Data-Link Layer III

Our goals:

- Understand principles behind link layer services:
  - Introduction, Services
  - Error detection, correction
  - Multiple access protocols
  - Sharing a broadcast channel: multiple access
  - Data-Link Layer Addressing
  - ARP/RARP
  - LAN: Ethernet
  - LAN: Switches
  - LAN: VLANS
  - Simple web request description
- Implementation of various link layer technologies

MAC Address

MAC addresses

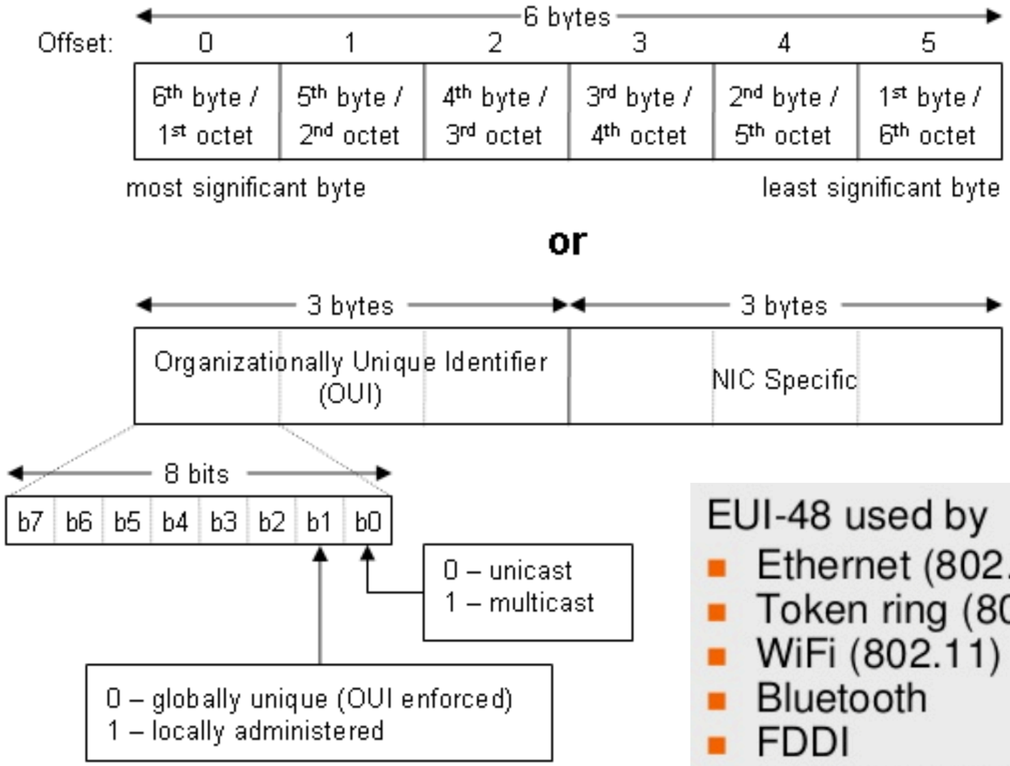
- 32-bit IP address:
  - *network-layer* address for interface
  - used for layer 3 (network layer) forwarding
- MAC (or LAN or physical or Ethernet) address:
  - function: *used 'locally" to get frame from one interface to another physically-connected interface (same network, in IP-addressing sense)*
  - 48 bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
  - e.g.: 1A-2F-BB-76-09-AD

hexadecimal (base 16) notation
(each "number" represents 4 bits)

MAC addresses = EUI

- EUI-64 – Modify EUI-48
- EUI-48 (MAC-48) 2^{48} = 281 474 976 710 656 - Extended Unique Identifier
- EUI-48 = OUI-24 + IAB-12 + NIC-12
- OUI – 24bit - Organizationally Unique Identifier from IEEE Committee
- IAB - Individual Address Block – 12bit - from IEEE Committee
- MAC Address Block and Venders Search (IEEE.org). https://regauth.standards.ieee.org/standards-ra-web/pub/view.html#registries
- 45-67-89-AB-CD-EF (canonical Windows)
- 45:67:89:AB:CD:EF (bit-reverse Linux)
- 4567.89AB.CDEF (CISCO)
- 45,67,89,AB,CD,EF
- 456789ABCDEF

MAC Structure



EUI-48 used by

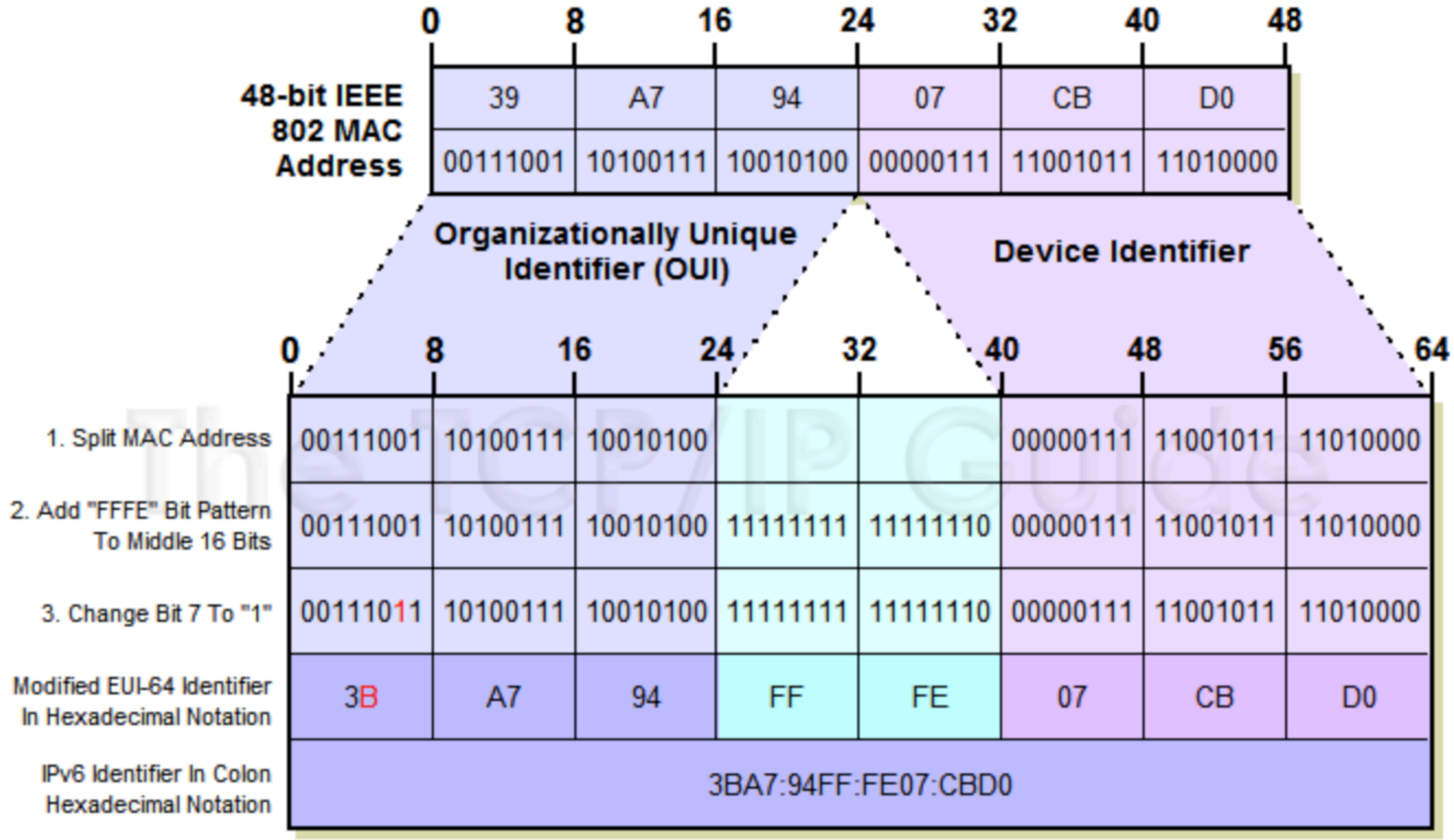
- Ethernet (802.3)
- Token ring (802.5)
- WiFi (802.11)
- Bluetooth
- FDDI
- SCSI/fiber-channel

IEEE defined a "next generation" 8-byte address called EUI-64

EUI-64 used for

- IEEE 1394 (firewire)
- 802.15.4 (personal area networks)
- IPv6 (LSBs of non-temporary unicast address)

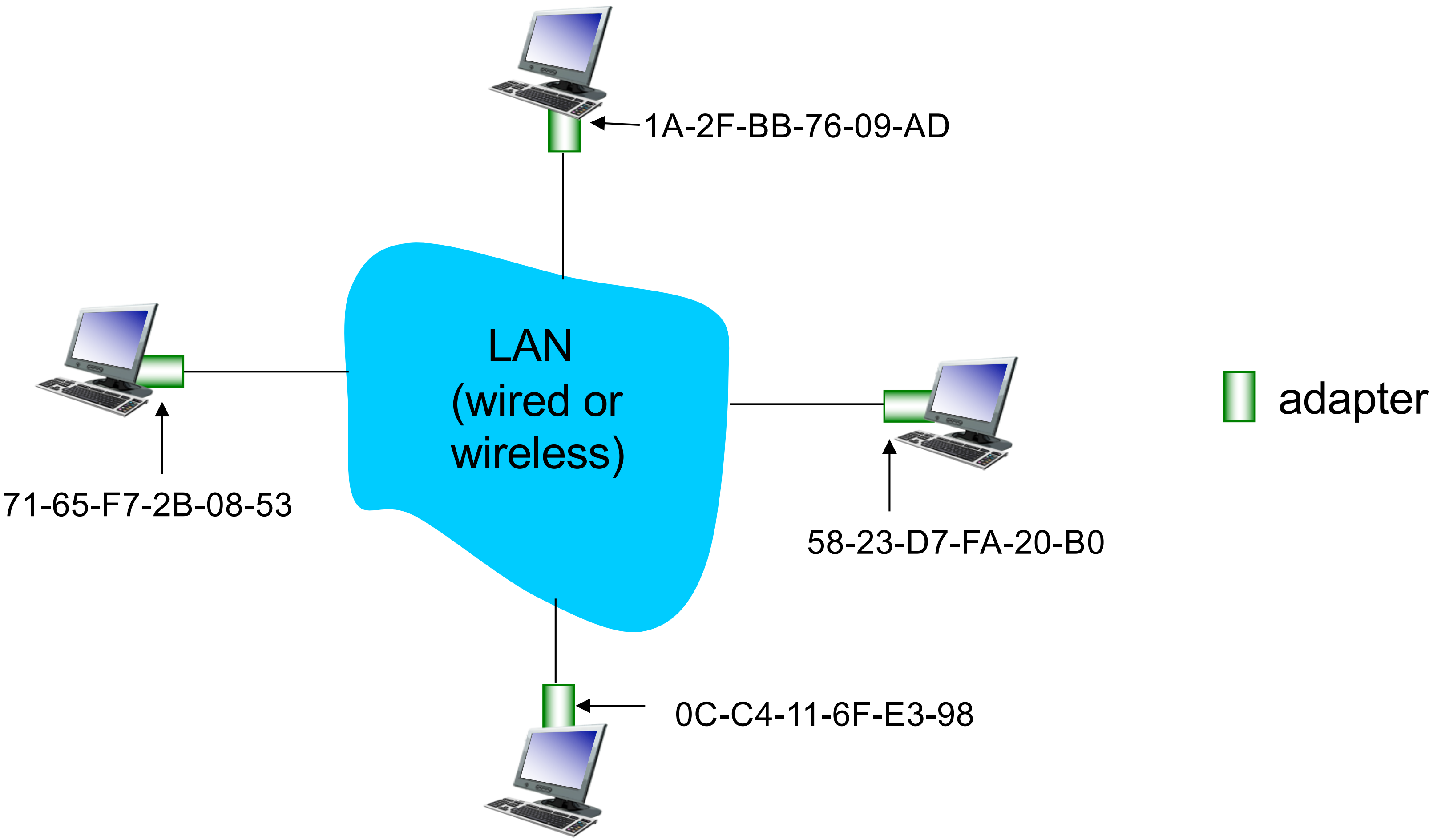
EUI-48 → EUI-64



64-Bit IPv6 Modified EUI-64 Interface Identifier

LAN addresses

each adapter on LAN has unique *LAN* address

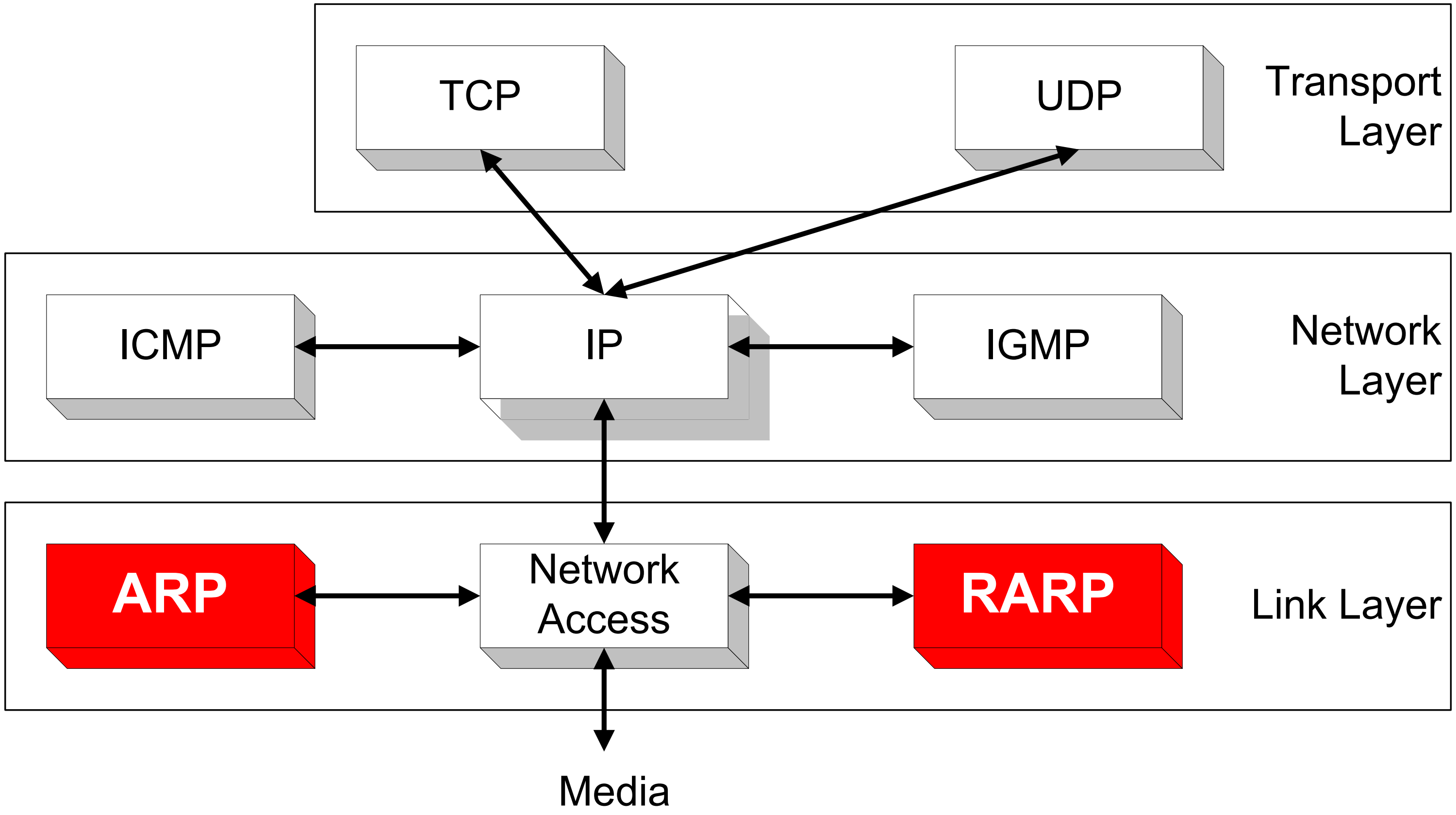


LAN addresses (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
  - MAC address: like Social Security Number
  - IP address: like postal address
- MAC flat address ➜ portability
  - can move LAN card from one LAN to another
- IP hierarchical address *not* portable
  - address depends on IP subnet to which node is attached

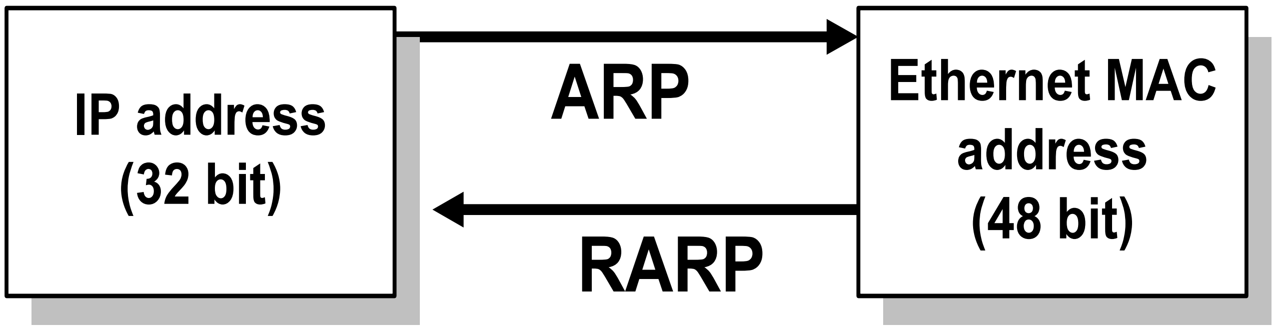
ARP/RARP

ARP/RARP


TCP
UDP
Transport Layer
ICMP
IP
IGMP
Network Layer
ARP
Network Access
RARP
Link Layer
Media

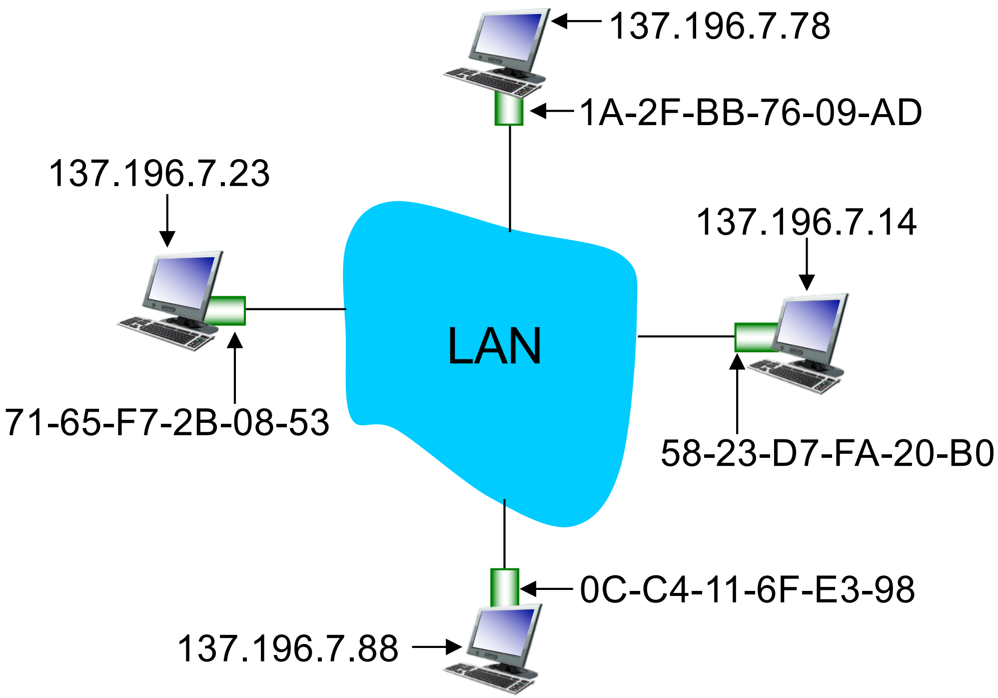
ARP: IP to MAC, RARP: MAC to IP

- Note:
  - The Internet is based on IP addresses
  - Data link protocols (Ethernet, FDDI, ATM) may have different (MAC) addresses
- The ARP and RARP protocols perform the translation between IP addresses and MAC layer addresses
- RARP - reverse address resolution protocol



ARP: address resolution protocol

Question: how to determine interface's MAC address, knowing its IP address?



ARP table: each IP node (host, router) on LAN has table

- IP/MAC address mappings for some LAN nodes:

  < IP address; MAC address; TTL>

- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

ARP Packet Format

Ethernet II header

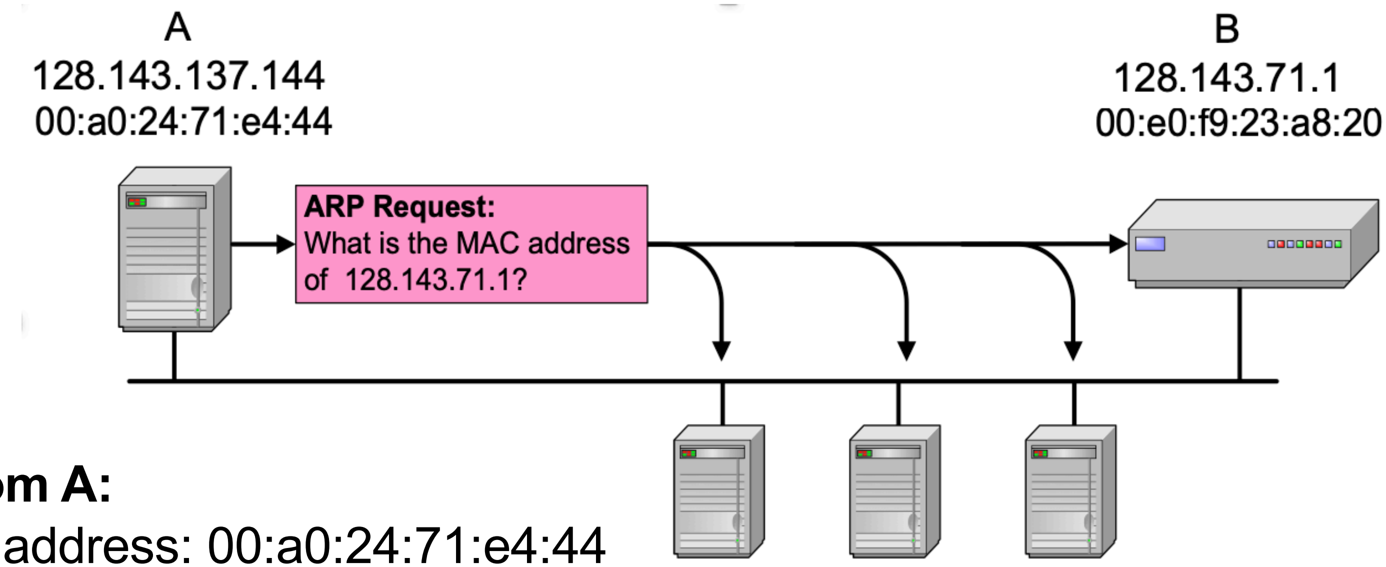
Destination address	Source address	Type 0x8060	ARP Request or ARP Reply	Padding	CRC
6	6	2	28	10	4

<table>
<tr><td colspan="2">Hardware type (2 bytes)</td><td>Protocol type (2 bytes)</td></tr>
<tr><td>Hardware address length (1 byte)</td><td>Protocol address length (1 byte)</td><td>Operation code (2 bytes)</td></tr>
<tr><td colspan="3">Source hardware address*</td></tr>
<tr><td colspan="3">Source protocol address*</td></tr>
<tr><td colspan="3">Target hardware address*</td></tr>
<tr><td colspan="3">Target protocol address*</td></tr>
</table>

* Note: The length of the address fields is determined by the corresponding address length fields

ARP protocol: same LAN

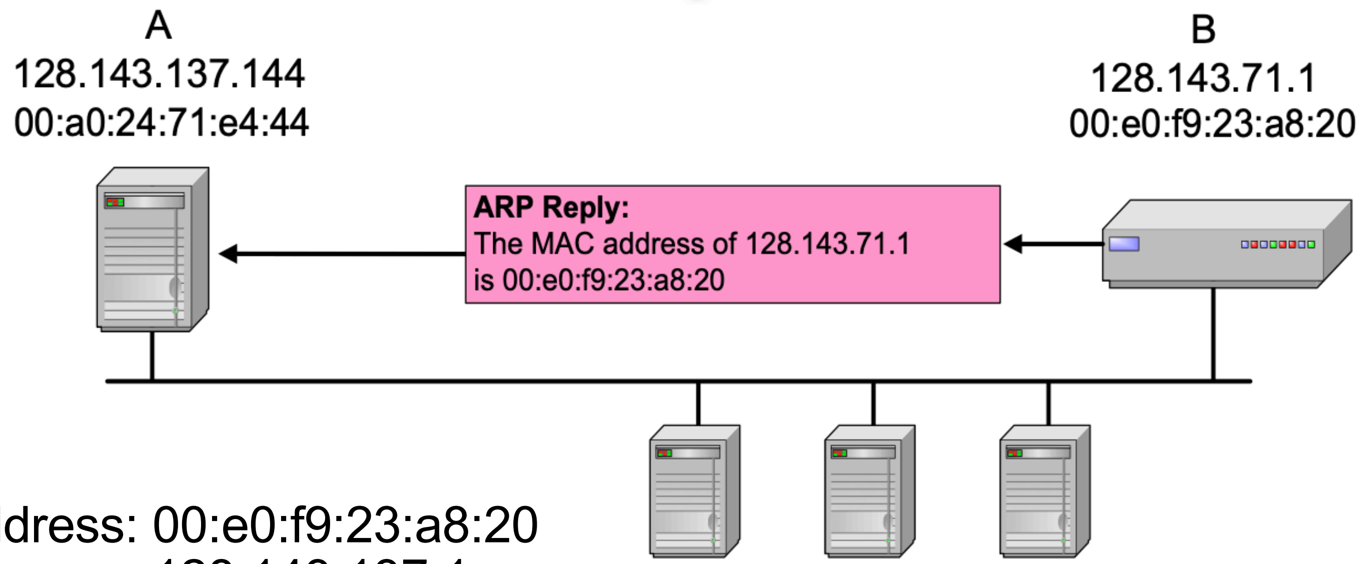
- A wants to send datagram to B
  - B's MAC address not in A's ARP table.
- A broadcasts ARP query packet, containing B's IP address
  - dest MAC address = FF-FF-FF-FF-FF-FF
  - all nodes on LAN receive ARP query
- Only B answers A



ARP Request from A:
Source hardware address: 00:a0:24:71:e4:44
Source protocol address: 128.143.137.144
Target hardware address: 00:00:00:00:00:00
Target protocol address: 128.143.137.1

ARP protocol: same LAN

- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables *without net administrator*



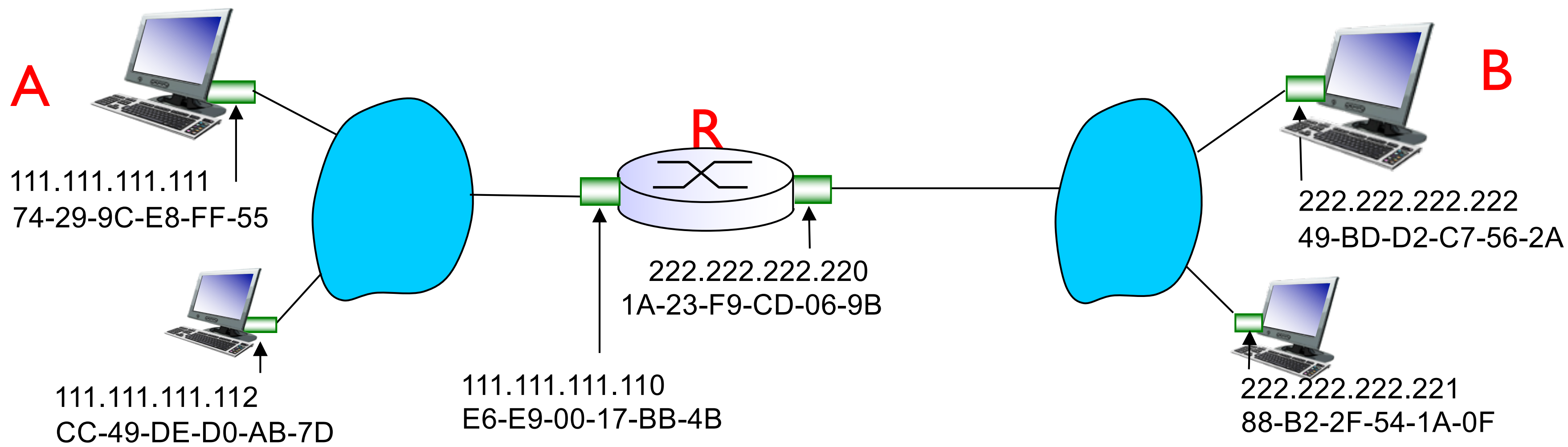
ARP Reply from B:

Source hardware address: 00:e0:f9:23:a8:20
Source protocol address: 128.143.137.1
Target hardware address: 00:a0:24:71:e4:44
Target protocol address: 128.143.137.144

Addressing: routing to another LAN

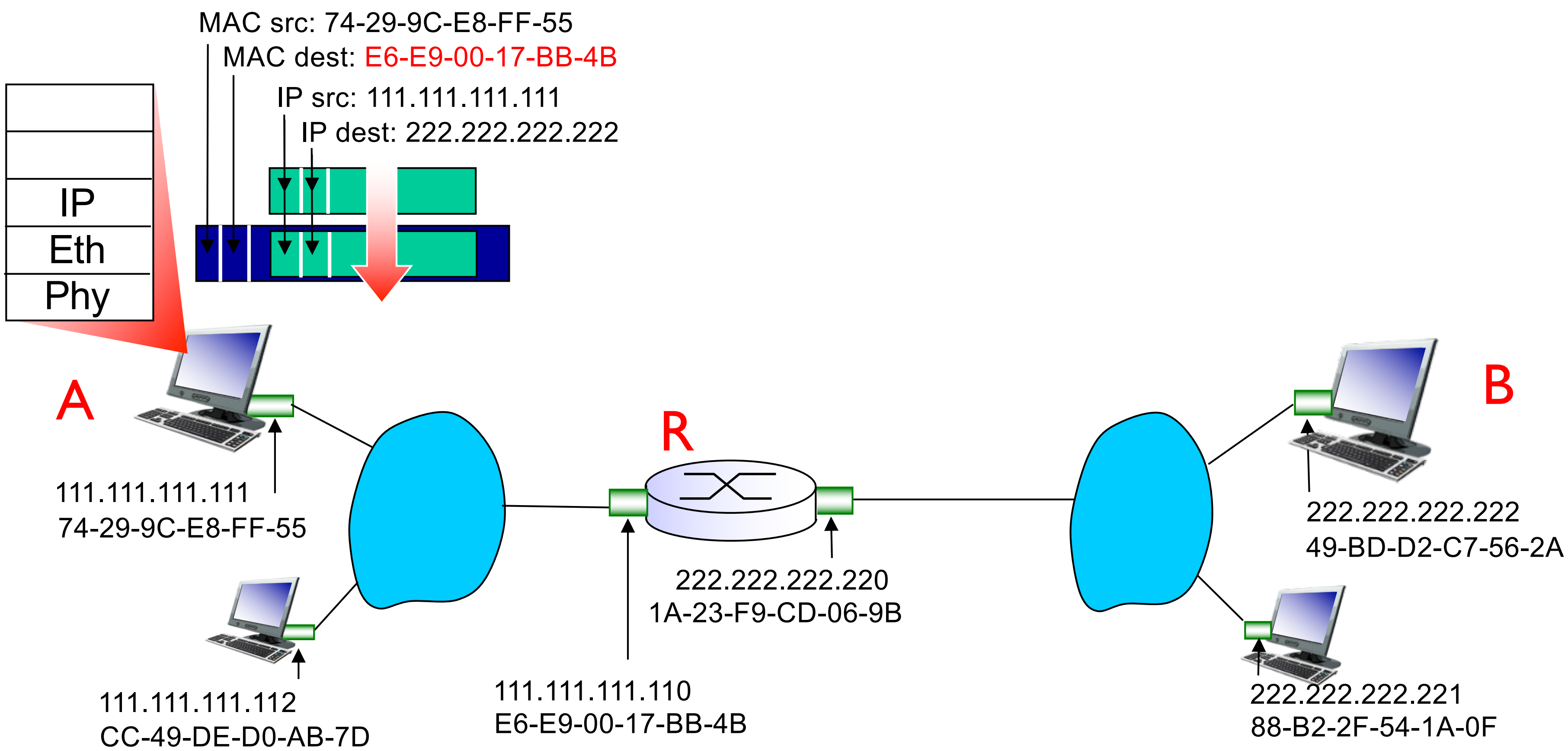
walkthrough: send datagram from A to B via R

- focus on addressing – at IP (datagram) and MAC layer (frame)
- assume A knows B's IP address
- assume A knows IP address of first hop router, R (how?)
- assume A knows R's MAC address (how?)



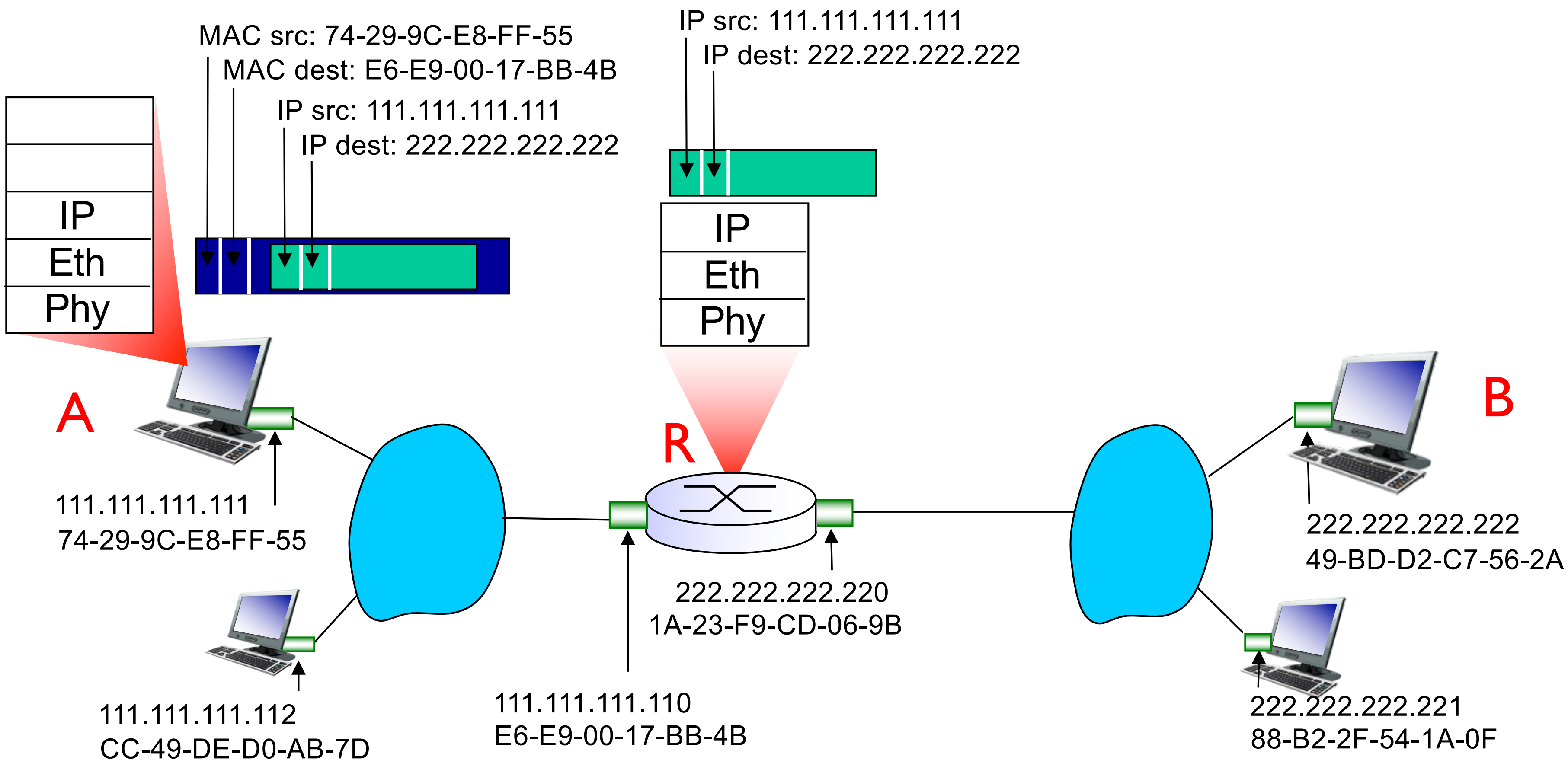
Addressing: routing to another LAN

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame with R's MAC address as dest, frame contains A-to-B IP datagram



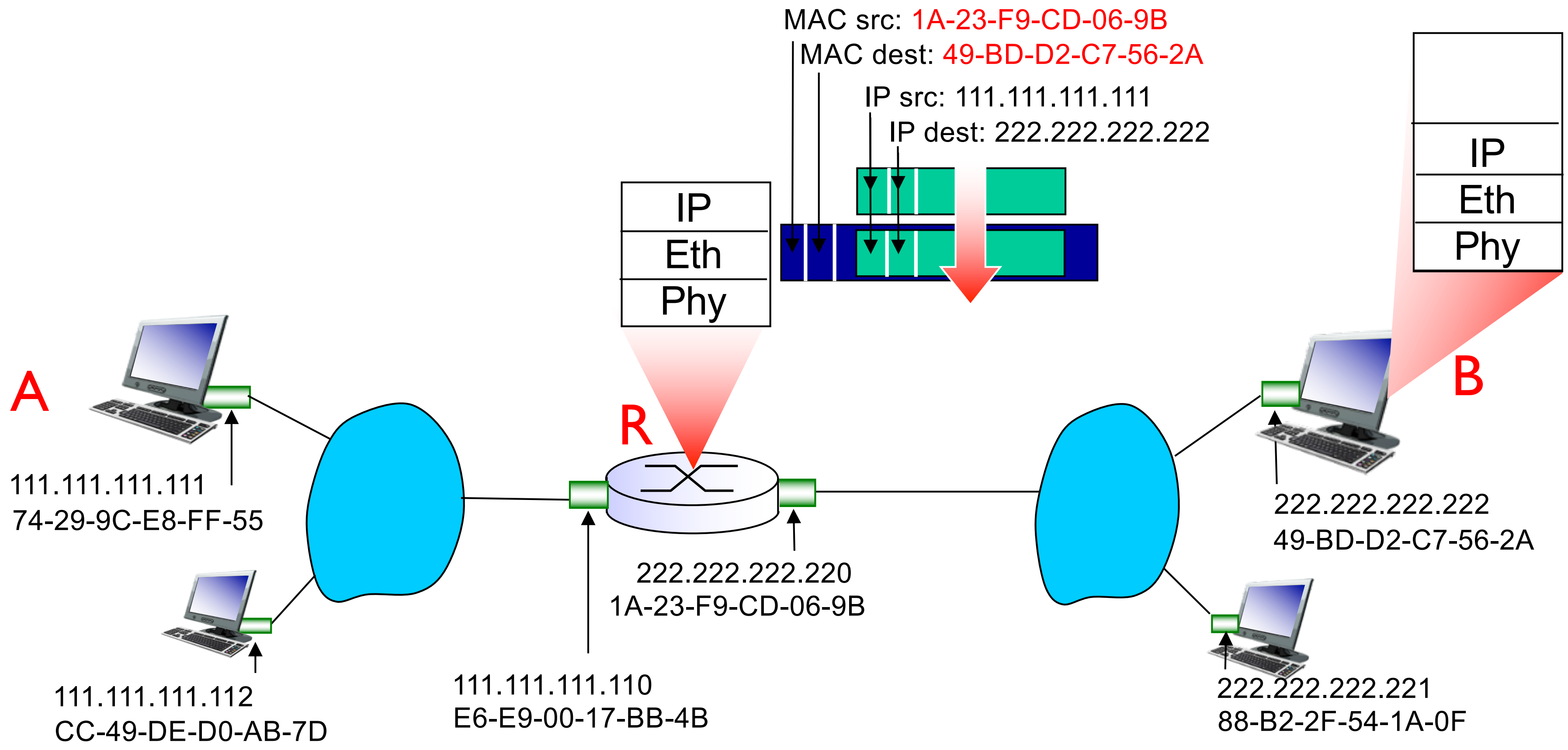
Addressing: routing to another LAN

- frame sent from A to R
- frame received at R, datagram removed, passed up to IP



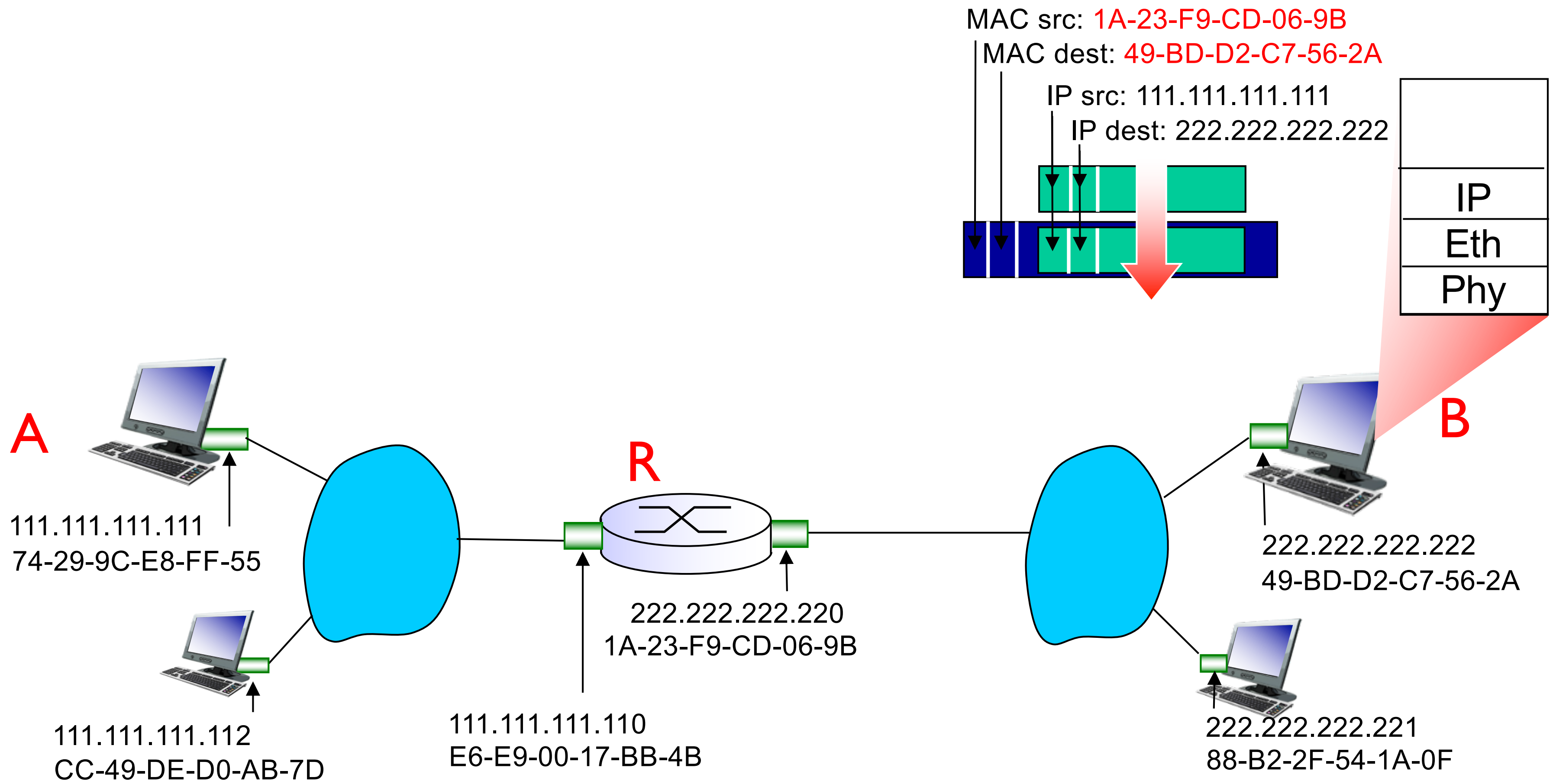
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



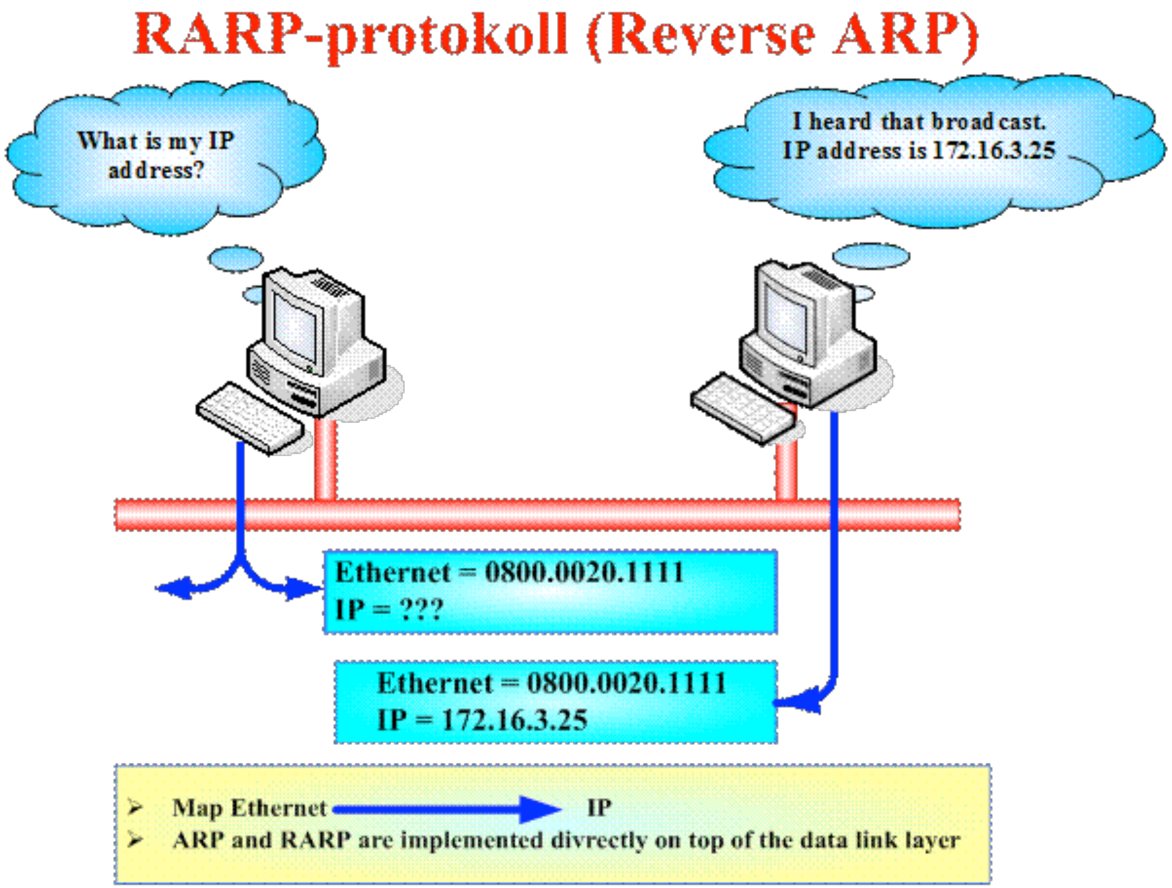
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



RARP: Reverse ARP

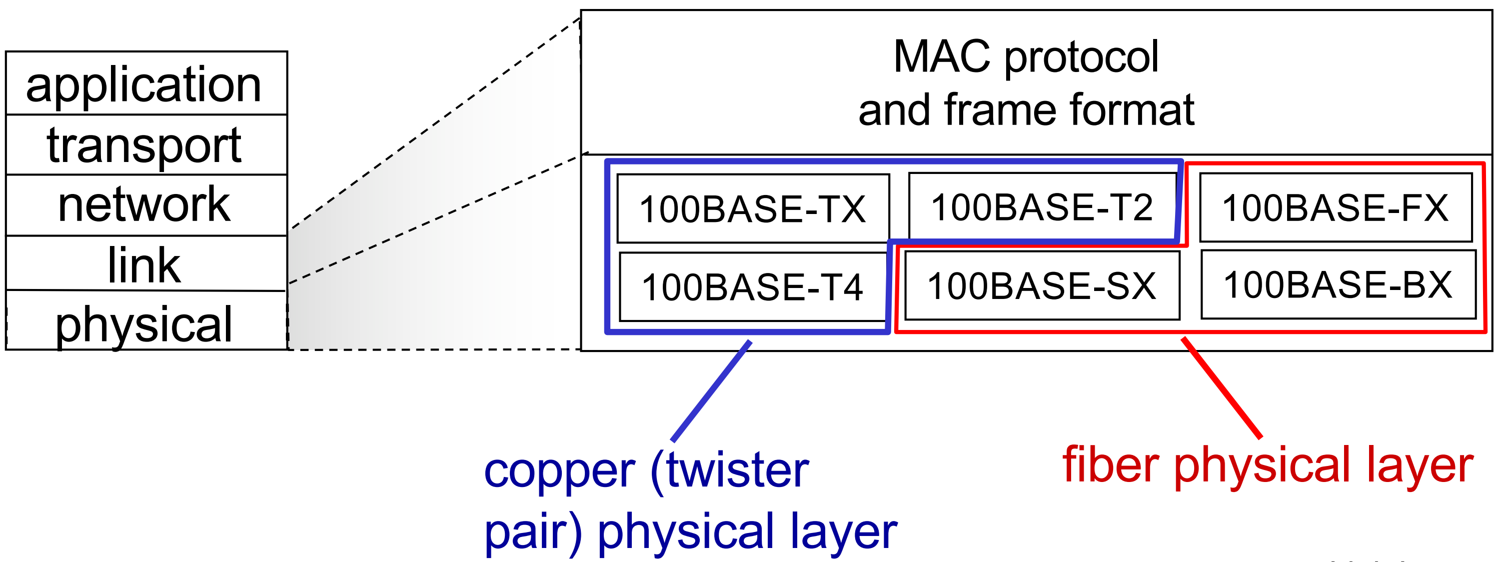
- The Reverse Address Resolution Protocol (RARP) is an obsolete computer networking protocol used by a client computer to request its Internet Protocol (IPv4) address from a computer network, when all it has available is its MAC address.
- RARP has been rendered obsolete by the BOOTP and the modern DHCP, which both support a much greater feature set than RARP.



LAN: Ethernet

802.3 Ethernet standards: link & physical layers

- *many* different Ethernet standards
  - common MAC protocol and frame format
  - different speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10Gbps
  - different physical layer media: fiber, cable



Ethernet and IEEE 802.3: Any Difference?

- On a conceptual level, they are identical. But there are subtle differences that are relevant if we deal with TCP/IP.
- ***"Ethernet" (Ethernet II, DIX)***
  - An industry standards from 1982 that is based on the first implementation of CSMA/CD by Xerox.
  - Predominant version of CSMA/CD in the US.
- **802.3:**
  - IEEE's version of CSMA/CD from 1985.
  - Interoperates with 802.2 (LLC) as higher layer.
- **Difference for our purposes:** Ethernet and 802.3 use different methods to encapsulate an IP datagram.

Ethernet II, DIX Encapsulation (RFC 894)

destination address	source address	type	data	CRC
6	6	2	46-1500	4

0800	IP datagram
2	38-1492

0806	ARP request/reply	PAD
2	28	10

0835	RARP request/reply	PAD
2	28	10

Ethernet frame structure

Sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame

7+1 Pre+SFD	6 DA	6 SA	2 T|L	0-1500 LLC data	<46 (Pad)	4 FCS
Preamble +Starting Frame Delimiter	Destination Address	Source Address	Frame Type or Length	Logical Link Control+ Payload Data	Padding Field	Frame Control Sum

- *Pre+SFD:* 7 bytes with pattern 10101010 + one byte with pattern 10101011 (SFD - Starting Frame Delimiter), used to synchronize receiver, sender clock rates
- *DA/SA:* 6 byte source, destination MAC addresses
  - if adapter receives frame with matching destination address, or with broadcast address (e.g. ARP packet), it passes data in frame to network layer protocol
  - otherwise, adapter discards frame
- *T/L:* 2 bytes indicates higher layer protocol Type (mostly IP but others possible, e.g., Novell IPX, AppleTalk) or Length of Frame (old-outdated)
- *Pad:* – have only if LLC data less then 46 byte
- *FCS:* cyclic redundancy check at receiver for every field exclude Pre+SFD and FCS
  - error detected: frame is dropped

IEEE 802.2/802.3 Encapsulation (RFC 1042)

←802.3 MAC→←802.2 LLC→←802.2 SNAP→

destination address	source address	length	DSAP AA	SSAP AA	cntl 03	org code 0	type	data	CRC
6	6	2	1	1	1	3	2	38-1492	4

- **destination address, source address:** MAC addresses are 48 bit
- **length**: frame length in number of bytes
- **DSAP, SSAP**: always set to `0xaa`
- **Ctrl:** set to 3
- **org code:** set to 0
- **type field** identifies the content of the data field
- **CRC:** cylic redundancy check

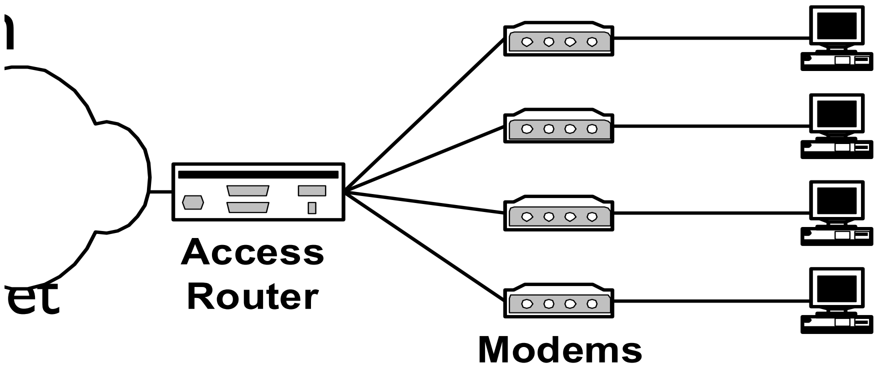
0800	IP datagram
2	38-1492

0806	ARP request/reply	PAD
2	28	10

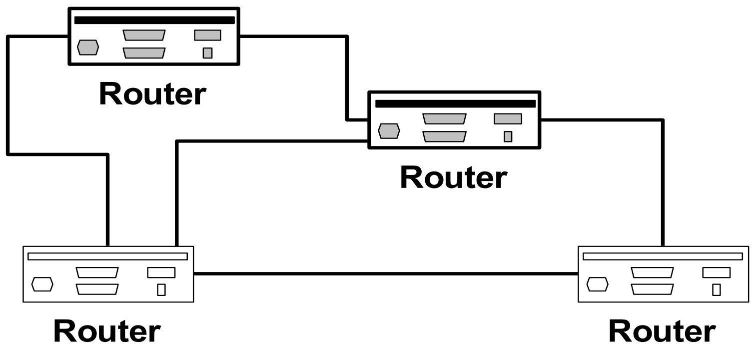
0835	RARP request/reply	PAD
2	28	10

PPP - Point-to-Point Protocol

- The **PPP protocol** is a data link protocol for transmission on a serial link
- Use of PPP today:
  - Dial-in or DSL access to Internet
  - Routers connected by point-to-point links
- Main purpose of PPP is encapsulation of IP datagrams
- PPP was proposed in 1992; a predecessor of PPP was the Serial Link IP (SLIP) protocol



Dial-Up Access



Point-to-Point Network

PPP - IP encapsulation

- The frame format of PPP is similar to HDLC and the 802.2 LLC frame format:

flag	addr	ctrl	protocol	data	CRC	flag
7E	**FF**	**03**				**7E**
1	1	1	2	<= 1500	2	1

0021	**IP datagram**
C021	**link control data**
8021	**network control data**

PPP

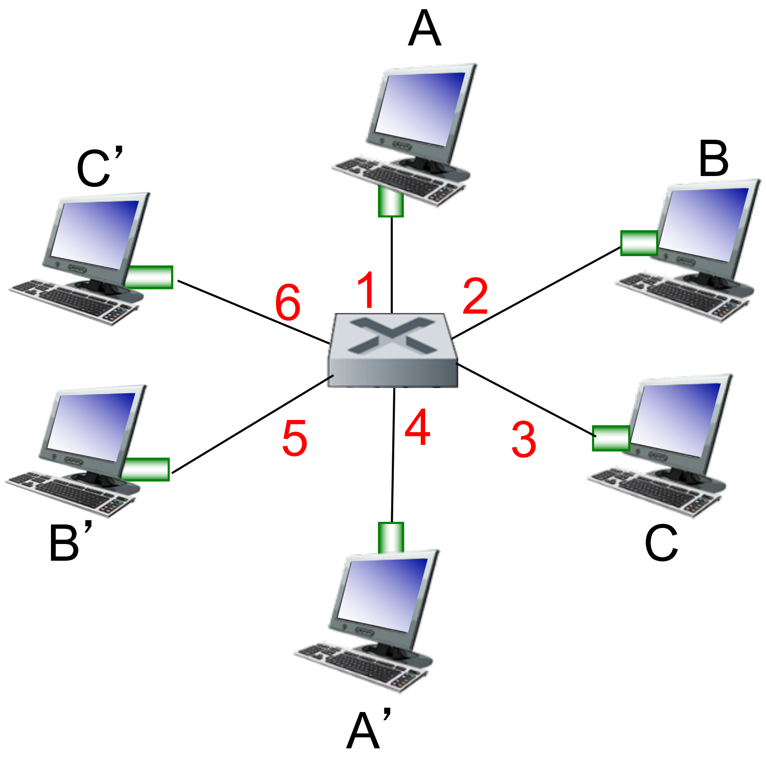
Other than a framing method PPP provides:

- The **link control protocol (LCP)** which is responsible for establishing, configuring, and negotiating a data-link connection
  - LCP is specified in RFC 1331.

- For each network layer protocol supported by PPP, there is one **network control protocol (NCP)**
  - The NCP for IP is specified in RFC 1332

Ethernet Switch

- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, but no collisions; full duplex
  - each link is its own collision domain
- *switching:* A-to-A' and B-to-B' can transmit simultaneously, without collisions



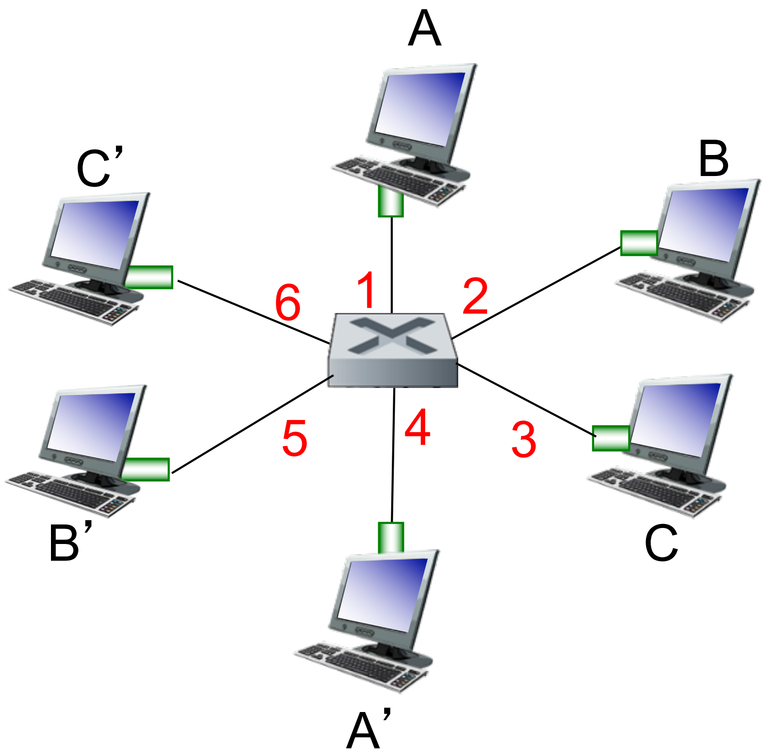
switch with six interfaces (1,2,3,4,5,6)

Switch forwarding table

Q: how does switch know A' reachable via interface 4, B' reachable via interface 5?

- *A: each switch has a switch table, each entry:*
  - *(MAC address of host, interface to reach host, time stamp)*
  - *looks like a routing table!*

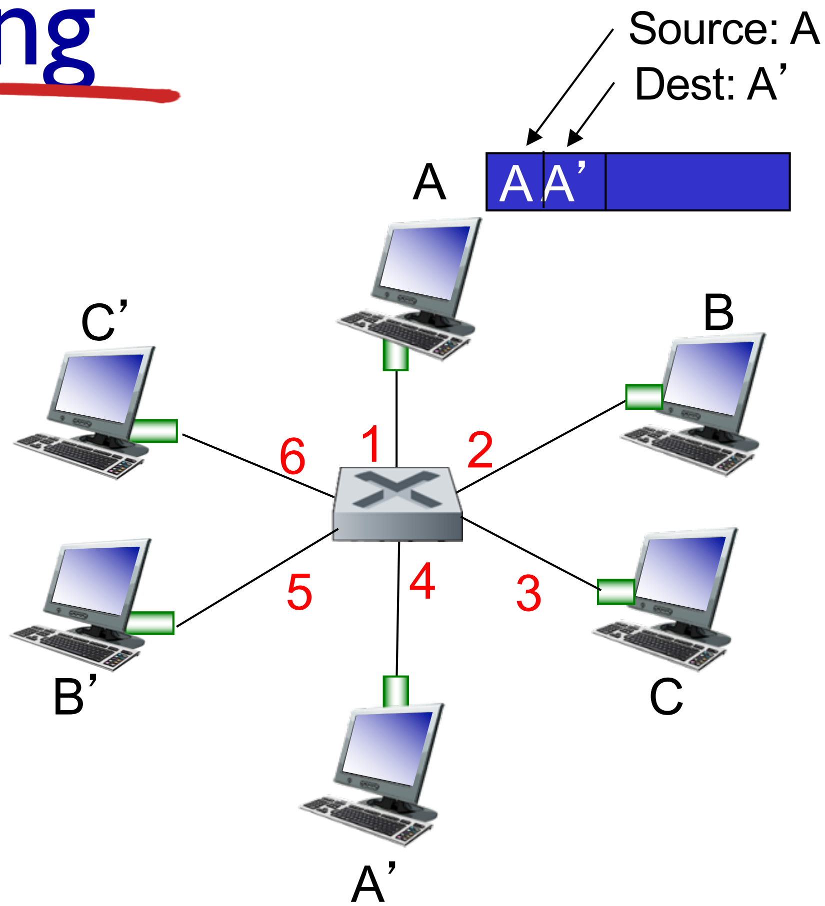
Q: how are entries created, maintained in switch table?
  - *something like a routing protocol?*



switch with six interfaces (1,2,3,4,5,6)

Switch: self-learning

- switch *learns* which hosts can be reached through which interfaces
  - when frame received, switch "learns" location of sender: incoming LAN segment
  - records sender/location pair in switch table



MAC addr	interface	TTL
A	*1*	*60*

Switch table (initially empty)

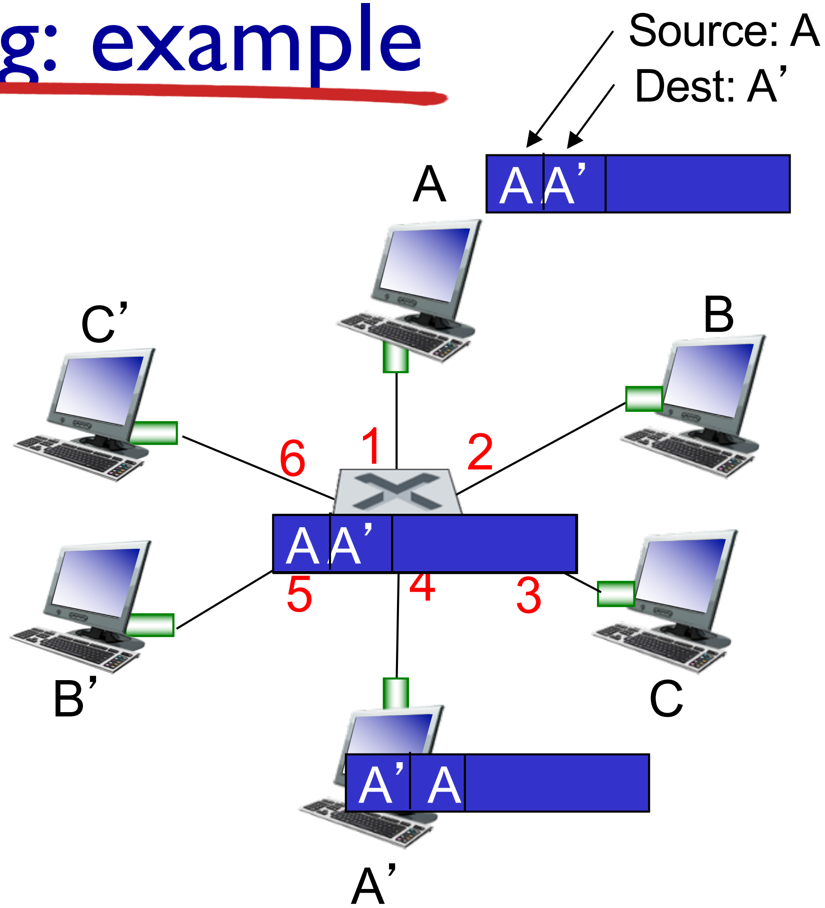
Switch: frame filtering/forwarding

when frame received at switch:

1. record incoming link, MAC address of sending host
2. index switch table using MAC destination address
3. if entry found for destination
   then {
   if destination on segment from which frame arrived
      then drop frame
      else forward frame on interface indicated by entry
   }
   else flood /* forward on all interfaces except arriving interface */

Self-learning, forwarding: example

- frame destination, A', locaton unknown: *flood*
- destination A location known: *selectively send on just one link*

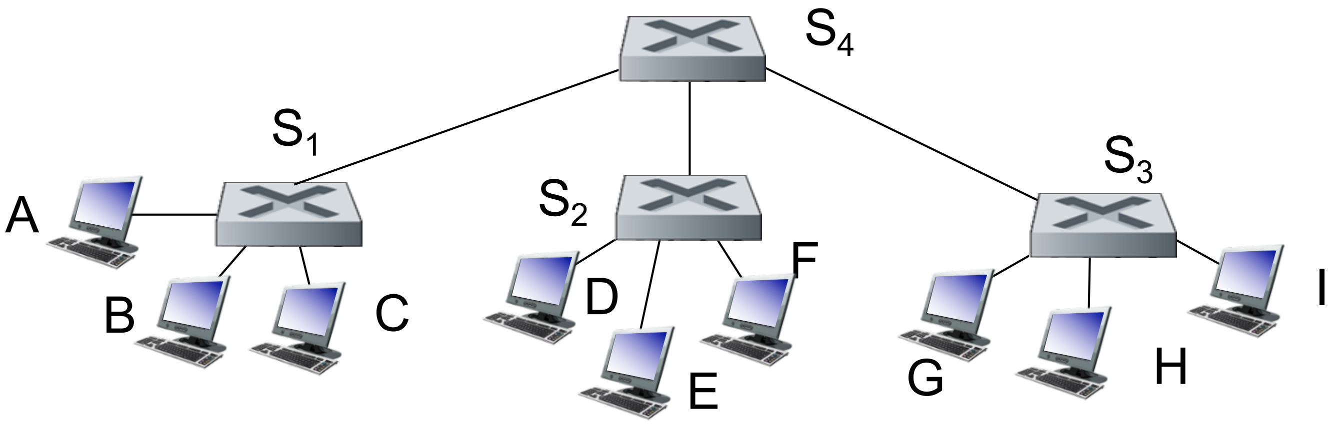


MAC addr	interface	TTL
A	*1*	*60*
A'	*4*	*60*

switch table (initially empty)

Interconnecting switches

- switches can be connected together

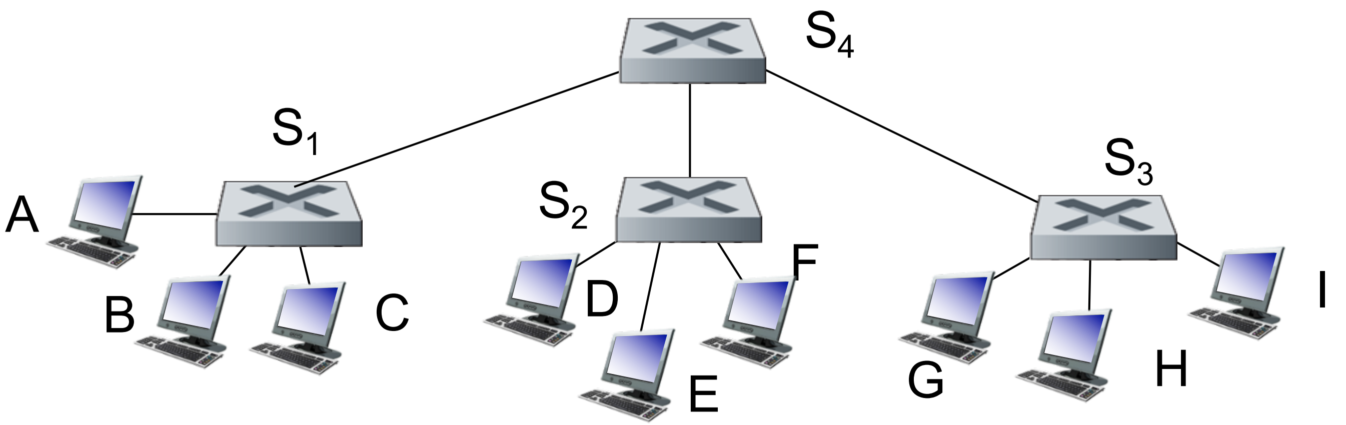


Q: sending from A to G - how does S_1 know to forward frame destined to F via S_4 and S_3?

- *A:* self learning! (works *exactly* the same as in single-switch case!)

Self-learning multi-switch example

Suppose C sends frame to I, I responds to C



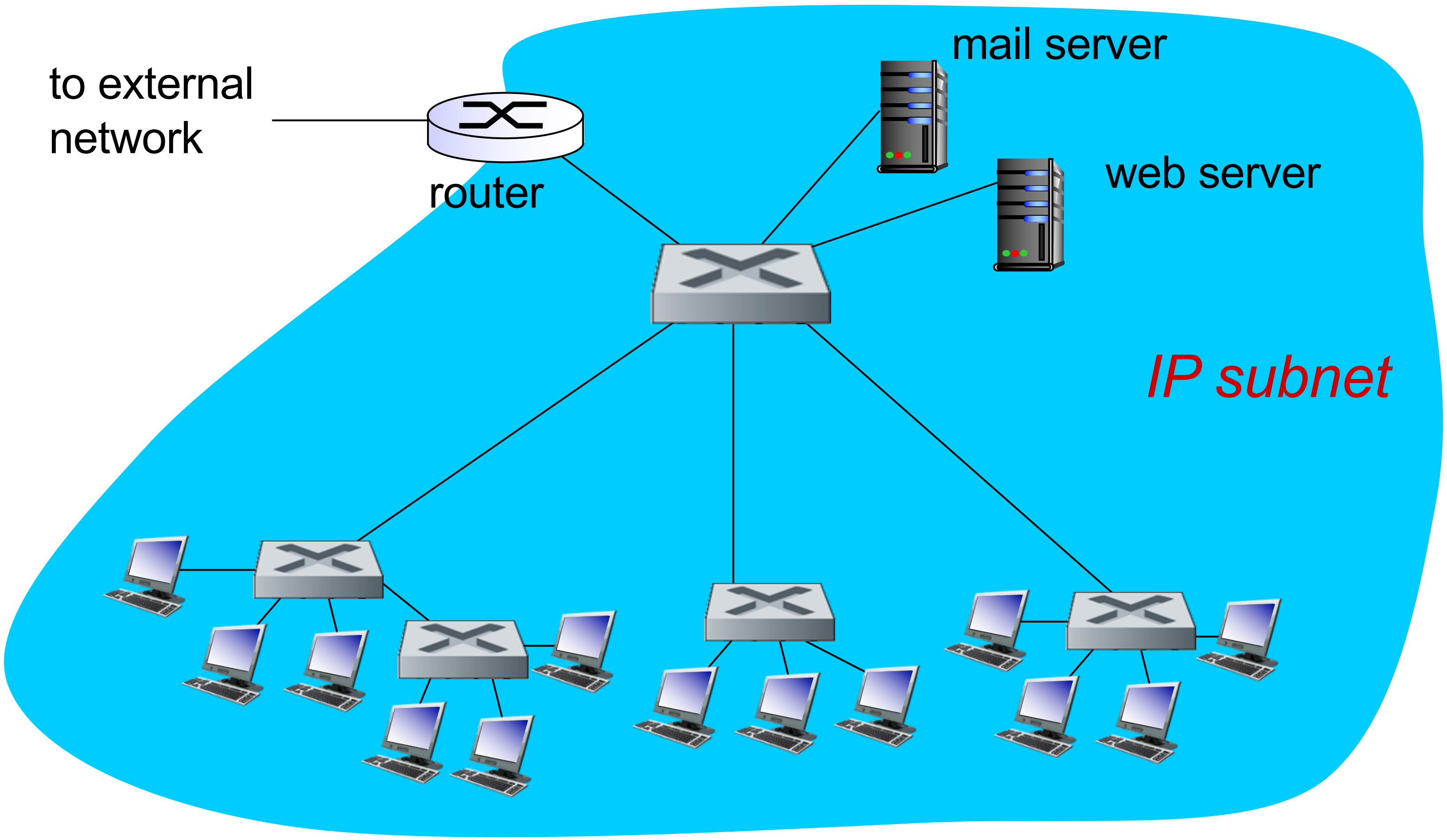
- *Q:* show switch tables and packet forwarding in S_1, S_2, S_3, S_4

MAC addr	interface	TTL

…

MAC addr	interface	TTL

Organisational network



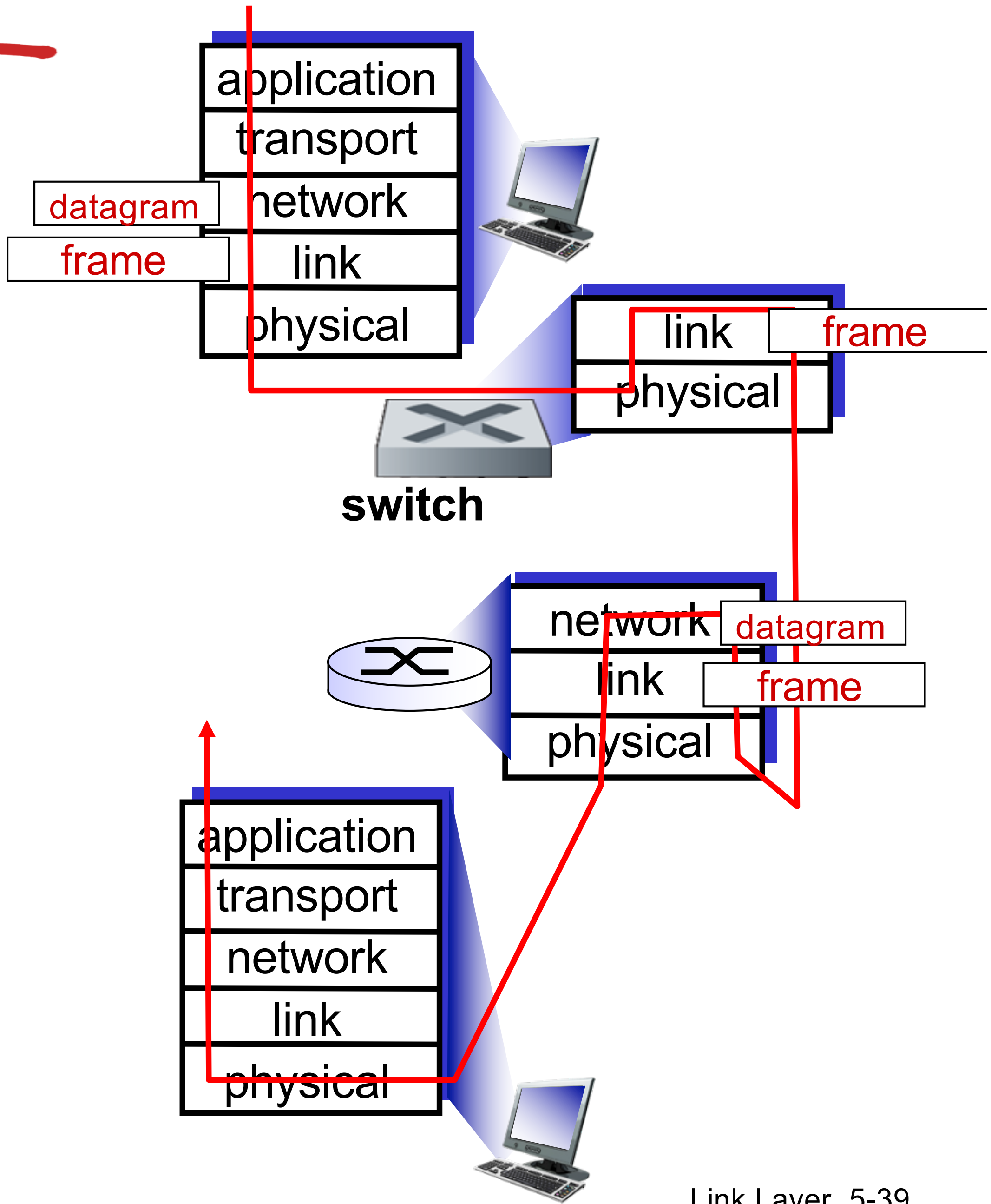
Switches vs. routers

both are store-and-forward:

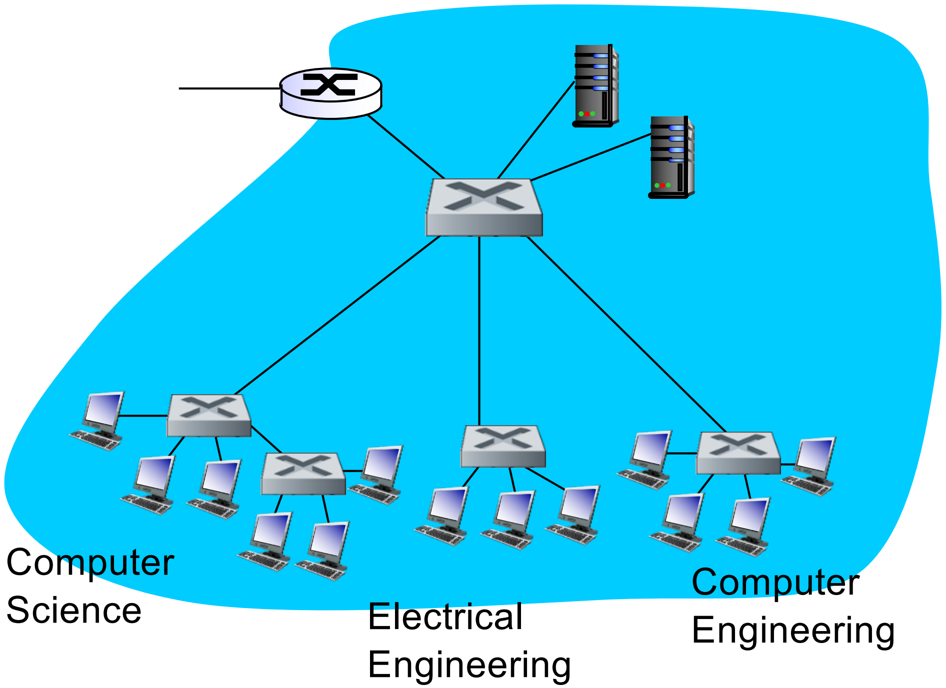
- *routers:* network-layer devices (examine network-layer headers)
- *switches*: link-layer devices (examine link-layer headers)

both have forwarding tables:

- *routers:* compute tables using routing algorithms, IP addresses
- *switches:* learn forwarding table using flooding, learning, MAC addresses



VLANs: motivation



consider:

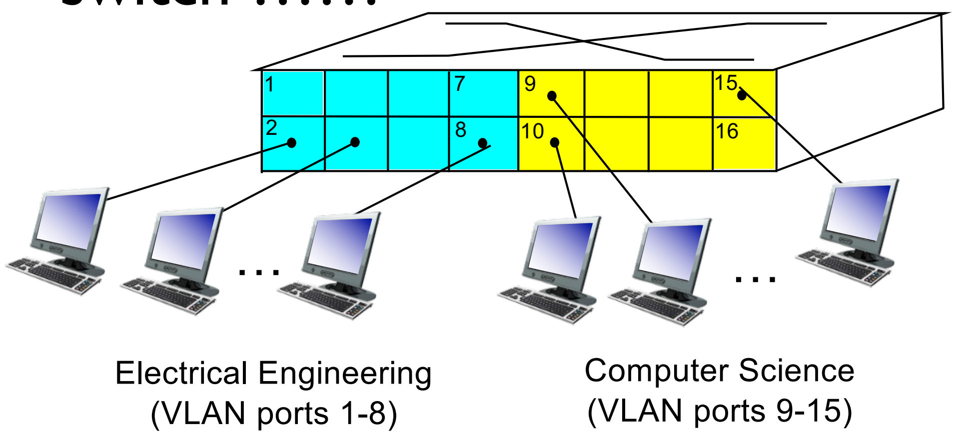
- CS user moves office to EE, but wants connect to CS switch?
- single broadcast domain:
  - all layer-2 broadcast traffic (ARP, DHCP, unknown location of destination MAC address) must cross entire LAN
  - security/privacy, efficiency issues

VLANs

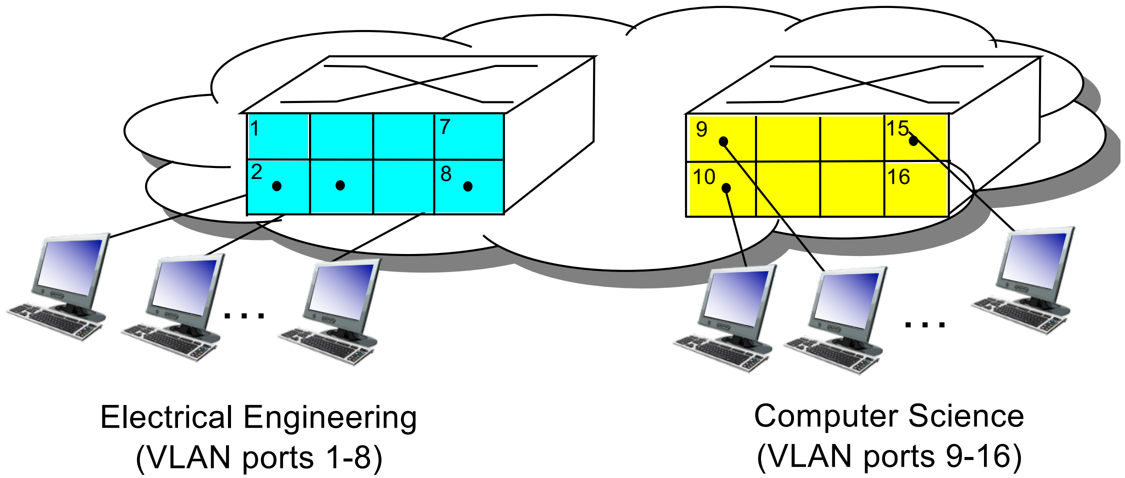
Virtual Local Area Network

switch(es) supporting VLAN capabilities can be configured to define multiple ***virtual*** LANS over single physical LAN infrastructure.

port-based VLAN: switch ports grouped (by switch management software) so that *single* physical switch ……

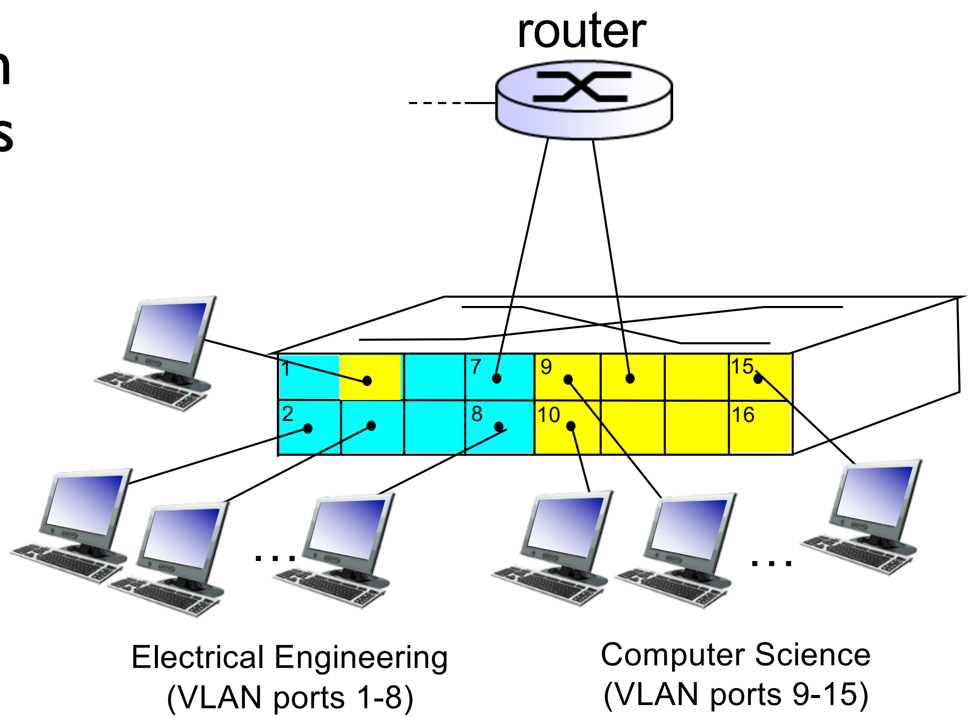


… operates as *multiple* virtual switches



Port-based VLAN

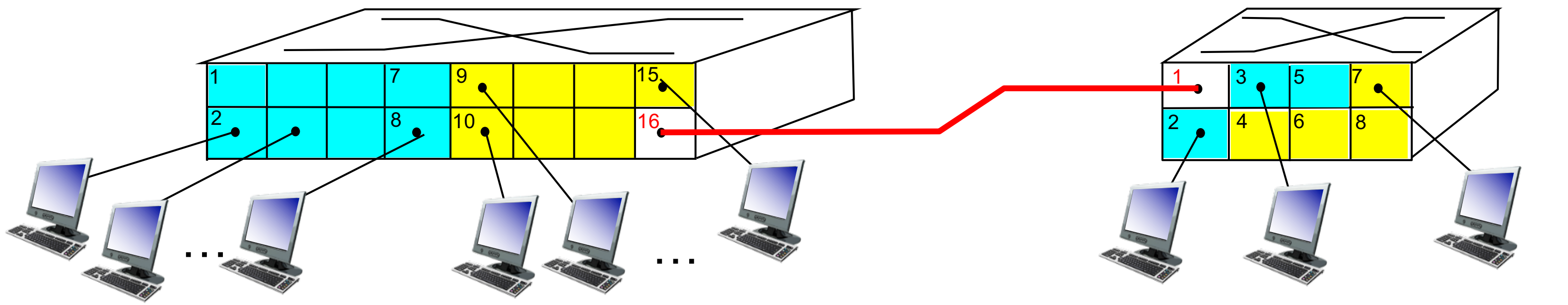
- *traffic isolation:* frames to/from ports 1-8 can *only* reach ports 1-8
  - can also define VLAN based on MAC addresses of endpoints, rather than switch port



- *dynamic membership:* ports can be dynamically assigned among VLANs

- *forwarding between VLANS:* done via routing (just as with separate switches)
  - in practice vendors sell combined switches plus routers

VLANS spanning multiple switches



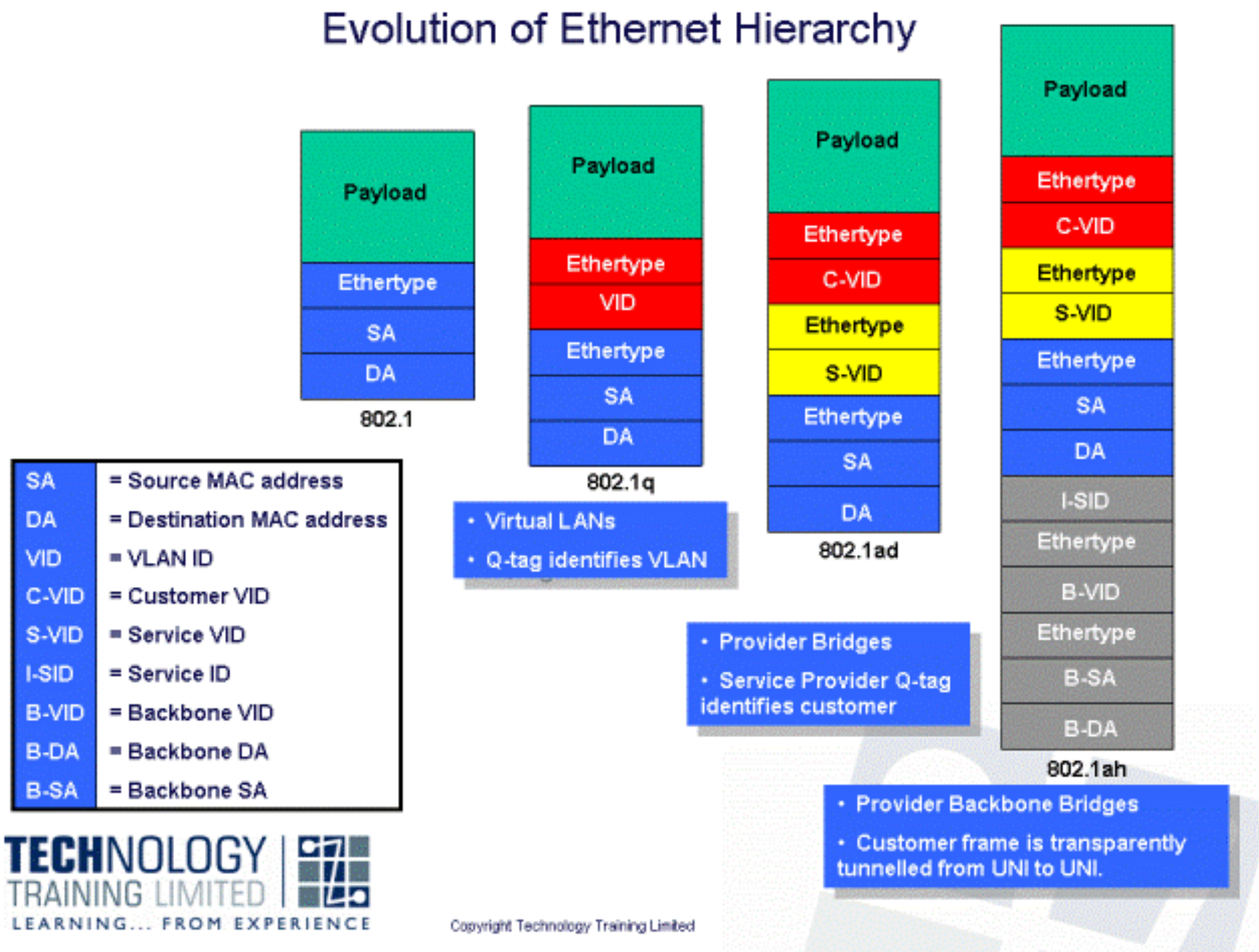
Electrical Engineering
(VLAN ports 1-8)

Computer Science
(VLAN ports 9-15)

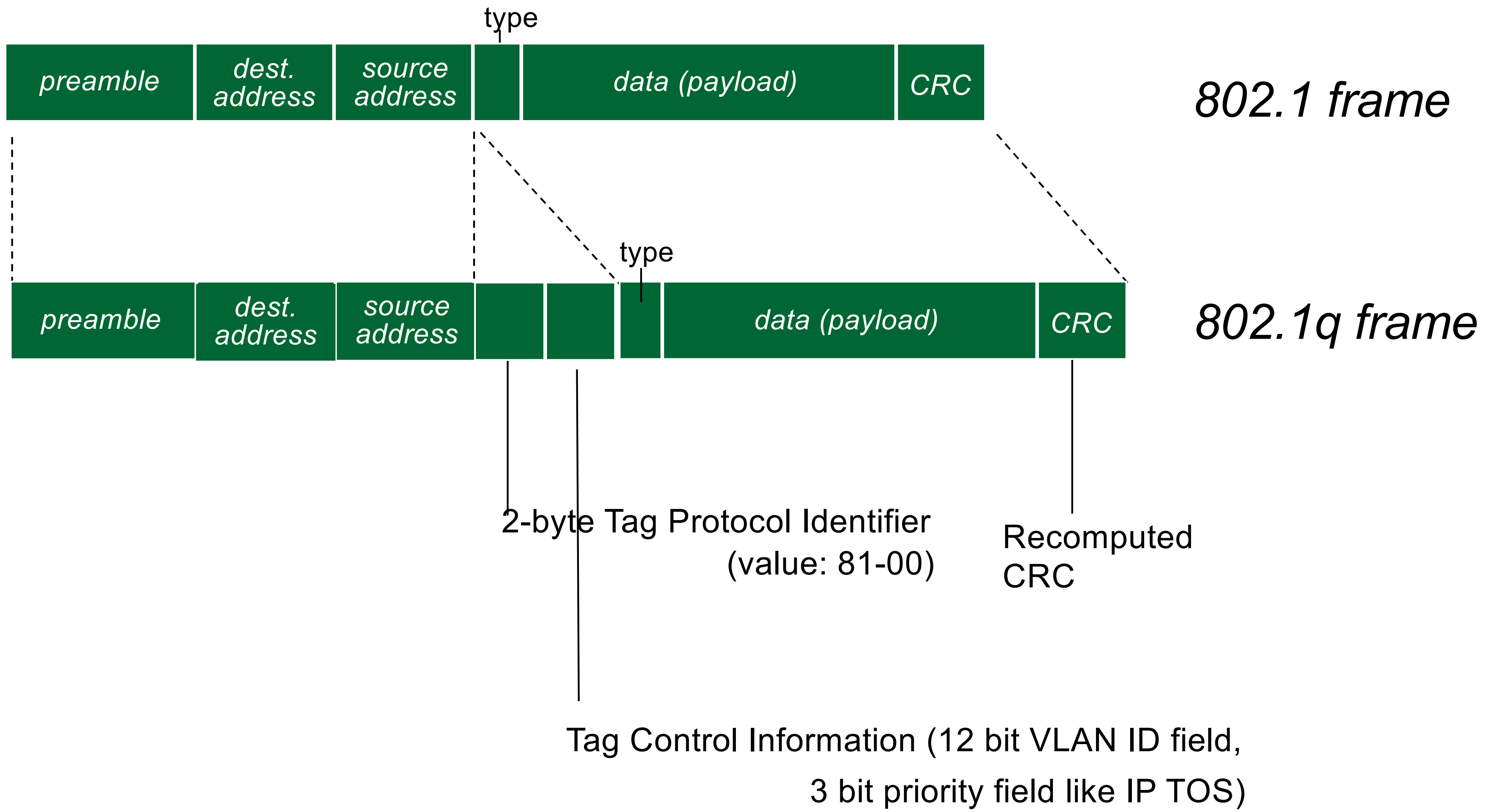
Ports 2,3,5 belong to EE VLAN
Ports 4,6,7,8 belong to CS VLAN

- *trunk port:* carries frames between VLANS defined over multiple physical switches
  - frames forwarded within VLAN between switches can't be vanilla 802.1 frames (must carry VLAN ID info)
  - 802.1q protocol adds/removed additional header fields for frames forwarded between trunk ports

VLAN: Ethernet



802.1Q VLAN frame format

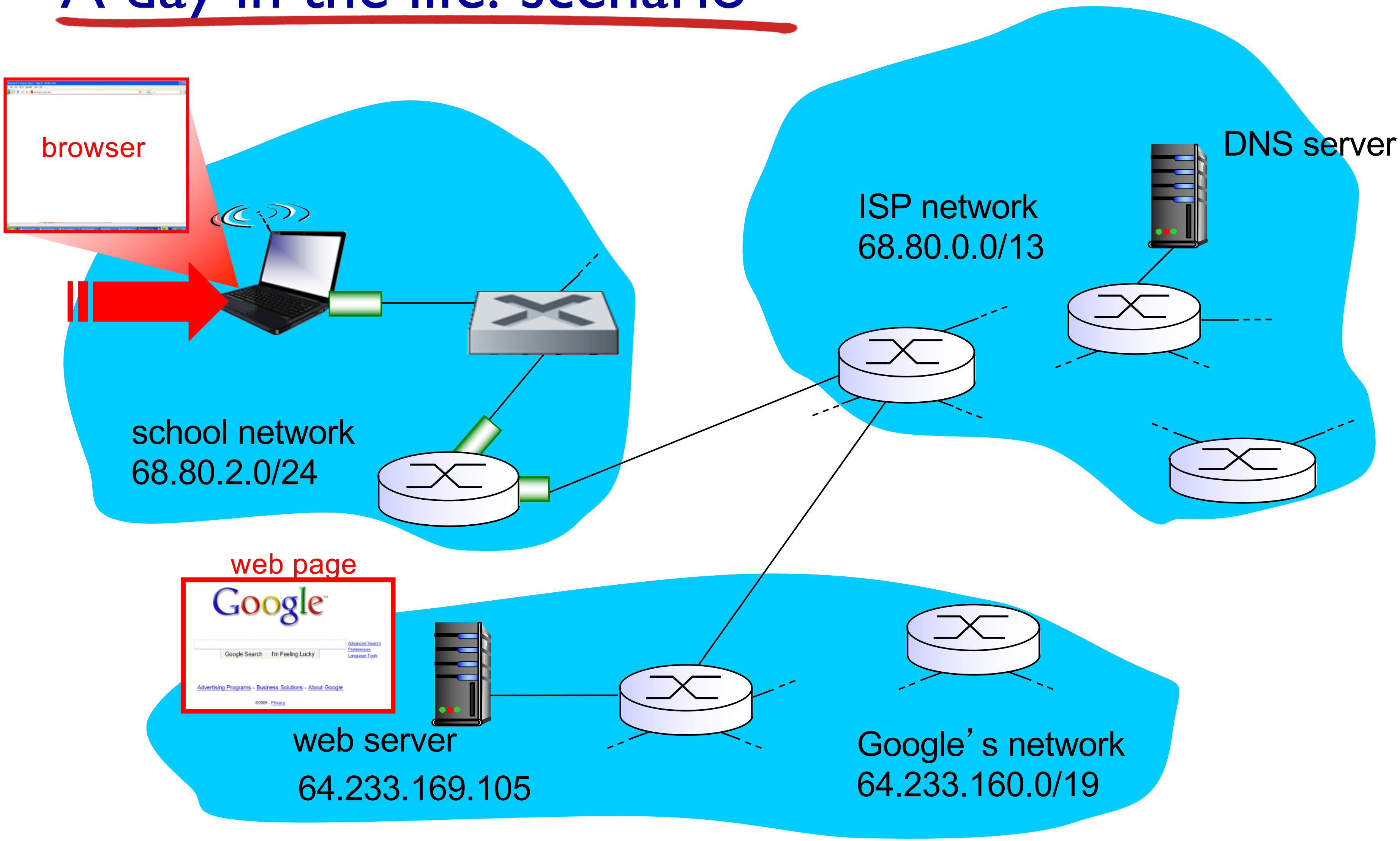


Simple web request description

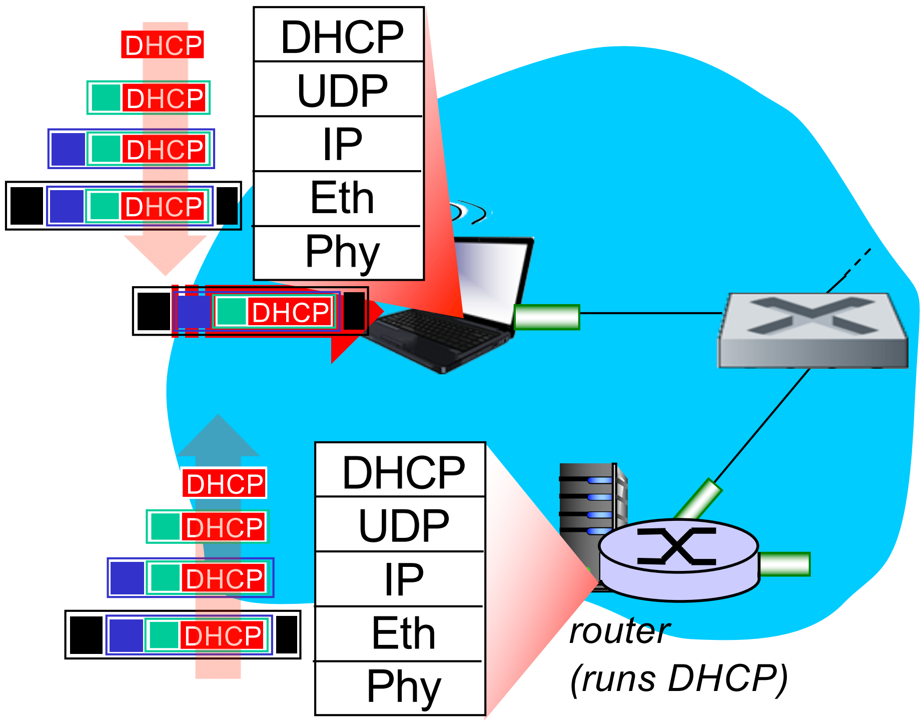
Synthesis: a day in the life of a web request

- Complete look protocol stack!
  - application, transport, network, data-link
- Putting-it-all-together: synthesis!
  - *goal:* identify, review, understand protocols (at all layers) involved in seemingly simple scenario: requesting www page
  - *scenario:* You attaches laptop to campus network, requests/receives www.google.com

A day in the life: scenario

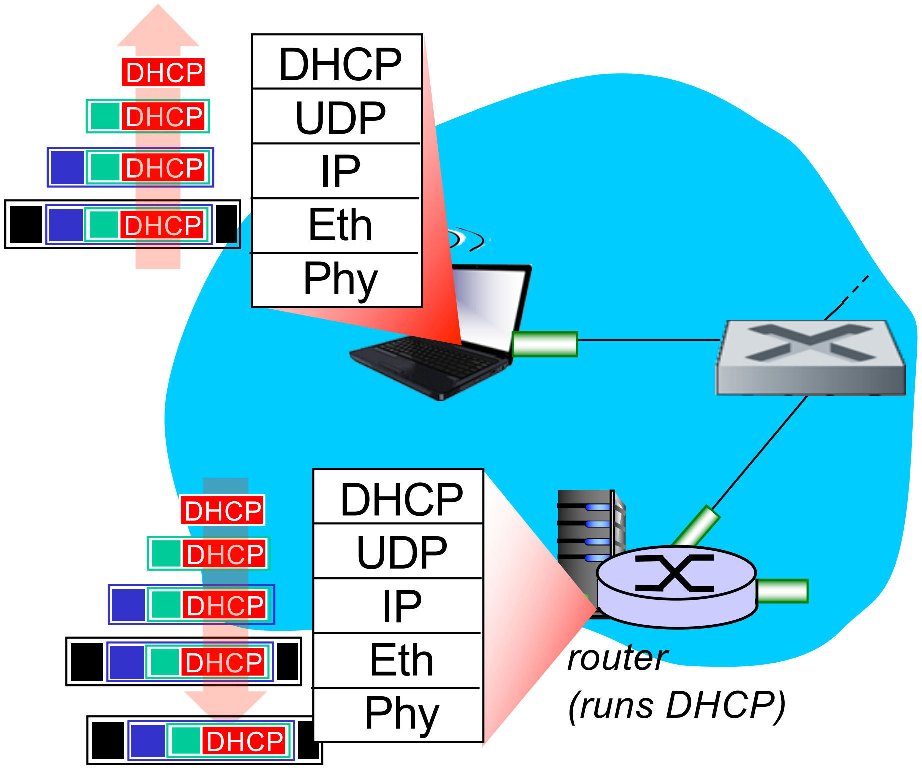


A day in the life… connecting to the Internet



- connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use *DHCP*
- DHCP request *encapsulated* in *UDP*, encapsulated in *IP*, encapsulated in *802.3* Ethernet
- Ethernet frame *broadcast* (dest: FFFFFFFFFFFF) on LAN, received at router running *DHCP* server
- Ethernet *demuxed* to IP demuxed, UDP demuxed to DHCP

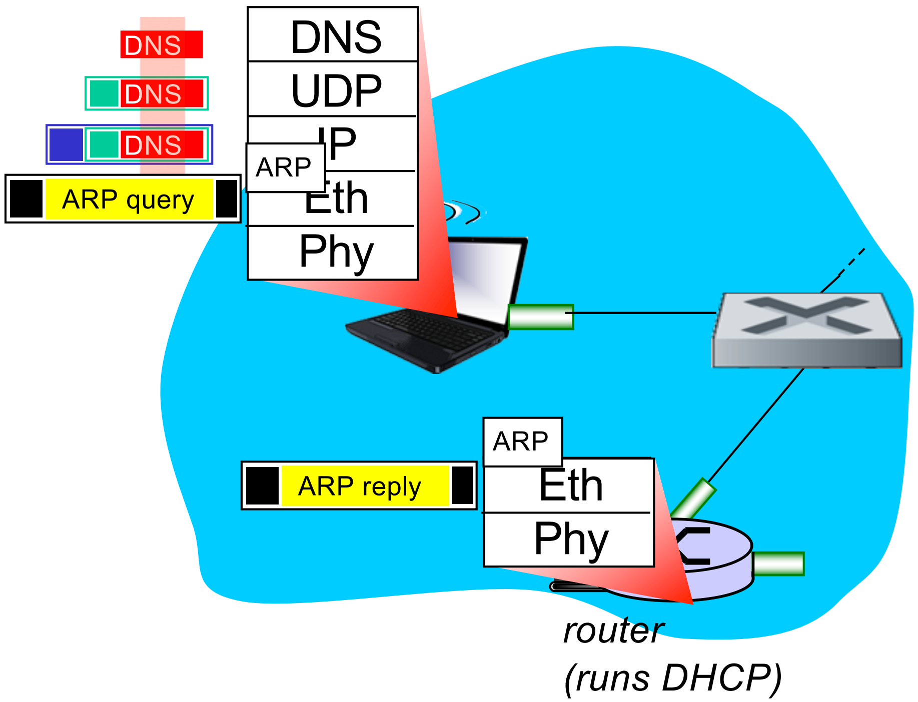
A day in the life… connecting to the Internet



- DHCP server formulates *DHCP ACK* containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation at DHCP server, frame forwarded (*switch learning*) through LAN, demultiplexing at client
- DHCP client receives DHCP ACK reply

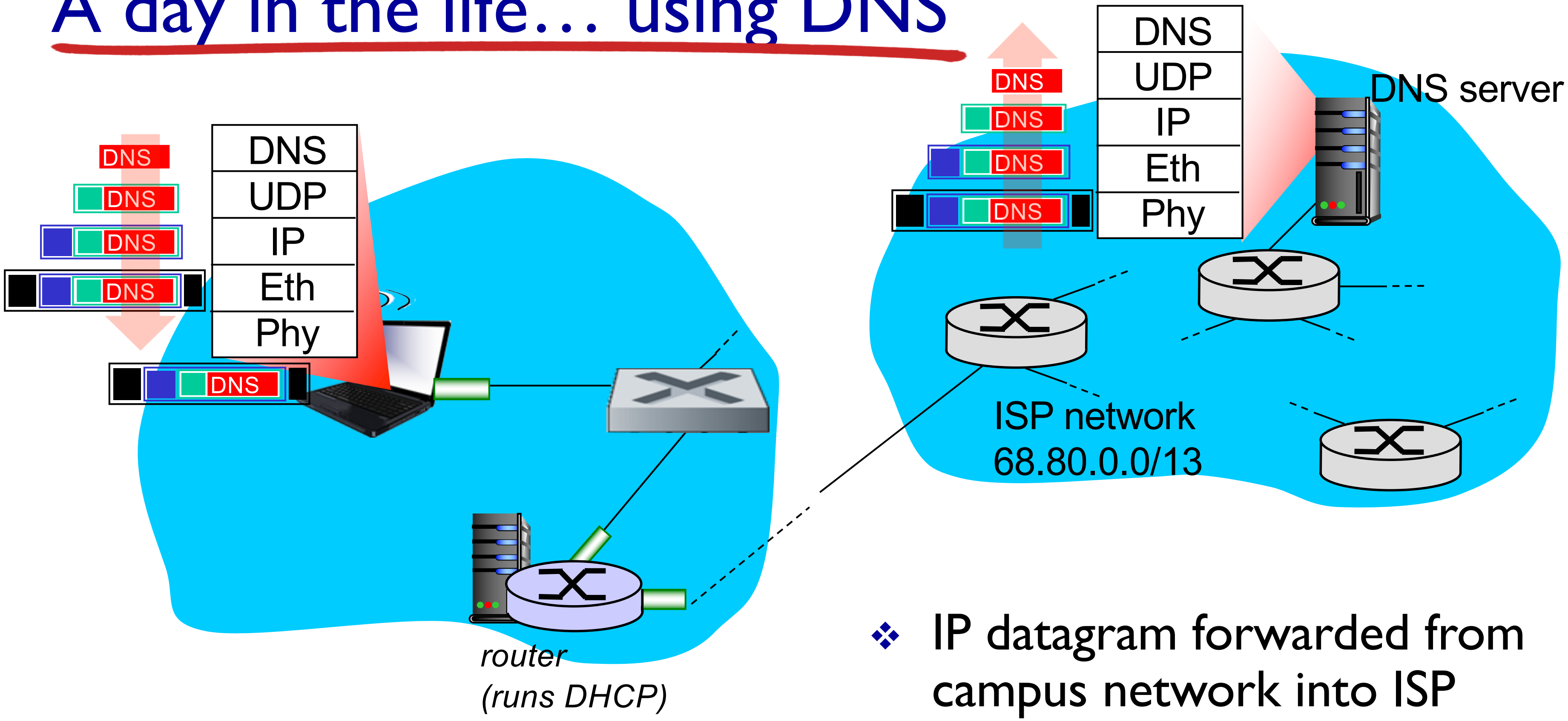
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

A day in the life… ARP (before DNS, before HTTP)



- before sending *HTTP* request, need IP address of www.google.com: *DNS*
- DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. To send frame to router, need MAC address of router interface: *ARP*
- *ARP query* broadcast, received by router, which replies with *ARP reply* giving MAC address of router interface
- client now knows MAC address of first hop router, so can now send frame containing DNS query

A day in the life… using DNS

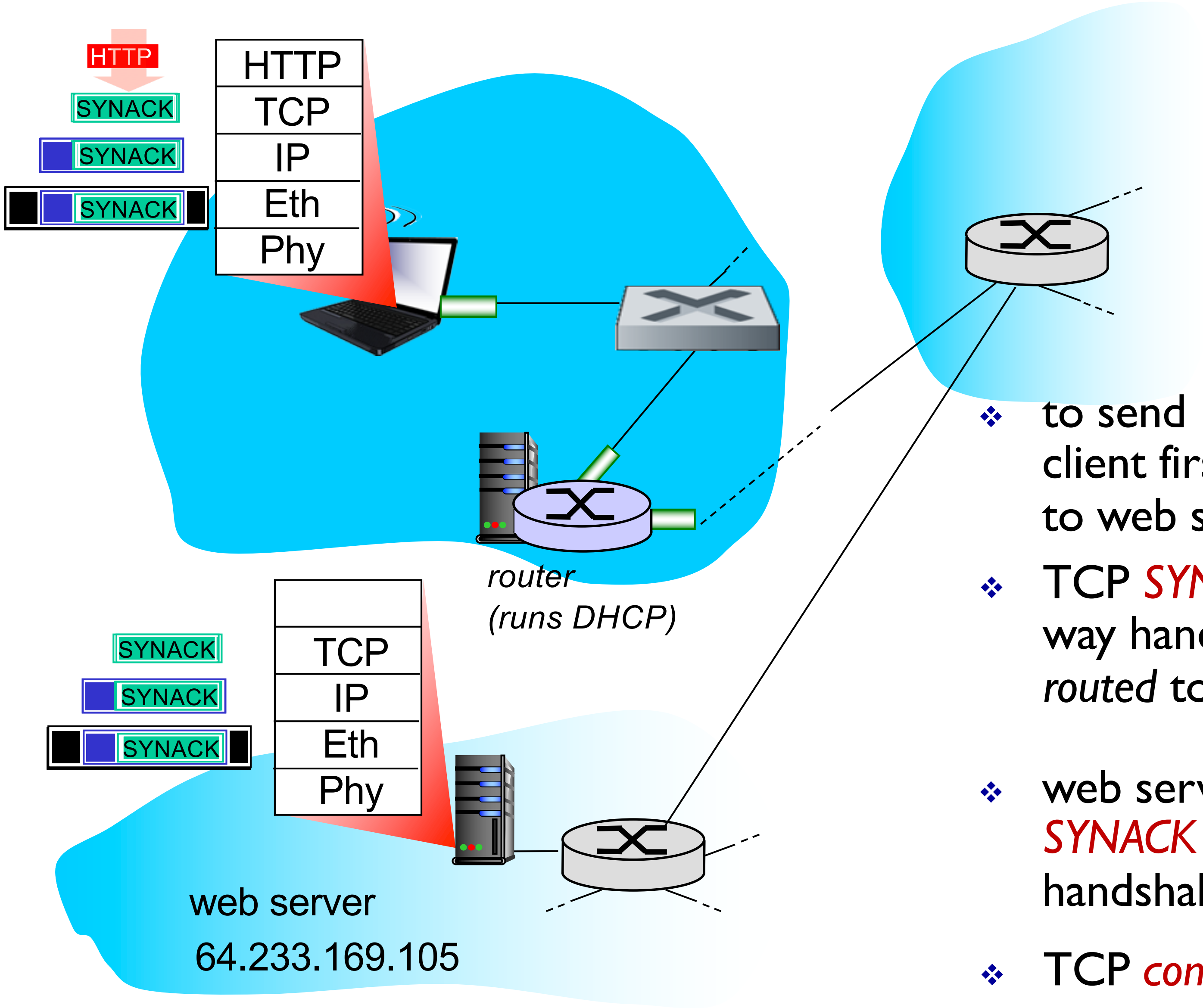


- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router

Client now has IP address of www.google.lv

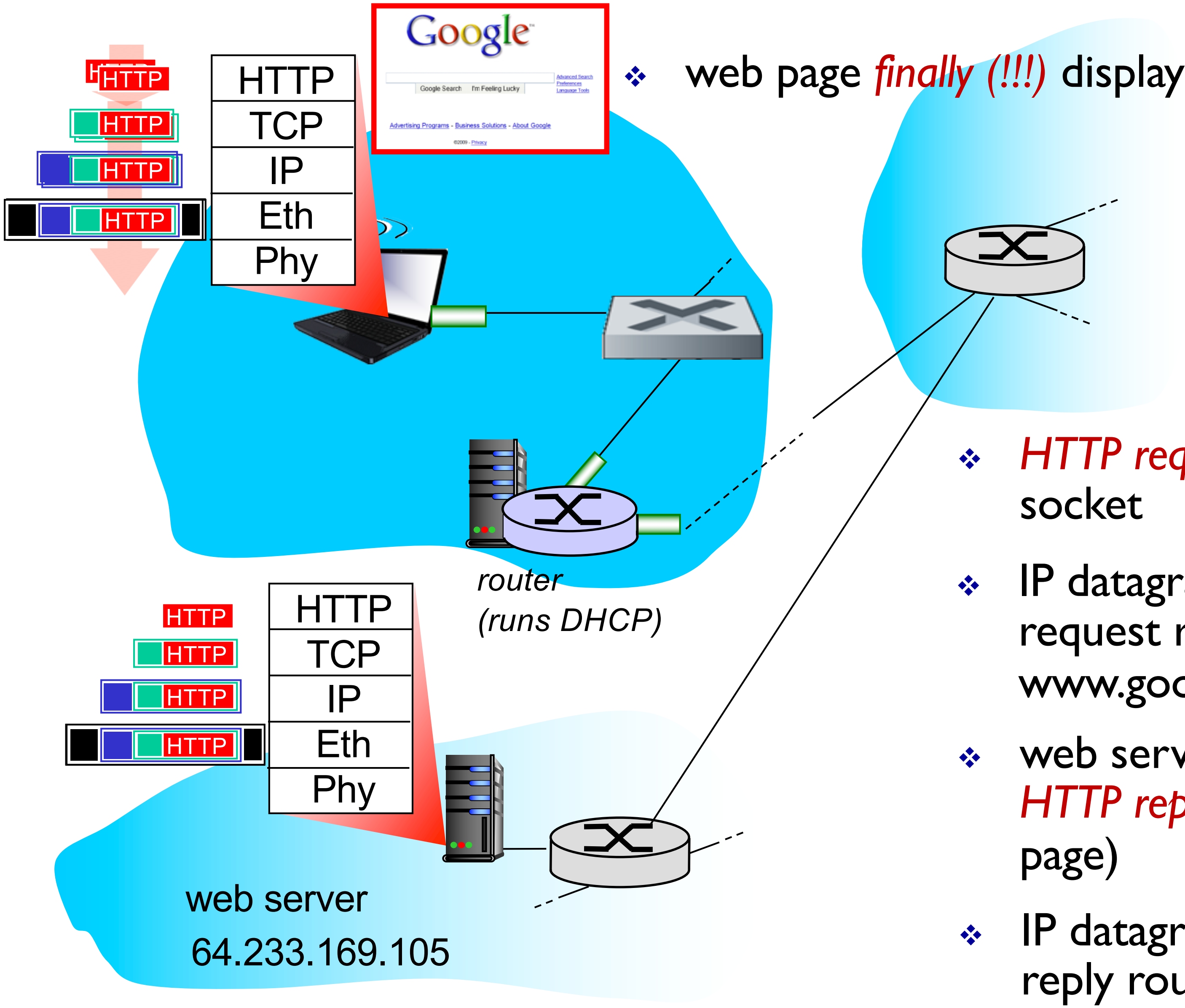
- IP datagram forwarded from campus network into ISP network, routed (tables created by *RIP, OSPF, IS-IS* and/or *BGP* routing protocols) to DNS server
- demux'ed to DNS server
- DNS server replies to client with IP address of www.google.com

A day in the life...TCP connection carrying HTTP



- to send HTTP request, client first opens *TCP socket* to web server
- TCP *SYN segment* (step 1 in 3-way handshake) *inter-domain routed* to web server
- web server responds with *TCP SYNACK* (step 2 in 3-way handshake)
- TCP *connection established!*

A day in the life… HTTP request/reply



- web page *finally (!!!)* displayed

- *HTTP request* sent into TCP socket
- IP datagram containing HTTP request routed to www.google.com
- web server responds with *HTTP reply* (containing web page)
- IP datagram containing HTTP reply routed back to client