Applied Virtual Networks

COMP 4912

Instructor: **Dawood Sajjadi**

PhD, SMIEEE, CISSP

ssajjaditorshizi@bcit.ca

Winter-Spring 2025

Week #1

BCIT™

Class Introduction (15 sec per student)

1. Your Name
2. You Program
3. Windows/Mac/Linux
4. What Expect to Learn

BCIT™

A Brief History of Time

Universe created ~13.7 billion years ago.

Planet Earth created ~4.5 billion years ago.

Modern Humans showed up ~300,00 years ago.

First **Civilizations** emerged ~12,000 years ago.

First **Written records** created ~5,000 years ago.

First **Analog Computer** created ~2,200 years ago.

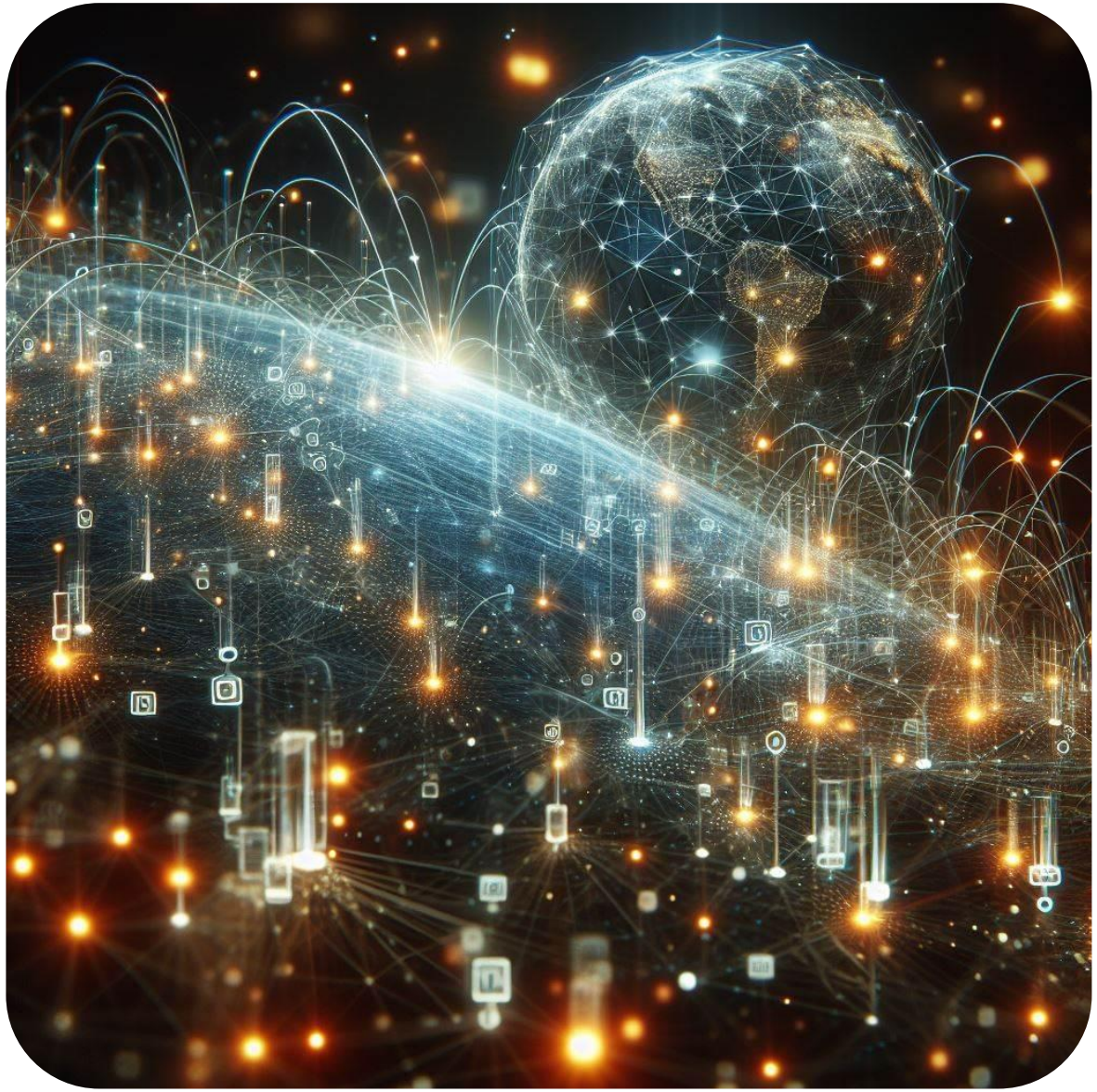
First **Digital Computer** created ~80 years ago.

First version of **the Internet** invented ~50 years ago.

BCIT™

Course Logistics

- ✓ Evaluation and course components
- ✓ Lectures and labs
- ✓ Attendance
- ✓ Academic integrity
- ✓ Learning Outcome



BCIT™

Evaluation

Criteria	%	Comments
Quiz	10	8 quizzes
Project	10	On a selected topic
Lab Assignments	25	5 assignments
Midterm Exam	20	Online
Final Exam	35	Online

BCIT™

Lectures and Labs

- ✓ All lectures and labs are **Online**.
- ✓ Lab and lecture materials are posted **Weekly on the Learning Hub**.
- ✓ Some complementary notes may be posted on the Learning Hub during the term and students have to study them.
- ✓ For quizzes, the **Quiz tool of the Learning Hub** will be used.
- ✓ **All quizzes and exams are closed-book** unless stated otherwise**.**

BCIT™

Attendance Requirements

- ✓ Regular **attendance** in **lectures** and **labs** is **mandatory** unless approved by the instructor.
- ✓ **Unapproved absence of 2 or more classes**

  may result in failure or forced withdrawal from the course or program.

- ✓ Please see Policy 5101

  Student Regulations: **http://www.bcit.ca/files/pdf/policies/5101.pdf**

BCIT™

Academic Integrity

- ✓ Violation of academic integrity, including plagiarism, dishonesty in assignments, examinations, or other academic performances are prohibited and will be handled in accordance with

  Policy 5104 - Academic Integrity and Appeals, and accompanying procedures.

- ✓ Please do not share the course materials with people outside of your class.

BCIT™

How the Internet Works?

Networking Protocols

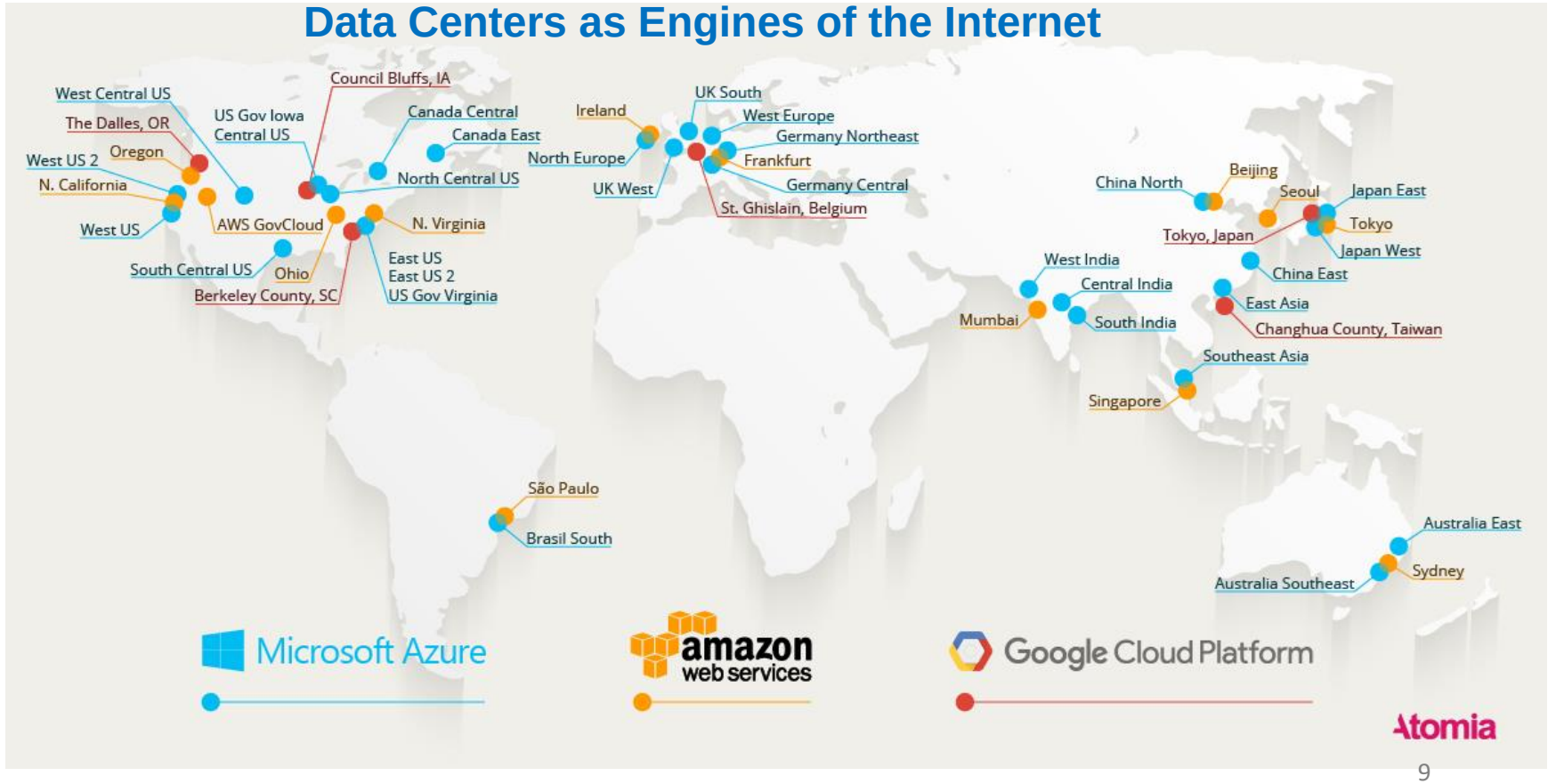
BGP
OSPF
FTP
TLS
SMTP
SSH
IMAP
IPSec
DNS
HTTPs
SIP
SNMP

Networking/Computing Devices

Servers
Switch
VMs
Containers
Router
Storage
Firewalls
Load Balancer
UTM
WAF
VPN

Internet Applications/Sites

Data Centers as Engines of the Internet



What is DevOps/SRE?

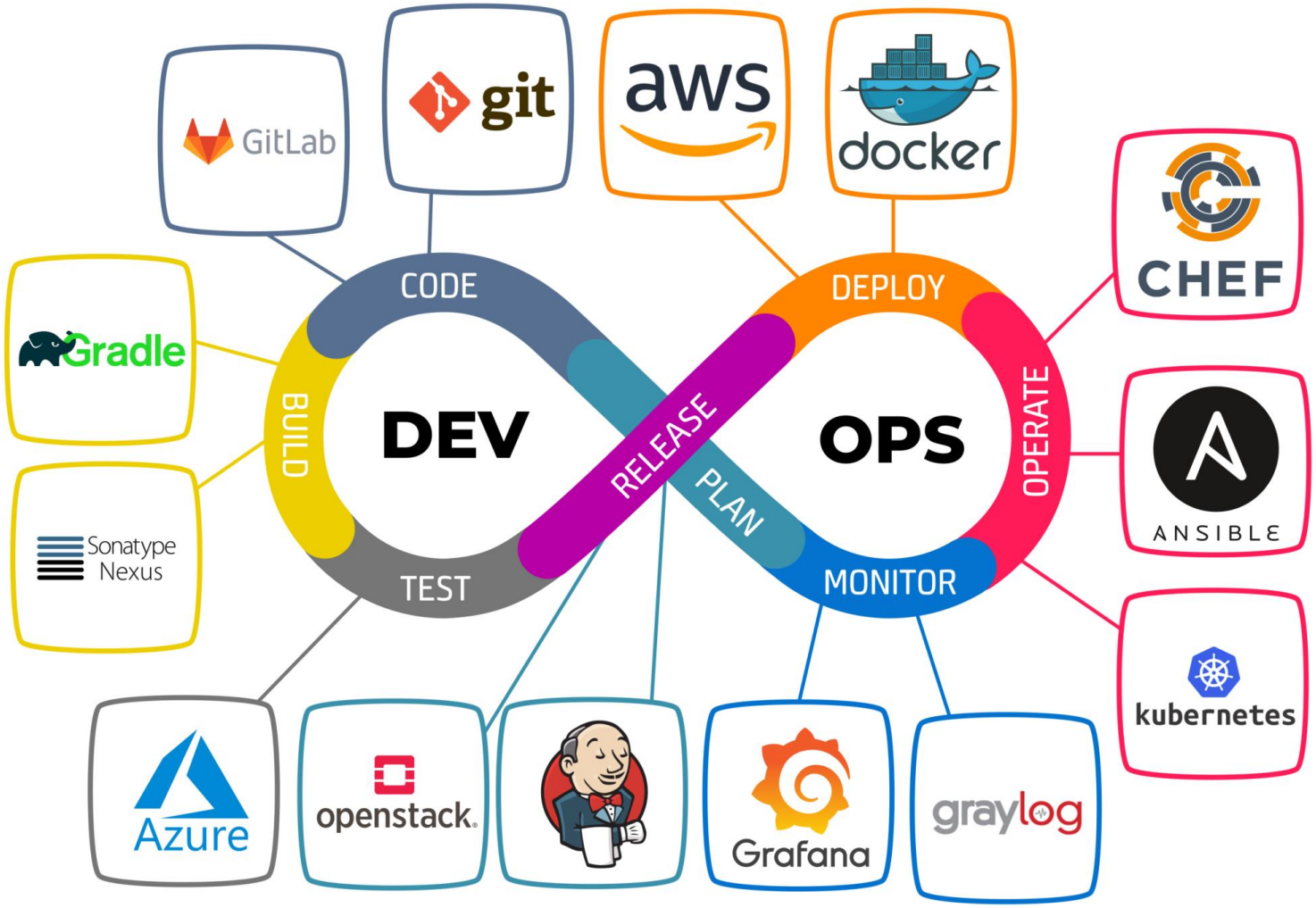
Speed

Software Developers — DEV

Stability

IT/System Operators — PROD

A **DevOps Engineer** bridges the gap between software development and IT operations to make software delivery faster, more reliable, and efficient.



What You Learn From This Course Will Help You to …

Devops in Canada
7,272 results
Job Alert Off

DevOps
OneSpan
Montreal, QC
1 company alum works here
Promoted · 18 applicants

Azure Devops, Infrastructure and Security Specialist
GS1 Canada
Toronto, ON
1 alum works here
Promoted · 5 applicants

DevOps Engineer
Avanade
Vancouver, BC
4 alumni work here
Promoted · 4 applicants

Cloud Engineer, Elastic Engineering
Onica, a Rackspace Technology Company
Vancouver, BC
2 alumni work here
4 weeks ago · 3 applicants

DevOps Engineering Director - K8s and Python
Canonical
Vancouver, BC Remote
1 connection works here
1 month ago · 0 applicants

Intermediate DevOps/SRE Engineer (Linux)
Global Relay
Vancouver, BC

Devops in United States
72,023 results
Job Alert Off

Senior Staff DevOps Engineer (Information Security)
Palo Alto Networks
Santa Clara, CA
2 alumni work here
Promoted · 2 applicants · Easy Apply

DevOps, Infrastructure Manager
Optello
Boise, ID
Medical, Vision, Dental, 401(k)
Actively recruiting
Promoted · 3 applicants · Easy Apply

Principal Consultant - DevOps - Nationwide Opportunities
Amazon Web Services (AWS)
San Francisco, CA
Medical benefit
6 connections work here
Promoted · 0 applicants

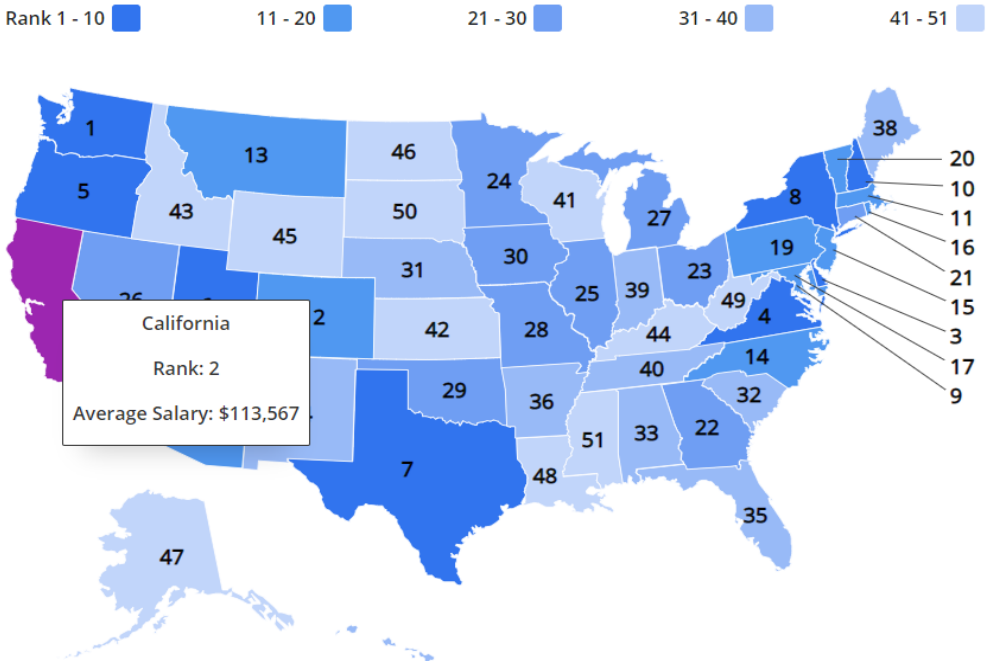
Security Infrastructure DevOps Engineer
Apple
Cupertino, CA
2 connections work here
3 hours ago

Platform Services QA Lead and DevOps Engineer
Viasat Inc.
Remote, OR
1 alum works here
2 weeks ago · 0 applicants

Senior CI/CD DevOps Engineer
Microsoft

DevOps/SRE

BEST STATES FOR A DEVOPS ENGINEER



Learning Outcomes of the course

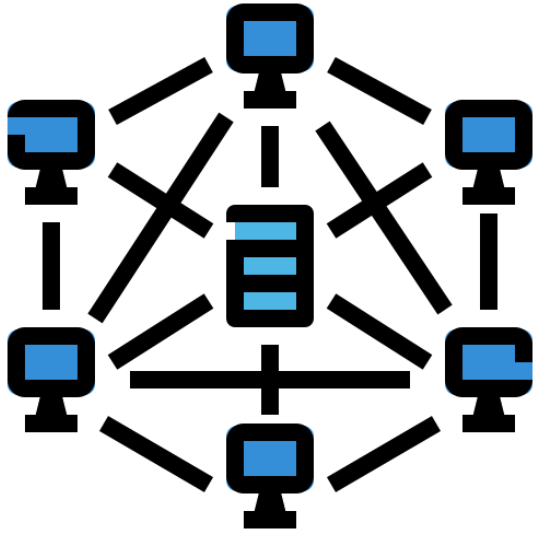
- ✓ Describe the different types of **Hypervisors** and Cloud technologies.
- ✓ Determine environment load potential and the methods required to **scale**.
- ✓ Describe the different technologies and protocols used for **network** and **storage virtualization**.
- ✓ Identify common **performance** bottlenecks and different ways to detect and prevent them.
- ✓ Explain fundamental concepts surrounding **High availability** (HA) and **Disaster Recovery** (DR) in a virtual environment.
- ✓ Describe a **backup strategy** for a virtual environment.
- ✓ Explain various monitoring protocols (**SNMP** and **syslog**) and tools.
- ✓ Describe fundamental concepts in **Public and Private Cloud** environments.
- ✓ Conduct a performance assessment combined with a review of **security baselines** in cloud environments.
- ✓ Describe **container management and orchestration** for cloud-based network solutions.
- ✓ Automate **resource provisioning** process in virtual environments.
- ✓ Explain the **Infrastructure-as-a-Code (IaaC)** concept for real-world applications.

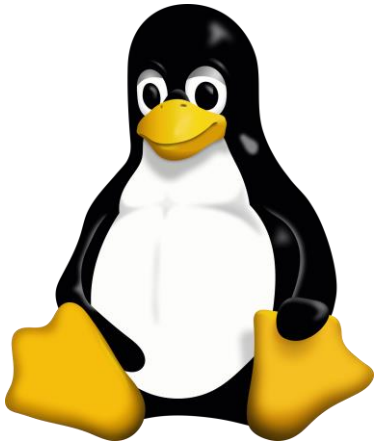
Learning Outcomes of Week #1

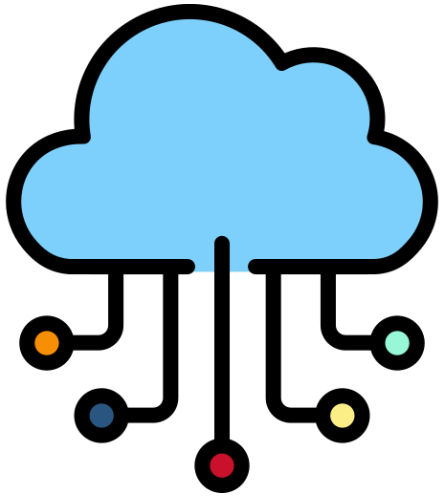
1. Describe how Internet works?
2. Explain how big companies serve **Billions** of people on the Internet.
3. Describe the general concept of Virtualization on the Internet.
4. Understand the Concept of Cloud Computing/Technology.
5. Explain different types of Cloud environments/architecture.

Quick Recap on Requirements of this Course

1. Basics of Computer Networking
2. Basics of Linux Operating System
3. Cloud Computing Platforms



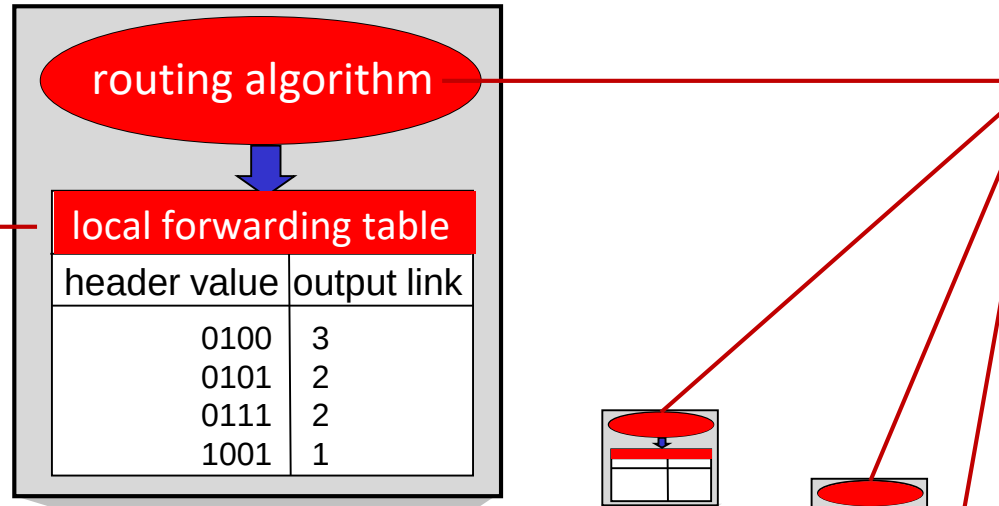


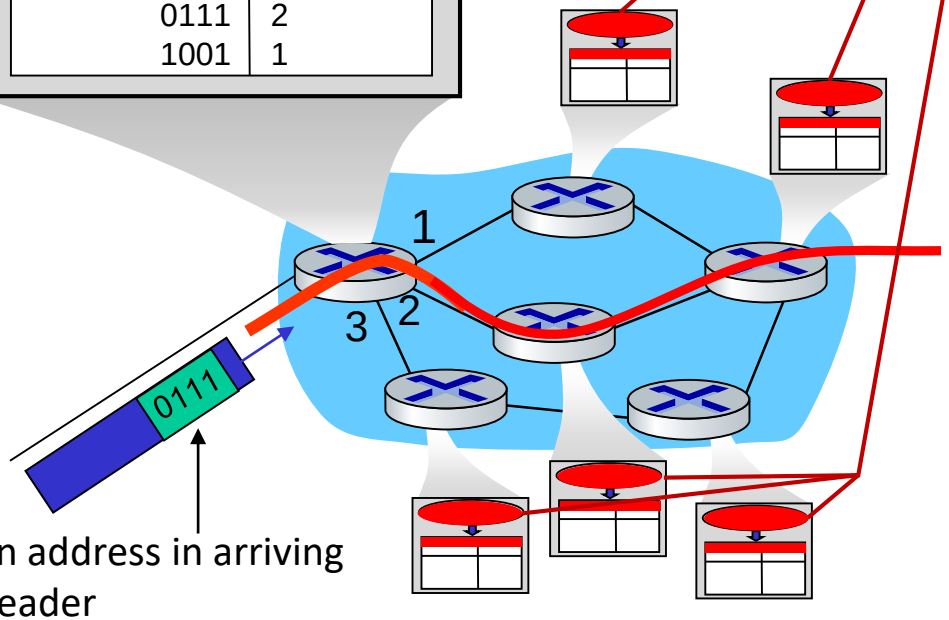


Two key network-core functions

Forwarding:

- aka "switching"
- *local* action: move arriving packets from router's input link to appropriate router output link





Routing:

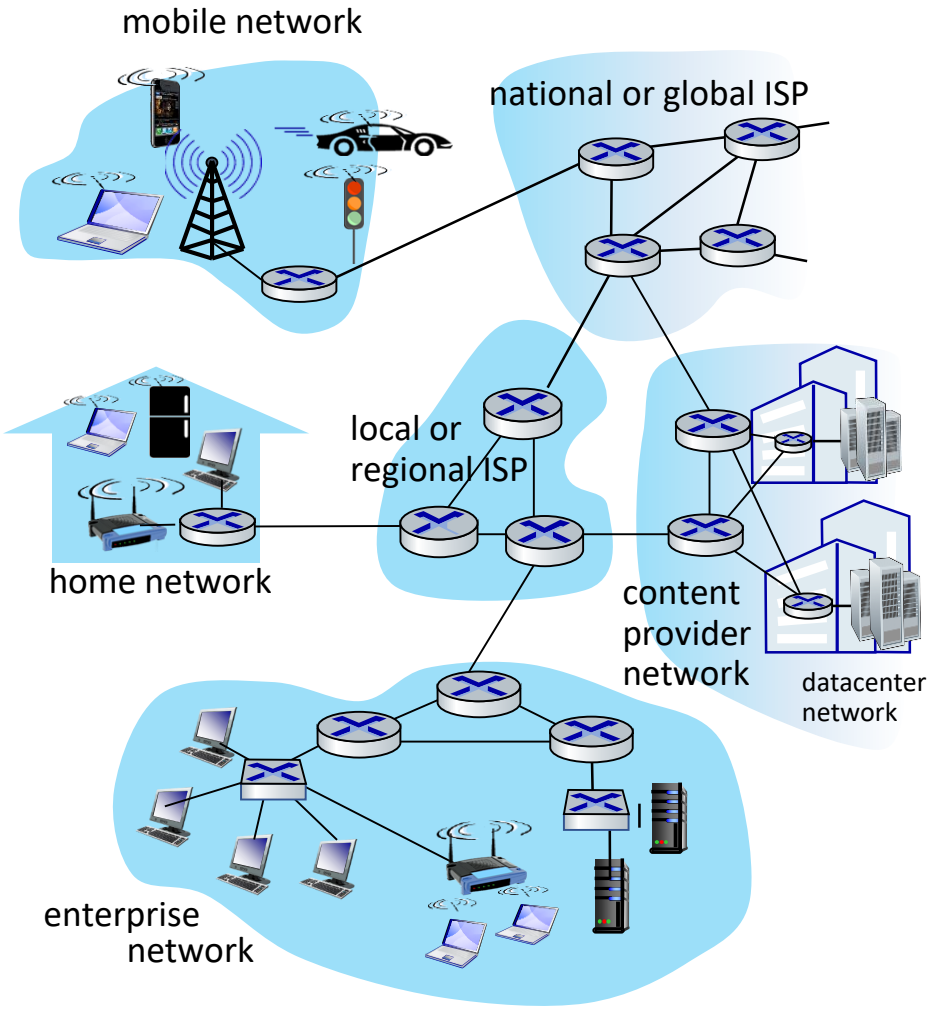
- *global* action: determine source-destination paths taken by packets
- routing algorithms


routing
45 h
3,053 miles
48 h
3,309 miles
United States
Victoria
Seattle
Portland
San Francisco
San Jose
Sacramento
Las Vegas
Los Angeles
San Diego
Phoenix
Tucson
El Paso
Dallas
Austin
Houston
San Antonio
New Orleans
Kansas City
Minneapolis
Chicago
Detroit
Toronto
Ottawa
Montreal
Quebec City
Northampton
New York
Philadelphia
Indianapolis
Nashville
Charlotte
Atlanta
Jacksonville
Orlando
Tampa
WASHINGTON
OREGON
CALIFORNIA
NEVADA
IDAHO
MONTANA
WYOMING
UTAH
COLORADO
ARIZONA
NEW MEXICO
NORTH DAKOTA
SOUTH DAKOTA
MINNESOTA
IOWA
WISCONSIN
MICHIGAN
ILLINOIS
INDIANA
OHIO
KENTUCKY
TENNESSEE
MISSISSIPPI
ALABAMA
GEORGIA
FLORIDA
SOUTH CAROLINA
NORTH CAROLINA
VIRGINIA
WEST VIRGINIA
PENNSYLVANIA
MARYLAND
DE
NJ
CT
RI
VERMONT
NEW HAMPSHIRE
MAINE
OKLAHOMA
TEXAS
ARKANSAS
LOUISIANA
BAJA CALIFORNIA
SONORA
CHIHUAHUA
COAHUILA

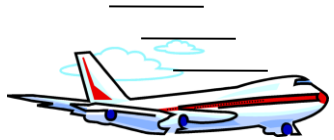

forwarding
forwarding
45 h
3,053 miles
48 h
3,309 miles
United States
Victoria
Seattle
Portland
WASHINGTON
NORTH DAKOTA
MINNESOTA
SOUTH DAKOTA
Minneapolis
WISCONSIN
MICHIGAN
Toronto
Quebec City
Montreal
Ottawa
MAINE
VERMONT
NEW HAMPSHIRE
Northampton
WYOMING
IOWA
Chicago
Detroit
ILLINOIS
INDIANA
OHIO
Kansas City
Indianapolis
COLORADO
KANSAS
MISSOURI
KENTUCKY
Nashville
TENNESSEE
San Francisco
San Jose
CALIFORNIA
Las Vegas
OKLAHOMA
ARKANSAS
Atlanta
Los Angeles
ARIZONA
NEW MEXICO
MISSISSIPPI
ALABAMA
San Diego
Phoenix
Dallas
Tucson
El Paso
TEXAS
BAJA CALIFORNIA
SONORA
Austin
Houston
LOUISIANA
New Orleans
CHIHUAHUA
San Antonio
COAHUILA
Tampa
FLORIDA
Cleveland
DOWNTOWN
CENTRAL
OHIO CITY

1 Breakout Rooms

1. Visit **ping.pe** web site
2. Enter **Google.com** in the box
3. Check the 2nd column from the right (**show**) for different Geo locations



Example: organization of air travel



end-to-end transfer of person plus baggage →

ticket (purchase)	ticket (complain)
baggage (check)	baggage (claim)
gates (load)	gates (unload)
runway takeoff	runway landing
airplane routing	airplane routing

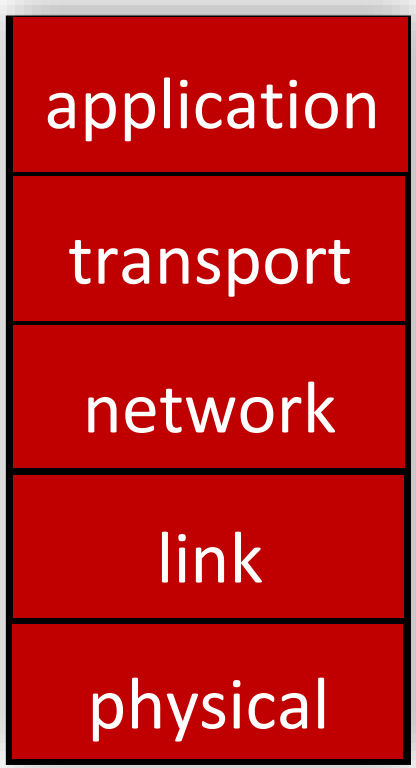
airplane routing

How would you *define/discuss* the *system* of airline travel?

- a series of steps, involving many services

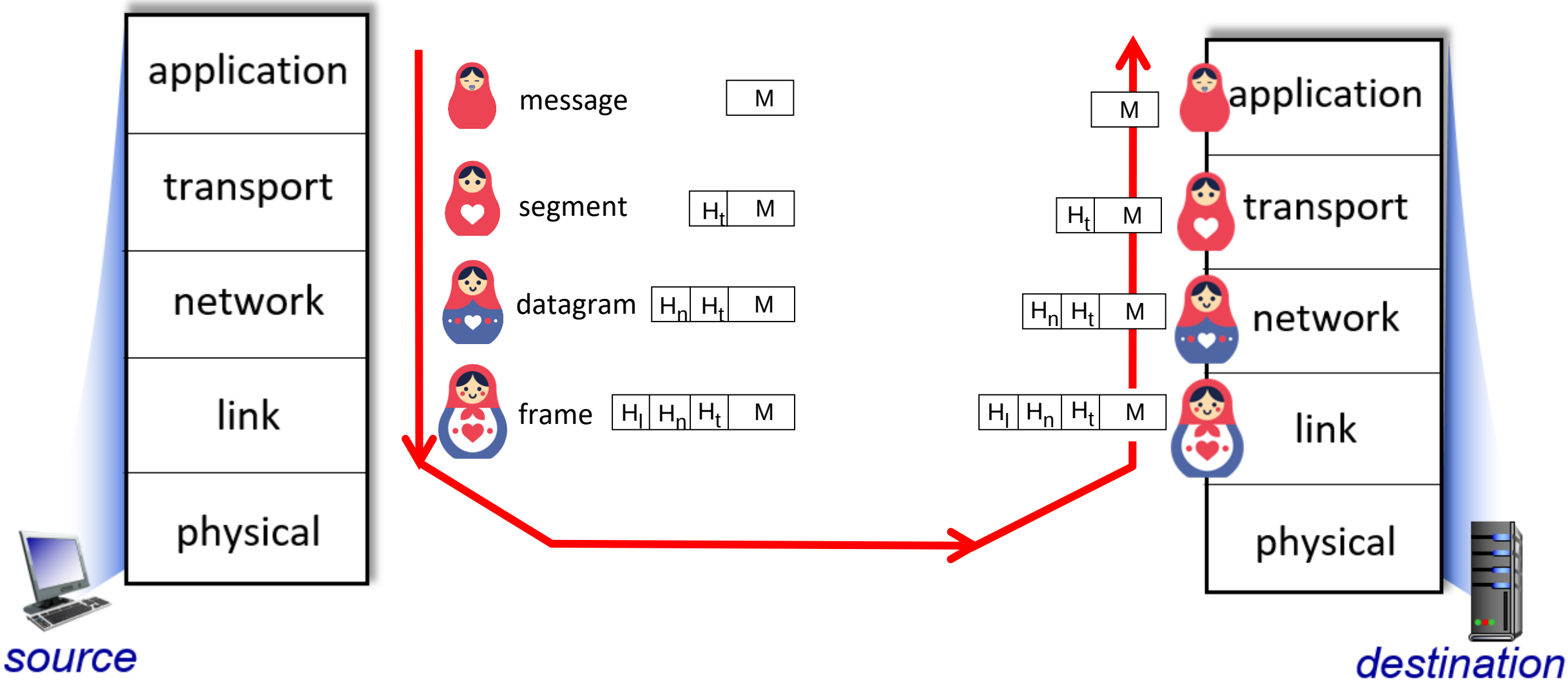
Layered Internet protocol stack

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



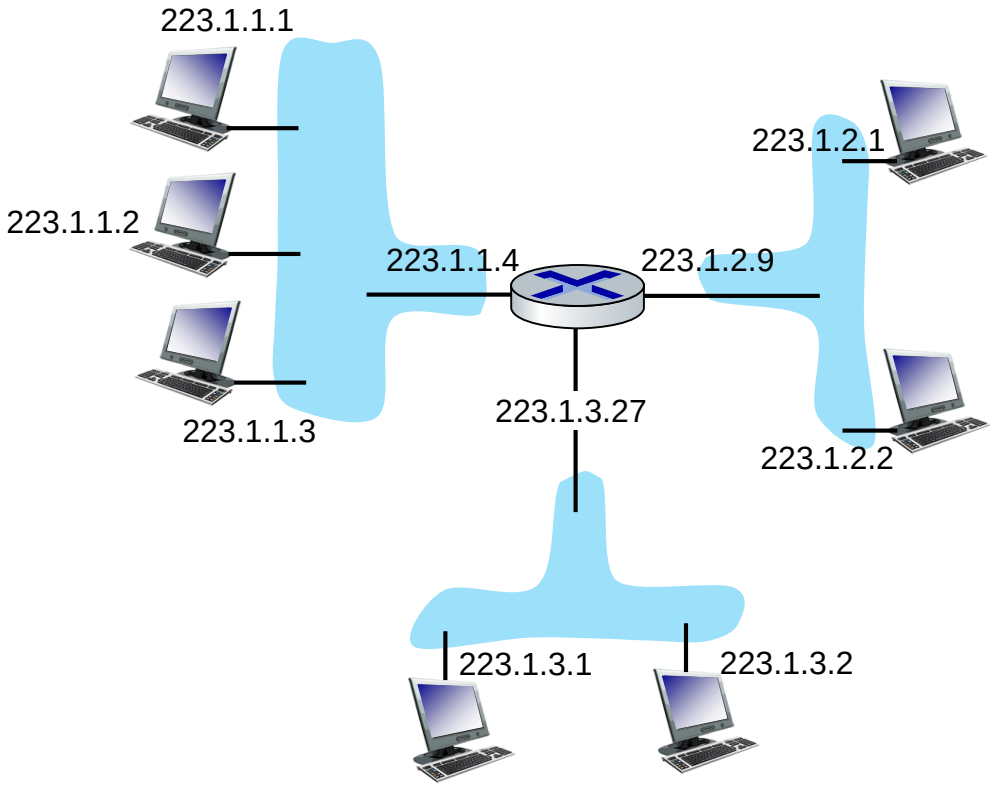
Five-Layer Internet Protocol Stack

Services, Layering and Encapsulation



IP addressing: introduction

- IP address: 32-bit identifier associated with each host or router *interface*
- interface: connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)



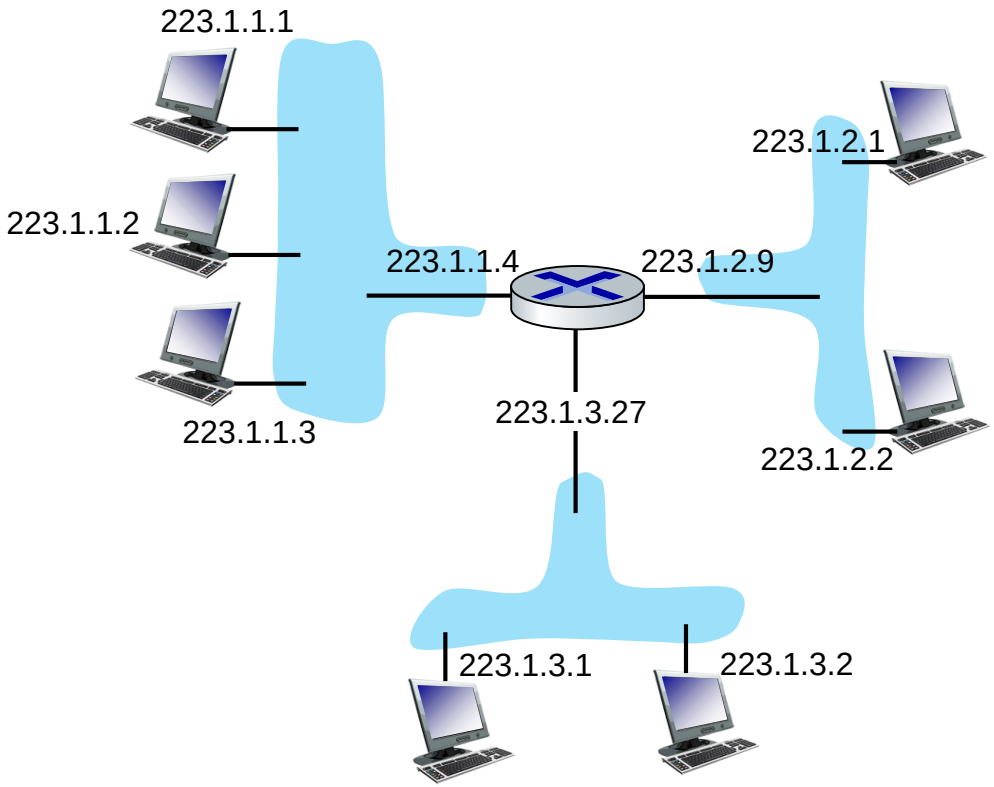
dotted-decimal IP address notation:

223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

IP addressing: introduction

- IP address: 32-bit identifier associated with each host or router *interface*
- interface: connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)



dotted-decimal IP address notation:

223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

IP addressing: CIDR

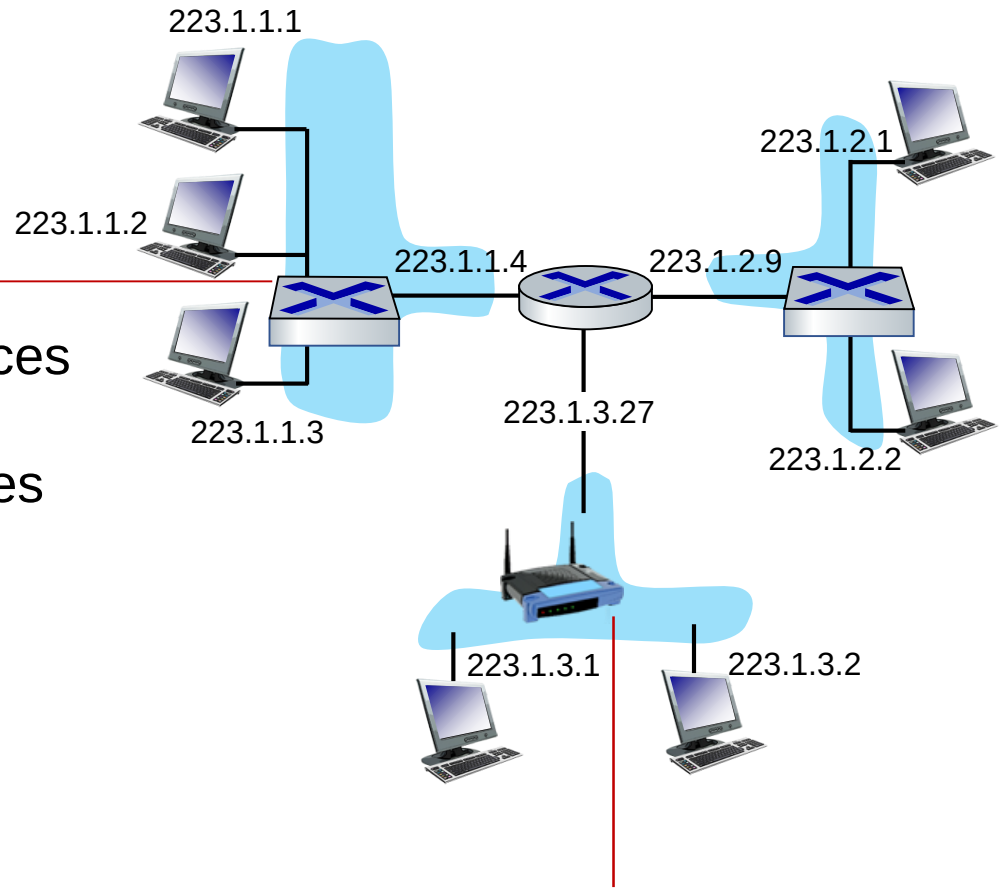
CIDR: Classless InterDomain Routing (pronounced “cider”)

- subnet portion of address of arbitrary length
- Variable Length Subnet Masks (VLSM)
- address format: a.b.c.d/x, where x is # bits in subnet portion of address

200.23.16.0/23

IP addressing: introduction

Q: how are interfaces actually connected?

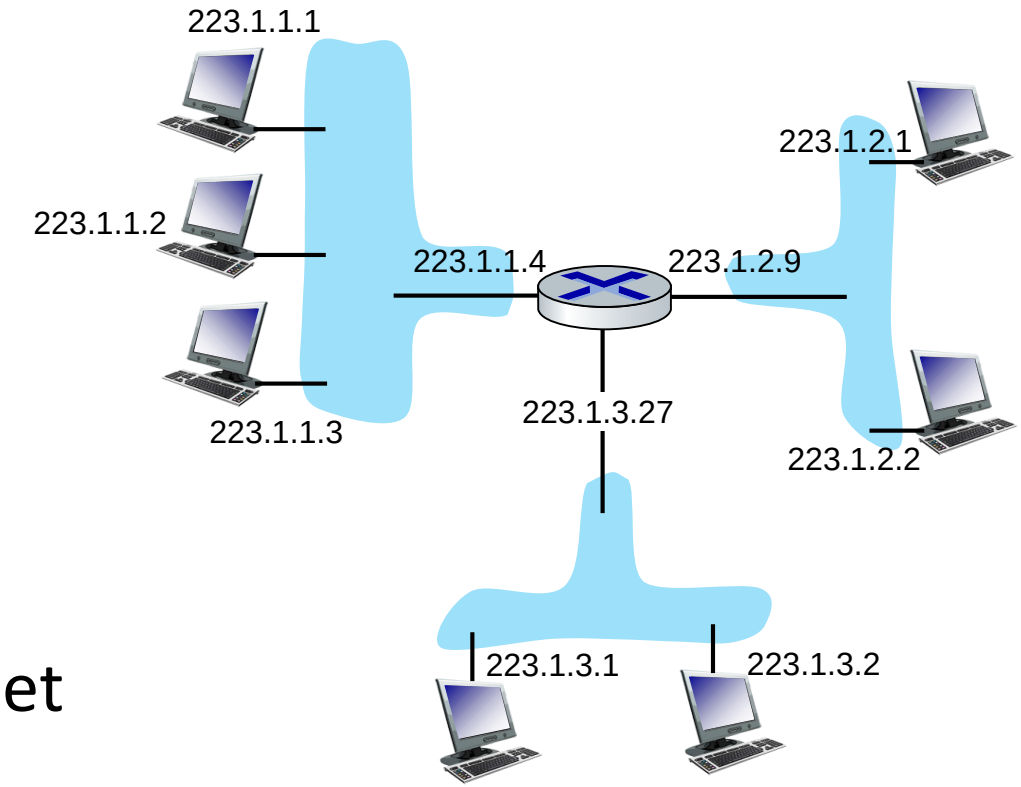


A: wired Ethernet interfaces connected by Ethernet switches

A: wireless WiFi interfaces connected by WiFi base station

Subnets

- *What's a subnet ?*
  - device interfaces that can physically reach each other without passing through an intervening router
- IP addresses have structure:
  - subnet part: devices in same subnet have common high order bits
  - host part: remaining low order bits

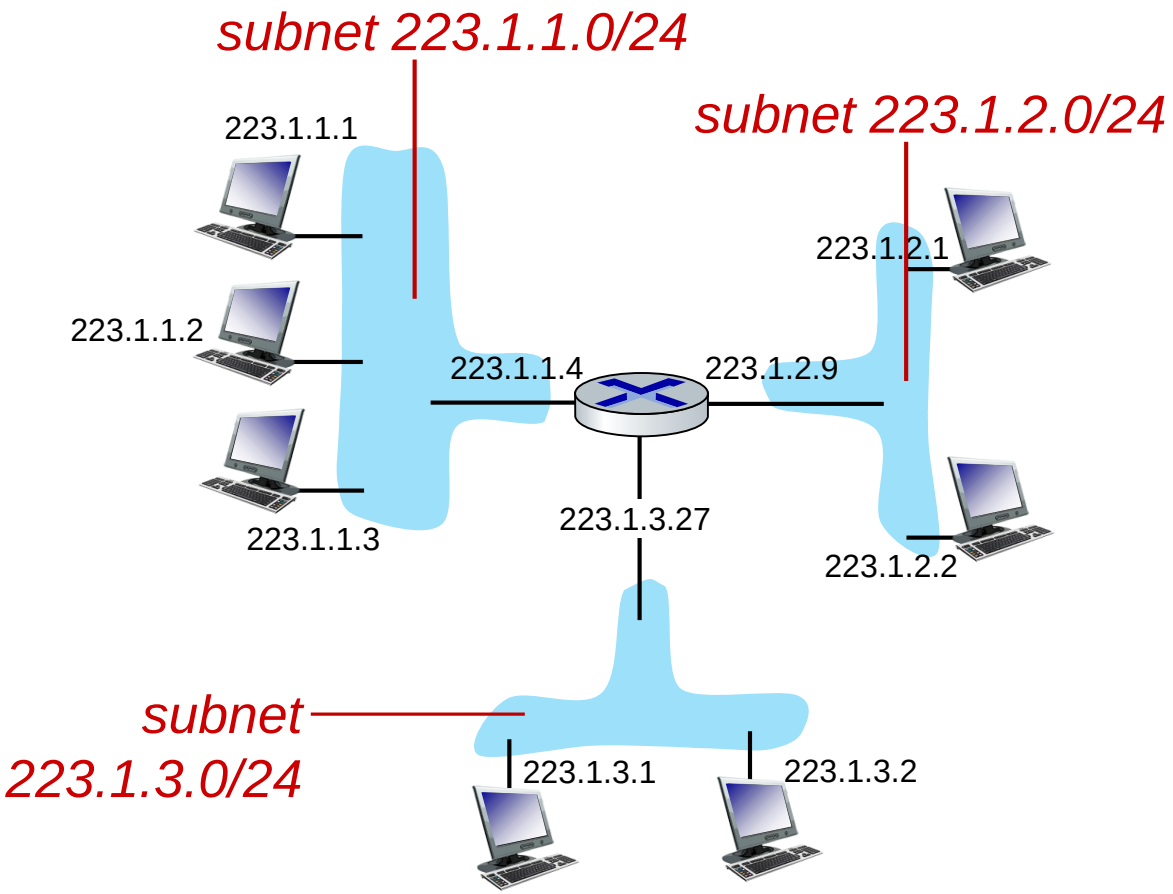


network consisting of 3 subnets

Subnets

Recipe for defining subnets:

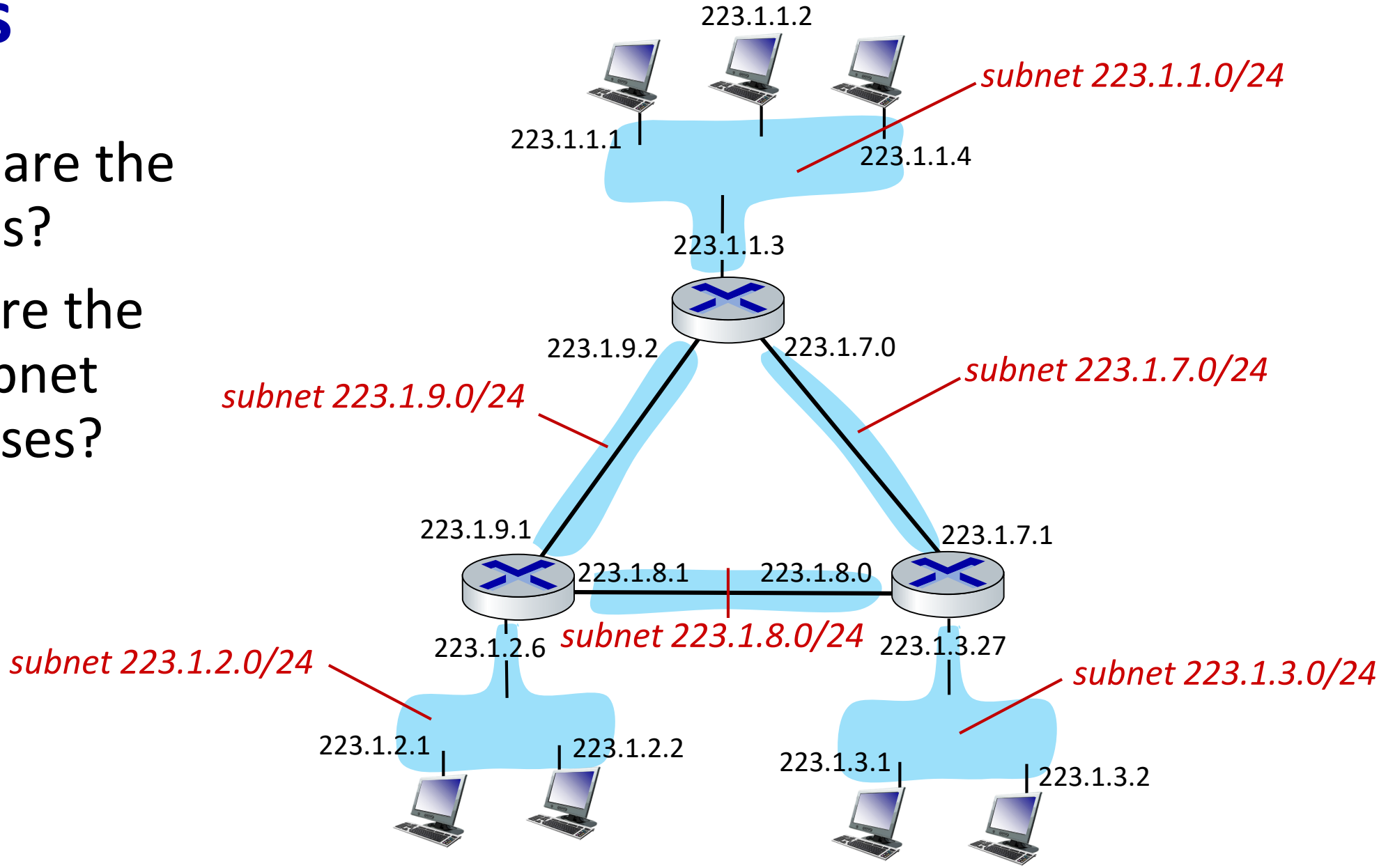
- detach each interface from its host or router, creating "islands" of isolated networks
- each isolated network is called a *subnet*



subnet mask: /24
(high-order 24 bits: subnet part of IP address)

Subnets

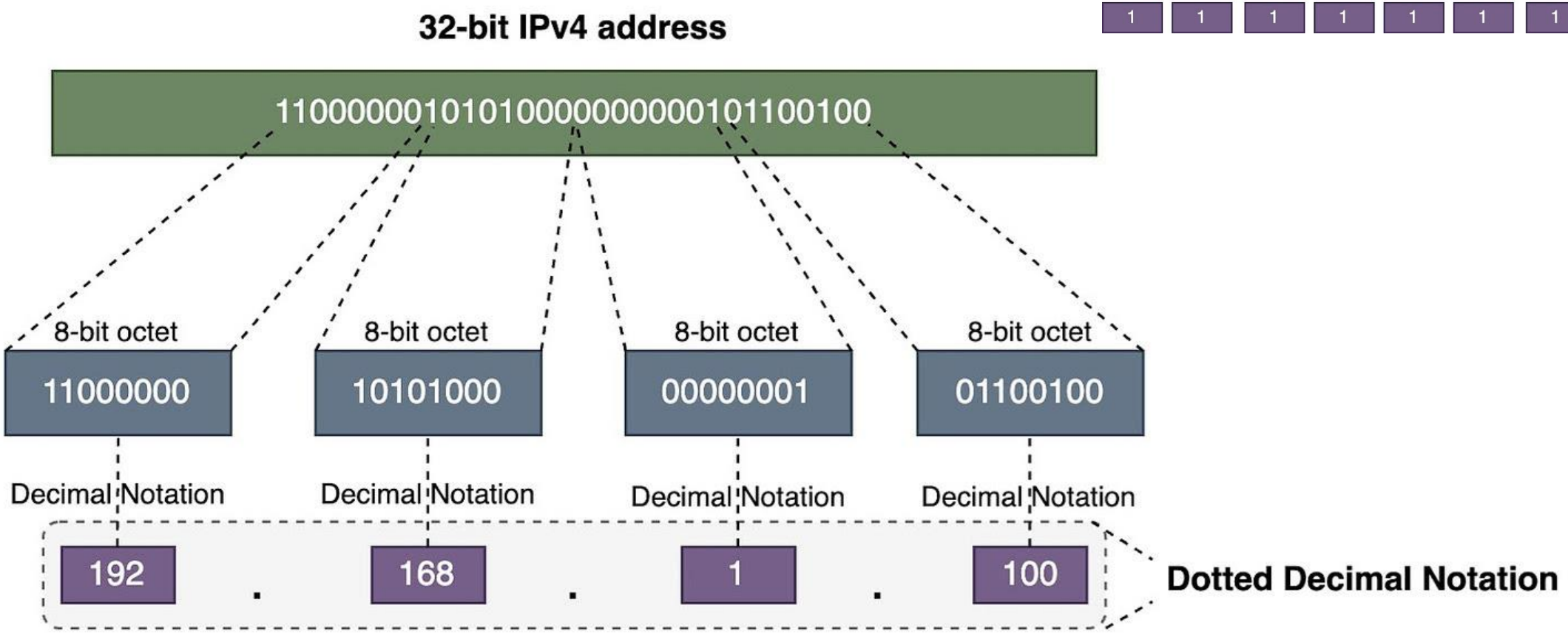
- where are the subnets?
- what are the /24 subnet addresses?



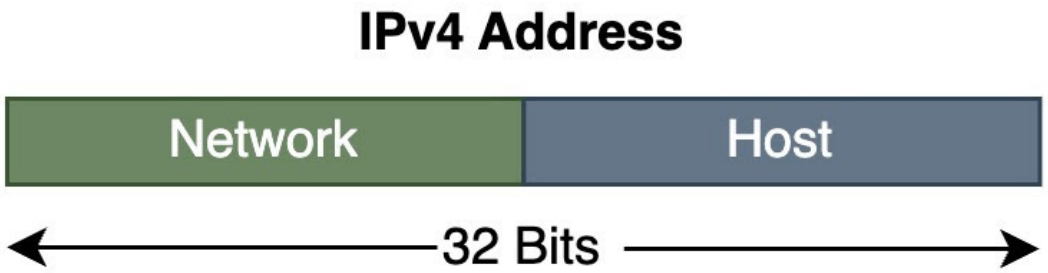
IP addressing: introduction

Valid Subnet Mask Values

128	64	32	16	8	4	2	1	Decimal Value
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	128
1	1	0	0	0	0	0	0	192
1	1	1	0	0	0	0	0	224
1	1	1	1	0	0	0	0	240
1	1	1	1	1	0	0	0	248
1	1	1	1	1	1	0	0	252
1	1	1	1	1	1	1	0	254
1	1	1	1	1	1	1	1	255

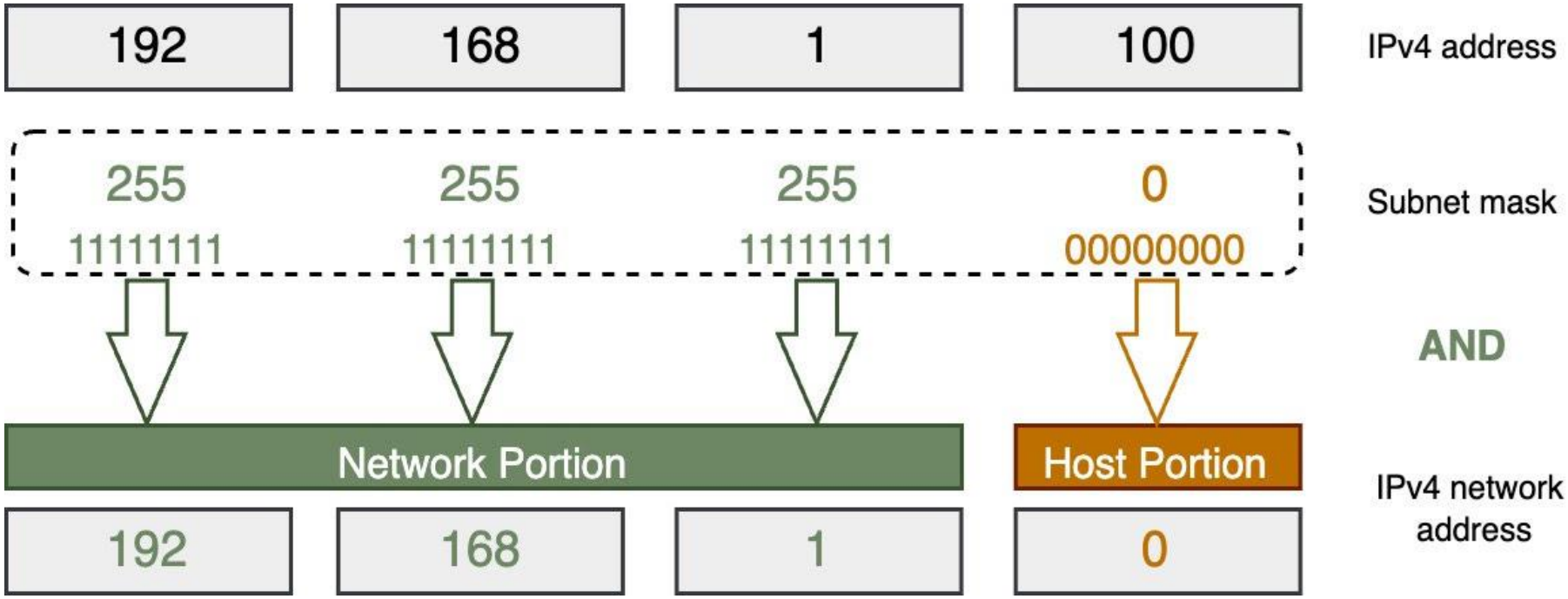


IP addressing: introduction

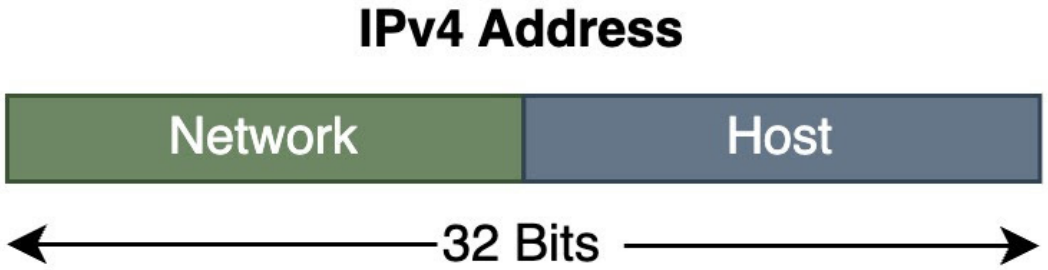


Valid Subnet Mask Values

128	64	32	16	8	4	2	1	Decimal Value
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	128
1	1	0	0	0	0	0	0	192
1	1	1	0	0	0	0	0	224
1	1	1	1	0	0	0	0	240
1	1	1	1	1	0	0	0	248
1	1	1	1	1	1	0	0	252
1	1	1	1	1	1	1	0	254
1	1	1	1	1	1	1	1	255

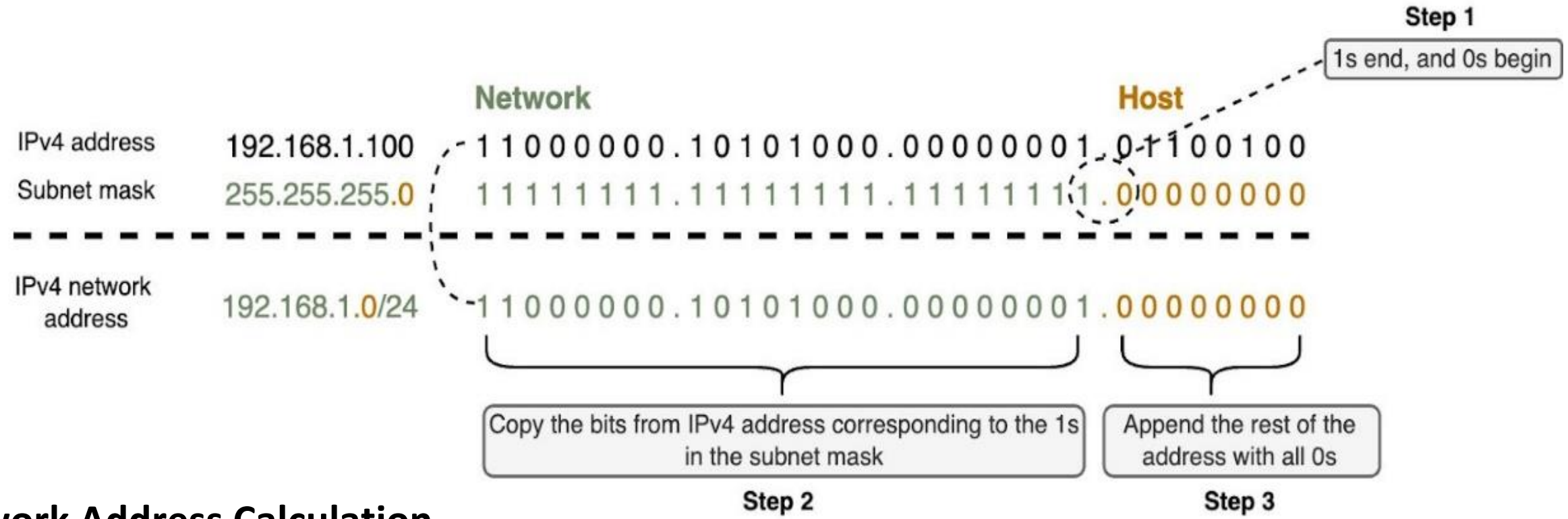


IP addressing: introduction



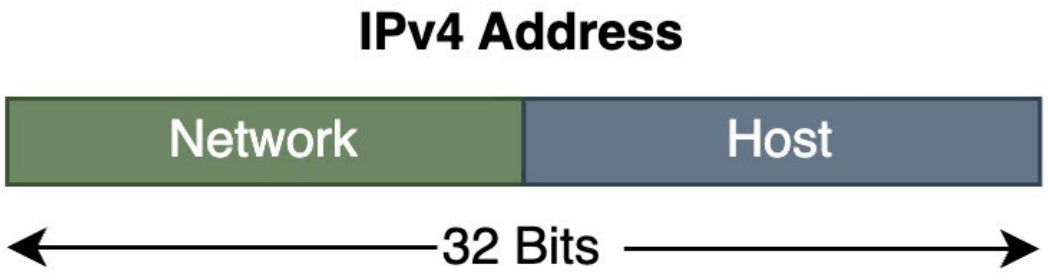
Valid Subnet Mask Values

128	64	32	16	8	4	2	1	Decimal Value
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	128
1	1	0	0	0	0	0	0	192
1	1	1	0	0	0	0	0	224
1	1	1	1	0	0	0	0	240
1	1	1	1	1	0	0	0	248
1	1	1	1	1	1	0	0	252
1	1	1	1	1	1	1	0	254
1	1	1	1	1	1	1	1	255



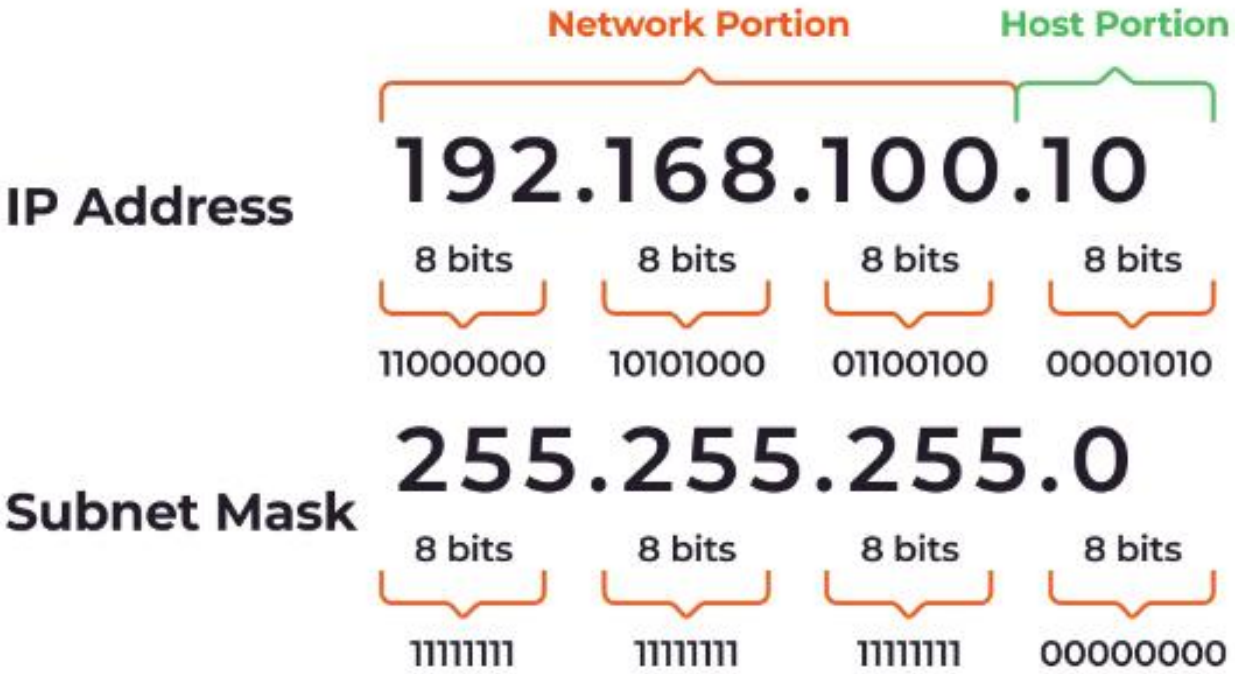
Network Address Calculation

IP addressing: introduction



Valid Subnet Mask Values

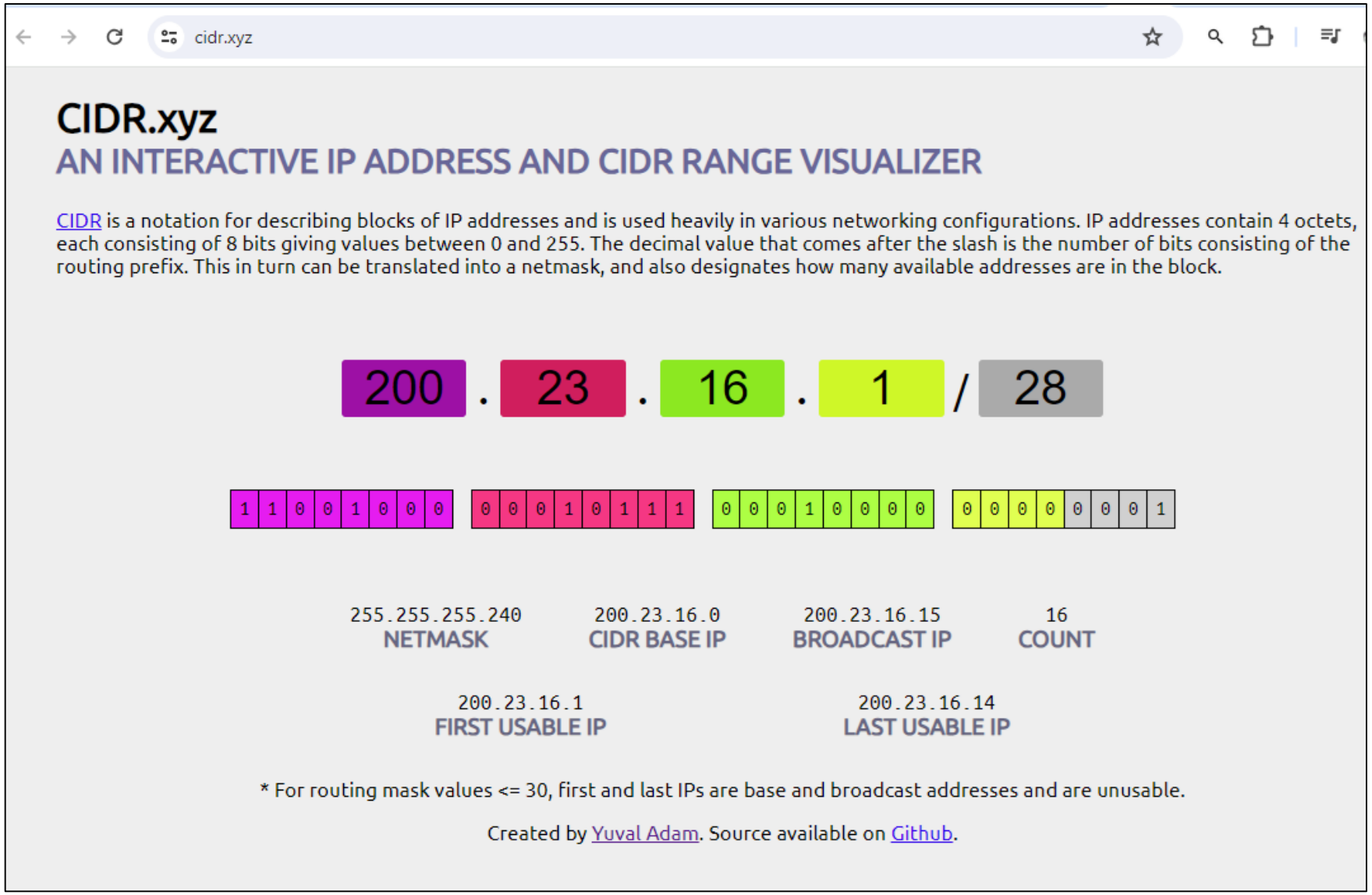
128	64	32	16	8	4	2	1	Decimal Value
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	128
1	1	0	0	0	0	0	0	192
1	1	1	0	0	0	0	0	224
1	1	1	1	0	0	0	0	240
1	1	1	1	1	0	0	0	248
1	1	1	1	1	1	0	0	252
1	1	1	1	1	1	1	0	254
1	1	1	1	1	1	1	1	255



Network IP → 192.168.100.0
Broadcast IP →192.168.100.255

First Useable IP→ 192.168.100.1
Last Useable IP →192.168.100.254

IP addressing: introduction



https://cidr.xyz

IP addresses: how to get one?

That's actually two questions:

1. Q: How does a *host* get IP address within its network (host part of address)?
2. Q: How does a *network* get IP address for itself (network part of address)

How does *host* get IP address?

- hard-coded by sysadmin in config file (e.g., /etc/rc.config in UNIX)
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

DHCP: Dynamic Host Configuration Protocol

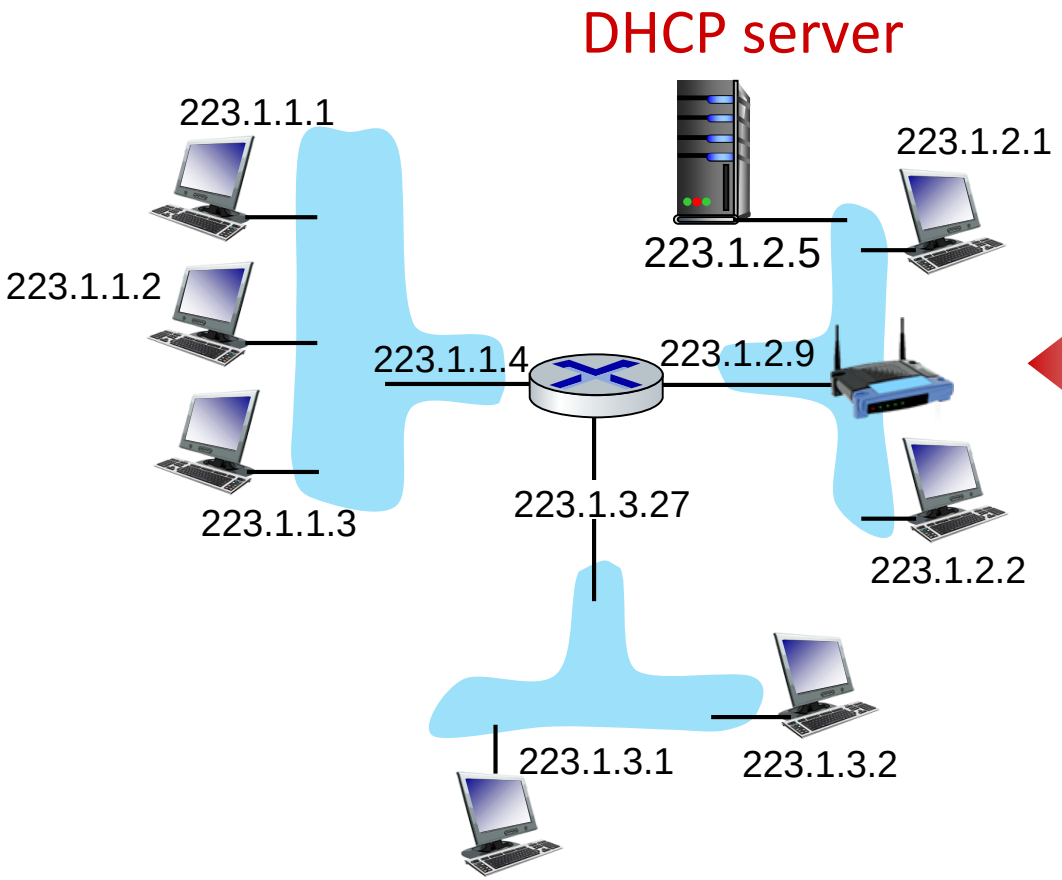
goal: host *dynamically* obtains IP address from network server when it "joins" network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/on)
- support for mobile users who join/leave network

DHCP overview:

- host broadcasts DHCP discover msg [optional]
- DHCP server responds with DHCP offer msg [optional]
- host requests IP address: DHCP request msg
- DHCP server sends address: DHCP ack msg

DHCP client-server scenario



Typically, DHCP server will be co-located in router, serving all subnets to which router is attached

arriving DHCP client needs address in this network

MAC addresses

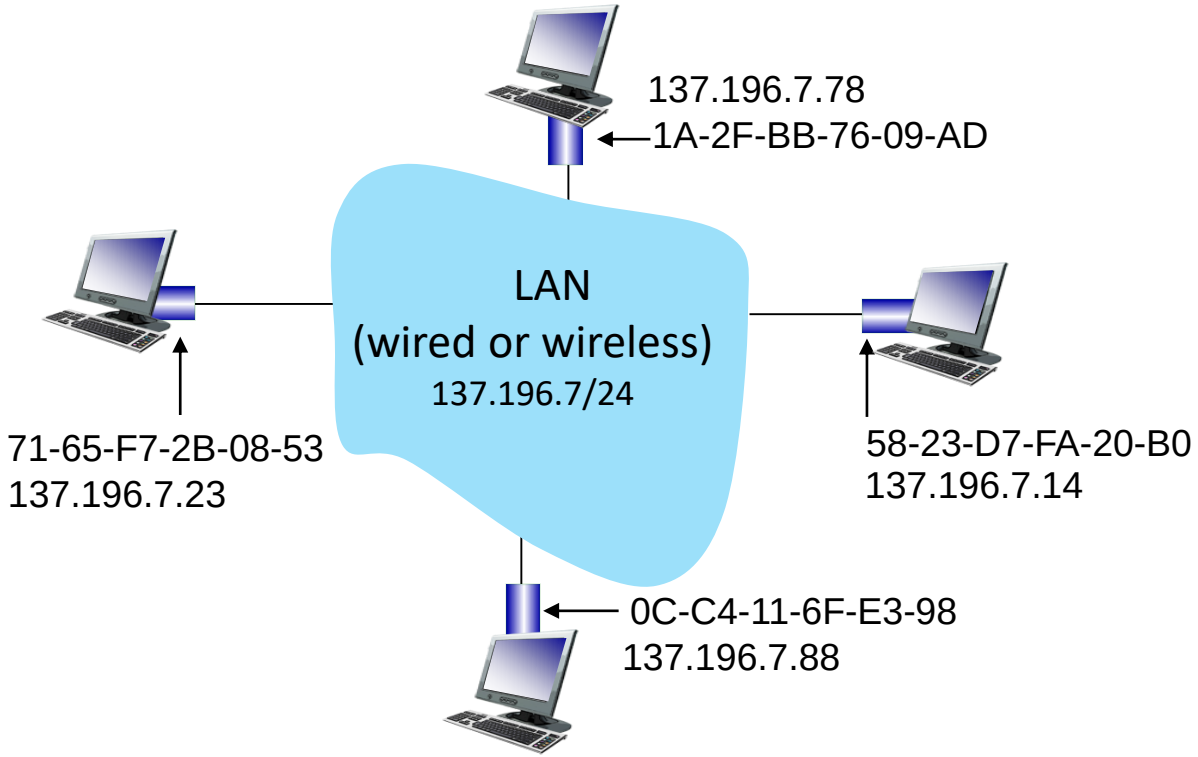
- 32-bit IP address:
  - *network-layer* address for interface
  - used for layer 3 (network layer) forwarding
  - e.g.: 128.119.40.136
- MAC (or LAN or physical or Ethernet) address:
  - function: used "locally" to get frame from one interface to another physically-connected interface (same subnet, 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)

MAC addresses

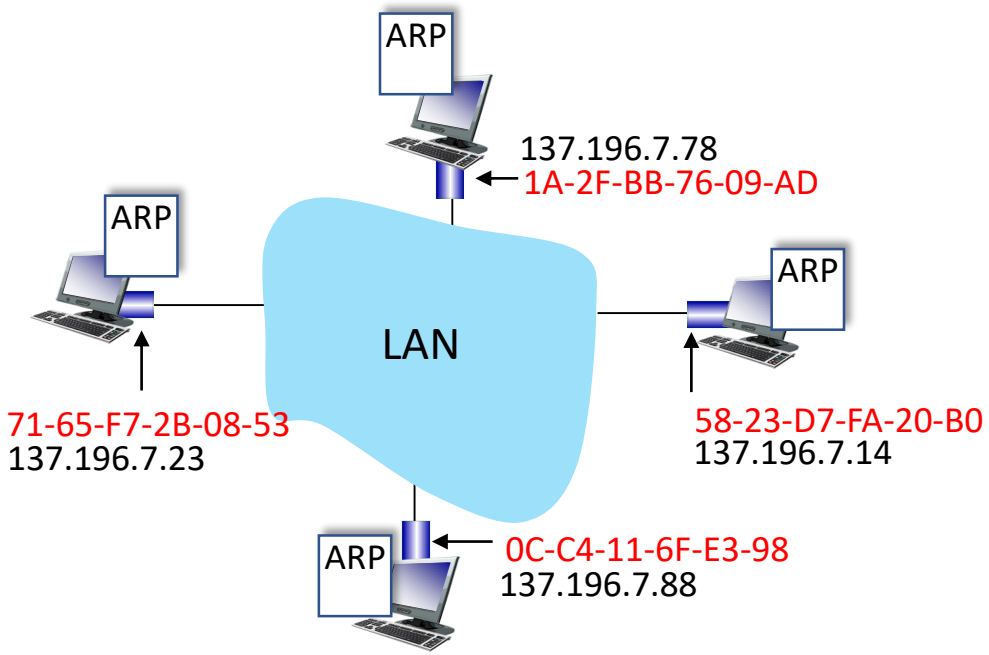
each interface on LAN

- has unique 48-bit MAC address
- has a locally unique 32-bit IP address (as we've seen)



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 in action

example: A wants to send datagram to B

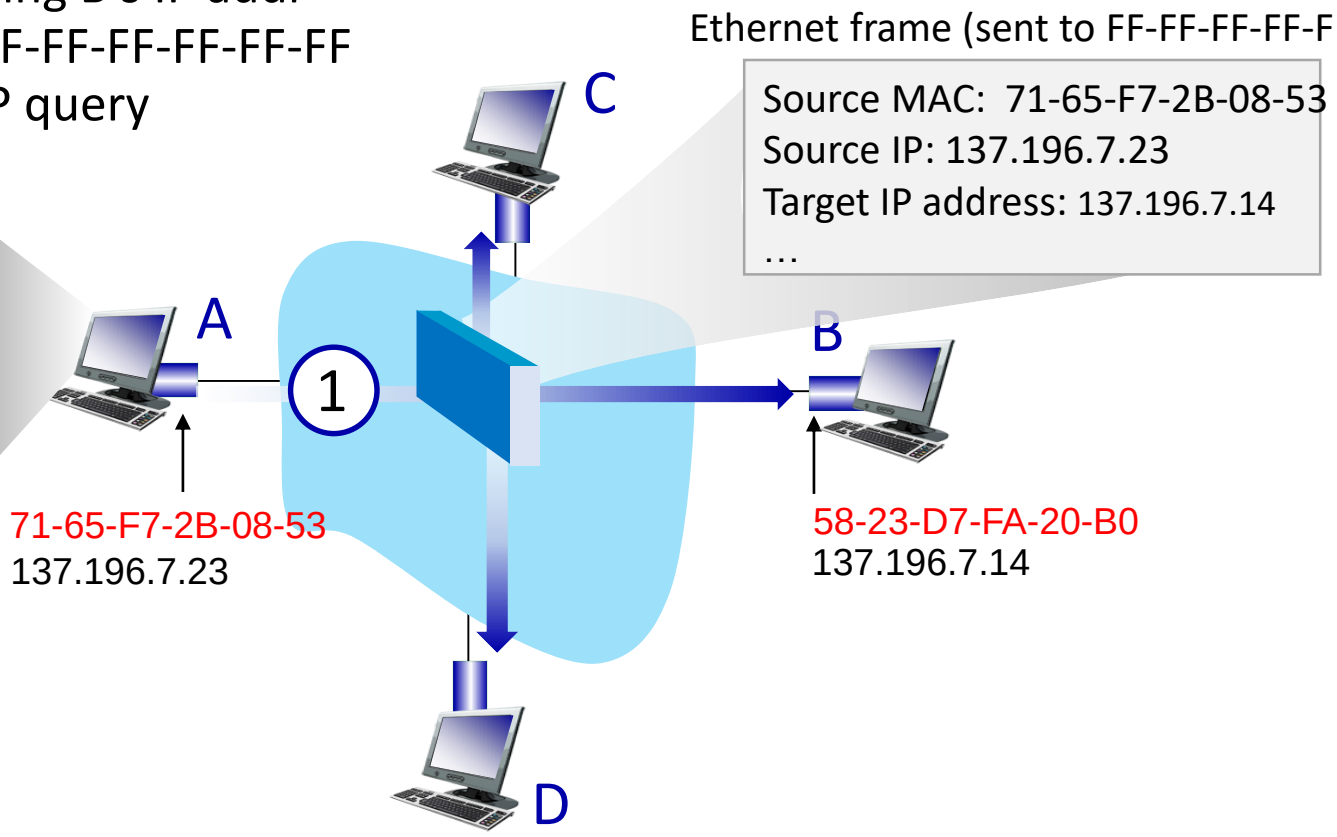
- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address

A broadcasts ARP query, containing B's IP addr

1 • destination MAC address = FF-FF-FF-FF-FF-FF
• all nodes on LAN receive ARP query

ARP table in A

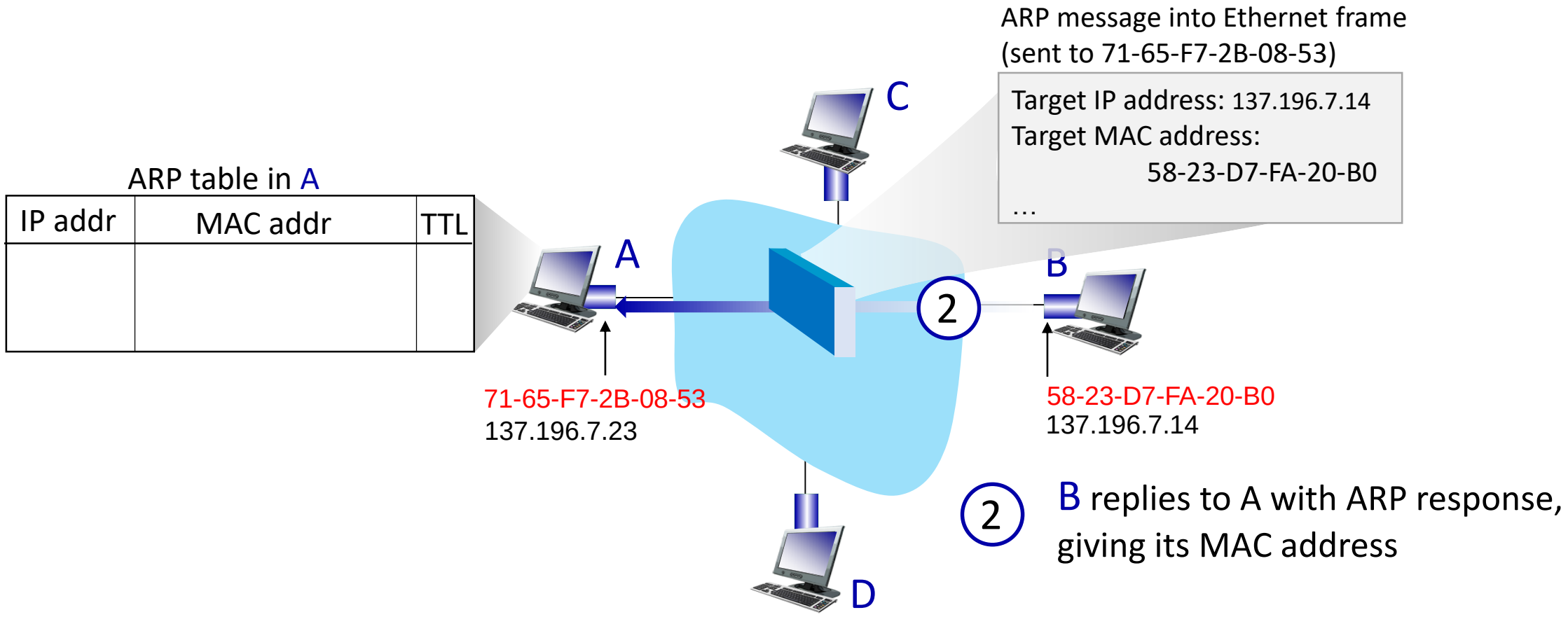
IP addr	MAC addr	TTL



ARP protocol in action

example: A wants to send datagram to B

- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address



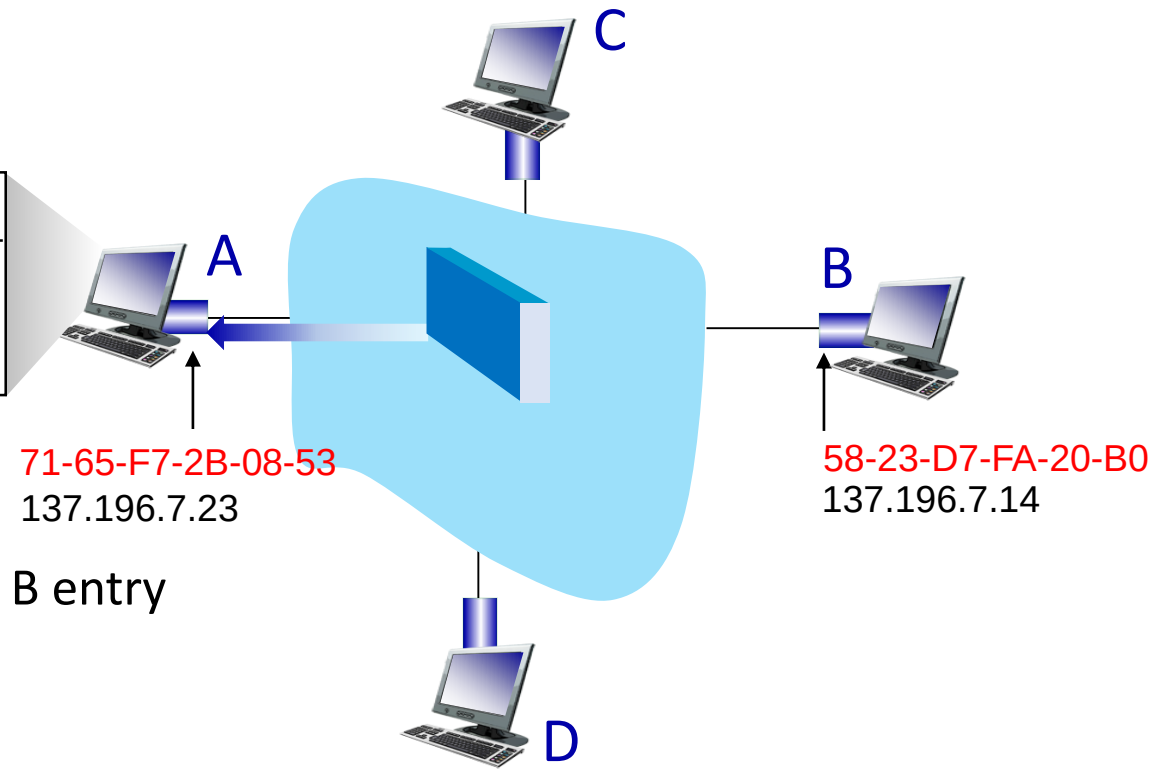
ARP protocol in action

example: A wants to send datagram to B

- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address

ARP table in A

IP addr	MAC addr	TTL
137.196.7.14	58-23-D7-FA-20-B0	500

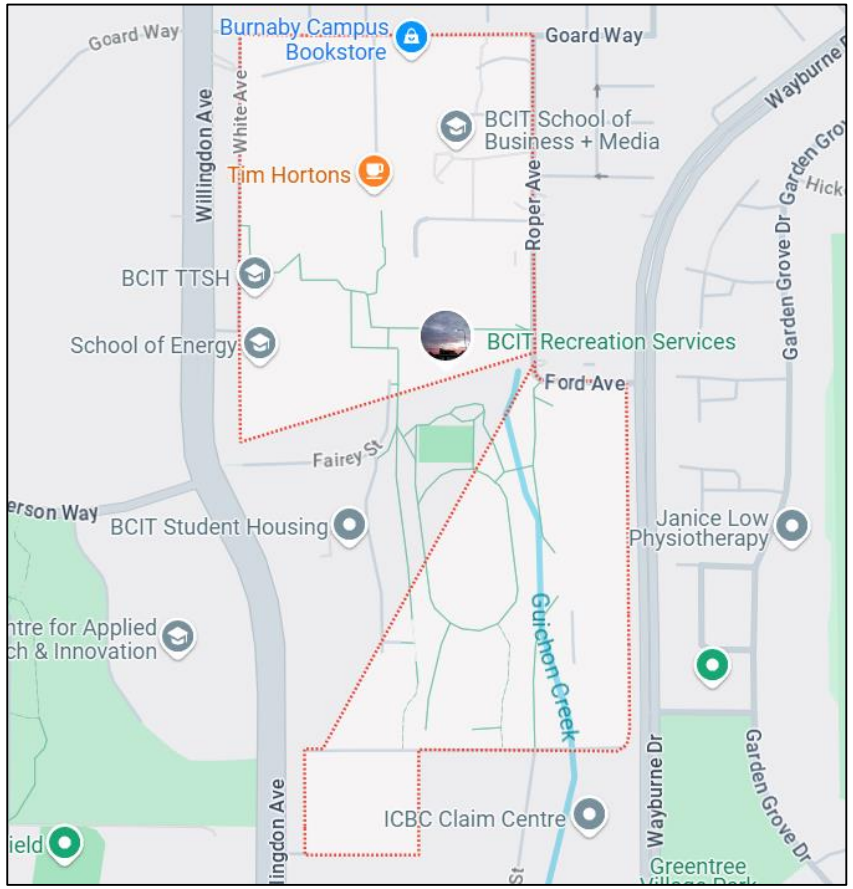


③ A receives B's reply, adds B entry into its local ARP table

IP address vs MAC address

House/Unit/Apt Number → MAC address

BCIT Postal Code: V5G 3H2 → IP address



Component	Postal System	Computer Network
MAC Address	Home/Mailbox Number	Unique Hardware Address (e.g., 00:1A:2B:3C:4D:5E)
IP Address	Postal Code (e.g., V3N 4Y7)	Logical Network Address (e.g., 192.168.1.10)

First, check the IP address and MAC address of your Network Interface Card (NIC)?

Windows: *ipconfig /all*

Linux/macOS: *ifconfig*

Then, visit the following website to find out who is the vendor of your NIC?

https://macaddress.io/

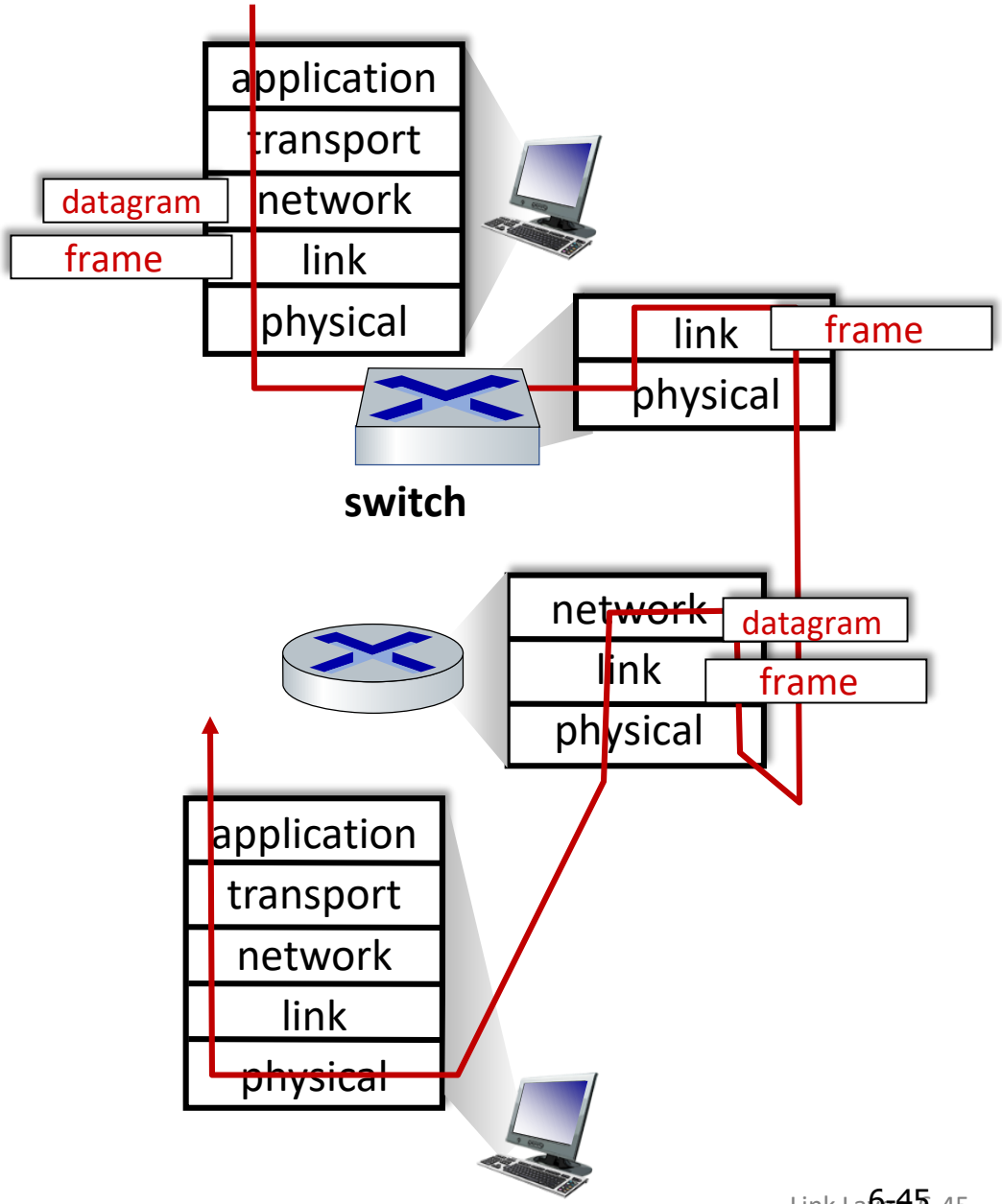
Switches vs. routers

both are store-and-forward:

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

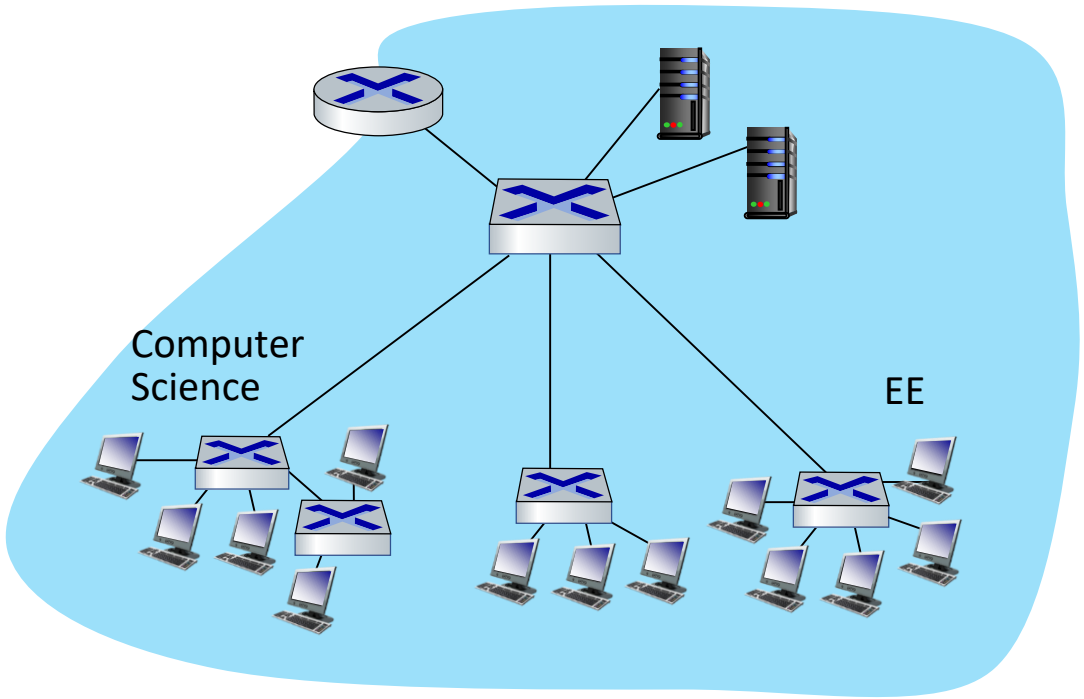
both have forwarding tables:

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



Virtual LANs (VLANs): motivation

Q: what happens as LAN sizes scale, users change point of attachment?

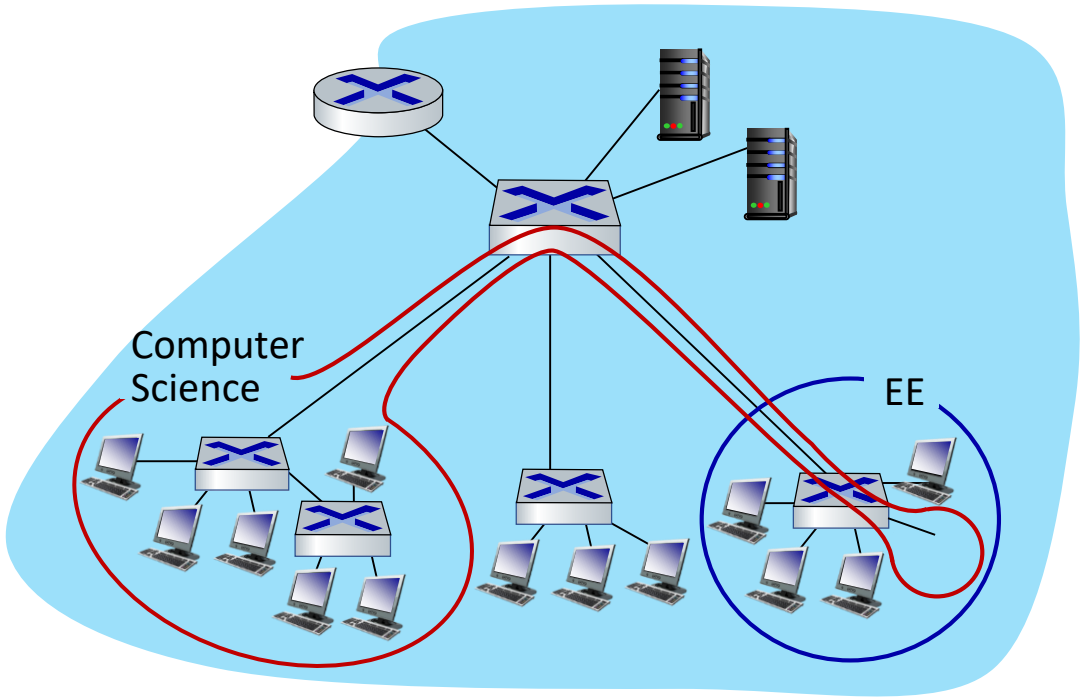


single broadcast domain:

- *scaling:* all layer-2 broadcast traffic (ARP, DHCP, unknown MAC) must cross entire LAN
- efficiency, security, privacy issues

Virtual LANs (VLANs): motivation

Q: what happens as LAN sizes scale, users change point of attachment?



single broadcast domain:

- *scaling:* all layer-2 broadcast traffic (ARP, DHCP, unknown MAC) must cross entire LAN
- efficiency, security, privacy, efficiency issues

administrative issues:

- CS user moves office to EE - *physically* attached to EE switch, but wants to remain *logically* attached to CS switch

IPv6: motivation

- initial motivation: 32-bit IPv4 address space would be completely allocated
- additional motivation:
  - speed processing/forwarding: 40-byte fixed length header
  - enable different network-layer treatment of “flows”

IPv4	IPv6
32-bit address space = 4,294,967,296 (2^32) unique addresses	128-bit address space = 340,282,366,920,938,463,463,374, 607,431,768,211,456 (2^128) unique addresses
example: 103.81.45.28	example: 2001:0DB8:0000:2F3B:02AA:00FF:FE28:9C5A

IPv4 vs. IPv6

Address Space:

- **IPv4:** Limited, leading to address exhaustion.
- **IPv6:** Vast, providing a nearly unlimited number of addresses.

Configuration:

- **IPv4:** Manual or DHCP.
- **IPv6:** Stateless Address Auto-configuration (SLAAC) and DHCPv6.

Security:

- **IPv4:** Optional IPSec support.
- **IPv6:** Mandatory IPSec support, integrated into the protocol.

Network Address Translation (NAT):

- **IPv4:** Commonly used due to limited address space.
- **IPv6:** Not required, supporting end-to-end connectivity.

Header Complexity:

- **IPv4:** More complex with various optional fields.
- **IPv6:** Simplified header structure for more efficient processing.

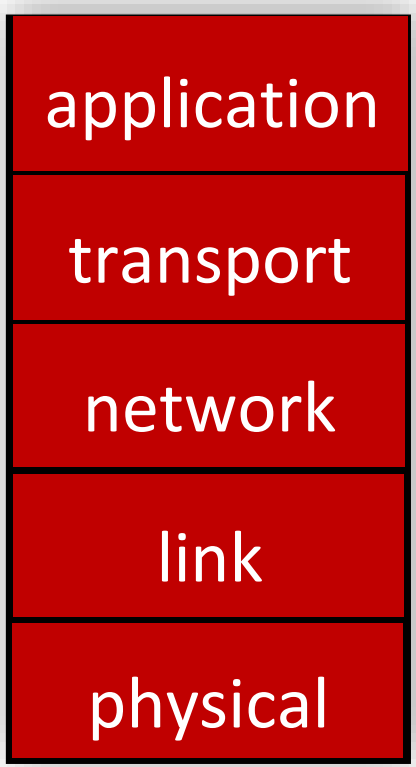
Performance:

- **IPv4:** Can be slower due to header complexity and NAT overhead.

- **IPv6:** Potentially faster due to simplified headers and elimination of NAT.

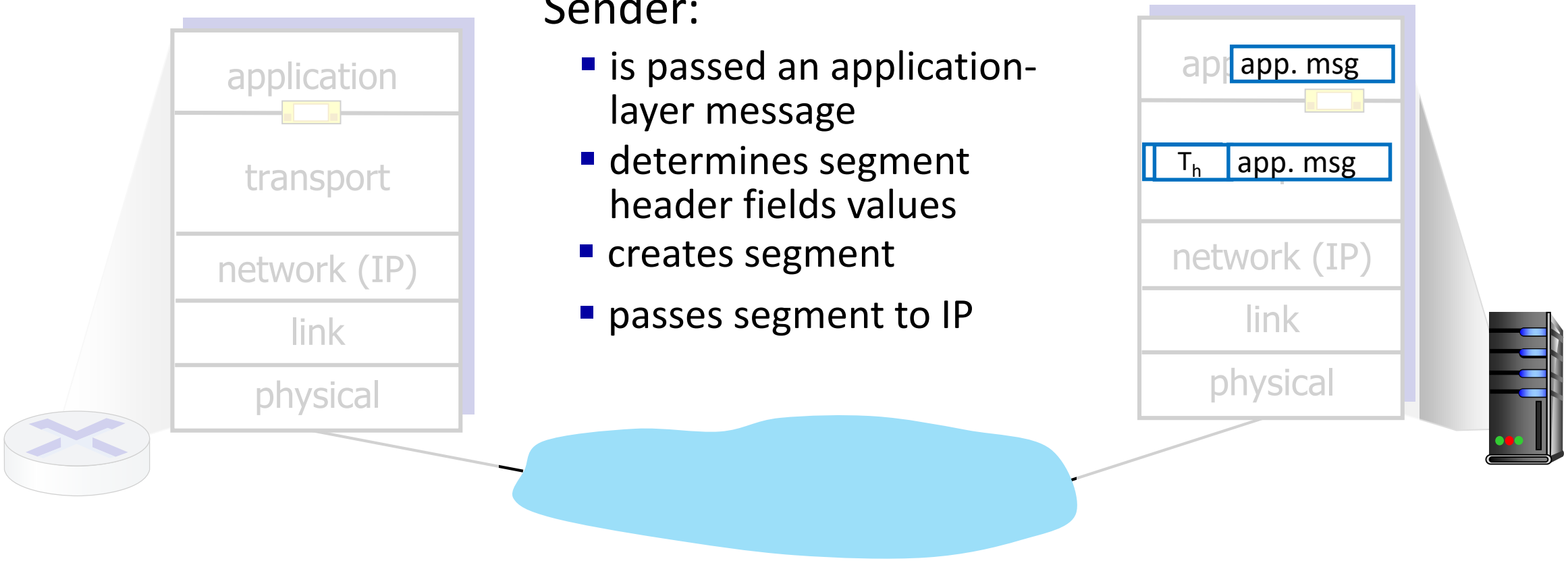
Layered Internet protocol stack

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



Five-Layer Internet Protocol Stack

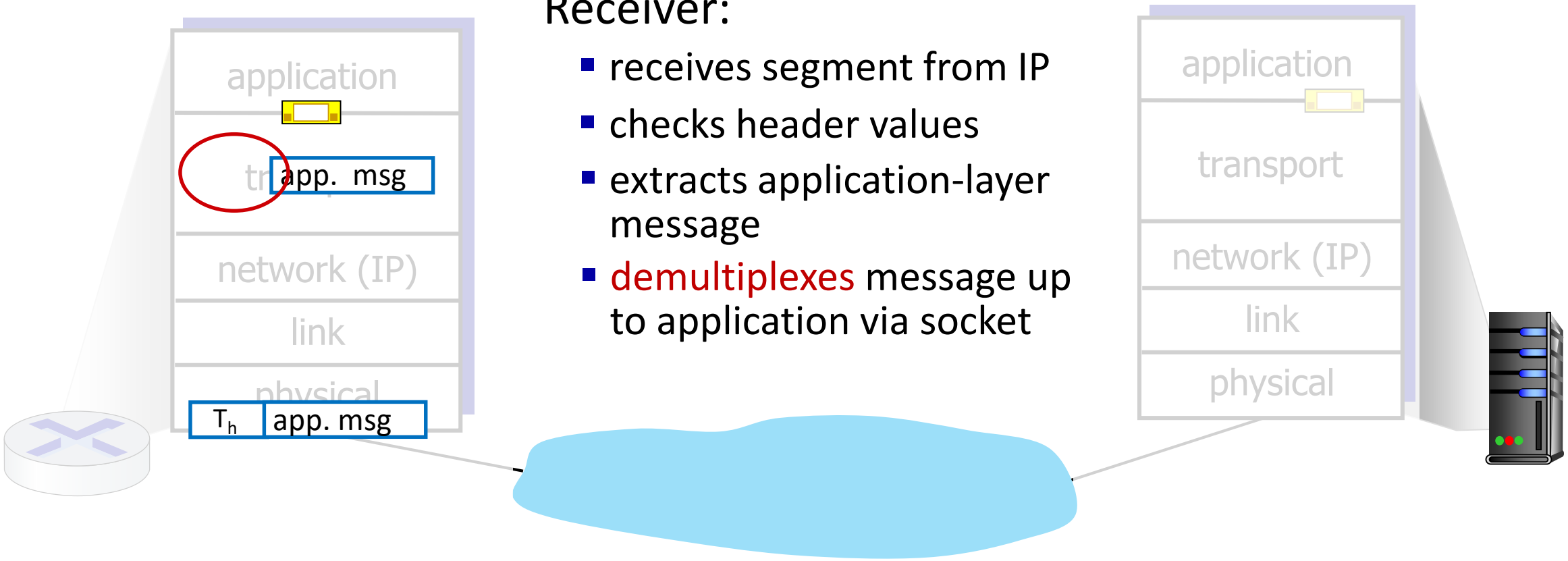
Transport Layer Actions



Sender:

- is passed an application-layer message
- determines segment header fields values
- creates segment
- passes segment to IP

Transport Layer Actions

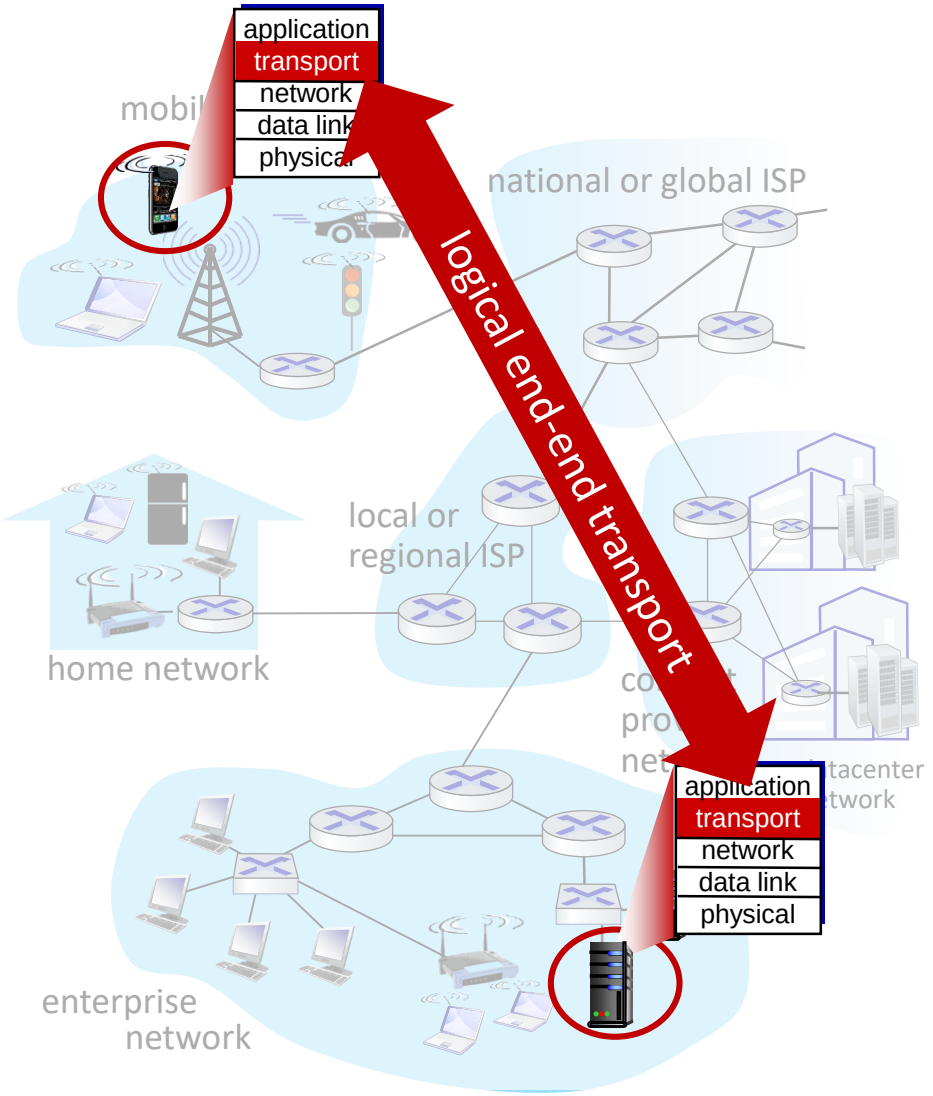


Receiver:

- receives segment from IP
- checks header values
- extracts application-layer message
- demultiplexes message up to application via socket

Two principal Internet transport protocols

- **TCP:** Transmission Control Protocol
  - reliable, in-order delivery
  - congestion control
  - flow control
  - connection setup
- **UDP:** User Datagram Protocol
  - unreliable, unordered delivery
  - no-frills extension of “best-effort” IP
- services *not* available:
  - delay guarantees
  - bandwidth guarantees



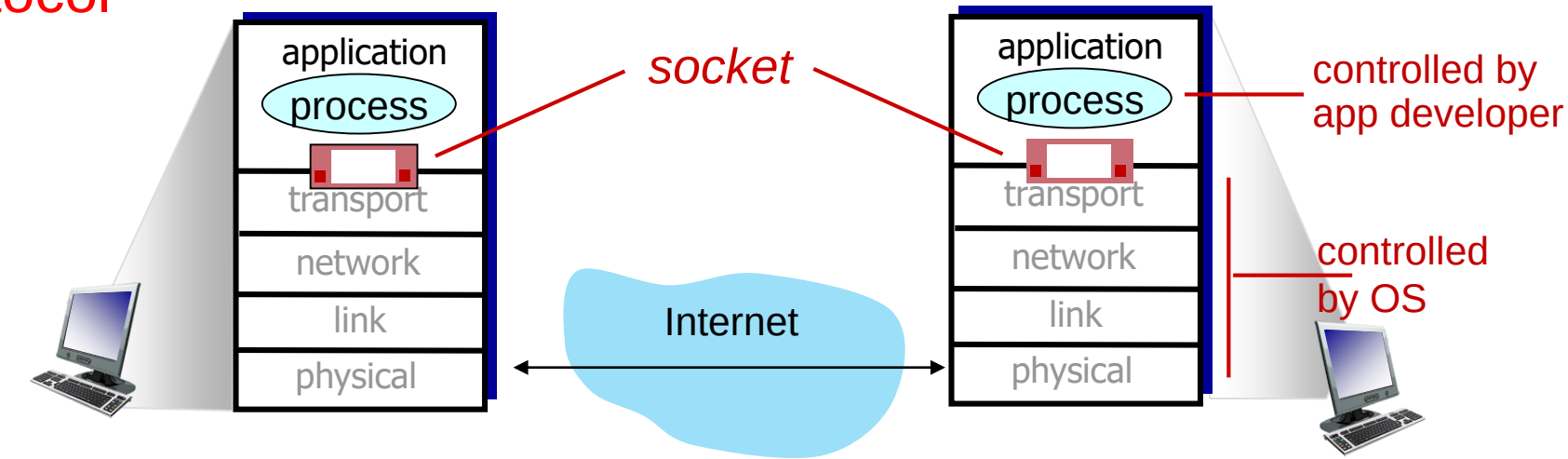
TCP vs UDP ;)

- **TCP:** Transmission Control Protocol
  - reliable, in-order delivery
  - congestion control
  - flow control
  - connection setup
- **UDP:** User Datagram Protocol
  - unreliable, unordered delivery
  - no-frills extension of “best-effort” IP
- services *not* available:
  - delay guarantees
  - bandwidth guarantees

Sockets

- process sends/receives messages to/from its socket
- socket analogous to door
  - sending process shoves message out door
  - sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process
  - two sockets involved: one on each side

Socket is a door between application process and end-end-transport protocol



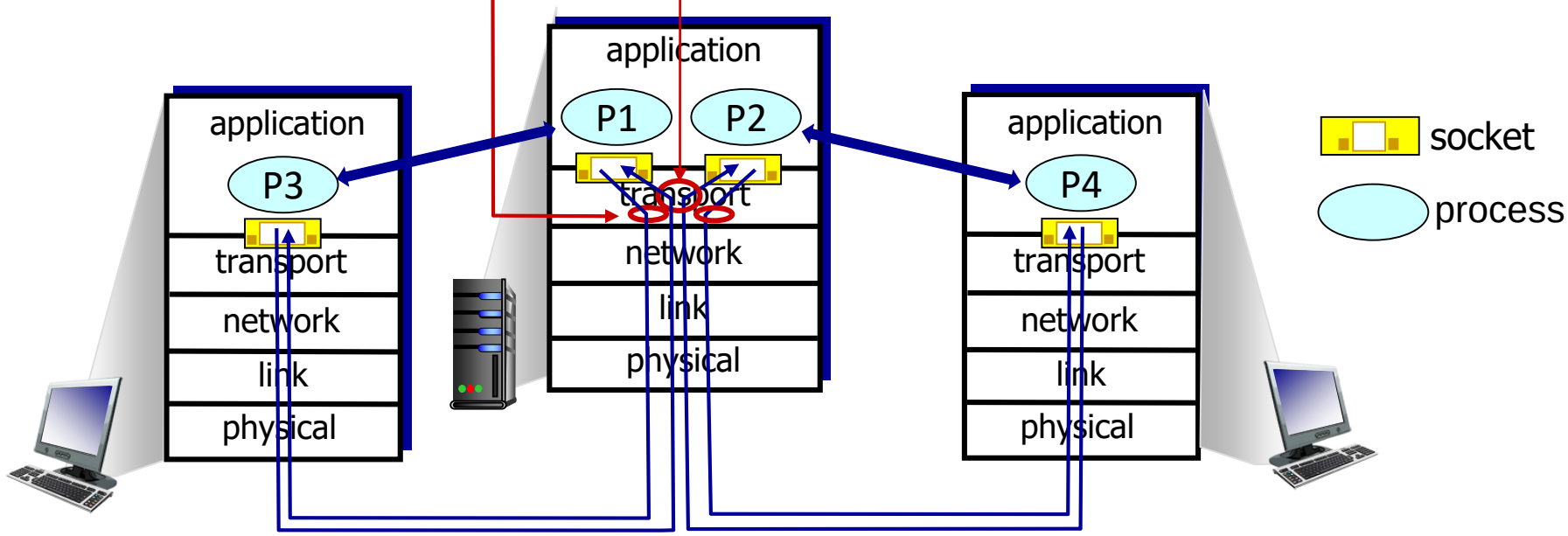
Multiplexing/demultiplexing

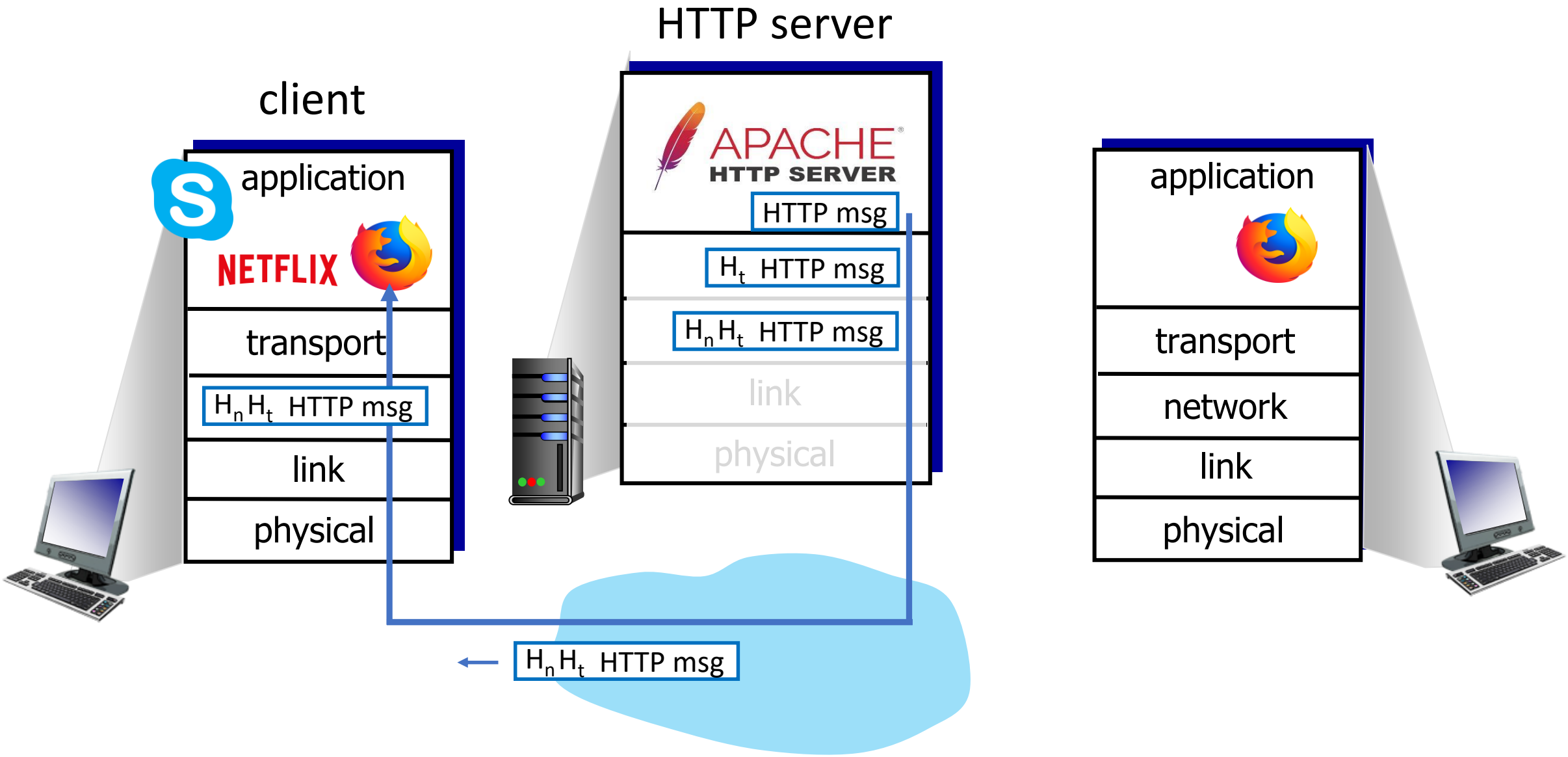
multiplexing as sender:

handle data from multiple sockets, add transport header (later used for demultiplexing)

demultiplexing as receiver:

use header info to deliver received segments to correct socket




HTTP server
client
application
NETFLIX
transport
$H_n H_t$ HTTP msg
link
physical
APACHE
HTTP SERVER
HTTP msg
H_t HTTP msg
$H_n H_t$ HTTP msg
link
physical
application
transport
network
link
physical
$H_n H_t$ HTTP msg

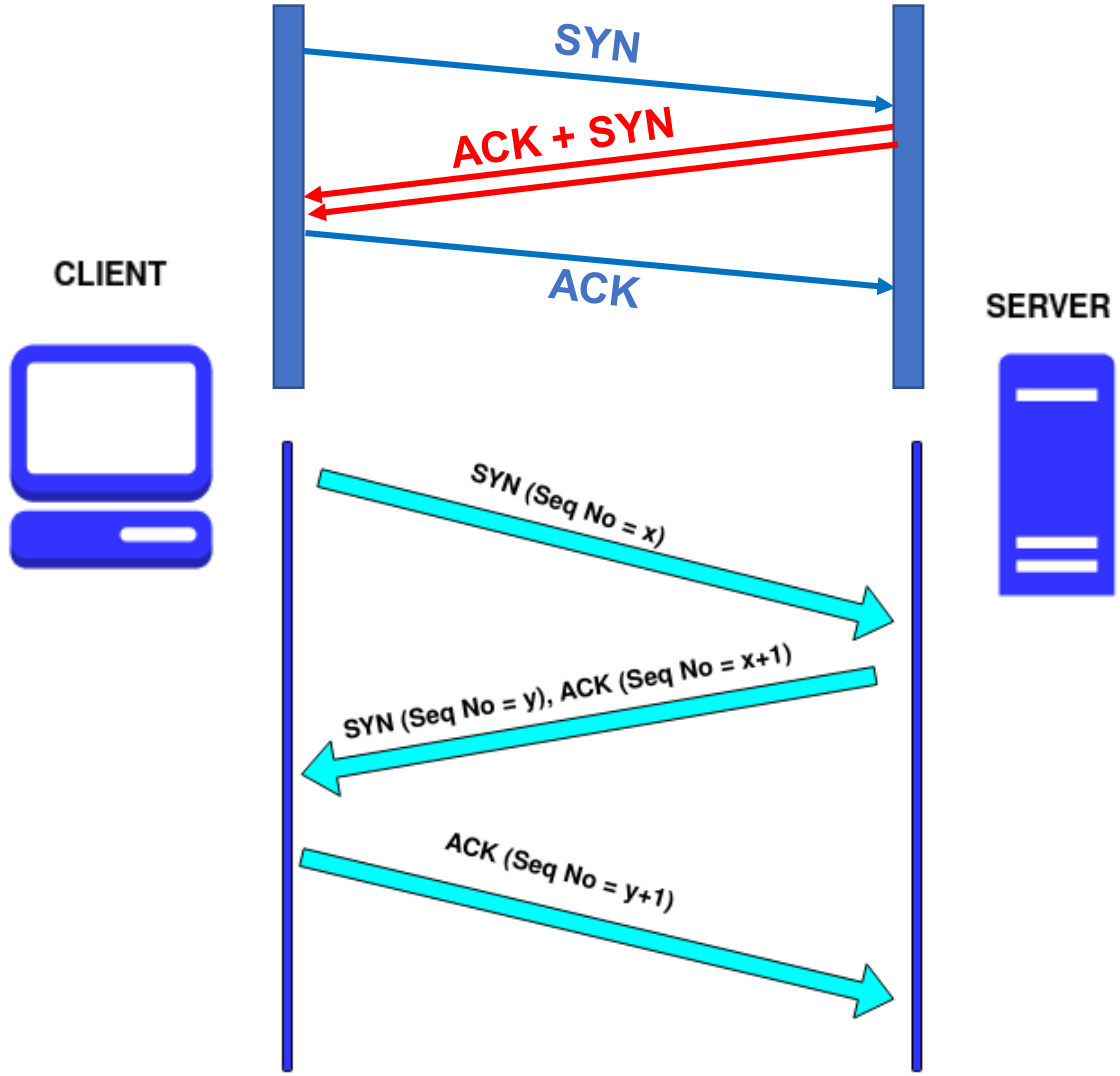
TCP: Three-way handshake

The TCP three-way handshake is a process used to establish a connection between a client and a server:

1. **SYN (Synchronize):**
The client sends a SYN packet to the server to initiate a connection.

2. **SYN-ACK (Synchronize-Acknowledge):**
The server responds with a SYN-ACK packet to acknowledge the client's request and indicate it is ready to establish a connection.

3. **ACK (Acknowledge):**
The client sends an ACK packet back to the server, confirming the connection is established



UDP: User Datagram Protocol

- "no frills," "bare bones" Internet transport protocol
- "best effort" service, UDP segments may be:
  - lost
  - delivered out-of-order to app
- *connectionless:*
  - no handshaking between UDP sender, receiver
  - each UDP segment handled independently of others

Why is there a UDP?

- no connection establishment (which can add RTT delay)
- simple: no connection state at sender, receiver
- small header size
- no congestion control
  - UDP can blast away as fast as desired!
  - can function in the face of congestion

UDP: User Datagram Protocol

- UDP use:
  - streaming multimedia apps (loss tolerant, rate sensitive)
  - DHCP, DNS, NTP, TFTP, SNMP, SIP (VoIP), …
- if reliable transfer needed over UDP (e.g., HTTP/3):
  - add needed reliability at application layer
  - add congestion control at application layer

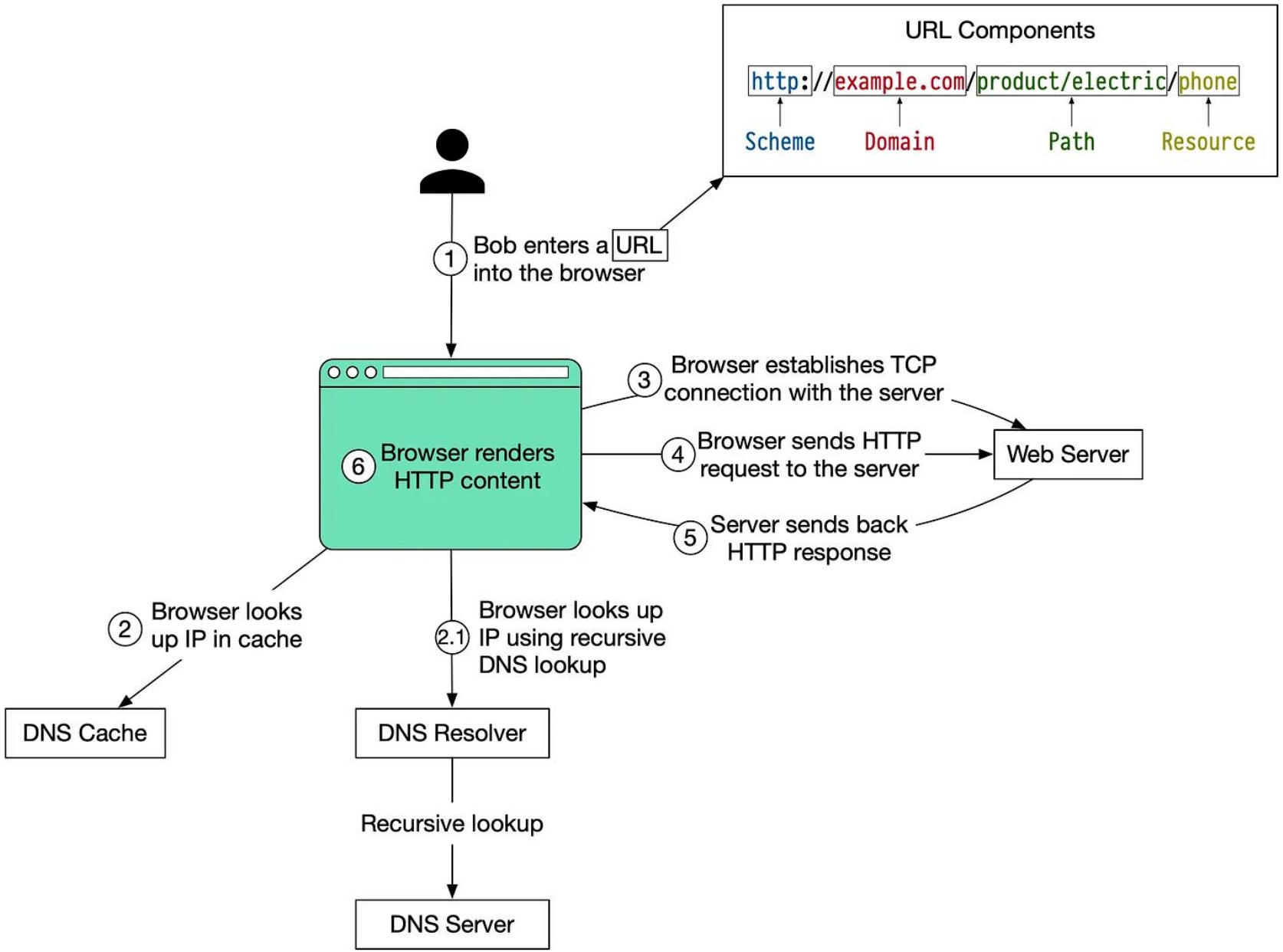
A day in the life of a web request

- *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

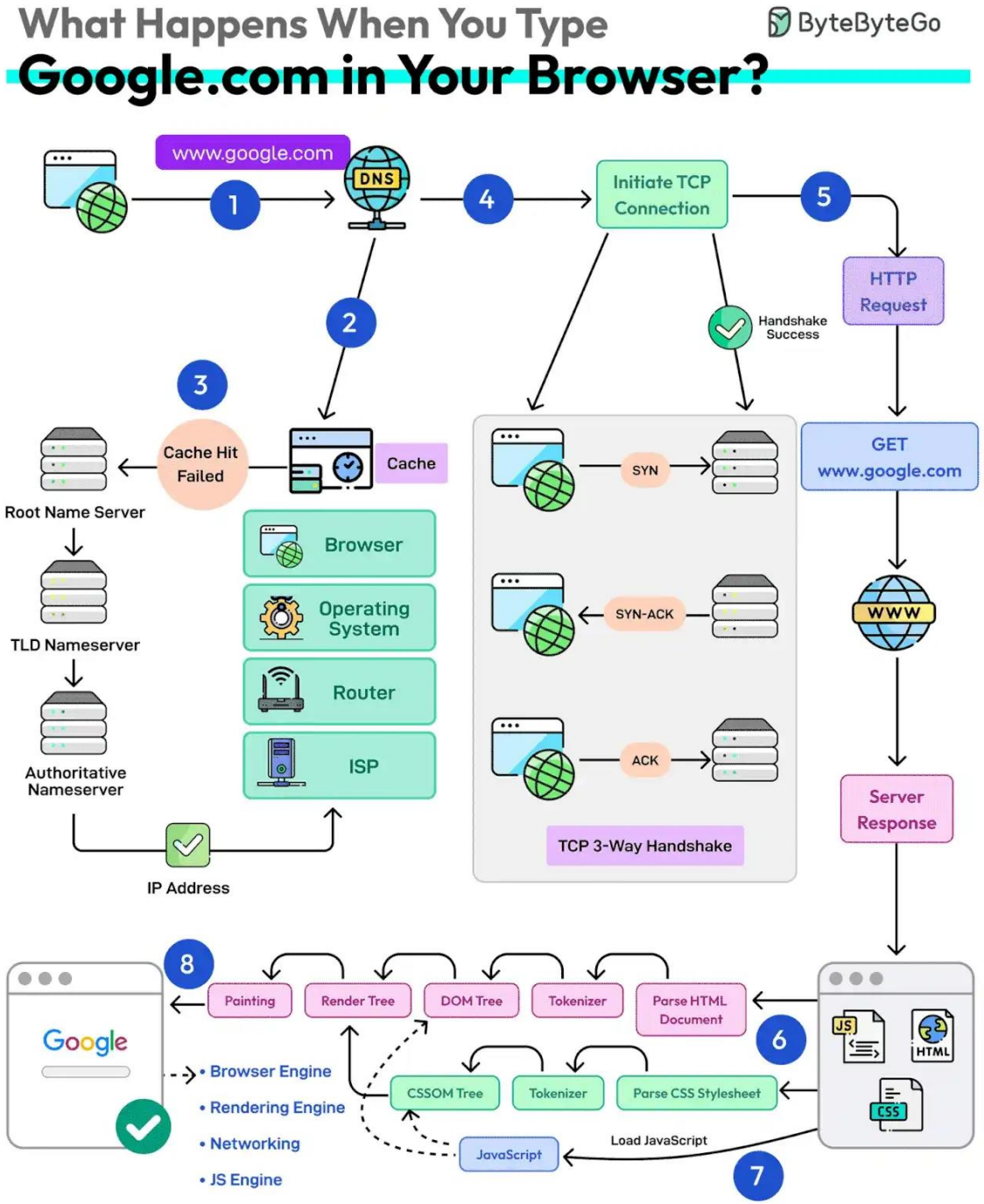
THERE IS NOT A SINGLE AND SIMPLE ANSWER TO THIS QUESTION

What happens when you type a URL into your browser?

ByteByteGo



https://blog.bytebytego.com/p/what-happens-when-you-type-a-url



1. First up, you type the website address in the browser's address bar.

2. The browser checks its cache first. If there's a cache miss, it must find the IP address.

3. DNS lookup begins (think of it as looking up a phone number). The request goes through different DNS servers (root, TLD, and authoritative). Finally, the IP address is retrieved.

4. Next, your browser initiates a TCP connection like a handshake. For example, in the case of HTTP 1.1, the client and server perform a TCP 3-way handshake with SYN, SYN-ACK, and ACK messages.

5. Once the handshake is successful, the browser makes an HTTP request to the server and the server responds with HTML, CSS, and JS files.

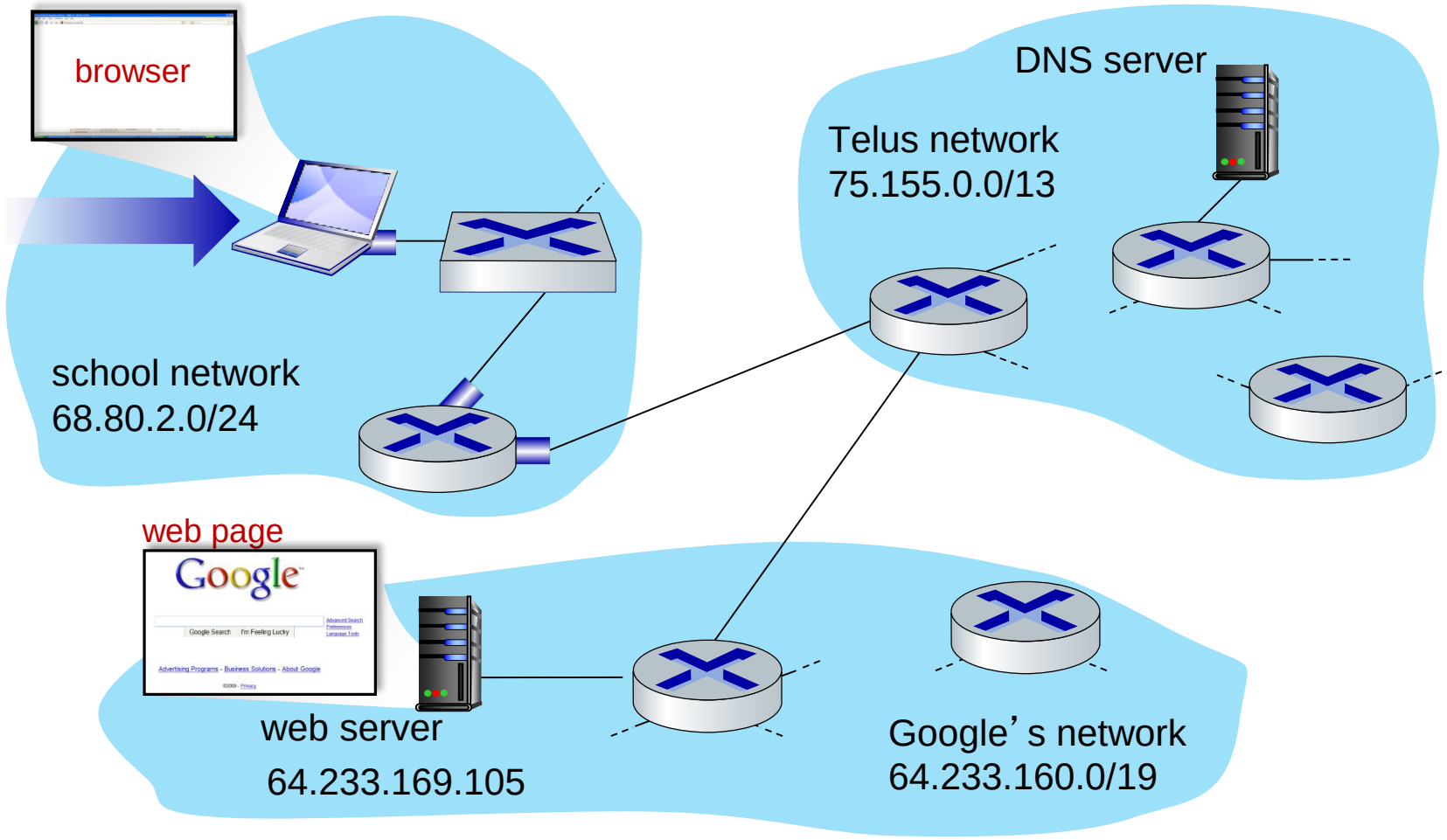
6. Finally, the browser processes everything. It parses the HTML document and creates DOM and CSSOM trees.

7. The browser executes the JavaScript and renders the page through various steps (tokenizer, parser, render tree, layout, and painting).

8. Finally, the webpage appears on your screen.

https://blog.bytebytego.com/p/ep145-infrastructure-as-code-landscape

A day in the life: scenario

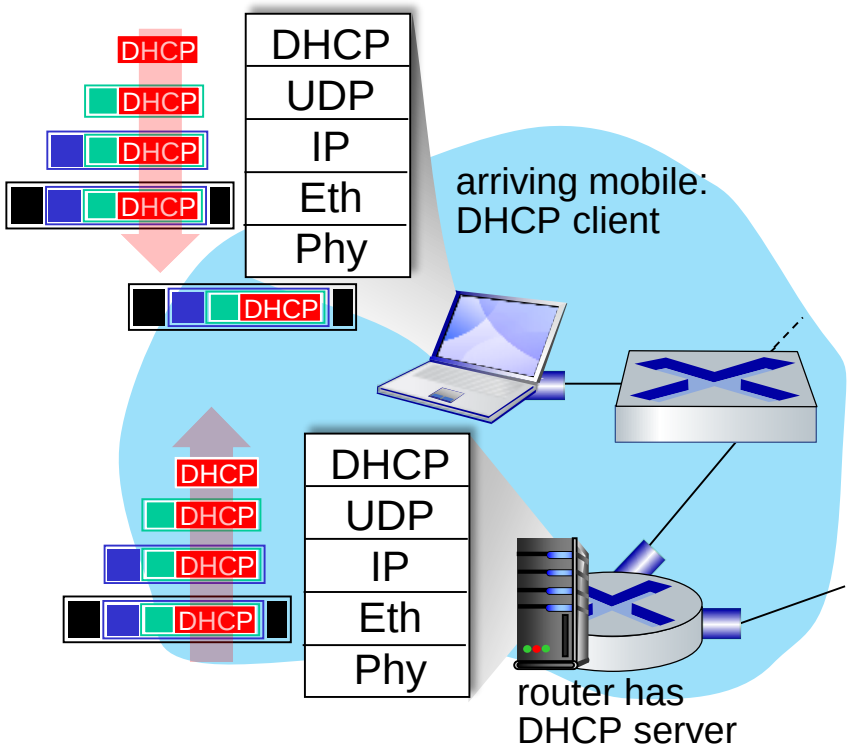


scenario:

- arriving mobile client attaches to network …
- requests web page: www.google.com

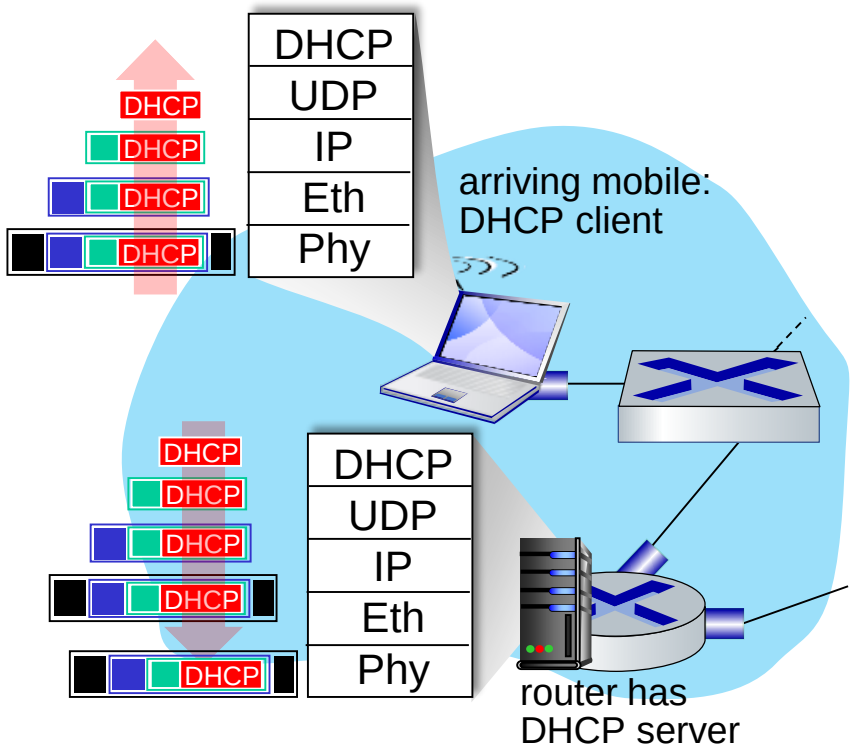
Sounds simple!

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 de-muxed to IP de-muxed, UDP de-muxed to DHCP

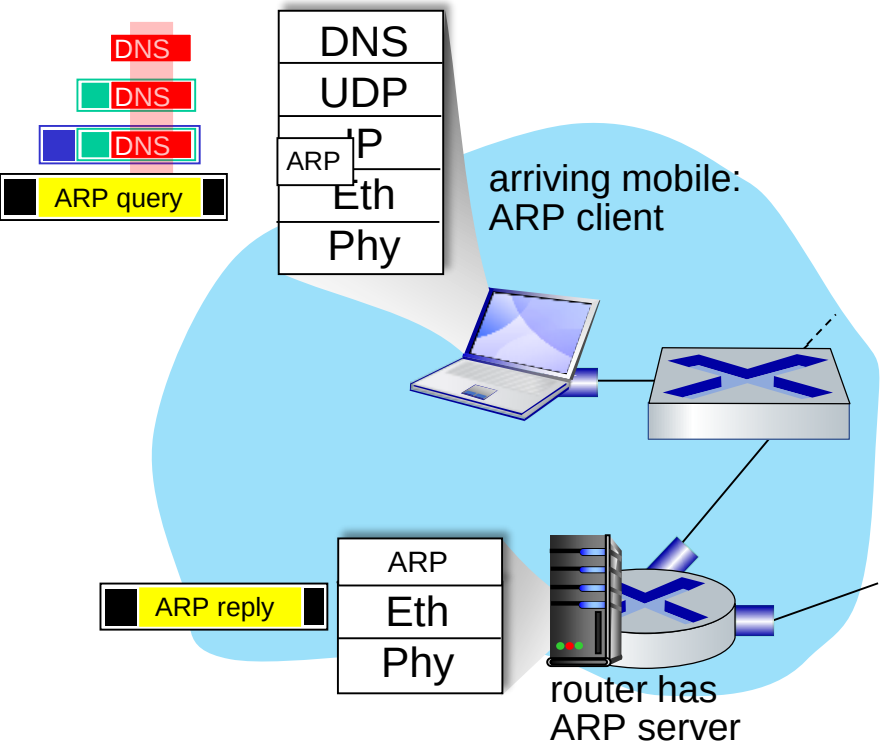
A day in the life: connecting to the Internet



- 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

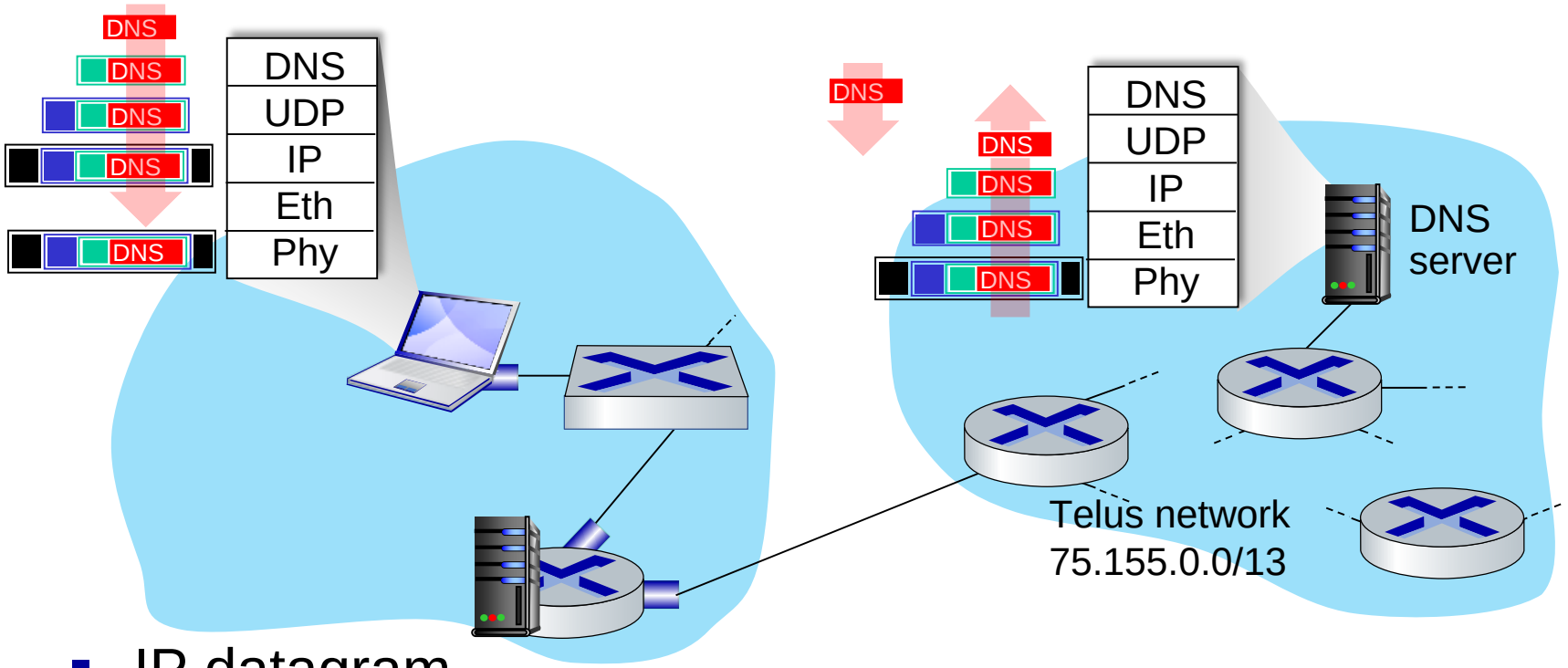
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

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

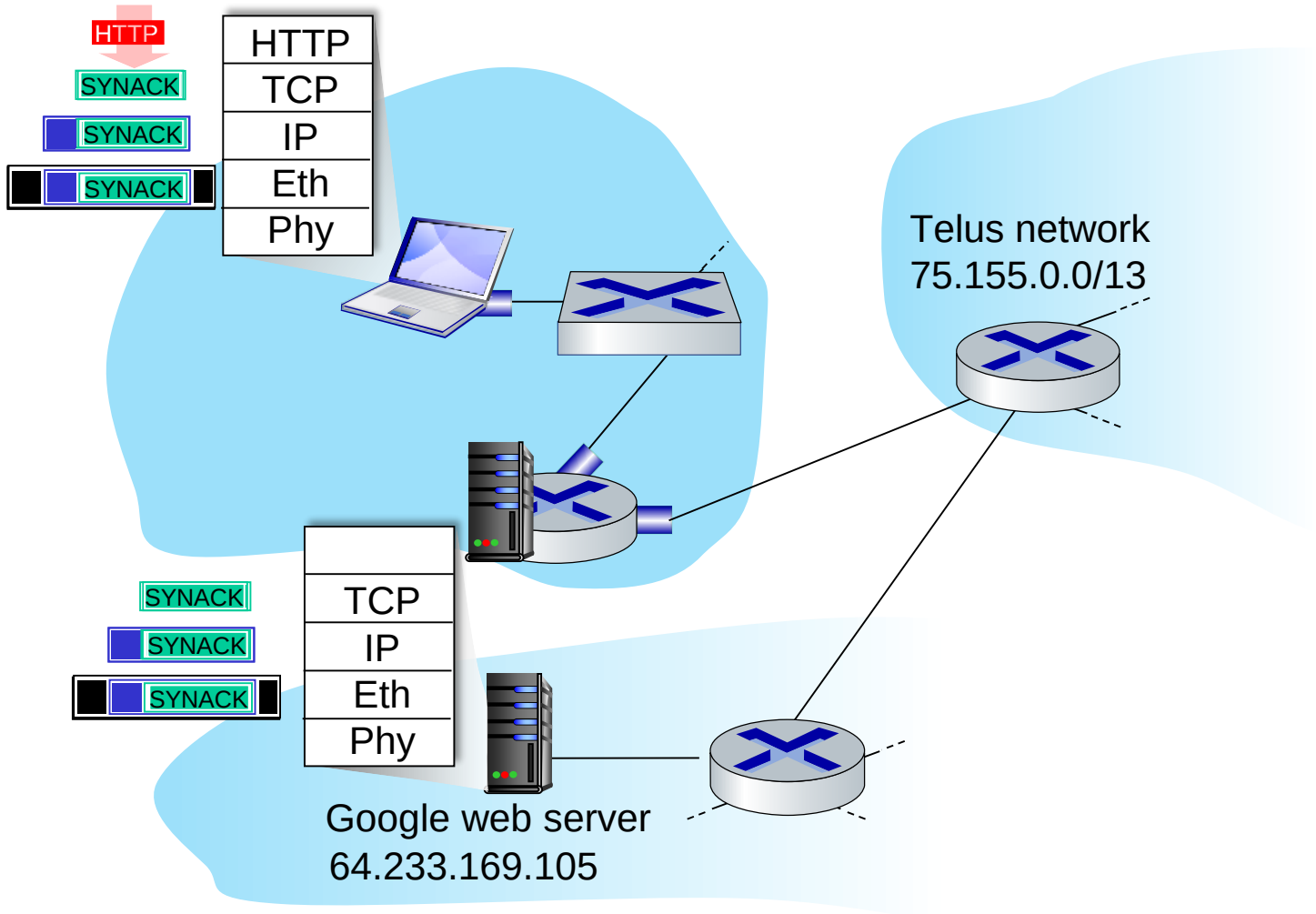


- de-muxed to DNS
- DNS replies to client with IP address of www.google.com

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

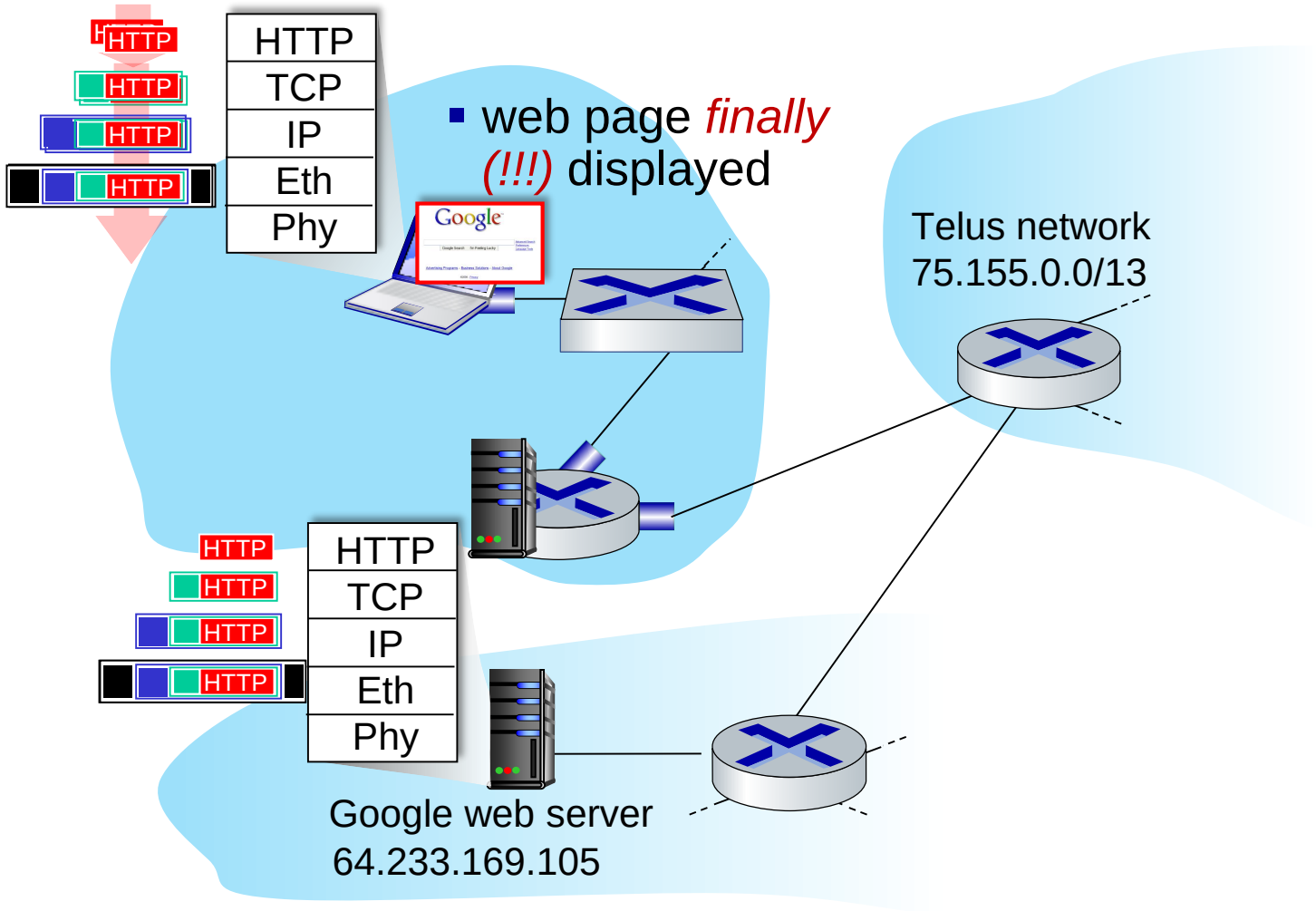
- IP datagram forwarded from campus network into Telus network, routed (tables created by RIP, OSPF, IS-IS and/or BGP routing protocols) to DNS server

A day in the life…TCP connection carrying HTTP



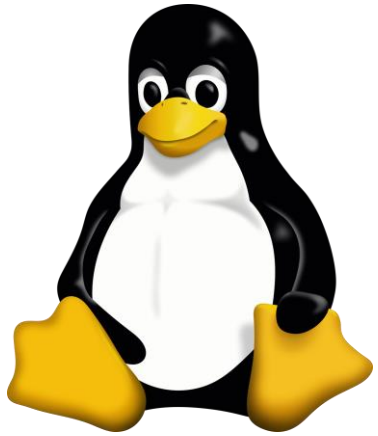
- to send HTTP request, client first opens TCP socket to web server
- TCP SYN segment (step 1 in TCP 3-way handshake) inter-domain routed to web server
- web server responds with TCP SYNACK (step 2 in TCP 3-way handshake)
- TCP connection established!

A day in the life… HTTP request/reply



- 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

A Brief Intro to Linux



debian

Origins of Linux

Unix Inspiration: Developed in the late 1960s at AT&T's Bell Labs.
GNU Project (1983): Launched by Richard Stallman to create a free OS.

Red Hat

Linux Kernel Creation

Linus Torvalds (1991): Created the Linux kernel as a personal project.
First Release (1991): v0.01 released on Minix; gained a developer community.

fedora

ubuntu

Linux Growth

1990s: Collaboration with GNU tools created a fully functioning OS (GNU/Linux).

CentOS

gentoo linux™

Amazon Linux

90%+ of public cloud servers run on Linux (AWS, Google Cloud, Azure).

Android powers over **2.5 billion active devices** worldwide,

Billions of IoT devices run Linux, particularly due to lightweight distributions

KALI
BY OFFENSIVE SECURITY

alpine Linux

BCIT™

A Brief Intro to Linux

- ✓ Master the Basics/Concepts
- ✓ Learn by Building Small Projects
- ✓ Embrace Curiosity
- ✓ Take Advantage of Online Materials
- ✓ Join Linux Communities

Some basic commands

cd, ls, pwd, hostname
cp, mv, rm, mkdir, rmdir, man
cat, vim, nano, touch, find, grep

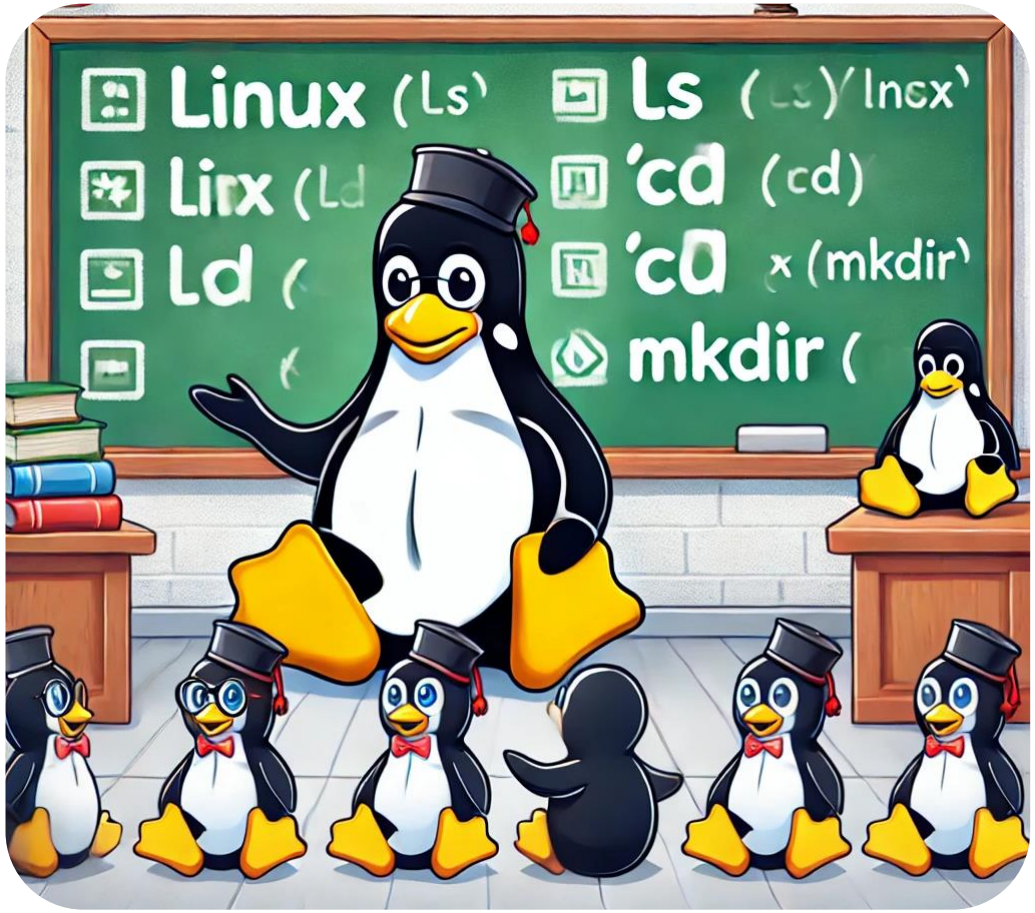
uname, df, du, tar, fdisk, whoami

top, ps, kill, jobs, init, uptime, date

sudo, yum/apt, ssh, scp, history

curl, wget, ping, mtr, ip, ifconfig
netstat, iptables

BCIT™



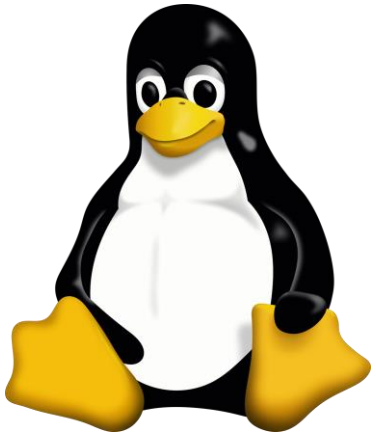
Some Nice References

https://linuxcommand.org/

https://linuxjourney.com/

https://linuxsurvival.com/

3 Breakout Rooms



1. Visit https://cocalc.com/features/terminal
2. Sign up the site.
3. Open "**New > Linux Terminal**" in your browser.

cd, ls, pwd
cp, mv, rm, mkdir, rmdir, man
cat, vim, nano, touch, find, grep

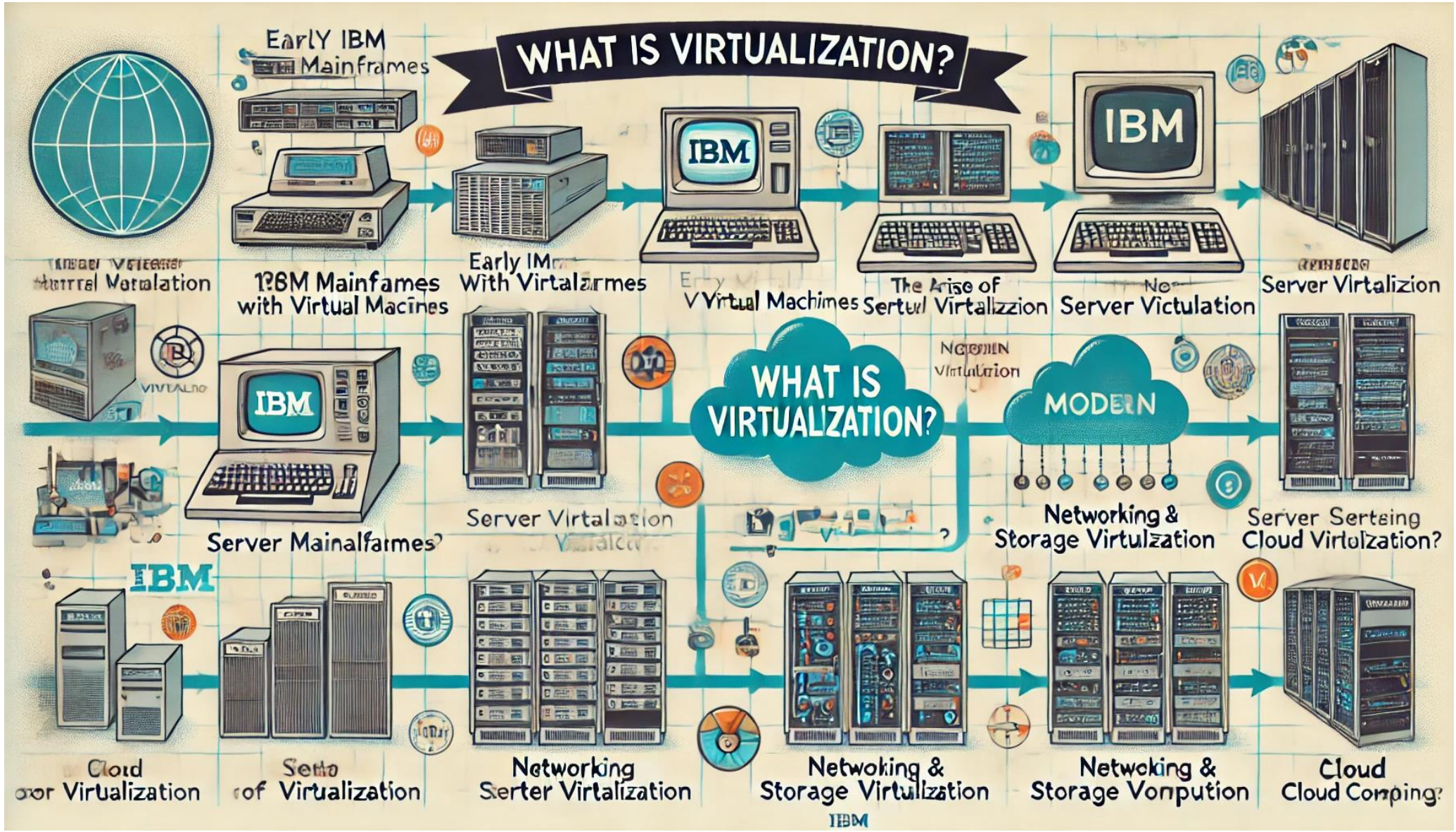
uname, df, du, tar, fdisk

top, ps, kill, jobs, init

sudo, yum/apt, ssh, scp

curl, wget, ping, mtr, ip
netstat, iptables

What is Virtualization?



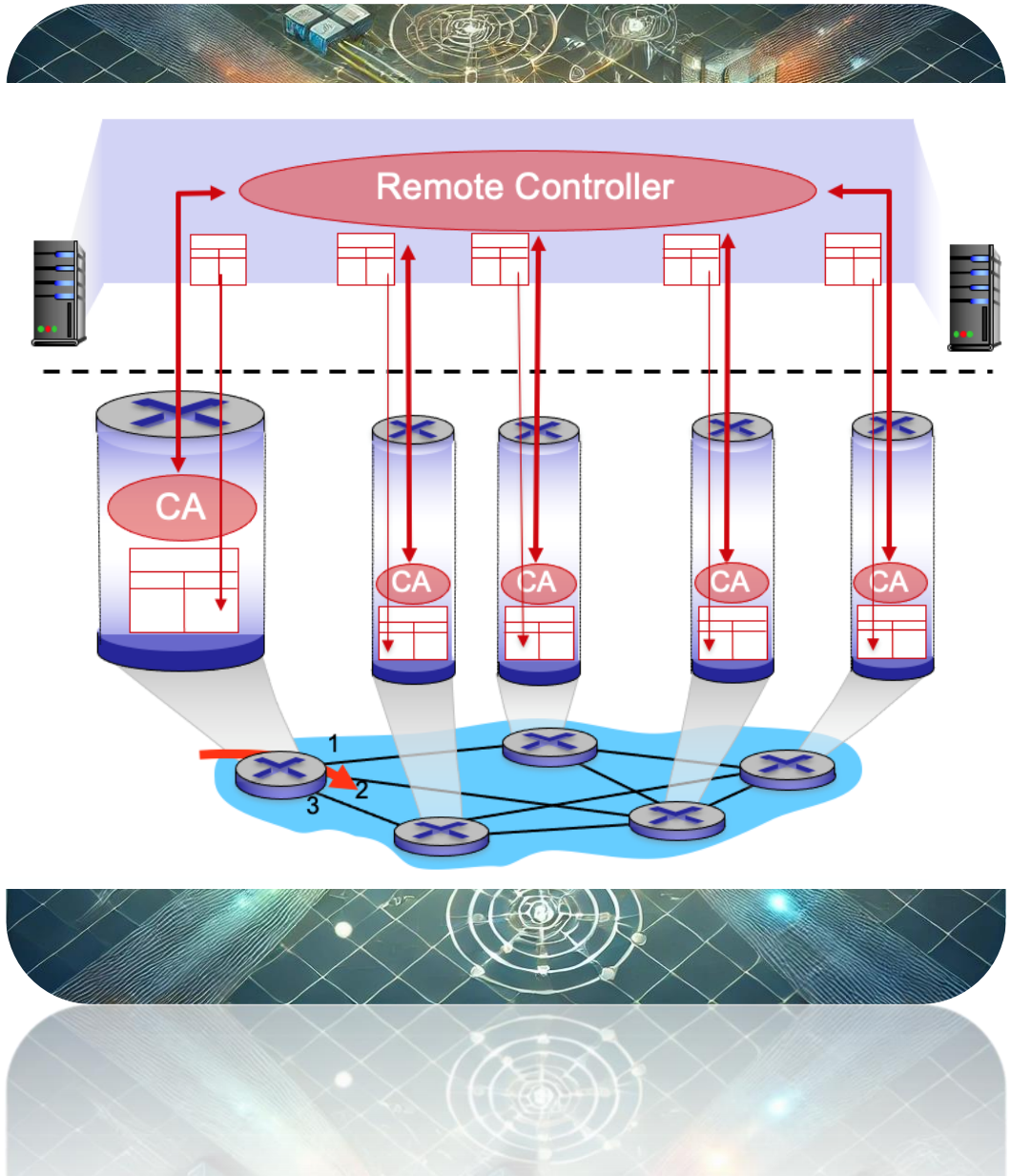
BCIT™

Network Virtualization

Abstracts physical network resources, such as Switches and Routers into Logical or Virtual Networks.

VLANs (Virtual LANs) or **VPNs** (Virtual Private Networks), **NFV** (Network Function Virtualization), **Overlay Networks** allow multiple networks to run over a single infrastructure.

Software-Defined Networking (**SDN**) is one the these technologies that separates control and data planes, allowing dynamic network management



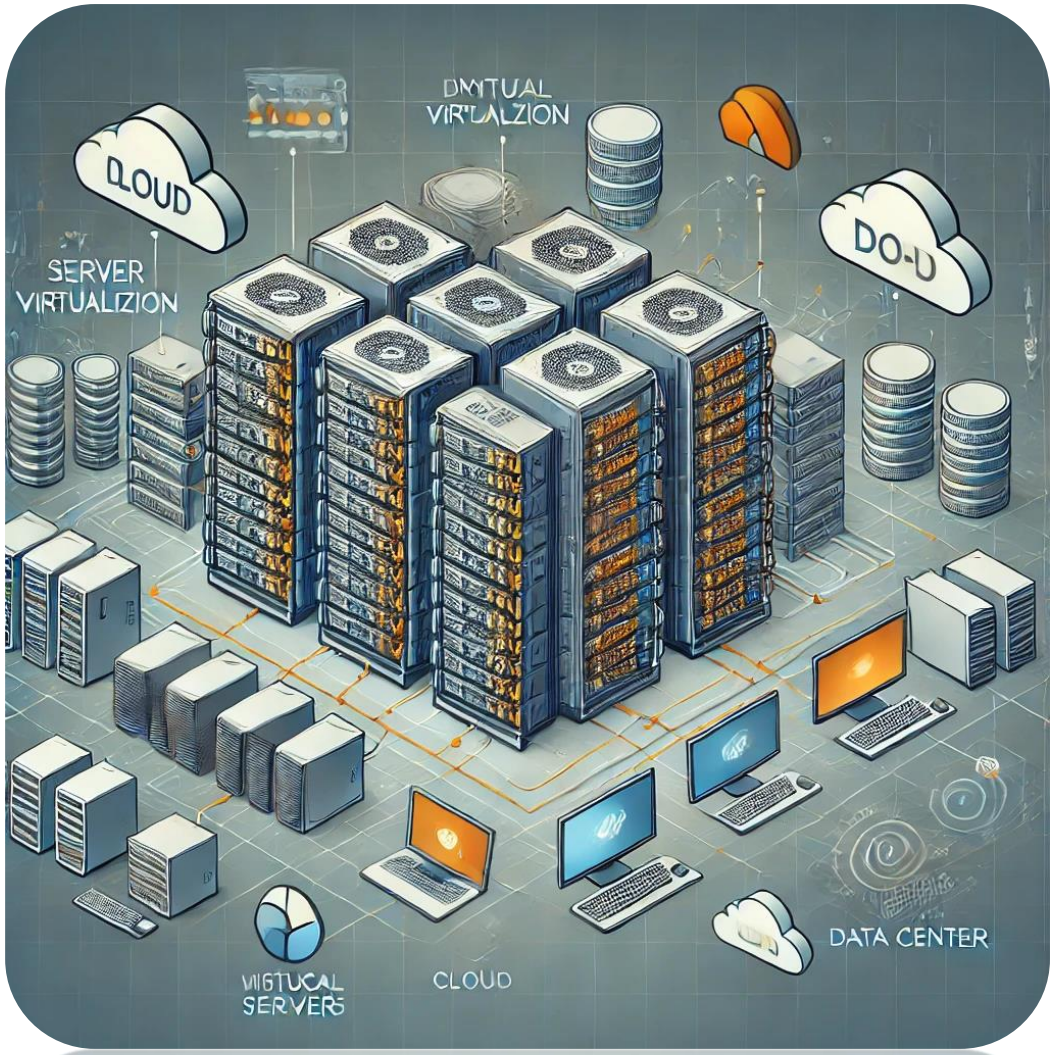
BCIT™

Server Virtualization

Multiple virtual machines (VMs) run on a single physical server

VMware, Hyper-V, and KVM allow servers to host isolated environments (VMs), each running a different Operating System or Application (can share the same Kernel).

Many Cloud providers such as AWS/GCP/Azure use virtualization to provision virtual servers to users.



BCIT™

Storage Virtualization

Combines multiple physical storage resources into a single virtual storage pool.

SAN (Storage Area Networks) or cloud storage solutions like Amazon S3. Ceph, vSAN, EBS, and Cinder also some of the known virtual storage technologies.

Enterprises use virtual storage to manage and allocate space efficiently across various applications.

BCIT™

Cloud Computing (Cloud Technology)

What is Cloud Technology?

Cloud Technology enables access to computing resources like storage, databases, servers, and software over the internet.

Types of Cloud Models

Public Cloud: Shared resources provided by third-party vendors.
Private Cloud: Dedicated resources for one organization.
Hybrid Cloud: Combination of public and private (on-premise) clouds.
Multi-Cloud: Use of multiple cloud services from different providers.

Benefits

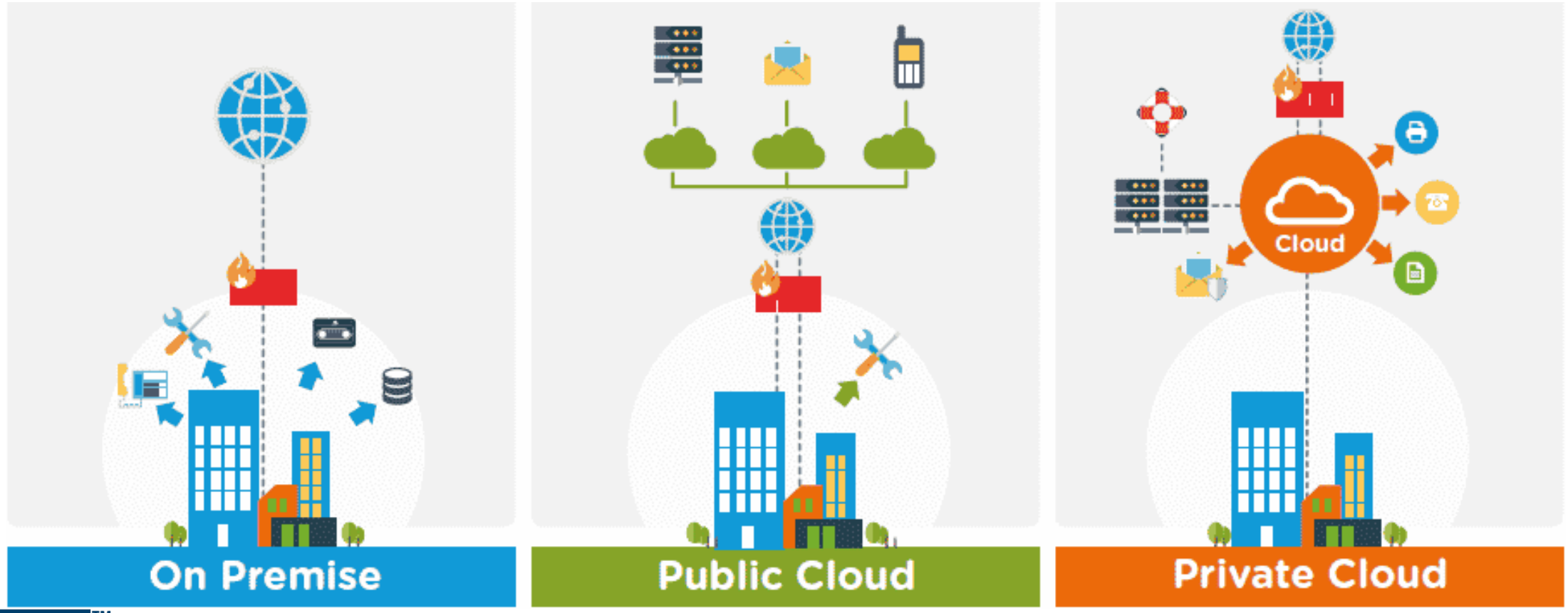
Scalability: Adjust resources based on demand.
Cost Efficiency: Pay-as-you-go pricing models.
Flexibility: Access from anywhere with internet.
Collaboration: Shared environments for teams and organizations.

BCIT™

Cloud Computing (Public Cloud)

BCIT™

Cloud Computing (Cloud Technology)



BCIT™

Cloud Computing (Cloud Technology)

Cloud service models provide different levels of management, from full applications to bare infrastructure

1. **SaaS (Software-as-a-Service)**:

Minimal user management (just access and settings).

NETFLIX Gmail Office 365

2. **PaaS (Platform-as-a-Service)**:

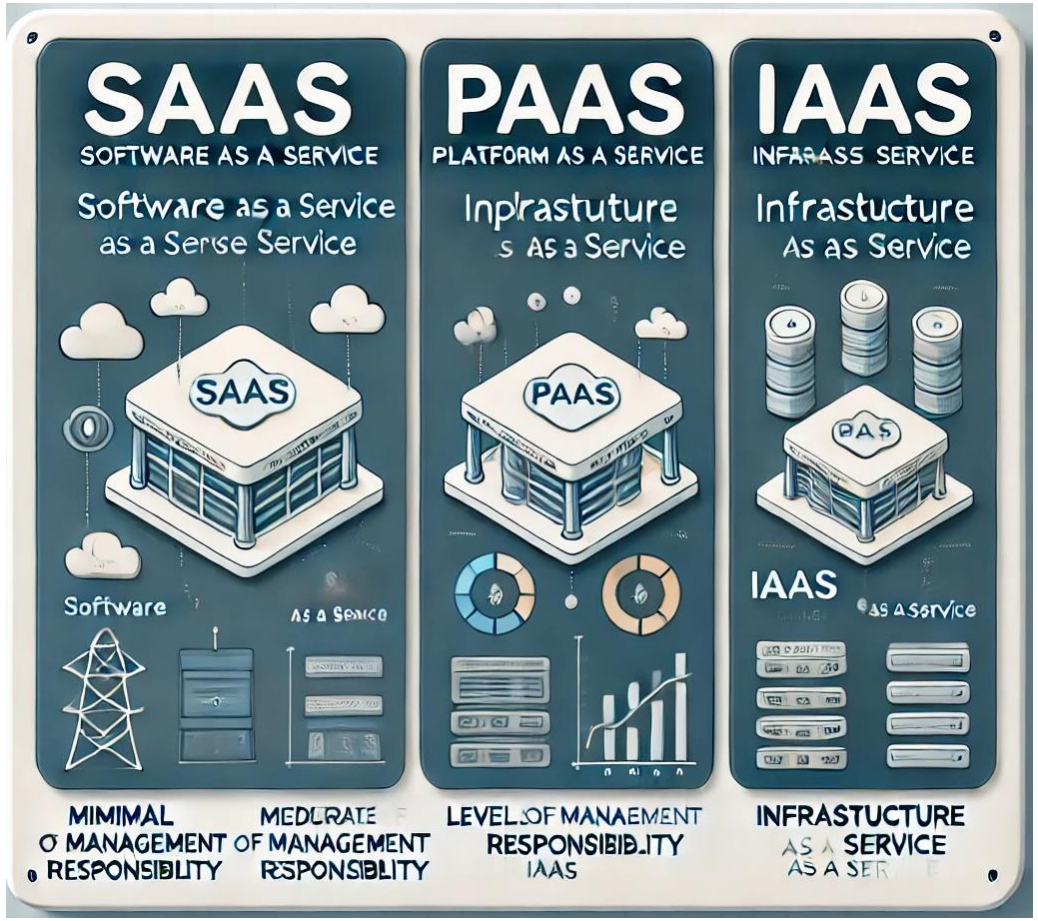
User manages applications and data, but not infrastructure.

HEROKU App Engine OPENSHIFT

3. **IaaS (Infrastructure-as-a-Service)**:

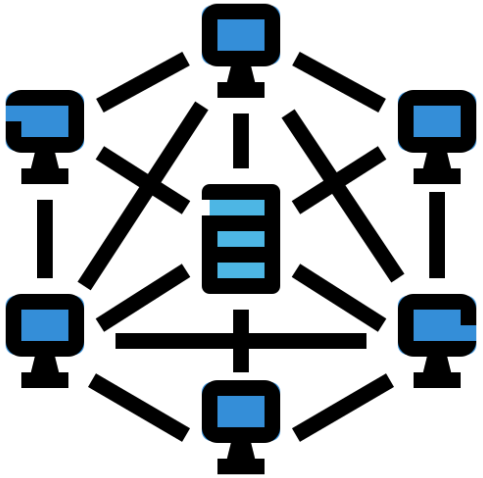
User manages applications, data, and operating systems, but the provider manages hardware.

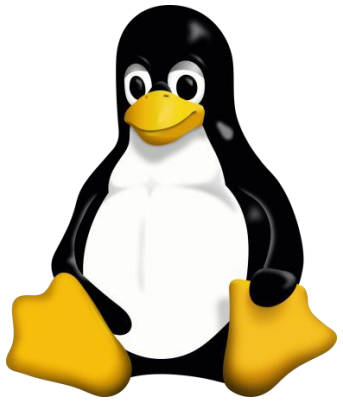
BCIT™ Amazon EC2 Google Cloud Platform Azure

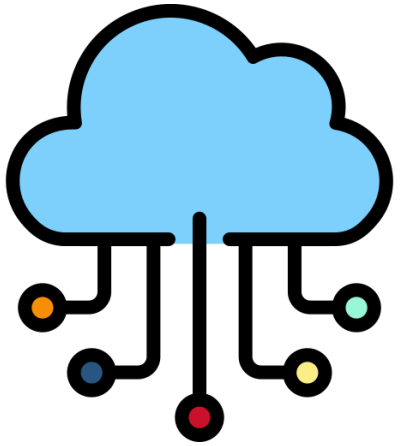


Lecture #1 Summary

1. Quick review of **Computer Networking** (Routing/Switching/IP/TCP/UDP)
2. Quick recap of Basic **Linux** Commands
3. Quick recap of the **Virtualization** and **Cloud Computing** Concepts

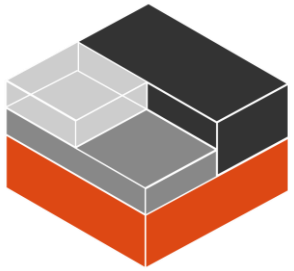


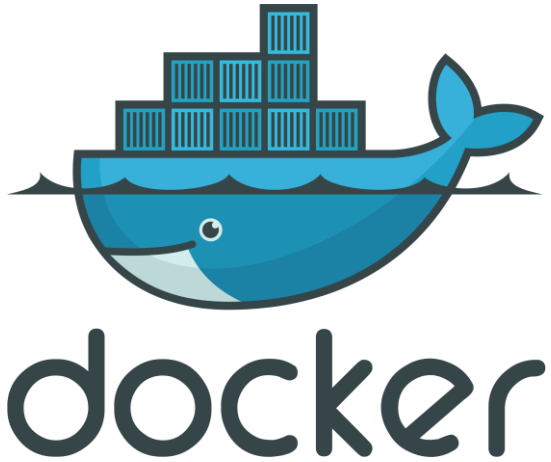


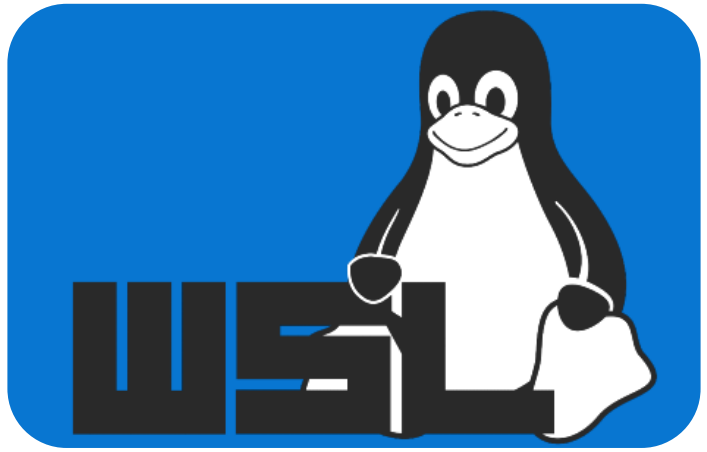


BCIT™

LAB/Project Requirements







BCIT™

Read **Lab0** PDF (Environment Setup)

End of Lecture #1

THANK
Y☺U

BCIT™