# CS118 Discussion 1A, Week 9

Yunqi Guo
Boelter Hall 5422, Friday 10:00—11:50 p.m.

# Outline

- Chapter 6 review
    - Media access links
    - Ethernet
    - A day in the life

# Medium Access Links and Protocols

- Two types: point-to-point, broadcast
- **Broadcast** channel shared by multiple hosts
  - What if we only have unicast channel?
  - What’s the pros and cons for a broadcast channel?
- Three classes of Multiple Access Control (MAC) protocols
  - Channel partitioning: FDMA, TDMA, CDMA
  - Random access: Aloha, CSMA/CD, Ethernet
  - Taking turns: Token ring/passing
  - **Pros and cons for each class of protocol?**

# Random access: 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

# Random access: slotted ALOHA

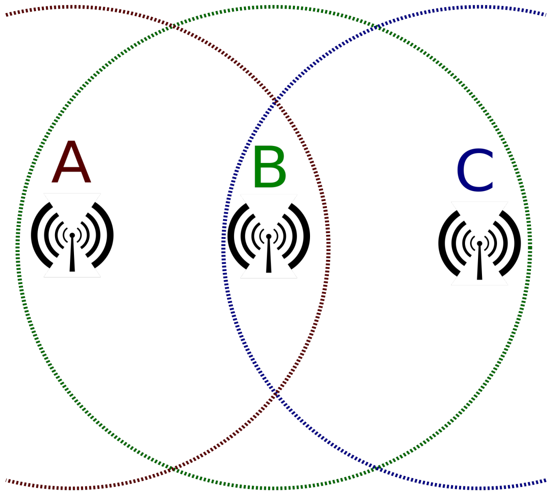
- suppose: N nodes with many frames to send, each transmits in slot with probability p
- Pr(given node has success in a slot) = p(1-p)^(N-1)
- Pr(any node has a success) = Np(1-p)^(N-1)
- max efficiency: find p* that maximizes Np(1-p)^(N-1)
- Take the limit of Np*(1-p*)^(N-1) as N goes to infinity, yields:
  - max efficiency = 1/e = .37

# Random access: ALOHA efficiency

- Slotted ALOHA max efficiency = 1/e = .37
- Unslotted ALOHA max efficiency = 1/2e = .18

# CSMA (carrier sense multiple access)

- Listen before transmit:
  - if channel sensed idle: transmit entire frame
  - if channel sensed busy, defer transmission
  - "don't interrupt others!"
- Channel busy?
  - 1-persistent CSMA: retry immediately
  - p-persistent CSMA: retry immediately with probability p
  - Non-persistent CSMA: retry after a random interval
- Collision?
  - hidden terminal problem



# 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

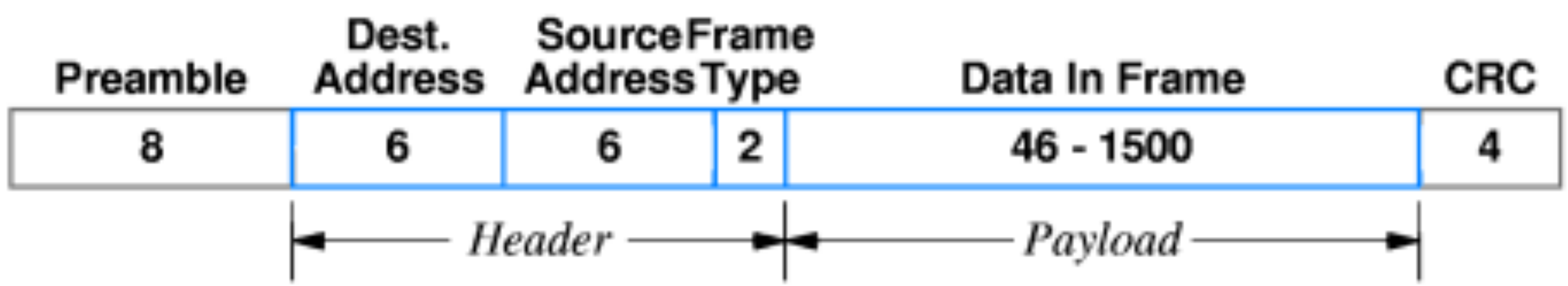
# Ethernet

- Connectionless and unreliable protocol
- MAC protocol: CSMA/CD + exponential backoff
- Switch-based Ethernet
  - No real broadcast channel anymore
  - Self-learning algorithm: support plug-and-play

# MAC address

- MAC address allocation by IEEE (who assigns IP?)
- MAC address is flat -> portability (IP address is ___?)
- Format: 48 bit address
  - AA-BB-CC-DD-EE-FF
  - Broadcast address: FF-FF-FF-FF-FF-FF

# Ethernet Frame




- Min frame size: 64 Bytes
- Max frame size: 1514 Bytes

# Ethernet CSMA/CD

- 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:

# Switch

- Examine each incoming frame's MAC address, forward to the destination LAN if dest. host is on a different LAN
- store-and-forward
- switch table: self-learning algorithm
  - (MAC address of host, interface to reach host, timestamp)

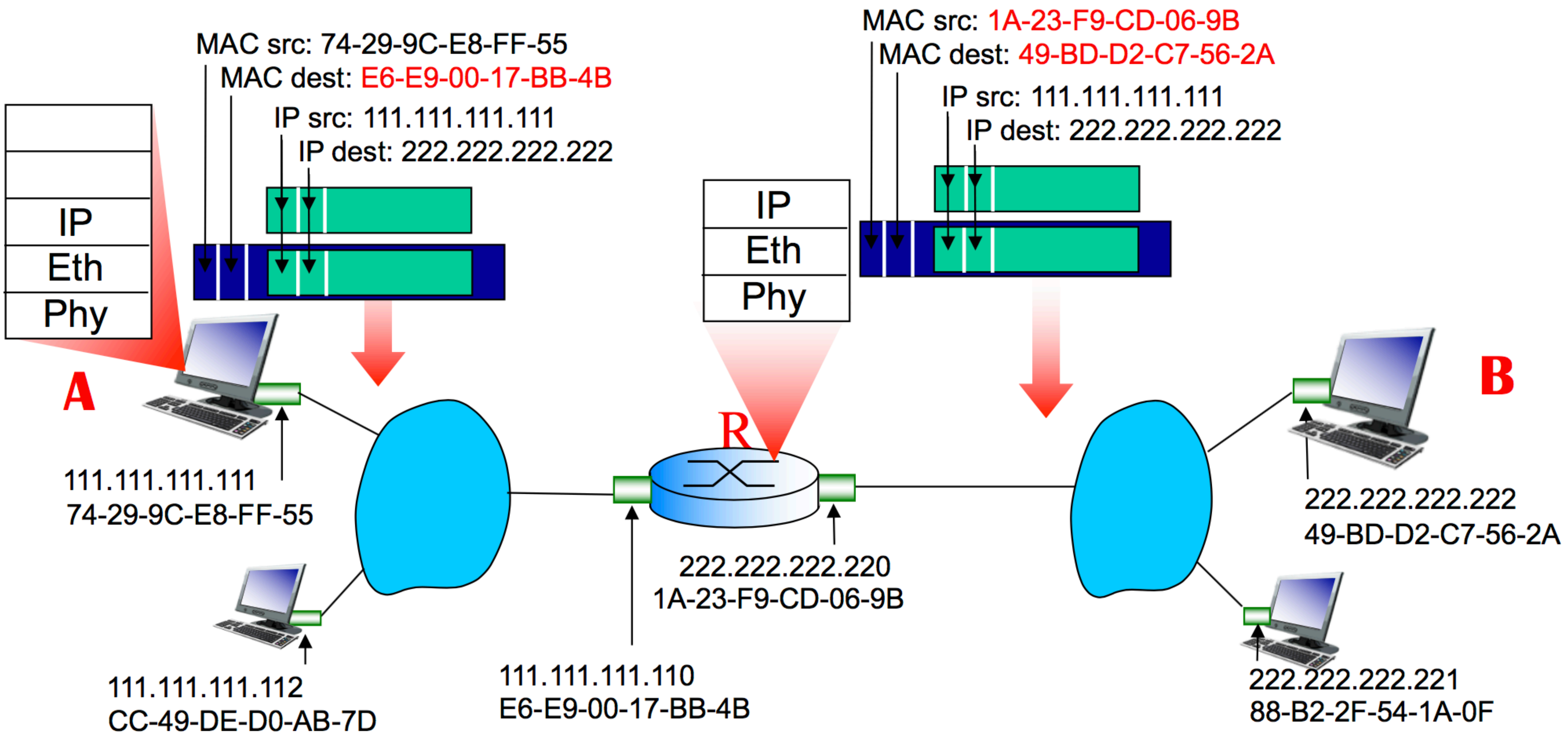
# ARP: address resolution protocol

- 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>
  - called PnP (plug-and-play)
  - soft-state design: information deletes itself after certain time unless being refreshed

# ARP: send an IP packet in the same subnet

- A wants to send IP packet to B, but B’s MAC address not in A’s ARP table.
- A broadcasts ARP query packet, containing B's IP address (all nodes on LAN receive ARP query)
  - dest MAC address = FF-FF-FF-FF-FF-FF
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A’s MAC address (unicast)
- A caches IP-to-MAC address pair in its ARP table until information becomes old (times out)

# ARP: send an IP packet across subnets



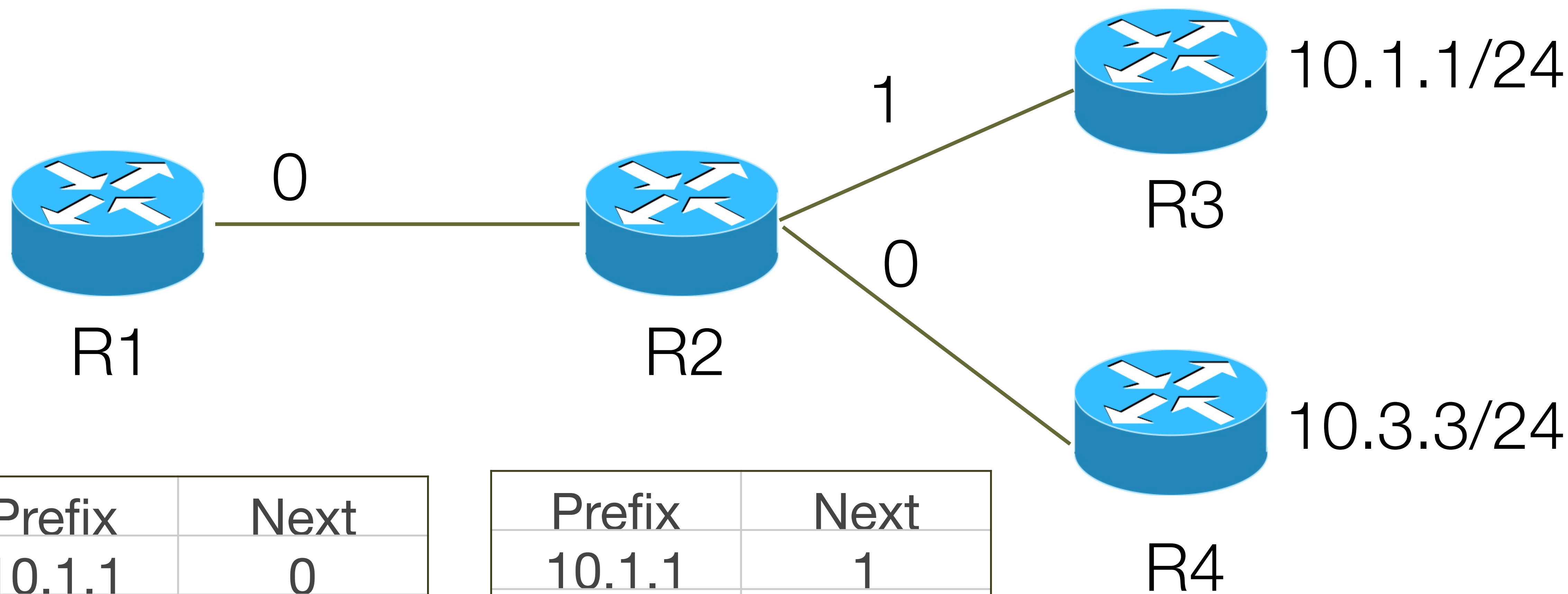
# Router vs. Switch

- Both are store-and-forward devices
  - routers: network layer devices (examine IP headers)
  - switches: link layer devices (examine Ethernet headers)
- Circuit-switch network: connection should be established before forwarding the data
  - At each hop, the circuit path is marked as a label
  - Data forwarding is based on label: **O(1) complexity**
  - **Vulnerable to link/node failures**
- Packet-switched network: connectionless, packets are forwarded based on IP header
  - Longest prefix matching: **O(N) complexity**
  - **Robust to link/node failures**
- **Can we take advantage of both, while preventing any vulnerabilities?**

# Multi-Protocol Label Switching

- Idea: **switching technique** into **connectionless** network
- In IP routing table, each entry is associated with a label
- Neighboring routers exchange labels, and forms an index of next hop's forwarding table
- When forwarding the packet, lookup the index only
  - Only the first hop performs longest prefix matching

# Exchanging labels between routers

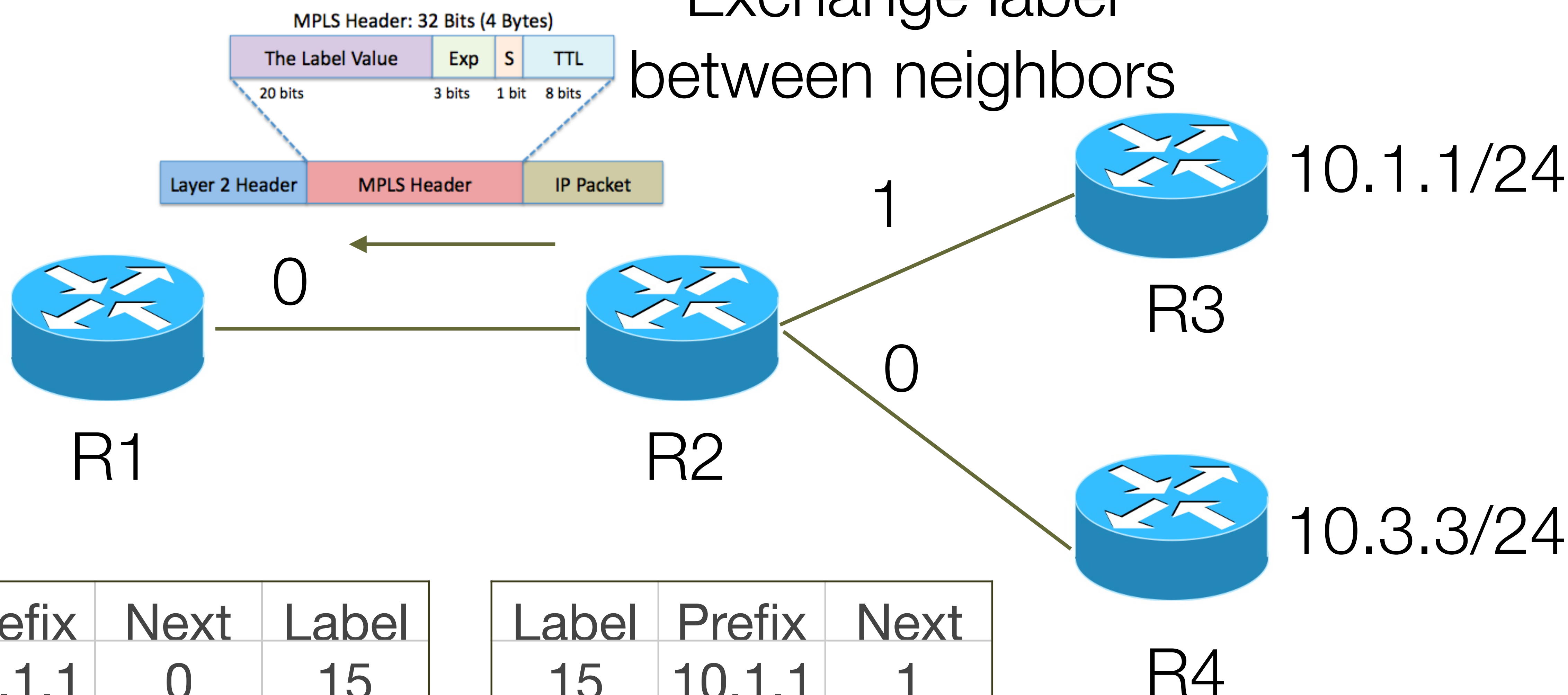



| Prefix | Next |
|---|---|
| 10.1.1 | 0 |
| 10.3.3 | 0 |

| Prefix | Next |
|---|---|
| 10.1.1 | 1 |
| 10.3.3 | 0 |

# Exchanging labels between routers

Exchange label between neighbors



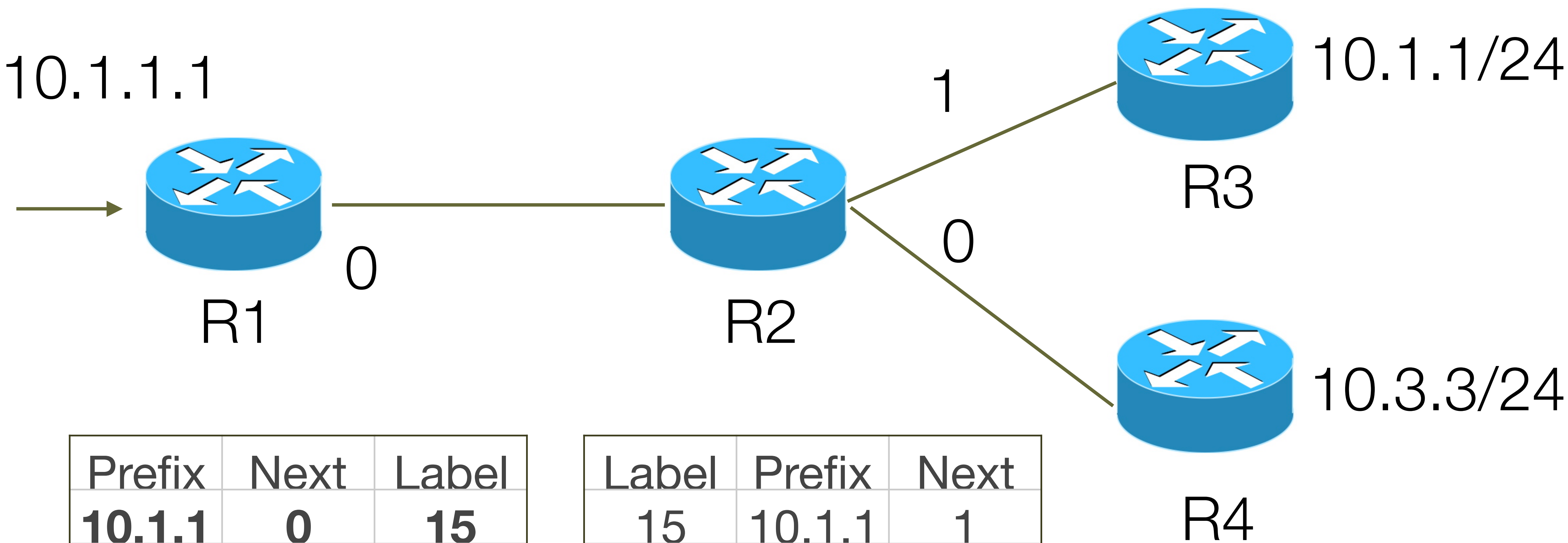

| Prefix | Next | Label |
|---|---|---|
| 10.1.1 | 0 | 15 |
| 10.3.3 | 0 | 16 |

Store remote labels

| Label | Prefix | Next |
|---|---|---|
| 15 | 10.1.1 | 1 |
| 16 | 10.3.3 | 0 |

Allocate label for each entry

# Forwarding packets

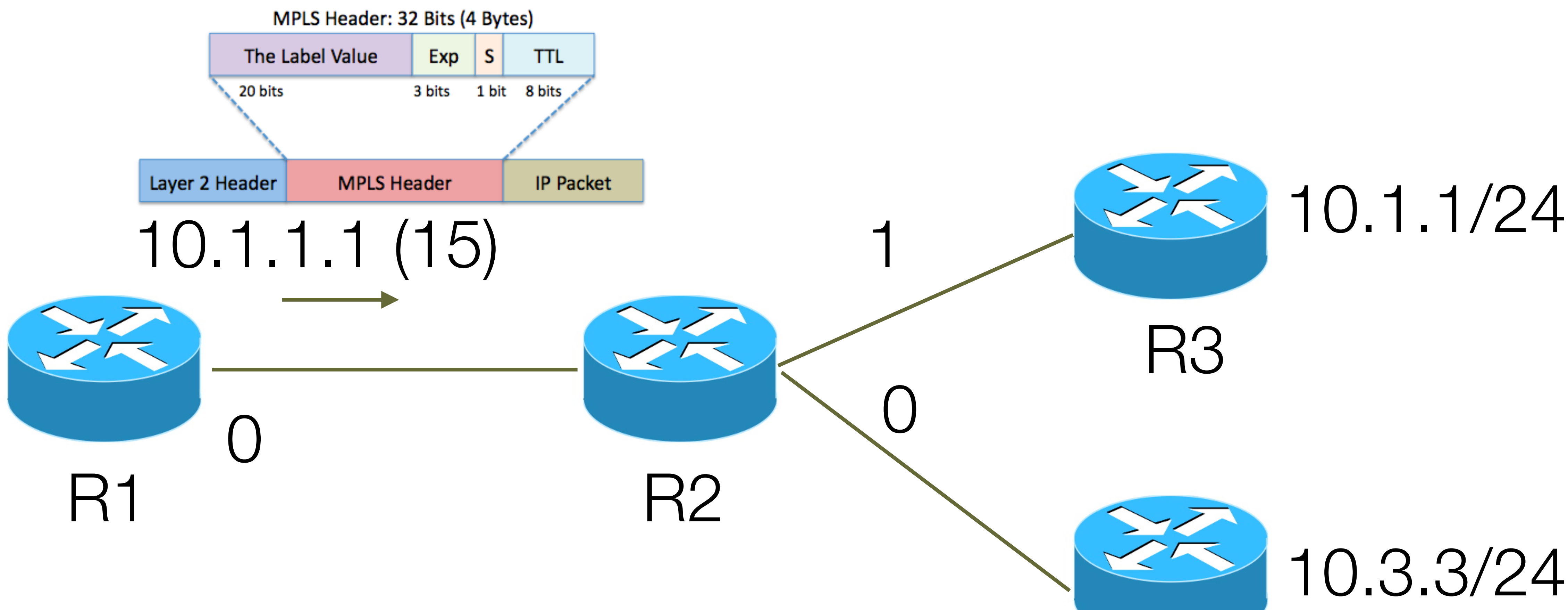



| Prefix | Next | Label |
|---|---|---|
| **10.1.1** | **0** | **15** |
| 10.3.3 | 0 | 16 |

| Label | Prefix | Next |
|---|---|---|
| 15 | 10.1.1 | 1 |
| 16 | 10.3.3 | 0 |

Longest Prefix Matching

# Forwarding packets




| Prefix | Next | Label |
|---|---|---|
| 10.1.1 | 0 | 15 |
| 10.3.3 | 0 | 16 |

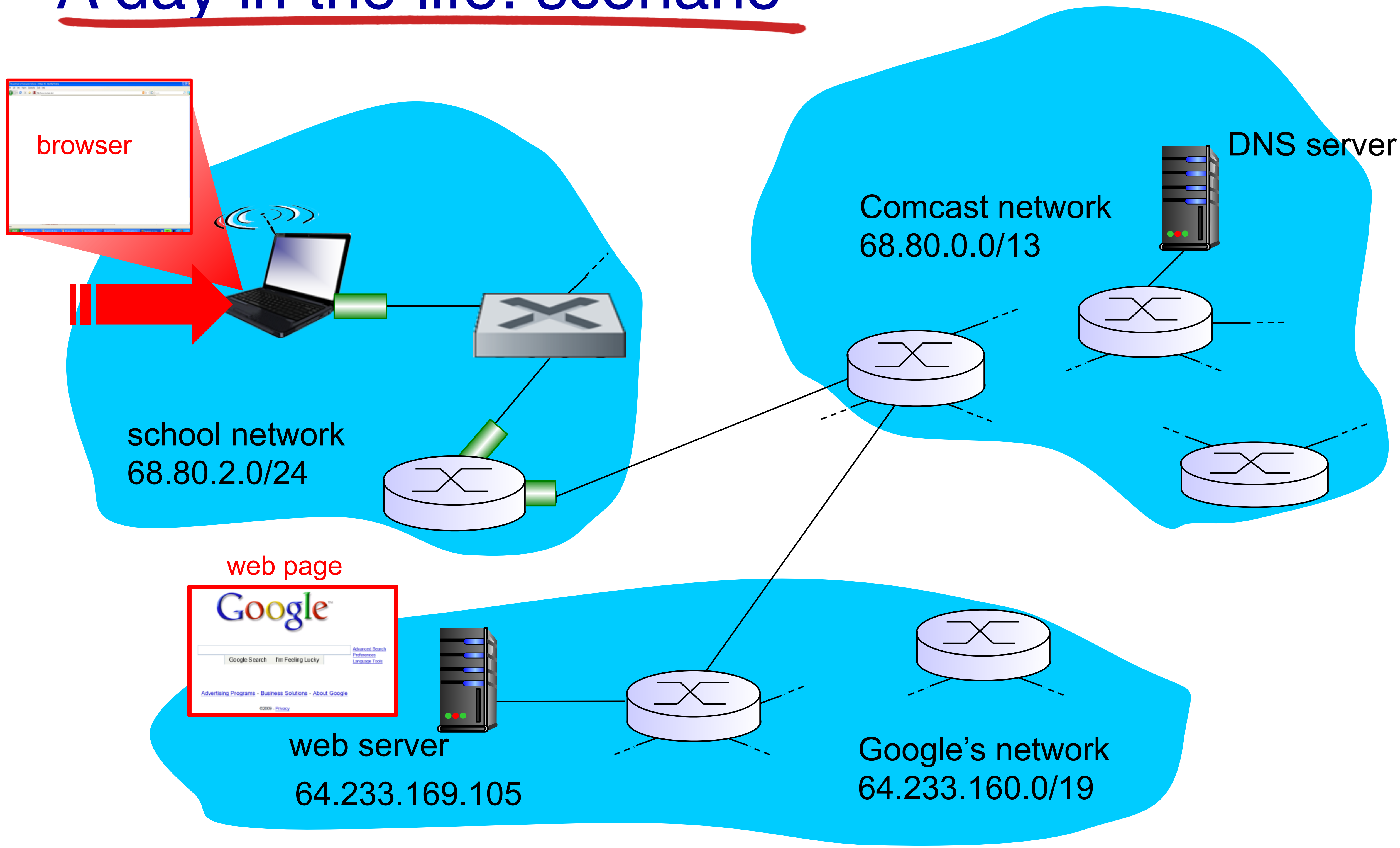
| Label | Prefix | Next |
|---|---|---|
| **15** | **10.1.1** | **1** |
| 16 | 10.3.3 | 0 |

Label-based Switching

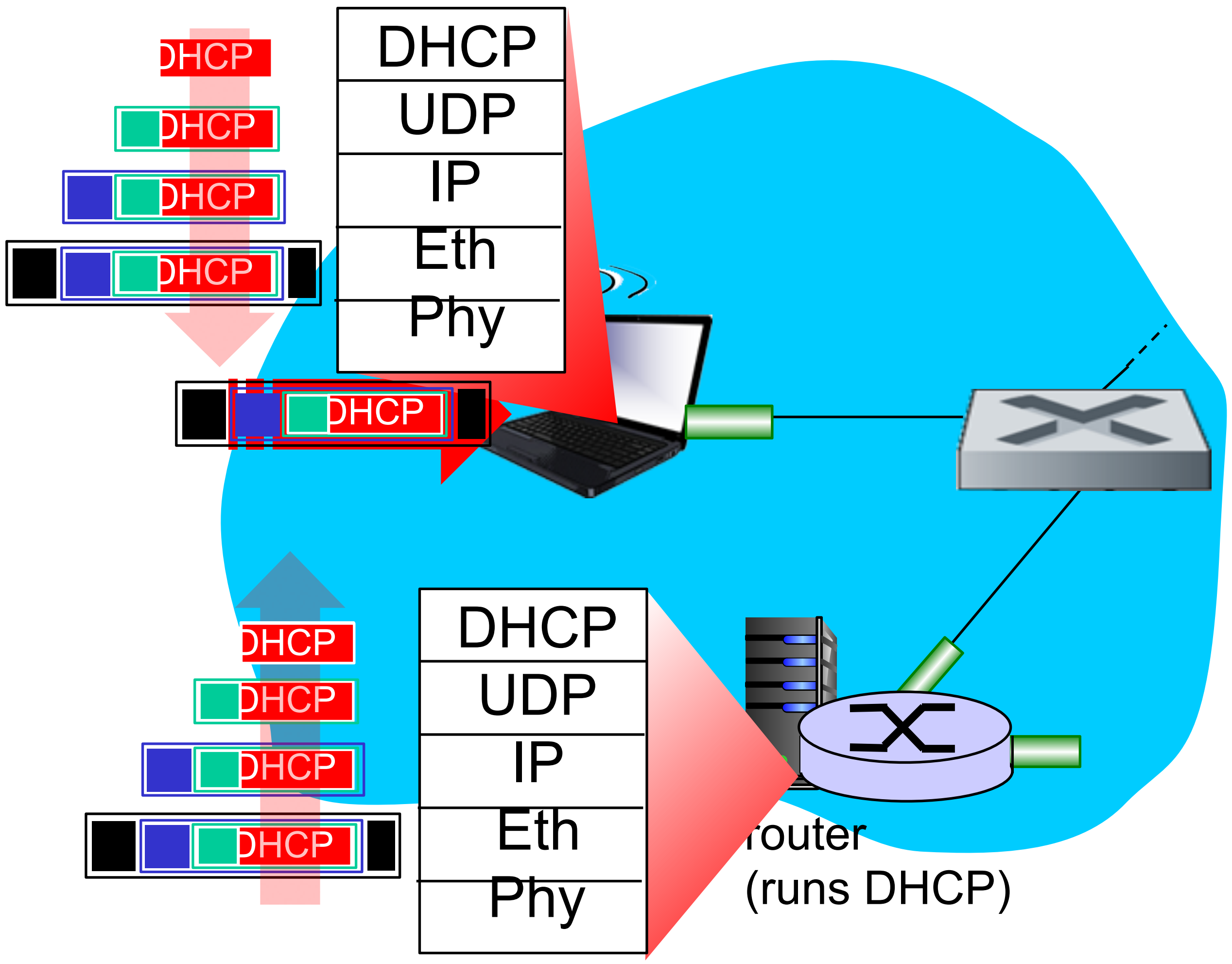
# Other Benefits

- In IP table, the IP addresses are aggregated based on prefixes
- In MPLS, the IP addresses can be aggregated in more flexible ways
  - How: assign same label for IP addresses in the same category
  - Benefit: better management (e.g., offer different levels of performance using different labels)

# A day in the life: scenario

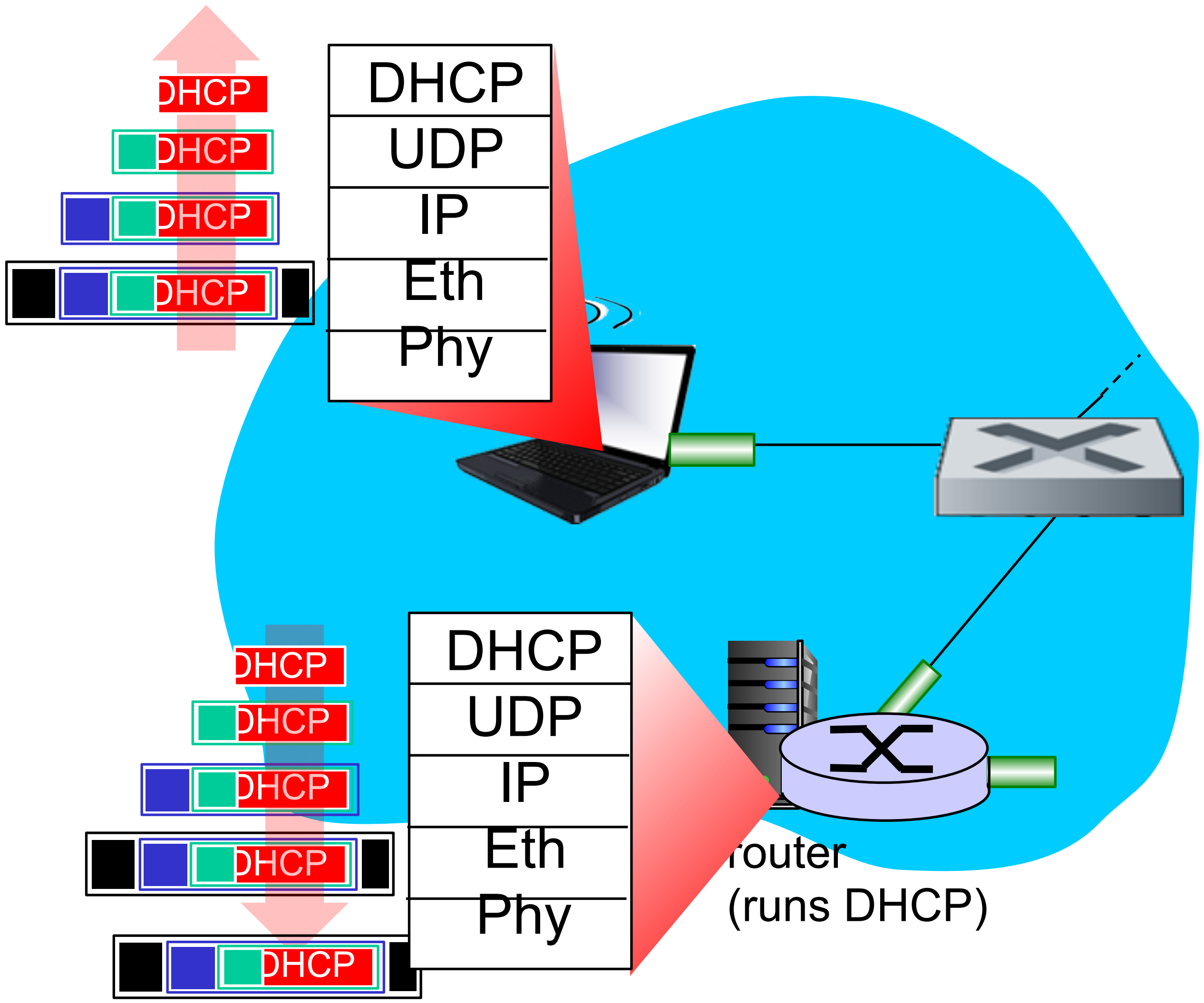


# A day in the life… connecting to the Internet




- 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 (ip.src = 0.0.0.0; ip.dst = 255.255.255.255)
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

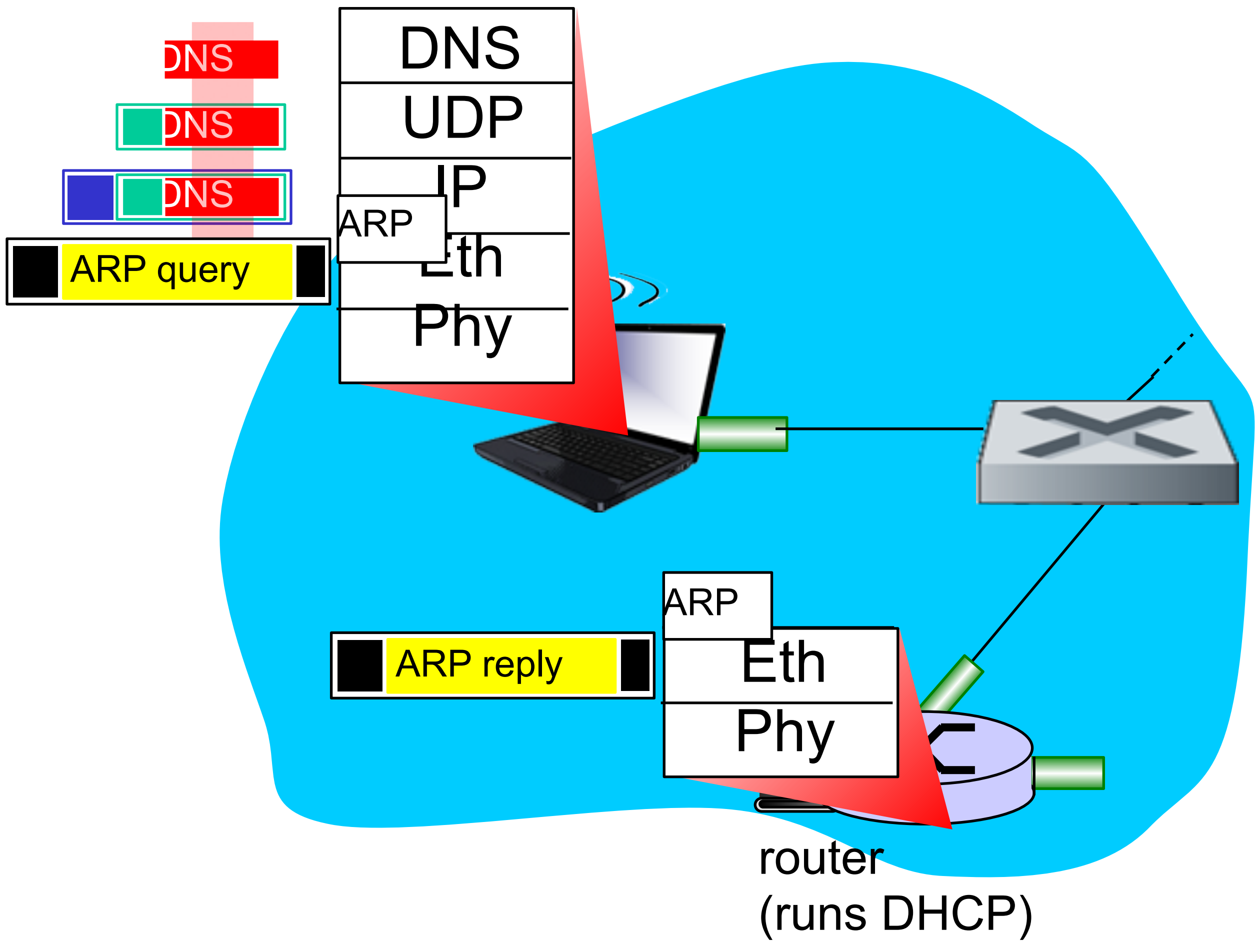
# A day in the life… connecting to the Internet




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

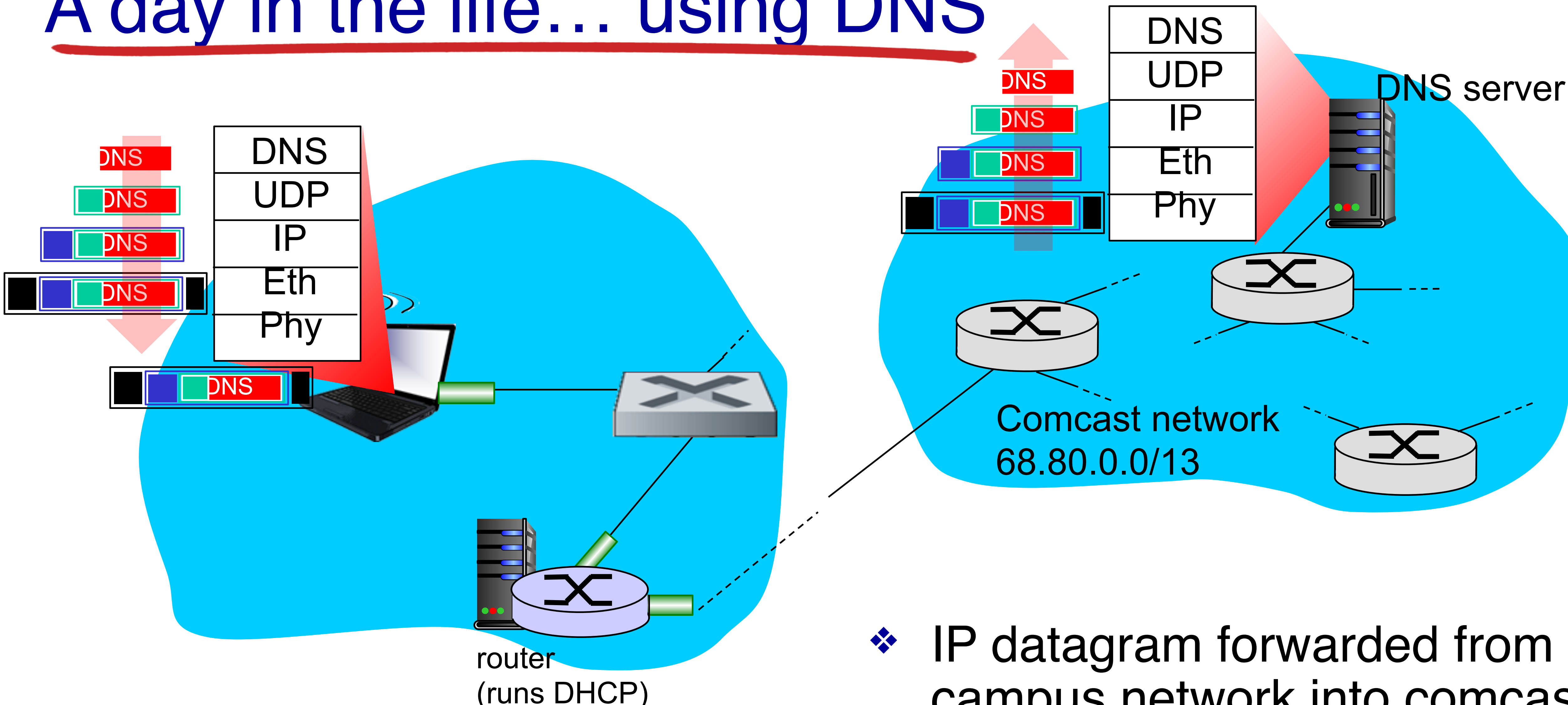
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

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



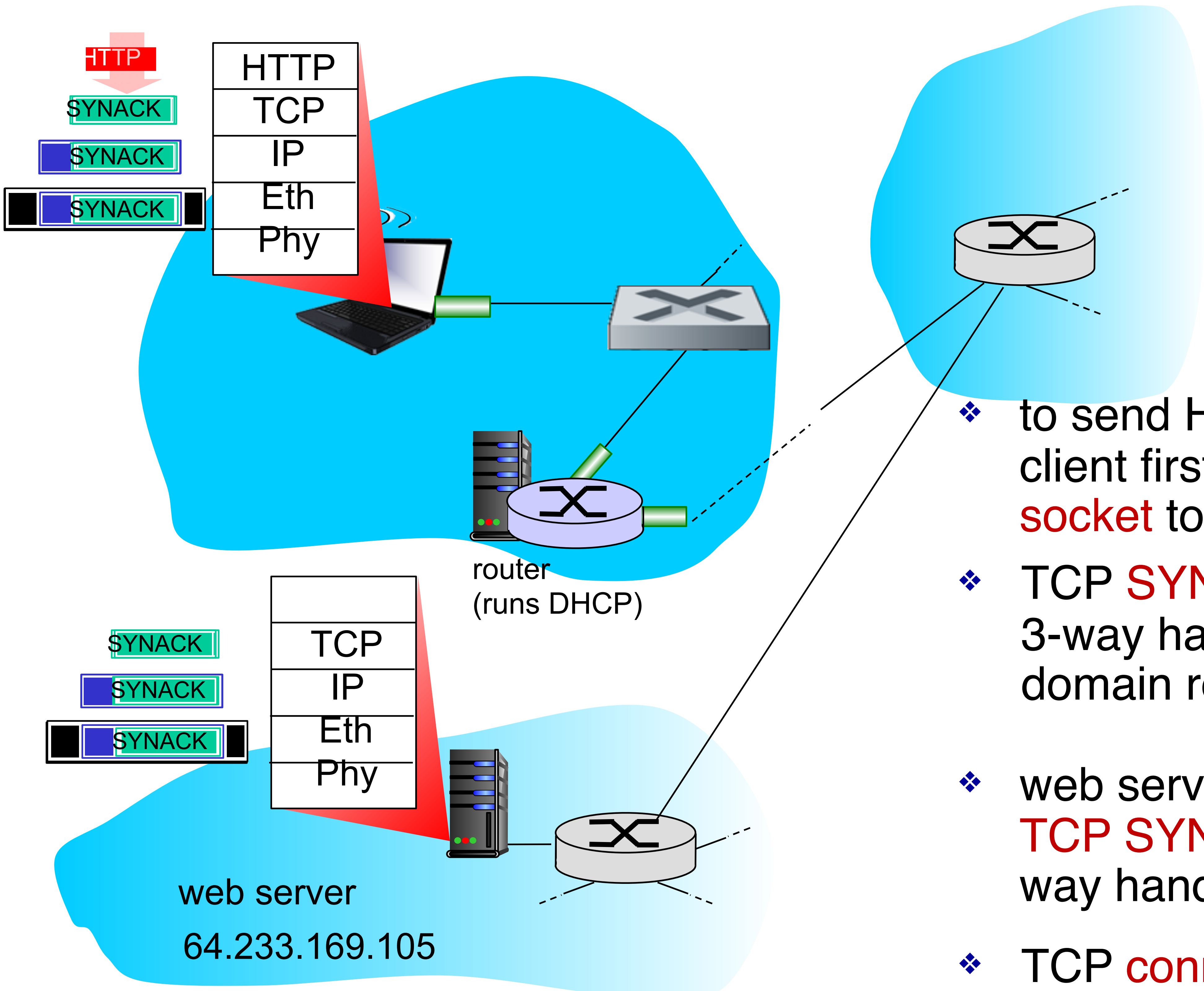

- before sending *HTTP* request, need IP address of www.google.com: *DNS*
- DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. To send frame to router, need MAC address of router interface: ARP
- ARP query broadcast, received by router, which replies with ARP reply giving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

# A day in the life… using DNS



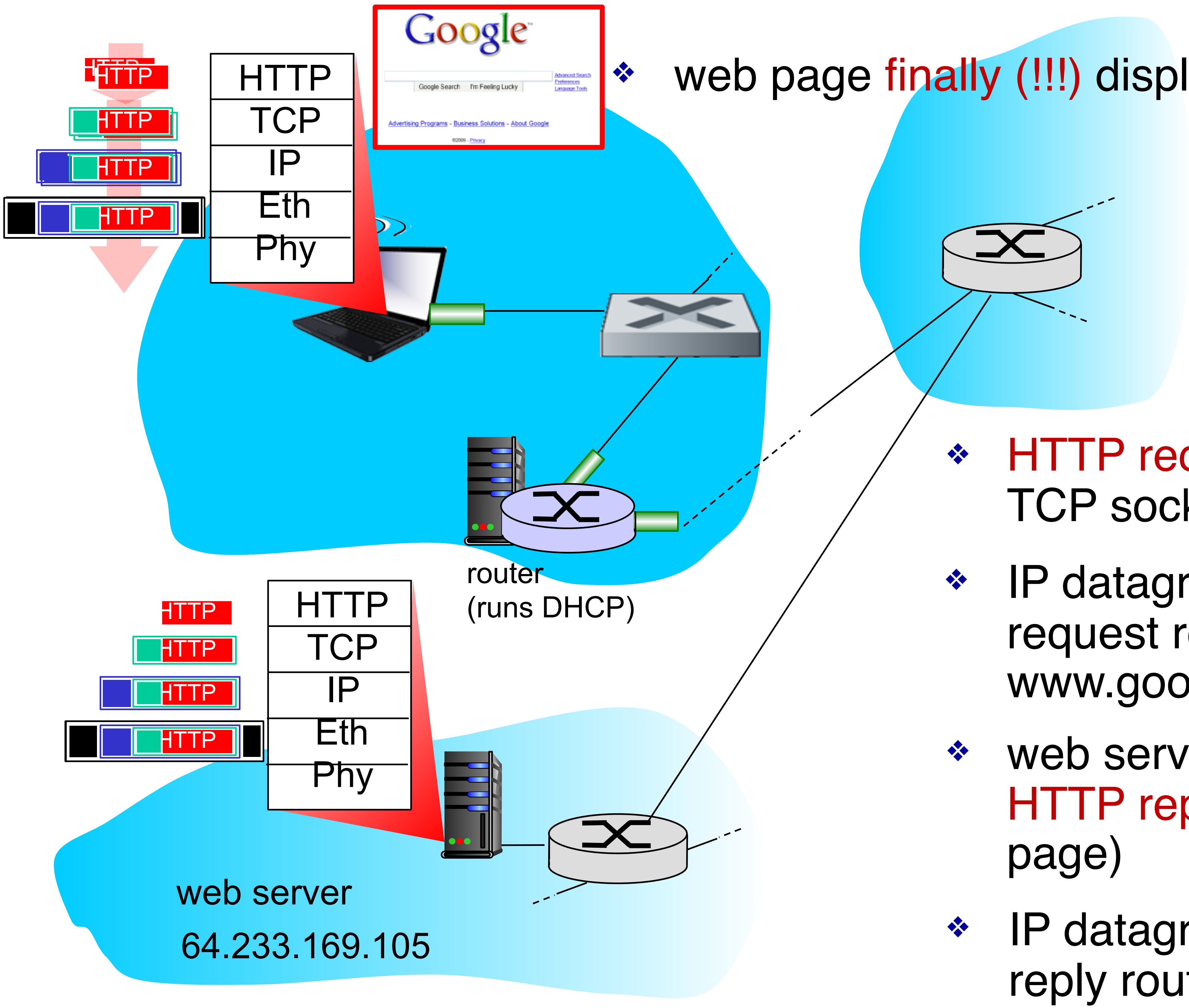

- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router
- IP datagram forwarded from campus network into comcast network, routed (tables created by RIP, OSPF, IS-IS and/or BGP routing protocols) to DNS server
- demux'ed to DNS server
- DNS server replies to client with 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