# Operating Systems

## Lecture 18: Networking

**Hong Xu**

https://github.com/henryhxu/CSCI3150

# Outline

- Brief history of computer networking and the Internet
- Basic concepts
- Layering architecture
- TCP/IP

# History

- How can computers be connected, and talk to each other?
- Roots can be traced back to telephone networks
- L. Kleinrock had first published work in ’61.
  - Showed packet switching was effective for bursty traffic
- Packet switching was developed and formed basis of ARPAnet (Advanced Research Projects Agency Network)

# Evolution – 1969

- First link on ARPAnet between UCLA and SRI (Stanford Research Institute)

```
6  Test messages between UCLA-SRI 10/15/69

   6a  Network configuration

SRI   |
      |
      |
      |
      |
      |
      |
      |
      |
UCLA  |
```

http://personalpages.manchester.ac.uk/staff/m.dodge/cybergeography/atlas/historical.html

# Evolution – 1969

- By the end of 1969, four nodes, UCLA, SRI, UCSB, Utah

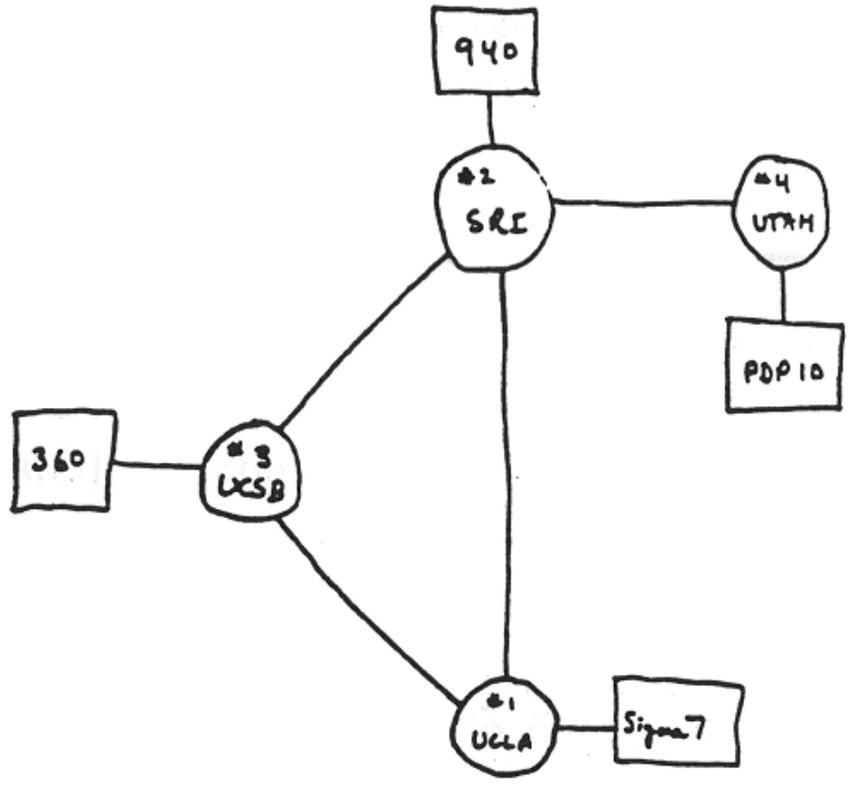
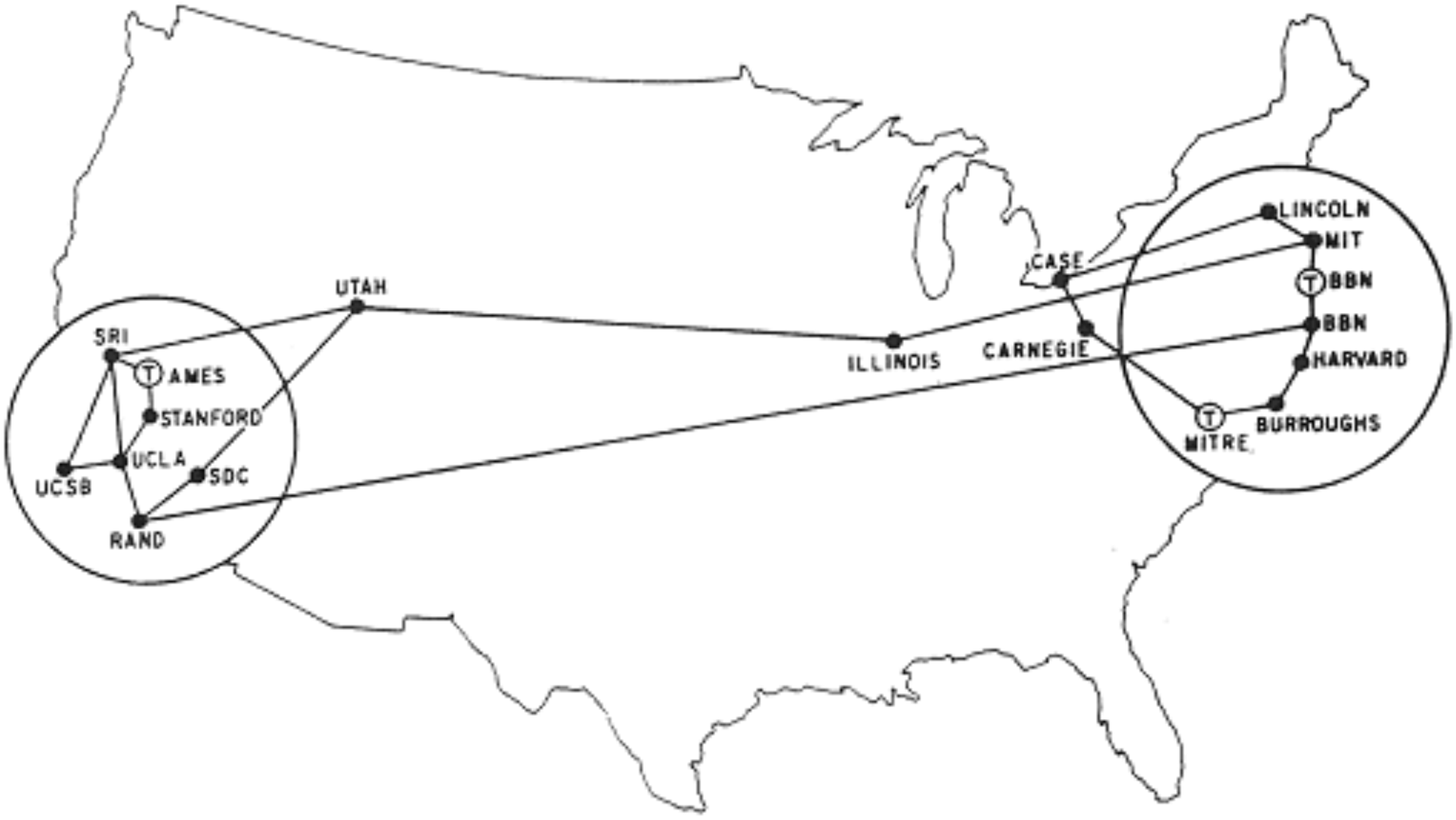



**FIGURE 6.2** Drawing of 4 Node Network (Courtesy of Alex McKenzie)

http://personalpages.manchester.ac.uk/staff/m.dodge/cybergeography/atlas/historical.html

# Evolution – 1971

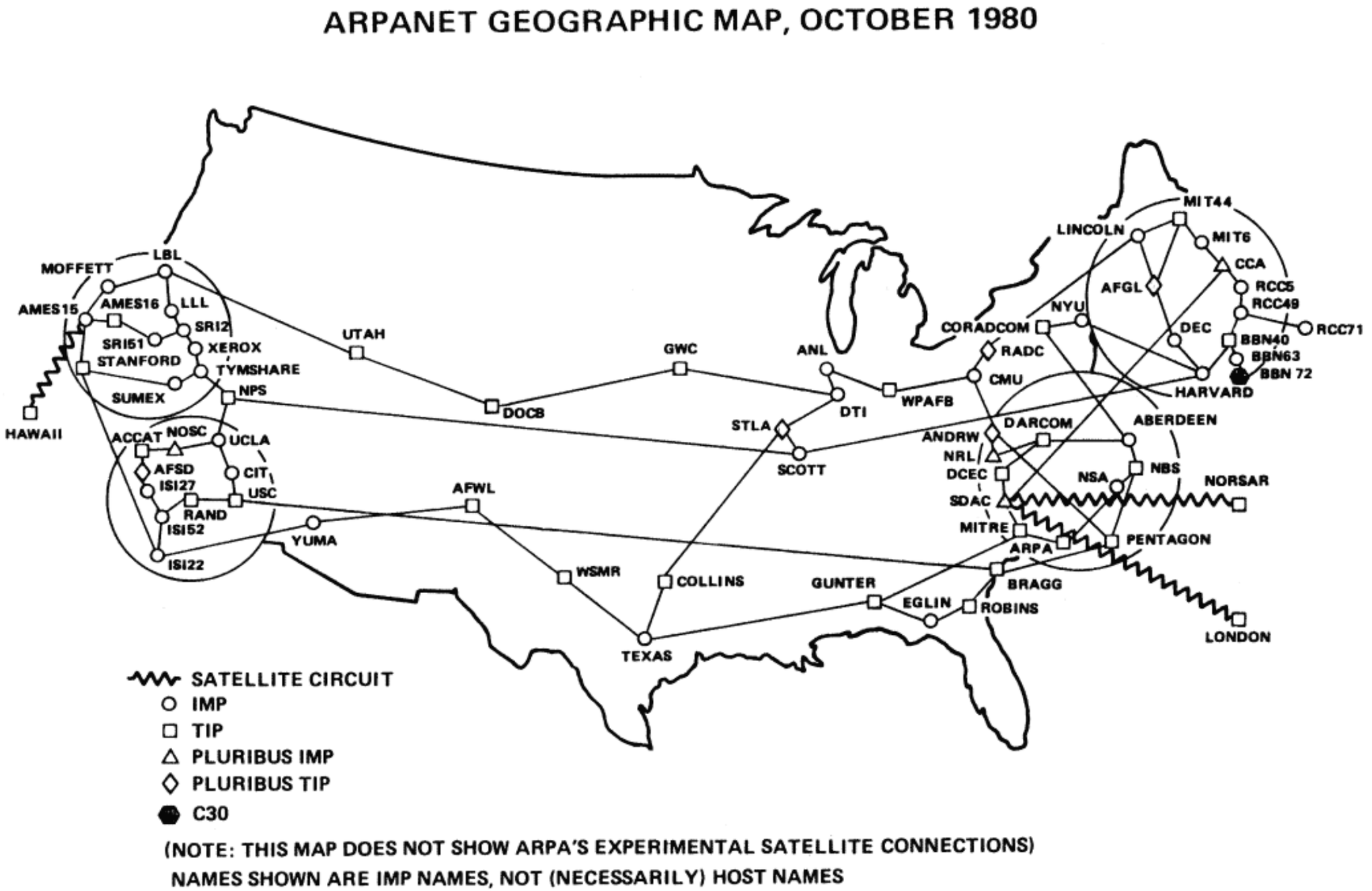
- September 1971




MAP 4 September 1971

http://personalpages.manchester.ac.uk/staff/m.dodge/cybergeography/atlas/historical.html

# Evolution – 1980




http://personalpages.manchester.ac.uk/staff/m.dodge/cybergeography/atlas/historical.html

# Evolution – Now

# Milestones

- In 1973, R. Metcalfe invented Ethernet (in Xerox PARC)
- In 1974, Vinton Cerf and Robert Kahn invented TCP/IP
- The term “internet” was adopted around 70’s as an abbreviation of *internetworking*
- Killer applications: Email, Web, peer-to-peer (P2P), search engine, video, mobile, social media

# Some basic concepts

# Building blocks

- Nodes
  - end-hosts, or hosts: PC, server
  - switches/routers, middleboxes
- Links: coax cable, optical fiber, wireless, …
  - point-to-point
  - multiple access (shared)

# Switching approaches

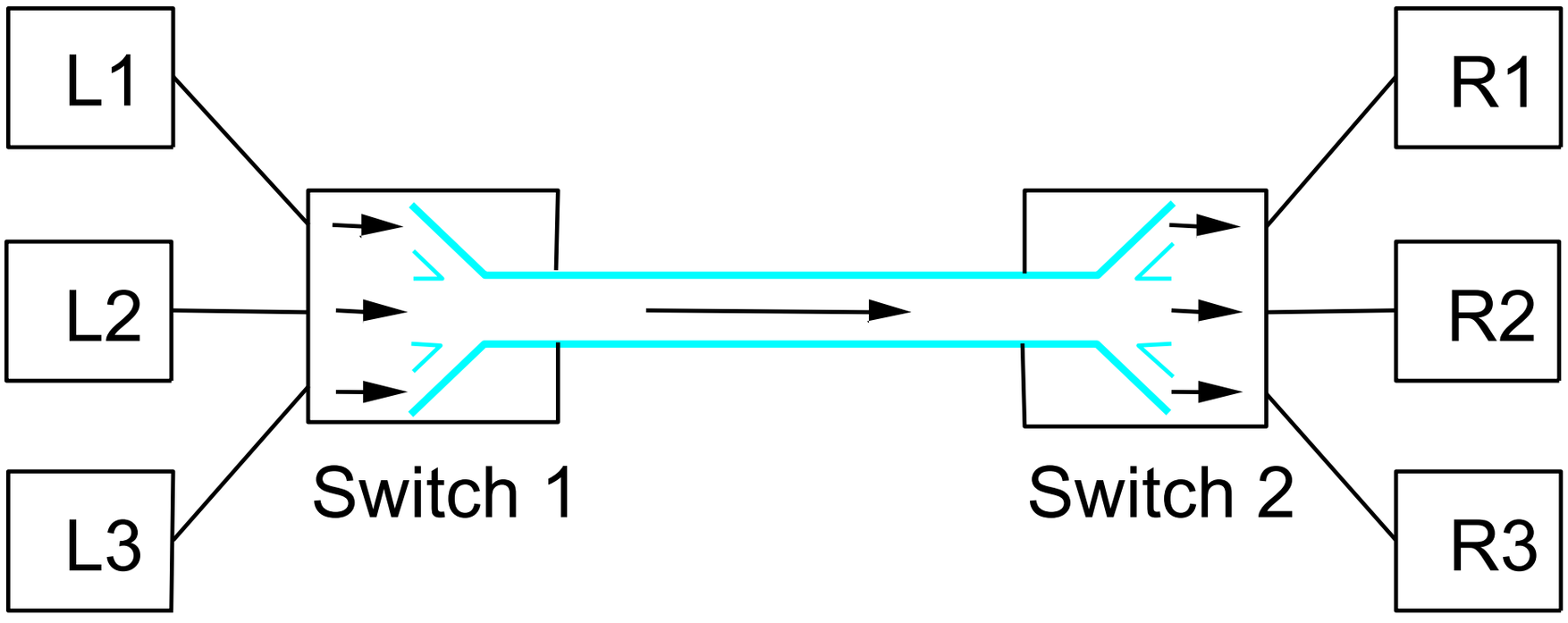
- Circuit switching
  - (early) telephone networks
  - Connection oriented, dedicated comm. link between two nodes
- Packet switching
  - computer networks
  - Connection-less, shared comm. links, no set-up
  - Data is divided into packets and transferred independently

# Addressing, routing

- Address: byte-string that identifies a node
- Routing: process of forwarding messages to the destination node based on its address
- Types of addresses:
  - unicast: node-specific
  - broadcast: all nodes on the network
  - multicast: a group/subset of nodes

# Multiplexing

- Time-division multiplexing (TDM)
- Frequency-division multiplexing (FDM)



# Statistical multiplexing

- On-demand time-division. Schedule link on a per-packet basis
- Packets that content for the link enter a/some buffer(s)/queue(s)
- Different scheduling disciplines can be used to decide which packet to transmit next
- Buffer (queue) overflow is called congestion

# Statistical multiplexing

- Say host A sends data to host B in a bursty manner. On average A generates 100Kbps in 10% of the time, and idles for 90% of the time

- Circuit switching, given a link with 1Mbps, how many hosts can be supported simultaneously?
  - 10 with no delay (no queueing)
- Packet switching, how many?
  - About 30 with very low probability of queueing delay

# Statistical multiplexing gain: not everybody is talking!

# Performance metrics

- Bandwidth, throughput
  - amount of data transmitted per unit time
  - 1 Mbps = $10^6$ bits per second, 1 MBps = 8 Mbps
- Latency (delay)
  - total time from one end to another
  - latency = propagation + transmission + queue
  - propagation = distance / C, transmission = size / bandwidth

# Performance metrics

- Relative importance
- 1B flow: queueing delay dominates
  - 1ms/100ms vs. 1MBps/100MBps
- 25MB flow: transmission delay, i.e. throughput, dominates
  - 1ms/100ms vs. 1MBps/100MBps

# A layering architecture

# Layering

- How to build an internet that connect all different kinds of networks, each of which may use a completely different technology?

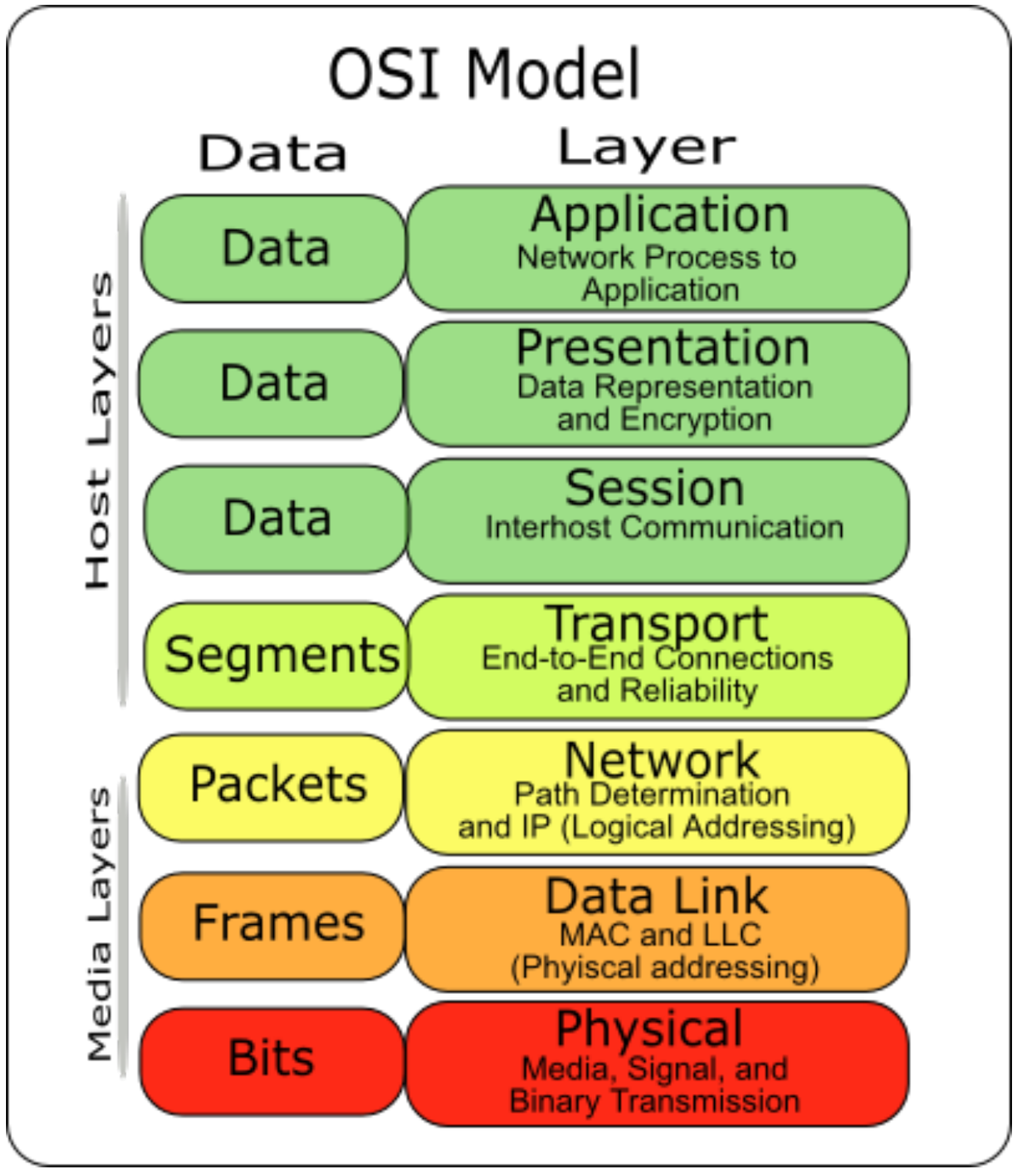
- Answer: layering, with well-defined interfaces

- Best summarized by David Clark, MIT, “The Design Philosophy of the DARPA Internet Protocols”

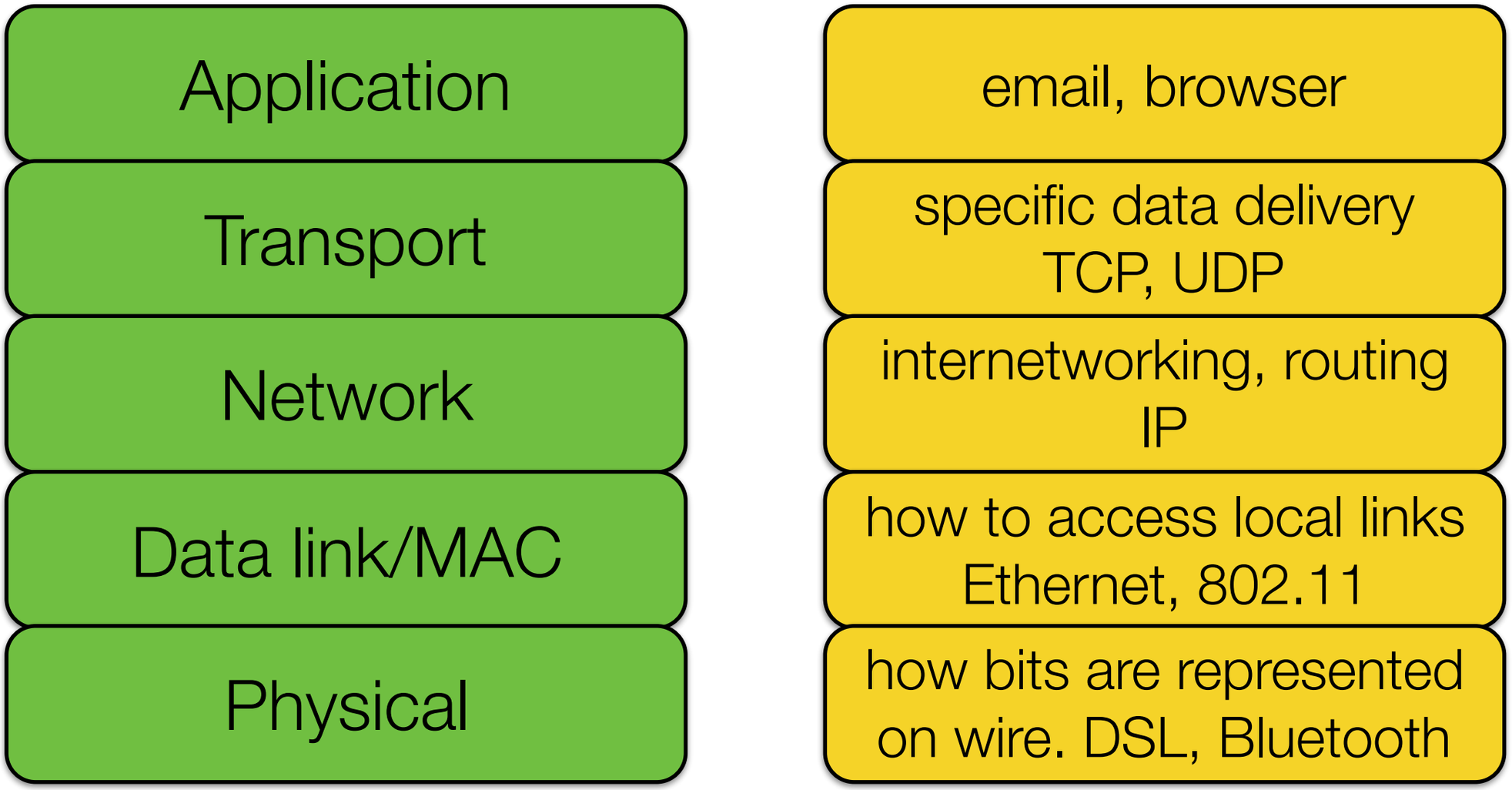
# Why layering?

- Functions are separated into layers. Layers are “blackboxes” to the upper- or lower-layer protocols.
- Allow distinct technologies for the same function.
  - physical: Ethernet, 3G, WiFi, satellite, optical, quantum, etc.
- Allow them to inter-operate using standardized interfaces

# OSI reference model

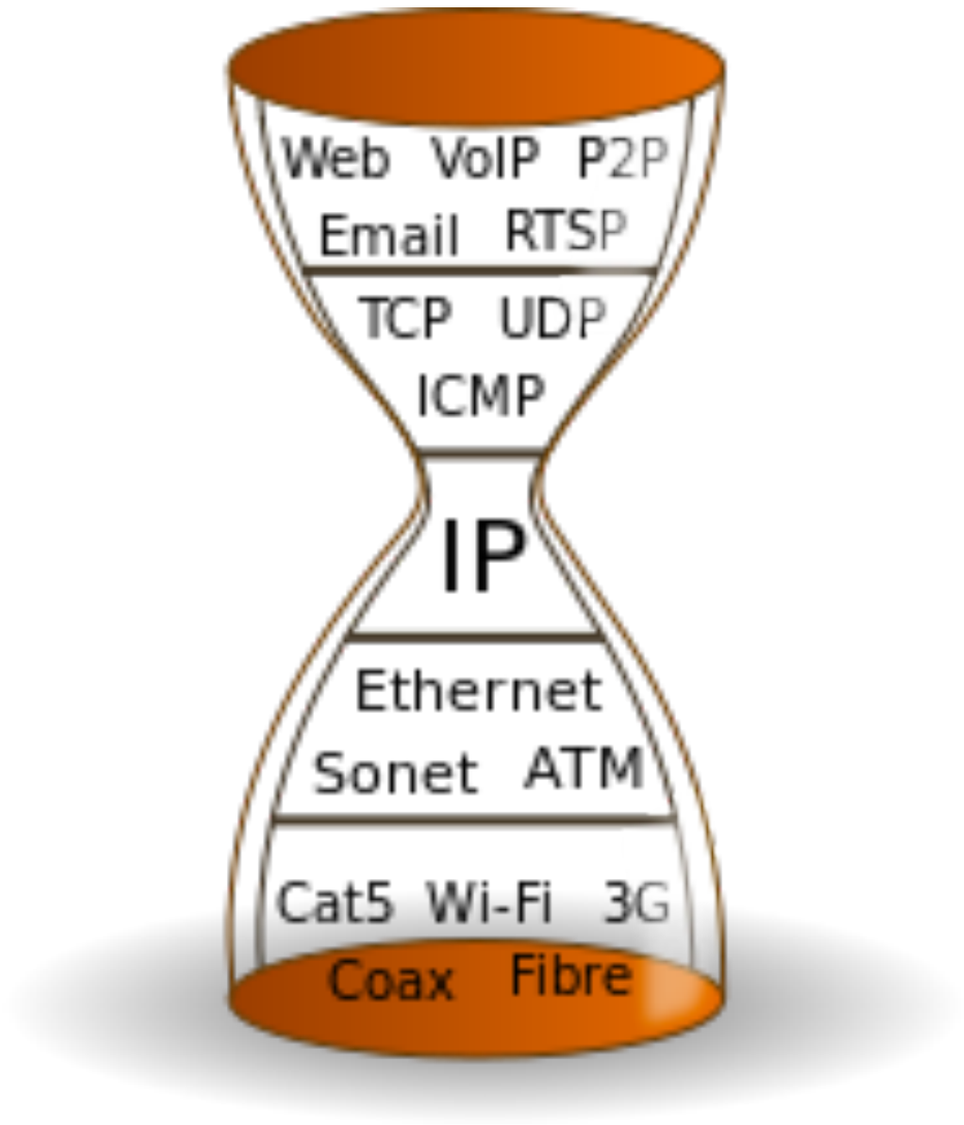


# More popular TCP/IP model



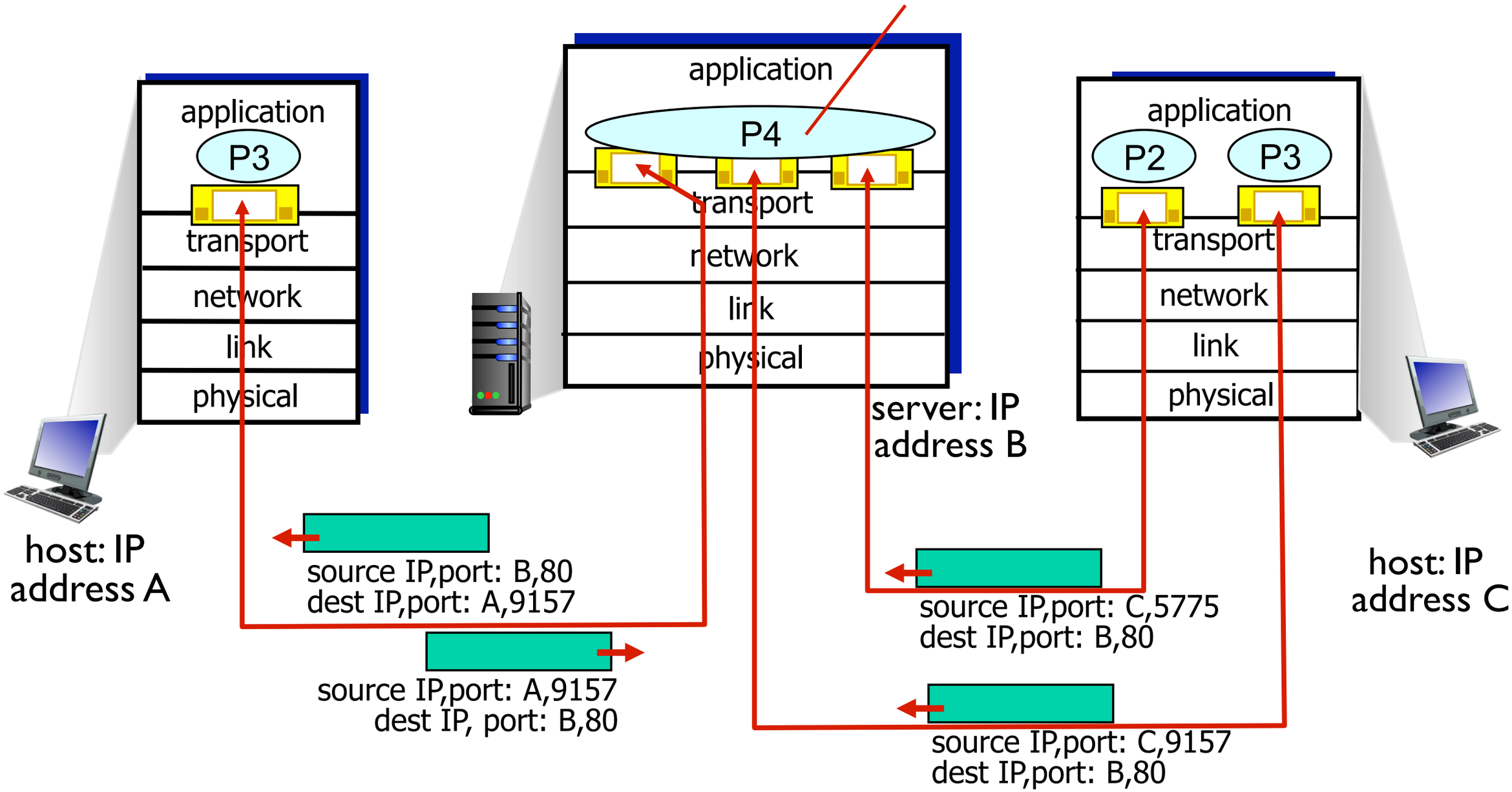
# Hourglass

- Our protocol stack



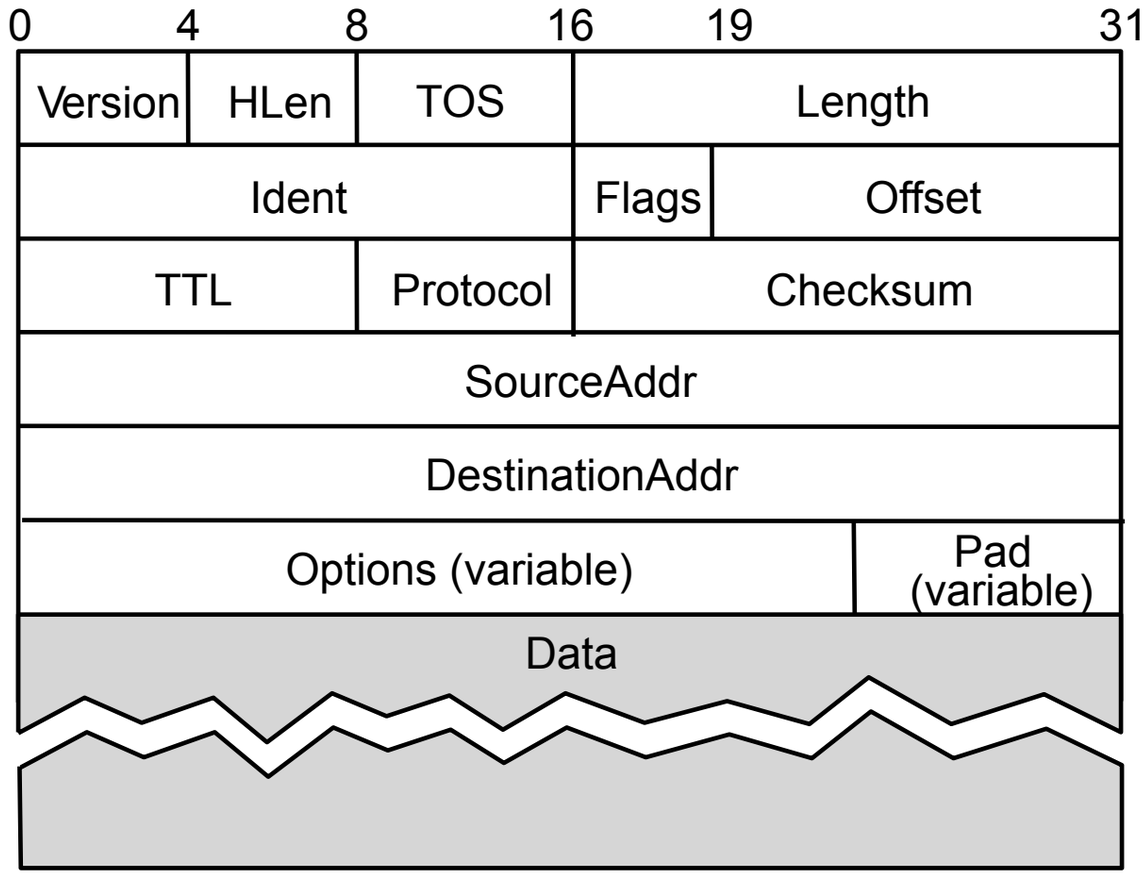
# TCP/IP

# TCP/IP


application
P3
transport
network
link
physical
application
P4
transport
network
link
physical
application
P2
P3
transport
network
link
physical
host: IP address A
server: IP address B
host: IP address C
source IP,port: B,80
dest IP,port: A,9157
source IP,port: A,9157
dest IP, port: B,80
source IP,port: C,5775
dest IP,port: B,80
source IP,port: C,9157
dest IP,port: B,80

# Internet protocol – IP

- Connectionless, packet-based
- Best-effort, inherently unreliable
  - packets may be lost, dropped, delayed, delivered out-of-order
  - no guarantee
- Header format



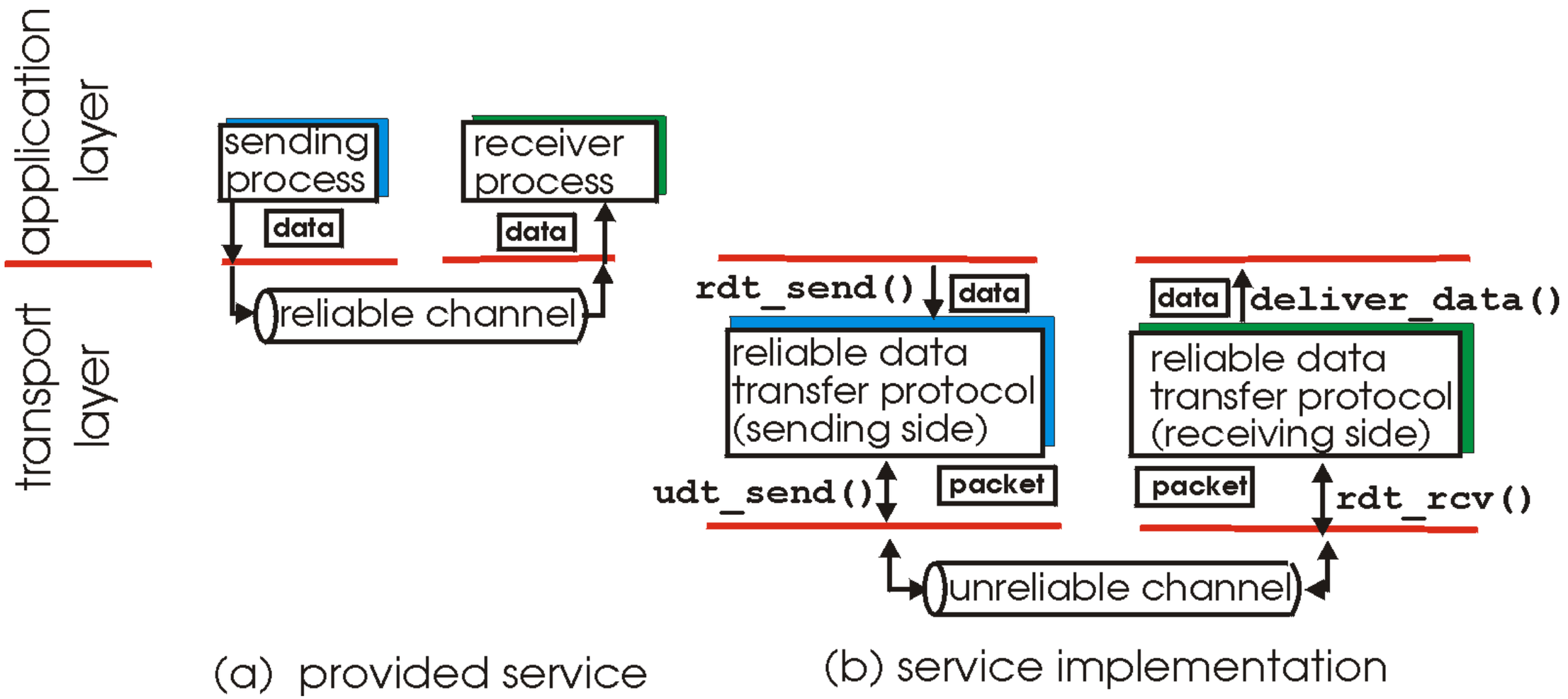
# Routing

- Routing in a nutshell:
  - Every packet header has the destination address
  - If the router is directly connected to the destination network, forward to the host
  - Else, forward to some other router
- Example

| Network | Next hop |
|---|---|
| Network 1 | R1 |
| Network 2 | R3 |
| All else | R0 |

# Transport layer

- Provide reliable data transfer to applications over unreliable IP network




(a) provided service

(b) service implementation

# TCP overview

- TCP: Transmission Control Protocol
- Point-to-point; reliable in-order byte stream (sequence numbers); full duplex
- Connection-oriented, handshaking before data exchange
- Flow control: not sending too fast for the receiver to process (src and dot may have different network speeds). recv window, rwnd
- Congestion control: avoid congestion collapse (too many sources sending too much). congestion window, cwnd

# Reliable transfer

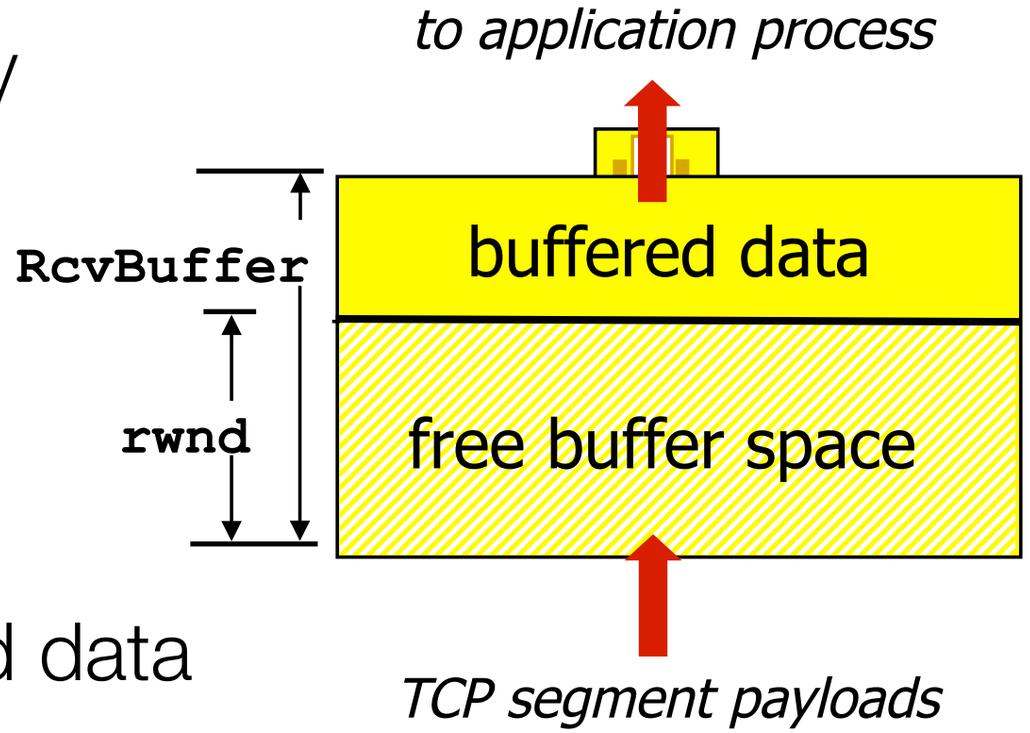
- Use sequence number, and send acknowledgement packets indicating the sequence number up to which the receiver has received

- If some packets are received out-of-order, or are lost, the receiver will only ACK the last seq num of the contiguous stream.

- Packet loss can be detected by timeouts and duplicated ACKs.

# TCP flow control

- Receiver controls sender, so the sender won't overflow the receiver's buffer by sending too fast
- Receiver advertises buffer space by including a rwnd value in the header
- Sender limits the amount of un-acked data to receiver's rwnd value




*receiver-side buffering*

# TCP congestion control

- “Too many sources sending too fast for the network to handle”
- Manifestations:
  - lost packets (buffer overflow at routers)
  - long delays (queueing in router buffers)

# TCP congestion control

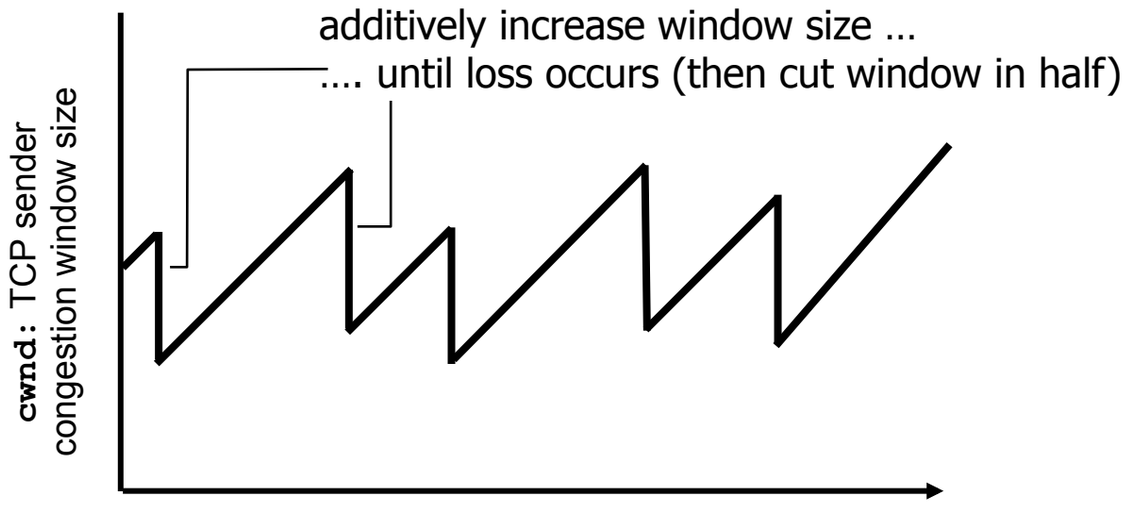
- TCP relies on packet drops as a signal of congestion. Packet drops are detected by duplicated ACKs. Thus when TCP sees duplicated ACKs, it will interpret as congestion is experienced.

- However
  - Packet drops may not be caused by congestion, e.g. in wireless networks
  - Duplicated ACKs may not be caused by packet drops. They may be due to a change of network path and some packets arrive late but not dropped

# TCP congestion control

- Approach: AIMD
- Sender increases sending rate (window size), probing for unused bandwidth, until loss occurs
- additive increase: increase cwnd by 1 MSS (maximum segment size) every RTT until loss detected
- multiplicative decrease: cut cwnd by half when loss is detected

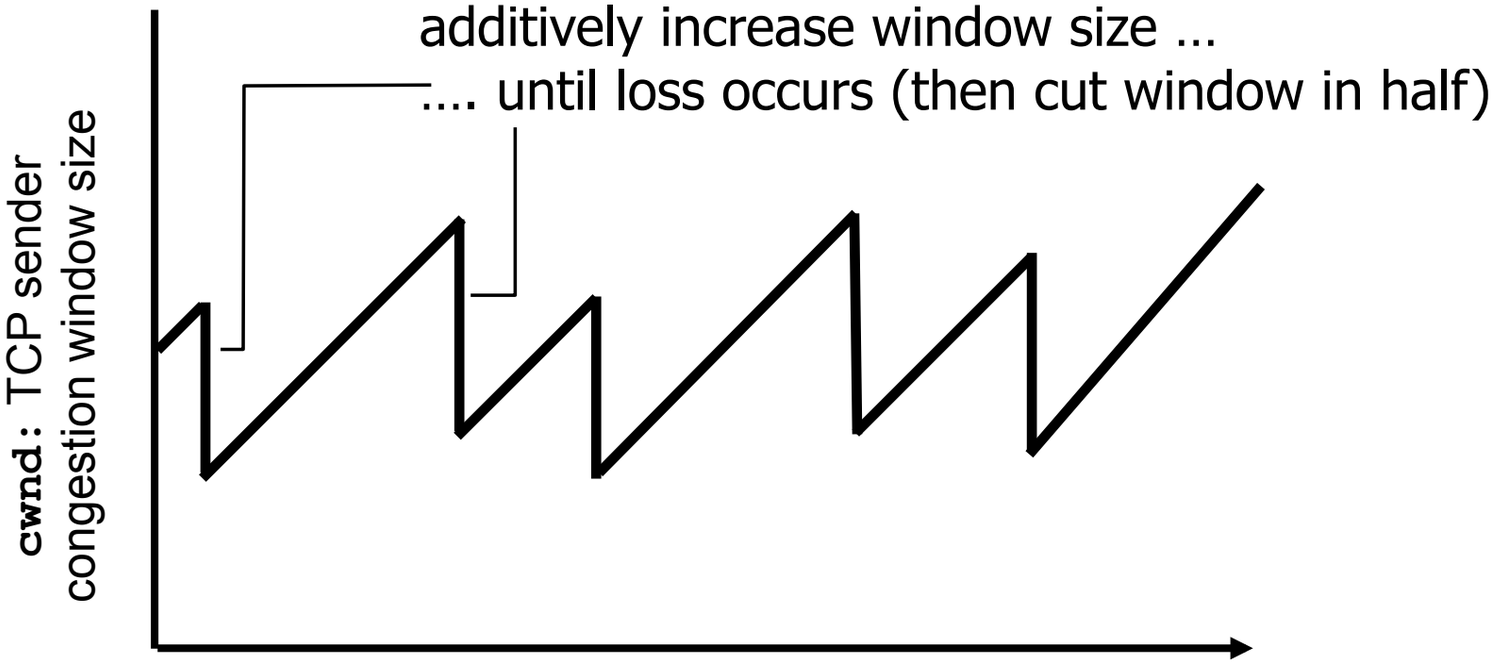
AIMD saw tooth behavior: probing for bandwidth



# TCP congestion control

- Typical TCP cwnd behavior

AIMD saw tooth behavior: probing for bandwidth



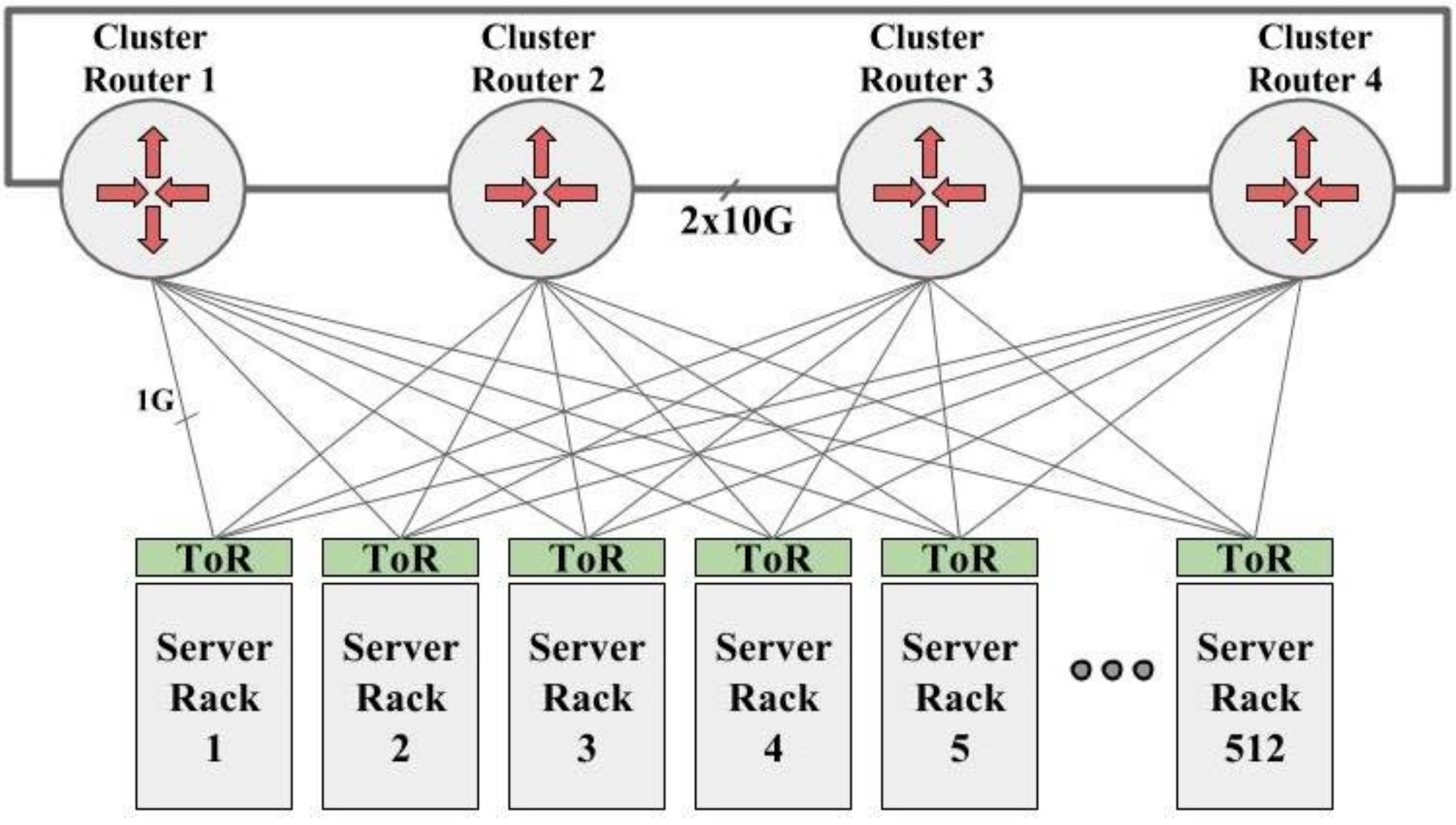
# Topology

# or an evolution of Google's network topologies

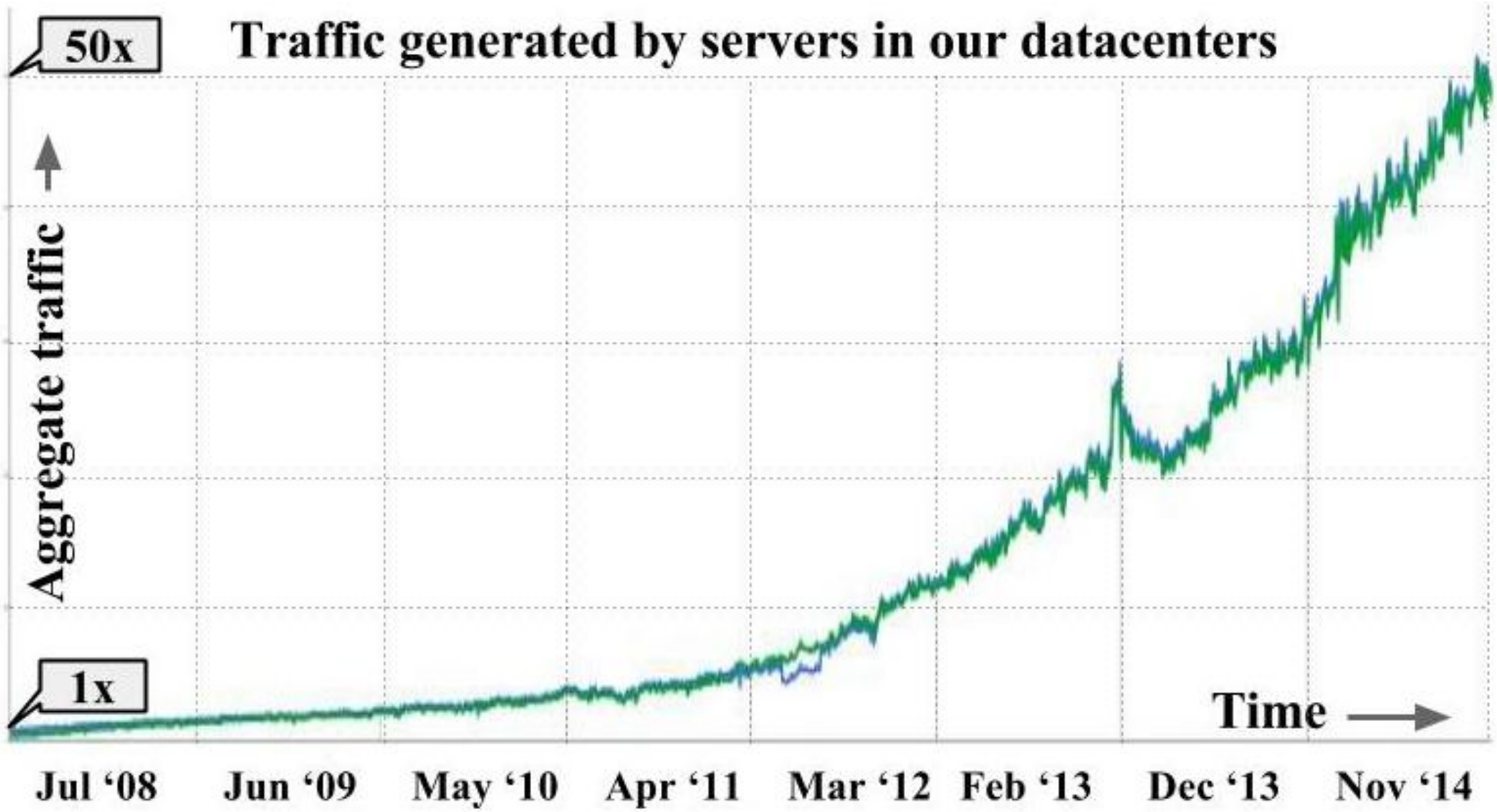
Credit:  A. Singh et al., "Jupiter rising: A decade of Clos topologies and centralized control in Google's datacenter network," ACM SIGCOMM'15.

# 2004: four-post cluster

- Supported 20k servers per cluster

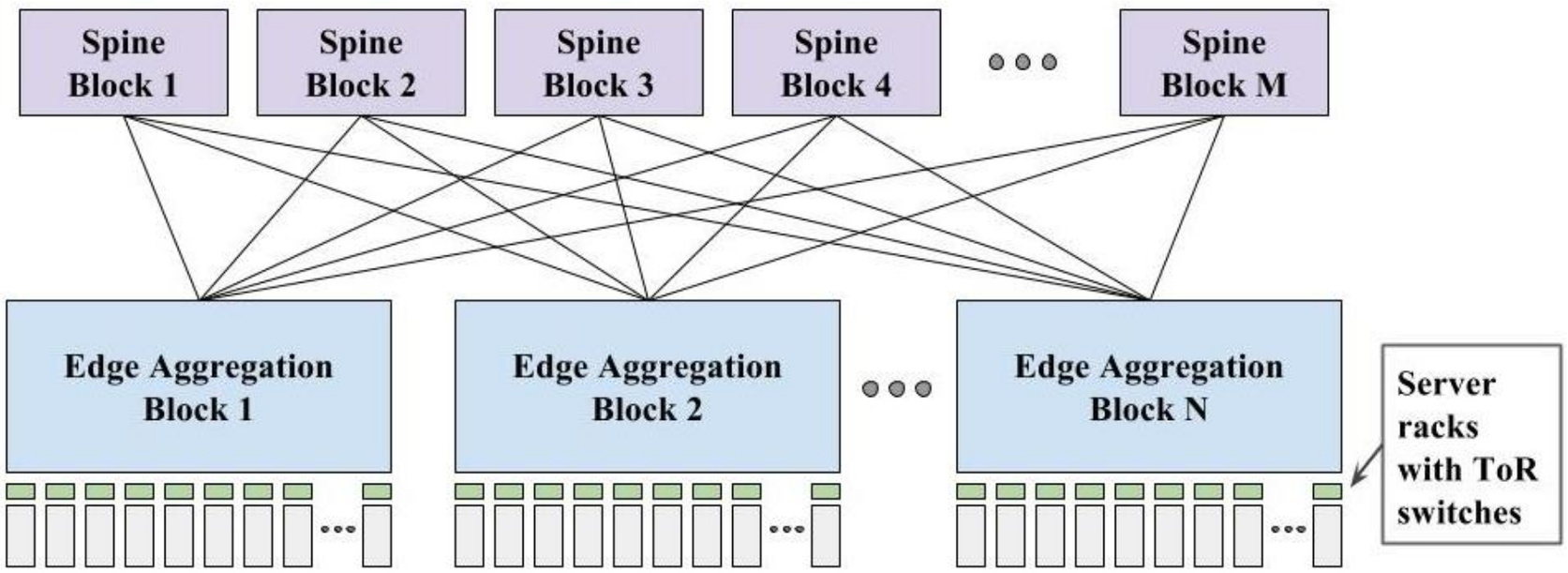


# But traffic keeps growing



# How to scale the network

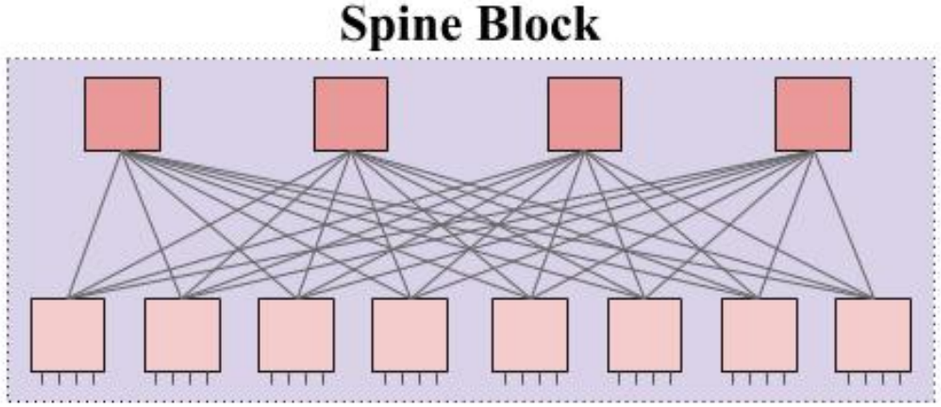
- Buying the largest switches with the most ports doesn't scale well
- And it's expensive
- Solution: Clos topologies



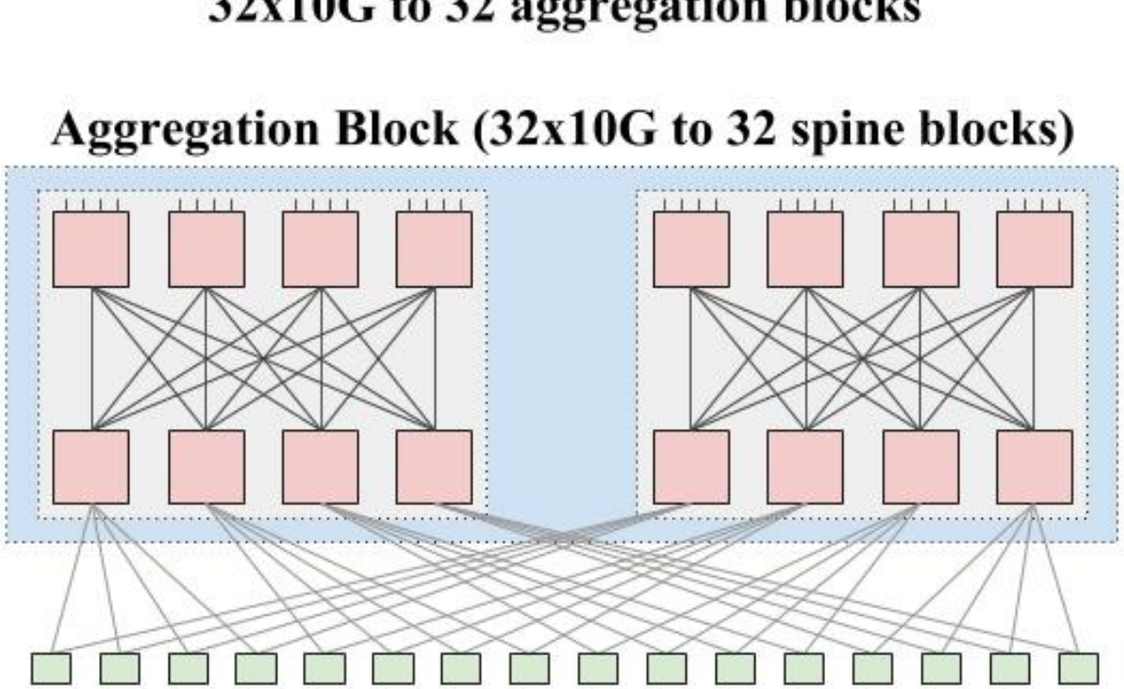
# A practical approach to scale

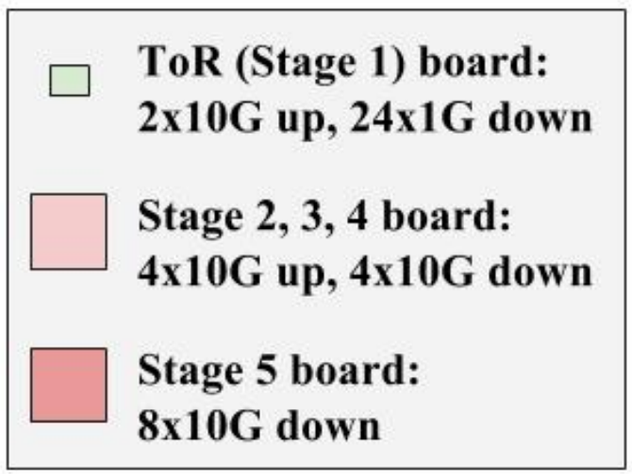
- Use Clos topologies:
  - Use many low-radix switches in multiple stages to scale to arbitrary size
  - Substantial path diversity and redundancy
- Merchant silicons
  - off-the-shelf, allows regular and rapid upgrades

# [2005] Firehose 1.0



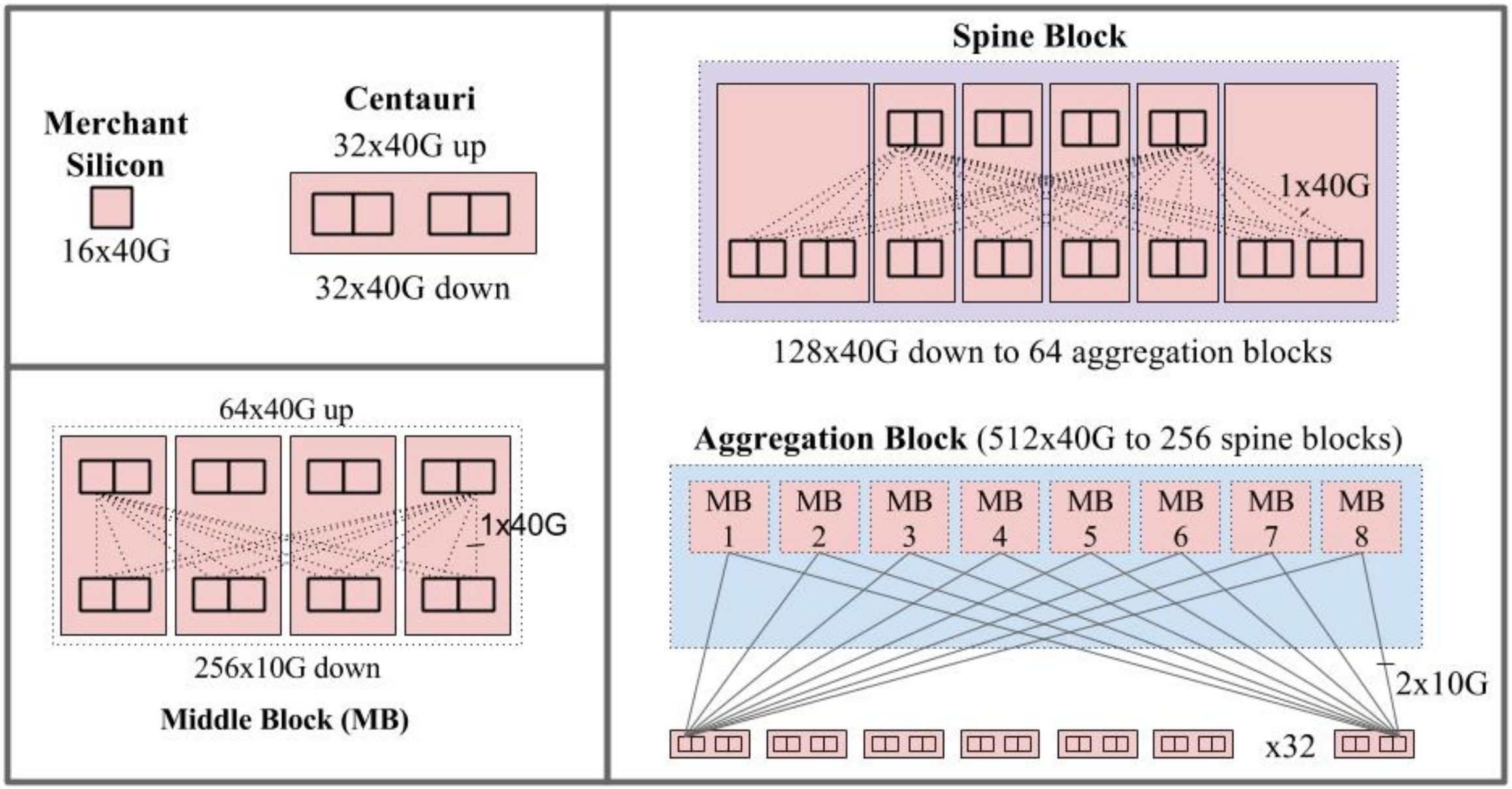




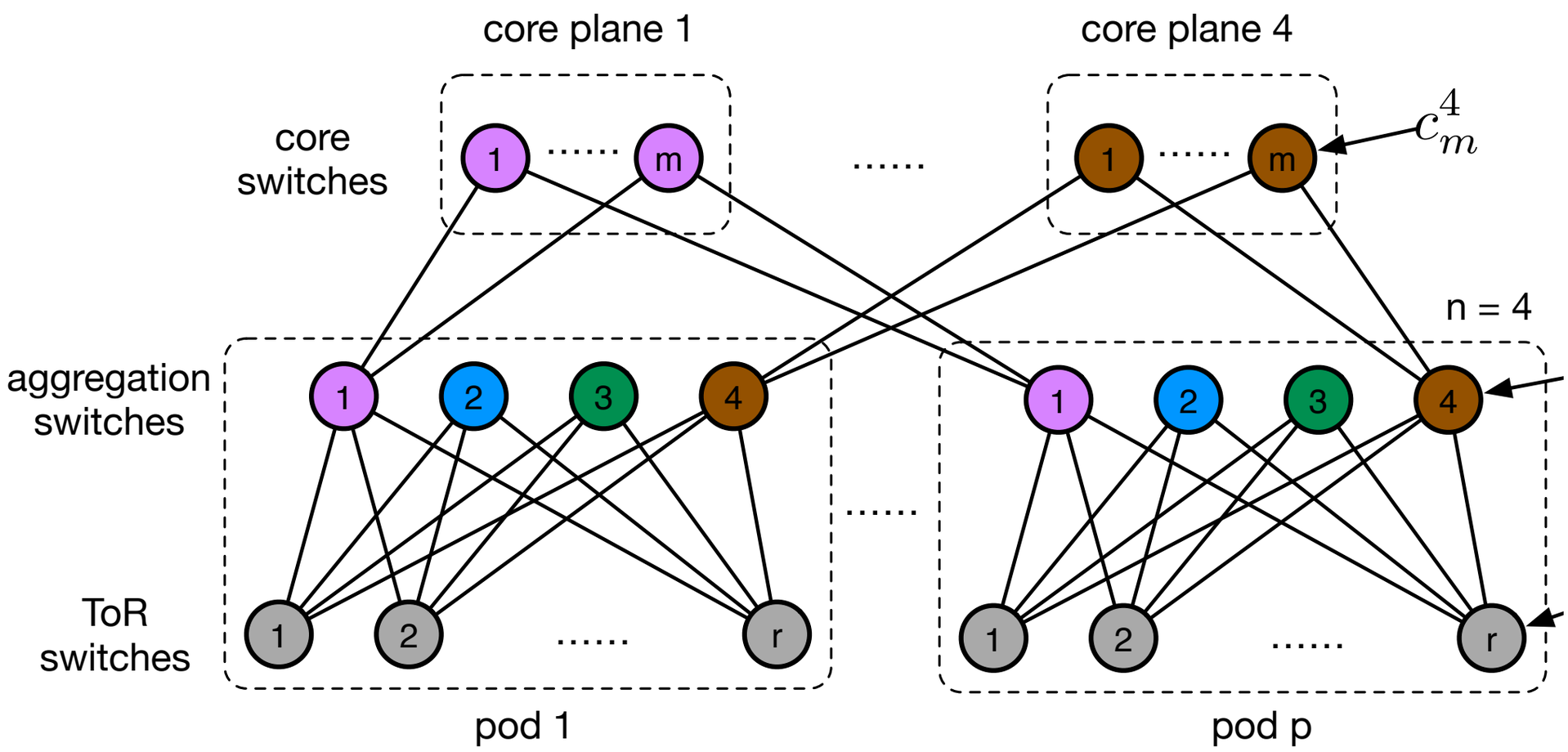




- An aggr block has 16 ToRs (320 machines). 32 spine blocks each connect to 32 aggr blocks, resulting in a fabric that scales to 10K machines with 1G average bandwidth

# [2012] Jupiter



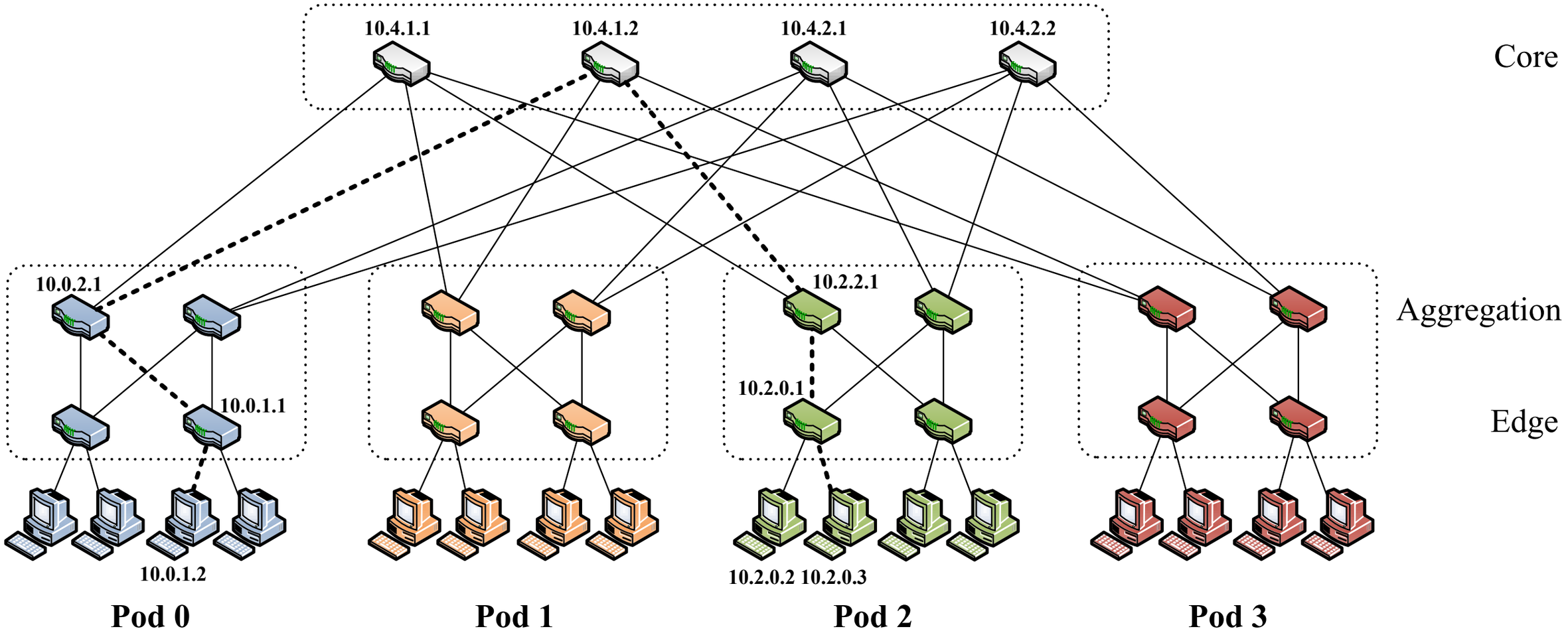
# Facebook's fabric




With 96 pods, the topology can accommodate 73,728 10Gbps hosts. In Facebook's Altoona data center, each aggregation switch connects to 48 ToR switches in its pod, and 12 out of the 48 possible core switches on its plane, resulting in a 4:1 oversubscription.

# Academia: Fat-tree

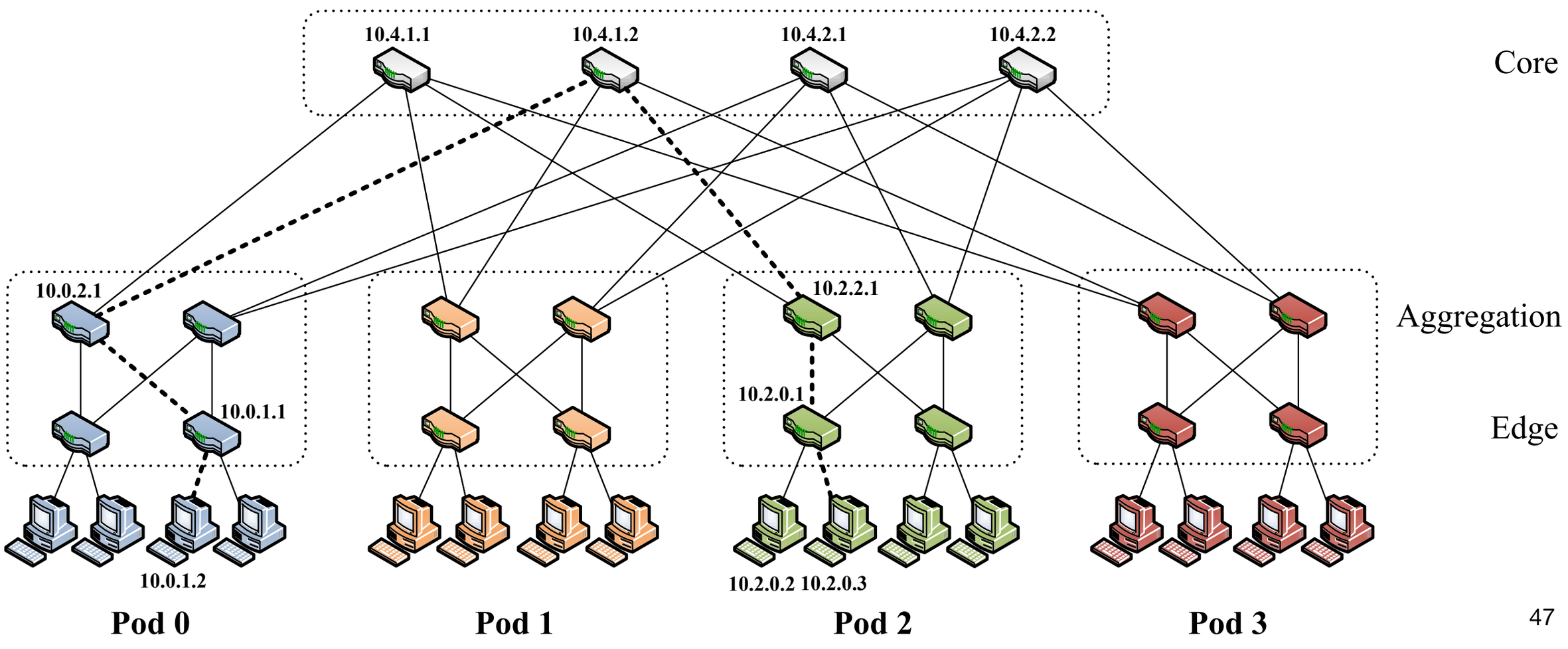
- Clos topology, built upon commodity switches




M. Al-Fares et al. A scalable, commodity data center network architecture. In Proc. of ACM SIGCOMM, 2008.

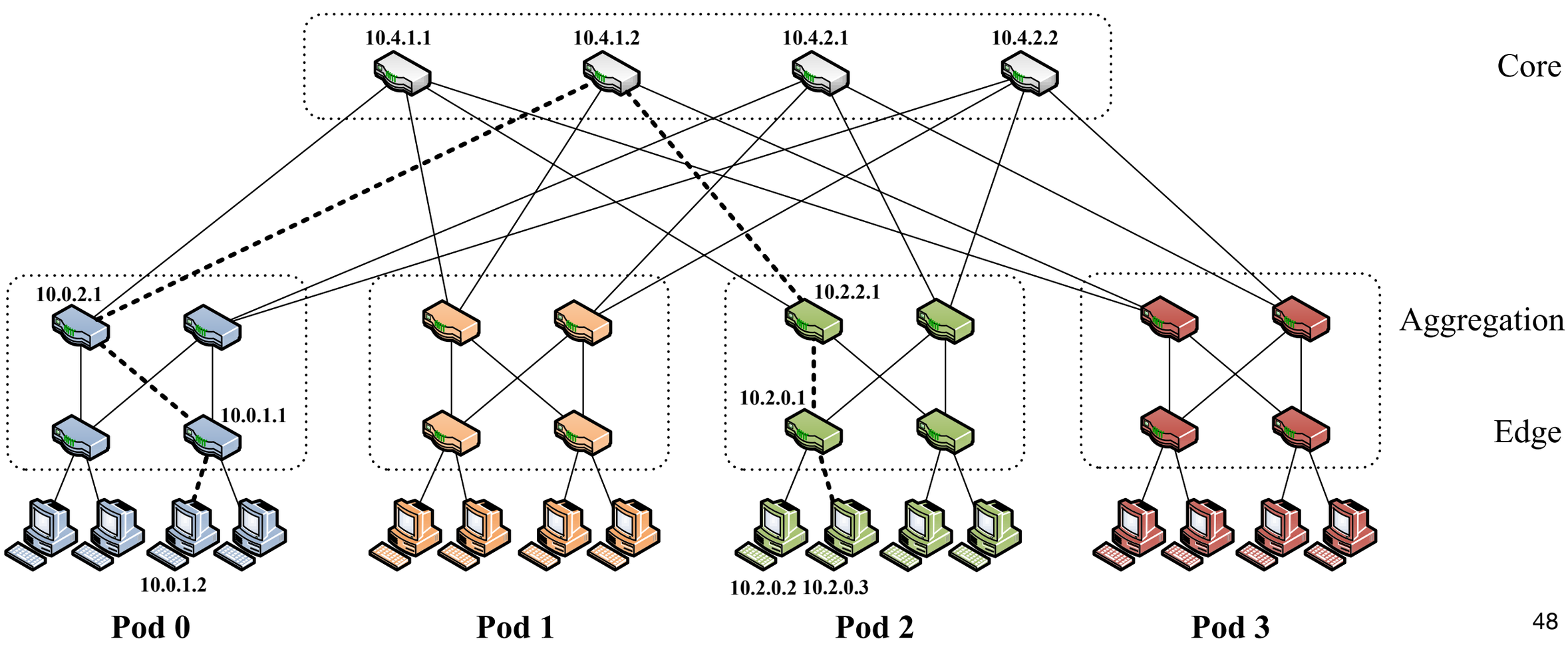
# Fat-tree

- k-pod fat-tree, k=4. There are k pods, each with two layers of k/2 switches.
- Each aggr. switch has k/2 ports to unique core switches; k/2 aggr. switches in a pod. So total number of core switches is $(k/2)^2$.



# Fat-tree

- Each core switch has one port to each pod.
- Each edge switch has k/2 hosts. k/2 edge switches in each of k pods. So a k-pod fat-tree can support $k^3/4$ hosts.



# Advantages of fat-tree

- In traditional hierarchical networks, switches in aggregation and core layers need to be more powerful, and have more ports per device. High-end high port density switches are extremely expensive.

- Scale out vs. scale up

- Fat-tree: $(5k^2/4)$ k-port switches support $k^3/4$ hosts

- 48-port 1GigE switches: 27,648 hosts using 2,880 switches.

# Advantages of fat-tree

- Rearrangeably non-blocking: for arbitrary communication patterns, there is some set of paths that will saturate all the bandwidth available to the end hosts in the topology.

- Hierarchical topologies are not rearrangeably non-blocking.