CSE333 Systems Programming

Networking: OSI Model, Sockets Intro

Instructors:

Naomi Alterman

Teaching Assistants:

Ann Baturytski	Barbora Buzkova	Mendel Carroll
Camden Harris	Hannah Hempstead	Rishabh Jain
Irene Lau	Jonathan Nister	Advay Patil
Aviel Wood	Angela Wu	Jennifer Xu

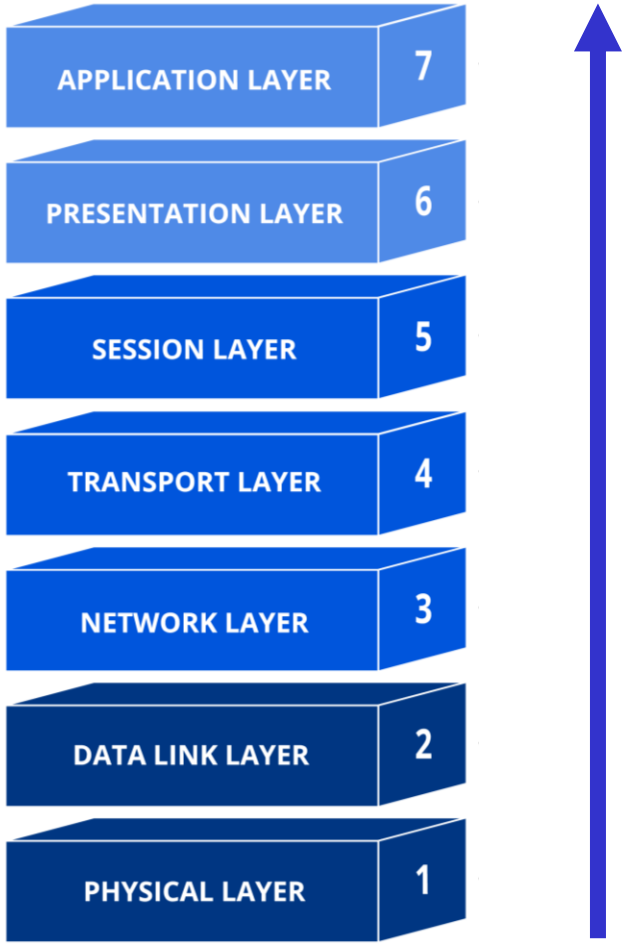
Administrivia

- Ex9 due Wednesday @ 11 **am**
- HW3 due Thursday @ 11:59 **pm**

Lecture Outline

- **Walkthrough of the OSI model**
- The Sockets API

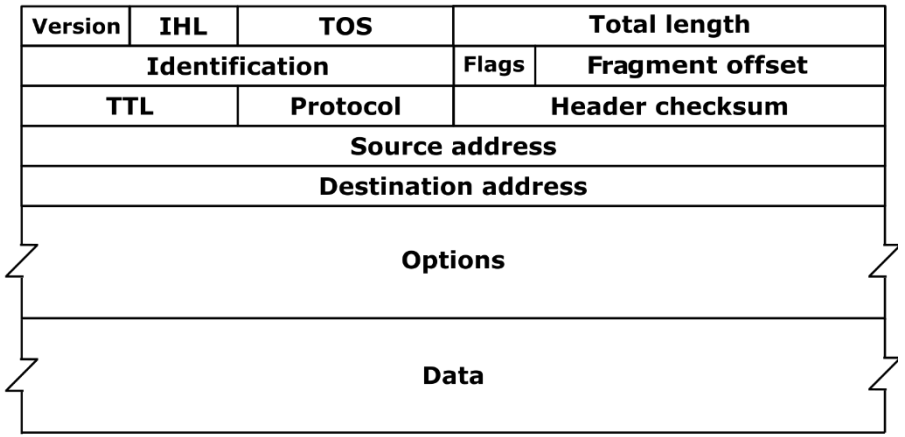
Let's go from the ground up



Protocols and headers

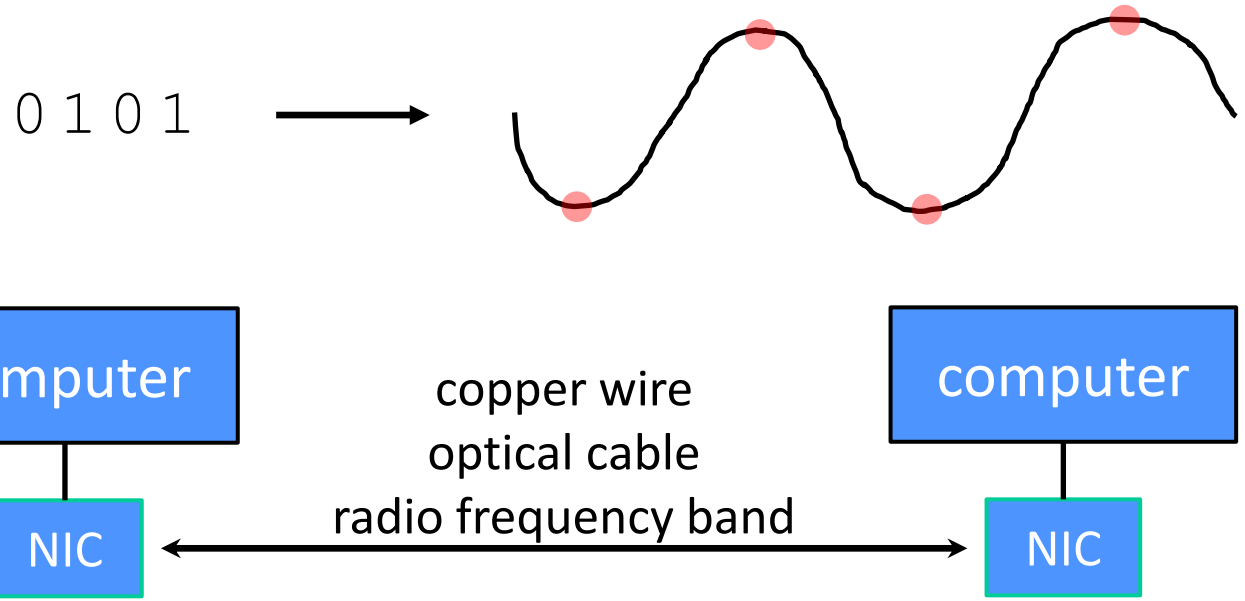
- Your data packet includes some metadata bytes for each layer so the next computer will know what do to with it. These bytes are called **packet headers**.
- We **standardize** those headers, and the behavior of the software reading them, in things called **network protocols**
  - IP! TCP! UDP! Ethernet! ARP! ICMP! RIP! N'Sync! BTS!

(*homework: figure out which of these are networking protocols and which of these are boy bands* 😈)



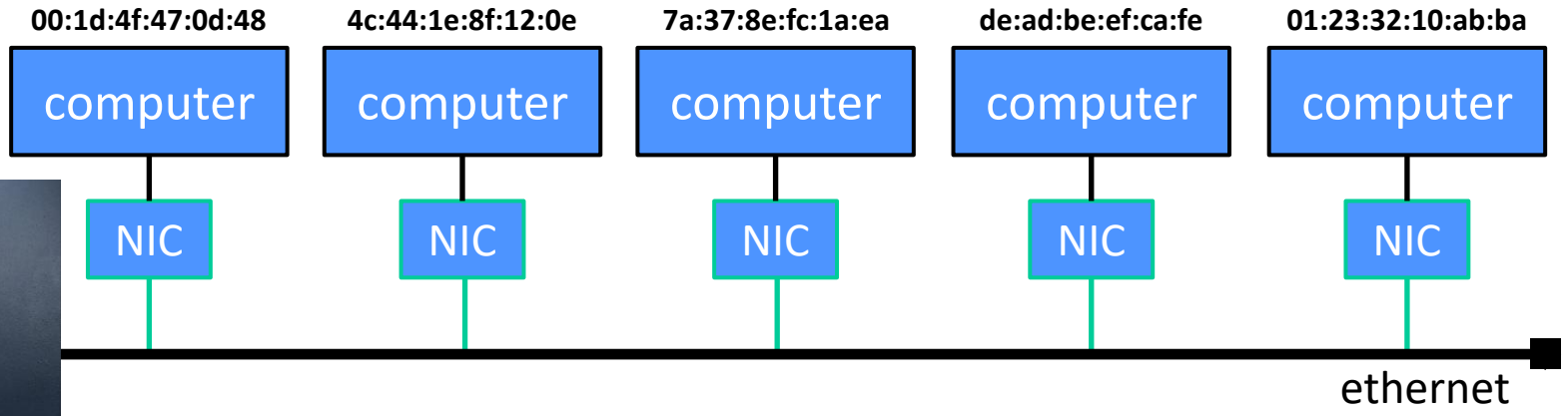
The Physical Layer

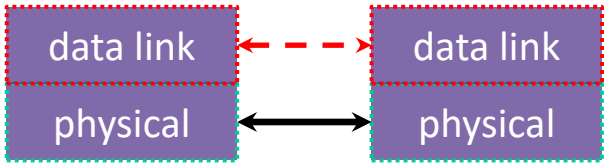
- Individual bits are modulated onto a wire or transmitted over radio
  - Physical layer specifies how bits are encoded at a signal level
  - Many choices, e.g., encode "1" as +1v, "0" as -0v; or "0"=+1v, "1"=-1v, …

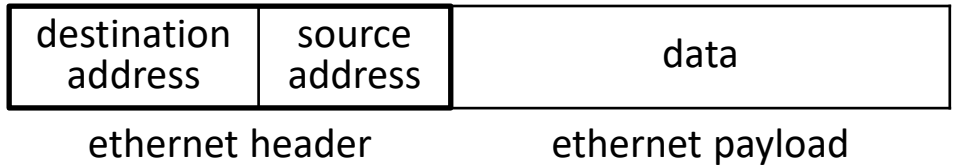


The Data Link Layer

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

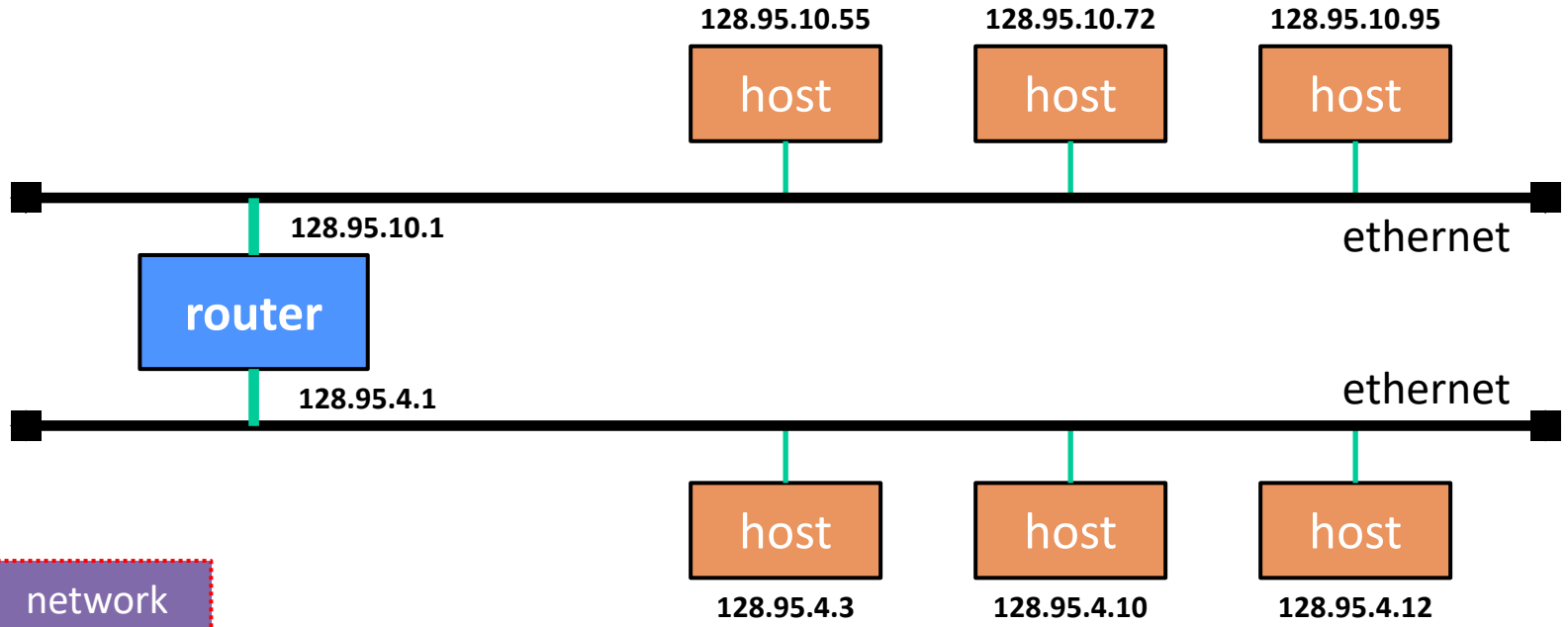


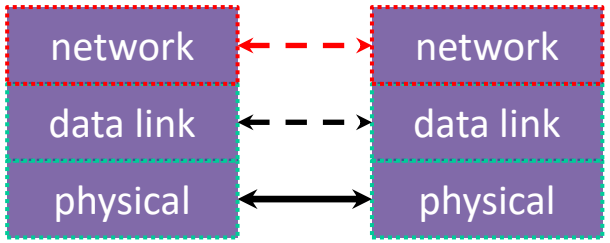




The Network Layer (IP)

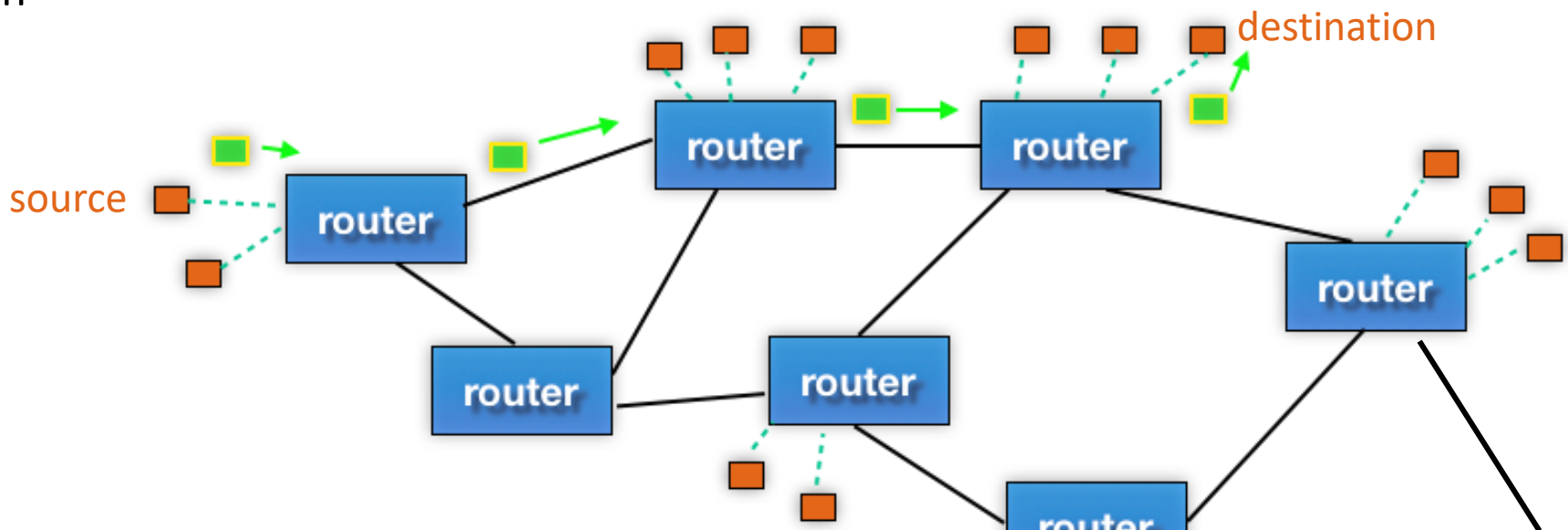
- Internet Protocol (IP) routes packets across multiple networks
  - Every computer has a unique IP address
  - Individual networks are connected by routers that span networks

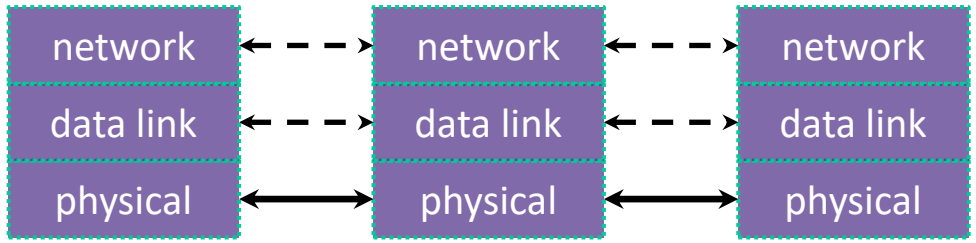




The Network Layer (IP)

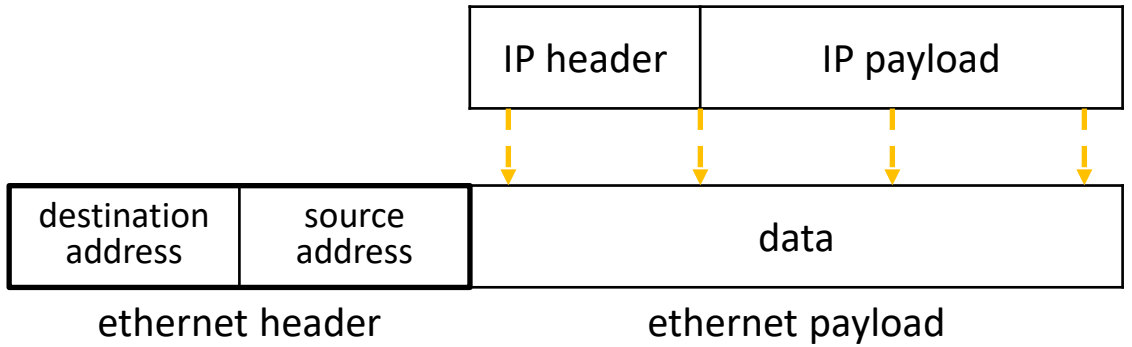
- There are protocols to:
  - Let a host map an IP to MAC address on the same network
  - Let a router learn about other routers to get IP packets one "hop" closer to their destination

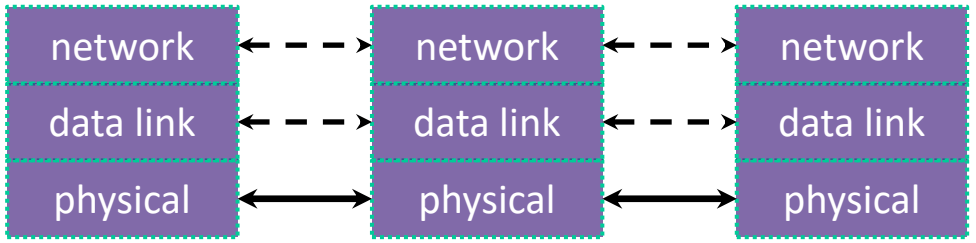




The Network Layer (IP)

- 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





Poll Everywhere

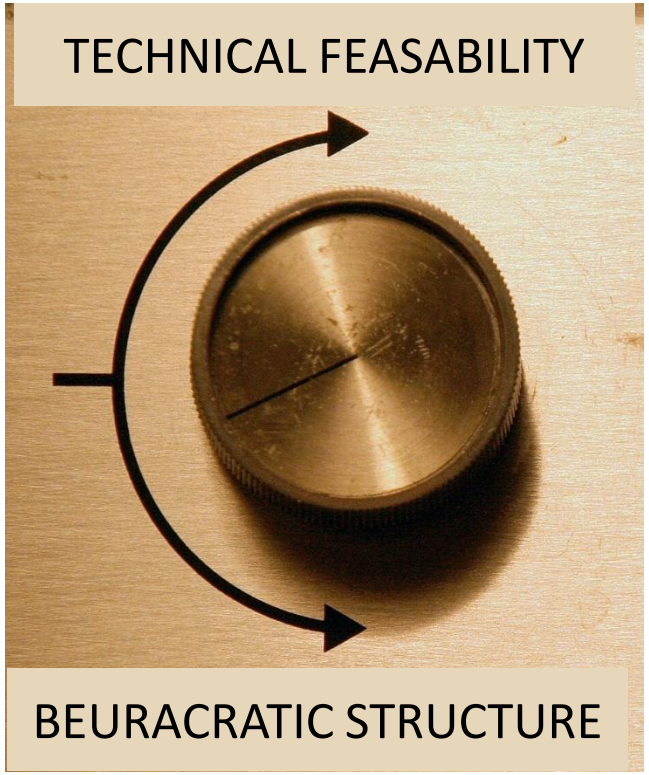
pollev.com/naomila

What's in a name?

- **IPv4 addresses are 32 bits**
  - So we've got **4,294,967,296** unique names
  - …how many computers could there possibly be 🙄 ?

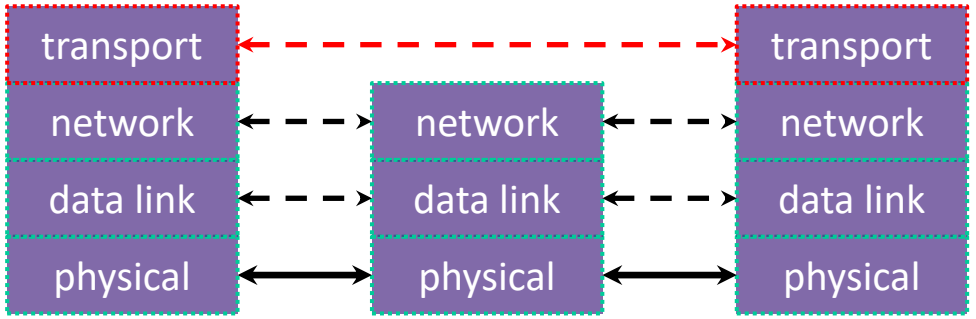
- **Think with me!**
  - What technical considerations could have led to them choosing 32 bit addresses?
  - What are some human (non-technical) consequences of IPv4 address exhaustion?

Engineering tradeoffs



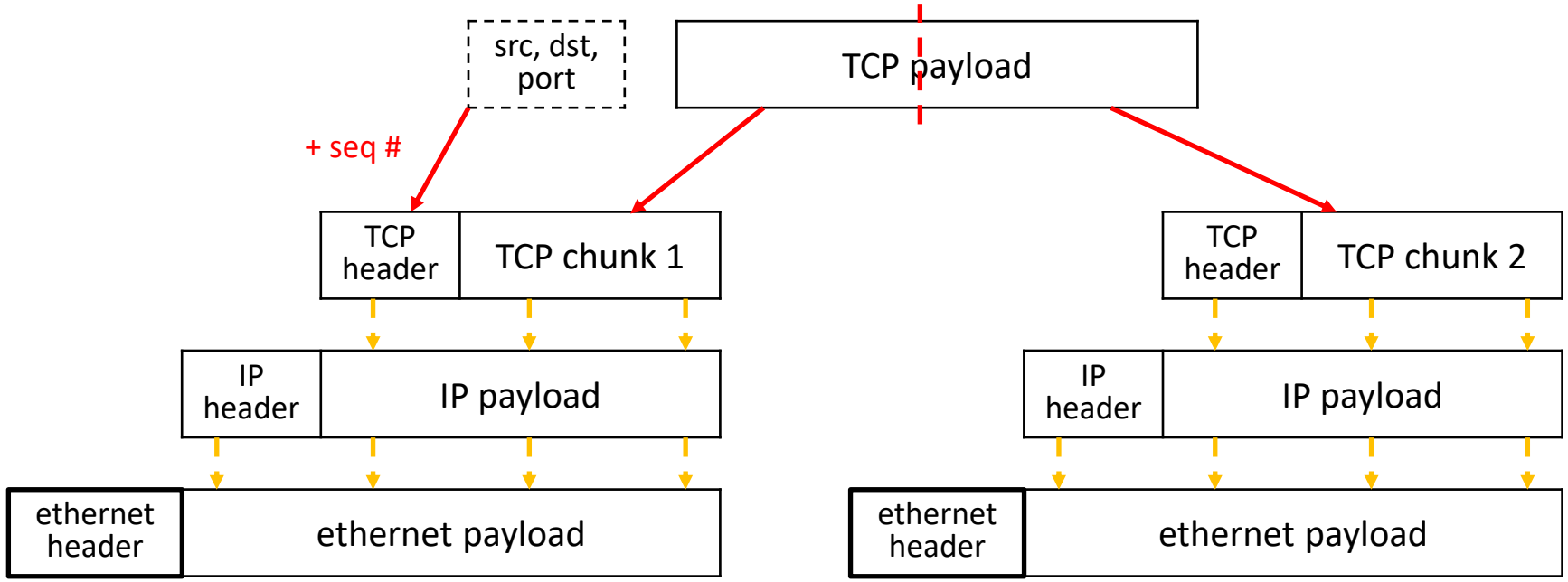
The Transport Layer (TCP)

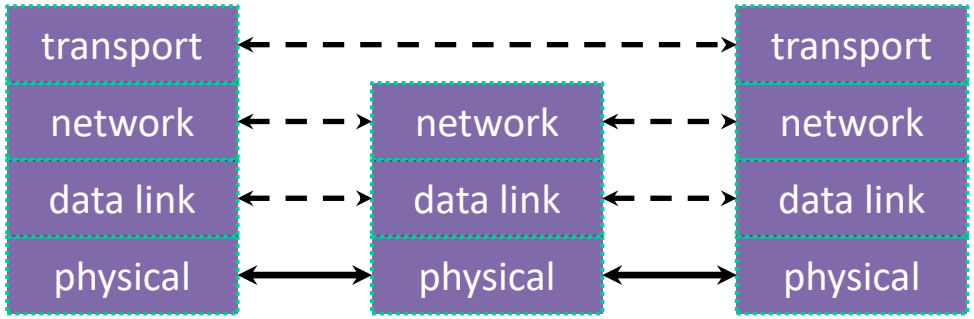
- Transmission Control Protocol (TCP):
  - Provides applications with reliable, ordered, congestion-controlled byte streams
    - Sends stream data as multiple IP packets (differentiated by sequence numbers) and retransmits them as necessary
    - When receiving, puts packets back in order and detects 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. #)



The Transport Layer (TCP)

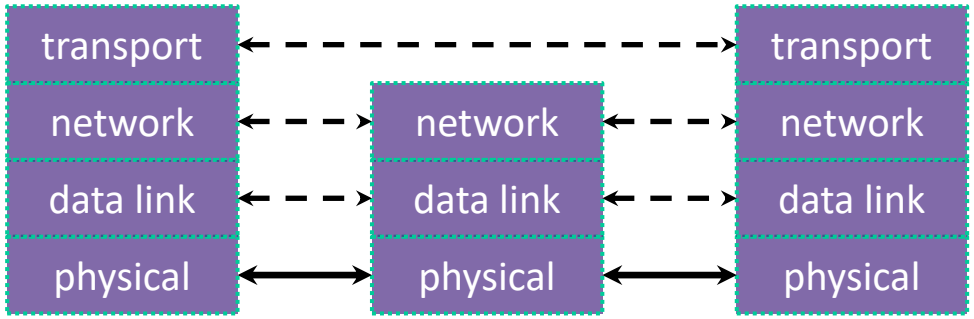
- Packet encapsulation – one more nested layer!





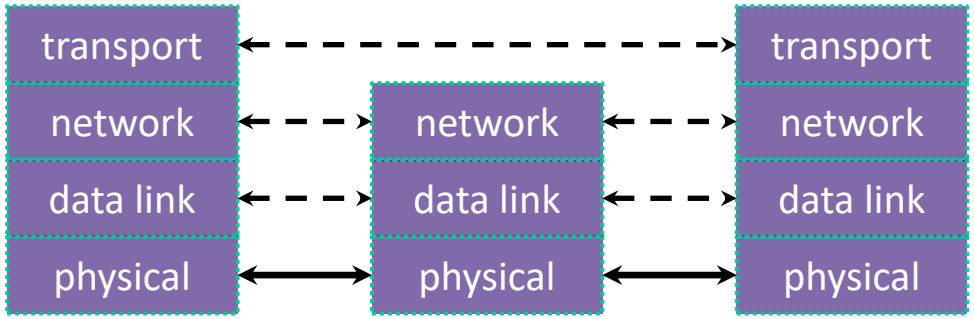
The Transport Layer (TCP)

- Applications use OS services to establish TCP streams:
  - The "Berkeley sockets" API
    - 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



The Transport Layer (UDP)

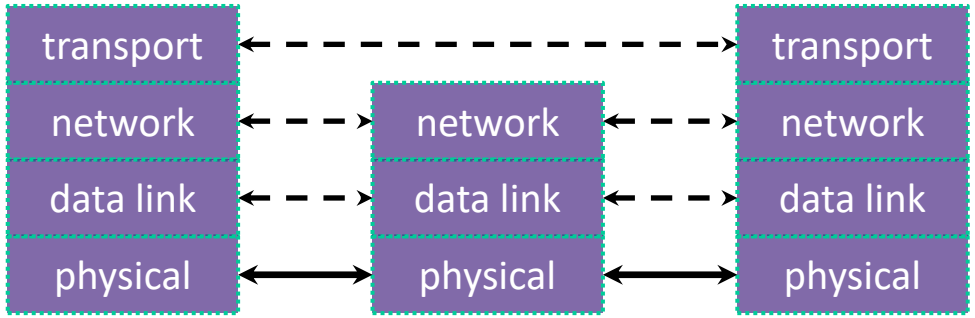
- User Datagram Protocol (UDP):
  - Provides applications with *unreliable* packet delivery
  - UDP is a really thin, simple layer on top of IP
    - Datagrams still are fragmented into multiple IP packets



The Transport Layer

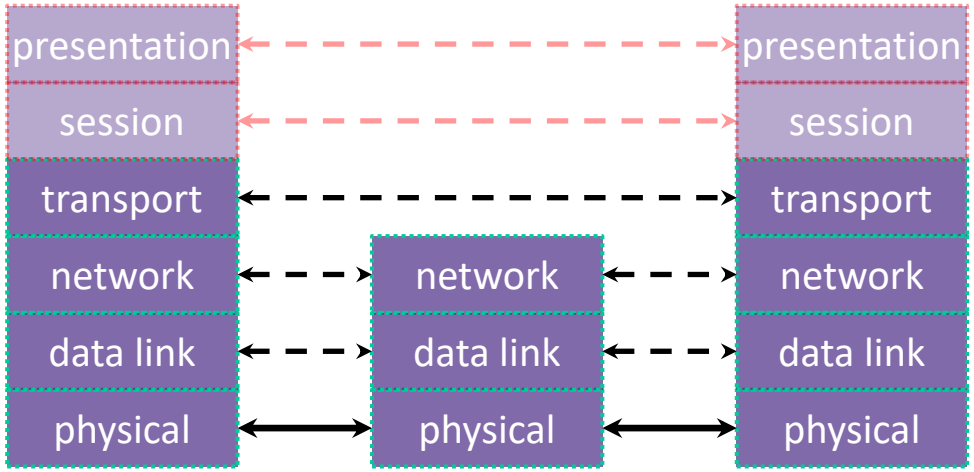
TCP:

UDP:



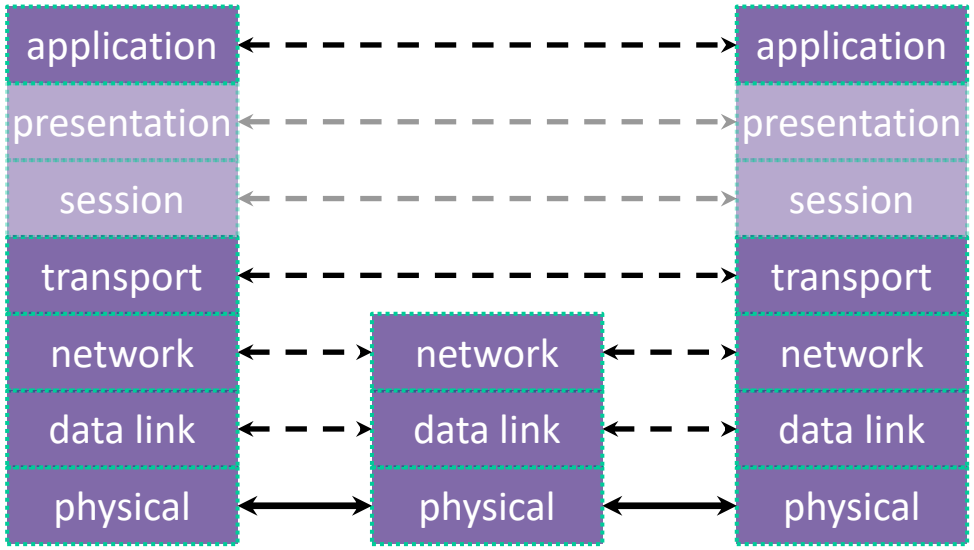
The (Mostly Missing) Layers 5 & 6

- Layer 5: Session Layer
  - Supposedly handles establishing and terminating application sessions
  - Remote Procedure Call (RPC) kind of fits in here
- Layer 6: Presentation Layer
  - Supposedly maps application-specific data units into a more network-neutral representation
  - Encryption (SSL) kind of fits in here



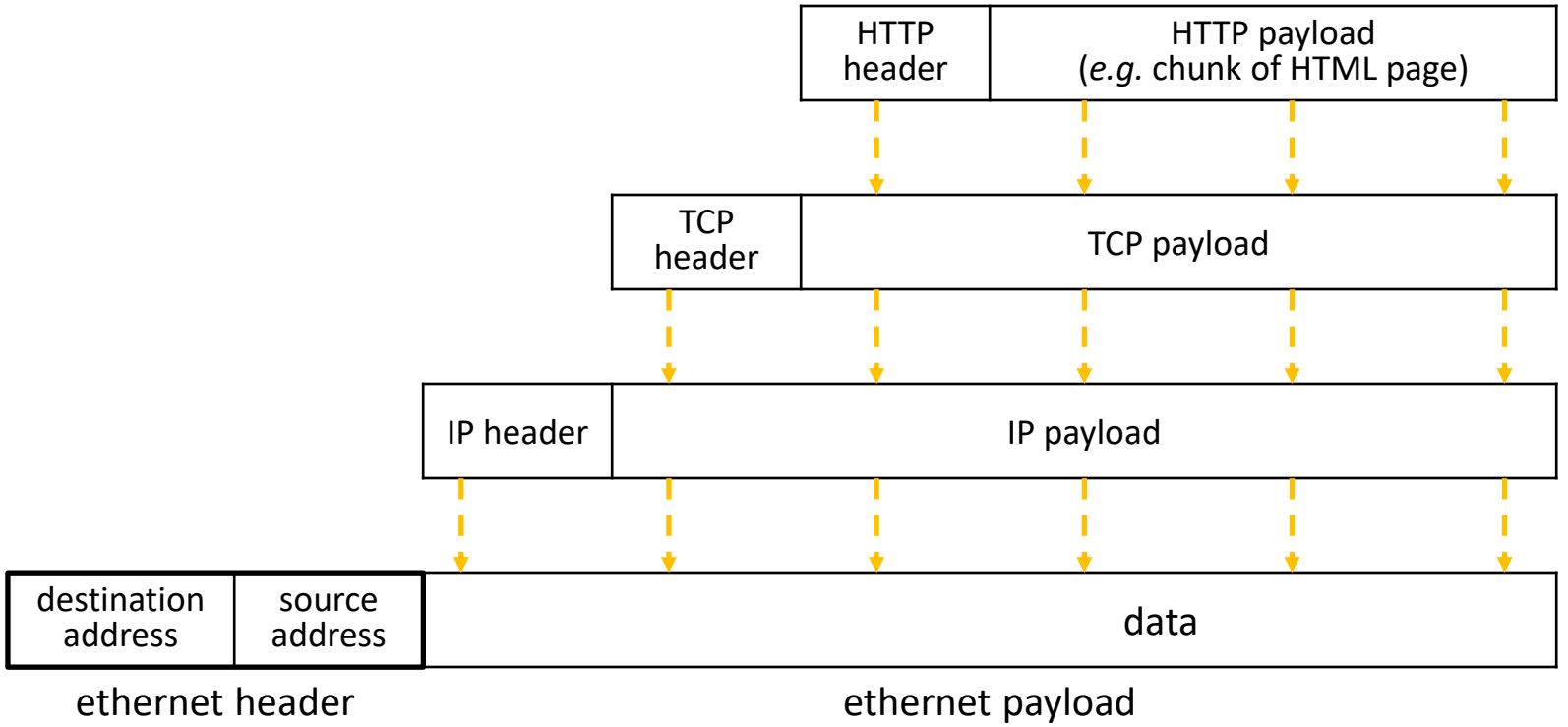
The Application Layer

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



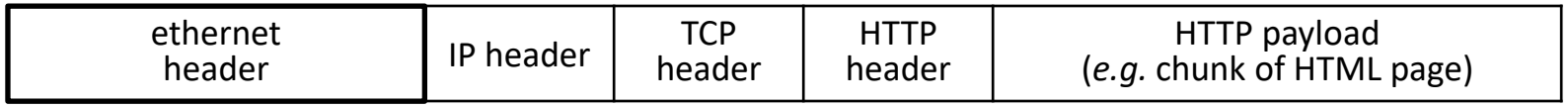
The Application Layer

- Packet encapsulation:



The Application Layer

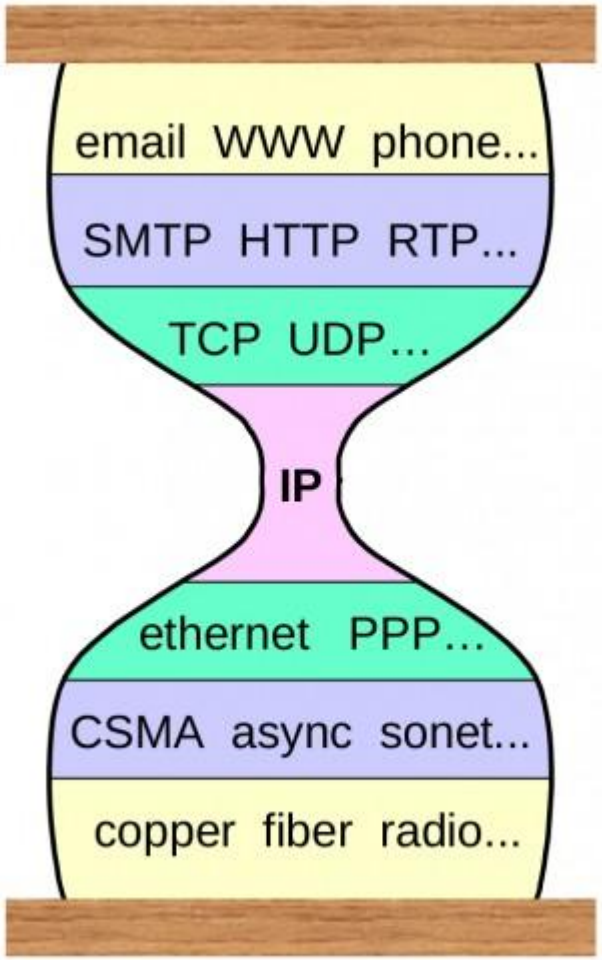
- Packet encapsulation:



The Application Layer

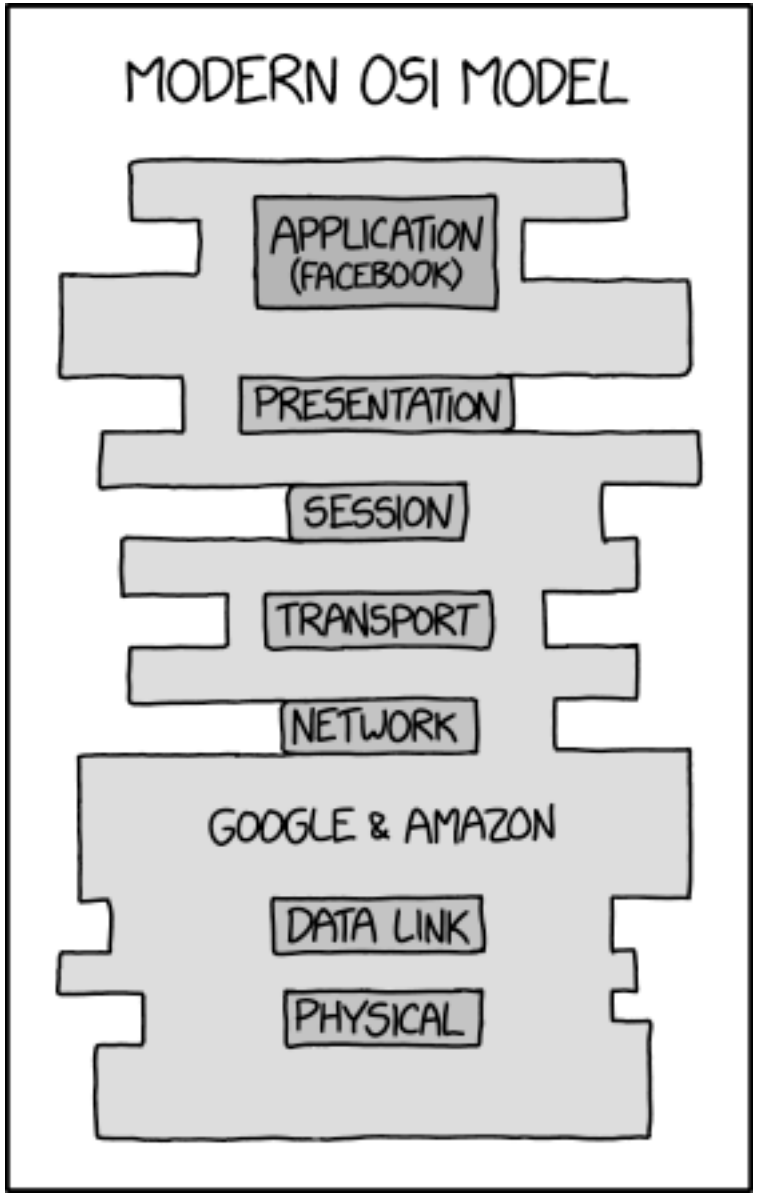
- Popular application-level protocols:
  - **DNS:** translates a domain name (*e.g.* www.google.com) into one or more IP addresses (*e.g.* 74.125.197.106)
    - Domain Name System
    - An hierarchy of DNS servers cooperate to do this
  - **HTTP:** web protocols
    - Hypertext Transfer Protocol
  - **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

The "narrow waist"



- Hahah that's a lot of protocols. How are we gonna do this…
- The bulk of the machinery running the internet **only deals with IP, TCP and UDP** ("TCP/IP")
  - How is the link layer handled? That's a **driver** problem
  - How does Instagram work? That's an **application** problem.
- This is called "**the narrow waist**" of the internet

The Future of Networking?



Lecture Outline

- Walkthrough of the OSI model
- **The Sockets API**

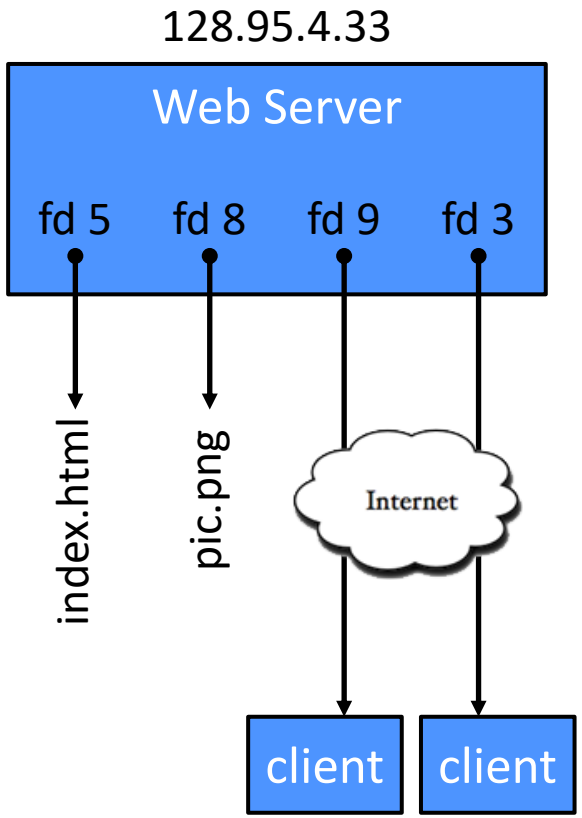
Files and File Descriptors

- Remember `open()`, `read()`, `write()`, and `close()`?
  - POSIX system calls for interacting with files
  - `open()` returns a file descriptor
    - An integer that represents an open file
    - This file descriptor is then passed to `read()`, `write()`, and `close()`
  - Inside the OS, the file descriptor is used to index into a table that keeps track of any OS-level state associated with the file, such as the file position

Networks and Sockets

- UNIX likes to make *all* I/O look like file I/O
  - You use `read()` and `write()` to communicate with remote computers over the network!
  - A file descriptor used for network communications is called a socket
  - Just like with files:
    - Your program can have multiple network channels open at once
    - You need to pass a file descriptor to `read()` and `write()` to let the OS know which network channel to use

File Descriptor Table



OS's File Descriptor Table for the Process

File Descriptor	Type	Connection
0	pipe	stdin (console)
1	pipe	stdout (console)
2	pipe	stderr (console)
3	TCP socket	local: 128.95.4.33:80 remote: 44.1.19.32:7113
5	file	index.html
8	file	pic.png
9	TCP socket	local: 128.95.4.33:80 remote: 102.12.3.4:5544

Types of Sockets

- Stream sockets
  - For connection-oriented, point-to-point, reliable byte streams
    - Using TCP, STCP, or other stream transports
- Datagram sockets
  - For connection-less, one-to-many, unreliable packets
    - Using UDP or other packet transports
- Raw sockets
  - For layer-3 communication (raw IP packet manipulation)