Università di Ferrara

Architettura di Reti

Chapter 6: Link Layer and LANs

Carlo Giannelli
carlo.giannelli@unife.it
http://www.unife.it/scienze/informatica/insegnamenti/architettura-reti/
http://docente.unife.it/carlo.giannelli

Computer Networking: A Top Down Approach
Jim Kurose, Keith Ross
Pearson, April 2016

Computer Networks
Andrew Tanenbaum
Prentice Hall

Slides adapted from "Computer Networking: A Top Down Approach", 7th Edition, Global Edition, Jim Kurose, Keith Ross, Pearson, April 2016

Chapter 6: Link layer and LANs

Our goals:

- understand principles behind link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - local area networks: Ethernet, VLANs
- instantiation, implementation of various link layer technologies

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

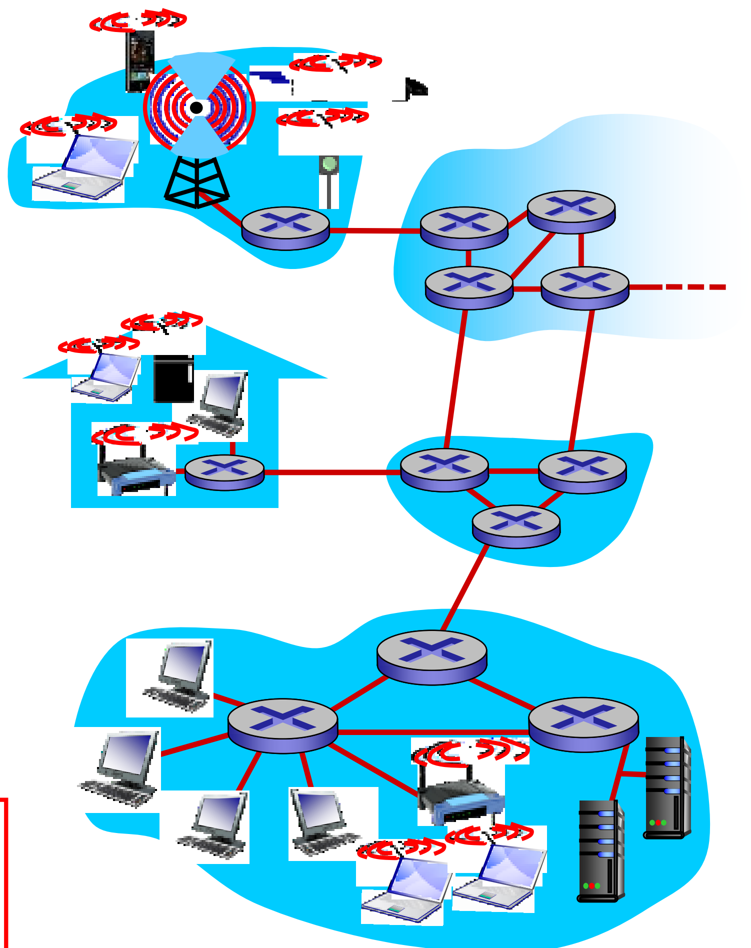
6.6 data center networking

6.7 a day in the life of a web request

Link layer: introduction

Terminology:

- hosts and routers: nodes
- communication channels that connect adjacent nodes along communication path: links
  - wired links
  - wireless links
  - LANs
- layer-2 packet: frame, encapsulates datagram



Data-link layer has responsibility of transferring datagram from one node to *physically adjacent* node over a link

Link layer: context

- Datagrams transferred by different link protocols over different links:
  - e.g., Ethernet on first link, frame relay on intermediate links, IEEE 802.11 (aka wifi) on last link
- Each link protocol provides different services
  - e.g., may or may not provide rdt over link

Transportation analogy:

- Trip from Princeton to Lausanne
  - limo: Princeton to JFK
  - plane: JFK to Geneva
  - train: Geneva to Lausanne

- Tourist = datagram
- Transport segment = communication link
- Transportation mode = link layer protocol
- Travel agent = routing algorithm

Link layer services

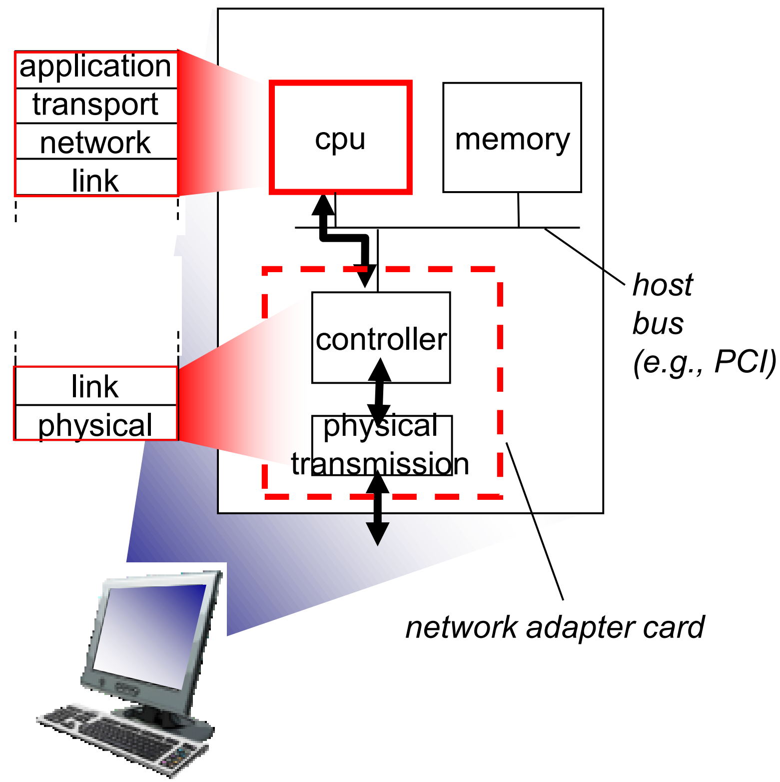
- *Framing, link access*
  - encapsulates datagrams into frames, adding header and trailer
  - rules channel access if shared medium
  - provides Medium Access Control (MAC) addresses used in frame headers to identify source, destination
    - different from IP addresses!
- *Reliable delivery between adjacent nodes*
  - we learned how to do this already (chapter 3)!
  - seldom used on low bit-error link (fiber, some twisted pair)
  - wireless links: high error rates
    - *Q:* why both link-level and end-to-end reliability?

Link layer services (more)

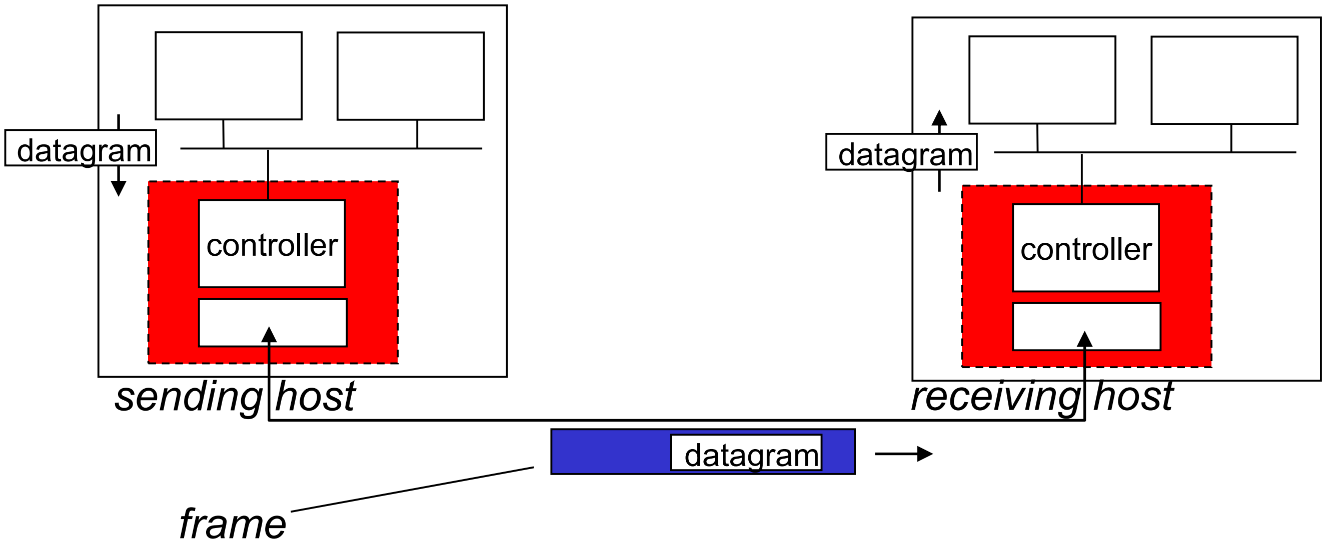
- *Flow control:*
  - pacing between adjacent sending and receiving nodes
- *Error detection:*
  - errors caused by signal attenuation, noise
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
- *Error correction:*
  - receiver identifies *and corrects* bit error(s) without resorting to retransmission
- *Half-duplex and full-duplex*
  - with half duplex, nodes at both ends of link can transmit, but not at same time

Where is the link layer implemented?

- In each and every host
- link layer implemented in "adaptor" (aka *Network Interface Card* NIC) or on a chip
  - Ethernet card, IEEE 802.11 card; Ethernet chipset
  - implements link, physical layer
- Attaches into host's system buses
- Combination of hardware, software, firmware



Adaptors communicating



- Sending side:
  - encapsulates datagrams in frames
  - adds error checking bits, rdt, flow control, etc.
- Receiving side
  - looks for errors, rdt, flow control, etc.
  - extracts datagrams, passes to upper layer at receiving side

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

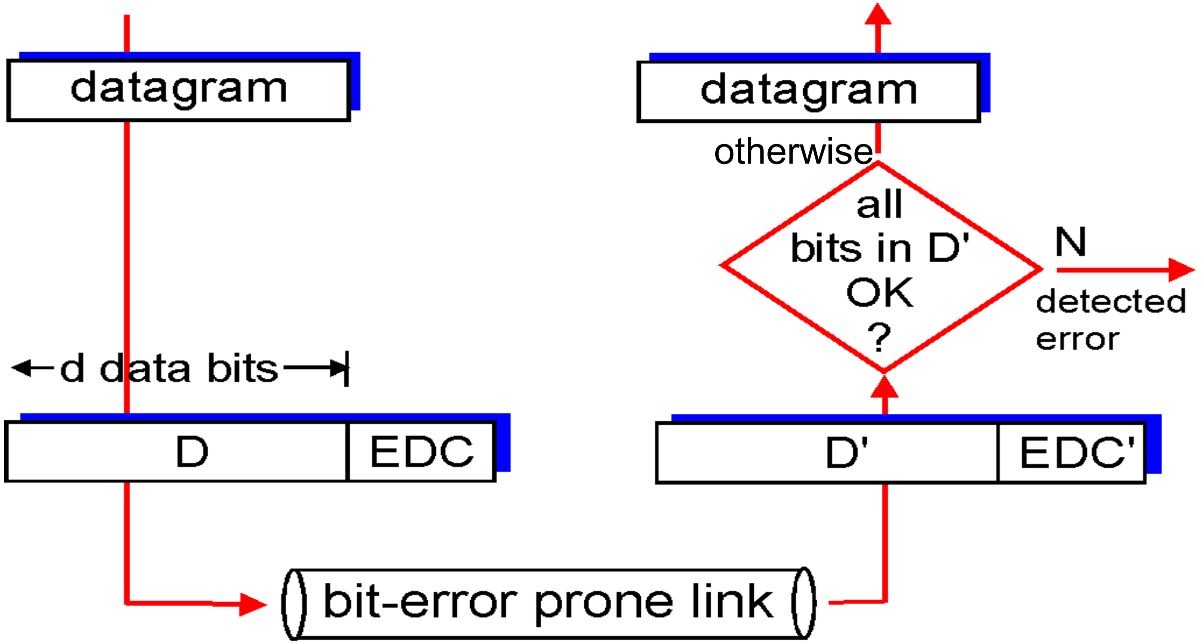
6.7 a day in the life of a web request

Error detection

EDC = Error Detection and Correction bits (redundancy)
D = Data protected by error checking, may include header fields

Error detection not 100% reliable!

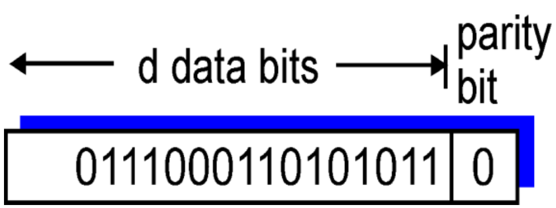
- protocol may miss some errors, but rarely
- larger EDC field yields better detection and correction



Parity checking

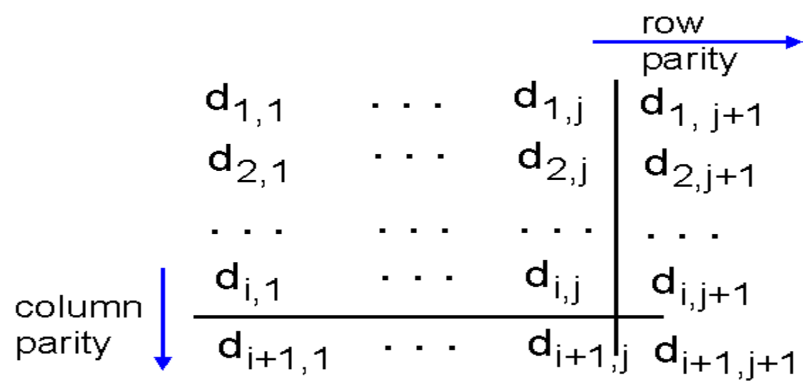
Single bit parity:

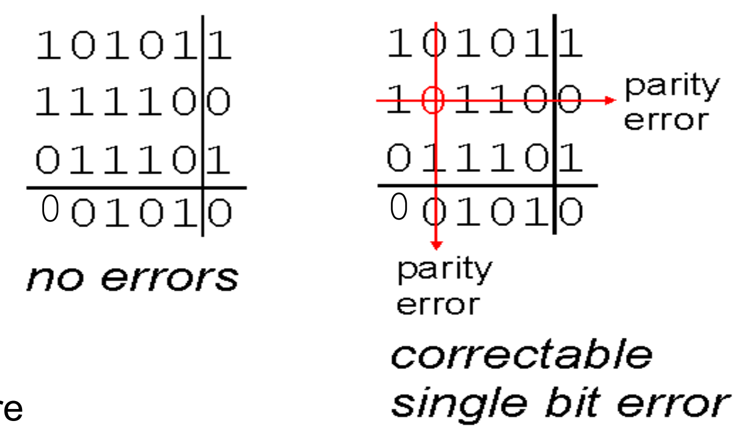
- detect single bit errors



Two-dimensional bit parity:

- detect and correct single bit errors





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

Internet checksum (review)

Goal: detect "errors" (e.g., flipped bits) in transmitted packet (note: used at transport layer only)

Sender:

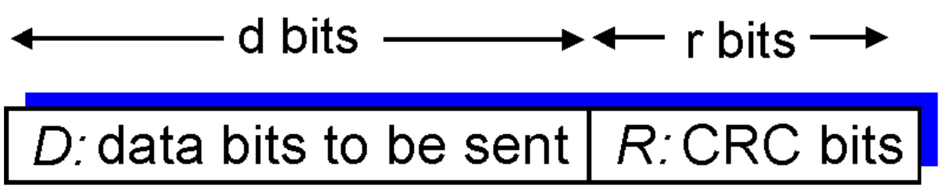
- treat segment contents as sequences of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

Receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?*

Cyclic Redundancy Check - CRC

- More powerful error-detection coding
- View data bits, D, as a binary number
- Choose r+1 bit pattern (generator), G
- Goal: choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
  - **can detect all burst errors less than r+1 bits**
- Widely used in practice (Ethernet, IEEE 802.11 WiFi, ATM)



bit pattern

$D * 2^r \text{ XOR } R$

mathematical formula

Additional Material

CRC example

Want:

$D \cdot 2^r$ XOR $R = nG$

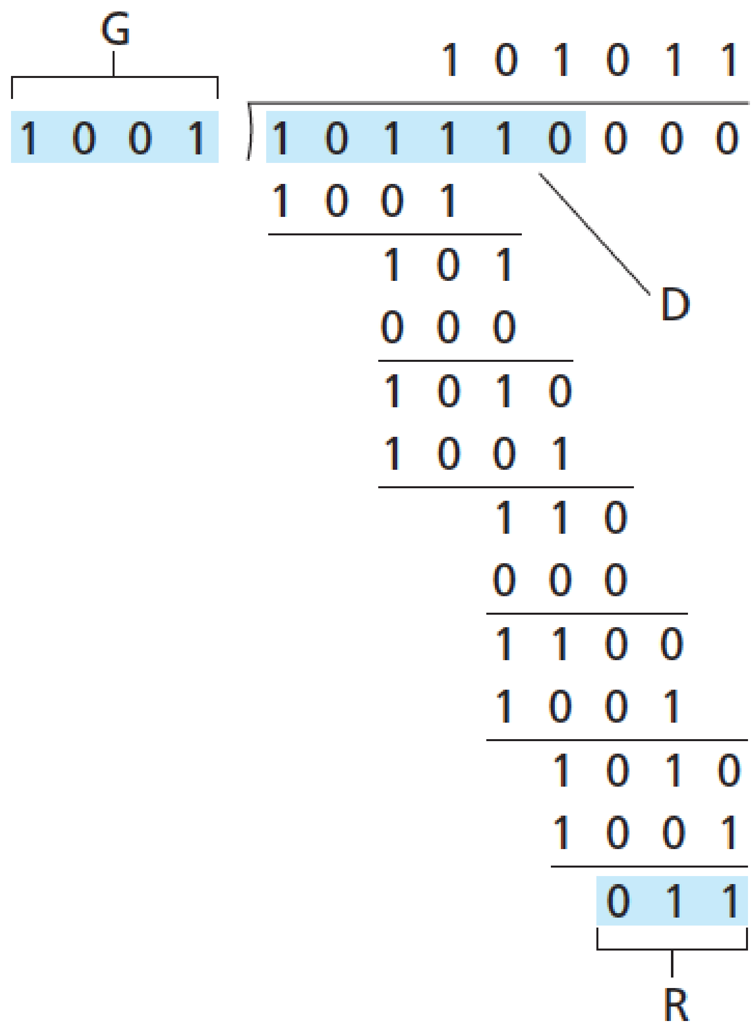
Equivalently:

$D \cdot 2^r = nG$ XOR R

Equivalently:

if we divide $D \cdot 2^r$ by G, want remainder R to satisfy:

$$R = remainder\left[\frac{D \cdot 2^r}{G}\right]$$



* 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

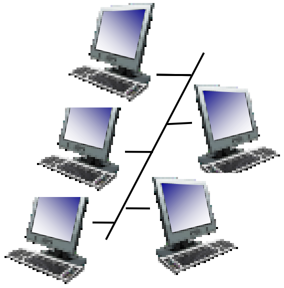
6.6 data center networking

6.7 a day in the life of a web request

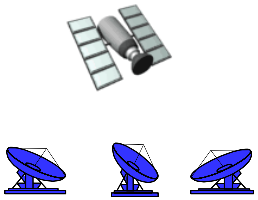
Multiple access links, protocols

Two types of "links"

- point-to-point
  - PPP for dial-up access
  - point-to-point link between Ethernet switch, host
- *broadcast (shared wire or medium)*
  - old-fashioned Ethernet
  - upstream HFC - hybrid fiber-coaxial
  - IEEE 802.11 wireless LAN


shared wire (e.g., cabled Ethernet)


shared RF (e.g., 802.11 WiFi)


shared RF (satellite)


humans at a cocktail party (shared air, acoustical)

Multiple access protocols

- Single shared broadcast channel accessed by multiple entities
- Two or more simultaneous transmissions by nodes: interference
  - *collision* if node receives two or more signals at the same time

Multiple access protocol

- distributed algorithm that determines how nodes share channels, i.e., determines when node can transmit
- communication about channel sharing must use channel itself!
  - no out-of-band channel for coordination

An ideal multiple access protocol

Given: broadcast channel of rate R bps

Desiderata:

1. when one node wants to transmit, it can send at rate R
2. when M nodes want to transmit, each can send at average rate R/M
3. fully decentralized:
   - no special node to coordinate transmissions
   - no synchronization of clocks, slots
4. simple

MAC protocols: taxonomy

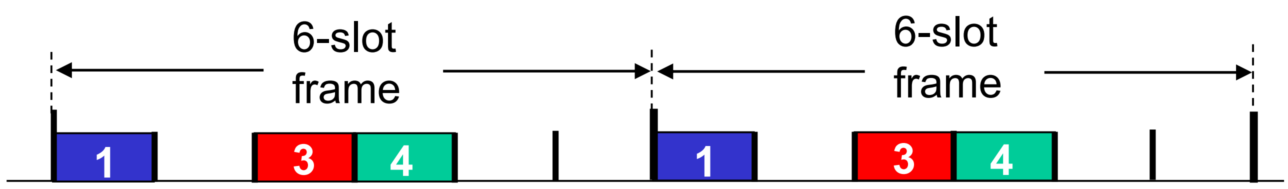
Three broad classes:

- *Channel partitioning*
  - divide channel into smaller "pieces" (time slots, frequency, code)
  - allocate piece to node for exclusive use
- *Random access*
  - channel not divided, allow collisions
  - "recover" from collisions
- *"Taking turns"*
  - nodes take turns, but nodes with more to send can take longer turns

Channel partitioning MAC protocols: TDMA

TDMA: Time Division Multiple Access

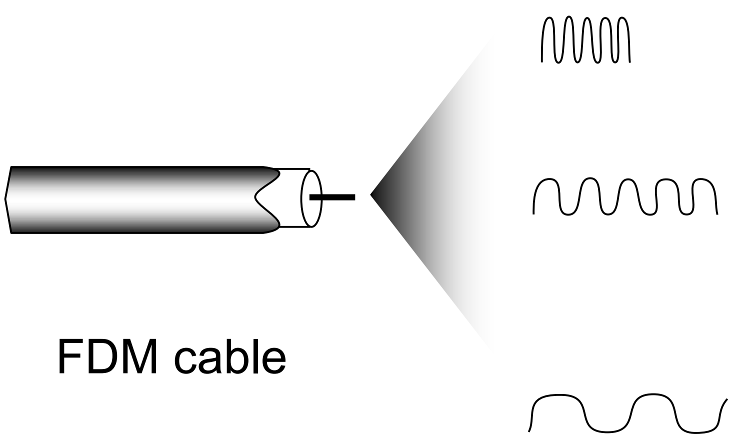
- access to channel in "rounds"
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle

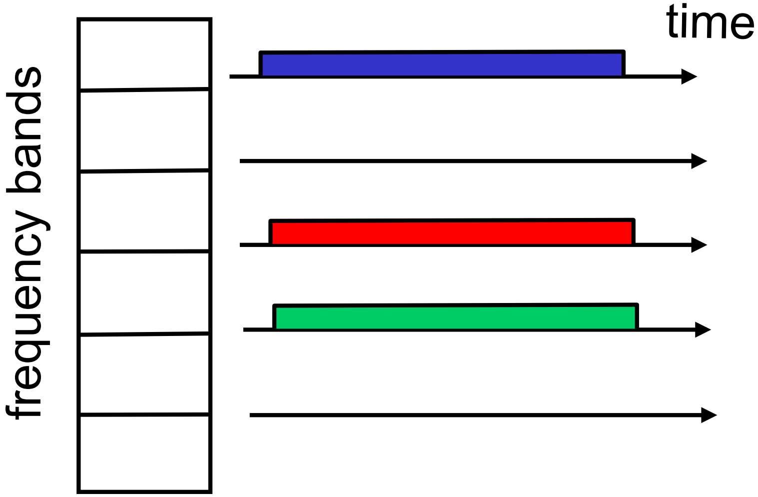


Channel partitioning MAC protocols: FDMA

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 packet to send, frequency bands 2,5,6 idle





Random access protocols

- When a node has packet to send
  - transmits at full channel data rate R
  - no *a priori* coordination among nodes
- Two or more transmitting nodes ➜ "collision"
- Random access MAC protocol specifies
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)
- Examples of random access MAC protocols:
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA

Slotted ALOHA

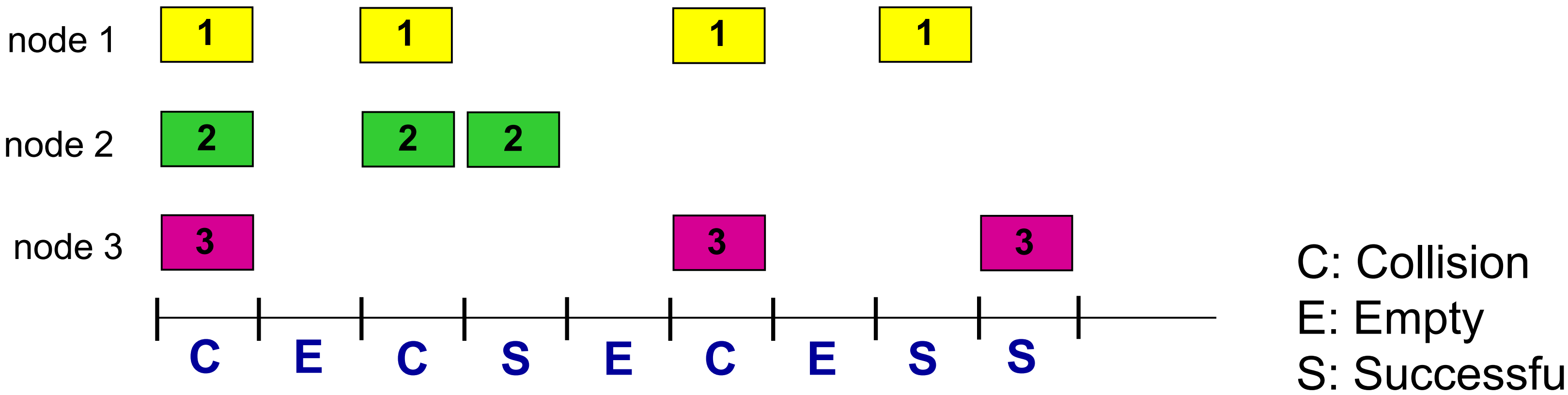
Assumptions:

- all frames have same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only at slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in the same slot, all nodes detect collision

Operation:

- when node obtains fresh frame, transmits in next slot
  - *if no collision:* node can send new frame in next slot
  - *if collision:* node retransmits frame in each subsequent slot with probability p until success

Slotted ALOHA



Pros:

- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in nodes need to be in sync
- simple

Cons:

- collisions, wasting slots
- idle slots
- nodes may be able to detect collision in less than time to transmit packet
- clock synchronization

Slotted ALOHA: efficiency

Efficiency: long-run fraction of successful slots (many nodes, all with many frames to send)

- *suppose: N* nodes with many frames to send, each transmits in slot with probability p
- prob that given node has success in a slot = $p(1-p)^{N-1}$
- prob that *any* node has a success = $Np(1-p)^{N-1}$

- max efficiency: find p^* that maximizes $Np(1-p)^{N-1}$
- for many nodes, take limit of $Np^*(1-p^*)^{N-1}$ as N goes to infinity, gives:

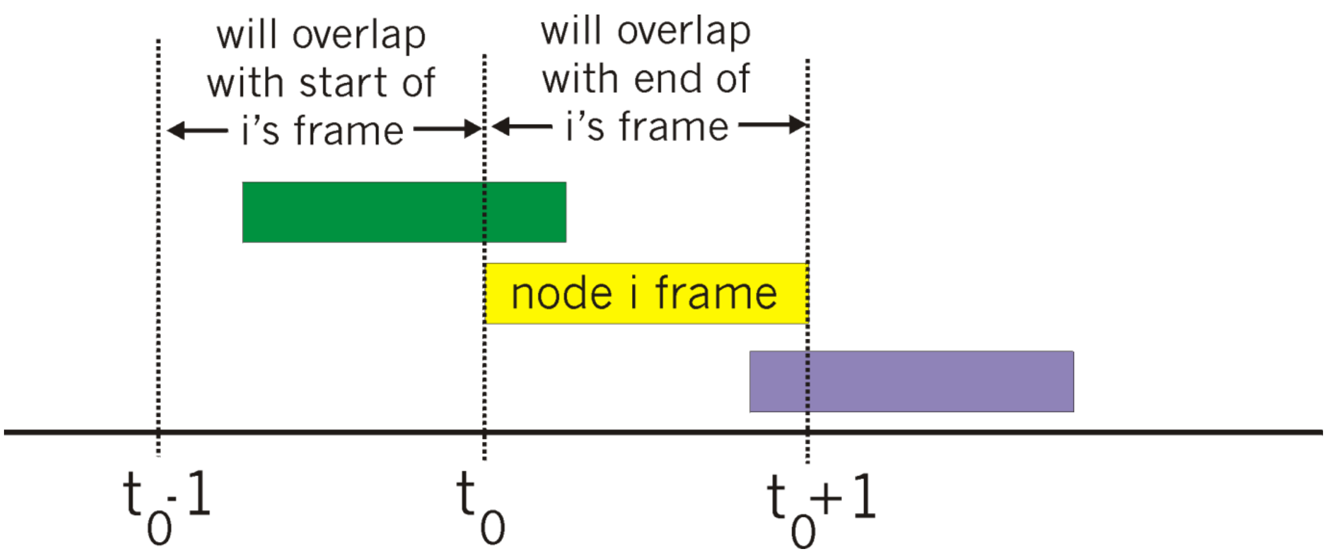
  max efficiency = 1/e = .37

At best: channel used for useful transmissions 37% of time!

!

Pure (unslotted) ALOHA

- Unslotted ALOHA: simpler, no synchronization
- When frame first arrives
  - transmit immediately
- Collision probability increases:
  - frame sent at t_0 collides with other frames sent in $[t_0-1, t_0+1]$



Pure ALOHA efficiency

P(success by given node) = P(node transmits) $\cdot$

P(no other node transmits in $[t_0-1, t_0]$ $\cdot$

P(no other node transmits in $[t_0, t_0+1]$

$$= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1}$$

$$= p \cdot (1-p)^{2(N-1)}$$

… choosing optimum p and then letting $n \longrightarrow \infty$

$$= 1/(2e) = .18 \rightarrow 18\%$$

even *worse* than slotted Aloha!

but simpler, no synchronization, fully decentralized

CSMA: Carrier Sense Multiple Access

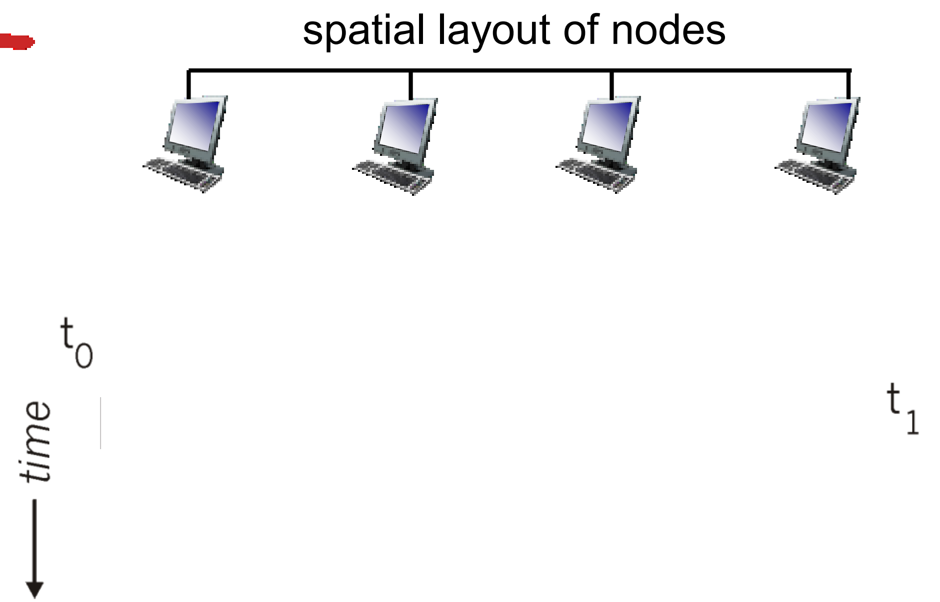
CSMA: listen before transmit:

- if channel sensed as idle, transmit entire frame
- if channel sensed as busy, defer transmission

- human analogy: don't interrupt others!

CSMA collisions

- Collisions *can* still occur: propagation delay means two nodes may not hear each other's transmission
- Collision: entire packet transmission time wasted
  - distance & propagation delay play role in determining collision probability



CSMA/CD: Collision Detection

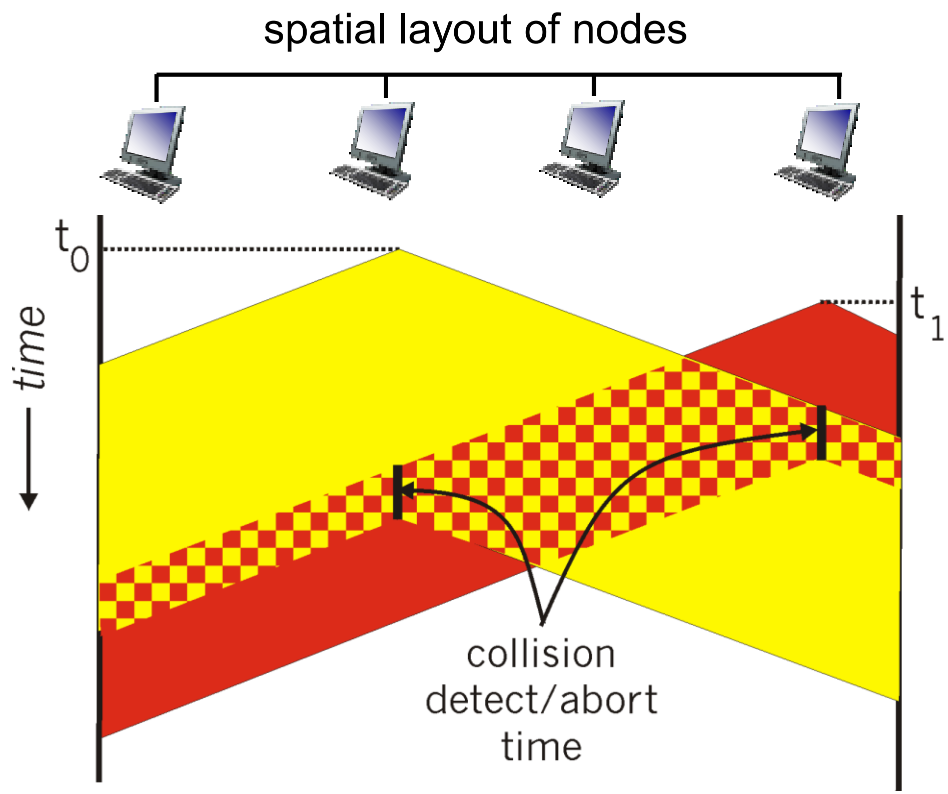
CSMA/CD: carrier sensing, deferral as in CSMA

- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage

- Collision detection:
  - easy in wired LANs: measure signal strengths, compare transmitted/received signals
  - difficult in wireless LANs: received signal strength overwhelmed by local transmission strength

- Human analogy: the polite conversationalist

CSMA/CD (collision detection)



Ethernet CSMA/CD algorithm

1. NIC receives datagrams from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission. If NIC senses channel busy, waits until channel is idle, then transmits.
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame!
4. **If NIC detects another transmission while transmitting**, aborts and sends jam signal
5. After aborting, NIC enters *binary (exponential) backoff:*
   - after m^{th} collision, NIC chooses K at random from $\{0,1,2, \ldots, 2^m-1\}$. NIC waits $K \cdot 512$ bit times, returns to Step 2
   - longer backoff interval with more collisions

CSMA/CD efficiency

- t_{prop} = max propagation delay between 2 nodes in LAN
- t_{trans} = time to transmit max-size frame

$$efficiency = \frac{1}{1 + 5t_{prop}/t_{trans}}$$

- efficiency goes to 1
  - as t_{prop} goes to 0: collisions immediately detected
  - as t_{trans} goes to infinity: no collisions since only one node transmitting; fairness?
- Better performance than ALOHA, and simple, cheap, decentralized!

"Taking turns" MAC protocols

Channel partitioning MAC protocols

- share channel *efficiently* and *fairly* at high load
- inefficient at low load: delay in channel access, 1/N bandwidth allocated even if only 1 active node!

Random access MAC protocols

- efficient at low load: single node can fully utilize channel
- high load: collision overhead

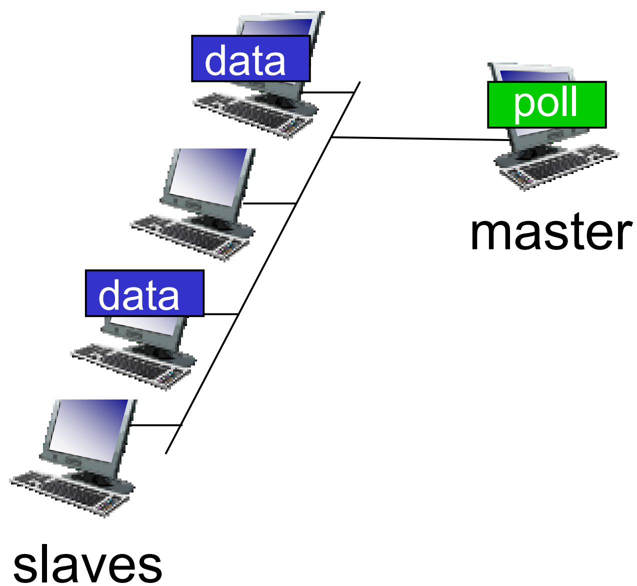
"Taking turns" protocols

look for best of both worlds!

"Taking turns" MAC protocols

Polling:

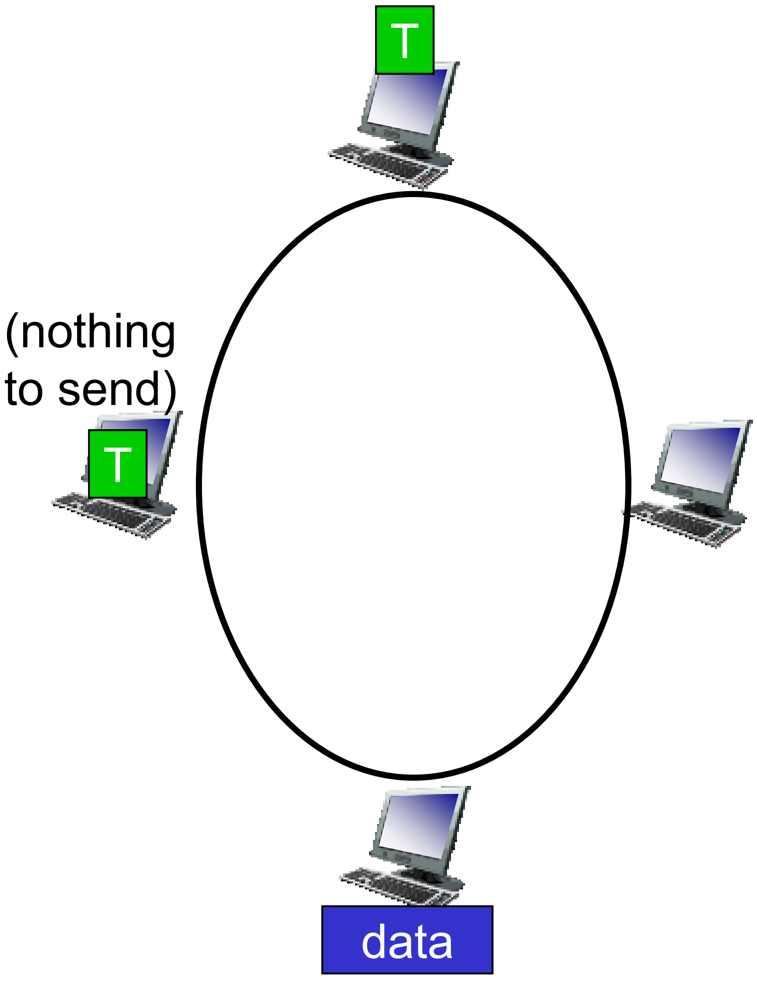
- master node "invites" slave nodes to transmit in turn
- typically used with "dumb" slave devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



"Taking turns" MAC protocols

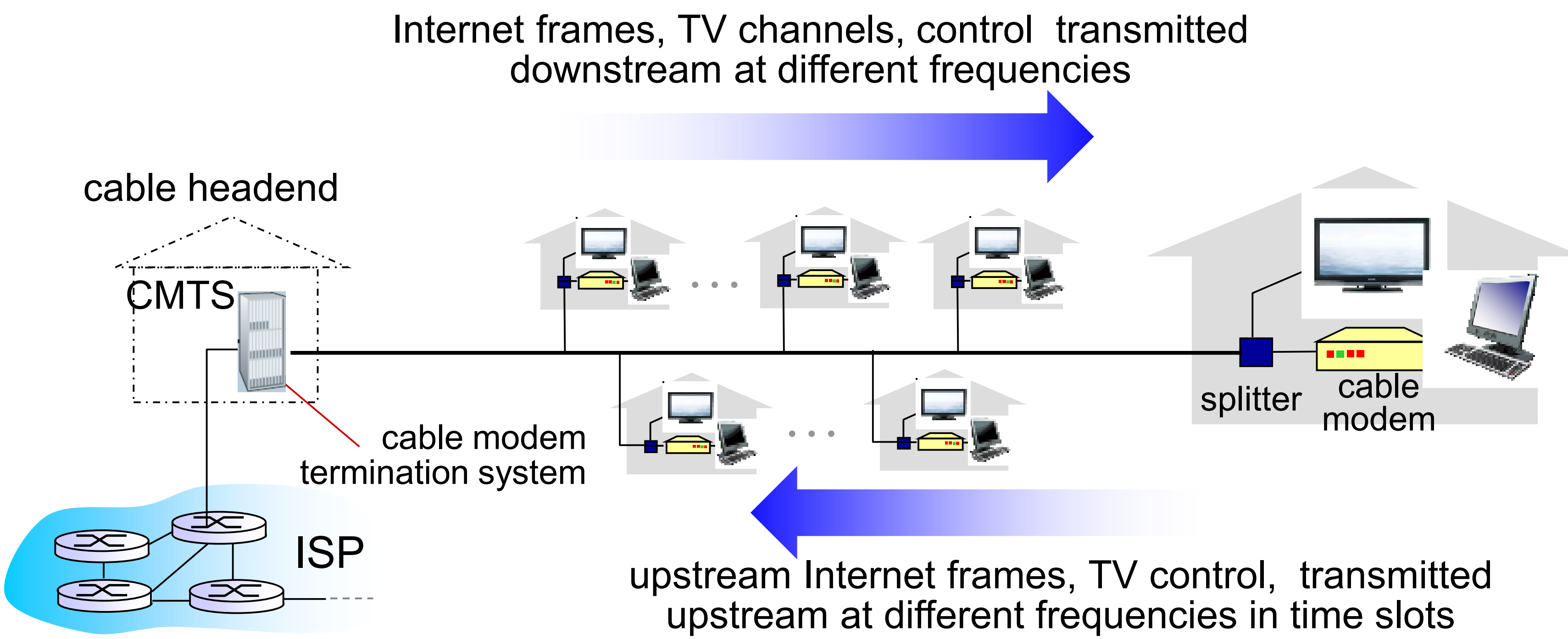
Token passing:

- control *token* passed from one node to next sequentially
- token message
- concerns:
  - token overhead
  - latency
  - single point of failure (token)



Additional Material

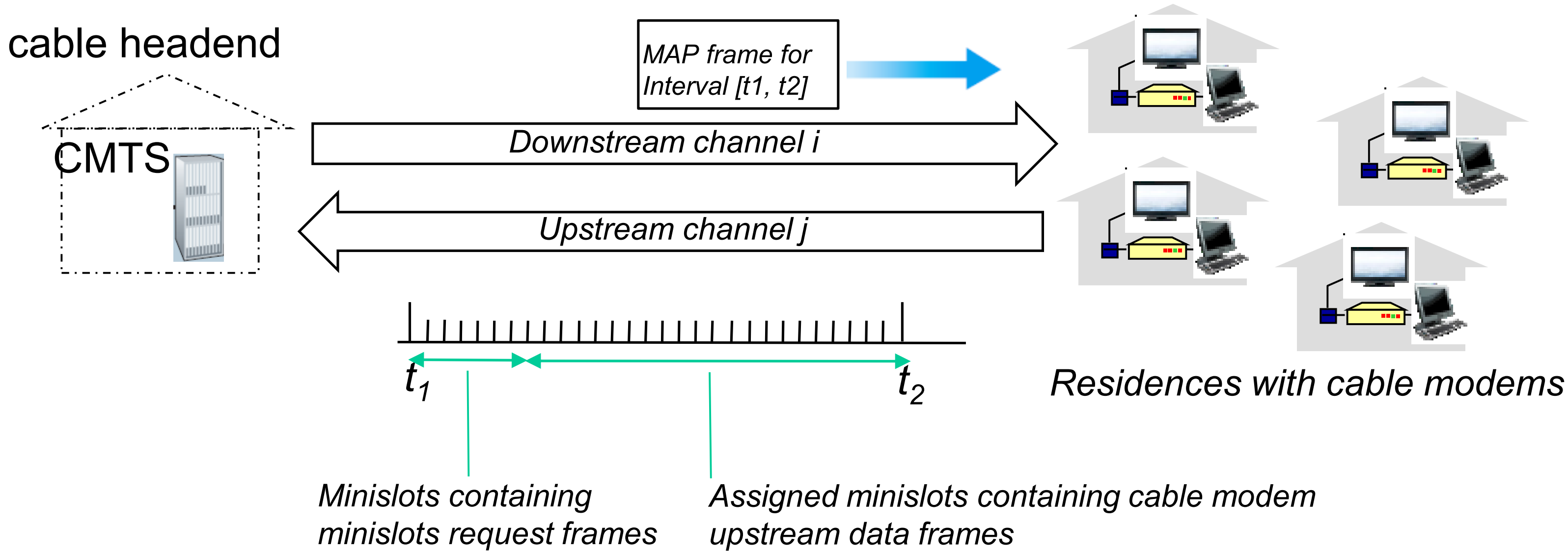
Cable access network



- Multiple 40Mbps downstream (broadcast) channels
  - single CMTS transmits into channels
- Multiple 30 Mbps upstream channels
  - multiple access: all users contend for certain upstream channel time slots (others assigned)

Additional Material

Cable access network



DOCSIS: data over cable service interface spec

- FDM over upstream, downstream frequency channels
- TDM upstream: some slots assigned, some have contention
  - downstream MAP frame: assigns upstream slots
  - request for upstream slots (and data) transmitted random access (binary backoff) in selected slots

Summary of MAC protocols

- *Channel partitioning,* by time, frequency or code
  - Time Division, Frequency Division
- *Random access* (dynamic)
  - ALOHA, S-ALOHA, CSMA, CSMA/CD
  - carrier sensing: easy in some technologies (wire), hard in others (wireless)
  - CSMA/CD used in Ethernet
  - CSMA/CA used in IEEE 802.11
- *Taking turns*
  - polling from central site, token passing
  - Bluetooth, Fiber Distributed Data Interface - FDDI, token ring

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

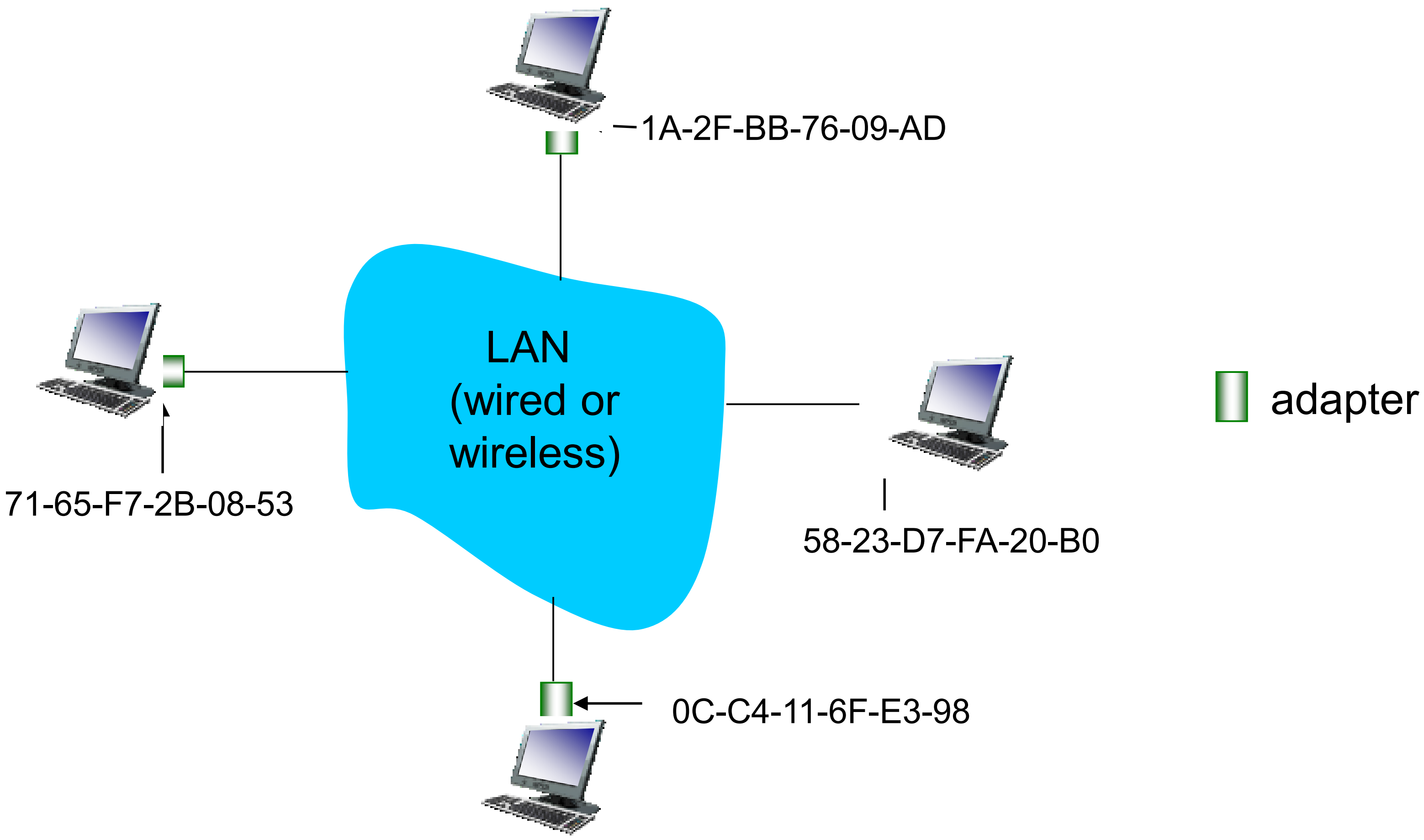
MAC addresses and ARP

- 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

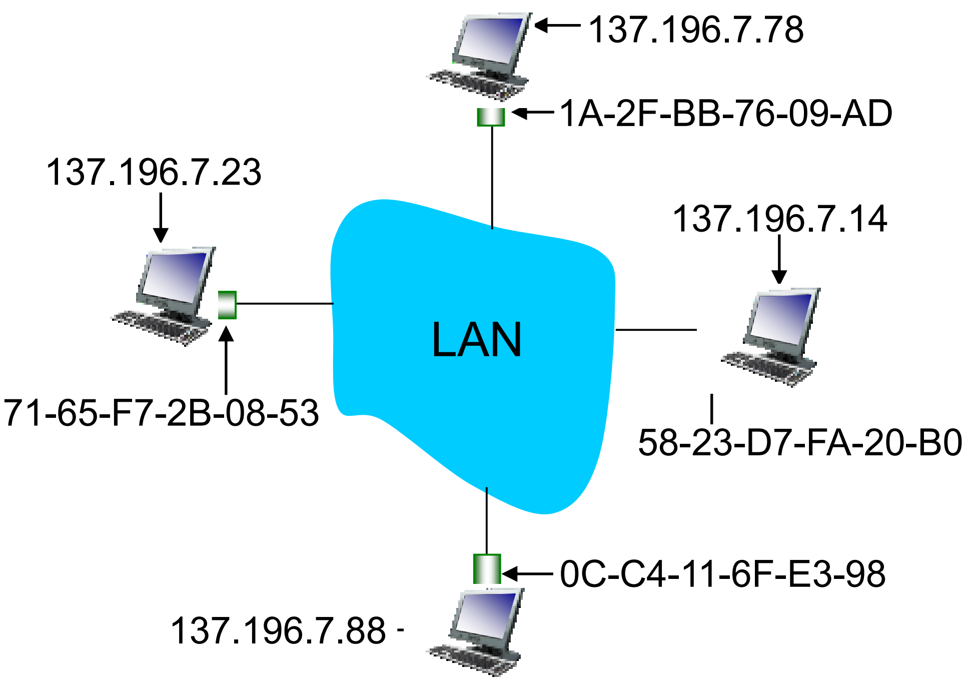


LAN addresses (more)

- MAC address allocation administered by IEEE
- Manufacturer buys portion of MAC address space (to assure uniqueness)
- Analogy:
  - MAC address: like Social Security Number
  - 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: 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 a table with

<IP address; MAC address; TTL>

- IP/MAC address mappings for some LAN nodes
- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

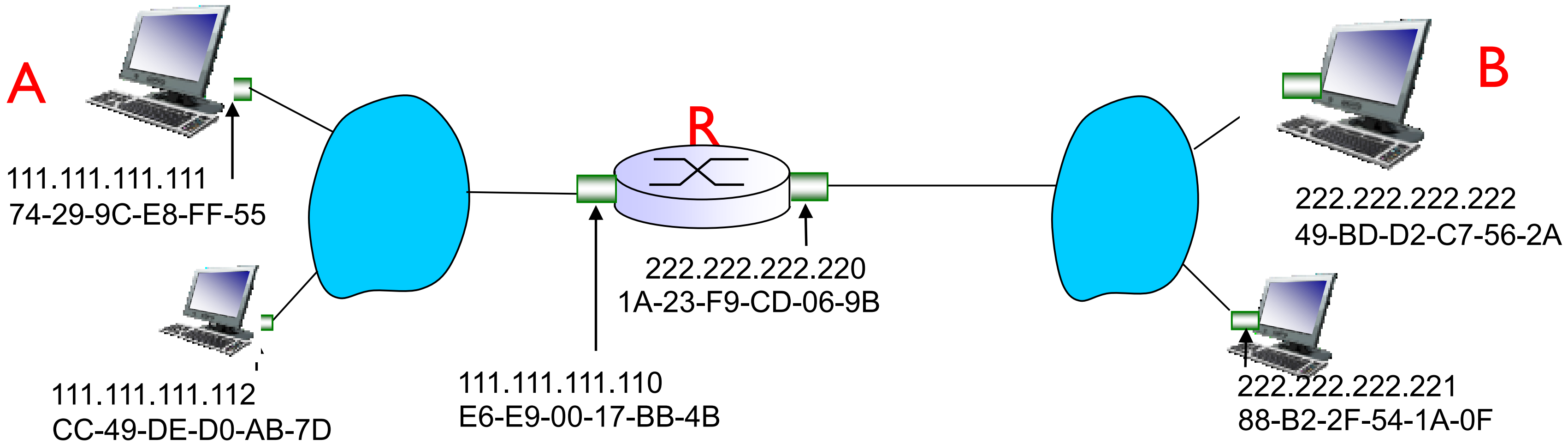
ARP protocol: same LAN

- A wants to send a 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

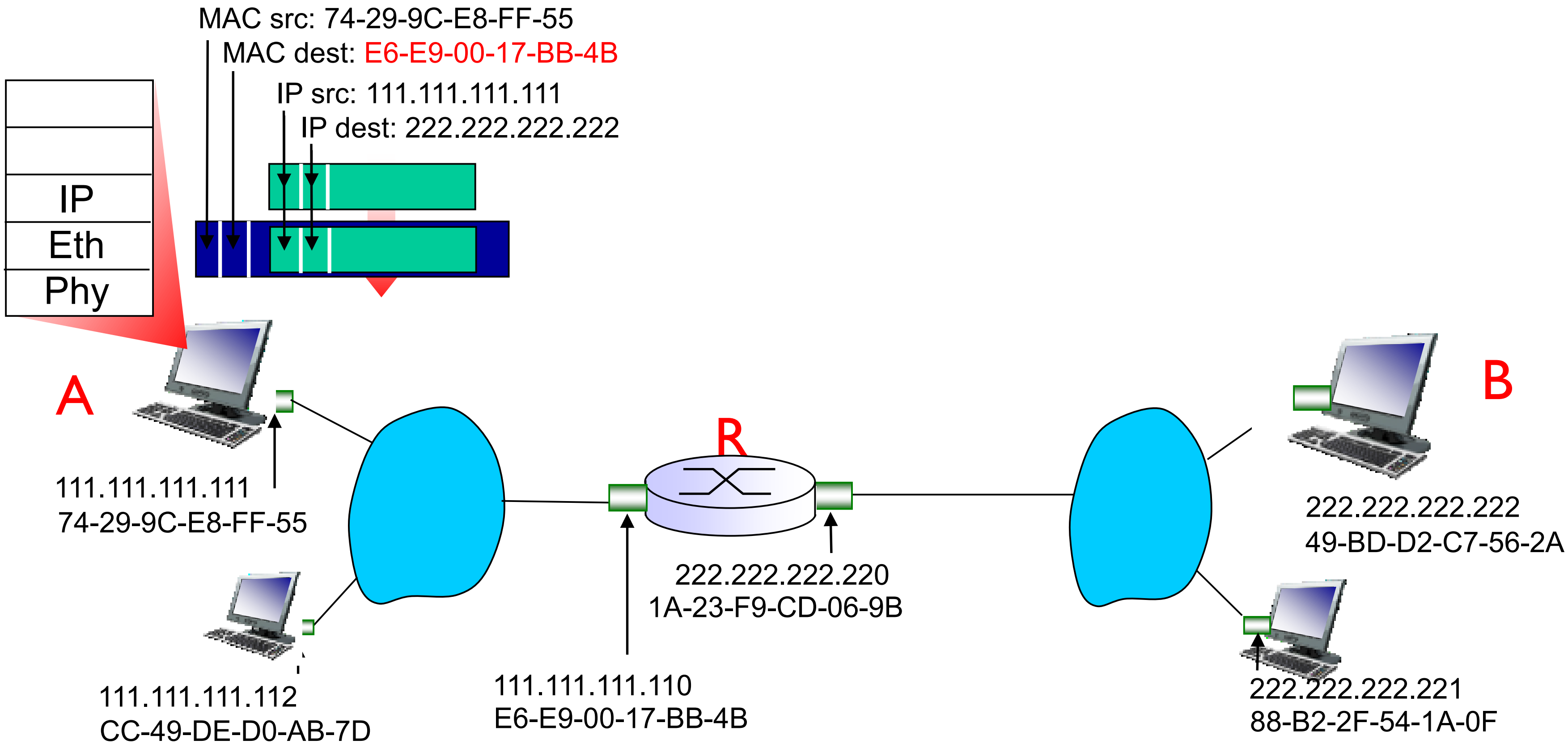
Walkthrough: send datagrams from A to B via R

- 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?)



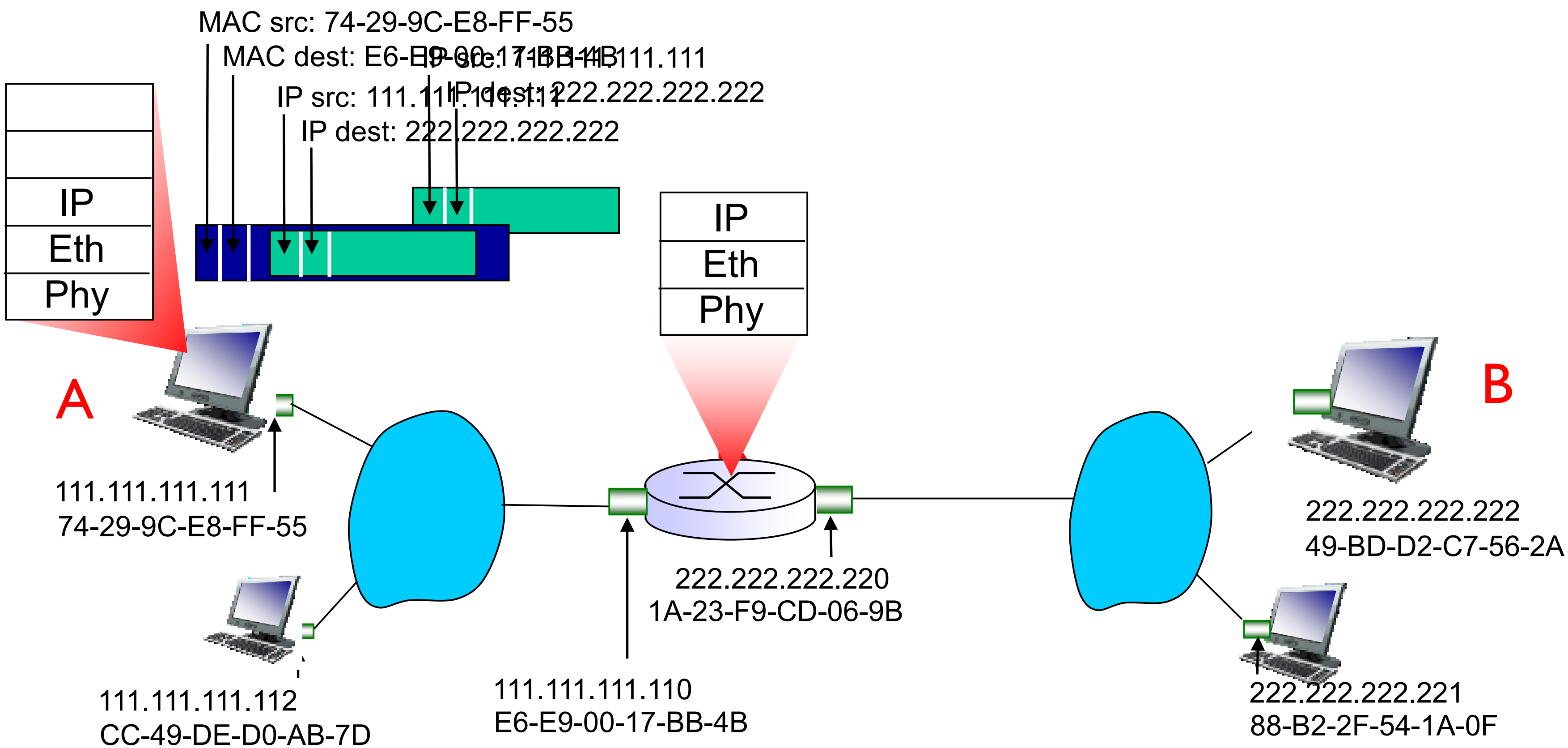
Addressing: routing to another LAN

- 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



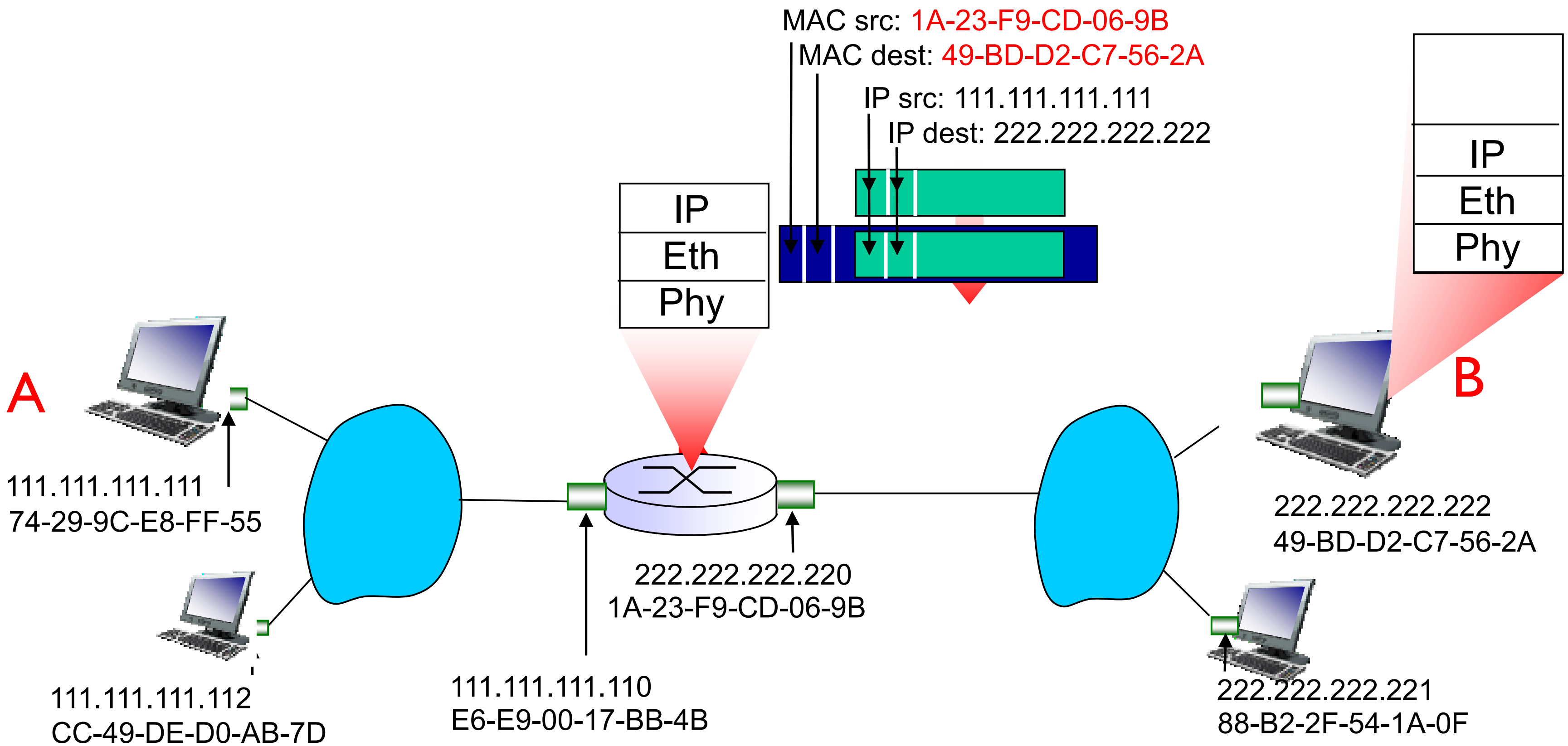
Addressing: routing to another LAN

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



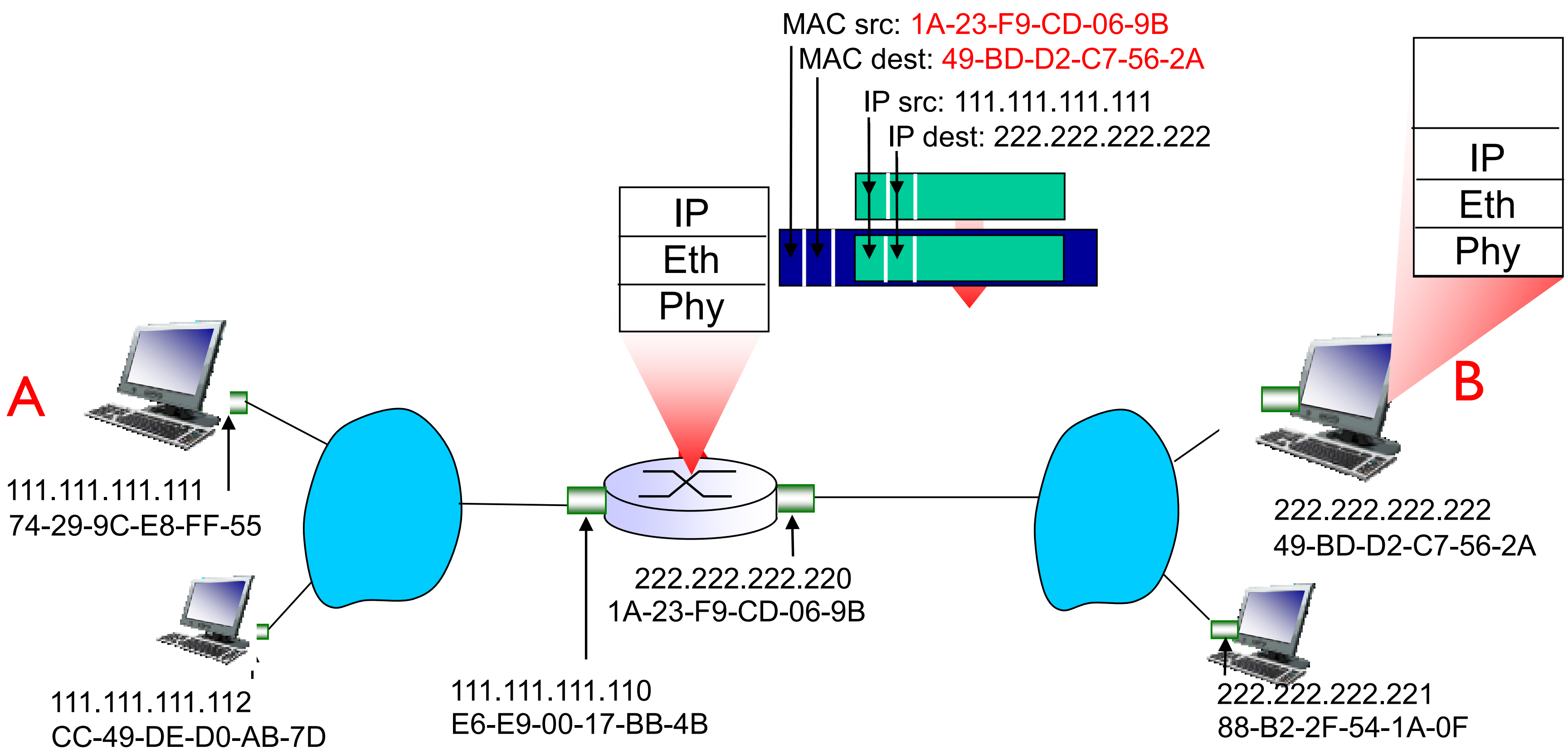
Addressing: routing to another LAN

- 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



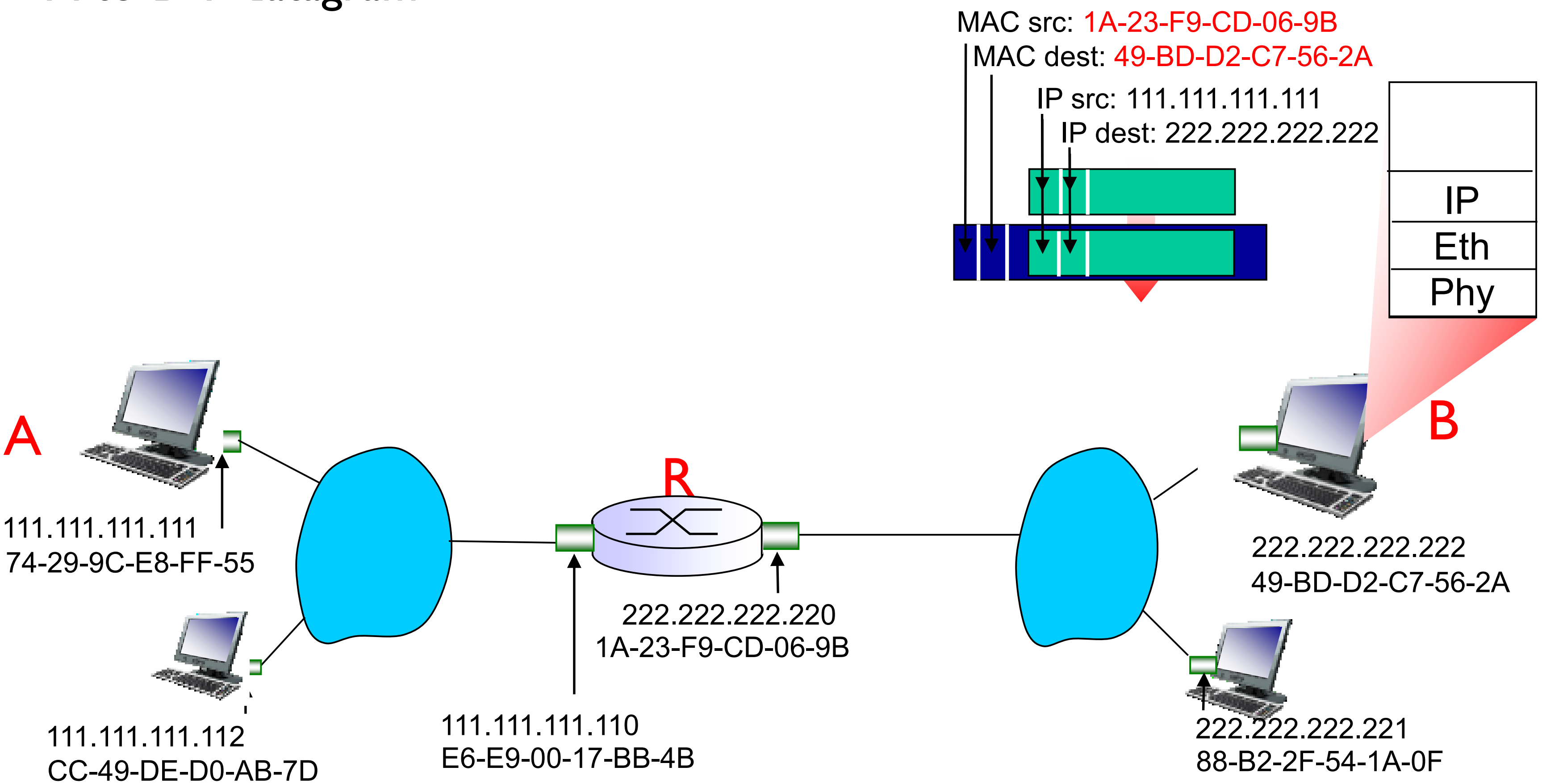
Addressing: routing to another LAN

- 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

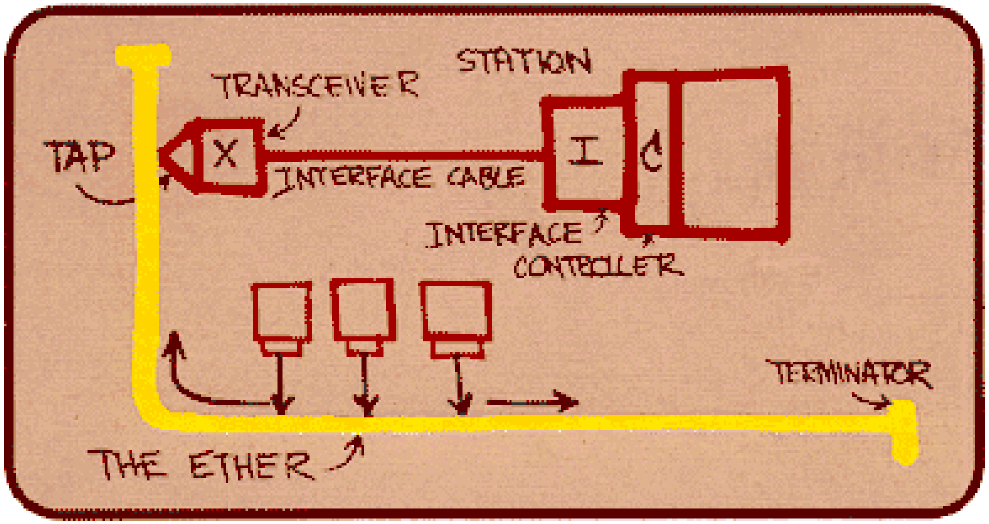
6.6 data center networking

6.7 a day in the life of a web request

Ethernet

"Dominant" wired LAN technology:

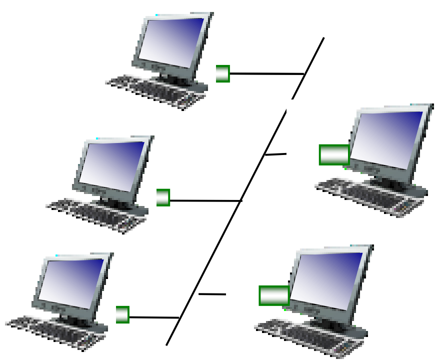
- single chip, multiple speeds (e.g., Broadcom BCM5761)
- first widely used LAN technology
- simpler, cheap
- kept up with speed race: 10 Mbps – 10 Gbps



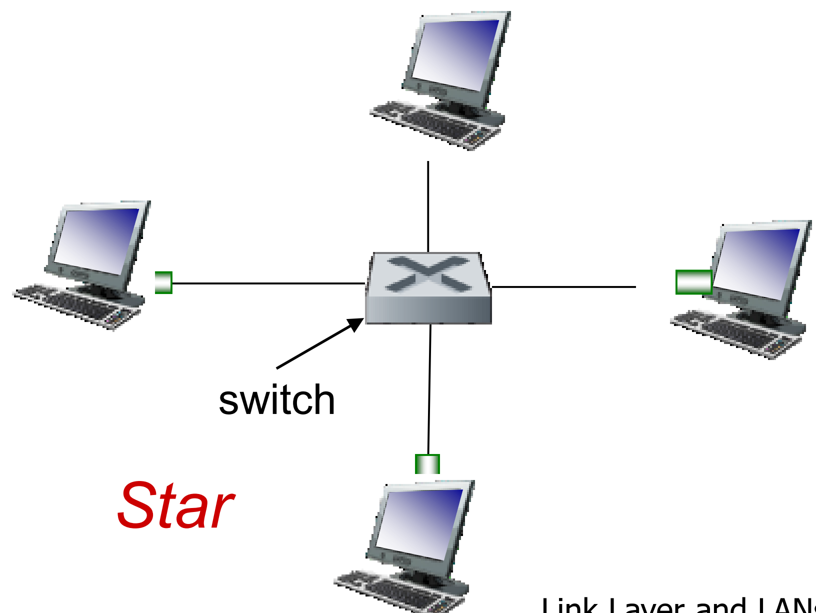
Metcalfe's Ethernet sketch

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



Star

Ethernet frame structure

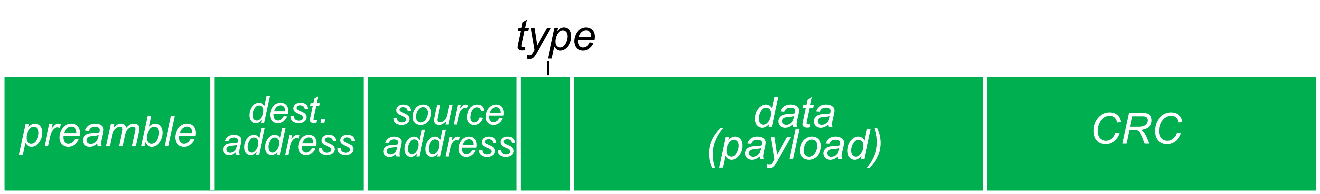
Sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame

Preamble:

- 8 bytes: 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:* 2 bytes, indicates higher layer protocol (mostly IP but others possible, e.g., Novell IPX, AppleTalk)
- *CRC:* 4 bytes, Cyclic Redundancy Check at receiver
  - error detected: frame is dropped

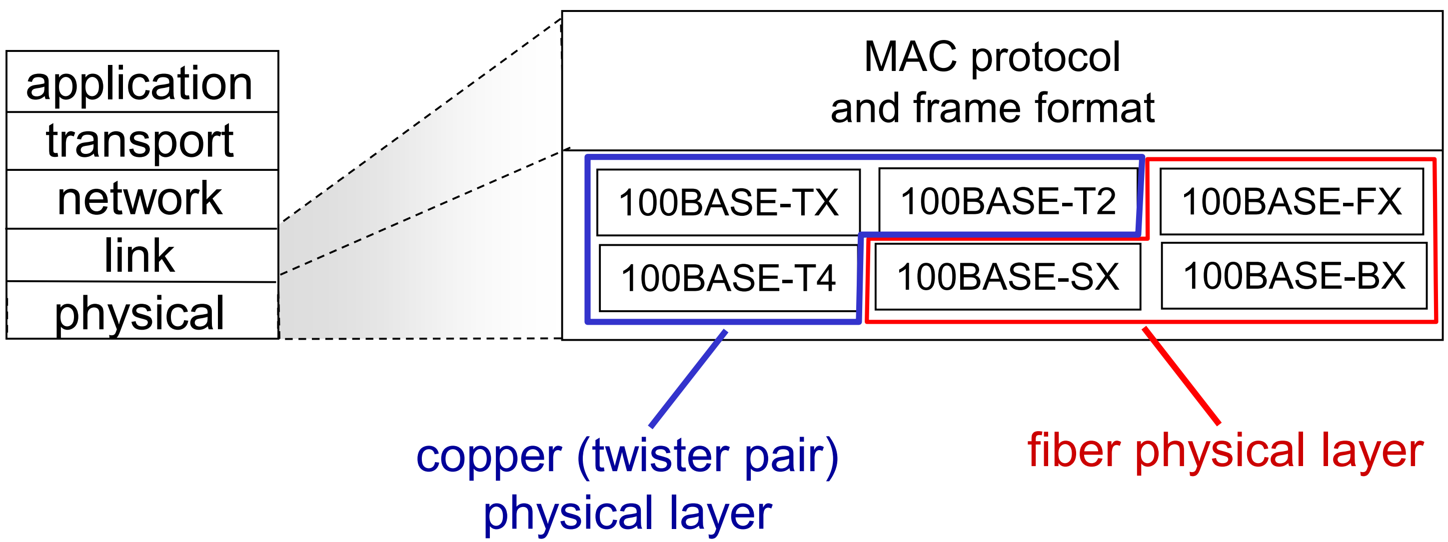


Ethernet: unreliable, connectionless

- *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*

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, 10 Gbps, 40 Gbps
  - different physical layer media: fiber, cable



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

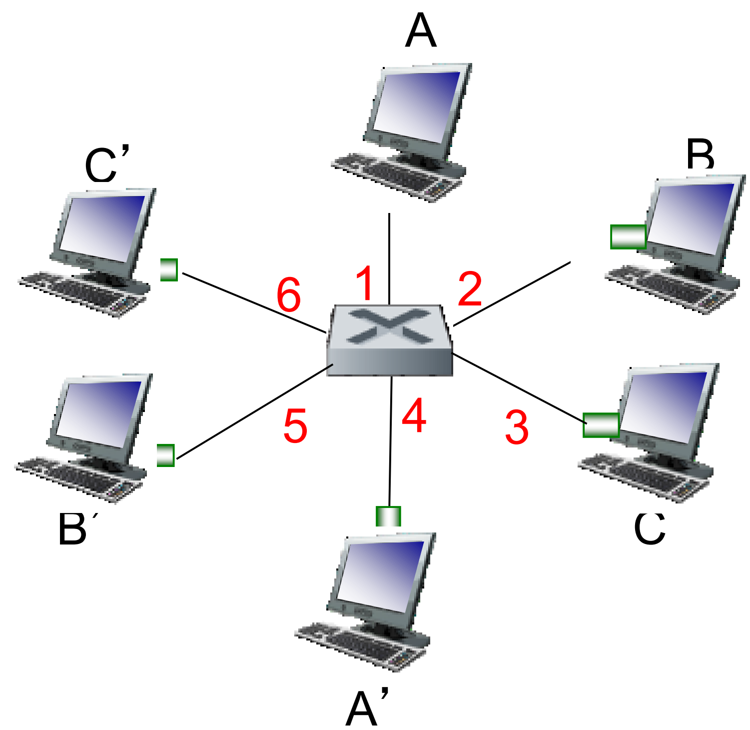
6.7 a day in the life of a web request

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 segment, uses CSMA/CD to access segment
- *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; full duplex
  - each link is its own collision domain
- *Switching:* A-to-A' and B-to-B' can transmit simultaneously, without collisions

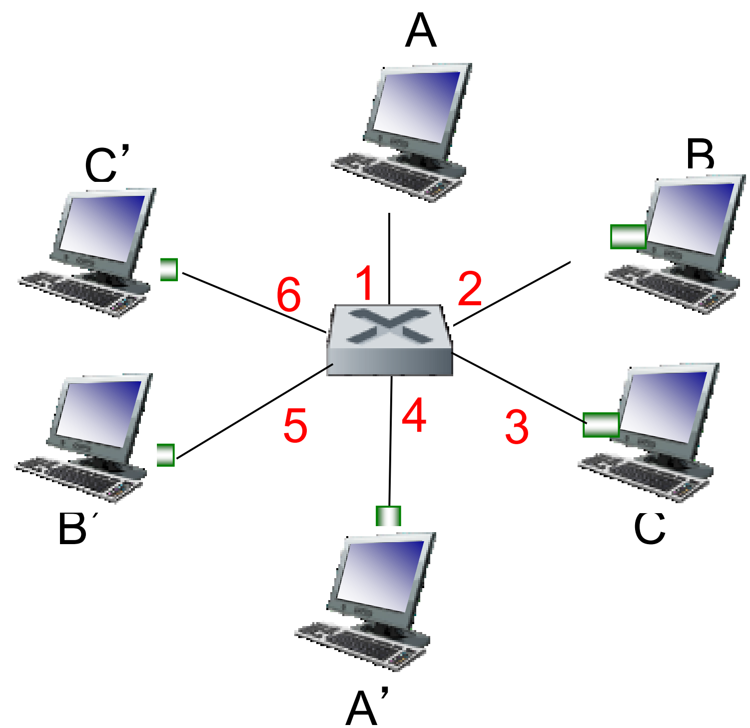


A switch with six interfaces (1,2,3,4,5,6)

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, timestamp
  - looks like a routing table!



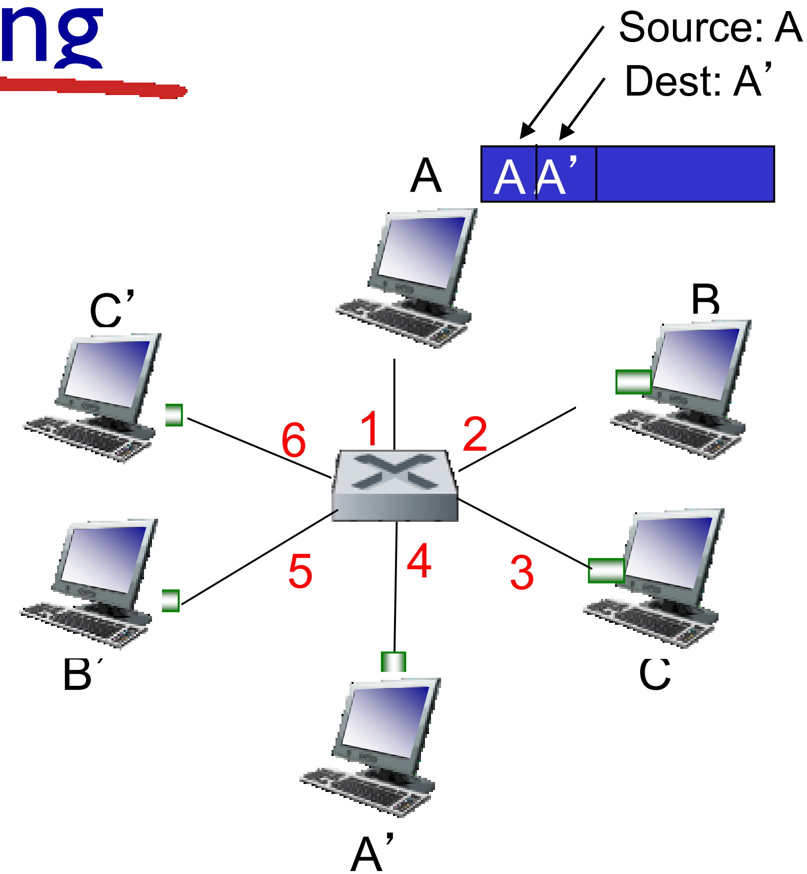
switch with six interfaces (1,2,3,4,5,6)

Q: how are entries created, maintained in switch table?

- something like a routing protocol?

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)

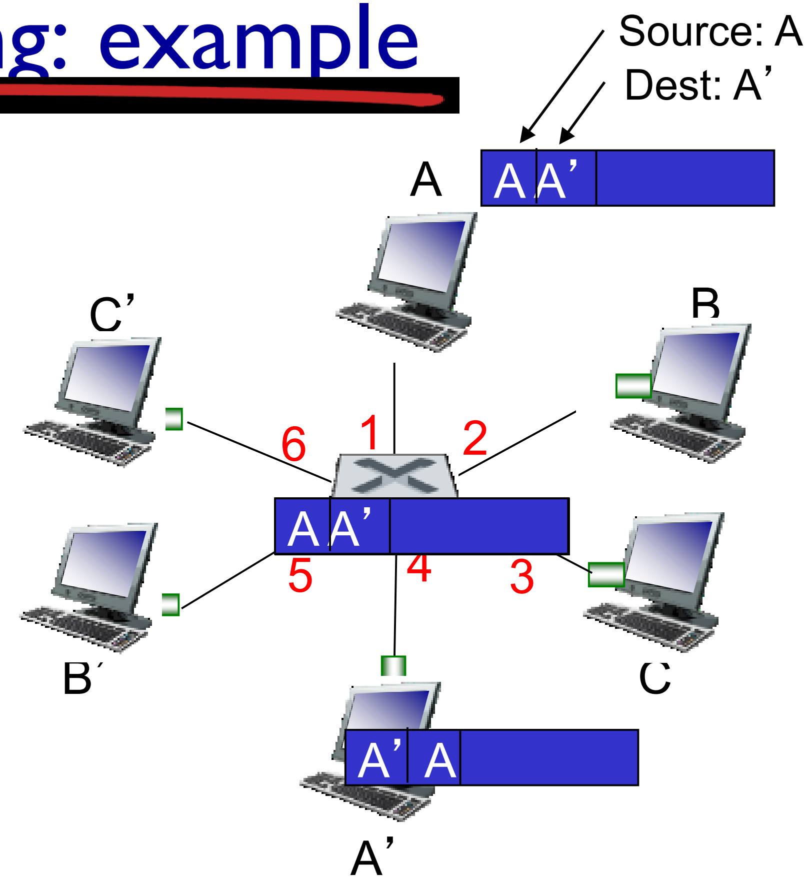
Switch: frame filtering/forwarding

When frames 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', location unknown: *flood*
- Destination A location known: selectively send on just one link

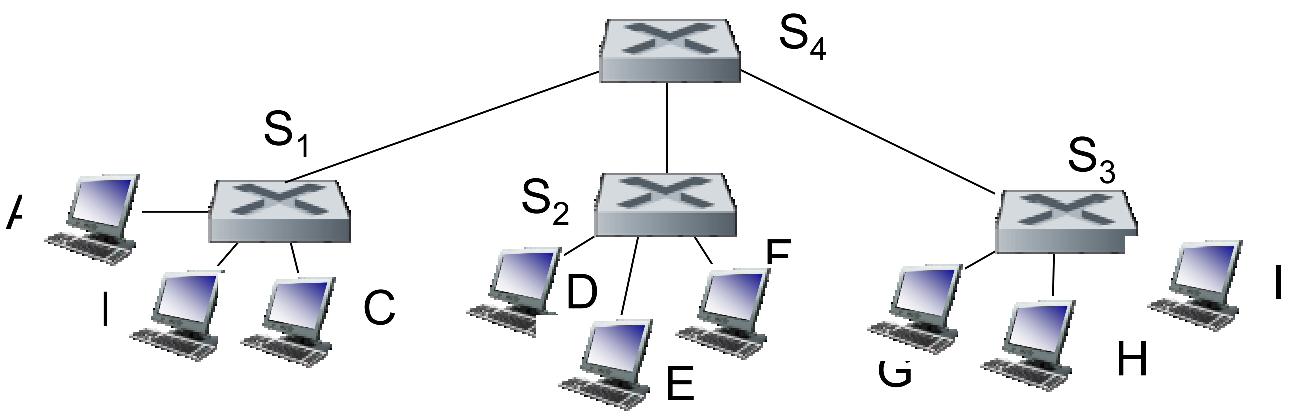


MAC addr	interface	TTL
A	*1*	*60*
A'	*4*	*60*

switch table (initially empty)

Interconnecting switches

Self-learning switches can be connected together:

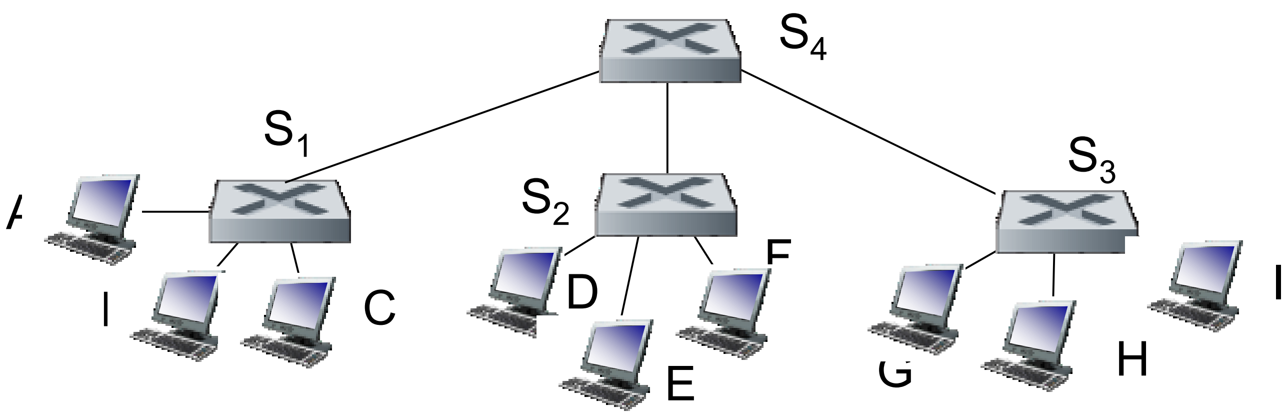


Q: sending from A to G - how does S_1 know to forward frame destined to G via S_4 and S_3?

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

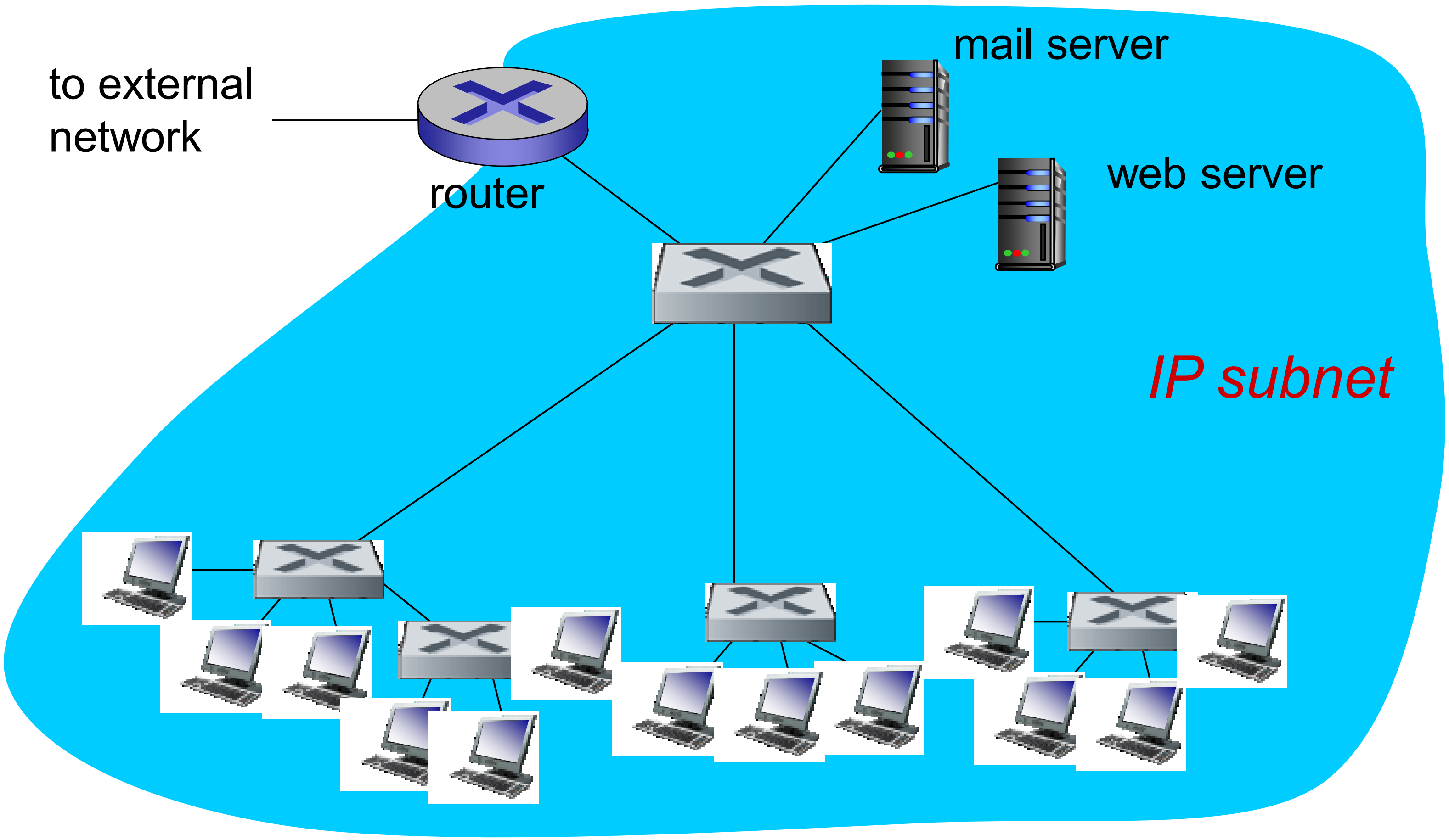
Self-learning multi-switch example

Suppose C sends frame to I, I responds to C

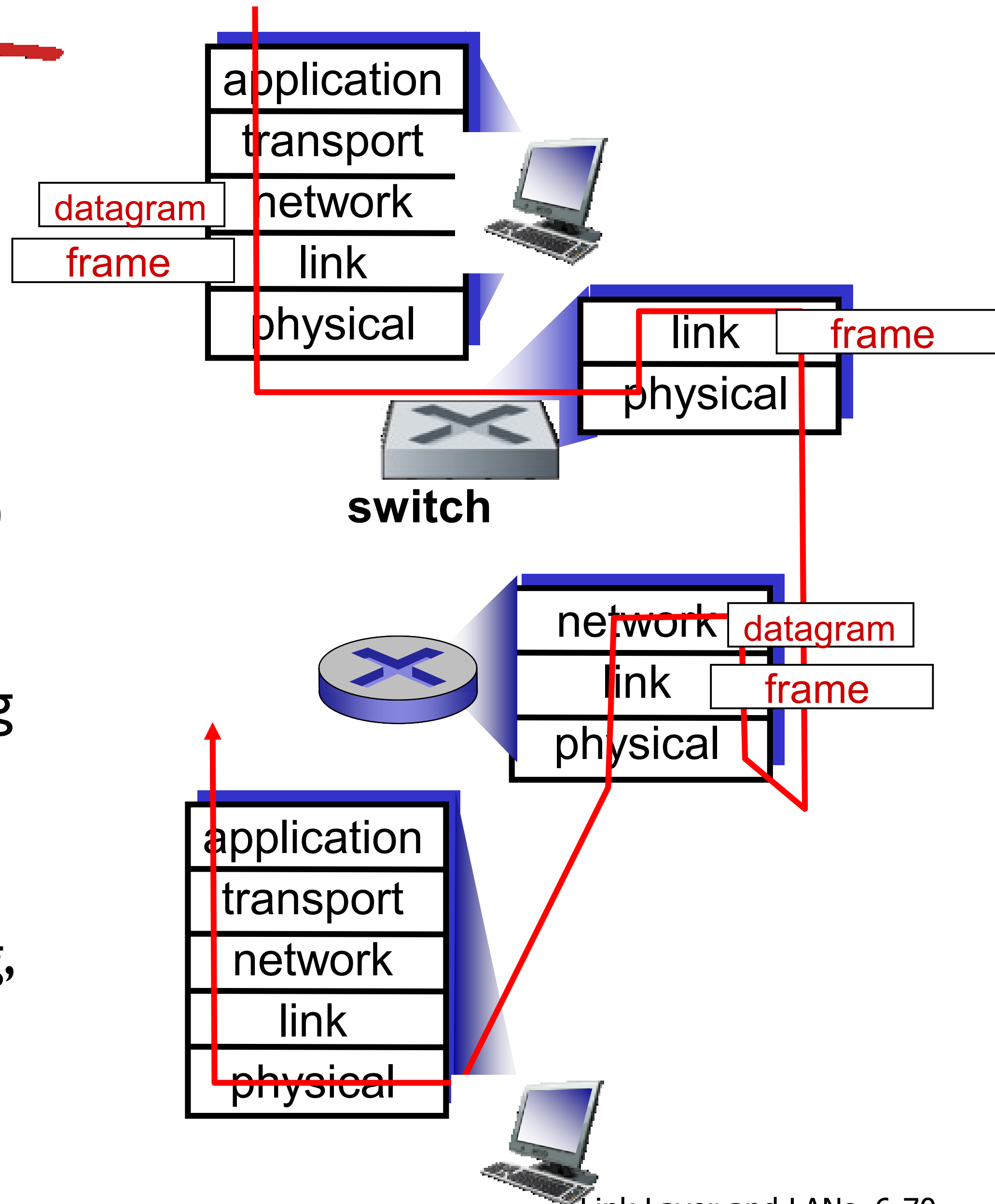


- Q: show switch tables and packet forwarding in S_1, S_2, S_3, S_4

Institutional network



Switches vs. routers



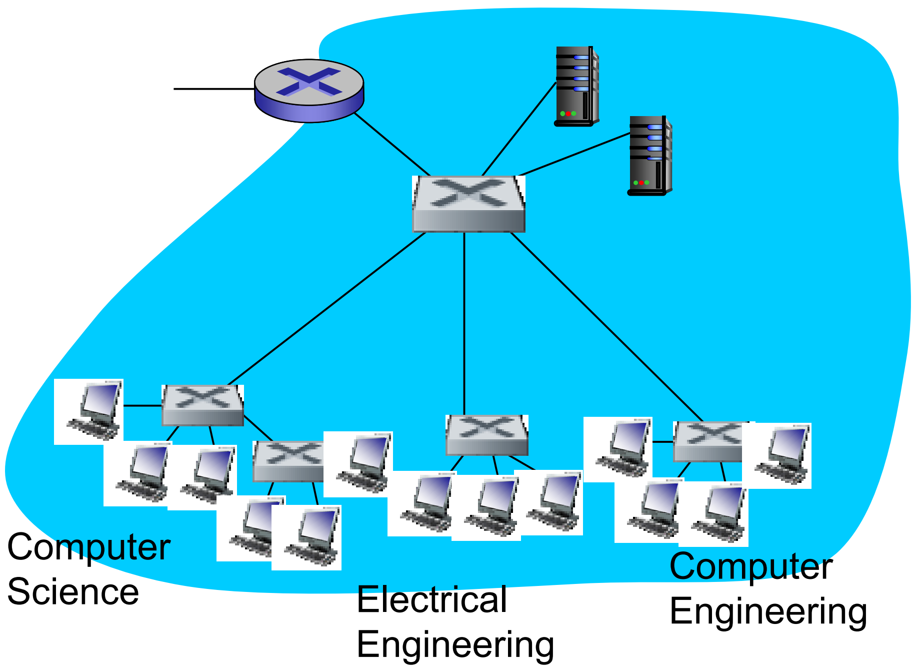
Both are store-and-forward:

- *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:

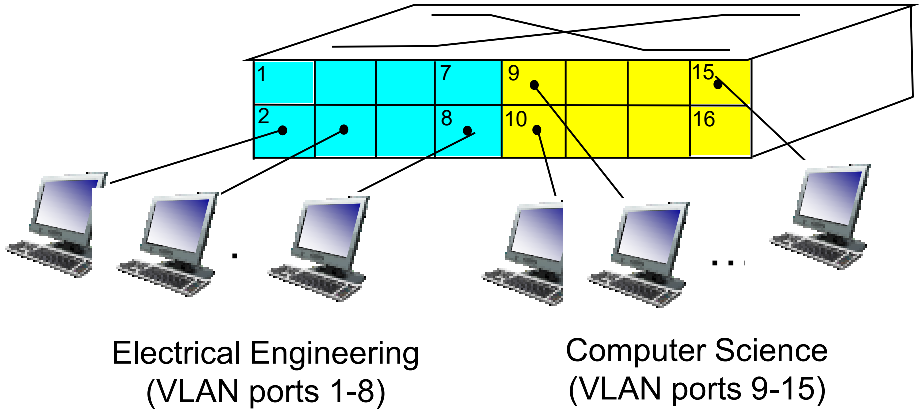
- CS user moves office to EE, but wants to connect to CS switch?
- Single broadcast domain:
  - all layer-2 broadcast traffic (ARP, DHCP, unknown location of destination MAC address) must cross entire LAN
  - security/privacy, efficiency issues

VLANs

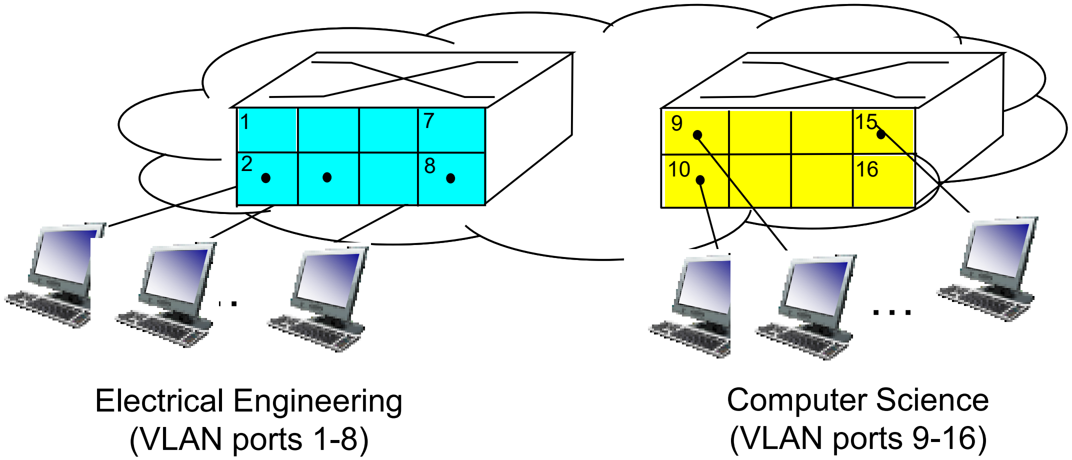
Virtual Local Area Network

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

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

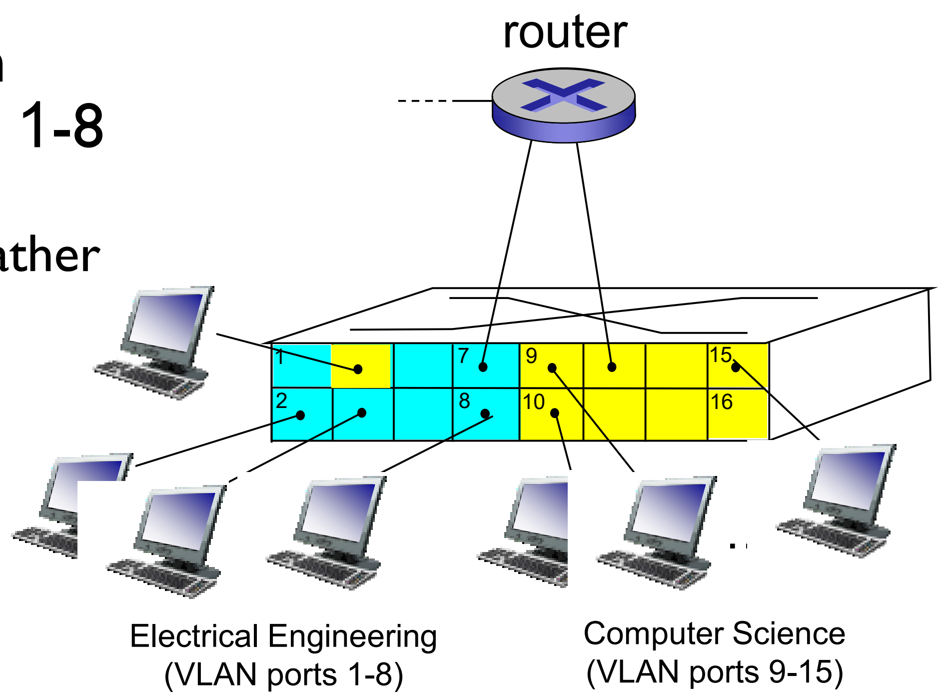


… operates as multiple virtual switches

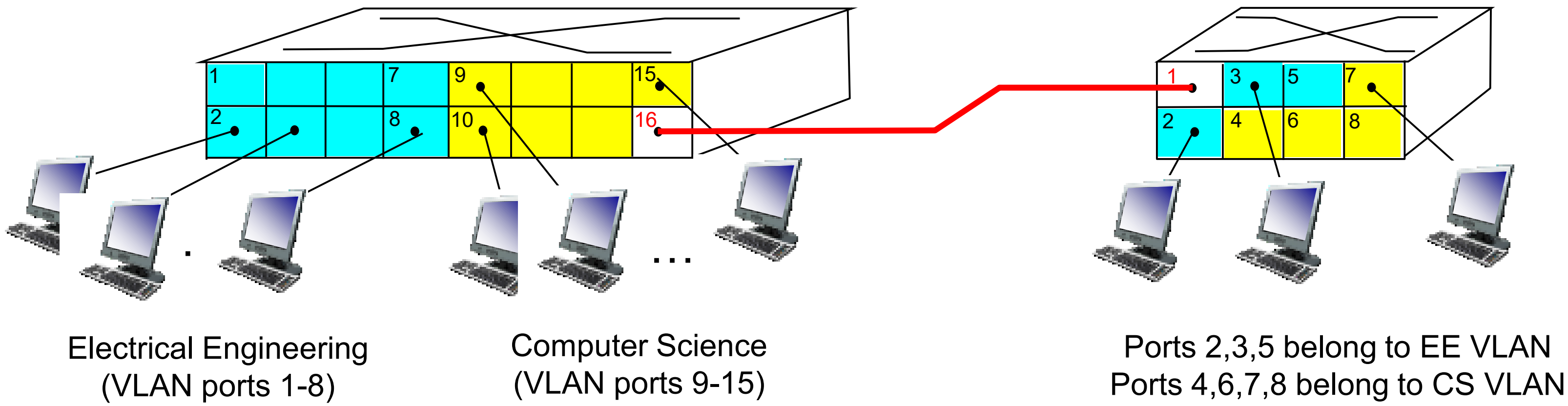


Port-based VLAN

- *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

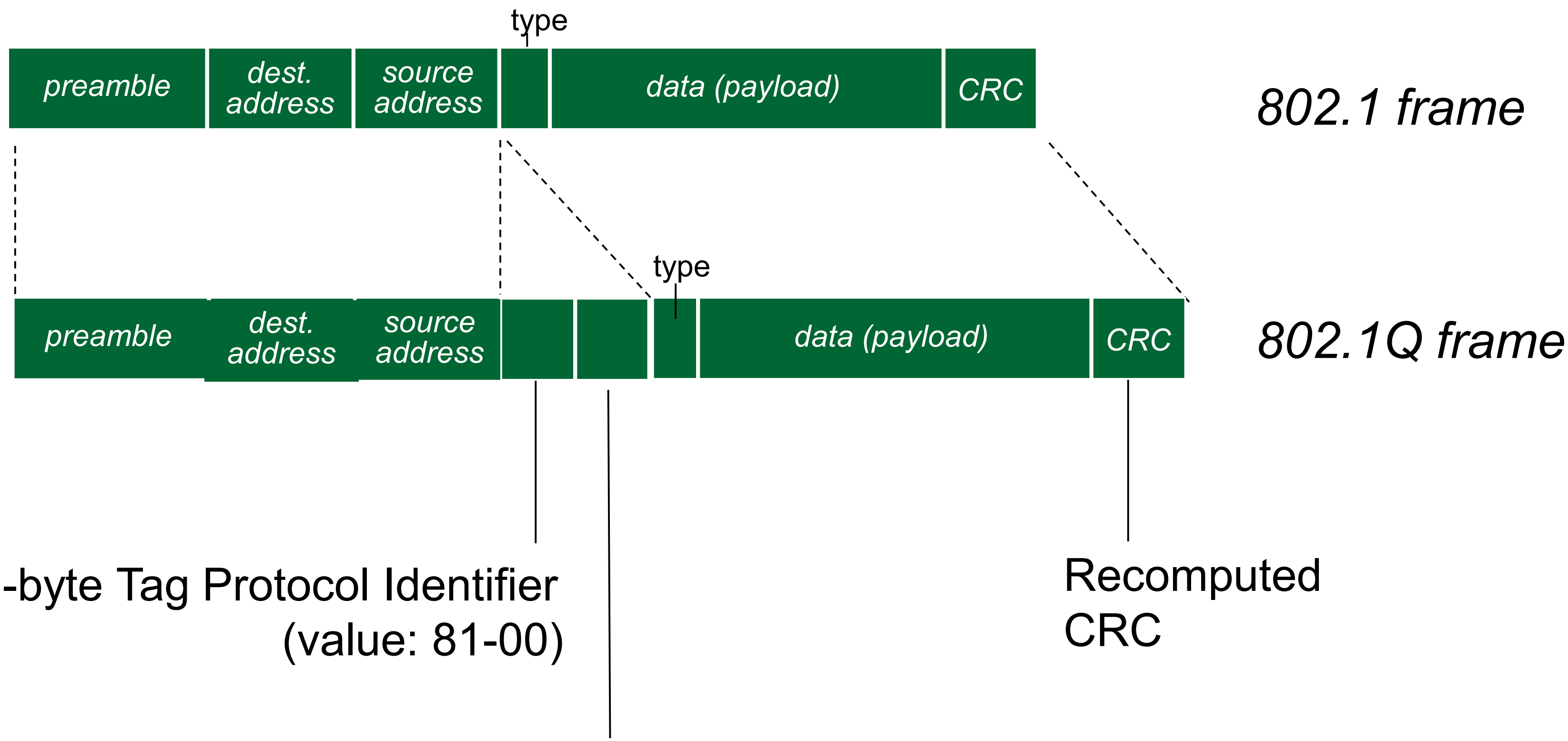


VLANs spanning multiple switches



- *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/removes additional header fields for frames forwarded between trunk ports

802.1Q VLAN frame format



2-byte Tag Protocol Identifier (value: 81-00)

Recomputed CRC

Tag Control Information

(12 bit VLAN ID field, 3 bit priority field like IP TOS)

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

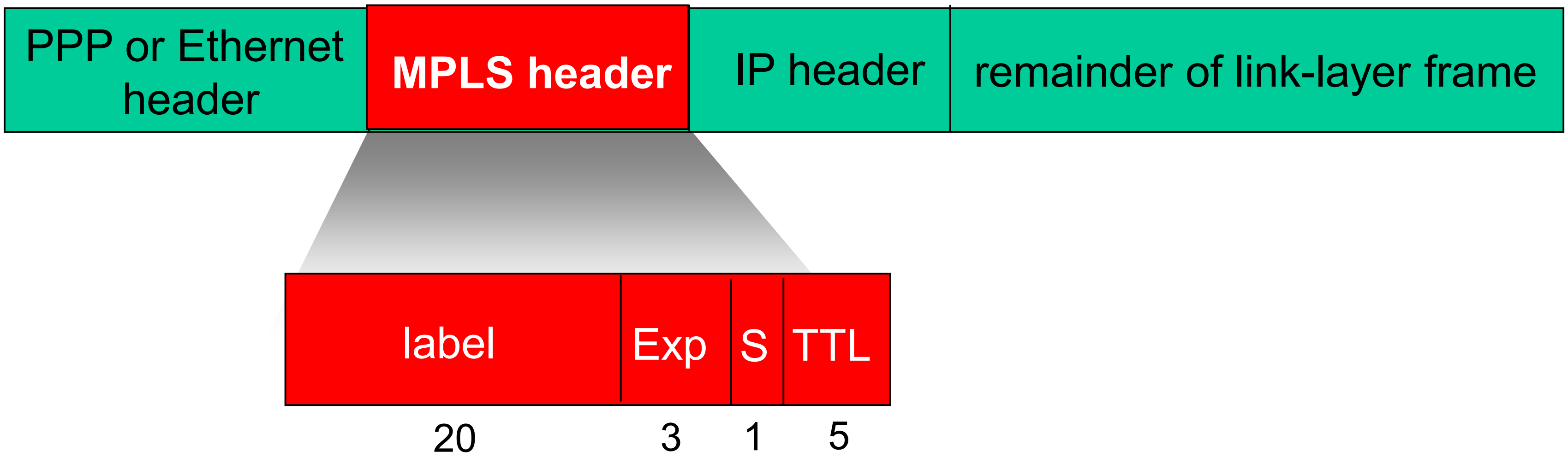
6.6 data center networking

6.7 a day in the life of a web request

Additional Material

MultiProtocol Label Switching (MPLS)

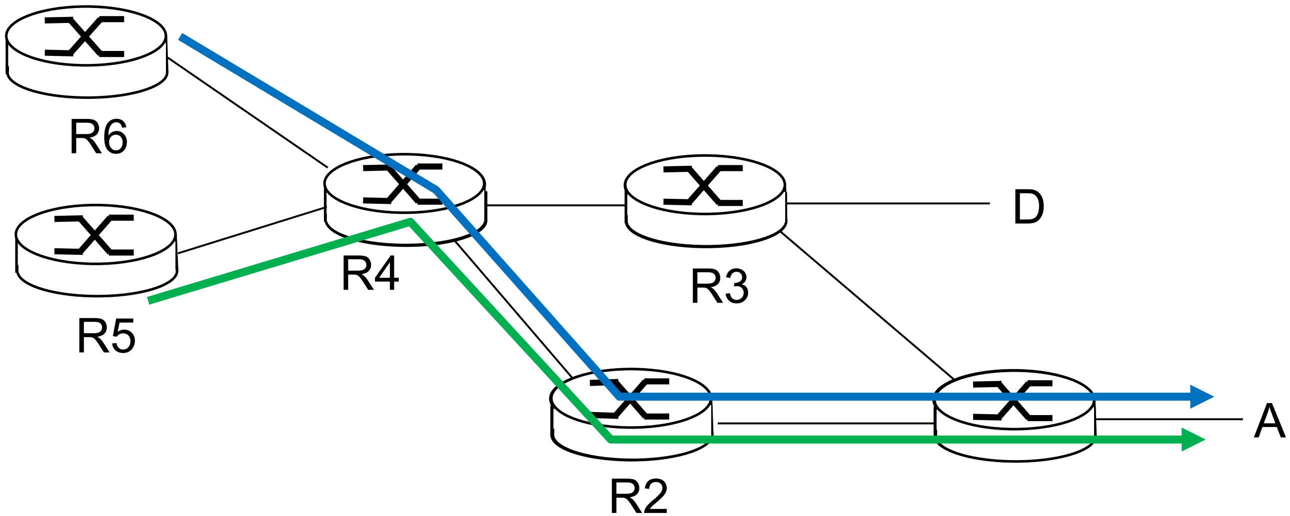
- Initial goal: high-speed IP forwarding using fixed length label (instead of IP address)
  - fast lookup using fixed length identifier (rather than shortest prefix matching)
  - borrowing ideas from Virtual Circuit (VC) approach
  - but IP datagram still keeps IP address!



MPLS capable routers

- a.k.a. label-switched router
- forward packets to outgoing interface based only on label value (*don't inspect IP address*)
  - MPLS forwarding table distinct from IP forwarding tables
- *flexibility:* MPLS forwarding decisions can *differ* from those of IP
  - use destination *and* source addresses to route flows to same destination differently (traffic engineering)
  - re-route flows quickly if link fails: pre-computed backup paths (useful for VoIP)

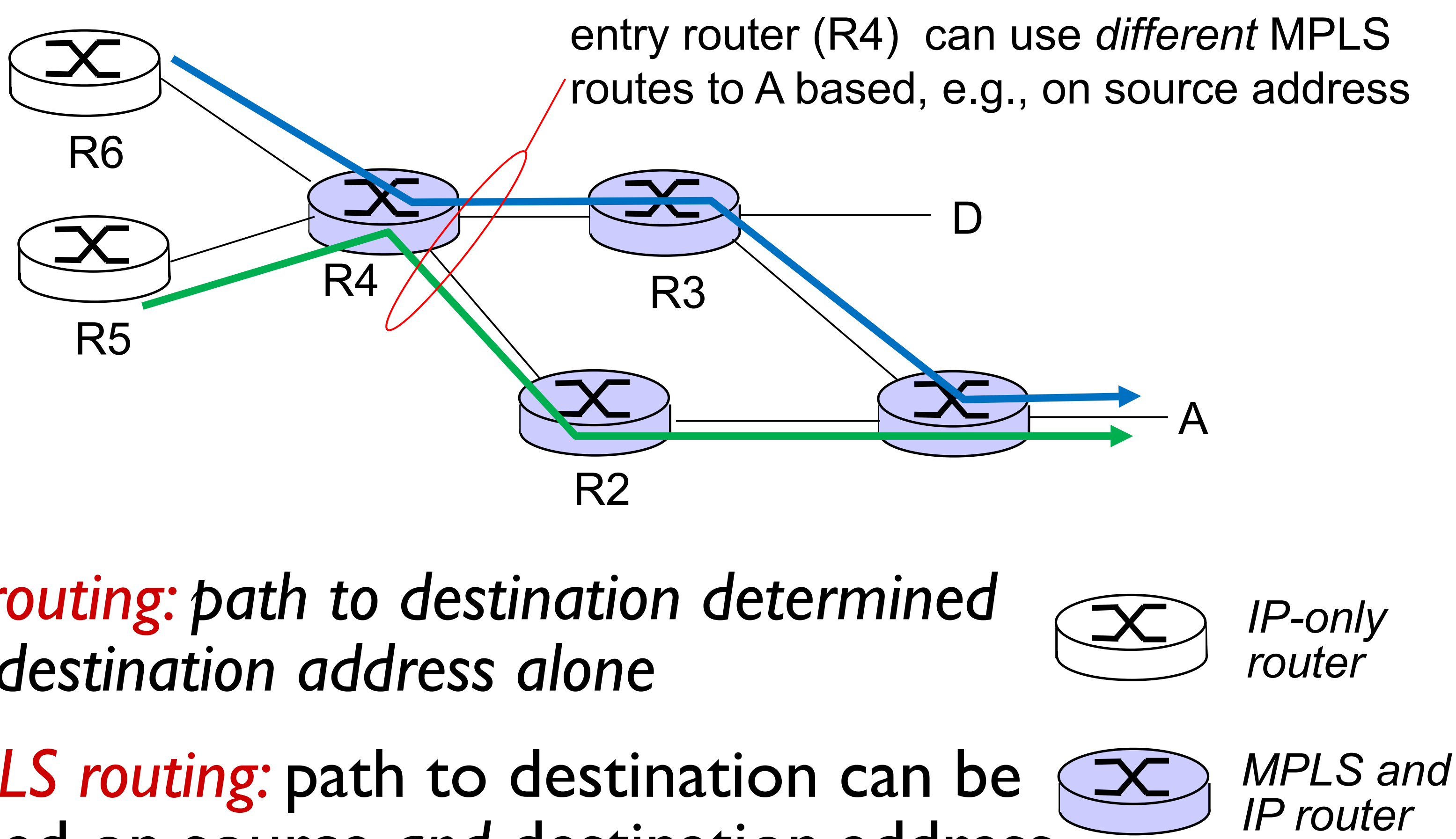
MPLS versus IP paths



- *IP routing: path to destination determined by destination address alone*

Additional Material

MPLS versus IP paths

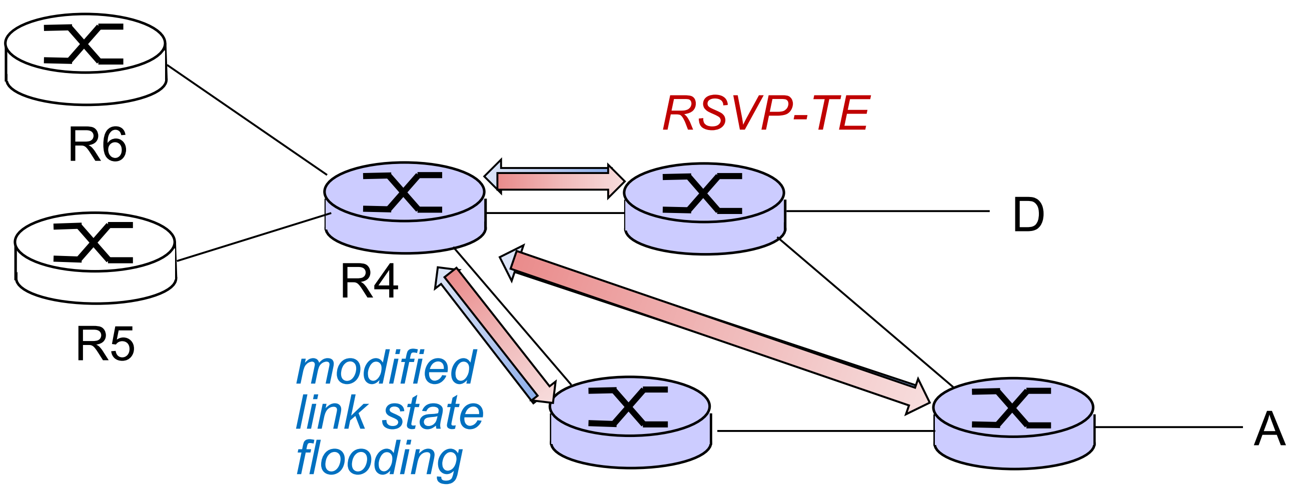


- *IP routing:* *path to destination determined by destination address alone*
- *MPLS routing:* path to destination can be based on source *and* destination address
  - *fast reroute:* precompute backup routes in case of link failure

Additional Material

MPLS signaling

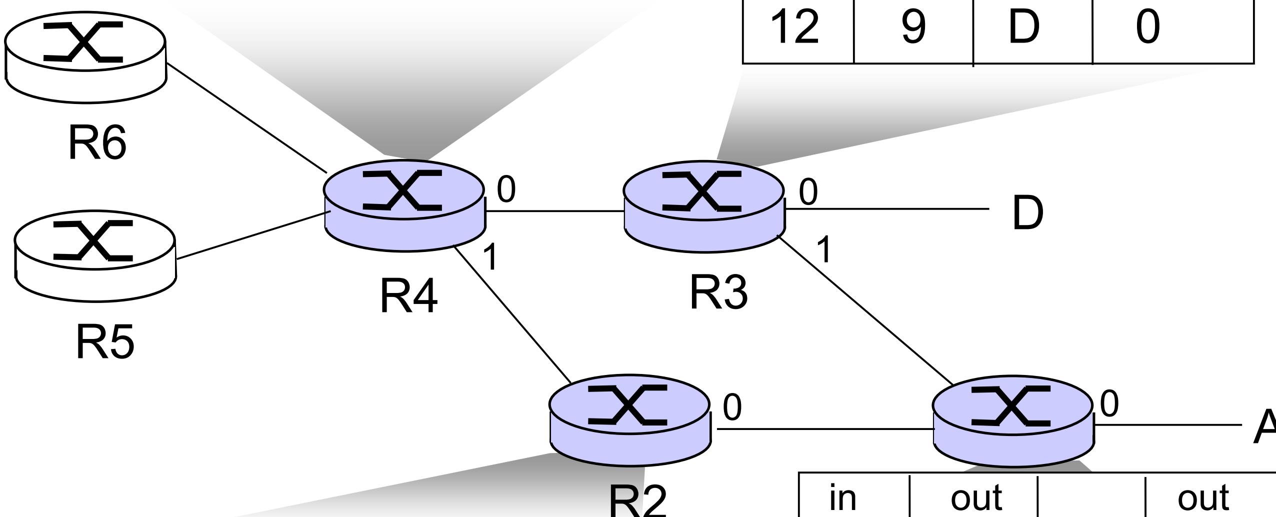
- modify OSPF, IS-IS link-state flooding protocols to carry info used by MPLS routing,
  - e.g., link bandwidth, amount of "reserved" link bandwidth
- *entry MPLS router uses RSVP-TE signaling protocol to set up MPLS forwarding at downstream routers*



MPLS forwarding tables

in label	out label	dest	out interface
	10	A	0
	12	D	0
	8	A	1

in label	out label	dest	out interface
10	6	A	1
12	9	D	0



in label	out label	dest	out interface
8	6	A	0

in label	out label	dest	out interface
6	-	A	0

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

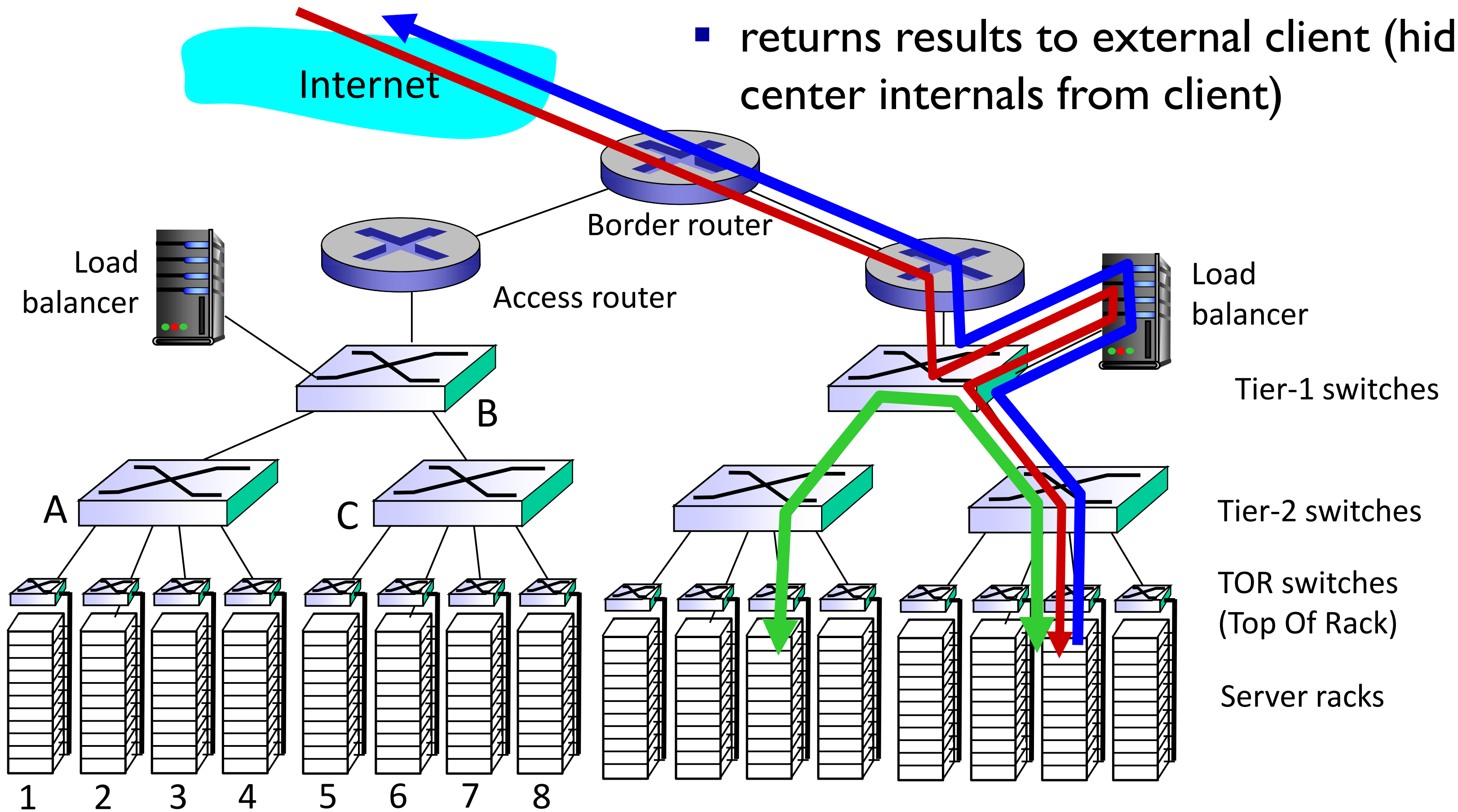
- 10's to 100's of thousands of hosts, often closely coupled, in close proximity:
  - e-business (e.g., Amazon)
  - content-servers (e.g., YouTube, Akamai, Apple, Microsoft)
  - search engines, data mining (e.g., Google)
- Challenges:
  - multiple applications, each serving massive numbers of clients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


Inside a 40-ft Microsoft container, Chicago data center

Data center networks

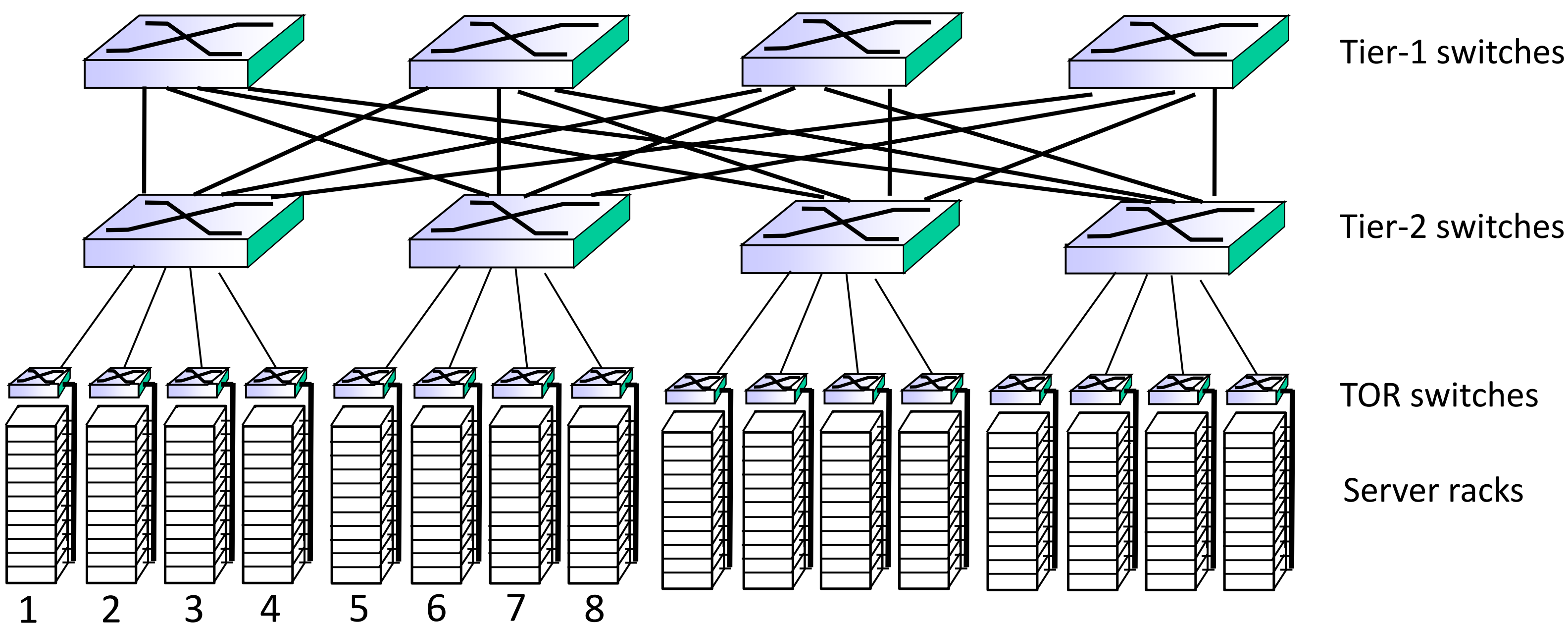
Load balancer: application-layer routing

- 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

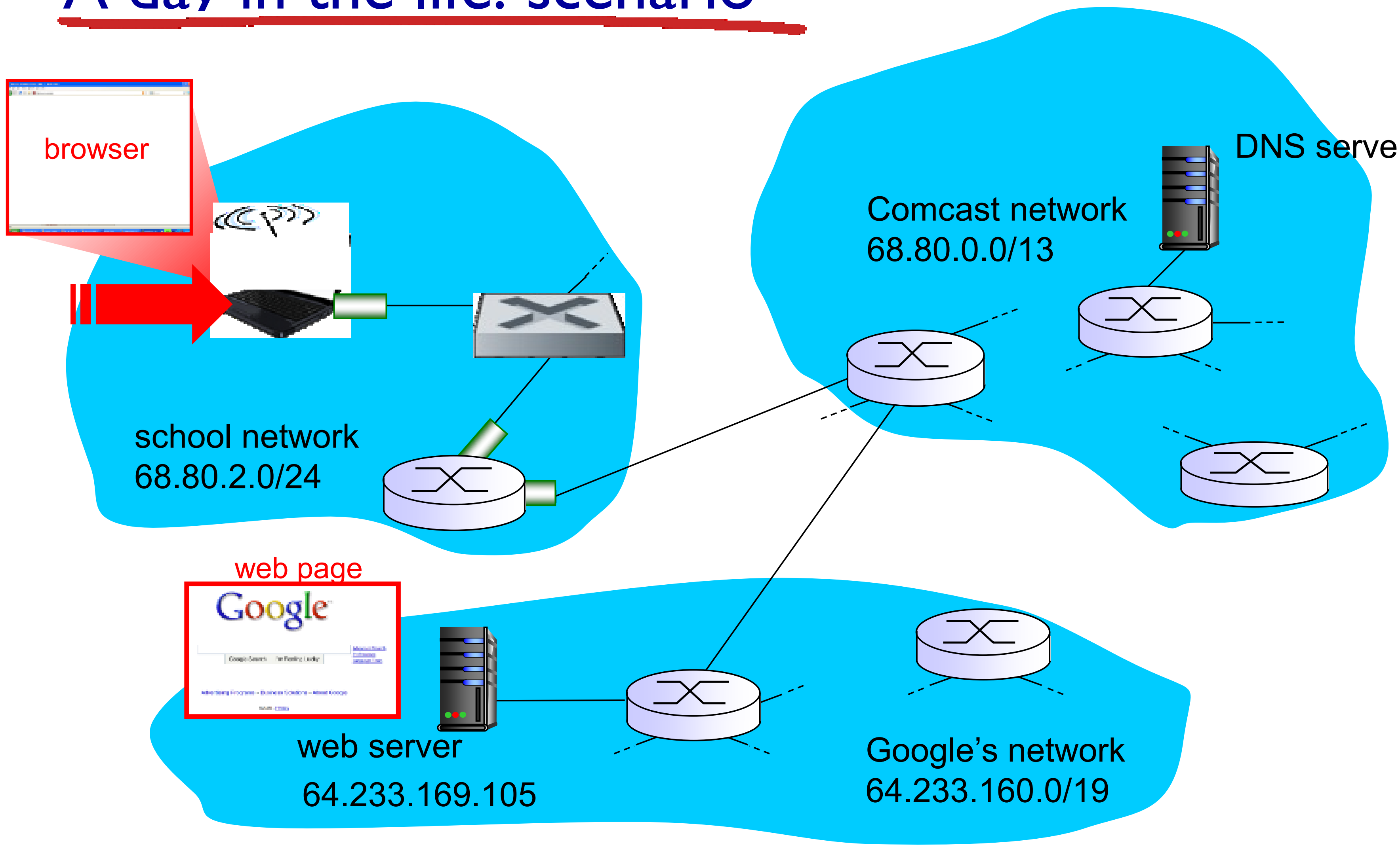
6.1 introduction, services
6.2 error detection, correction
6.3 multiple access protocols
64 LANs
- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS
6.6 data center networking
6.7 a day in the life of a web request

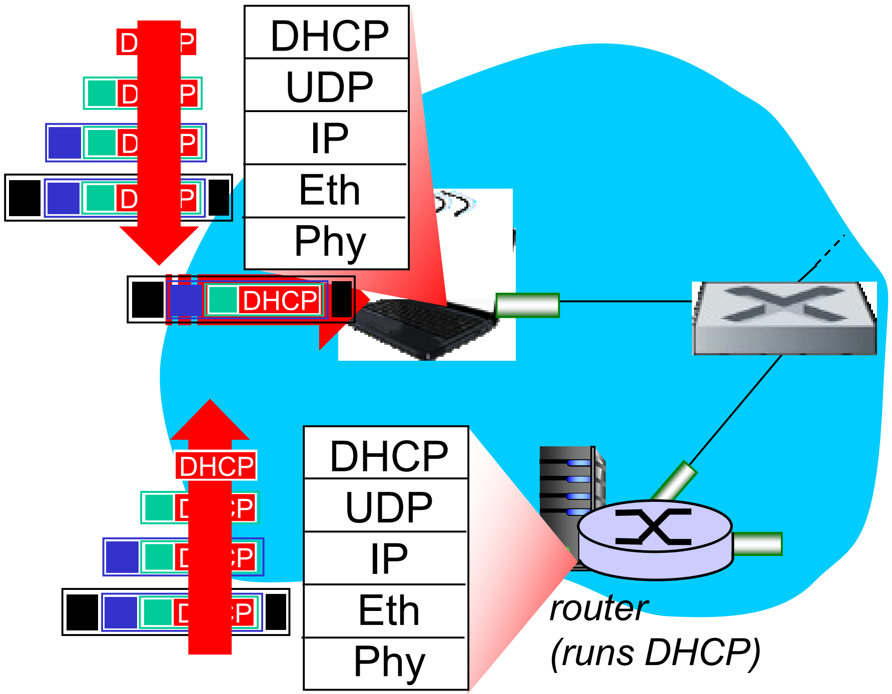
Synthesis: a day in the life of a web request

- 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 a www page
  - *scenario:* student attaches laptop to campus network, requests/receives www.google.com

A day in the life: scenario

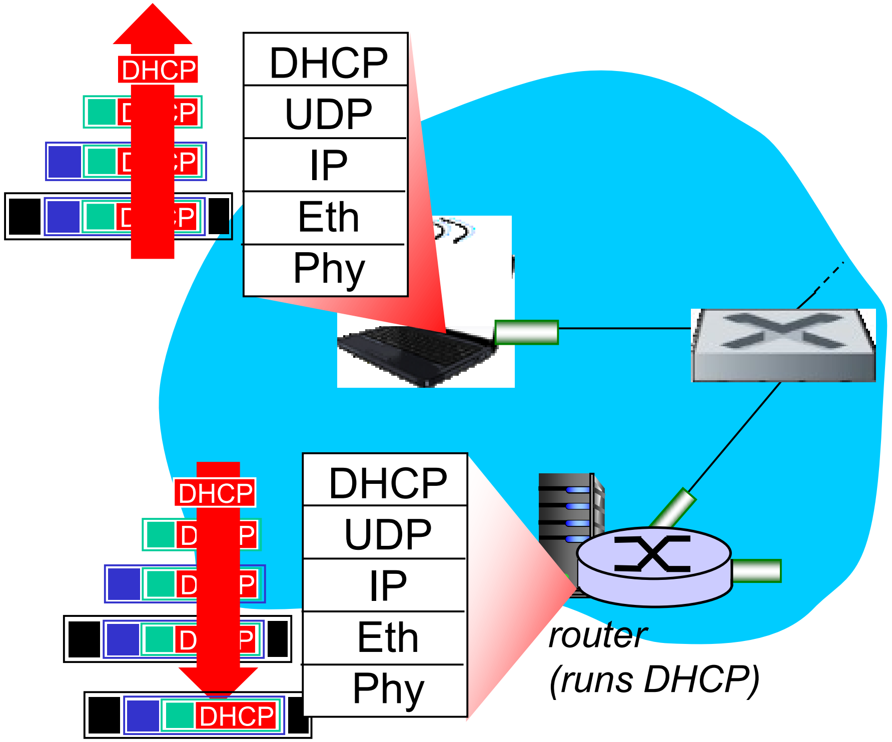


A day in the life… connecting to the Internet



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

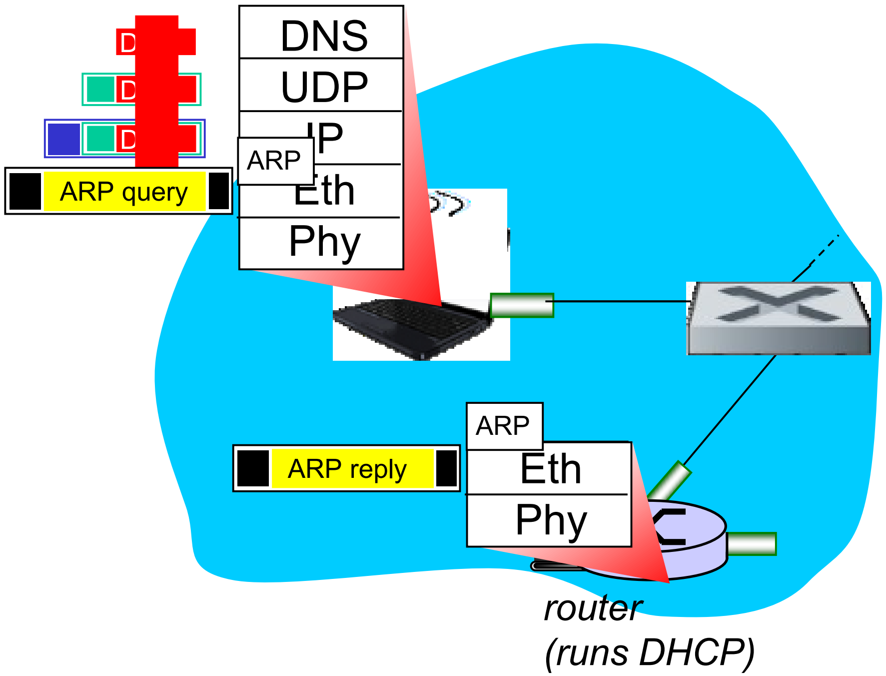
A day in the life… connecting to the Internet



- 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

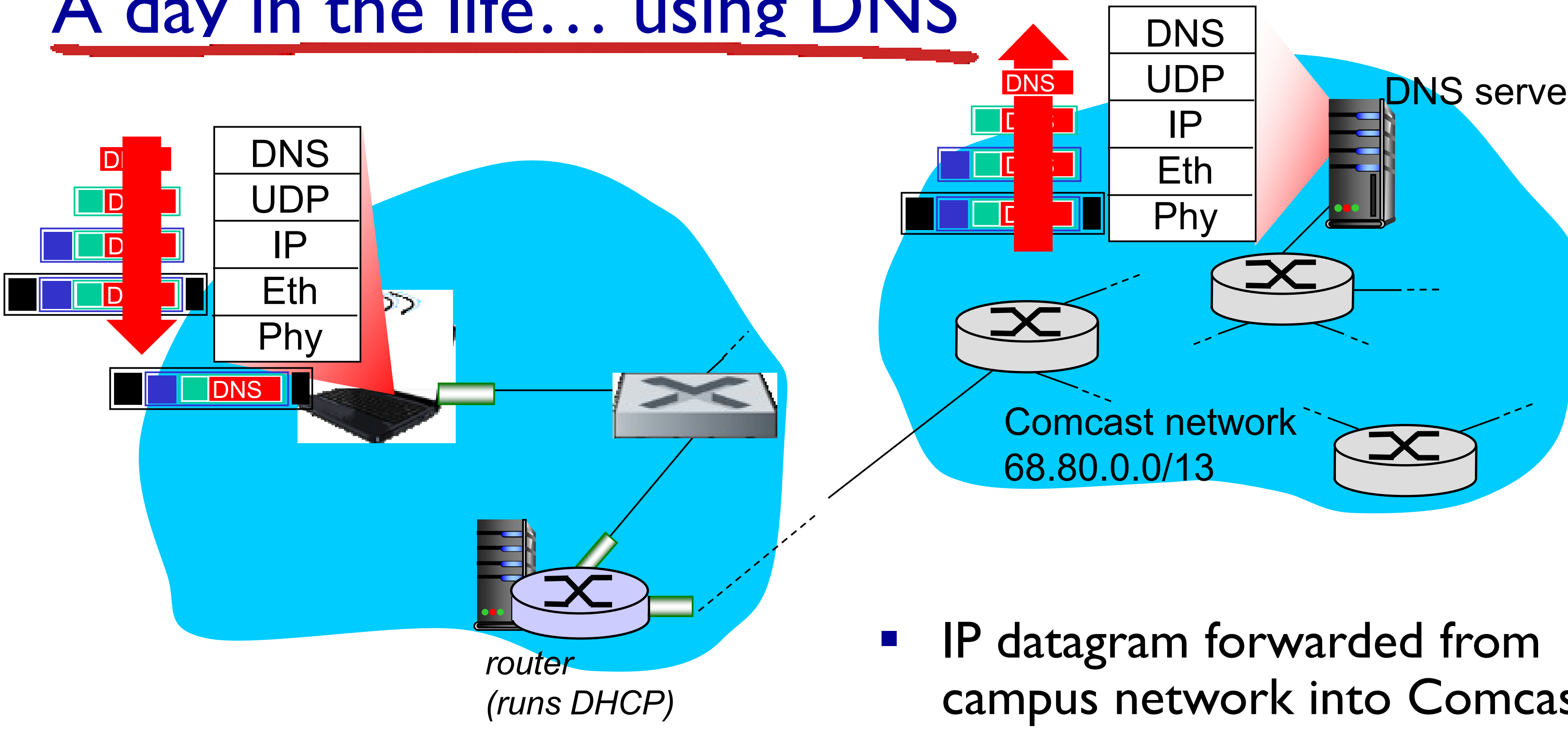
Client now has IP address, knows name & addr of DNS server, knows IP address of its first-hop router

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



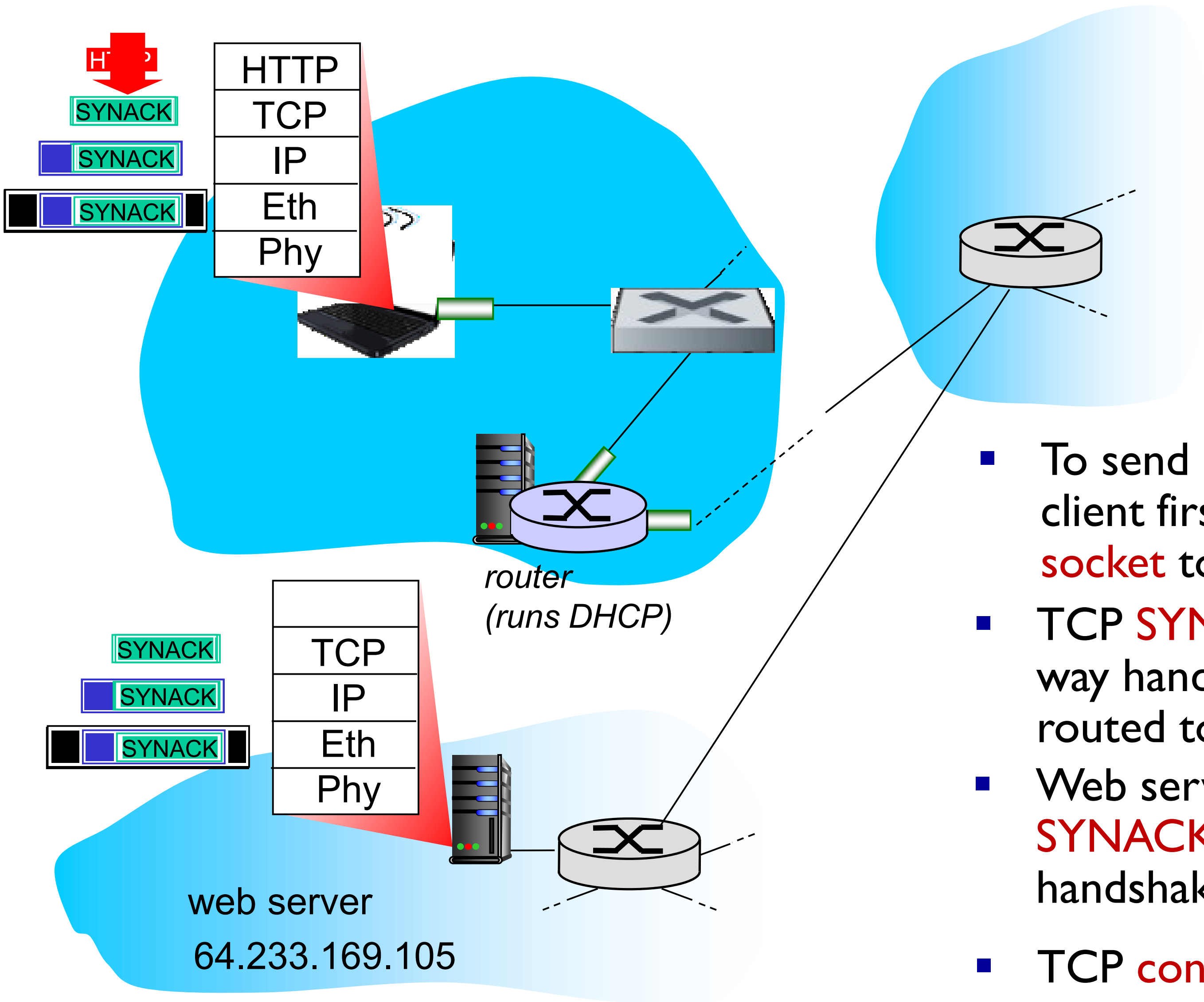
- before sending *HTTP* request, need IP address of www.google.com: *DNS*
- DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. To send frame to router, need MAC address of router interface: ARP
- ARP query broadcast, received by router, which replies with ARP reply giving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



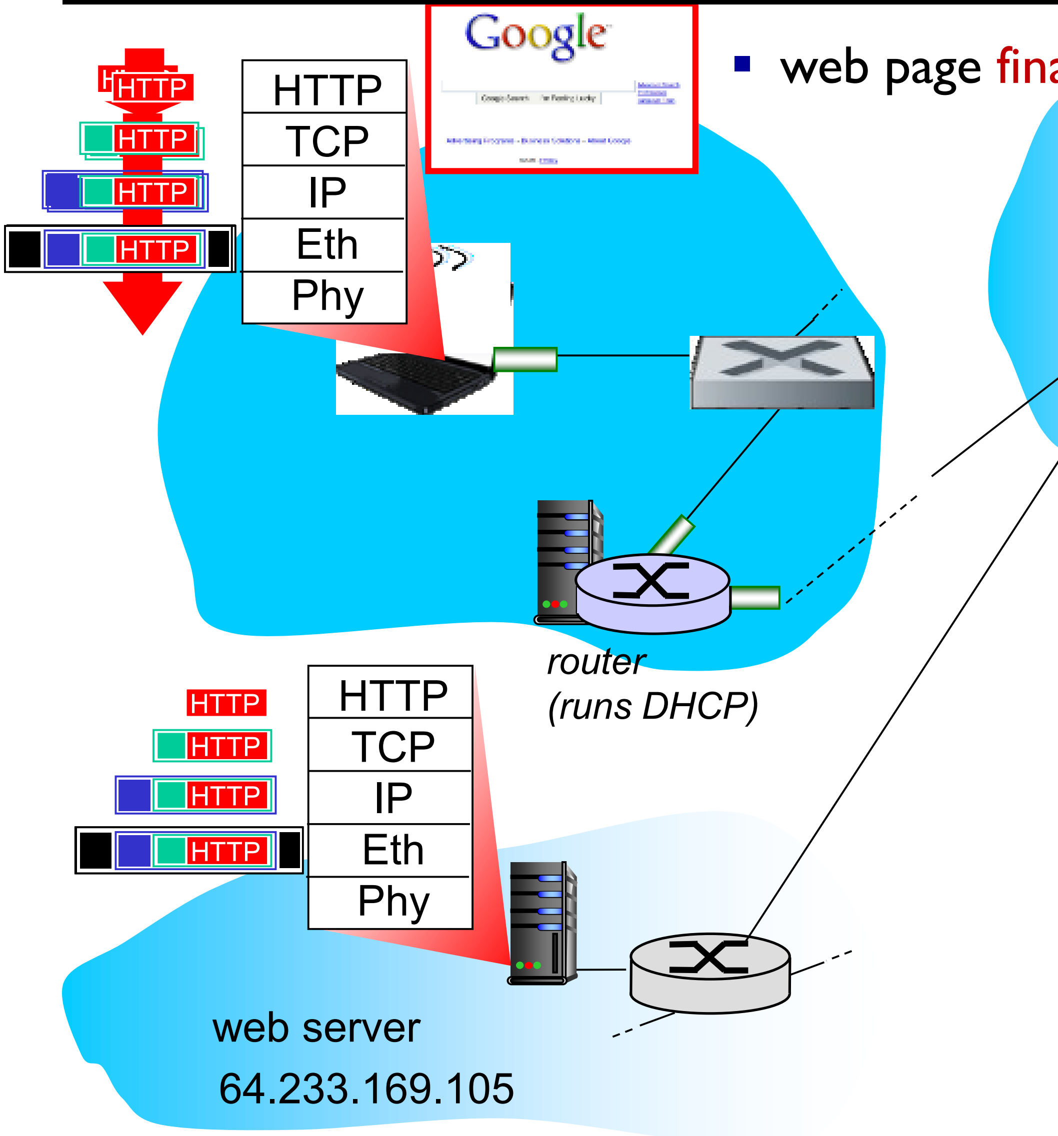
- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router
- IP datagram forwarded from campus network into Comcast network, routed (tables previously created by RIP, OSPF, IS-IS and/or BGP routing protocols) to DNS server
- Demuxed at DNS server
- DNS server replies to 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(s) 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