# Introduction to Networks

## Intermediate Computer Systems Programming, Fall 2025

**Instructor:** Travis McGaha

**TAs:** Theodor Bulacovschi Ash Fujiyama

# Administrivia

- Midterm Grades posted
  - Clobber Policy!
  - Feel free to reach out!
- Mid semester Survey results: I will look at eventually. I got sick :)))))))))
- HW08: Due tomorrow
- Final Project Partner Sign-up
  - If you want to work solo, let me know or you will be randomly assigned a partner
  - Sign-ups posted tonight? (I'm sick so maybe delayed)
- HW9 posted after class on Wed: implementing TCP

# Lecture Outline

- **Introduction to Networks**
  - **OSI model & TCPIP model**
    - **Layers upon layers upon layers…**
  - **Physical & Data Link Layer**
  - **IP Layer**
  - **Transport Layer**
  - **Application Layer**
- Analyzing this model

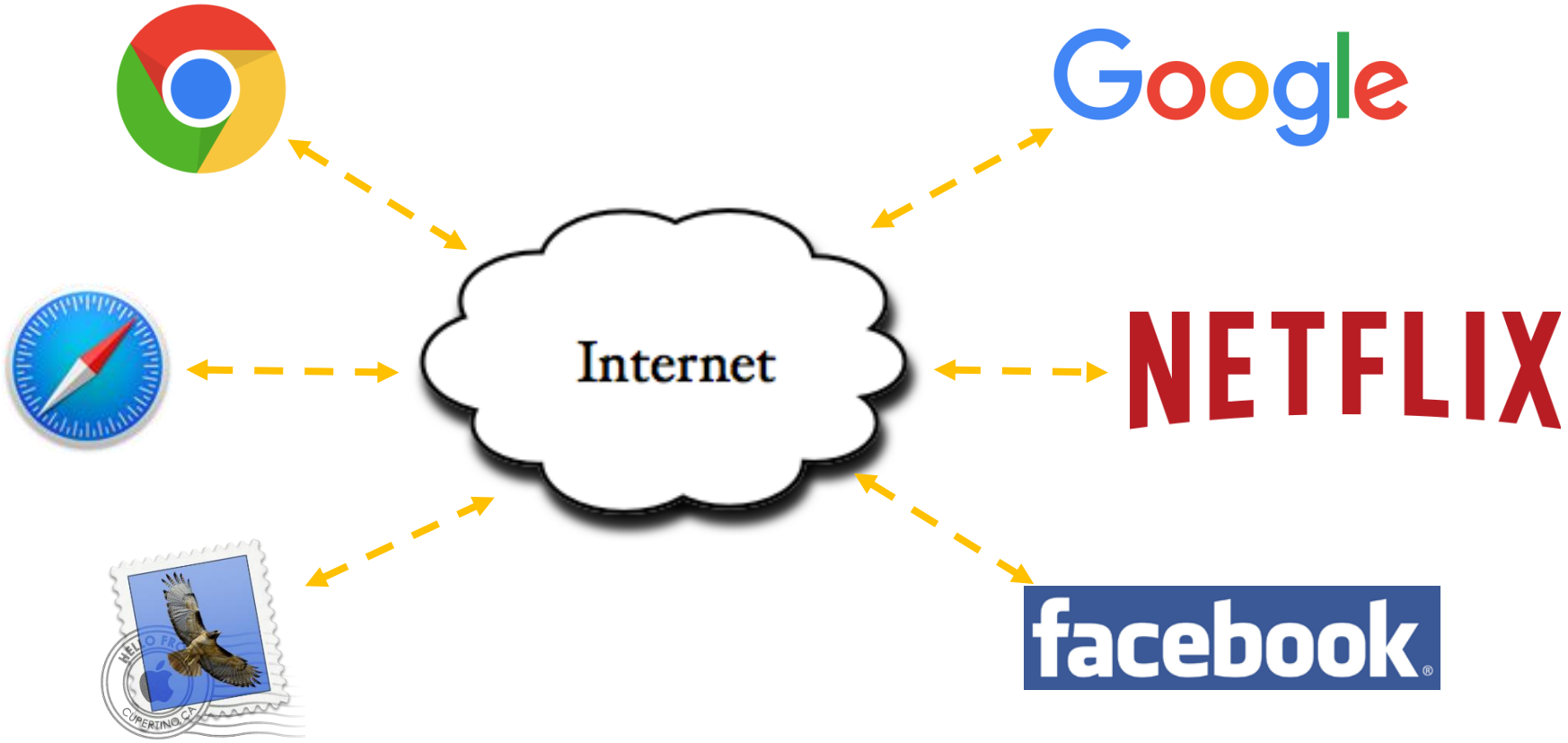
# Today's Goals

- Networking is a very common programming feature
  - You will likely have to create a program that will read/write over the network at some point in your career

- We want to give you a basic, high-level understanding of how networks work before you use them
  - Take CIS 5530 if you want to know more about the implementations of networks
    CIS 5050 is another option

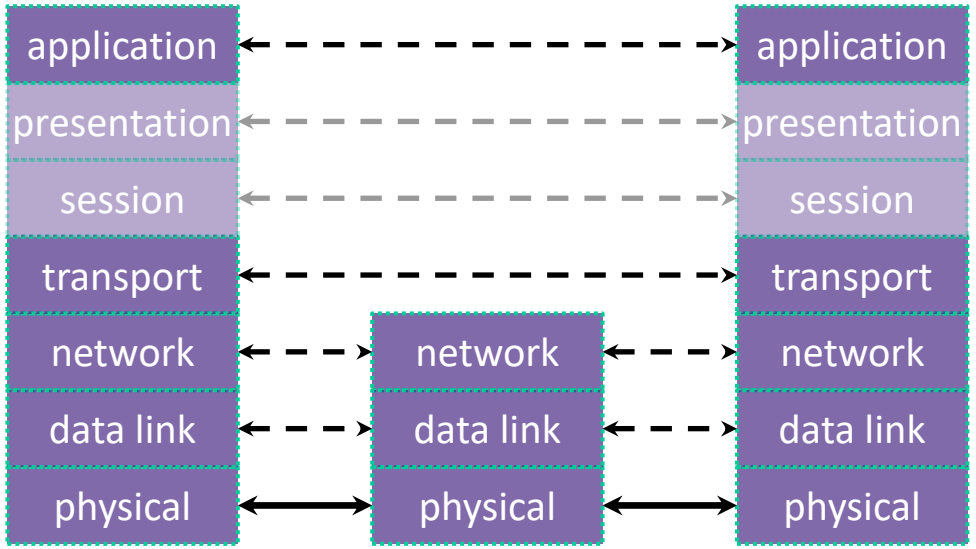
- Less activities
  - Hard to come up with
  - I am sick ☹

# Networks From 10,000 ft



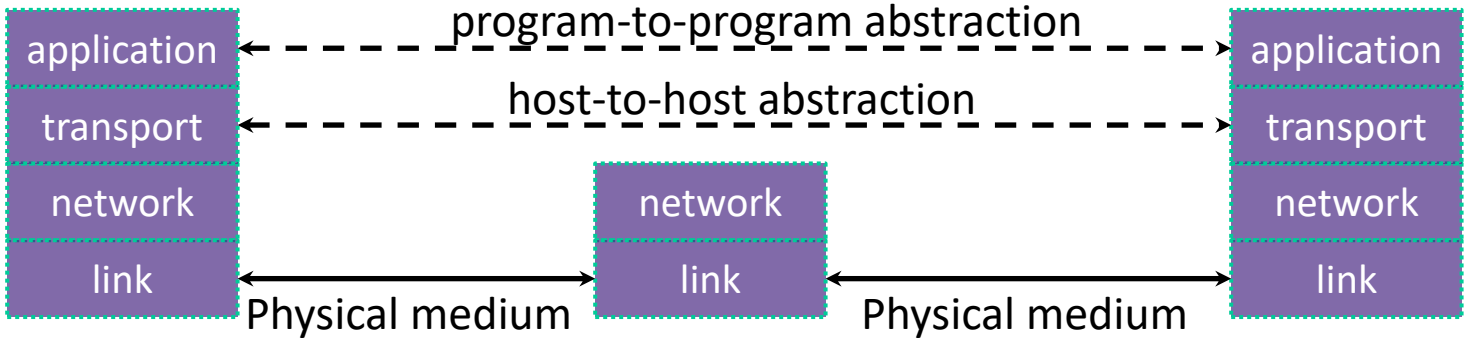
# 7-Layer OSI Model

- The “Full” model for how our network works
  - Developed in early stages of networking
  - Some of the layers don’t exist as much as others



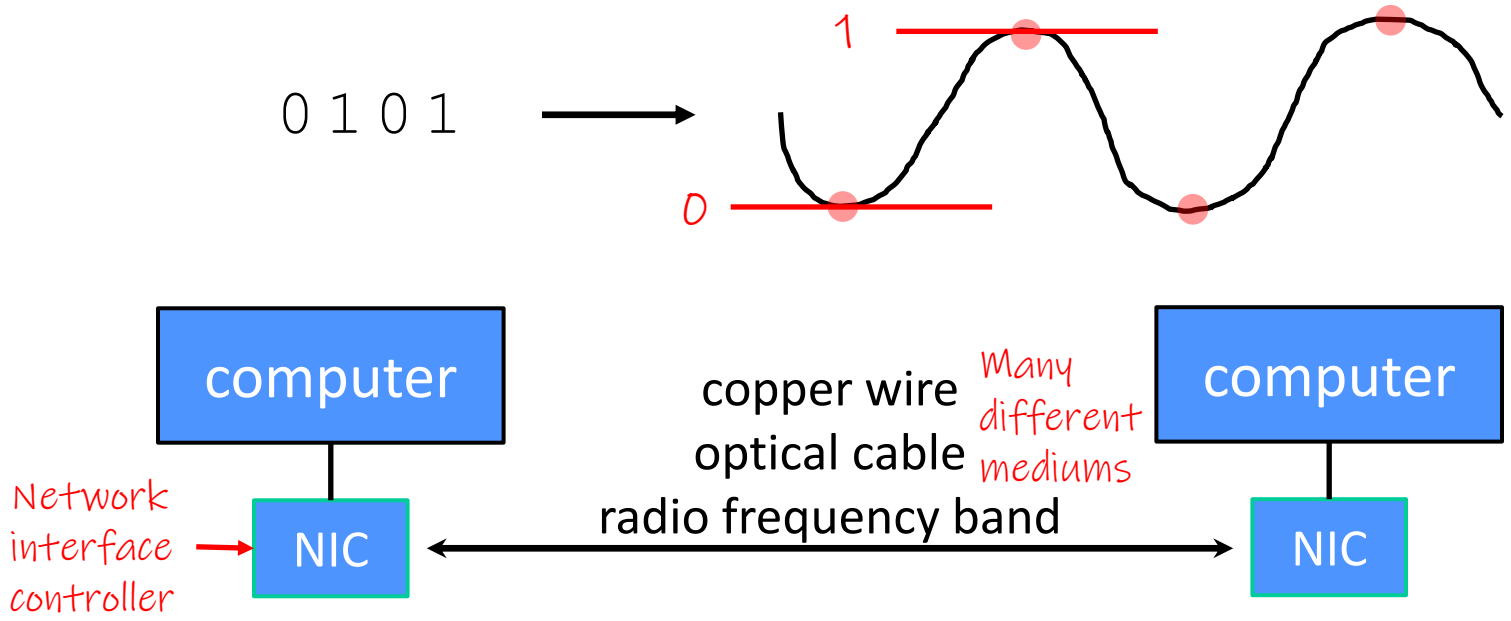
# Internet protocol Suite / TCPIP Model

- Simplified* model developed before the OSI model
  - Less comprehensive
  - Implicitly assumes lower levels are implemented for us
  - Focuses more on software-level concerns
    - We will use this to examine the network (but will go over the lower level briefly first)



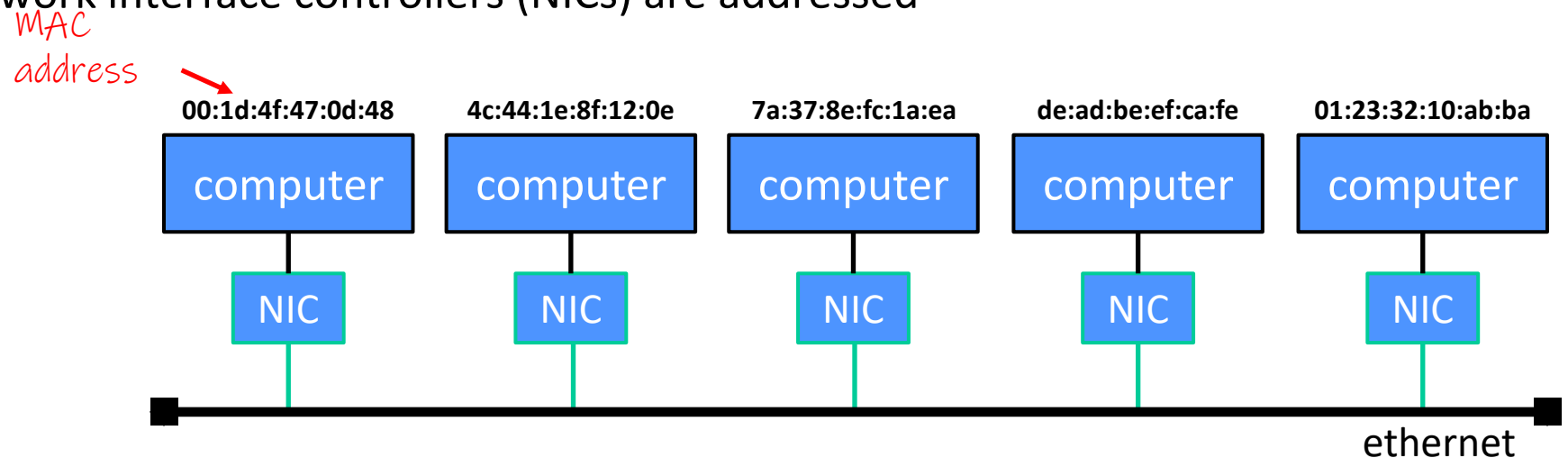
# The Physical Layer

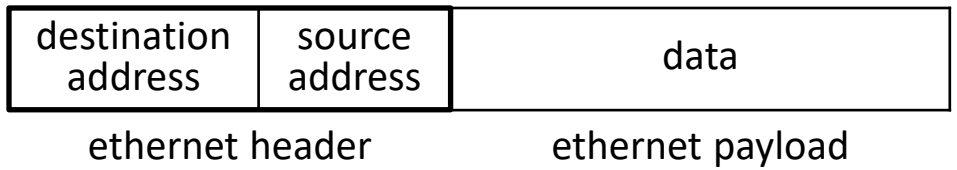
- Individual bits are **modulated onto a wire or transmitted over radio**
  - Physical layer specifies **how bits are encoded at a signal level**
  - Many choices, e.g., encode “1” as +1v, “0” as -0v; or “0”=+1v, “1”=-1v, …

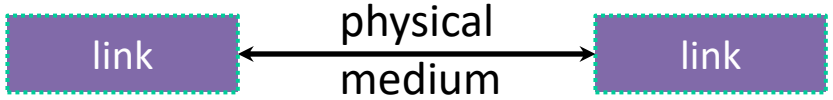


# The Data Link Layer

- Multiple computers on a LAN contend for the network medium
  - Media access control (MAC) specifies **how computers cooperate in a local network**
  - Link layer also specifies **how bits are "packetized"** and network interface controllers (NICs) are addressed



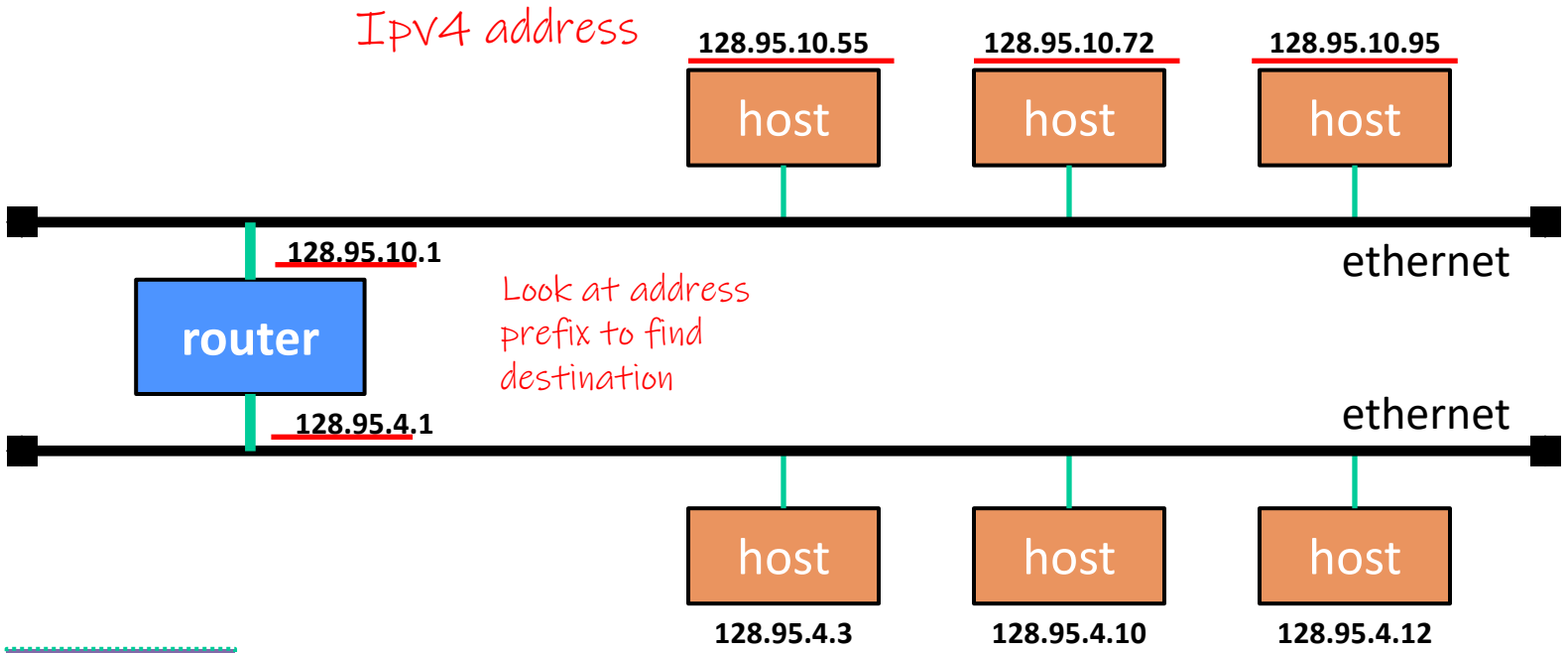



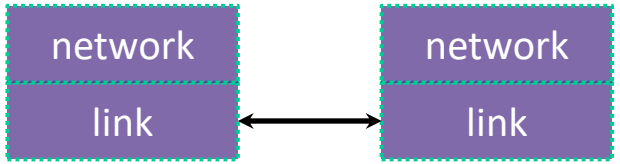
**Discuss & Raise Hands**

- Any guesses for how we get computers to share the same physical medium?
  - (Hint: how do we handle people sharing time in a conversation?)

# The Network Layer (IP)

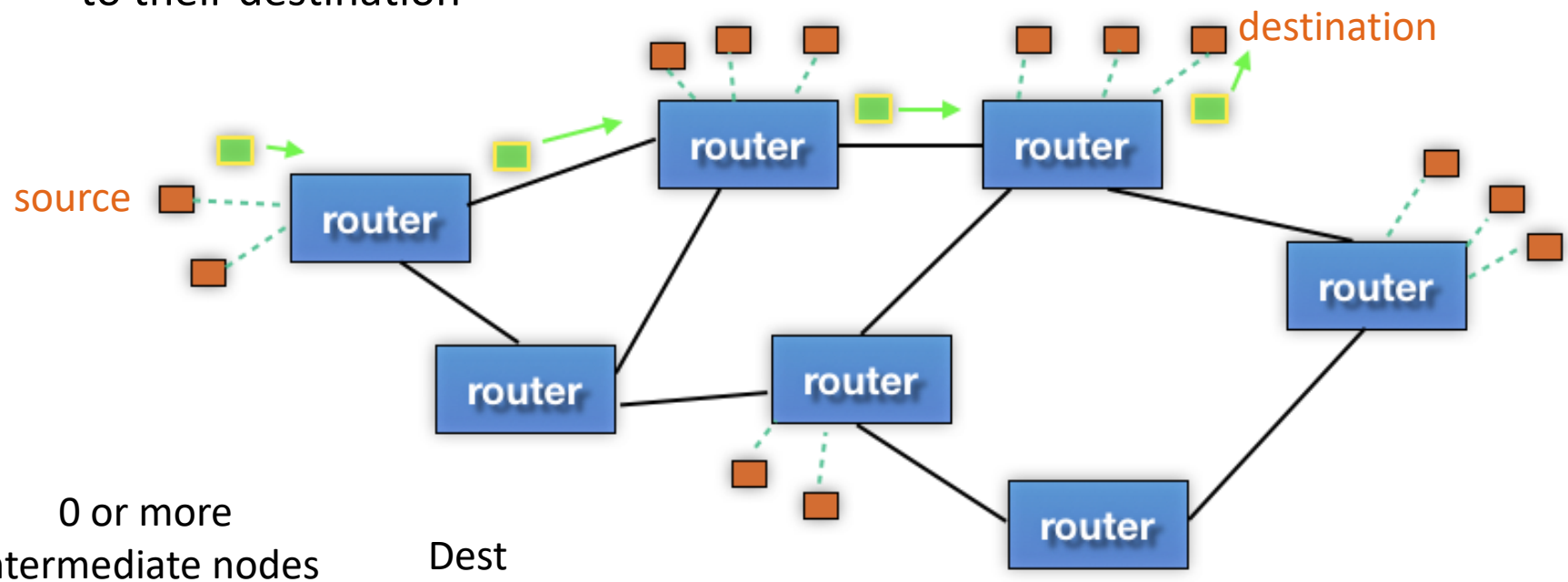
- Internet Protocol (IP) **routes packets across multiple networks**
  - Every computer has a unique IP address*
  - Individual networks are connected by routers that span networks

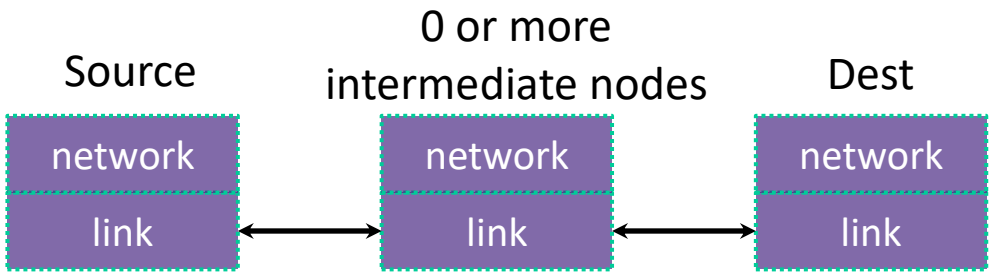


# The Network Layer (IP)

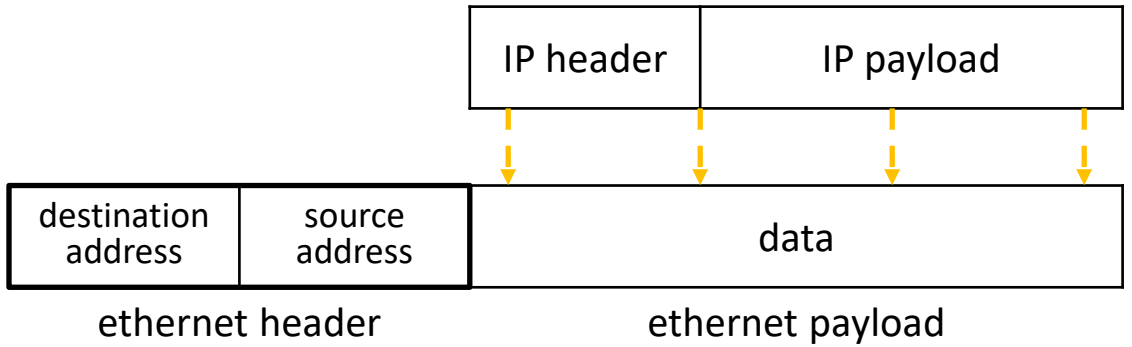
- There are protocols to:
  - Let a host map an IP to MAC address on the same network
  - Let a router learn about other routers to get IP packets one step closer to their destination



# The Network Layer (IP)

- Packet encapsulation:
  - An IP packet is encapsulated as the payload of an Ethernet frame
  - As IP packets traverse networks, routers pull out the IP packet from an Ethernet frame and plunk it into a new one on the next network



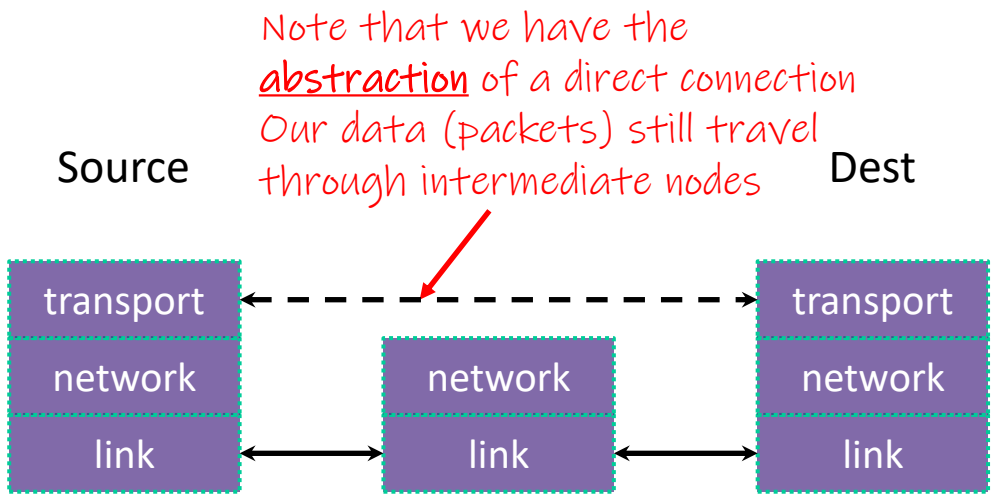
## Discuss & Raise Hands

- If I want to send my friend (in another country) a postcard, how does it get there?
  - What do I need to do?
    - Do I just put my friends name on it and put in a mailbox?
  - How does the post office deliver it?
    - Does one person drive it from my house to their house directly?
    - How does the post office know where to send it?

# The Transport Layer

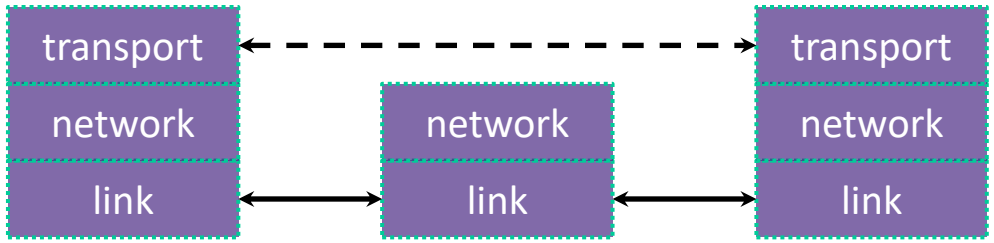
- Provides different protocols to interface between source and destination:
  - *e.g.*, Transmission Control Protocol (TCP), User Datagram Protocol (UDP)
  - These protocols still work with packets, but manages their order, reliability, multiple applications using the network…

- Some protocols at this layer (TCP) provide an interface to treat the network as a *data stream*



# The Transport Layer (TCP)

- Transmission Control Protocol (TCP):
  - Provides applications with reliable, ordered, congestion-controlled byte streams
    - Sends stream data as multiple IP packets (differentiated by sequence numbers) and retransmits them as necessary
    - When receiving, puts packets back in order and detects missing packets
  - A single host (IP address) can have up to $2^{16} = 65{,}535$ "ports"
    - Kind of like an apartment number at a postal address (your applications are the residents who get mail sent to an apt. #)

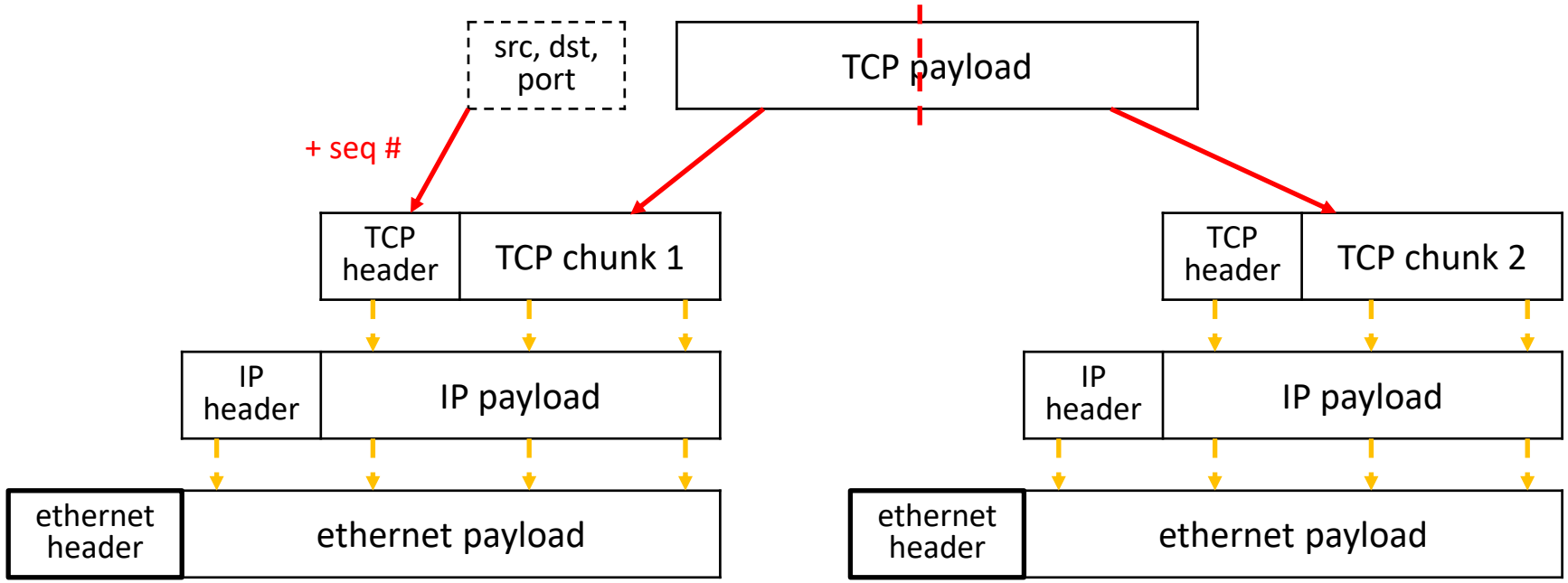


**Discuss & Raise Hands**

- Let's say we want to send a book to a friend in another state, but we can only send post cards (small pieces of paper)
  - How do we send the data to our friend?
  - How do we ensure the data gets to our friend?

# The Transport Layer (TCP)

- Packet encapsulation – one more nested layer!
- Data we send over TCP is broken up for us into packets of data that can travel over the network.
- These packets are “combined” back together at the destination! User doesn’t worry about how packets are broken-up or combined back together.



# The Transport Layer (TCP)

- Applications use OS services to establish TCP streams:
  - The "Berkeley sockets" API (Part of POSIX on linux)
    - A set of OS system calls
  - Clients `connect()` to a server IP address + application port number
  - Servers `listen()` for and `accept()` client connections
  - Clients and servers `read()` and `write()` data to each other

Used same as in File I/O

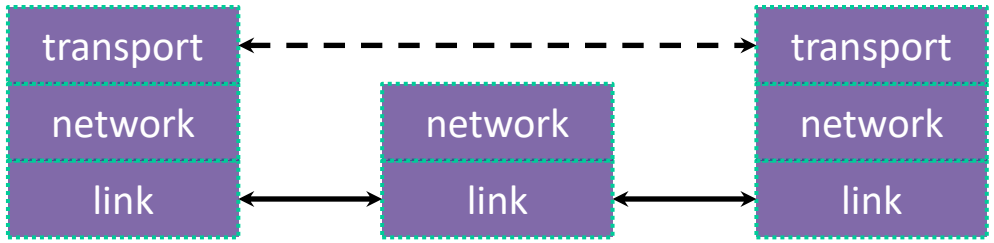
    - Also has functions called `rcv()` and `send()` which do almost the same thing

  - The interface is written in C! Not C++!
    - Does C++ have a std::networking module?
    - No! Why would they add something useful to the std library without arguing about it forever!
    - In this class we will use a thin wrapper for it that I wrote in the style of the Rust net module.

# The Transport Layer (UDP)

- User Datagram Protocol (UDP):
  - Provides applications with *unreliable* packet delivery
  - UDP is a really thin, simple layer on top of IP
    - Datagrams are still sent in IP packets
    - By default send() will fail if you try to write something that won't fit in a single packet. (this size is called the MTU -> Maximum Transmission Unit)
  - Very commonly is not used directly, instead people build their own "protocol" on top of UDP. Examples: QUIC, GameNetworkingSockets

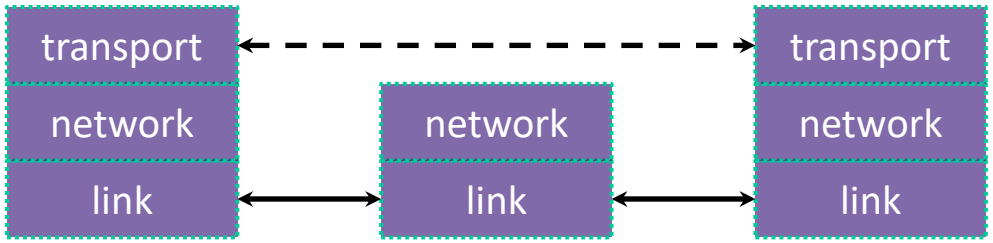
Ok when we want speed.
(VOIP or ZOOM)



# The Transport Layer

## TCP:


## UDP:

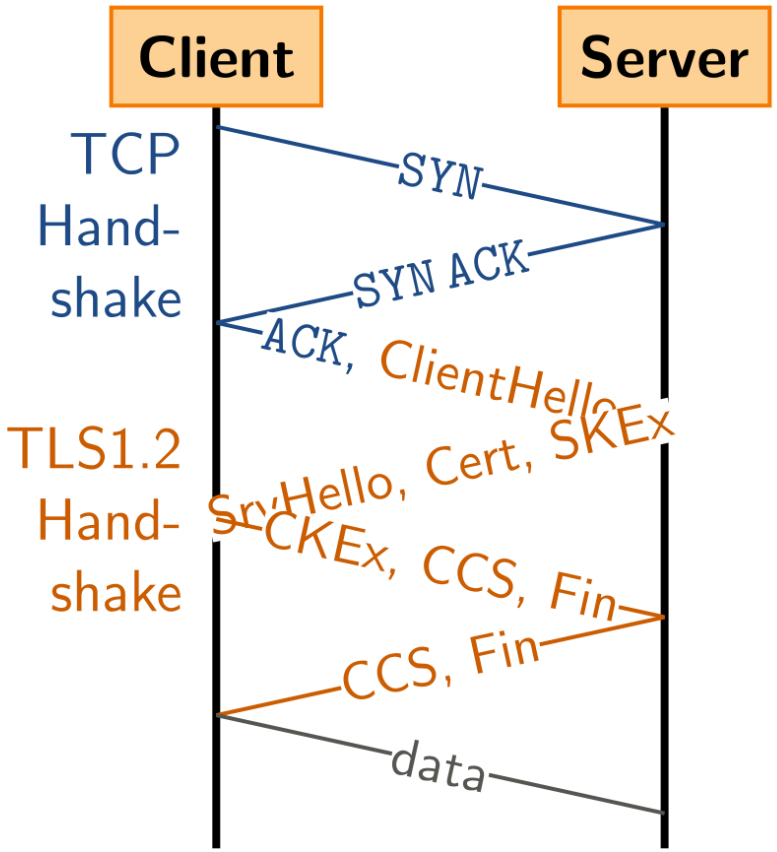




In reality, TCP goes back and forth 3 times before to make sure:

"Hey, do you want to share popcorn and are you who I think you are"

More on these on Wednesday (after the next slide)

# TCP Overhead




- Setting up a TCP connection typically requires **3 round trips**
  - (which is a relatively long time)
- If a packet in a sequence is dropped, then the "Stream" must wait to recover that packet before it can process other things in the stream.
- Solution: QUIC

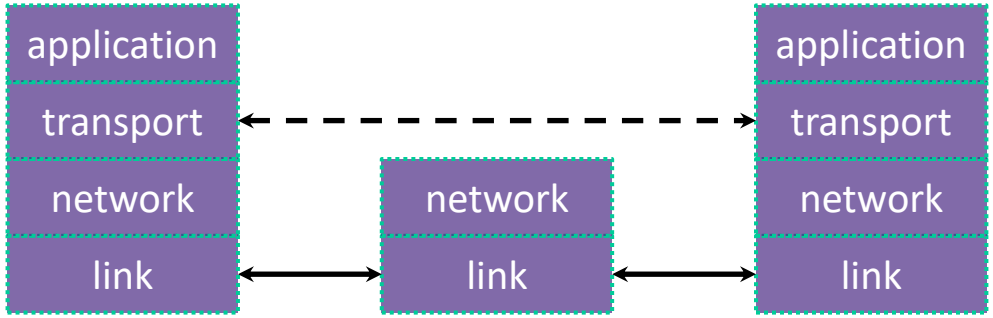
By Sedrubal - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=114587250


pollev.com/tqm

- Can we guarantee that data gets sent reliably from one computer to another?
  - If so, how?

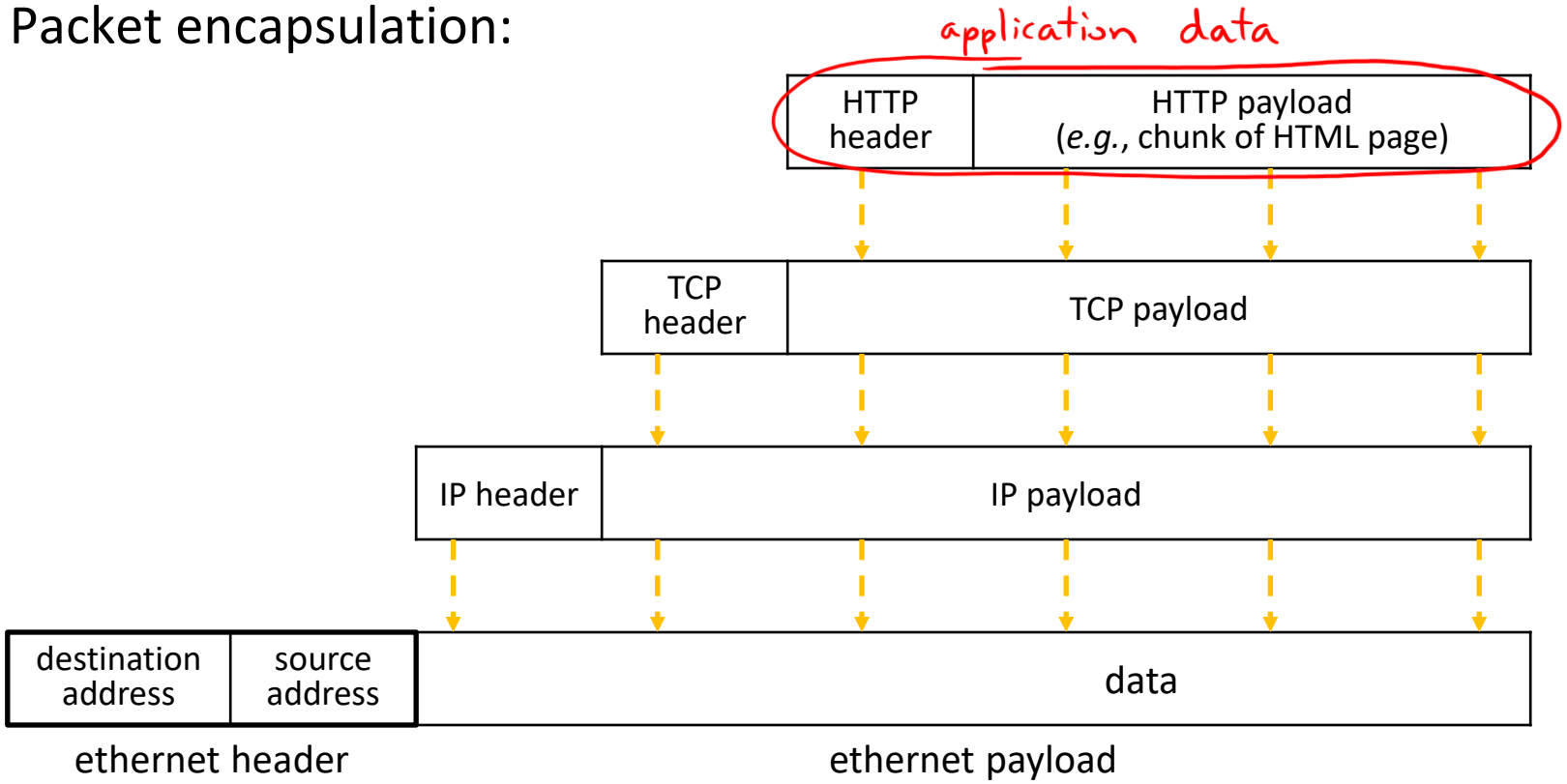
# The Application Layer

- Application protocols
  - The format and meaning of messages between application entities
  - *e.g.*, HTTP is an application-level protocol that dictates how web browsers and web servers communicate
    - HTTP is implemented *on top of* TCP streams



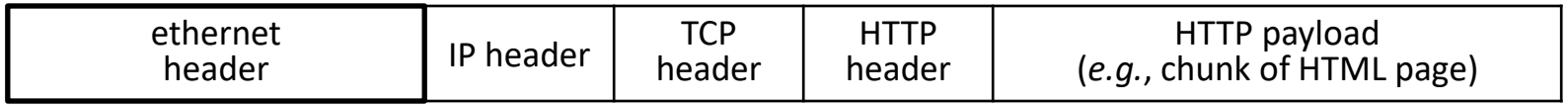
# The Application Layer

- Packet encapsulation:



# The Application Layer

- Packet encapsulation:



# The Application Layer

- Popular application-level protocols:
  - **DNS:** translates a domain name (*e.g.*, www.google.com) into one or more IP addresses (*e.g.*, 74.125.197.106)
    - Domain Name System
    - An hierarchy of DNS servers cooperate to do this
  - **HTTP:** web protocols
    - Hypertext Transfer Protocol
  - **SMTP, IMAP, POP:** mail delivery and access protocols
    - Secure Mail Transfer Protocol, Internet Message Access Protocol, Post Office Protocol
  - **SSH:** secure remote login protocol
    - Secure Shell
  - **bittorrent:** peer-to-peer, swarming file sharing protocol

# netcat demo (if time)

- netcat (`nc`) is "a computer networking utility for reading from and writing to network connections using TCP or UDP"
  - https://en.wikipedia.org/wiki/Netcat

  - Listen on port: `nc -l <port>`
  - Connect: `nc <IPaddr> <port>`
    - Local host: `127.0.0.1`

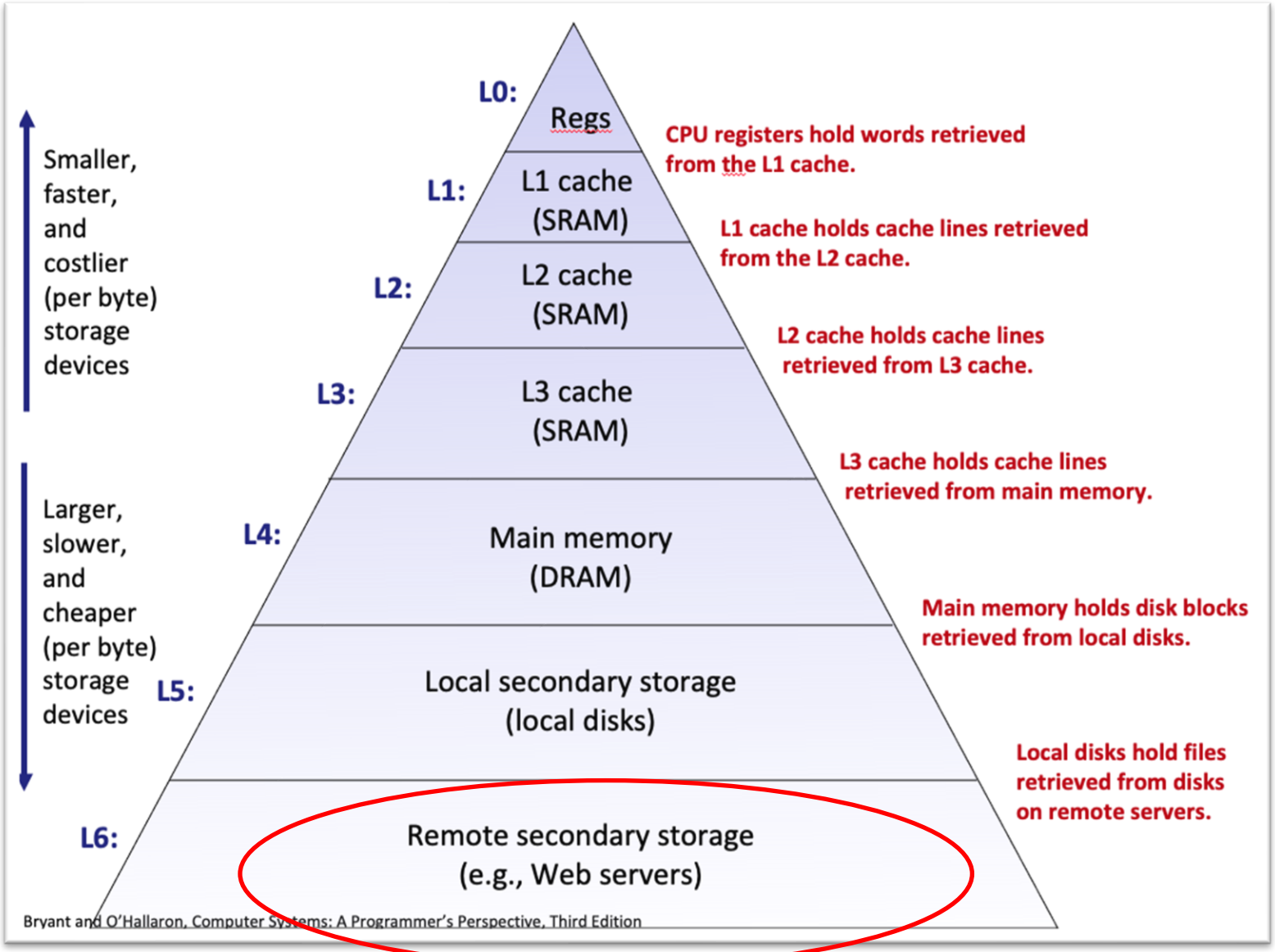
**Discuss & Raise Hands**

- Which of these are you familiar with? Do you know what they are?
  - HTML
  - HTTP
  - TCP
  - UDP
  - IP Address
  - Port
  - Mac Address

# Lecture outline

- Introduction to Networks
  - OSI model & TCPIP model
    - Layers upon layers upon layers…
  - Physical & Data Link Layer
  - IP Layer
  - Transport Layer
  - Application Layer
- **Analyzing this model**

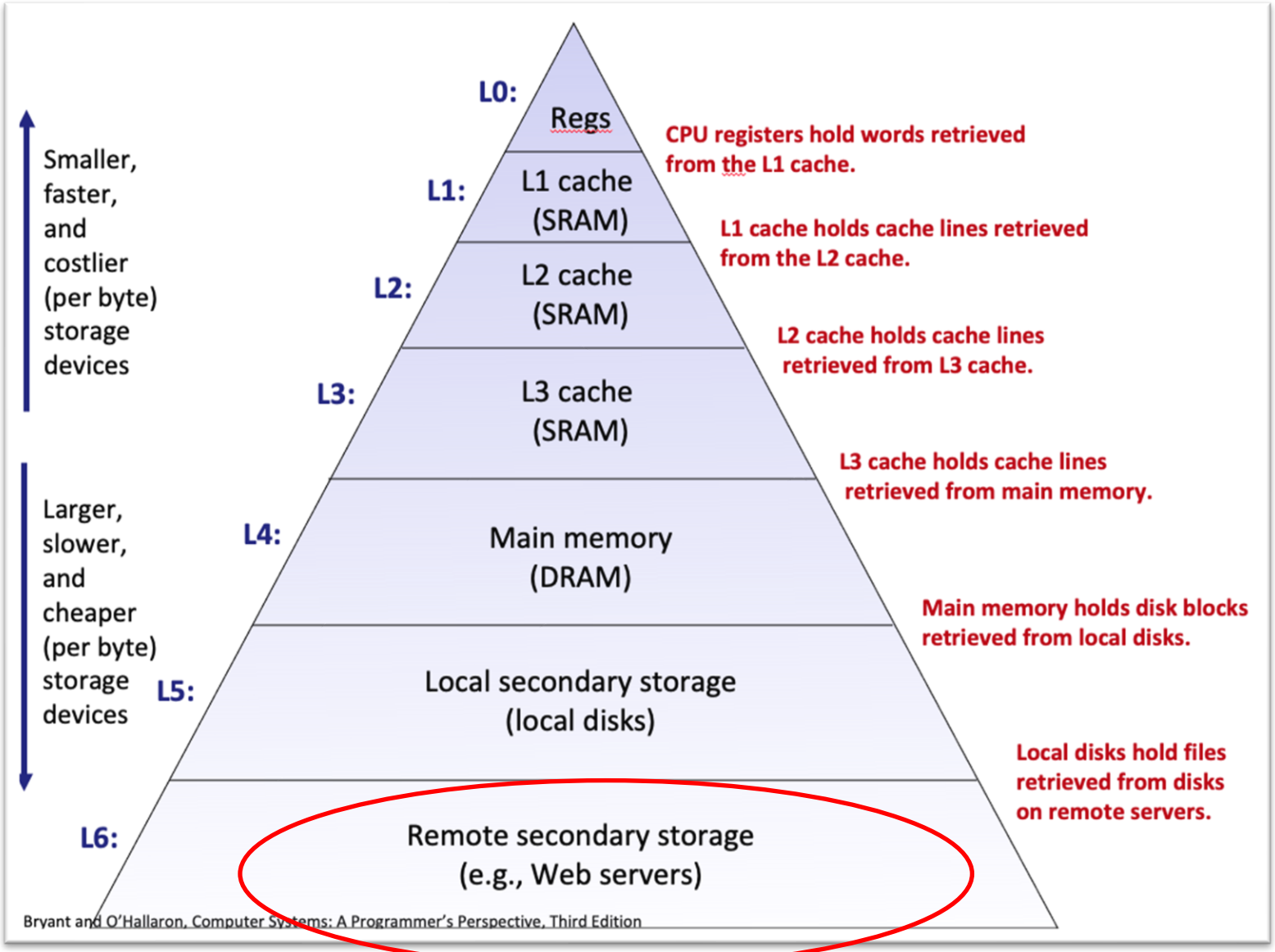
# Memory Hierarchy (again)



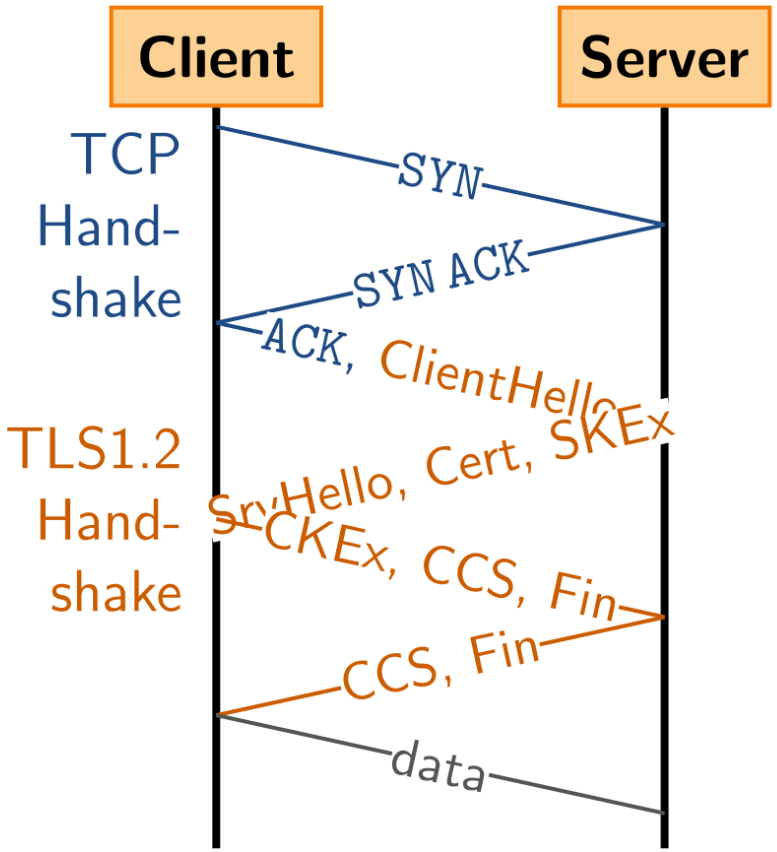
# Materials Matter – Latency

- Fiber optic cables are lower-latency and higher-bandwidth than traditional copper wiring
  - Much of the internet's "long haul" data is transmitted on these
  - (signal attenuation is much better too)

# Memory Hierarchy (again)



# TCP Overhead

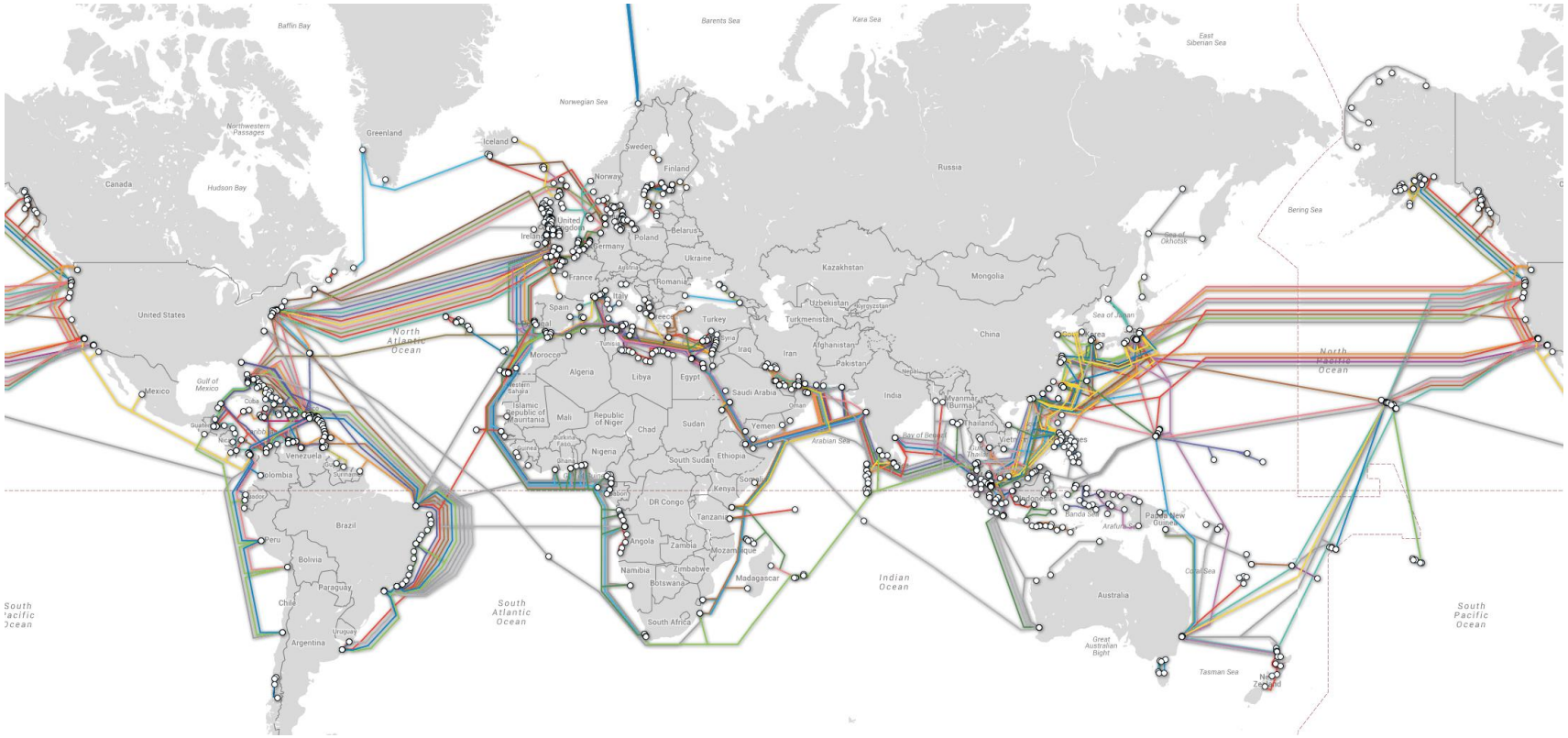



- Setting up a TCP connection typically requires **3 round trips**
  - (which is a relatively long time)
- If a packet in a sequence is dropped, then the "Stream" must wait to recover that packet before it can process other things in the stream.
- Solution: QUIC

By Sedrubal - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=114587250

# Reliability

- Packet loss?
- Physical interference?
- Link going down?

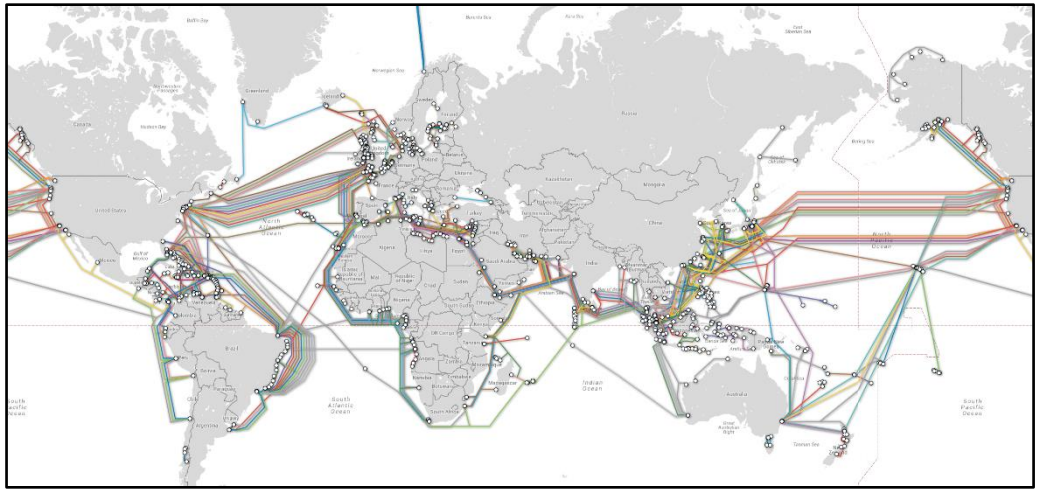


**Discuss & Raise Hands**

- If you had to guess a place on the map with higher internet speeds, where would you guess?
  - If you had to guess a place with slower internet speeds?

# Topology Matters – Latency and Reliability

- Some places are surprisingly well- or poorly-connected to “backbone” infrastructure like fiber optic cables
- Unintuitive topology can create interesting failures
  - *e.g.*, 2006 7.0-magnitude Hengchun Earthquake disrupted communications to Singapore, Philippines, Thailand, China, etc. for a month

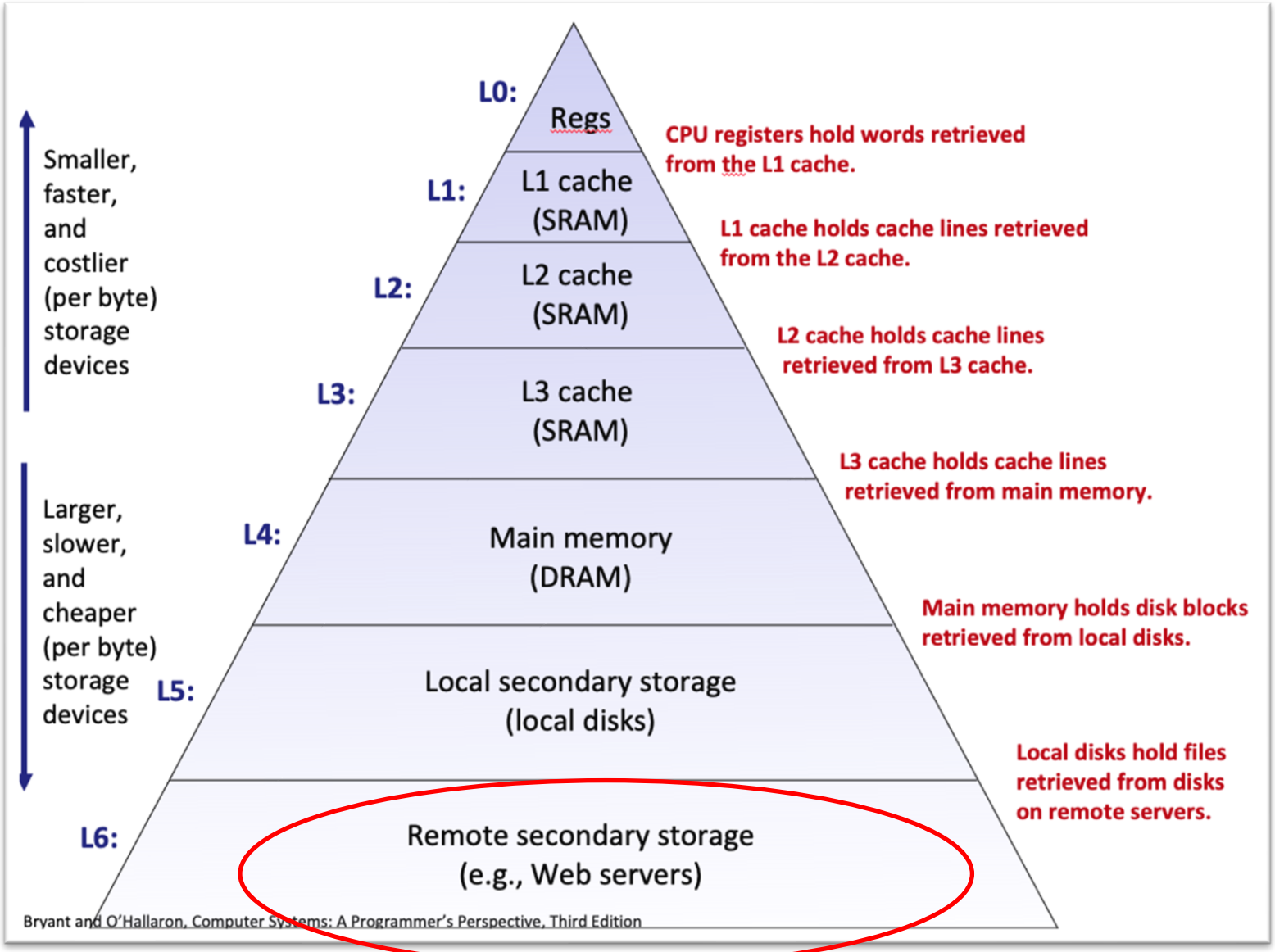



# Distance Matters – Latency

- Distances within a single datacenter are smaller than distances across continents
- Even within a datacenter, distances can sometimes matter

123Net Data Center, Wikimedia

# Memory Hierarchy (again)



# Next Lecture

- Socket Programming!