Networking in the Cloud

Gabriel Scalosub

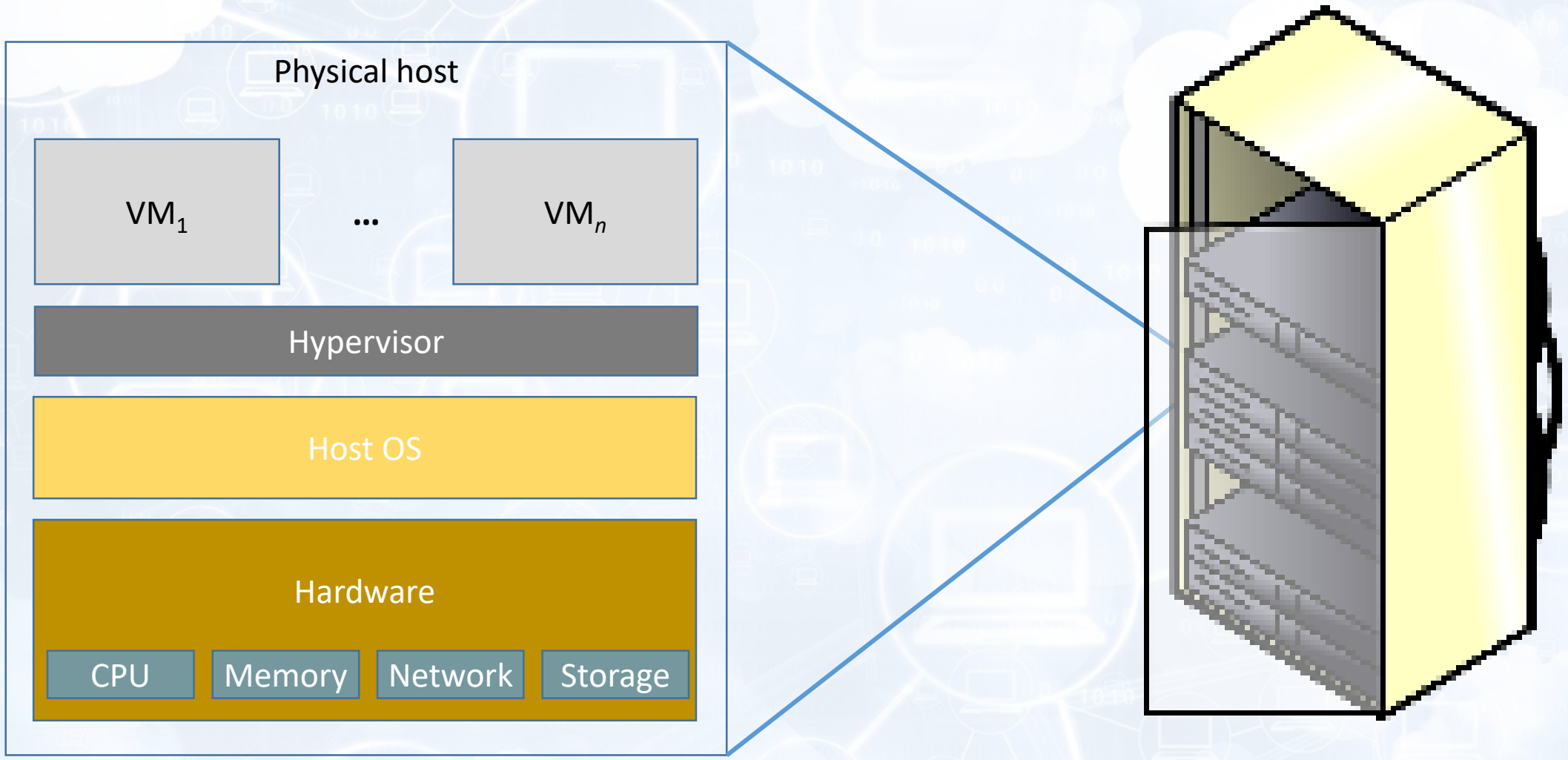
Borrowed extensively from:

Philip Brighten Godfrey (UIUC) and Ankit Singla (ETH), Jeff Mogul and Amin Vahdat (Google), Albert Greenberg (Microsoft), Raj Jain (WUSTL), and various other papers/resources (see list at the end)

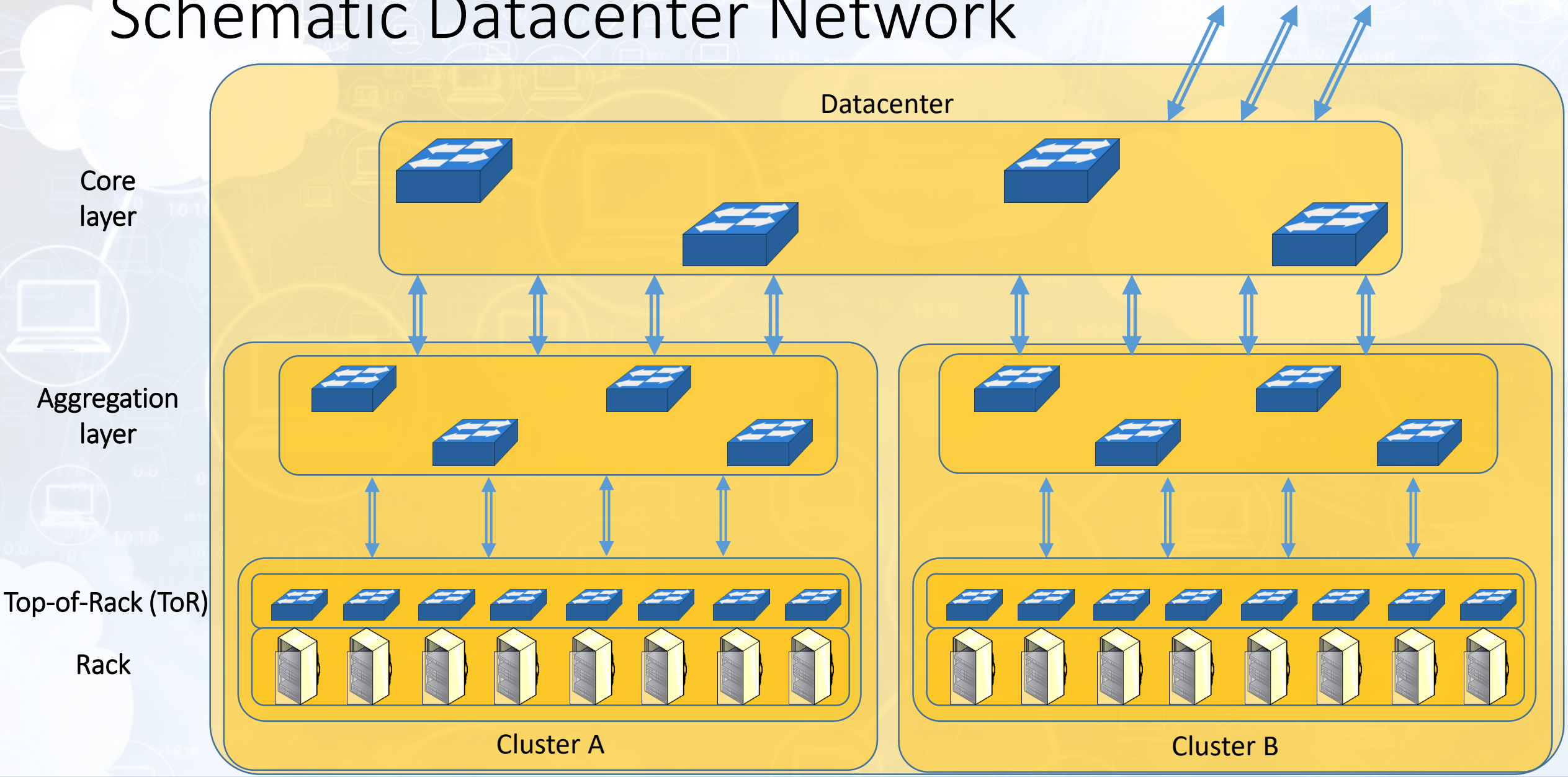
Outline

- Introduction
- Cloud Traffic
- Cloud Architectures
- Routing and Forwarding
  - Equal-cost Multi-path (ECMP)
- Congestion Control
  - Data center TCP (DCTCP)
- Misc

Schematic Cloud Infrastructure



Schematic Datacenter Network

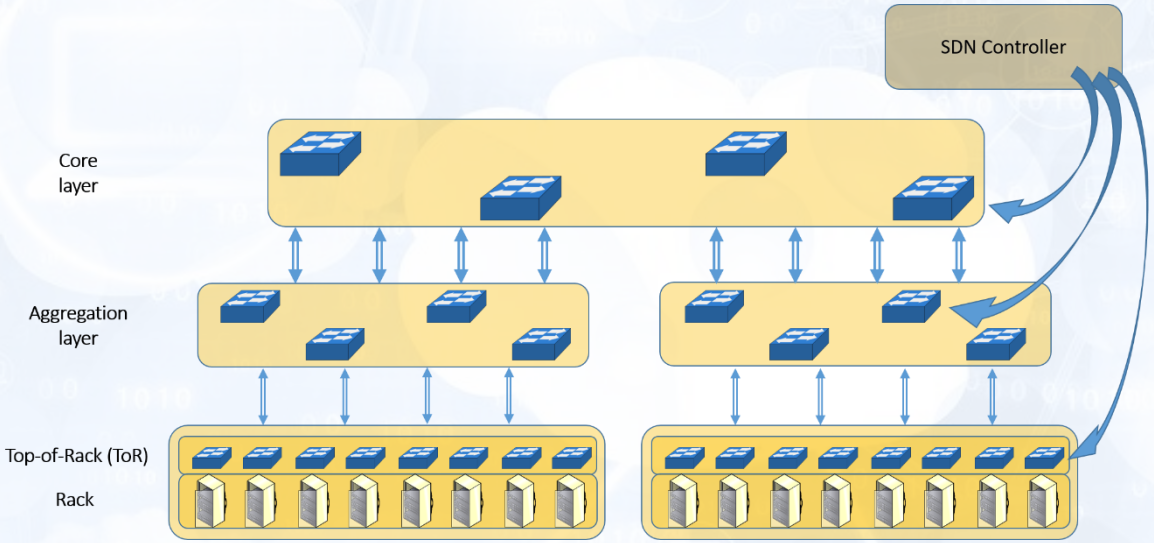


Datacenters: SDN's "killer-app"

- E.g., Google is strongly using SDN
  - In 2017: ~20% of its traffic
  - Today: presumably (much?) more…

Vahdat (2017)

- Trends:



Cisco Global Cloud Index 2018

Outline

- Introduction
- Cloud Traffic
- Cloud Architectures
- Routing and Forwarding
  - Equal-cost Multi-path (ECMP)
- Congestion Control
  - Data center TCP (DCTCP)
- Misc

Cloud Traffic: What's in There?

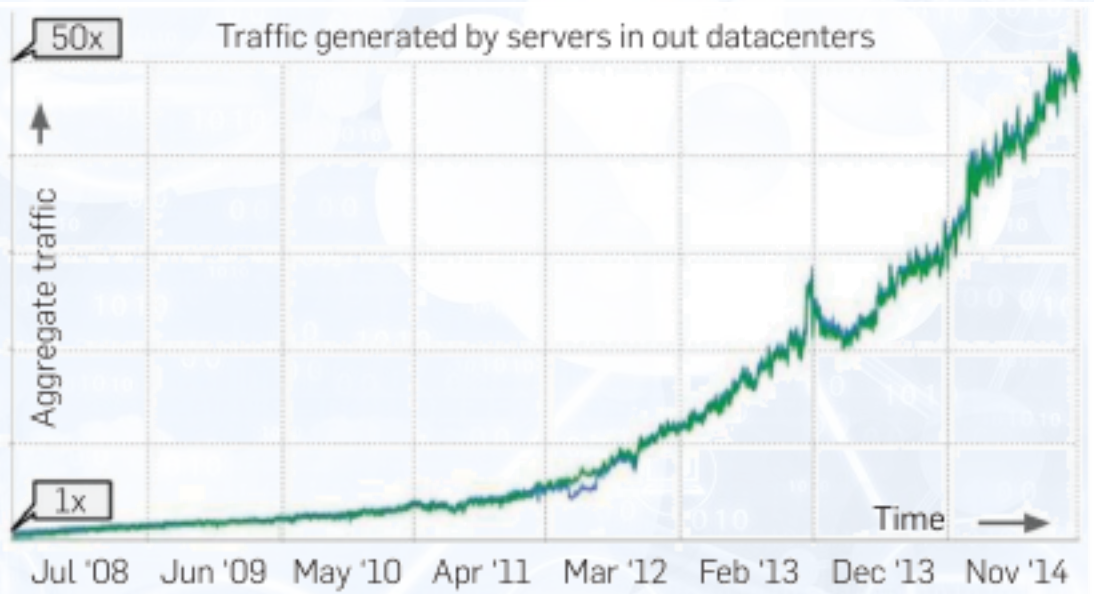
- In plain sight:
  - Applications front-end / servers
    - Web, Facebook, Google, Office, Dropbox, Mail, git
  - Internet-of-Things (IoT) / sensors aggregators
- In the background:
  - Applications back-end
    - Crawling, algorithms, indexing, replication, storage, synchronization
  - "HPC" use-cases
    - Big-data analytics, simulations (weather, scientific, industry, …)
  - Algorithms in general
    - Graphs, ML (sparkML, TensorFlow, …)

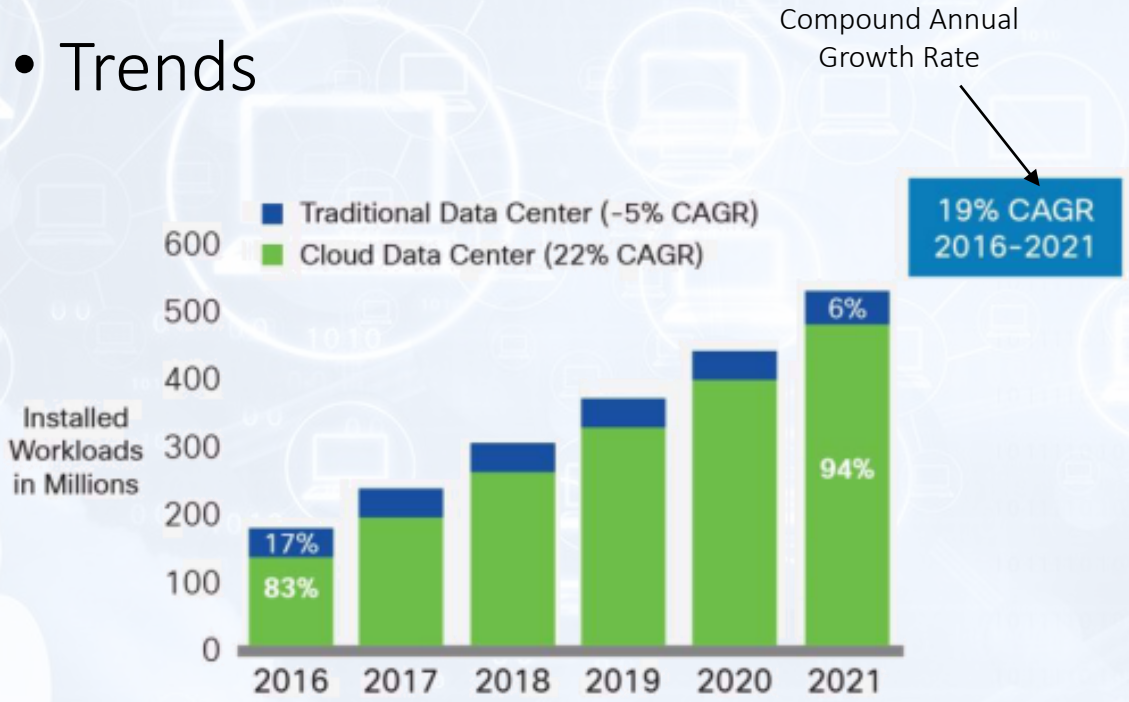
Machine to machine

Cloud Traffic: How Much Is In There?

Singh et al.: “Jupiter Rising” (2015) [Google]

- Tremendous increase in traffic
  - 50x increase over 6 years
- Trends

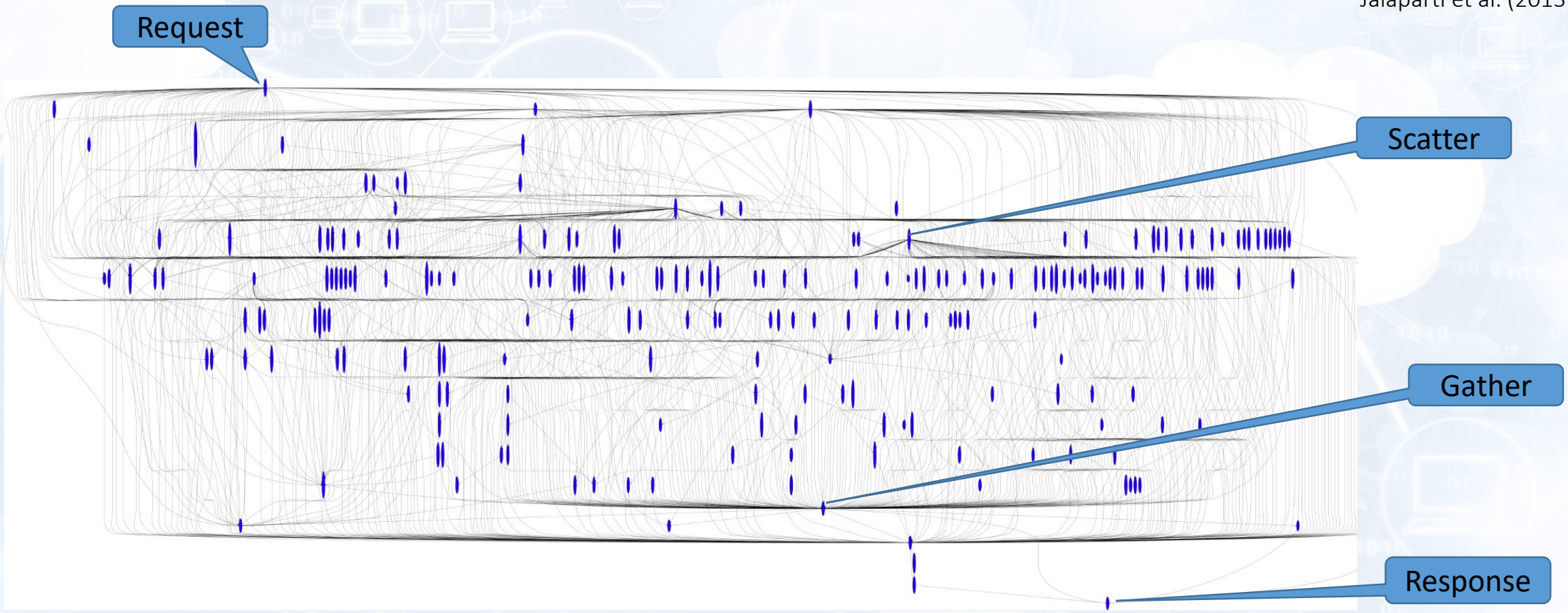




Cisco Global Cloud Index 2018

Cloud Traffic: The Scatter-Gather Use-case

Jalaparti et al. (2013)



- Very short per-hop deadlines (10ms)
- Up to 150 stages (servers…), degree of 40, path length of 10

Cloud Traffic: The Scatter-Gather Use-case

- Assume
  - Single server response < 10ms: 99% of the time
  - Single server response > 1s: 1% of the time
- Consequence for 100 servers involved in single request:
  - Overall response > 1s: 63% of the time!!

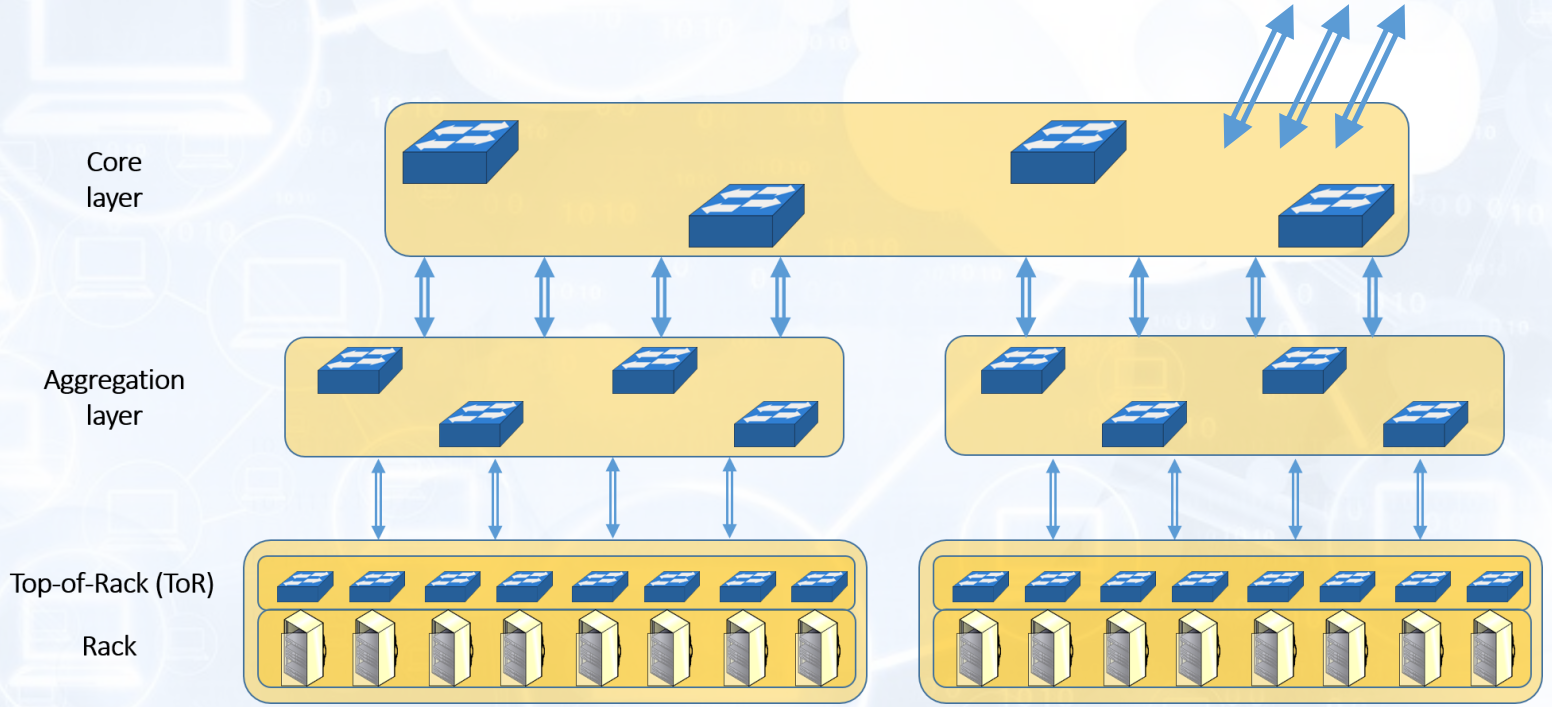
$$1 - (1 - p)^n = 1 - (1 - 0.01)^{100} \rightarrow 1 - 1/e = 0.63$$

- What could we do?
  - Parallel requests and take the shortest (redundancy)
  - use “fast” stages for gathering (e.g., leave out slow responses for search engine query)

- Very short per-hop deadlines (10ms)
- Up to 150 stages (servers…), degree of 40, path length of 10

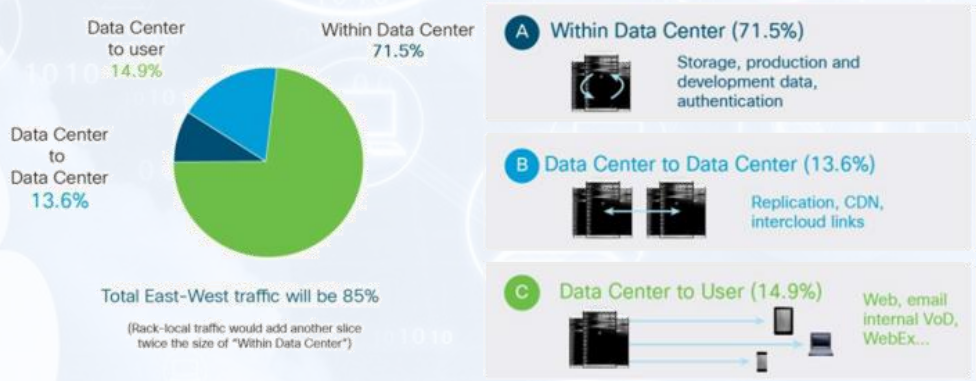
Cloud Traffic: Winds Rose

- North-South traffic
  - Out/In the DC
- East-West traffic
  - inter-cluster
  - intra-cluster inter-rack
  - Intra-rack



Cloud Traffic: Locality

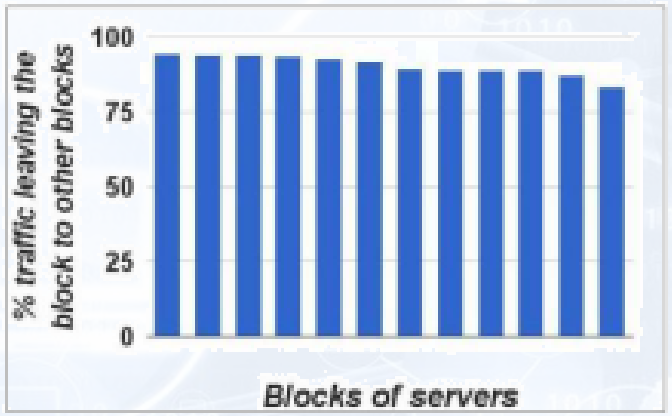
- FB: Most traffic is not intra-rack
  - ~30% not even intra-cluster
- Google: Racks in cluster organized in blocks
  - Storage/work split uniformly across blocks
  - Availability (high) vs. locality (low)
- Trends (by Cisco):



Cisco Global Cloud Index 2018

Locality	All
Rack	12.9
Cluster	57.5
DC	11.9
Inter-DC	17.7
Percentage	

Roy et al. (2015) [Facebook]



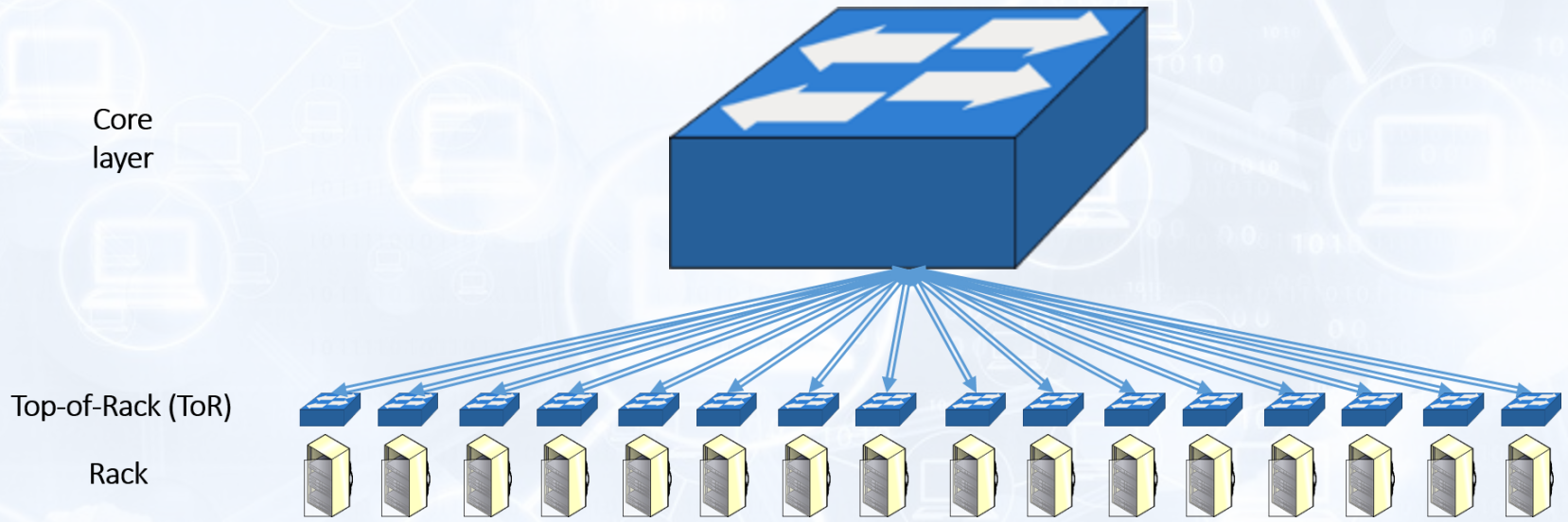
Singh et al., "Jupiter Rising" (2005) [Google]

Outline

- Introduction
- Cloud Traffic
- Cloud Architectures
- Routing and Forwarding
  - Equal-cost Multi-path (ECMP)
- Congestion Control
  - Data center TCP (DCTCP)
- Misc

Cloud Architecture: The Ideal

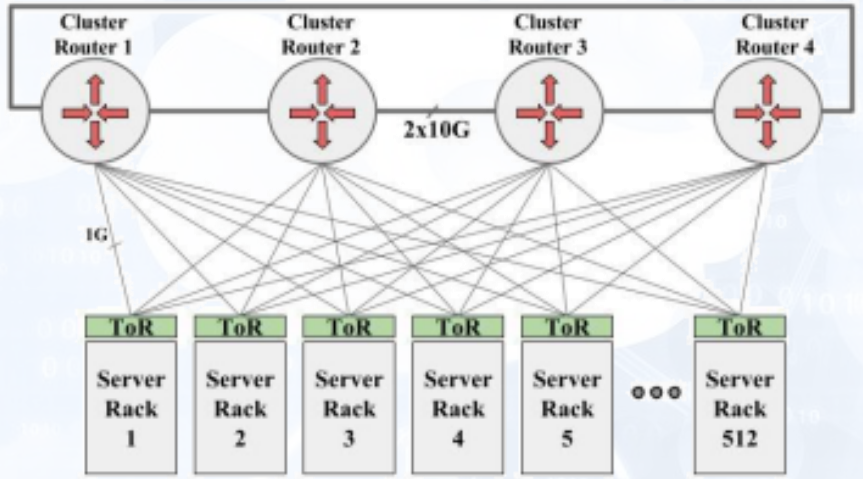
- A VERY big crossbar switch connecting all racks
  - n^2 internal connections, where n can easily be on the order of 1000s and more
  - Not feasible
- How do we solve hard problems (think “Algorithms 101”)?
  - “Special cases”
  - “Approximation”

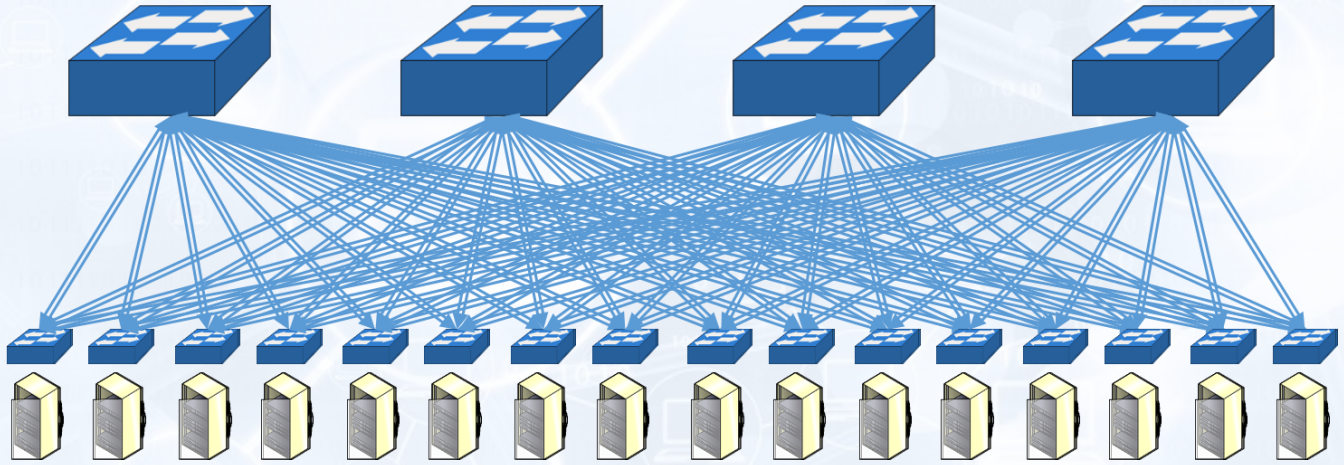


Cloud Architecture: Small DCs

Singh et al., “Jupiter Rising” (2015)

- Actually use such “big” crossbar switches
  - Several, for redundancy
- Googles 2004 DC network
  - 512 racks
  - 40 servers/rack
  - 1G / server: Total 40G
  - Only 4G out of rack…
- Highly non-scalable
  - Restricted deployment
  - Congestion
  - Port-count, cost

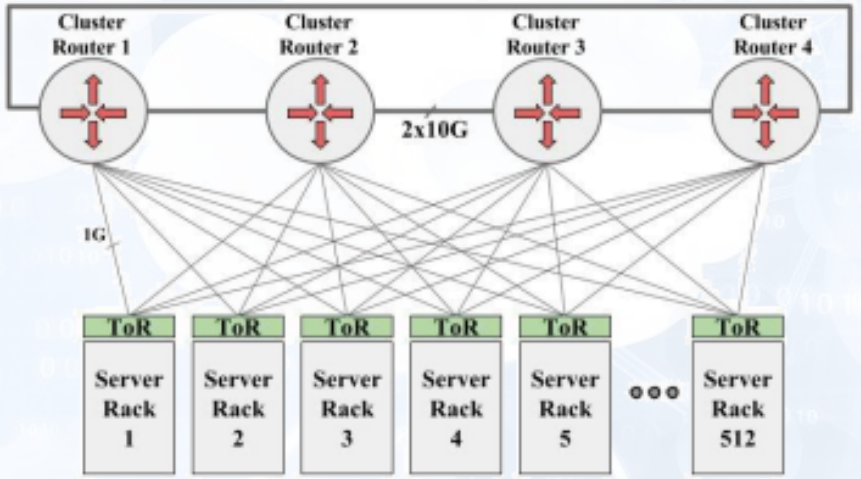


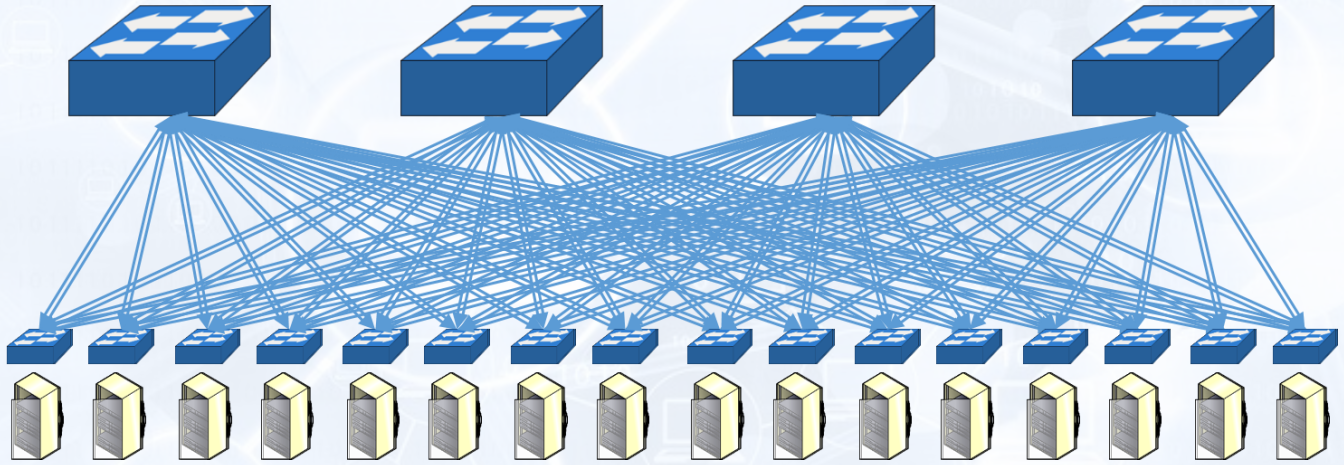


Cloud Architecture: (Non-)Blocking

Singh et al., “Jupiter Rising” (2015)

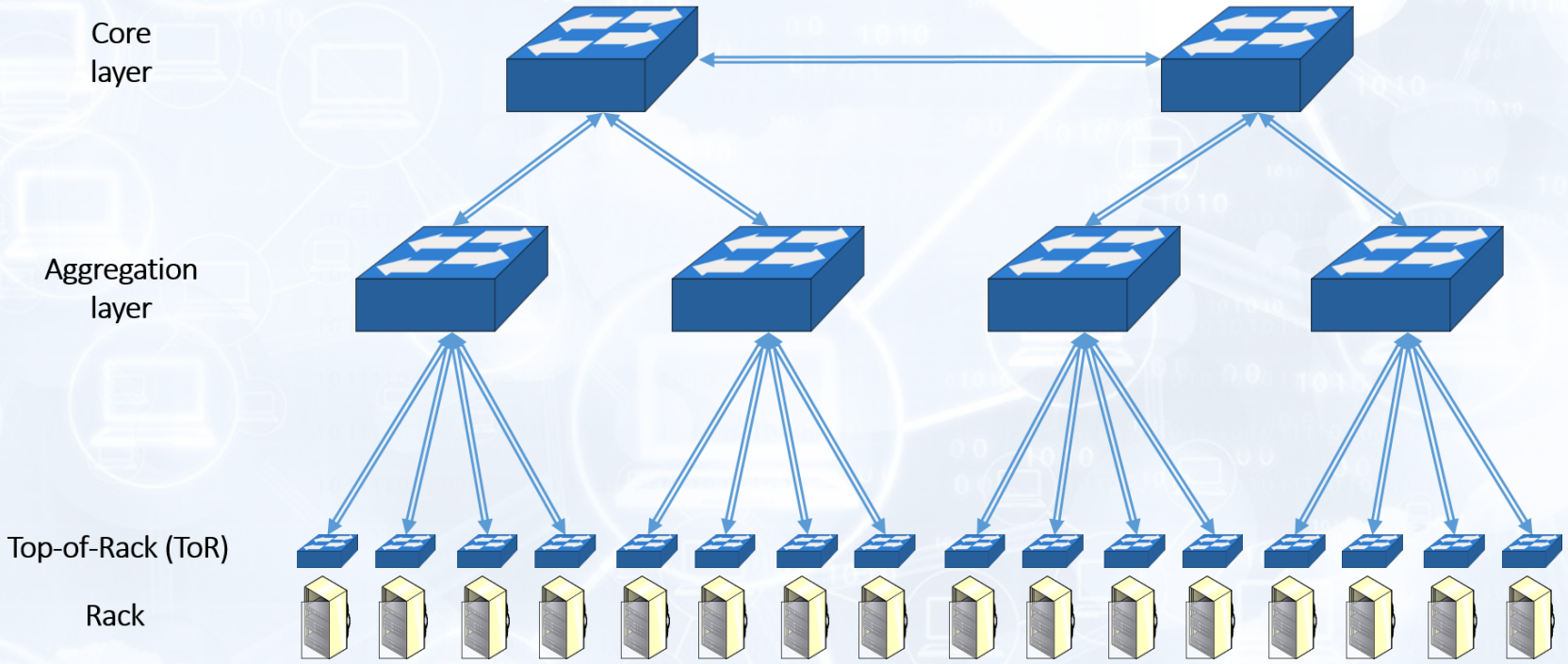
- Up:Down ratio capacity
  - 1:1 is called *non-blocking*
    - All hosts can transmit & receive at (host) line rate
  - 1:k (k>1) is called *blocking*
    - Reduced cost, higher congestion
- What’s the up:down ratio here?
  - 4:40 = 1:10 = 10%
    - Very congested…
    - Oversubscribed





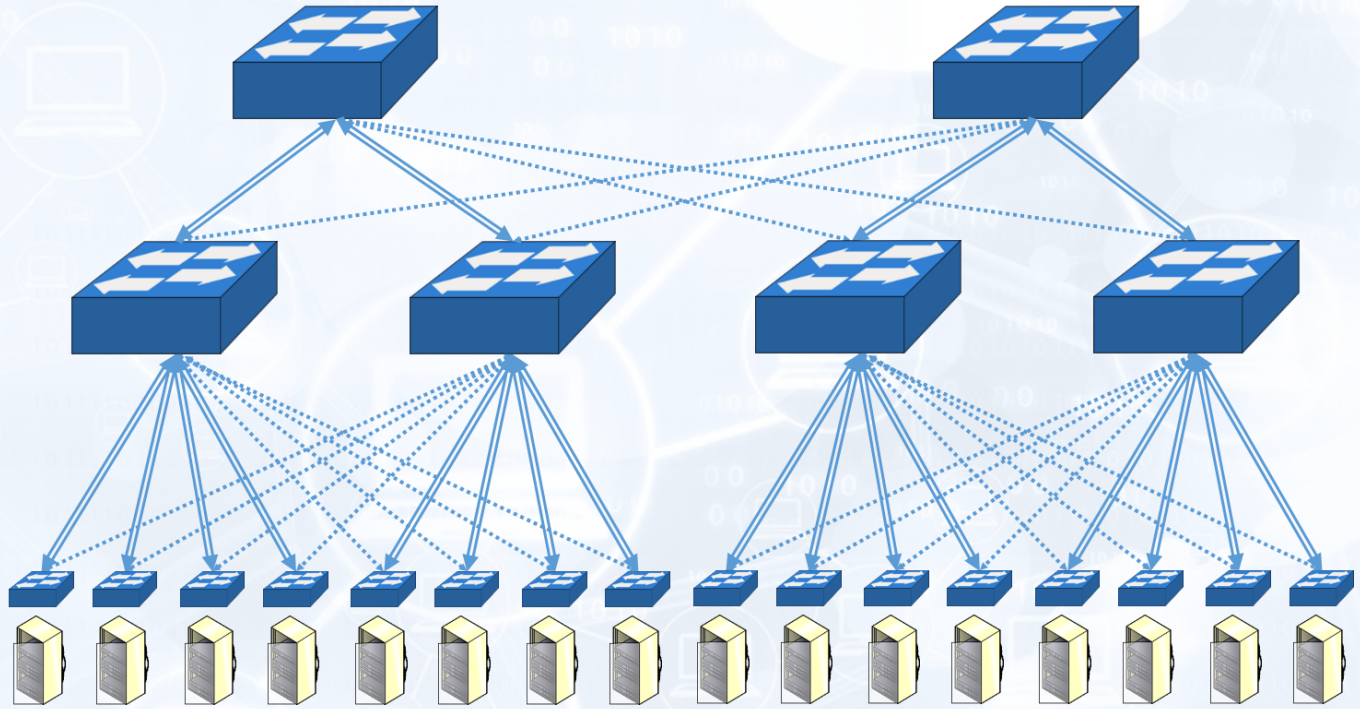
Cloud Architecture: Simple Hierarchies

- Simplest hierarchical network: Tree
  - Smaller port-count, relatively cheap



Cloud Architecture: Simple Hierarchies

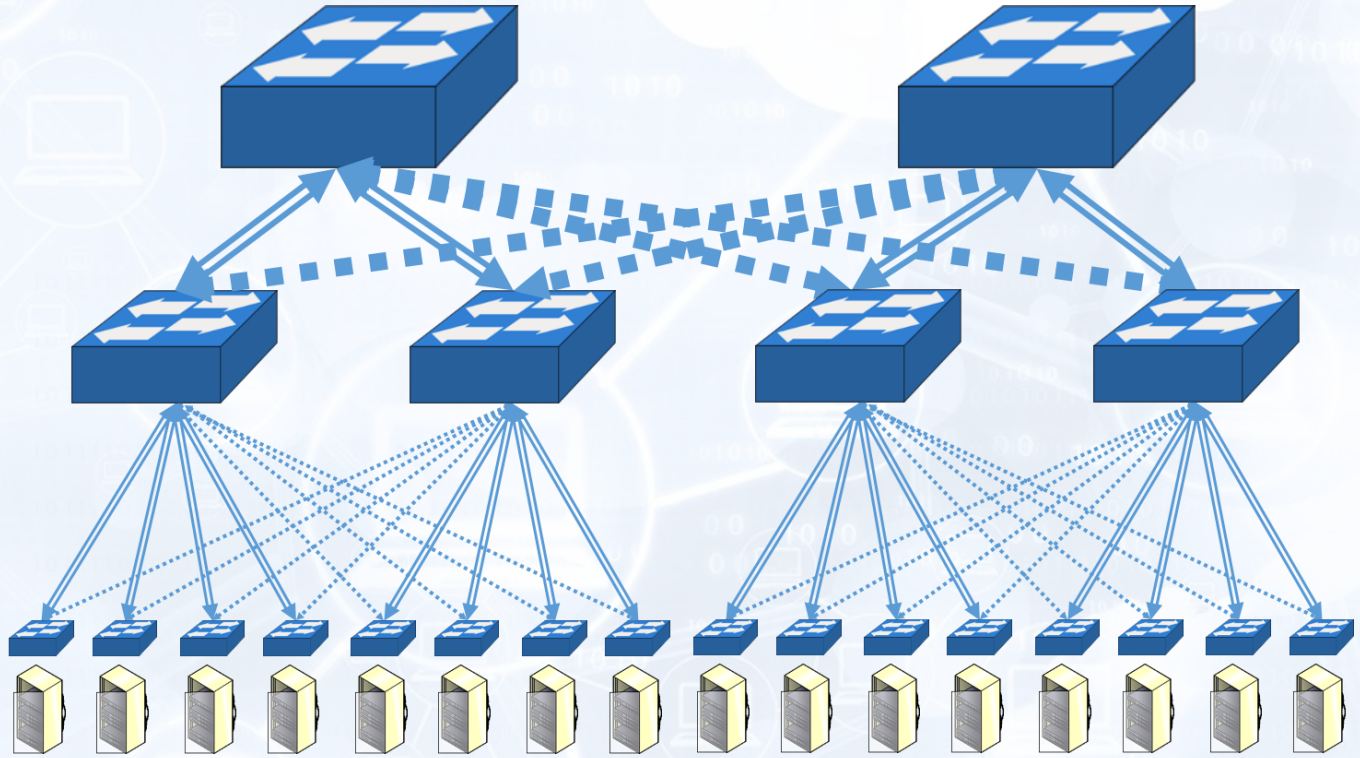
- Simplest hierarchical network: Tree
  - Smaller port-count, relatively cheap
- Usually with some redundancy
- Cost/Port-count: OK
- Semi-scalable
  - Inter-cluster traffic
    - All through “roots”
    - Highly congested
  - Restricted deployment
  - Failure-sensitive



Cloud Architecture: Simple Hierarchies

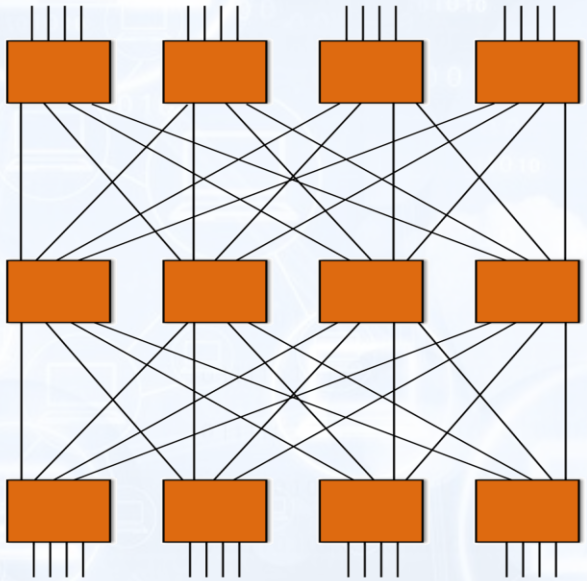
- Links/routers farther up: higher capacity
- Inter-cluster congestion: OK
- Still semi-scalable
  - Cost/port-count
  - Restricted deployment
  - Failure-sensitive
- AKA, Fat-trees…
- Remember placement
  - high traffic pairs:

    place “together”



Cloud Architecture: Clos Networks

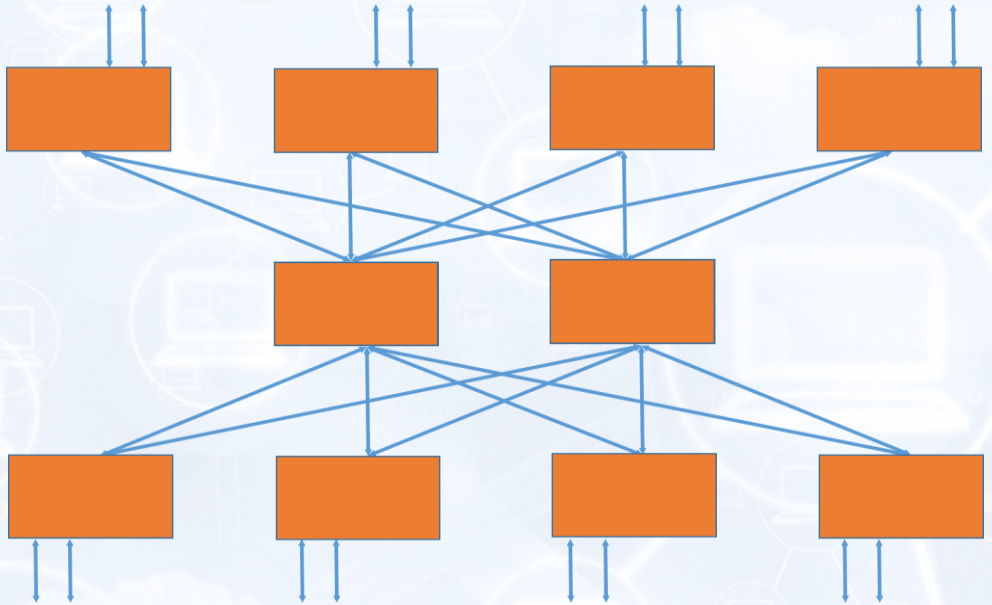
- Small-switches (8-port) to large-switches (32-port)
- Use simple (i.e., cheap…) switches to obtain high port-count



- “Capacity”: Min-Cut…

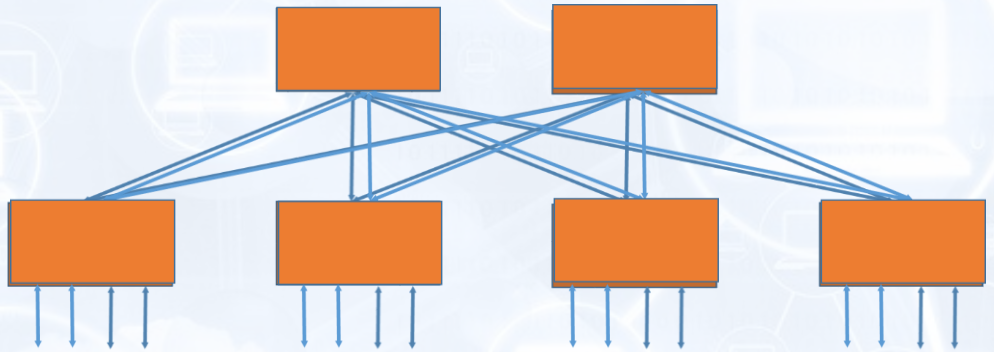
Cloud Architecture: From Clos to Fat-tree

- From Clos to Fat-tree
  - Clos: 10 × 4-port → 16-port



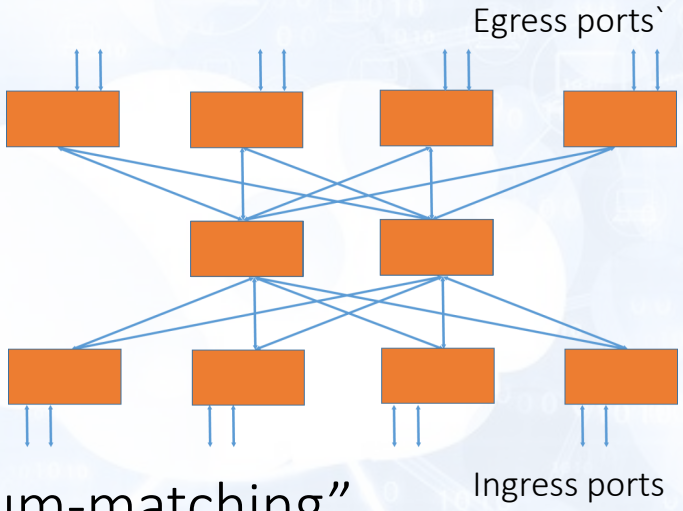
Cloud Architecture: From Clos to Fat-tree

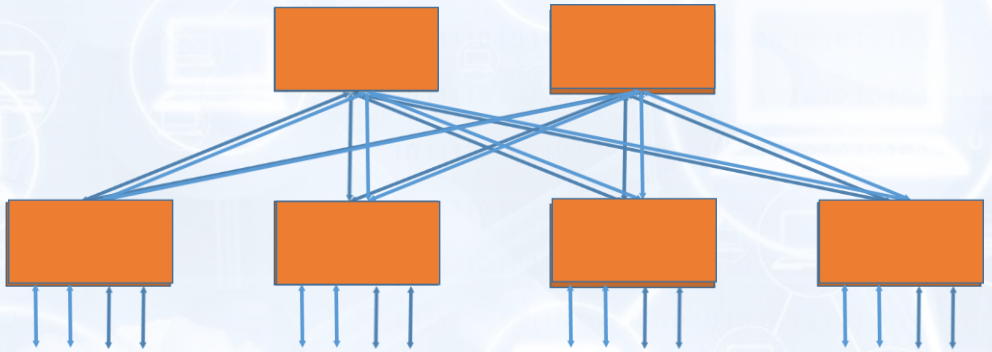
- From Clos to Fat-tree
  - Clos: 10 × 4-port → 16-port
  - Now fold…
  - Fat-tree



Cloud Architecture: From Clos to Fat-tree

- From Clos to Fat-tree
- Non-blocking:
  - *Strict-sense*: “maximal-matching” is “maximum-matching”
    - “sufficient capacity” in middle stages to add connections online
  - *Rearrangeable*: “maximal-matching” can become “maximum-matching”
    - “sufficient capacity” in middle stages if all connections available up-front, OR
    - Allowed to “reroute” some existing connections

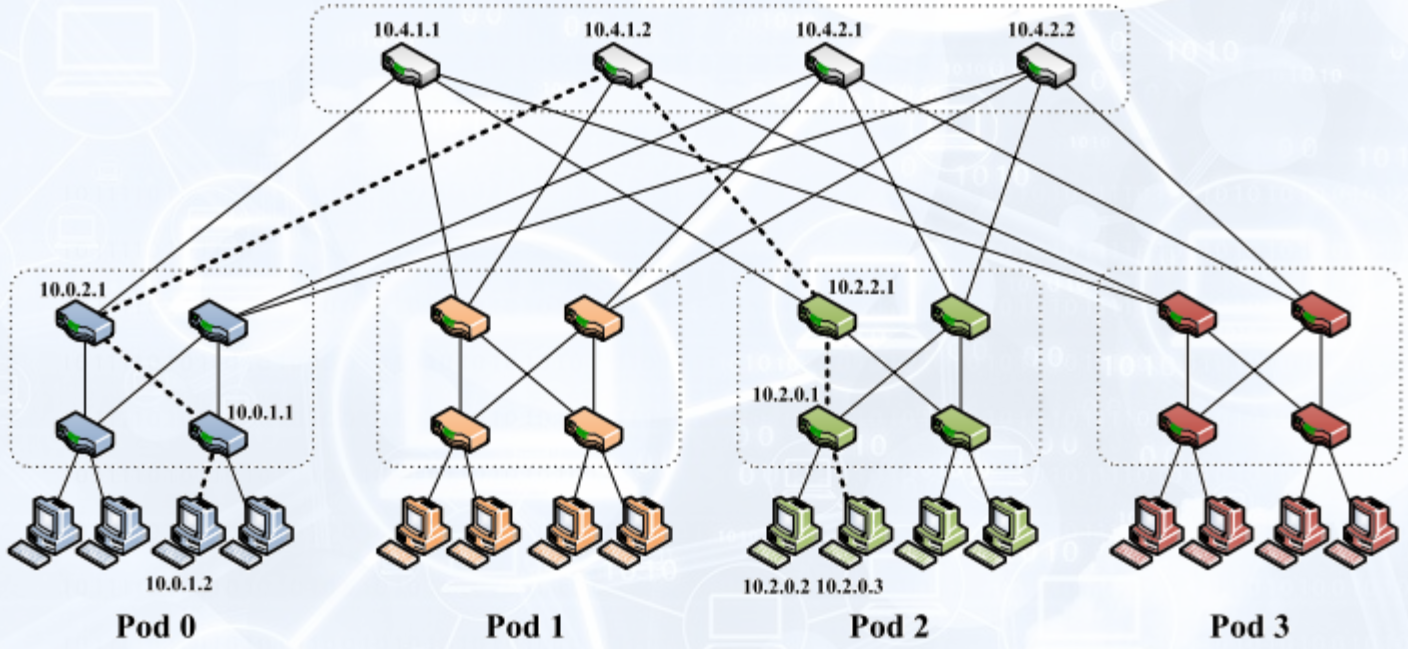




Cloud Architecture: Fat-trees

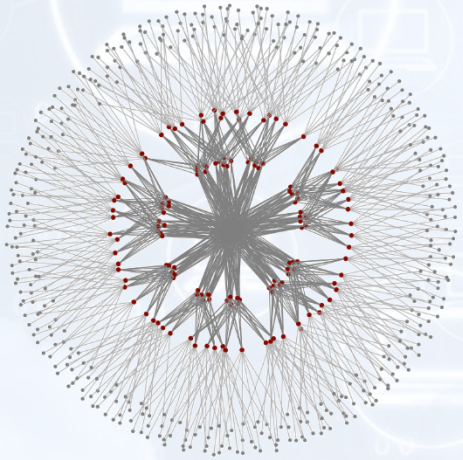
- Layers are structured as a collection of Clos networks
- E.g., all switches with 4 ports

Al-Fares et al. (2008)



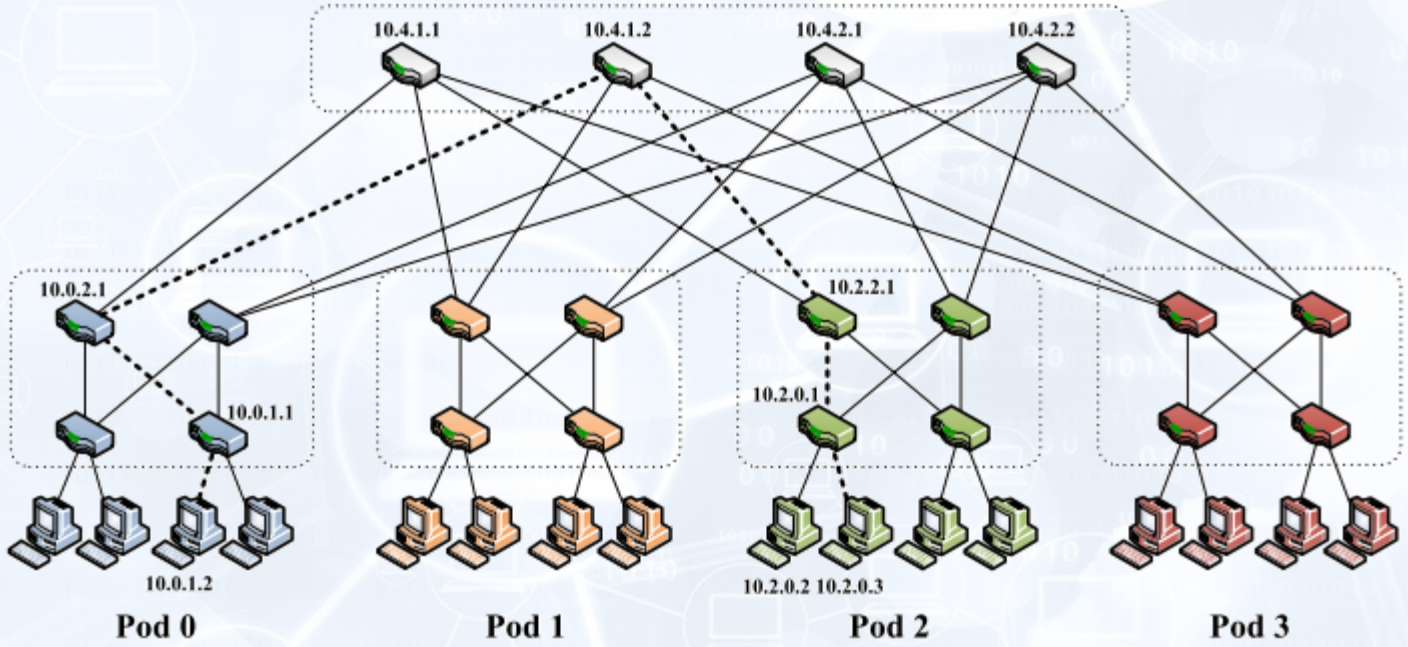
Cloud Architecture: Fat-trees

- Locality vs. capacity
  - Different layers -> different capacities
  - Might restrict deployment
- Redundancy:
  - Multiple paths



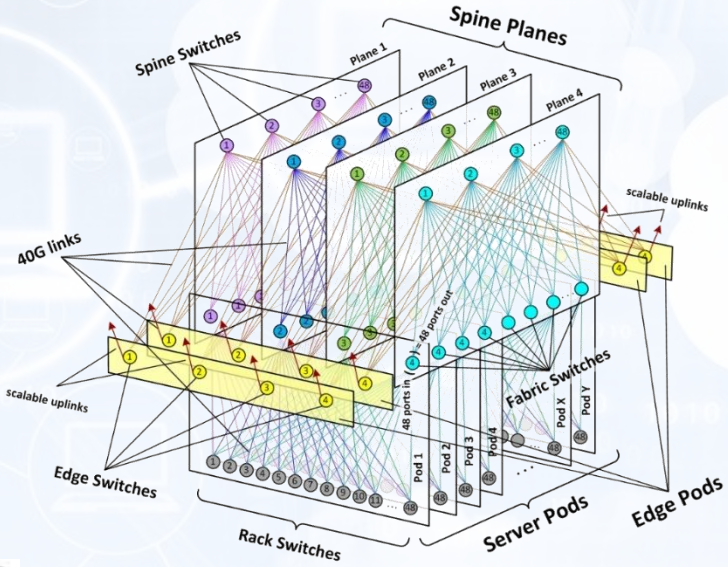
Singla et al, “Jellyfish” (2012)

Al-Fares et al. (2008)



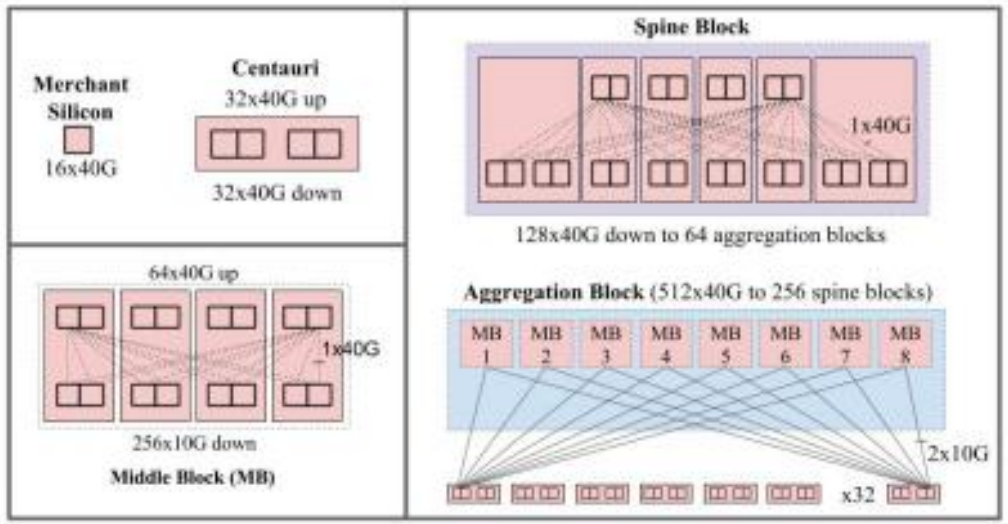
Cloud Architecture: Fat-trees Variants Deployed

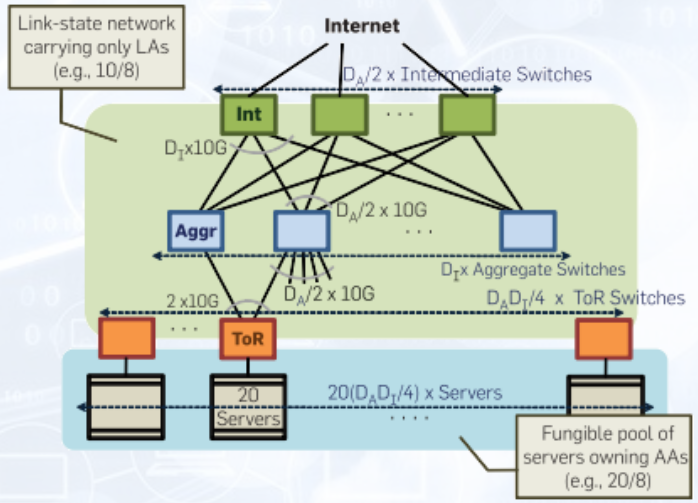
- VL2 @ Microsoft
- Jupiter @ Google
- Facebook DC



Andreyev (2014) @ FB

Singh et al., “Jupiter Rising” (2015) @ Google





Greenberg et al., “VL2” (2009) @ Microsoft

Outline

- Introduction
- Cloud Traffic
- Cloud Architectures
- Routing and Forwarding
  - Equal-cost Multi-path (ECMP)
- Congestion Control
  - Data center TCP (DCTCP)
- Misc

Routing and Forwarding

- Baseline: L2 spanning tree
  - Uses but a fraction of the capacity
  - Failure sensitive
- Link-state protocols (OSPF-like)
  - Flood link information
  - L3 address aggregation
  - “manipulable” by altering link costs
- BGP-like
  - L3 address aggregation
  - DC-interoperability
  - Relatively slower reaction to change/failure

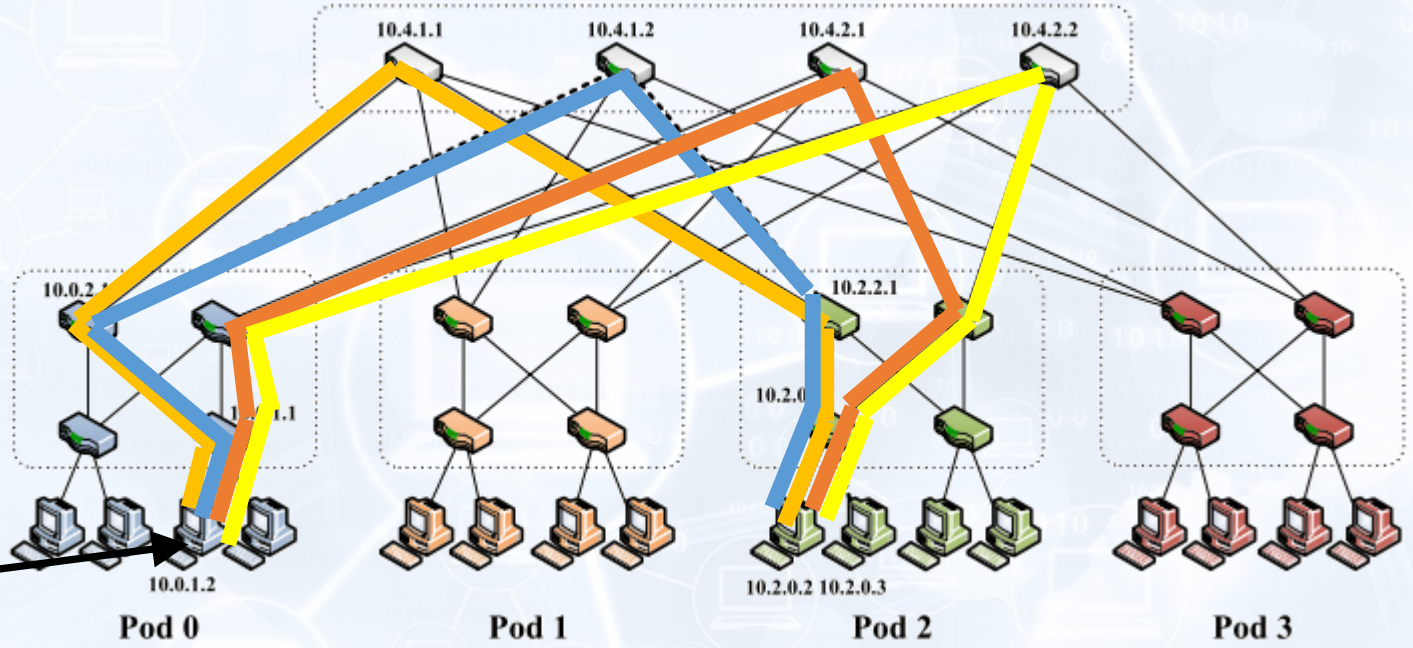
Single path from source to destination

Routing and Forwarding: Multiple Paths

Al-Fares et al. (2008)

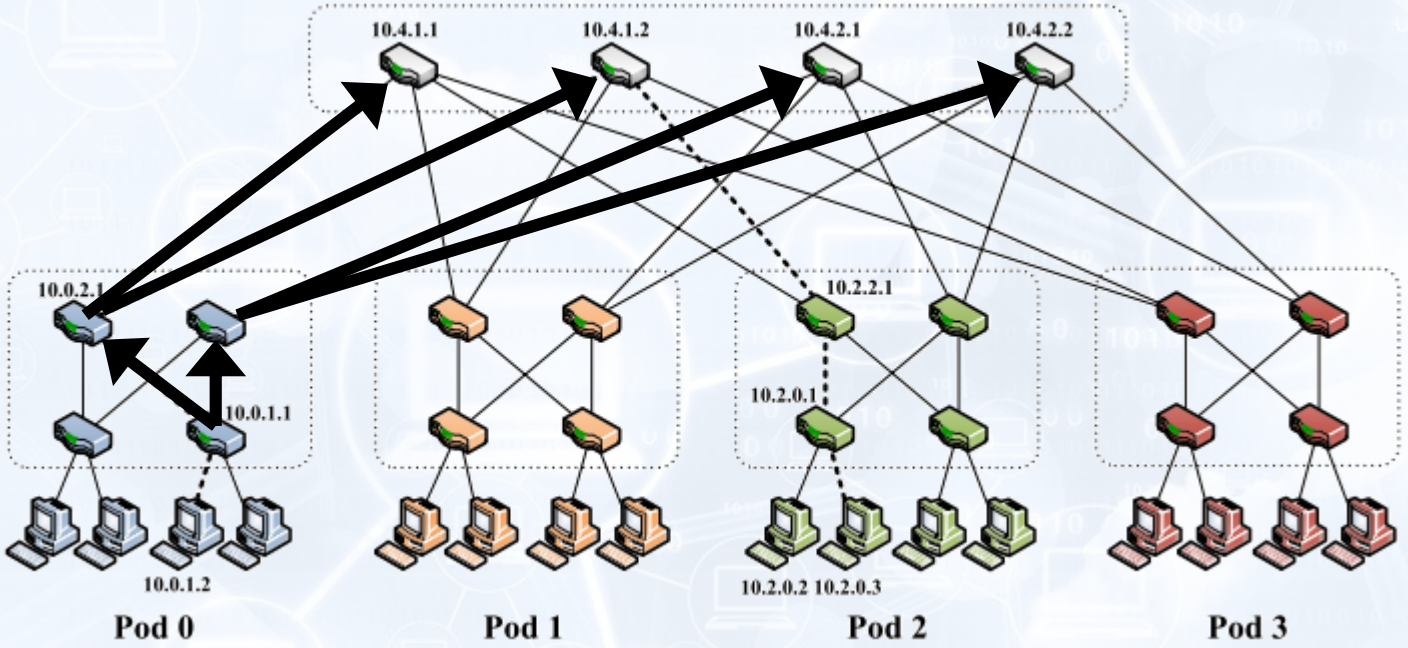
- We would like to take advantage of network density
  - Handle short-flows
  - Fast: no heavy “rerouting” protocol, gathering link states
  - Distributed (no central control)
- Effectively, load balancing
- Main idea: random selection
  - Problematic for TCP if done on packet-level
    - Variable queuing delay
    - reordering

Destination	Via port
10.2.0.2	1
10.2.0.2	2



Equal-Cost Multi-Path (ECMP)

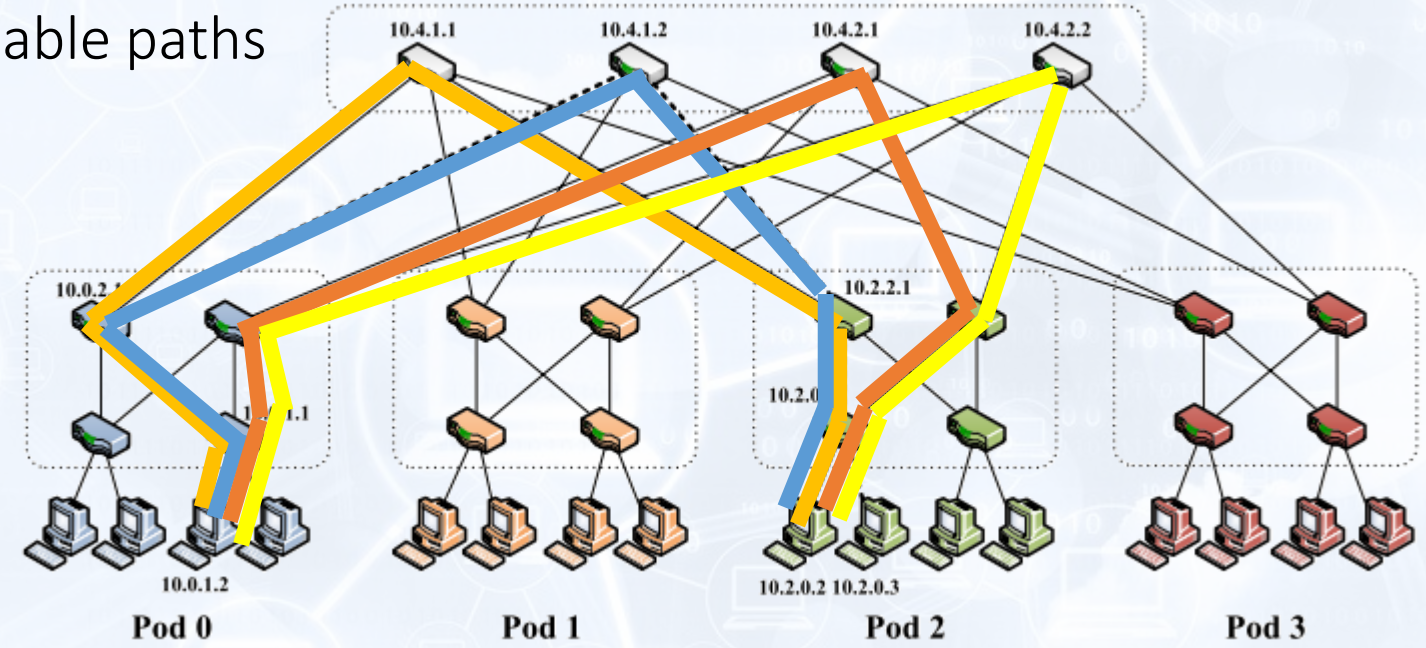
- Output-port = hash (packet header)
  - Per-flow random choice (L2/L3)
  - No TCP-reordering
- Performed in every hop, independently



Equal-Cost Multi-Path (ECMP)

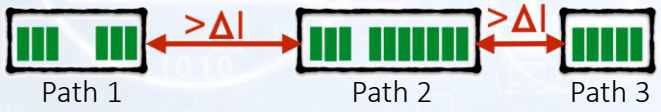
- Output-port = hash (packet header)
  - Per-flow random choice (L2/L3)
  - No TCP-reordering
- Performed in every hop, independently
  - Distributes flows over all available paths

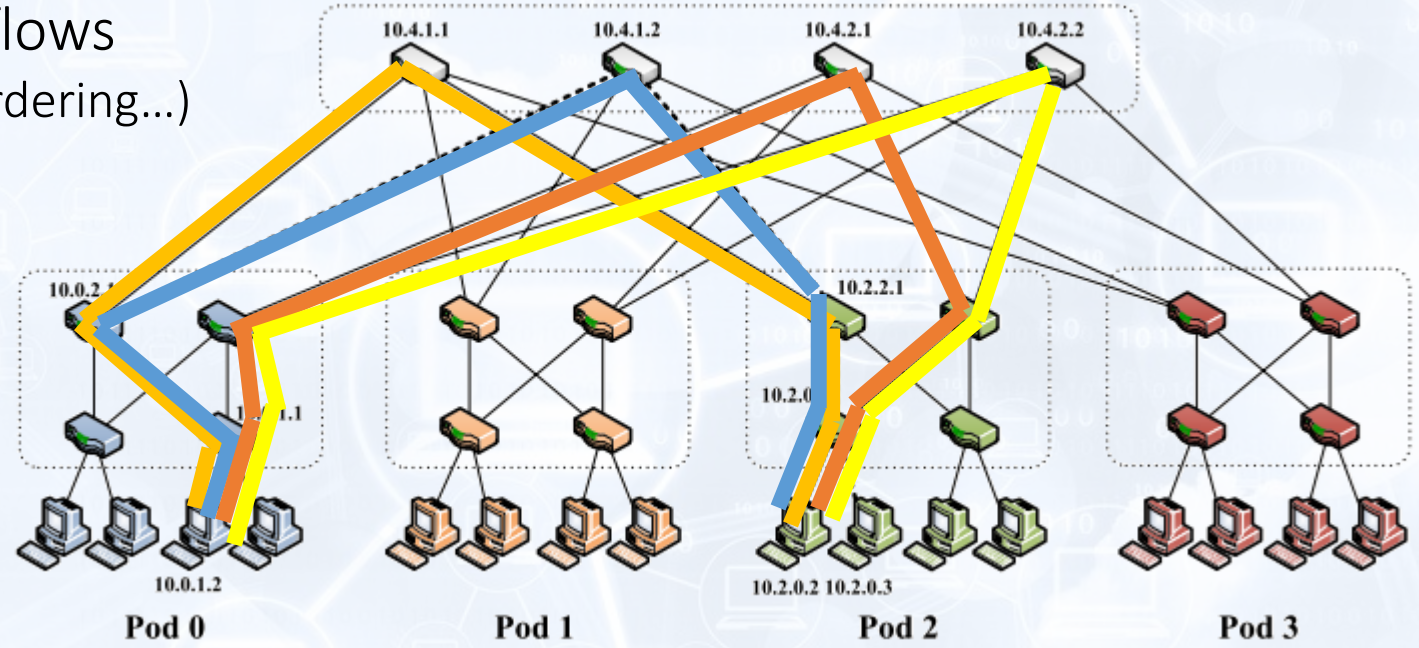
remember VXLAN dynamic UDP source port…



Equal-Cost Multi-Path (ECMP)

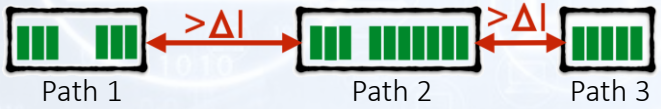
- Many improvements
  - Remote congestion awareness
    - Gather/maintain usage statistics
    - Pick least-congested path
  - Network-wide decomposition
  - Flowlets: well separated sub-flows
    - Can take distinct paths (no reordering…)



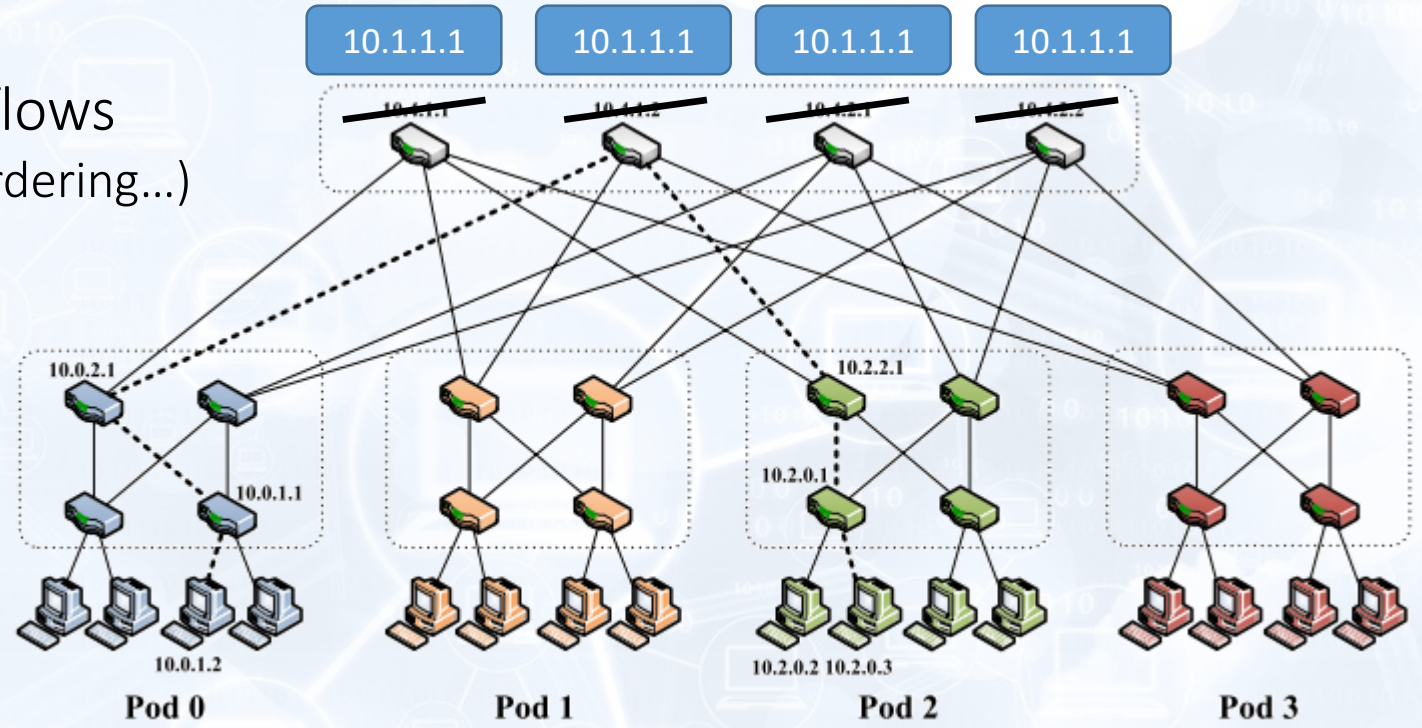


Equal-Cost Multi-Path (ECMP)

- Many improvements
  - Remote congestion awareness
    - Gather/maintain usage statistics
    - Pick least-congested path
  - Network-wide decomposition
  - Flowlets: well separated sub-flows
    - Can take distinct paths (no reordering…)



- Anycast addresses
  - Virtualization/encapsulation
  - Smaller forwarding tables
    - Switches on the way up
  - ECMP does “heavy lifting”

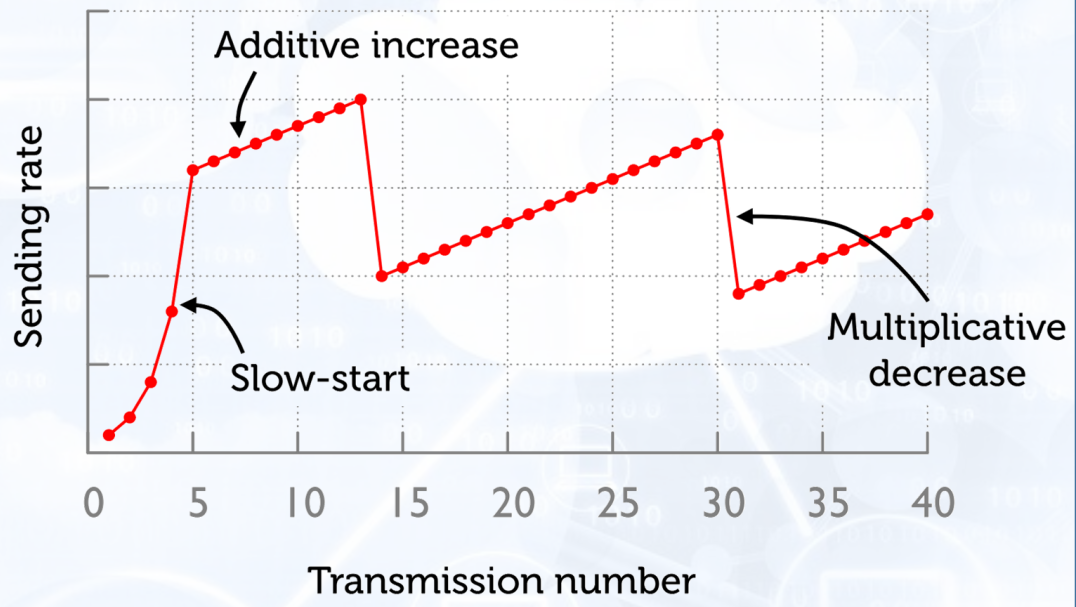


Outline

- Introduction
- Cloud Traffic
- Cloud Architectures
- Routing and Forwarding
  - Equal-cost Multi-path (ECMP)
- Congestion Control
  - Data center TCP (DCTCP)
- Misc

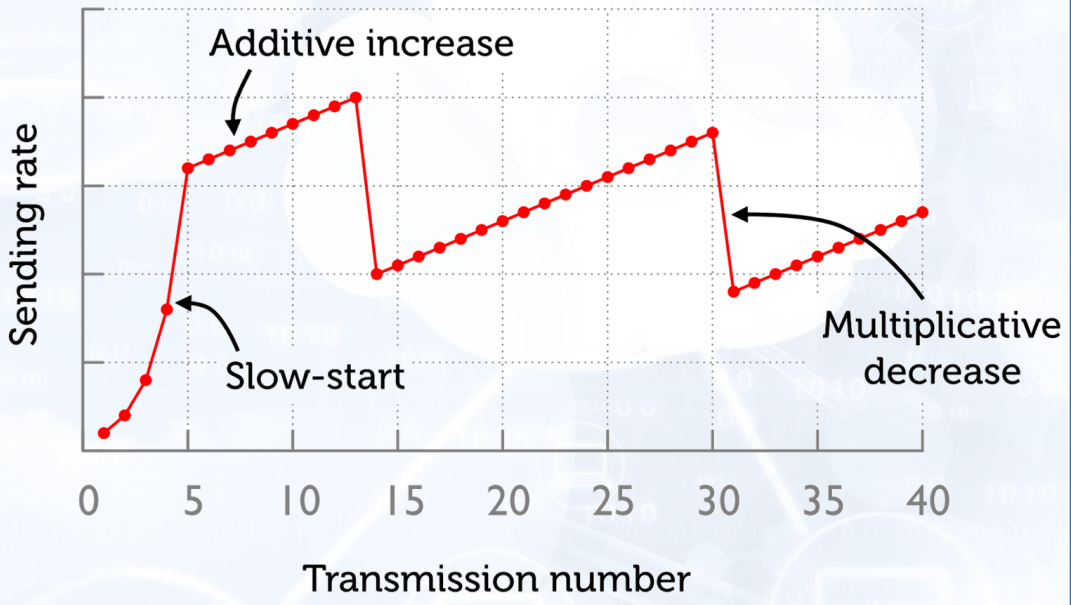
Congestion Control: Standard TCP

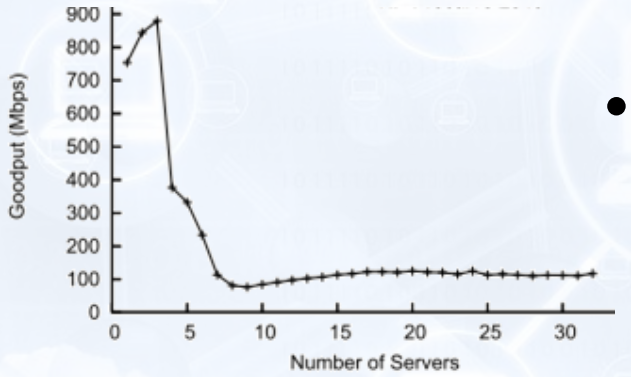
- Sharing the available capacity
  - High throughput
    - Use (large) buffers to absorb bursts
  - Low latency
    - (Large) full buffers delay packets
- Dynamic
  - Flows go up and down
  - Queueing delays
- TCP-like congestion control
  - Send packets
  - Wait for ACK (< RTT)
  - Adjust sending rate



Congestion Control: Standard TCP

- Issues (especially in the datacenter)
  - Latency mostly due to queueing delay
    - Sometimes just transient long buffer(s)
    - Multiplicative decrease too harsh
  - No flow isolation
  - TCP Incast
    - Multiple servers accessing single server
      - E.g., gather-phase of scatter-gather
    - Throughput collapse



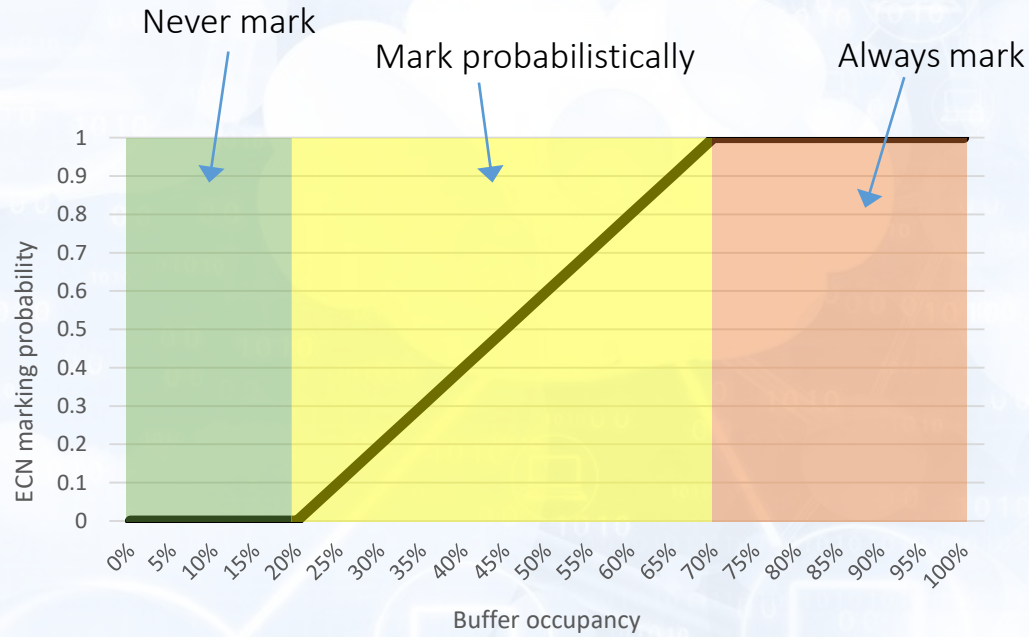


Phanishayee et al. (2008)

- Goal: try and hit the right rate
  - Early
  - More smoothly

Explicit Congestion Notification (ECN)

- Switch side:
  - Mark packet ECN bit by AVG buffer occupancy
  - High & low thresholds
- Receiver side:
  - Copy ECN bit onto ACK
- Sender side upon receiving ACK
  - If ECN bit is set, adjust rate
- Identify congestion before packet loss



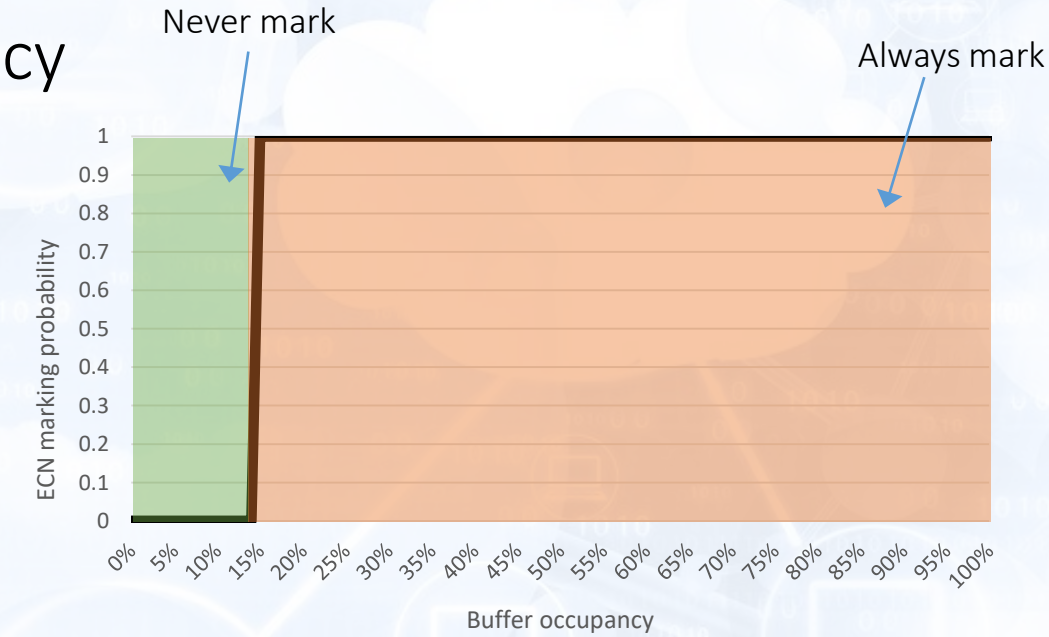
- Goal: try and hit the right rate
  - Early
  - More smoothly

Congestion Control: Datacenter TCP (DCTCP)

Alizadeh et al., "Data Center TCP (DCTCP)", SIGCOMM 2010

- ECN bit set: *instantaneous* buffer occupancy
  - Very fast reaction to buffer load
  - Single (low) threshold
    - Queues are kept small -> low latency
    - Can still accommodate large bursts
- Sender
  - Maintain moving average of marked packets
    - S = #marked / #acks
    - MA = (1-α)MA + αS
  - Adapt window
    - W = (1 - MA/2) W
    - Decrease factor between 1 and 2
    - No harsh multiplicative decrease

MA→ 1:	Many ECN 1-bits W → ???
MA→ 0:	Few ECN 1-bits W → ???



- Goal: try and hit the right rate
  - Early
  - More smoothly

Congestion Control: Datacenter TCP (DCTCP)

Judd (2015)

	TCP	DCTCP+
Mean	4.01	0.0422
Median	4.06	0.0395
Maximum	4.20	0.0850
Minimum	3.32	0.0280
σ	0.167	0.0106

- Performance
  - Per-packet latency in ms
  - Recall: queues are maintained small
  - What happens to TCP flows if DCTCP is used?
    - DCTCP is not TCP-friendly: hard grind of regular TCP flows
    - Can be partially solved by considering them as two separate QoS classes
  - Doesn't require special HW capabilities
- Smaller buffers suffice if #flows large (statistical multiplexing)
  - C×RTT/$\sqrt{n}$ instead of C×RTT

Appenzeller et al. (2004)

Outline

- Introduction
- Cloud Traffic
- Cloud Architectures
- Routing and Forwarding
  - Equal-cost Multi-path (ECMP)
- Congestion Control
  - Data center TCP (DCTCP)
- Misc

Accelerations

- Data Plane Development Kit (DPDK)
  - Libraries for accelerating packet processing
  - Traffic bypassing the host kernel / hypervisor
  - I/O intensive applications recompiled to use DPDK API
    - DPDK driver controls NIC
- Single Root Input/Output Virtualization (SR-IOV)
  - Virtualizing PCI Express (PCIe) devices
    - Actual device: Physical Function (PF)
    - Virtual devices: Virtual Function (VF)
      - Reduced functionality
      - Full functionality aided by hypervisor/host OS
  - Essentially, virtualizing NIC ports (for physical NICs supporting SR-IOV)
    - Reducing emulation/paravirtualization overhead
    - Close to native I/O performance

High-speed Networking

- InfiniBand (IB)
  - Switched fabric interconnect architecture
    - Not a shared bus
  - Benefits: multi-path, path redundancy, scalability
  - Within the datacenter / HPC cluster
    - Connecting server clusters, network storage
- RDMA over Converged Ethernet (RoCE)
  - RDMA: Remote Direct Memory Access
    - Transfer data directly from application memory buffers
      - Zero-copy, kernel bypass
      - Close to wire-speed performance
    - “Cornerstone” of HPC
      - Also in ML, big data, storage
  - Originally L2, recently also L3

(Partial) Bibliography

- Singh et al., “Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google’s Datacenter Network”, SIGCOMM 2015
- Jalaparti et al., “Speeding up Distributed Request-Response Workflows”, SIGCOMM 2013
- Roy et al., “Inside the Social Network’s (Datacenter) Network”, SIGCOMM 2015
- Cisco Global Cloud Index 2018
- Greenberg, “Container Networking”, ONS 2017
- Vahdat, “Networking Challenges for the Next Decade”, ONS 2017
- Al-Fares et al., “A Scalable, Commodity Data Center Network Architecture”, SIGCOMM 2008
- Greenberg et al., “VL2: A Scalable and Flexible Data Center Network”, SIGCOMM 2009
- Andreyev, “Introducing data center fabric, the next-generation Facebook data center network”, 2014
- Sinha et al., “Harnessing TCPs Burstiness using Flowlet Switching”, HotNets 2004
- Phanishayee et al. “Measurement and Analysis of TCP Throughput Collapse in Cluster-based Storage Systems”, USENIX FAST 2008
- Handley et al., “Re-architecting datacenter networks and stacks for low latency and high performance”, SIGCOMM 2017
- Alizadeh et al., “Data Center TCP (DCTCP)”, SIGCOMM 2010
- https://www.mellanox.com
- Appenzeller et al., “Sizing Router Buffers”, SIGCOMM 2004
- Judd, “Attaining the Promise and Avoiding the Test Pitfalls of TCP in the Datacenter”, USENIX NSDI 2015
- Noormohammadpour et al., “Datacenter Traffic Control: Understanding Techniques and Tradeoffs”, IEEE Comm. Surv. & Tut. 20(2), 2018