
NANYANG
TECHNOLOGICAL
UNIVERSITY
SINGAPORE
CE/CZ4052
Cloud Computing
Data Center Networking
– Basics, Topology
Dr. Tan, Chee Wei
Email: cheewei.tan@ntu.edu.sg
Office: N4-02c-104

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

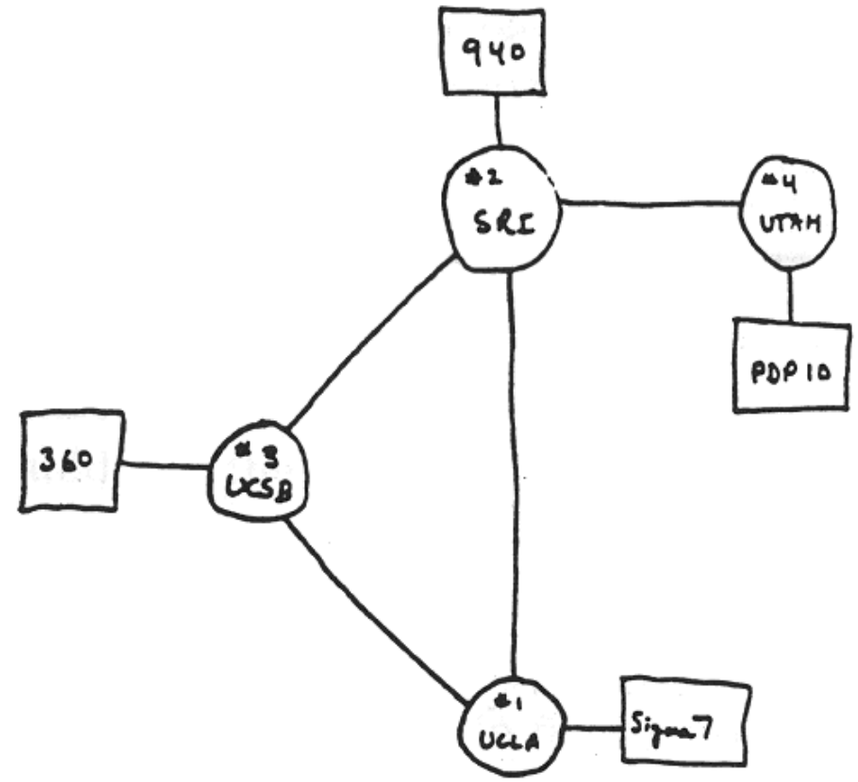
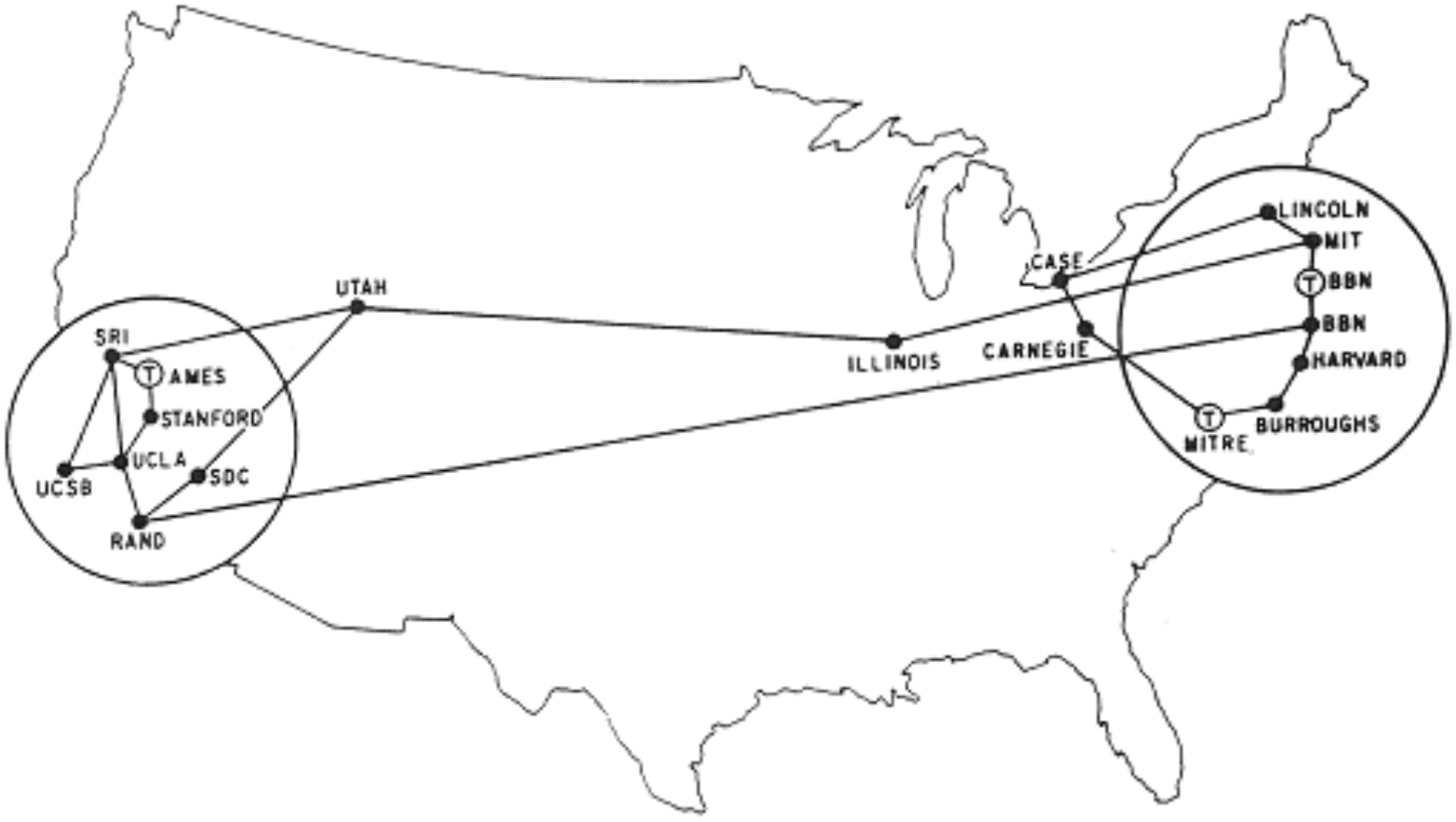


FIGURE 6.2 Drawing of 4 Node Network (Courtesy of Alex McKenzie)

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

Evolution – 1971

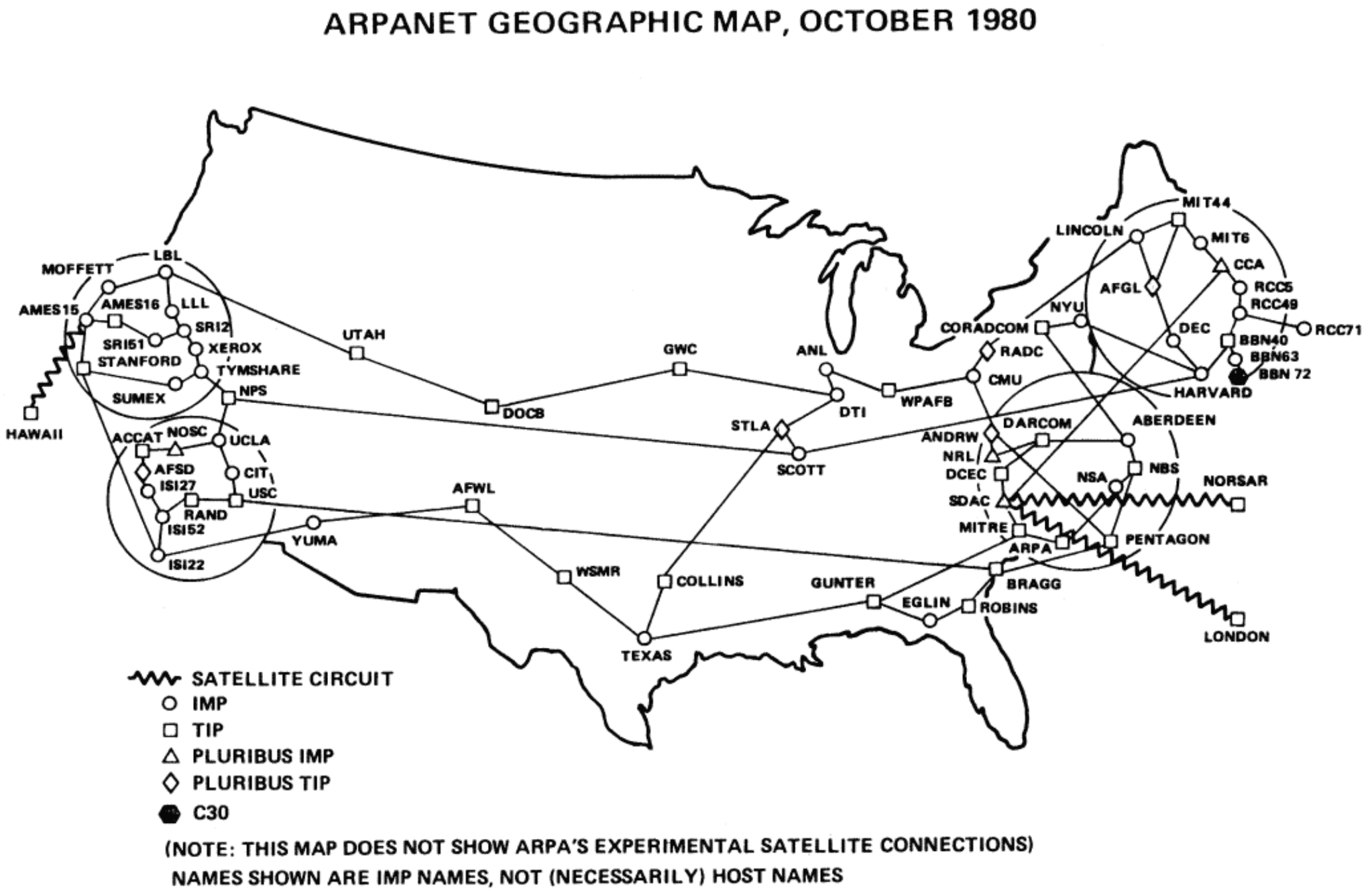
- September 1971



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

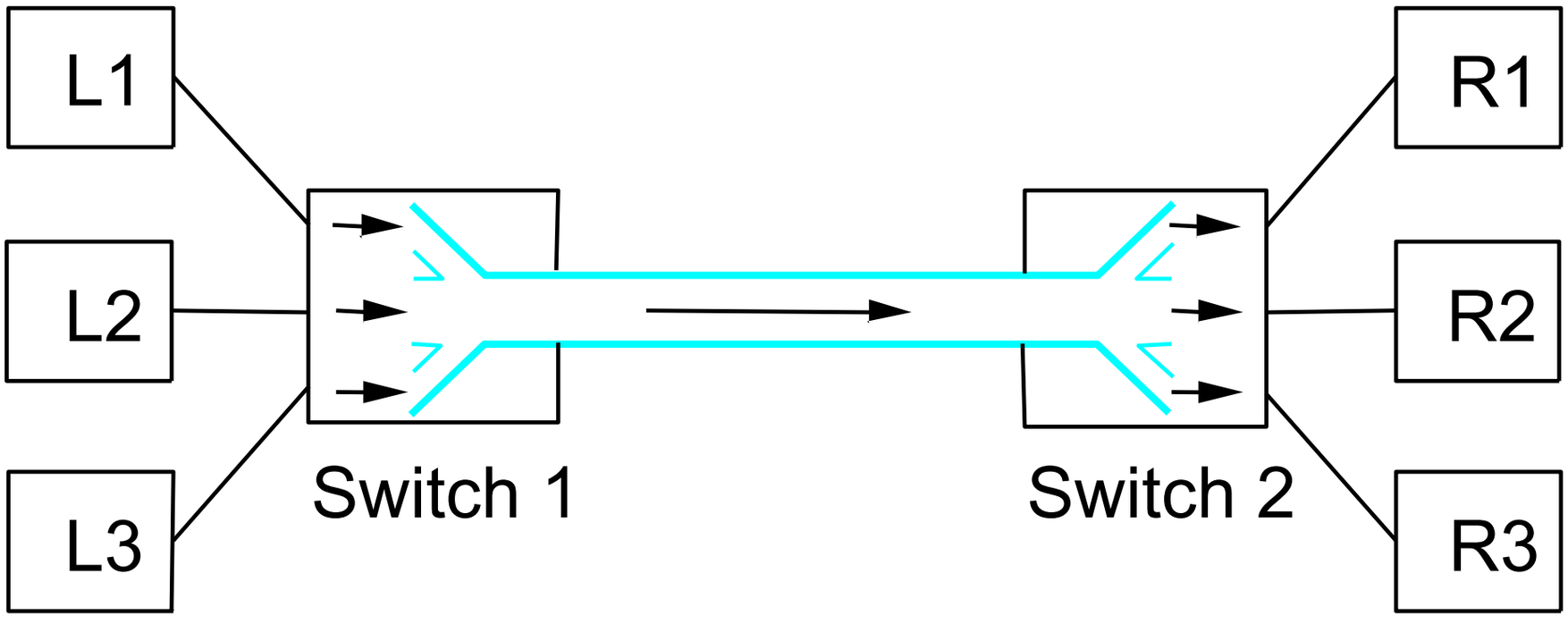
- Circuit switching
  - (early) telephone networks
  - Connection oriented, dedicated comm. link between two nodes
- Packet switching
  - computer networks
  - Connection-less, shared comm. links, no set-up
  - Data is divided into packets and transferred independently

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) Assumes worst case

- Packet switching, how many?
  - About 30 with very low probability of queueing delay

Binomial

Statistical multiplexing gain: not everybody is transmitting!

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?

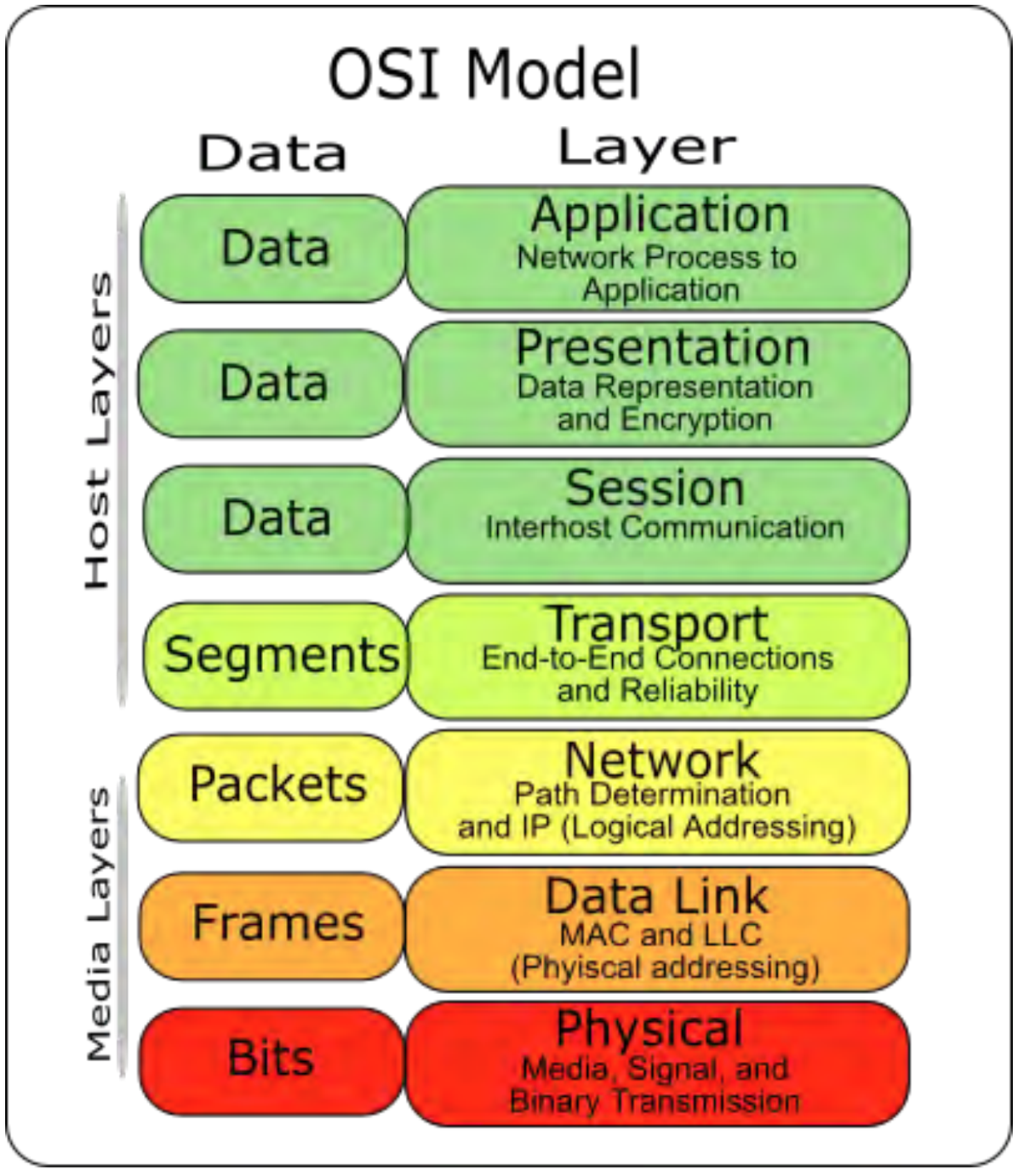
- Answer: layering, with well-defined interfaces

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

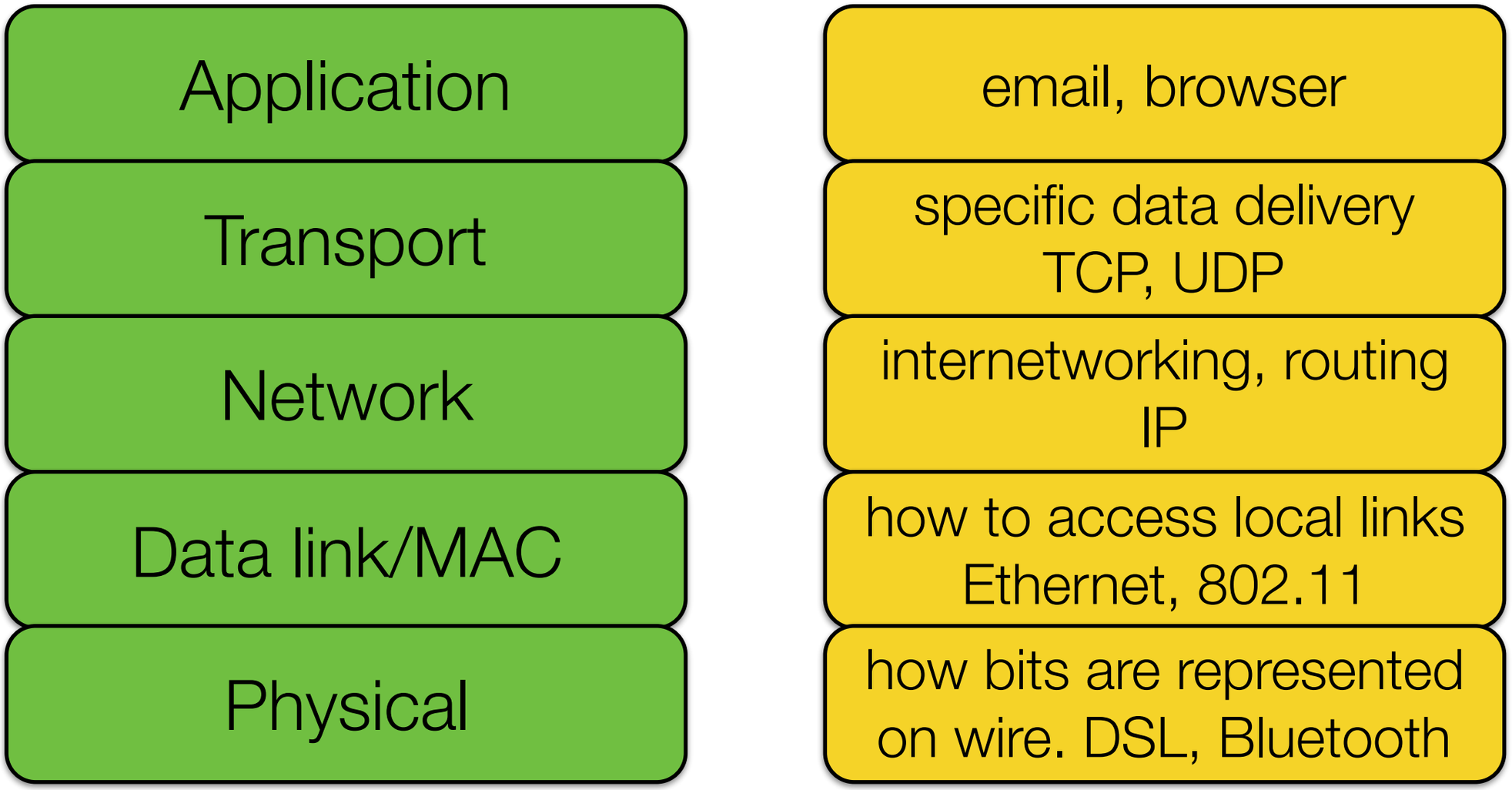
Why layering?

- 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

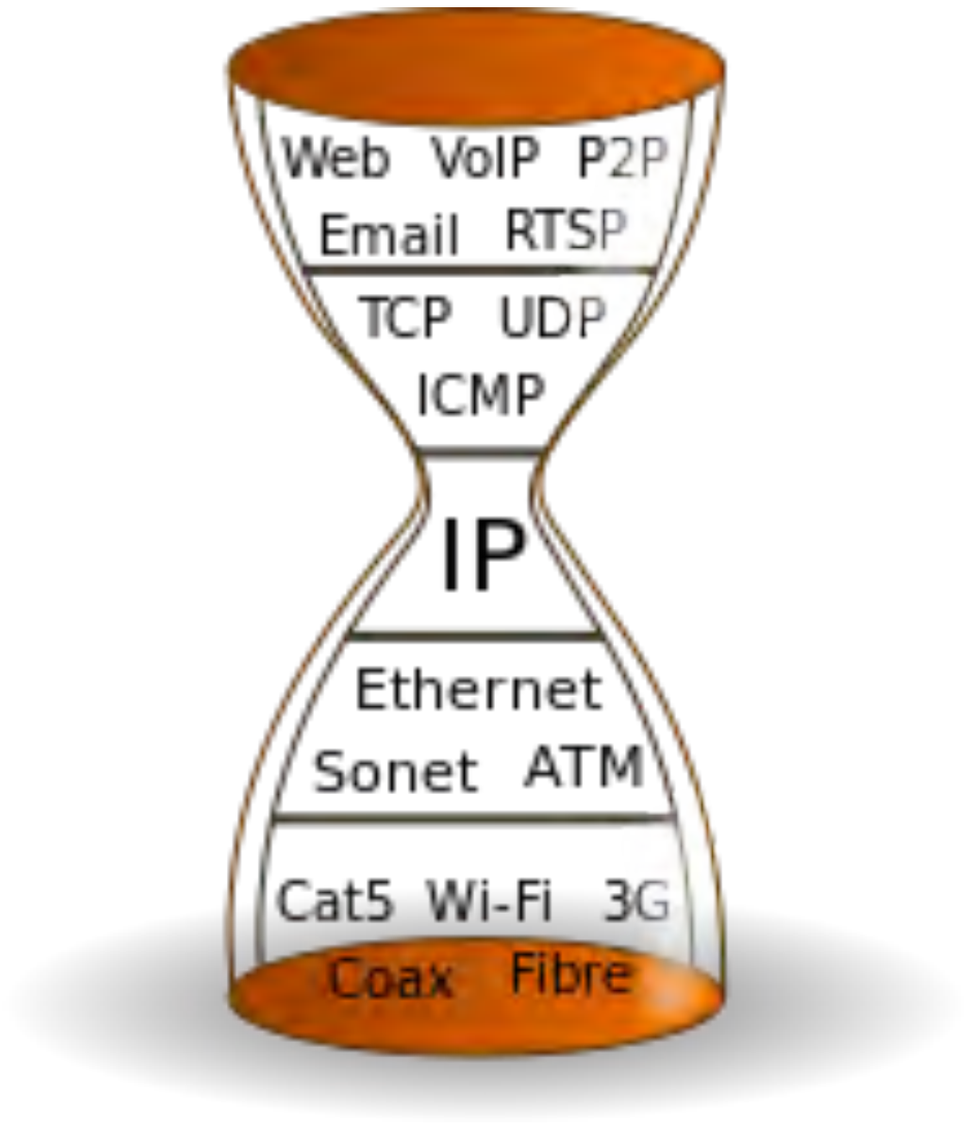


More popular TCP/IP model



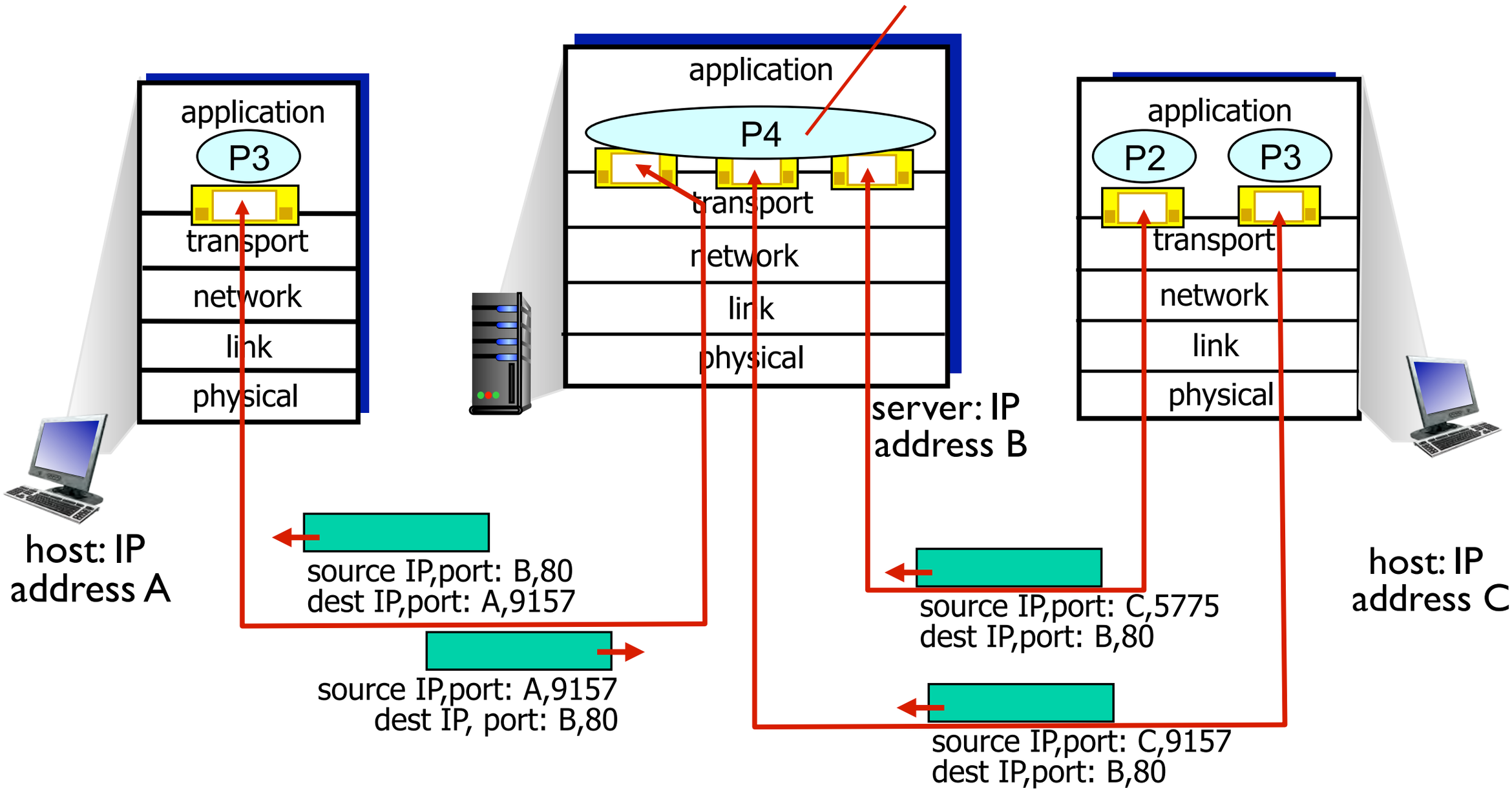
Hourglass

- Our protocol stack



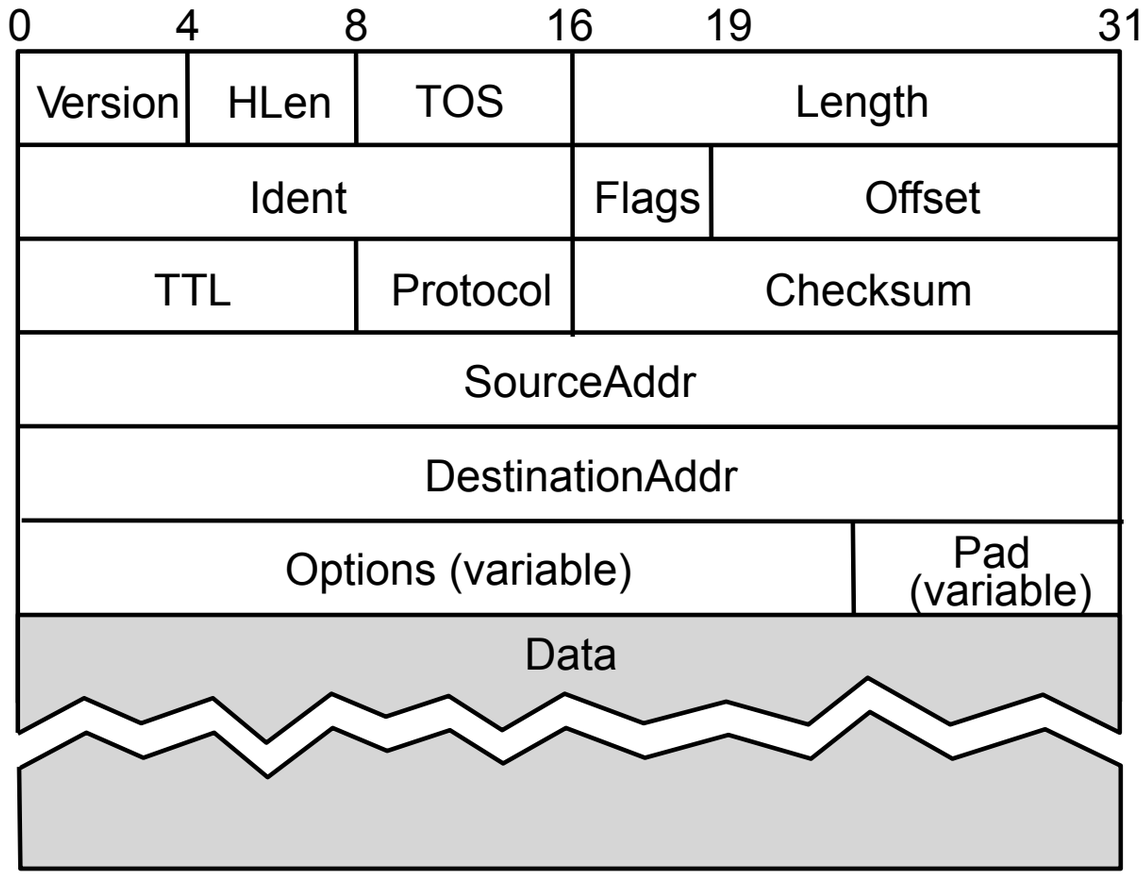
TCP/IP

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



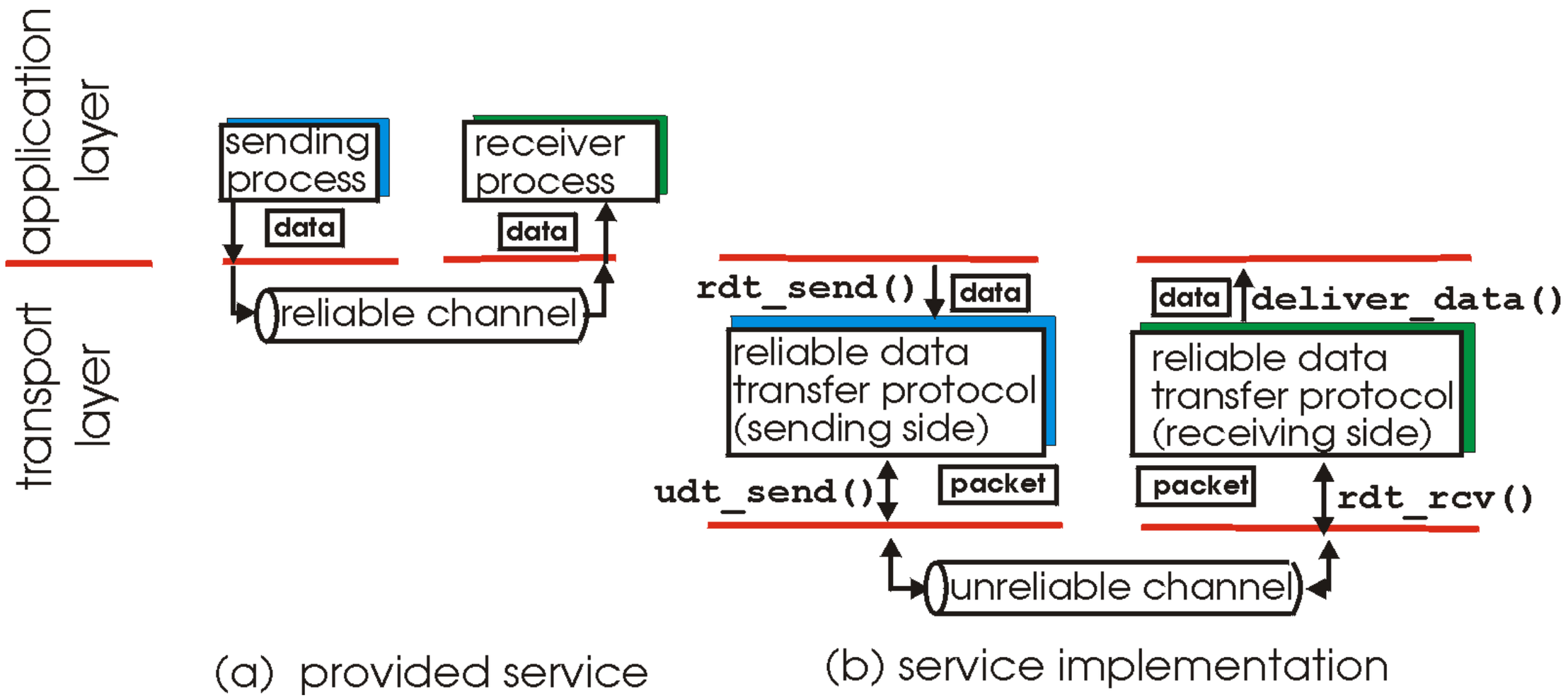
Routing

- 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 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

Regulate sending of bytes

- Congestion control: avoid congestion collapse (too many sources sending too much). congestion window, cwnd

Reliable transfer

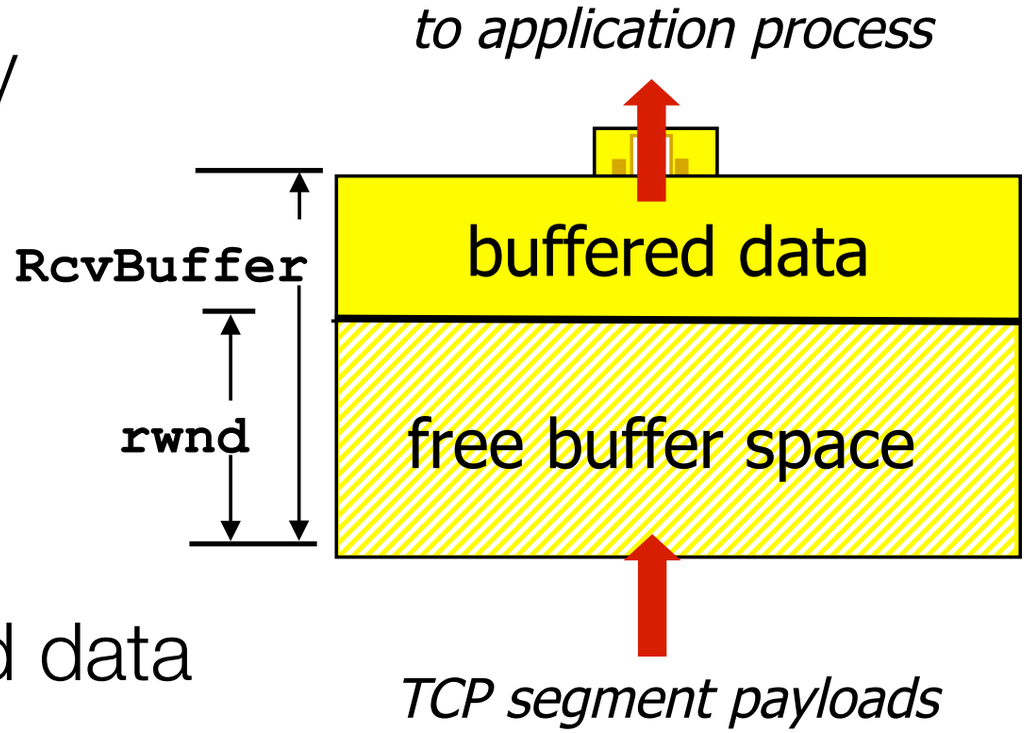
- Use sequence number, and send acknowledgement packets indicating the sequence number up to which the receiver has received

- 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 Additive increase/multiplicative decrease

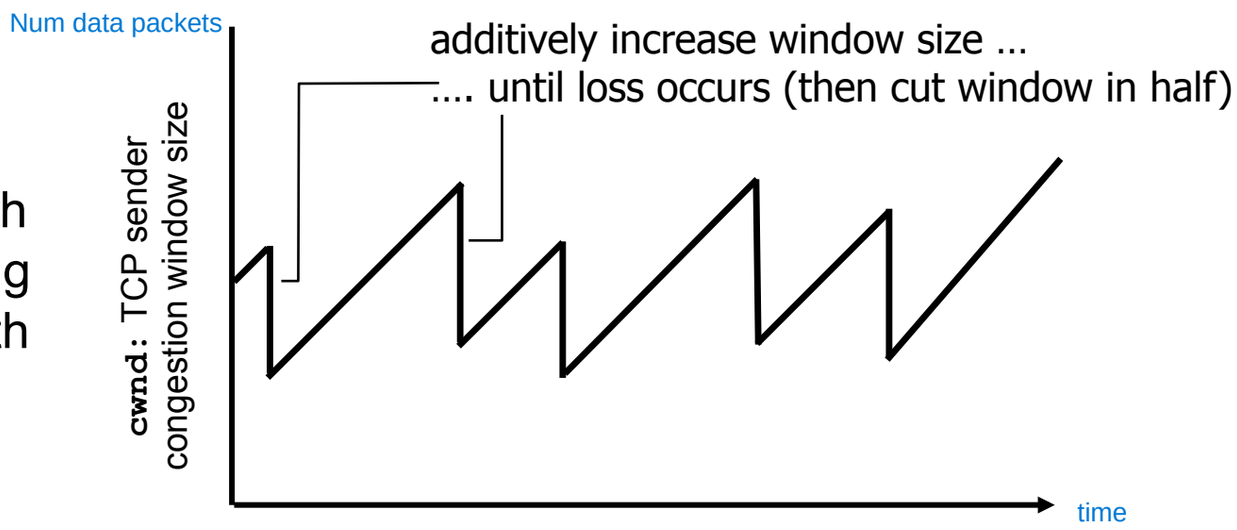
- Sender increases sending rate (window size), probing for unused bandwidth, until loss occurs

- additive increase: increase cwnd by 1 MSS (maximum segment size) every RTT until loss detected

  Round-trip time

- multiplicative decrease: cut cwnd by half when loss is detected

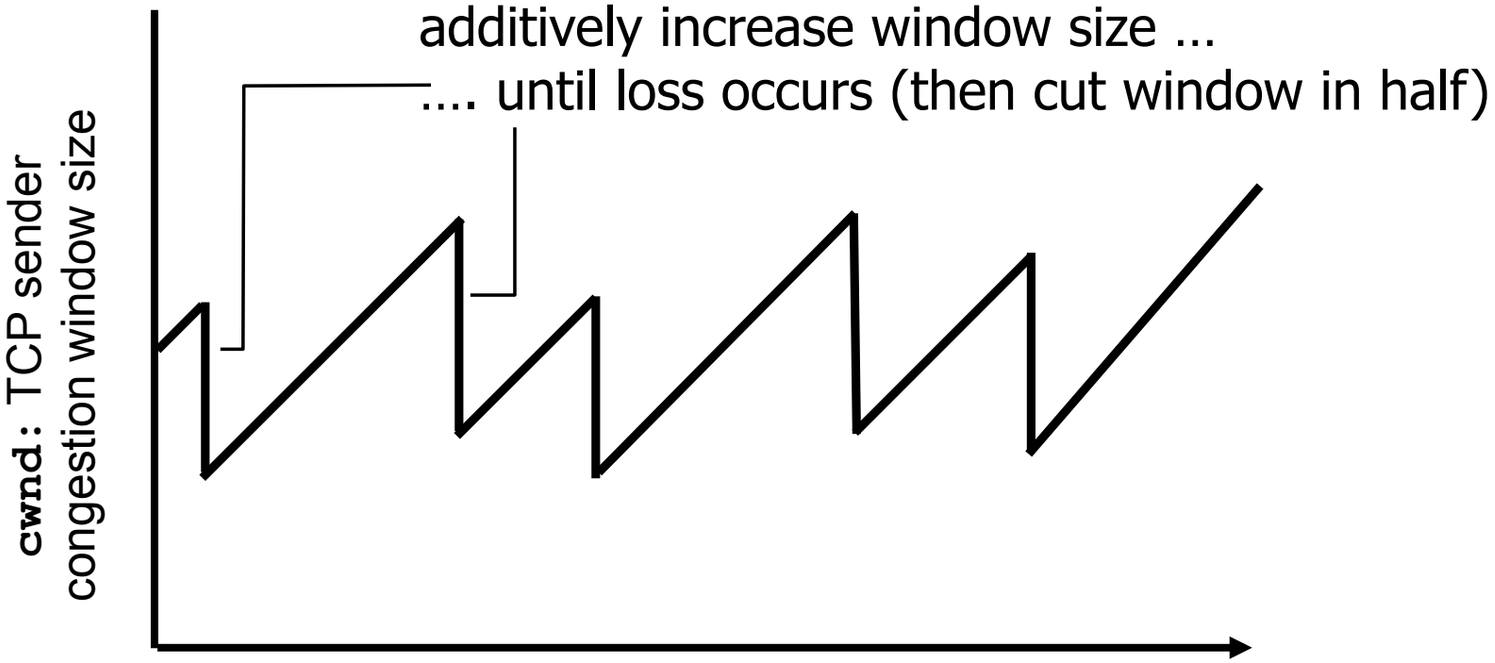
AIMD saw tooth behavior: probing for bandwidth



TCP congestion control

- Typical TCP cwnd behavior

AIMD saw tooth behavior: probing for bandwidth

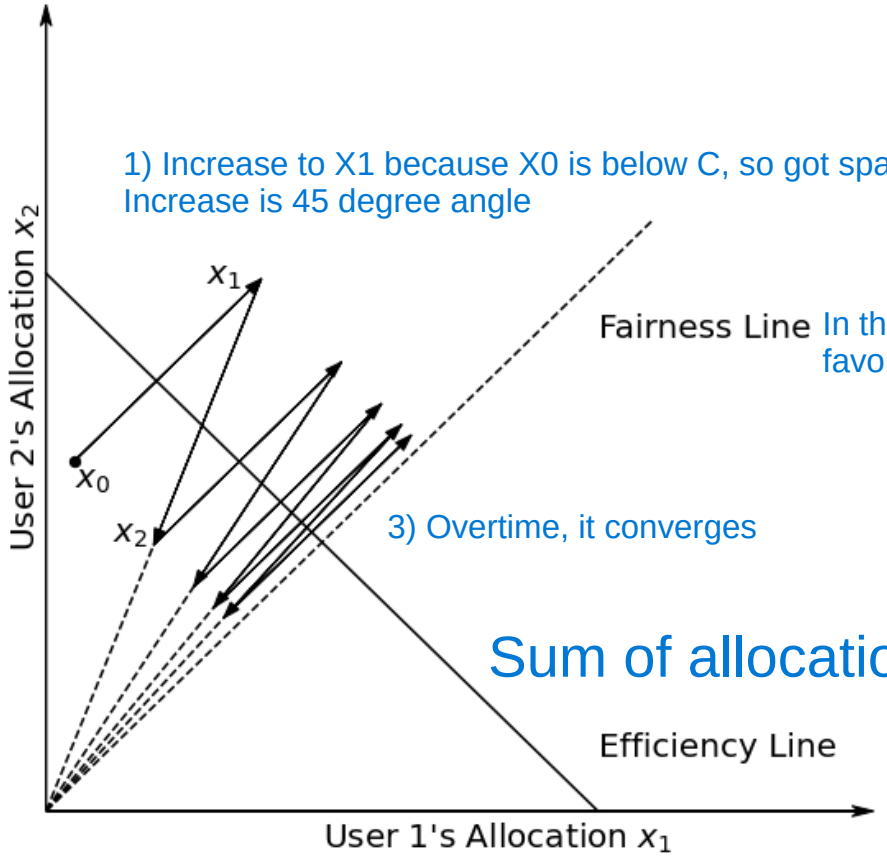


Classical Visual Proof of AIMD

Objective is to guide allocation to Fairness line

1) Increase to X1 because X0 is below C, so got spare capacity. Add 1 to both users. Increase is 45 degree angle

2) x1 to x2 -> cut by half for both as exceeded C. So X2 is the halfway point of the vector (origin to the X1 on diagram)



In this case, both x1 and 2 are equally favored, hence 45degrees

3) Overtime, it converges

Sum of allocation = C (capacity)

A visual proof of the AIMD abstraction

Perron-Frobenius Theory Approach to AIMD

Each user has their own
a = AI
b = MD

- Perspective via positive linear systems theory [12] [13]
- Let $w_s(k)$ denote the congestion window size of source s immediately before the kth network congestion event is detected by all the sources as shown in Figure 2.
- Let α_s and $0 < \beta_s < 1$ be the additive and multiplicative parameters of source s using the AIMD algorithm (that are conventionally set as 1 and 0.5) respectively
- Let $q_{\max}$ and P be, respectively, the maximum queue length of the congested bottleneck link and the maximum instantaneous number of sent unacknowledged packets that are in transit (e.g., $P = q_{\max} + BT$ where B is the bottleneck link service rate in packets per second and T is the round-trip time)

[12] Abraham Berman, Robert Shorten, and Douglas Leith. Positive matrices associated with synchronised communication networks. Linear Algebra and its Applications, 393:47–54, 2004.

[13] Martin Corless, Christopher King, Robert Shorten, and Fabian Wirth. AIMD Dynamics and Distributed Resource Allocation. Society for Industrial and Applied Mathematics, 2016.

Positive Linear System

Tells me the sending window size right before the k congestion event

- At the $(k+1)$th congestion event, source s's window satisfies

$$w_s(k+1) = \beta_s w_s(k) + \left(\frac{\alpha_s}{\sum_{i=1}^{n} \alpha_i} \right) \sum_{i=1}^{n} (1 - \beta_i)\, w_i(k).$$

Let $w(k) = (w_1(k), \ldots, w_n(k))^T$ and write a positive system:

$$w(k+1) = Aw(k),$$

At convergence, w(k+1) = Aw(k)
(1) w(k+1) = Aw(k)
so there exist a w(k) that is an eigen vector that equates to the direction/slope of the fairness line

where

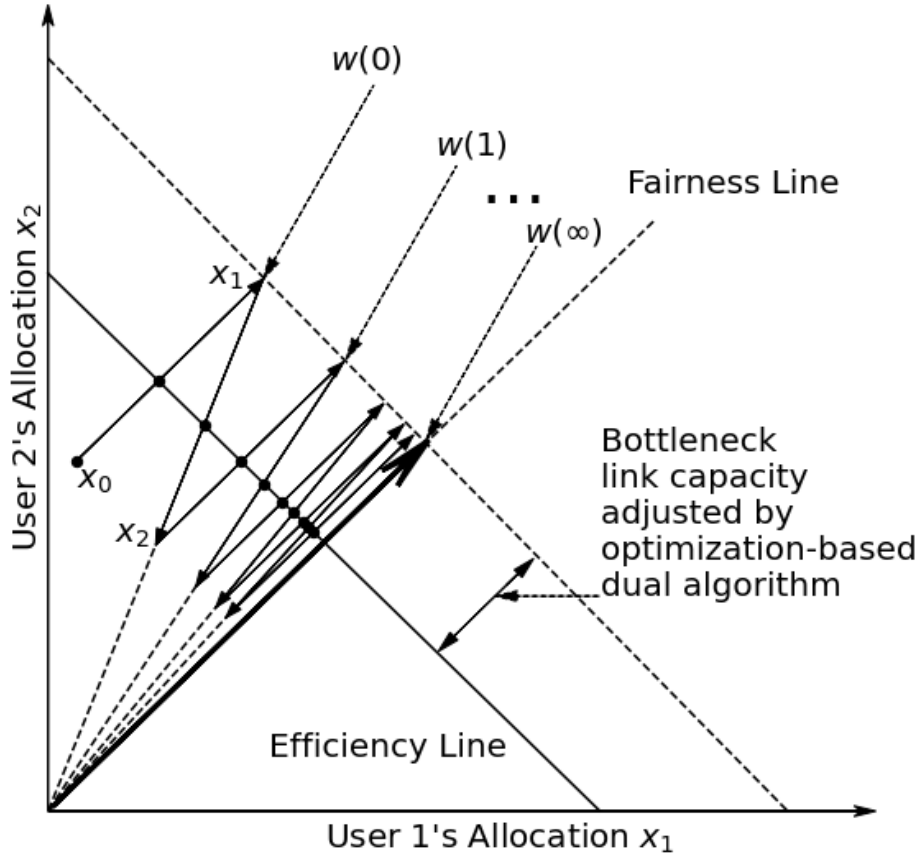
$$A = \begin{bmatrix} \beta_1 & 0 & \cdots & 0 \\ 0 & \beta_2 & 0 & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \cdots & \beta_n \end{bmatrix}$$

Sqaure Matrix!

Converted to linear algebra form

$$+ \frac{1}{\sum_{i=1}^{n} \alpha_i} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_n \end{pmatrix} (1 - \beta_1, \cdots, 1 - \beta_n)$$

Visualizing the Theories Behind AIMD



Teaching convex optimization theory and Perron-Frobenius Theory.

Positive Linear System

- The spectrum of the matrix **A** (e.g., the Perron-Frobenius eigenvalue and eigenvectors) provides insights on fairness, rate of convergence and transient response: [14]

$$\lim_{k\to\infty} w(k) = \left(\frac{\alpha_1}{1-\beta_1}, \cdots, \frac{\alpha_n}{1-\beta_n}\right)^T,$$

  which, if specialized to the case of $\alpha_i = 1$ and $\beta_i = 0.5$ for all i, is proportional to the all-ones vector as it should be
- Fairness line as Perron-Frobenius right eigenvector
- Classical power method algorithm can simulate AIMD and to visualize the iterates as shown in Figure 2

[14] Abraham Berman, Robert Shorten, and Douglas Leith. Positive matrices associated with synchronised communication networks. Linear Algebra and its Applications, 393:47–54, 2004.

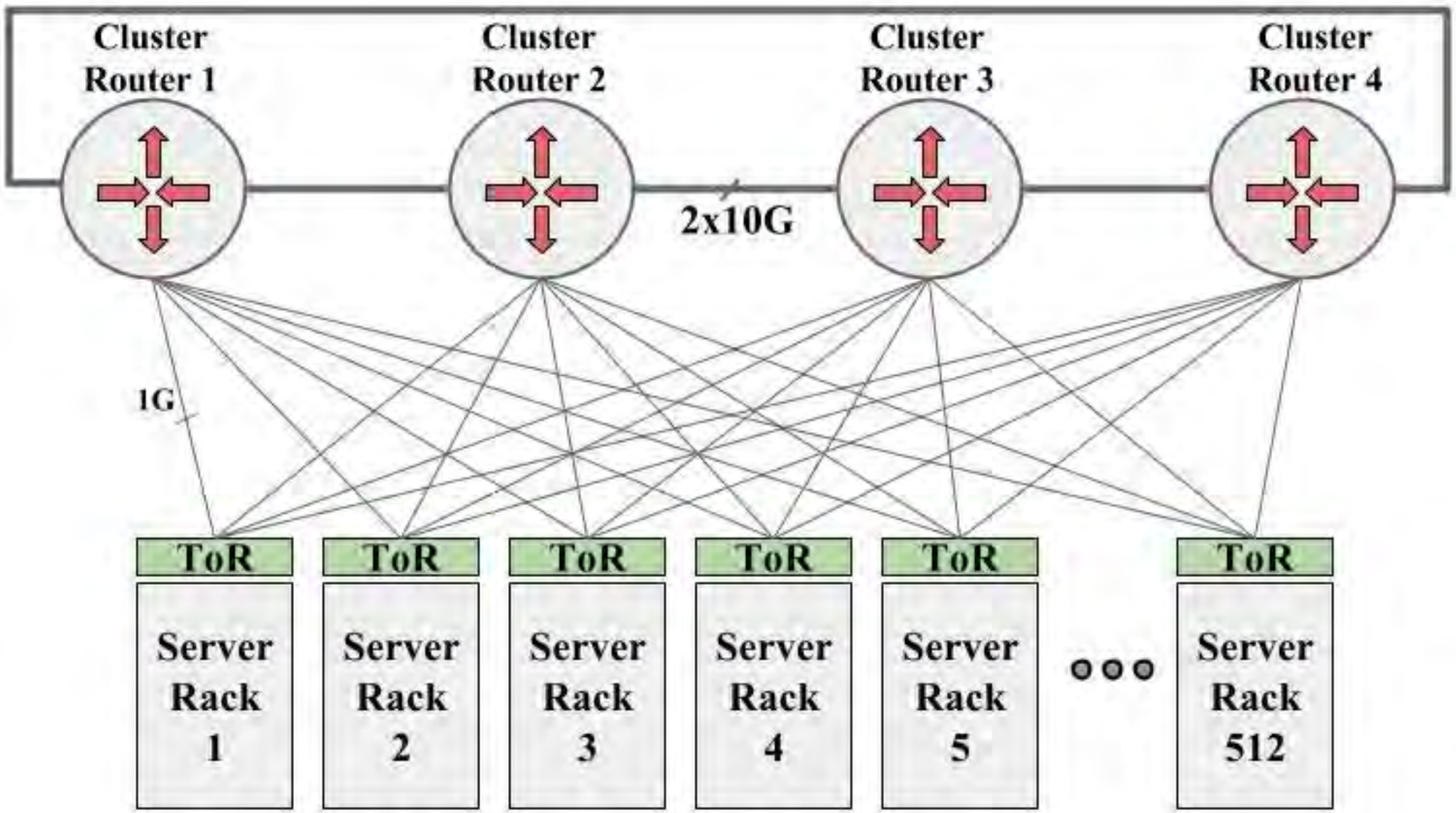
Topology

or an evolution of Google's network topologies

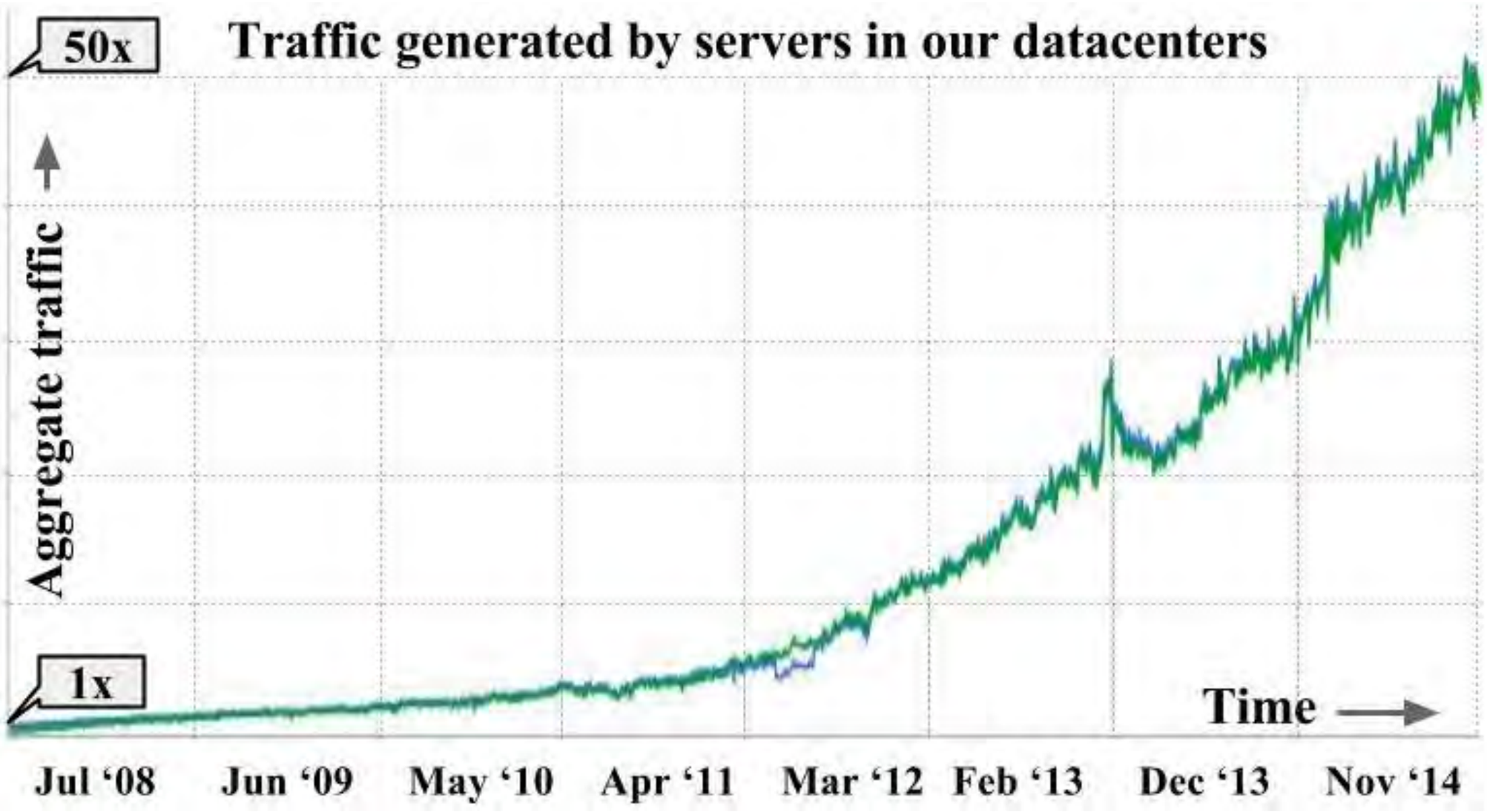
Credit: A. Singh et al., "Jupiter rising: A decade of Clos topologies and centralized control in Google's datacenter network," ACM SIGCOMM'15.

2004: four-post cluster

- Supported 20k servers per cluster

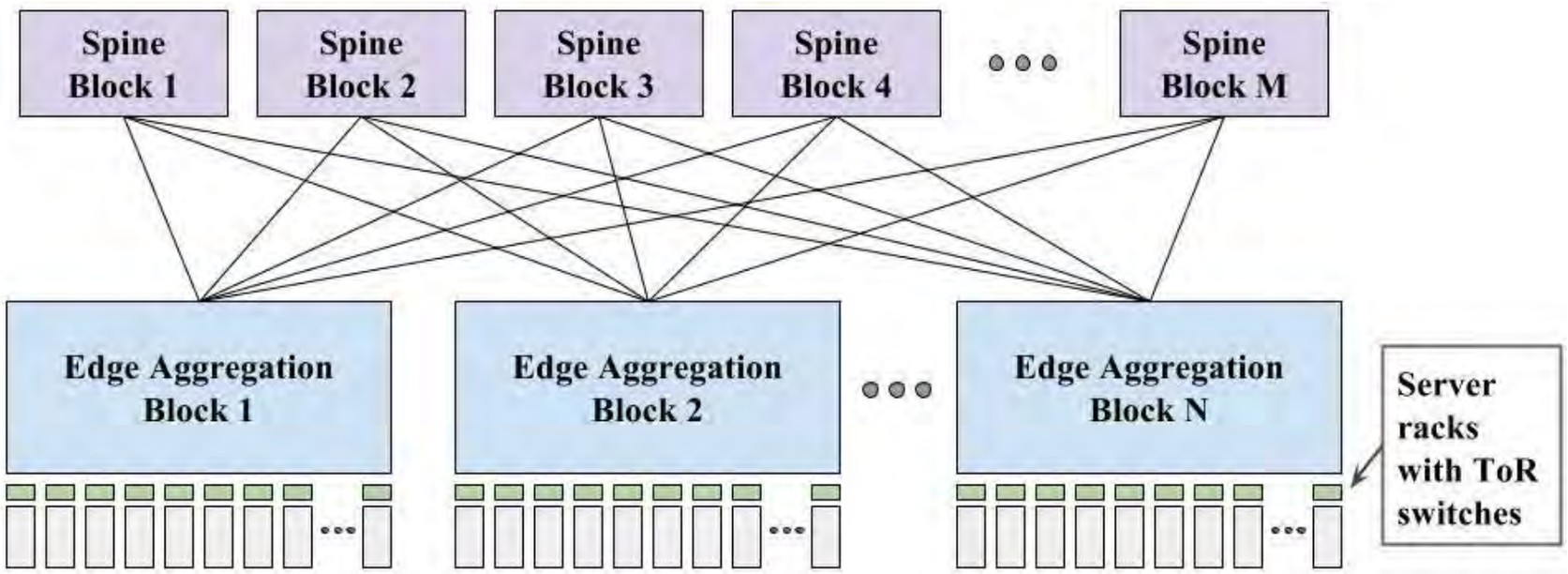


But traffic keeps growing



How to scale the network

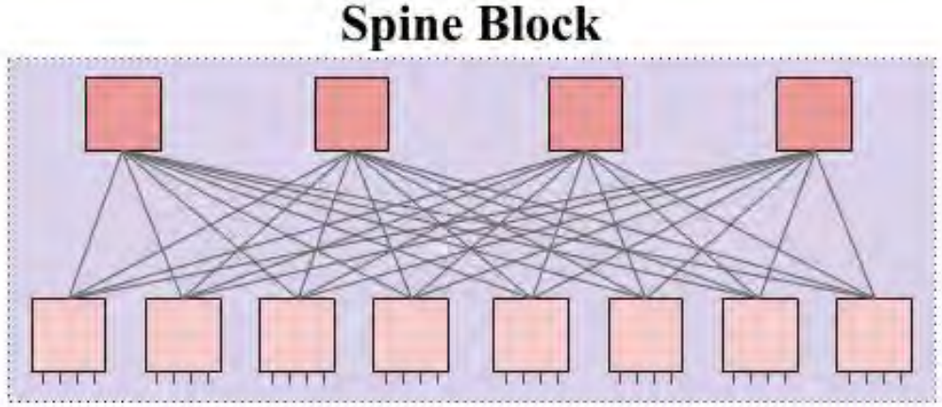
- Buying the largest switches with the most ports doesn't scale well
- And it's expensive
- Solution: Clos topologies

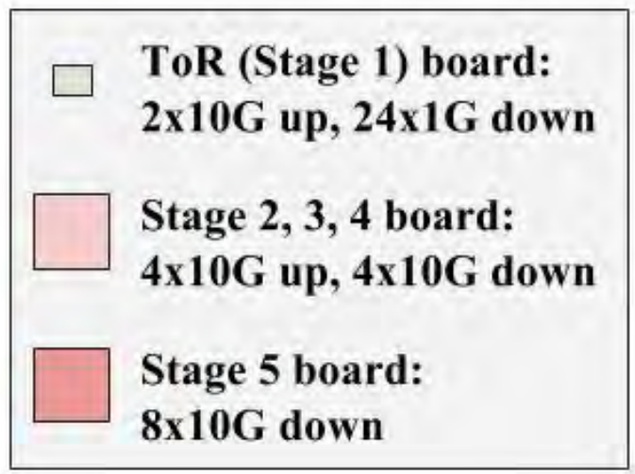


A practical approach to scale

- 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

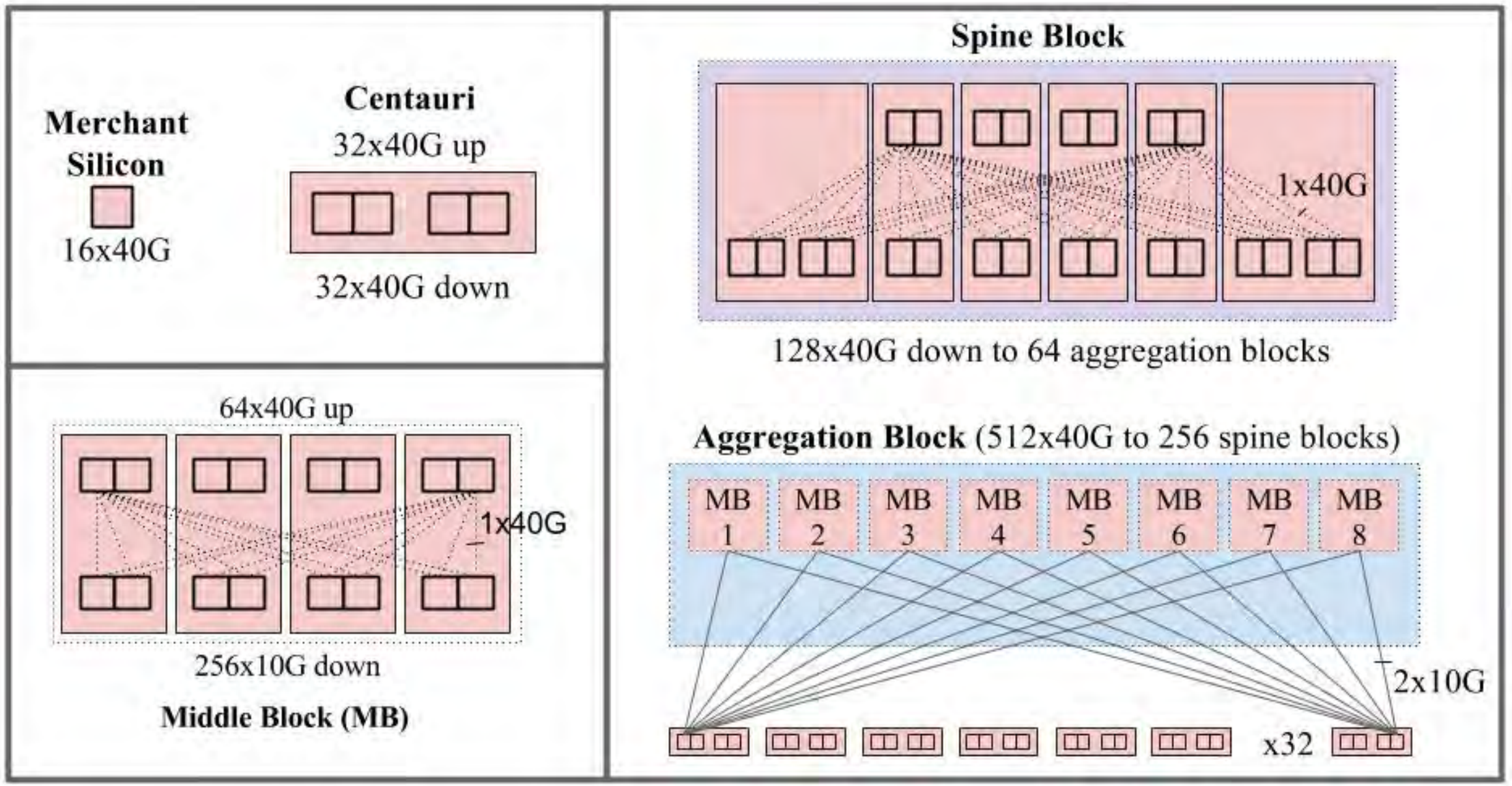
[2005] Firehose 1.0



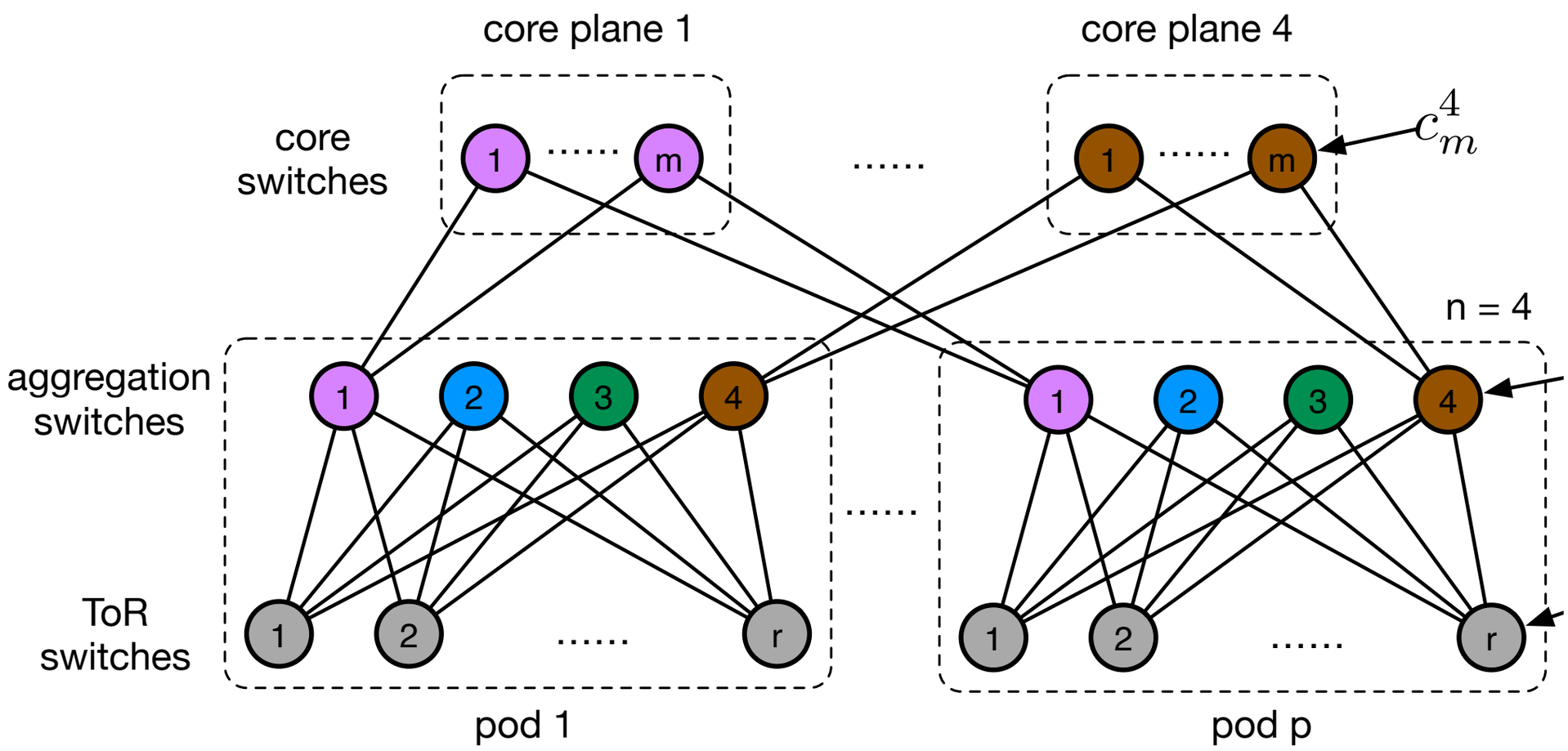


- An aggr block has 16 ToRs (320 machines). 32 spine blocks each connect to 32 aggr blocks, resulting in a fabric that scales to 10K machines with 1G average bandwidth

[2012] Jupiter



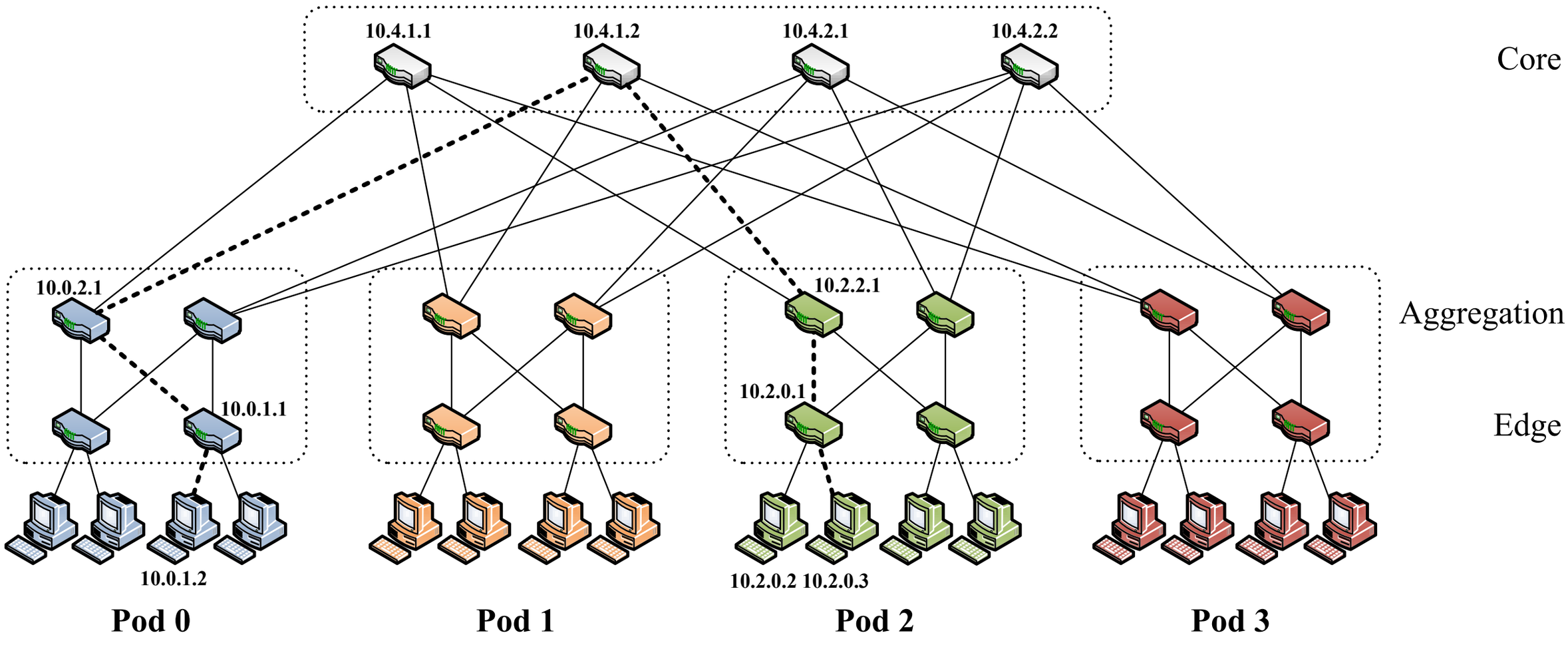
Facebook's fabric



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

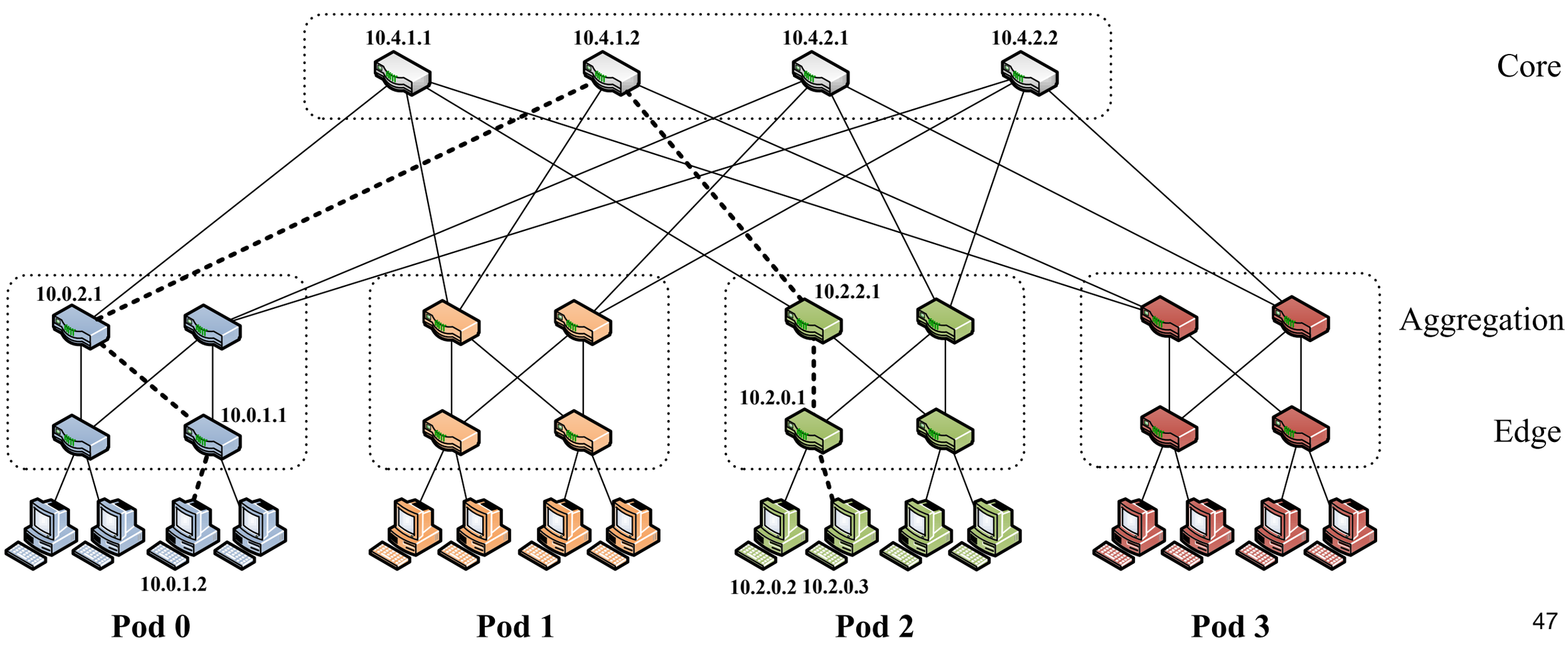
- Clos topology, built upon commodity switches



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

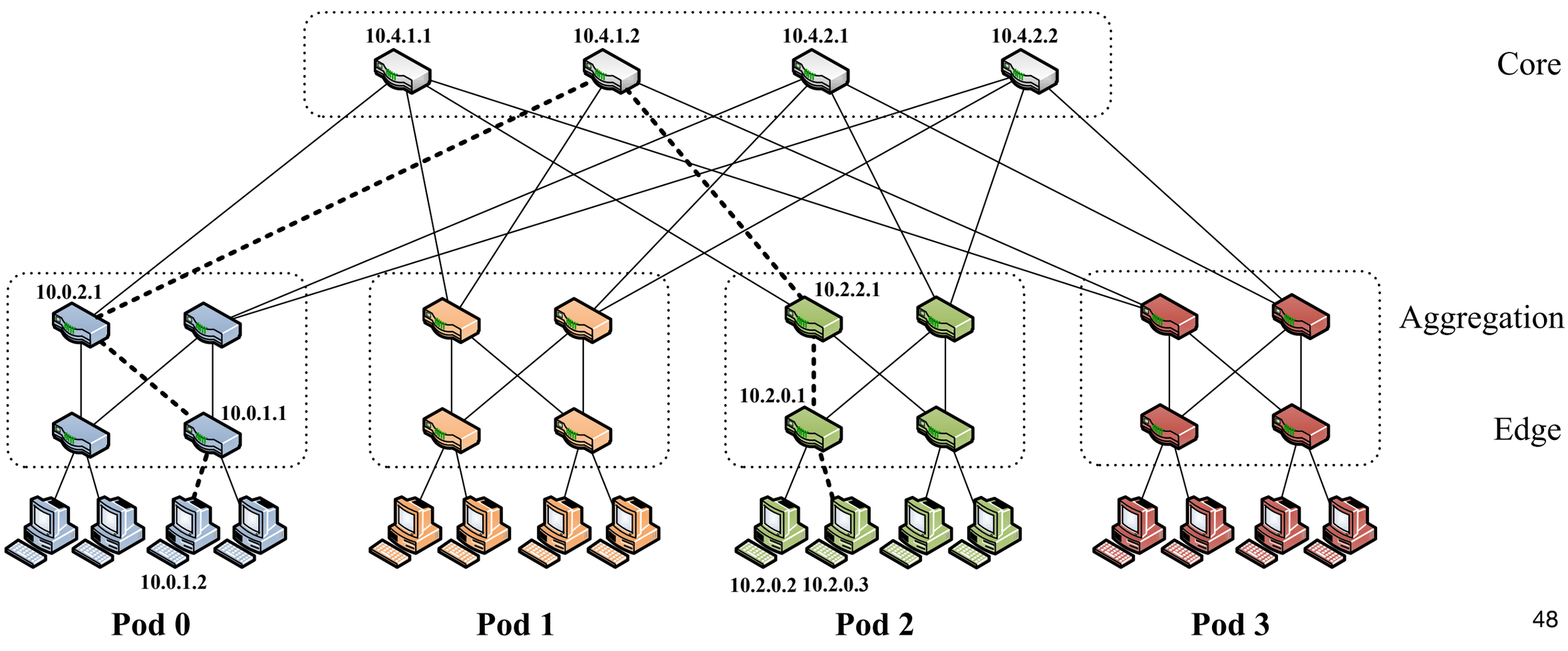
Fat-tree

- 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.

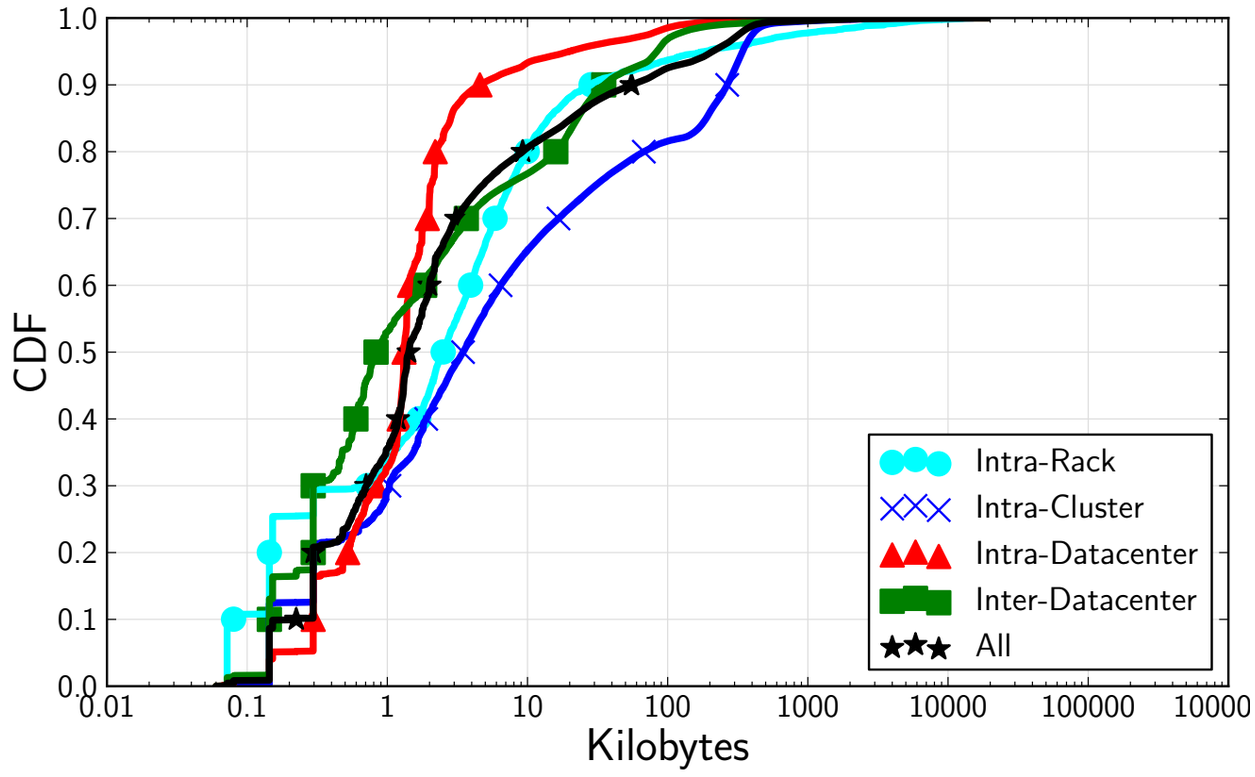
Advantages of fat-tree

- 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.

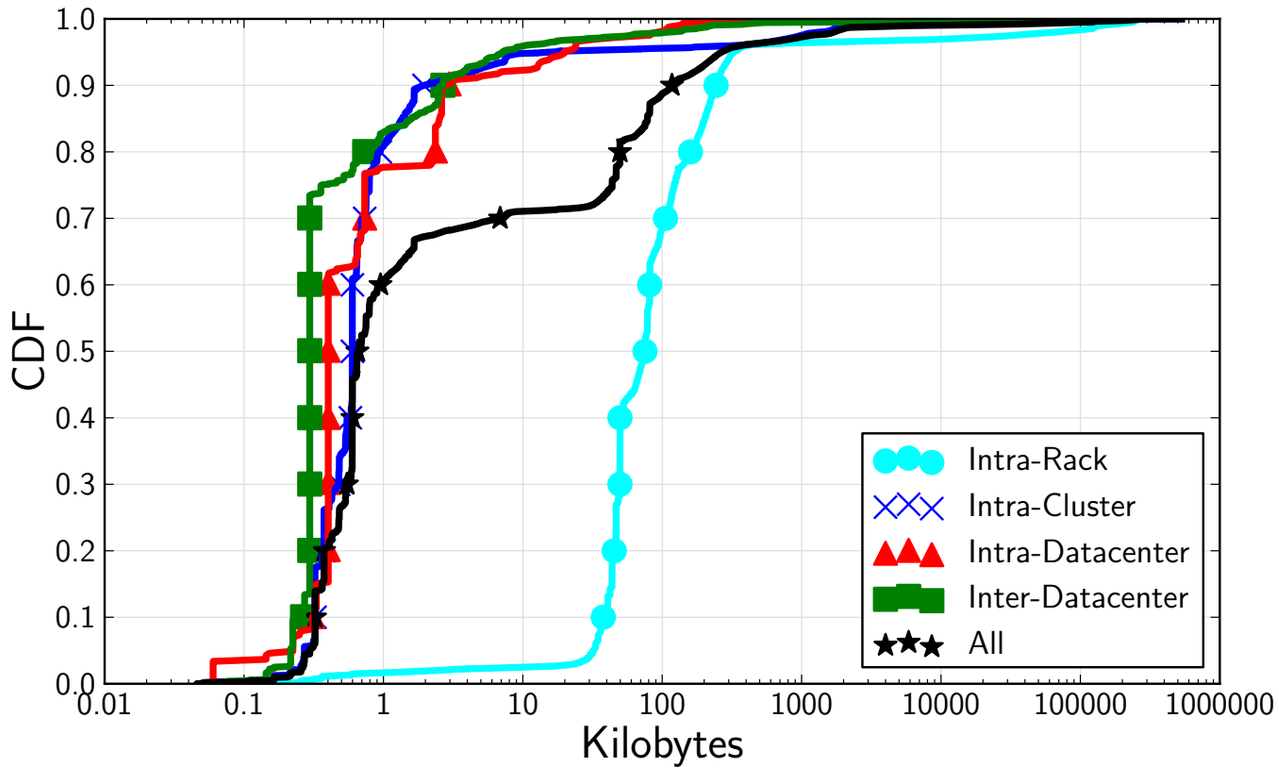
- Hierarchical topologies are not rearrangeably non-blocking.

Traffic characteristics

Flow size, Facebook



(a) Web servers



(c) Hadoop

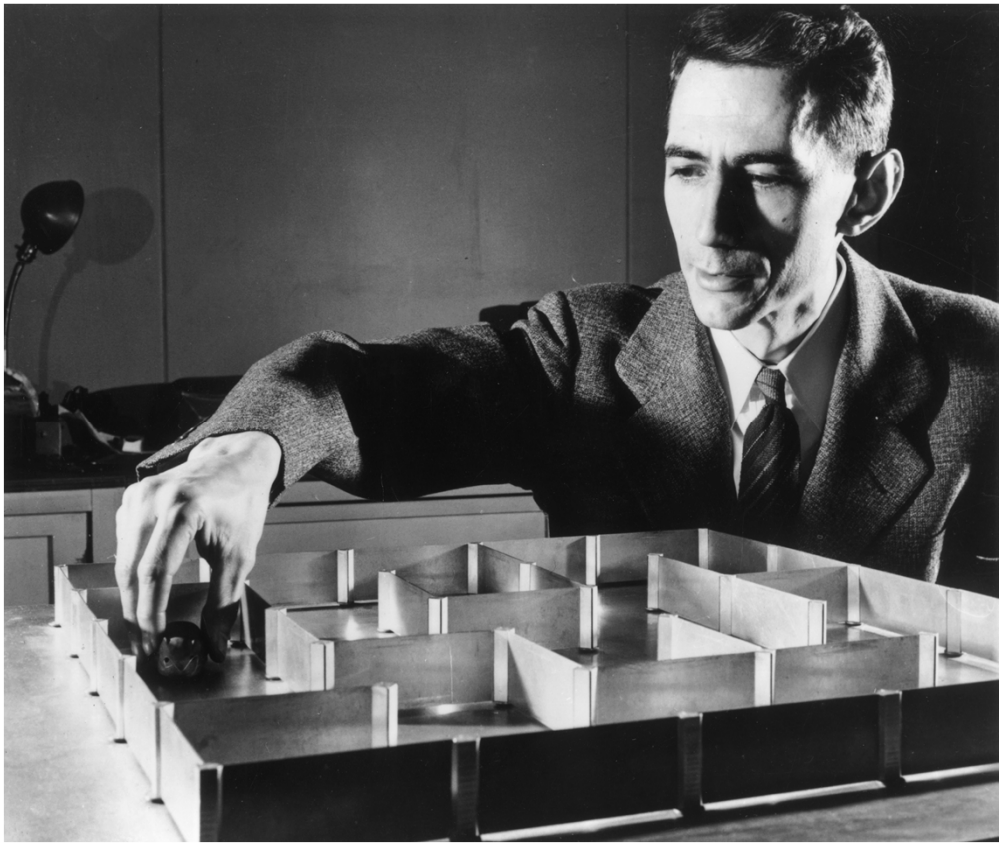
- Mice vs. elephants: many mice flows, a few elephants carrying most of the bytes -> Routing elephants is important

A. Roy et al. Inside the social network's (datacenter) network. In Proc. of ACM SIGCOMM, 2015.

Theseus: Shannon's Mouse-in-Maze

Theseus (1952) AT&T Bell Labs

Video demo by Shannon: https://www.youtube.com/watch?v=nS0luYZd4fs



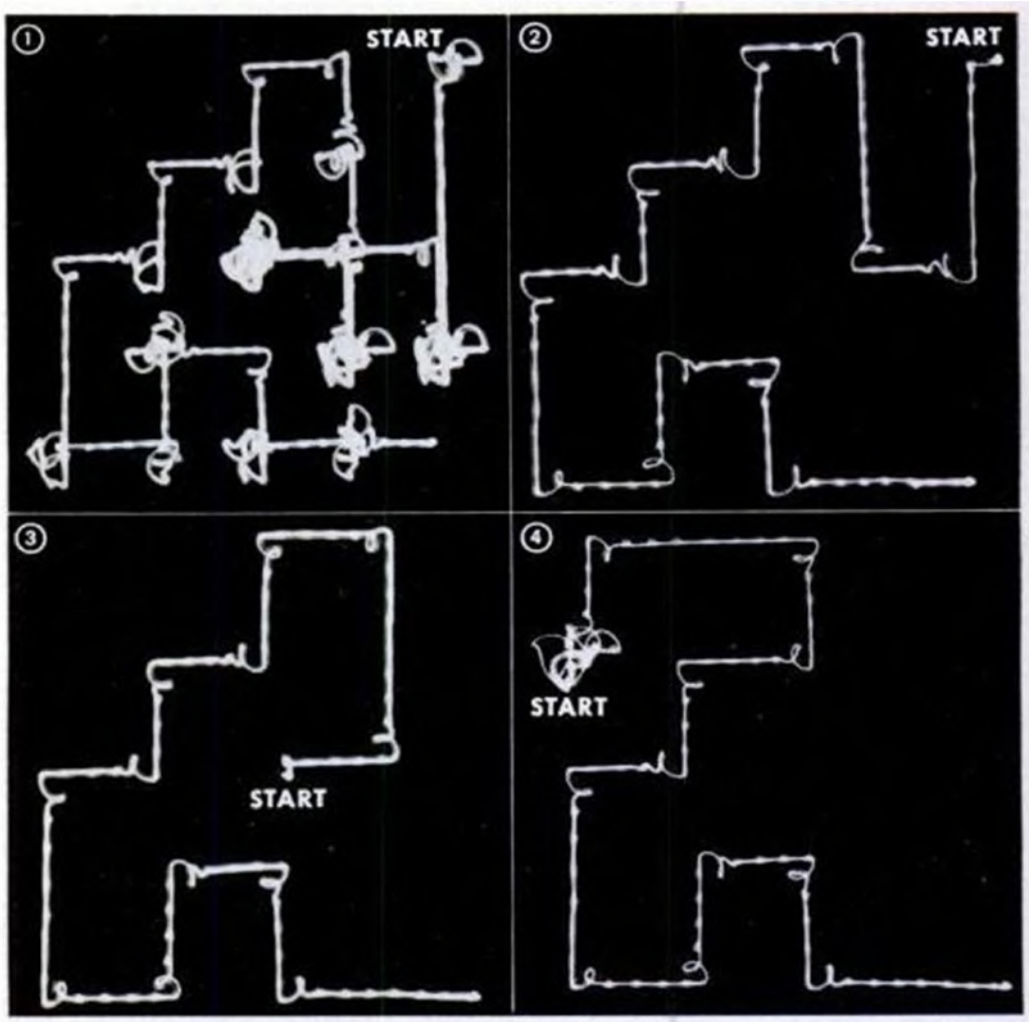
https://time.com/4311107/claude-shannon-100-years/

Theseus: Shannon's Mouse-in-Maze

A life-sized mouse robot in 1952… The Maze *solves* the Mouse!

http://cyberneticzoo.com/mazesolvers/1952-%E2%80%93-theseus-maze-solving-mouse-%E2%80%93-claude-shannon-american/

Theseus: Shannon's Mouse-in-Maze

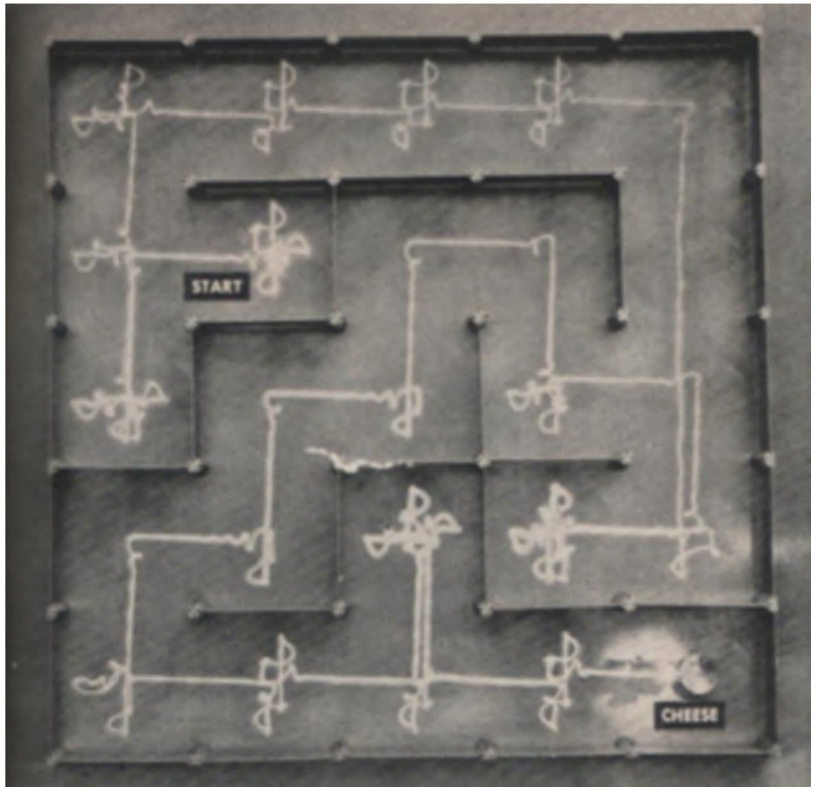


MEMORY TESTS show how Theseus learns. In first trial the mouse makes wrong turns, leaves complicated trail. Second time he starts from the same place, goes straight to the goal. In third trial he is started from different spot but is on the original trail, so has no trouble. The fourth time he is put in an unfamiliar square, blunders around until he gets on the course he has learned.

- Theseus learns by experience and trial-and-error
- Memory of its route such that, when placed in a new spot that was on the previous route, Theseus can ignore blind alleys (i.e., previous errors made) and navigate correctly to end point.
- When maze topology changes, Theseus *forgets outdated knowledge, relearns and incorporates new knowledge* to existing ones in memory
- This Shannon's maze opens door to new results in many fields such as graph theory (breadth-first-search) and AI applications (e.g., our Internet!).

Theseus: Shannon's Mouse-in-Maze

The trail of Theseus is highlighted by trial-and-error means. It does not necessarily choose the best way if there are two different ways to reach the target, although choosing the shorter one is highly probable.



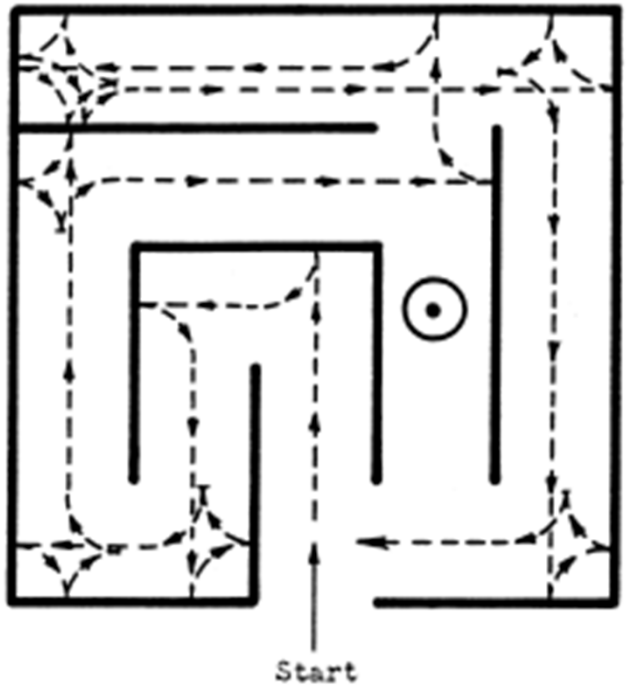


Fig. 6--Shannon's 1952 Maze

http://cyberneticzoo.com/mazesolvers/1952-%E2%80%93-theseus-maze-solving-mouse-%E2%80%93-claude-shannon-american/

Our Internet is a-Maze-ing

Another that I learned was that in building self-learning systems it is equally important to forget, as it is to learn. For example, when you destroy parts of a network, the network must quickly adapt to routing traffic entirely differently. I found that by using two different time constants, one for learning and the other for forgetting provided the balanced properties desired. And, I found it helpful to view the network as an organism, as it had many of the characteristics of an organism as it responds to overloads, and sub-system failures.

Dynamic Routing, 1961

Baran:

I first thought that it might be possible to build a system capable of smart routing through the network after reading about Shannon's mouse through a maze mechanism . But instead of remembering only a single path, I wanted a scheme that not only remembered, but also knew when to forget, if the network was chopped up. It is interesting to note that the early simulation showed that after the hypothetical network was 50% instantly destroyed, that the surviving pieces of the network reconstituted themselves within a half a second of real world time and again worked efficiently in handling the packet flow.

Hochfelder:

How would the packets know how to do that?

Baran:

Through the use of a very simple routing algorithm. Imagine that you are a hypothetical postman and mail comes in from different directions, North, South, East and West. You, the postman would look at the cancellation dates on the

Paul Baran: Father of Packet Switching

Packet switching is the method by which the Internet works, as it features delivery of packets of data between devices over a shared network.

https://ethw.org/Oral-History:Paul_Baran