N stations have packets to send
  - Each transmits in slot with probability *p*

- Probability of successful transmission:

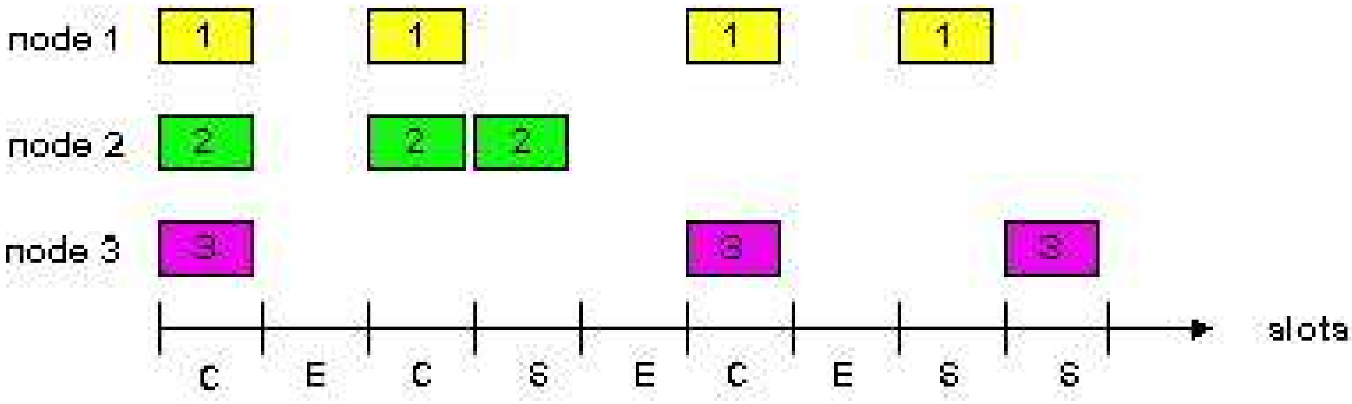
  by a particular node i: $S_i = p\,(1-p)^{(N-1)}$

  by any of N nodes: $S = N\,p\,(1-p)^{(N-1)}$

- What value of p maximizes prob. of success:
  - For fixed p, S ➔ 0 as N increases
  - But if p = 1/N, then S ➔ 1/e = 0.37 as N increases

- Max efficiency is only slightly greater than 1/3!

Pros and Cons of Slotted Aloha



Pros

- Single active node can continuously transmit at full rate of channel
- Highly decentralized: only need slot synchronization
- Simple

Cons

- Wasted slots:
  - Idle
  - Collisions
- Collisions consume entire slot
- Clock synchronization

Improving on Slotted Aloha

- Fewer wasted slots
  - *Need to decrease collisions and empty slots*
- Don’t waste full slots on collisions
  - *Need to decrease time to detect collisions*
- Avoid need for synchronization
  - *Synchronization is hard to achieve*
  - ***And Aloha performance drops if you don’t have slots***

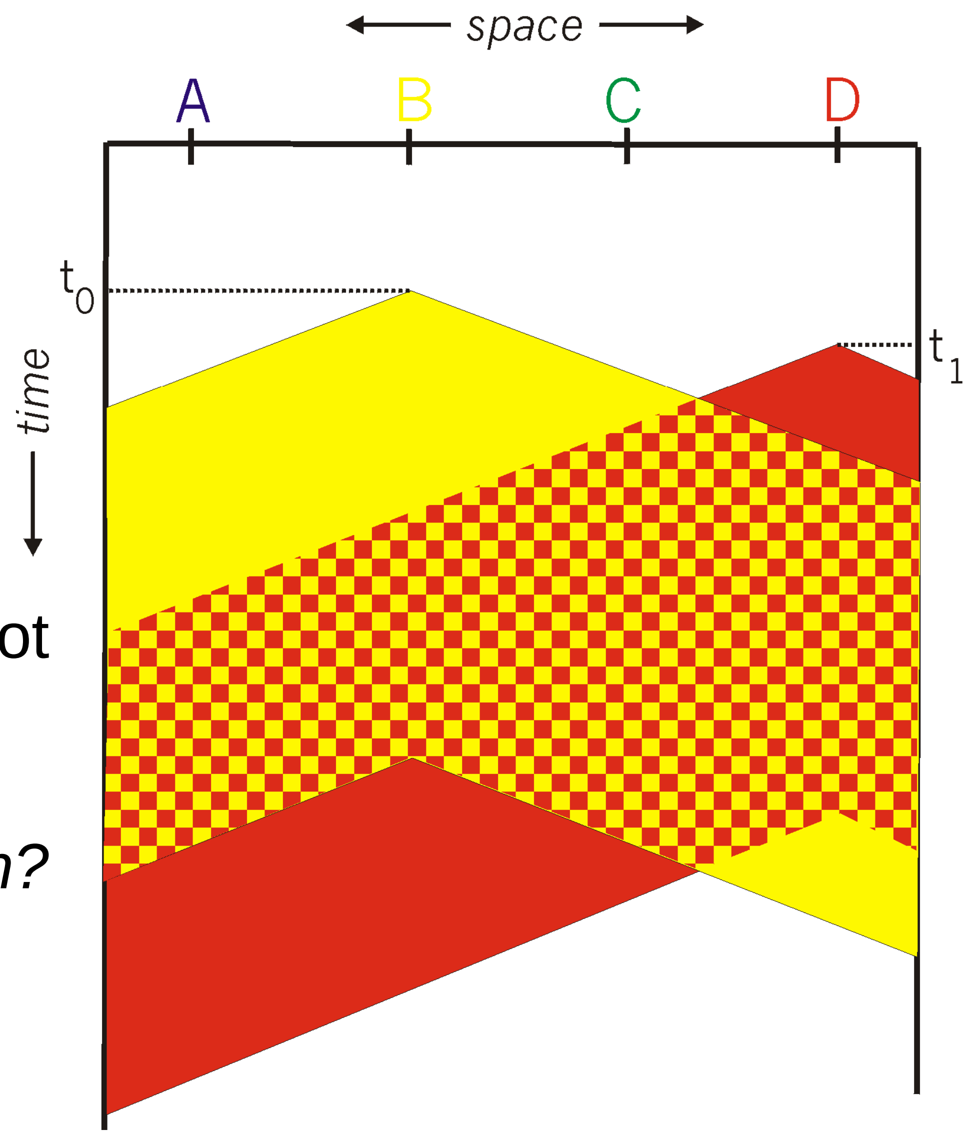
CSMA (Carrier Sense Multiple Access)

- CSMA: listen before transmit
  - If channel sensed idle: transmit entire frame
  - If channel sensed busy, defer transmission
- Human analogy: don’t interrupt others!
- Does this eliminate all collisions?
  - No, because of nonzero propagation delay

CSMA Collisions

Propagation delay: two nodes may not hear each other's before sending.

Would slots hurt or help?



CSMA reduces but does not eliminate collisions

Biggest remaining problem?

Collisions still take full slot!

How do you fix that?

CSMA/CD (Collision Detection)

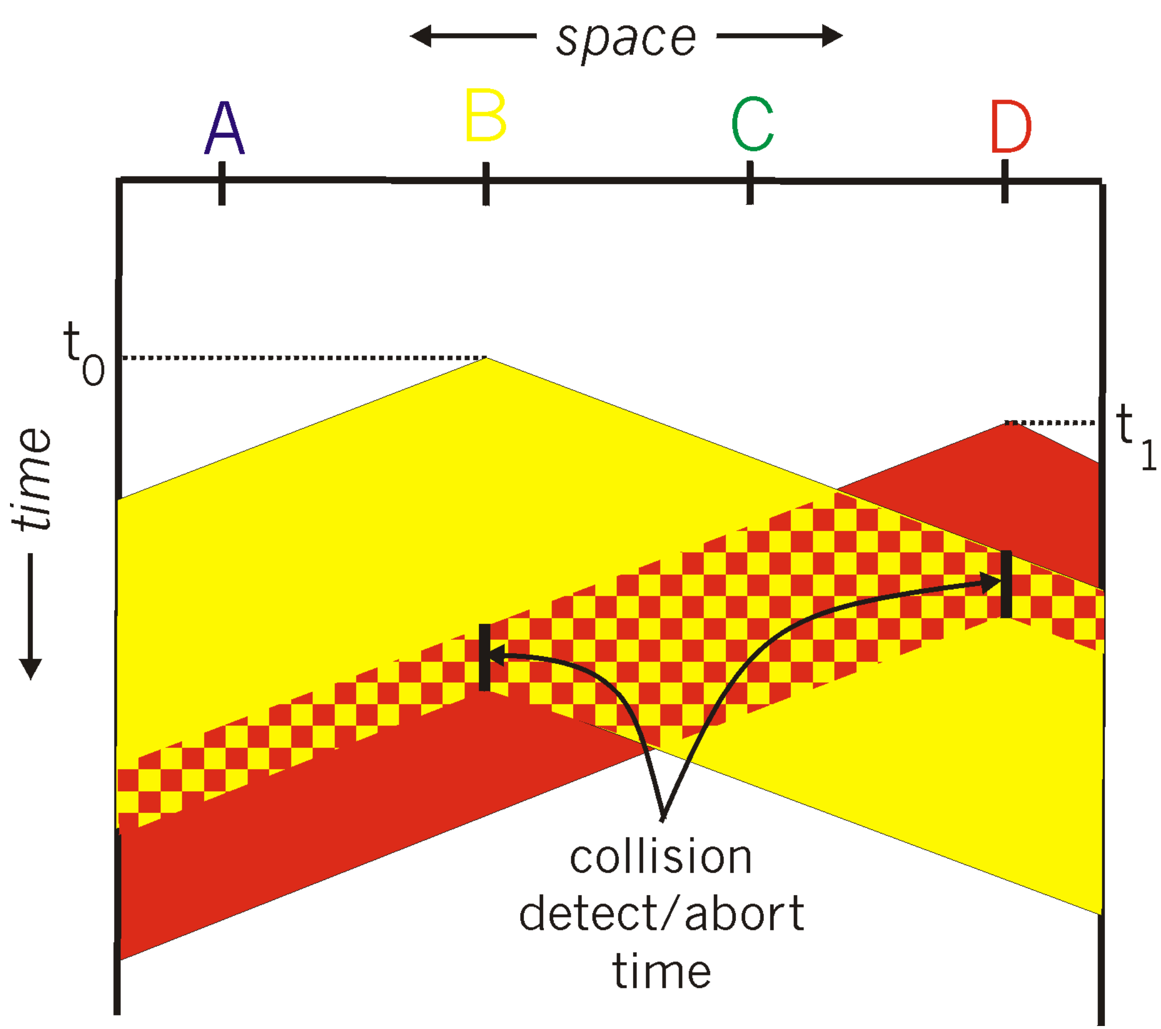
- CSMA/CD: carrier sensing, deferral as in CSMA
  - **Collisions detected within short time**
  - Colliding transmissions aborted, reducing wastage
- Collision detection easy in wired LANs:
  - Compare transmitted, received signals
- Collision detection difficult in wireless LANs:
  - Reception shut off while transmitting (well, perhaps not)
  - Not perfect broadcast (limited range) so collisions local
  - Leads to use of *collision avoidance* instead
    - ***Will discuss in wireless lecture***

CSMA/CD Collision Detection

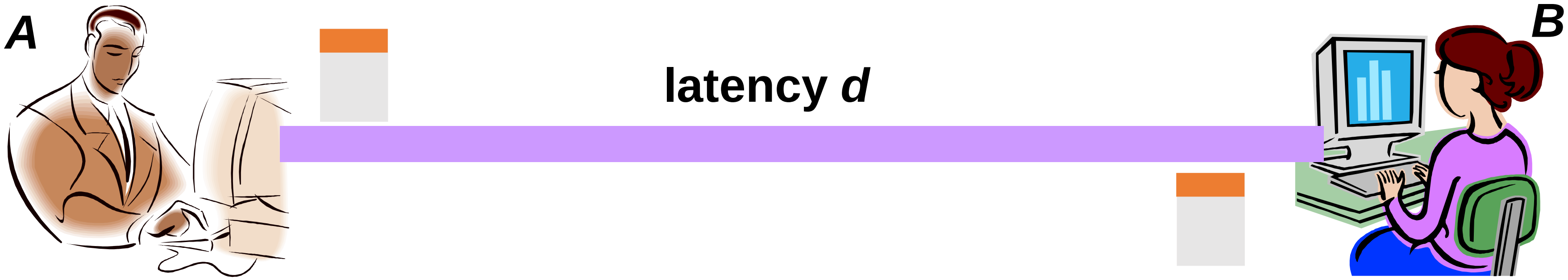
B and D can tell that collision occurred.

Note: for this to work, need restrictions on minimum frame size and maximum distance.

Why?

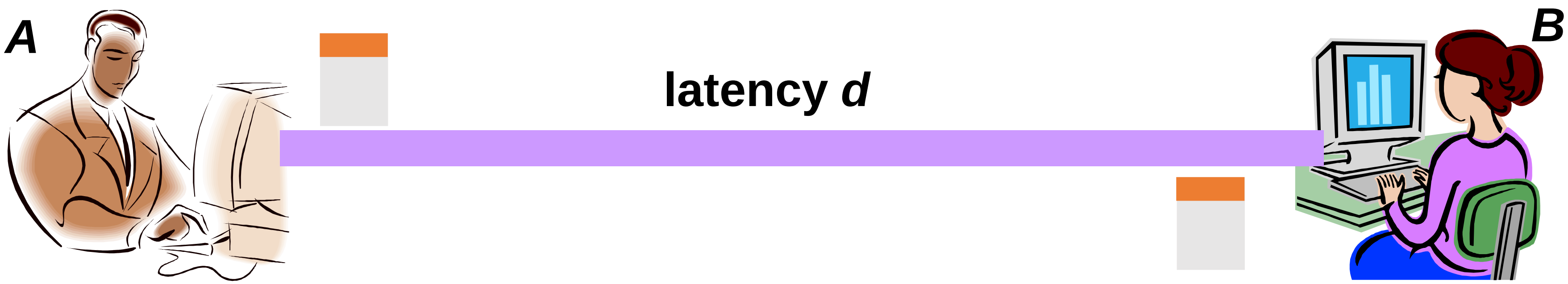


Limits on CSMA/CD Network Length



- Latency depends on physical length of link
  - Time to propagate a packet from one end to the other
- Suppose *A* sends a packet at time ***t***
  - And *B* sees an idle line at a time just before ***t*+*d***
  - … so *B* happily starts transmitting a packet
- *B* detects a collision, and sends jamming signal
  - But *A* can't see collision until ***t*+*2d***

Limits on CSMA/CD Network Length



- *A* needs to wait for time ***2d*** to detect collision
  - So, *A* should keep transmitting during this period
  - … and keep an eye out for a possible collision
- Imposes restrictions. E.g., for 10 Mbps Ethernet:
  - Maximum length of the wire: 2,500 meters
  - Minimum length of a frame: 512 bits (64 bytes)
    - 512 bits = 51.2 μsec (at 10 Mbit/sec)
    - For light in vacuum, 51.2 μsec ≈ 15,000 meters vs. 5,000 meters "round trip" to wait for collision
  - What about 10Gbps Ethernet?

Performance of CSMA/CD

- Time wasted in collisions
  - Proportional to distance d
- Time spend transmitting a packet
  - Packet length p divided by bandwidth b
- Rough estimate for efficiency (K some constant)

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

- Note:
  - For l
  - As bandwidth increases, E decreases
  - That is why high-speed LANs are all switched

Ethernet Multiple Access

First widely deployed multiple access

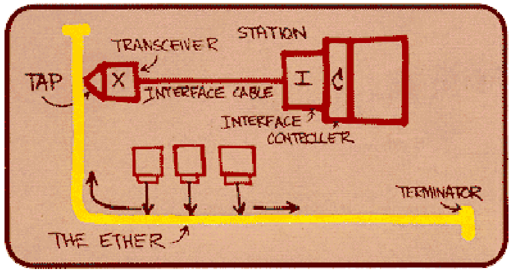
Benefits of Ethernet

- Easy to administer and maintain
- Inexpensive
- Increasingly higher speed
- Evolvable!

Evolution of Ethernet

- Changed everything except the frame format
  - From single coaxial cable to hub-based star
  - From shared media to switches
  - From electrical signaling to optical
- Lesson #1
  - The right interface can accommodate many changes
  - Implementation is hidden behind interface
- Lesson #2
  - Really hard to displace the dominant technology
  - Slight performance improvements are not enough

Ethernet: CSMA/CD Protocol



- **Carrier sense**: wait for link to be idle
- **Collision detection**: listen while transmitting
  - No collision: transmission is complete
  - Collision: abort transmission & send **jam** signal
- **Random access**: binary exponential back-off
  - After collision, wait a random time before trying again
  - After m^{th} collision, choose K randomly from {0, …, 2^m-1}
  - … and wait for K*512 bit times before trying again
    - Using min packet size as "slot"
    - **If transmission occurring when ready to send, wait until end of transmission (CSMA)**

Binary Exponential Backoff (BEB)

- Think of time as divided in slots
- After each collision, pick a slot randomly within next 2^m slots
  - Where m is the number of collisions since last successful transmission

- Questions:
  - Why backoff?
  - Why random?
  - Why 2^m?
  - Why not listen while waiting?

Behavior of BEB Under Light Load

Look at collisions between two nodes

- First collision: pick one of the next two slots
  - Chance of success after first collision: 50%
  - Average delay 1.5 slots
- Second collision: pick one of the next four slots
  - Chance of success after second collision: 75%
  - Average delay 2.5 slots
- In general: after m^{th} collision
  - Chance of success: $1-2^{-m}$
  - Average delay (in slots): $½ + 2^{(m-1)}$

BEB: Theory vs Reality

In theory, there is no difference between theory and practice. But, in practice, there is.

BEB Reality

- Performs well (far from optimal, but no one cares)
  - *Large packets are ~23 times as large as minimal slot*
- Is mostly irrelevant
  - *Almost all current ethernets are* ***switched***

BEB Theory

- A very interesting algorithm
- Stability for finite N only proved in 1985
  - Ethernet can handle nonzero traffic load without collapse
    - Greenberg et al. (AT&T)
- All backoff algorithms unstable for infinite N (1985)
  - Poisson model: infinite user pool, total demand is finite
    - David Aldous (UCB Statistics)
- Not of practical interest, but gives important insight
  - Multiple access should be in your “bag of tricks”

Question

- Two hosts, each with infinite packets to send
- What happens under BEB?
- Throughput high or low?
- Bandwidth shared equally or not?

MAC “Channel Capture” in BEB

- Finite chance that first one to have a successful transmission will never relinquish the channel
  - The other host will *never* send a packet

- Therefore, asymptotically channel is fully utilized and completely allocated to one host

Example

- Two hosts, each with infinite packets to send
  - Slot 1: collision
  - Slot 2: each resends with prob ½
    - Assume host A sends, host B does not
  - Slot 3: A and B both send (collision)
  - Slot 4: A sends with probability ½, B with prob. ¼
    - Assume A sends, B does not
  - Slot 5: A definitely sends, B sends with prob. ¼
    - Assume collision
  - Slot 6: A sends with probability ½, B with prob. 1/8
- Conclusion: if A gets through first, the prob. of B sending successfully halves with each collision

Another Question

- Hosts now have large but finite # packets to send
- What happens under BEB?
- Throughput high or low?

Answer

- Efficiency less than one, no matter how many packets

- Time you wait for loser to start is proportional to time winner was sending….

Different Backoff Functions

- Exponential: backoff ~ a^i
  - Channel capture?
  - Efficiency?
- Superlinear polynomial: backoff ~ i^p $p>1$
  - Channel capture?
  - Efficiency?
- Sublinear polynomial: backoff ~ i^p $p\leq1$
  - Channel capture?
  - Efficiency?

Different Backoff Functions

- Exponential: backoff ~ a^i
  - Channel capture *(loser might not send until winner idle)*
  - Efficiency less than 1 *(time wasted waiting for loser to start)*
- Superlinear polynomial: backoff ~ i^p $p>1$
  - Channel capture
  - Efficiency is 1 (for any finite # of hosts N)
- Sublinear polynomial: backoff ~ i^p $p \le 1$
  - No channel capture *(loser not shut out)*
  - Efficiency is less than 1 (and goes to zero for large N)
    - *Time wasted resolving collisions*

Why Do We Care?

- Until this work was done, no one knew about capture, or what properties of the backoff enabled it.
- You don't understand something until you've ***played*** with it. Just getting it to work isn't enough.