CSE 124 AND CSE 224:

FUNDAMENTALS OF NETWORKING AND GO'S NET PACKAGE

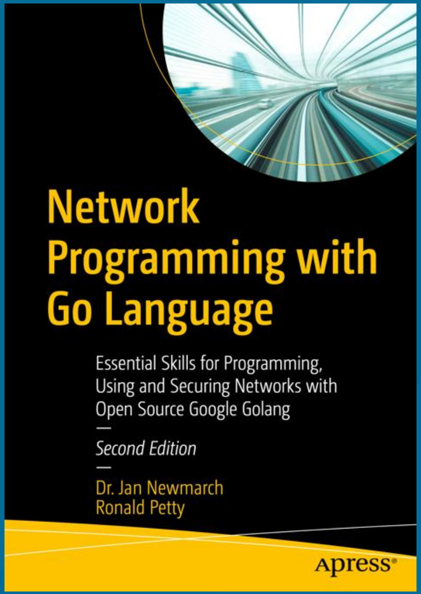
George Porter
April 8, 2025

UC San Diego

ATTRIBUTION

UC San Diego

REFERENCE MATERIAL



Chapter 1
First part of Chapter 3

UC San Diego

ANNOUNCEMENTS

1. One-day extension form available off the modules page in canvas

2. Friday's TA materials linked off the modules page (the podcast system wasn't set up correctly to record the Friday meeting, which has now been fixed)

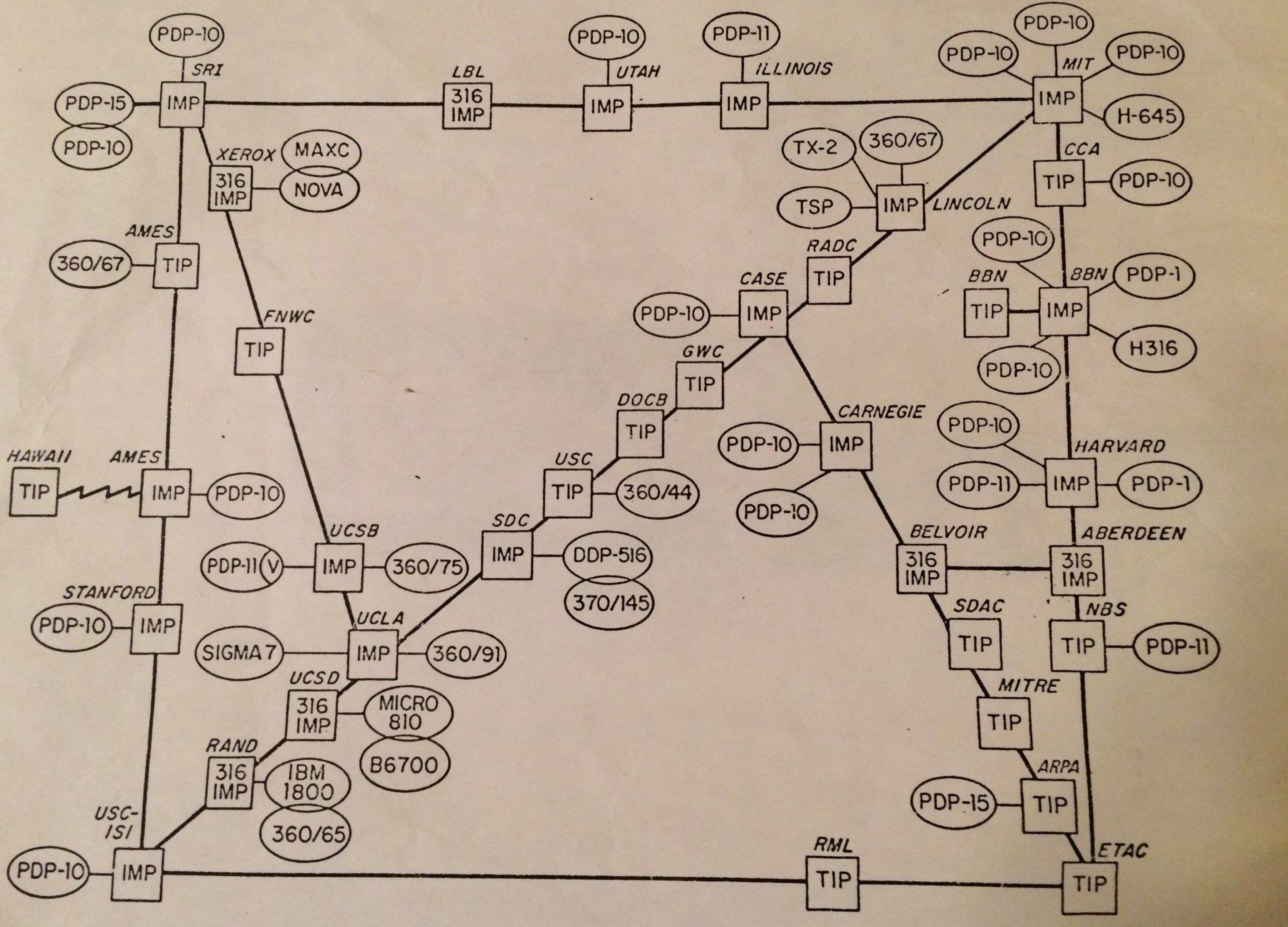
UC San Diego

BRIEF HISTORY OF THE INTERNET

- 1968 - DARPA (Defense Advanced Research Projects Agency) contracts with BBN (Bolt, Beranek & Newman) to create ARPAnet
- 1970 - First five nodes:
  - UCLA
  - Stanford
  - UC Santa Barbara
  - U of Utah, and
  - BBN
- 1974 - TCP specification by Vint Cerf & Kahn
- 1984 – On January 1, the Internet with its 1000 hosts converts en masse to using TCP/IP for its messaging

Data from the Internet Society

ARPA NETWORK, LOGICAL MAP, MAY 1973



Outline

1. Protocols and Layering
2. Addressing
3. Wrap-up/Q&A

KEY NETWORKING CONCEPT: LAYERING

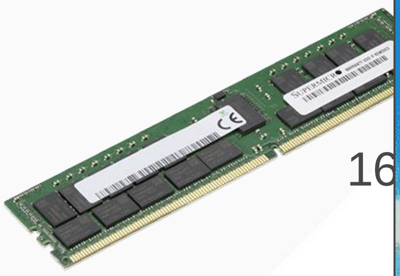
"All problems in computer science can be solved by another level of indirection."

David J. Wheeler

LET'S IMAGINE WRITING A STRING TO A FILE…

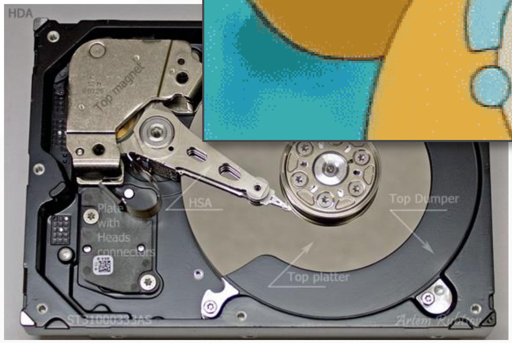
Where is the string?

32GB 288-Pin DDR 2400 ram?



16 ?

Where is the file w

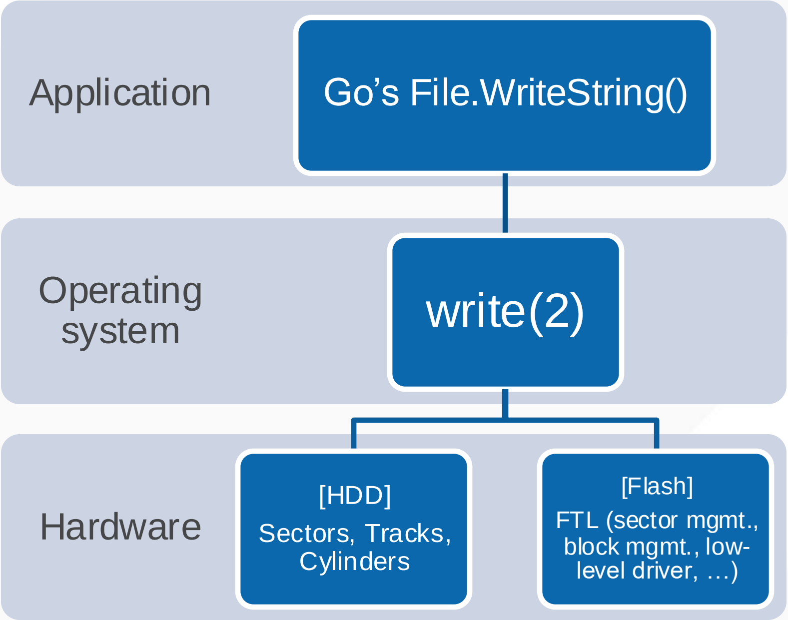


Hard disk drive? (SAS? SATA?)

M.2 NVMe Flash drive?

LAYERING AND TA SECTION DEMO

```
file, err := os.Create("example.txt")
…
writer := bufio.NewWriter(file)
…
writer.WriteString("Greetings!" + "\n")
writer.Flush()
…
file.Close()
```



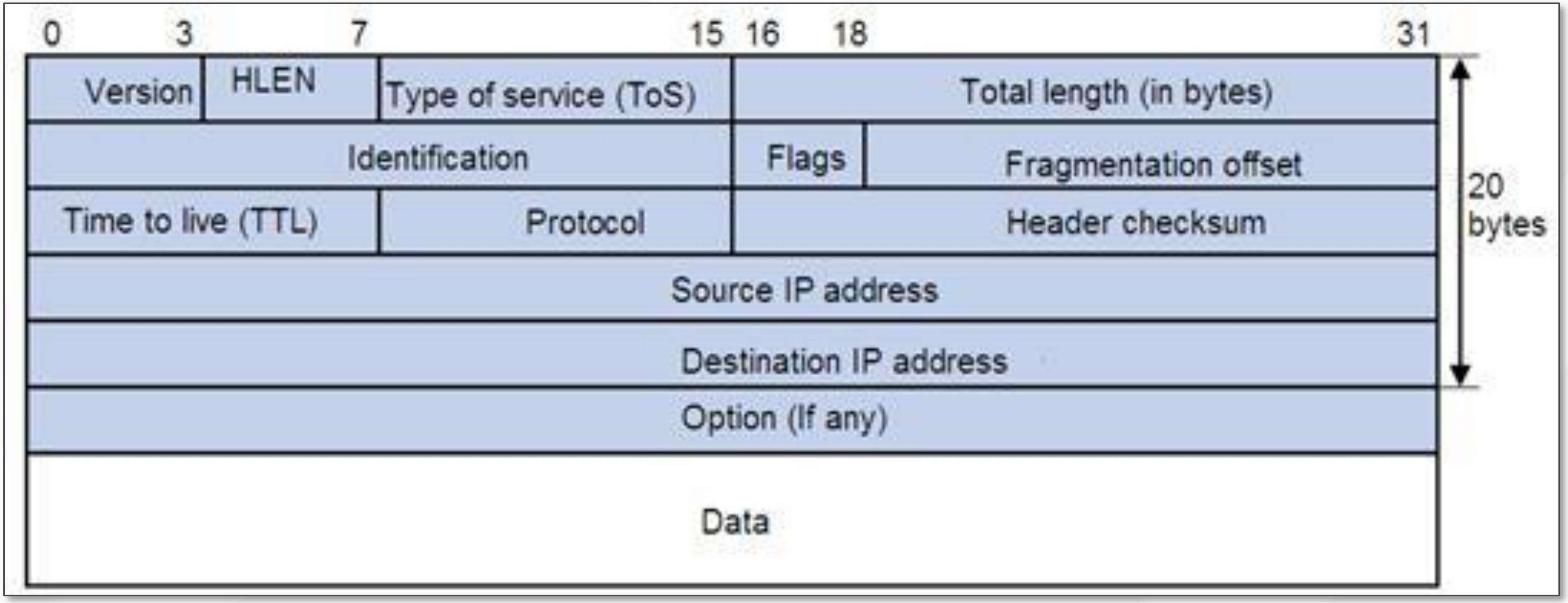
- Sub-divide the problem
  - Each layer relies on services from layer below
  - Each layer exports services to layer above

- Interface between layers defines interaction
  - Hides implementation details (encapsulation)
  - Layers can change without disturbing other layers (modularity)

INTERNET DELIVERY MODEL

- Packets are communicated between *hosts*, which are computers such as laptops, desktops, servers, phones, PS5s, Nintendo Switch, Nintendo Switch 2 (???!), etc.
- Send and receive *packets* of data, up to 64KB in size
  - Though 1500 bytes is the norm
- Connection-less (every packet is handled separately and independently)
- “Best-effort” delivery
  - Arbitrary order of packet delivery
  - Packets can be lost, and there is no automatic retransmission
  - Possible duplicates
  - Packets can get corrupted during transit
  - Packets can be delayed arbitrarily (how to know when it’s lost vs. just really late??)

AN IP PACKET


0
3
7
15
16
18
31
Version
HLEN
Type of service (ToS)
Total length (in bytes)
Identification
Flags
Fragmentation offset
Time to live (TTL)
Protocol
Header checksum
Source IP address
Destination IP address
Option (If any)
Data
20 bytes

COMMUNICATION MODEL: MESSAGE PASSING

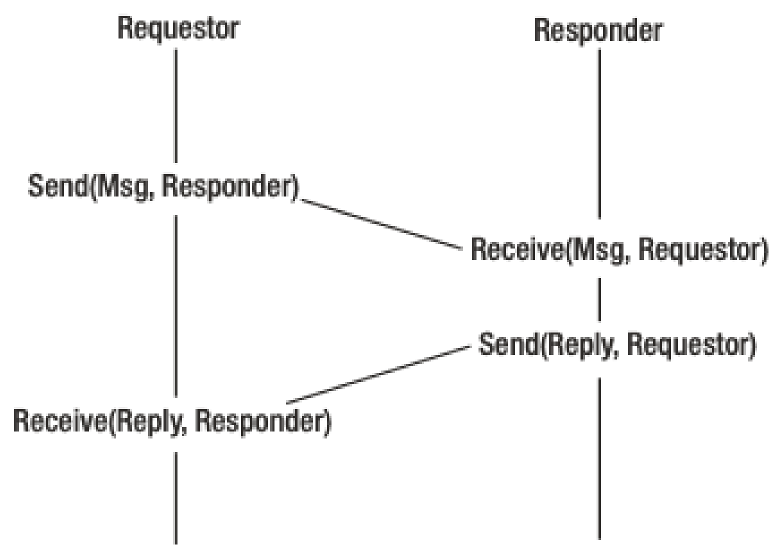


Figure 1-4. *The message passing communications model*

- UDP: "Packets" of data (up to ~1500 bytes) sent from one application to another (same delivery model as the Internet at large)
- TCP: A "stream" of bytes is transmitted reliably from one application to another (the TCP protocol ensures that the data arrives reliably and in the same order as it was sent)

COMMUNICATION MODEL: REMOTE PROCEDURE CALLS

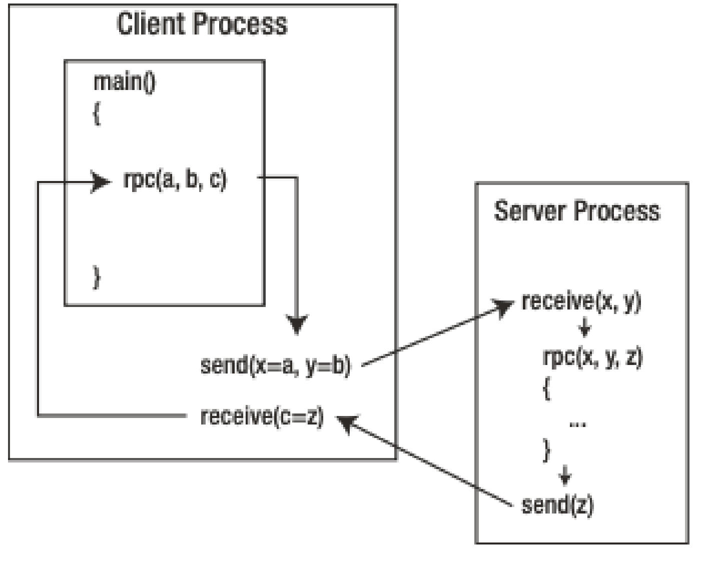


Figure 1-5. *The remote procedure call communications model*

- Just as you can call functions/procedures/methods in a local library, linked into your code, RPCs allow you to “call into” code on another machine/server

BUILDING ON TOP OF THE RAW INTERNET PROTOCOL

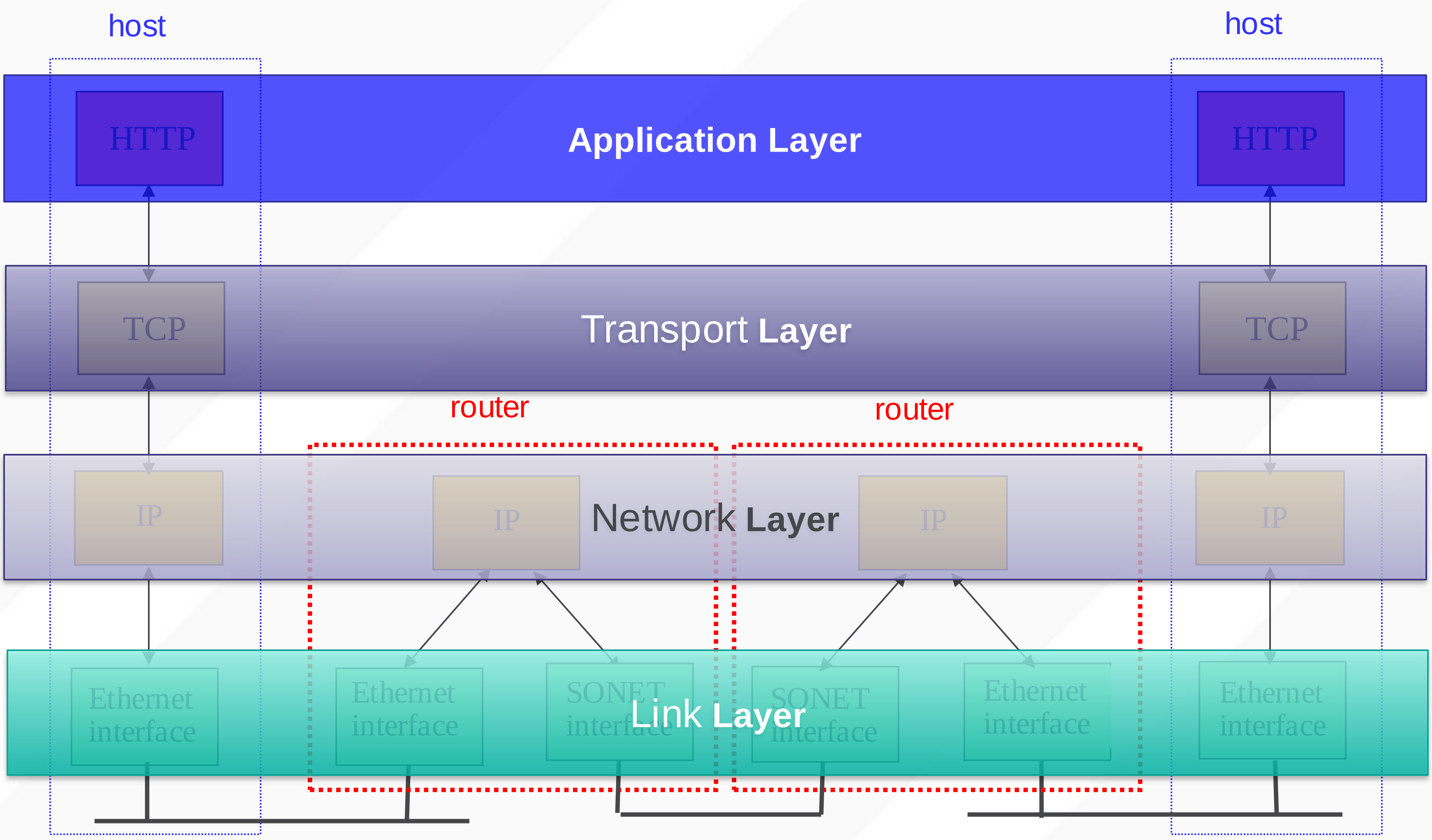
TCP protocol

- Connection-oriented
  - Requires connection establishment & termination
- Interface: “Infinite bytestream”
- Reliable delivery
  - In-order delivery
  - Retransmission
  - No duplicates
- High variance in latency
  - Cost of the reliable service
- E.g., HTTP, SSH, FTP, …

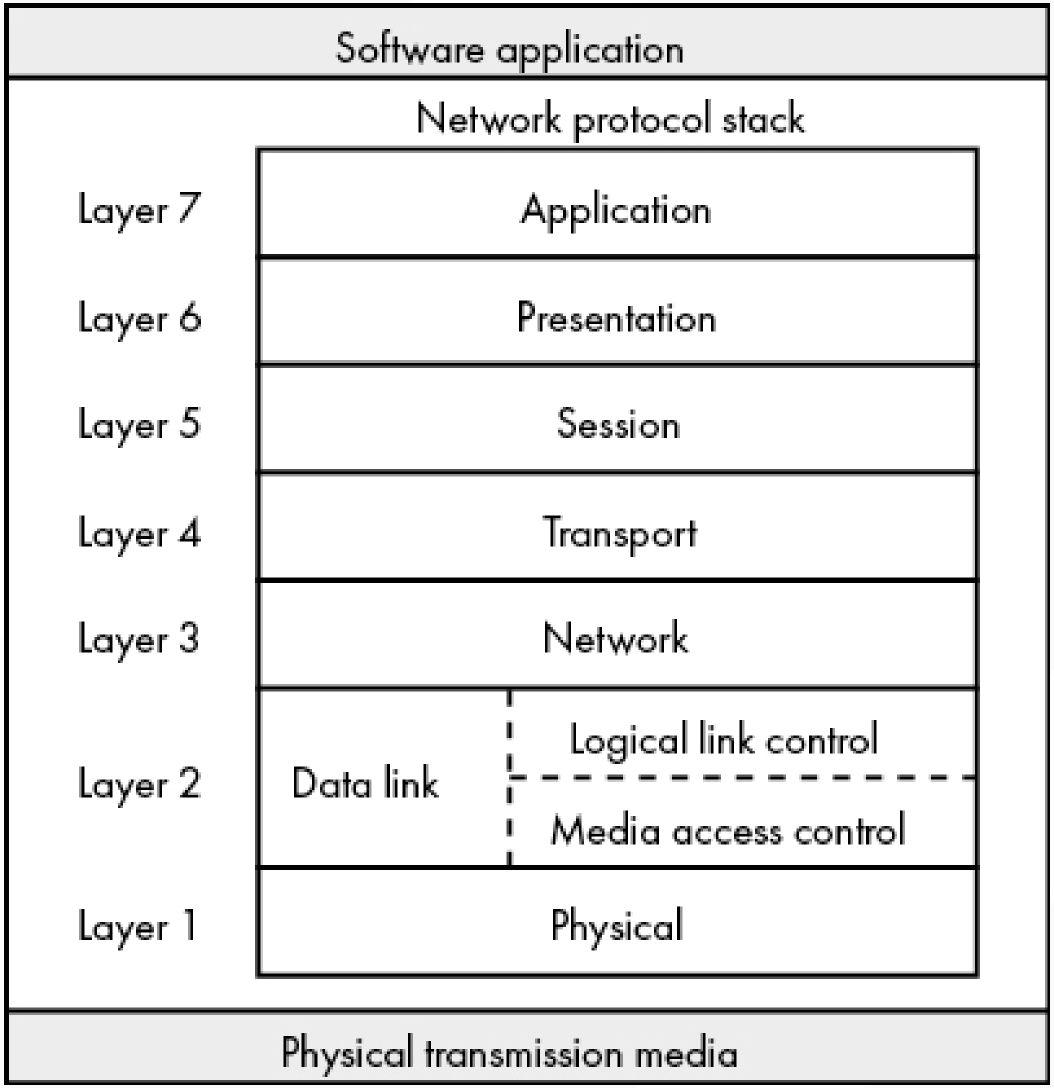
UDP protocol

- Connection-less
- “Best-effort” delivery
  - Arbitrary order of packet delivery
  - No retransmission
  - Possible duplicates
- Low variance in latency
- Packet-like interface
  - Requires packetizing
- E.g., DNS, VoIP, VOD, …

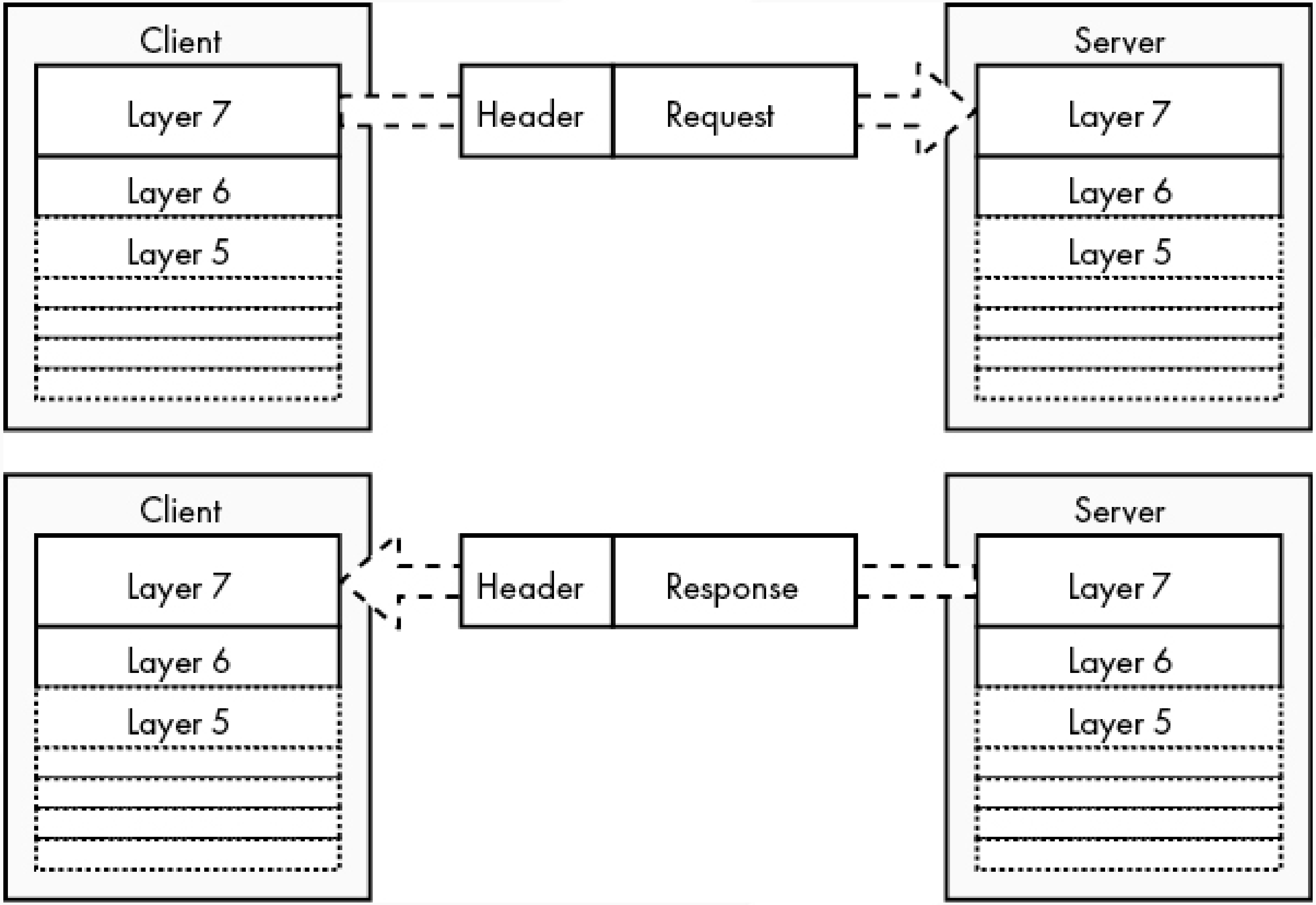
NETWORK LAYERING



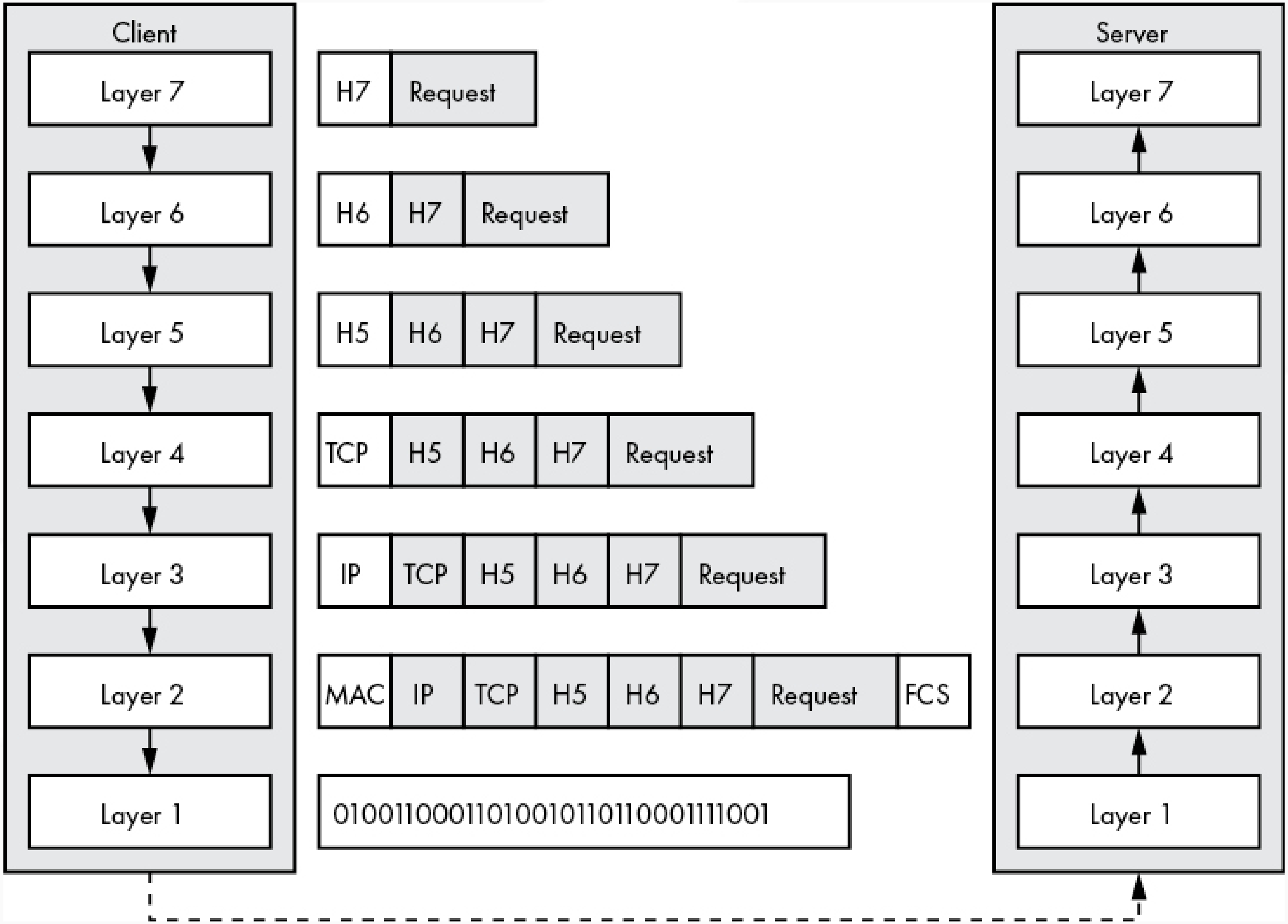
OSI NETWORK STACK



PROTOCOLS GOVERN MESSAGES EXCHANGED WITHIN A SINGLE LAYER



ENCAPSULATION VIA HEADERS



TCP/IP MODEL (VS OSI MODEL)

<table>
<tr><td colspan="2">Software application</td></tr>
<tr><th>TCP/IP</th><th>OSI</th></tr>
<tr><td rowspan="3">Application</td><td>Application</td></tr>
<tr><td>Presentation</td></tr>
<tr><td>Session</td></tr>
<tr><td>Transport</td><td>Transport</td></tr>
<tr><td>Internet/network</td><td>Network</td></tr>
<tr><td rowspan="2">Link</td><td>Data link</td></tr>
<tr><td>Physical</td></tr>
<tr><td colspan="2">Physical transmission media</td></tr>
</table>

Outline

1. Protocols and Layering
2. Addressing
3. Wrap-up/Q&A

MOTIVATION FOR OUR DISCUSSION OF ADDRESSING

1. When you implement a *client* application, you will typically need to communicate with a remote system/service via its *IP Address*

2. When you implement a cloud-hosted *network service*, you will typically be assigned a *subnet (group)* of IP addresses to use for your various server programs, and will need to use and manage those appropriately

IP PROTOCOL

- Scenario: An internet connected device wants to send a message to another internet connected device anywhere in the world
  - IP protocol handles this
  - Prepend an “IP header” to the message, set the destination IP address, set the source IP address, and Internet routers will forward it along the shortest path till it gets to the destination network
  - From there, the destination network forwards the packet to the intended host

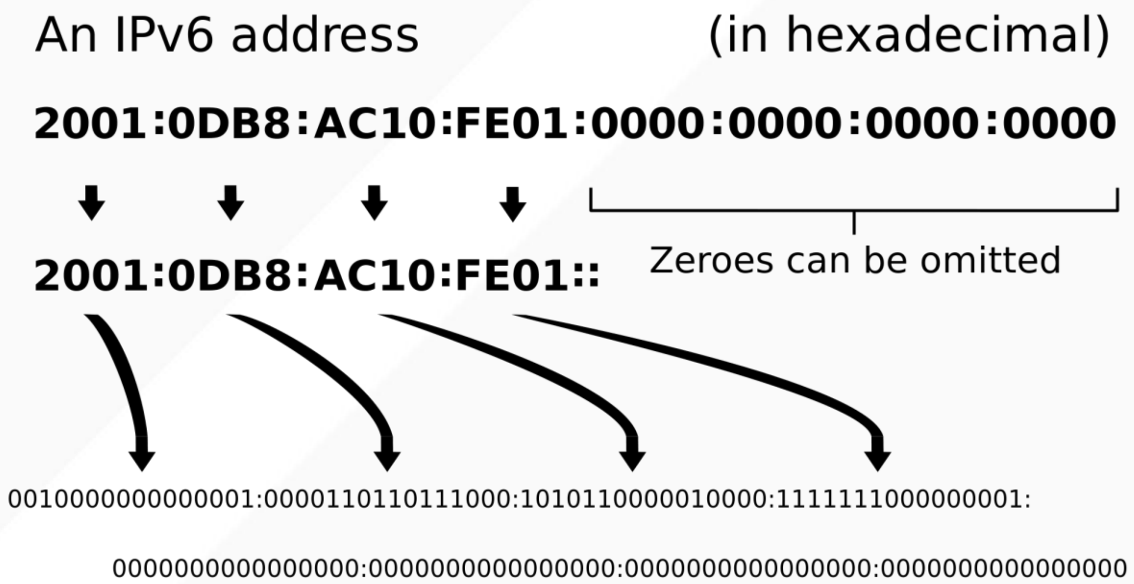
** Simplified model but sufficient for our purposes*

IP VERSION 4 (IPV4)

11000000	.	10101000	.	00000001	.	00001010	(Binary)
192	.	168	.	1	.	10	(Decimal)

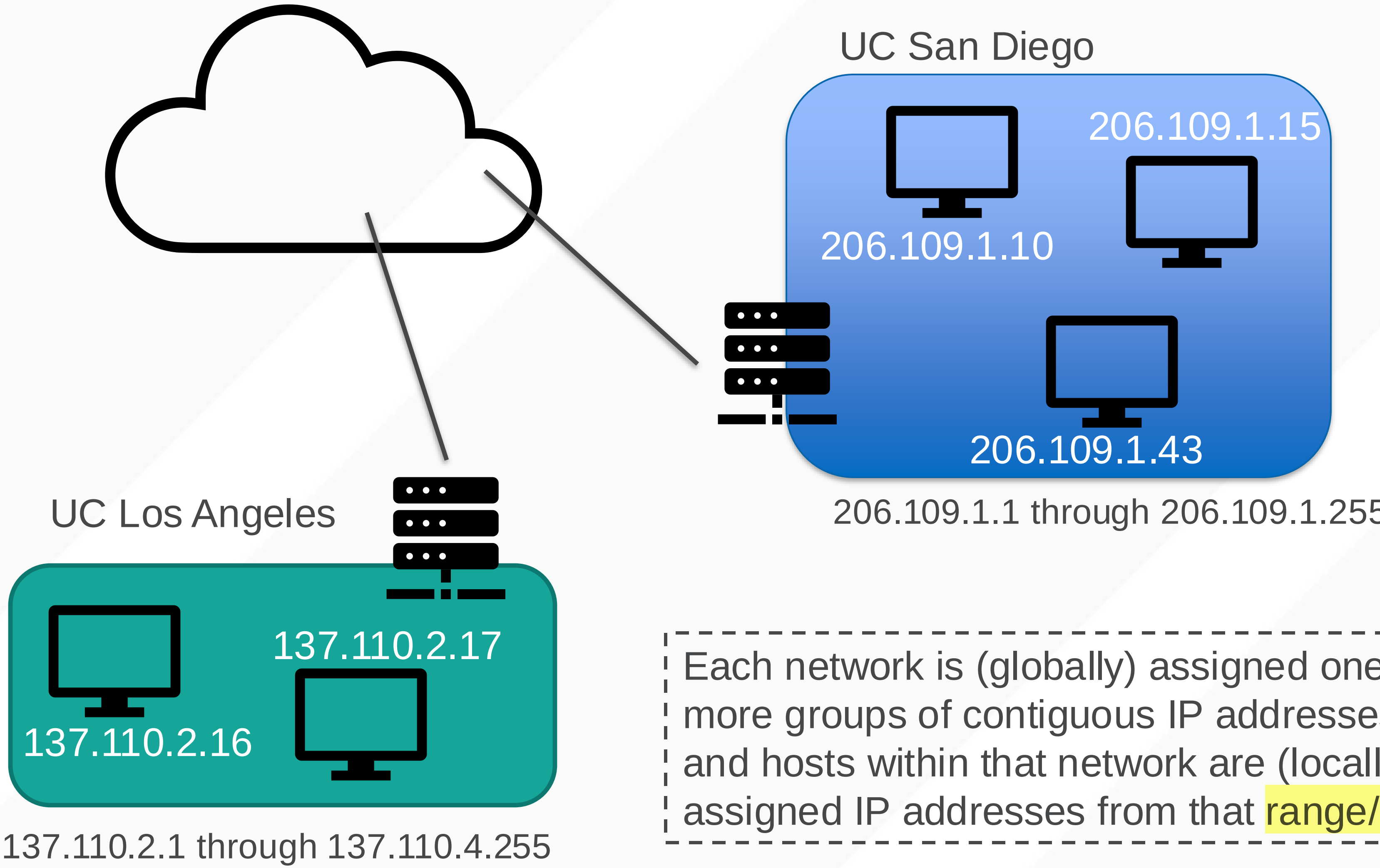
- 32-bits
- Usually represented as four 8-bit bytes (octets)
- E.g. 206.109.1.6, 127.0.0.1, 192.168.1.10

IP VERSION 6 (IPV6)



- 128-bits
- Represented as 8 16-bit blocks, in hex, separated by colons
- E.g. 2001:4860:4860::8888, ::1, 2345:0425:2CA1:0000:0000:0567:5673:23b5

THE INTERNET: A NETWORK OF SUB-NETWORKS



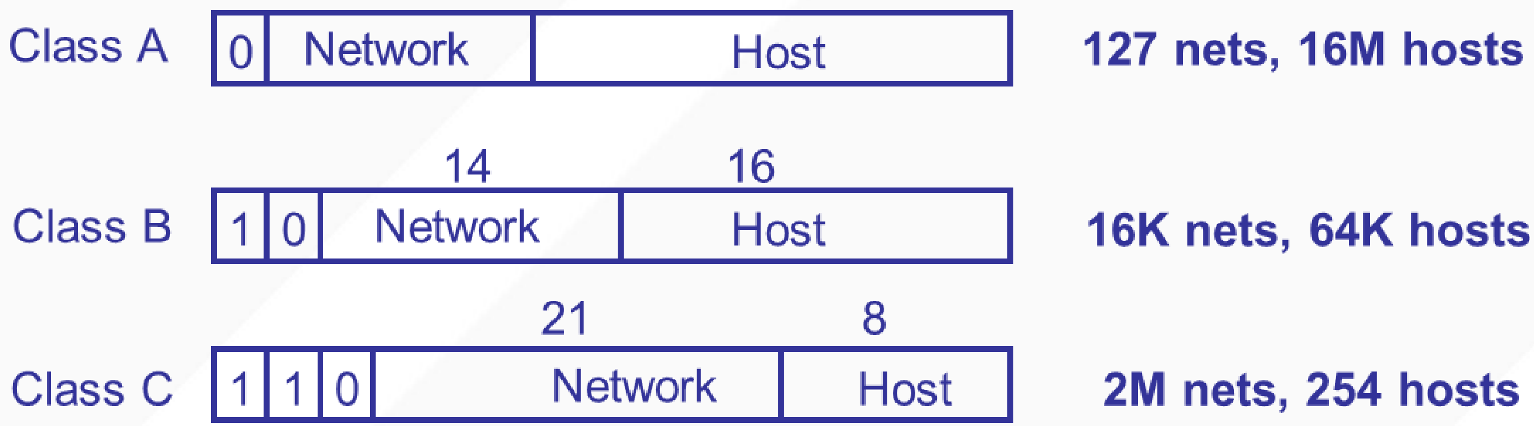
Each network is (globally) assigned one or more groups of contiguous IP addresses, and hosts within that network are (locally) assigned IP addresses from that range/pool

ROUTE AGGREGATION

- 2^32 possible (IPv4) addresses spread across ~100,000 independent networks
- Each router keeps the "next hop" on a per-network basis, not per-host basis
- But networks can be different sizes (e.g. UCSD is bigger than a small startup)
- Each router has to keep a list of networks (and their next hops) + how "big" each network is

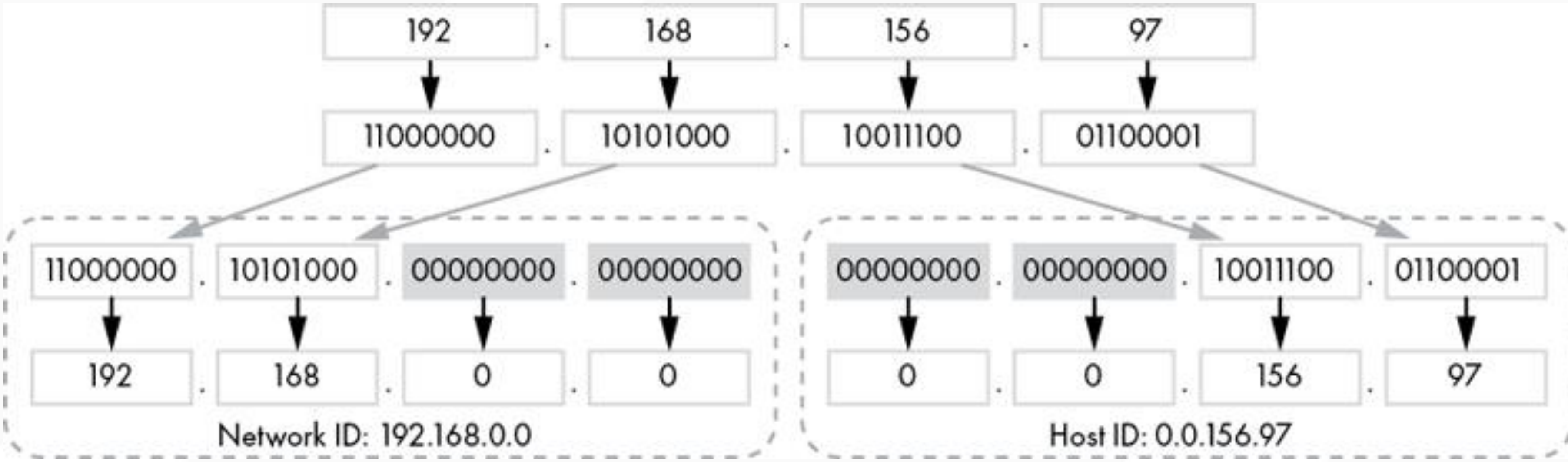
CLASS-BASED ADDRESSING (NOT REALLY USED ANYMORE)

- Most significant bits determines “class” of address



- Special addresses
  - Class D (1110) for multicast, Class E (1111) experimental
  - 127.0.0.1: local host (a.k.a. the loopback address)
  - Host bits all set to 0: network address
  - Host bits all set to 1: broadcast address (sent to every host in the local network, though often disabled for large networks to avoid too much cross-talk/background traffic)

ADDRESSING EXAMPLE FOR 192.168.0.0/16



IP ADDRESS PROBLEM (1991)

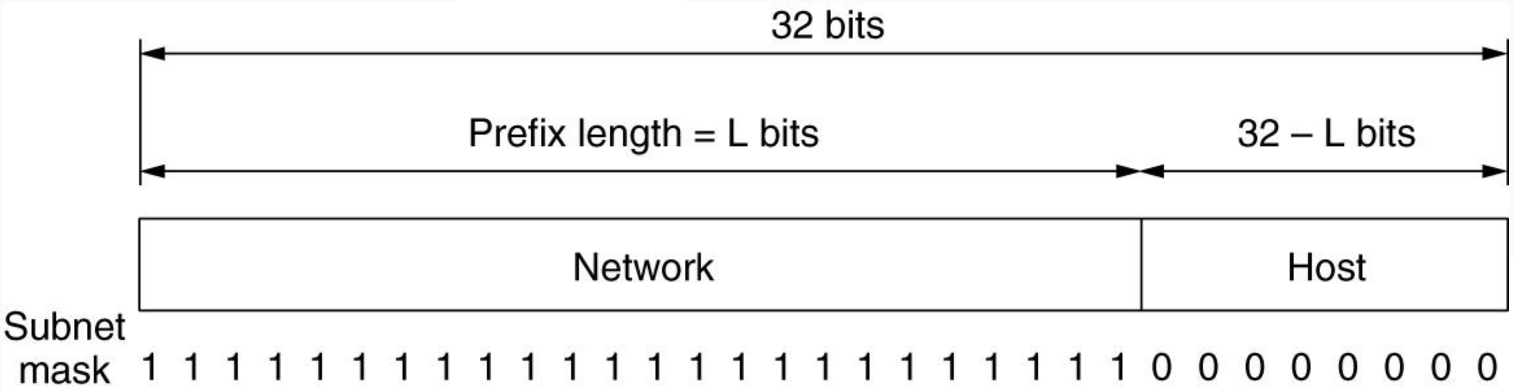
- Address space depletion
  - In danger of running out of classes A and B
- Why?
  - Class C too small for most organizations (only ~250 addresses)
  - Very few class A – very careful about giving them out (who has 16M hosts anyway?)
  - Class B – greatest problem

CIDR

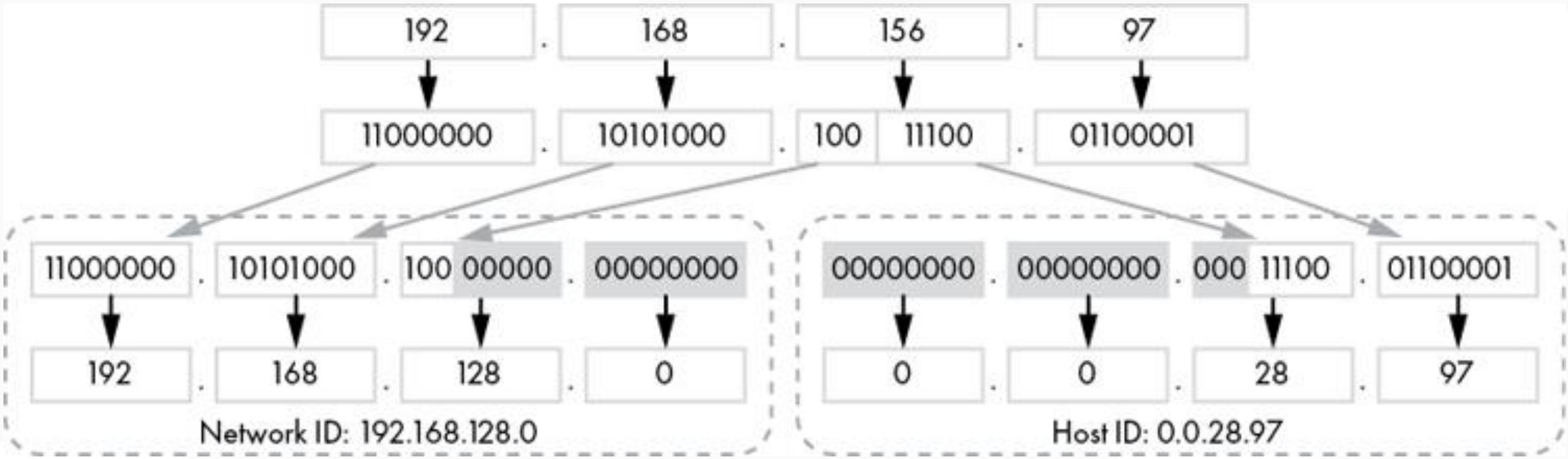
- Classless Inter-Domain Routing (1993)
  - Networks described by variable-length prefix and length
  - Allows arbitrary allocation between network and host address

  - e.g. 10.95.1.2 contained within 10.0.0.0/8:
    - 10.0.0.0 is network and remainder (95.1.2) is host
- Pro: Finer grained allocation; aggregation
- Con: More expensive lookup: longest prefix match

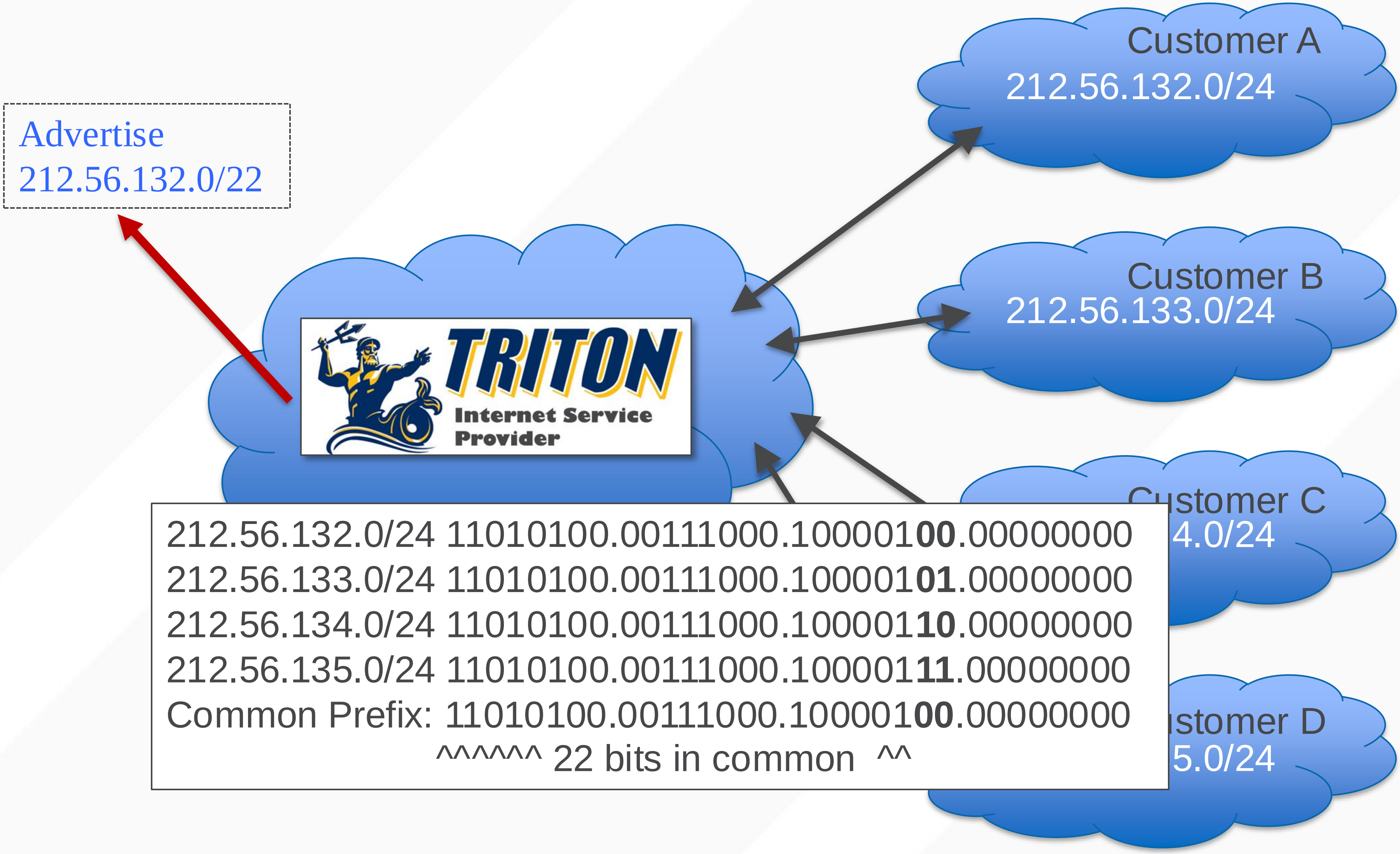
SUBNETS AND NETMASKS



192.168.156.97/19



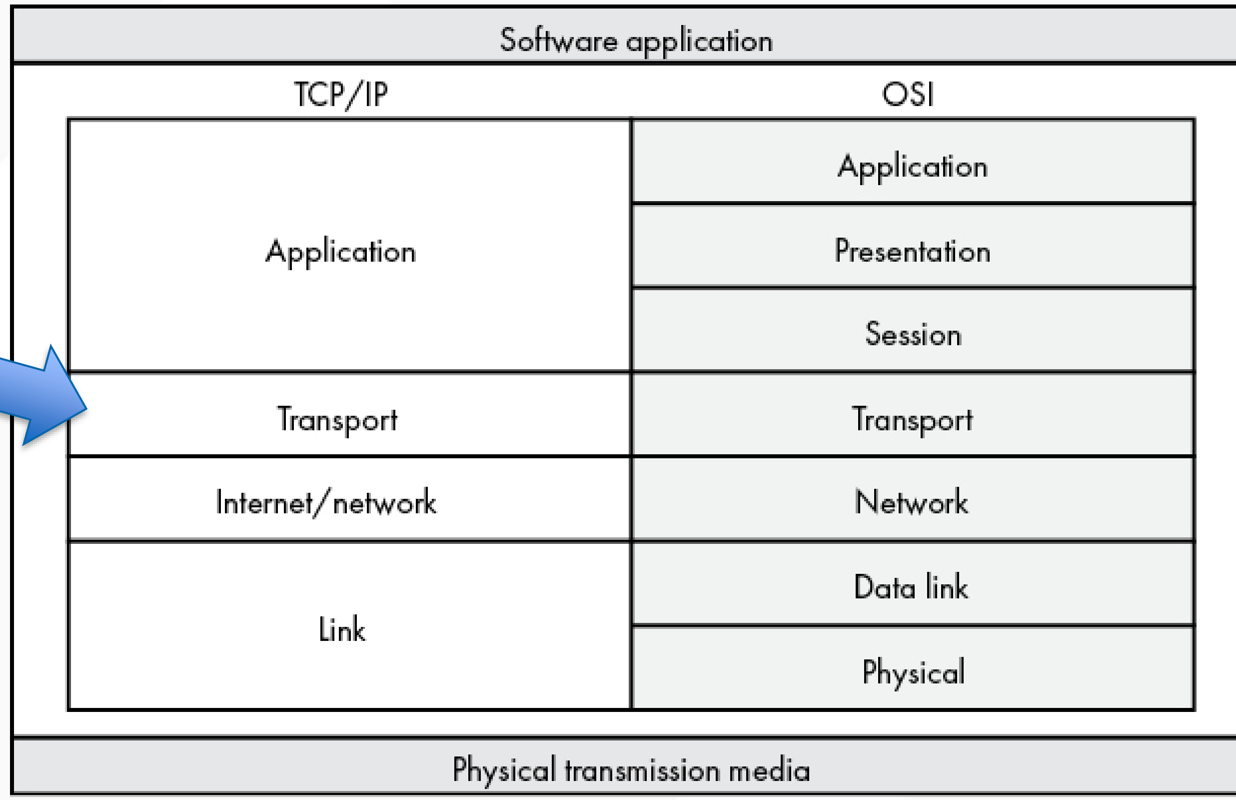
ADDRESS AGGREGATION EXAMPLE



212.56.132.0/24 11010100.00111000.100001**00**.00000000
212.56.133.0/24 11010100.00111000.100001**01**.00000000
212.56.134.0/24 11010100.00111000.100001**10**.00000000
212.56.135.0/24 11010100.00111000.100001**11**.00000000
Common Prefix: 11010100.00111000.100001**00**.00000000
^^^^^^ 22 bits in common ^^

TCP/IP MODEL (VS OSI MODEL)

Which *process* on the destination machine? What if I want to send a stream of bytes, not just a limited-size message?



PORTS

- IP addresses identify a *machine*
  - Actually they identify a network interface on a machine
- How to identify different programs on the machine?
  - Process ID/PID? (no… why not?)
  - Instead we use a port (which is a 16-bit number)
  - 0-1024 reserved for the OS, you can use 1025-65535

CONVERTING FROM A NAME TO AN IP ADDRESS

- Domain name system (DNS)
  - Converts from names to addresses
  - (And a lot more, actually… we have a whole lecture on DNS coming up)
- In Go, can rely on net.LookupIP(name):
  - ips, err := net.LookupIP(“www.google.com”)
  - Note that LookupIP returns a slice, not a single response…
    - Names can map to more than one IP address
- [demo code in lookup.go]

Outline

1. Protocols and Layering
2. Addressing
3. Wrap-up/Q&A

UC San Diego