

# Chapter 6

## The Link Layer and LANs

A note on the use of these Powerpoint slides:

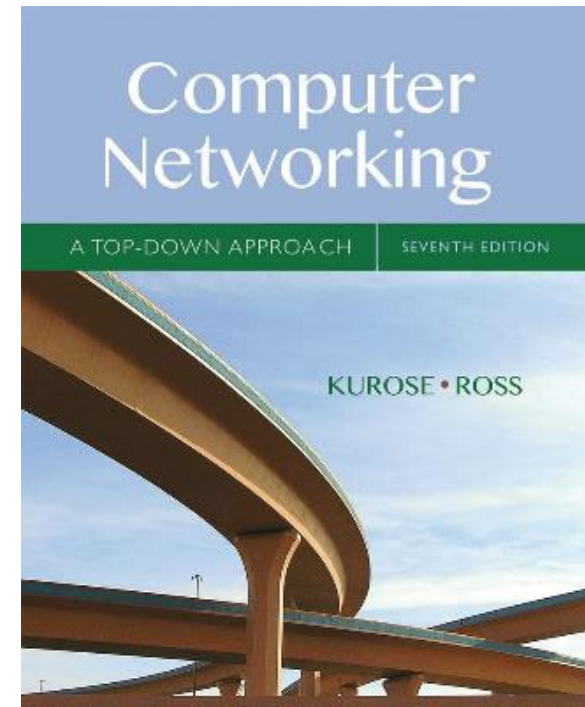
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a lot of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

All material copyright © 1996-2016

J.F Kurose and K.W. Ross, All Rights Reserved



**Computer Networking: A Top  
Down Approach**

7<sup>th</sup> edition

Jim Kurose, Keith Ross

Pearson/Addison Wesley

April 2016

Link Layer and LANs 6-1

# 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

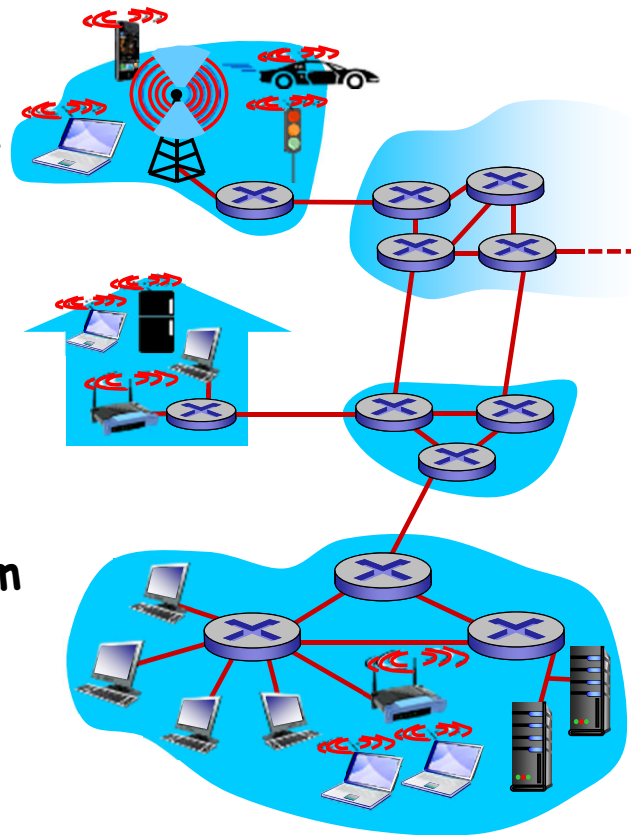
6.7 a day in the life of a  
web request

# 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

- 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

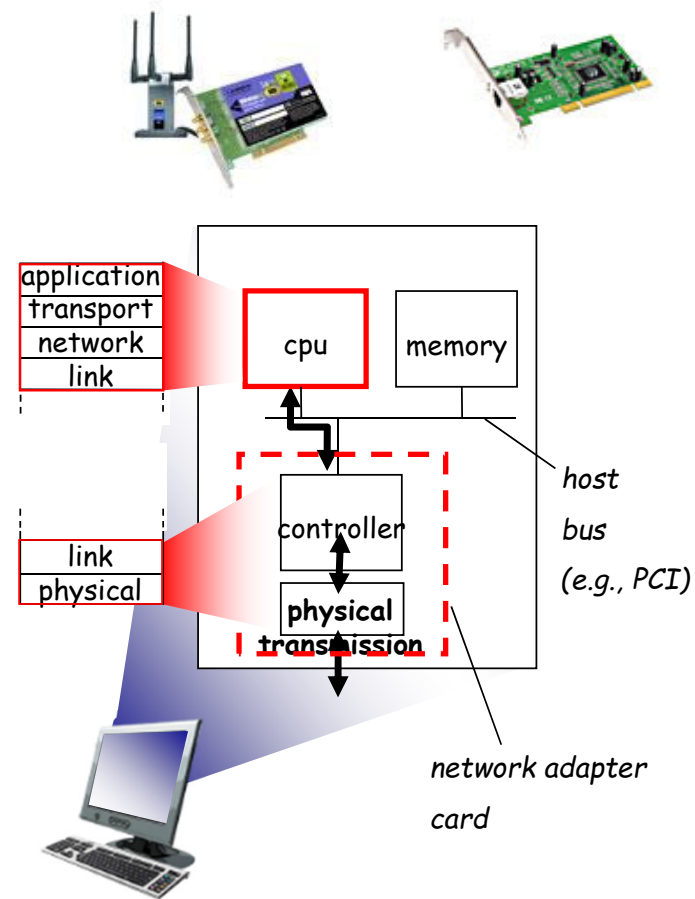
- *framing, link access:*
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - “MAC” addresses used in frame headers to identify source, destination
    - different from IP address!
- *reliable delivery between adjacent nodes*
  - we learned how to do this already (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-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

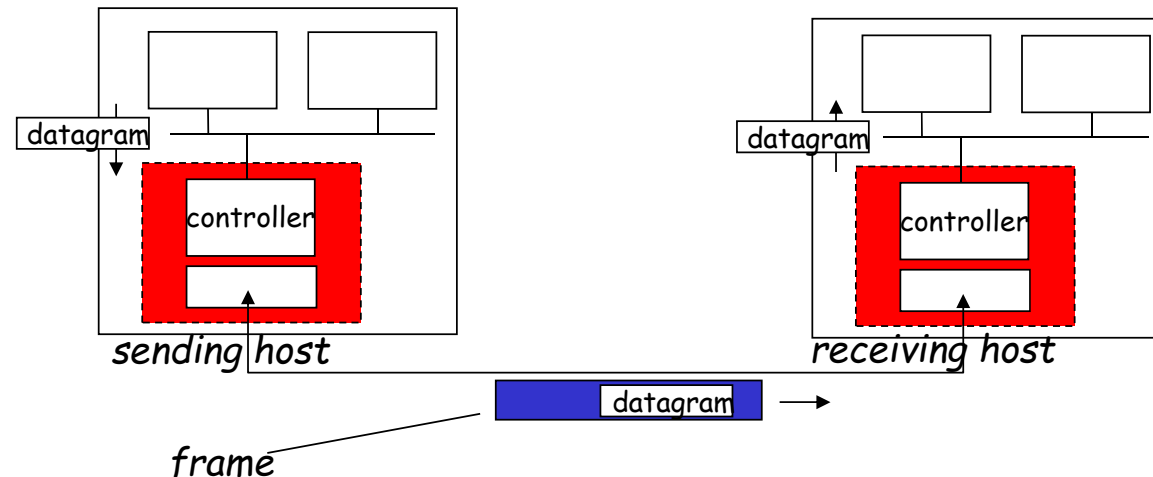
# Where is the link layer implemented?

- in each and every host
- link layer implemented in “adaptor”  
(aka *network interface card* NIC) or on a chip
  - Ethernet card, 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, rdt, flow control, etc.

## ■ receiving side

- looks for errors, rdt, flow control, etc.
- extracts datagram, 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

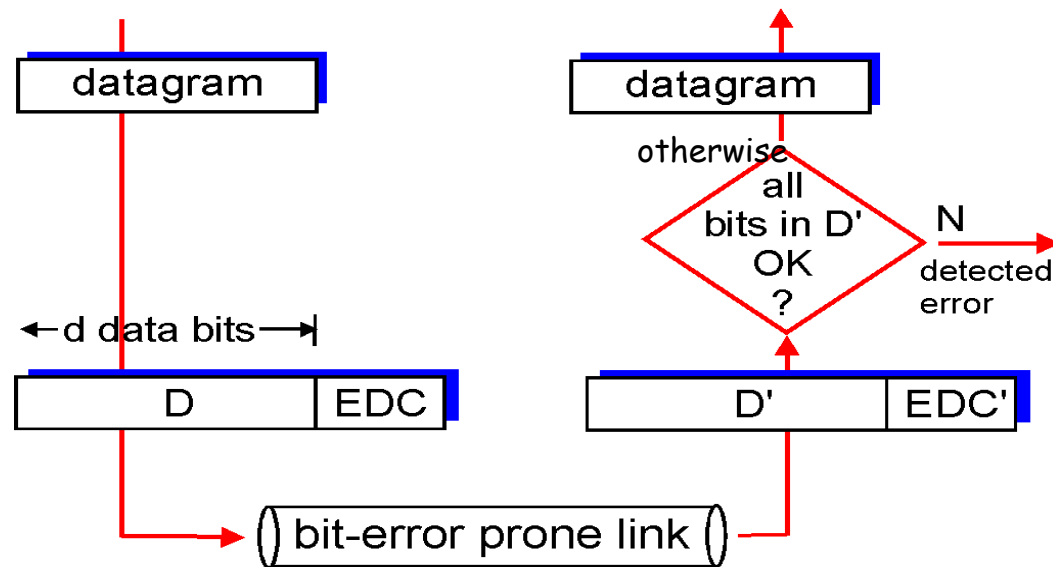
# Error detection

EDC= Error Detection and Correction bits (redundancy)

D = Data protected by error checking, may include header fields

Error detection not 100% reliable!

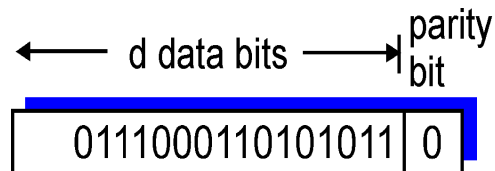
- protocol may miss some errors, but rarely
- larger EDC field yields better detection and correction



# Parity checking

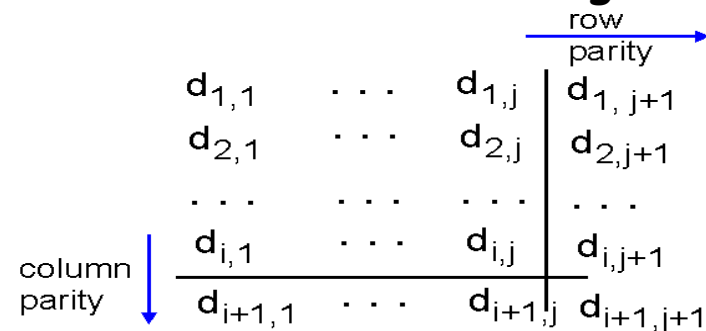
## single bit parity:

- detect single bit errors



## two-dimensional bit parity:

- detect and correct single bit errors



1	0	1	0	1	1
1	1	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

no errors

1	0	1	0	1	1
1	0	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

parity error

parity error

correctable  
single bit error

\* Check out the online interactive exercises for more examples:

[http://gaia.cs.umass.edu/kurose\\_ross/interactive/](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:**

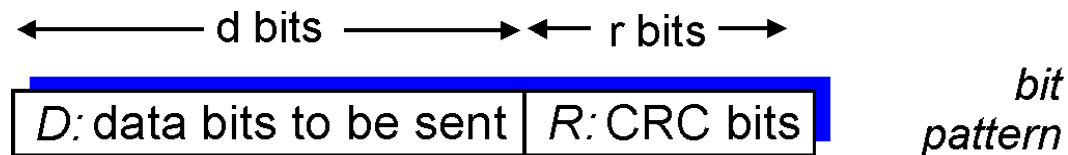
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

## **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

- 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
  - $\langle D, R \rangle$  exactly divisible by  $G$  (modulo 2)
  - receiver knows  $G$ , divides  $\langle D, R \rangle$  by  $G$ . If non-zero remainder: error detected!
  - can detect all burst errors less than  $r+1$  bits
- widely used in practice (Ethernet, 802.11 WiFi, ATM)



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

*mathematical formula*

## CRC example

**want:**

$$D \cdot 2^r \text{ XOR } R = nG$$

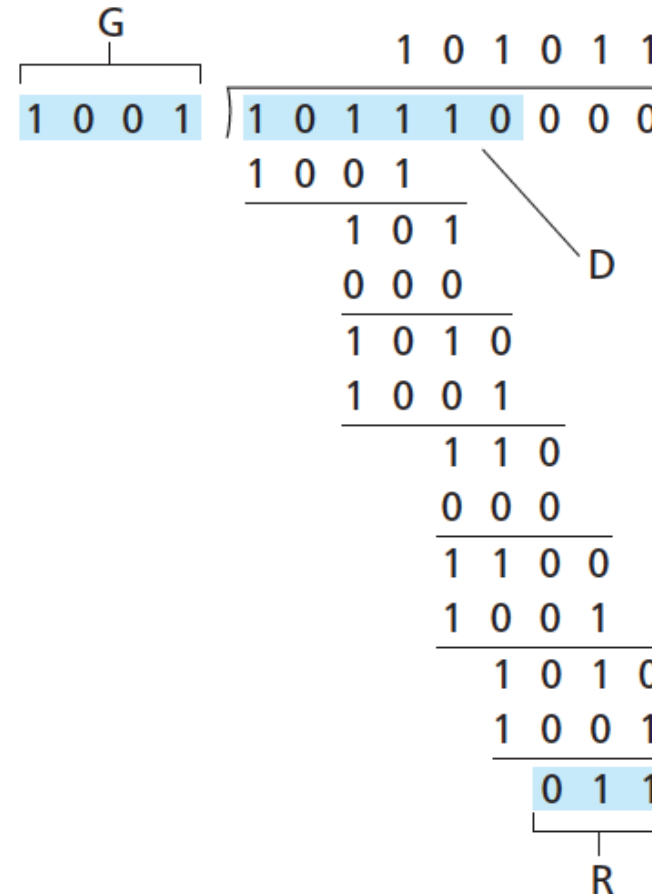
*equivalently:*

$$D \cdot 2^r = nG \text{ XOR } R$$

*equivalently:*

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

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



\* Check out the online interactive exercises for more examples:

[http://gaia.cs.umass.edu/kurose\\_ross/interactive/](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

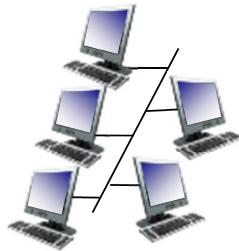
6.7 a day in the life of a web  
request



# Multiple access links, protocols

two types of “links” :

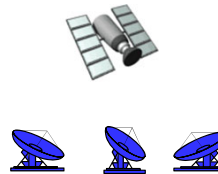
- point-to-point
  - PPP for dial-up access
  - point-to-point link between Ethernet switch, host
- **broadcast (shared wire or medium)**
  - old-fashioned Ethernet
  - upstream HFC
  - 802.11 wireless LAN



shared wire (e.g.,  
cabled Ethernet)



shared RF  
(e.g., 802.11 WiFi)



shared RF  
(satellite)



humans at a  
cocktail party  
(shared air, acoustical)  
Link Layer and LANs 6-17

# 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

## *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

# 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

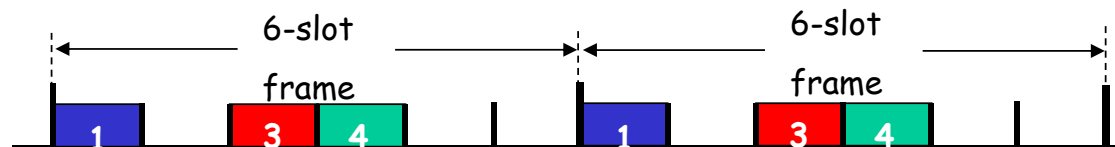
- *“taking turns”*

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

# Channel partitioning MAC protocols: TDMA

## TDMA: time division multiple access

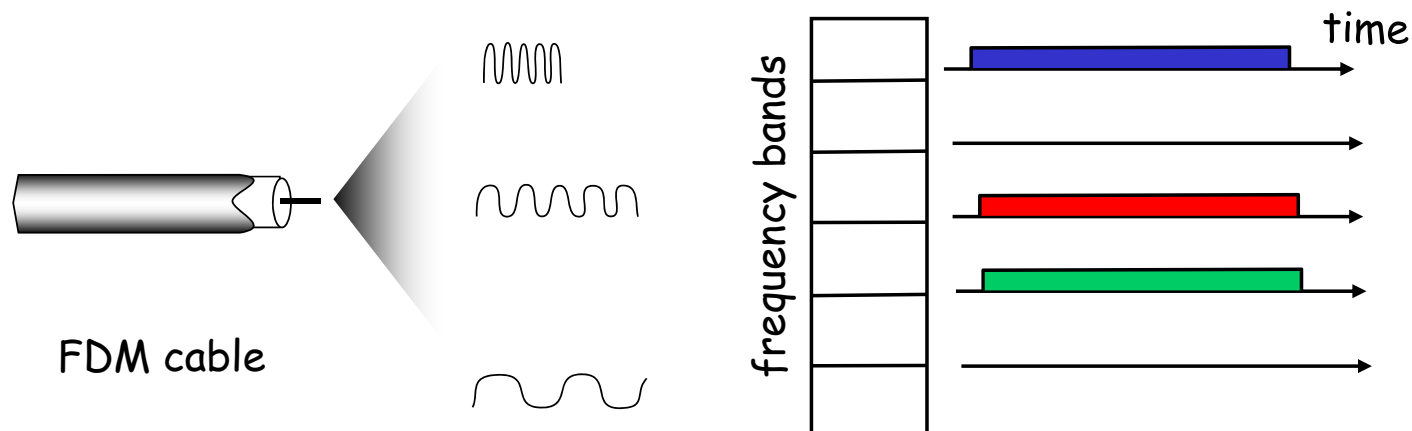
- access to channel in "rounds"
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



# 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 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)
- examples of random access MAC protocols:
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA

# Slotted ALOHA

## *assumptions:*

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

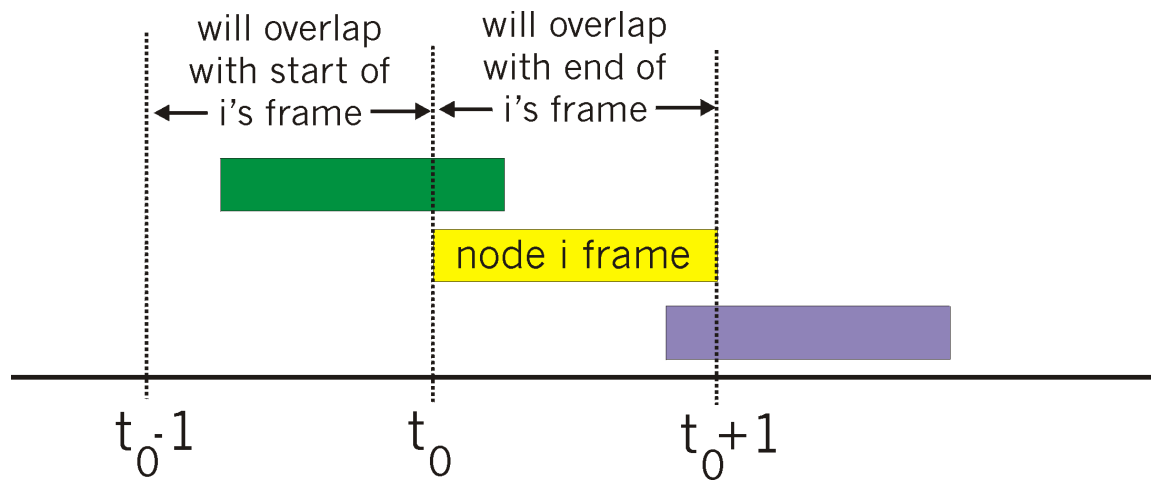
## *operation:*

- 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 prob.  $p$  until success



# 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

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

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

$$P(\text{no other node transmits in } [t_0-1, t_0])$$

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

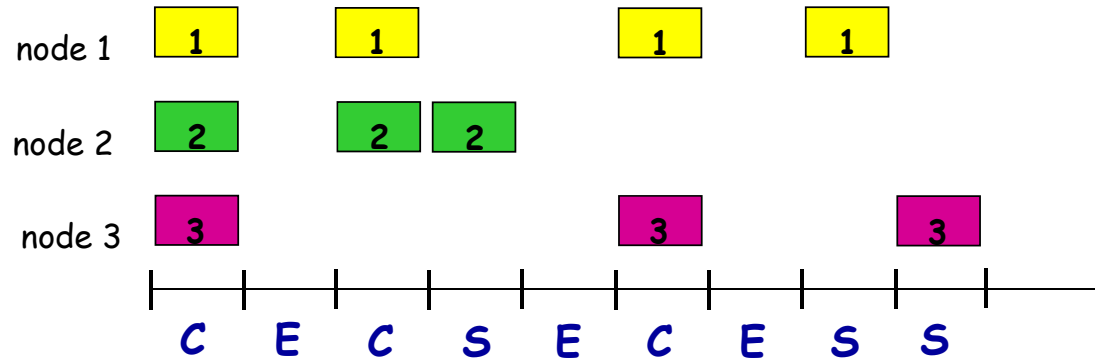
$$= p \cdot (1-p)^{2(N-1)} \rightarrow \infty$$

... choosing optimum  $p$  and then letting  $n$

$$= 1/(2e) = .18$$

*even worse than slotted Aloha!*

# 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!



# CSMA (carrier sense multiple access)

**CSMA:** listen before transmit:

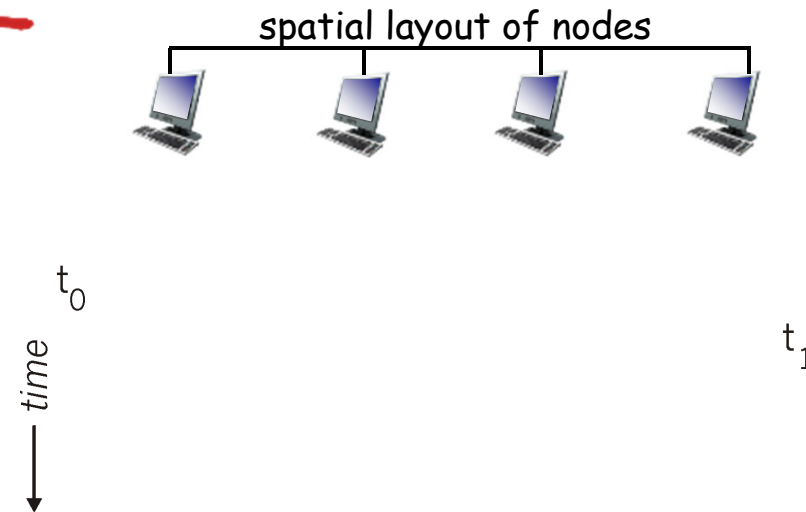
if channel sensed idle: transmit entire frame

- if channel sensed 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

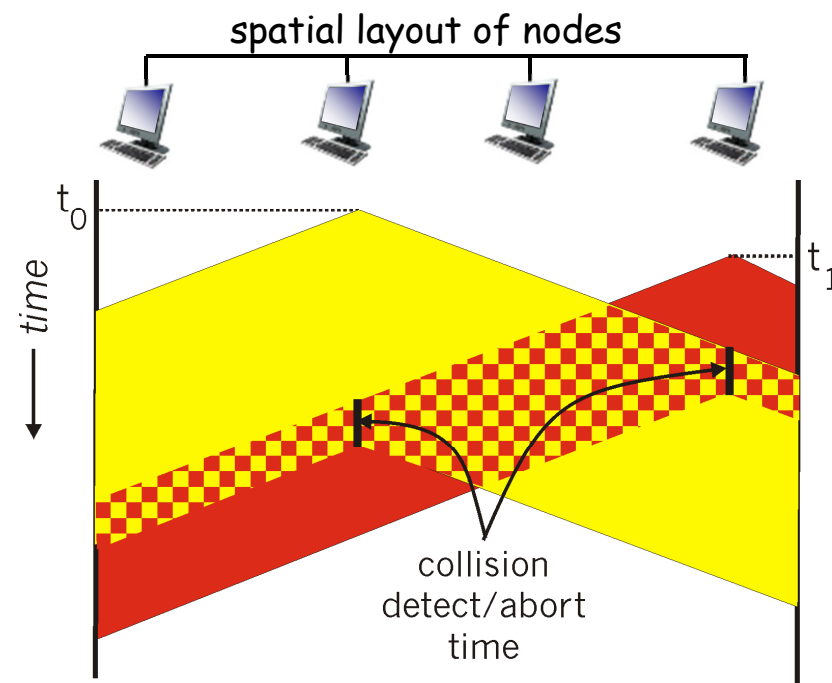


# CSMA/CD (collision detection)

**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 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 and sends jam signal
5. After aborting, NIC enters **binary (exponential) backoff**:
  - after  $m$ th collision, NIC chooses  $K$  at random from  $\{0, 1, 2, \dots, 2^m - 1\}$ . 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

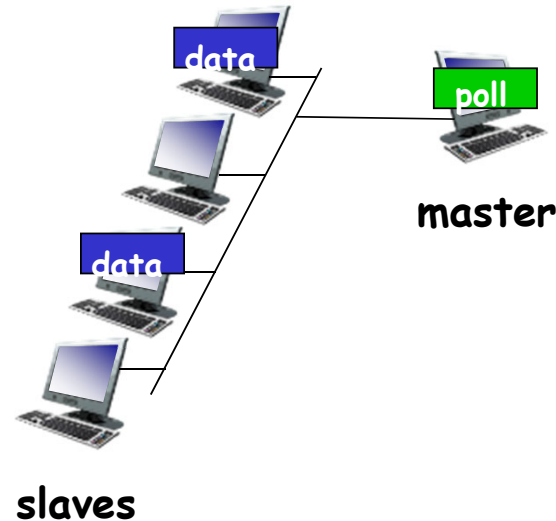
## "taking turns" protocols

look for best of both worlds!

# "Taking turns" MAC protocols

## *polling:*

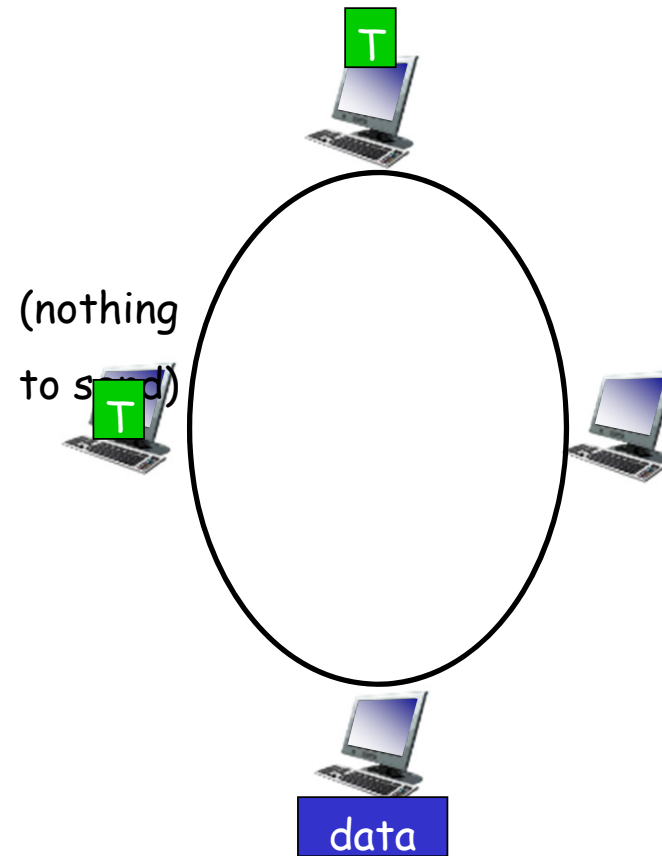
- master node "invites" slave nodes to transmit in turn
- typically used with "dumb" slave devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



# "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

# 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

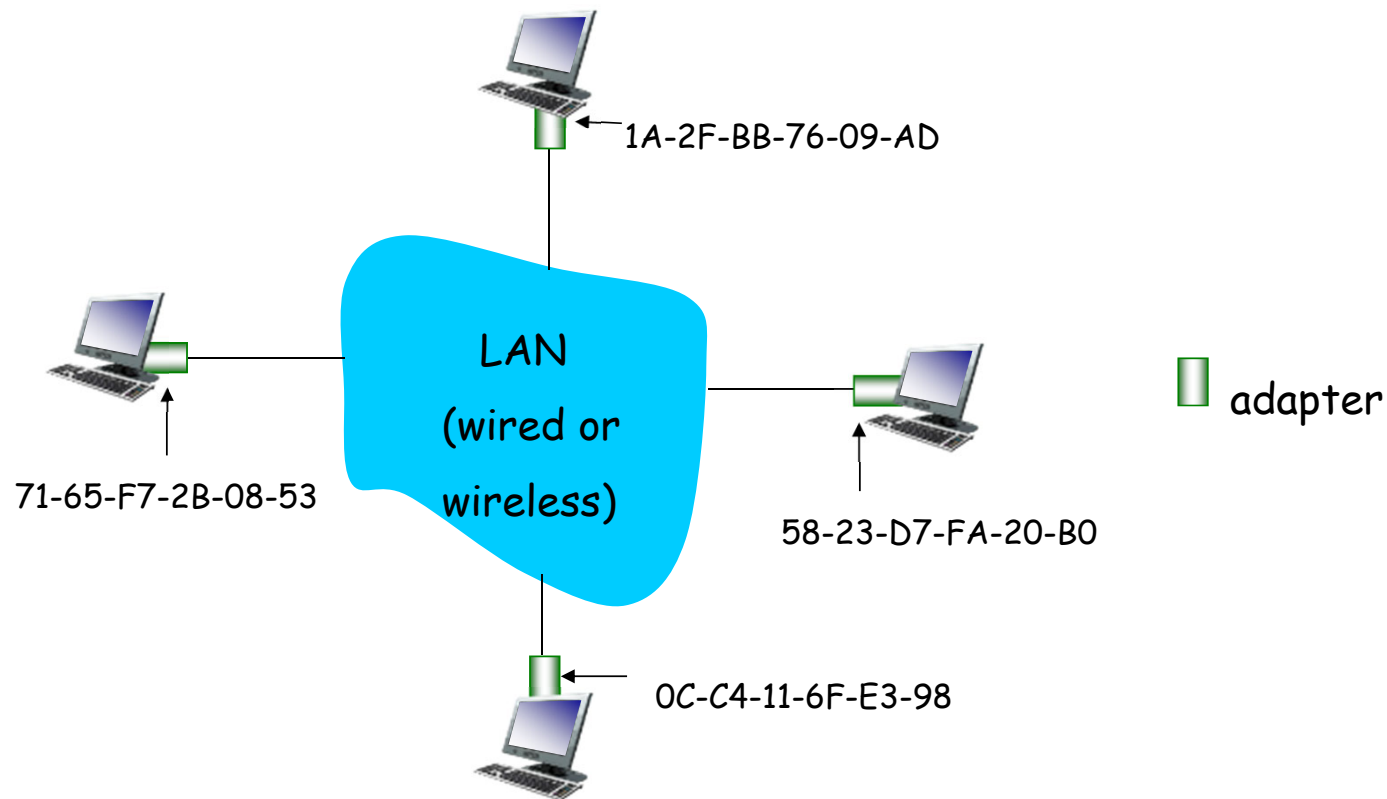
6.7 a day in the life of a web  
request

# 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



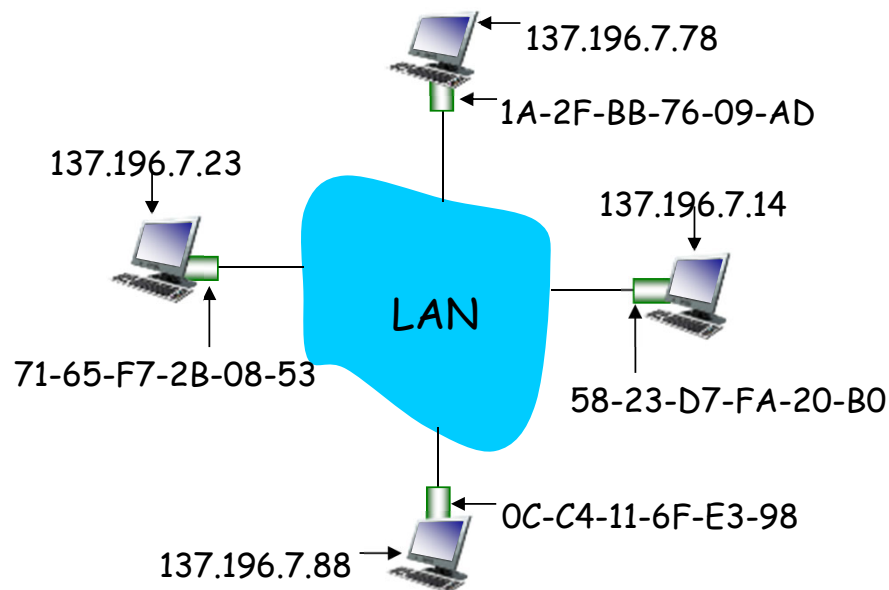


## LAN addresses (more)

- 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 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)

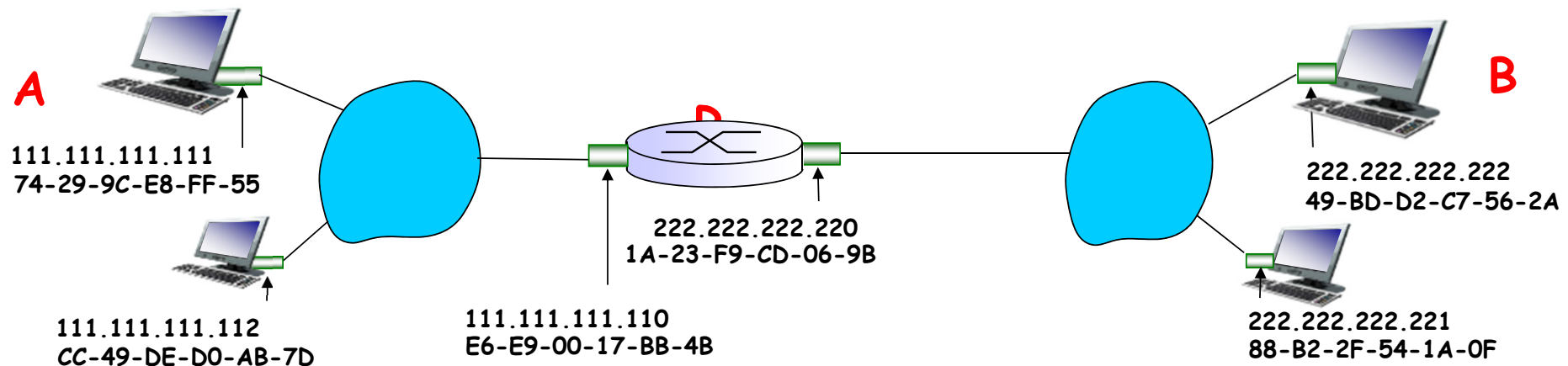
# ARP protocol: same LAN

- 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

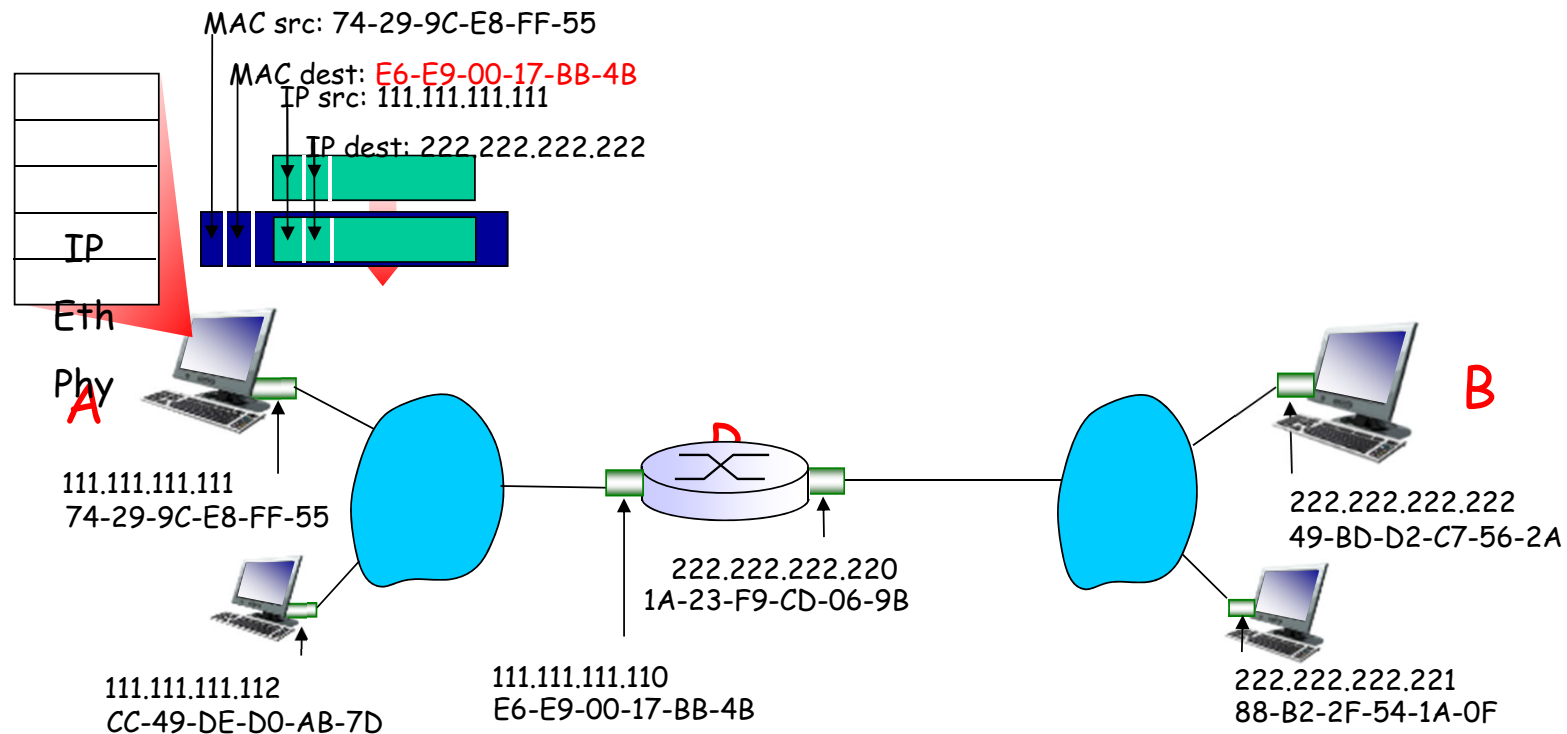
walkthrough: **send datagram from A to B via R**

- 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?)



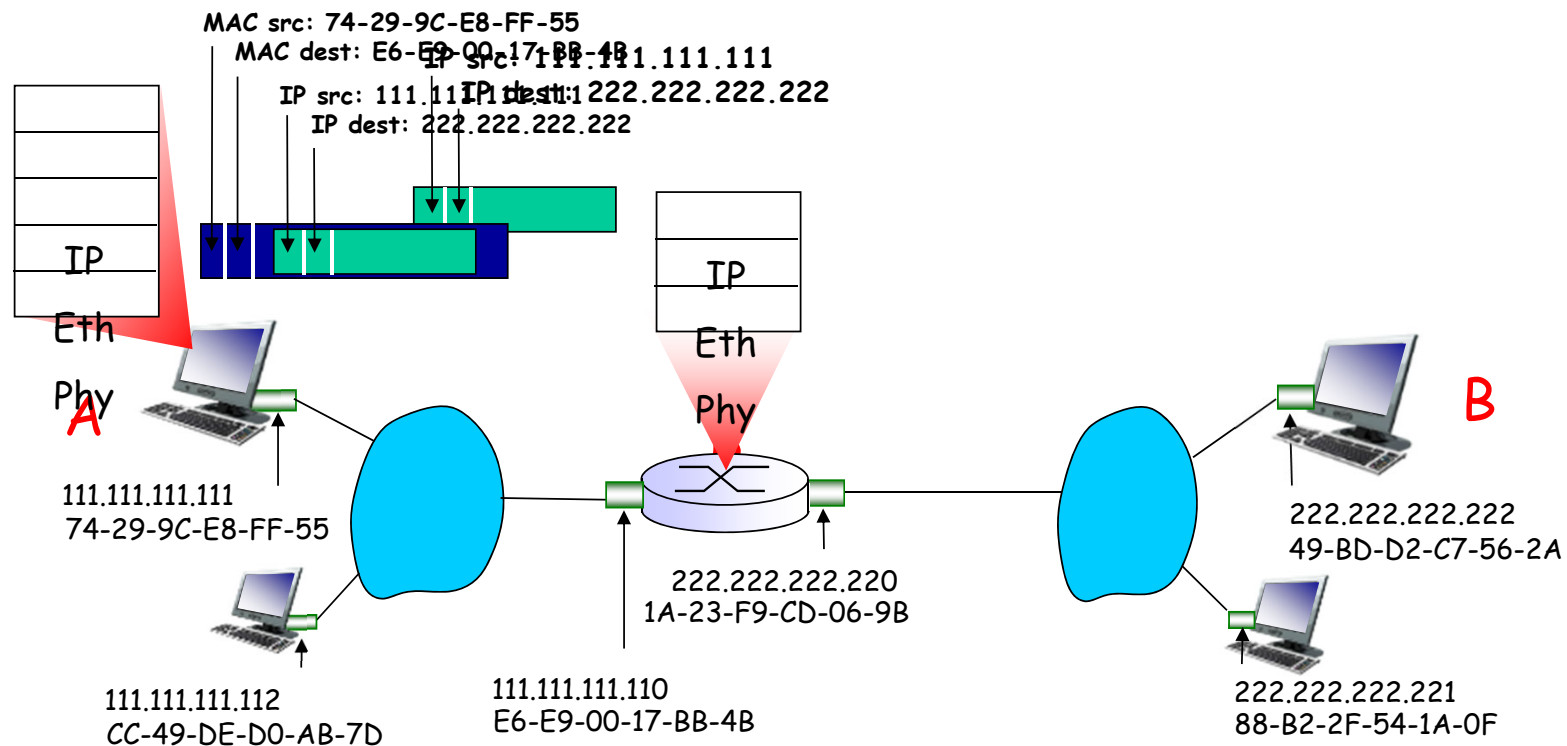
# Addressing: routing to another LAN

- 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



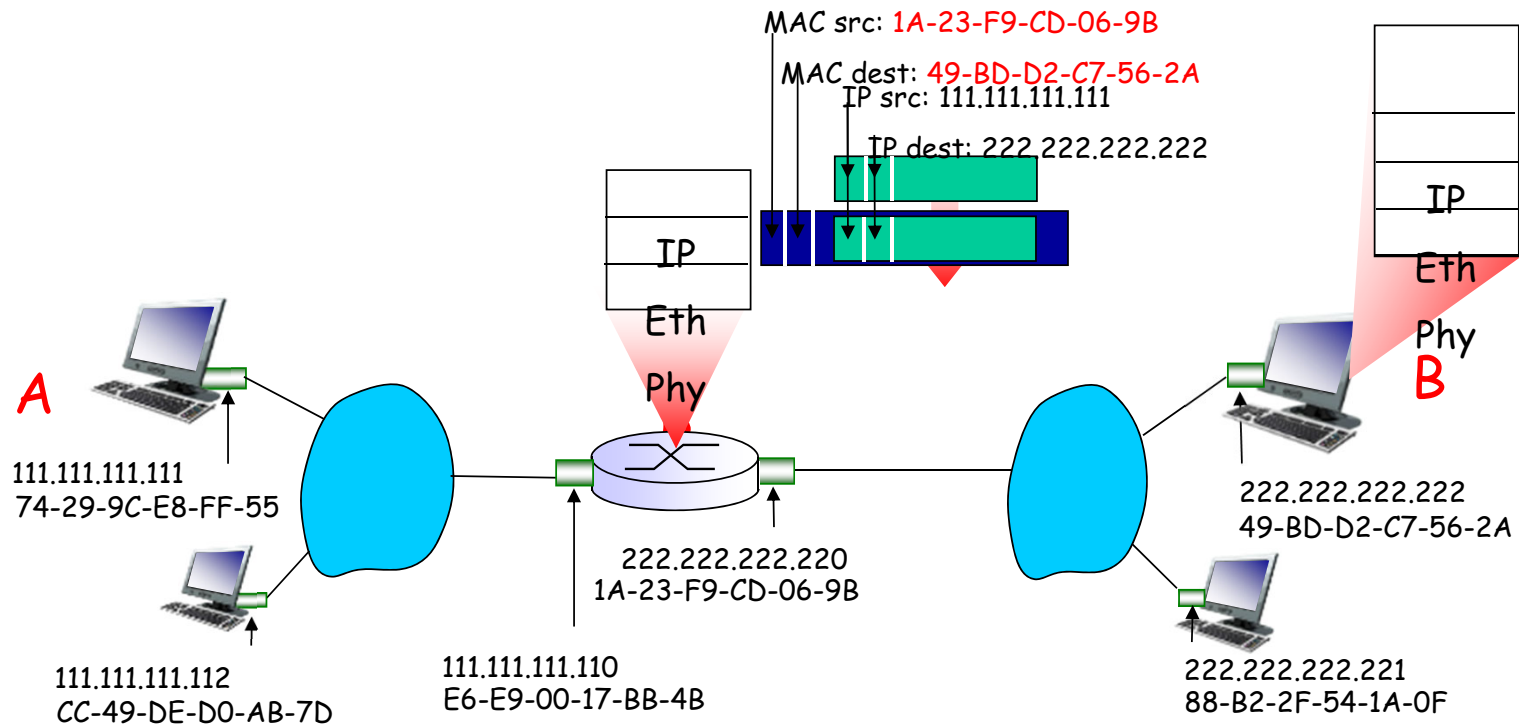
# Addressing: routing to another LAN

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



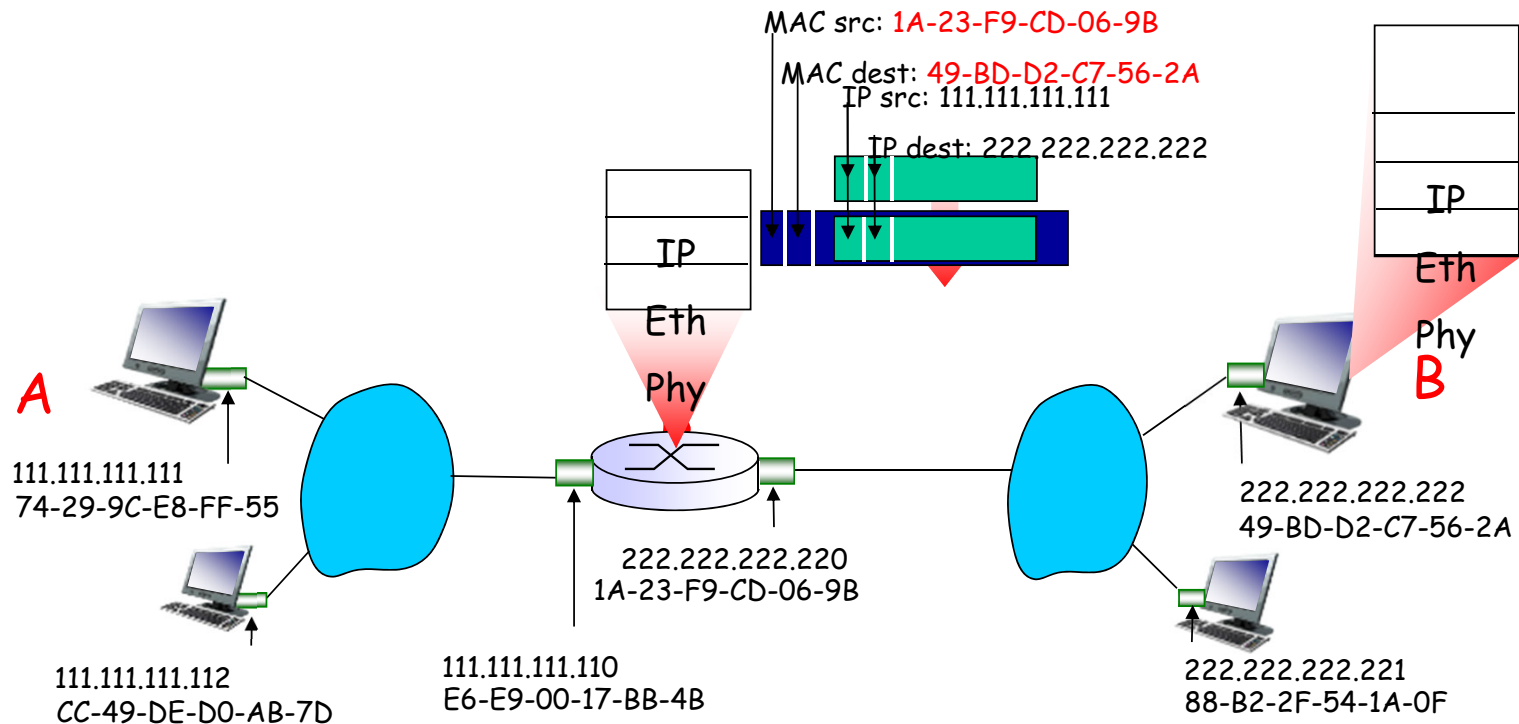
# Addressing: routing to another LAN

- 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

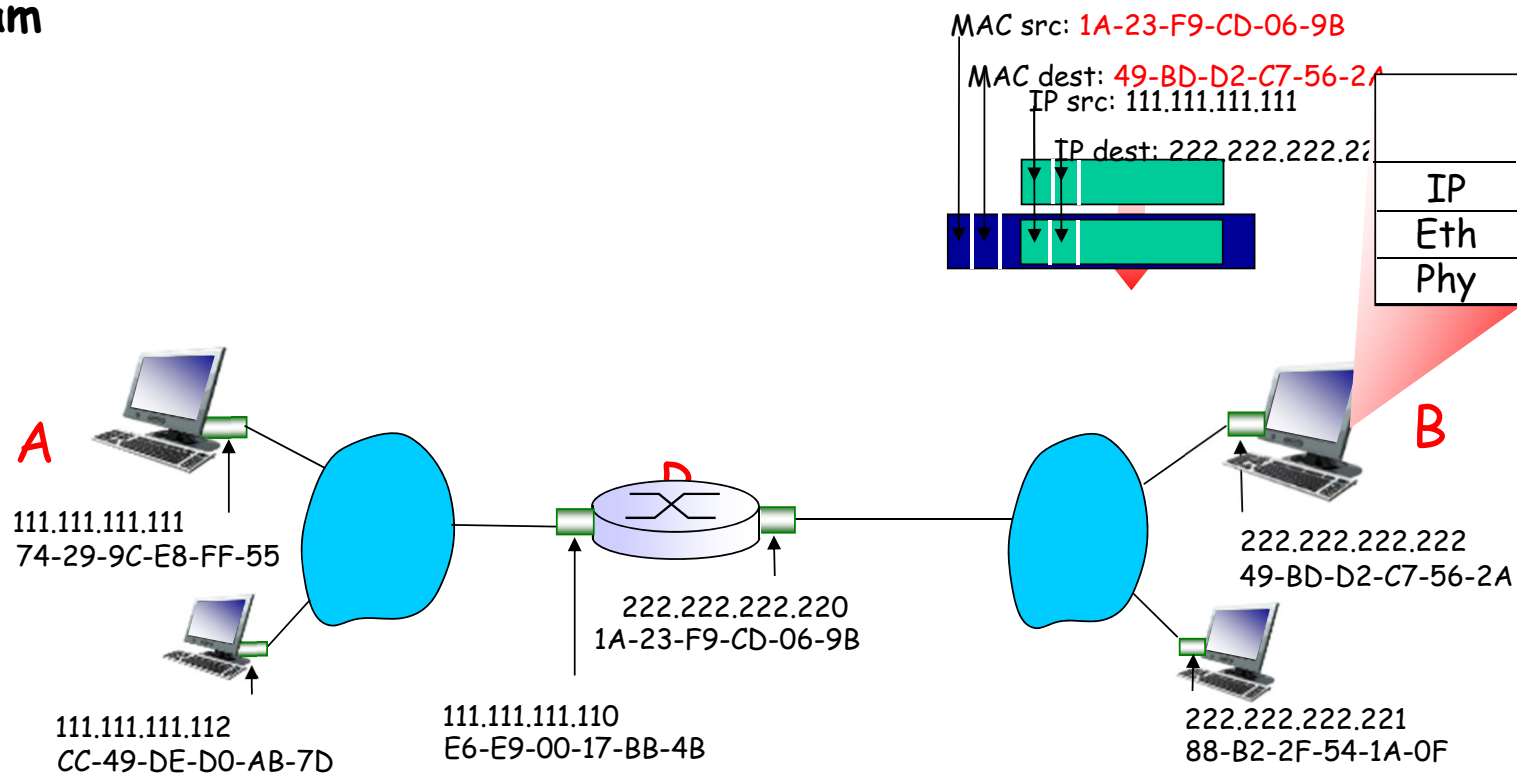




# Addressing: routing to another LAN

- 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/](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

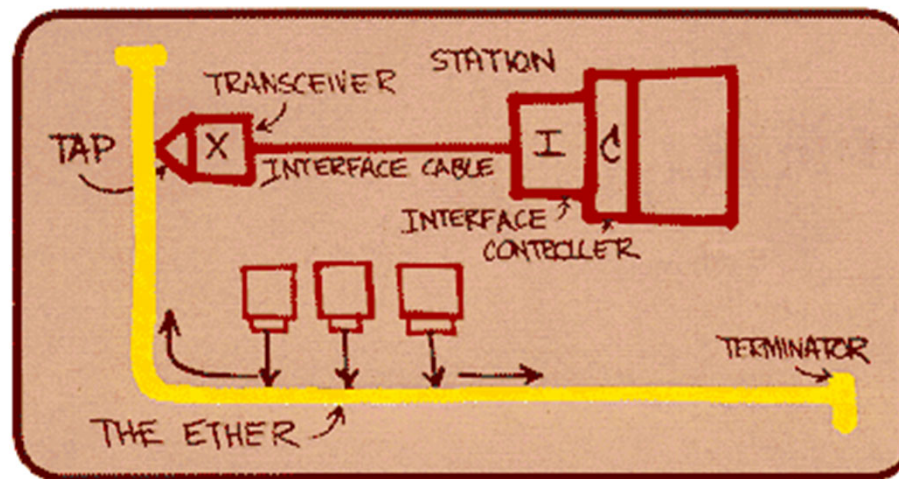
6.6 data center networking

6.7 a day in the life of a web  
request

# Ethernet

“dominant” wired LAN technology:

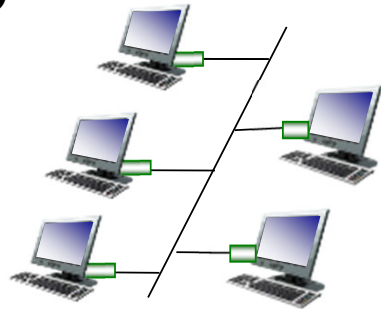
- single chip, multiple speeds (e.g., Broadcom BCM5761)
- first widely used LAN technology
- simpler, cheap
- kept up with speed race: 10 Mbps – 10 Gbps



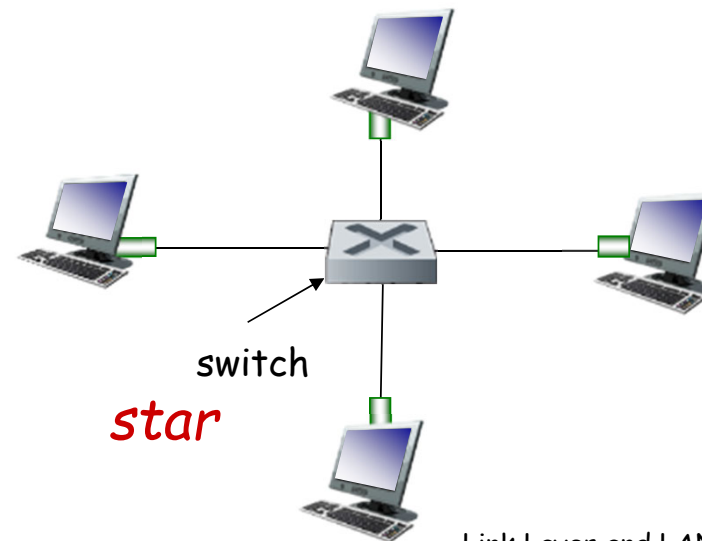
*Metcalfe's Ethernet sketch*

# 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



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

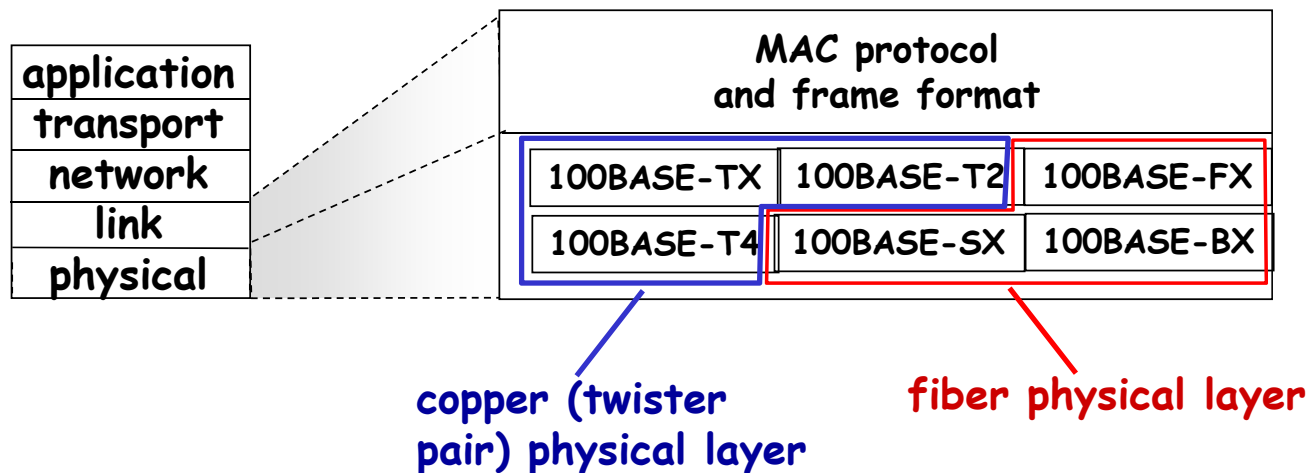


# Ethernet: unreliable, connectionless

- **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 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

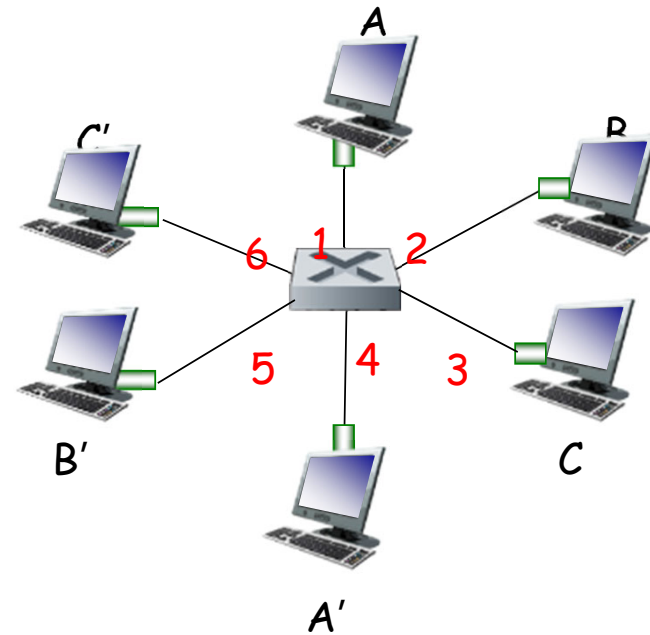
6.7 a day in the life of a web  
request

# 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



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

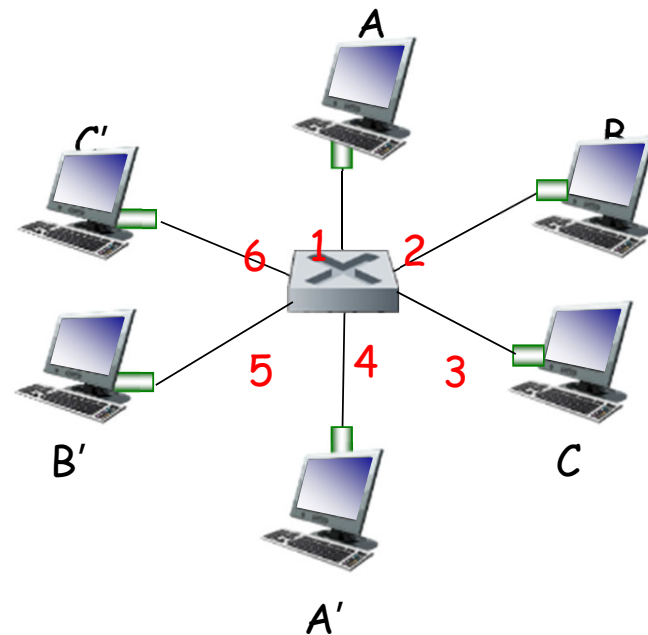
# Switch forwarding table

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?

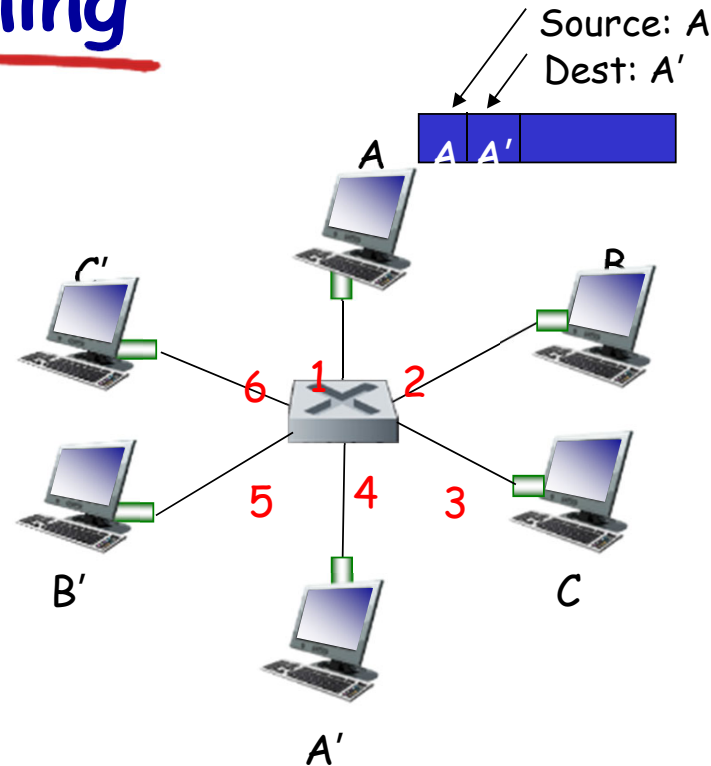
- something like a routing protocol?



*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)

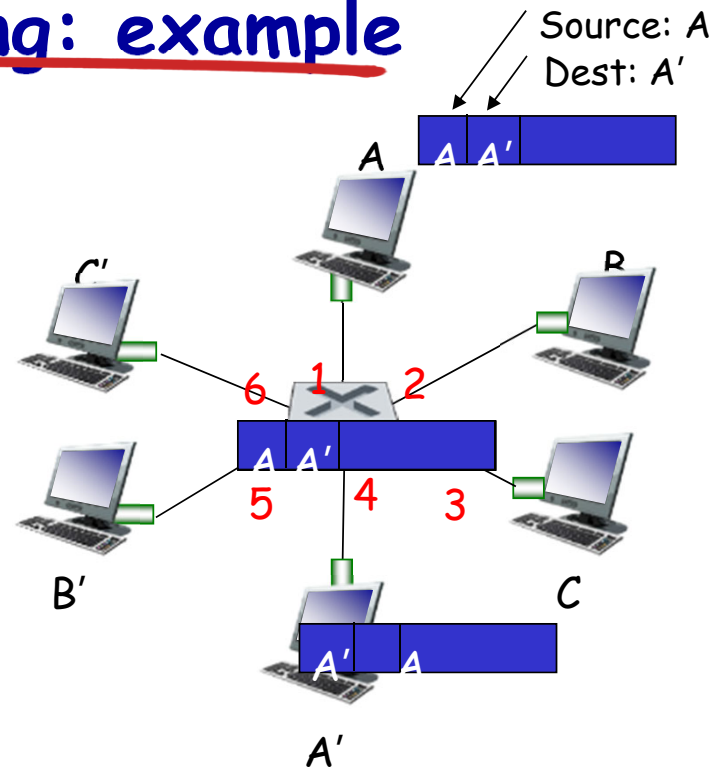
# Switch: frame filtering/forwarding

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*

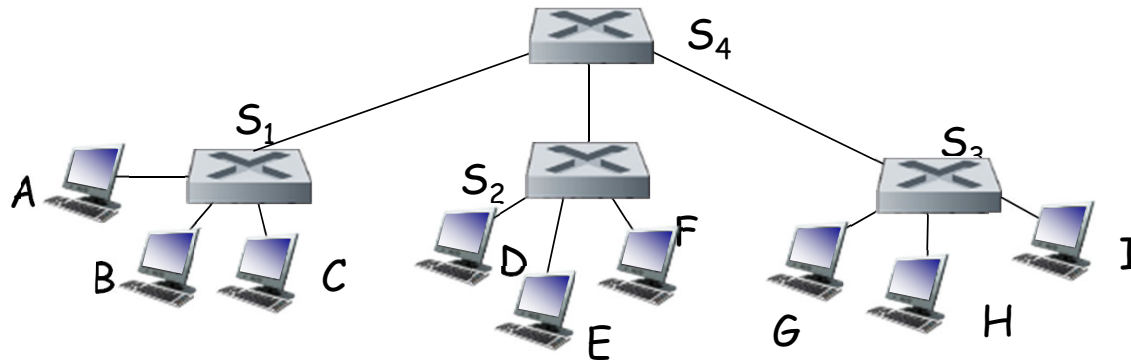


MAC addr	interface	TTL
A	1	60
A'	4	60

*switch table*  
*(initially empty)*

# Interconnecting switches

self-learning switches can be connected together:



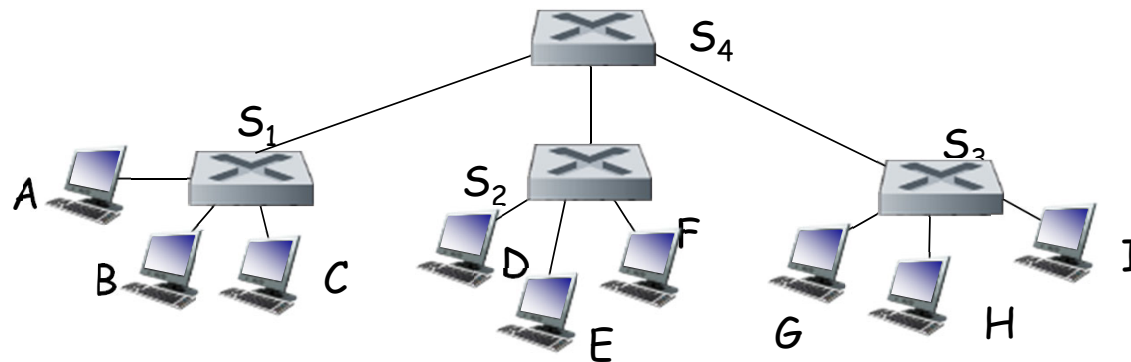
**Q:** sending from A to G - how does S<sub>1</sub> know to forward frame destined to G via S<sub>4</sub> and S<sub>3</sub>?

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



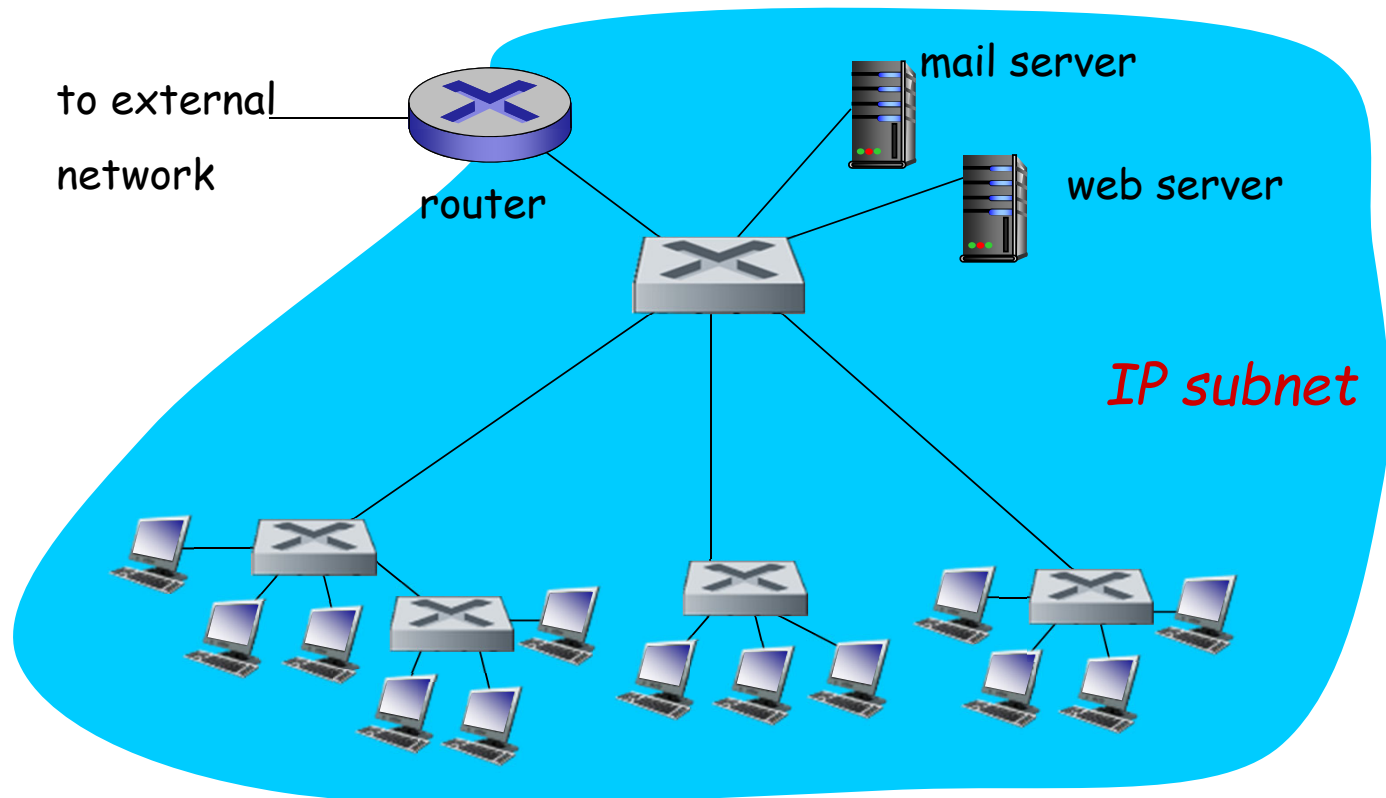
# Self-learning multi-switch example

Suppose C sends frame to I, I responds to C



- Q: show switch tables and packet forwarding in S<sub>1</sub>, S<sub>2</sub>, S<sub>3</sub>, S<sub>4</sub>

# Institutional network



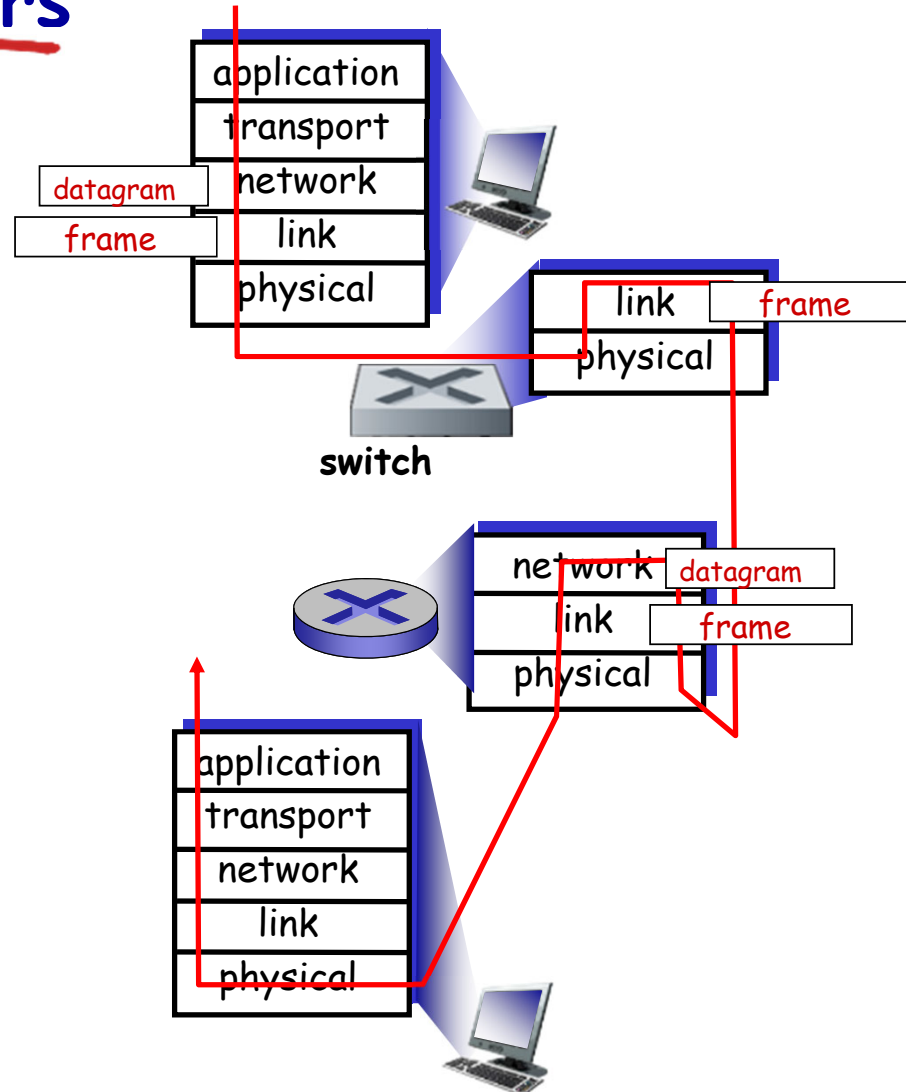
# Switches vs. routers

both are store-and-forward:

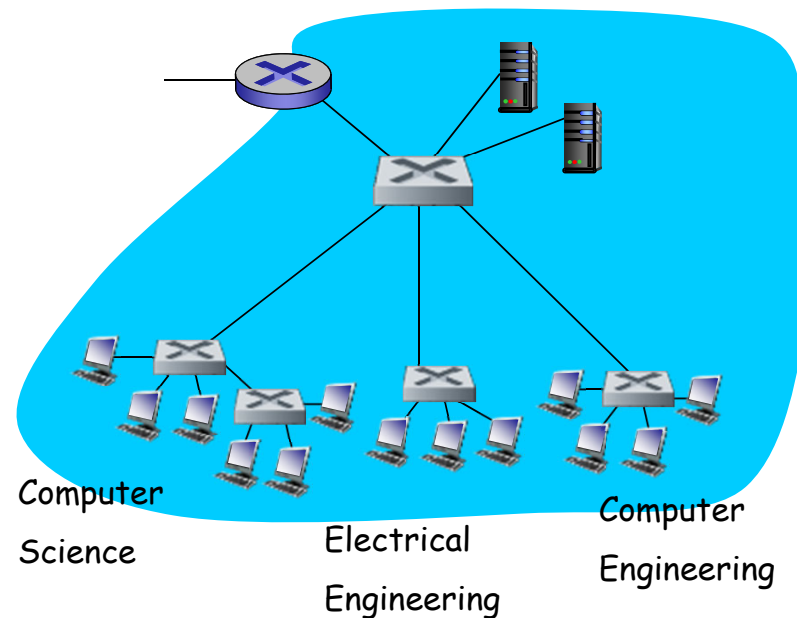
- **routers:** network-layer devices (examine network-layer headers)
- **switches:** link-layer devices (examine link-layer headers)

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

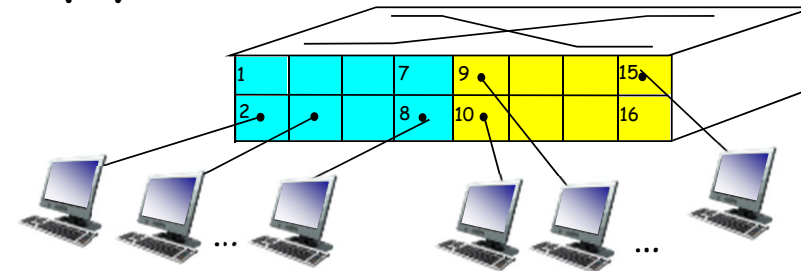
# VLANs

*Virtual Local*

*Area Network*

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

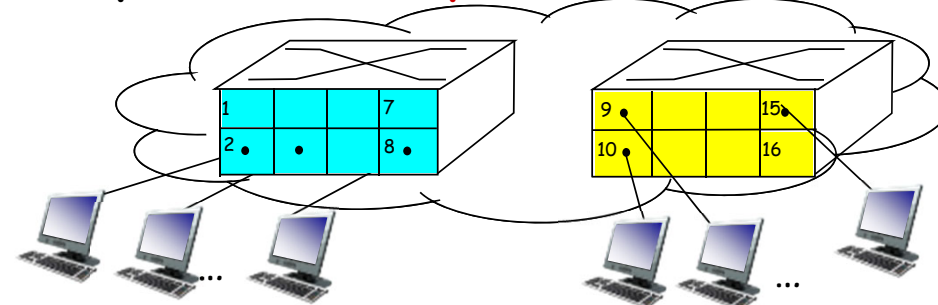
**port-based VLAN:** switch ports grouped (by switch management software) so that *single* physical switch .....



Electrical Engineering  
(VLAN ports 1-8)

Computer Science  
(VLAN ports 9-15)

... operates as **multiple** virtual switches



Electrical Engineering  
(VLAN ports 1-8)

Computer Science  
(VLAN ports 9-16)

# Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

6.4 LANs

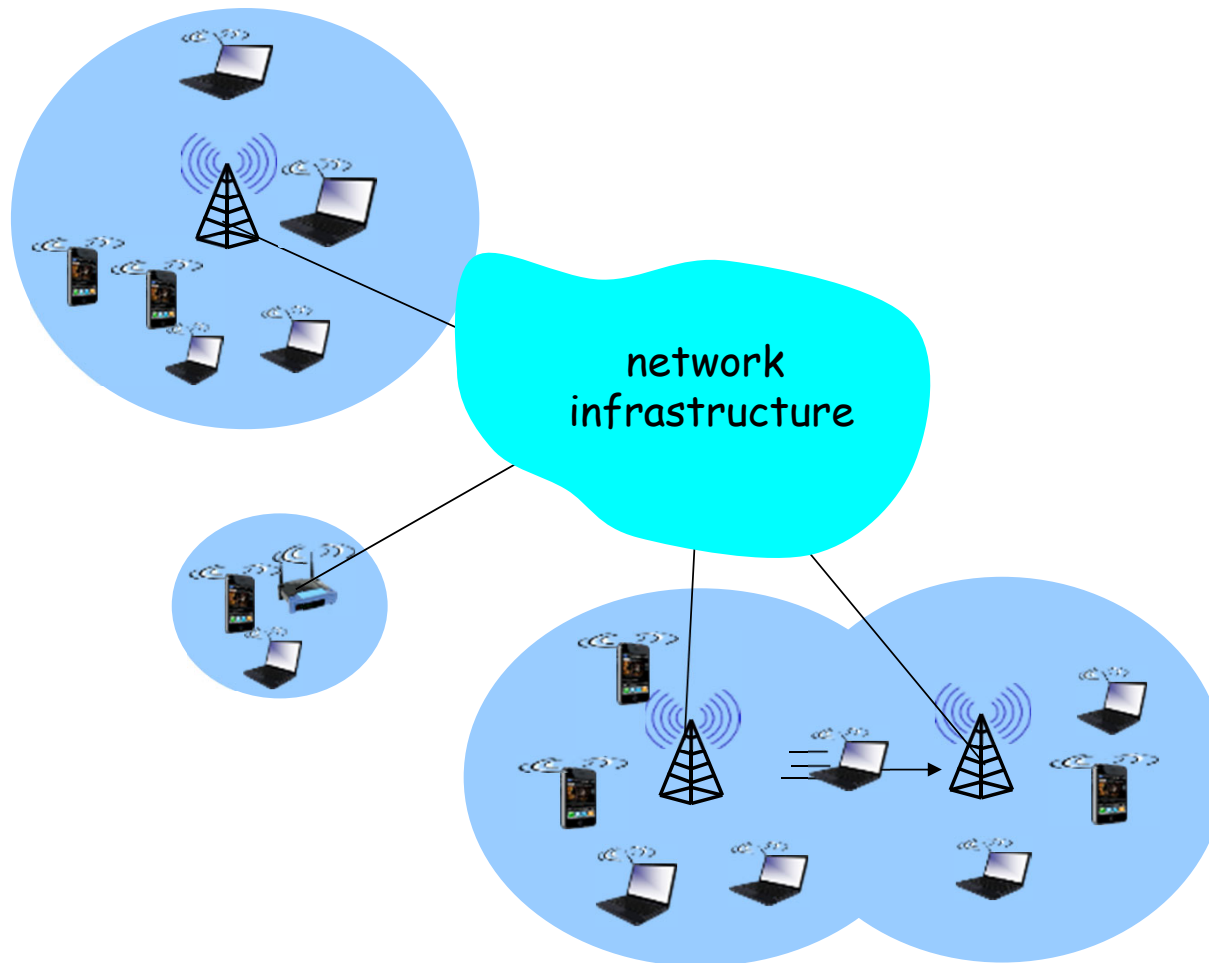
- addressing, ARP
- Ethernet
- switches
- VLANs

6.5 link virtualization: MPLS

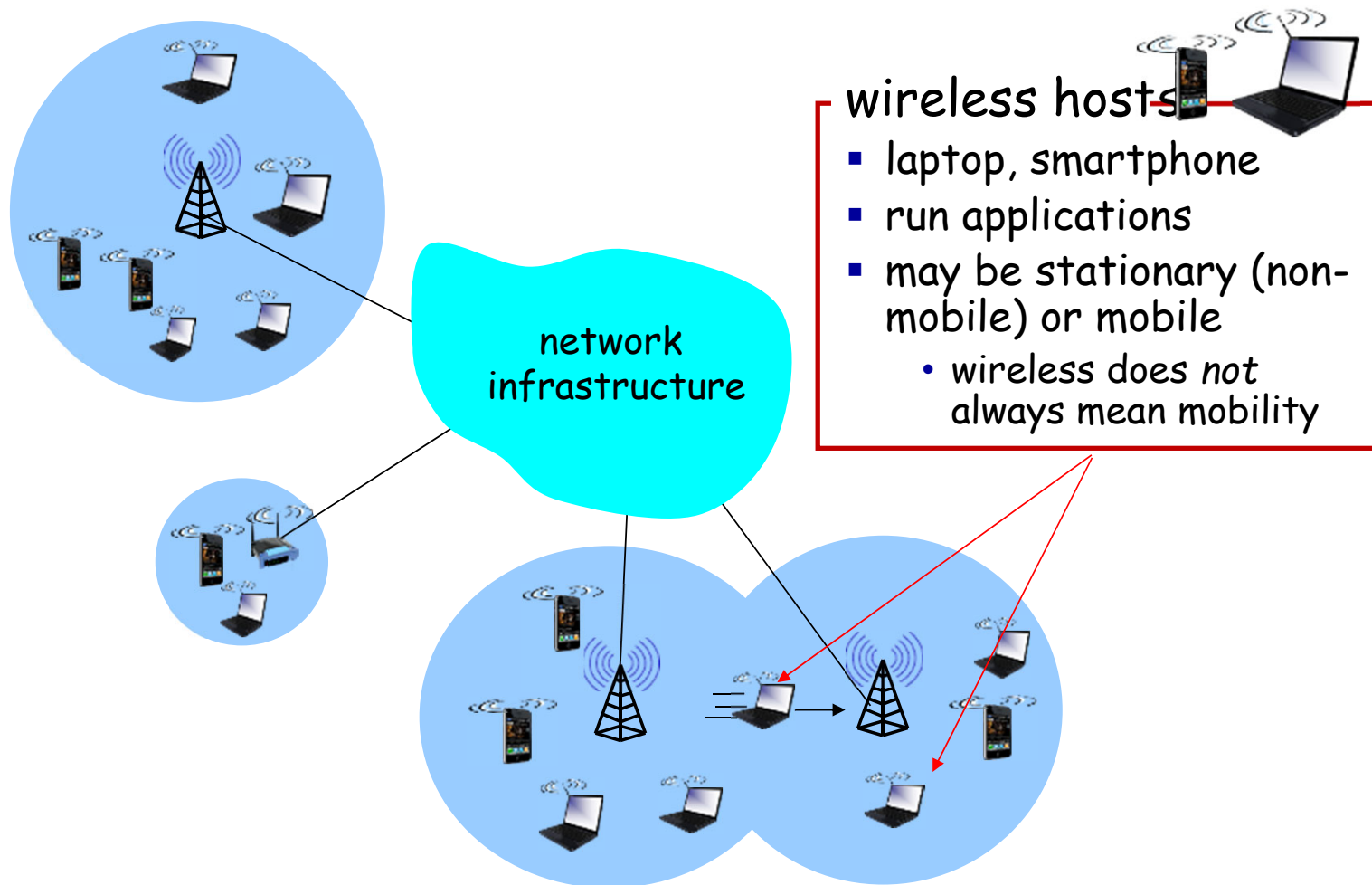
6.6 data center networking

6.7 a day in the life of a web  
request

# Elements of a wireless network

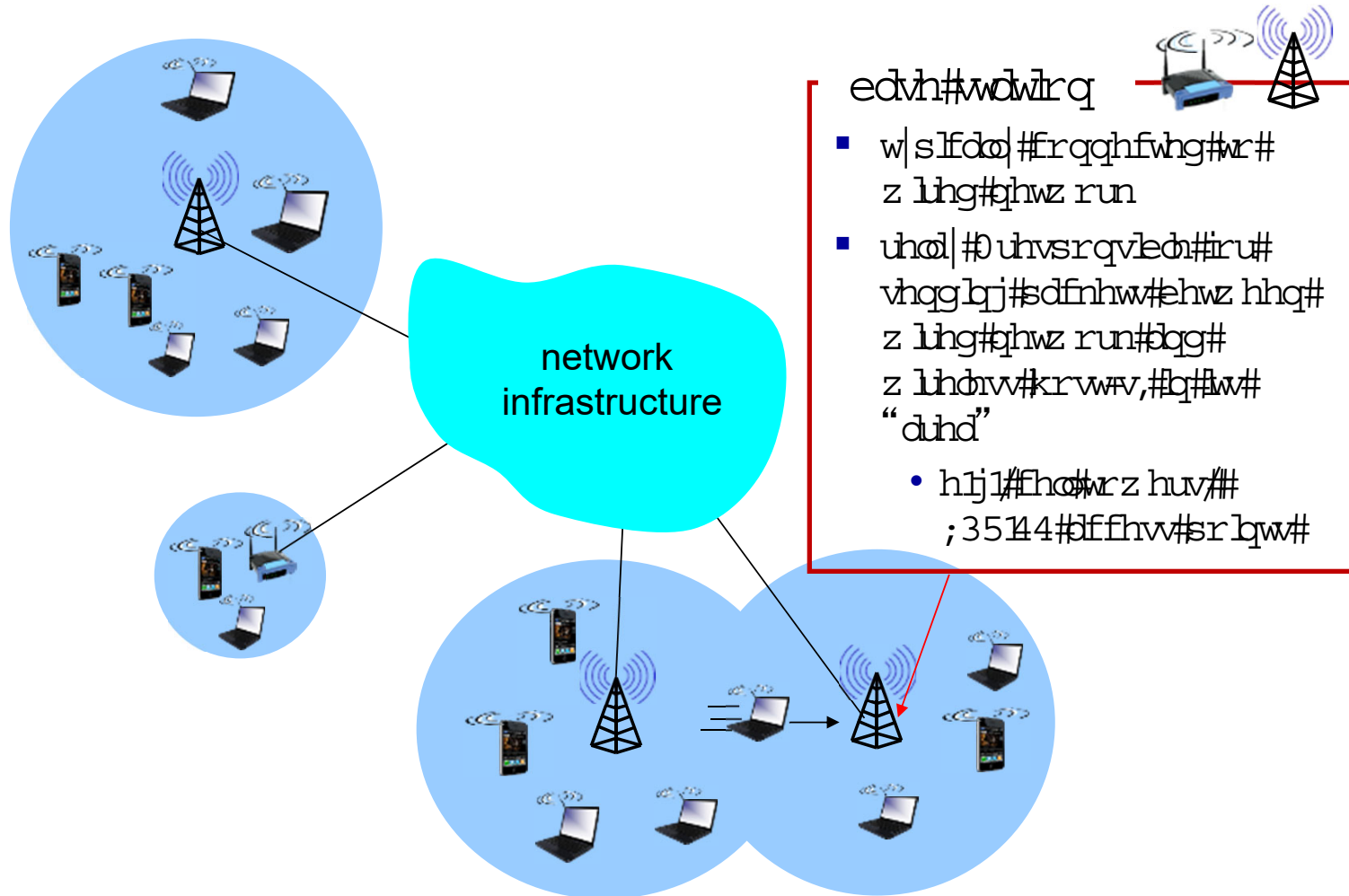


# Elements of a wireless network

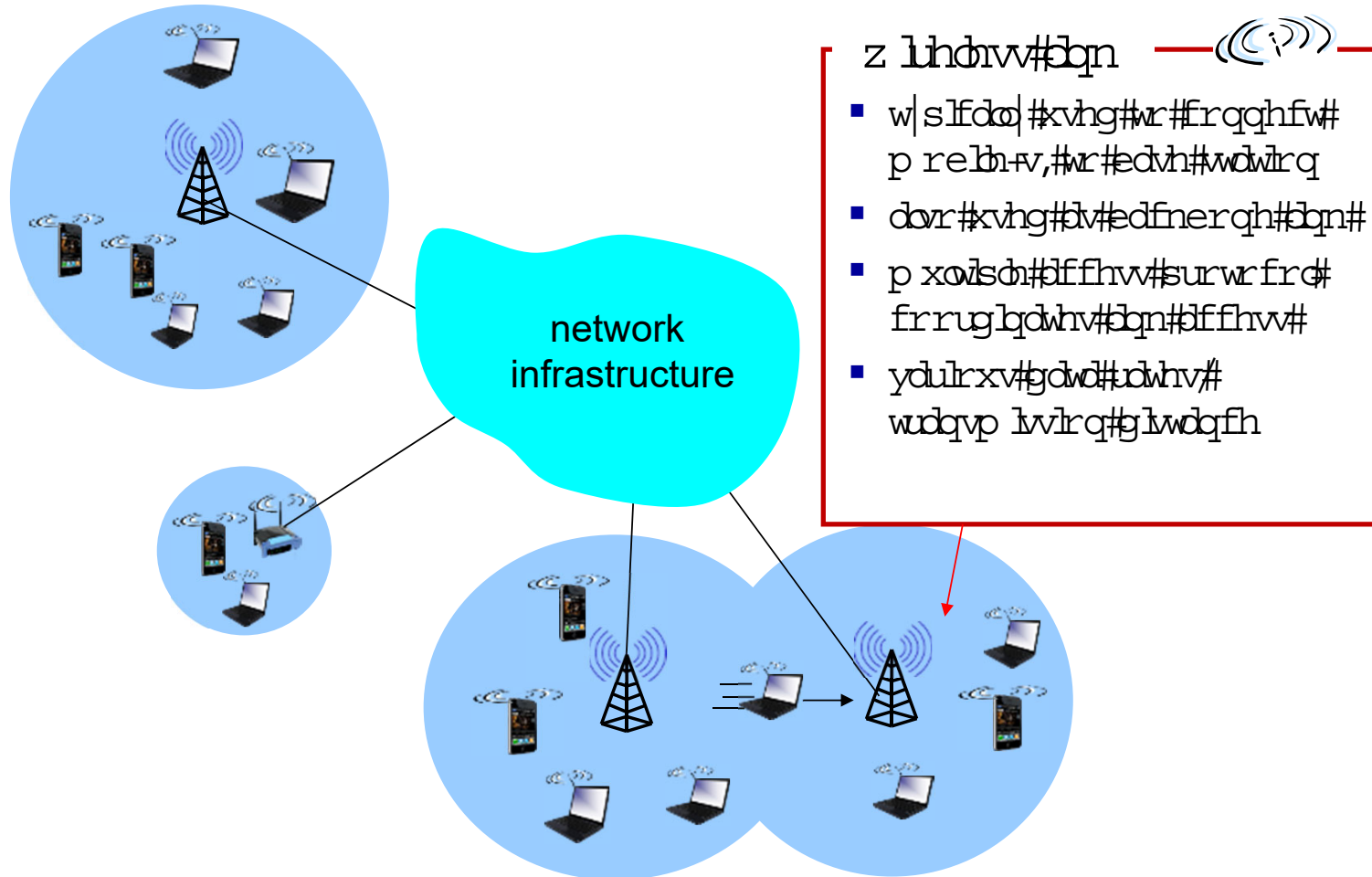




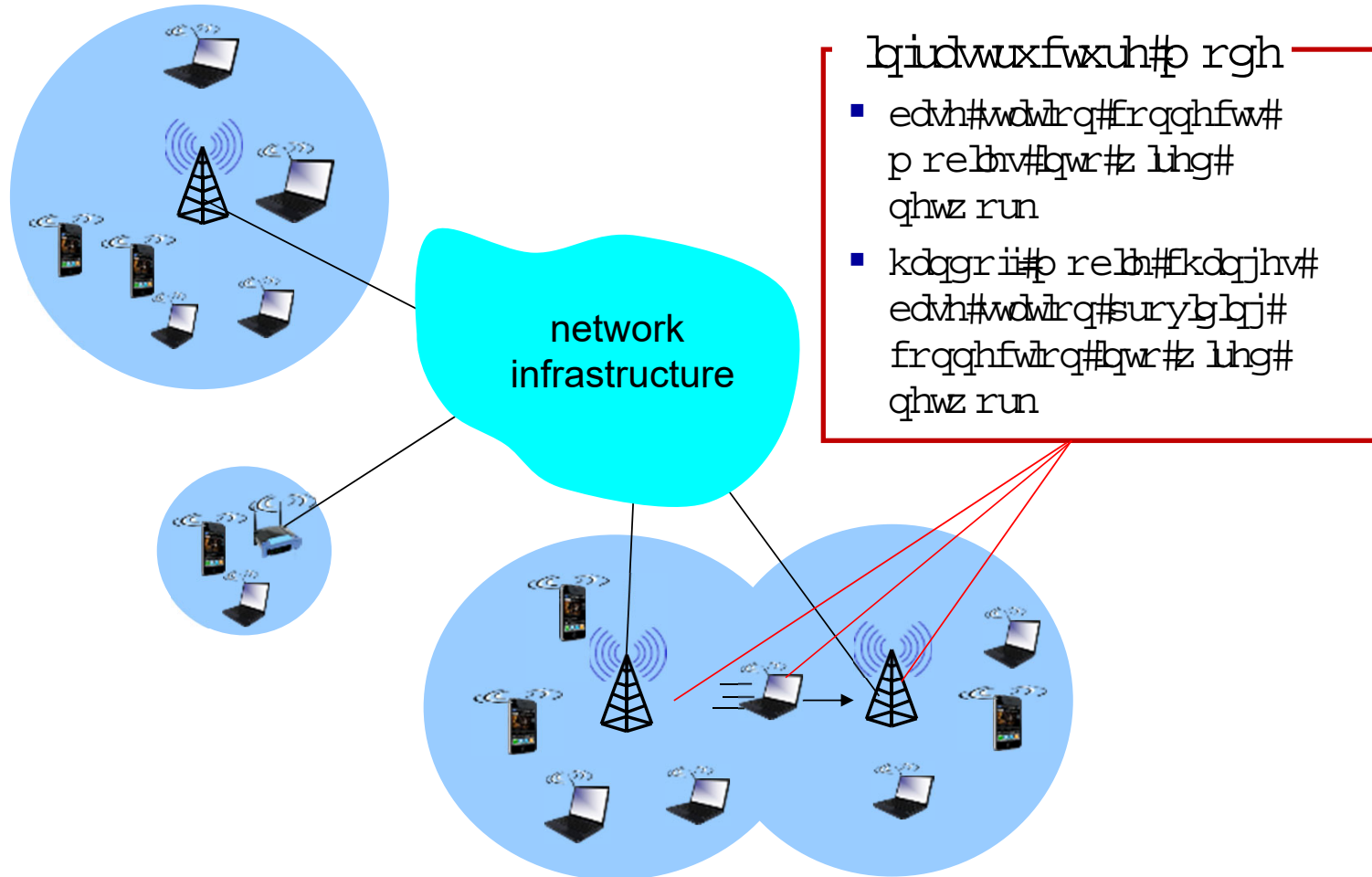
# How how the network infrastructure run



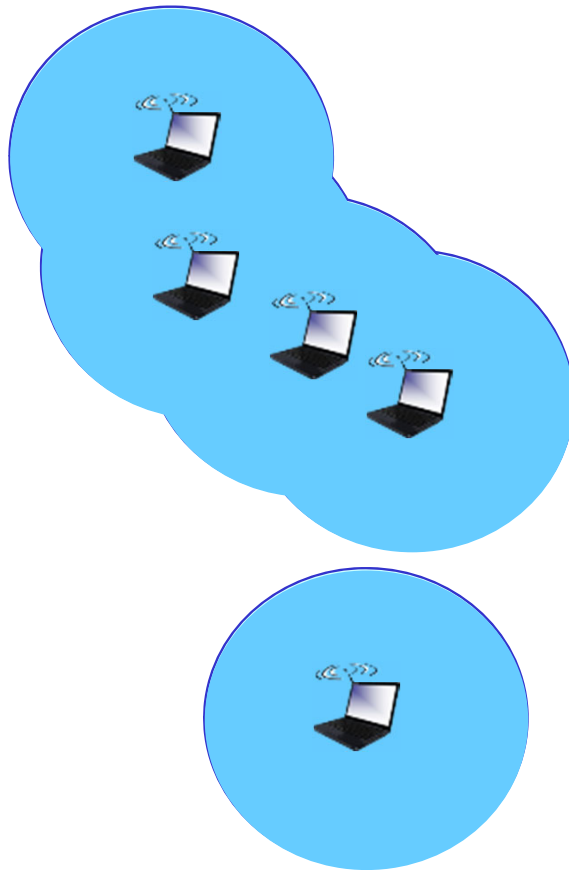
# Elements of a wireless network



# How the network runs



# How how to run



dg#krf#p rgh

- qr#edv#wdwtrqv
- qrghv#fdq#rqd#  
wdqvp l#wr#wkhu#  
qrghv#z lwb#bn#  
fryhudjh
- qrghv#rujdqljh#  
wkhp vhoqv#qwr#b#  
qhwz run#urxwh#  
dp rqj#wkhp vhoqv

## Important differences from wired link ...

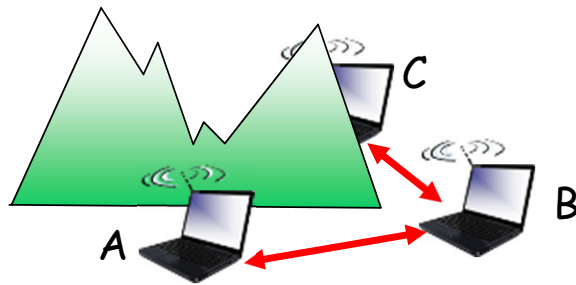
**important** differences from wired link ....

- **decreased signal strength:** radio signal attenuates as it propagates through matter (path loss)
- **interference from other sources:** standardized wireless network frequencies (e.g., 2.4 GHz) shared by other devices (e.g., phone); devices (motors) interfere as well
- **multipath propagation:** radio signal reflects off objects ground, arriving at destination at slightly different times

.... make communication across (even a point to point) wireless link much more “difficult”

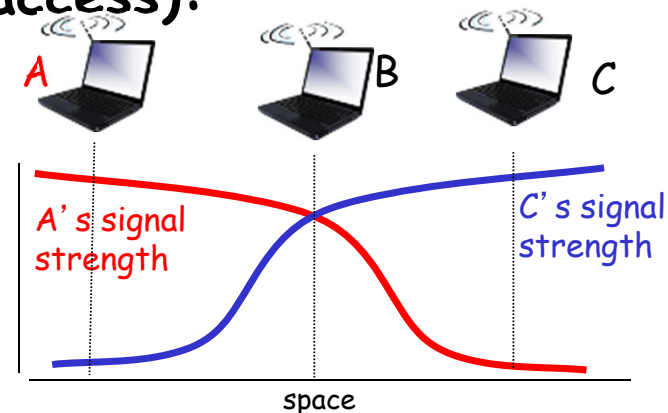
# Wireless network characteristics

Multiple wireless senders and receivers create additional problems (beyond multiple access):



## *Hidden terminal problem*

- B, A hear each other
- B, C hear each other
- A, C can not hear each other means A, C unaware of their interference at B



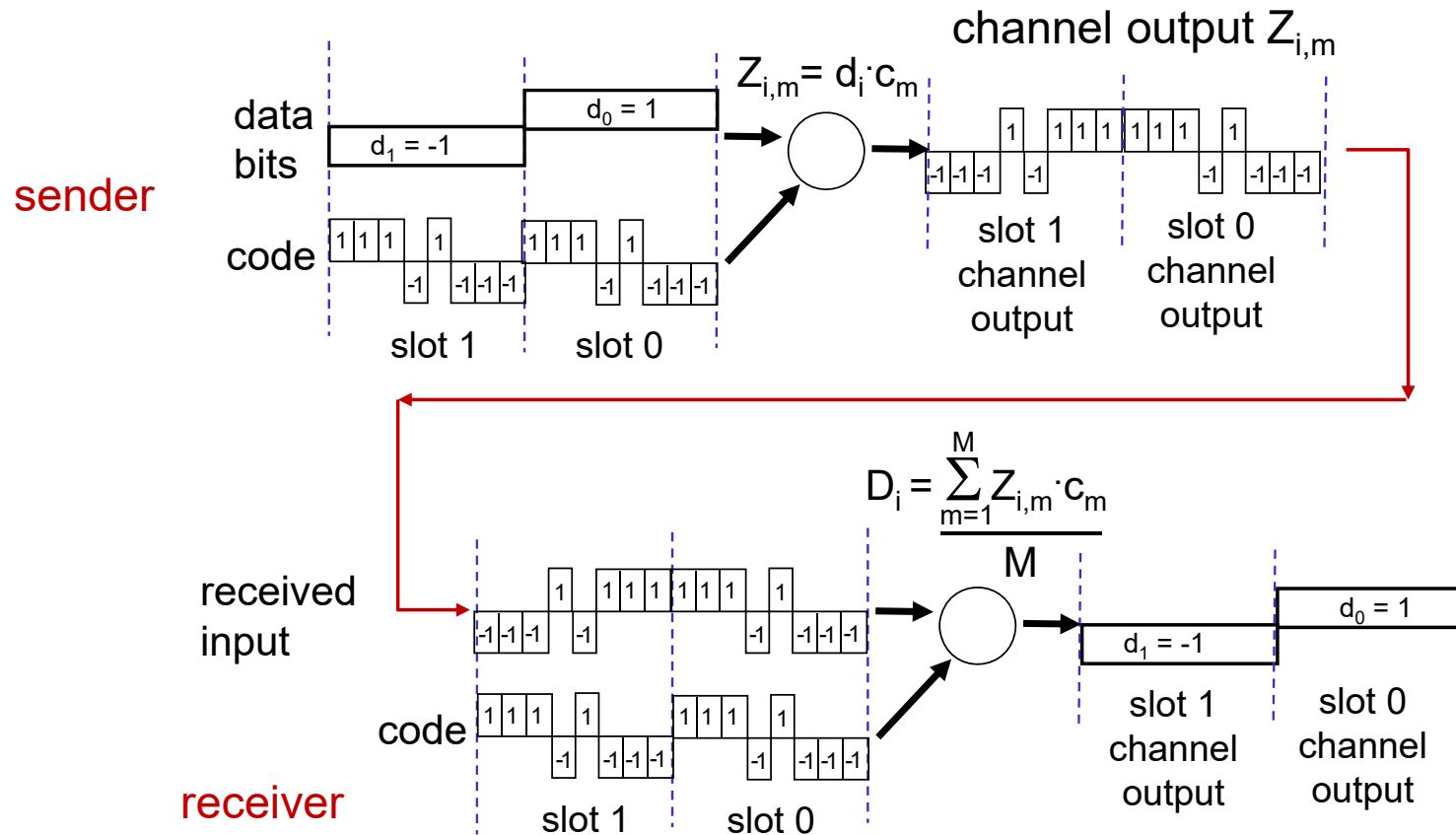
## *Signal attenuation:*

- B, A hear each other
- B, C hear each other
- A, C can not hear each other interfering at B

# Code Division Multiple Access (CDMA)

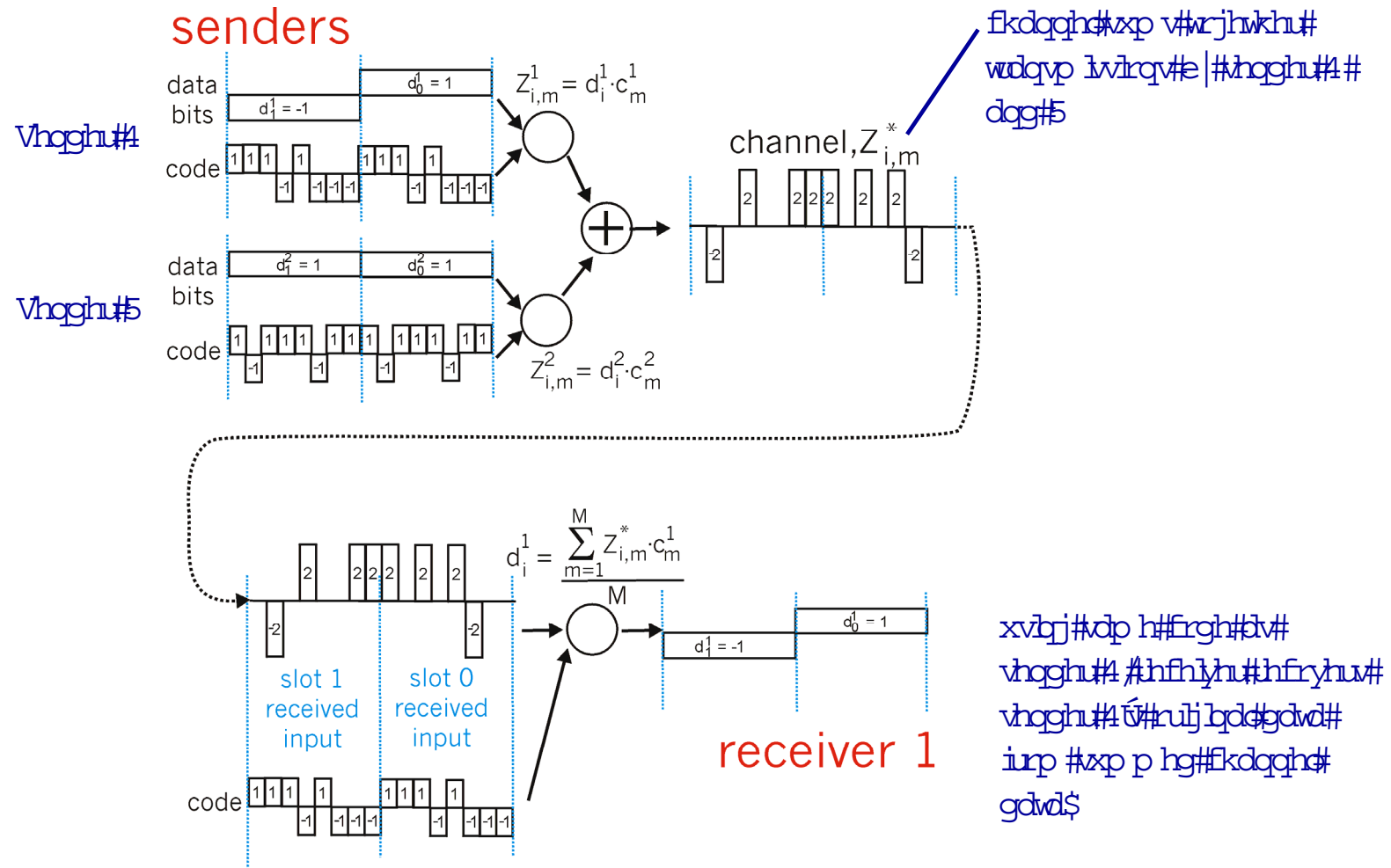
- unique “code” assigned to each user; i.e., code set partitioning
  - all users share same frequency, but each user has own “chipping” sequence (i.e., code) to encode data
  - allows multiple users to “coexist” and transmit simultaneously with minimal interference (if codes are “orthogonal”)
- *encoded signal* = (original data) X (chipping sequence)
- *decoding*: inner-product of encoded signal and chipping sequence

# CDMA encode/decode





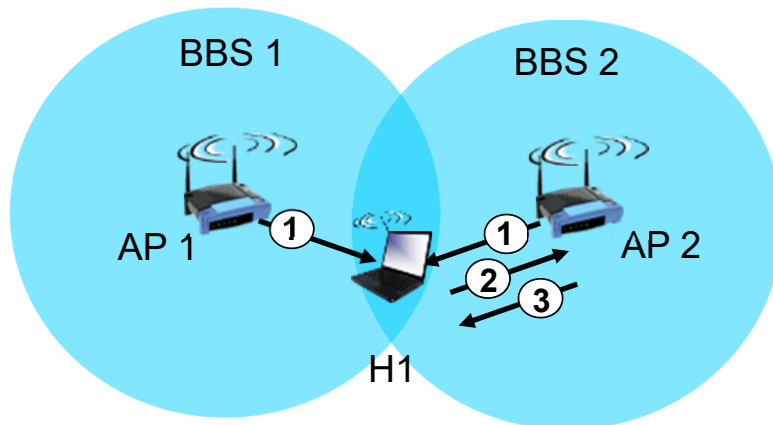
# FDMA



# 802.11: Channels, association

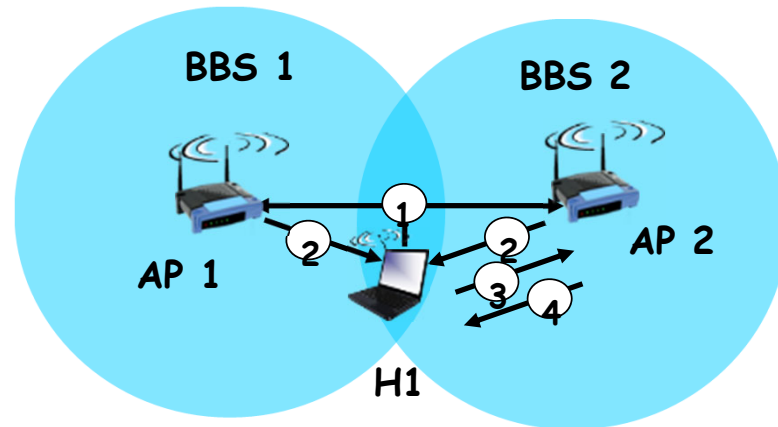
- 802.11b: 2.4GHz-2.485GHz spectrum divided into 11 channels at different frequencies
  - AP admin chooses frequency for AP
  - interference possible: channel can be same as that chosen by neighboring AP!
- host: must *associate* with an AP
  - scans channels, listening for *beacon frames* containing AP's name (SSID) and MAC address
  - selects AP to associate with
  - may perform authentication [Chapter 8]
  - will typically run DHCP to get IP address in AP's subnet

# 802.11: passive/active scanning



## passive scanning:

- (1) beacon frames sent from APs
- (2) association Request frame sent: H1 to selected AP
- (3) association Response frame sent from selected AP to H1

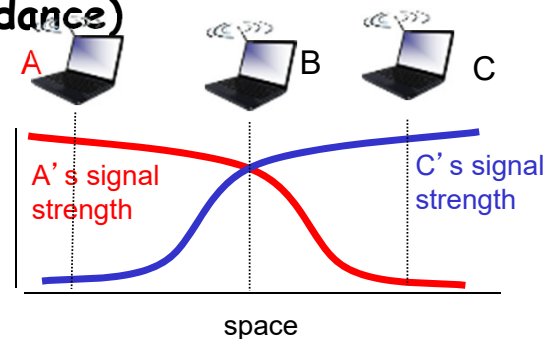
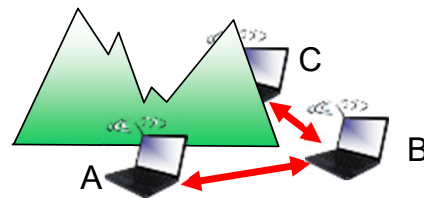


## active scanning:

- (1) Probe Request frame broadcast from H1
- (2) Probe Response frames sent from APs
- (3) Association Request frame sent: H1 to selected AP
- (4) Association Response frame sent from selected AP to H1

## IIIH#; 35144#b xovs#dfhvw

- avoid collisions: 2+ nodes transmitting at same time
- 802.11: CSMA - sense before transmitting
  - don't collide with ongoing transmission by other node
- 802.11: *no collision detection!*
  - difficult to receive (sense collisions) when transmitting due to weak received signals (fading)
  - can't sense all collisions in any case: hidden terminal, fading
  - goal: **avoid collisions: CSMA/C(ollision)A(voidance)**



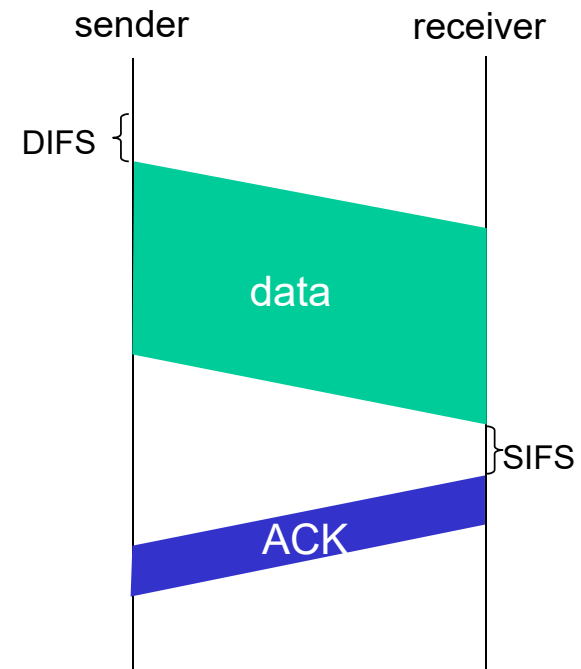
# IEEE 802.11 DCF Survival

## 802.11 sender

- 1 if sense channel idle for DIFS then  
transmit entire frame (no CD)
- 2 if sense channel busy then  
start random backoff time  
timer counts down while channel idle  
transmit when timer expires  
if no ACK, increase random backoff interval, repeat 2

## 802.11 receiver

- if frame received OK  
return ACK after SIFS (ACK needed due to hidden terminal problem)



## Dvrbbj#frövlrav#o ruh,

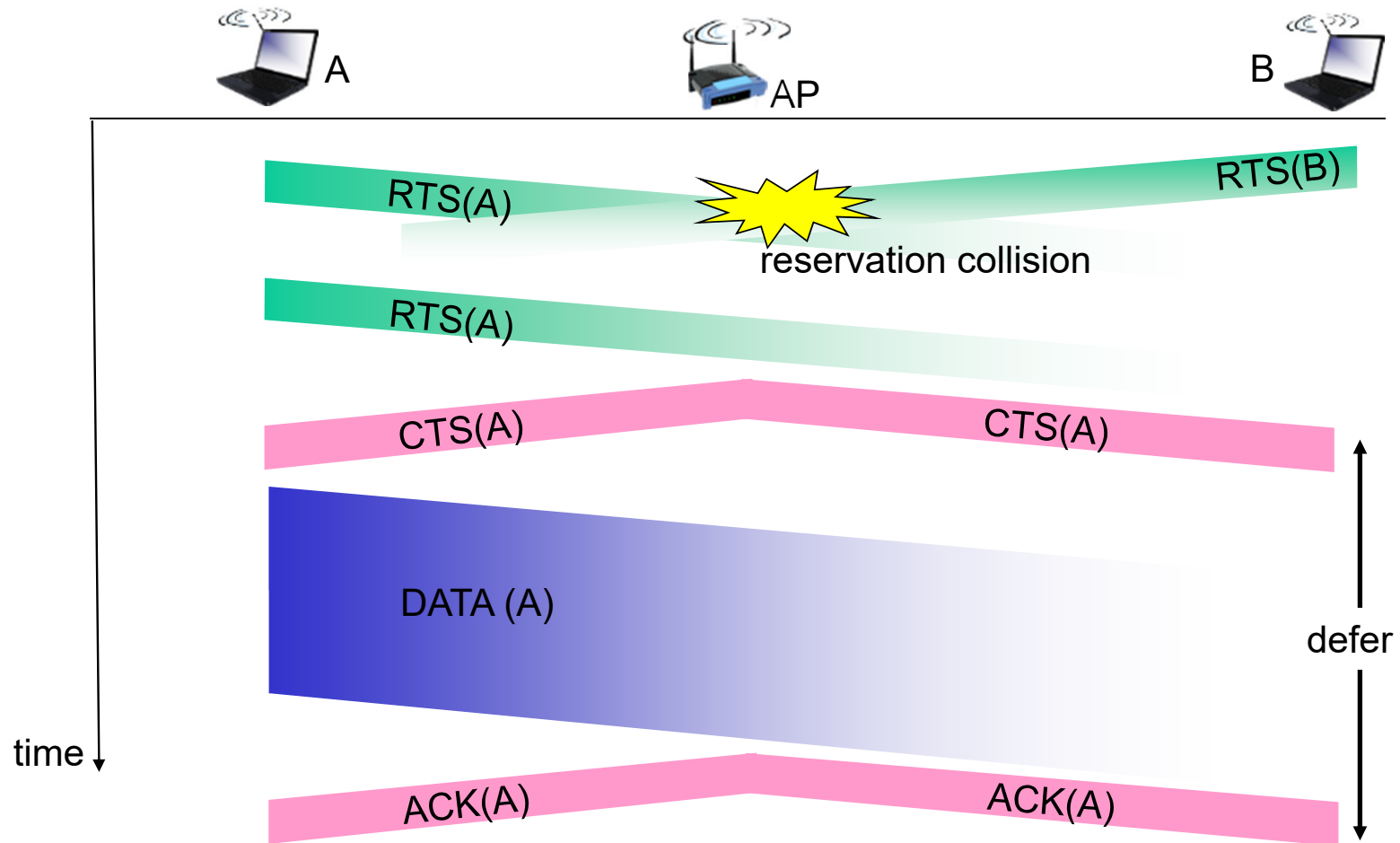
**idea:** allow sender to “reserve” channel rather than random access of data

frames: avoid collisions of long data frames

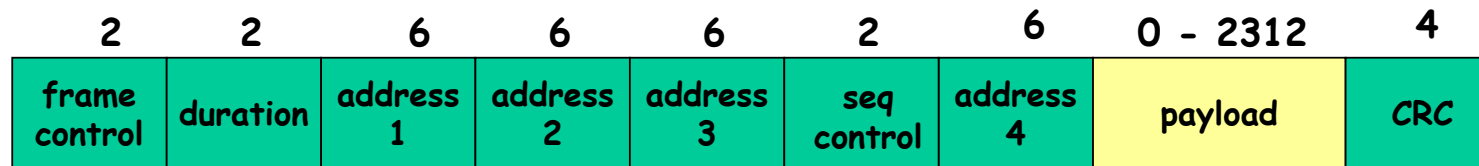
- sender first transmits *small* request-to-send (RTS) packets to BS using CSMA
  - RTSs may still collide with each other (but they’re short)
- BS broadcasts clear-to-send CTS in response to RTS
- CTS heard by all nodes
  - sender transmits data frame
  - other stations defer transmissions

*avoid data frame collisions completely  
using small reservation packets!*

# Flow Control



:35144#iud h=#brquhwvbj



**Address 1:** MAC address of wireless host or AP to receive this frame

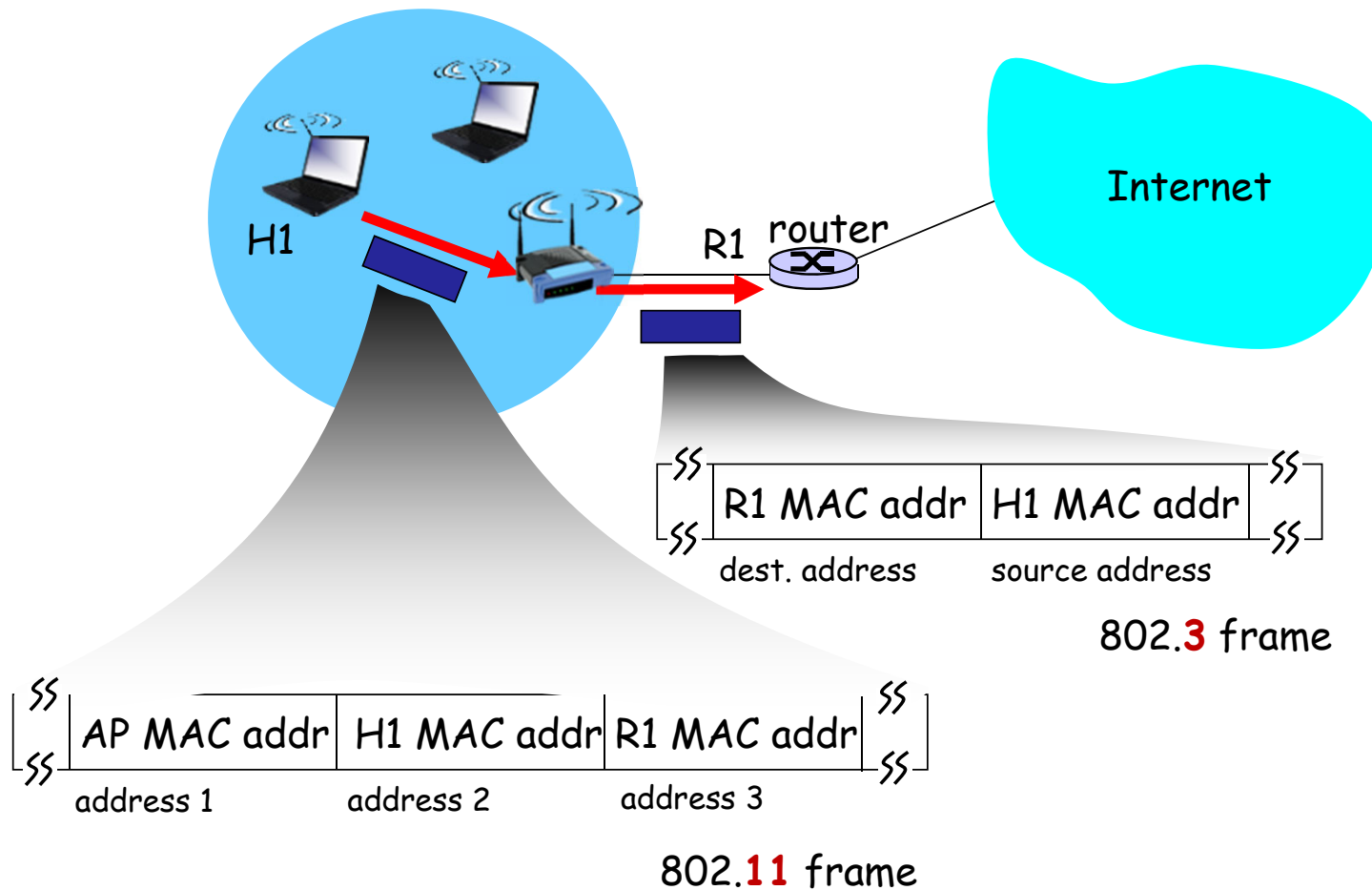
**Address 2:** MAC address of wireless host or AP transmitting this frame

**Address 3:** MAC address of router interface to which AP is attached

**Address 4:** used only in ad hoc mode



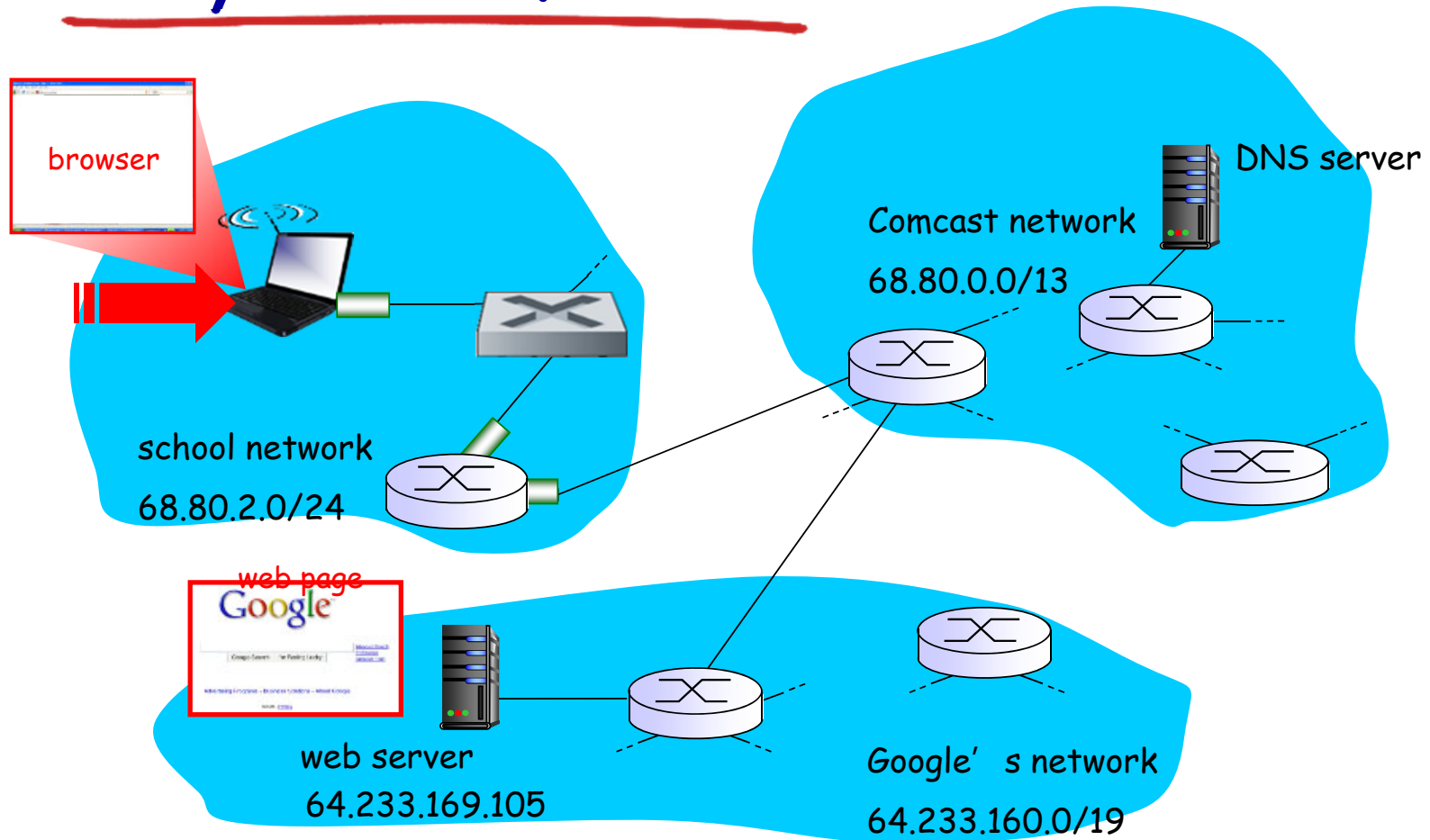
# 802.11 frame: addressing



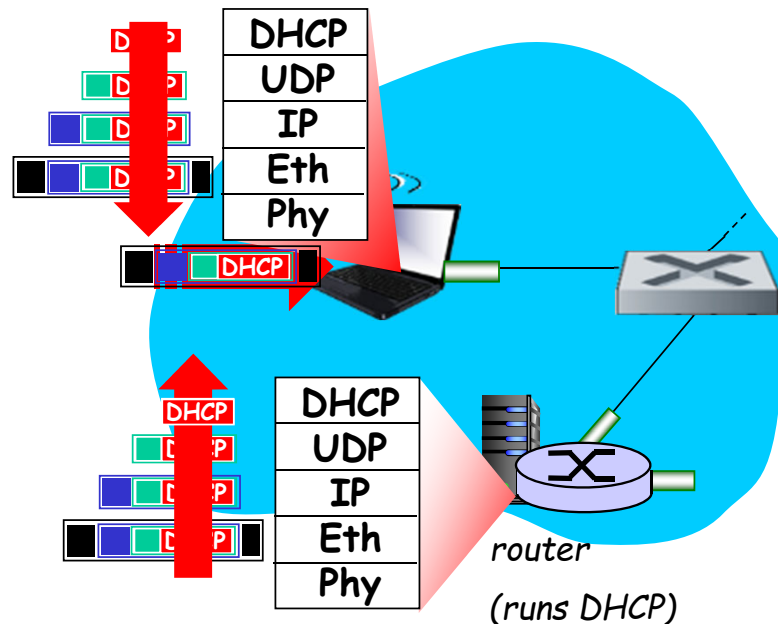
## Synthesis: a day in the life of a web request

- 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

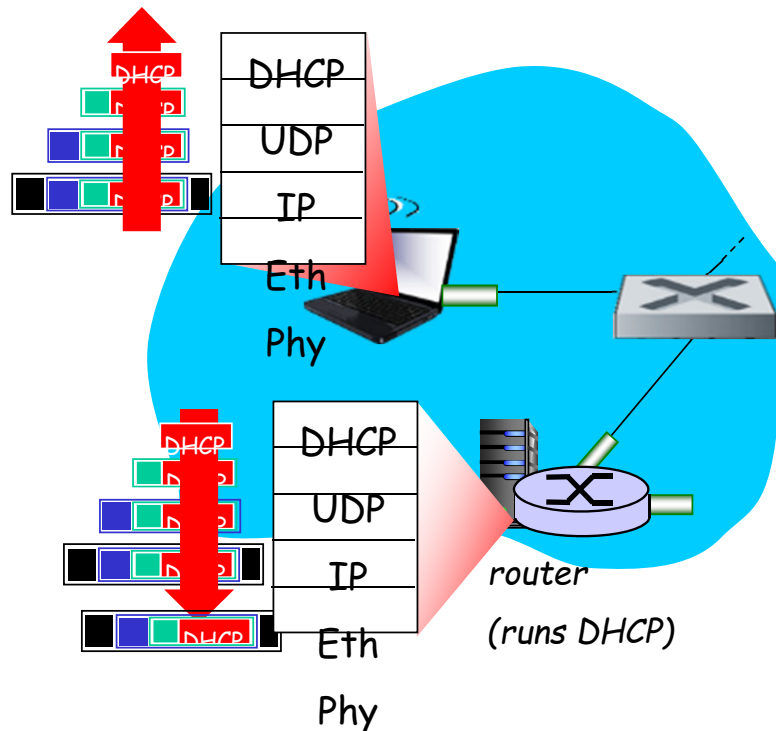


# A day in the life... connecting to the Internet



- 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

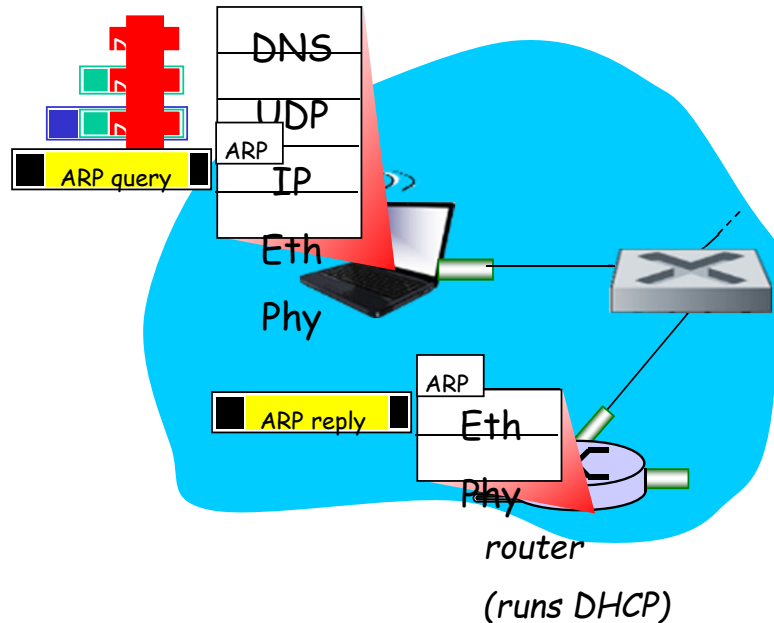
# A day in the life... connecting to the Internet



- 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

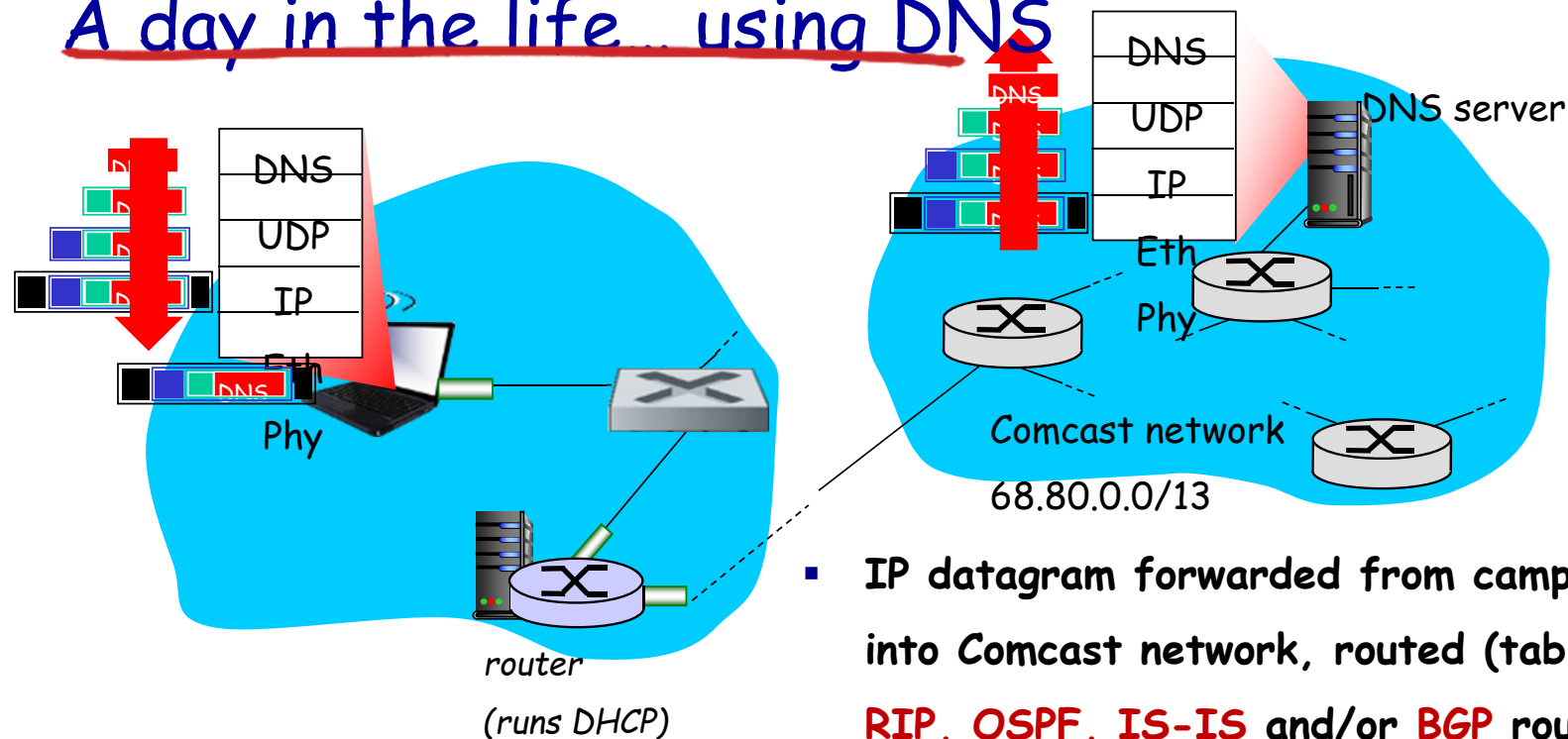
*Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router*

## 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

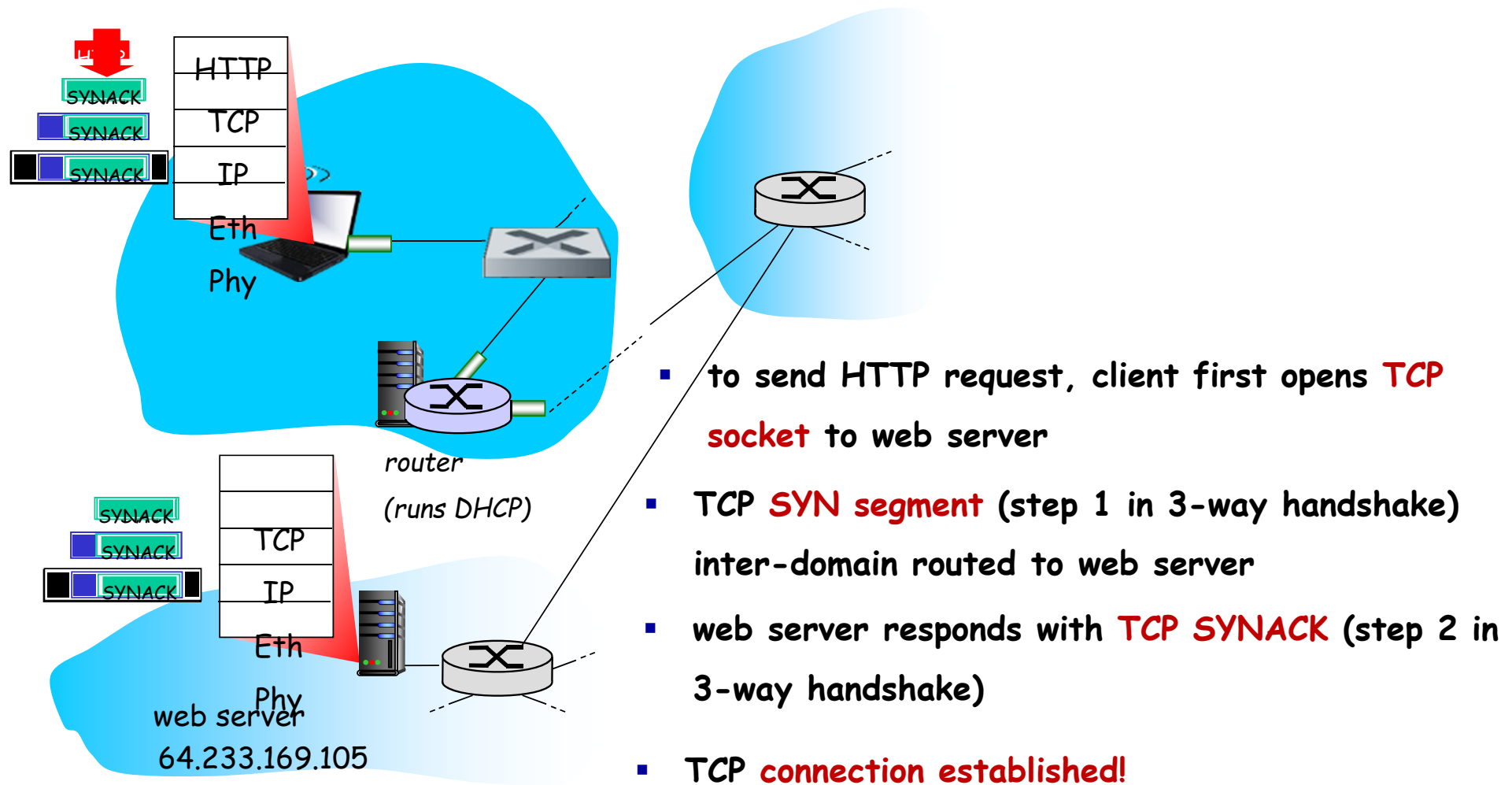
## A day in the life... using DNS



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

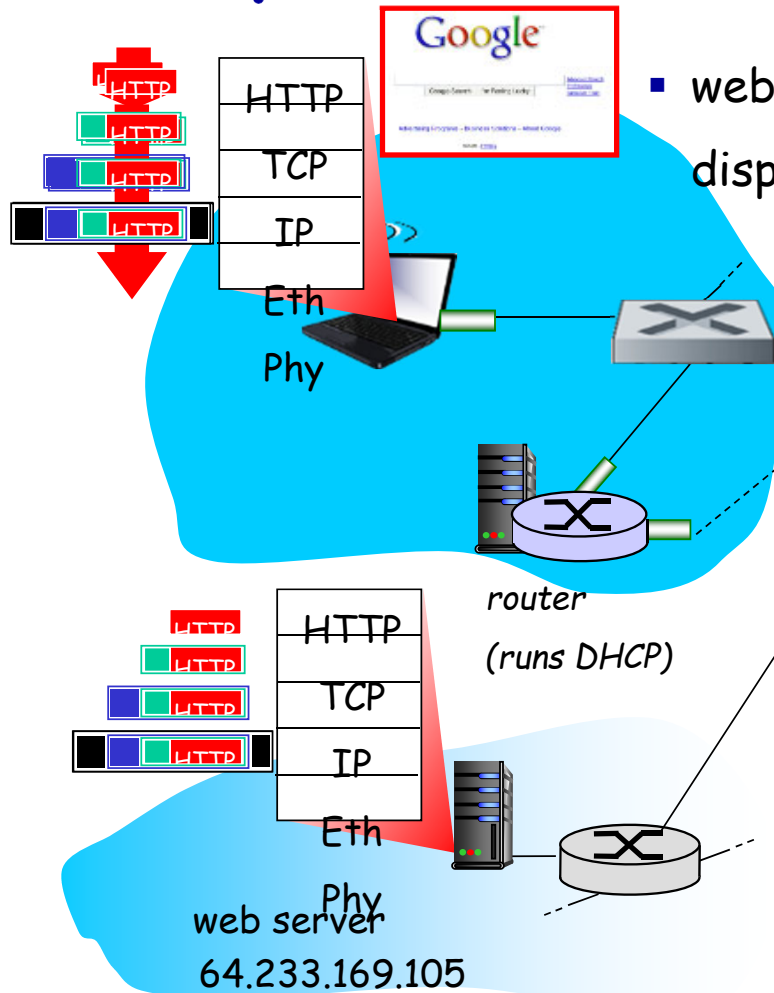
- IP datagram forwarded from campus network into Comcast network, routed (tables created by **RIP**, **OSPF**, **IS-IS** and/or **BGP** routing protocols) to DNS server
- demuxed to DNS server
- DNS server replies to client with IP address of **www.google.com**

# 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