



Computer Networks

CS3953

Link Layer-Part 2

Haiming Jin

The slides are adapted from those provided by Prof. J.F Kurose and K.W. Ross.

Link layer, LANs: outline

6.1 introduction, services

6.2 error detection,
correction

6.3 multiple access
protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANs

6.5 link virtualization:
MPLS

6.6 data center
networking

6.7 a day in the life of a
web request

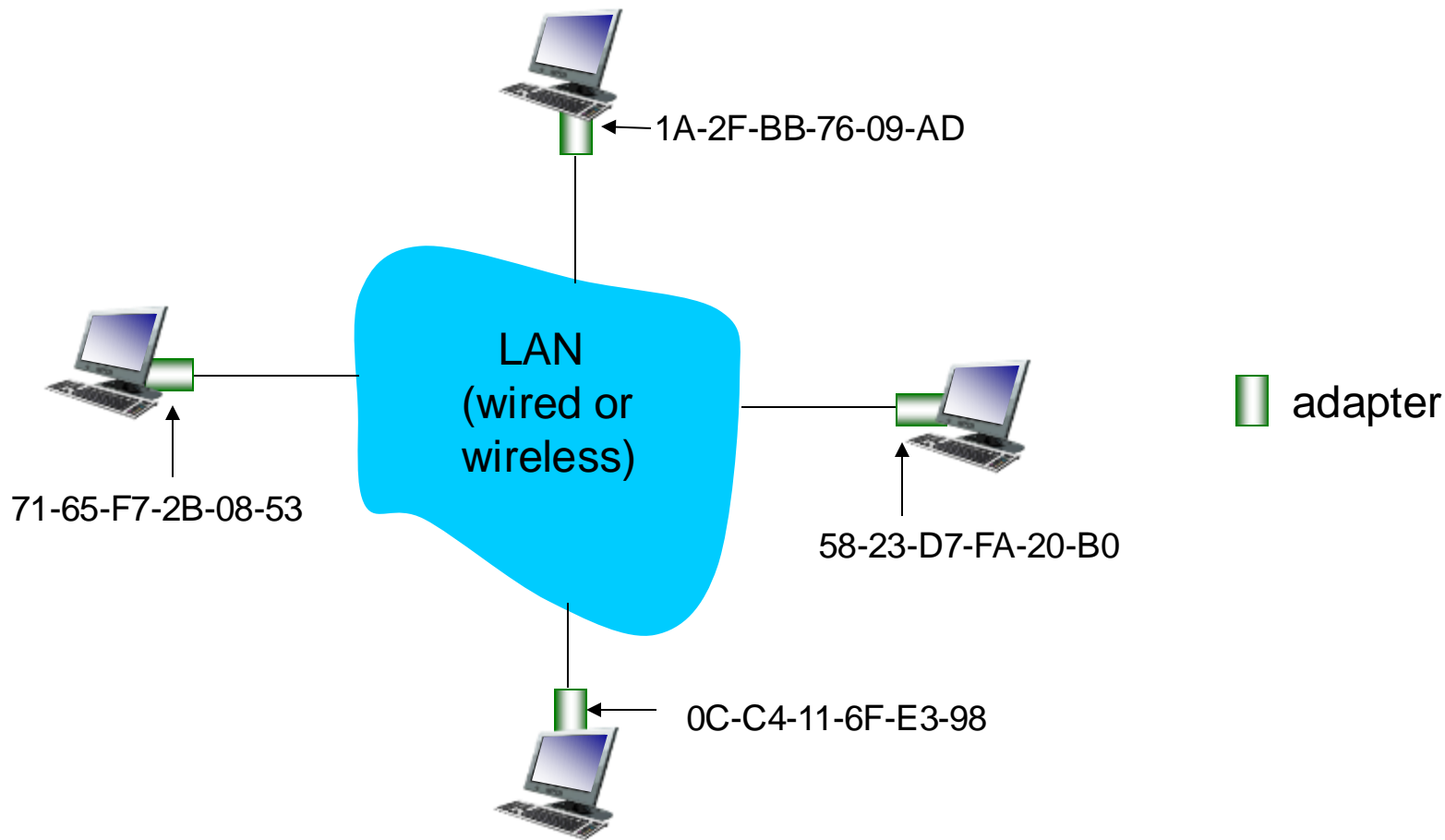
MAC addresses and ARP

- 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 “numeral” represents 4 bits)

LAN addresses and ARP

each adapter on LAN has unique *LAN* address

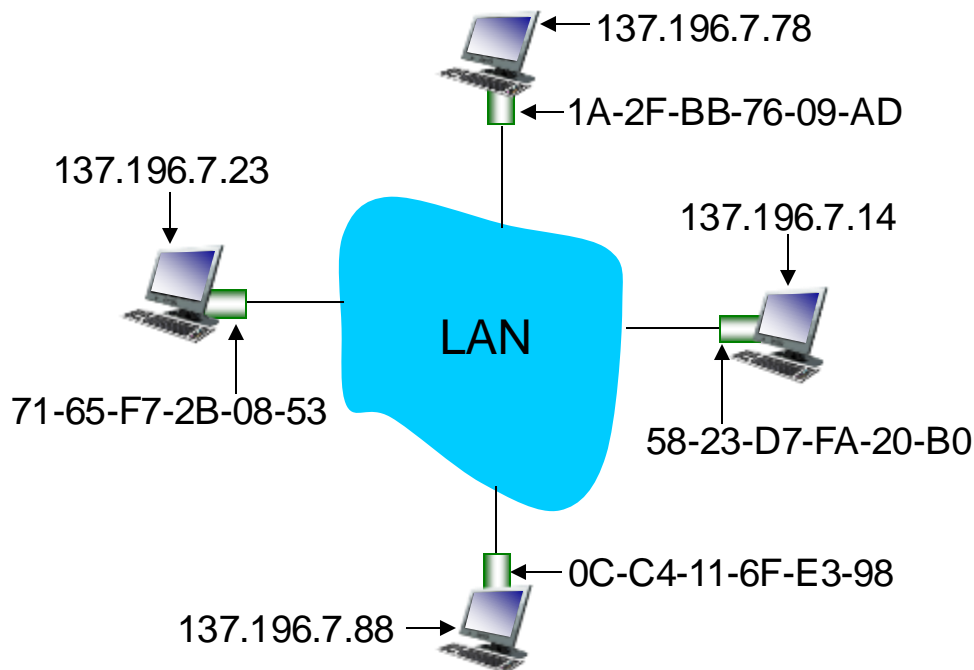


LAN addresses (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
 - MAC address: like Social Security Number (身份证号)
 - IP address: like postal address (家庭住址)
- 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: 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)

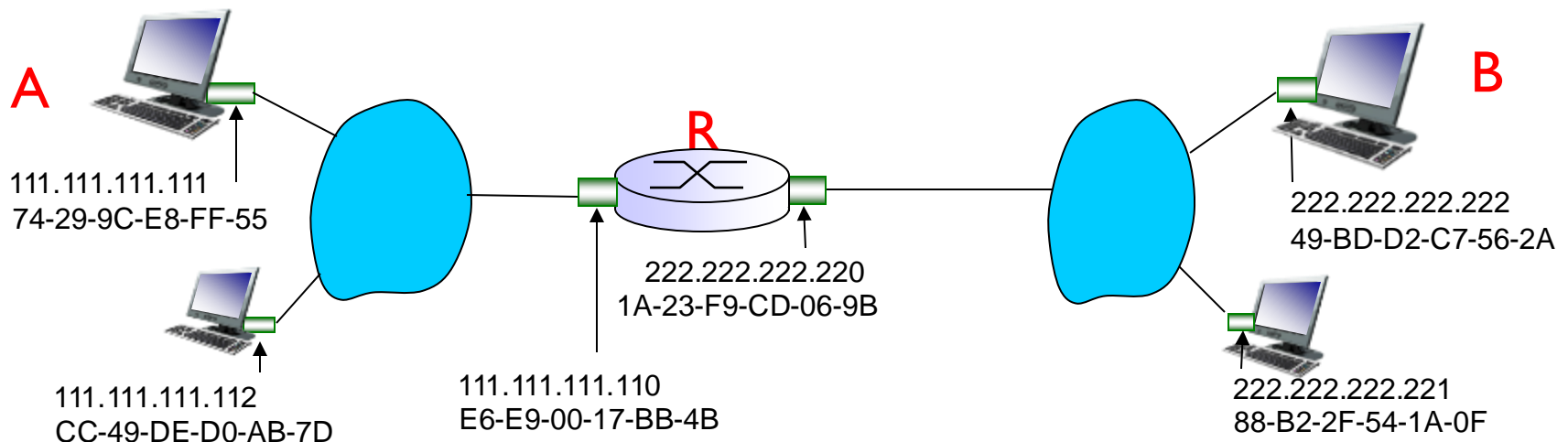
ARP protocol: same LAN

- 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
 - destination MAC address = FF-FF-FF-FF-FF-FF
 - all nodes on LAN receive ARP query
- 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 intervention from net administrator*

Addressing: routing to another LAN

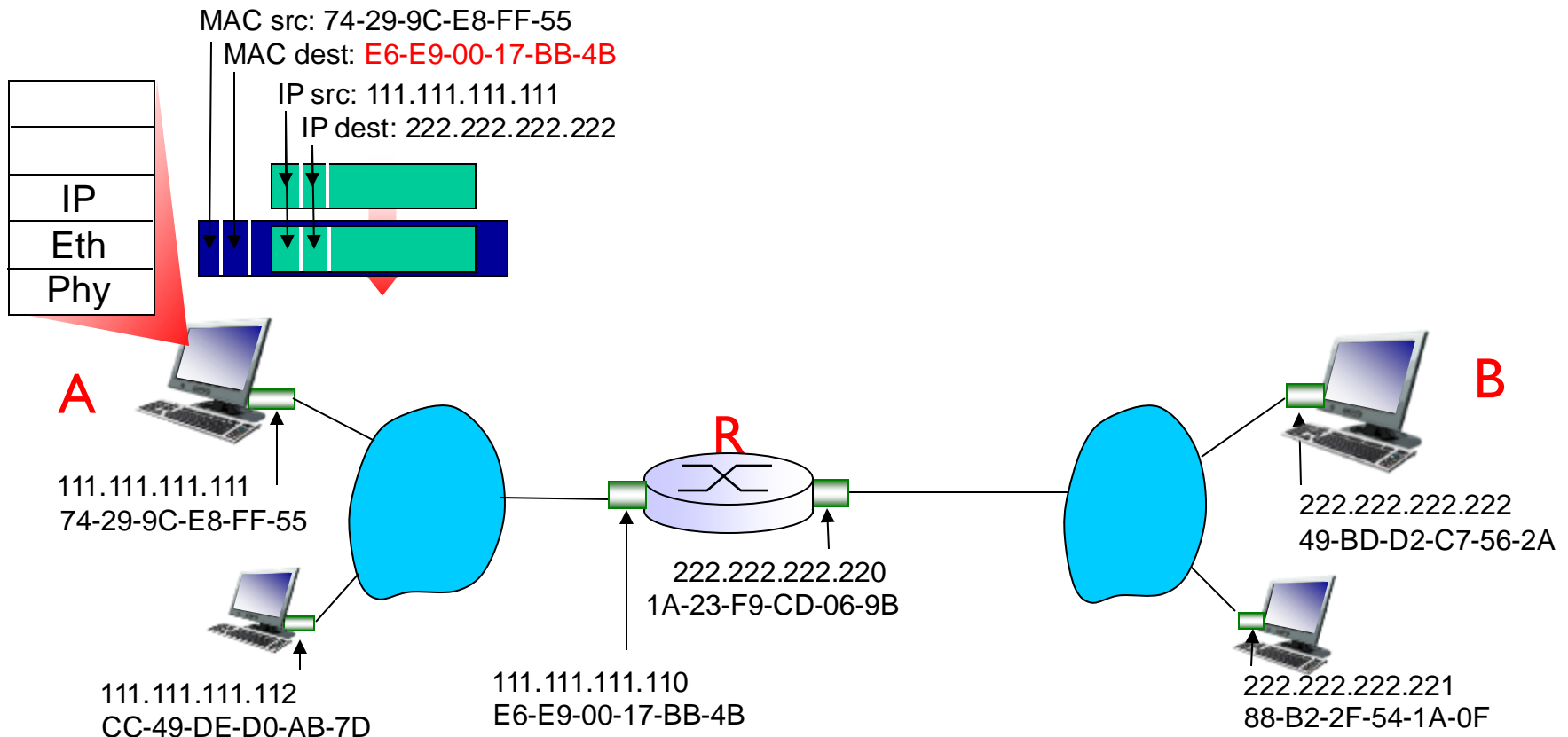
walkthrough: **send datagram from A to B via R**

- 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?)



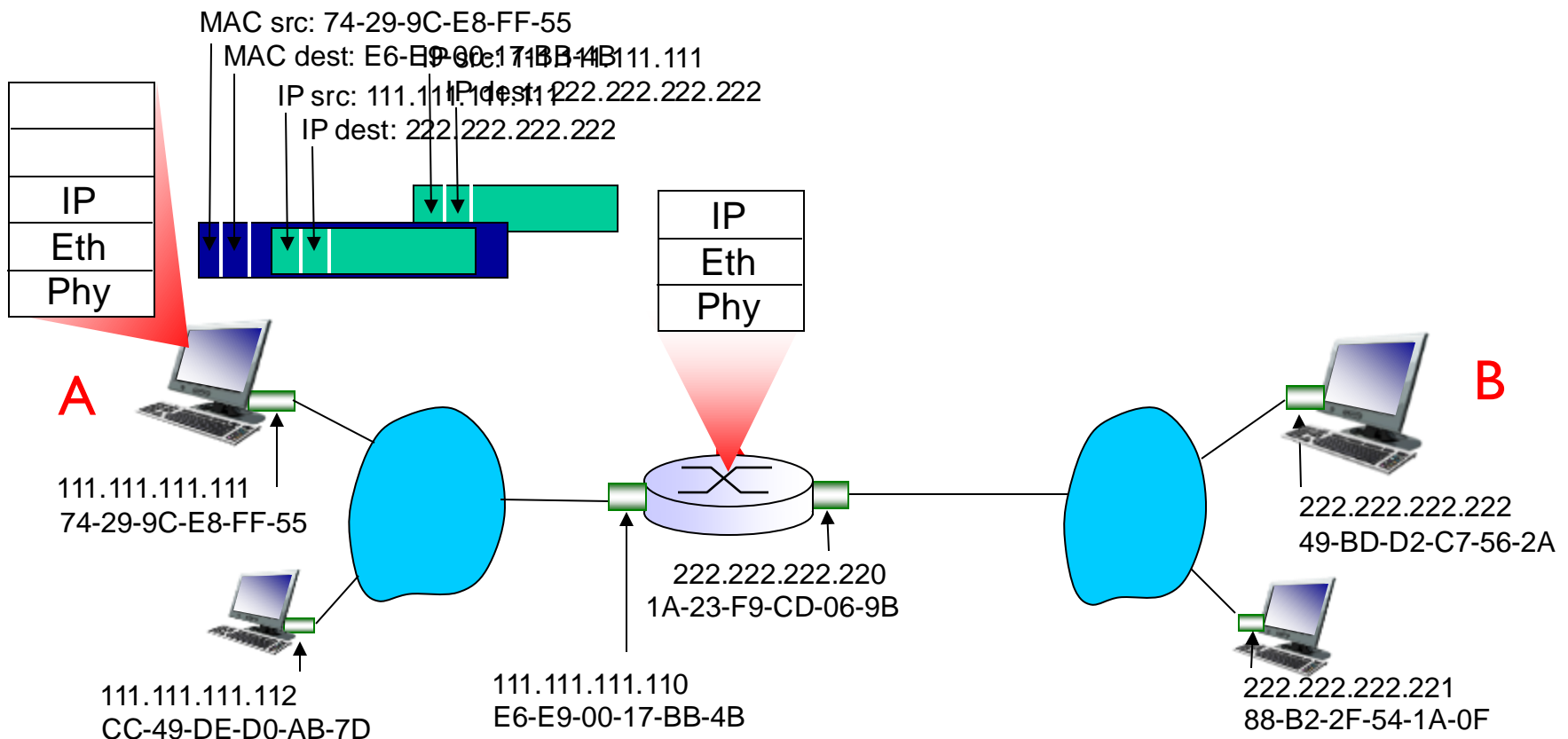
Addressing: routing to another LAN

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



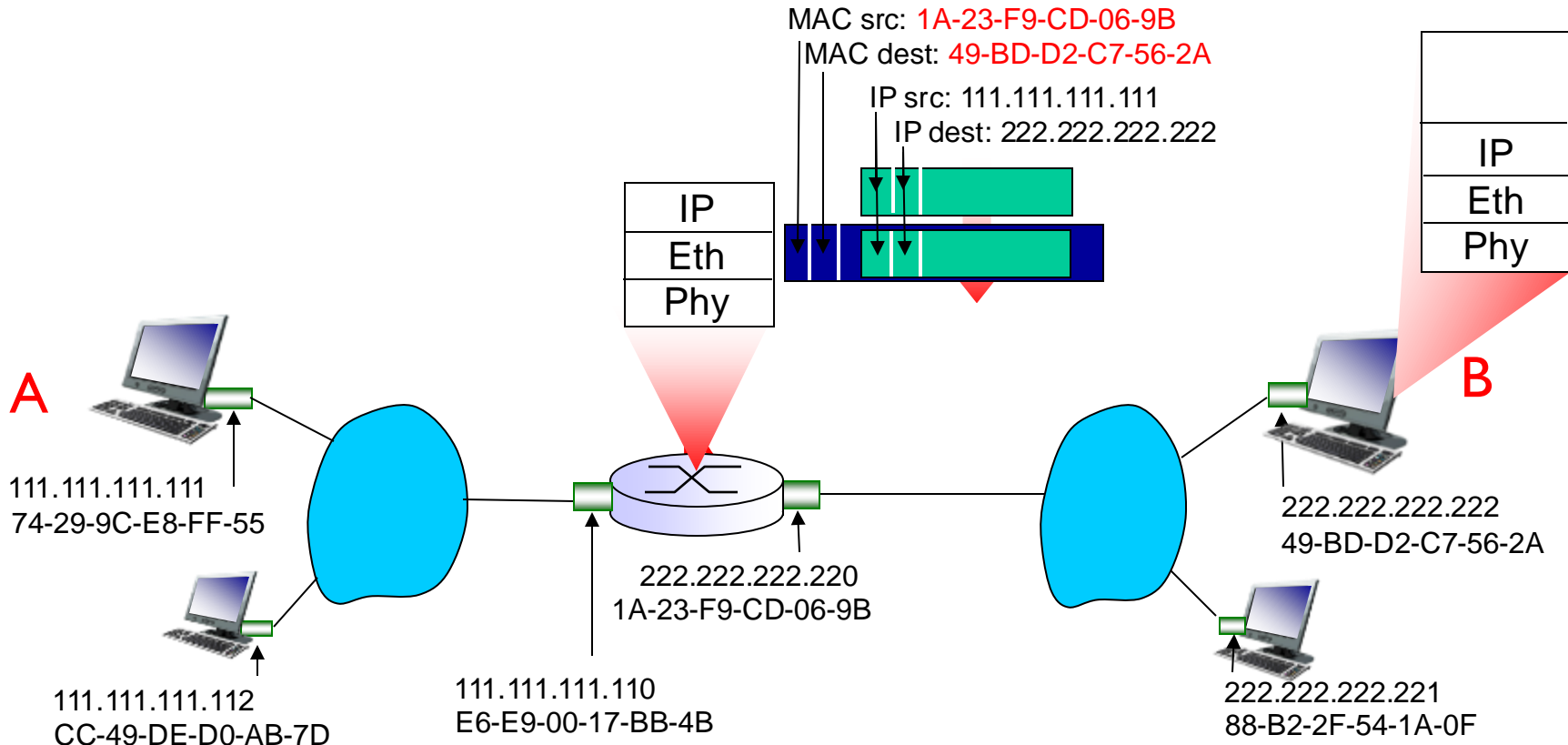
Addressing: routing to another LAN

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



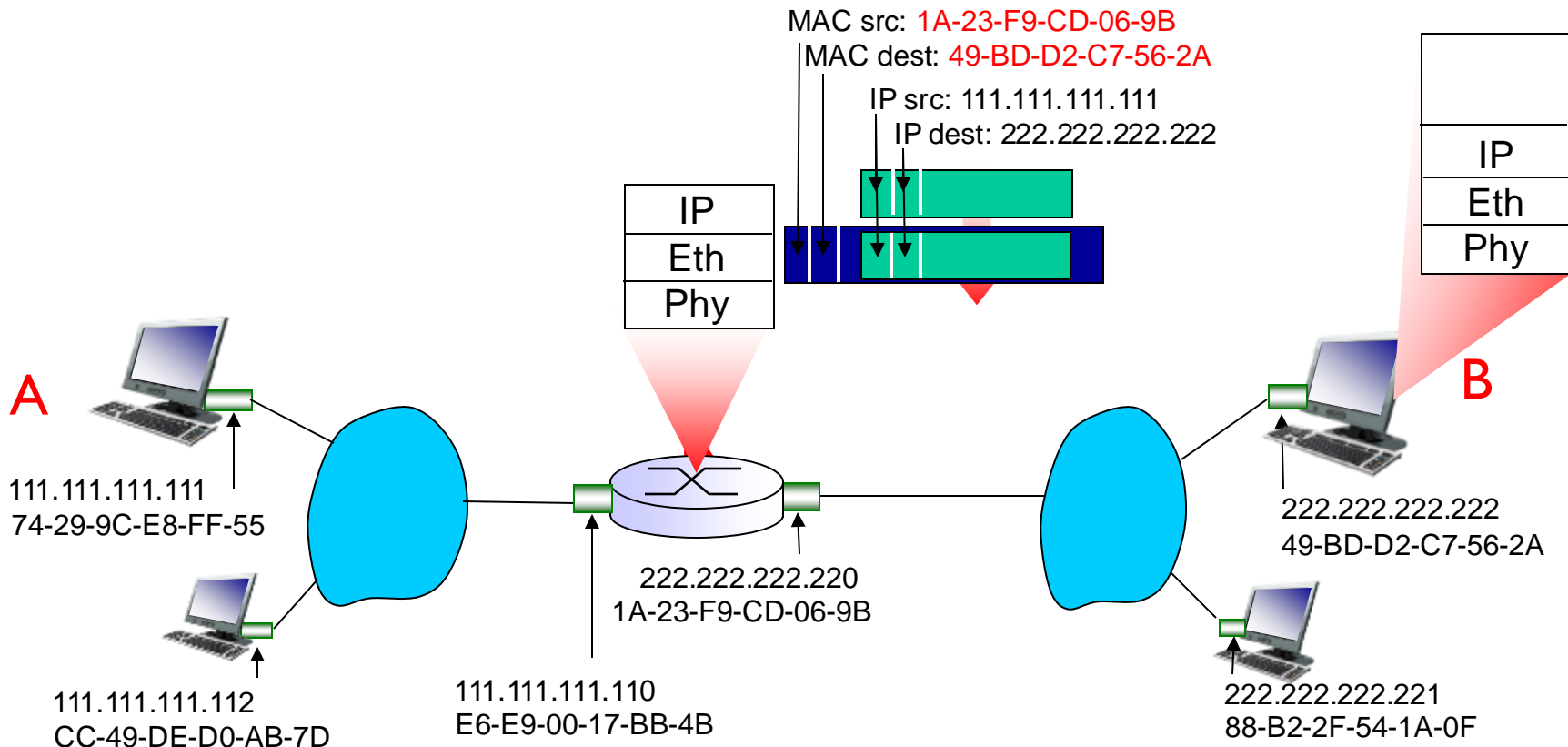
Addressing: routing to another LAN

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



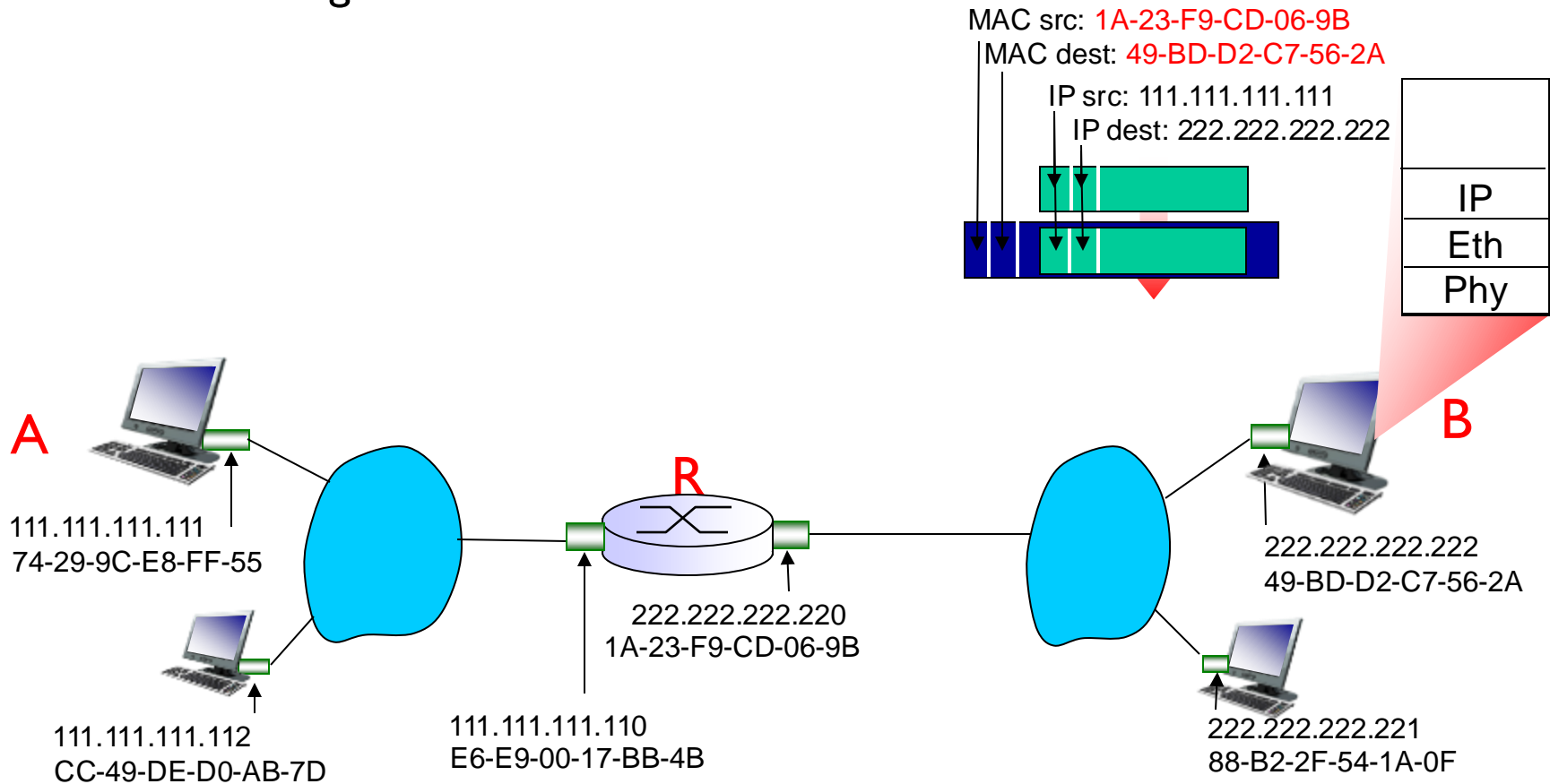
Addressing: routing to another LAN

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



* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Link layer, LANs: outline

6.1 introduction, services

6.2 error detection,
correction

6.3 multiple access
protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANs

6.5 link virtualization:
MPLS

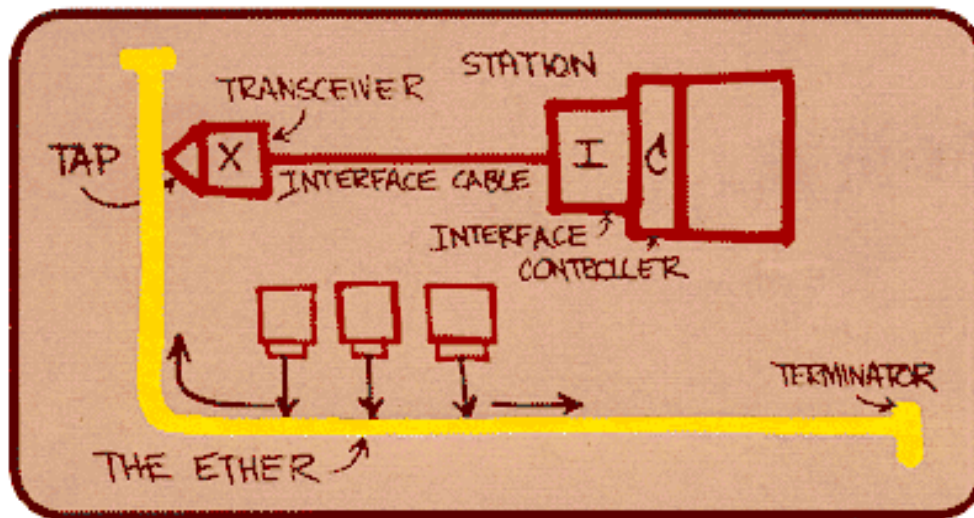
6.6 data center
networking

6.7 a day in the life of a
web request

Ethernet

“dominant” wired LAN technology:

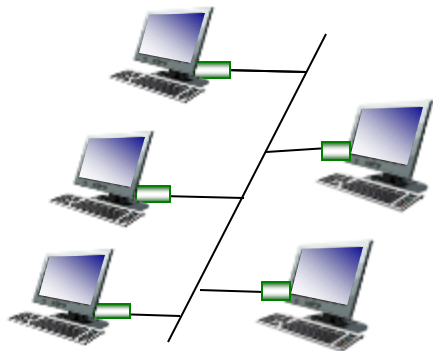
- first widely used LAN technology
- simpler, cheap
- kept up with speed race: 10 Mbps – 10 Gbps



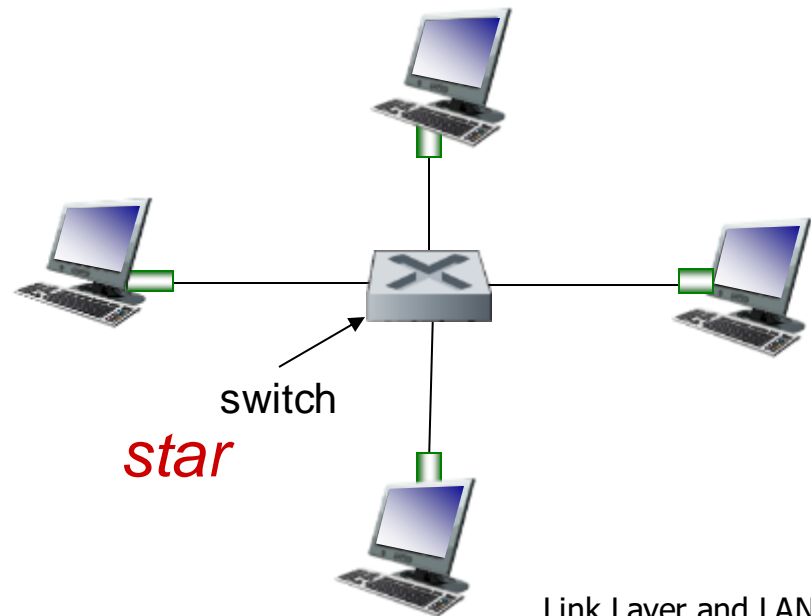
Metcalfe's Ethernet sketch

Ethernet: physical topology

- **bus:** popular through mid 90s
 - all nodes in same collision domain (can collide with each other)
- **star:** prevails today
 - active **switch** in center
 - each “spoke” runs a (separate) Ethernet protocol (nodes do not collide with each other)



bus: coaxial cable



Ethernet frame structure

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



preamble:

- 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
- used to synchronize receiver, sender clock rates

Ethernet frame structure (more)

- **addresses:** 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
- **type:** indicates higher layer protocol (mostly IP)
- **CRC:** cyclic redundancy check at receiver
 - error detected: frame is dropped



Ethernet: unreliable, connectionless

- *connectionless*: no handshaking between sending and receiving NICs
- *unreliable*: receiving NIC doesn't send acks or nacks to sending NIC
 - data in dropped frames recovered only if initial sender uses higher layer rdt (e.g., TCP), otherwise dropped data lost
- Ethernet's MAC protocol: unslotted *CSMA/CD with binary backoff*

Link layer, LANs: outline

6.1 introduction, services

6.2 error detection,
correction

6.3 multiple access
protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANs

6.5 link virtualization:
MPLS

6.6 data center
networking

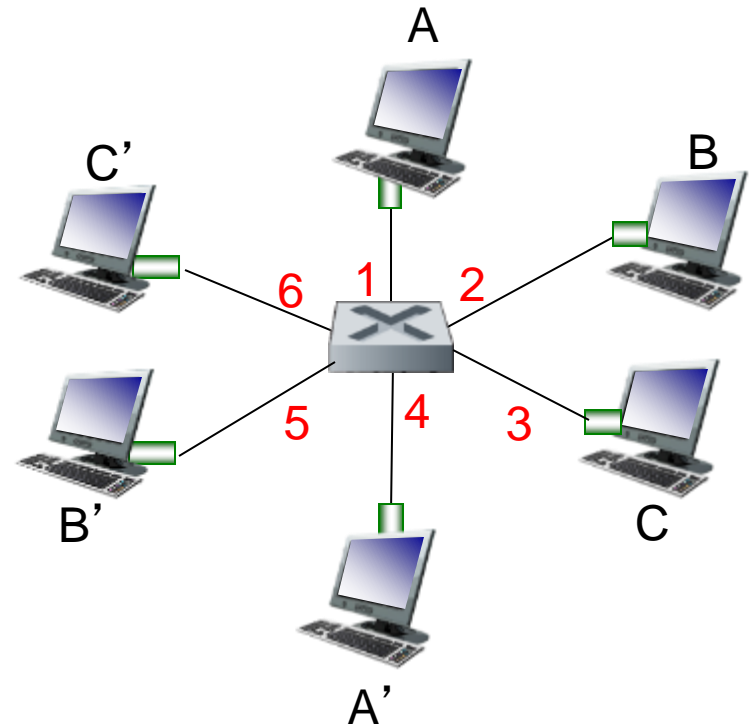
6.7 a day in the life of a
web request

Ethernet switch

- **link-layer device: takes an *active* role**
 - store, forward Ethernet frames
 - examine incoming frame's MAC address, **selectively** forward frame to one-or-more outgoing links when frame is to be forwarded on link, uses CSMA/CD to access link
- ***transparent***
 - hosts are unaware of presence of switches
- ***plug-and-play, self-learning***
 - switches do not need to be configured

Switch: *multiple* simultaneous transmissions

- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, but no collisions; 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)*

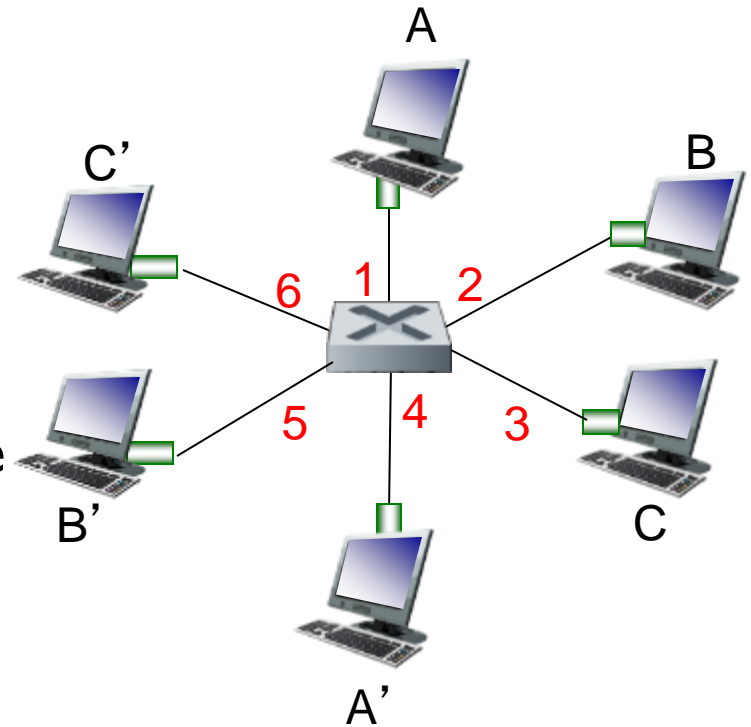
Switch forwarding table

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?

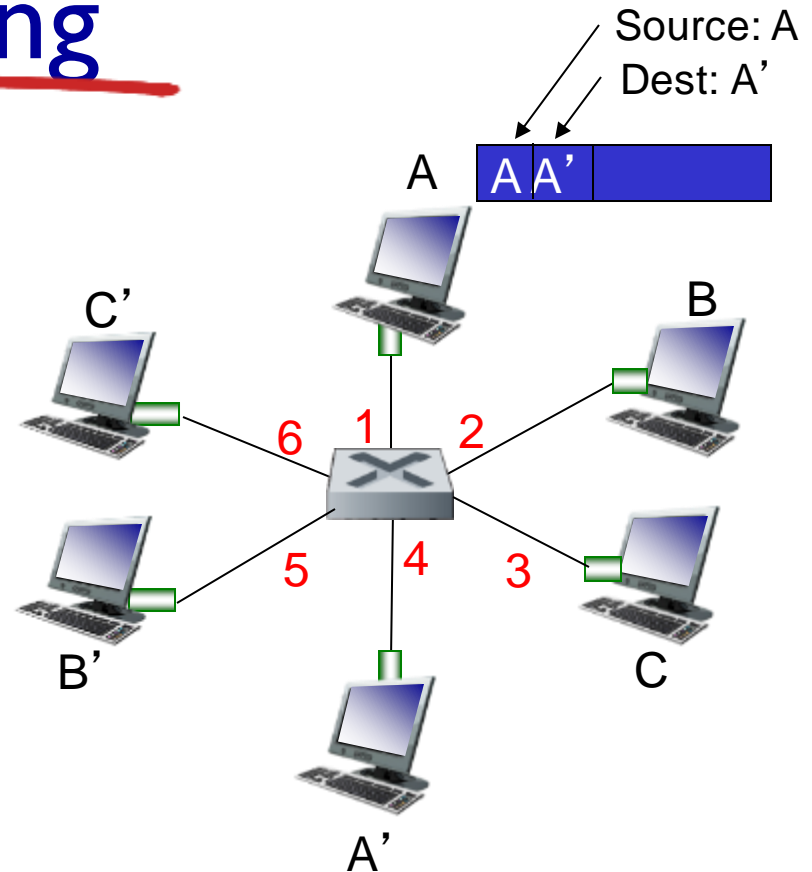
- something like a routing protocol?



*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

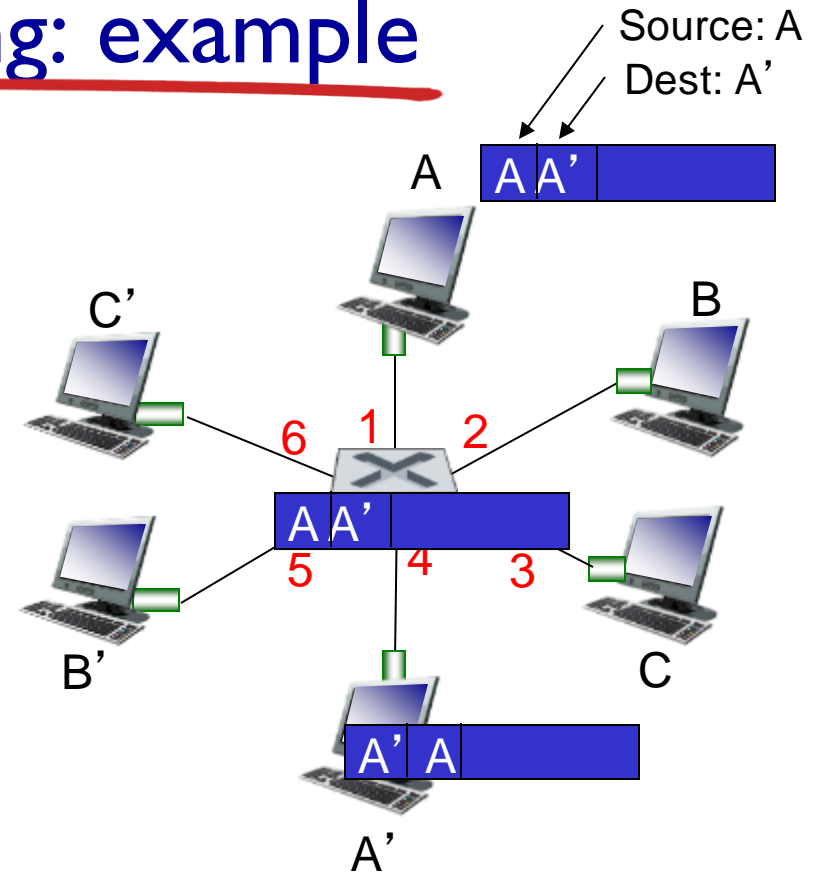
3. if entry found for destination
then {

 else forward frame on interface indicated by
entry
}

else flood /* forward on all interfaces except arriving
interface */

Self-learning, forwarding: example

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

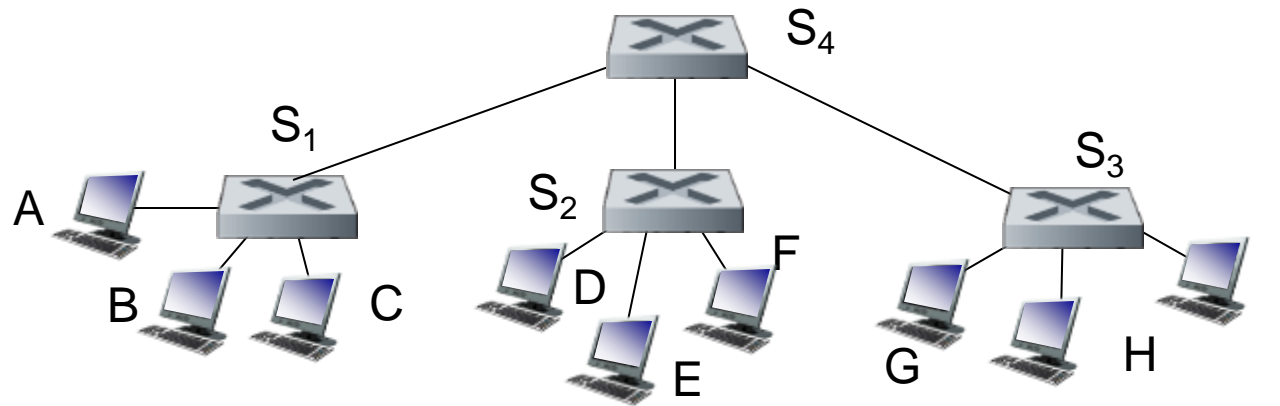


MAC addr	interface	TTL
A	1	60
A'	4	60

*switch table
(initially empty)*

Interconnecting switches

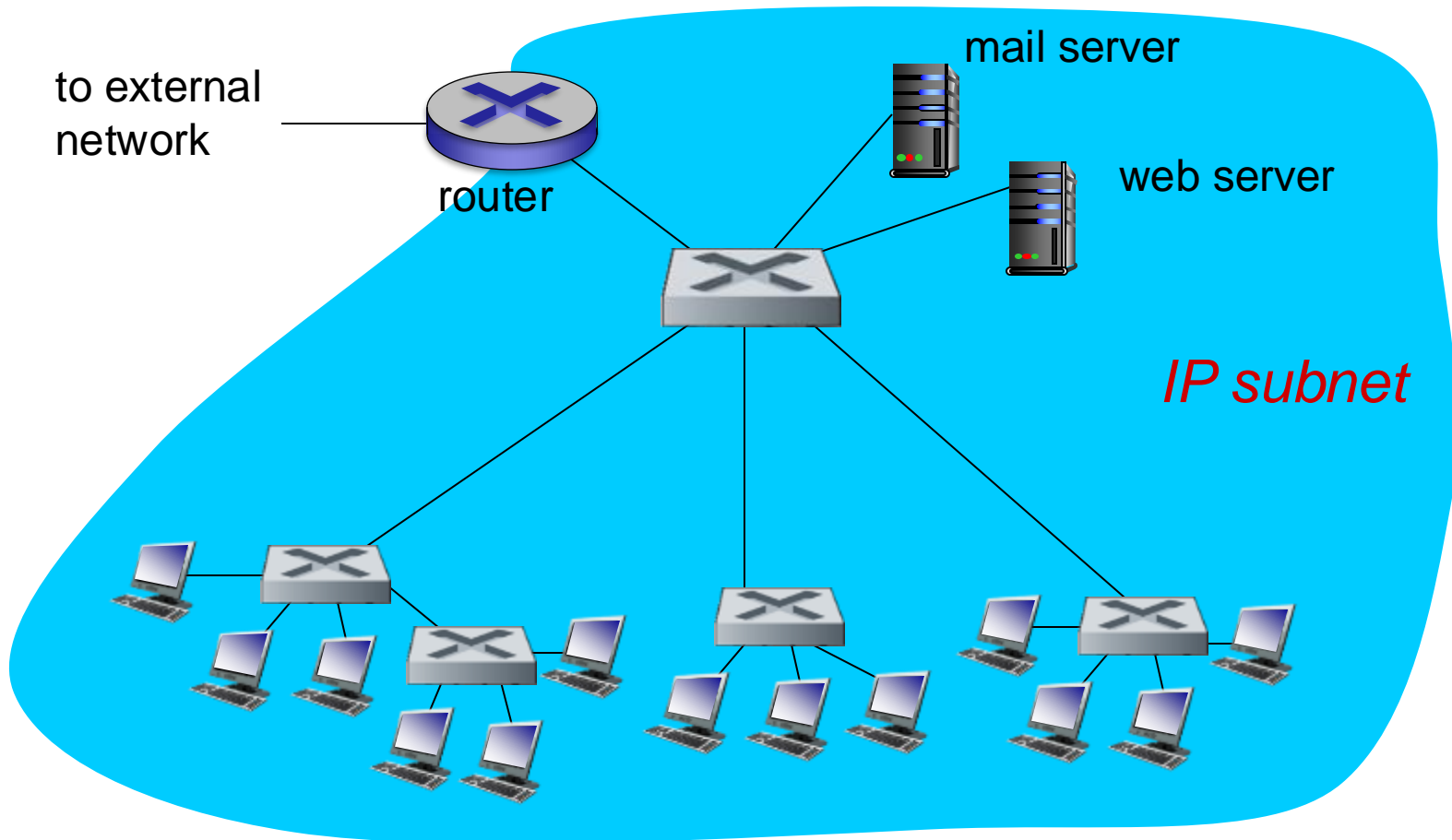
self-learning switches can be connected together:



Q: sending from A to G - how does S₁ know to forward frame destined to G via S₄ and S₃?

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

Institutional network



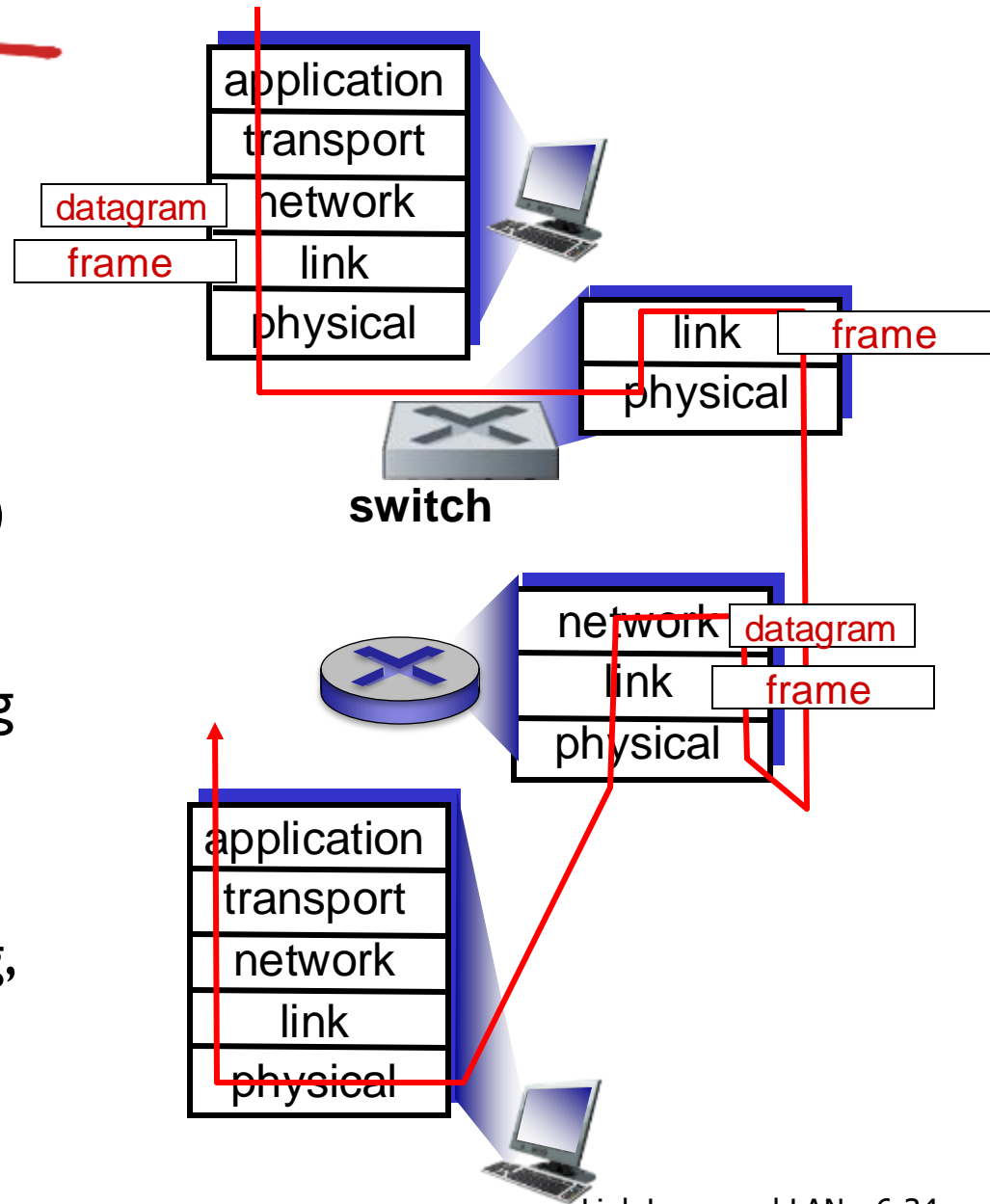
Switches vs. routers

both are store-and-forward:

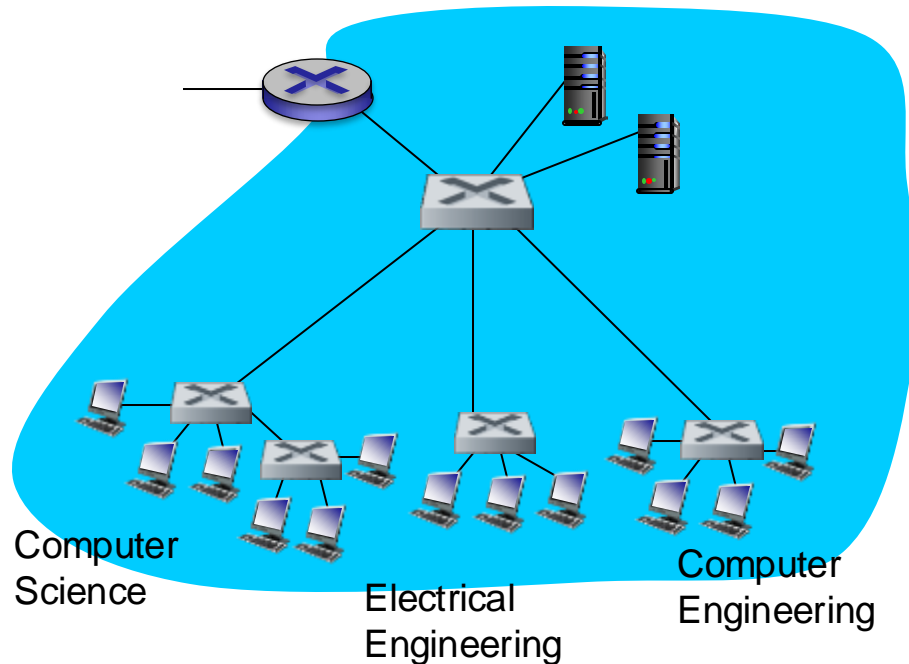
- **routers:** network-layer devices (examine network-layer headers)
- **switches:** link-layer devices (examine link-layer headers)

both have forwarding tables:

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



VLANs: motivation



consider:

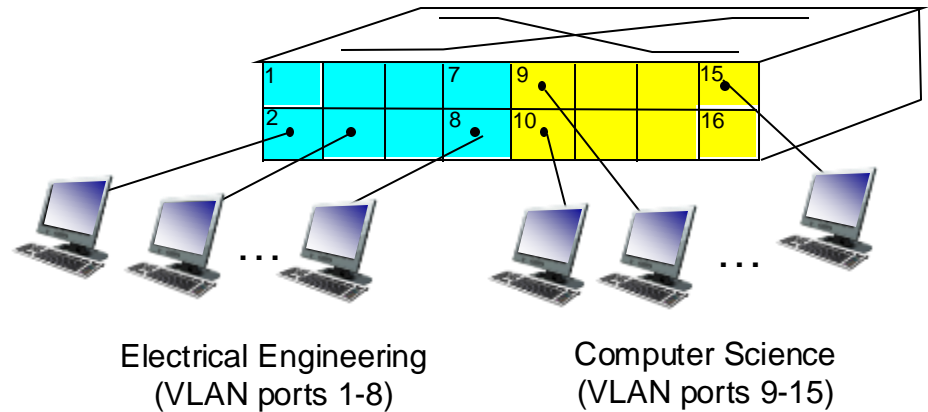
- CS user moves office to EE, but wants connect to CS switch?
- single broadcast domain:
 - all layer-2 broadcast traffic (ARP, DHCP, unknown location of destination MAC address) must cross entire LAN
 - security/privacy, efficiency issues

VLANs

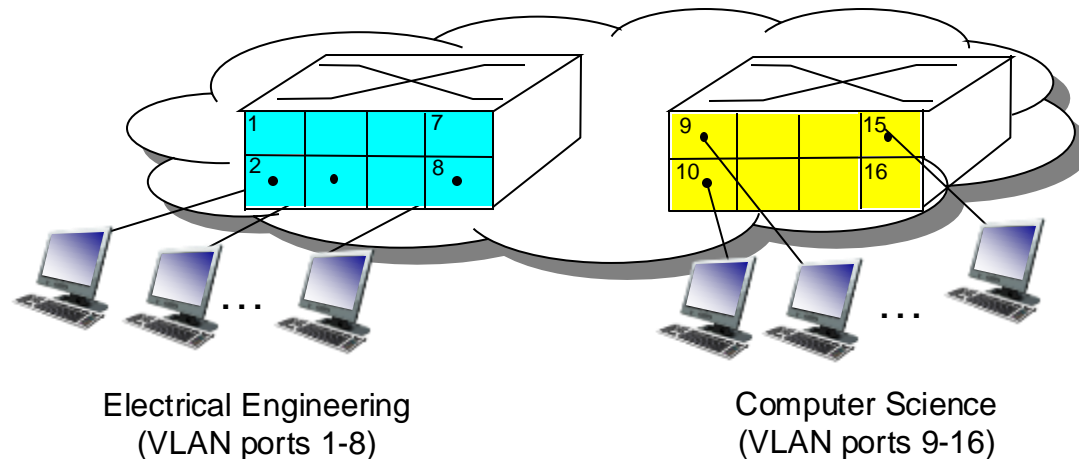
Virtual Local Area Network

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

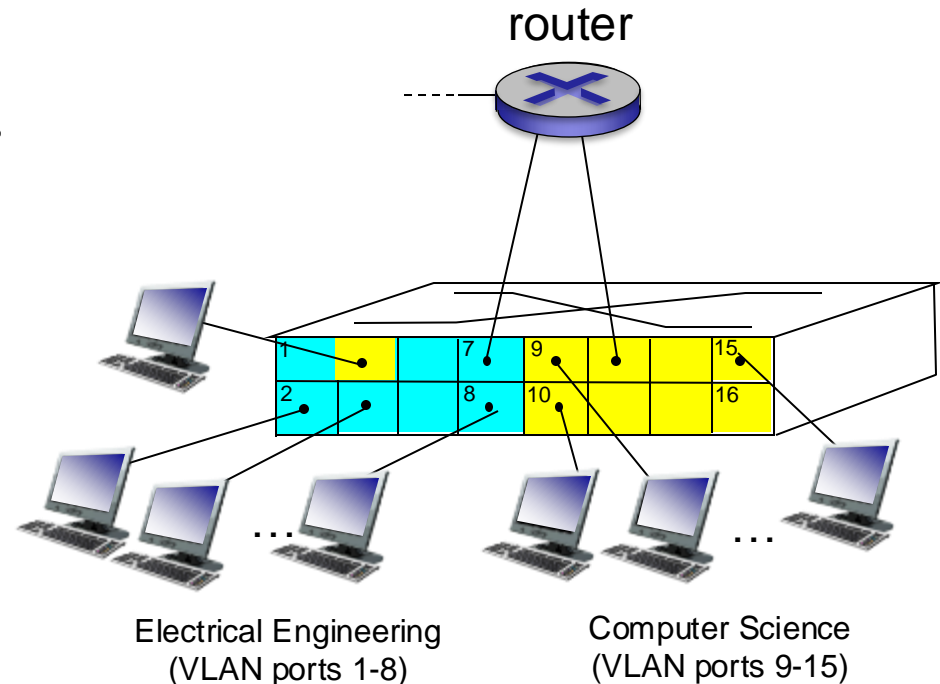


... operates as *multiple* virtual switches

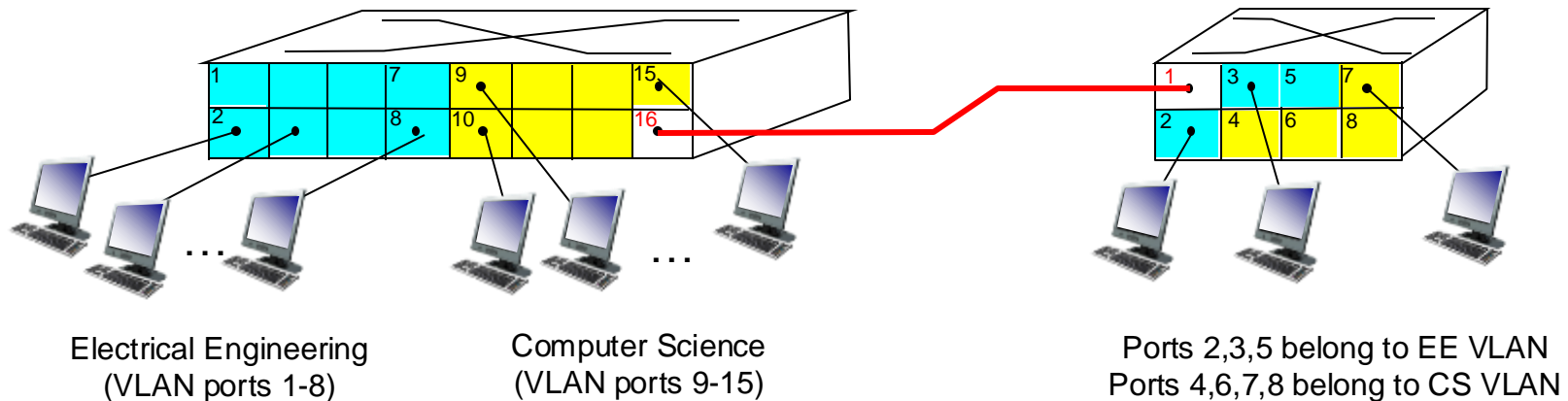


Port-based VLAN

- **traffic isolation:** frames to/from ports 1-8 can *only* reach ports 1-8
- **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

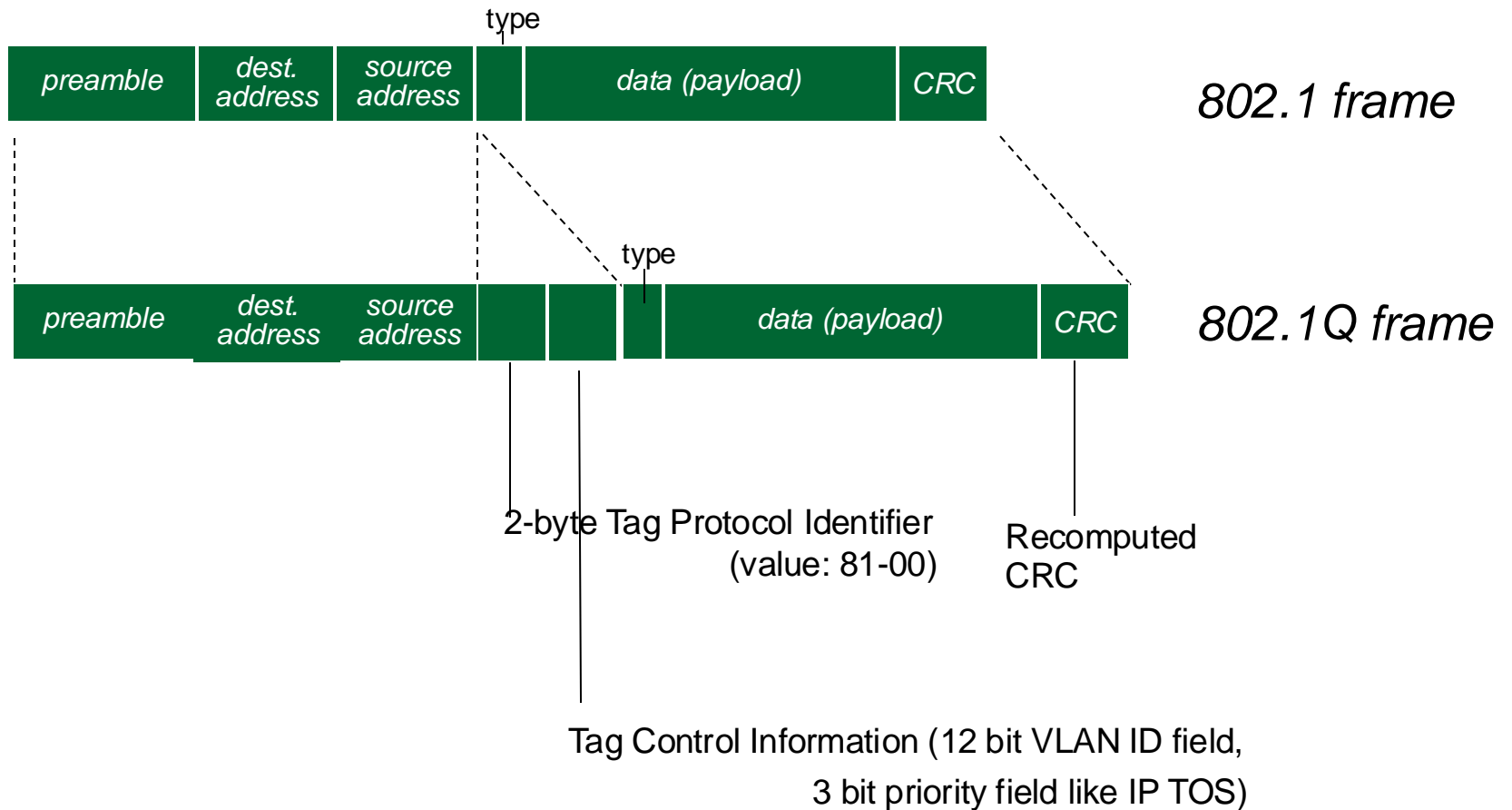


VLANs spanning multiple switches



- **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

802.1Q VLAN frame format



Link layer, LANs: outline

6.1 introduction, services

6.2 error detection,
correction

6.3 multiple access
protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANs

6.5 link virtualization:
MPLS

6.6 data center
networking

6.7 a day in the life of a
web request

Link layer, LANs: outline

6.1 introduction, services

6.2 error detection,
correction

6.3 multiple access
protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANs

6.5 link virtualization:
MPLS

6.6 data center
networking

6.7 a day in the life of a
web request

Data center networks

- 10's to 100's of thousands of hosts, often closely coupled, in close proximity:
 - e-business (e.g. Amazon)
 - content-servers (e.g., YouTube, Akamai, Apple, Microsoft)
 - search engines, data mining (e.g., Google)
- challenges:
 - multiple applications, each serving massive numbers of clients
 - managing/balancing load, avoiding processing, networking, data bottlenecks

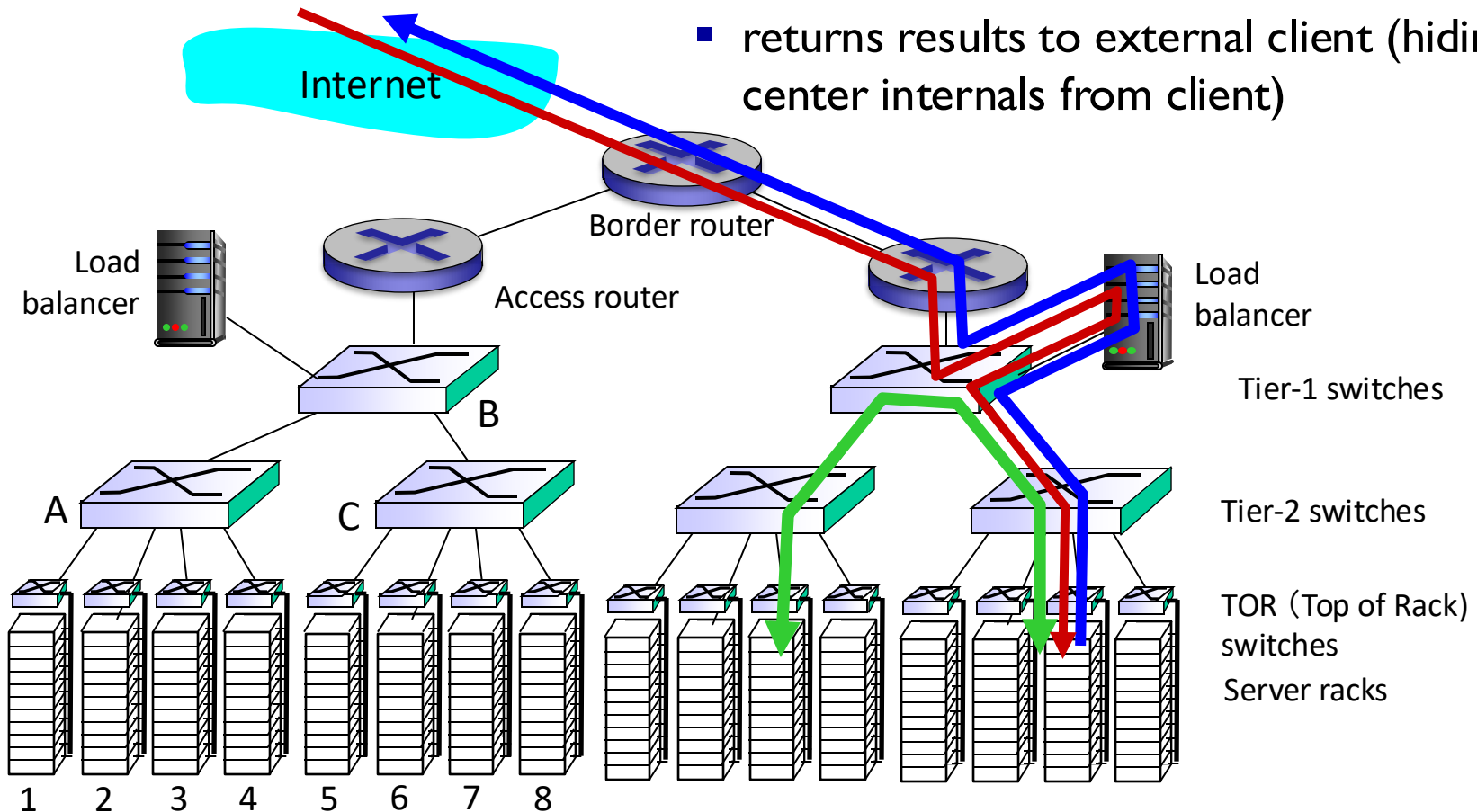


Inside a 40-ft Microsoft container,
Chicago data center

Data center networks

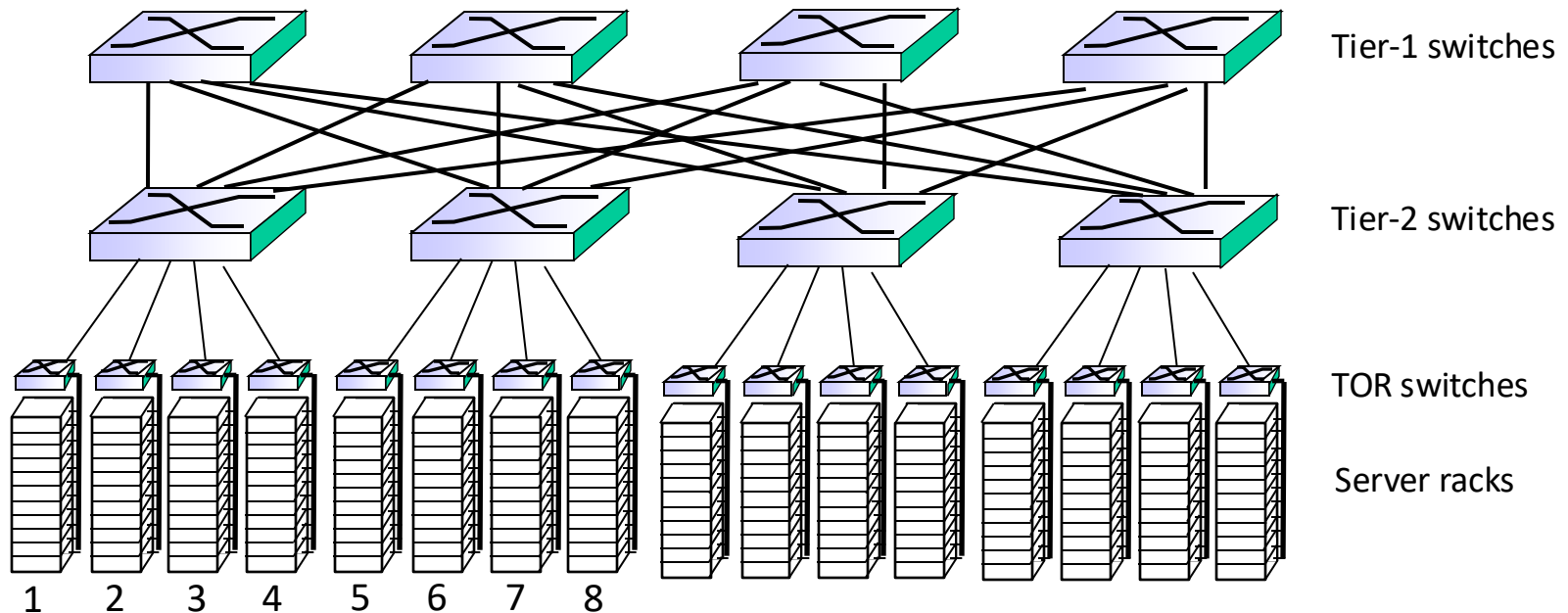
load balancer: application-layer routing

- receives external client requests
- directs workload within data center
- returns results to external client (hiding data center internals from client)



Data center networks

- rich interconnection among switches, racks:
 - increased throughput between racks (multiple routing paths possible)
 - increased reliability via redundancy



Link layer, LANs: outline

6.1 introduction, services

6.2 error detection,
correction

6.3 multiple access
protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANs

6.5 link virtualization:
MPLS

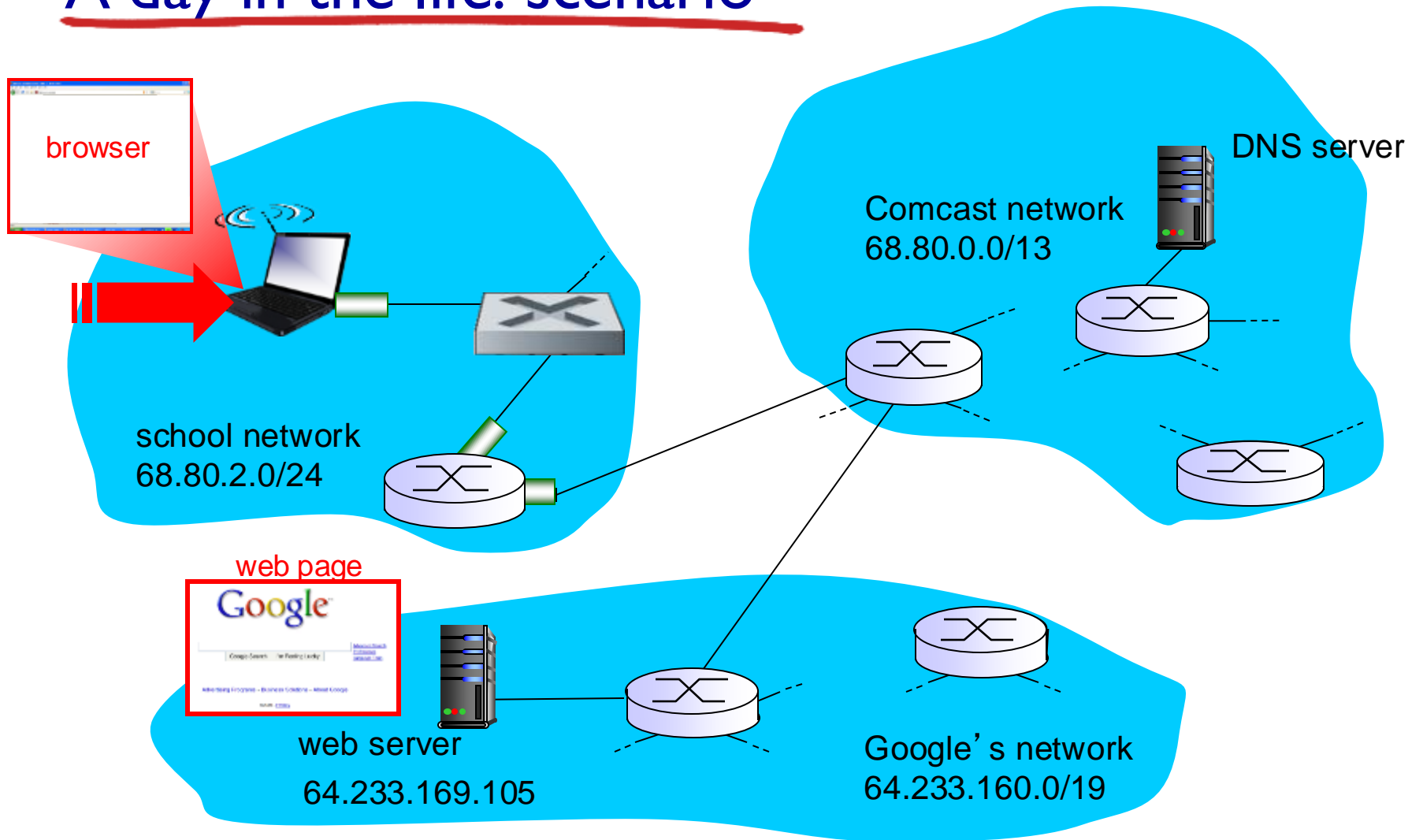
6.6 data center
networking

6.7 a day in the life of a
web request

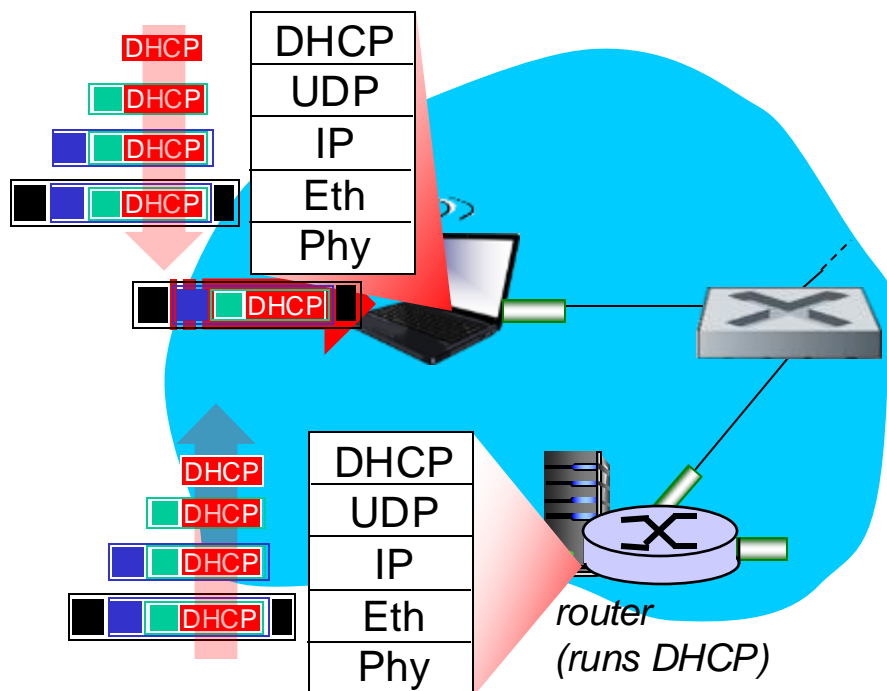
Synthesis: a day in the life of a web request

- journey down protocol stack complete!
 - application, transport, network, link
- putting-it-all-together: synthesis!
 - *goal*: identify, review, understand protocols (at all layers) involved in seemingly simple scenario: requesting www page
 - *scenario*: student attaches laptop to campus network, requests/receives `www.google.com`

A day in the life: scenario

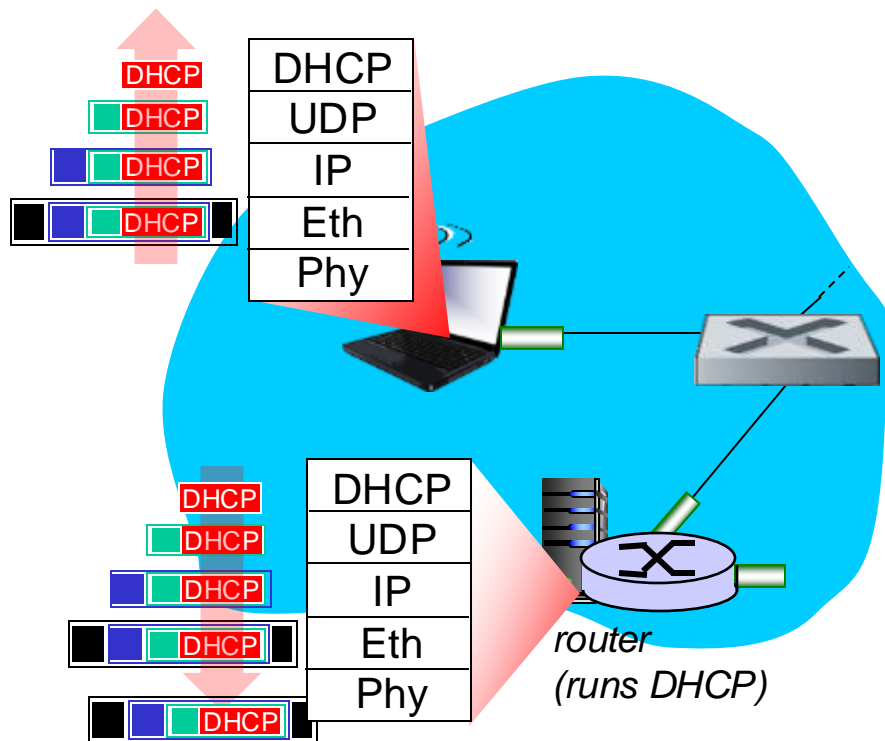


A day in the life... connecting to the Internet



- 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: FFFFFFFFFFFFFFFF) on LAN, received at router running **DHCP** server
- Ethernet **demuxed** to IP demuxed, UDP demuxed to DHCP

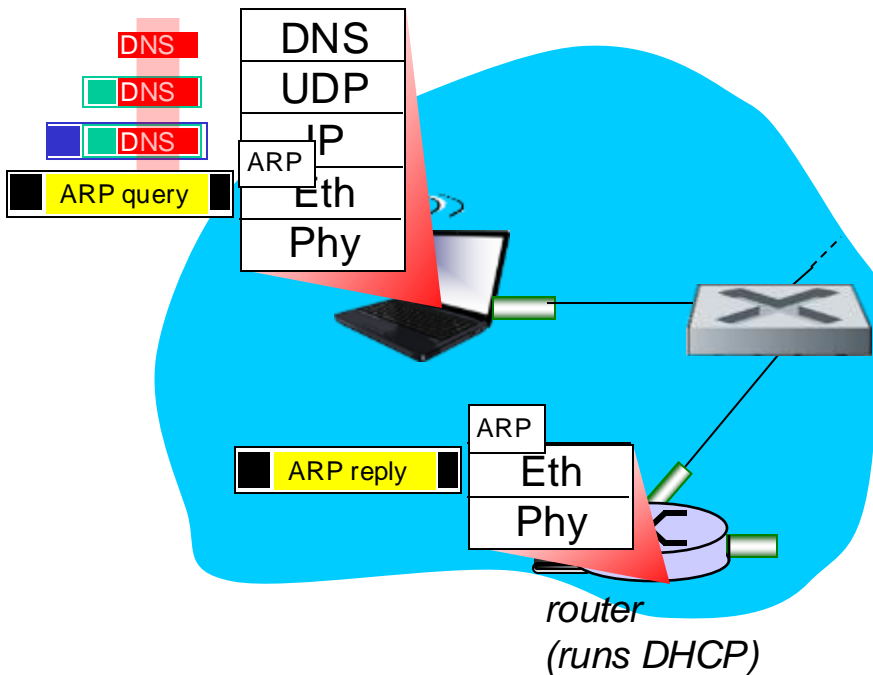
A day in the life... connecting to the Internet



- 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

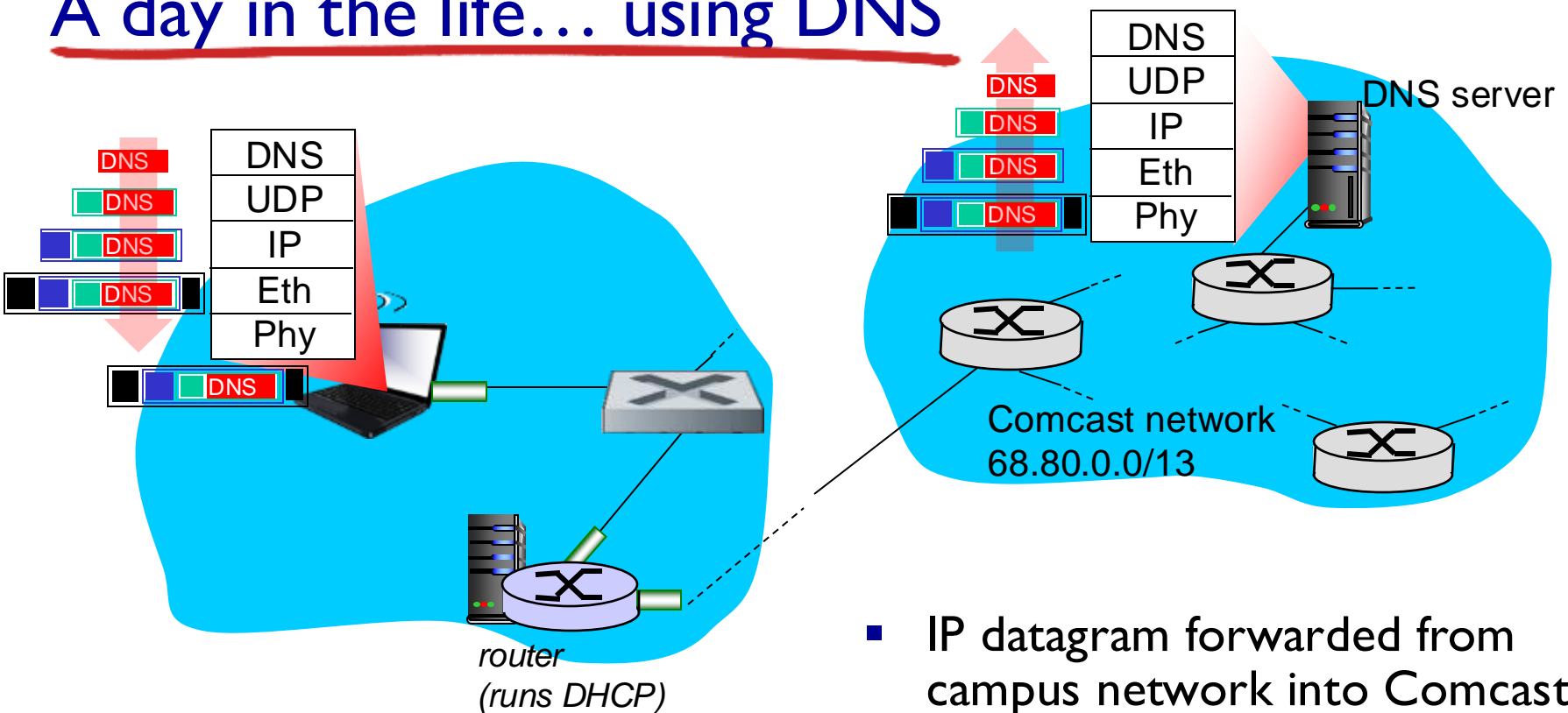
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

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

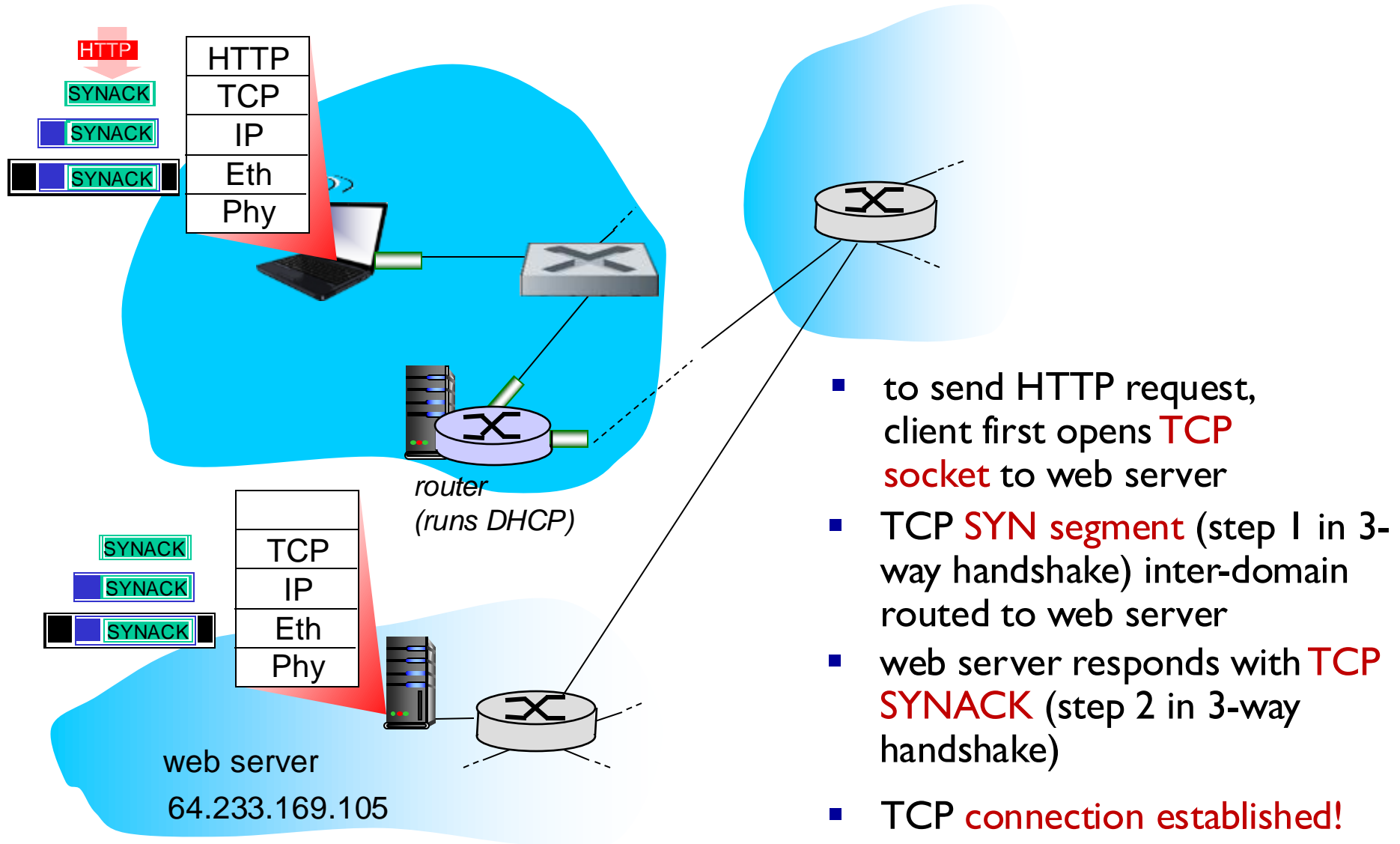
A day in the life... using DNS



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

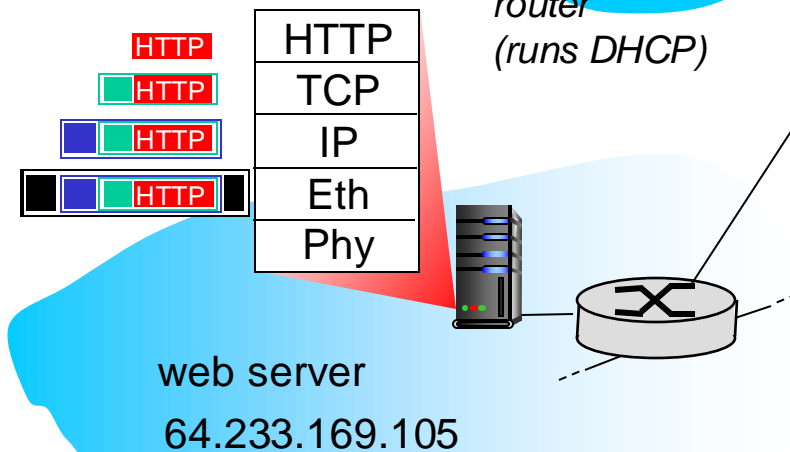
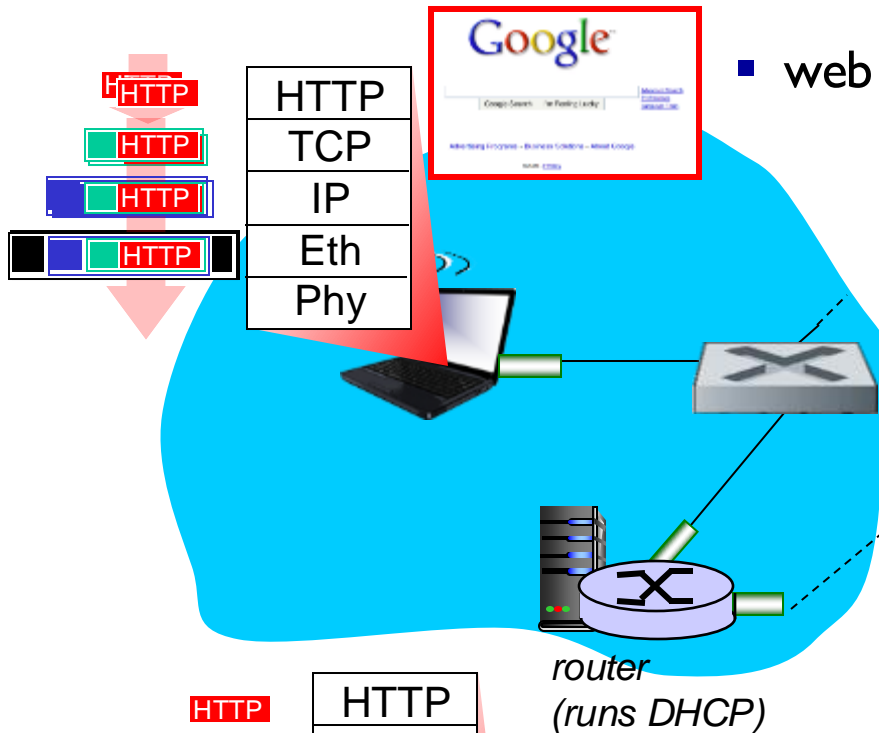
- IP datagram forwarded from campus network into Comcast network, routed (tables created by **RIP**, **OSPF**, **IS-IS** and/or **BGP** routing protocols) to DNS server
- demuxed to DNS server
- DNS server replies to client with IP address of **www.google.com**

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



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

Chapter 6: Summary

- principles behind data link layer services:
 - error detection, correction
 - sharing a broadcast channel: multiple access
 - link layer addressing
- instantiation and implementation of various link layer technologies
 - Ethernet
 - switched LANS, VLANs
 - virtualized networks as a link layer: MPLS
- synthesis: a day in the life of a web request

Chapter 6: let's take a breath

- journey down protocol stack *complete* (except PHY)
- solid understanding of networking principles, practice
- could stop here but *lots* of interesting topics!
 - wireless
 - multimedia
 - security