

Redes de Comunicação 2023/2024

T05 Link layer

Jorge Granjal
University of Coimbra



T05: outline

5.1 introduction, services

5.2 multiple access
protocols

5.3 LANs

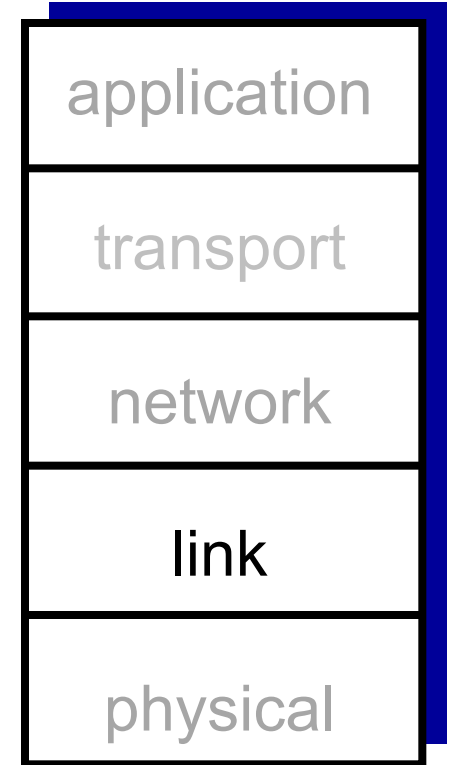
- addressing, ARP
- Ethernet
- switches

5.4 data center
networking

5.5 a day in the life of a
web request

Internet (TCP/IP) protocol stack

- *application*: supporting network applications
 - FTP, SMTP, HTTP, ...
- *transport*: process-process data transfer
 - TCP, UDP
- *network*: routing of datagrams from source to destination
 - IP, routing protocols
- *link*: data transfer between neighboring network elements
 - Ethernet, 802.11 (WiFi), ...
- *physical*: bits “on the wire”

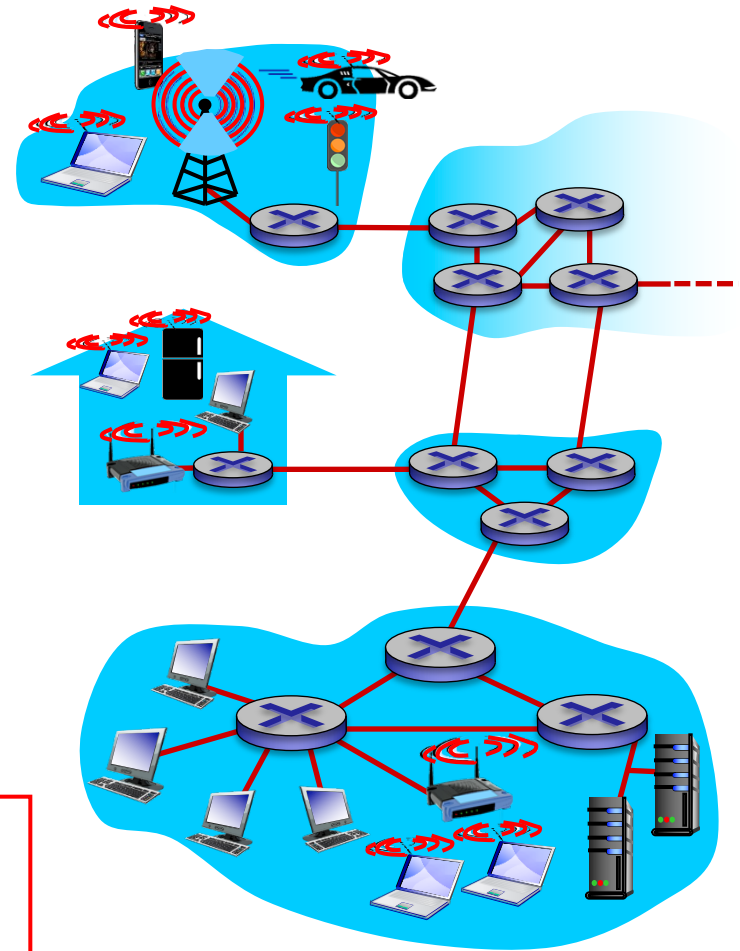


Link layer: introduction

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

- datagram transferred by *different link protocols* over *different links*:
 - e.g., Ethernet on first link, frame relay on intermediate links, 802.11 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:*

- encapsulate (network layer) datagram into frame, adding header, trailer
- Medium Access Control (MAC) protocol defines rules for channel access if shared medium
- “MAC” addresses used in frame headers to identify source, destination
 - different from IP address!

■ *reliable delivery between adjacent nodes*

- we learned how to do this already (network layer)!
- seldom used on low bit-error link (fiber, some twisted pair)
- wireless links: high error rates
 - *Q*: why both link-level and end-end reliability?

Link layer services (more)

■ *flow control:*

- pacing between adjacent sending and receiving nodes

■ *error detection:*

- errors caused by signal attenuation, noise.
- more sophisticated than at network and transport layer (Internet checksum) and implemented at the hardware
- receiver detects presence of errors:
 - transmitting node includes error-detection bits in frame, receiving node performs an error check
 - signals sender for retransmission or drops frame

■ *error correction:*

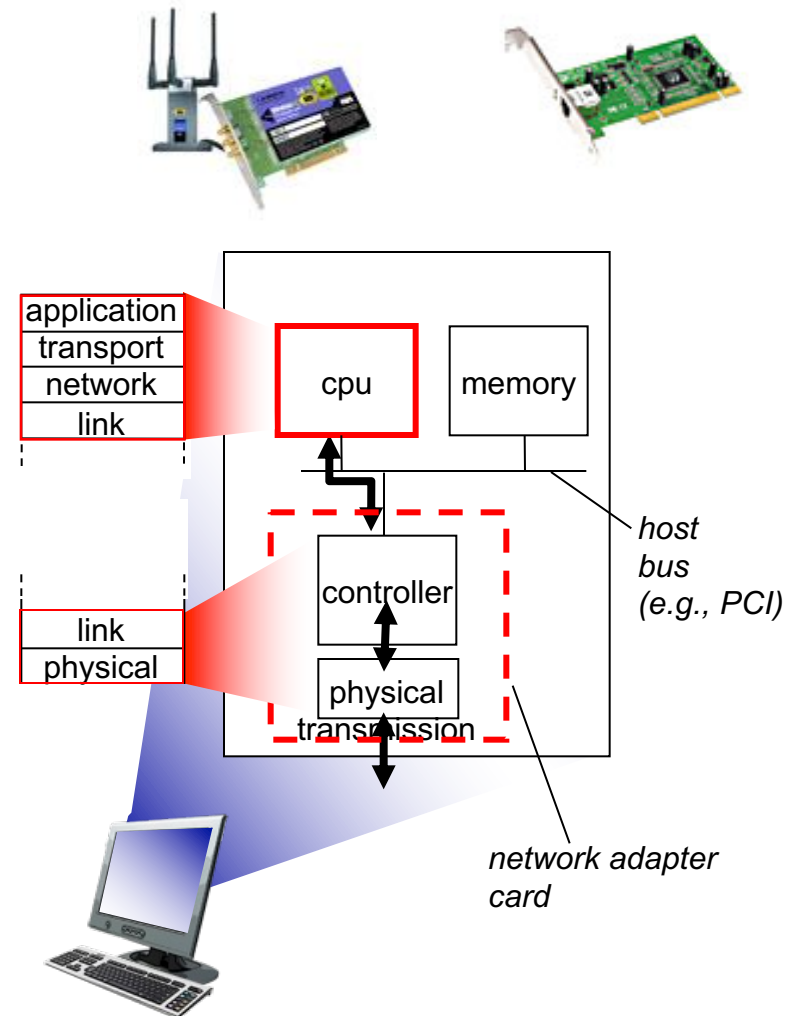
- receiver identifies *and corrects* bit error(s) without resorting to retransmission

■ *half-duplex and full-duplex*

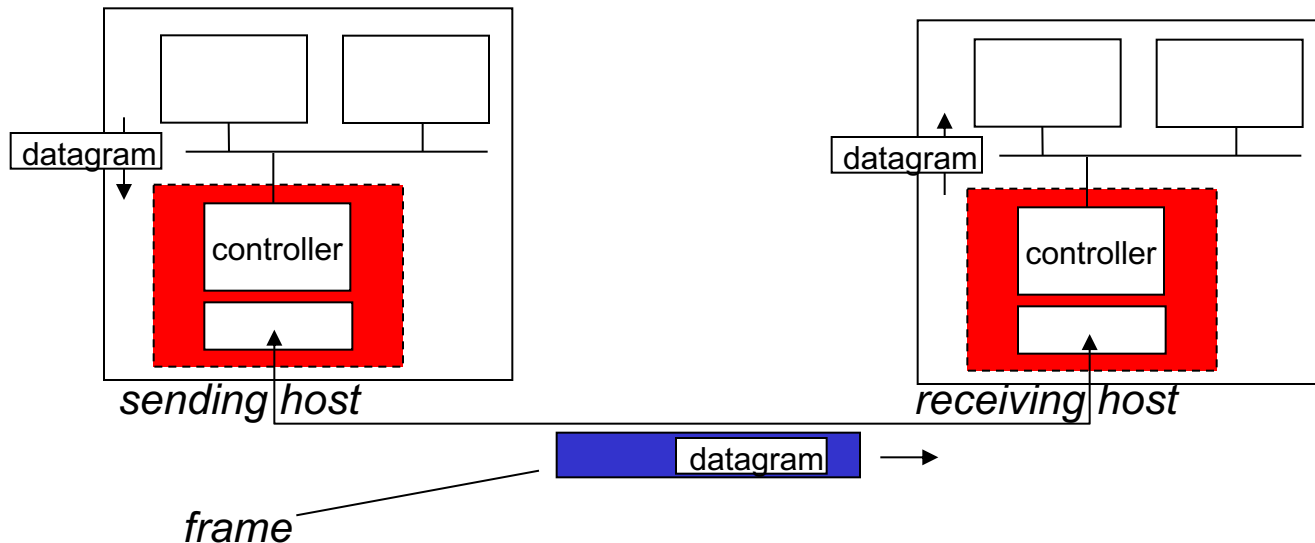
- with half duplex, nodes at both ends of link can transmit, but not at same time

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, 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 datagram in frame
 - adds error checking bits, flow control, etc.
- receiving side
 - looks for errors, flow control, etc.
 - extracts datagram, passes to upper layer at receiving side

T05: outline

5.1 introduction, services

5.2 multiple access
protocols

5.3 LANs

- addressing, ARP
- Ethernet
- switches

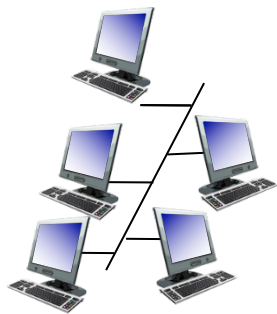
5.4 data center
networking

5.5 a day in the life of a
web request

Multiple access links, protocols

two types of “links”:

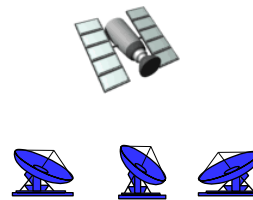
- point-to-point
 - PPP for dial-up access
 - point-to-point link between Ethernet switch, host
- ***broadcast (shared wire or medium)***
 - **broadcast**: when a node transmits a frame each of the other nodes in the link receives a copy
 - old-fashioned Ethernet
 - 802.11 wireless LAN



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
- two or more simultaneous transmissions by nodes: interference
 - *collision* if node receives two or more signals at the same time
 - bandwidth is wasted during collisions

multiple access protocol

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

MAC protocols: taxonomy

three broad classes:

- *channel partitioning*

- divide channel into smaller “pieces”: time-division multiplexing (TDM), frequency-division multiplexing (FDM), code division multiplexing (CDMA)
- allocate piece to node for exclusive use

- *random access*

- channel not divided, allow collisions
- “recover” from collisions

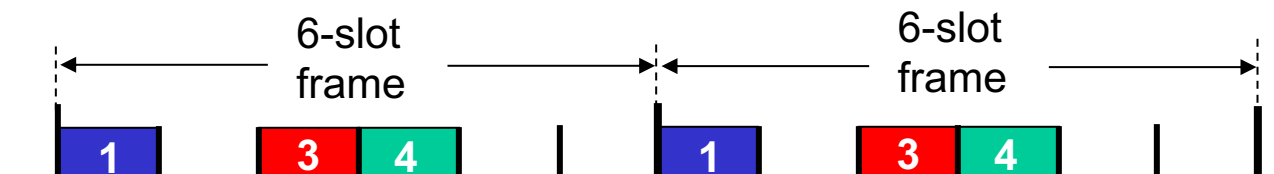
- *“taking turns”*

- nodes take turns, but nodes with more to send can take longer turns

Channel partitioning MAC protocols: TDMA

TDMA: time division multiple access

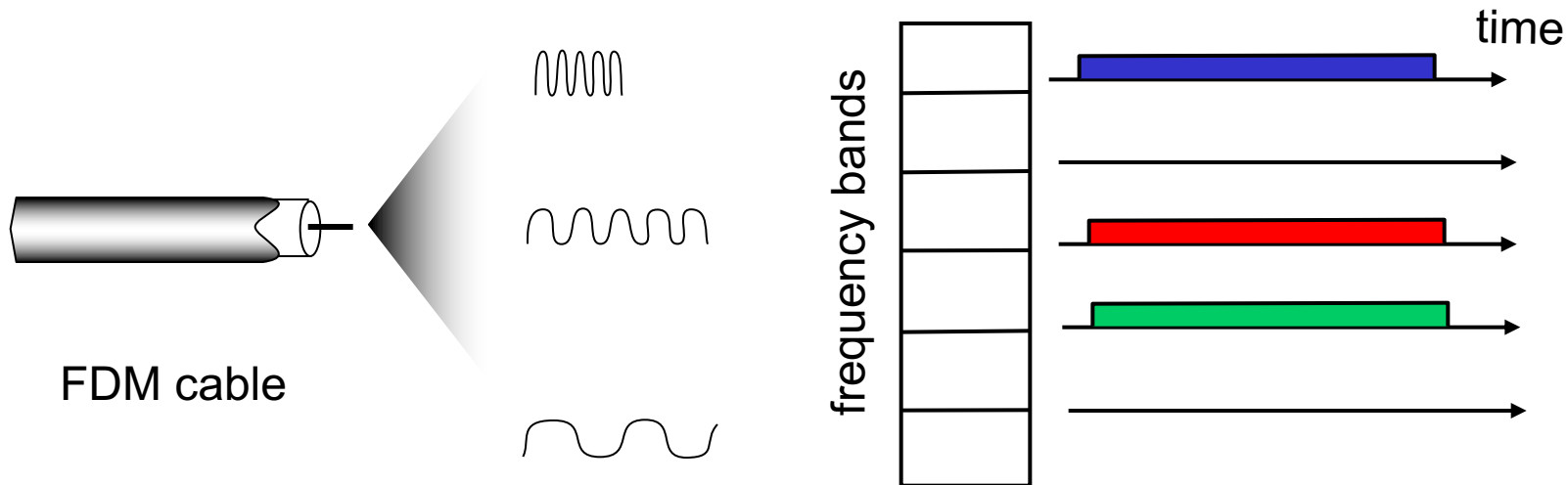
- access to channel in "rounds"
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle
- eliminates collisions and is perfectly fair, but..
- limited transmission rate per node even if other nodes are not transmitting, node always waits for its turn to transmit



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

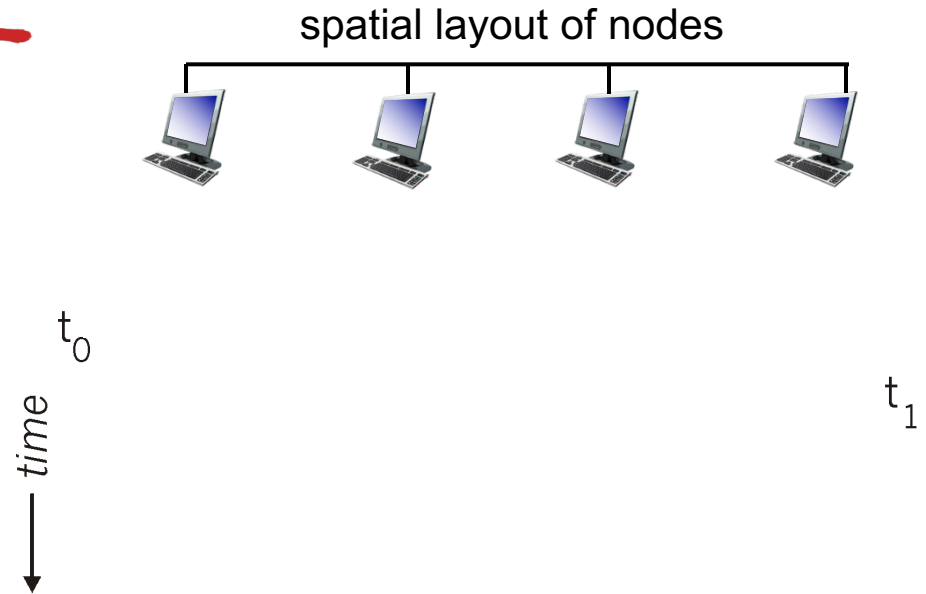


Random access protocols

- when node has packet to send
 - transmit 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): *each node selects an independent random delay before retransmitting*
- examples of random access MAC protocols (hundreds are available in the literature!):
 - slotted ALOHA
 - ALOHA
 - CSMA, CSMA/CD, CSMA/CA

CSMA collisions

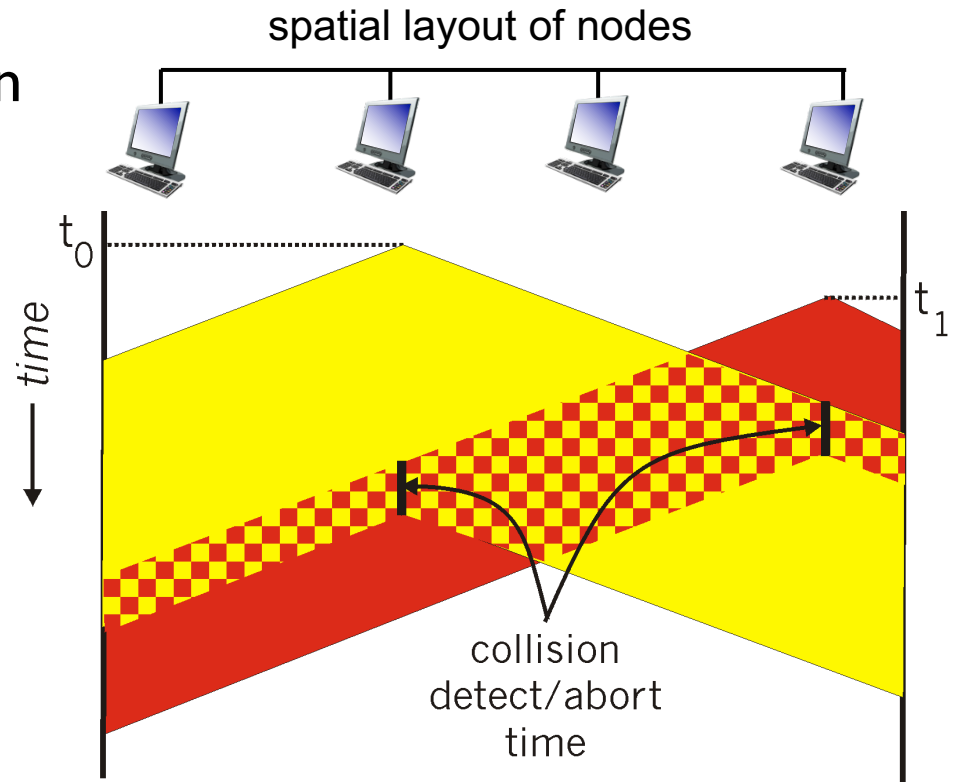
- collisions *can* still occur:
propagation delay means
two nodes may not hear
each other's
transmission
- collision: entire packet
transmission time
wasted
 - distance &
propagation delay
play role in
determining collision
probability



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:
measure signal strengths,
compare transmitted,
received signals
- May be used in wired networks (e.g. Ethernet)
 - Very hard in wireless networks (Wi-Fi 802.11): use CSMA/CA



Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission. If NIC senses channel busy, *waits until channel 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
5. After aborting, NIC enters *binary (exponential) backoff*:
 - after m th collision, NIC chooses K at random from $\{0, 1, 2, \dots, 2^m - 1\}$. NIC waits $K \cdot 512$ bit times, *returns to Step 2*
 - longer backoff interval with more collisions

“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

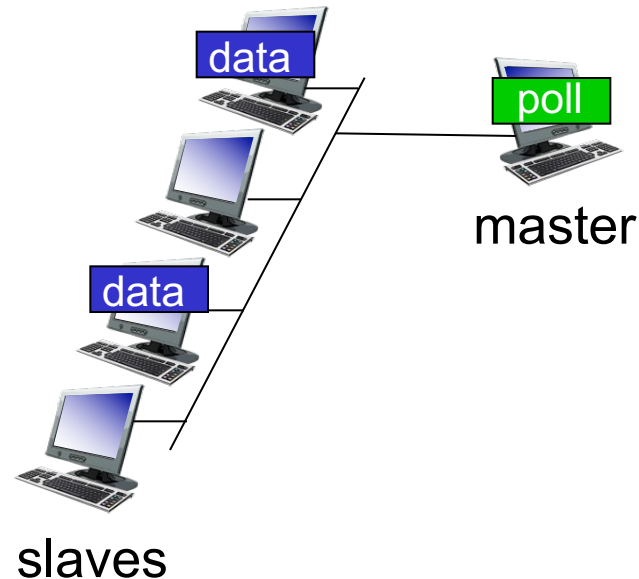
“taking turns” protocols

look for best of both worlds!

“Taking turns” MAC protocols

polling:

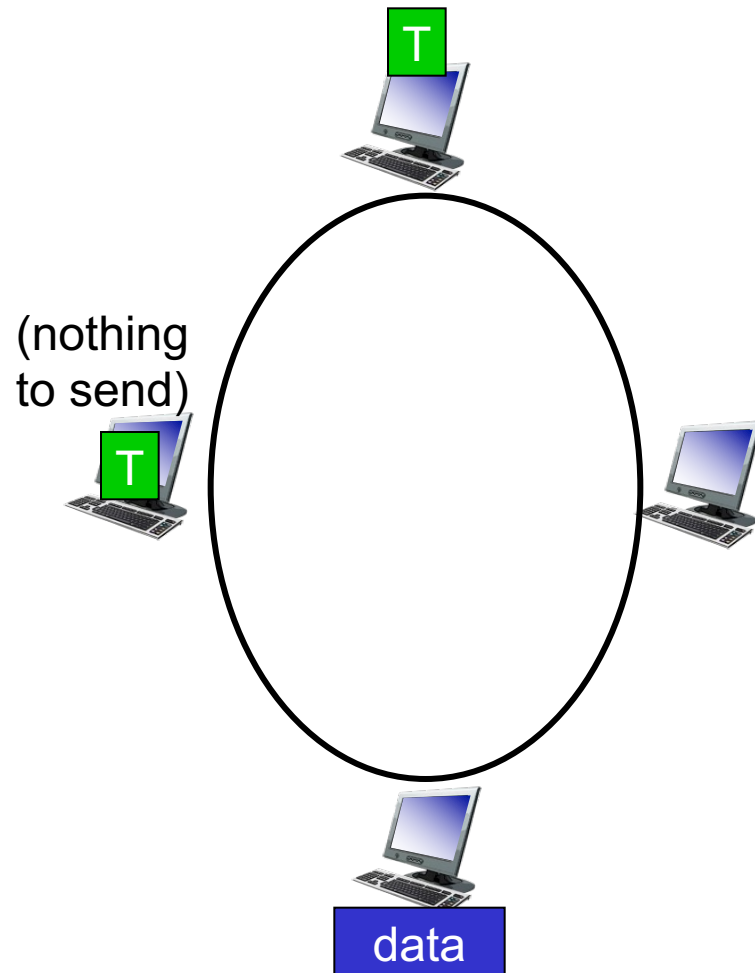
- master node “invites” slave nodes to transmit in turn
- typically used with “dumb” slave devices
- concerns:
 - polling overhead
 - latency
 - single point of failure (master)



“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)



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 802.11
- *taking turns*
 - polling from central site, token passing
 - Bluetooth, FDDI, token ring

T05: outline

5.1 introduction, services

5.2 multiple access
protocols

5.3 LANs

- addressing, ARP
- Ethernet
- switches

5.4 data center
networking

5.5 a day in the life of a
web request

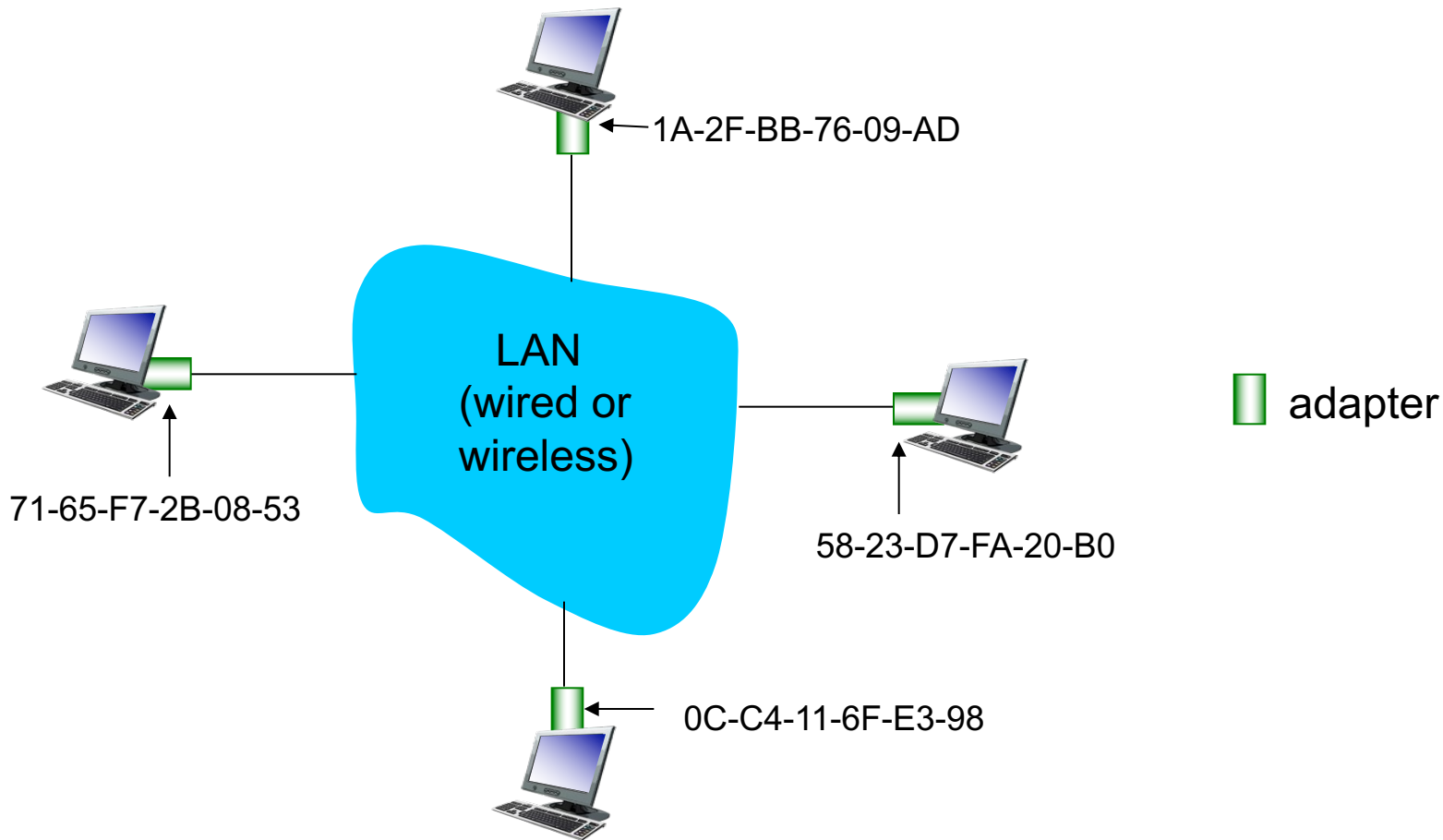
MAC addresses and ARP

- 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

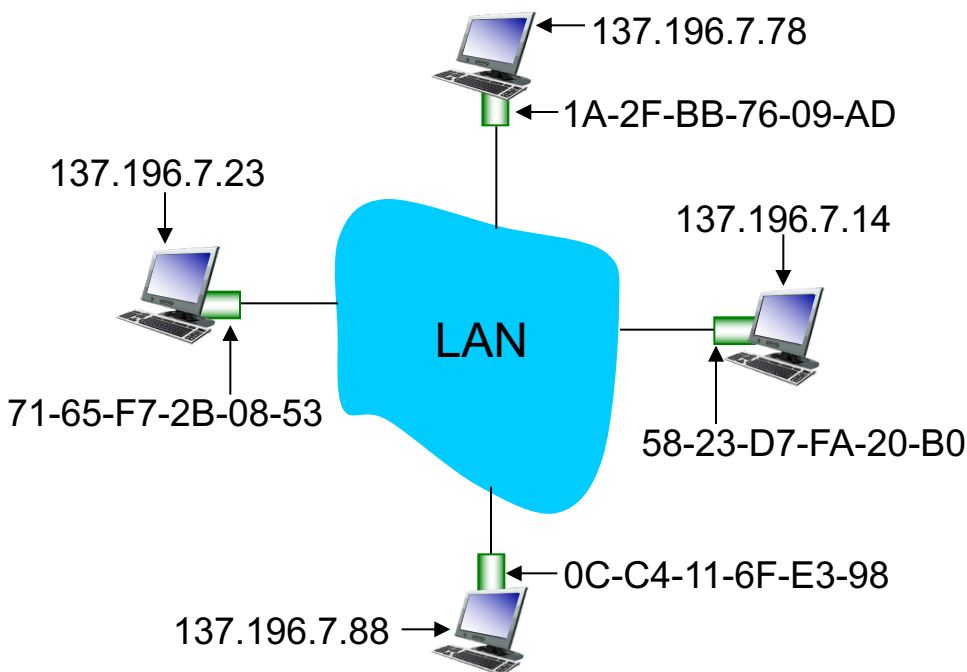


LAN addresses (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- 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)

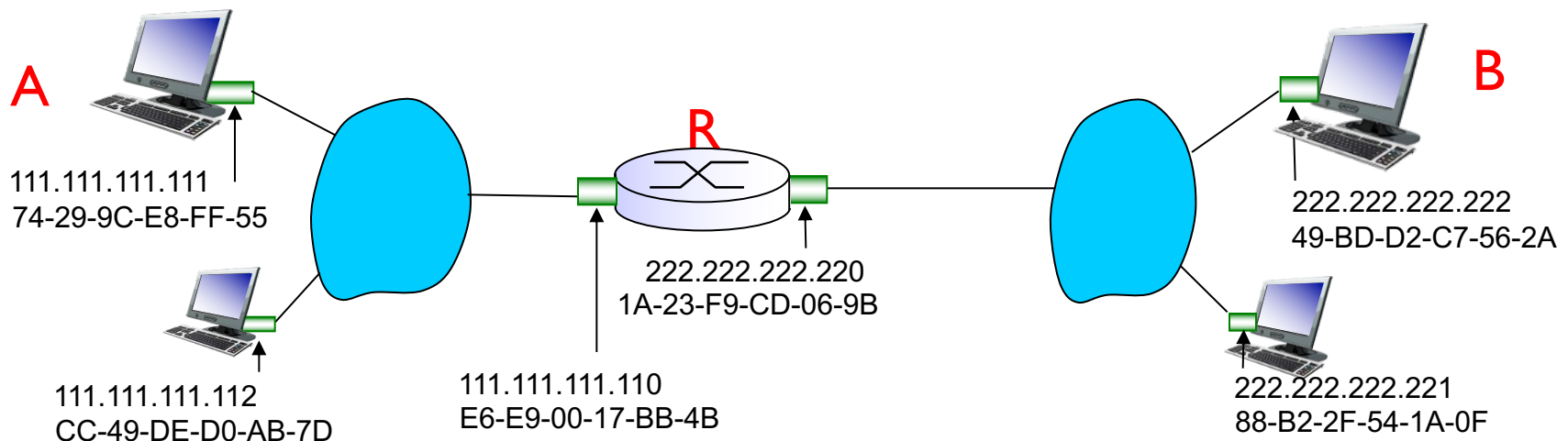
ARP protocol: same LAN

- 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

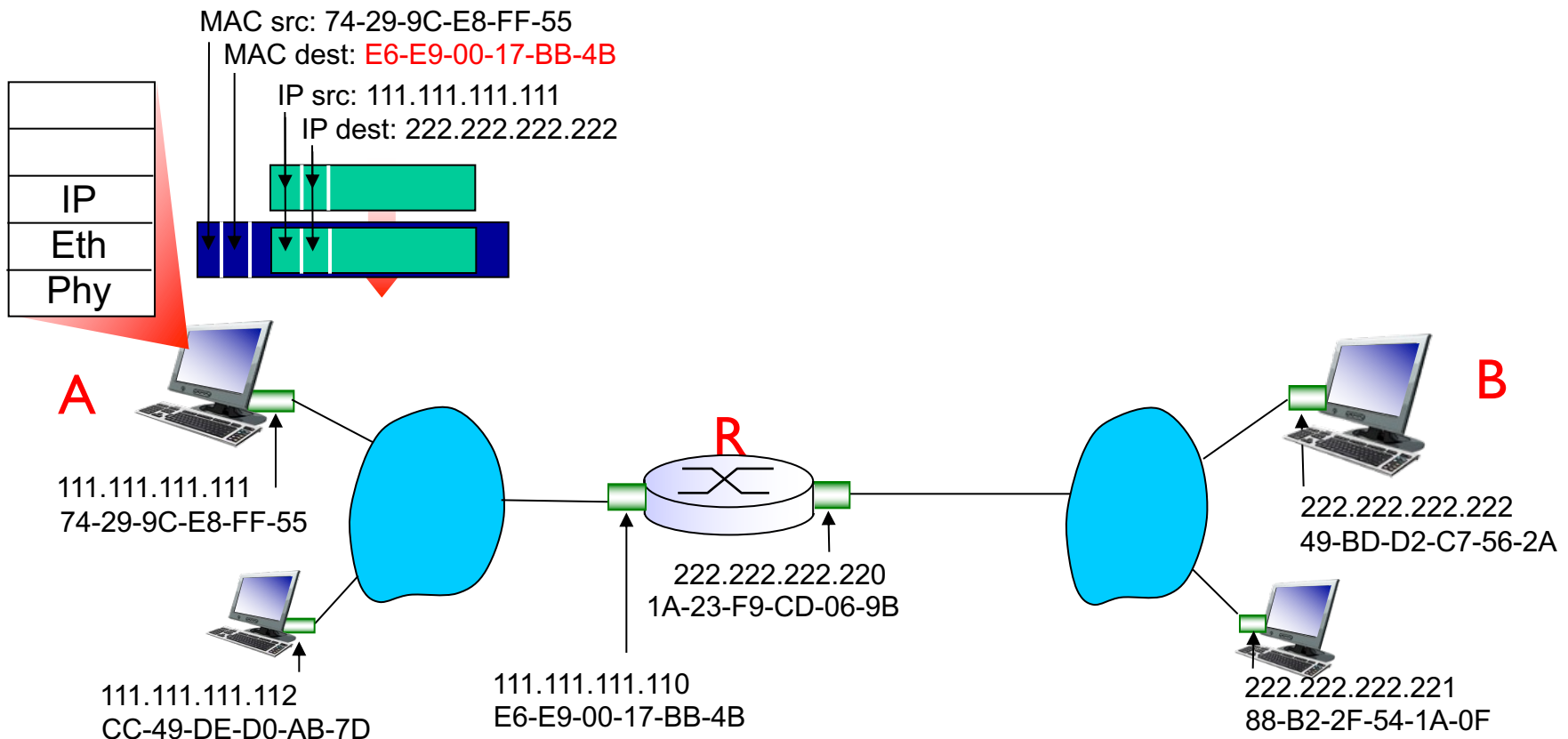
walkthrough: **send datagram from A to B via R**

- 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?)



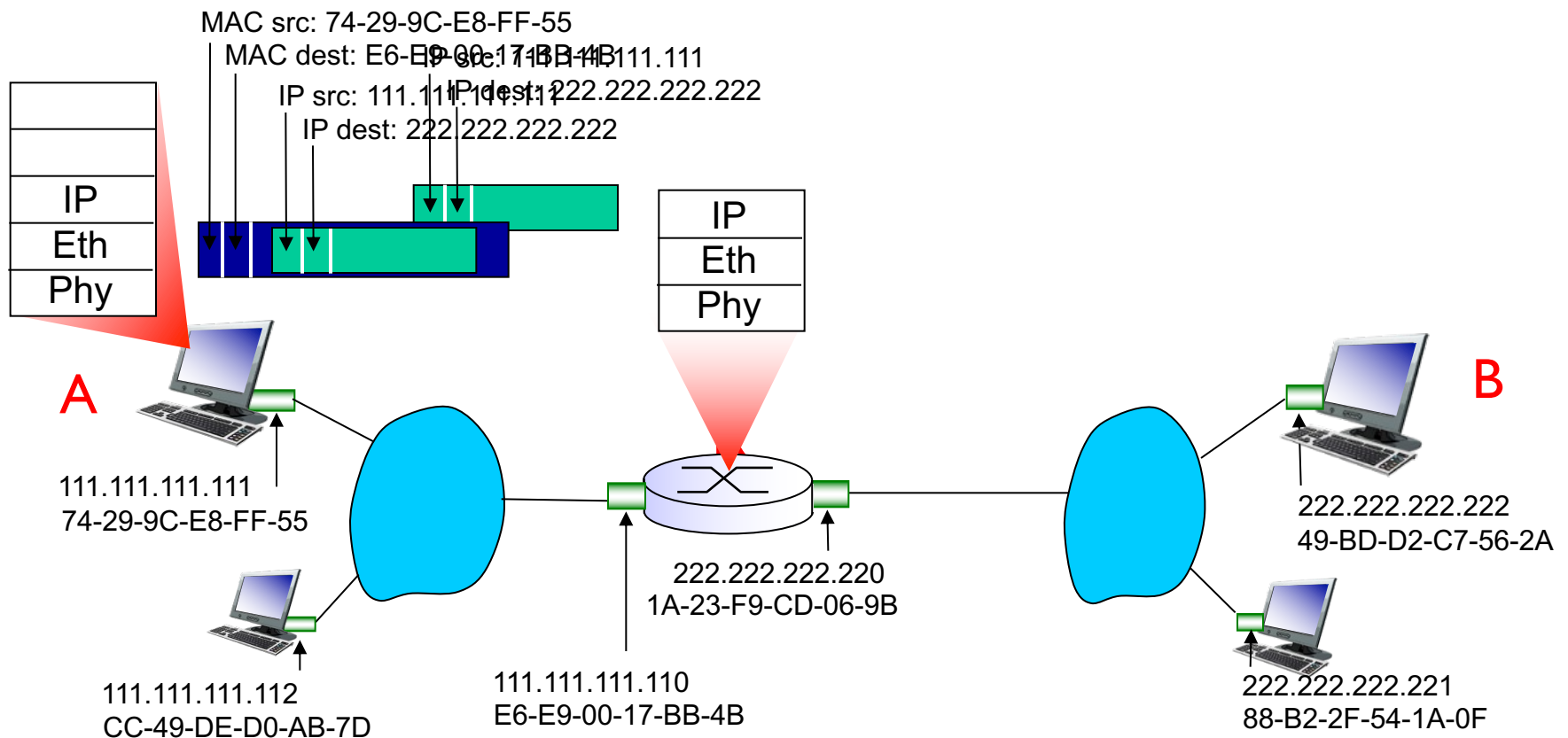
Addressing: routing to another LAN

- 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



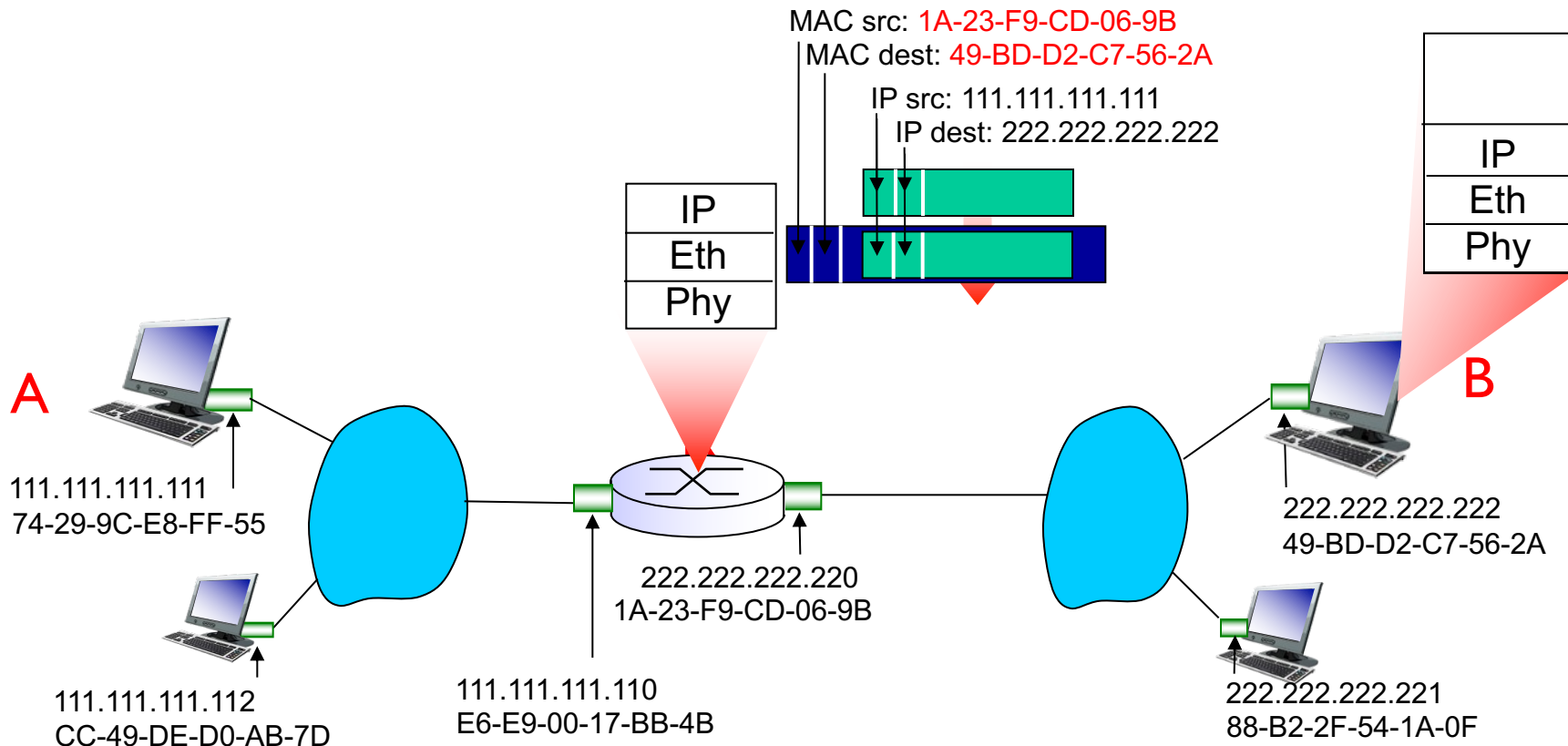
Addressing: routing to another LAN

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



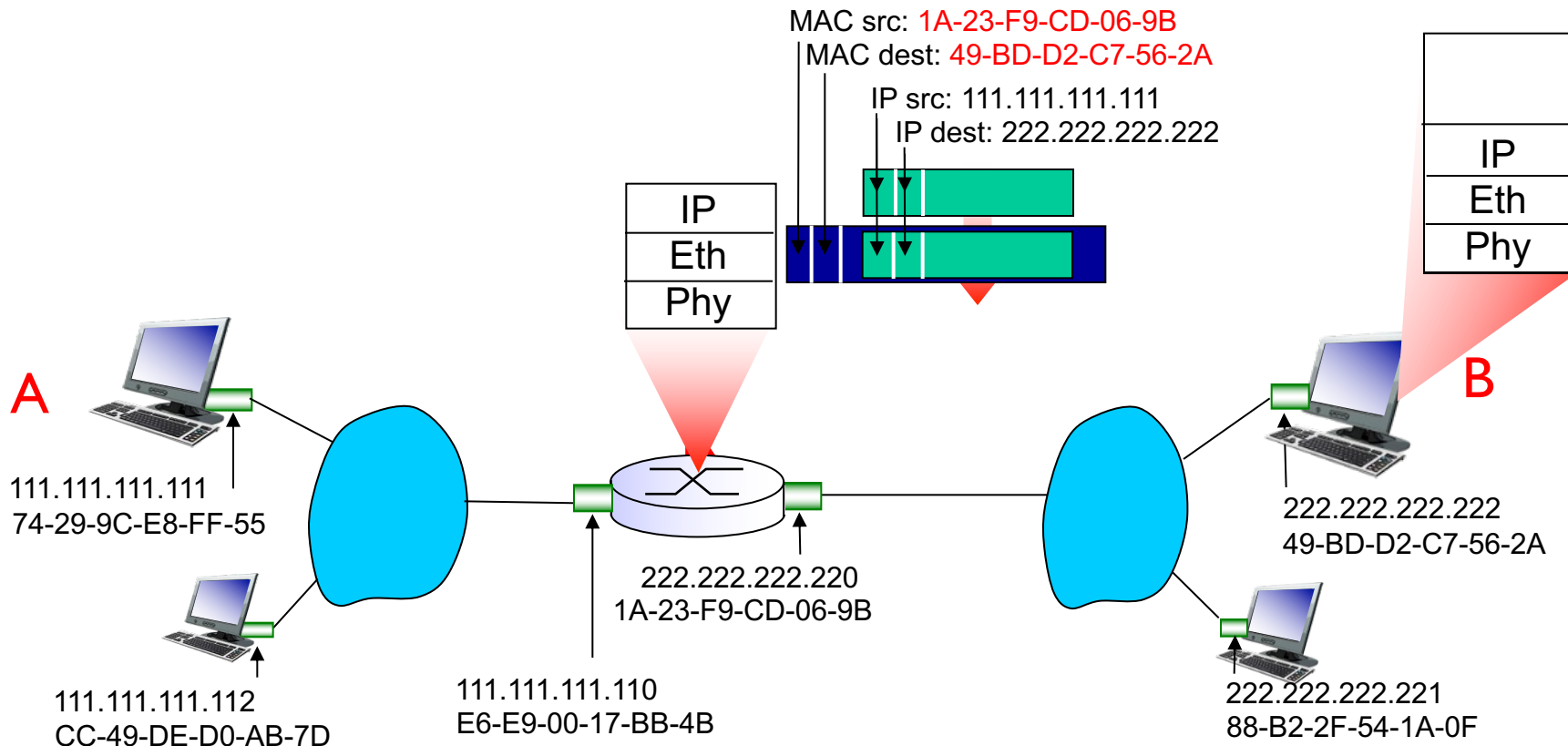
Addressing: routing to another LAN

- 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



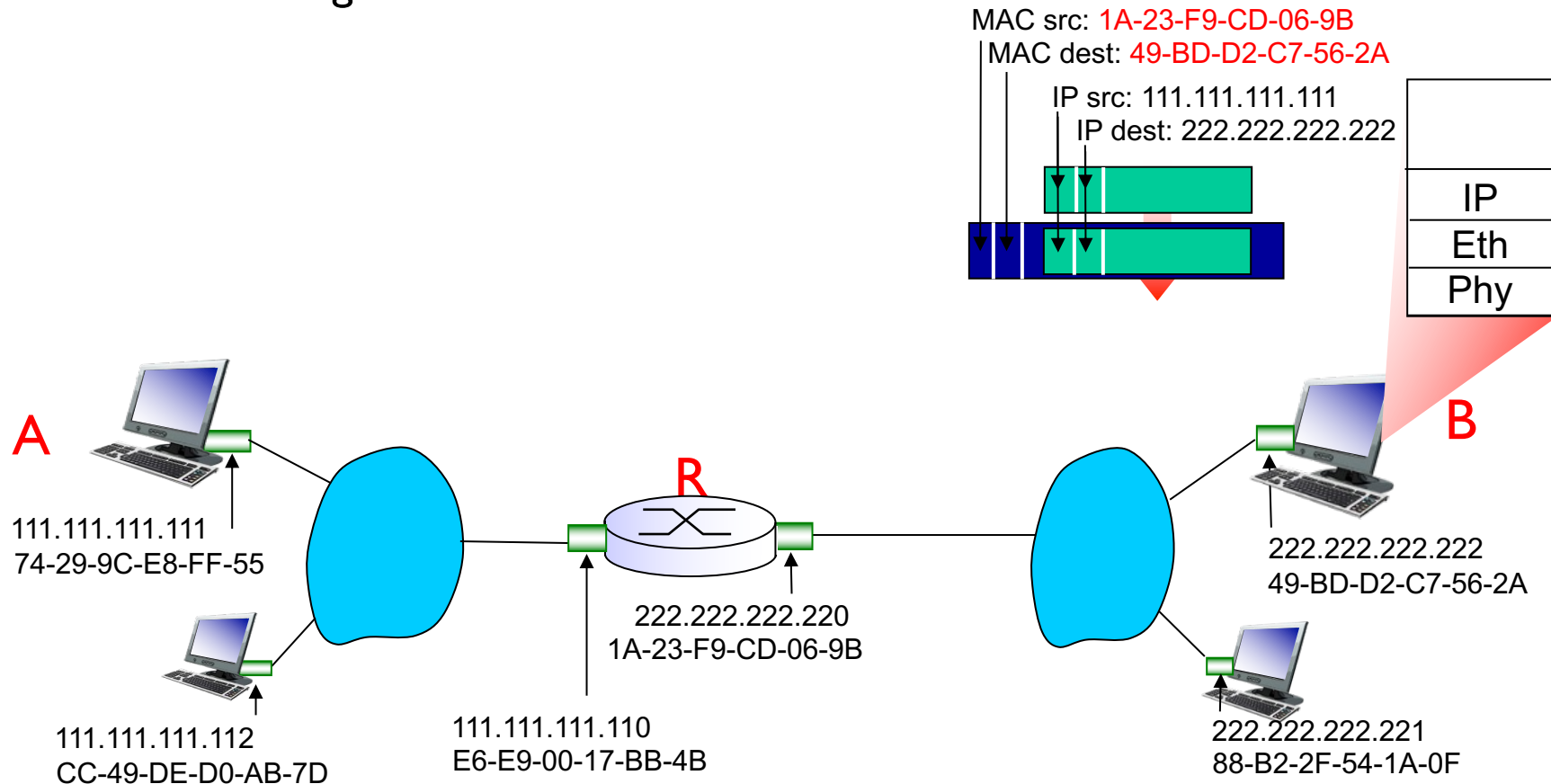
Addressing: routing to another LAN

- 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



T05: outline

5.1 introduction, services

5.2 multiple access
protocols

5.3 LANs

- addressing, ARP
- Ethernet
- switches

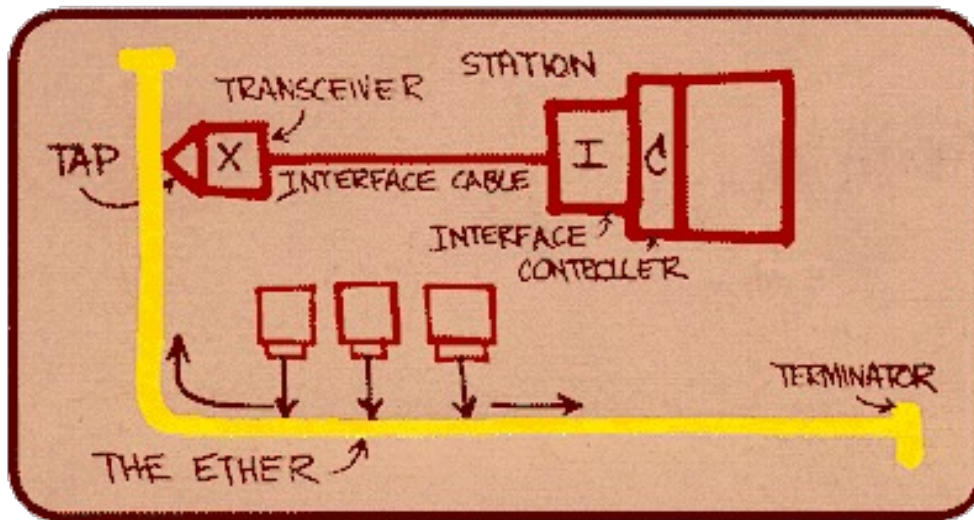
5.4 data center
networking

5.5 a day in the life of a
web request

Ethernet

“dominant” wired LAN technology:

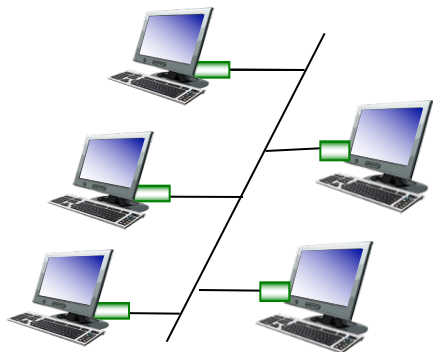
- single chip, multiple speeds (e.g., Broadcom BCM5761)
- first widely used LAN technology
- simpler, cheap
- kept up with speed race: 10 Mbps – 10 Gbps



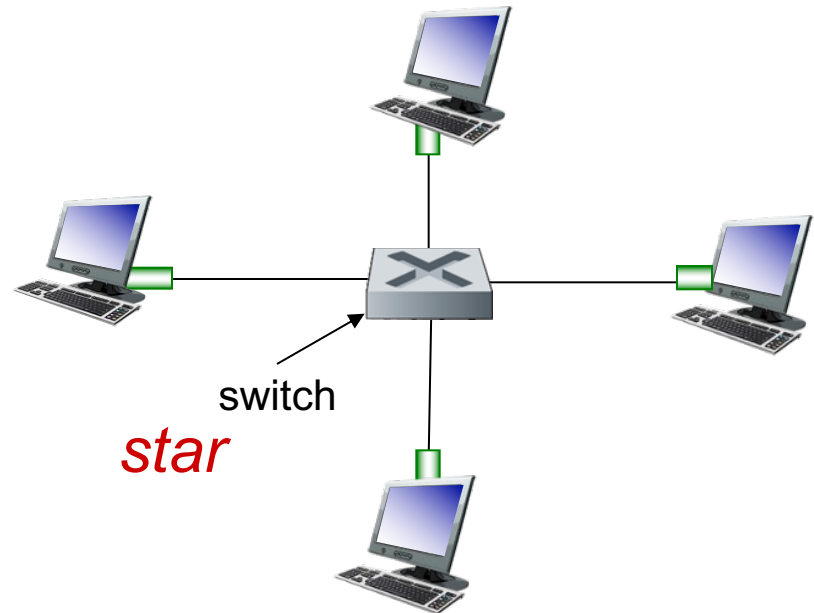
Metcalfe's Ethernet sketch

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



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 (48 bit) 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 but others possible, e.g., Novell IPX, AppleTalk)
- **CRC:** cyclic redundancy check at receiver
 - error detected: frame is dropped

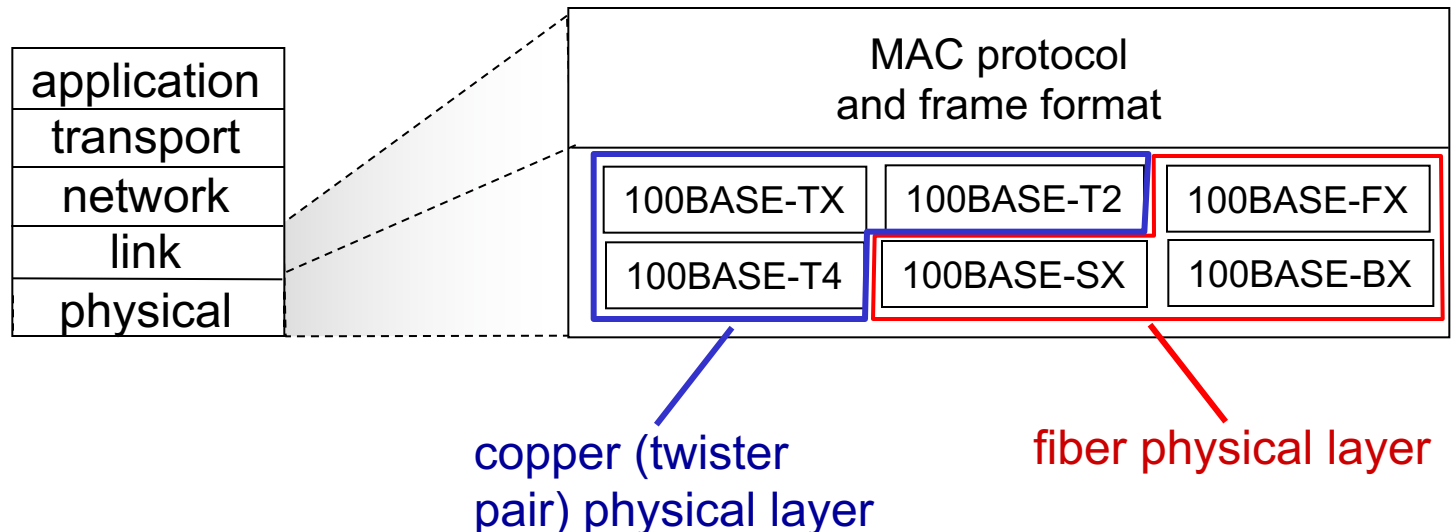


Ethernet: unreliable, connectionless

- *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 exponential 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, 1 Gbps, 10 Gbps, 40 Gbps
 - different physical layer media: fiber, cable



T05: outline

5.1 introduction, services

5.2 multiple access
protocols

5.3 LANs

- addressing, ARP
- ethernet
- switches

5.4 data center
networking

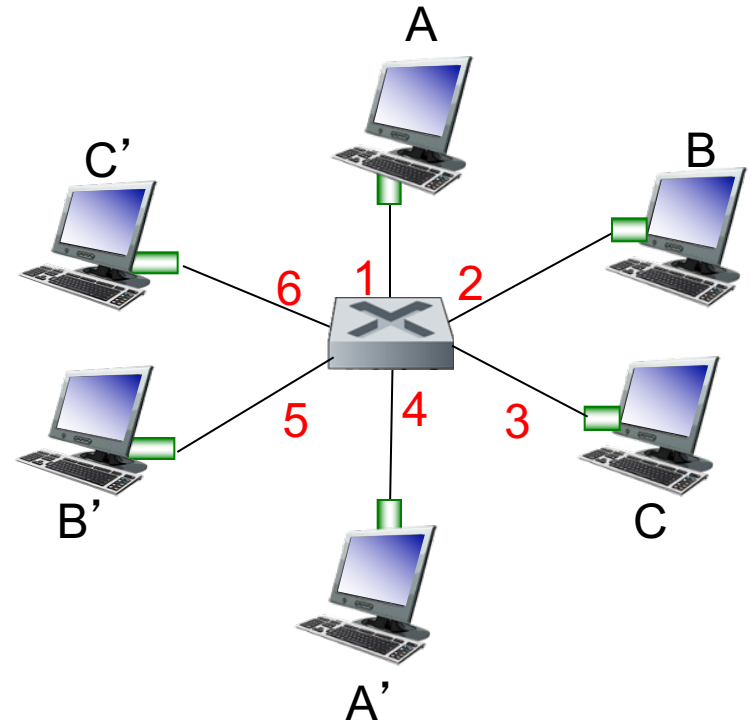
5.5 a day in the life of a
web request

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



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

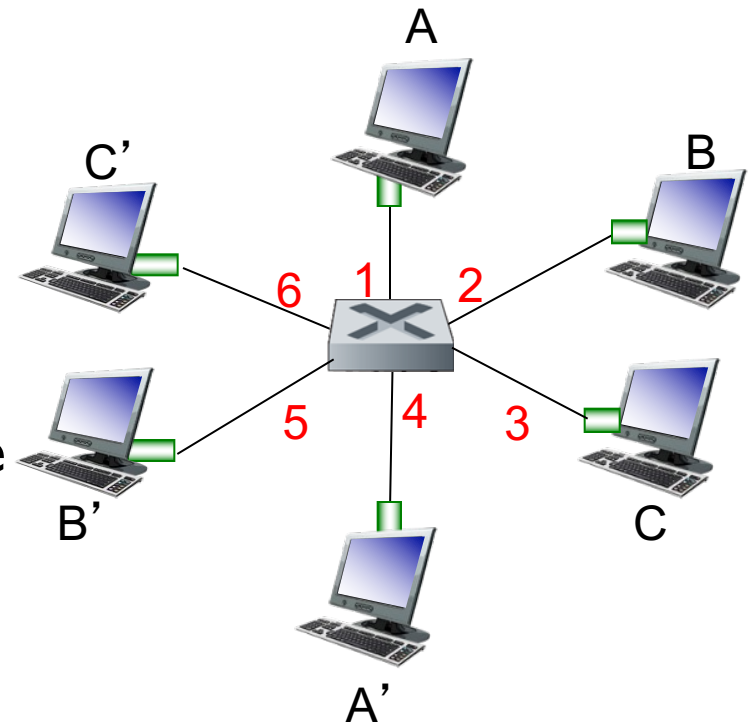
Switch forwarding table

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?

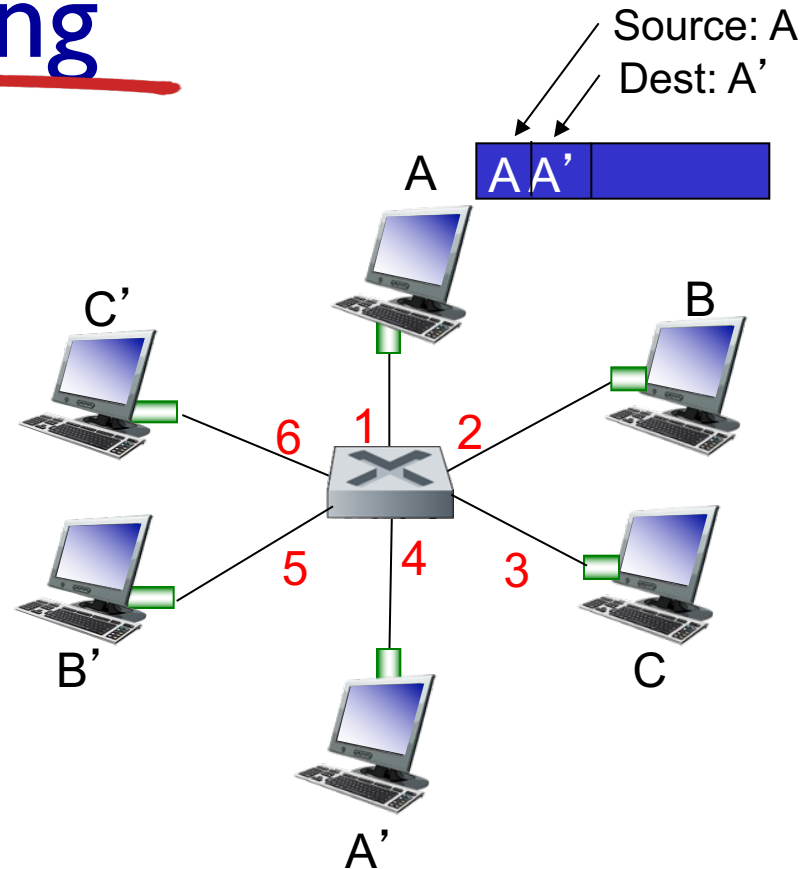
- something like a routing protocol?



*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)*

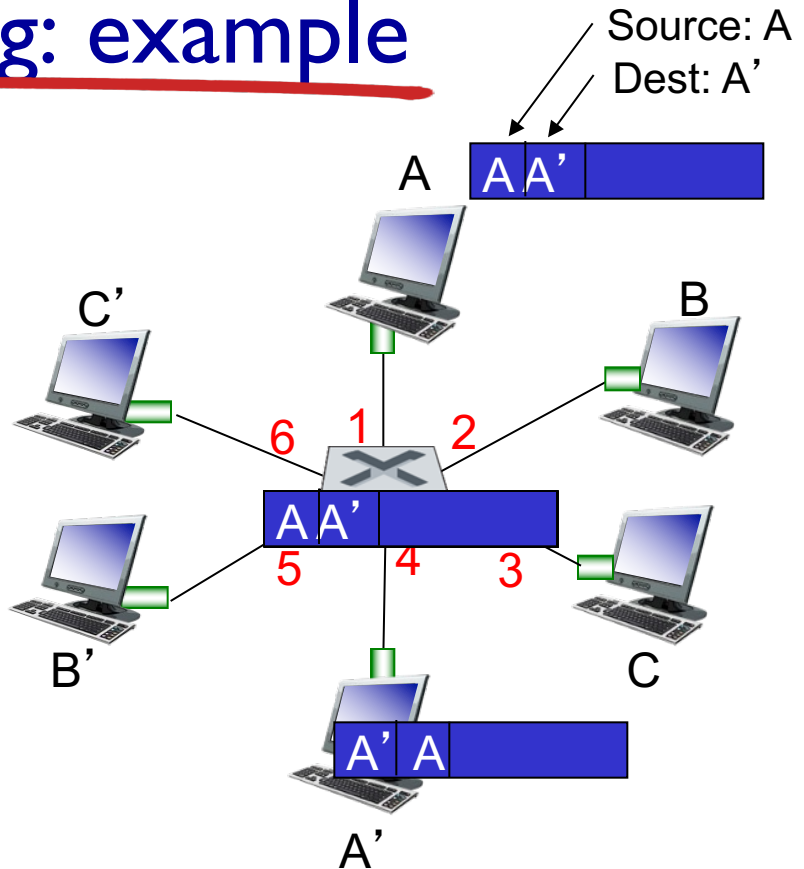
Switch: frame filtering/forwarding

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

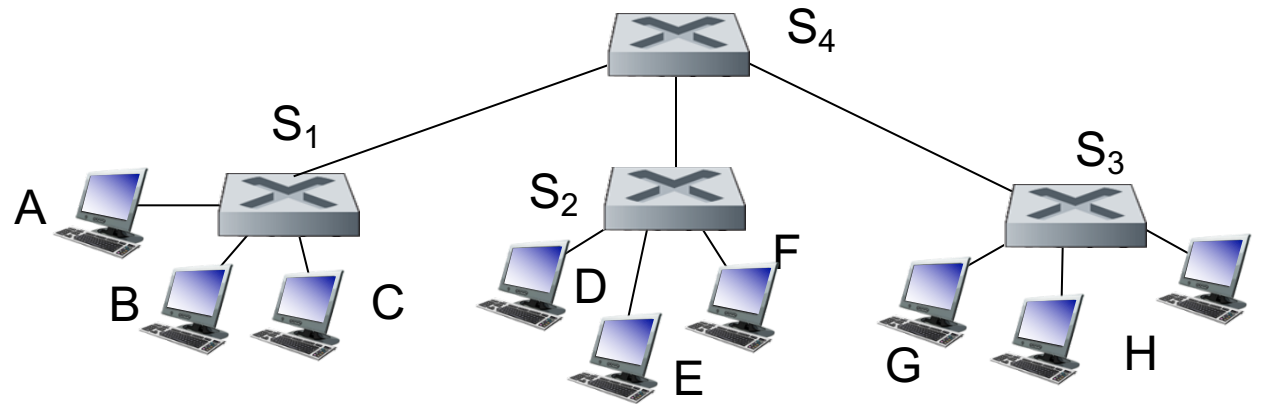


MAC addr	interface	TTL
A	1	60
A'	4	60

*switch table
(initially empty)*

Interconnecting switches

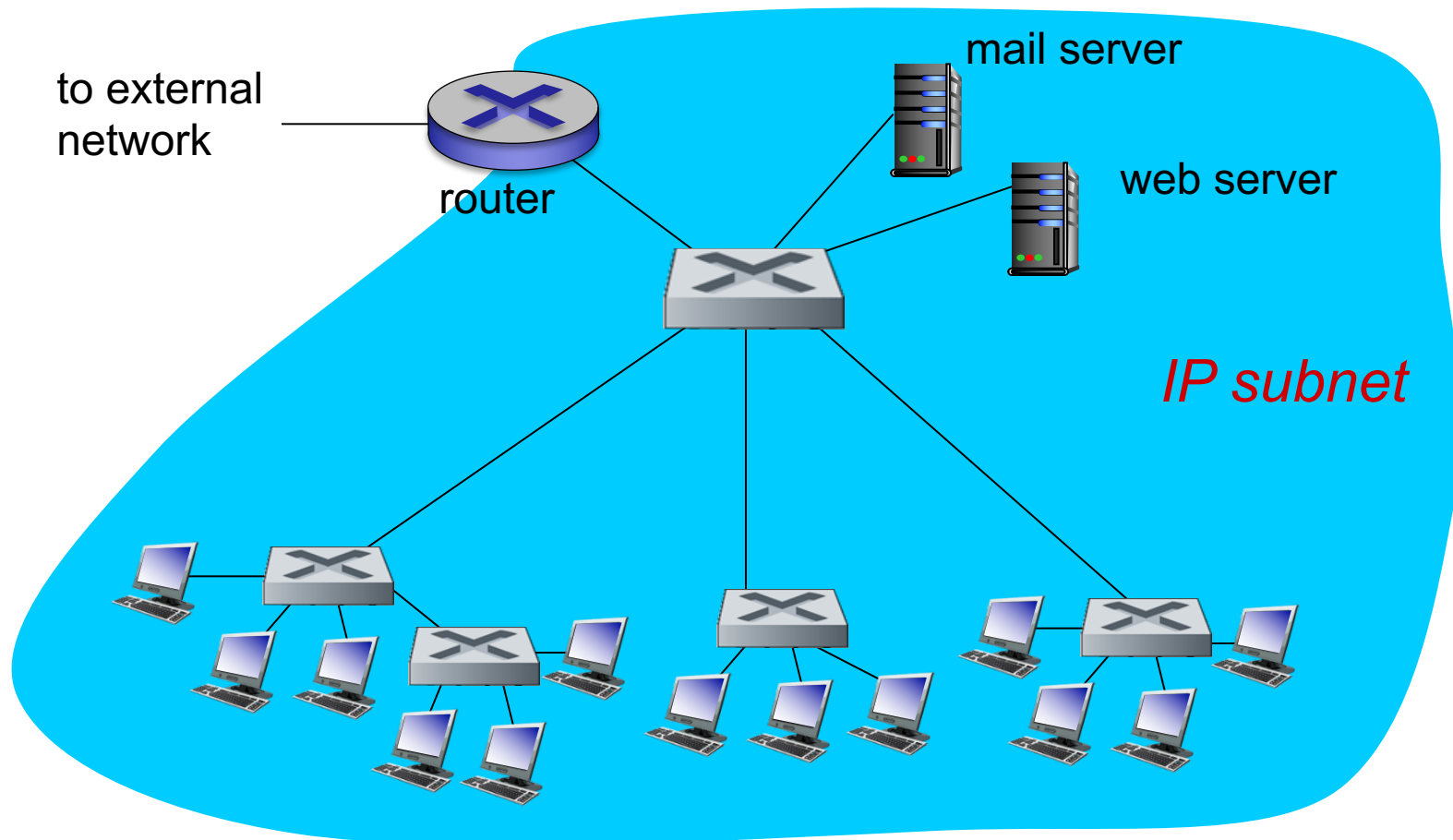
self-learning switches can be connected together:



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



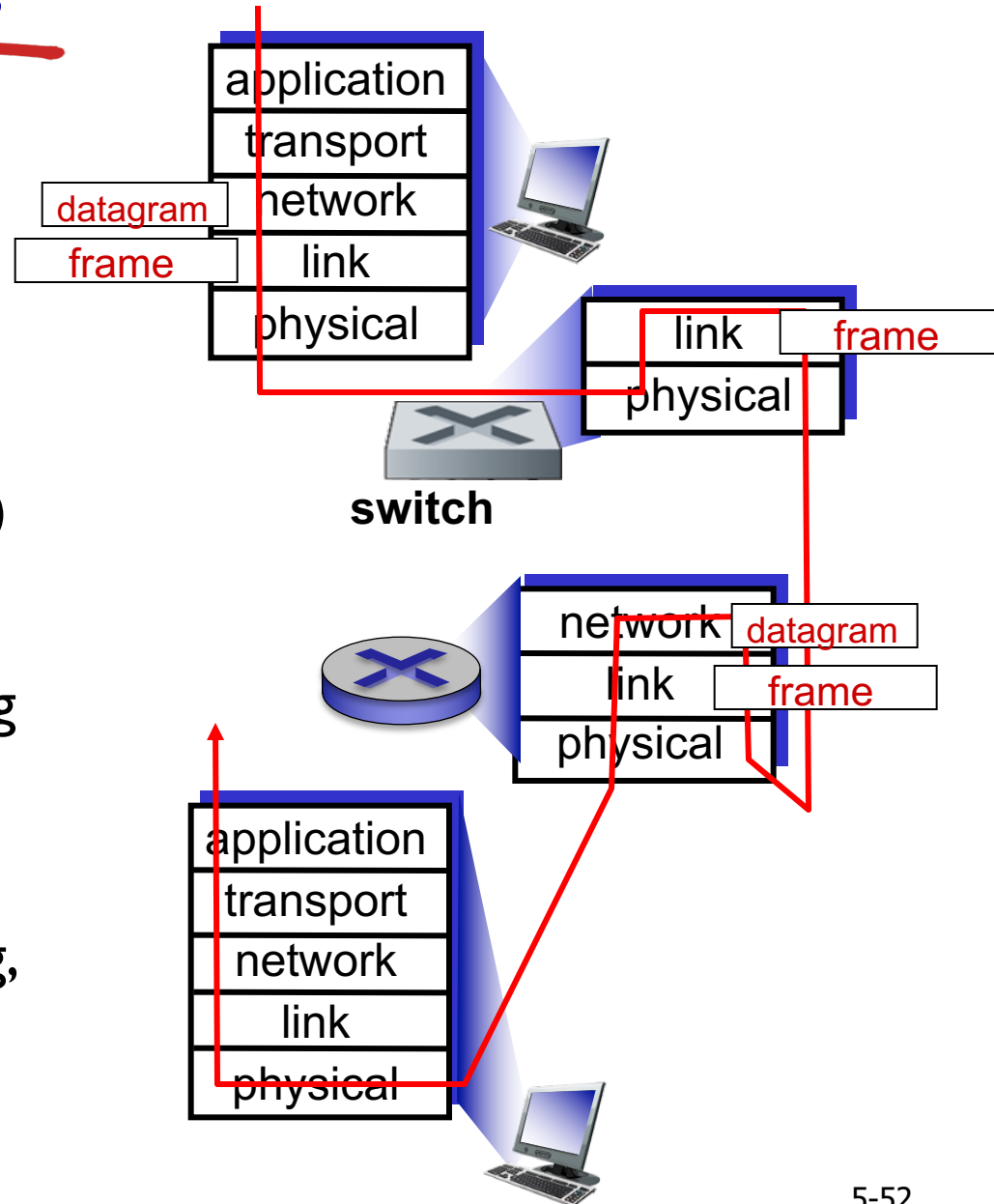
Switches vs. routers

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



T06: outline

6.1 introduction, services

6.2 multiple access
protocols

6.3 LANs

- addressing, ARP
- Ethernet
- switches

6.4 data center
networking

6.5 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

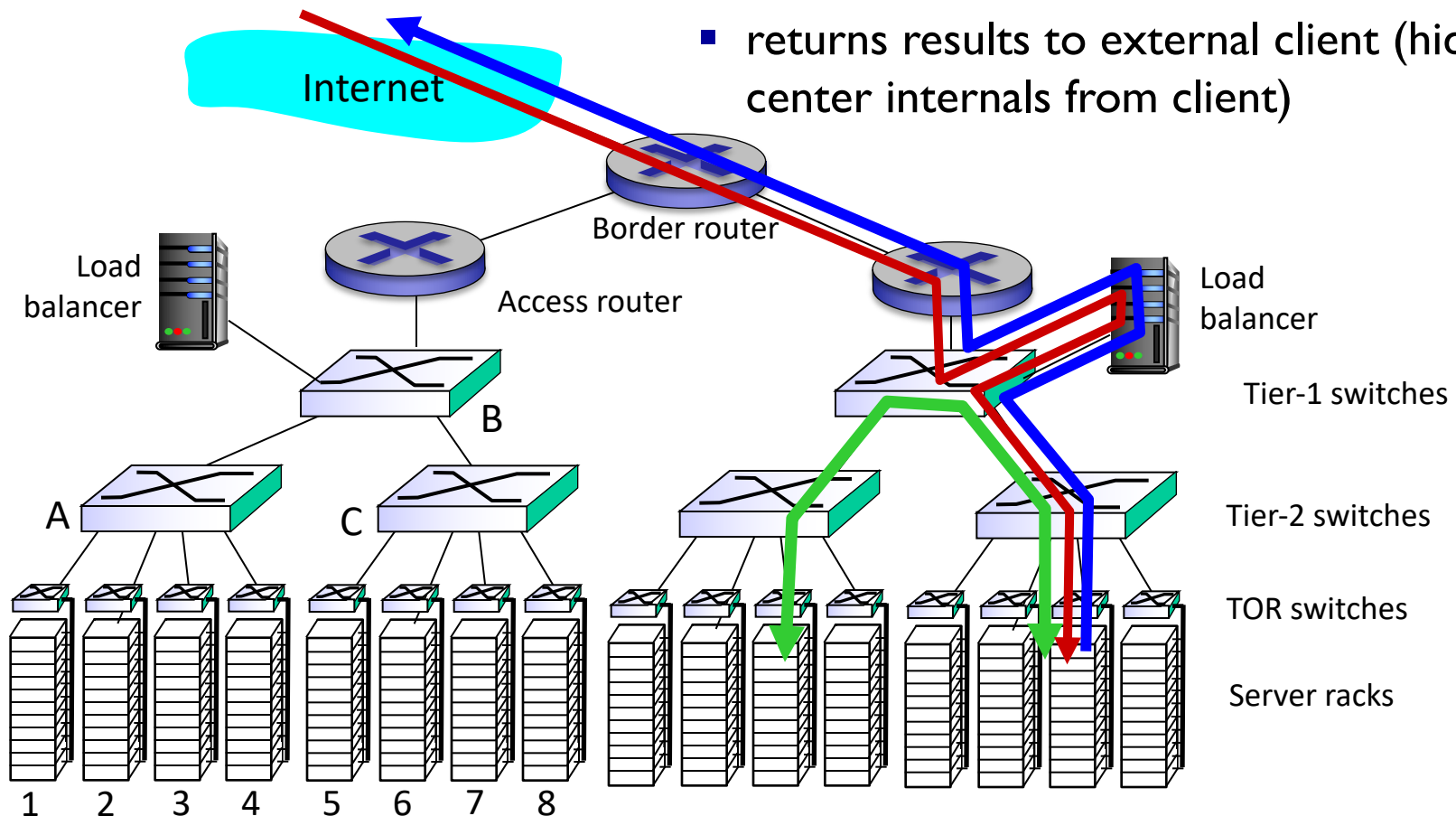


Inside a 40-ft Microsoft container,
Chicago data center

Data center networks

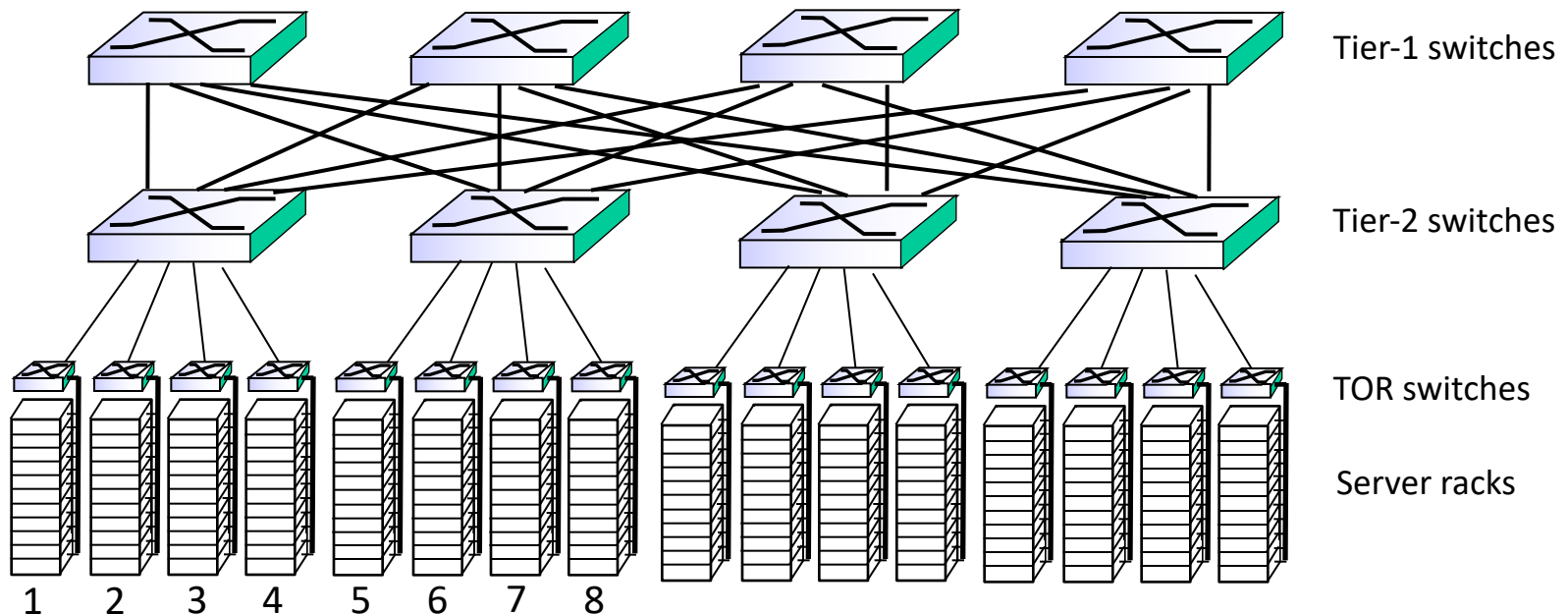
load balancer: application-layer routing

- 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



T05: outline

5.1 introduction, services

5.2 multiple access
protocols

5.3 LANs

- addressing, ARP
- Ethernet
- switches

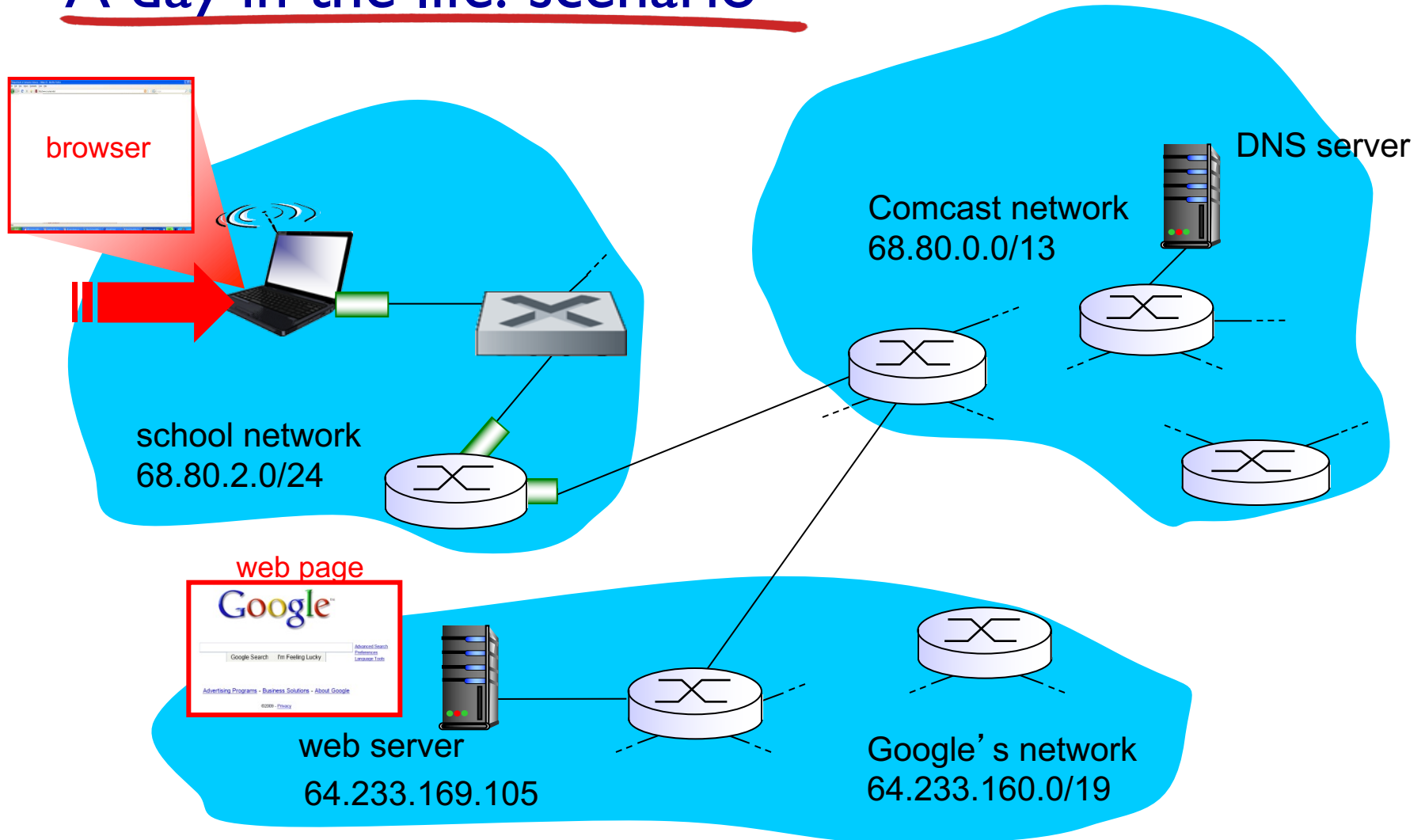
5.4 data center
networking

5.5 a day in the life of a
web request

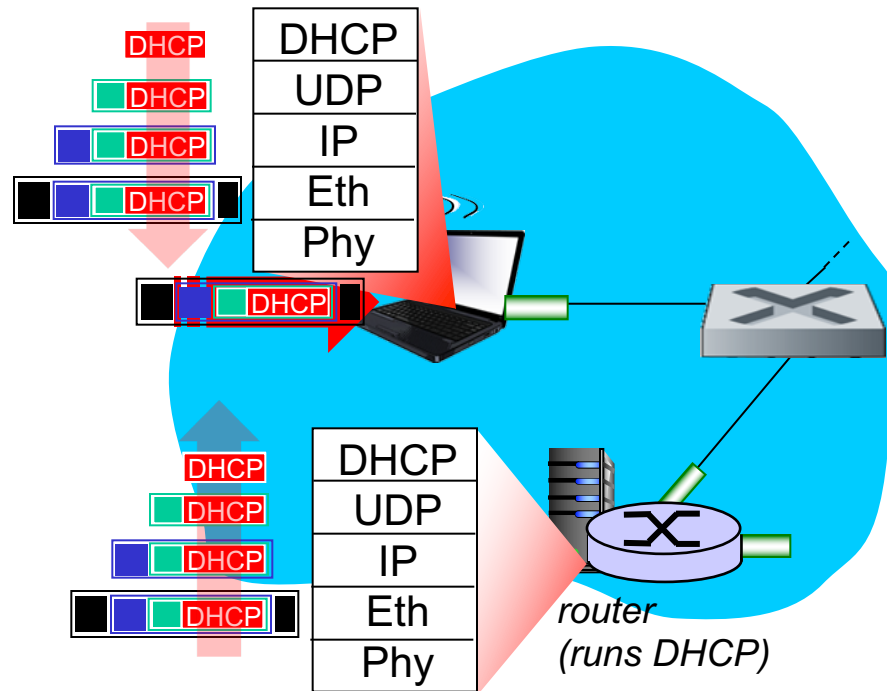
Synthesis: a day in the life of a web request

- 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

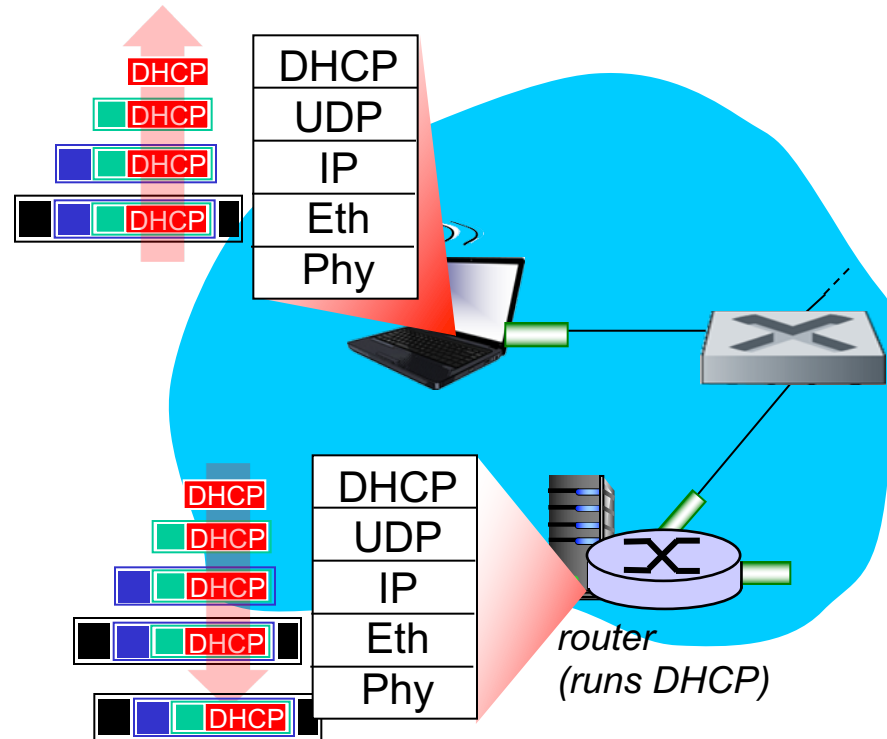


A day in the life... connecting to the Internet



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

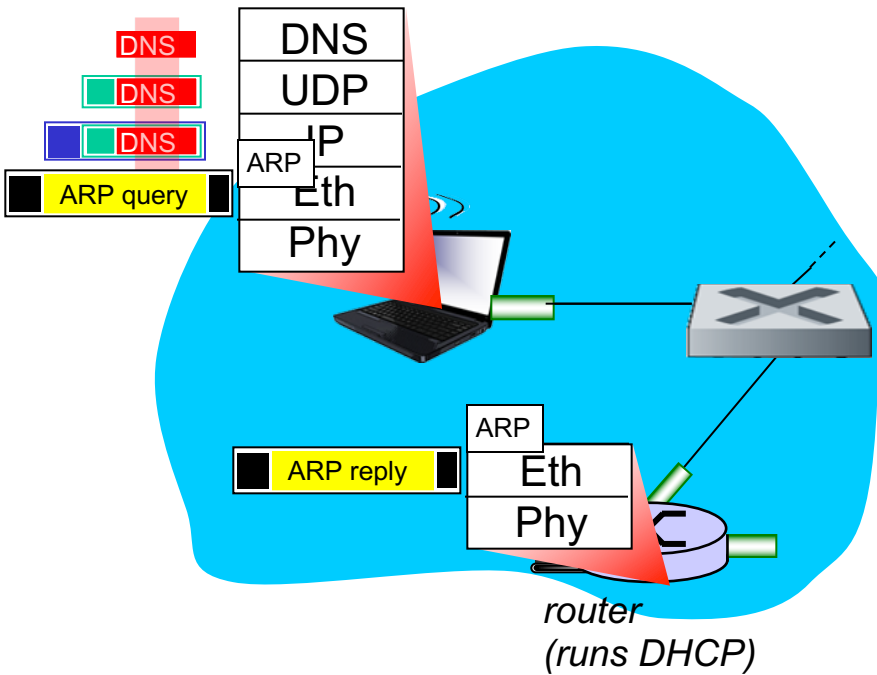
A day in the life... connecting to the Internet



- 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

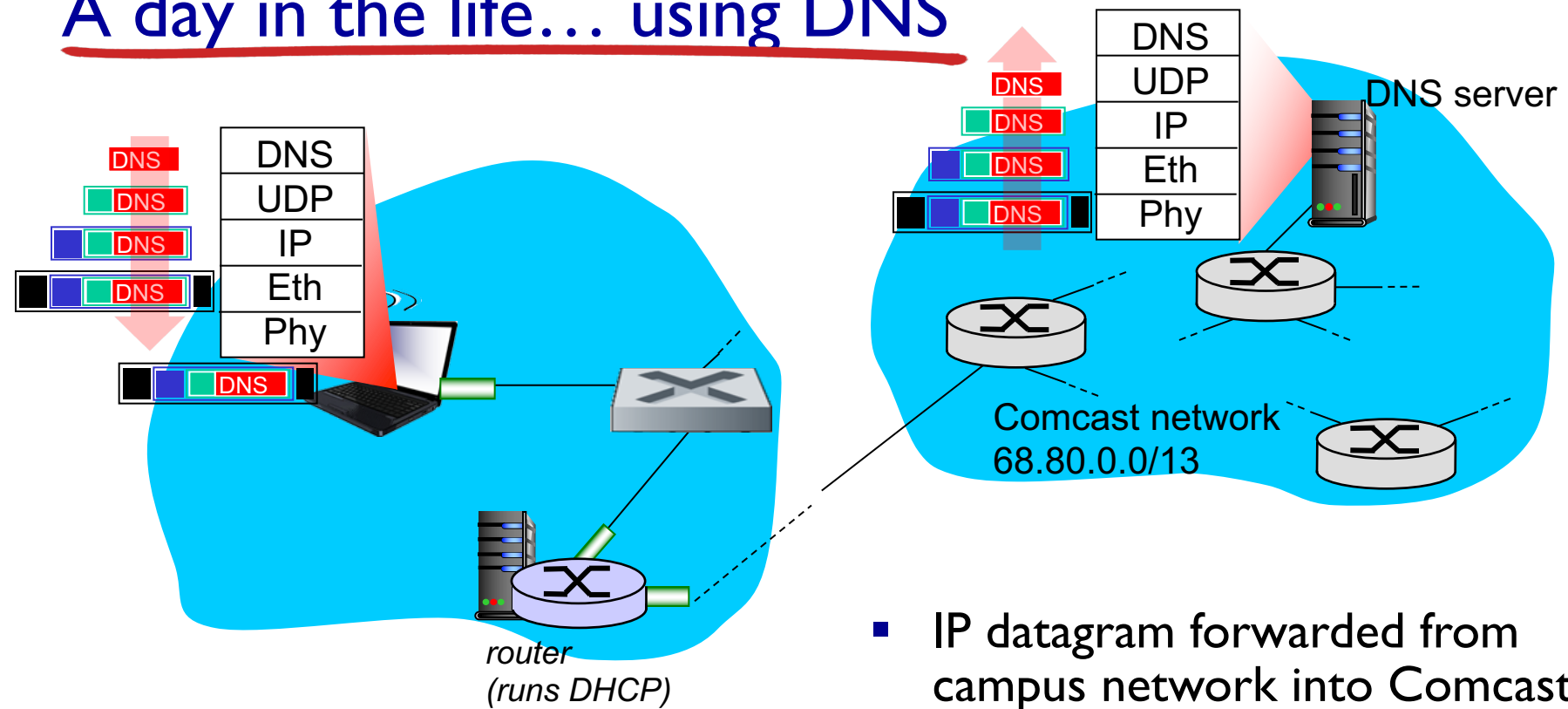
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

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

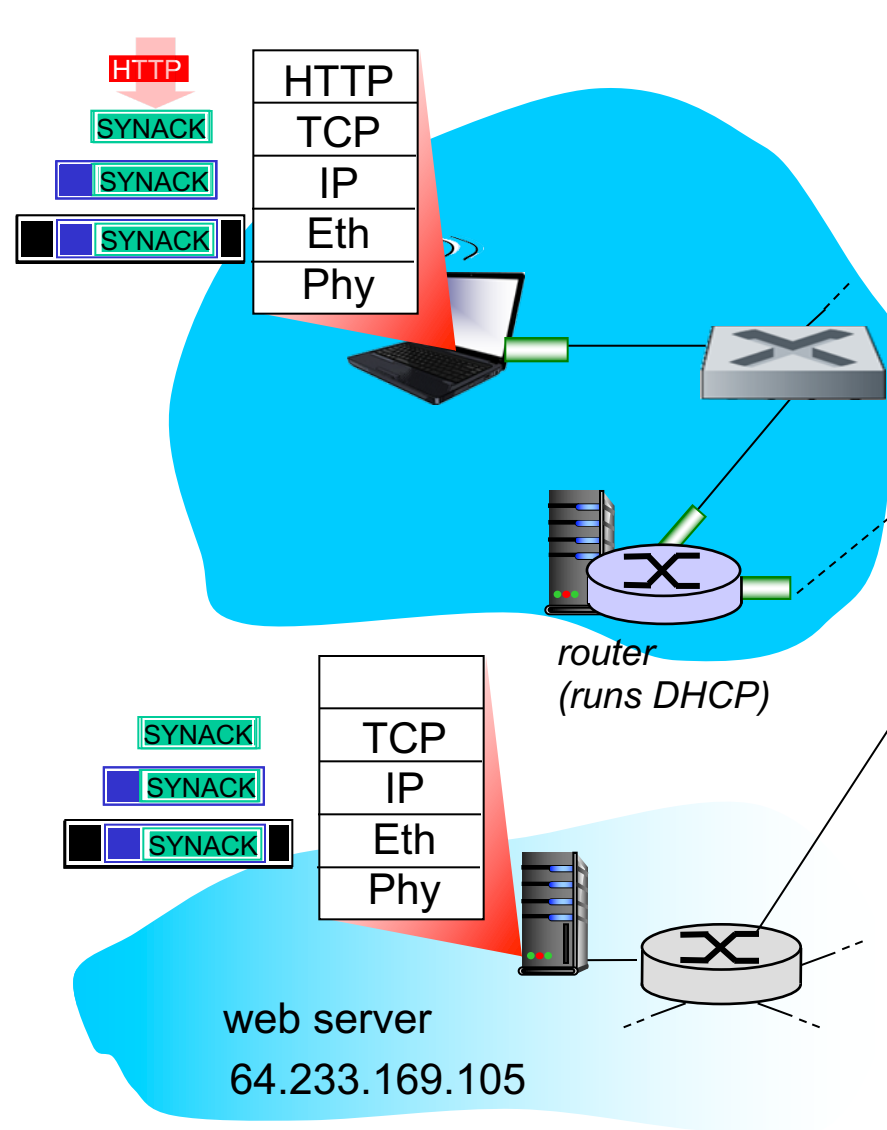
A day in the life... using DNS



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

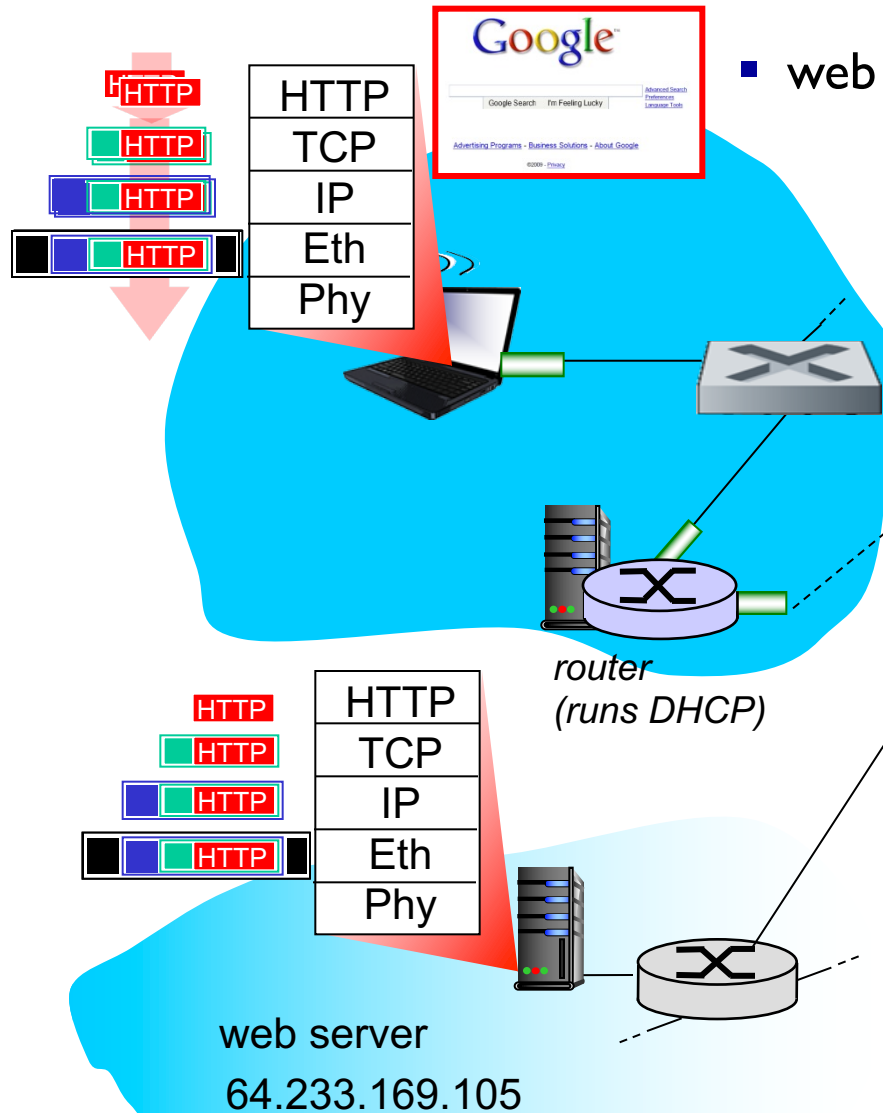
- IP datagram forwarded from campus network into Comcast network, routed (tables created by **RIP**, **OSPF**, **IS-IS** and/or **BGP** routing protocols) to DNS server
- demuxed to DNS server
- DNS server replies to client with IP address of www.google.com

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



- 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

T05: 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

T05: Bibliography

J. Kurose and K. Ross, “Computer Networking – a top-down approach”, Pearson. Chapter 5: The Link Layer: Links, Access Networks, and LANs

