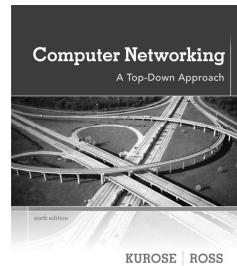


Chapter 5 Link Layer



*Computer
Networking: A Top
Down Approach*
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

All material copyright 1996-2012
J.F. Kurose and K.W. Ross, All Rights Reserved

Link Layer 5-1

Link layer, LANs: outline

- 5.1 introduction, services
- 5.2 error detection, correction
- 5.3 multiple access protocols
- 5.4 LANs
 - addressing, ARP
 - Ethernet
 - switches
 - VLANs
- 5.5 link virtualization: MPLS
- 5.6 data center networking
- 5.7 a day in the life of a web request

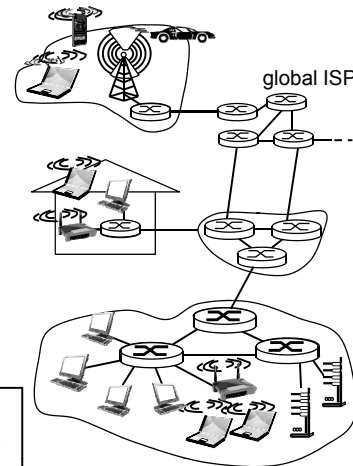
Link Layer 5-2

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 5-3

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 5-4

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, dest
 - 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 5-5

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

Link Layer 5-6

Link layer, LANs: outline

- | | |
|---|--|
| 5.1 introduction, services | 5.5 link virtualization: MPLS |
| 5.2 error detection, correction | 5.6 data center networking |
| 5.3 multiple access protocols | 5.7 a day in the life of a web request |
| 5.4 LANs <ul style="list-style-type: none"> ▪ addressing, ARP ▪ Ethernet ▪ switches ▪ VLANs | |

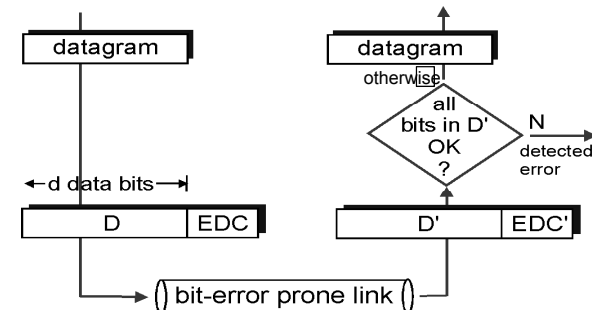
Link Layer 5-7

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

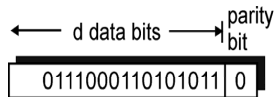


Link Layer 5-8

Parity checking

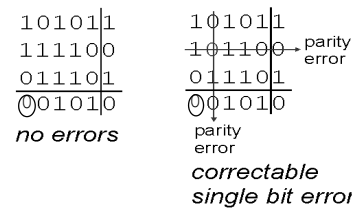
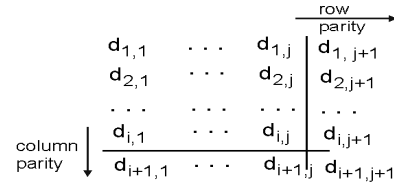
single bit parity:

- ❖ detect single bit errors



two-dimensional bit parity:

- ❖ detect and correct single bit errors



Link Layer 5-9

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?*

Link Layer 5-10

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)



$$D \cdot 2^r \text{ XOR } R \quad \text{mathematical formula}$$

Link Layer 5-11

CRC example

want:

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

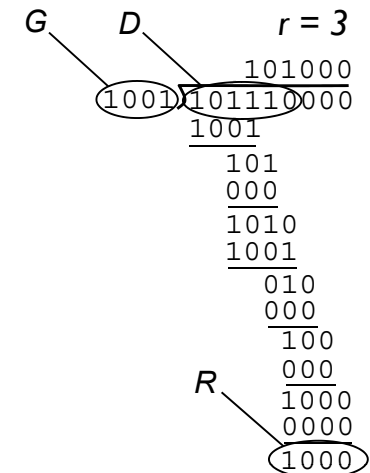
equivalently:

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

equivalently:

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

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



Link Layer 5-12

Link layer, LANs: outline

- 5.1 introduction, services
- 5.2 error detection, correction
- 5.3 multiple access protocols
- 5.4 LANs
 - addressing, ARP
 - Ethernet
 - switches
 - VLANs
- 5.5 link virtualization: MPLS
- 5.6 data center networking
- 5.7 a day in the life of a web request

Link Layer 5-13

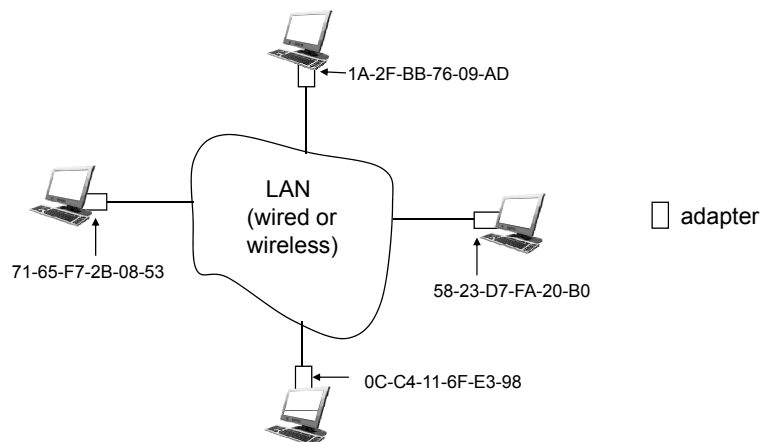
MAC addresses and ARP

- ❖ 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 "number" represents 4 bits)

Link Layer 5-14

LAN addresses and ARP

each adapter on LAN has unique *LAN* address



Link Layer 5-15

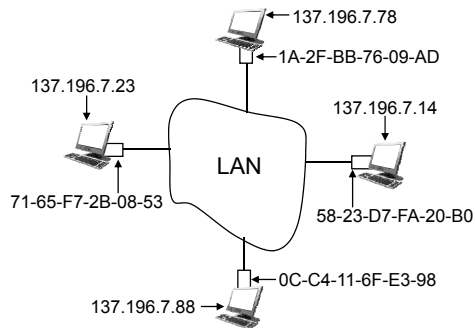
LAN addresses (more)

- ❖ 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

Link Layer 5-16

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)

Link Layer 5-17

ARP protocol: same LAN

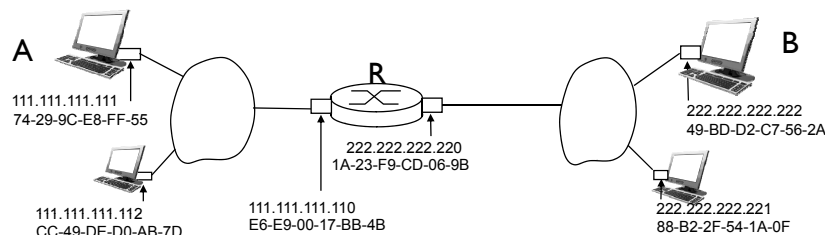
- ❖ A wants to send datagram to B
 - B's MAC address not in A's ARP table.
- ❖ A broadcasts ARP query packet, containing B's IP address
 - dest MAC address = FF-FF-FF-FF-FF-FF
 - all nodes on LAN receive ARP query
- ❖ B receives ARP packet, replies to A with its (B's) MAC address
 - frame sent to A's 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*

Link Layer 5-18

Addressing: routing to another LAN

walkthrough: send datagram from A to B via R

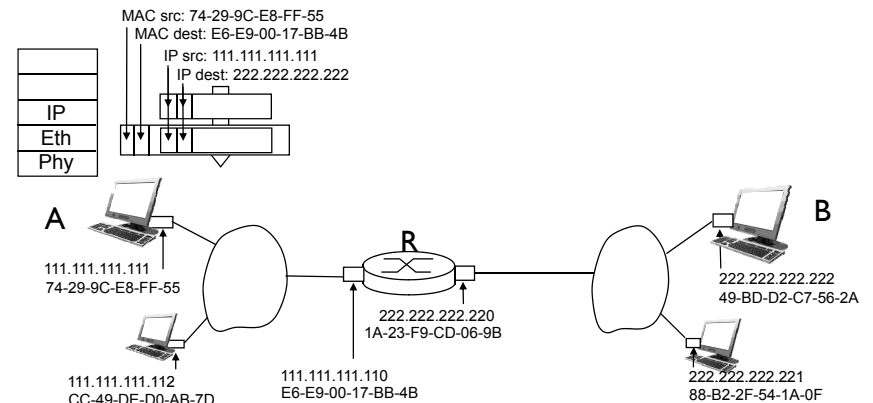
- focus on addressing – at IP (datagram) and MAC layer (frame)
- assume A knows B's IP address
- assume A knows IP address of first hop router, R (how?)
- assume A knows R's MAC address (how?)



Link Layer 5-19

Addressing: routing to another LAN

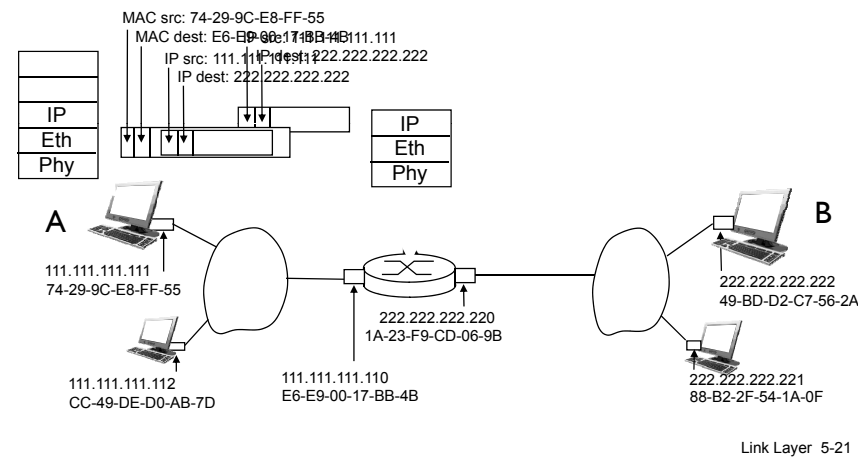
- ❖ A creates IP datagram with IP source A, destination B
- ❖ A creates link-layer frame with R's MAC address as dest, frame contains A-to-B IP datagram



Link Layer 5-20

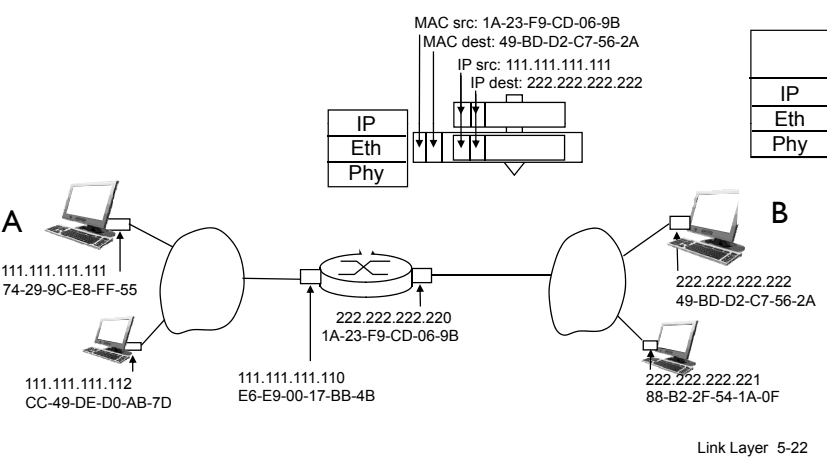
Addressing: routing to another LAN

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



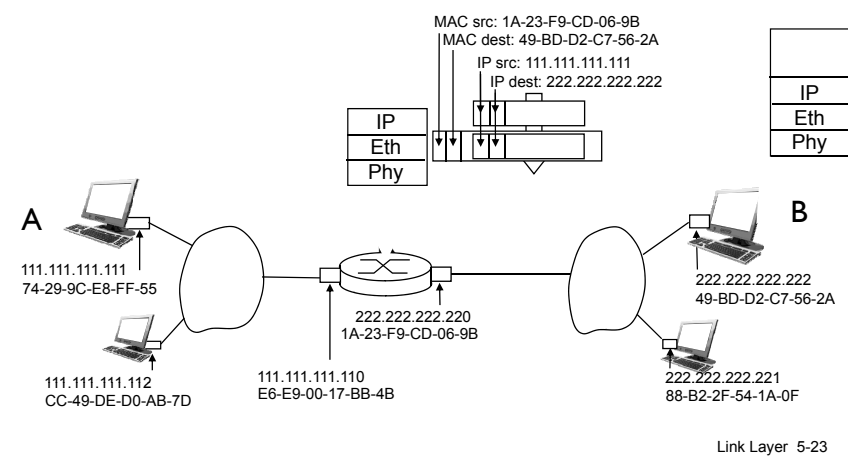
Addressing: routing to another LAN

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



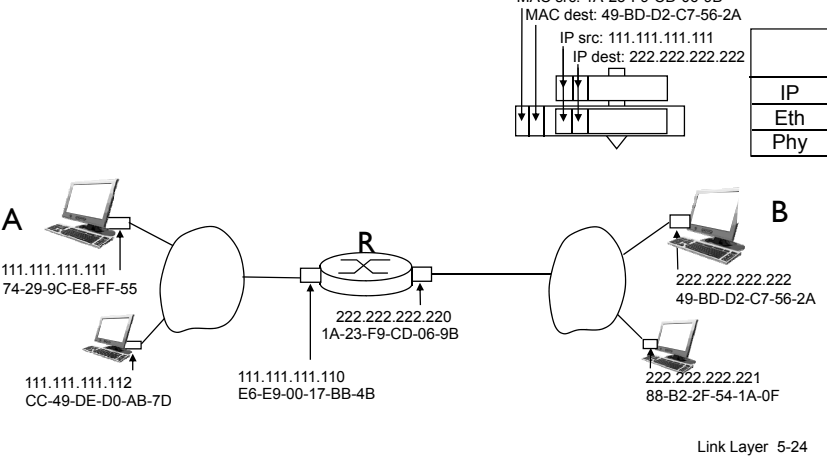
Addressing: routing to another LAN

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



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



Link layer, LANs: outline

- 5.1 introduction, services
- 5.2 error detection, correction
- 5.3 multiple access protocols
- 5.4 LANs
 - addressing, ARP
 - Ethernet
 - switches
 - VLANs
- 5.5 link virtualization: MPLS
- 5.6 data center networking
- 5.7 a day in the life of a web request

Link Layer 5-25

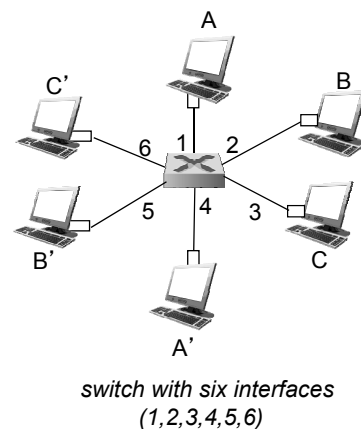
Ethernet switch

- ❖ 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

Link Layer 5-26

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



Link Layer 5-27

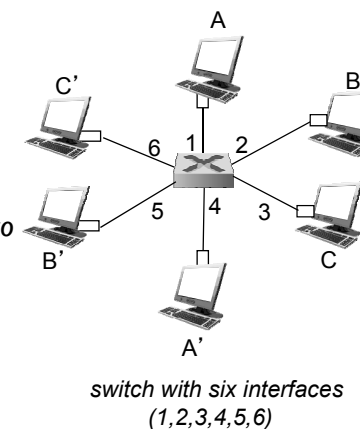
Switch forwarding table

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?

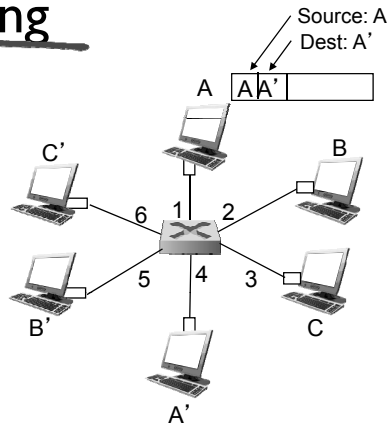


Link Layer 5-28

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)

Link Layer 5-29

Switch: frame filtering/forwarding

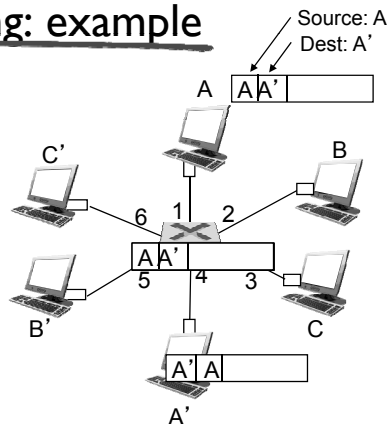
when frame received at switch:

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 */

Link Layer 5-30

Self-learning, forwarding: example

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



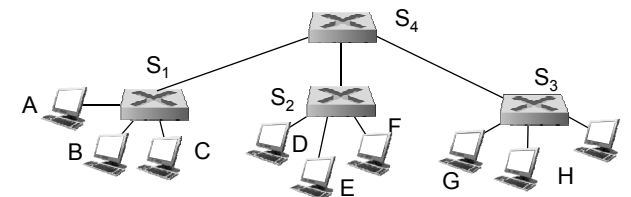
MAC addr	interface	TTL
A	1	60
A'	4	60

switch table
(initially empty)

Link Layer 5-31

Interconnecting switches

- ❖ switches can be connected together



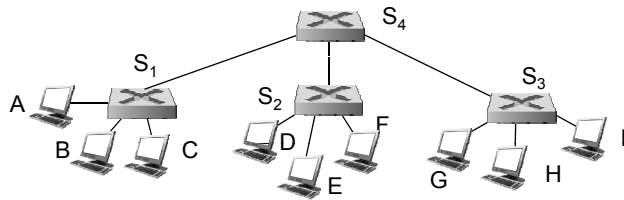
Q: sending from A to G - how does S₁ know to forward frame destined to F via S₄ and S₃?

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

Link Layer 5-32

Self-learning multi-switch example

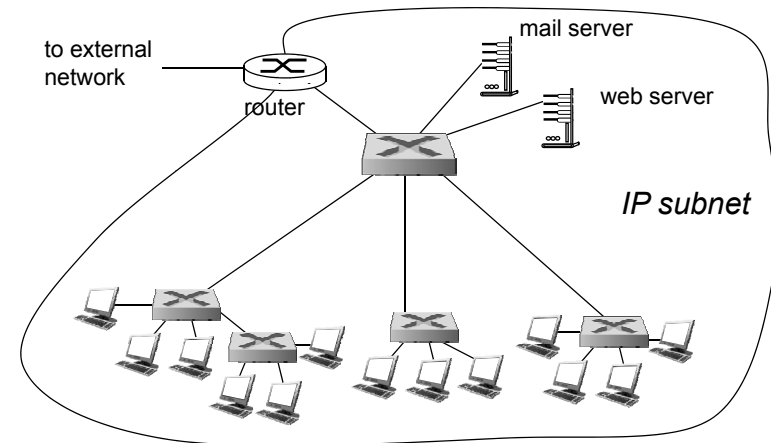
Suppose C sends frame to I, I responds to C



❖ Q: show switch tables and packet forwarding in S₁, S₂, S₃, S₄

Link Layer 5-33

Institutional network



Link Layer 5-34

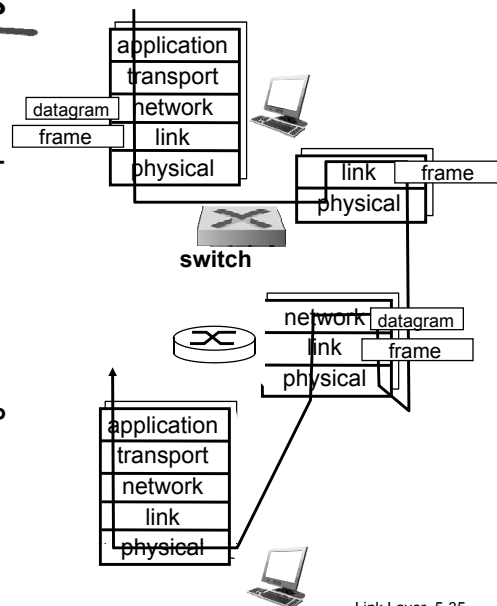
Switches vs. routers

both are store-and-forward:

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

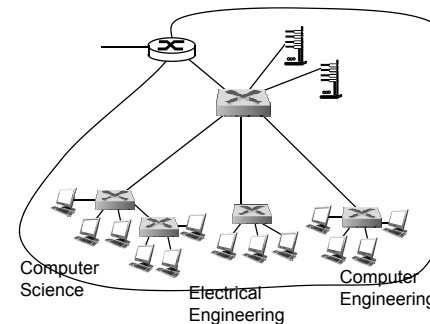
both have forwarding tables:

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



Link Layer 5-35

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

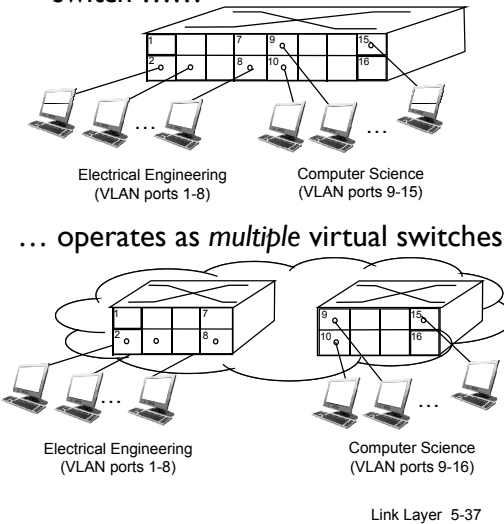
Link Layer 5-36

VLANs

Virtual Local Area Network

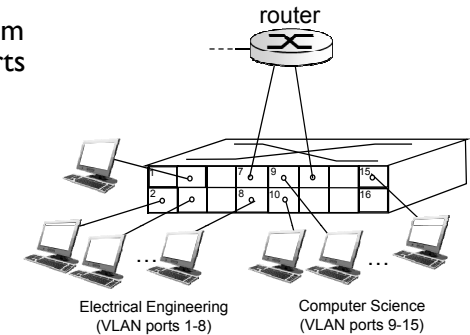
switch(es) supporting VLAN capabilities can be configured to define multiple **virtual** LANS over single physical LAN infrastructure.

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

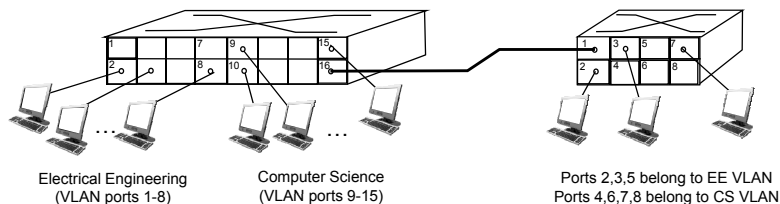


Port-based VLAN

- ❖ *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

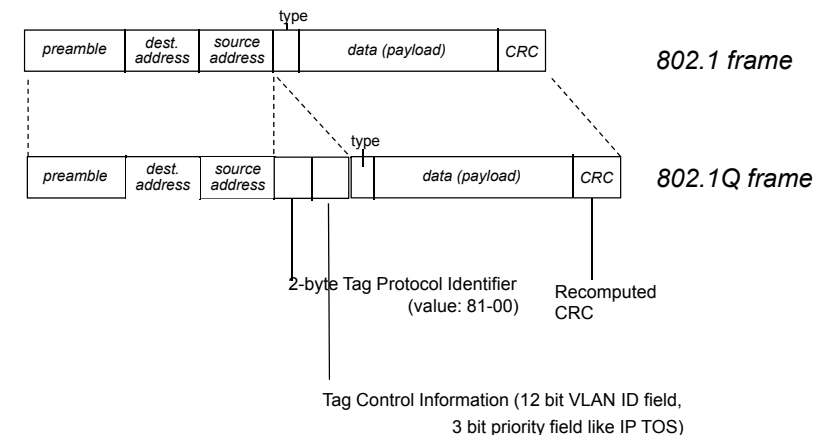


VLANs spanning multiple switches



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

802.1Q VLAN frame format



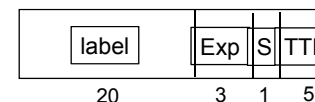
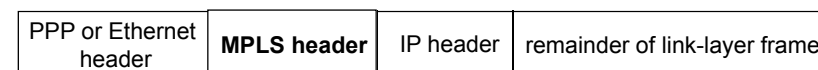
Link layer, LANs: outline

- 5.1 introduction, services
- 5.2 error detection, correction
- 5.3 multiple access protocols
- 5.4 LANs
 - addressing, ARP
 - Ethernet
 - switches
 - VLANs
- 5.5 link virtualization: MPLS
- 5.6 data center networking
- 5.7 a day in the life of a web request

Link Layer 5-41

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!



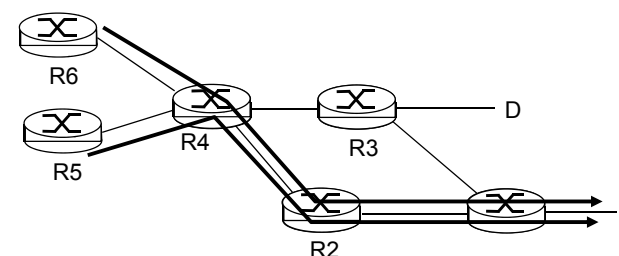
Link Layer 5-42

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)

Link Layer 5-43

MPLS versus IP paths

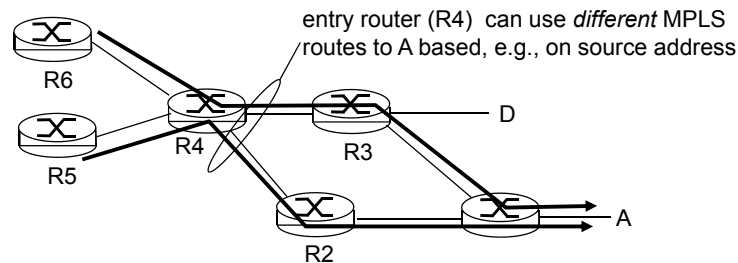


- ❖ *IP routing: path to destination determined by destination address alone*



Link Layer 5-44

MPLS versus IP paths



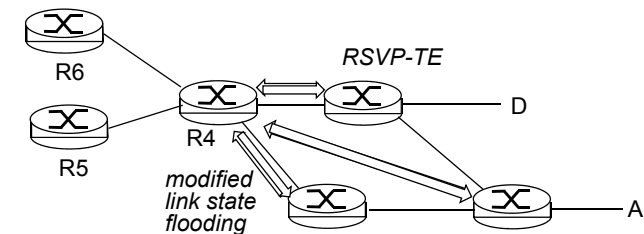
- ❖ **IP routing:** path to destination determined by destination address alone
- ❖ **MPLS routing:** path to destination can be based on source *and* dest. address
 - fast reroute: precompute backup routes in case of link failure



Link Layer 5-45

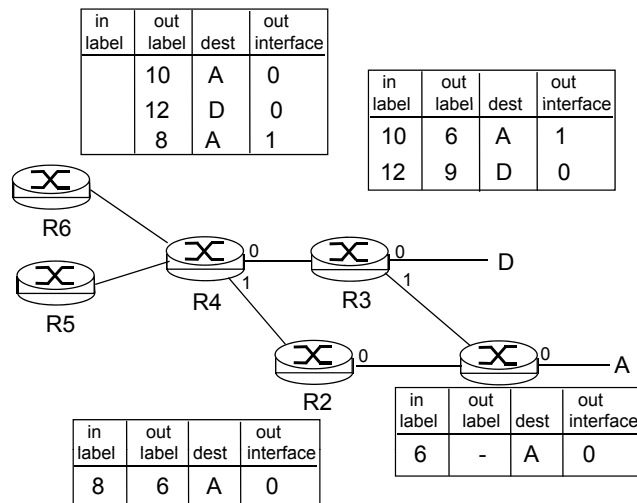
MPLS signaling

- ❖ 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



Link Layer 5-46

MPLS forwarding tables



Link Layer 5-47

Link layer, LANs: outline

- 5.1 introduction, services
- 5.2 error detection, correction
- 5.3 multiple access protocols
- 5.4 LANs
 - addressing, ARP
 - Ethernet
 - switches
 - VLANs
- 5.5 link virtualization: MPLS
- 5.6 data center networking
- 5.7 a day in the life of a web request

Link Layer 5-48

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



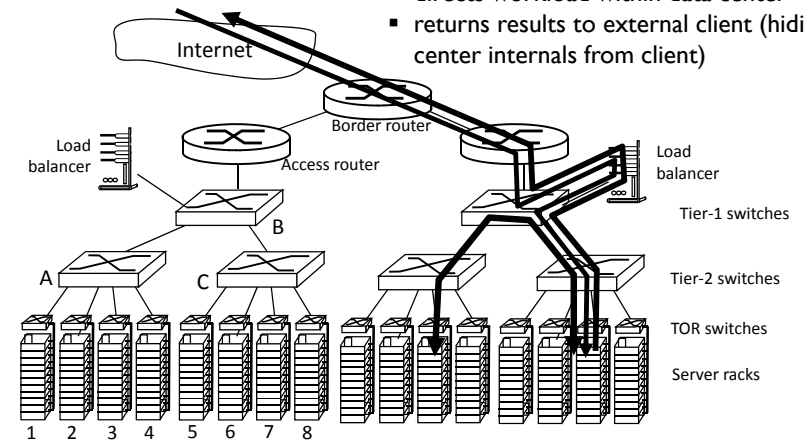
Inside a 40-ft Microsoft container, Chicago data center

Link Layer 5-49

Data center networks

load balancer: application-layer routing

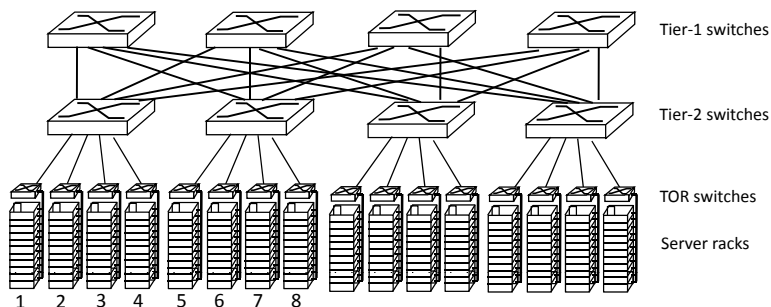
- receives external client requests
- directs workload within data center
- returns results to external client (hiding data center internals from client)



Link Layer 5-50

Data center networks

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



Link layer, LANs: outline

- | | |
|---|--|
| 5.1 introduction, services | 5.5 link virtualization: MPLS |
| 5.2 error detection, correction | 5.6 data center networking |
| 5.3 multiple access protocols | 5.7 a day in the life of a web request |
| 5.4 LANs <ul style="list-style-type: none"> ▪ addressing, ARP ▪ Ethernet ▪ switches ▪ VLANs | |

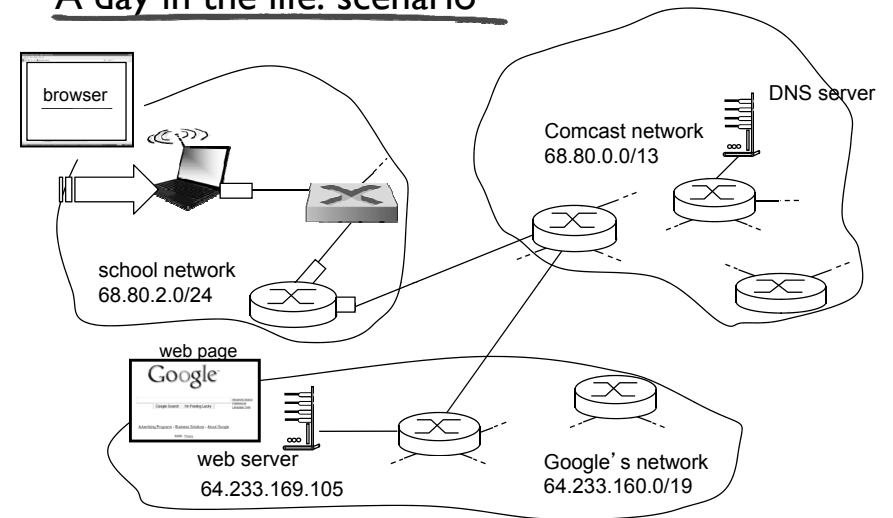
Link Layer 5-52

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`

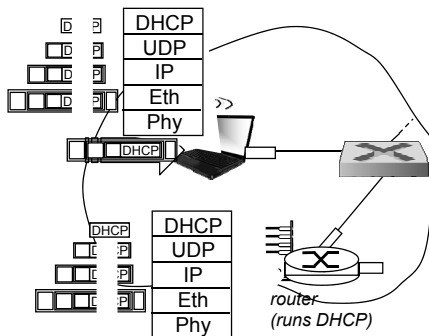
Link Layer 5-53

A day in the life: scenario



Link Layer 5-54

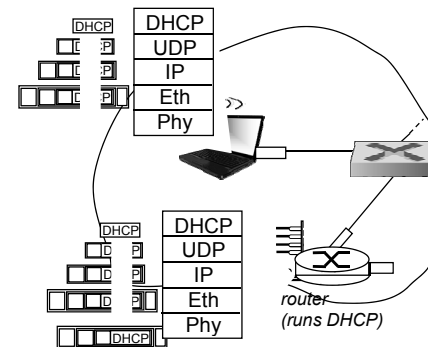
A day in the life... connecting to the Internet



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

Link Layer 5-55

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

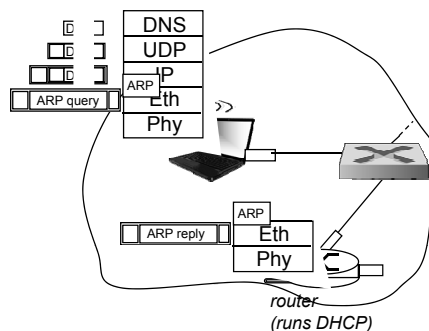


- ❖ DHCP server formulates *DHCP ACK* containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- ❖ encapsulation at DHCP server; frame forwarded (*switch learning*) through LAN, demultiplexing at client
- ❖ DHCP client receives *DHCPACK* reply

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

Link Layer 5-56

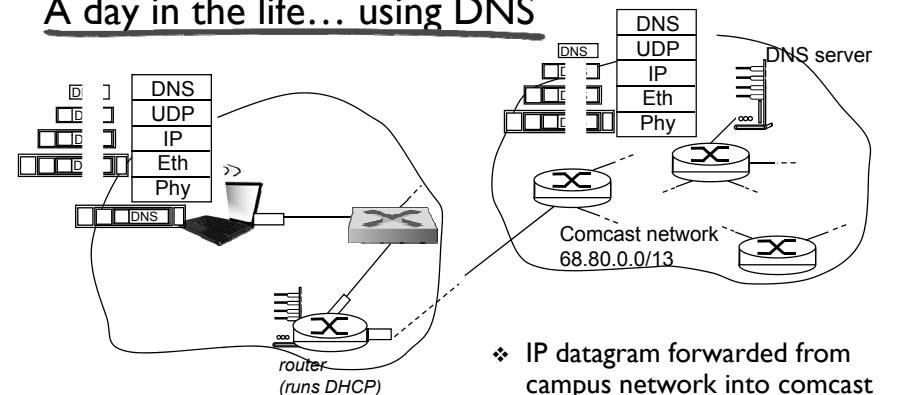
A day in the life... ARP (before DNS, before HTTP)



- ❖ 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

Link Layer 5-57

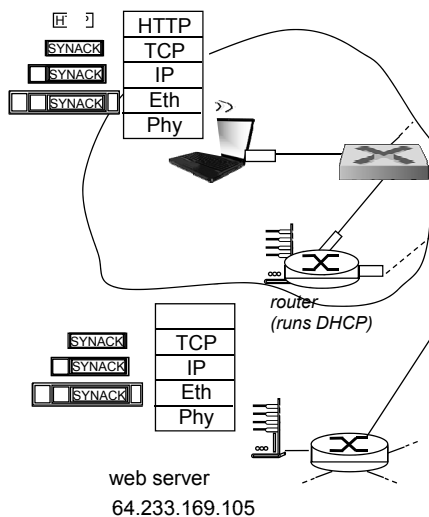
A day in the life... using DNS



- ❖ 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
- ❖ demux'ed to *DNS* server
- ❖ *DNS* server replies to client with IP address of *www.google.com*

Link Layer 5-58

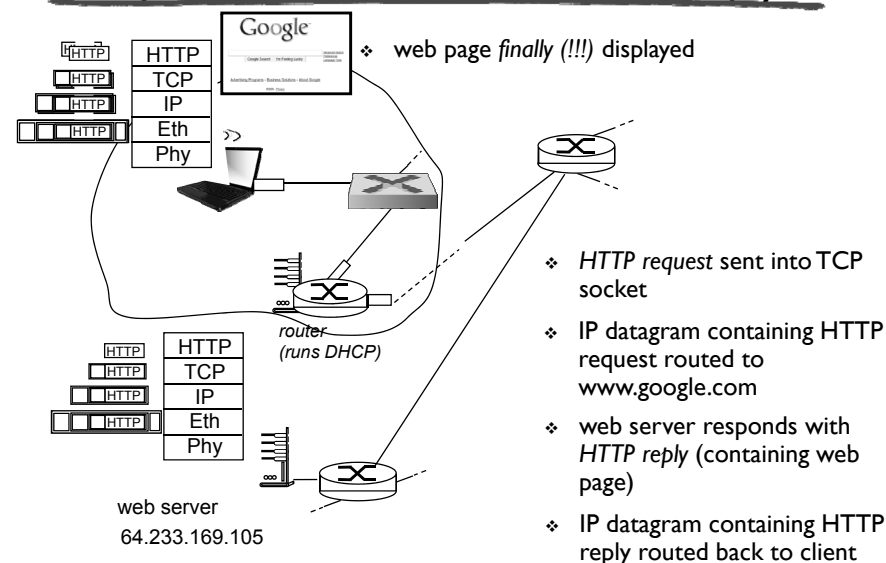
A day in the life...TCP connection carrying HTTP



- ❖ 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!

Link Layer 5-59

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

Link Layer 5-60

Chapter 5: 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
 - network management