Cycle 2: Enterprise Networking

CS 436: Fall 2017

Matthew Caesar

http://www.cs.Illinois.edu/~caesar/cs436

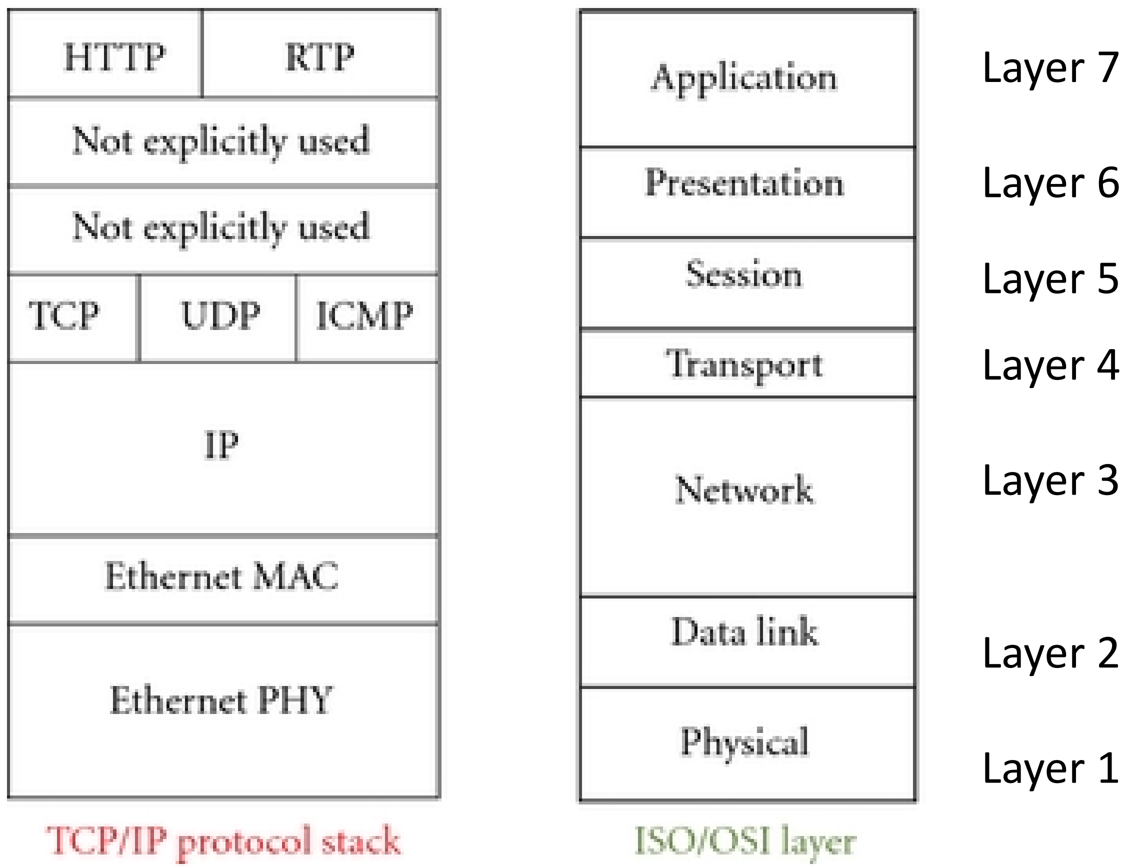
Lab 1 Recap

- Skills acquired: Network configuration, Layer 3 network administration, protocols (BGP, OSPF), Cisco/Cisco IOS

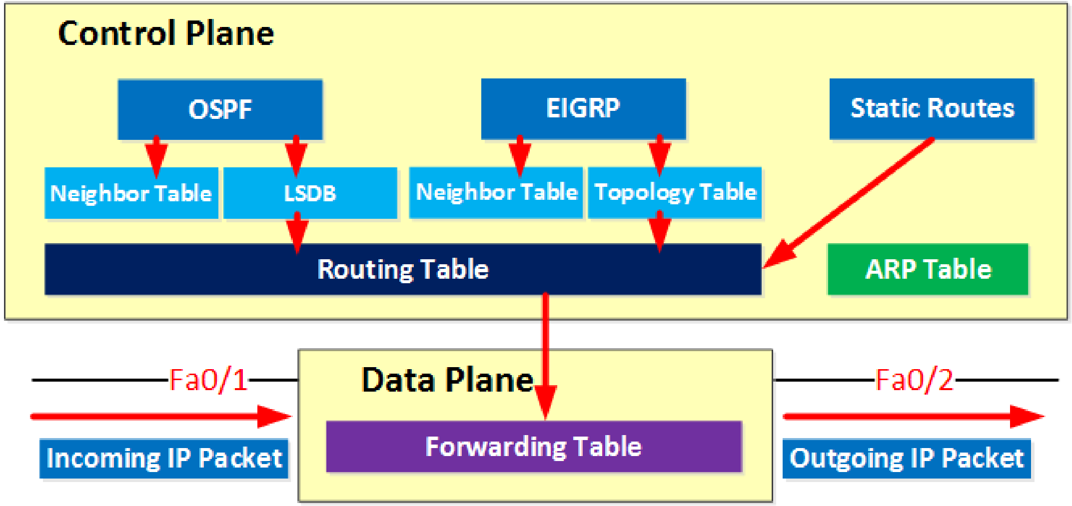
- Demos today and Wednesday – Sign up if you have not

- How to go deeper:
  - GNS3: play around with more protocols, vendors
  - Lurk at https://www.reddit.com/r/networking/
  - CCNA/CCIE certification

Conceptual network architecture (TCP/IP vs ISO)



Data vs Control planes

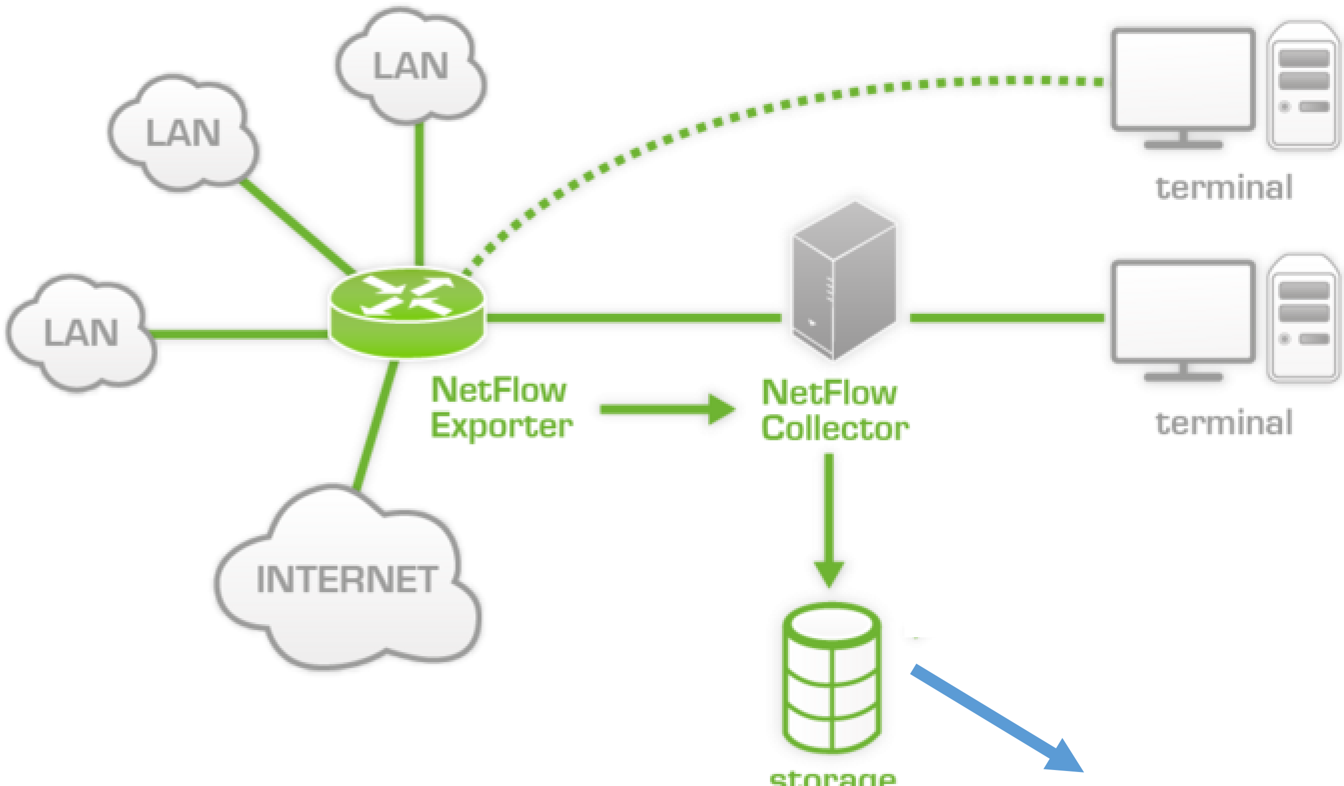


- Two kinds of packets in the internet:
- **Data packets** – traffic computers send
  - Goes through the router, not to the router
  - Video streaming, VoIP, downloading web pages, bittorrent, etc.
- **Control packets** – packets used to "control" the internet
  - Destined to or originated from router itself
  - Routing protocol messages (BGP, OSPF, etc), session setup messages
- Can think of "planes" associated with each of these functions
  - Control vs data plane
  - Layer 2 control plane more plug-and-play, can focus more on data plane in Lab 2

Lab 2 Overview

- This lab is less about configuration
  - Layer 2 is “plug and play”!
- Focus on understanding traffic and network inputs
  - Netflow (large-scale packet monitoring)
  - Tcpdump/pcap (edge packet monitoring)
  - Routing traces (BGP feeds)

Lab 2 Overview: Netflow



Source IP	Destination IP	Application	Source Port	Dest . Port	Protocol	DSCP Name	TCP FLAGS	Traffic	No of Packets	NextH
59.93.161.133	203.199.206.240	TCP_App	3593	2967	TCP	Default	S	2.73 KB	57	203.199
59.93.161.133	203.199.206.192	TCP_App	3543	2967	TCP	Default	S	2.73 KB	57	203.199
59.93.161.133	203.199.206.208	TCP_App	3560	2967	TCP	Default	S	2.73 KB	57	203.199
59.93.161.133	203.199.206.224	TCP_App	3577	2967	TCP	Default	S	2.73 KB	57	203.199
59.93.161.133	203.199.206.243	TCP_App	3596	2967	TCP	Default	S	2.73 KB	57	203.199

Cycle 2: Enterprise Networking

CS 436: Fall 2017

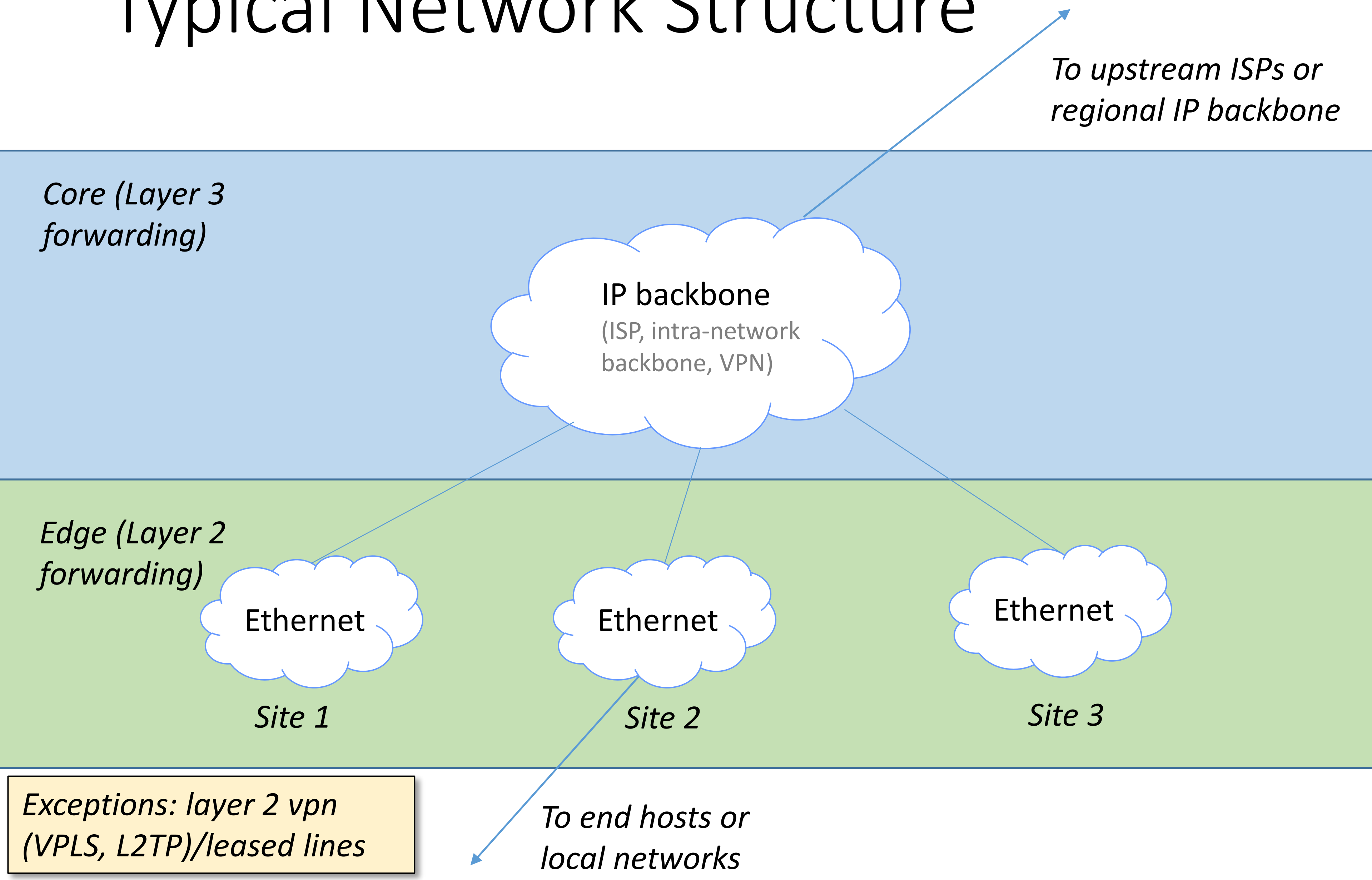
Matthew Caesar

http://www.cs.Illinois.edu/~caesar/cs436

Motivating Questions

- We are running out of address space at our Miami branch office. What are our options?
- New legislation requires companies storing PCI data to log IP sources of all accesses – how to do that?
- Judge requires you to start sending warning notices to people bittorrenting TV shows on your network – how to do that?
- Here is a budget of $100,000. Build a network for our Chicago branch office.

Typical Network Structure



Ethernet (layer 2) vs IP (layer 3) routing

- Ethernet is “plug and play”
  - Easy to build networks
  - May optionally configure ACLs, SSIDs (wireless), spanning tree properties, etc.
- Each host assigned a topology-independent *MAC address*
  - E.g., 00-14-22-01-23-45
- Uses “dumb” flooding (broadcast) to get data packets where they need to go
  - Less efficient than link-state (unicast)

Command Prompt

```
C:\Users\mccae>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : LAPTOP-M33LKCPO
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : SJC-WIFI-DHCP1

Ethernet adapter Ethernet:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) Ethernet Connection (4) I219-LM
   Physical Address. . . . . . . . . : 54-EE-75-DB-6D-44
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Microsoft Wi-Fi Direct Virtual Adapter
   Physical Address. . . . . . . . . : 1C-4D-70-72-4F-12
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . : SJC-WIFI-DHCP1
   Description . . . . . . . . . . . : Intel(R) Dual Band Wireless-AC 8265
   Physical Address. . . . . . . . . : 1C-4D-70-72-4F-11
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::3df4:a06e:801d:3b1%15(Preferred)
   IPv4 Address. . . . . . . . . . . : 10.47.13.145(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.128.0
   Lease Obtained. . . . . . . . . . : Thursday, September 14, 2017 6:35:18 AM
   Lease Expires . . . . . . . . . . : Thursday, September 14, 2017 9:42:30 AM
   Default Gateway . . . . . . . . . : 10.47.1.1
   DHCP Server . . . . . . . . . . . : 10.47.1.100
   DHCPv6 IAID . . . . . . . . . . . : 85740912
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-21-2B-72-CC-54-EE-75-DB-6D-44
   DNS Servers . . . . . . . . . . . : 8.8.8.8
                                       8.8.4.4
   NetBIOS over Tcpip. . . . . . . . : Enabled

Ethernet adapter Bluetooth Network Connection:
```

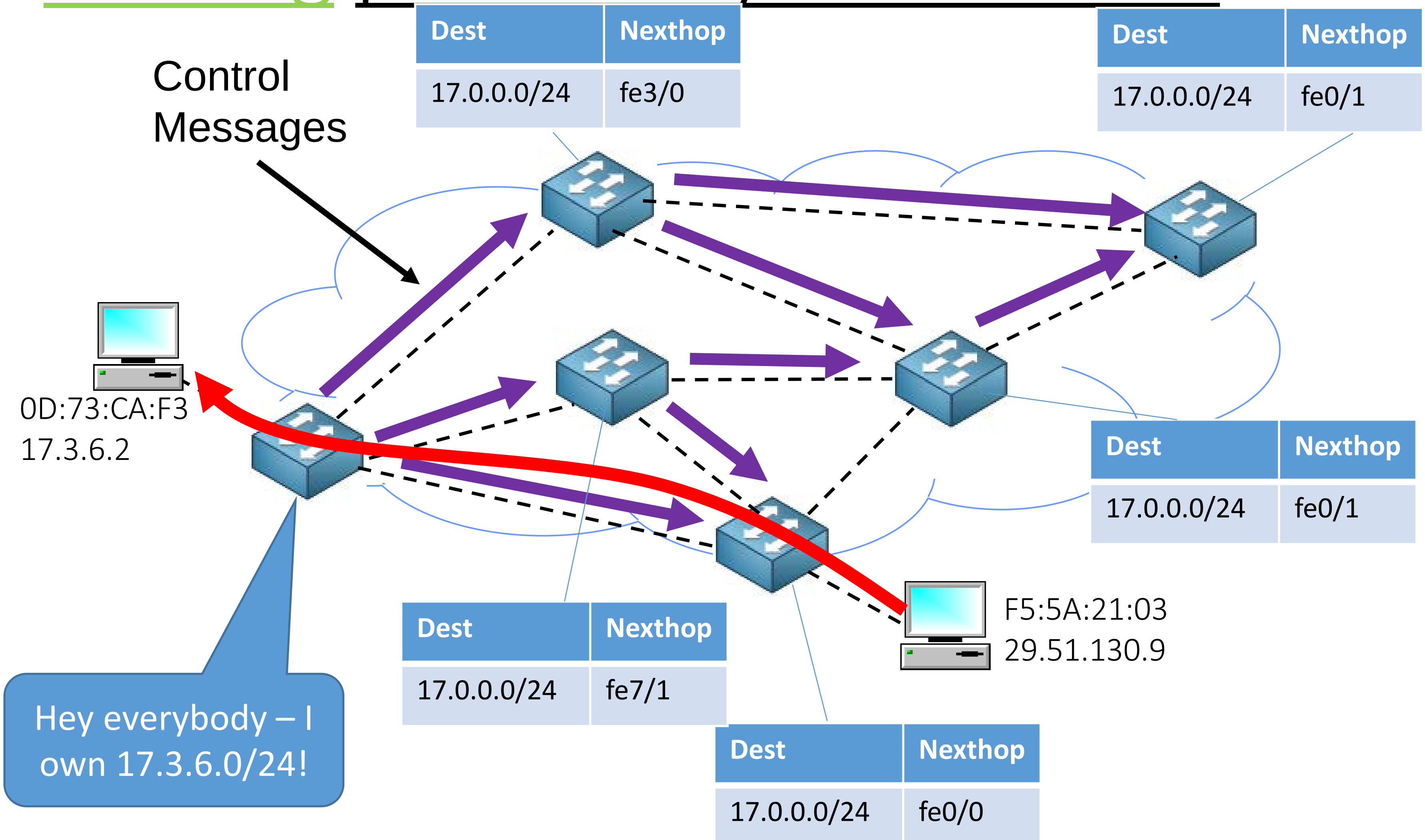
MAC address

Automatically discover host network configuration

Which IP addresses are on my subnet

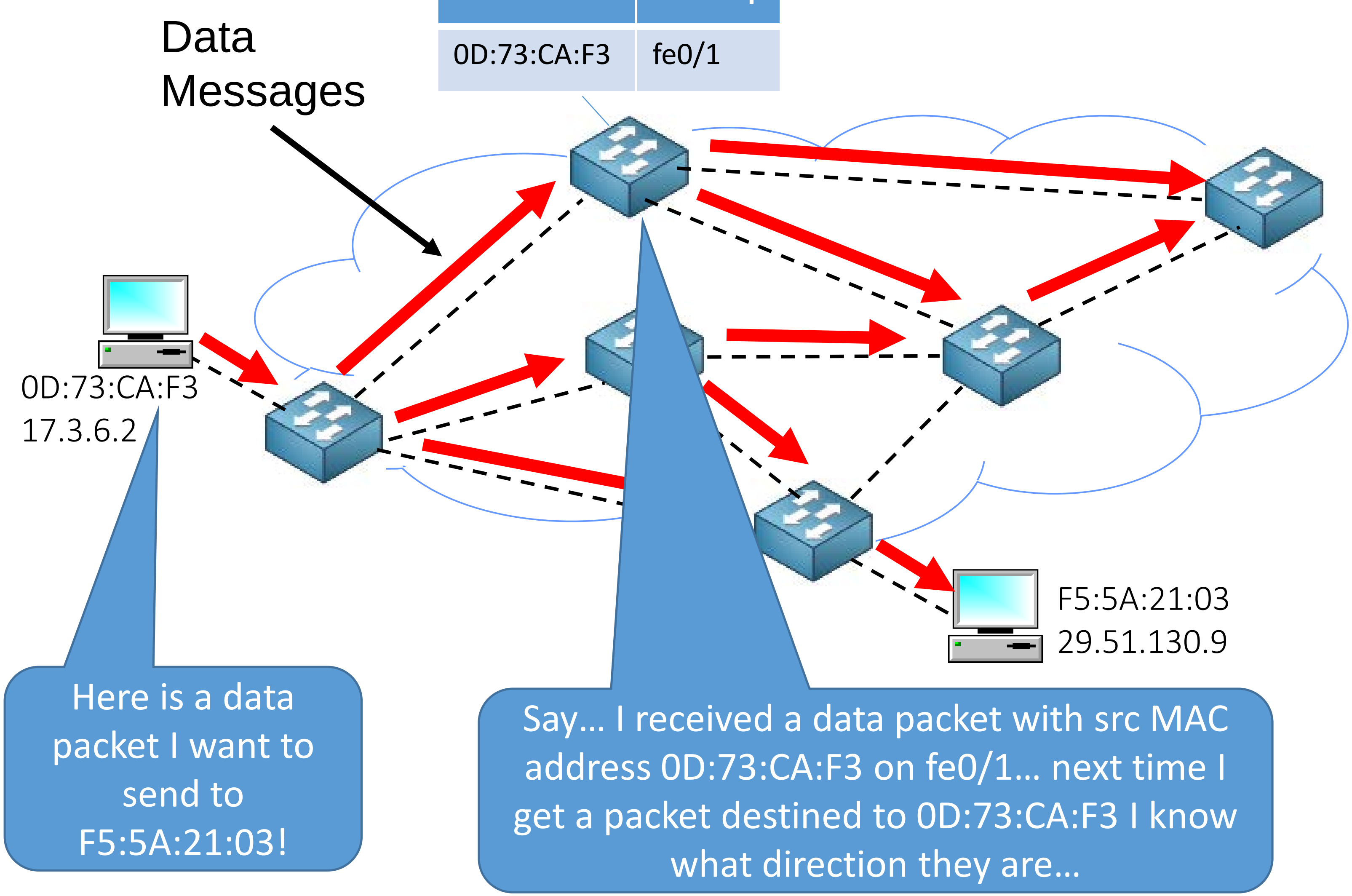
Where to forward packets if destination is off my subnet

L2 Switching vs L3 Routing: Routing proactively builds state

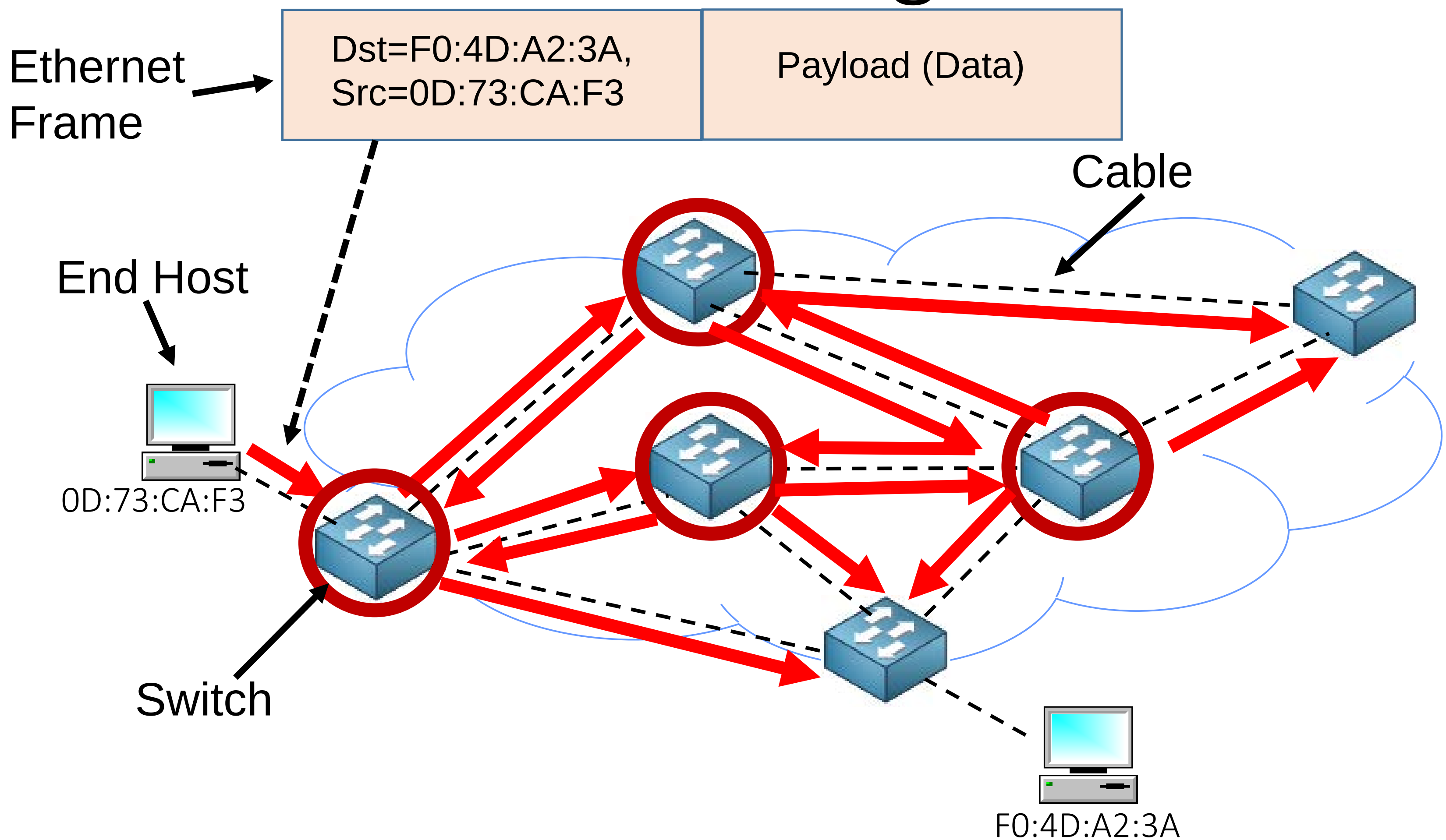


L2 Switching vs L3 Routing: Switching relies on broadcast

Dest	Nexthop
0D:73:CA:F3	fe0/1

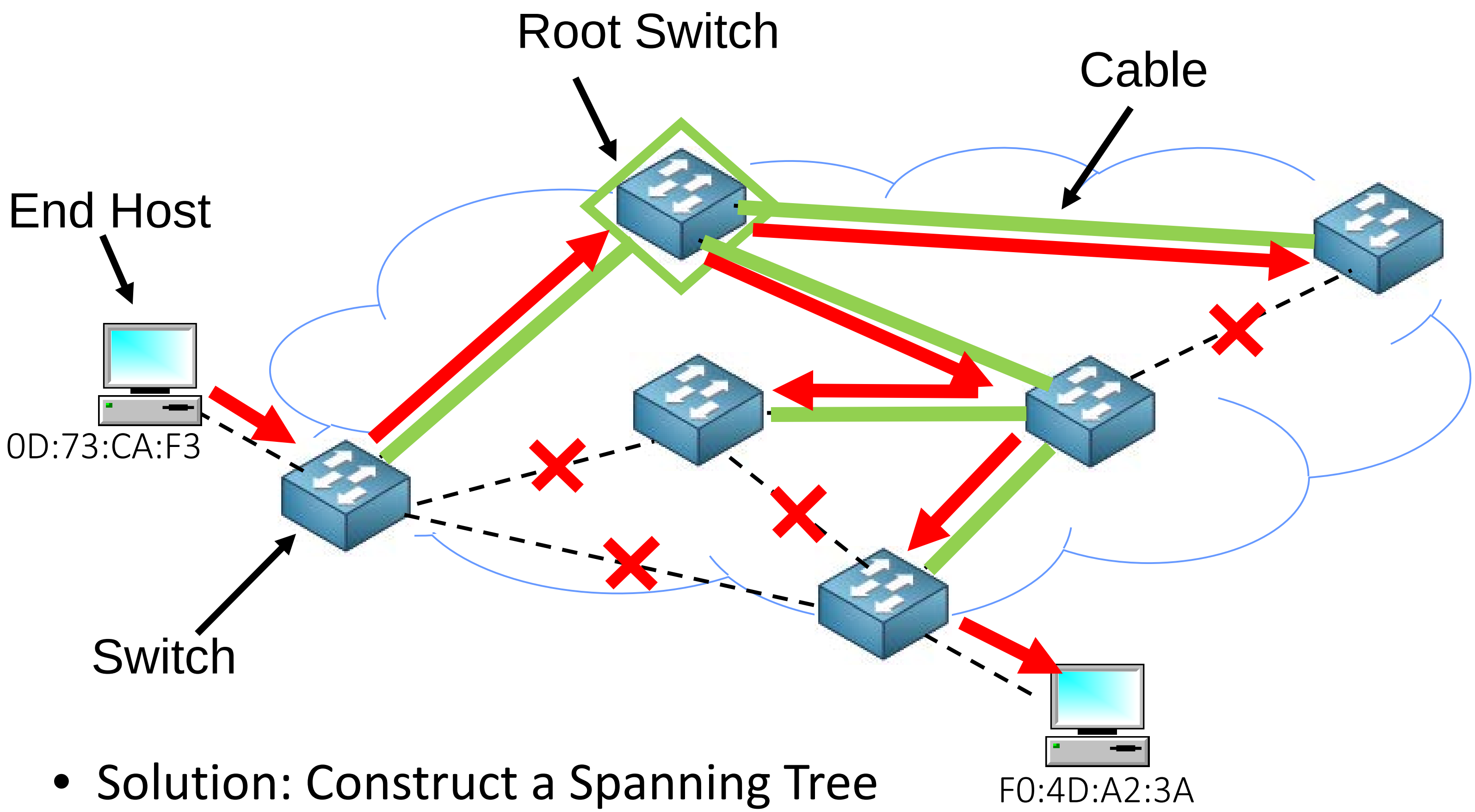


Ethernet Forwarding



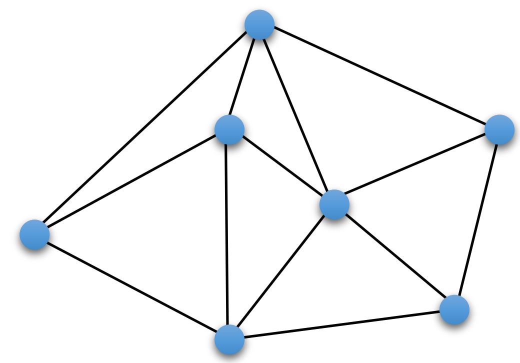
- Problem: Broadcast Storms
- How to flood with stateless switches?

Ethernet Forwarding



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

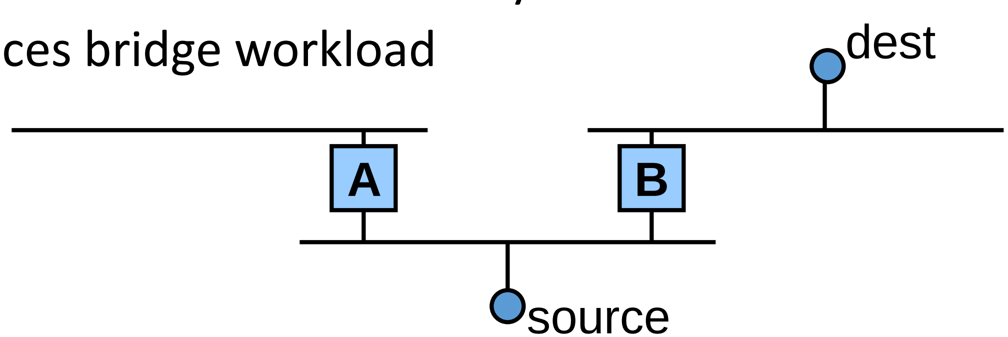
Avoiding Flooding



- 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

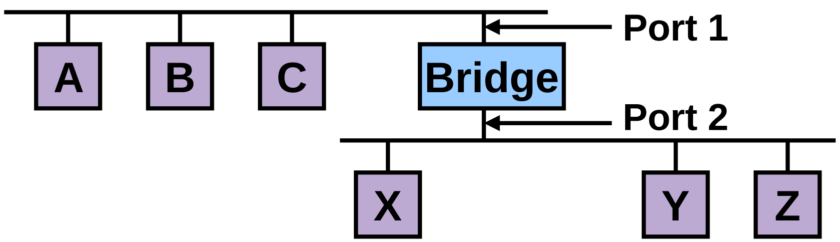
Avoiding Flooding

- 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

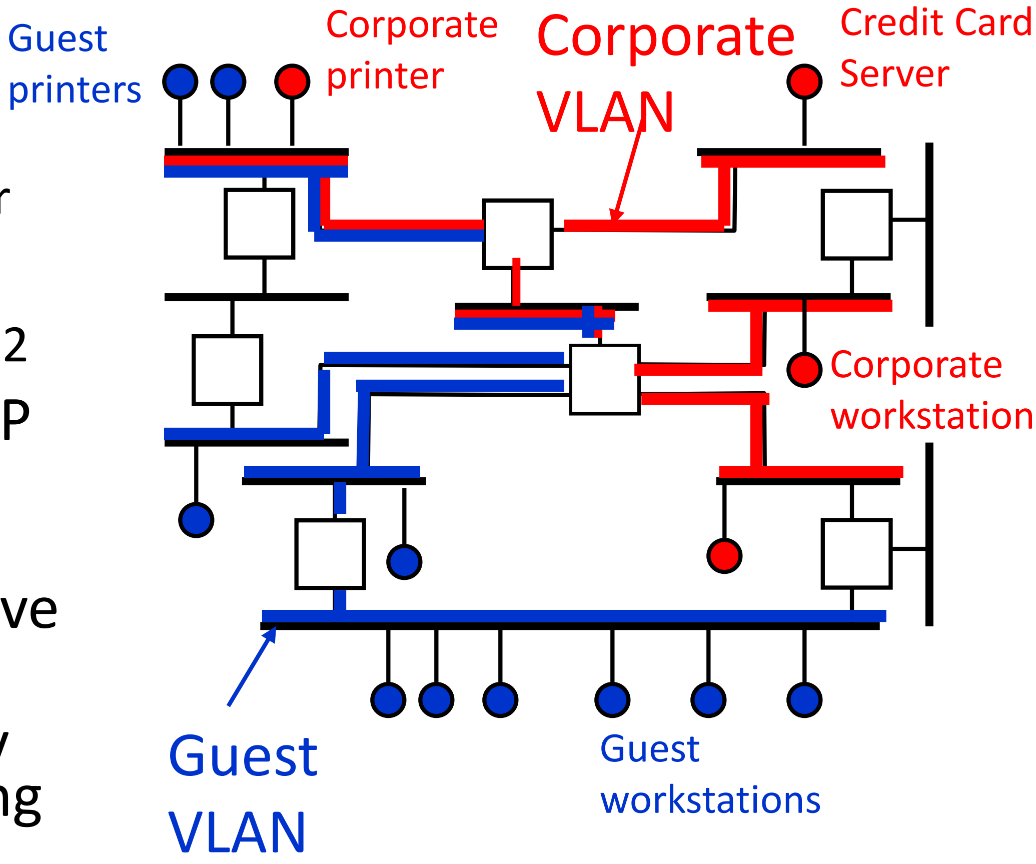
- Bridge learns table entries based on source address
  - When receive frame from A on port 1
    add A to list of hosts on port 1
  - Time out entries to allow movement of hosts
- Table is an “optimization”, meaning it helps performance but is not mandatory
- Always forward broadcast frames



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

Scaling Ethernet with VLANs

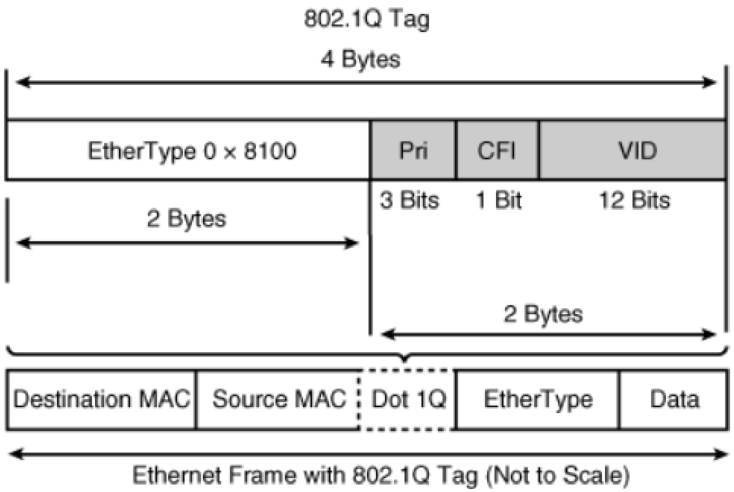
- Divide up hosts into logical groups called **VLANs**
  - Like virtual machines, but for LANs (creates "virtual networks")
  - VLANs isolate traffic at layer 2
- Each VLAN corresponds to IP subnet, single broadcast domain
- Ethernet packet headers have VLAN tag
- Bridges forward packet only on subnets on corresponding VLAN

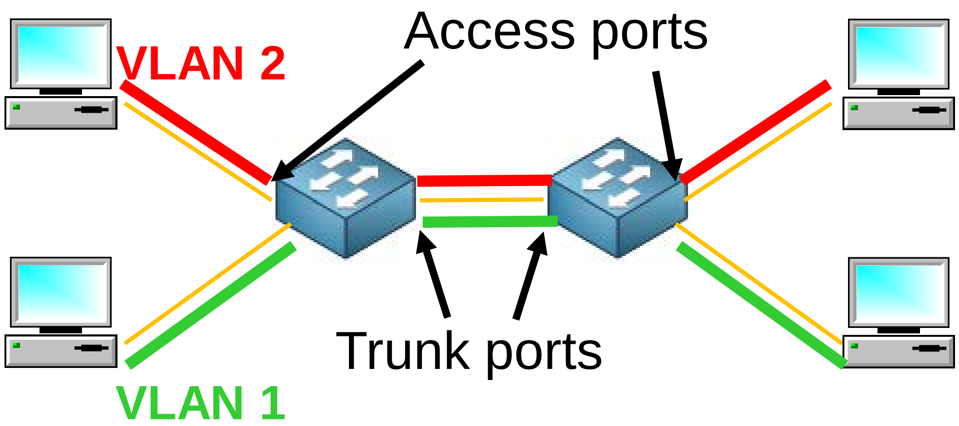


Virtual LANs

- Downsides of VLANs
  - Are (usually) 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

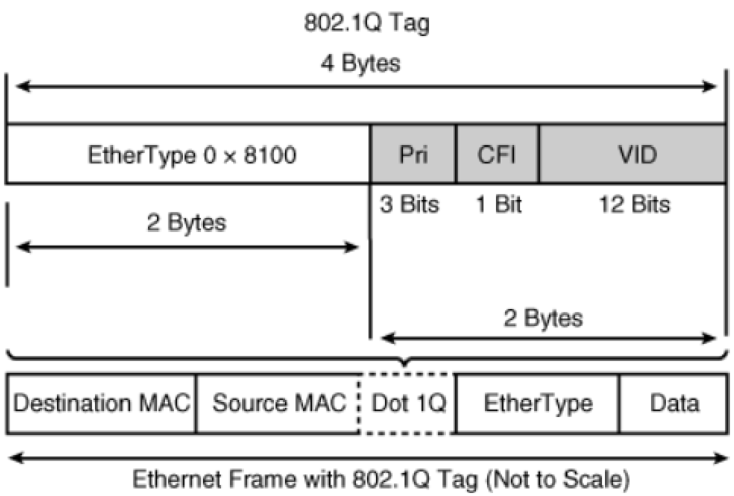
How VLANs are implemented

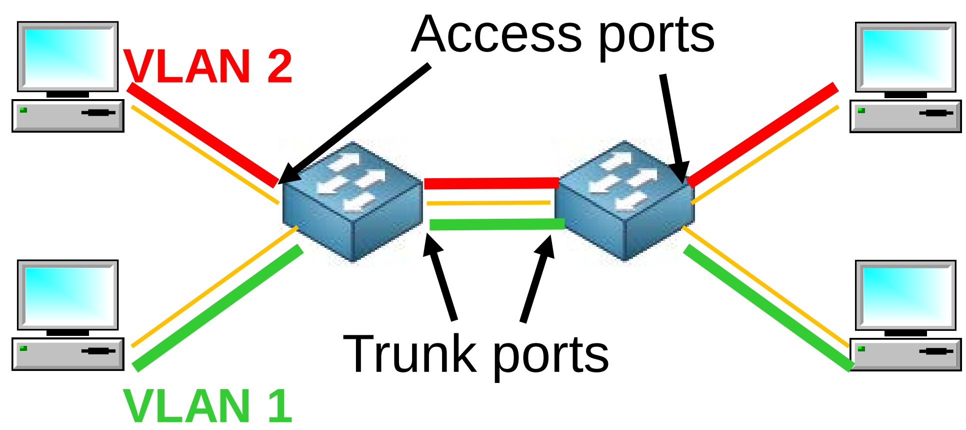




- Packets are annotated with 12-bit VLAN tags
  - Up to 4096 VLANs can be encapsulated within a single VLAN ID
- LAN switches can configure ports as access ports or trunk ports
  - Access ports append tags on packets
  - VLAN membership almost always statically encoded in access switch's configuration file
  - Trunk ports can multiplex several VLANs

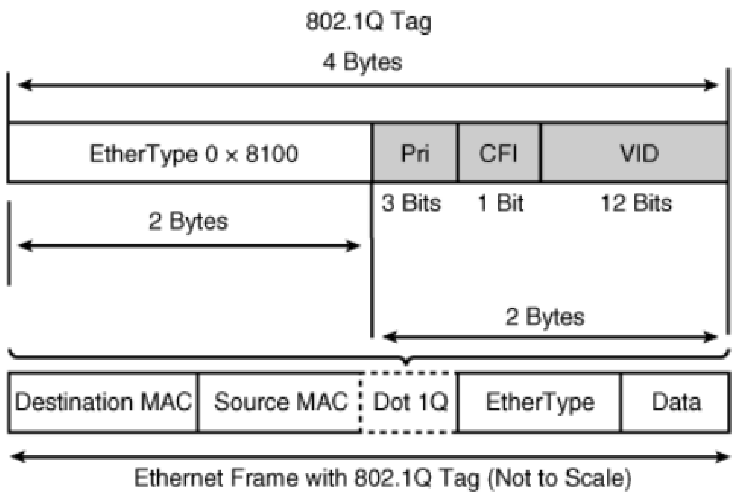
How VLANs are implemented

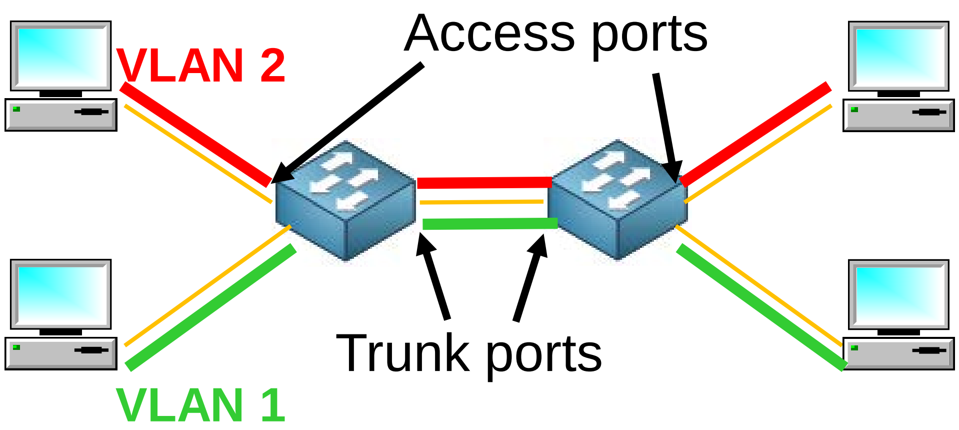




- 802.1Q (VLAN spec) defines a few other fields too
  - Ethertype of 0x8100 instructs switch to decode next 2 bytes as VLAN header
  - 3 bits of priority (like IP ToS)
  - 1 bit for compatibility with token ring
- What if 4096 VLANs isn't enough?
  - QinQ (802.1ad) – can encapsulate VLANs within VLANs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

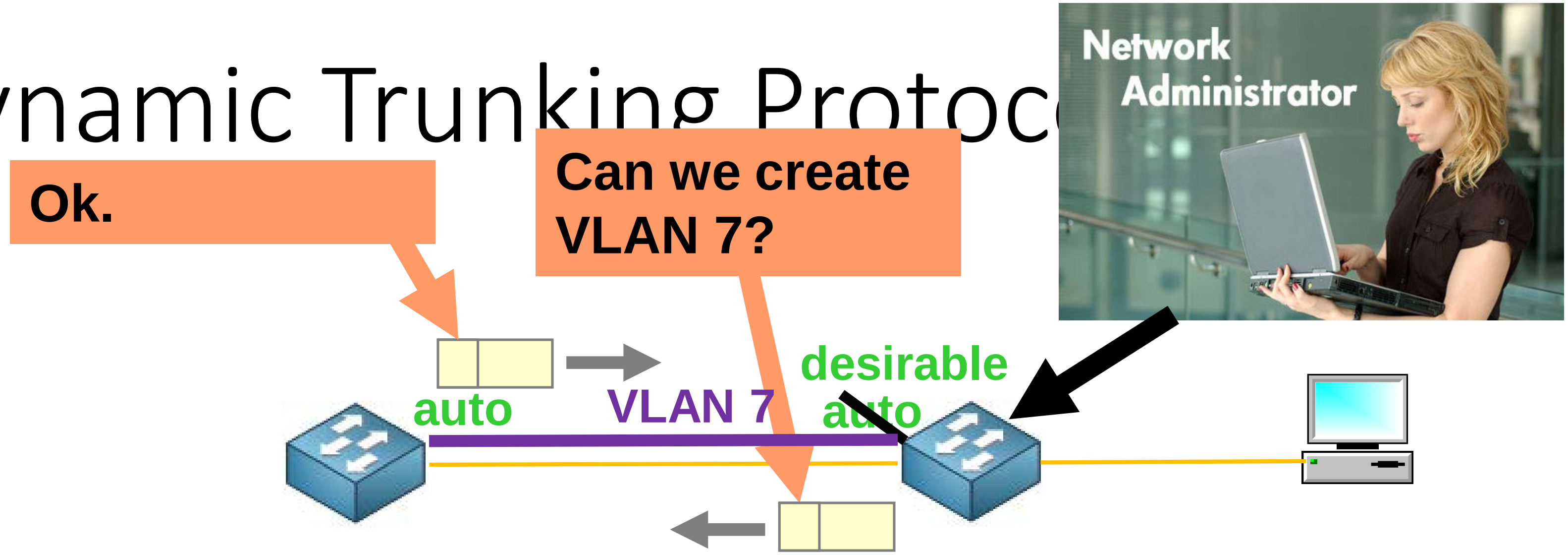
How VLANs are implemented





- 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

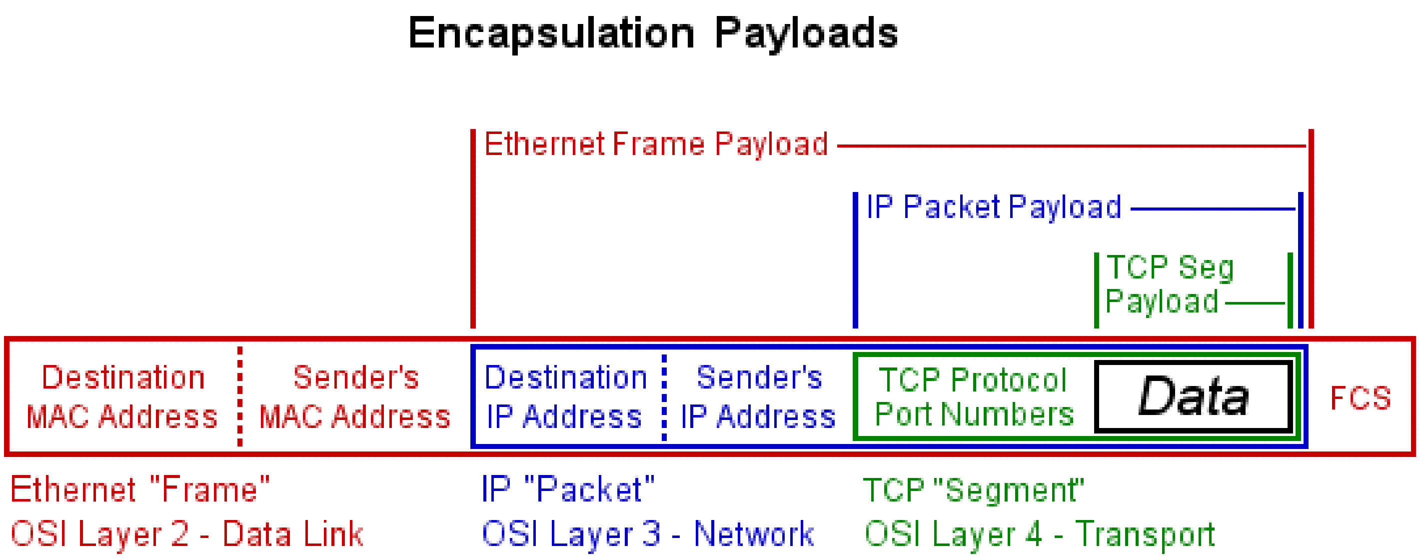
Dynamic Trunking Protoc



- 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)

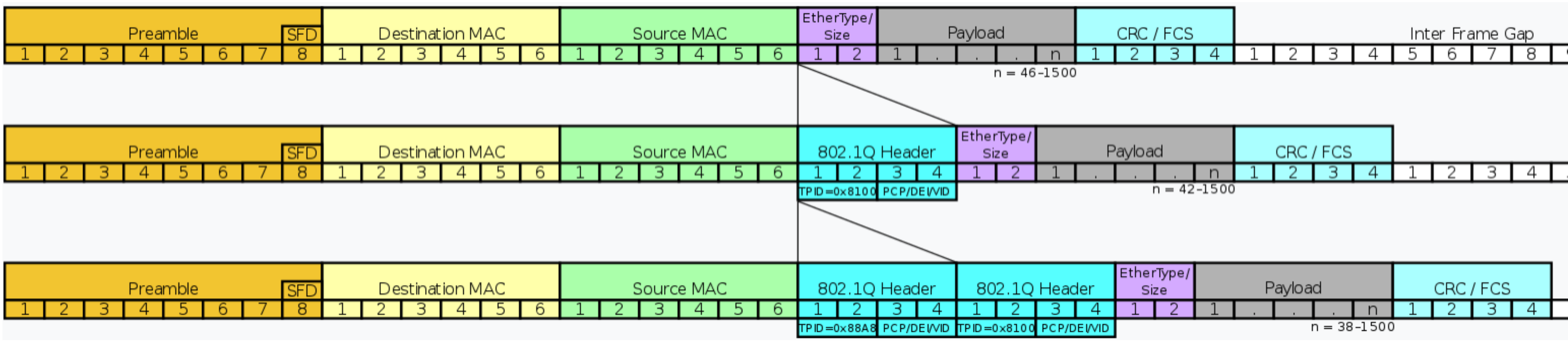
Traffic

Encapsulation



- Each layer of protocol stack encapsulates data passed to it.
- Each forwarding layer inspects data only at that encapsulation layer
  - Switching only looks at Ethernet header, Routing only looks at IP header, etc.
  - Terminology: “Layer-3 switch”, “Layer-4 load balancer”, “Layer-7 load balancer”

Ethernet Header



- Preamble – 56-bit pattern used to sync clocks
- Protocol ID: set to 0x8100 to identify frame as a 802.1Q-tagged frame
- Priority (PCP): 1 (background), 0 (best-effort, default), 2 (excellent effort), 3 (critical application), 4 (video), 5 (voice), 6 (internetwork control), 7 (network control)
- DEI – indicates if frame can be dropped during congestion
- VID – VLAN identifier. 12 bits allows for 4096 VLANs.
  - 802.1ad allows double tagging; 24 bits for VLANs

IPv4 Header

Offsets	Octet	0								1								2								3							
Octet	**Bit**	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	**0**	Version				IHL				DSCP						ECN		Total Length															
4	**32**	Identification																Flags			Fragment Offset												
8	**64**	Time To Live								Protocol								Header Checksum															
12	**96**	Source IP Address																															
16	**128**	Destination IP Address																															
20	**160**	Options (if IHL > 5)																															
24	**192**																																
28	**224**																																
32	**256**																																

- Version: 4; IHL: specifies size of header (words)
- DSCP: Diffserv marking (priority/QoS)
- Identification/Frag offset: fragment counter/offset. Flags: don’t fragment/more fragments/fragment ok
- TTL: hop counter; Protocol: TCP, ICMP, UDP, etc.
- Options: source routing, record route, etc. typically filtered.

TCP header

<table>
<tr><th>Offsets</th><th>Octet</th><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>Octet</th><th>Bit</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td>0</td><td>0</td><td colspan="16">Source port</td><td colspan="16">Destination port</td></tr>
<tr><td>4</td><td>32</td><td colspan="32">Sequence number</td></tr>
<tr><td>8</td><td>64</td><td colspan="32">Acknowledgment number (if ACK set)</td></tr>
<tr><td>12</td><td>96</td><td colspan="4">Data offset</td><td colspan="3">Reserved
0 0 0</td><td>NS</td><td>CWR</td><td>ECE</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="16">Window Size</td></tr>
<tr><td>16</td><td>128</td><td colspan="16">Checksum</td><td colspan="16">Urgent pointer (if URG set)</td></tr>
<tr><td>20
...</td><td>160
...</td><td colspan="32">Options (if data offset > 5. Padded at the end with "0" bytes if necessary.)
...</td></tr>
</table>

- Ports: assigned to application. Local may be ephemeral.
- Seq num: used for reassembly; Initial seq num.
- Ack num: next seqno the sender is expecting
- Data offset: size of TCP header
- Flags: SYN, ACK, FIN are main used ones
- Window size: size of sender’s receive window
- Options: Selective Acks Permitted, Max Segment Size announcement, etc

Addressing

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 (often) 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!

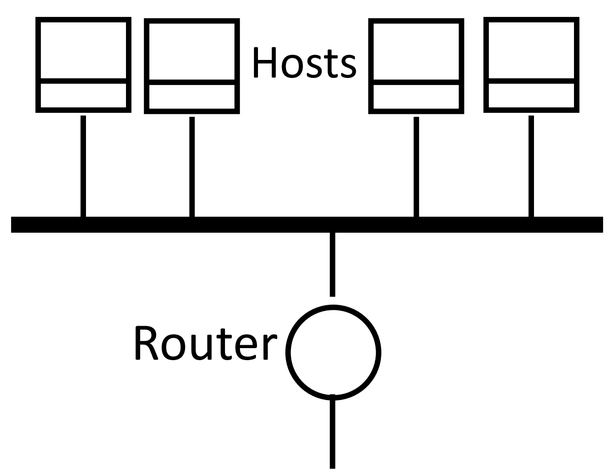
Discovery

- 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

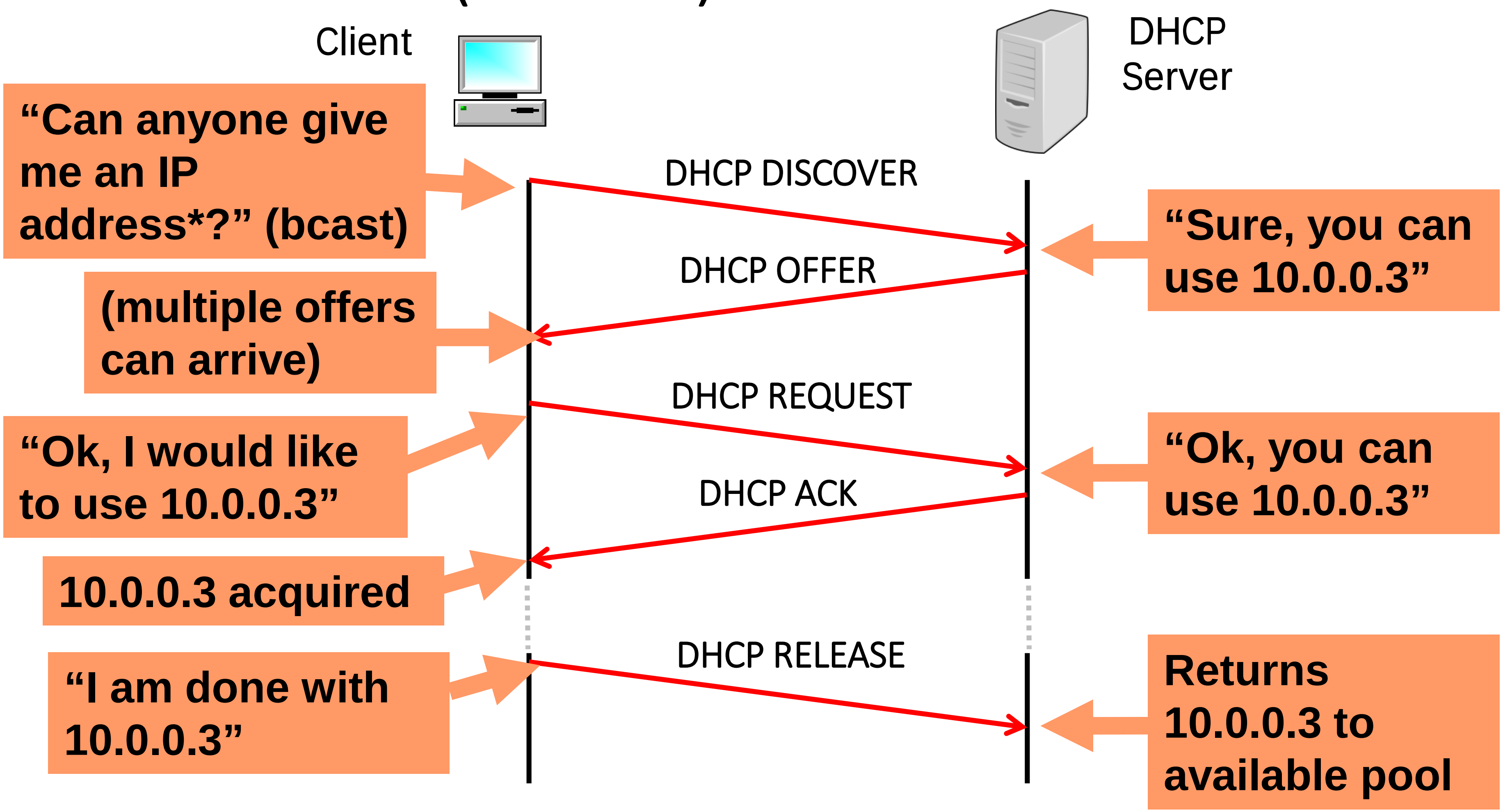
ARP and DHCP

- 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

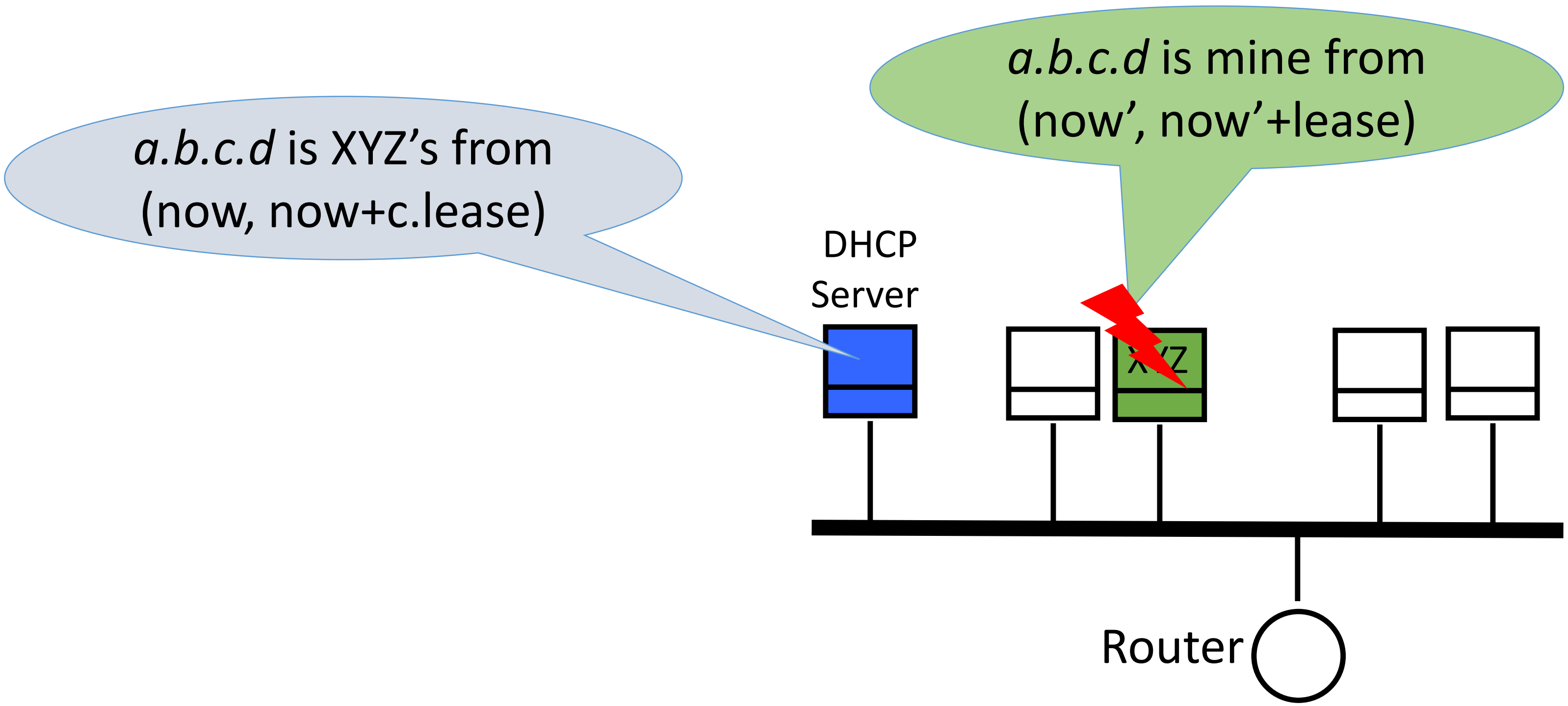
1. One or more local DHCP servers maintain required information
2. Client broadcasts a DHCP discovery message
3. One or more DHCP servers responds with a DHCP “offer” message
4. Client broadcasts a DHCP request message
5. Selected DHCP server responds with an ACK

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

DHCP uses "soft state"

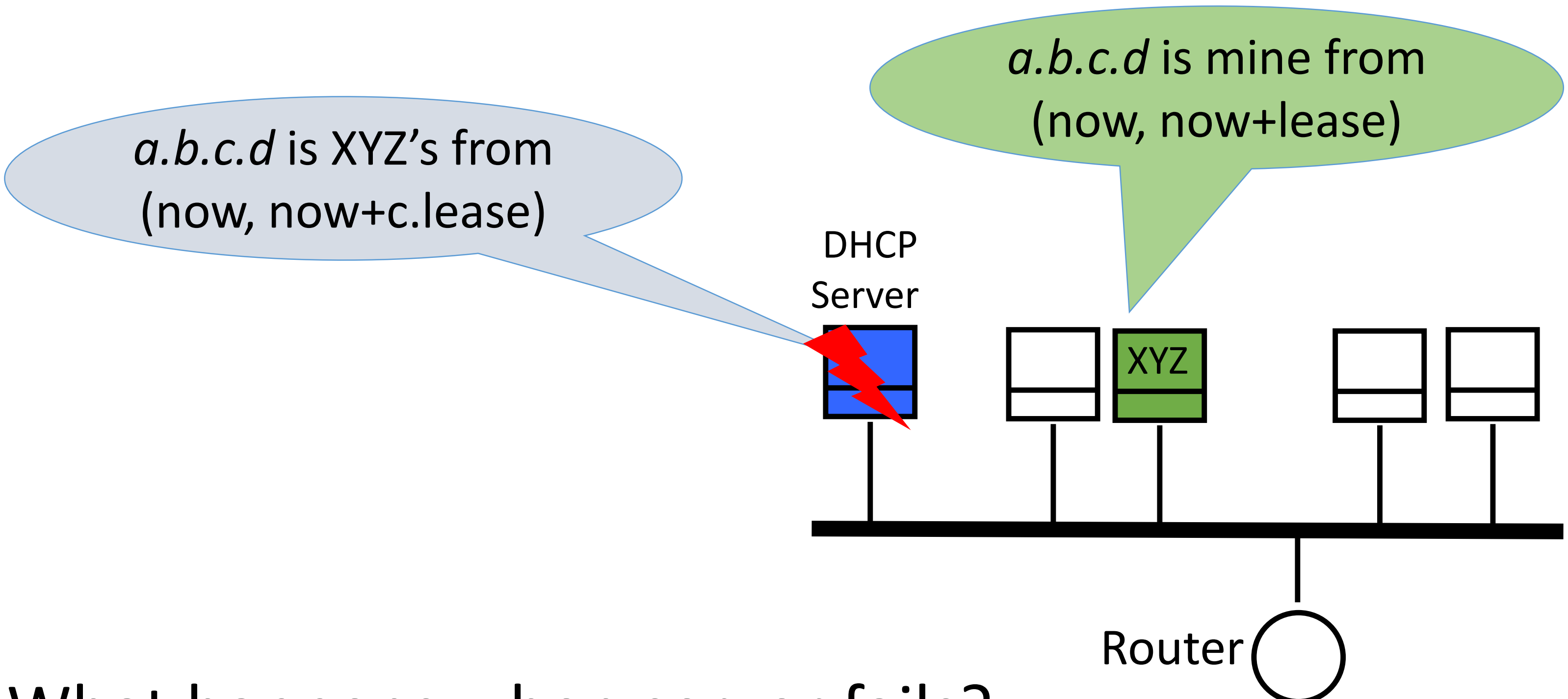
- Soft state: if not refreshed, state is forgotten
  - hard state: allocation is deliberately returned/withdrawn
  - used to track address allocation in DHCP
- Implementation
  - address allocations are associated with a lease period
  - server: sets a timer associated with the record of allocation
  - 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



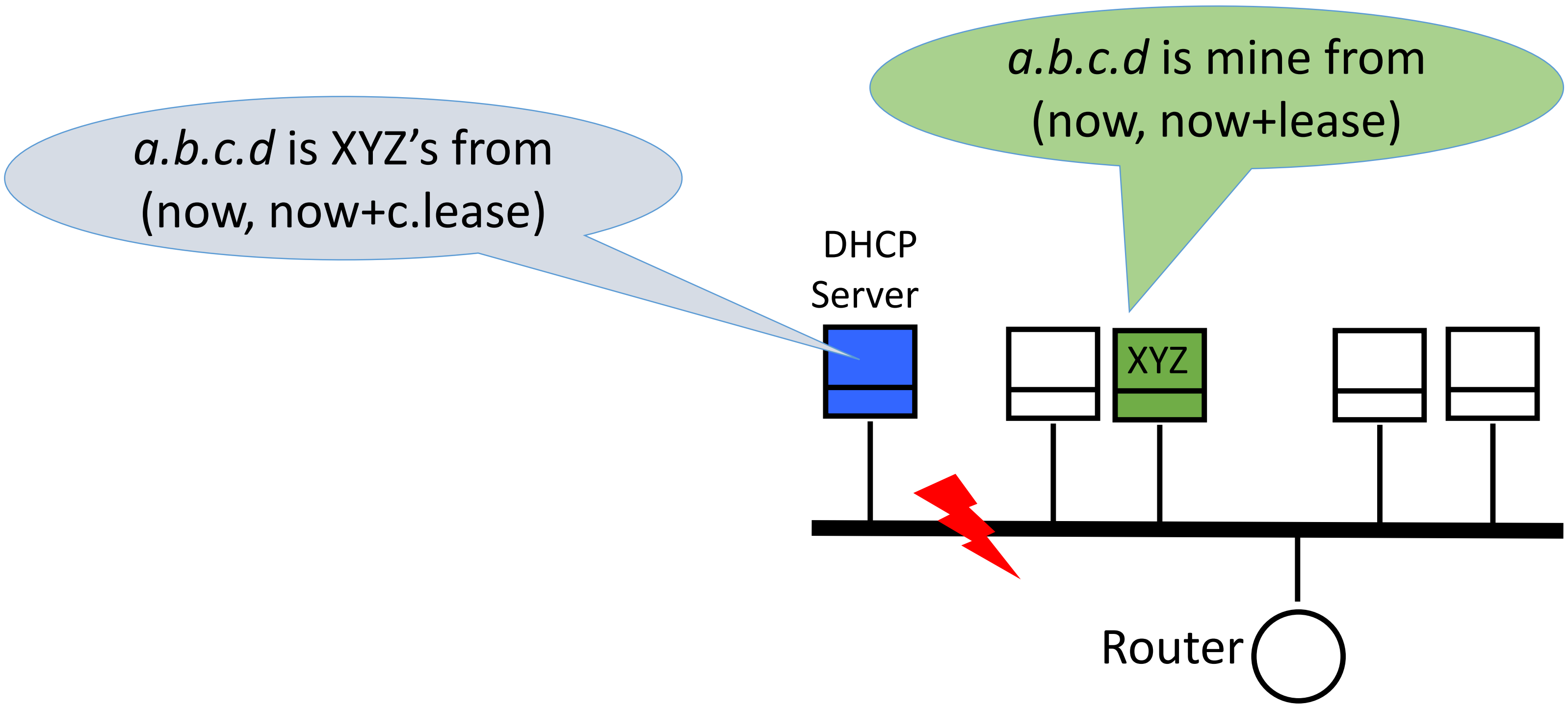
- What happens when host XYZ fails?
  - refreshes from XYZ stop
  - server reclaims *a.b.c.d* after O(lease period)

Soft state under failure



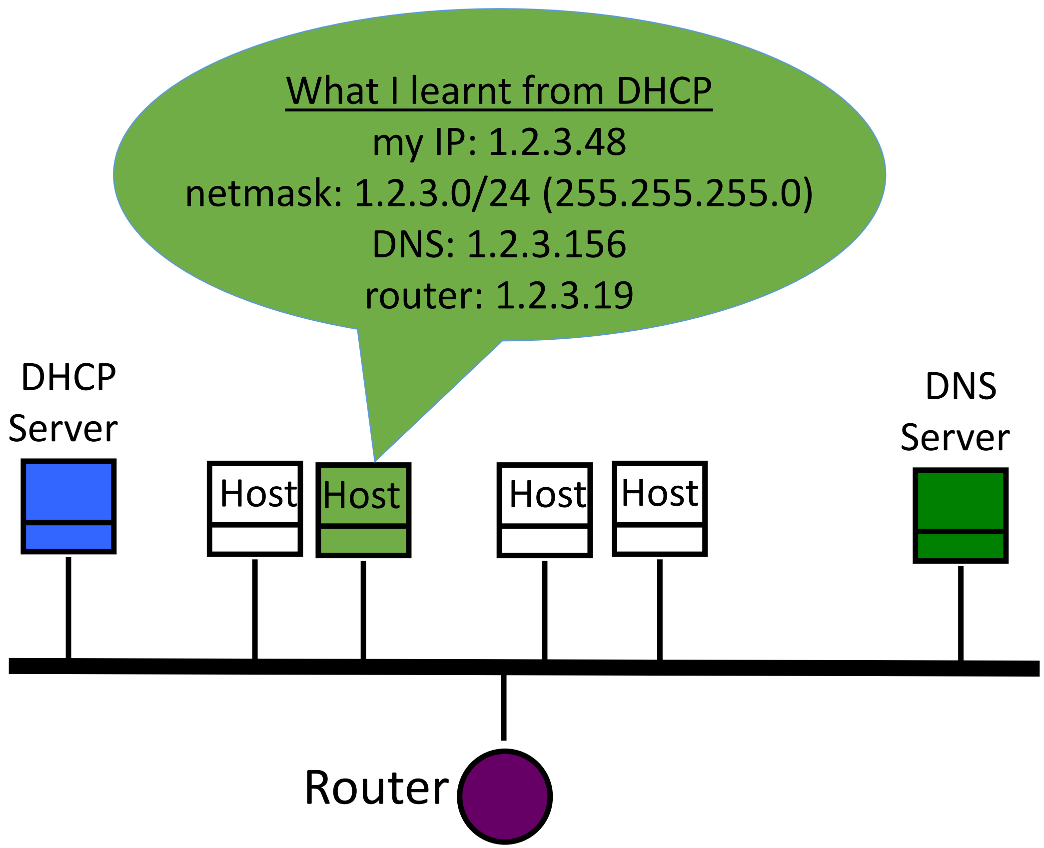
- What happens when server fails?
  - ACKs from server stop
  - XYZ releases address after O(lease period); send new request
  - A new DHCP server can come up from a `cold start' and we're back on track in ~lease time

Soft state under failure

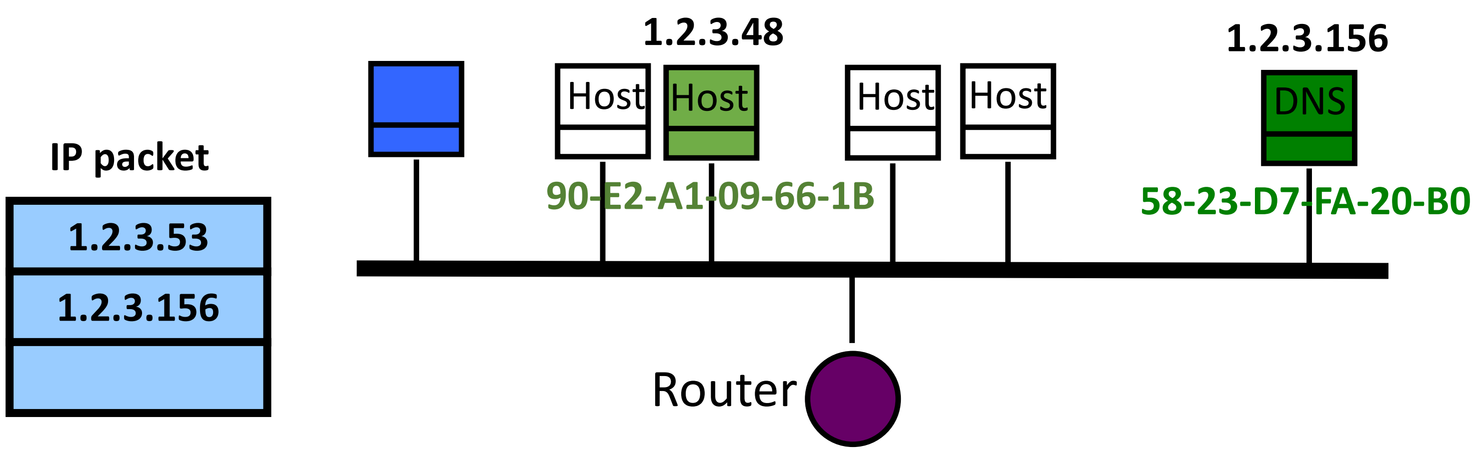


- What happens if the network fails?
  - refreshes and ACKs don't get through
  - XYZ release address; DHCP server reclaims it

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

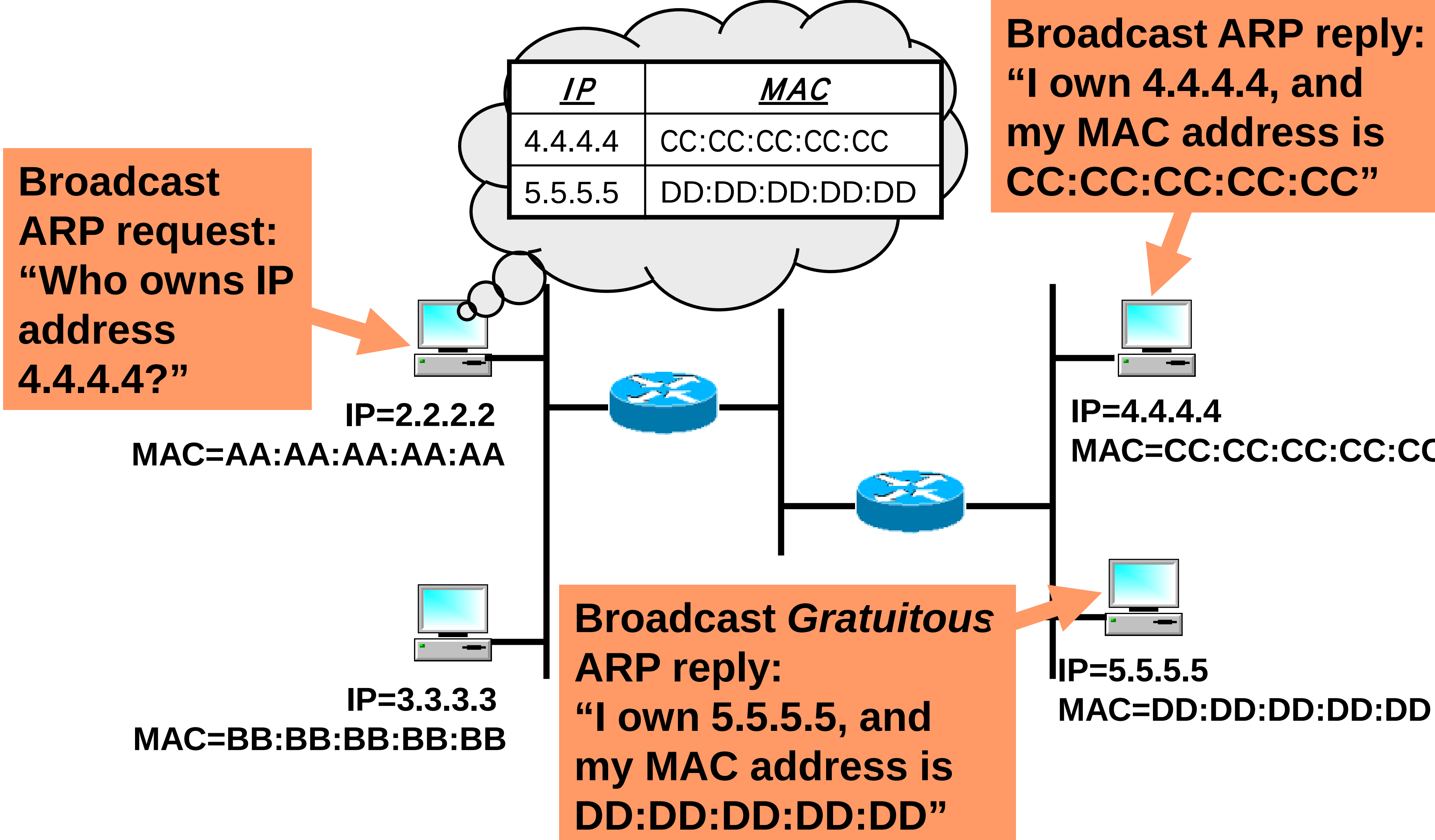
```
C:\Users\Matthew Caesar>arp -a

Interface: 192.168.1.84 --- 0x8
  Internet Address      Physical Address      Type
  192.168.1.76          84-d6-d0-a1-05-84     dynamic
  192.168.1.87          00-80-92-cb-aa-0c     dynamic
  192.168.1.254         f8-2c-18-94-96-7d     dynamic
  192.168.1.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

- 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)

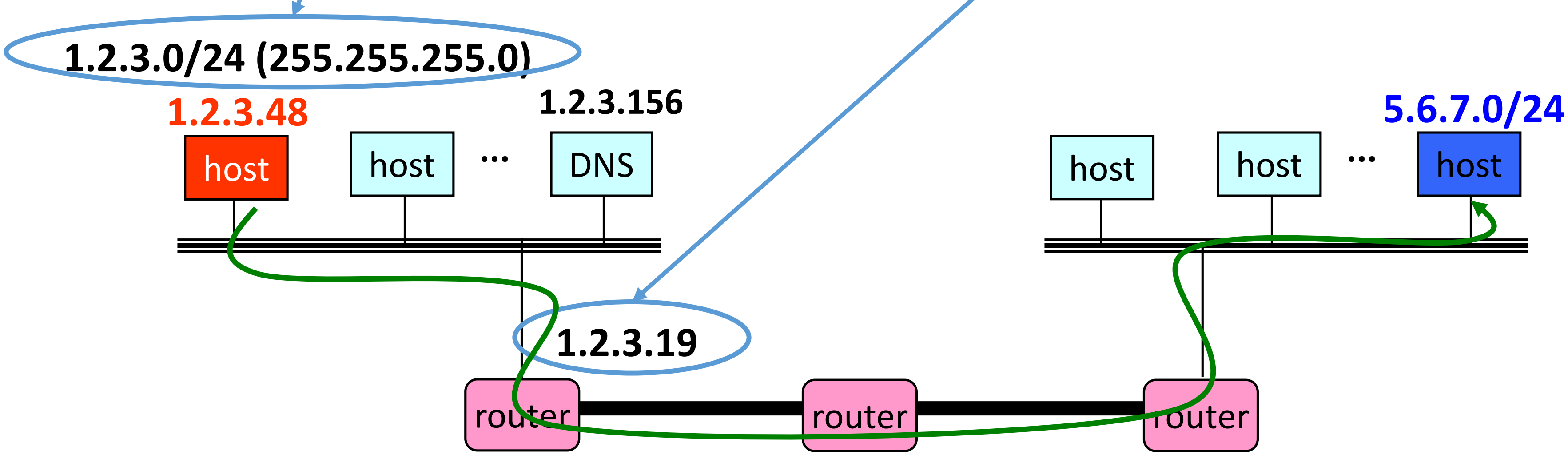
- Networked applications are programmed to deal with IP addresses
- But Ethernet forwards to MAC address
- How can OS know the MAC address corresponding to a given IP address?
- Solution: Address Resolution Protocol
    - Broadcasts ARP request for MAC address owning a given IP address



- 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 (entered manually or discovered via DHCP)
- How does the red host know about 1.2.3.19?
  - Also entered manually or DHCP (assigned as the "gateway")



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
- Solutions
  - Dynamic Arp Inspection (DAI) – disallow ARP requests from MAC address not manually whitelisted on port
  - Port Security – configure port to disallow data traffic from MAC address not manually whitelisted to port

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**

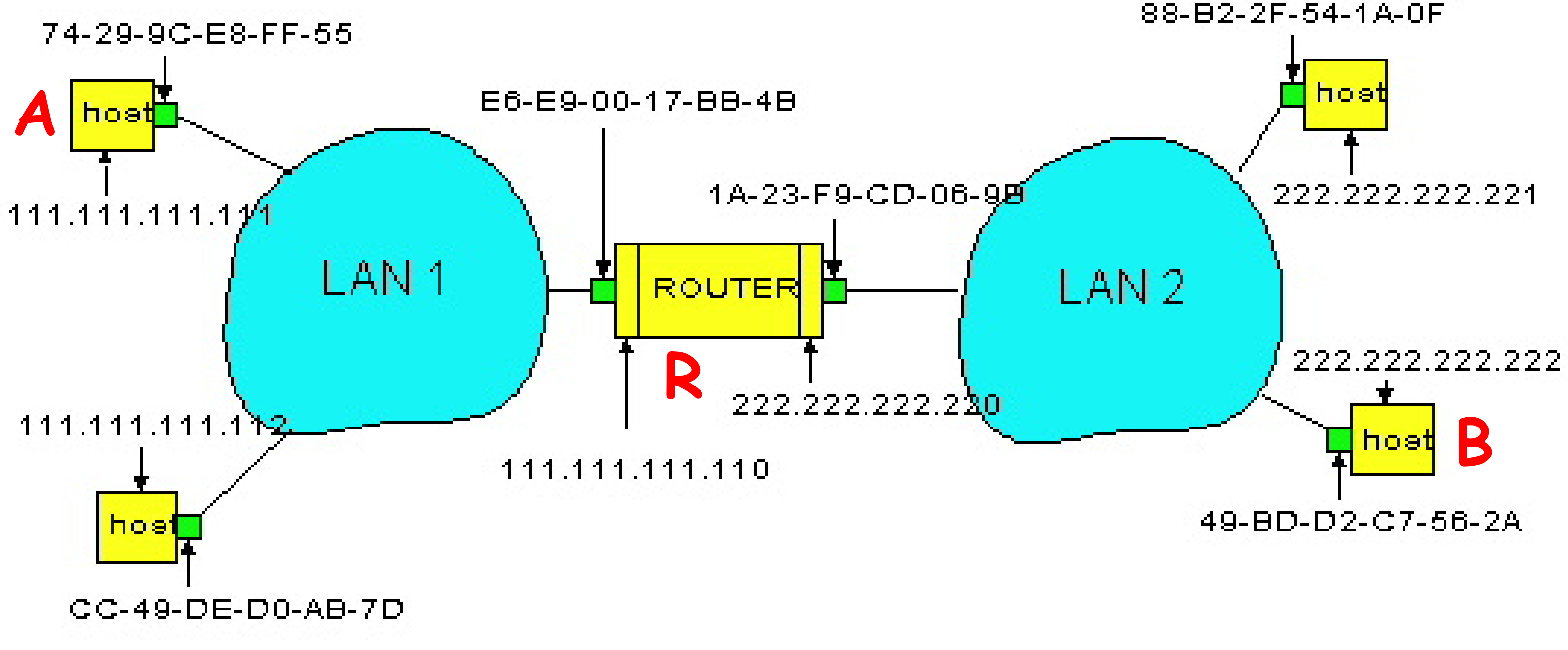
Sending a Packet

- On same subnet:
  - Use MAC address of destination.
  - ***ARP***

- On some other subnet:
  - Use MAC address of first-hop router.
  - ***DHCP + ARP***

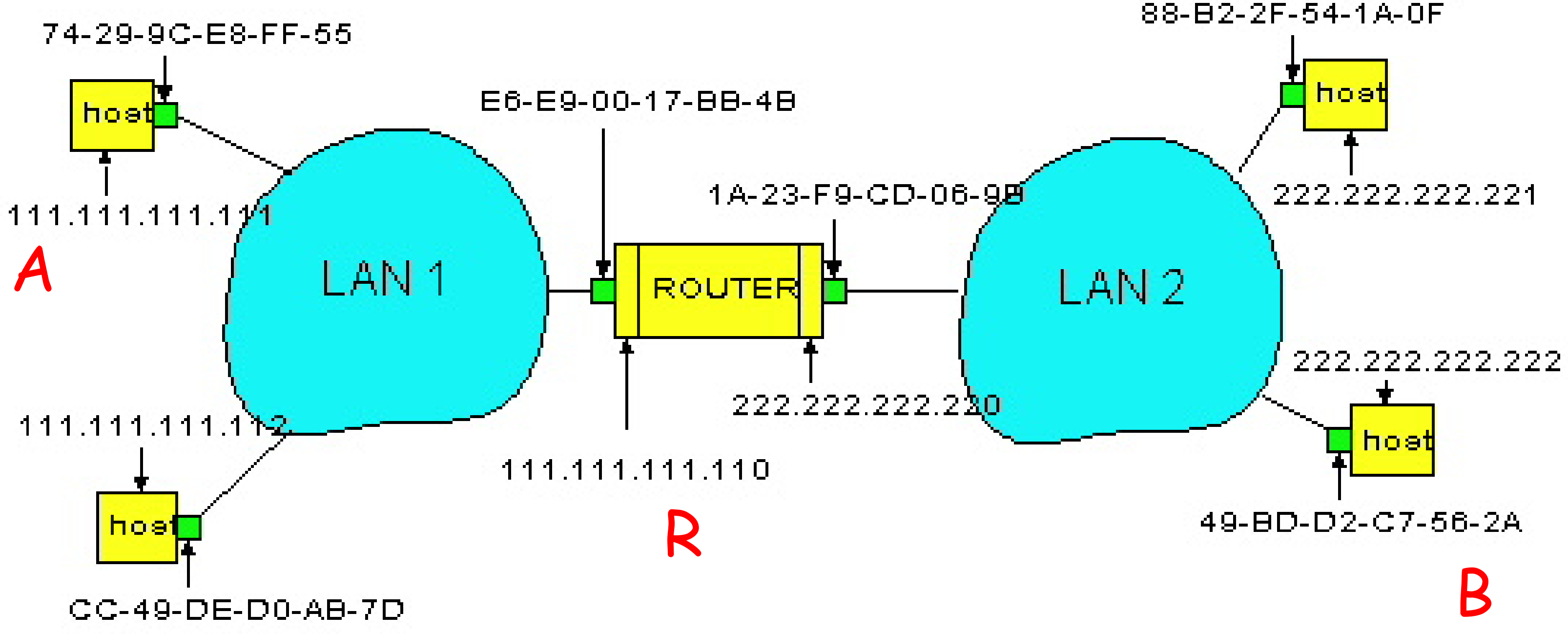
- And how can a host tell whether destination is on same or other subnet?
  - Use the netmask
  - ***DHCP***

Example: A Sending a Packet to B



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

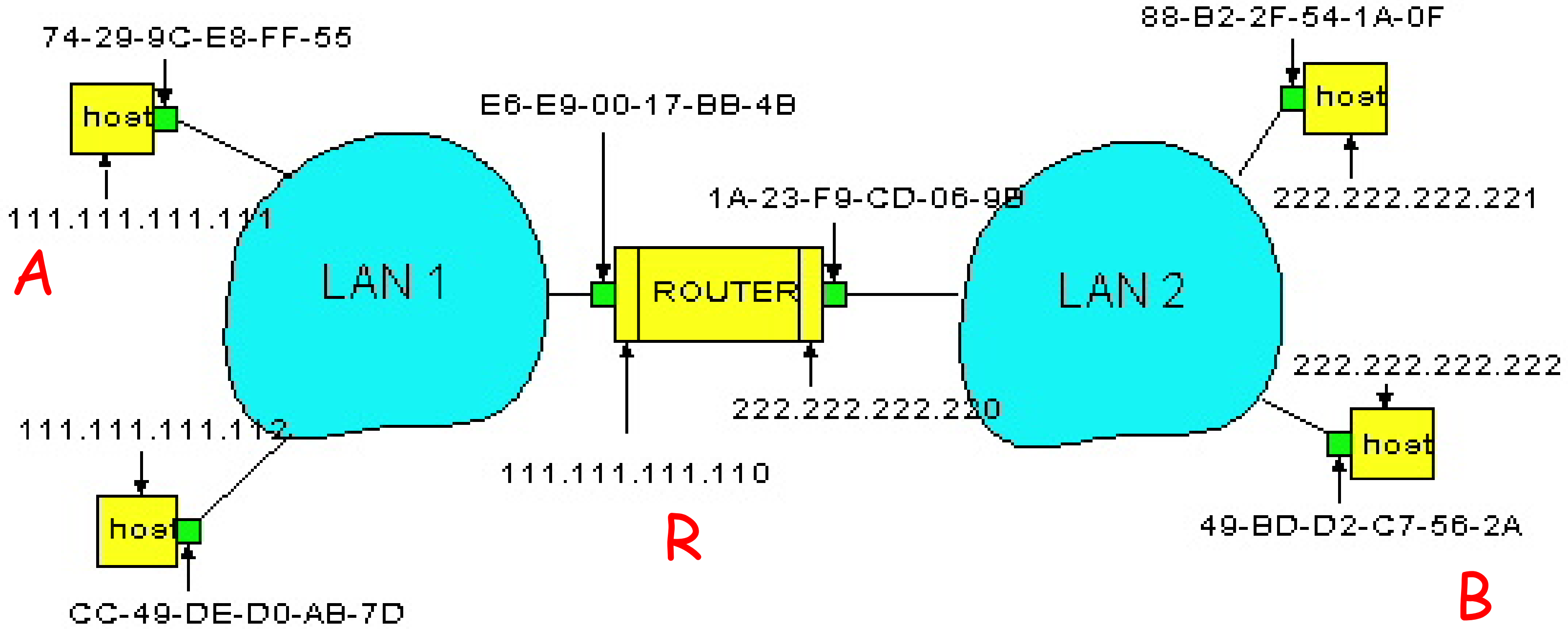
Example: A Sending a Packet to B



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

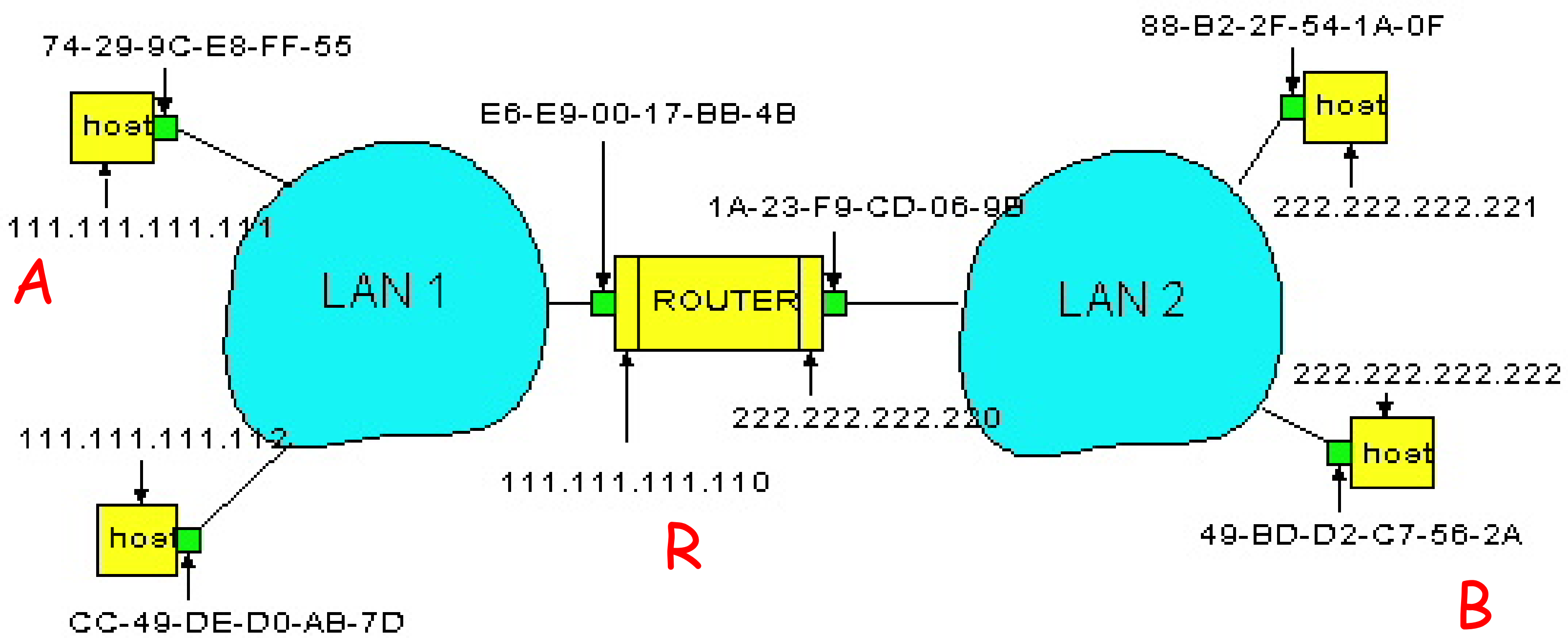
Host A Decides to Send Through R

- 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



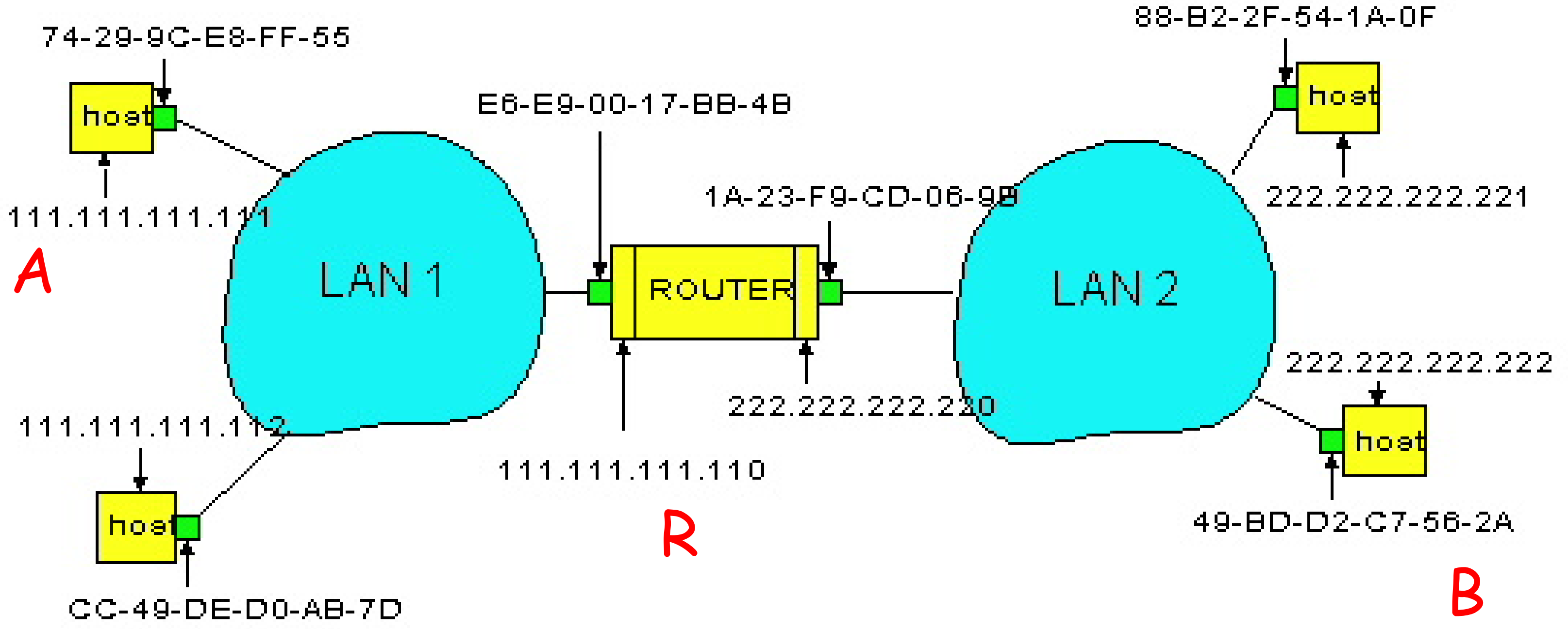
Host A Sends Packet Through R

- 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



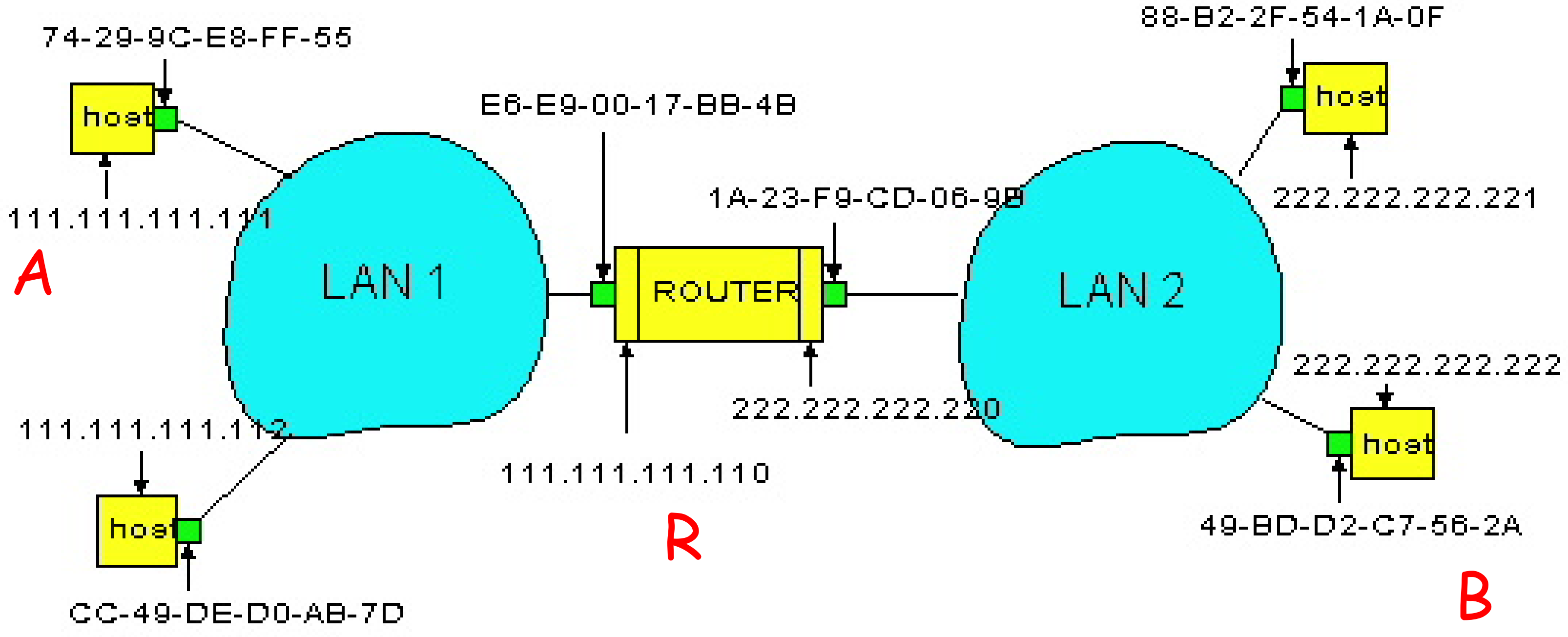
R Decides how to Forward Packet

- Router R's adapter receives the packet
  - R extracts the IP packet from the Ethernet frame
  - R sees the IP packet is destined to 222.222.222.222
- Router R consults its forwarding table
  - Packet matches 222.222.222.0/24 via other adapter (port)



R Sends Packet to B

- Router R 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?