EN.601.414/614 Computer Networks

Switched LAN

Xin Jin

Spring 2019 (MW 3:00-4:15pm in Shaffer 301)

JOHNS HOPKINS UNIVERSITY

https://github.com/xinjin/course-net

Agenda

- **Ethernet wrap-up**
- **Putting everything together**

Recap: Switched Ethernet

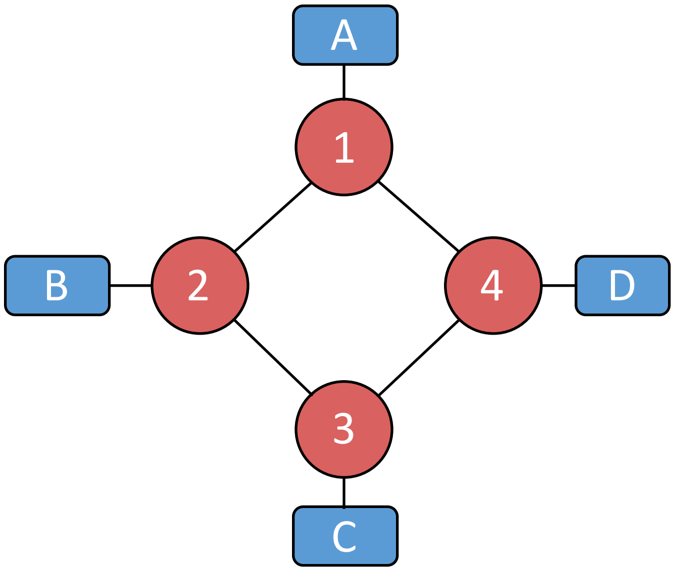
- **Ethernet started as a broadcast medium**
  - Faced broadcast storm in larger setups due to flooding
- **Constraints of switched Ethernet (for backward compatibility)**
  - No changes to end-hosts
  - Maintain plug-n-play aspect

Ethernet topics

- Frames and framing
- Addressing
- **Routing**
- **Forwarding**
- **Discovery**

Flooding (still) leads to loops

- **Example: A wants to broadcast a message**
  - A sends packet to 1
  - 1 Floods to 2 and 4
  - 2 Floods to B and 3
  - 4 Floods to D and 3
  - 3 Floods packet from 2 to C and 4
  - 3 Floods packet from 4 to C and 2
  - 4 Floods packet from 3 to D and 1
  - 2 Floods packet from 3 to B and 1
  - 1 Floods packet from 2 to A and 4
  - 1 Floods packet from 4 to B and 2
  - ....



- **Broadcast storm still happens in a switched network if it contains a cycle of switches**

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

Breaking ties

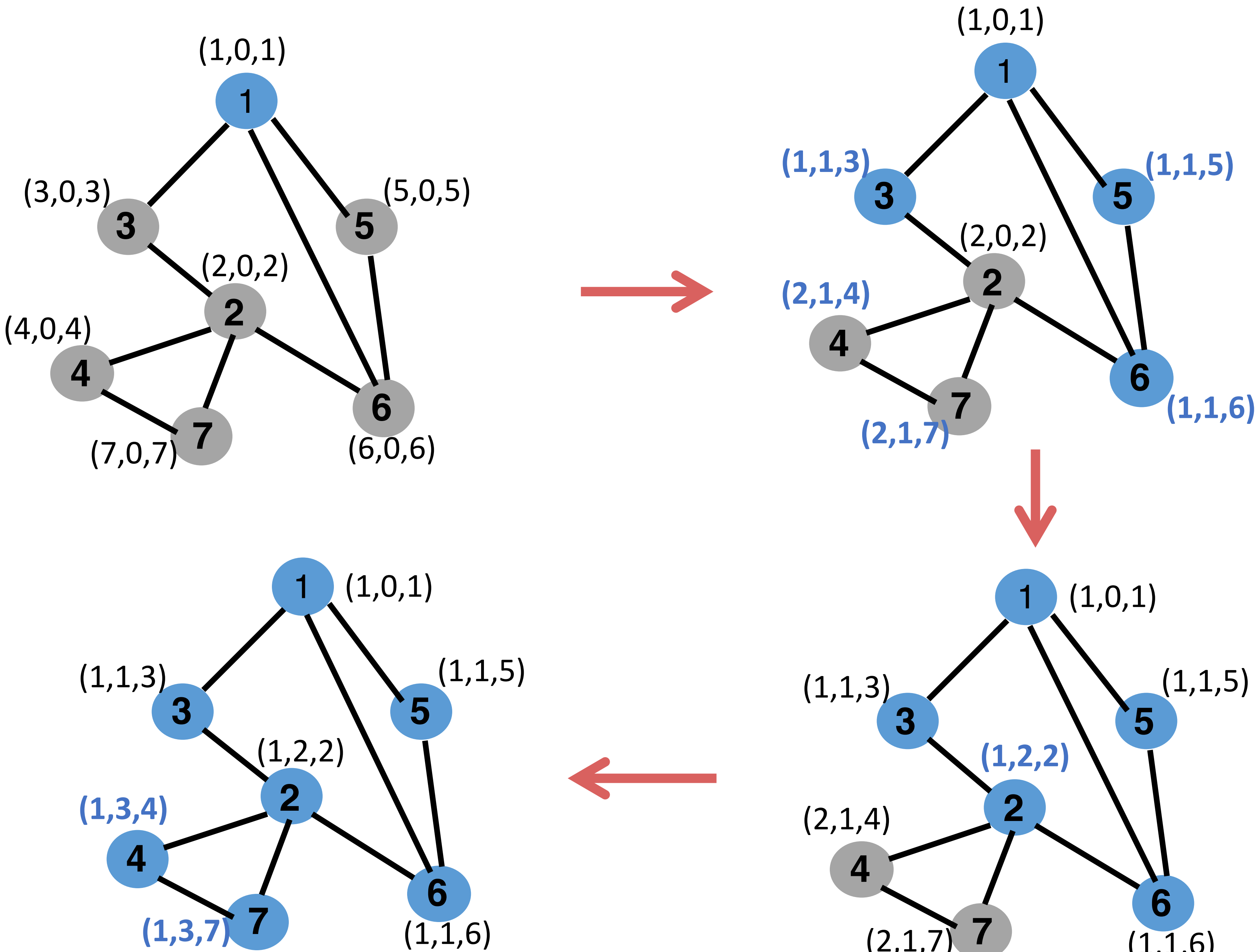
- **When there are multiple shortest paths to the root, choose the path that uses the neighbor switch with the lower ID**
  - One could use any tiebreaking system, but this is an easy one to remember and implement

Constructing a spanning tree

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

Steps in the spanning tree algorithm

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


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

Robust spanning tree algorithm

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

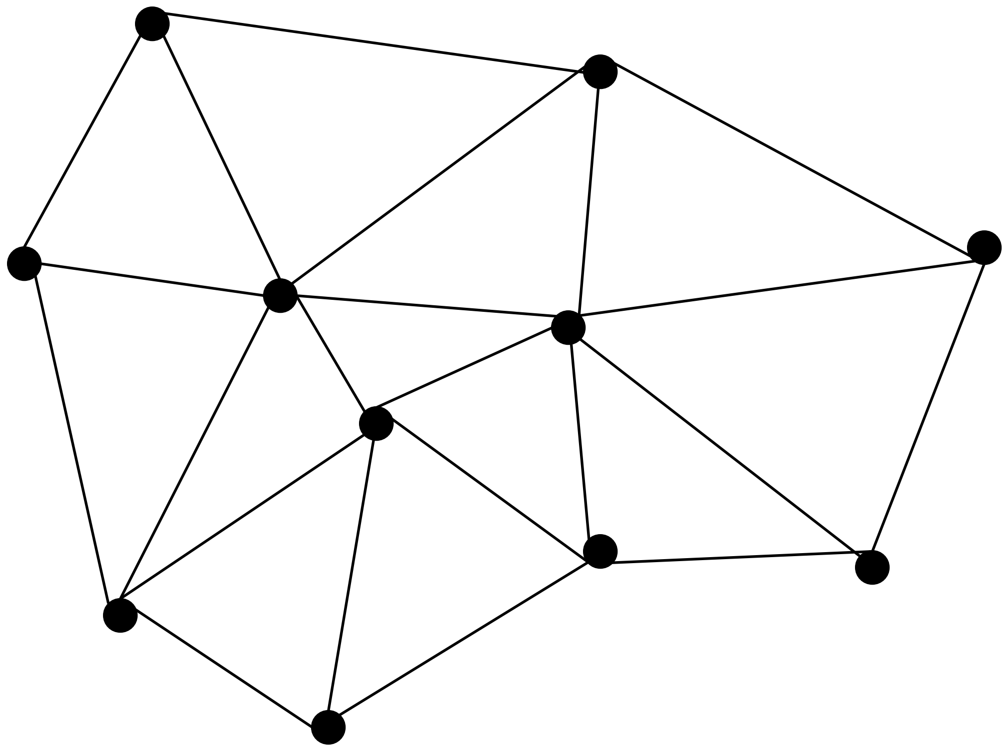
Ethernet topics

- Frames and framing
- Addressing
- Routing
- **Forwarding**
- **Discovery**

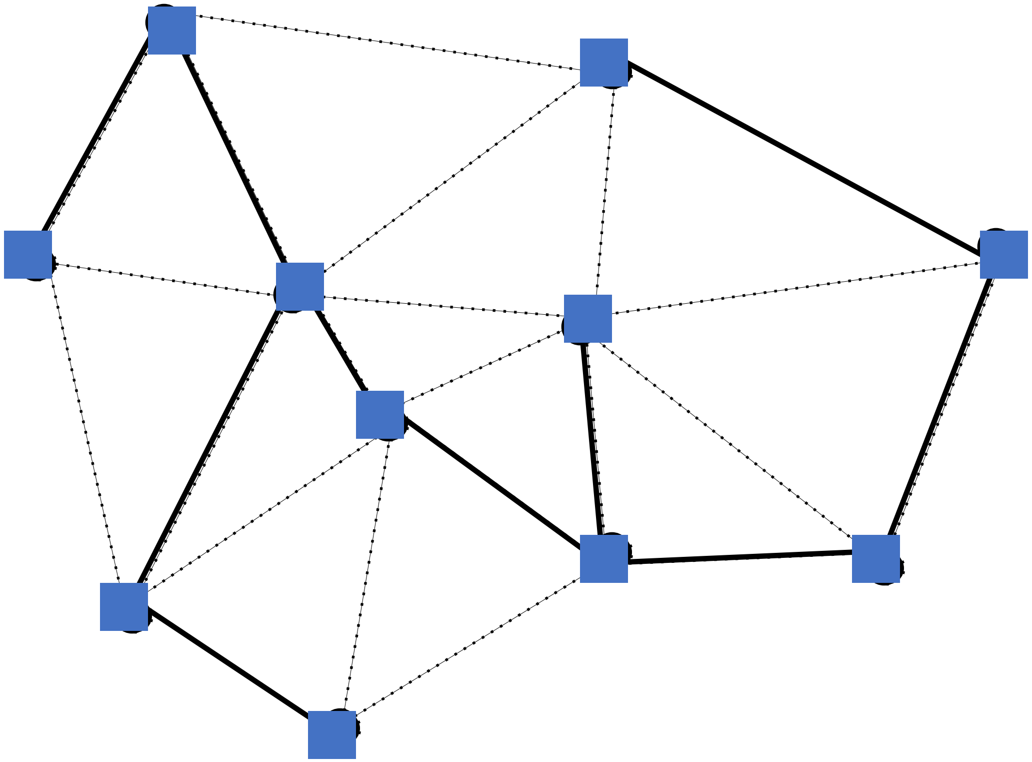
Flooding on a spanning tree

- **Switches flood using the following rule:**
  - (Ignore all ports not on spanning tree!)
  - Originating switch sends packet out all ports
  - When a packet arrives on one incoming port, send it out 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 particular port, it knows what port to use to reach B!
  - ➢ Works because there is only one path to B

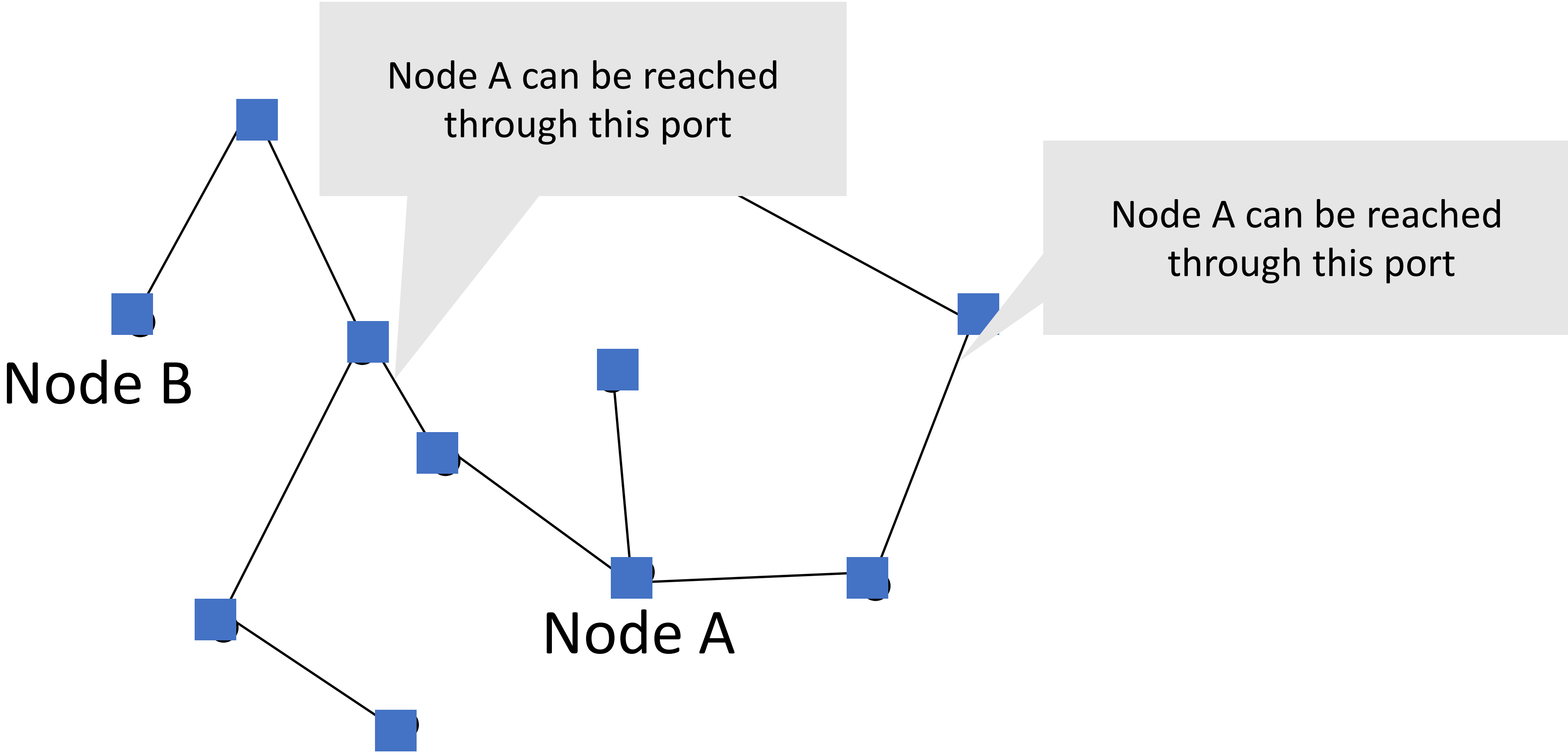
Nodes can “learn” routes

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

General approach

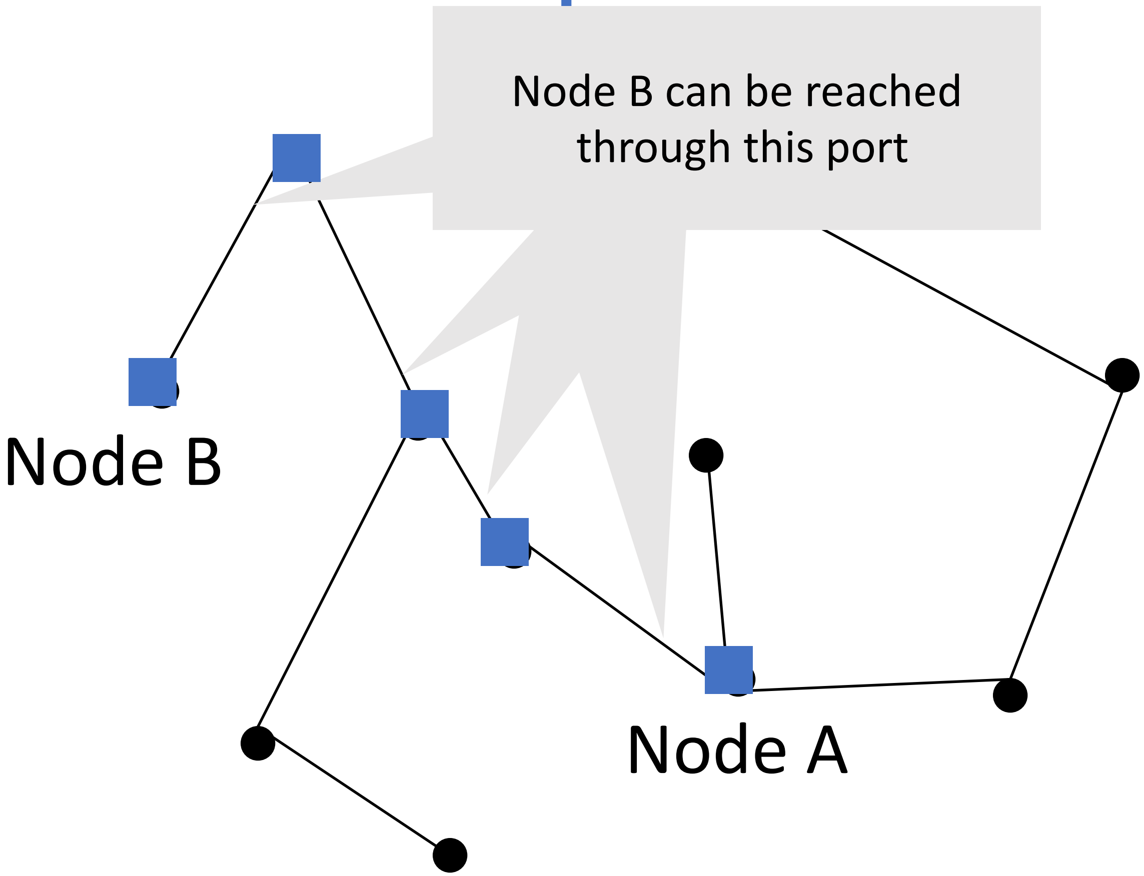
- **Flood first packet to node you are trying to reach**
- **All switches learn where you are**
- **When destination responds, some switches learn where it is…**
  - Only some switches, because packet to you follows direct path, and is not flooded

Learning from flood packets



Once a node has sent a flood message, all other switches know how to reach it….

Node B responds

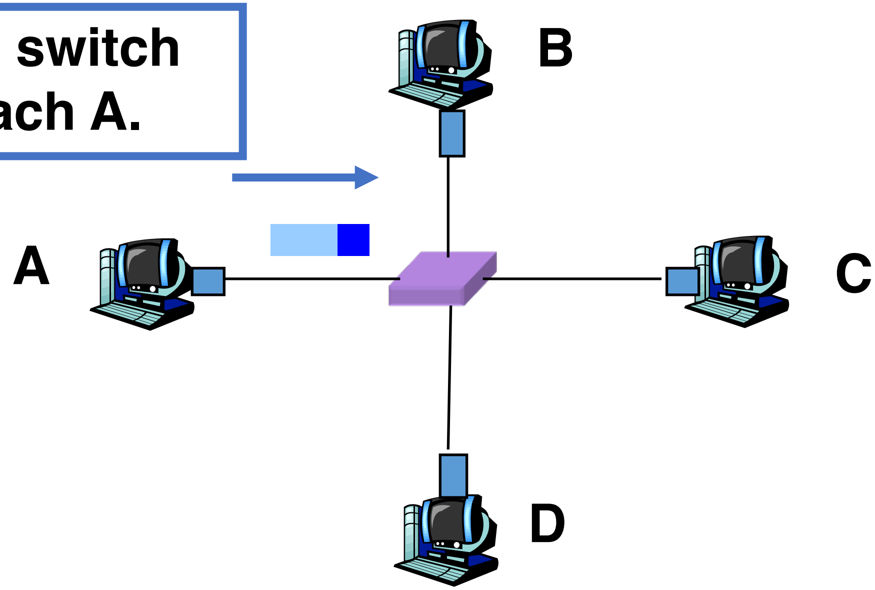


When a node responds, <u>some</u> of the switches learn where it is

Ethernet switches are “self learning”

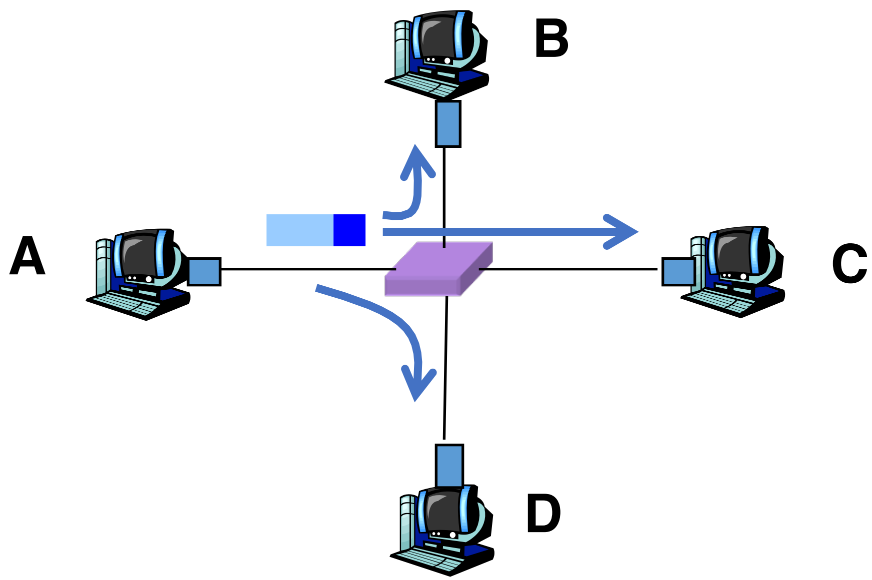
- **When a packet arrives:**
  - ➢Inspect source MAC address, associate with incoming port
  - ➢Store mapping in the switch table
  - ➢Use time-to-live field to eventually forget mapping

Packet tells switch how to reach A.



Self learning: Handling misses

- When 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 source of flood**
- **No route “computation”**
- **Forwarding entries a consequence of traffic pattern**

Contrast

IP

- Packets forwarded on all available links
- Addresses can be aggregated
- 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(+ spanning tree for floods)

Strengths of Ethernet's approach

- **Plug-n-Play: zero-configuration / self-***
- **Simple**
- **Cheap?**

Weaknesses of Ethernet’s 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
  - Rebuilt spanning tree may be quite different
- **Slow to react to host movement**
  - Entries must time out
- **Poor predictability**
  - Location of root and traffic pattern determines forwarding efficiency

Link layer topics

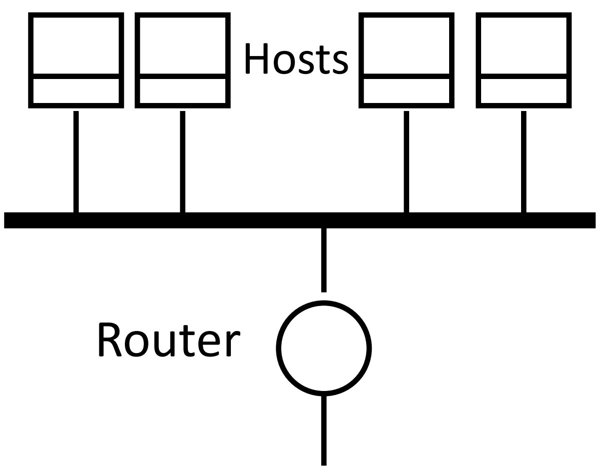
- Frames and framing
- Addressing
- Routing
- Forwarding
- **Discovery and bootstrapping**

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**
  - ARP → Address Resolution Protocol
  - DHCP → Dynamic Host Configuration Protocol
  - Confined to a single local-area network (LAN)
  - Rely on broadcast capability



ARP and DHCP

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

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

- **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

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

DHCP: Operation

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

DHCP: Operation

- **One or more local DHCP servers maintain required information**
- **Client broadcasts a DHCP discovery message**
- **One or more DHCP servers responds with a DHCP "offer" message**
- **Client broadcasts a DHCP request message**
  - ➢Specifies which offer it wants
  - ➢Echoes accepted parameters
  - ➢Other DHCP servers learn they were not chosen

DHCP: Operation

- **One or more local DHCP servers maintain required information**
- **Client broadcasts a DHCP discovery message**
- **One or more DHCP servers responds with a DHCP “offer” message**
- **Client broadcasts a DHCP request message**
- **Selected DHCP server responds with an ACK**

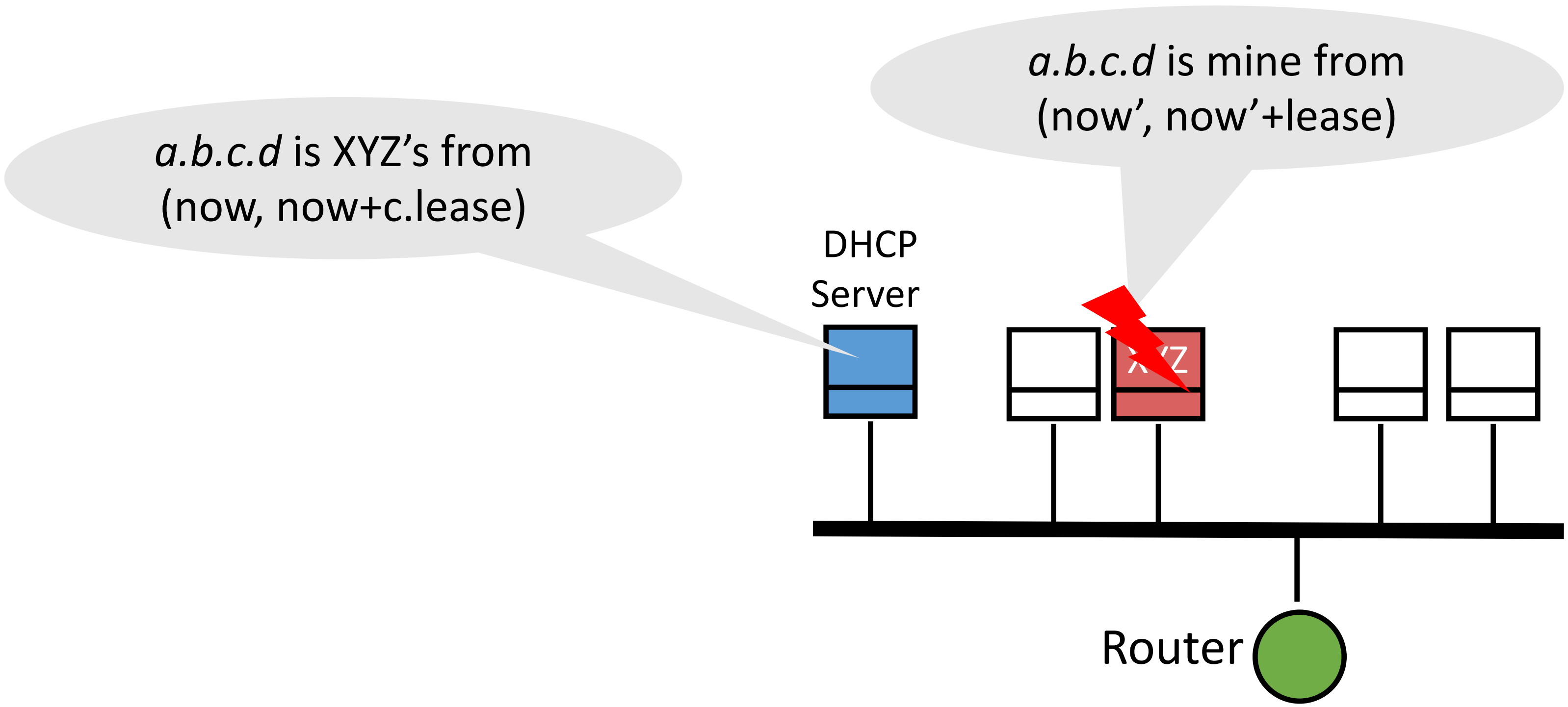
DHCP: Operation

- **One or more local DHCP servers maintain required information**
- **Client broadcasts a DHCP discovery message**
- **One or more DHCP servers responds with a DHCP “offer” message**
- **Client broadcasts a DHCP request message**
- **Selected DHCP server responds with an ACK**
- **DHCP “relay agents” used when the DHCP server is not on the same broadcast domain**

DHCP uses “soft state”

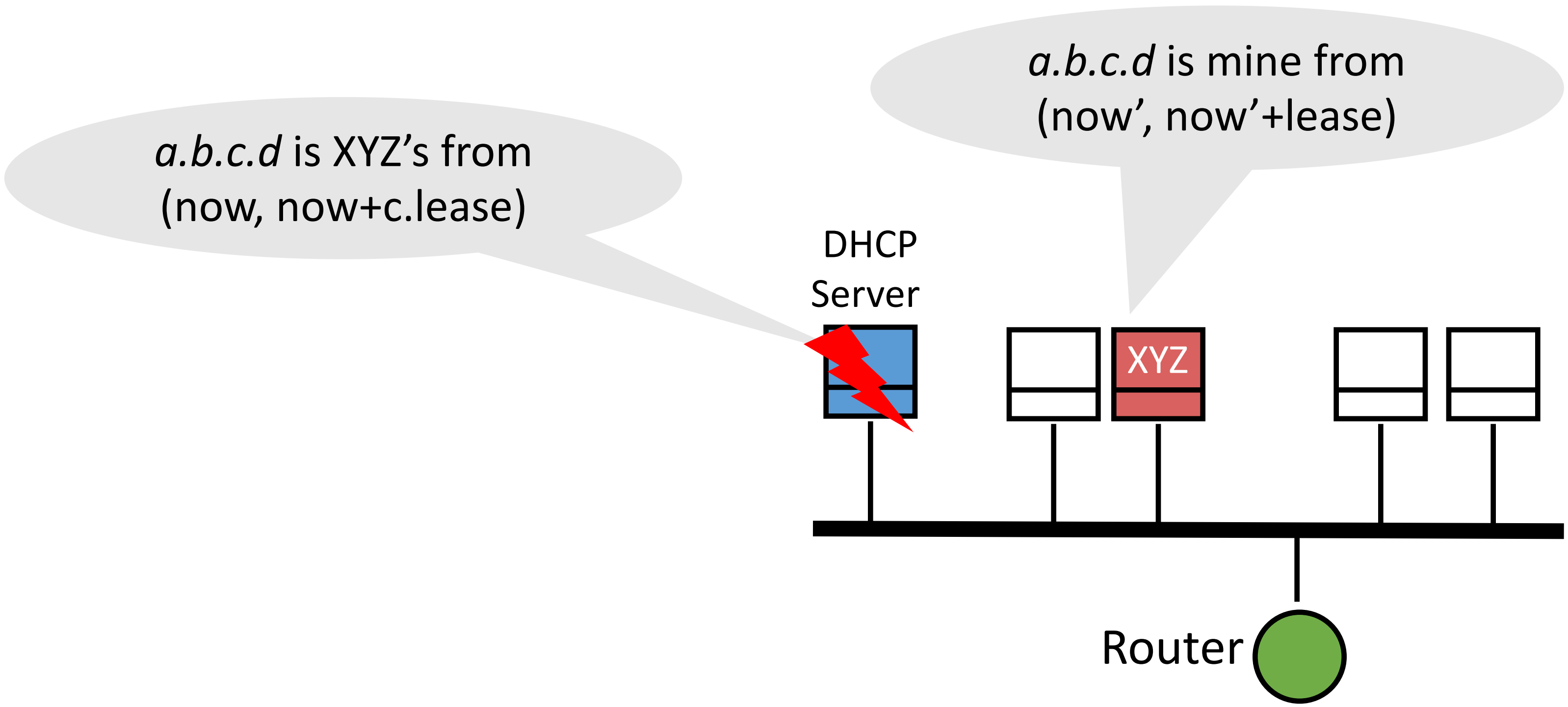
- **Soft state: if not refreshed, state is forgotten**
  - ➢ Hard state: allocation/revocation is deliberate
- **Implementation:**
  - ➢ Address allocations have a lease period
  - ➢ Server sets a timer for each allocation
  - ➢ Client must request a refresh before lease expires
  - ➢ Server resets timer when a refresh arrives and ACKs
    - OR reclaims allocated address when timer expires
- **Simple, yet robust under failure**

Soft state under failure



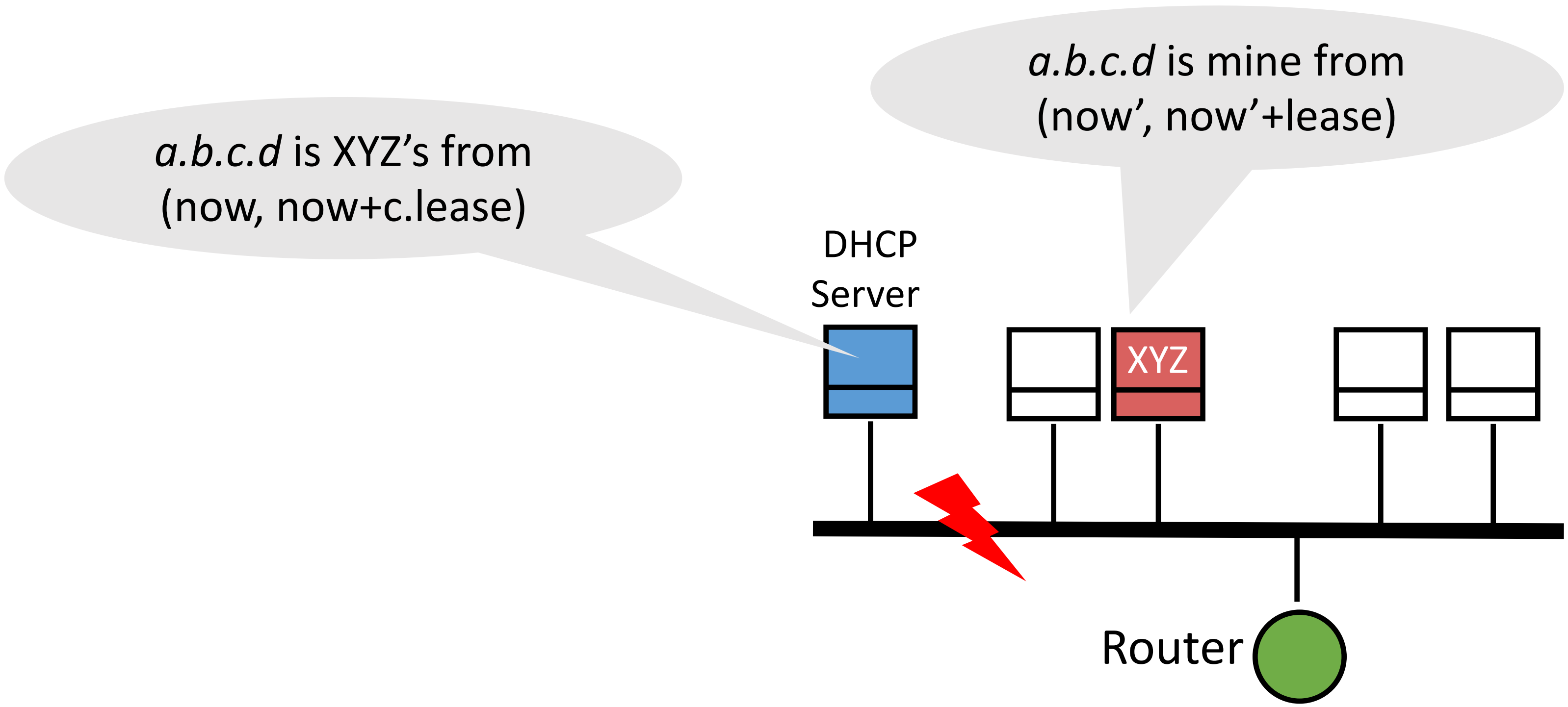
- **What happens when host XYZ fails?**
  - Refreshes from XYZ stop
  - Server reclaims a.b.c.d after O(lease period)

Soft state under failure



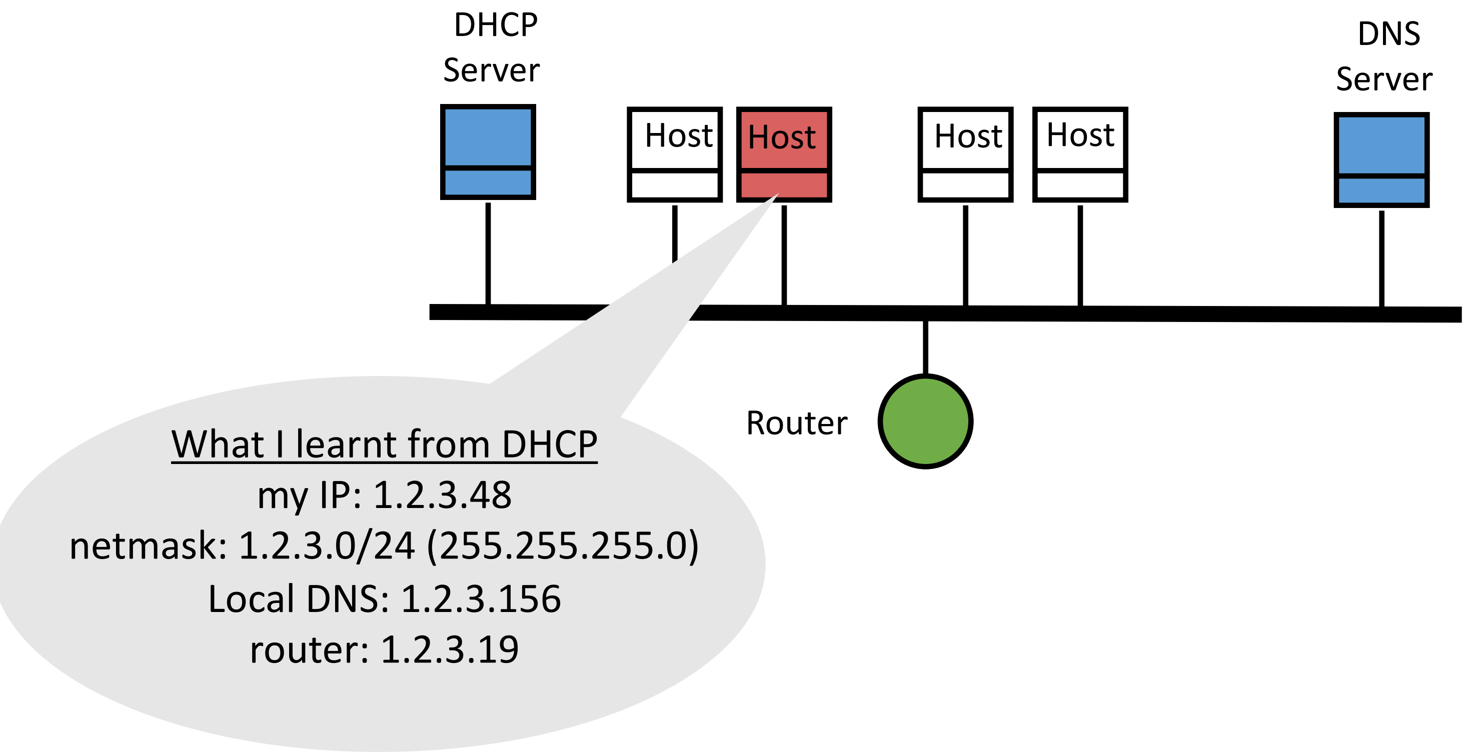
- **What happens when server fails?**
  - ACKs from server stop
  - XYZ releases address after O(lease period); send new request
  - A new DHCP server can come up from a `cold start' and we are back on track in ~lease time

Soft state under failure

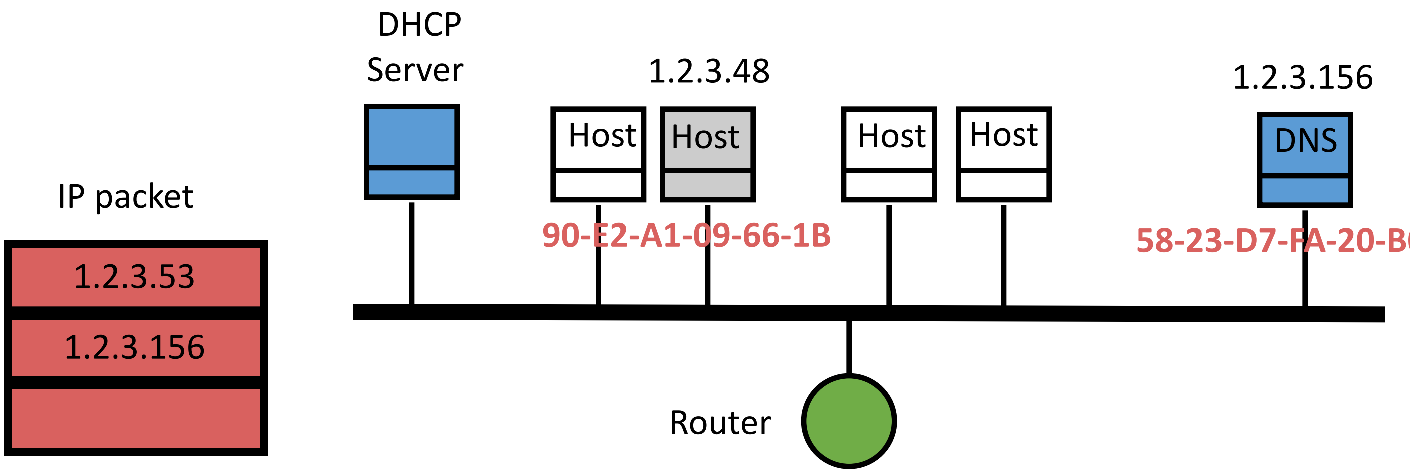


- **What happens if the network fails?**
  - ➢ Refreshes and ACKs don't get through
  - ➢ XYZ release address; DHCP server reclaims it

Are we there yet?



Sending packets over link Layer



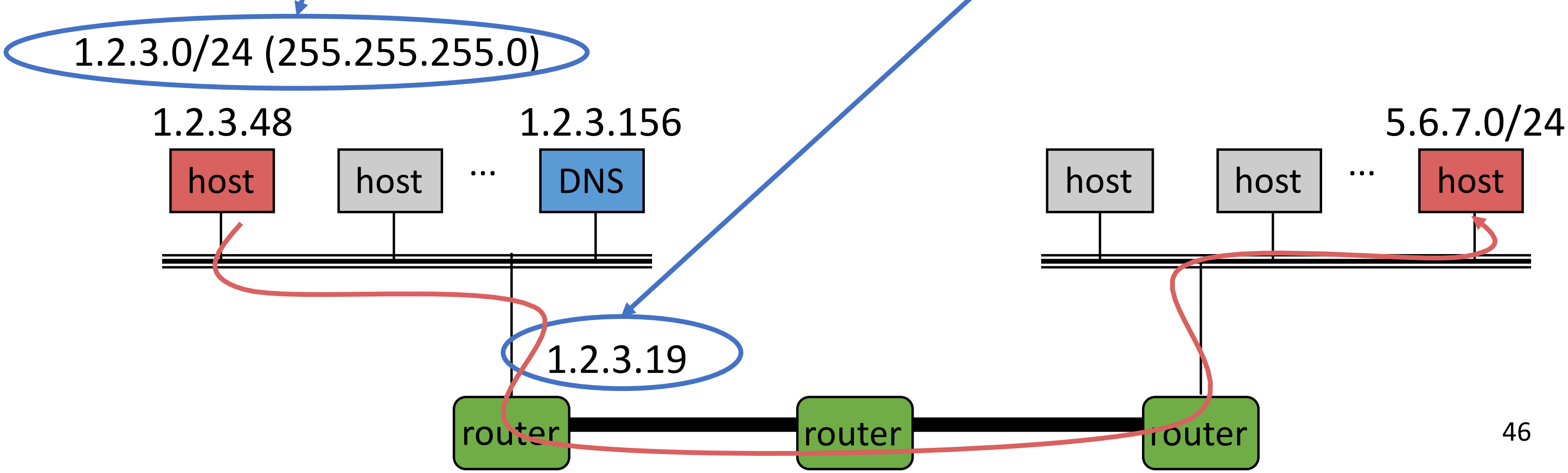
- **Link layer only understands MAC addresses**
  - ➢Translate the destination IP address to MAC address
  - ➢Encapsulate the IP packet in a link-level (Ethernet) frame

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 dest. IP address to dest. MAC address
  - Encapsulate (IP) data packet with MAC header; xmit
- **What if IP address not in the table?**
  - Sender broadcasts: Who has IP address 1.2.3.156?
  - Receiver replies: MAC address 58-23-D7-FA-20-B0
  - Sender caches result in its ARP table

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



Key ideas in both ARP and DHCP

- **Broadcasting: Can use broadcast to make contact**
  - Scalable because of limited size
- **Caching: remember the past for a while**
  - Store the information you learn to reduce overhead
- **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 in the face of unpredictable change

ID resolution in the networking stack

Layer	Examples	Structure	Configuration	Resolution Service
App. Layer	cse.umich.edu	Organizational hierarchy	~ manual	DNS
Network Layer	123.45.6.78	topological hierarchy	DHCP	ARP
Link layer	45-CC-4E-12-F0-97	vendor (flat)	hard-coded	

Discovery mechanisms

- **We have seen two approaches**
  - ➢ Broadcast (ARP, DHCP)
    - Flooding does not scale
    - No centralized point of failure
    - Zero configuration
  - ➢ Directory service (DNS)
    - No flooding / scalable
    - Root of the directory is vulnerable (caching is key)
    - Needs configuration to bootstrap (local, root servers, etc.)

Summary

- **Spanning tree enables Ethernet to efficiently flood a network to learn routes while forwarding packets**
- **DHCP and ARP form the discovery backplane of networking and make everything work together**

- **Next lecture: Wireless and Security**

Thanks!
Q&A