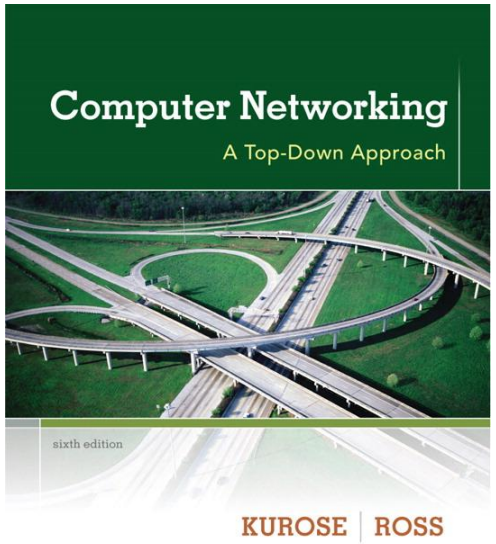
Κεφάλαιο 5
Επίπεδο συνδέσμου δεδομένων



A note on the use of these ppt 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 1996-2012
J.F Kurose and K.W. Ross, All Rights Reserved

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

Επίπεδο συνδέσμου δεδομένων

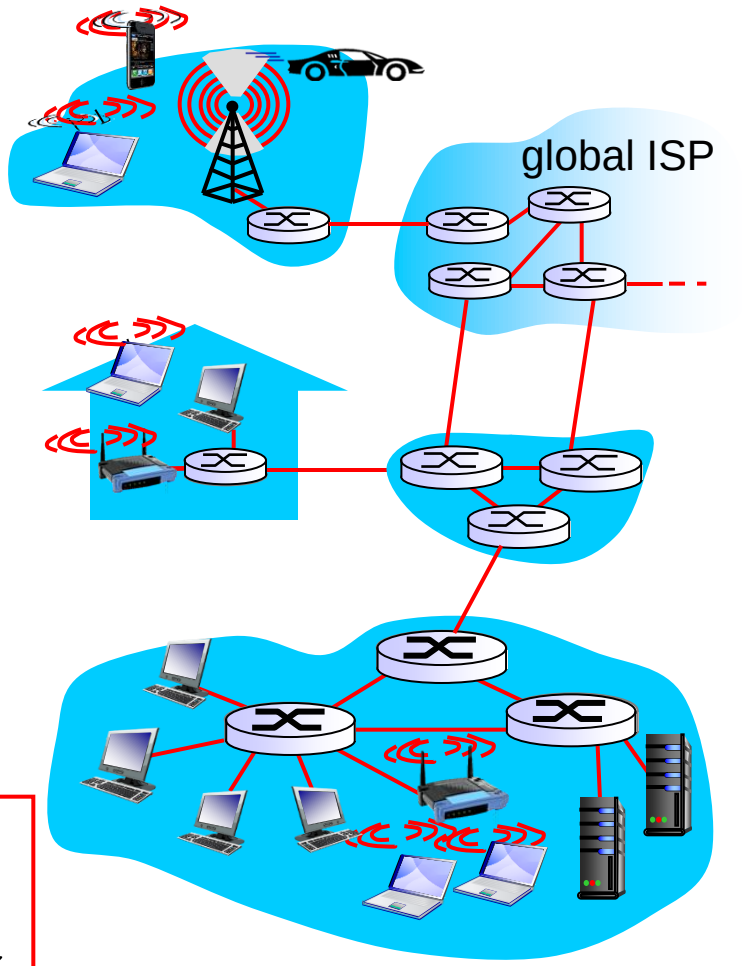
Στόχοι:

- Κατανόηση των υπηρεσιών που προσφέρει το επίπεδο συνδέσμου δεδομένων:
  - Ανίχνευση και διόρθωση σφαλμάτων
  - Κοινή χρήση διαύλων επικοινωνίας, πολλαπλή πρόσβαση.
  - Διευθυνσιοδότηση στο επίπεδο συνδέσμου.
  - Τοπικά δίκτυα: Ethernet, VLANs

Link layer: Εισαγωγή

ορολογία:

- hosts and routers: nodes (κόμβοι)
- Δίαυλοι επικοινωνίας που συνδέουν γειτονικούς κόμβους: links (σύνδεσμοι)
  - wired links (ενσύρματοι)
  - wireless links (ασύρματοι)
  - LANs (τοπικά δίκτυα
- Πακέτο επιπέδου 2: frame (πλαίσιο), ενθυλακώνει ένα datagram



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

Link layer υπηρεσίες

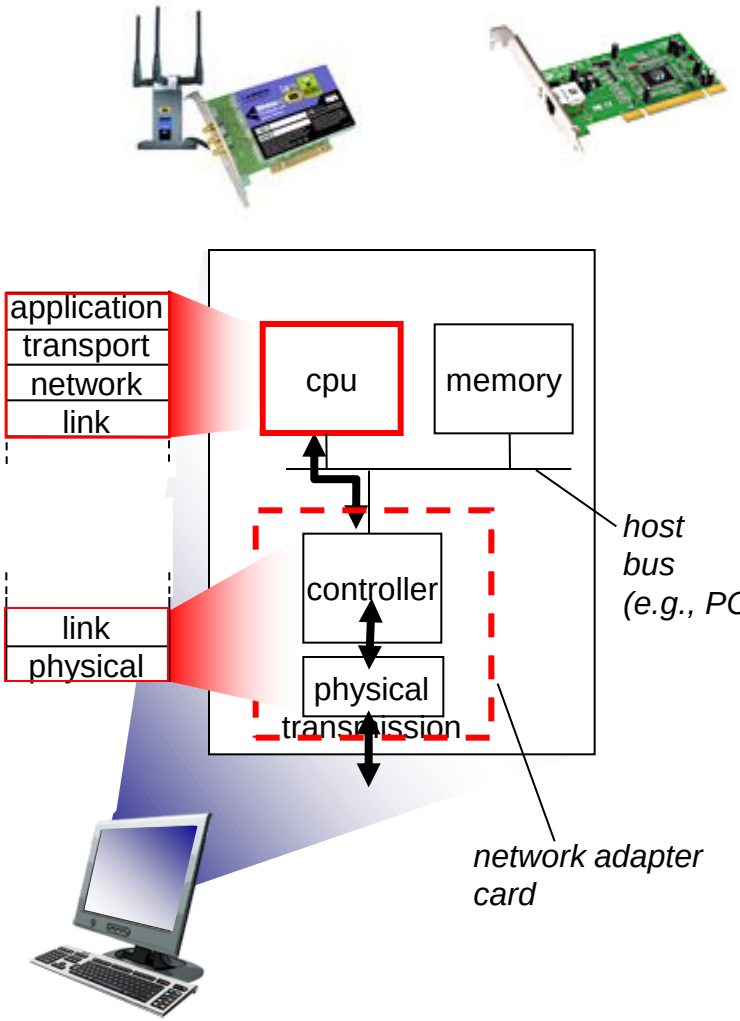
- *Δημιουργία πλαισίου, πρόσβαση στον σύνδεσμο:*
  - Ενθυλάκωση του datagram σε ένα frame,προσθήκη επικεφαλίδας
  - Πρόσβαση στον σύνδεσμο κοινής χρήσης.
  - Η διεύθυνση "MAC" χρησιμοποιείται στις κεφαλίδες των πλαισίων για να προσδιορίσουν την προέλευση και τον προορισμό.
    - Η διεύθυνση MAC είναι διαφορετική από την διεύθυνση IP!
- *Αξιόπιστη μεταφορά μεταξύ γειτνιαζόντων κόμβων.*
  - Μελετήθηκε στο προηγούμενο κεφάλαιο.
  - Χρησιμοποιείται σπάνια σε συνδέσμους με μικρό ποσοστό σφαλμάτων.
  - Ασύρματες συνδέσεις έχουν υψηλό ρυθμό σφαλμάτων.
    - *Ε:Γιατί χρειάζεται αξιοπιστία και στο επίπεδο 2?*

Link layer υπηρεσίες (συνεχ.)

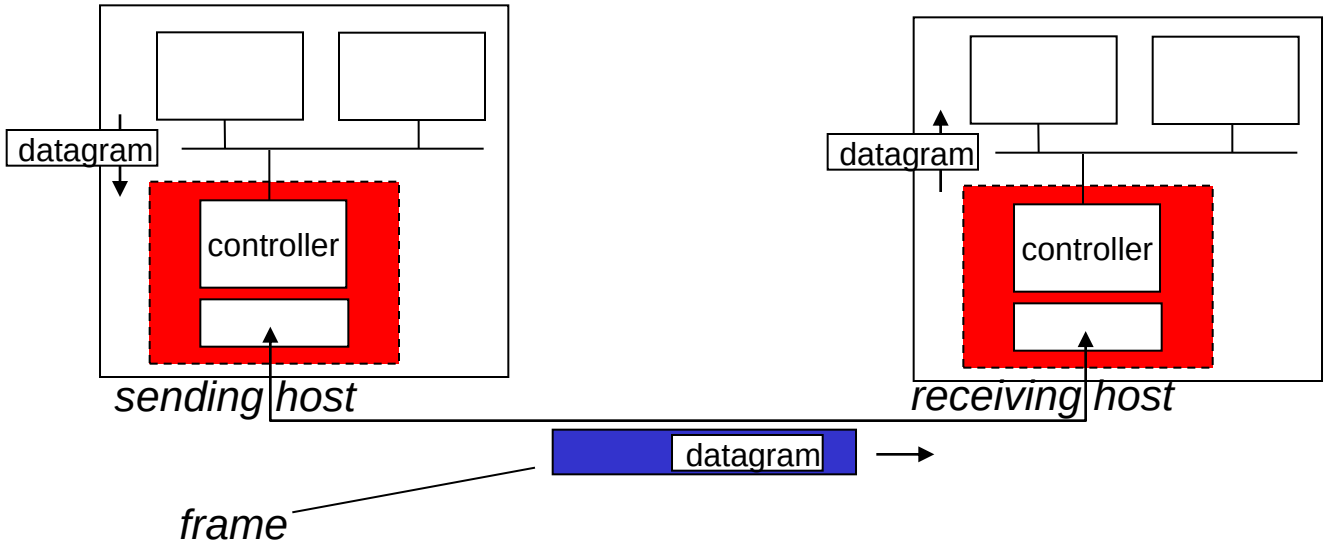
- *flow control (έλεγχος ροής):*
  - Προσαρμογή στις ταχύτητες μεταξύ αποστέλλοντα και λαμβάνοντα κόμβου.
- *error detection (ανίχνευση σφαλμάτων)*:
  - Σφάλματα προκαλούνται από την εξασθένηση του σήματος και θόρυβο.
  - Ο παραλήπτης ανιχνεύει την παρουσία σφαλμάτων:
    - Στέλνει σήμα στον αποστολέα για επανάληψη αποστολής και απορρίπτει το πλαίσιο.
- *error correction(διόρθωση σφαλμάτων):*
  - Ο παραλήπτης ανιχνεύει και διορθώνει λανθασμένα ψηφία χωρίς να απαιτείται επανάληψη της αποστολής.
- *half-duplex and full-duplex*
  - Με half duplex, και οι δύο κόμβοι μπορούν να στέλνουν δεδομένα αλλά όχι ταυτόχρονα.

Που υλοποιείται το link layer?

- Σε κάθε υπολογιστή ή δικτυακή συσκευή.
- Το link layer υλοποιείται στον «προσαρμογέα-adaptor» (*network interface card* NIC) ή σε ένα ολοκληρωμένο.
- Ethernet card, 802.11 card; Ethernet chipset
  - Υλοποιεί και το link layer και το φυσικό επίπεδο (physical layer)
  - Συνδέεται στους διαύλους του υπολογιστή.
  - Είναι συνδυασμός hardware, software, firmware



Επικοινωνία προσαρμογέων



- Αποστέλλουσα πλευρά:
  - Ενθυλακώνει datagram σε frame
  - προσθέτει ψηφία για έλεγχο σφαλμάτων, ρυθμίζει την ροή κλπ.
- Λαμβάνουσα πλευρά
  - Ελέγχει για σφάλματα, ενημερώνει για ροή, κλπ.
  - Εξάγει το datagram, και το προωθεί στο ανώτερο επίπεδο.

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

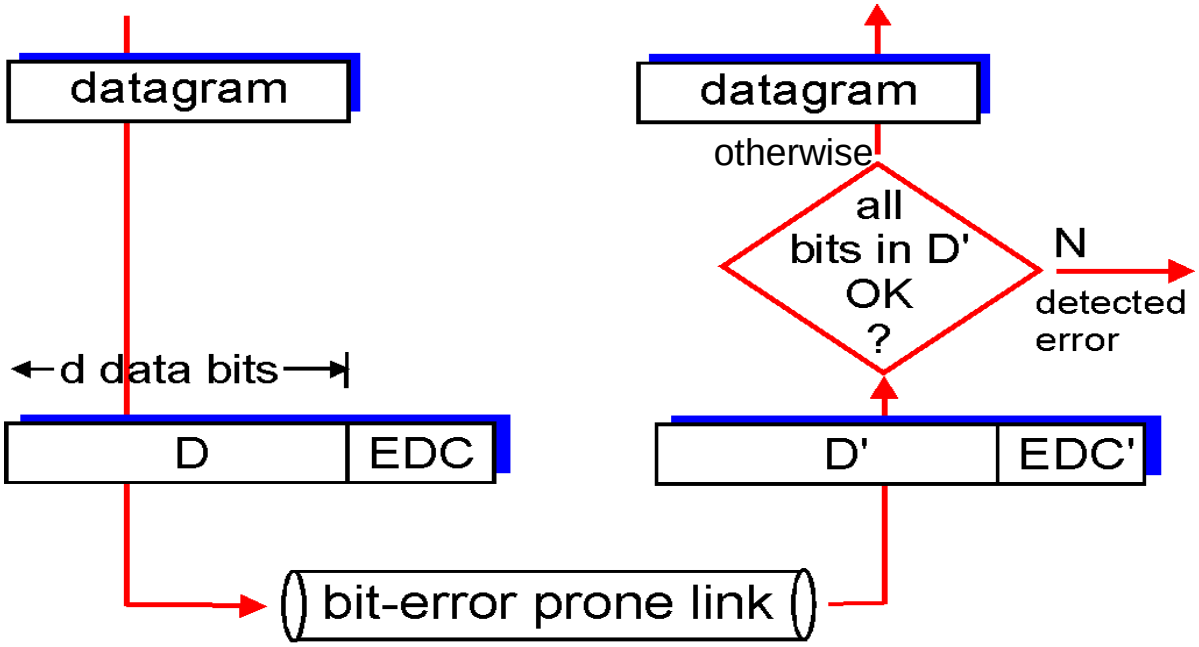
5.6 data center networking

5.7 a day in the life of a web request

Ανίχνευση Σφαλμάτων

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

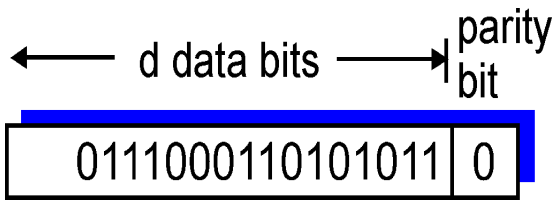
- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



Έλεγχος ισοτιμίας

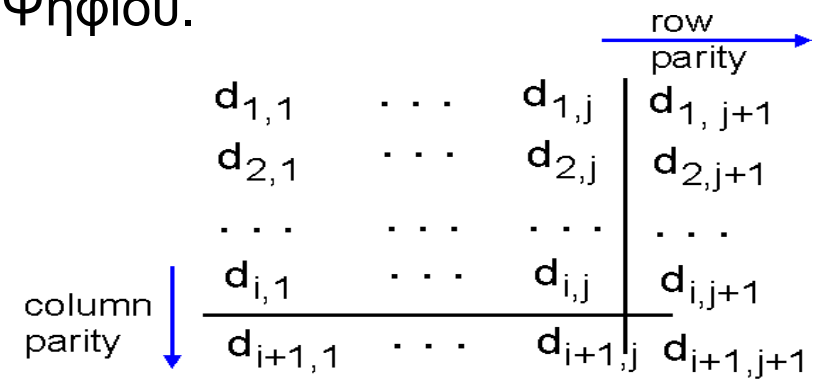
Ισοτιμία ενός bit:

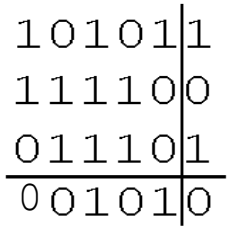
- *Ανιχνεύει σφάλματα ενός ψηφίου*



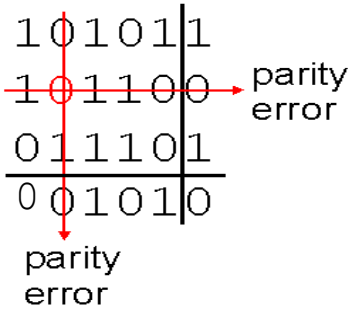
Διδιάστατη ισοτιμία:

- ανιχνεύει και διορθώνει σφάλματα ενός Ψηφίου.





no errors



correctable single bit error

Internet checksum (review)

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

sender:

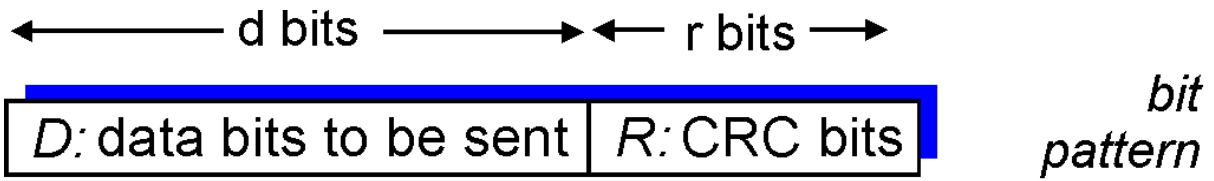
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

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



$D * 2^r \quad XOR \quad R$

mathematical formula

CRC example

want:

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

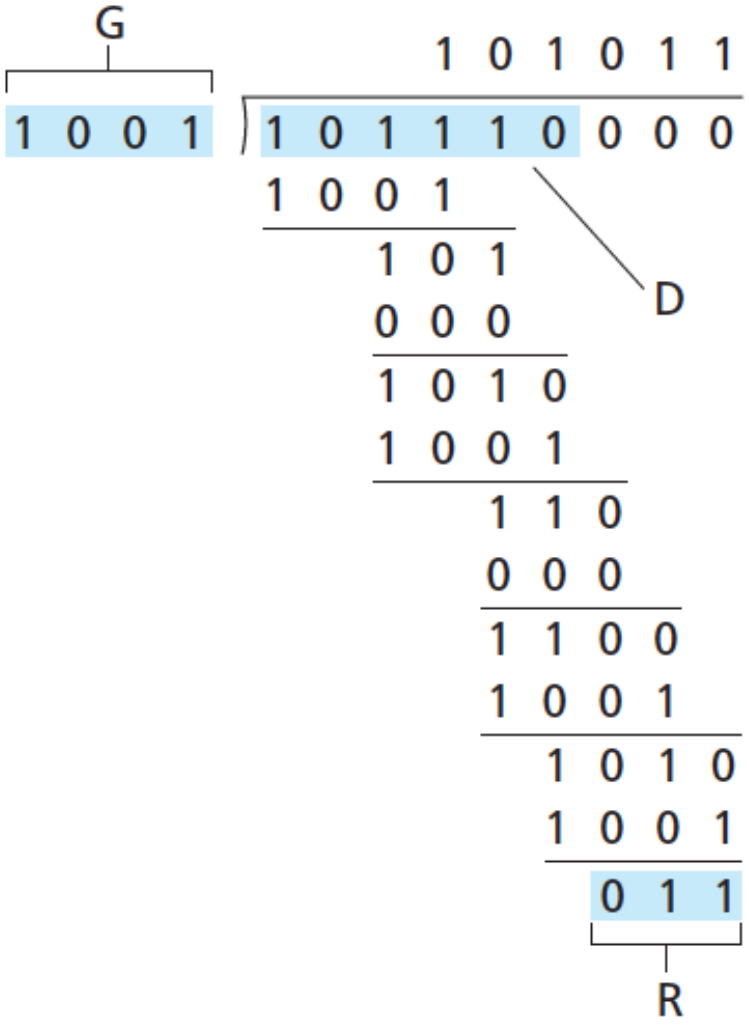
equivalently:

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

equivalently:

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

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



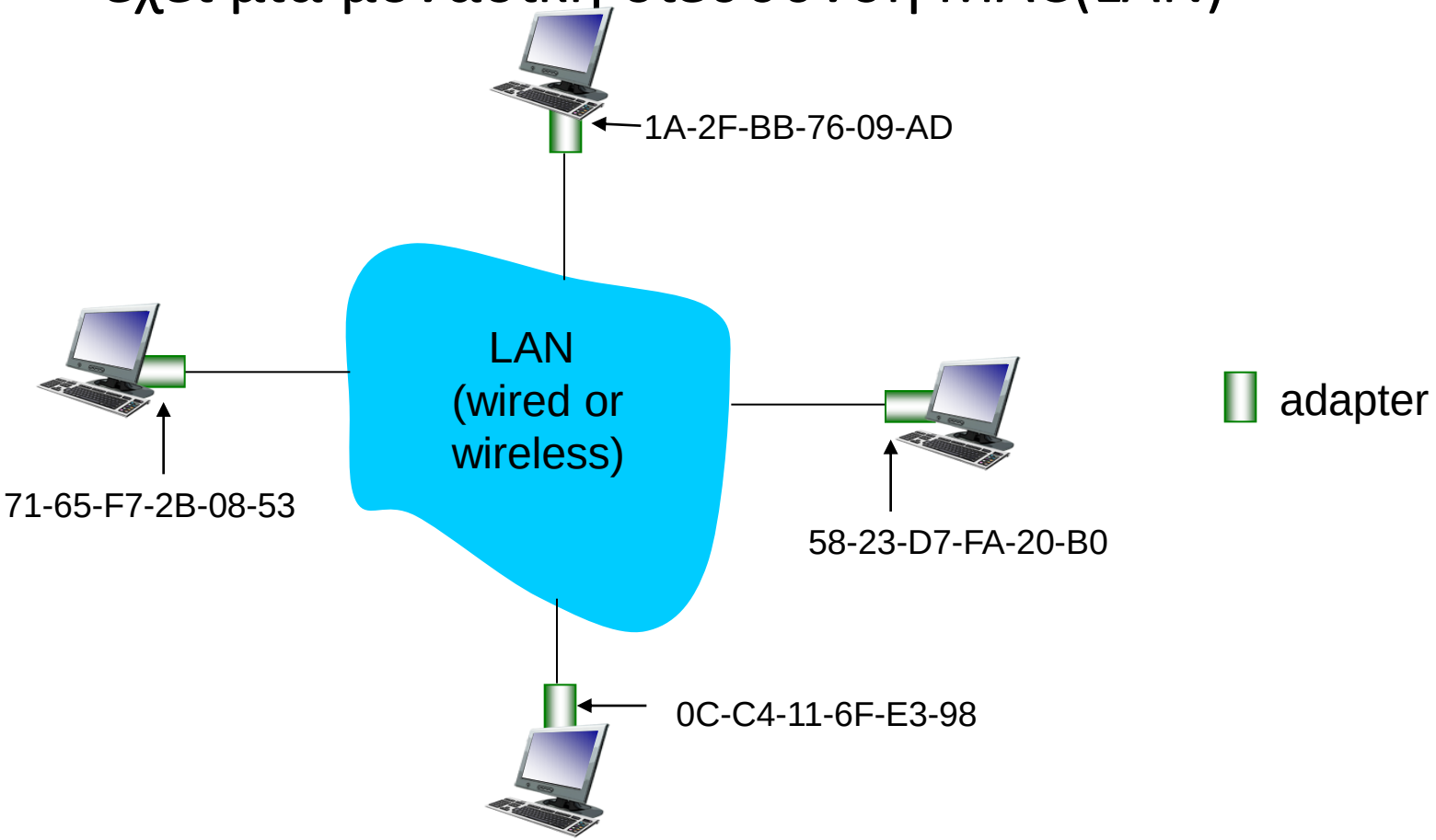
Διεύθυνση MAC and ARP

- 32-bit διεύθυνση IP :
  - *Διεύθυνση επιπέδου δικτύου.*
- Διεύθυνση MAC (ή LAN ή physical ή Ethernet):
  - *Χρησιμοποιείται τοπικά για να μεταφέρει ένα πλαίσιο από ένα interface σε ένα άλλο φυσικά συνδεδεμένο interface (στο ίδιο δίκτυο, όσον αφορά την διεύθυνση IP*
  - Διεύθυνση 48 bit (για την πλειοψηφία των LAN) εγγραμμένη στην ROM του προσαρμογέα δικτύου (σε μερικές περιπτώσεις μπορεί να οριστεί μέσω λογισμικού)
  - e.g.: 1A-2F-BB-76-09-AD

Δεκαεξαδικος (base 16) συμβολισμός
(κάθε "σύμβολο" αντιπροσωπεύει 4 represents 4 bits)

Διευθύνσεις LAN and ARP

Κάθε προσαρμογέας σε ένα τοπικό δικτυο έχει μια μοναδική διεύθυνση MAC(LAN)

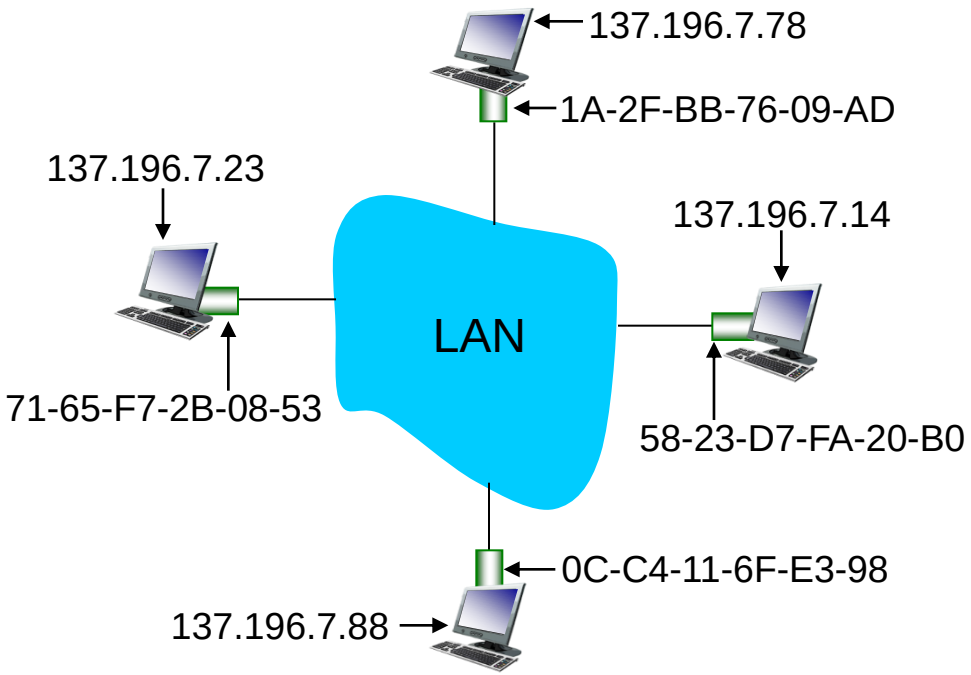


Διευθύνσεις LAN (περισσοτερα)

- Τις διευθύνσεις MAC τις διαχειρίζεται το IEEE
- Οι κατασκευαστές αγοράζουν ένα εύρος διευθύνσεων MAC (για να διασφαλιστεί η μοναδικότητα)
- αναλογία:
  - MAC : το ΑΜΚΑ
  - IP : η ταχυδρομική διεύθυνση
- MAC address ➜ φορητότητα
  - Μπορούμε να μεταφέρουμε μια κάρτα δικτύου από δίκτυο σε άλλο.
- IP address δεν είναι φορητή
  - Η διεύθυνση εξαρτάται από το υποδίκτυο στο οποίο είμαστε συνδεδεμένοι.

ARP: address resolution protocol
Πρωτόκολλο επίλυσης διεύθυνσης

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)

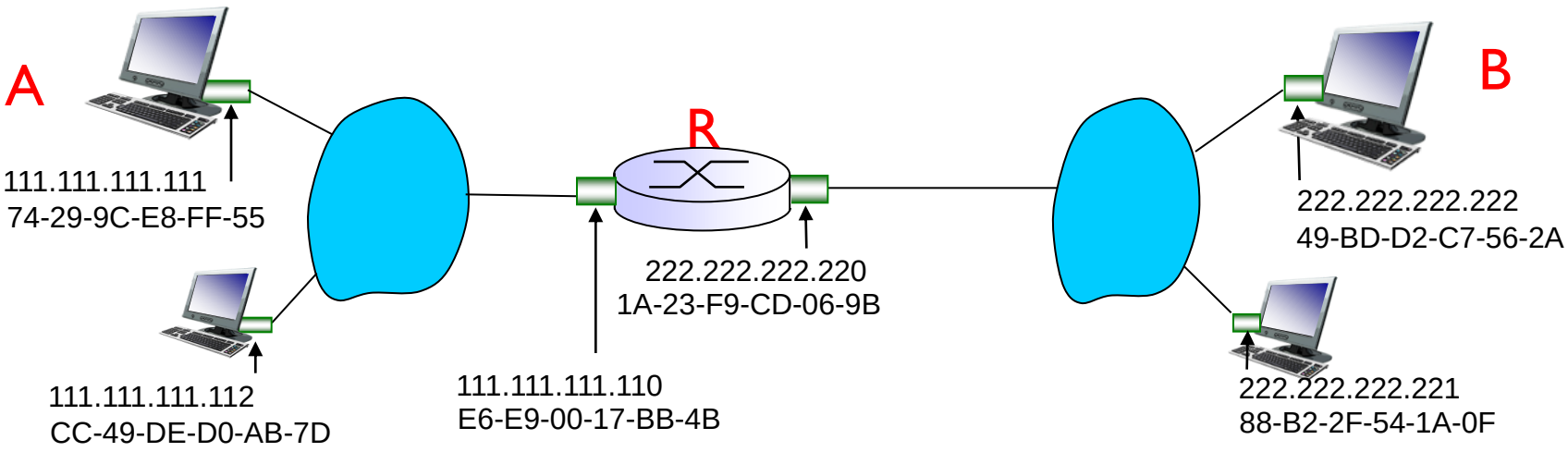
Πρωτόκολλο ARP : στο ίδιο LAN

- 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*

Addressing: δρομολόγηση σε άλλο LAN

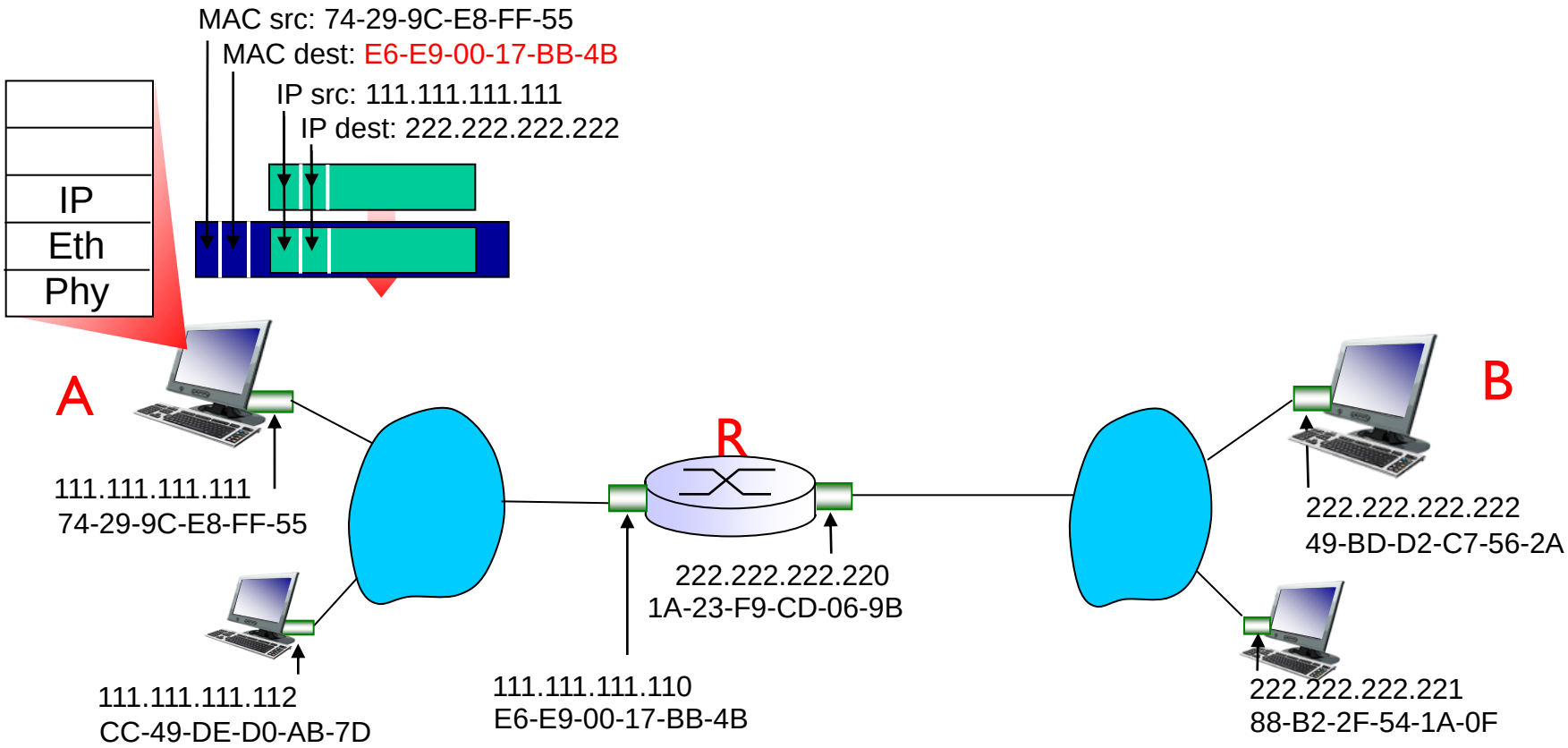
walkthrough: send datagram from A to B via R

- 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?)



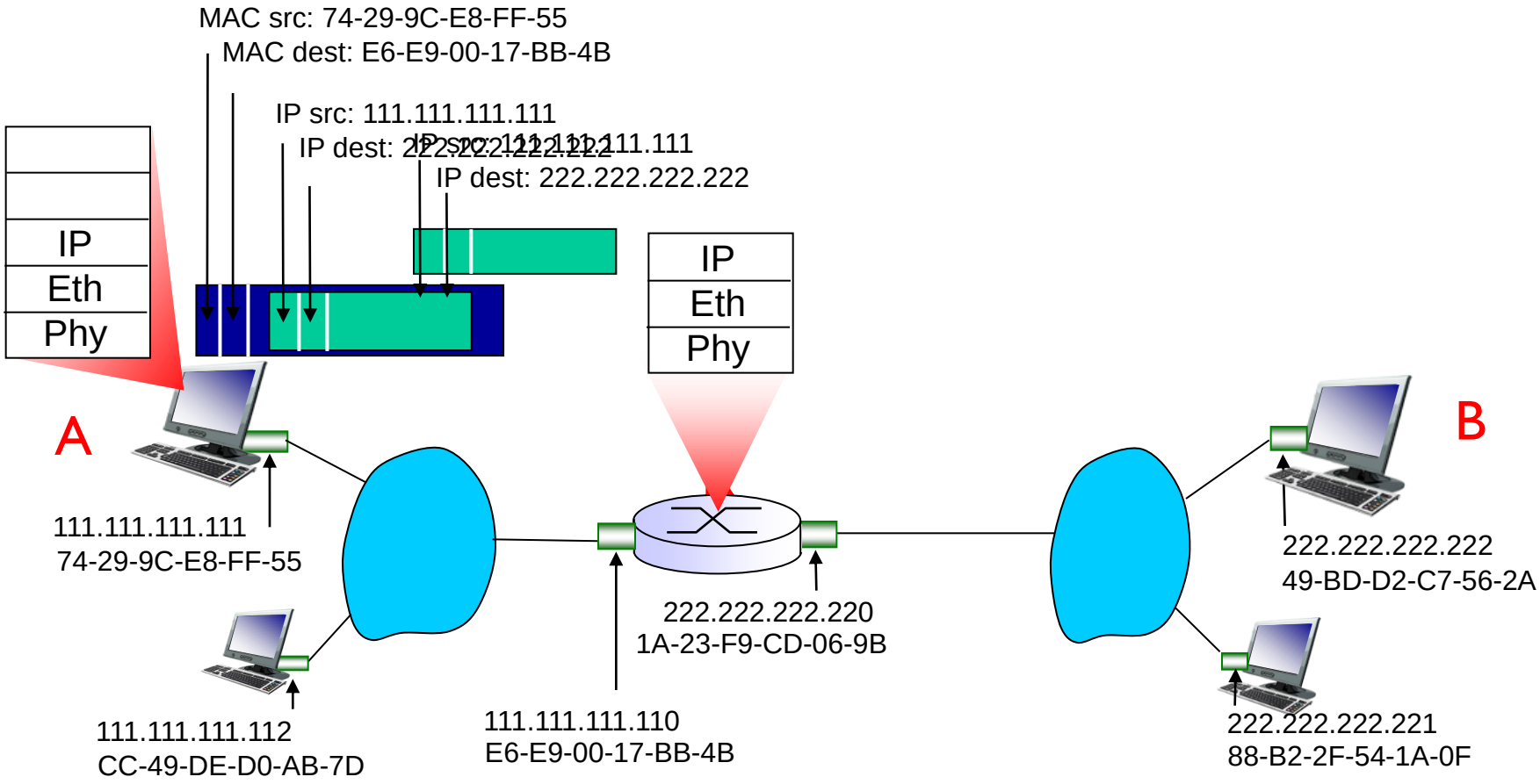
Addressing: δρομολόγηση σε άλλο LAN

- 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



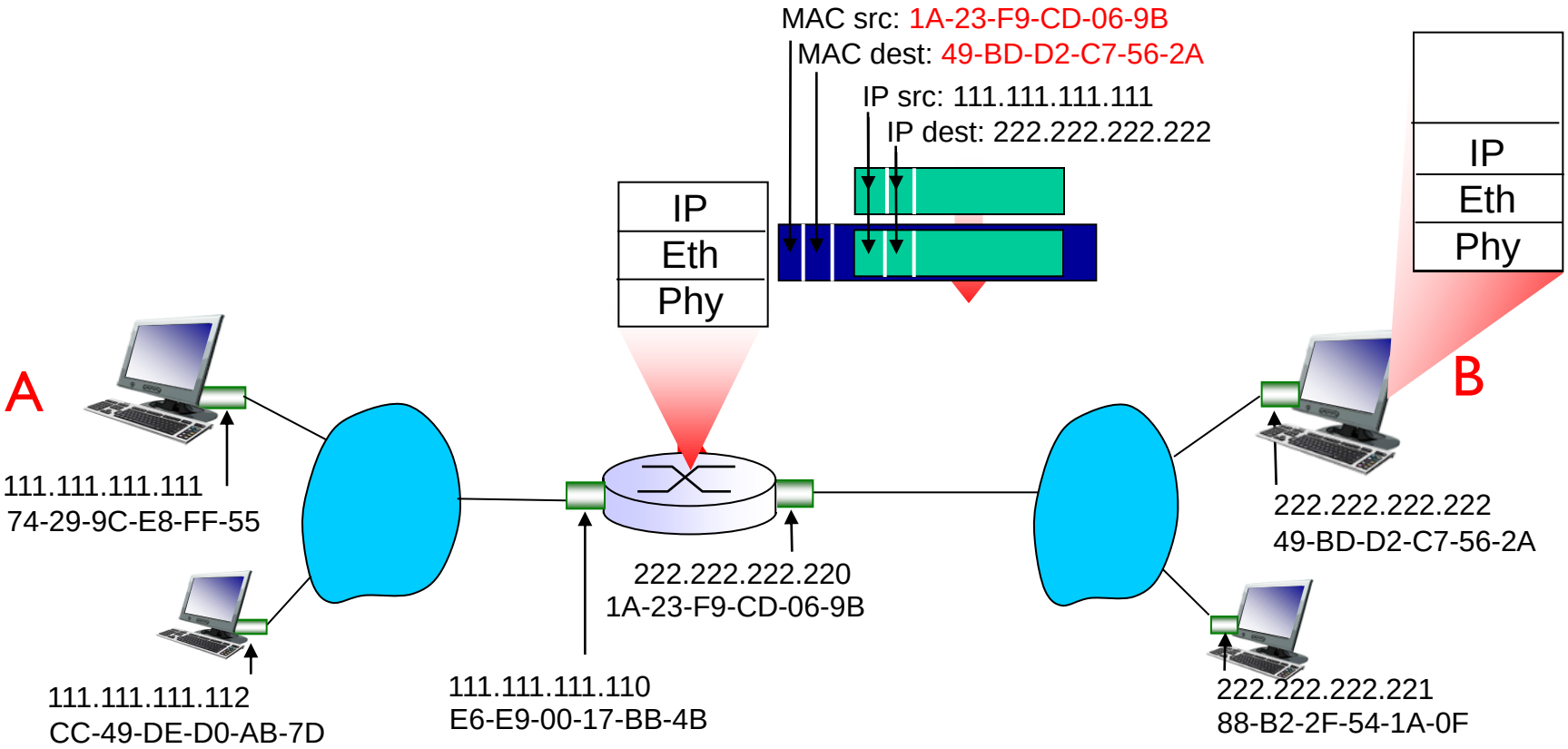
Addressing: δρομολόγηση σε άλλο LAN

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



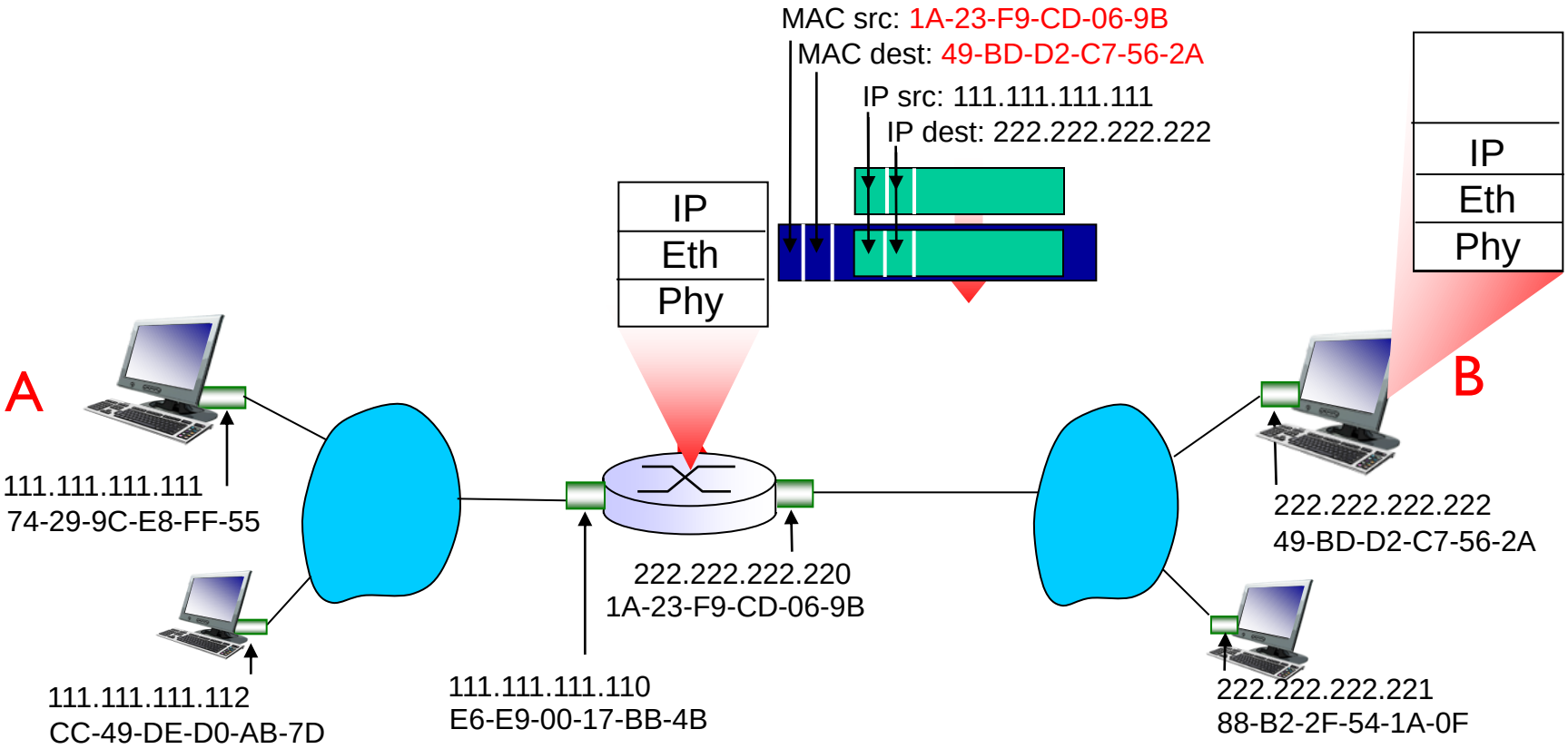
Addressing: δρομολόγηση σε άλλο LAN

- 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



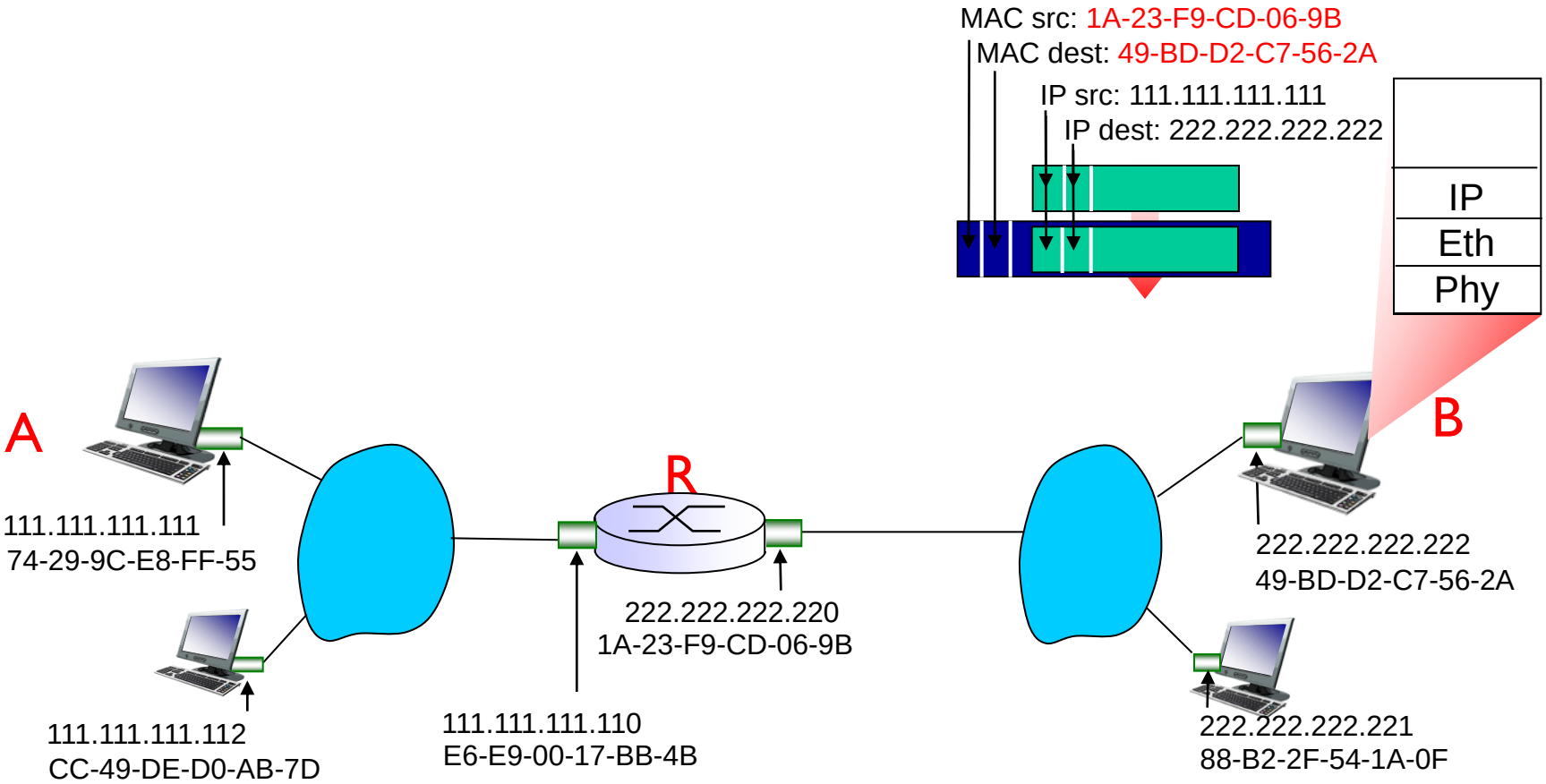
Addressing: δρομολόγηση σε άλλο LAN

- 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



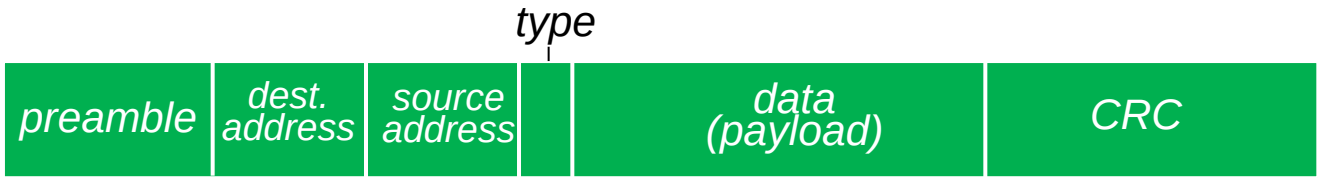
Addressing: δρομολόγηση σε άλλο LAN

- 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



Ethernet frame structure

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

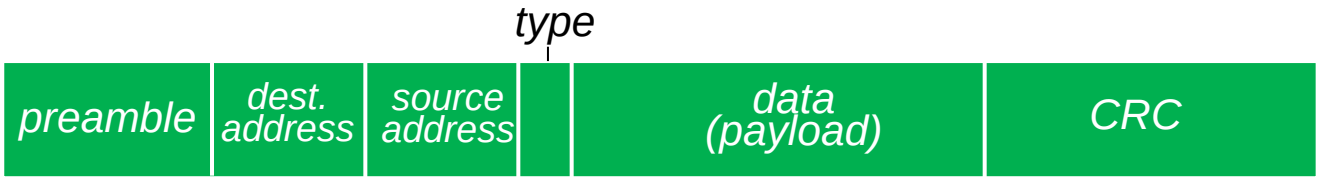


preamble:

- 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

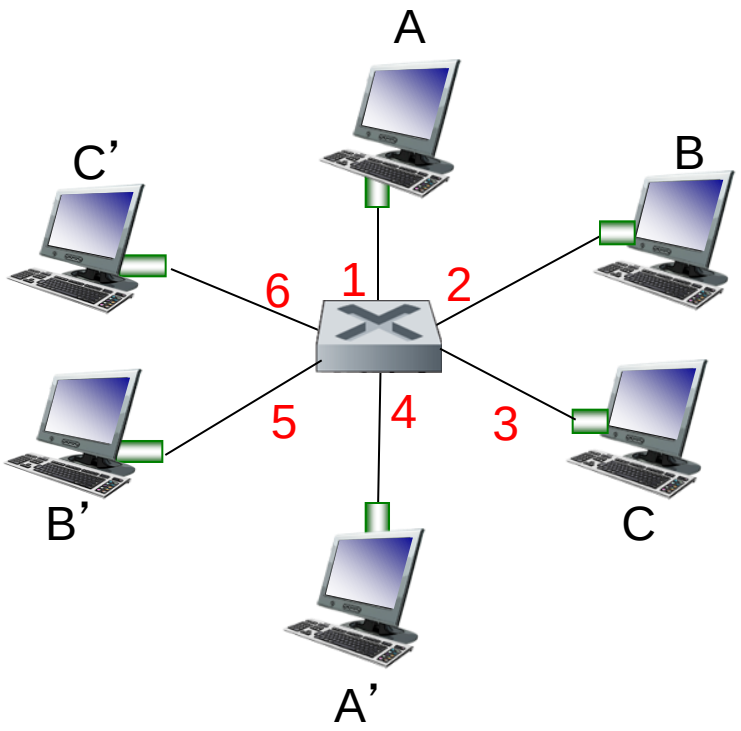


Ethernet switch

- 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)

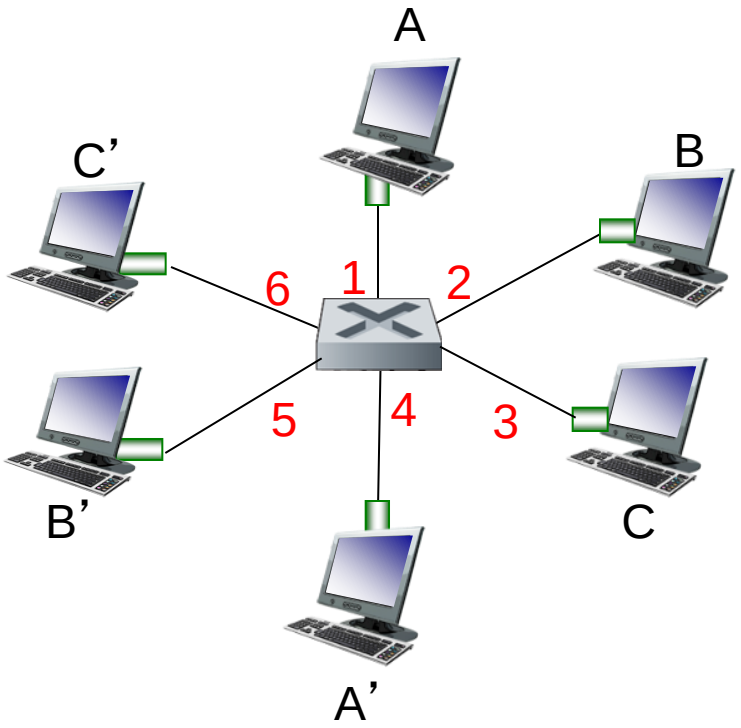
Switch forwarding table

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?

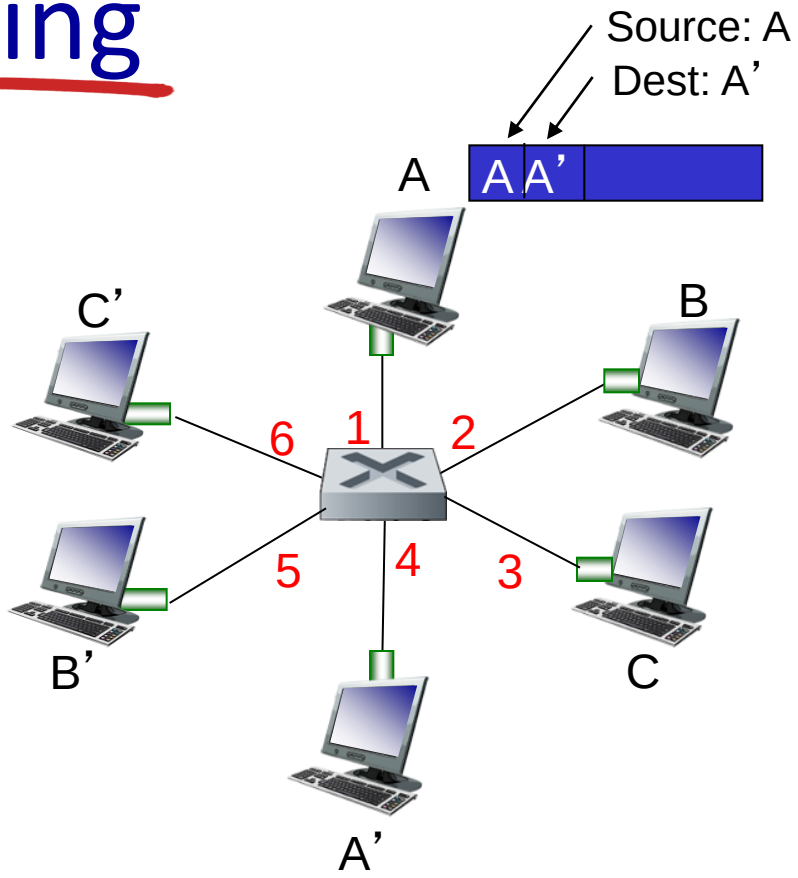
- *something like a routing protocol?*



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)

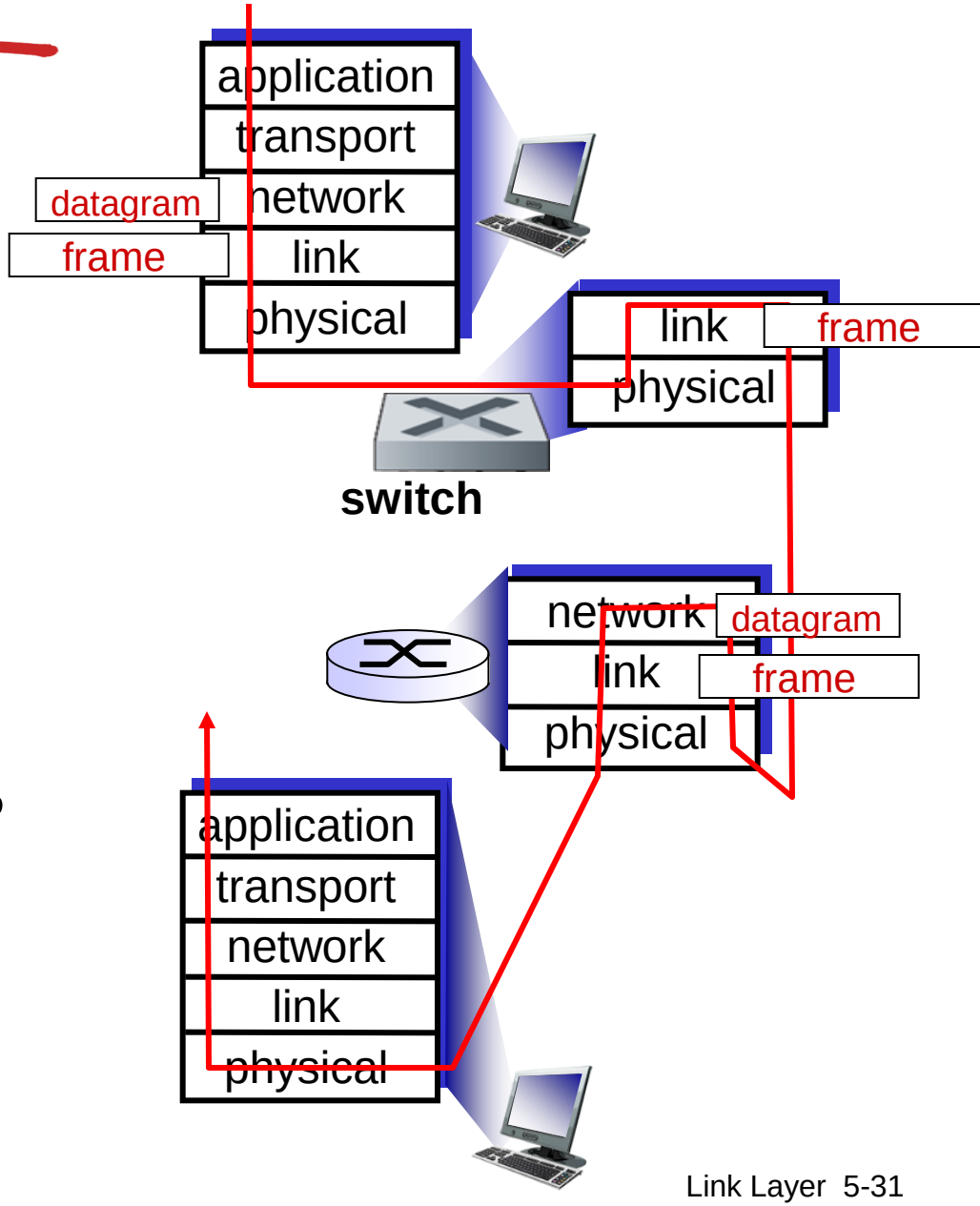
Switches vs. routers

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:

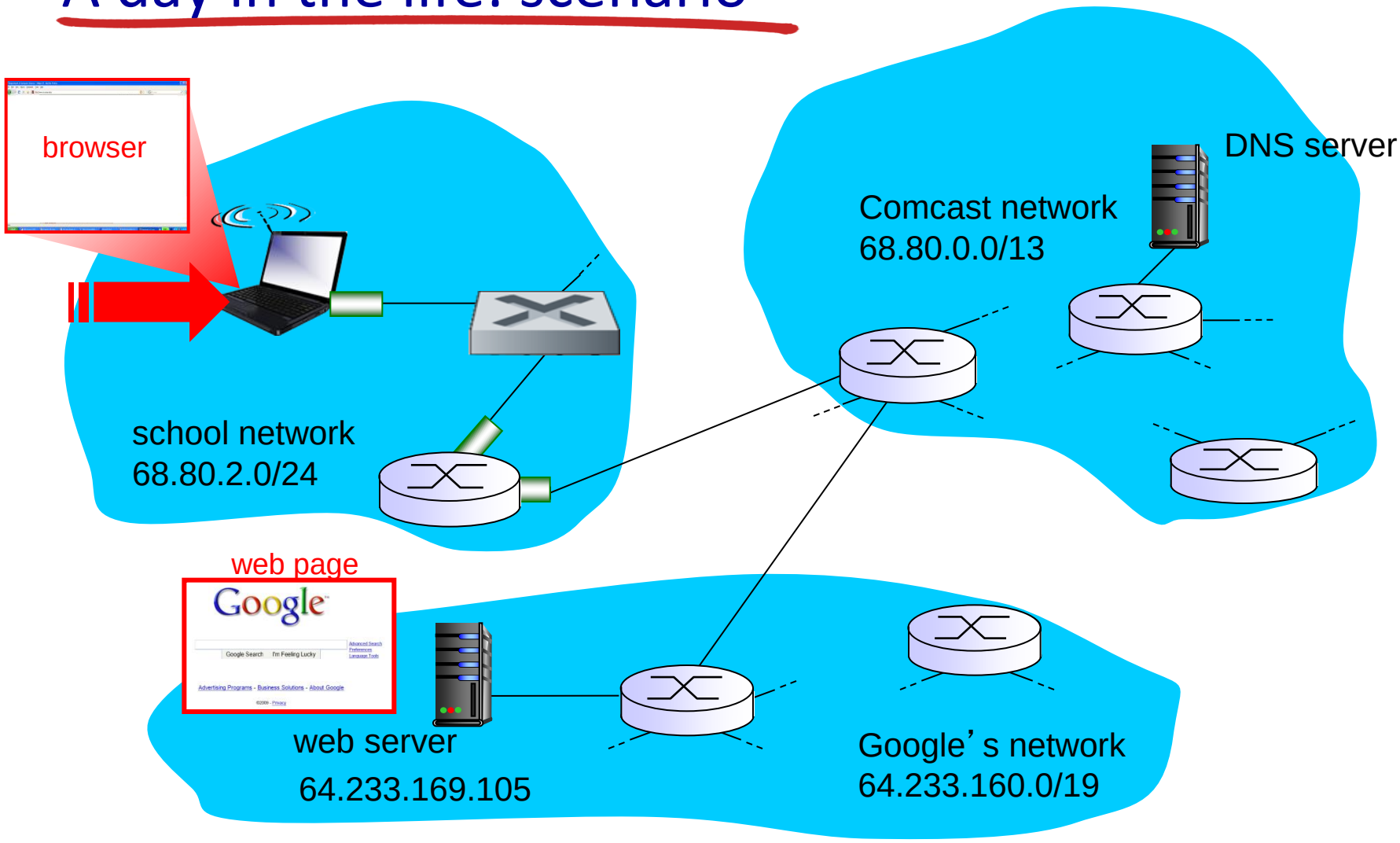
- *routers:* compute tables using routing algorithms, IP addresses
- *switches:* learn forwarding table using flooding, learning, MAC addresses



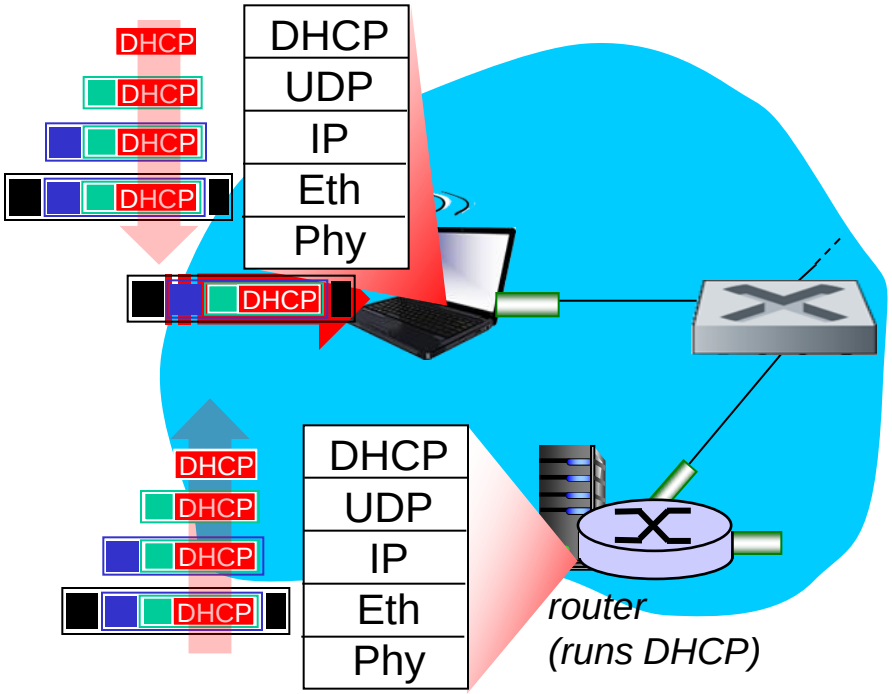
Synthesis: a day in the life of a web request

- 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

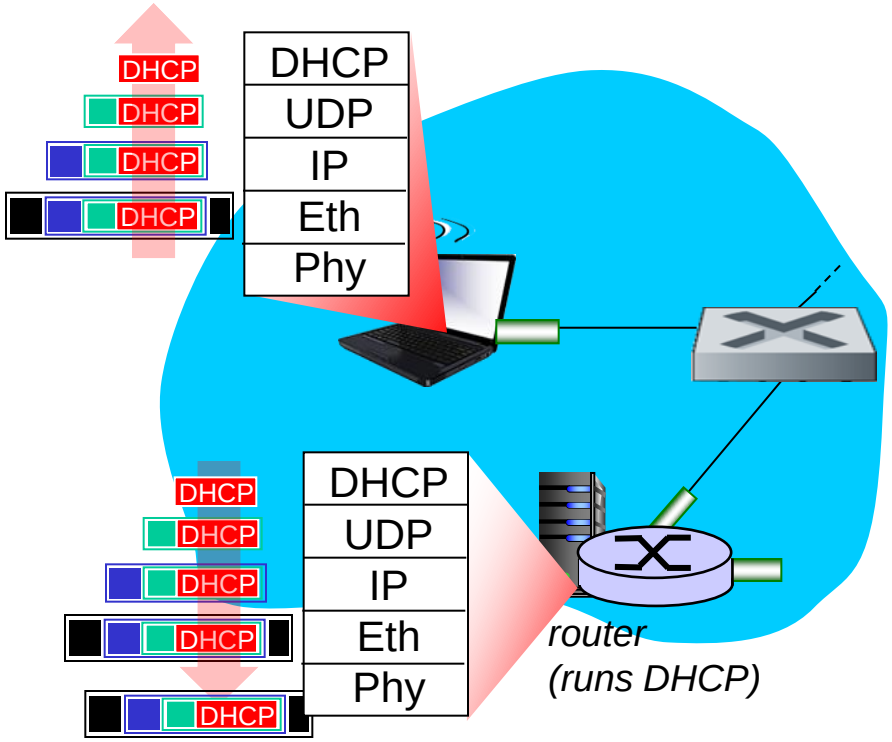


A day in the life… connecting to the Internet



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

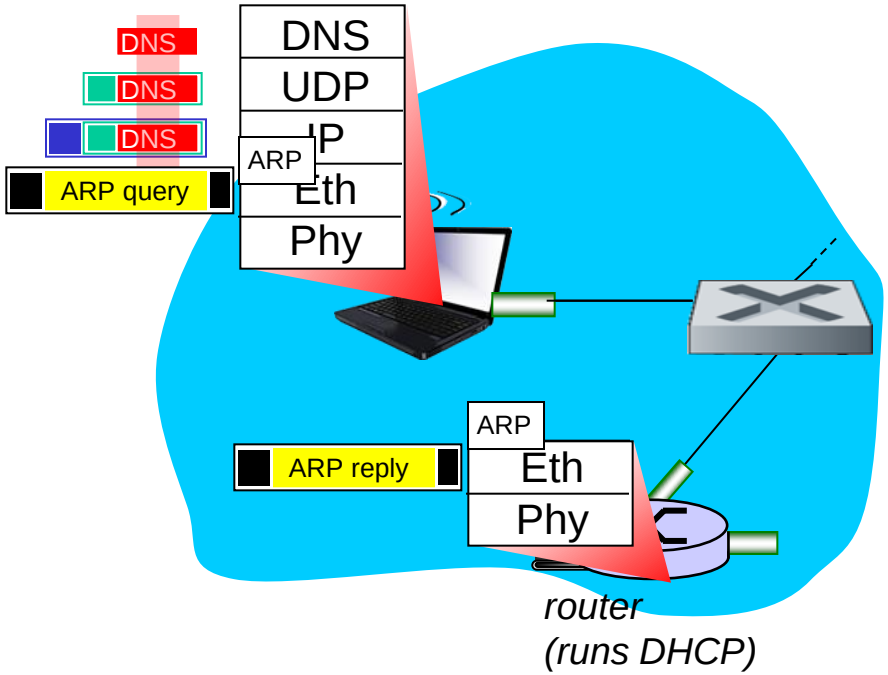
A day in the life… connecting to the Internet



- 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

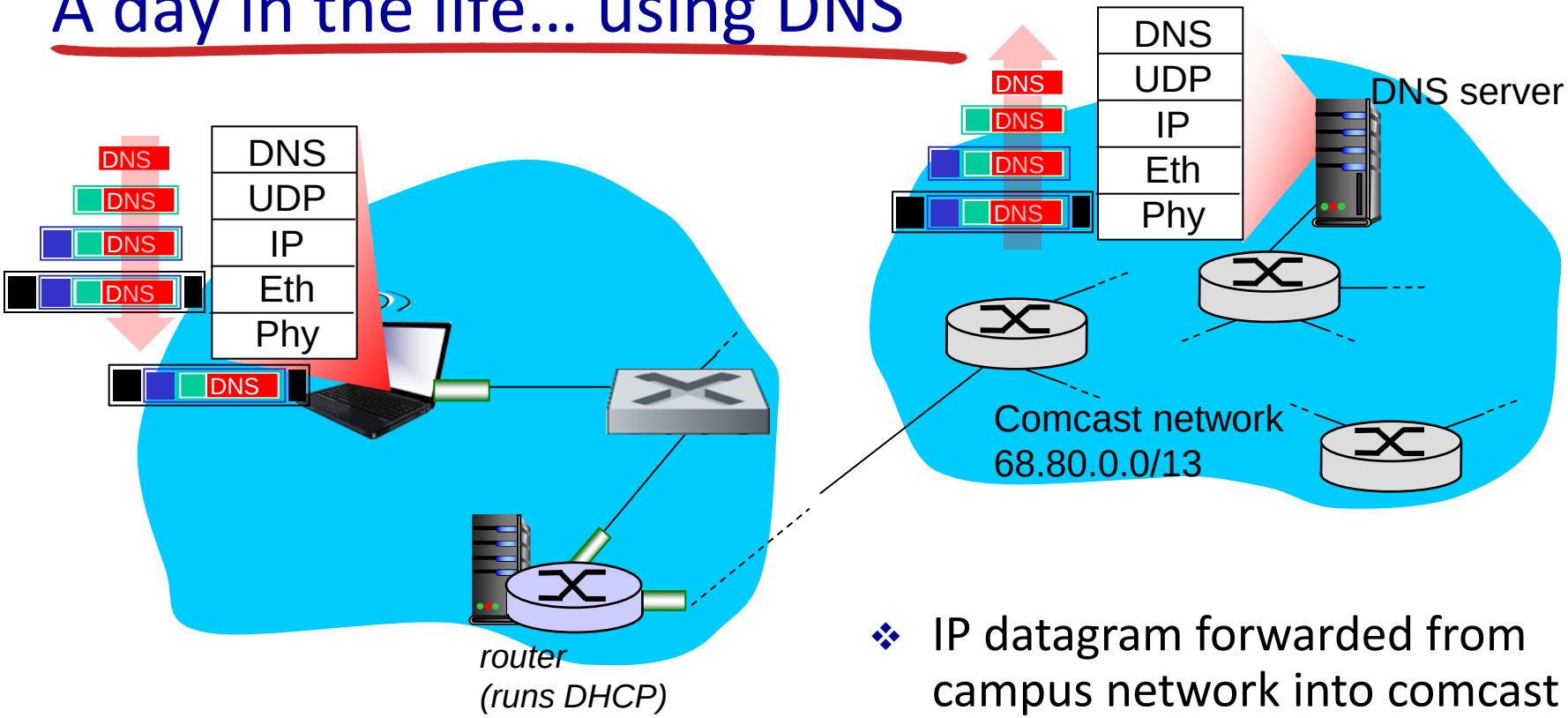
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

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

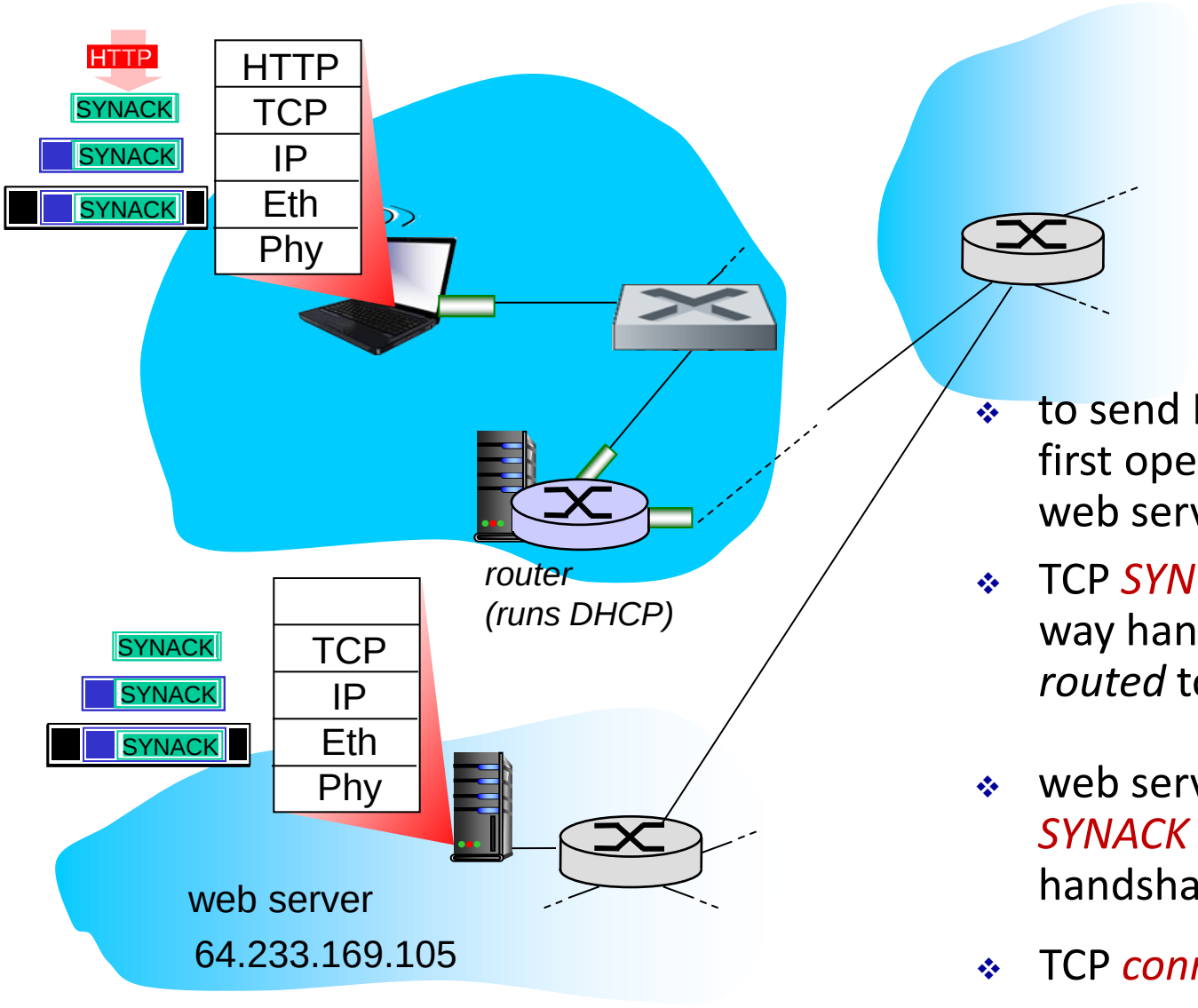
A day in the life… using DNS



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

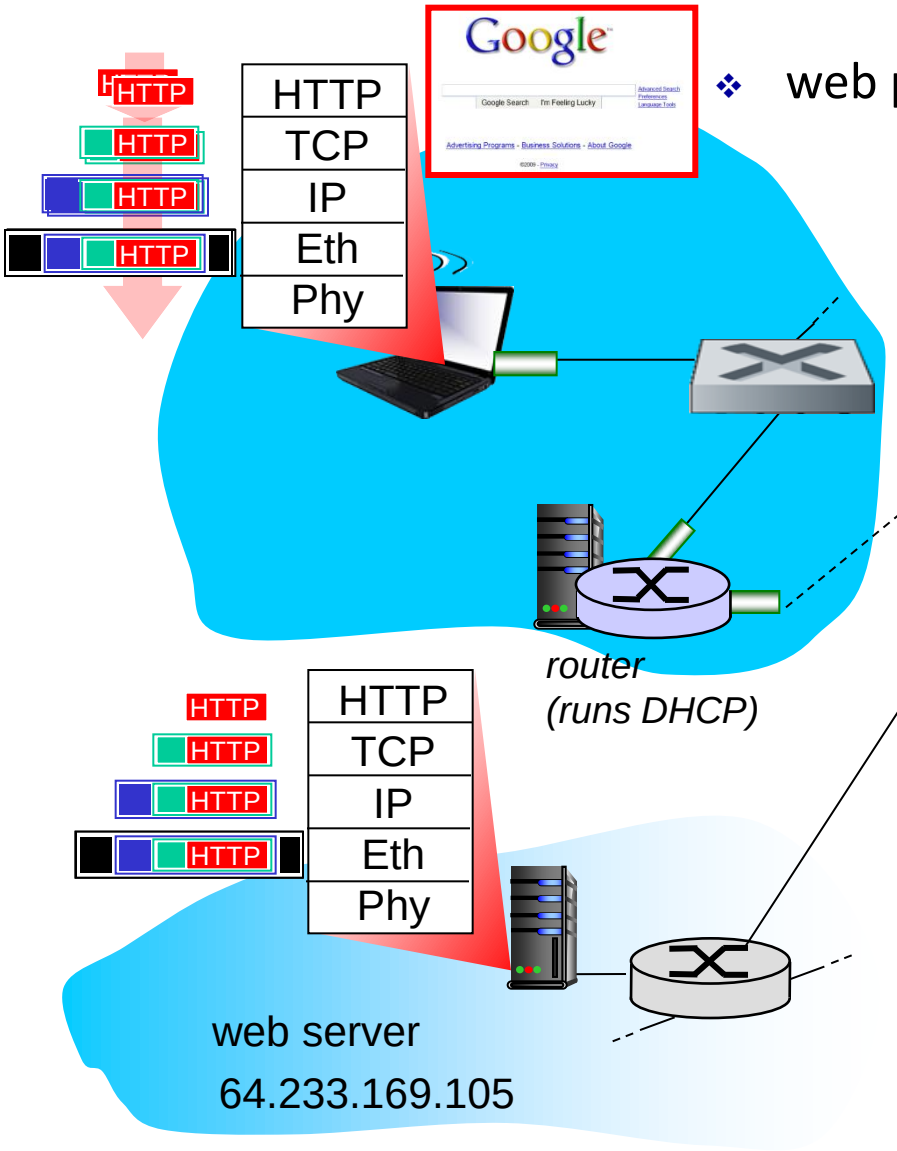
- IP datagram forwarded from campus network into comcast network, routed (tables created by *RIP, OSPF, IS-IS* and/or *BGP* routing protocols) to DNS server
- demux'ed to DNS server
- DNS server replies to client with IP address of www.google.com

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