NETWORKING FUNDAMENTALS

George Porter
Module 1
Fall 2020

UC San Diego

ATTRIBUTION

- These slides incorporate material from:
  - Alex C. Snoeren, UC San Diego
  - Michael Freedman and Kyle Jamieson, Princeton University
  - Internet Society
  - Computer Networking: A Top Down Approach

UC San Diego

Outline

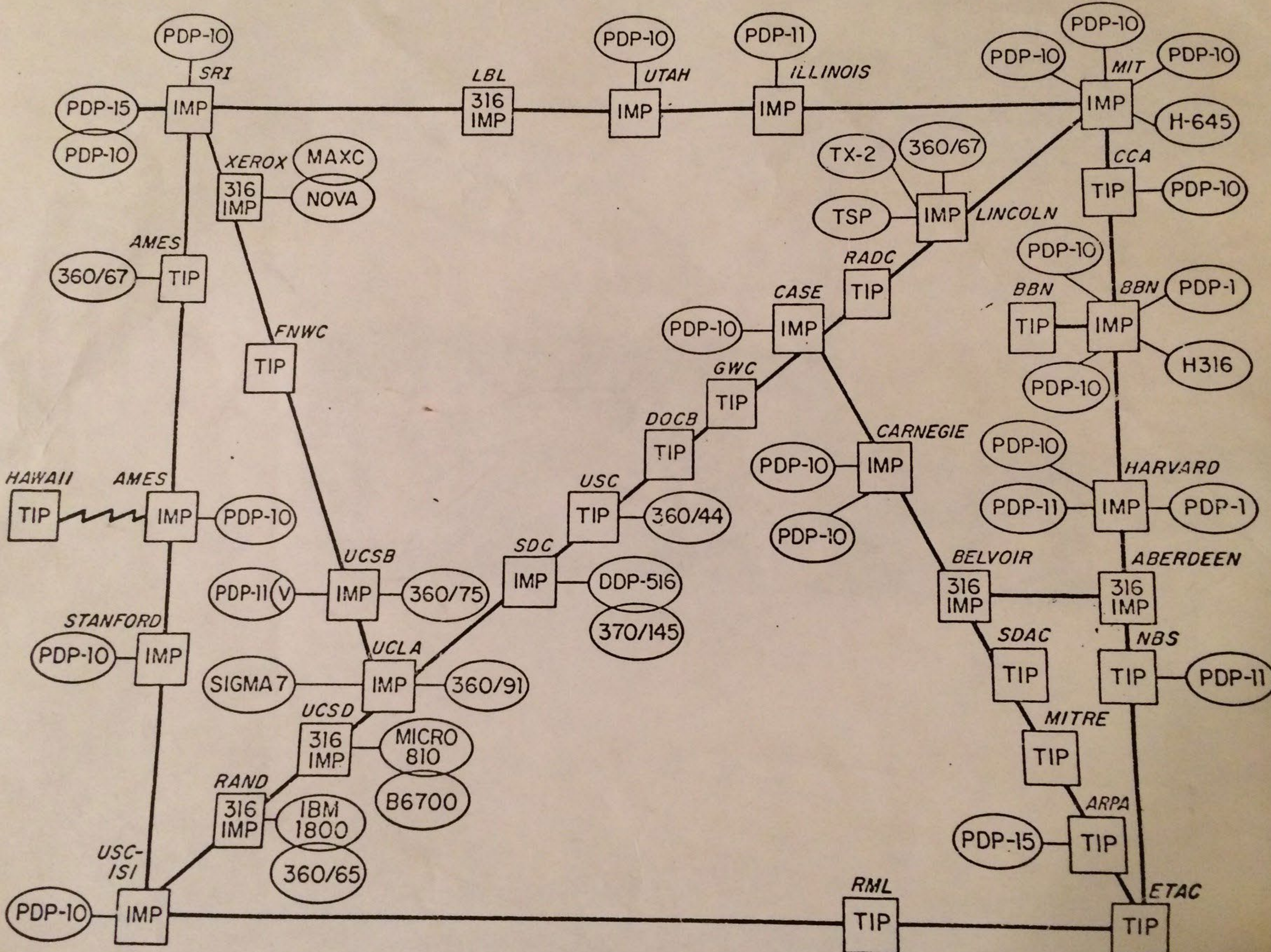
1. Packet switching
2. Addressing

BRIEF HISTORY OF THE INTERNET

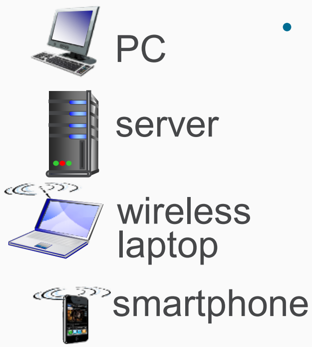
- 1968 - DARPA (Defense Advanced Research Projects Agency) contracts with BBN (Bolt, Beranek & Newman) to create ARPAnet
- 1970 - First five nodes:
  - UCLA
  - Stanford
  - UC Santa Barbara
  - U of Utah, and
  - BBN
- 1974 - TCP specification by Vint Cerf
- 1984 – On January 1, the Internet with its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

Data from the Internet Society

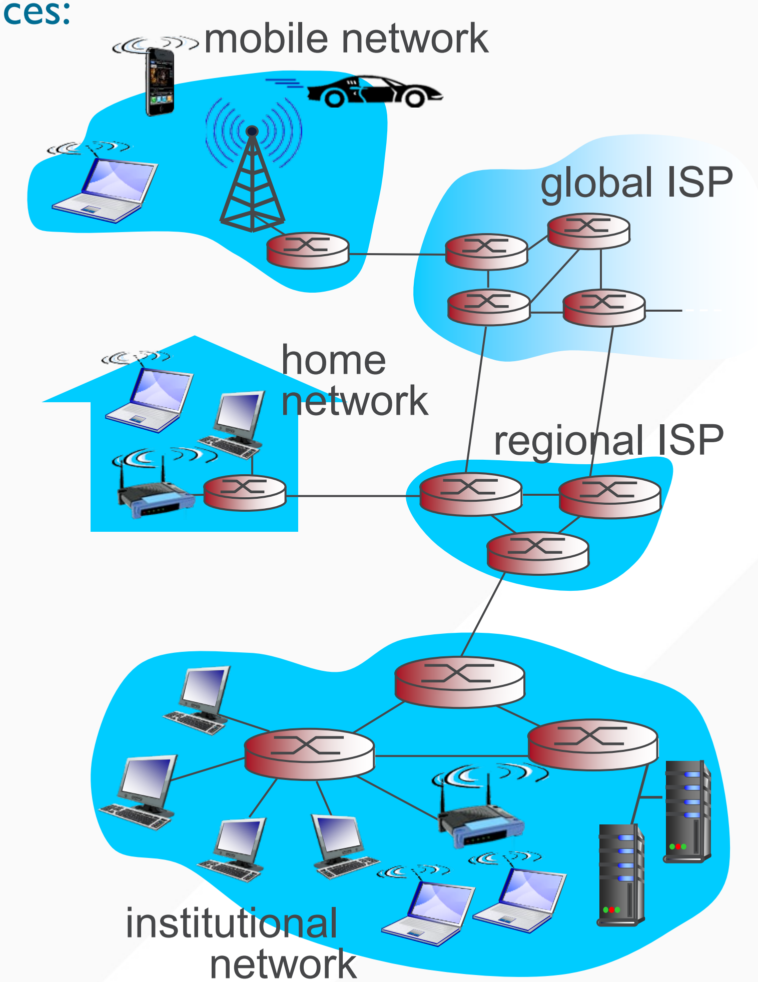
ARPA NETWORK, LOGICAL MAP, MAY 1973

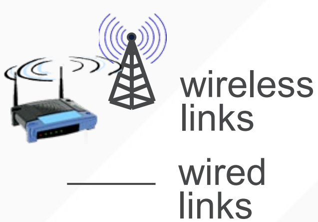


AN INTER-NETWORK

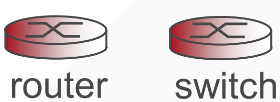


- billions of connected computing devices:
  - *hosts = end systems*
  - running *network apps*





- *communication links*
  - fiber, copper, radio, satellite
  - transmission rate: *bandwidth*

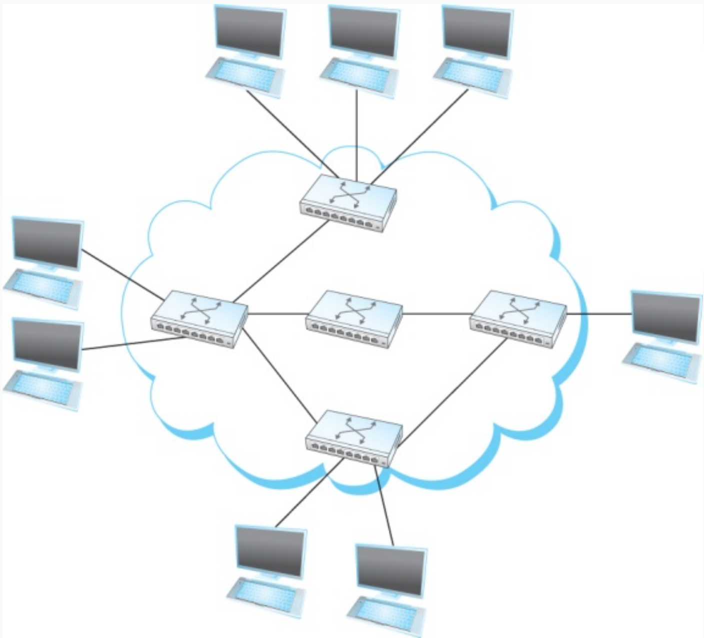


- *Packet switches:* forward packets (chunks of data)
  - *routers* and *switches*

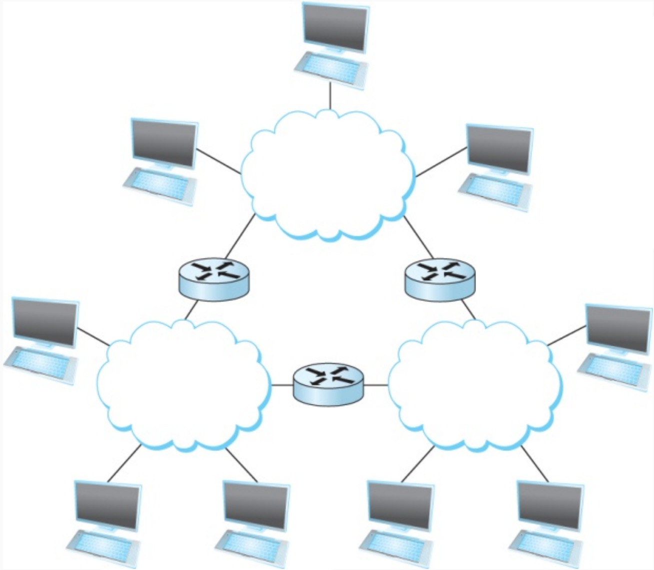
PACKET SWITCHING

- Data transmitted from source to destination in discrete *packets*
  - Variable size, usually a maximum transmission unit (MTU) of ~1500 bytes
  - 1 GB file is approx 715,000 packets
- Each packet routed independently
  - Has its own source and destination *address* used to find the route to the ultimate destination

NETWORK TERMINOLOGY

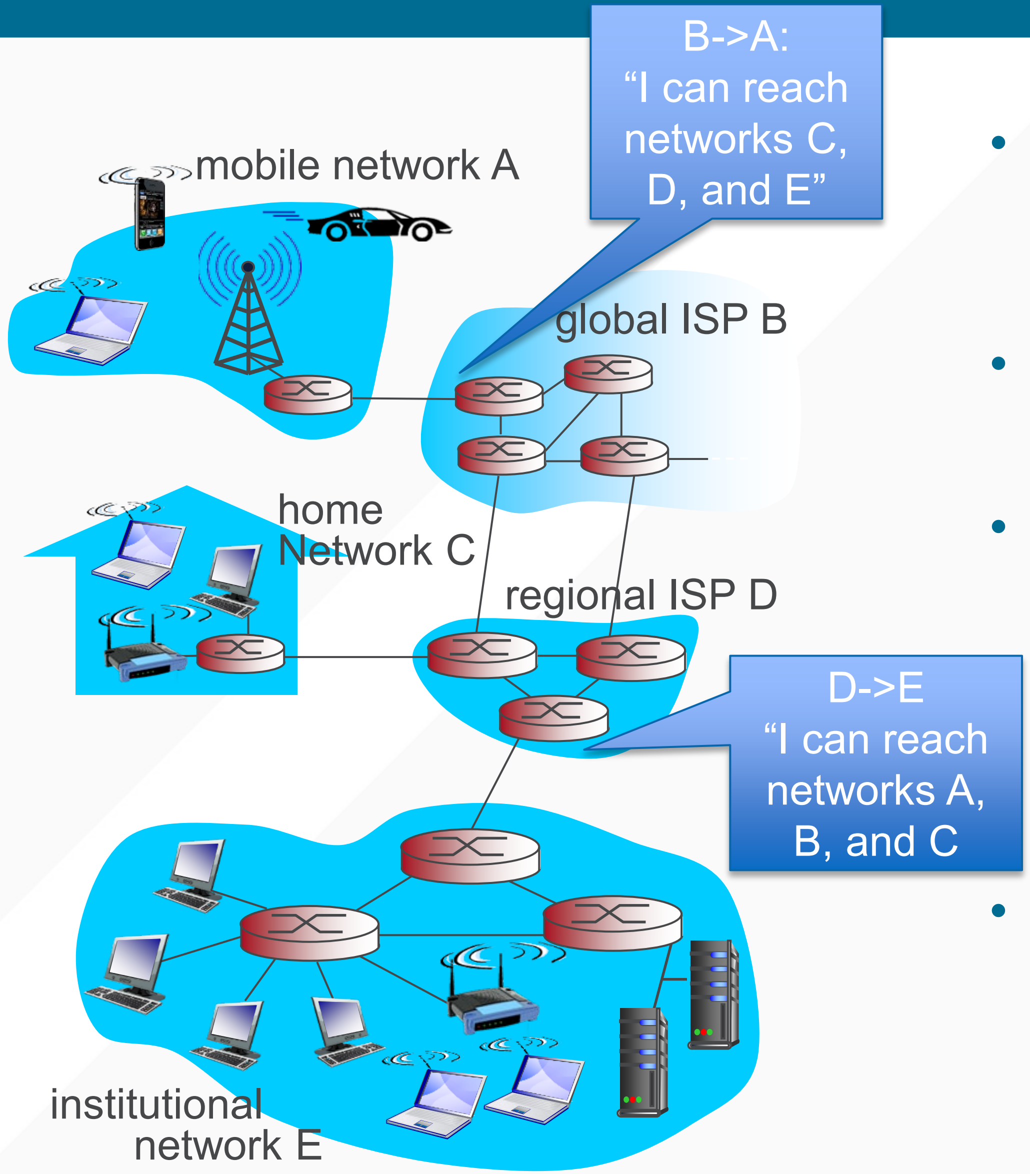


A single packet-switched network

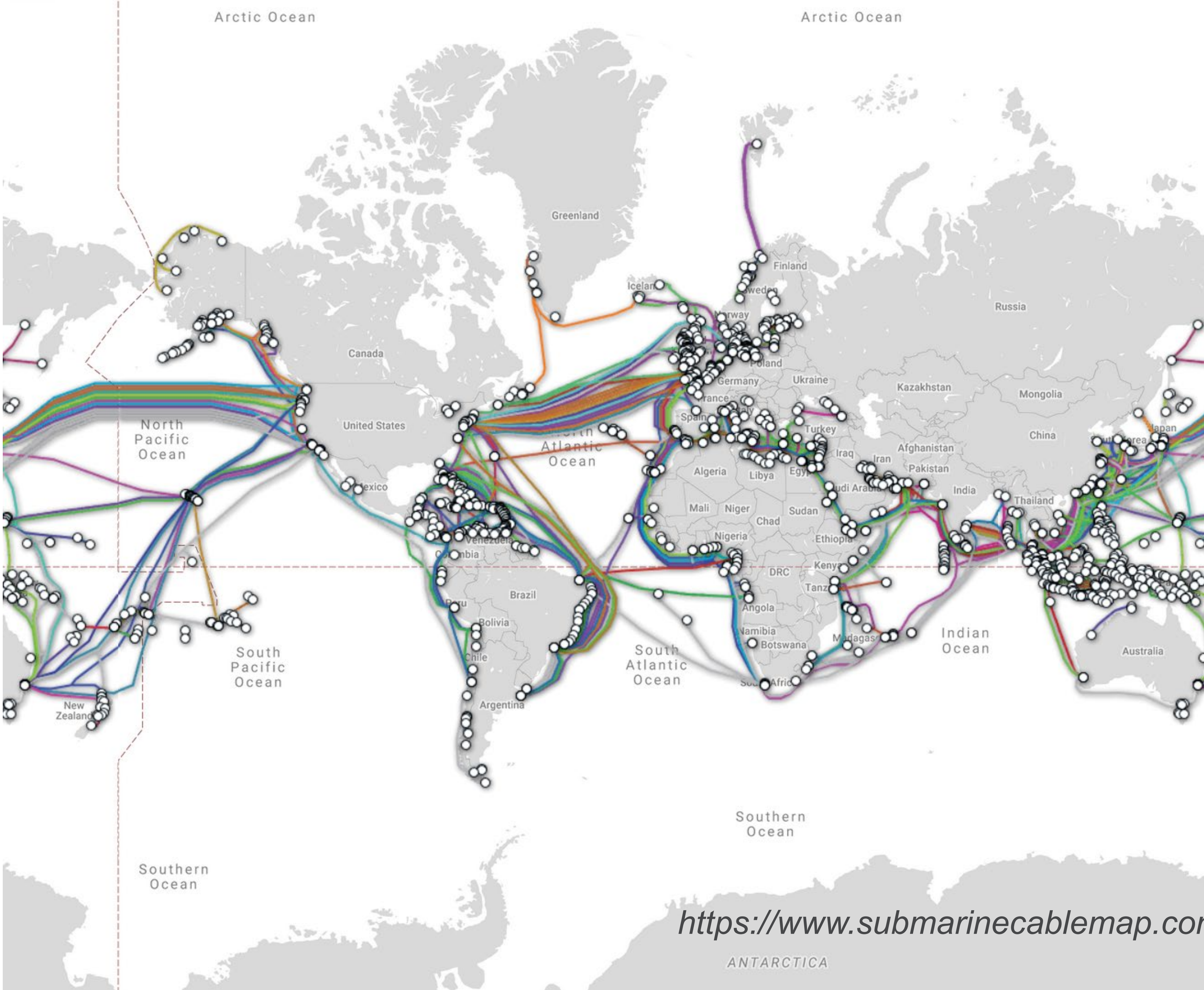


Interconnection of networks

ROUTING PACKETS BETWEEN NETWORKS



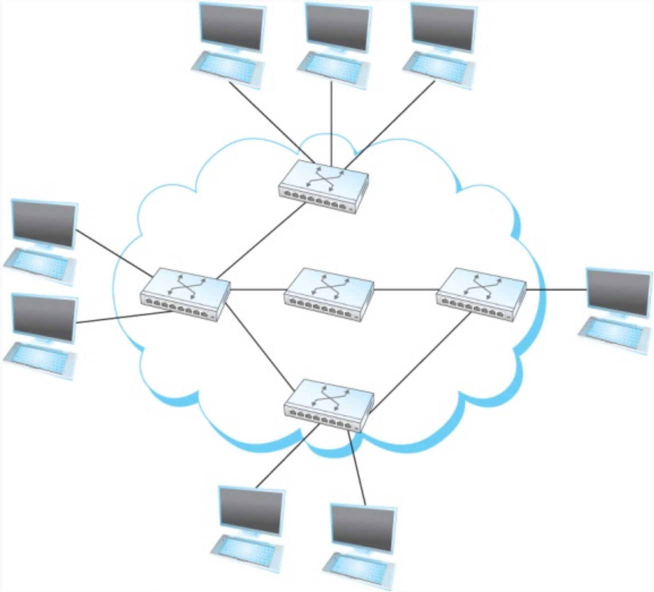
- Networks use *Border Gateway Protocol (BGP)* to announce reachability
- Each network talks just with its neighbors
- Goal is to get a packet to the destination network
  - It is up to that destination network to get individual packets to their ultimate destination
- Back-to-back packets from the same “connection” might take different paths!
  - Might arrive out of order too



https://www.submarinecablemap.com/

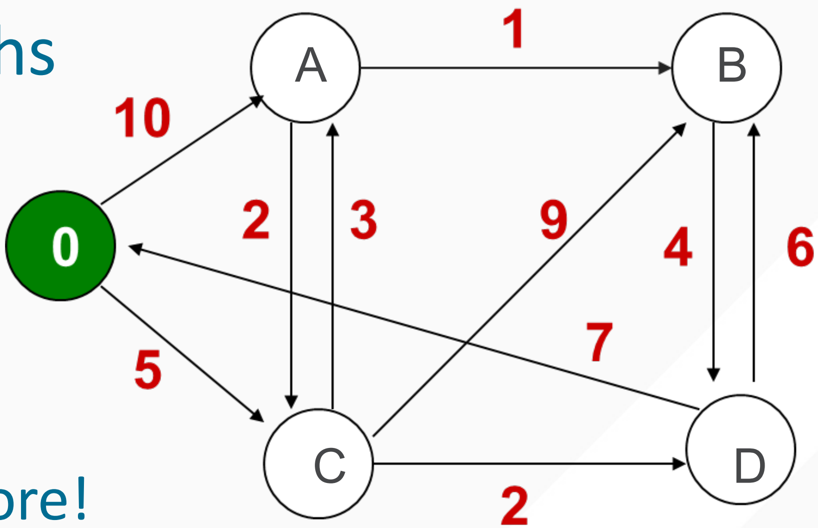
ROUTING PACKETS WITHIN A SINGLE NETWORK

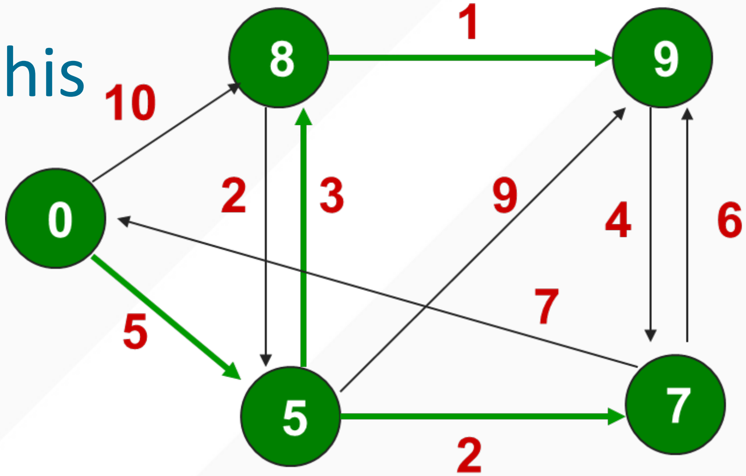
- E.g. UCSD, San Diego Comcast, a Google datacenter
- Packet switches communicate with each other using an *intra-domain* protocol called a Link-state protocol
  - Each packet switch sends a list of its neighbors to every other switch via a *broadcast*
  - As a result, each switch has the same "map" of what the network's *topology* looks like



ROUTING PACKETS WITHIN A SINGLE NETWORK

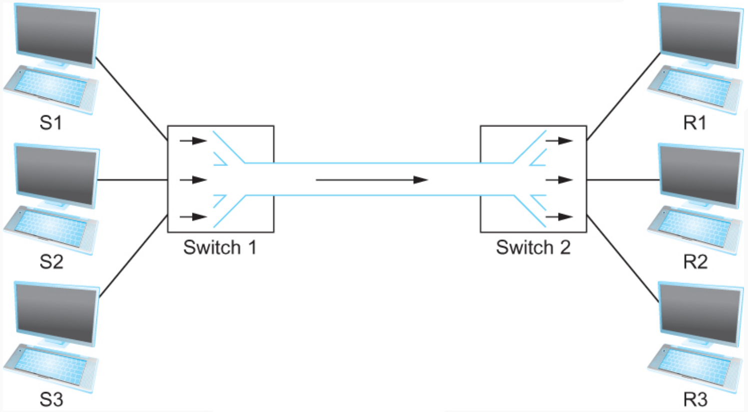
- Each packet switch uses its local topology map to choose paths to each destination
- Typically using Dijkstra's shortest path algorithm
    - Take CSE 123/222a to learn more!
- Packets are forwarded along this shortest path *tree*

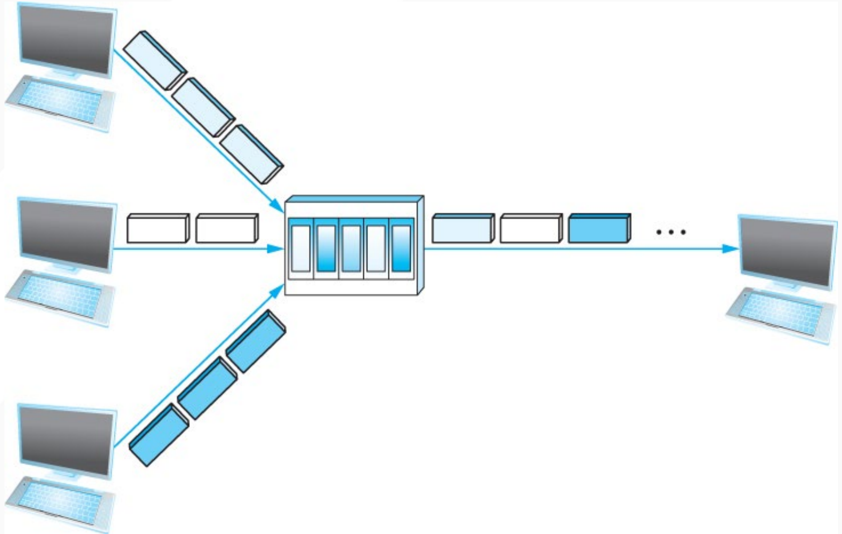




RESOURCE SHARING

Multiplexing multiple logical flows over a single physical link



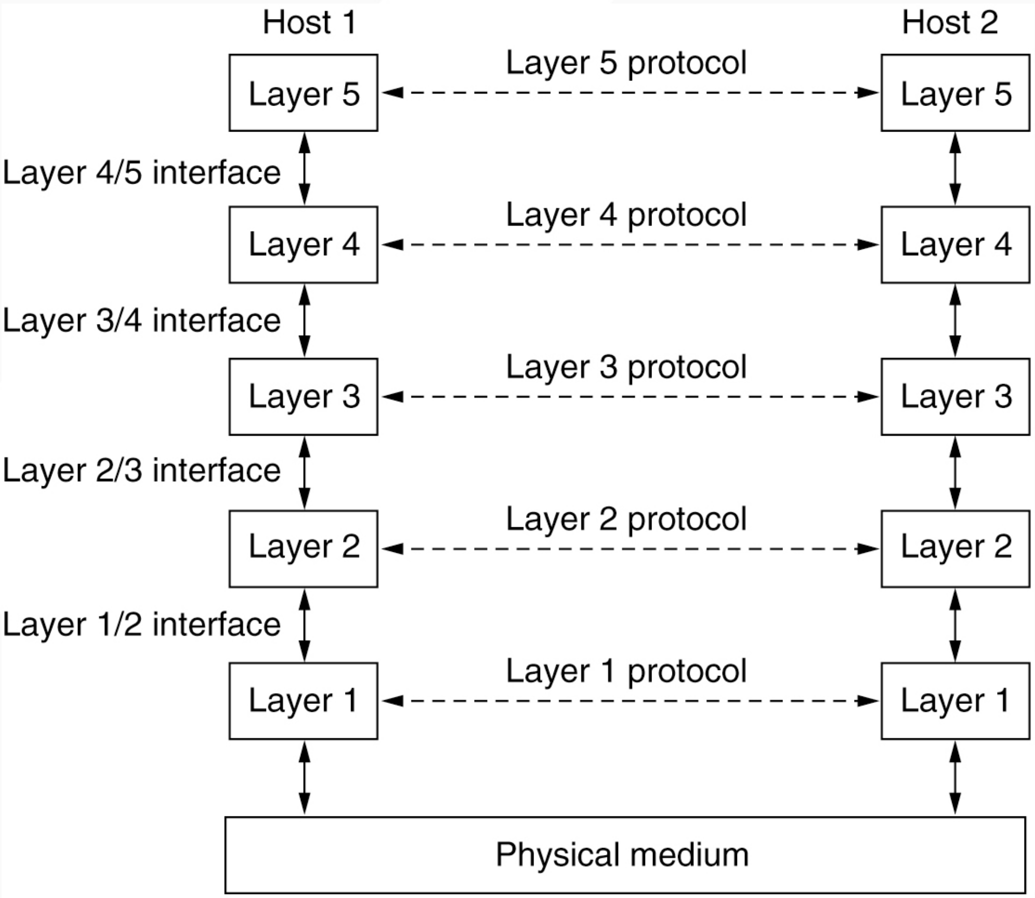


- Resource: links and nodes
- How to share a link?
  - Multiplexing
  - De-multiplexing
  - Queueing

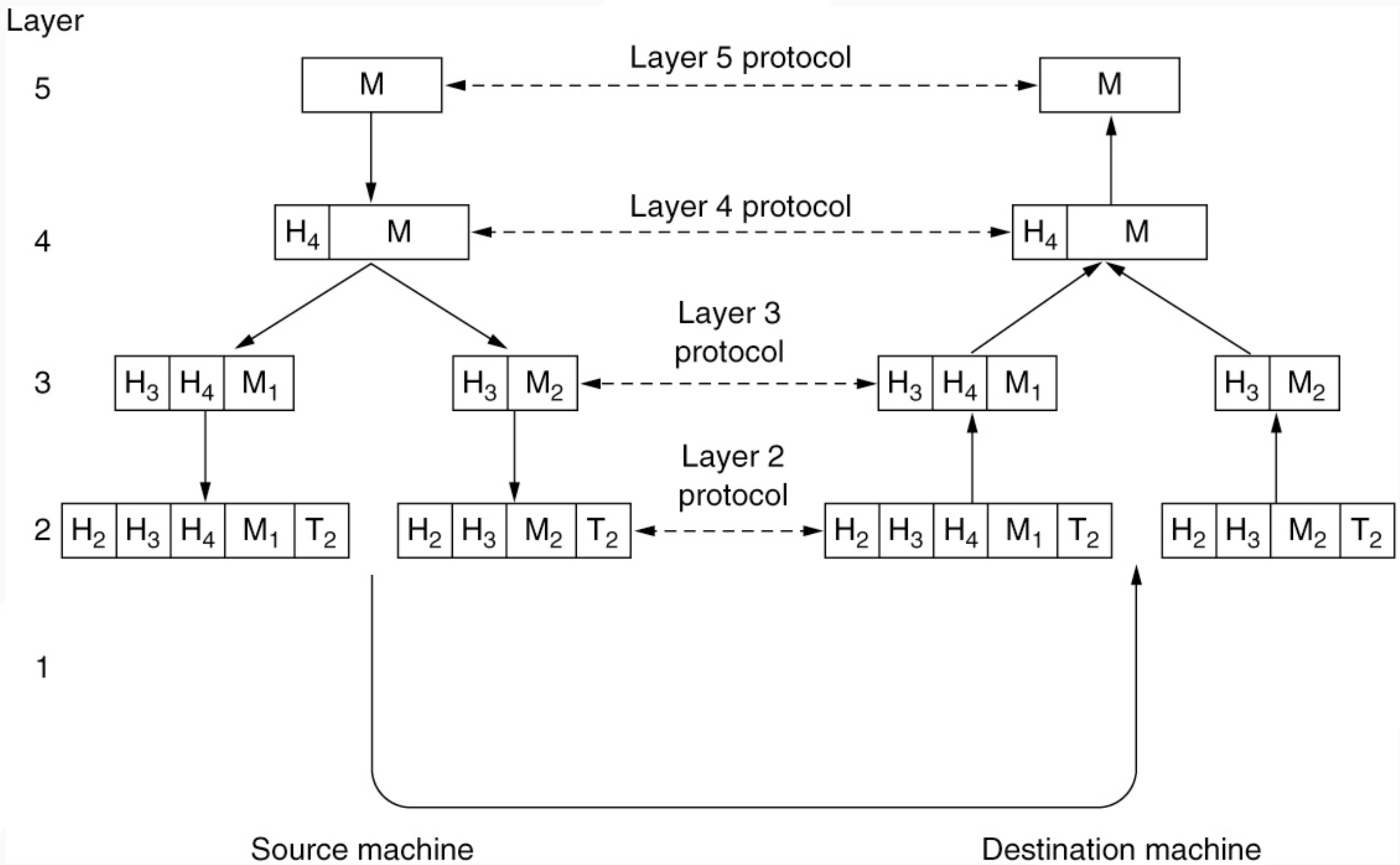
PROTOCOL LAYERING

- Networks organized as a stack of layers or levels
  - Each layer built upon the one below it
- Communication between corresponding layers
  - Use a common protocol referred to as a “layer n protocol”
  - Below layer 1 is the physical medium through which actual communication occurs
  - Interface lies between each pair of adjacent layers
- Network architecture: a set of layers and protocols
- Protocol stack: a list of the protocols used by a certain system, one protocol per layer

LAYERED PROTOCOL “STACK”



INFORMATION FLOW THROUGH LAYERS



Outline

1. Packet switching
2. Addressing

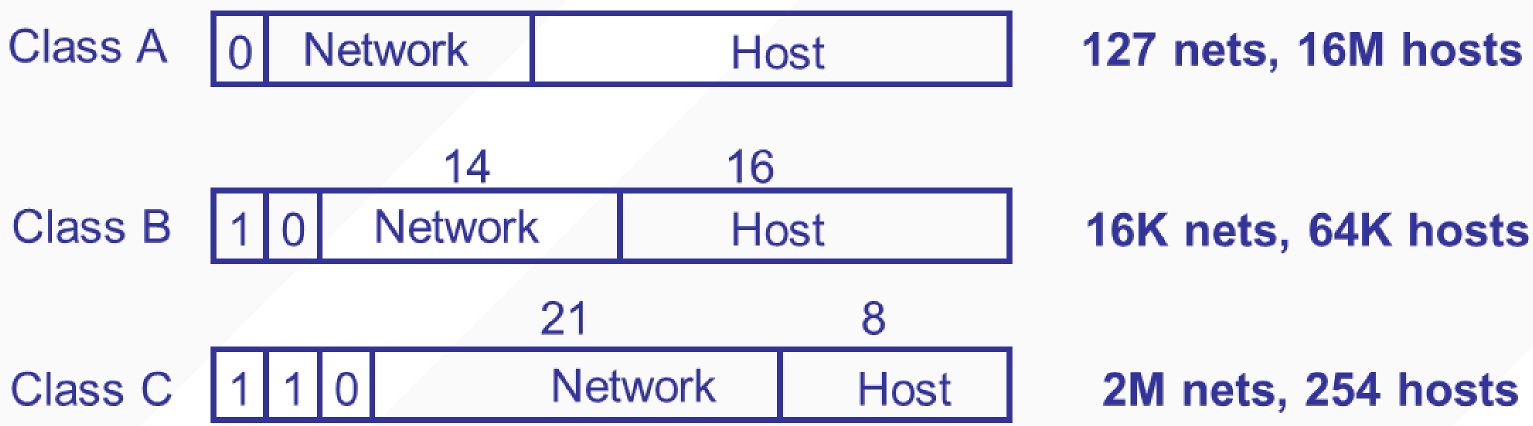
ADDRESSING CONSIDERATIONS

- Fixed length or variable length addresses?
- Issues:
  - Flexibility
  - Processing costs
  - Header size
- Engineering choice: IP uses fixed length addresses

- 32-bits in an IPv4 address
  - Dotted decimal format a.b.c.d
  - Each represent 8 bits of address
- Hierarchical: Network part and host part
  - E.g. IP address 128.54.70.238
  - 128.54 refers to the UCSD campus network
  - 70.238 refers to the host ieng6.ucsd.edu
- Which part is network *vs.* host?

CLASS-BASED ADDRESSING (NOT REALLY USED ANYMORE)

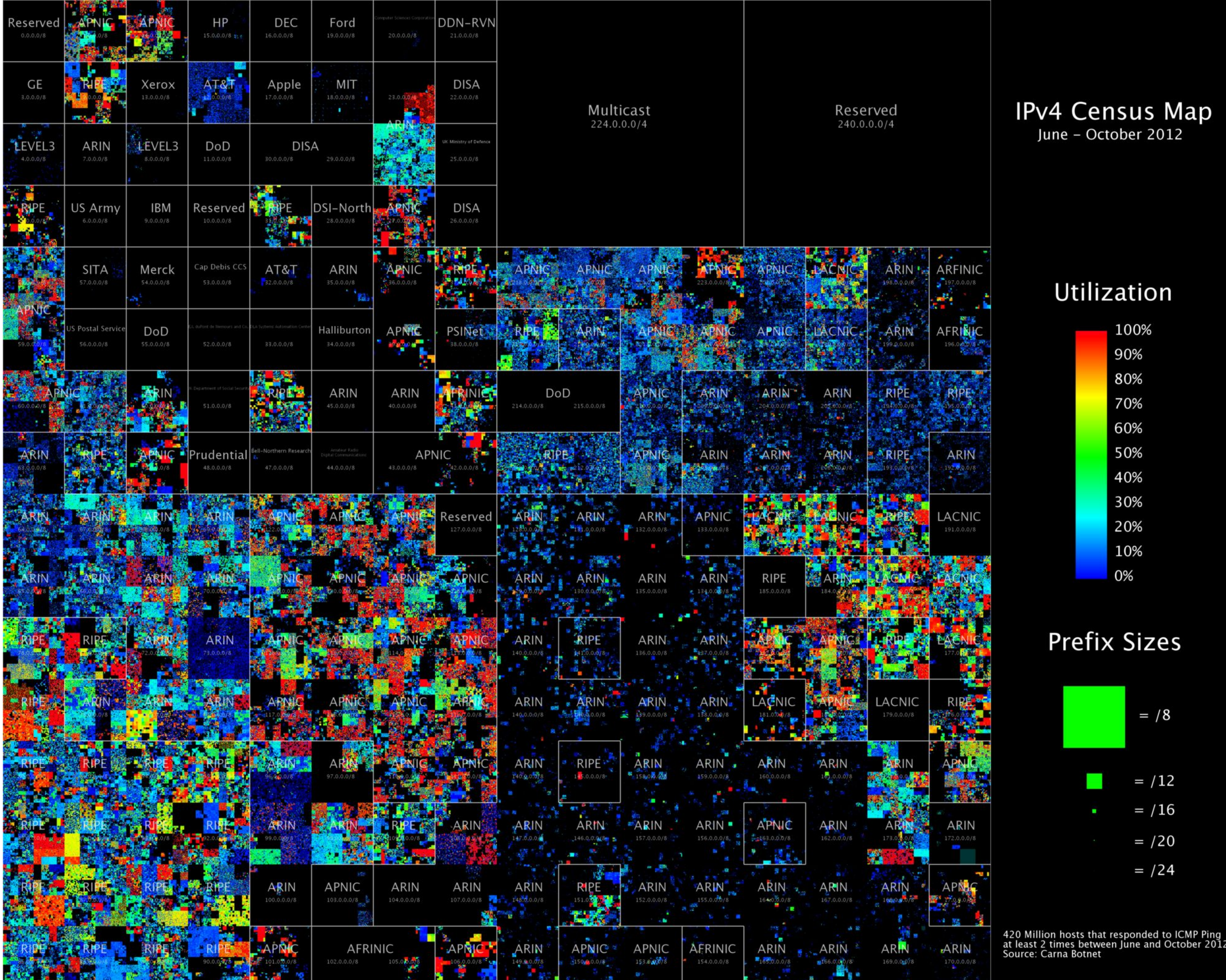
- Most significant bits determines “class” of address



- 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

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



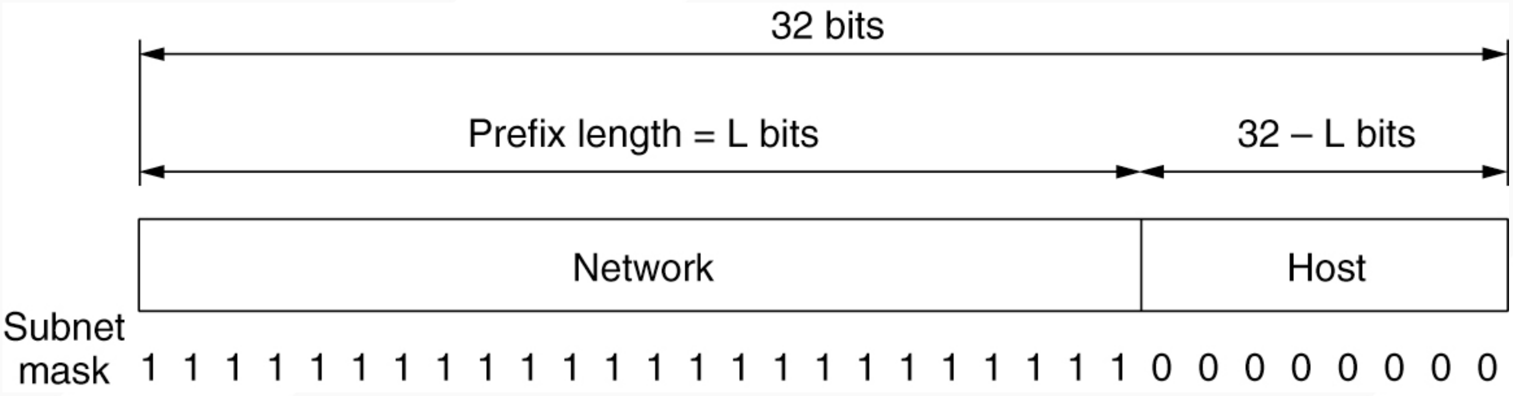
Reserved APNIC APNIC HP DEC Ford DDN-RVN
Multicast
224.0.0.0/4
Reserved
240.0.0.0/4
GE RIPE Xerox AT&T Apple MIT DISA
LEVEL3 ARIN LEVEL3 DoD DISA ARIN UK Ministry of Defence
RIPE US Army IBM Reserved RIPE DSI-North APNIC DISA
SITA Merck Cap Debis CCS AT&T ARIN APNIC RIPE APNIC APNIC APNIC APNIC APNIC LACNIC ARIN ARFINIC
APNIC US Postal Service DoD Halliburton APNIC PSINet RIPE ARIN APNIC APNIC APNIC LACNIC ARIN AFRINIC
APNIC ARIN ARIN ARIN AFRINIC DoD APNIC ARIN ARIN ARIN RIPE RIPE
ARIN RIPE APNIC Prudential Bell-Northern Research APNIC RIPE APNIC ARIN ARIN ARIN RIPE ARIN
ARIN ARIN ARIN ARIN APNIC APNIC APNIC Reserved ARIN ARIN ARIN APNIC LACNIC LACNIC RIPE LACNIC
ARIN ARIN ARIN ARIN APNIC APNIC APNIC APNIC ARIN ARIN ARIN ARIN RIPE ARIN LACNIC LACNIC
RIPE RIPE ARIN ARIN APNIC APNIC APNIC APNIC ARIN RIPE ARIN ARIN APNIC APNIC RIPE LACNIC
RIPE ARIN ARIN ARIN APNIC APNIC APNIC APNIC ARIN ARIN ARIN ARIN LACNIC APNIC LACNIC RIPE
RIPE RIPE RIPE RIPE ARIN ARIN APNIC APNIC ARIN RIPE ARIN ARIN ARIN ARIN ARIN APNIC
RIPE RIPE RIPE RIPE ARIN ARIN RIPE ARIN ARIN ARIN ARIN ARIN APNIC ARIN ARIN ARIN
RIPE RIPE RIPE RIPE ARIN APNIC ARIN ARIN ARIN RIPE ARIN ARIN ARIN ARIN ARIN APNIC
RIPE RIPE RIPE RIPE APNIC AFRINIC APNIC ARIN APNIC APNIC AFRINIC ARIN ARIN ARIN ARIN
IPv4 Census Map
June – October 2012
Utilization
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Prefix Sizes
= /8
= /12
= /16
= /20
= /24
420 Million hosts that responded to ICMP Ping
at least 2 times between June and October 2012
Source: Carna Botnet

CIDR

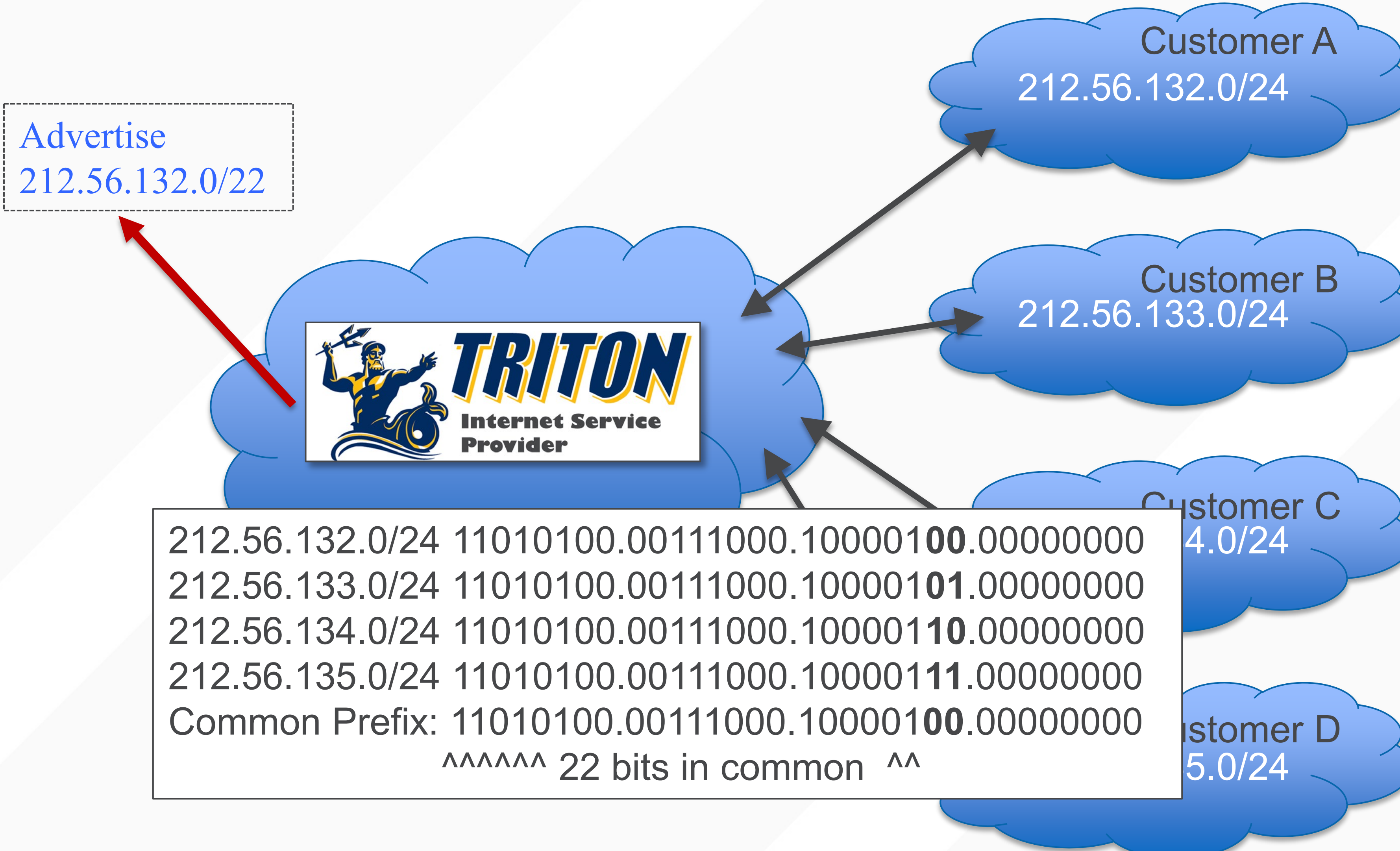
- Classless Inter-Domain Routing (1993)
  - Networks described by variable-length prefix and length
  - Allows arbitrary allocation between network and host address

  - 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



ADDRESS AGGREGATION EXAMPLE



DIVIDING ADDRESSES WITHIN YOUR NETWORK

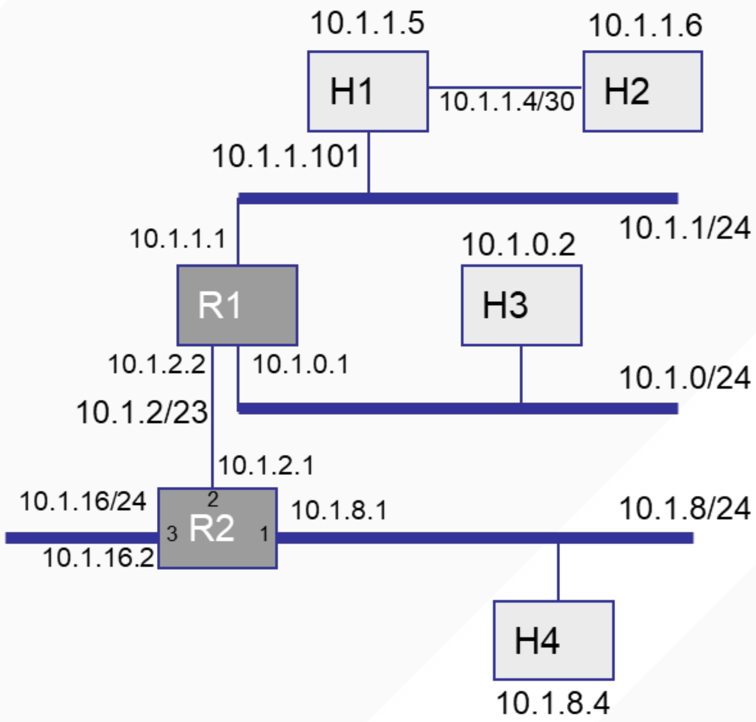
University	First address	Last address	How many	Prefix
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12.0/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

- Rule: **longest prefix matching**
  - Forward packets to the ***sub-network with the longest prefix***

FORWARDING TABLE EXAMPLE (R2)

- Packet to 10.1.1.6
- Matches 10.1.0.0/23

Destination	Next Hop
127.0.0.1	loopback
Default or 0/0	10.1.16.1
10.1.8.0/24	interface1
10.1.2.0/23	interface2
10.1.0.0/23	10.1.2.2
10.1.16.0/24	interface3

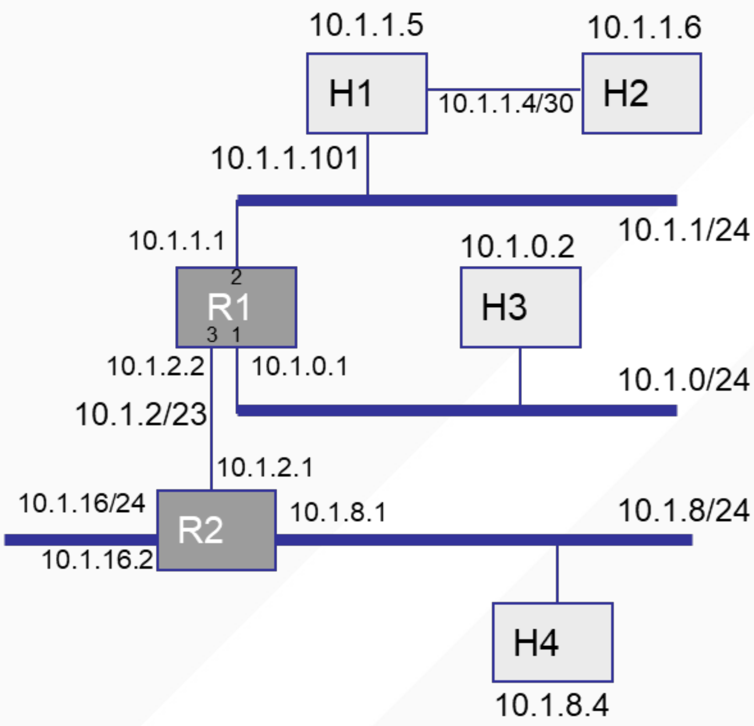


FORWARDING TABLE EXAMPLE 2 (R1)

- Packet to 10.1.1.6
- Matches 10.1.1.4/30
  - Longest prefix match

Routing table at R1

Destination	Next Hop
127.0.0.1	loopback
Default or 0/0	10.1.2.1
10.1.0.0/24	interface1
10.1.1.0/24	interface2
10.1.2.0/23	interface3
10.1.1.4/30	10.1.1.101

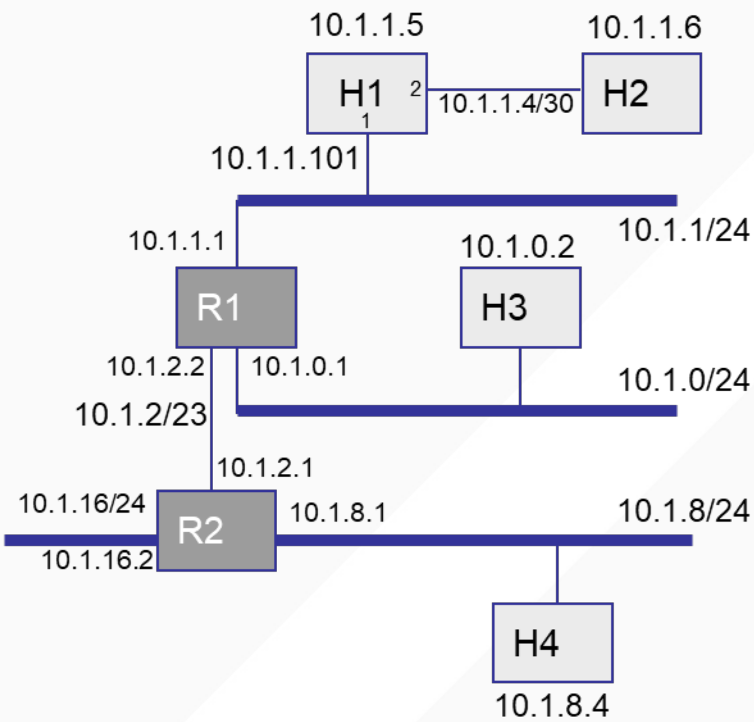


FORWARDING TABLE EXAMPLE 3 (H1)

- Packet to 10.1.1.6
- Direct route
  - Longest prefix match

Routing table at H1

Destination	Next Hop
127.0.0.1	loopback
Default or 0/0	10.1.1.1
10.1.1.0/24	interface1
10.1.1.4/30	interface2

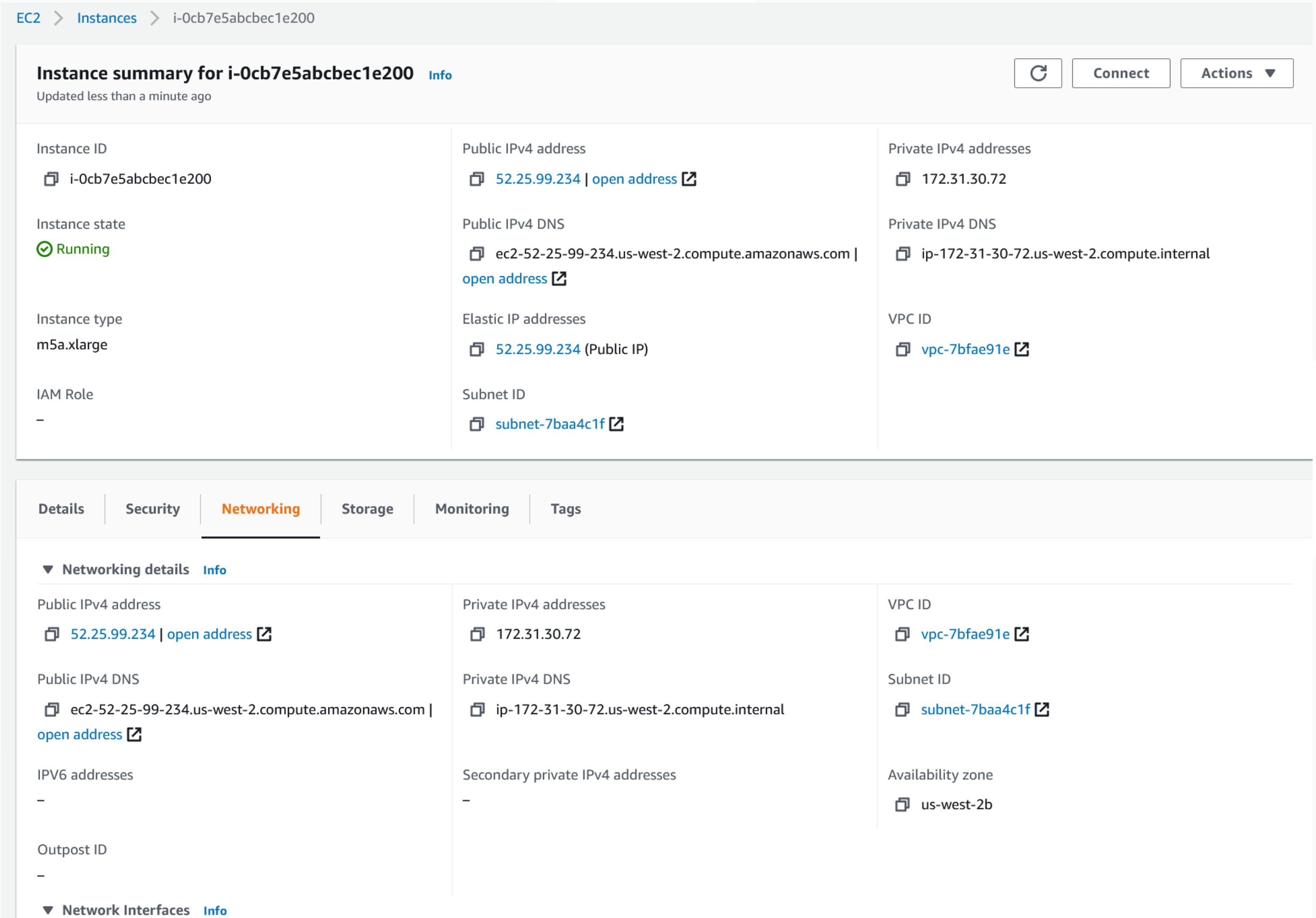


KEY TAKE-AWAY

```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
8 package(s) needed for security, out of 16 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-30-72 ~]$ ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9001
        inet 172.31.30.72  netmask 255.255.240.0  broadcast 172.31.31.255
        inet6 fe80::65:43ff:fe81:f0  prefixlen 64  scopeid 0x20<link>
        ether 02:65:43:81:00:f0  txqueuelen 1000  (Ethernet)
        RX packets 67181  bytes 5406474 (5.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 72005  bytes 5648089 (5.3 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[ec2-user@ip-172-31-30-72 ~]$
```

- To reach a computer, you need:
  - Its IP address
- But to configure that server, you need:
  - Its IP address
  - It’s netmask
  - The “default” gateway address

AMAZON EC2 VIRTUAL MACHINE VIEW

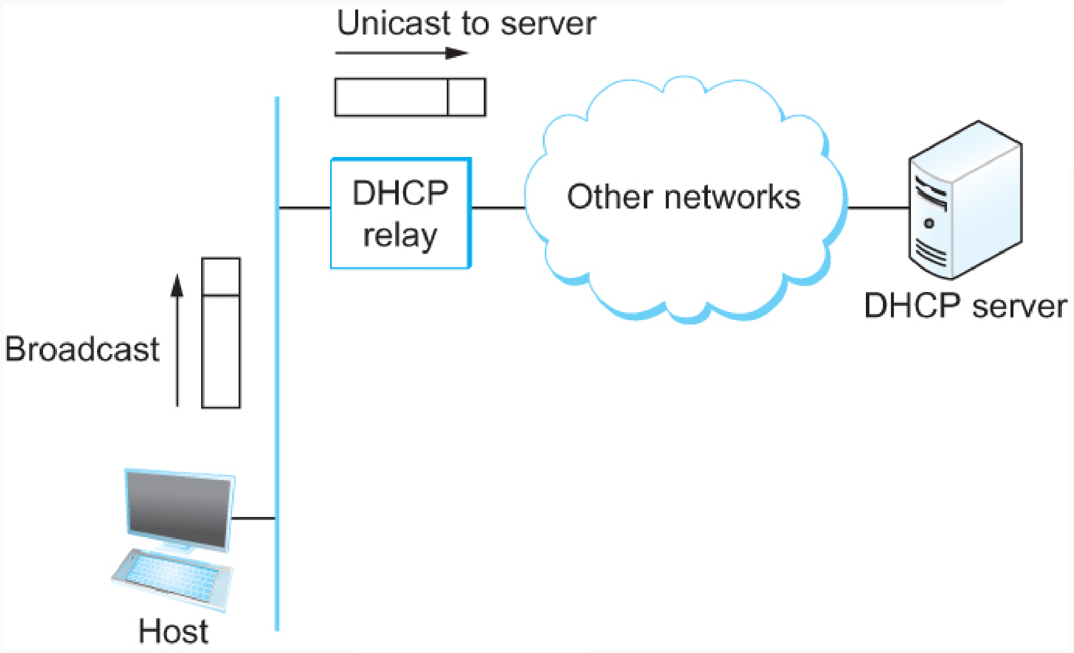


ASSIGNING ADDRESSES VIA DHCP

- DHCP server is responsible for providing configuration information to hosts
- There is at least one DHCP server for an administrative domain
- DHCP server maintains a pool of available addresses

DHCP IN ACTION

- Newly booted or attached host sends DHCPDISCOVER message to a special IP address (255.255.255.255)
- DHCP relay agent unicasts the message to DHCP server and waits for the response



DNS HOSTNAME VERSUS IP ADDRESS

- **DNS host name** (e.g. www.cs.ucsd.edu)
  - **Mnemonic** name appreciated by humans
  - **Variable length**, full alphabet of characters
  - Provides **little** (if any) information about **location**
- **IP address** (e.g. 128.112.136.35)
  - Numerical address appreciated by **routers**
  - **Fixed length**, decimal number
  - **Hierarchical** address space, related to host **location**

MAPPING NAMES TO ADDRESSES

```
GETADDRINFO(3)            Linux Programmer's Manual            GETADDRINFO(3)

NAME
       getaddrinfo,  freeaddrinfo,  gai_strerror - network address and service
       translation

SYNOPSIS
       #include <sys/types.h>
       #include <sys/socket.h>
       #include <netdb.h>

       int getaddrinfo(const char *node, const char *service,
                       const struct addrinfo *hints,
                       struct addrinfo **res);

       void freeaddrinfo(struct addrinfo *res);

       const char *gai_strerror(int errcode);
```

A FACT ABOUT NAME→ADDRESS MAPPINGS

```
struct addrinfo {
    int              ai_flags;
    int              ai_family;
    int              ai_socktype;
    int              ai_protocol;
    socklen_t        ai_addrlen;
    struct sockaddr *ai_addr;
    char            *ai_canonname;
    struct addrinfo *ai_next;
};
```

- Looking up a name results in a *linked list* of results. Why?
  - Support for newer "IPv6 (128-bit)" and classic "IPv4 (32-bit)"
  - A name can map to many IP addresses. E.g. google.com