DATACENTER NETWORKING

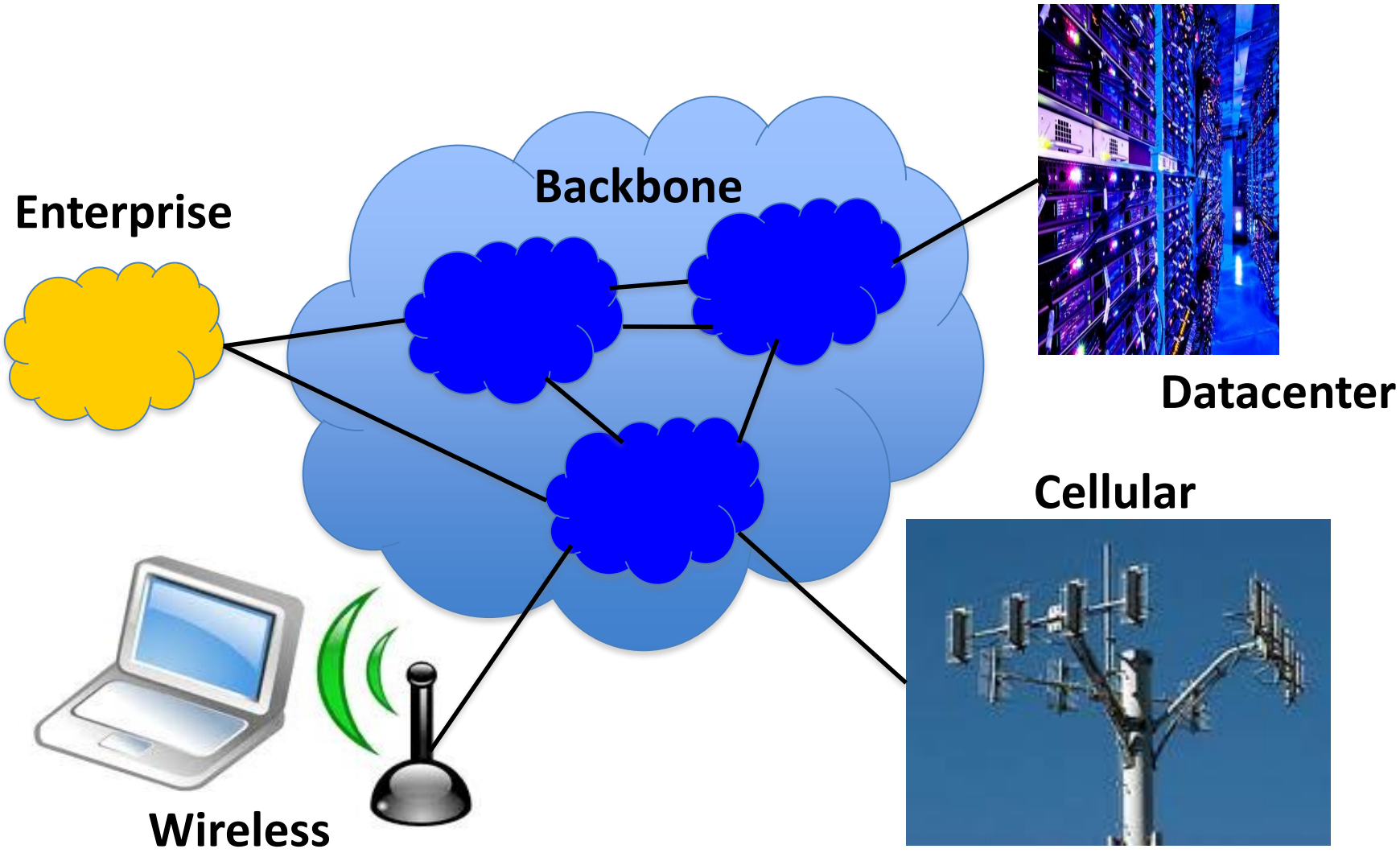
Advaned Computer Networks

David Koll

Exam

Today is the last day to register!

Networking Case Studies



Cloud Computing

- Elastic resources
  - Expand and contract resources
  - Pay-per-use
  - Infrastructure on demand
- Multi-tenancy
  - Multiple independent users
  - Security and resource isolation
  - Amortize the cost of the (shared) infrastructure

Note: First set of slides based on Alex C. Snoeren's lecture on DCN at UCSD: https://cseweb.ucsd.edu/classes/wi14/cse222A-a/lectures/222A-wi14-l7.pdf

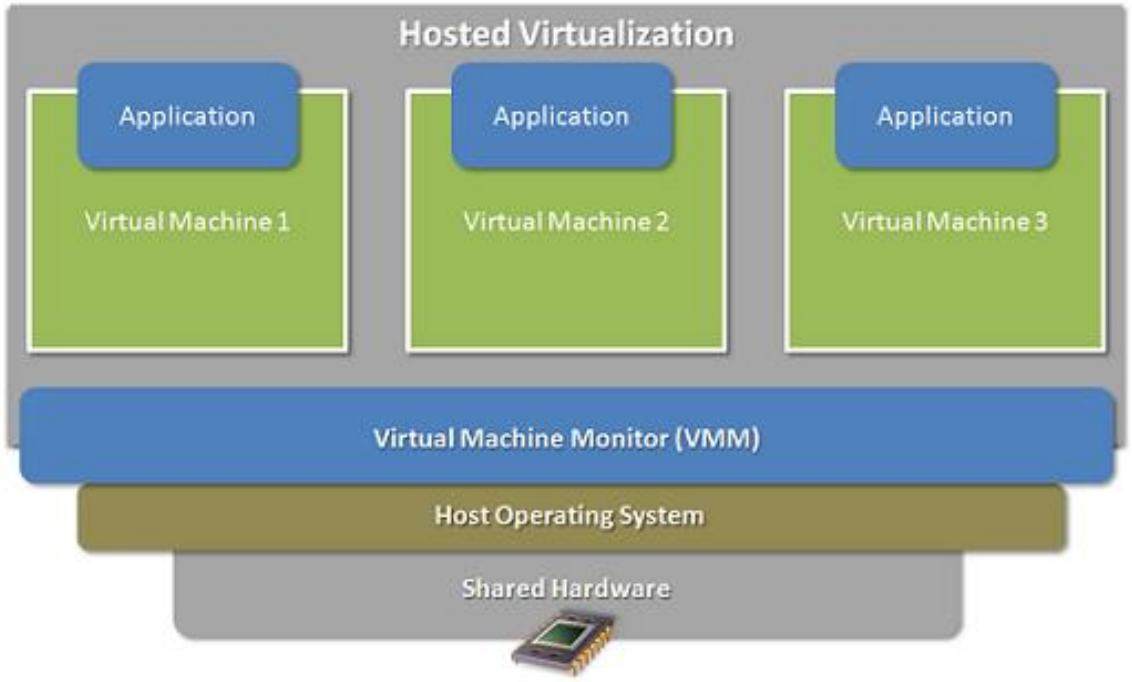
Cloud Service Models

- Software as a Service
  - Provider licenses applications to users as a service
  - E.g., customer relationship management, e-mail, ..
  - Avoid costs of installation, maintenance, patches, …
- Platform as a Service
  - Provider offers platform for building applications
  - E.g., Google's App-Engine, Amazon S3 storage
  - Avoid worrying about scalability of platform

Cloud Service Models

- Infrastructure as a Service
  - Provider offers raw computing, storage, and network
  - E.g., Amazon's Elastic Computing Cloud (EC2)
  - Avoid buying servers and estimating resource needs

Enabling Technology: Virtualization

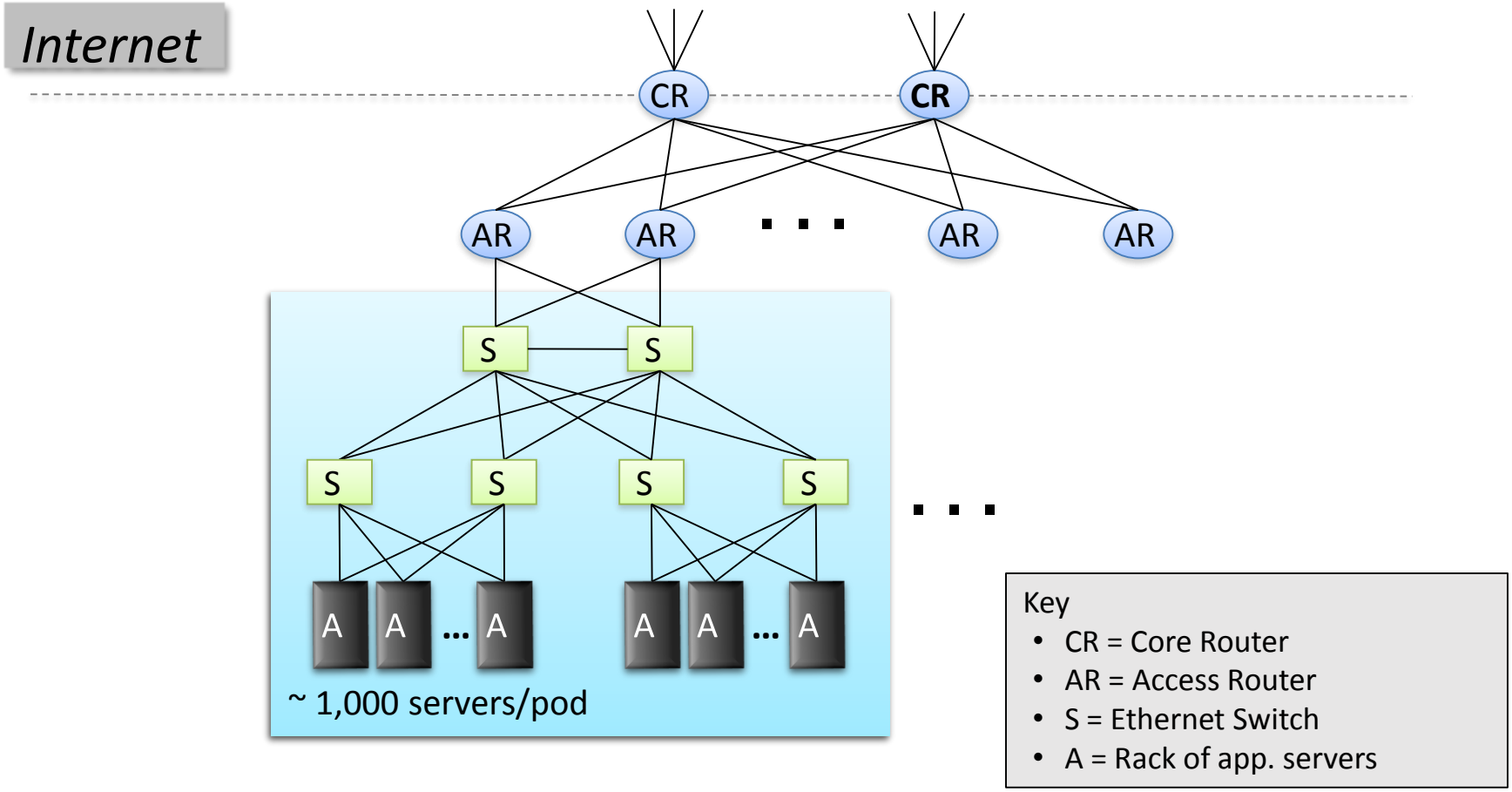


- Multiple virtual machines on one physical machine
- Applications run unmodified as on real machine
- VM can migrate from one computer to another

Multi-Tier Applications

- Applications consist of tasks
  - Many separate components
  - Running on different machines
- Commodity computers
  - Many general-purpose computers
  - Not one big mainframe
  - Easier scaling

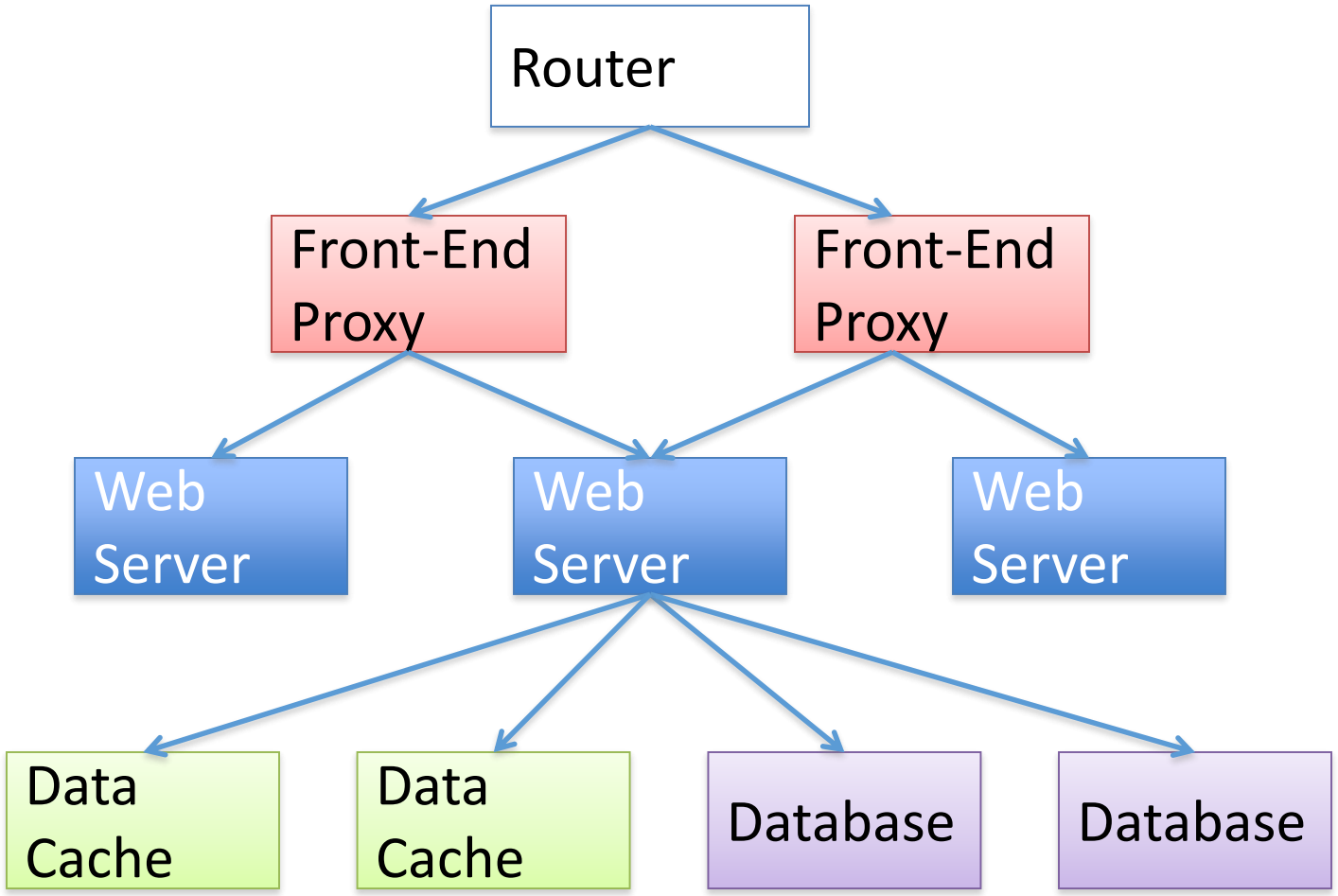
Datacenter Network Topology



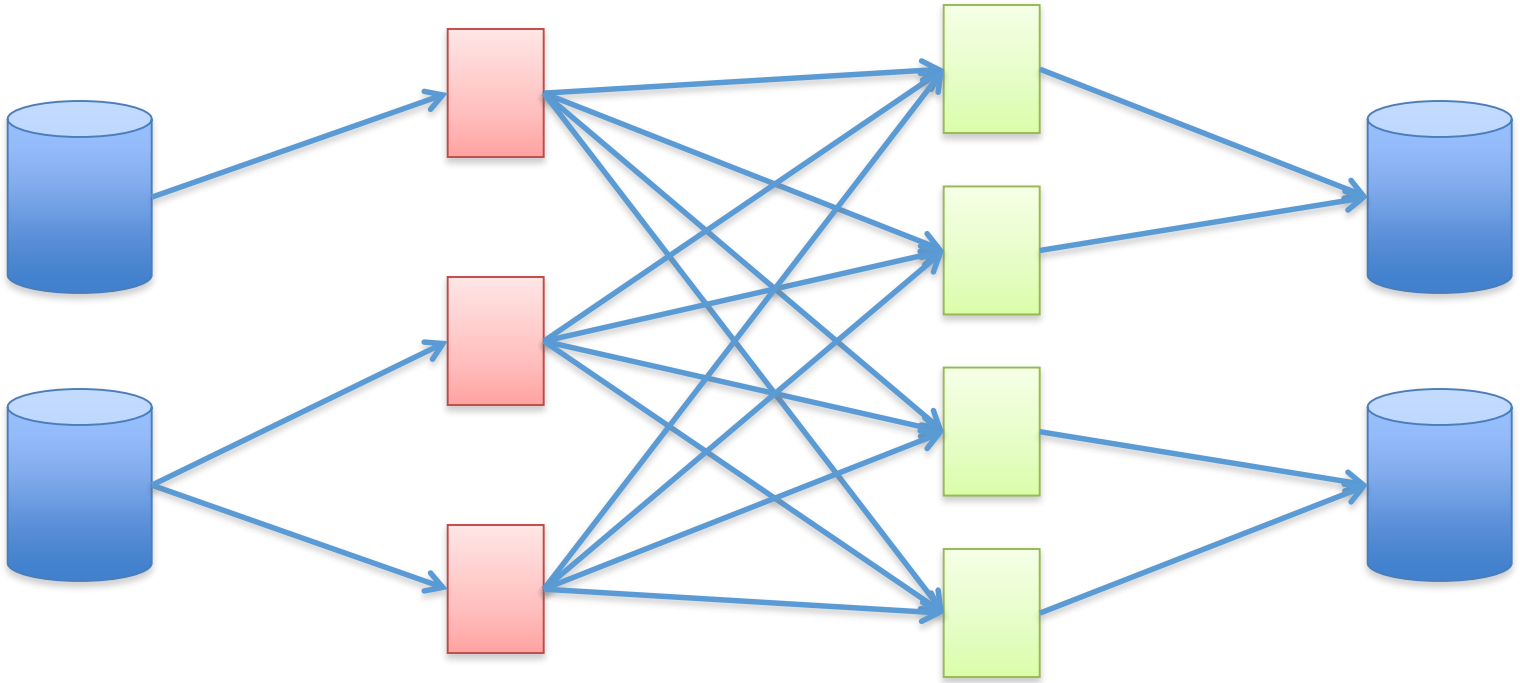
Different types of network traffic in DC

- "North-South traffic"
  - Traffic to/from external clients (outside of datacenter)
  - Handled by front-end (web) servers, mid-tier application servers, and back-end databases
  - Traffic patterns fairly stable, though diurnal variations
- "East-West traffic"
  - Traffic within data-parallel computations within datacenter (e.g. "Partition/Aggregate" programs like Map Reduce)
  - Data in distributed storage, partitions transferred to compute nodes, results joined at aggregation points, stored back into FS
  - Traffic may shift on small timescales (e.g., minutes)

North-South Traffic



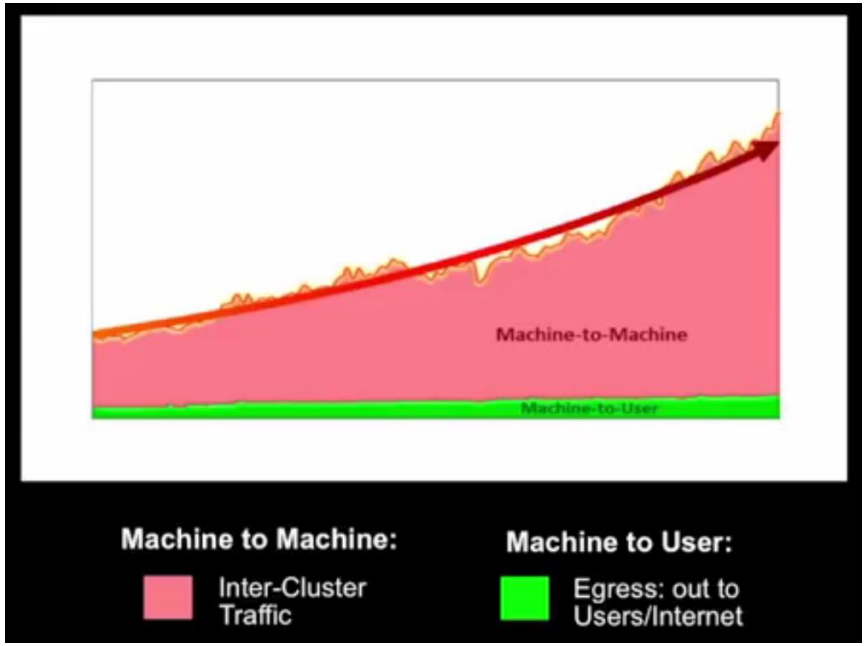
East-West Traffic



Distributed Storage **Map Tasks** **Reduce Tasks** **Distributed Storage**

Measuring Traffic in Today's Data Centers

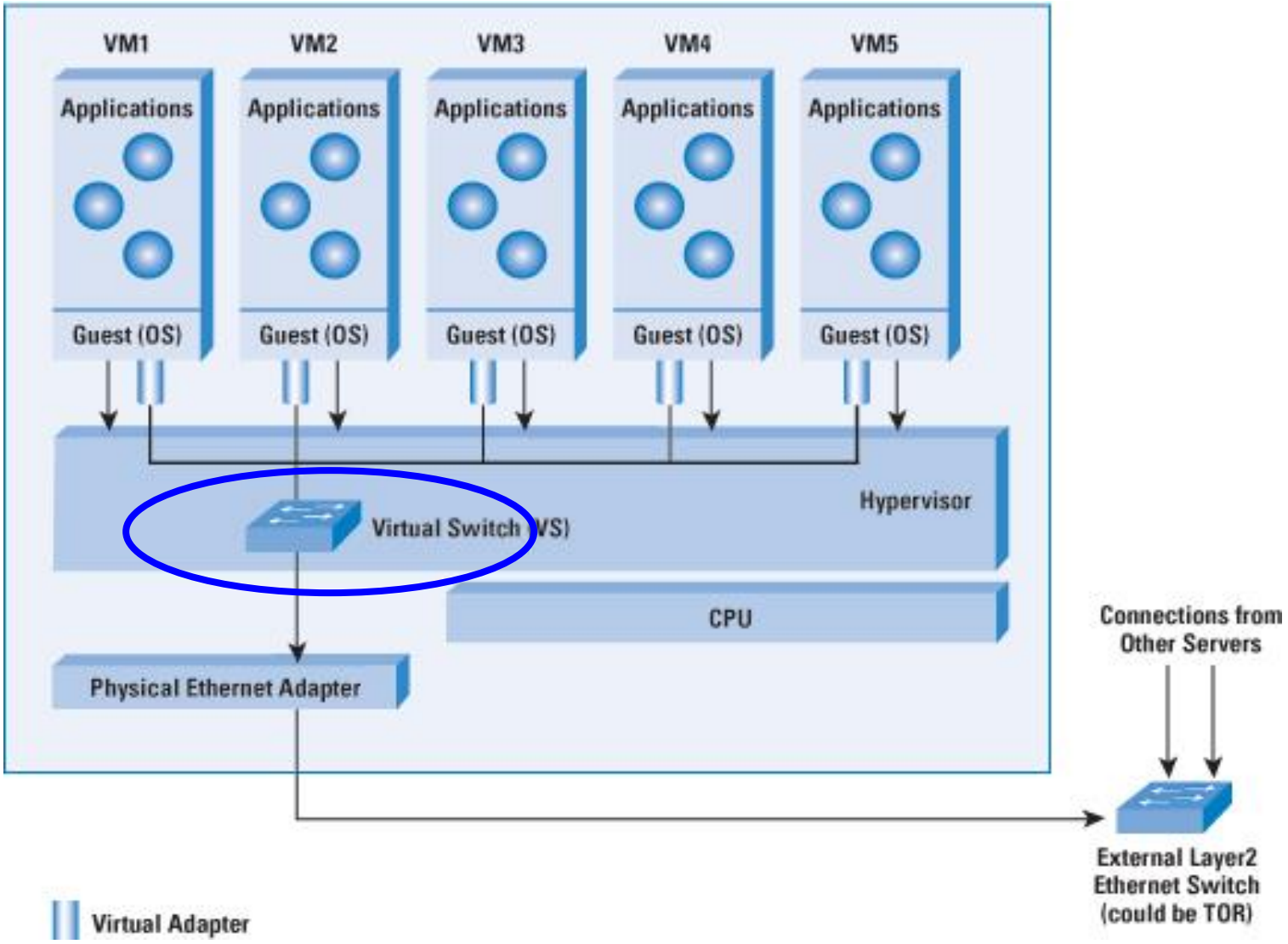
- 80% of the packets stay inside the data center [1]
  - Data mining, index computations, back end to front end



  - Trend is towards even more internal communication [2]

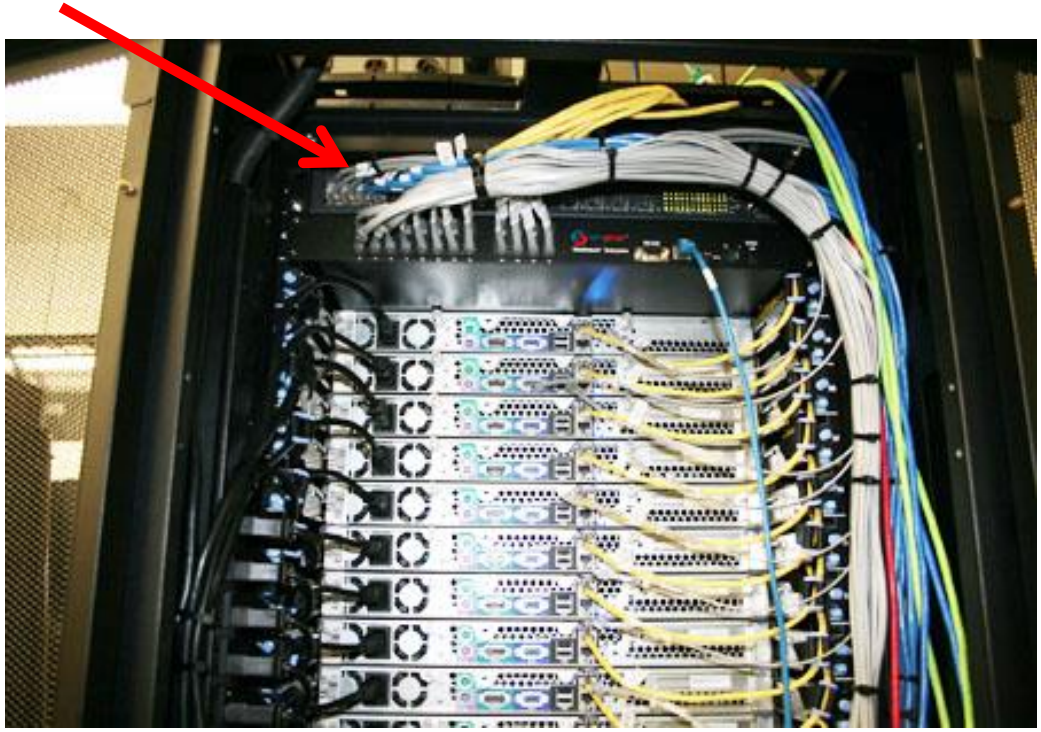
[1] VL2: A Flexible and Scalable Data Center Network. Sigcomm 2009.
Greenberg, Jain, Kandula, Kim, Lahiri, Maltz, Patel, Sengupta.
[2] In Facebook Datacenter: https://www.youtube.com/watch?v=mLEawo6OzFM

Datacenter Networks – Bottom-Up Perspective

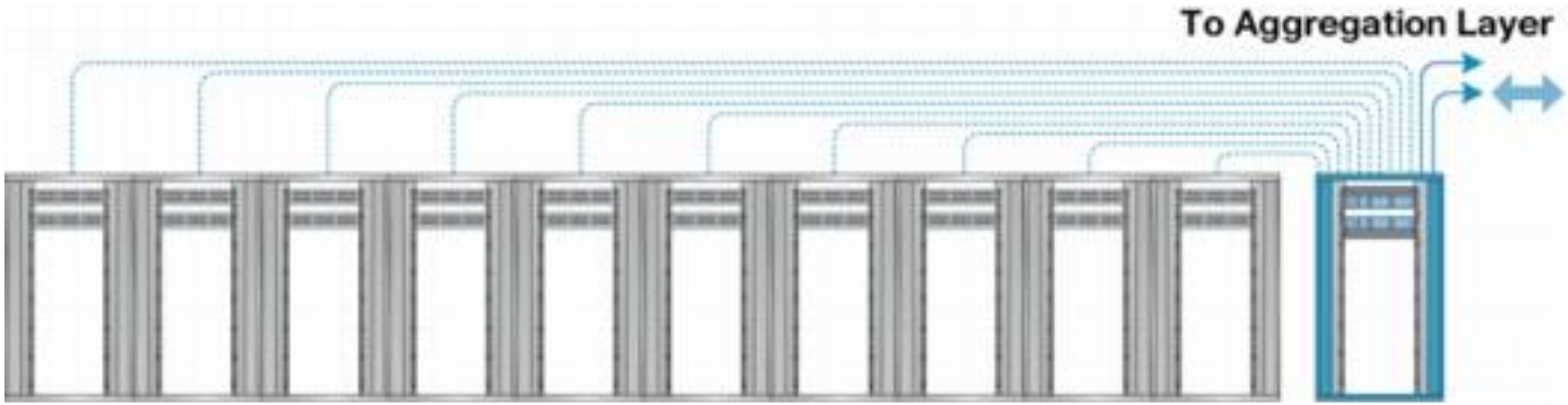


Top-of-Rack Architecture

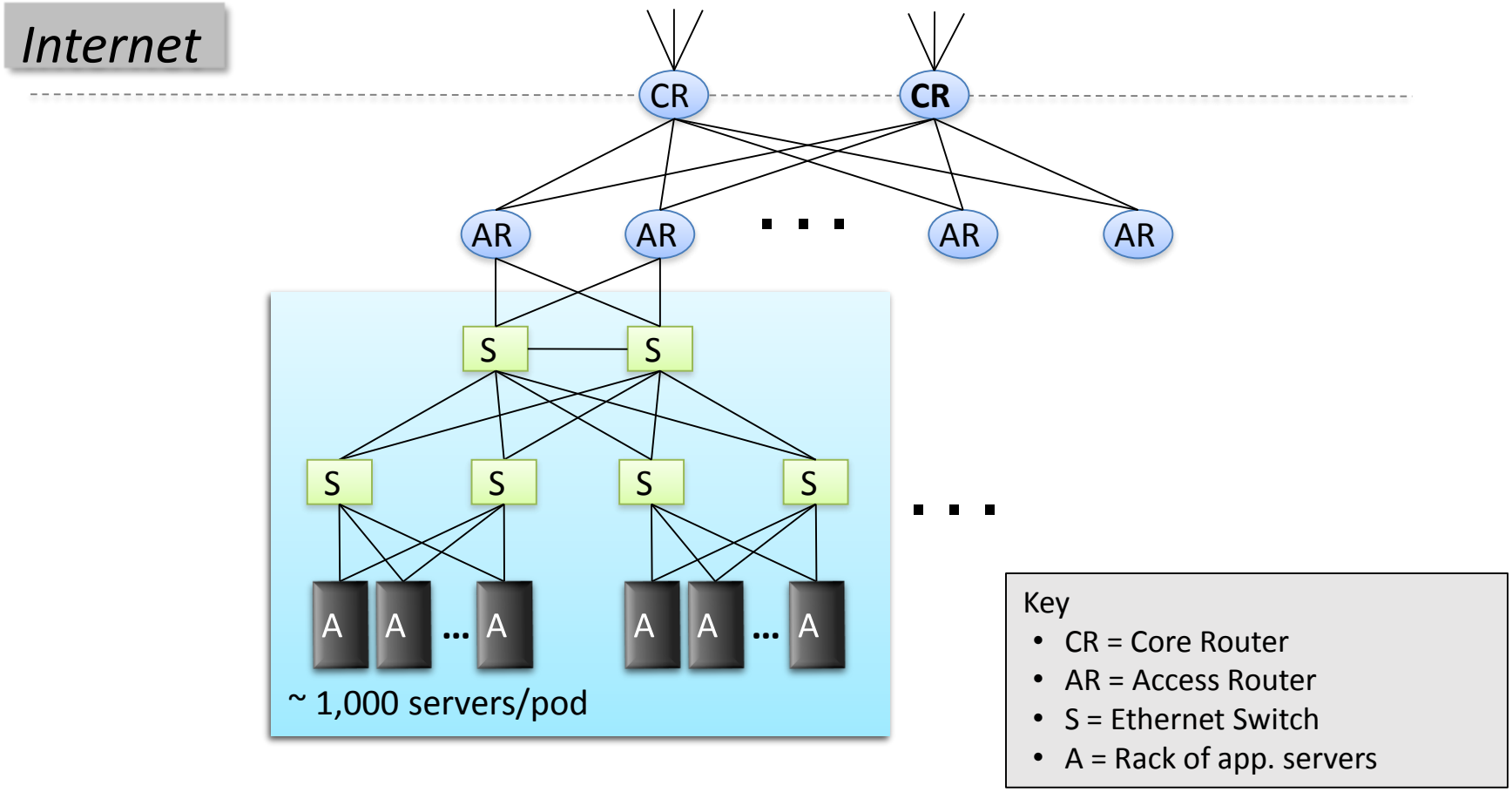
- Rack of servers
  - Commodity servers
  - And top-of-rack switch
- Modular design
  - Preconfigured racks
  - Power, network, and storage cabling



Aggregate to the Next Level



Datacenter Network Topology

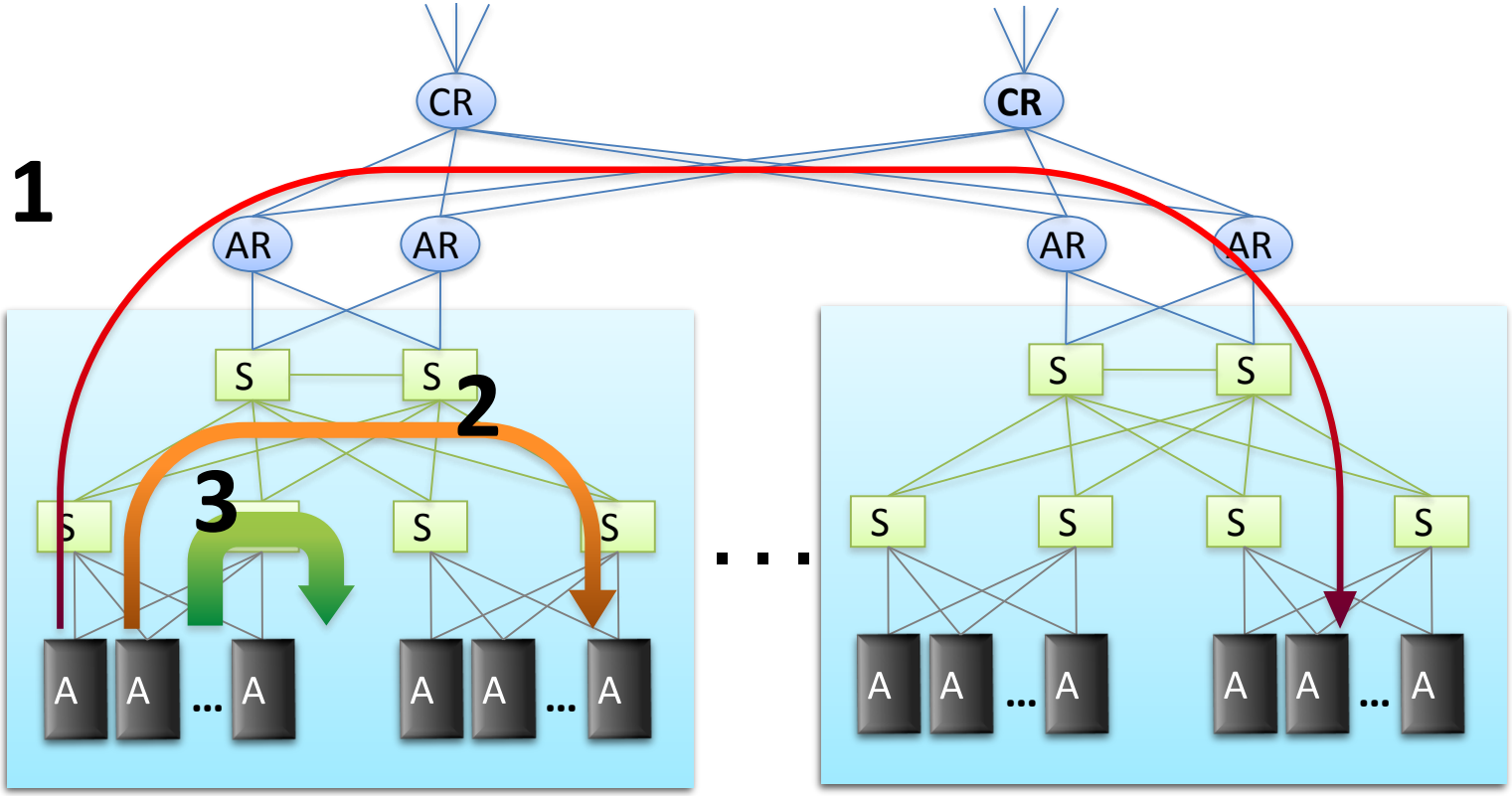


Modularity, Modularity, Modularity

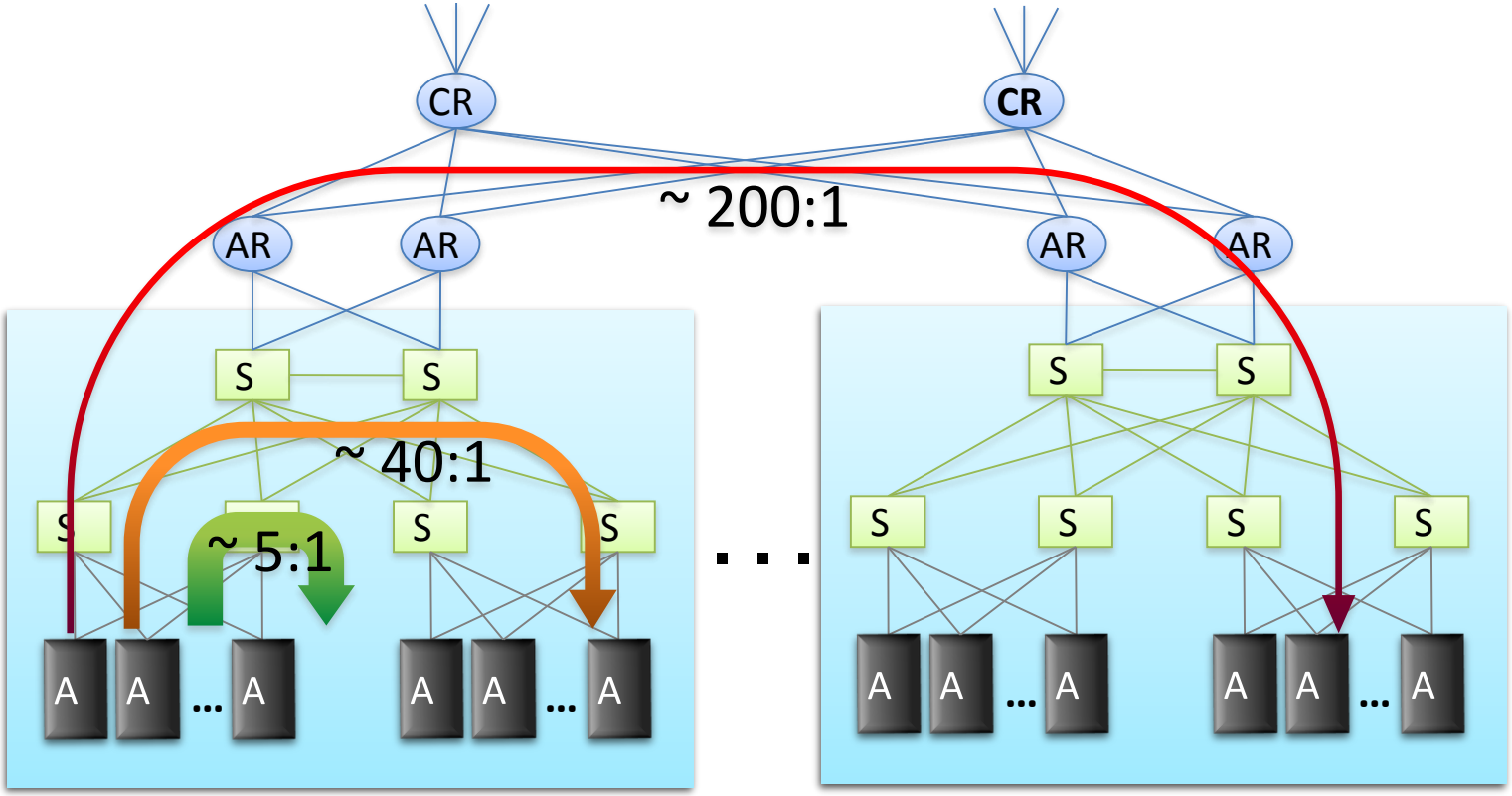
- Containers

- Many containers

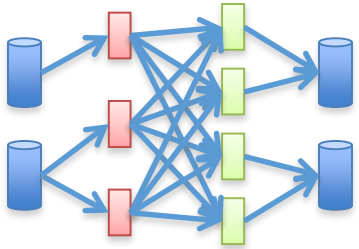
Capacity Mismatch?



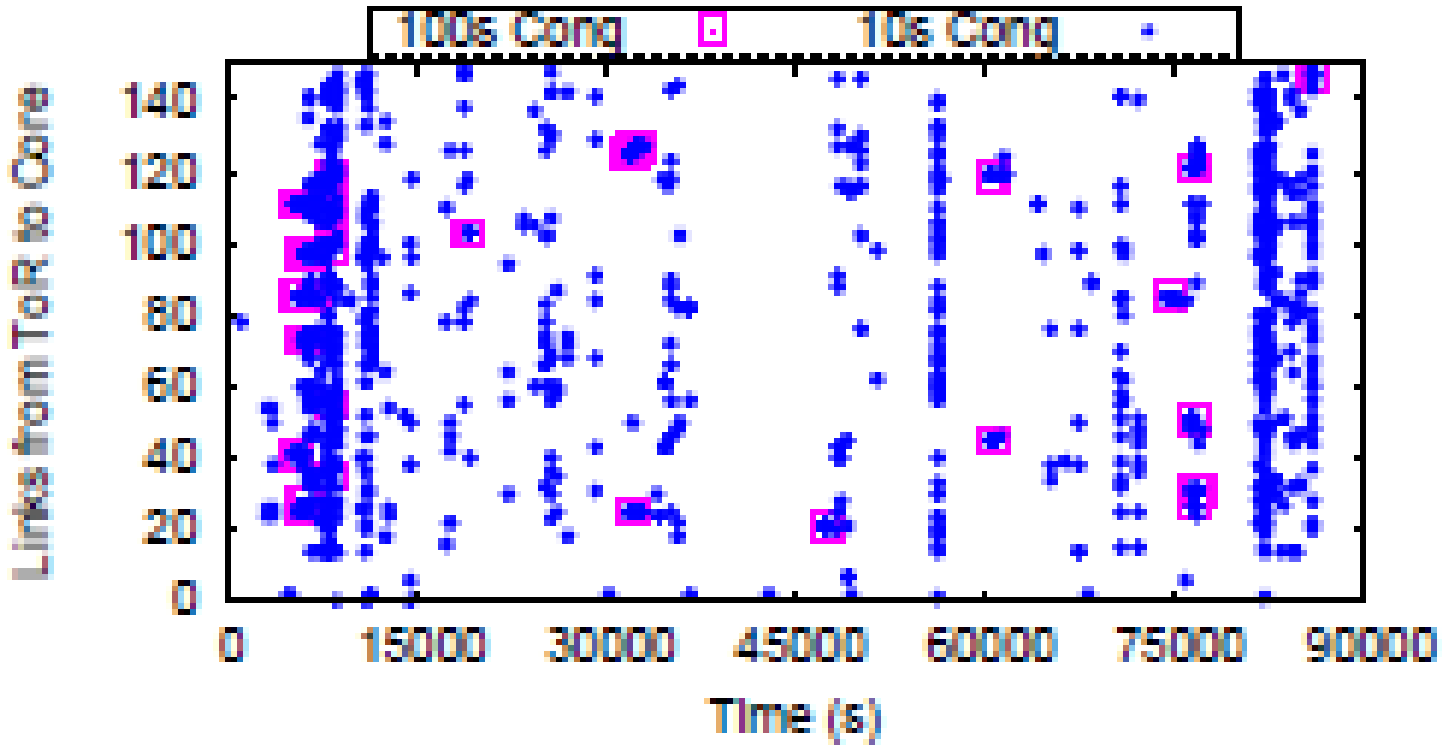
Capacity Mismatch!



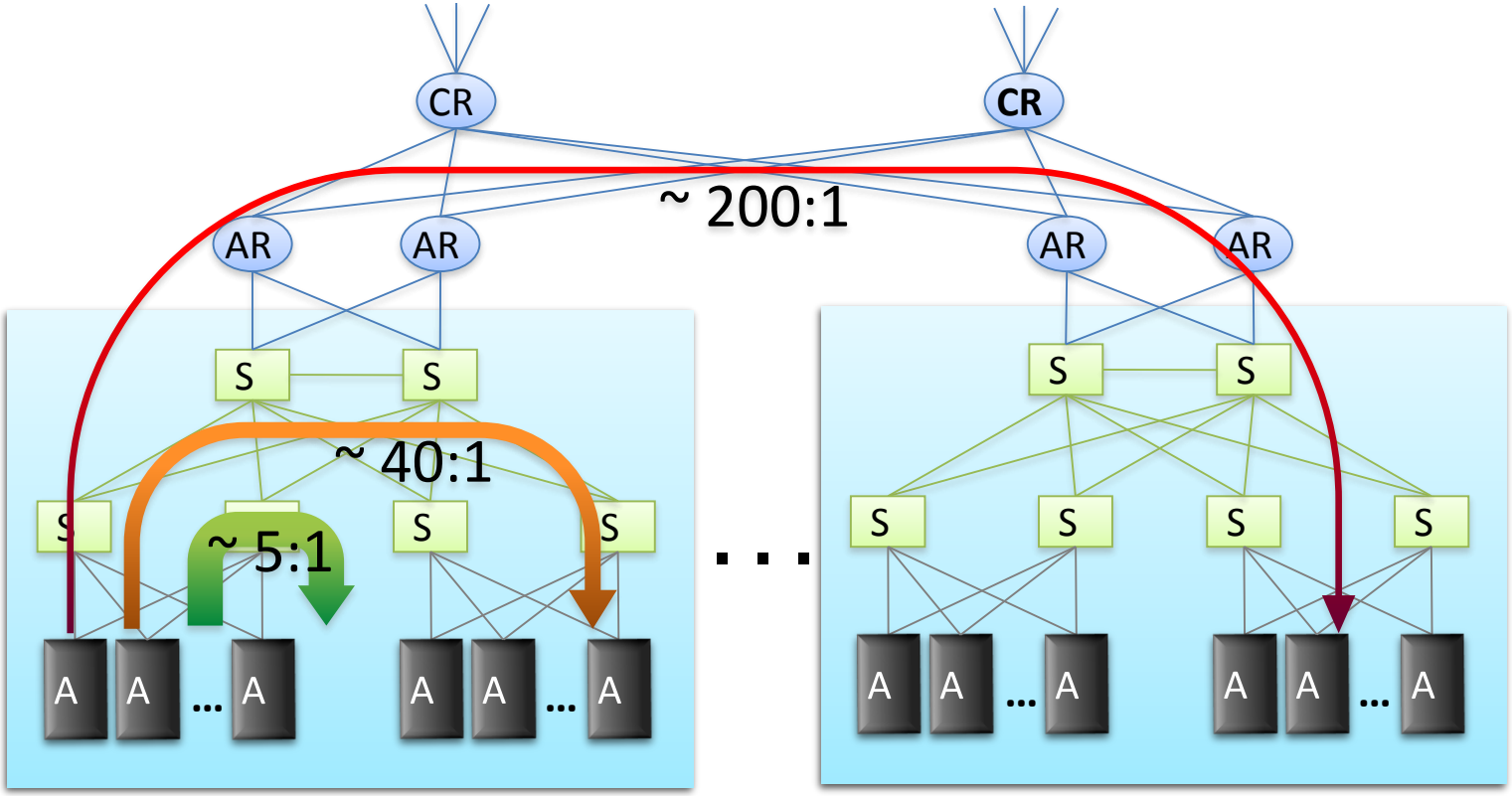
Particularly bad for east-west traffic

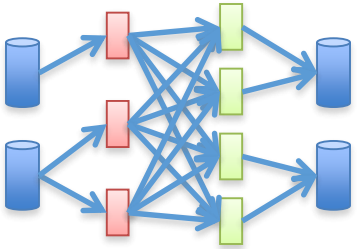


Result: Congestion on Upper Links



Capacity Mismatch!





Is there a better topology for DCN?

Server Costs

- 30% utilization considered “good” in data centers
- Causes include:
- Uneven application fit:
  - Each server has CPU, memory, disk, network: most applications exhaust one resource, stranding the others
- Uncertainty in demand:
  - Demand for a new service can spike quickly

Background

- *Topology:*
  - 2 layers: 5K to 8K hosts
  - 3 layer: >25K hosts
  - Switches:
    - Leaves: have N GigE ports (48-288) + N 10 GigE uplinks to one or more layers of network elements
    - Higher levels: N 10 GigE ports (32-128)

Background Cont.

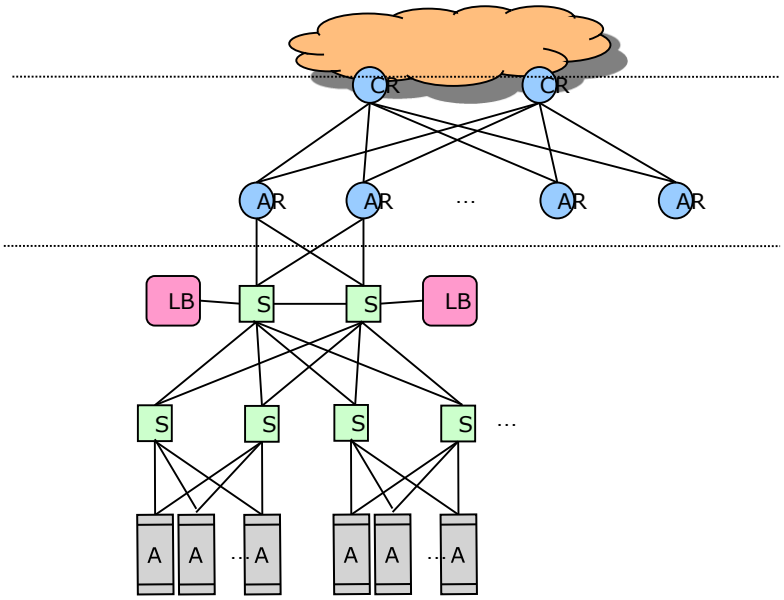
- *Oversubscription:*
  - Ratio of the worst-case achievable aggregate bandwidth among the end hosts to the total bisection bandwidth of a particular communication topology
  - Lower the total cost of the design
  - Typical designs: factor of 2:5:1 (400 Mbps)to 8:1(125 Mbps)
- *Cost:*
  - Edge: $7,000 for each 48-port GigE switch
  - Aggregation and core: $700,000 for 128-port 10GigE switches
  - Cabling costs are not considered!

Problems with common DC topology

- Leverages specialized hardware and communication protocols, such as InfinniBand, Myrinet.
  - These solutions can scale to clusters of thousands of nodes with high bandwidth
  - Expensive infrastructure, incompatible with TCP/IP applications

Problems with common DC topology

- Need very high reliability near top of the tree
  - Very hard to achieve
    - Example: failure of a temporarily unpaired core switch affected ten million users for four hours [1]
- 0.3% of failure events knocked out all members of a network redundancy group
- Single point of failure



Ref: Data Center: Load Balancing Data Center Services, Cisco 2004

[1] VL2: A Flexible and Scalable Data Center Network. Sigcomm 2009. Greenberg, Jain, Kandula, Kim, Lahiri, Maltz, Patel, Sengupta.

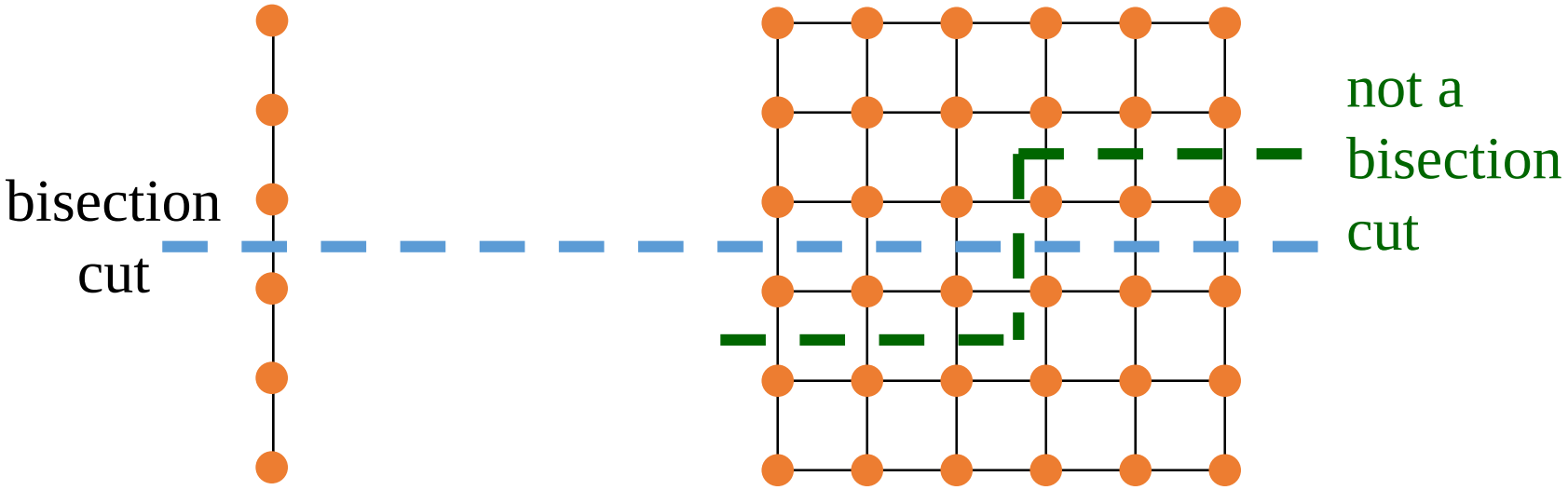
Desired properties of a solution

- Backwards compatible with existing infrastructure
  - No changes in application
  - Support of layer 2 (Ethernet)
- Cost effective
  - Low power consumption & heat emission
  - Cheap infrastructure
- Allows host communication at line speed
- No single point of failure
- A solution: Fat-Tree [1]

[1] Al-Fares et al. "A scalable, commodity data center network architecture." *ACM SIGCOMM Computer Communication Review* 38.4 (2008): 63-74.

Important Metric: Bisection Bandwidth

- **Bisection bandwidth:** bandwidth across smallest cut that divides network into two equal halves
- Bandwidth across "narrowest" part of the network



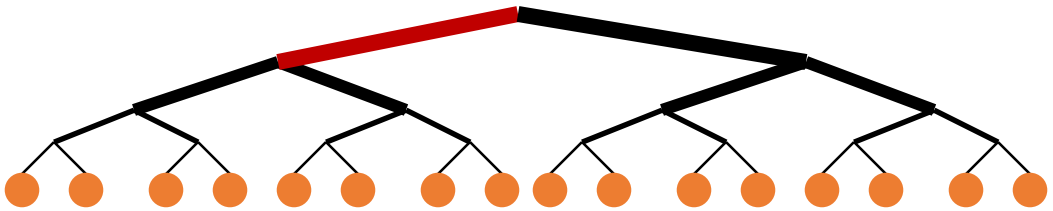
bisection bw= link bw

*bisection bw = sqrt(n) * link bw*

- Bisection bandwidth is important for algorithms in which all processors need to communicate with all others

Typical DC structure: tree

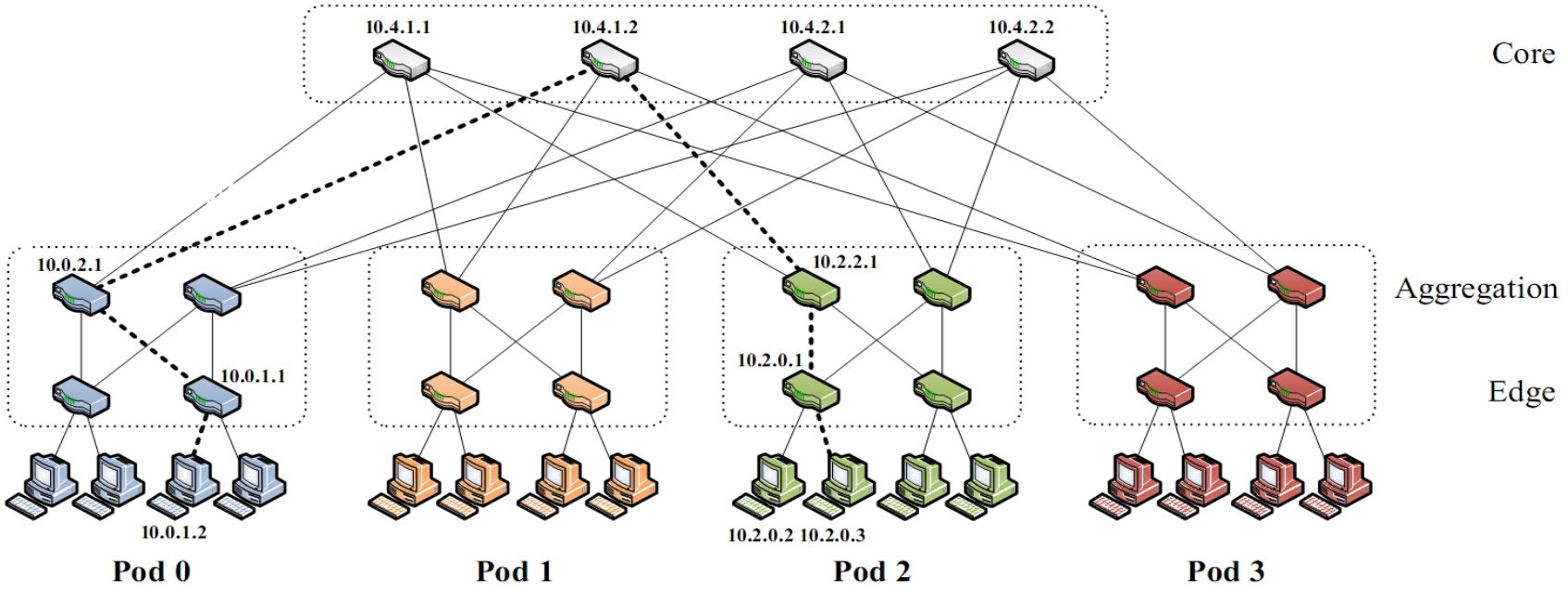
- Problem: Bisection bandwidth = 1.
- Fat trees avoid bisection bandwidth problem:
  - More (or wider) links near top.



Fat-Tree Based DC Architecture

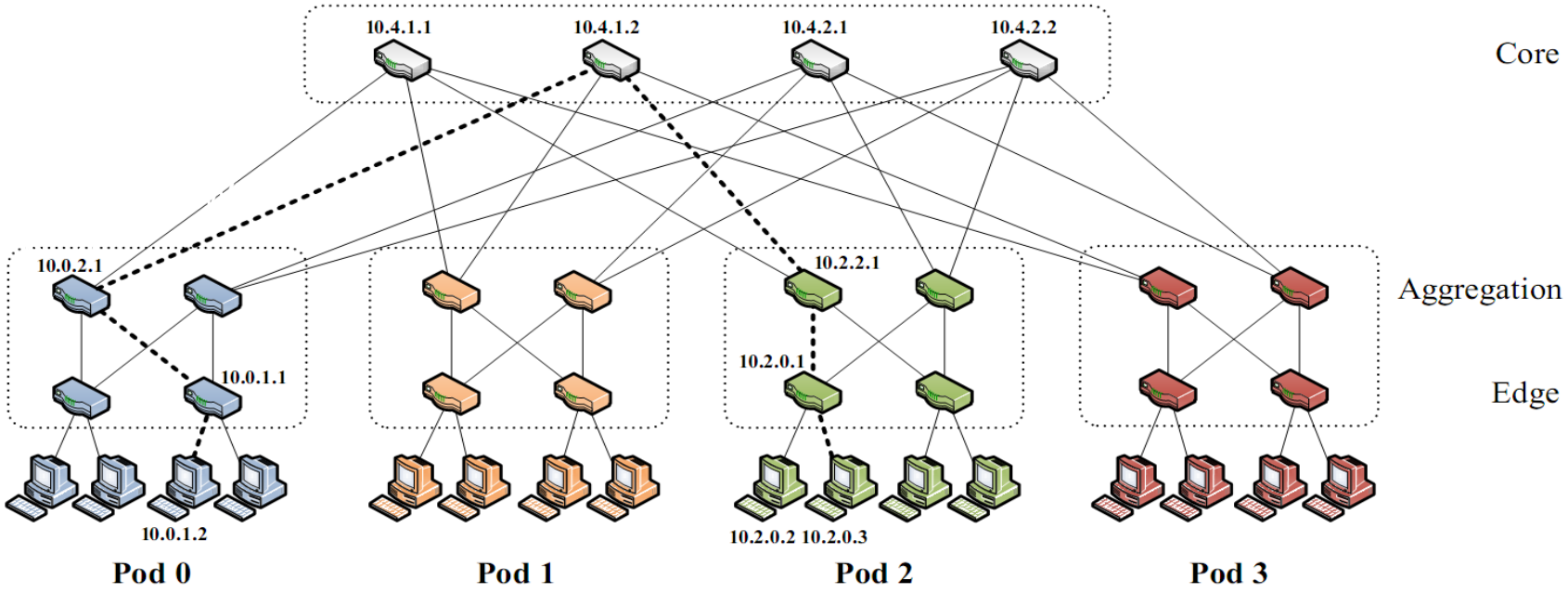
- *Inter-connect racks (of servers) using a fat-tree topology*

  K-ary fat tree: three-layer topology (edge, aggregation and core)
  - each pod consists of $(k/2)^2$ servers & 2 layers of k/2 k-port switches



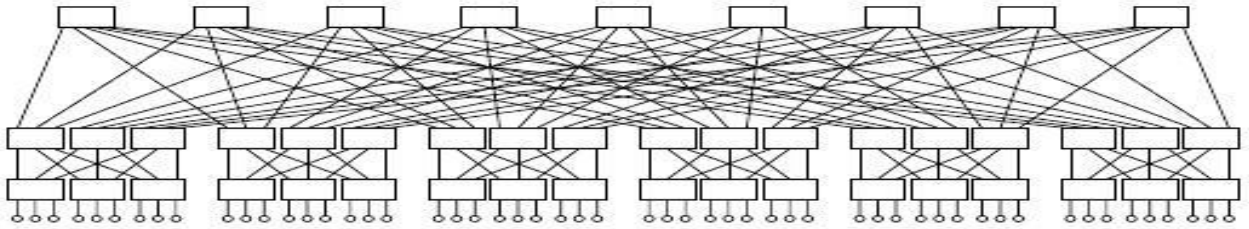
Fat-Tree Based DC Architecture

- *Inter-connect racks (of servers) using a fat-tree topology*
  - each edge (lower) switch connects to k/2 servers & k/2 aggr. switches
  - each aggr. (upper) switch connects to k/2 edge & k/2 core switches
  - $(k/2)^2$ core switches: each connects to k pods



Fat-Tree Based Topology

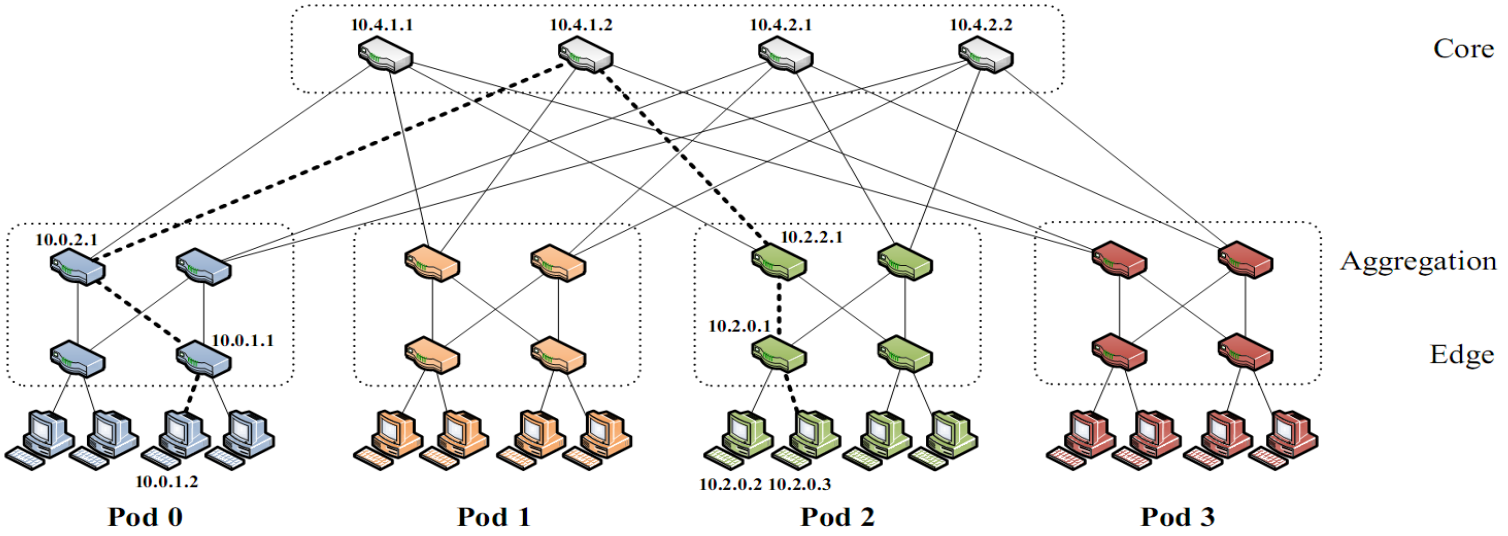
- **Why Fat-Tree?**
  - Fat tree has identical bandwidth at any bisections
  - Each layer has the same aggregated bandwidth
- Can be built using cheap devices with uniform capacity
  - Each port supports same speed as end host
  - All devices can transmit at line speed if packets are distributed uniform along available paths
- Great scalability: k-port switch supports $k^3/4$ servers



Fat tree network with K = 6 supporting 54 hosts

Problems with Fat-tree

- Layer 3 will only use one of the existing equal cost paths
  - Bottlenecks up and down the fat-tree
- Packet re-ordering occurs if layer 3 blindly takes advantage of path diversity;
- Load may not necessarily be well-balanced



FAT-Tree Modified

- Enforce a special (IP) addressing scheme in DC
  - unused.PodNumber.switchnumber.Endhost
  - Allows host attached to same switch to route only through switch
  - Allows inter-pod traffic to stay within pod

FAT-Tree Modified

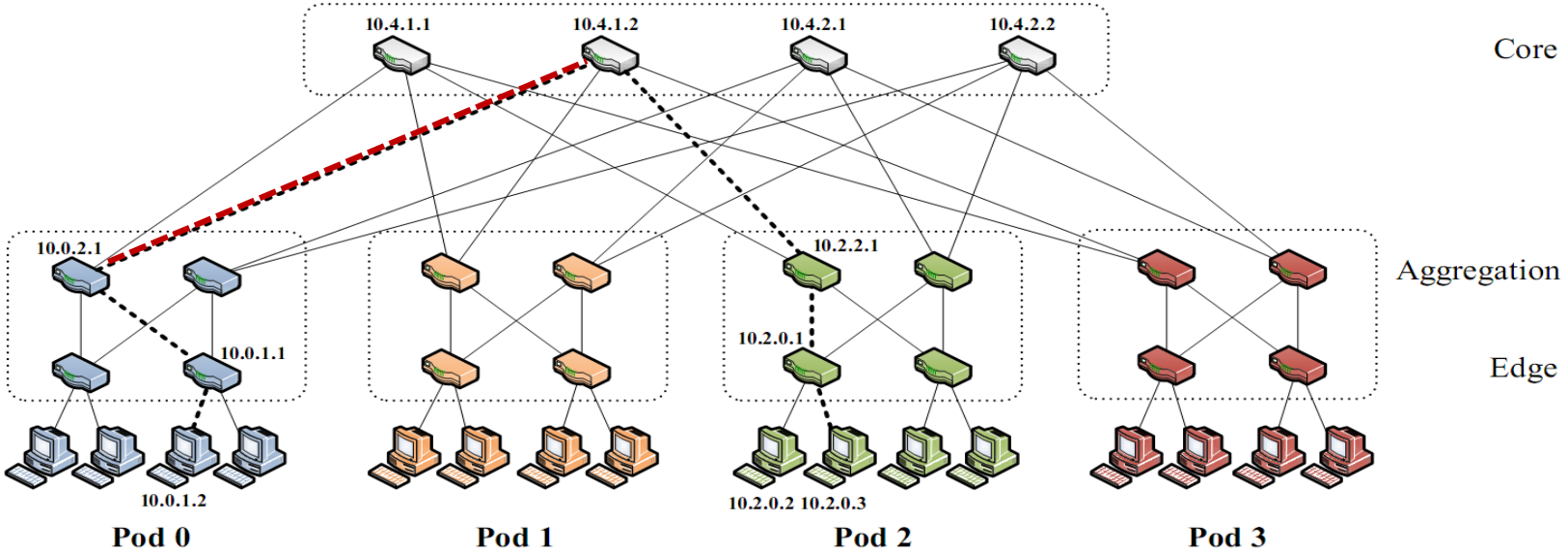
- Use two level look-ups to distribute traffic and maintain packet ordering
  - First level is prefix lookup
    - used to route down the topology to servers
  - Second level is a suffix lookup
    - used to route up towards core
    - maintain packet ordering by using same ports for same server
    - Diffuses and spreads out traffic

Prefix	Output port
10.2.0.0/24	0
10.2.1.0/24	1
0.0.0.0/0	

Suffix	Output port
0.0.0.2/8	2
0.0.0.3/8	3

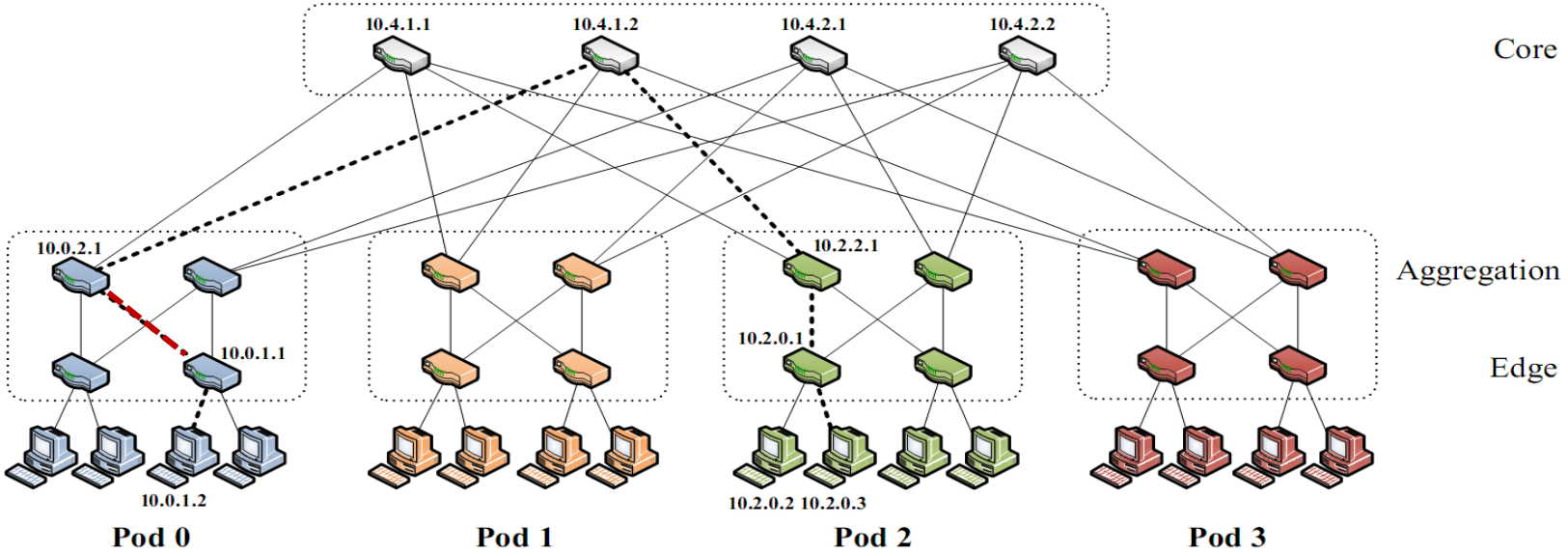
Fault-Tolerance

- Failure b/w upper layer and core switches
  - **Outgoing inter-pod traffic:** local routing table marks the affected link as unavailable and chooses another core switch
  - **Incoming inter-pod traffic:** core switch broadcasts a tag to upper switches directly connected signifying its inability to carry traffic to that entire pod, then upper switches avoid that core switch when assigning flows destined to that pod



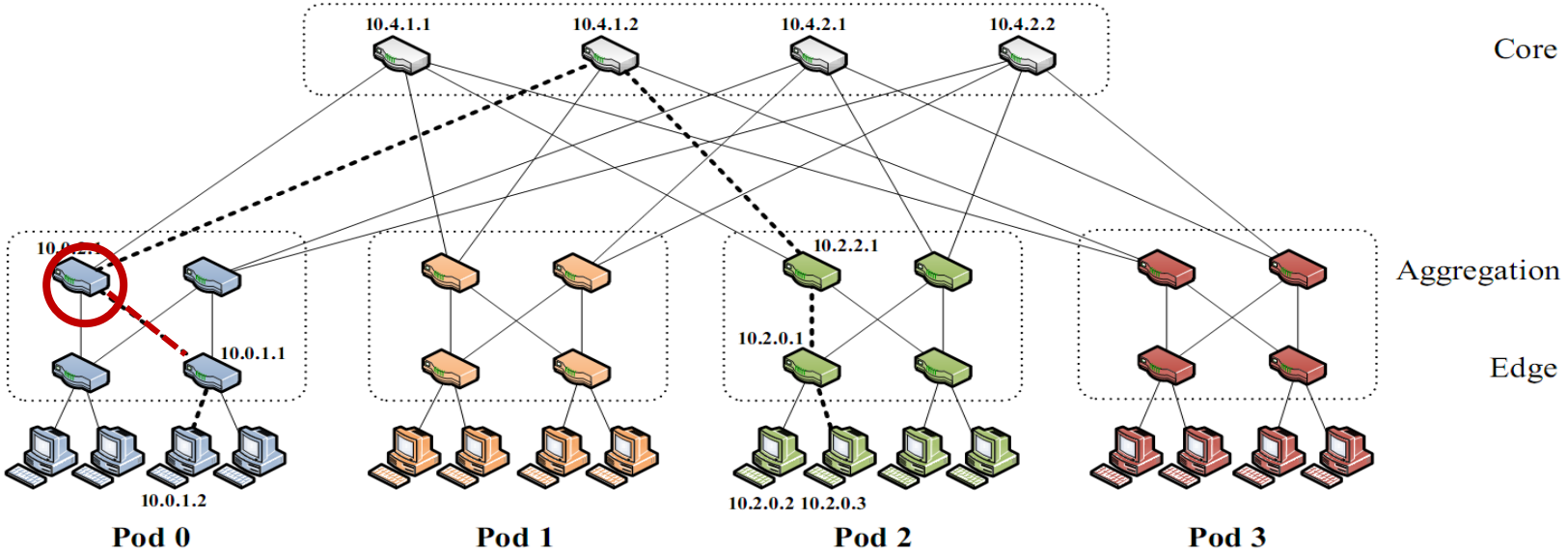
Fault-Tolerance

- Failure b/w lower and upper layer switches
  - **Outgoing inter- and intra pod traffic from lower-layer:**
    - the local flow classifier sets the cost to infinity and does not assign it any new flows, chooses another upper layer switch



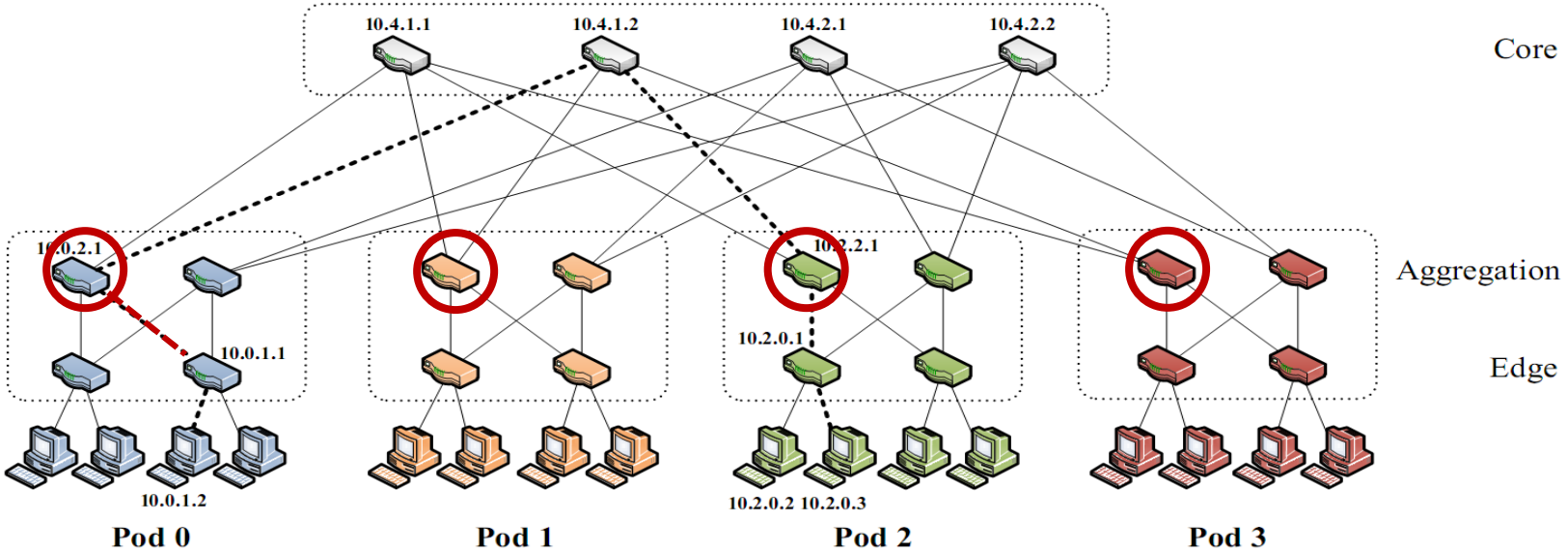
Fault-Tolerance

- Failure b/w lower and upper layer switches
  - **Intra-pod traffic using upper layer switch as intermediary:**
    - Switch broadcasts a tag notifying all lower level switches, these would check when assigning new flows and avoid it



Fault-Tolerance

- Failure b/w lower and upper layer switches
  - **Inter-pod traffic coming into upper layer switch:**
    - Tag to all its core switches signifying its inability to carry traffic, core switches mirror this tag to all upper layer switches, then upper switches avoid affected core switch when assigning new flows



Packing

- Increased wiring overhead is inherent to the fat-tree topology
- Minimize total cable length by placing racks around the pod switch in two dimensions

Packing

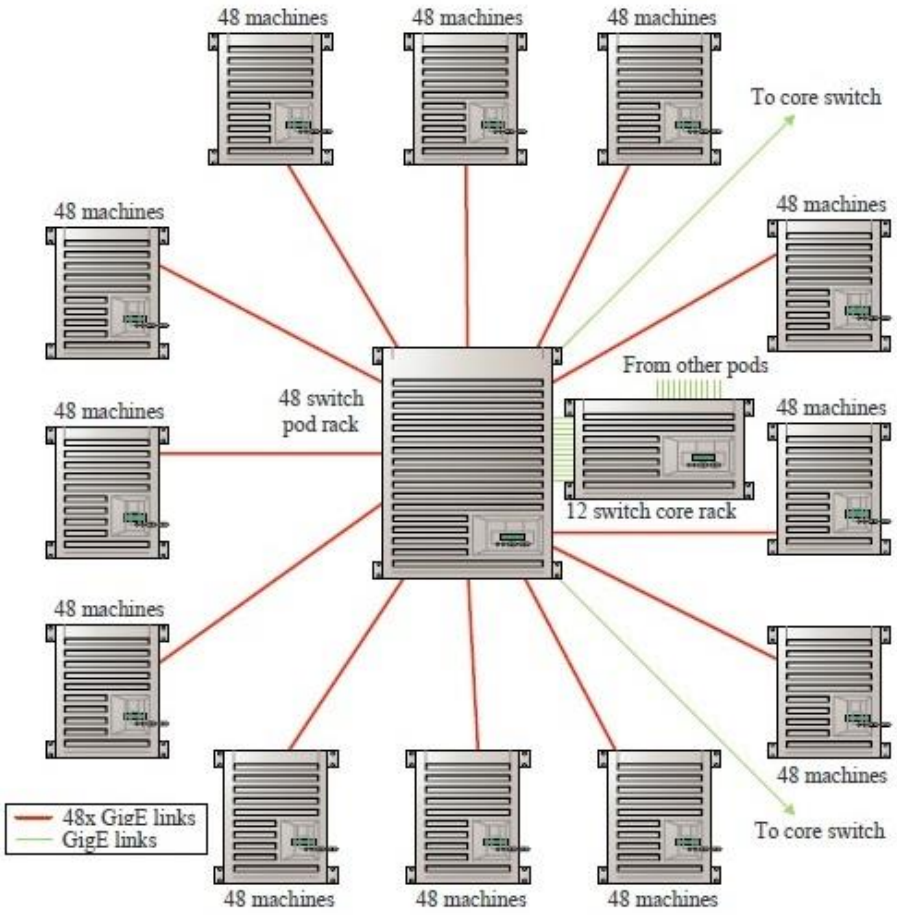


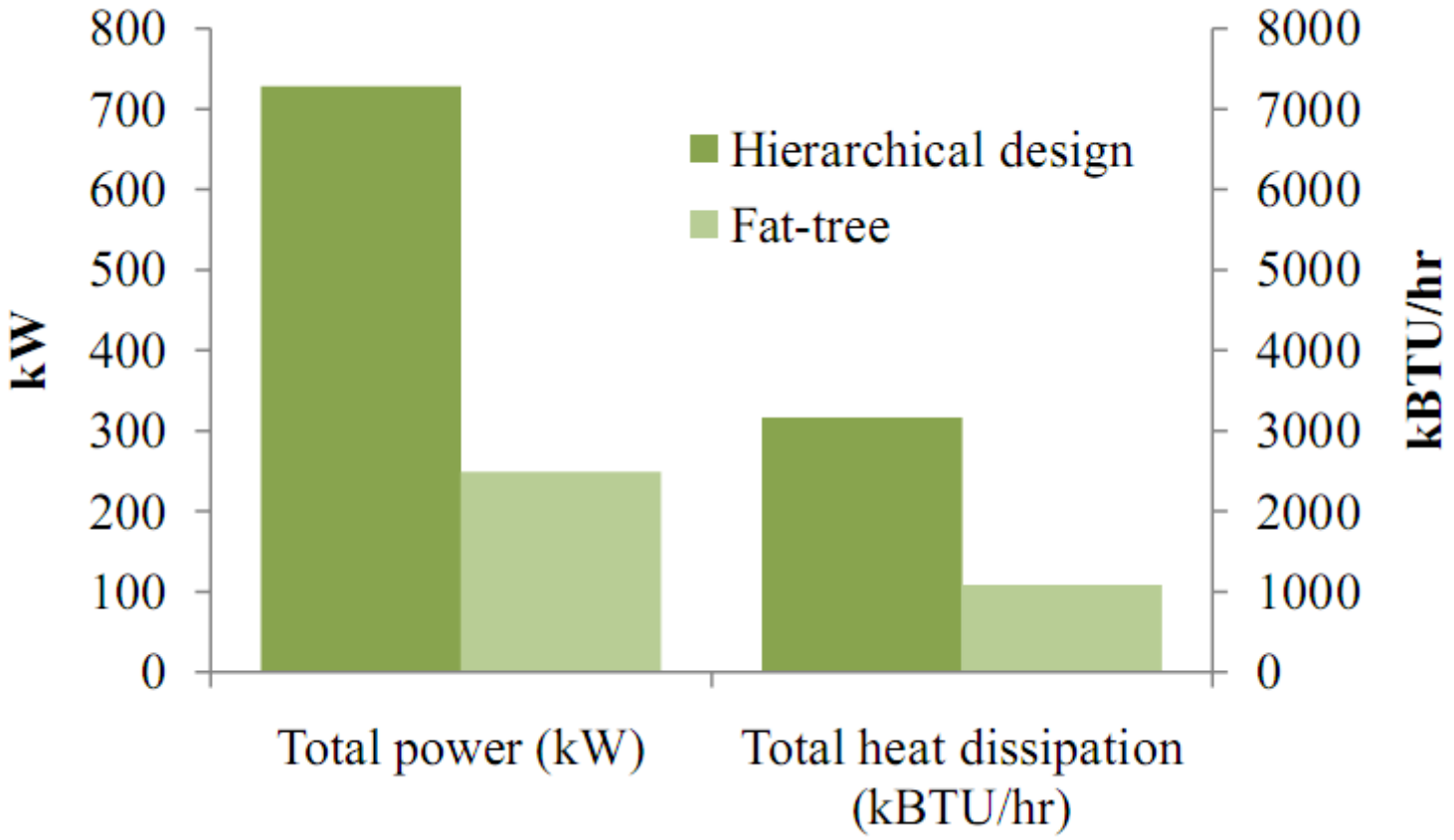
Figure 8: Proposed packaging solution. The only external cables are between the pods and the core nodes.

Results: Network Utilization

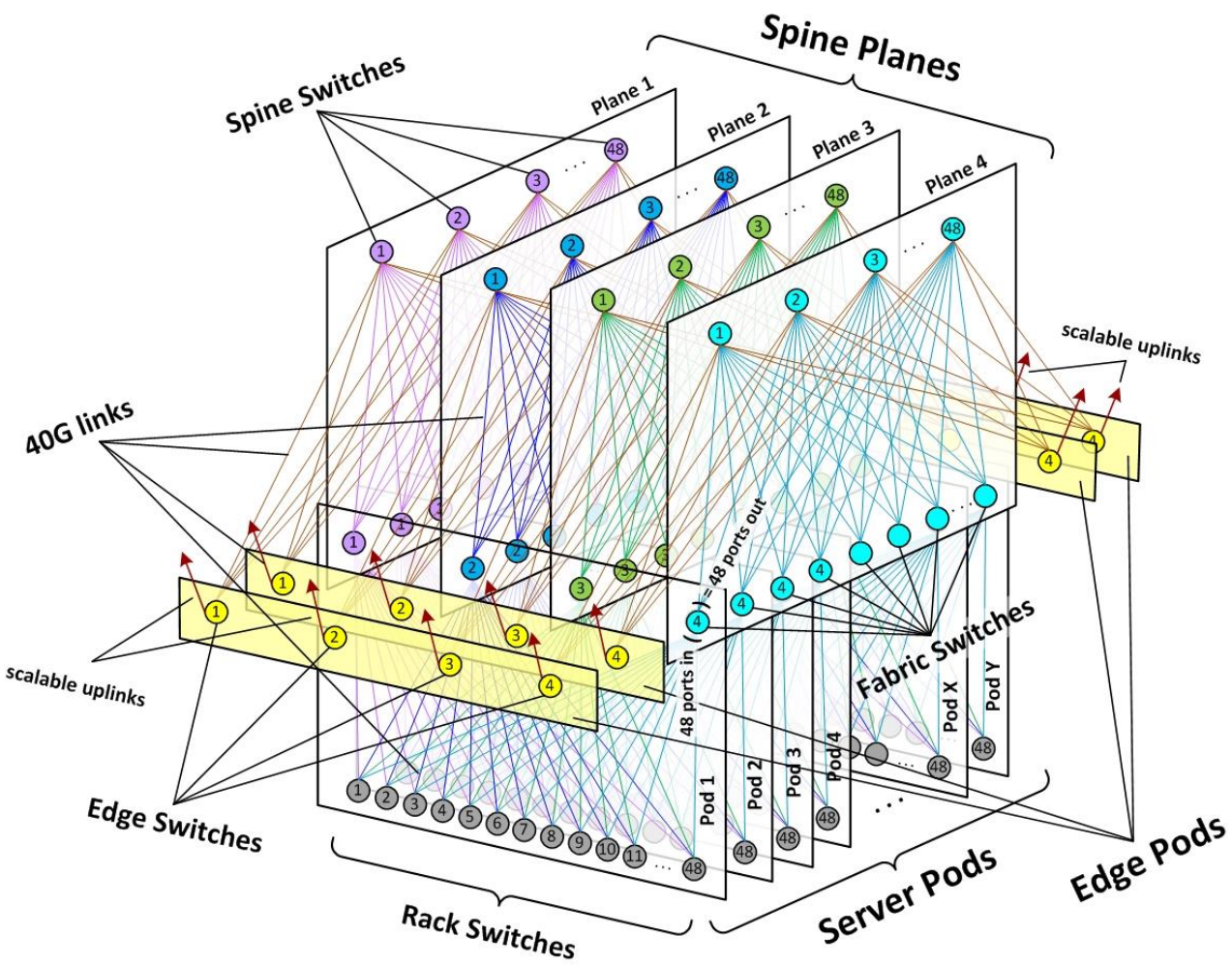
Test	Tree	Two-Level Table	Flow Classification	Flow Scheduling
Random	53.4%	75.0%	76.3%	93.5%
Stride (1)	100.0%	100.0%	100.0%	100.0%
Stride (2)	78.1%	100.0%	100.0%	99.5%
Stride (4)	27.9%	100.0%	100.0%	100.0%
Stride (8)	28.0%	100.0%	100.0%	99.9%
Staggered Prob (1.0, 0.0)	100.0%	100.0%	100.0%	100.0%
Staggered Prob (0.5, 0.3)	83.6%	82.0%	86.2%	93.4%
Staggered Prob (0.2, 0.3)	64.9%	75.6%	80.2%	88.5%
Worst cases:				
Inter-pod Incoming	28.0%	50.6%	75.1%	99.9%
Same-ID Outgoing	27.8%	38.5%	75.4%	87.4%

Table 2: Aggregate Bandwidth of the network, as a percentage of ideal bisection bandwidth for the Tree, Two-Level Table, Flow Classification, and Flow Scheduling methods. The ideal bisection bandwidth for the fat-tree network is 1.536Gbps.

Results: Heat & Power Consumption

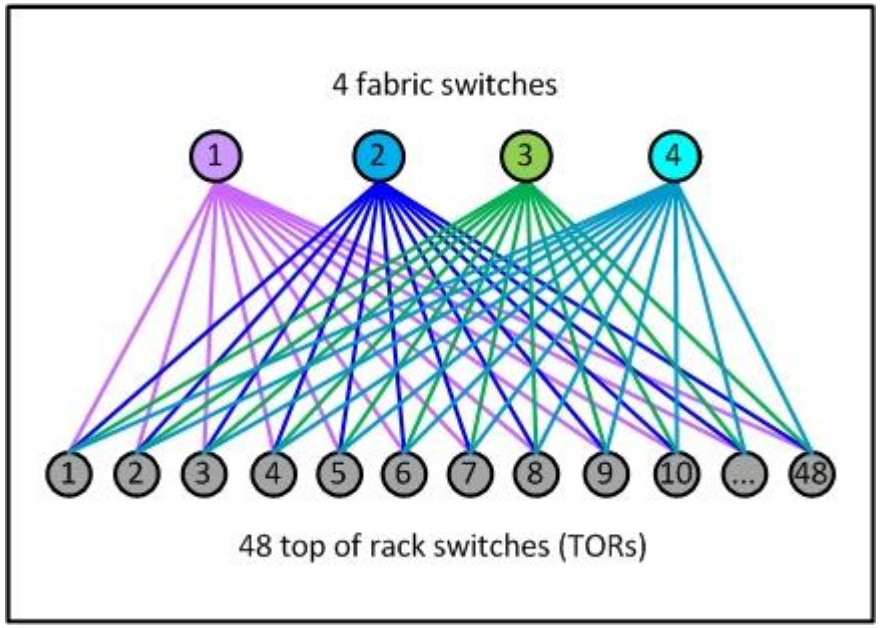


Another Look: Facebook [1]



[1] Roy et al. "**The Many Faces of Facebook's Datacenter Network** " (probably?) *Proceedings of the 2015 ACM conference on SIGCOMM*. ACM, 2015.

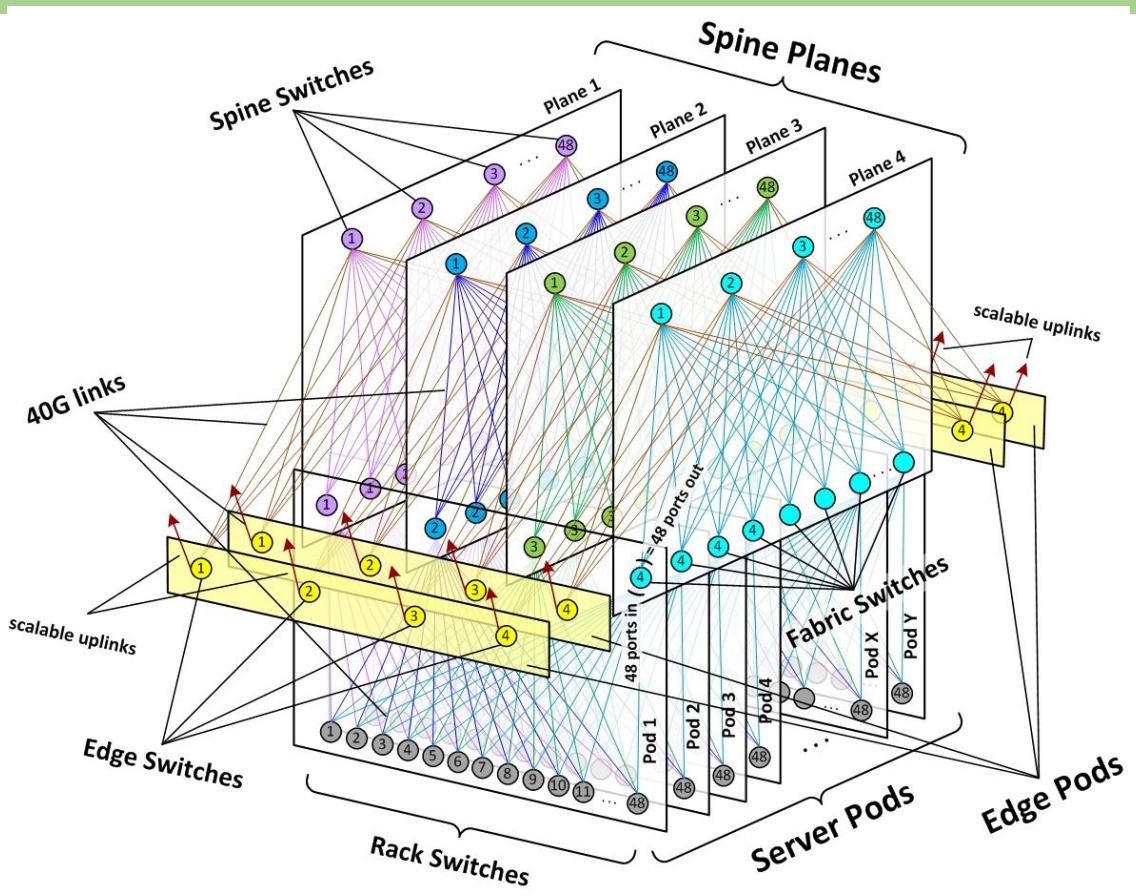
Facebook's new datacenter



- ToR with 4 40G ports: each connected to one fabric switch
- **For each ToR – FS link, same amount of bandwidth reserved outgoing from FS**

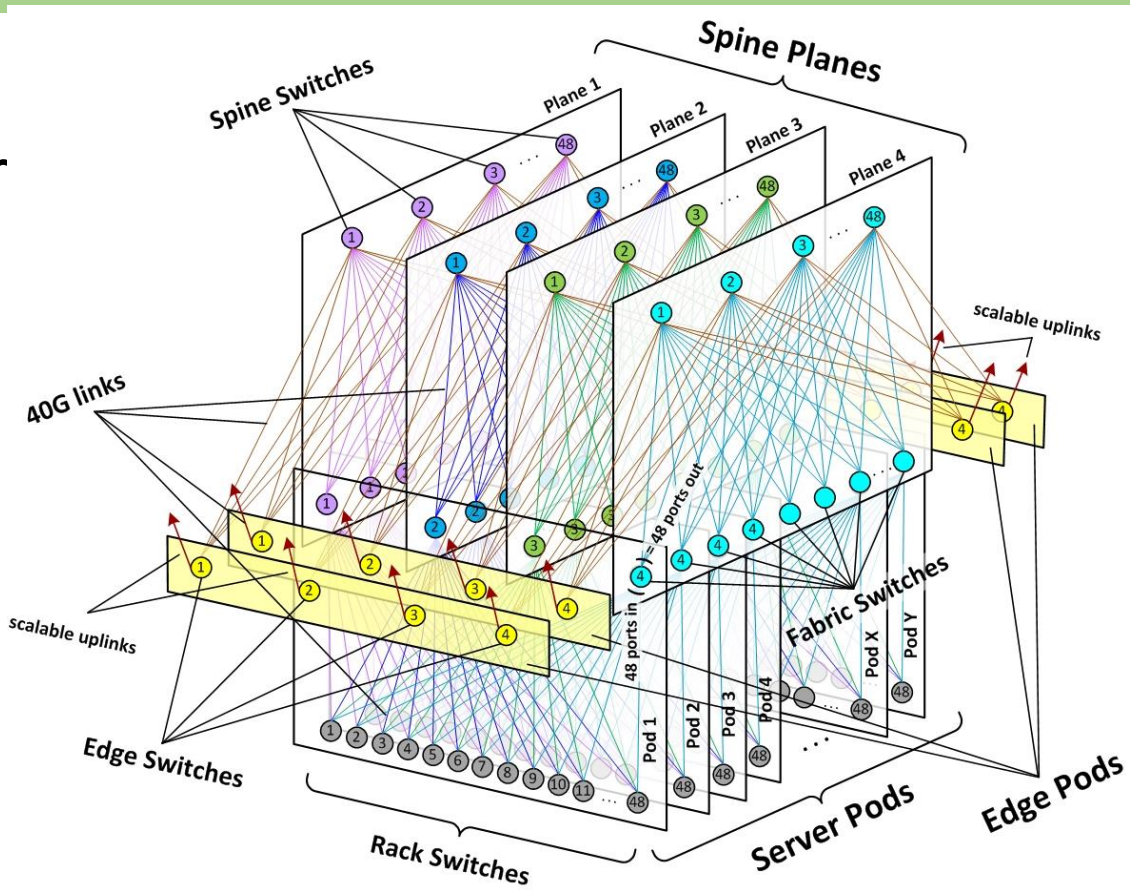
Facebook's new datacenter

- Four spine planes
- Each accommodates 48 spine switches (40G links)
- Spine planes are interconnected
- Multi-petabit bisection bandwidth
- Possible to achieve zero oversubscription

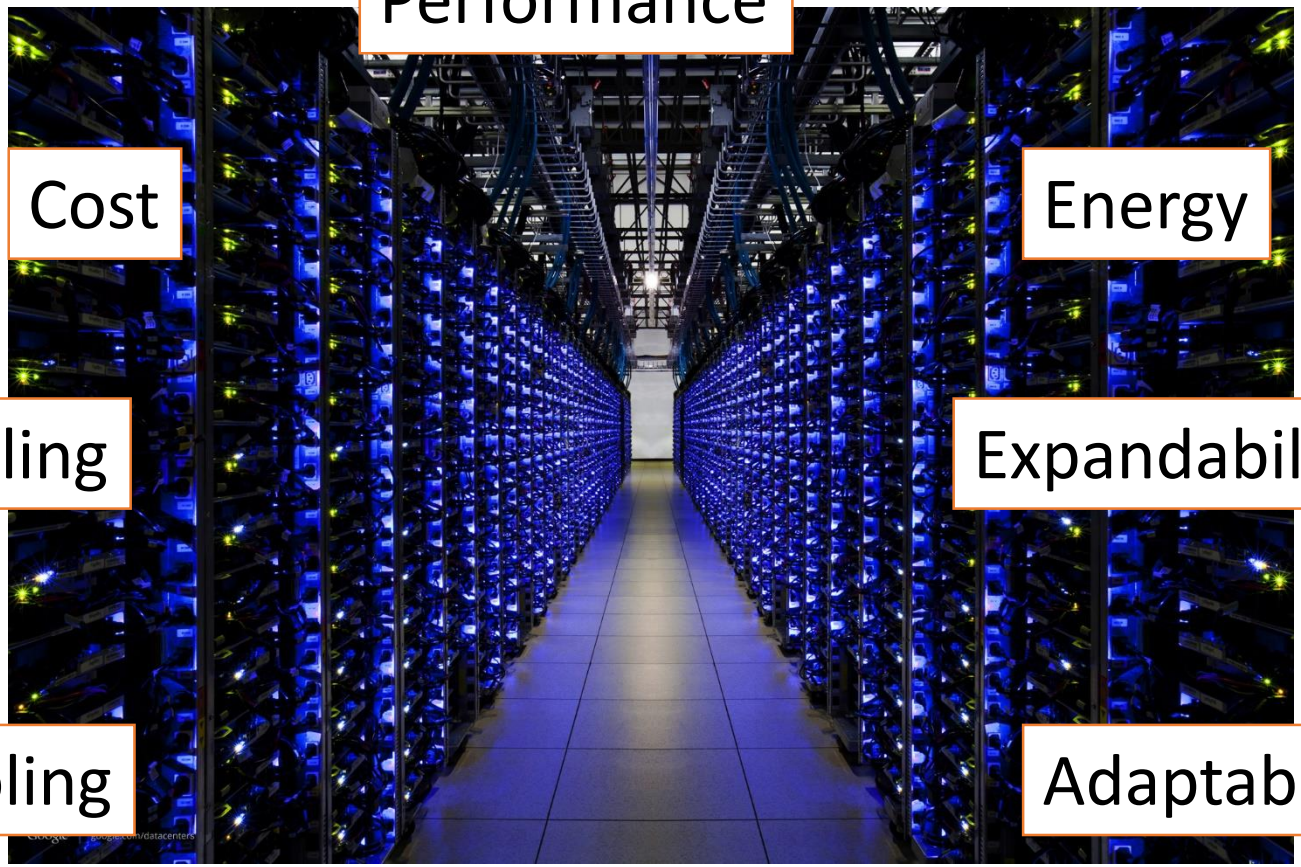


Facebook's new datacenter

- Protocol BGP
- Centralized controller to override BGP decisions

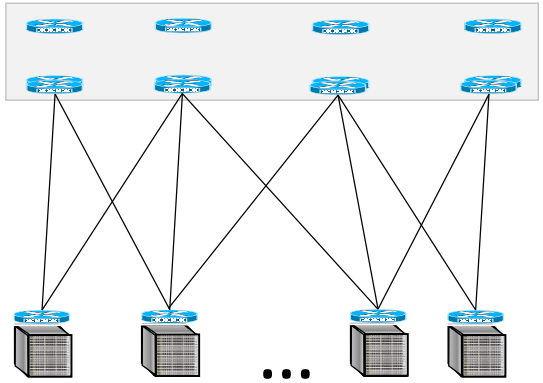


A completely different approach to DCN

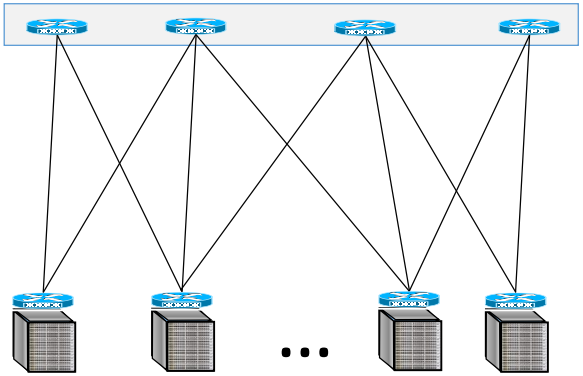


Data Center Network Architectures

Over provisioned
(e.g. FatTree, Jellyfish)

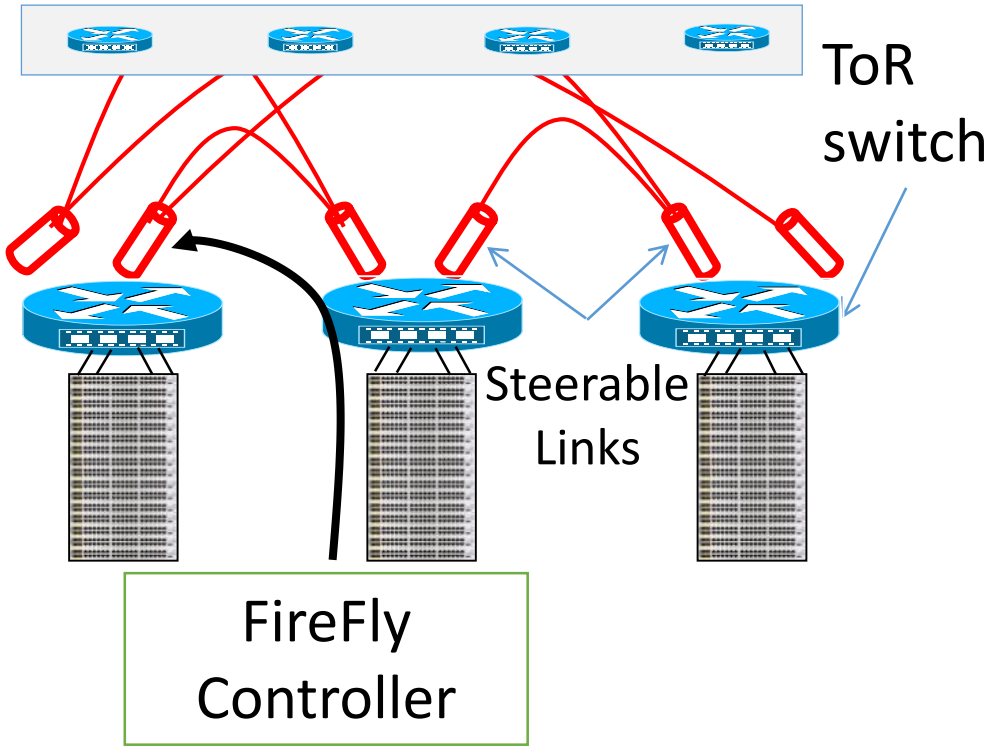


Over subscribed
(e.g. simple tree)



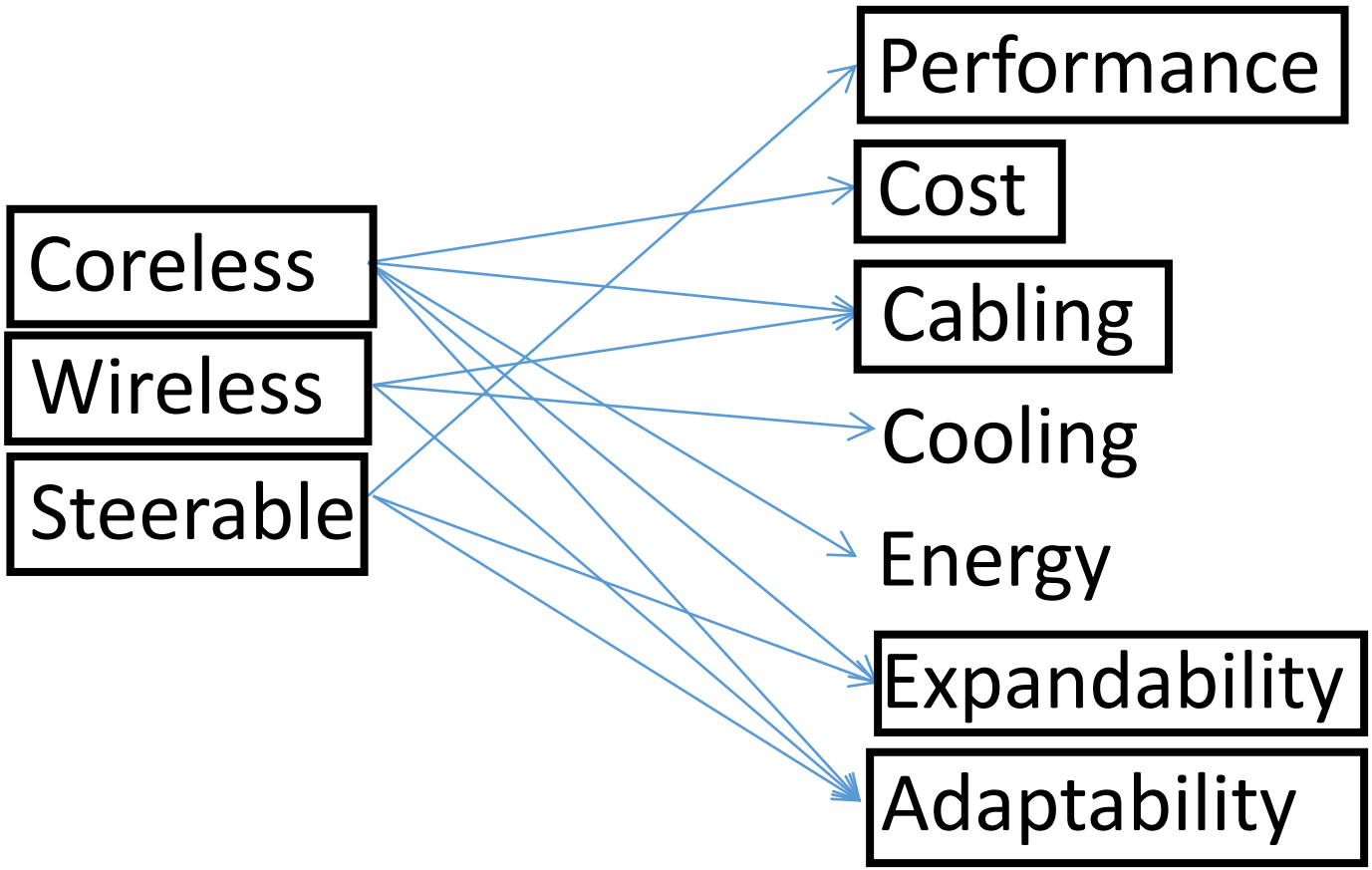
Enter Firefly [1]

- Coreless
- Wireless
- Steerable



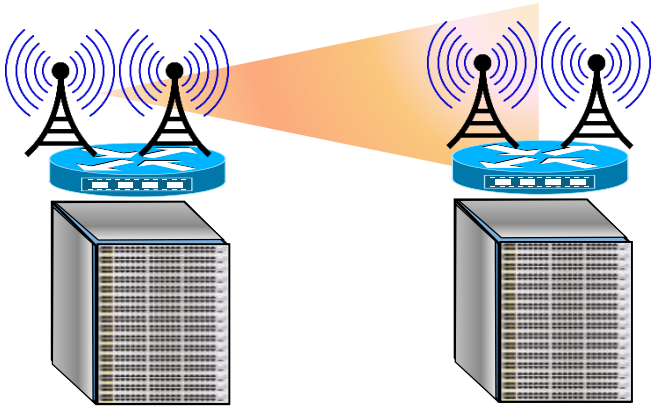
[1] Hamedazimi et al. "FireFly: a reconfigurable wireless data center fabric using free-space optics.“ *Proceedings of the 2014 ACM conference on SIGCOMM*. ACM, 2014.

Potential Benefits of This Vision



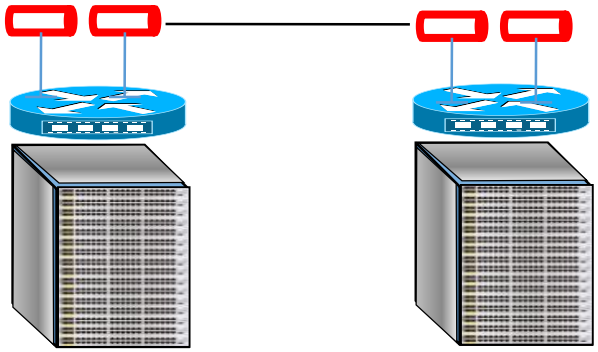
How to achieve Wireless Switches?

RF (e.g. 60GHZ)



Wide beam ➔
High interference
Limited active links
Limited Throughput

FSO (Free Space Optical)



Narrow beam ➔
Zero interference
No limit on active links
High Throughput

Today's FSO

- Cost: $15K per FSO
- Size: 1sqm
- Power: 30w
- Non steerable

- Current: bulky, power-hungry, and expensive
- Required: small, low power and low expense

Reducing size, price and power consumption

- Traditional use : outdoor, long haul
  - High power
  - Weatherproof
- Data centers: indoor, short haul
- Feasible roadmap via commodity fiber optics
  - E.g. Small form transceivers (Optical SFP)

Steerability

Shortcomings of current FSOs

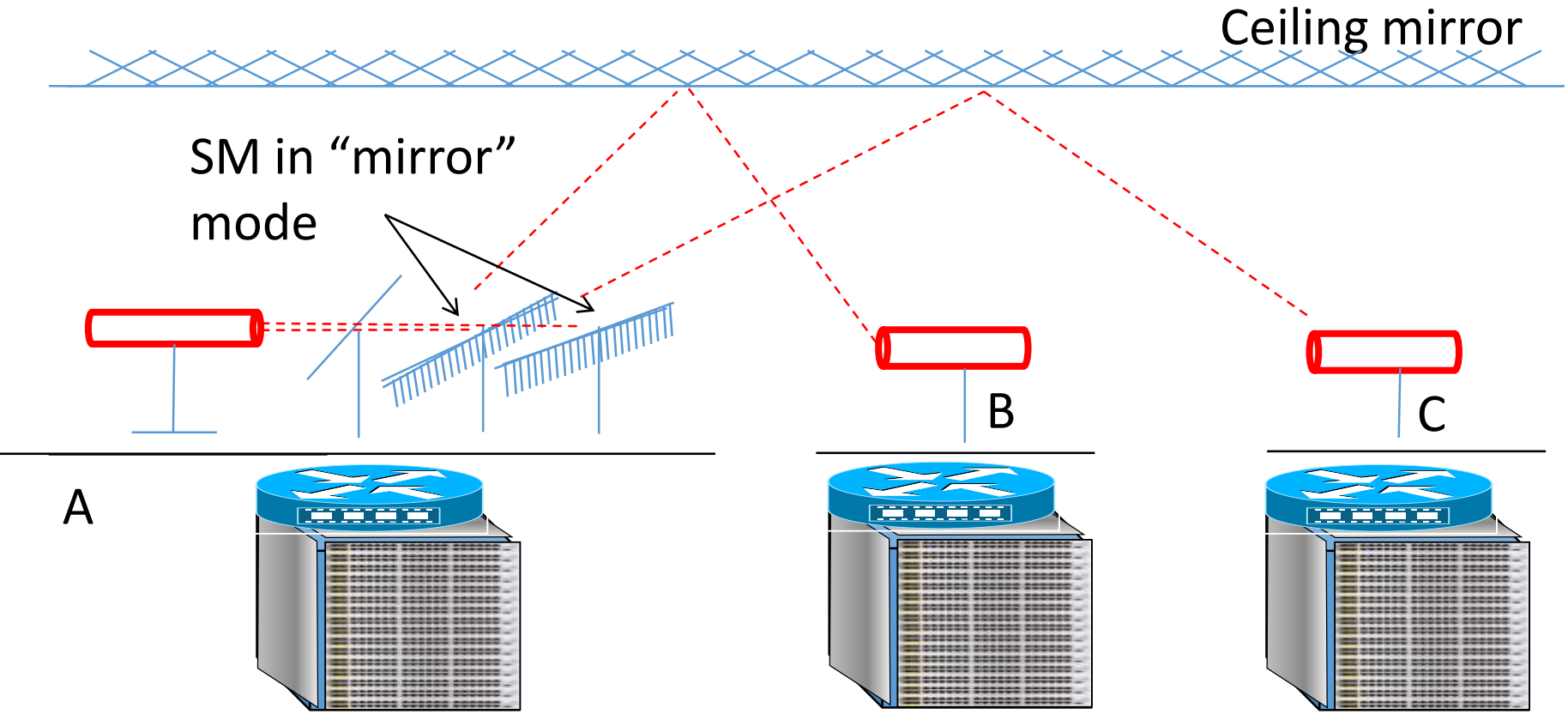
- ✓Cost
- ✓Size
- ✓Power
- Not Steerable

FSO design using SFP

Via Switchable mirrors or Galvo mirrors

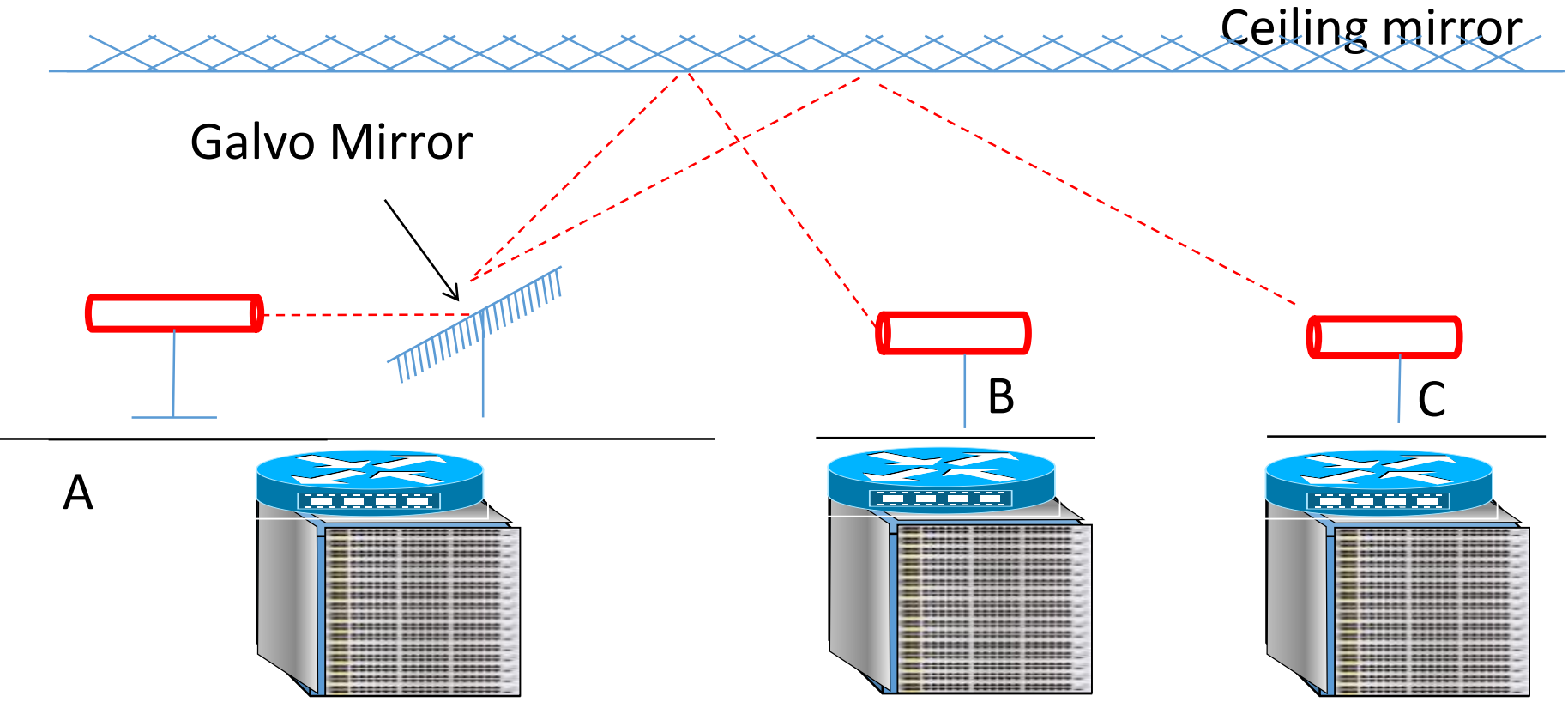
Steerability via Switchable Mirror

- Switchable Mirror: glass ⟷ mirror
- Electronic control, low latency

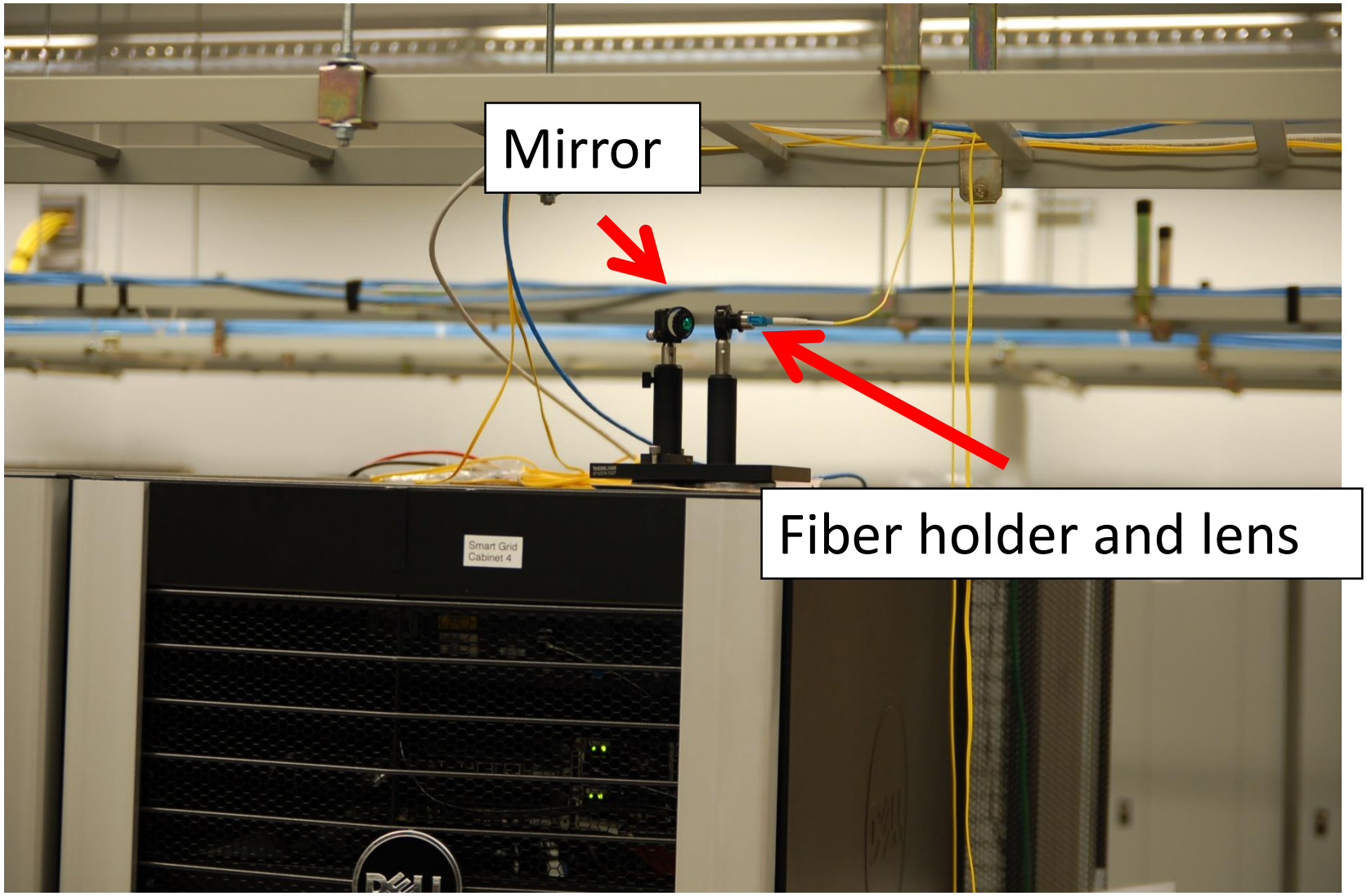


Steerability via Galvo Mirror

- Galvo Mirror: small rotating mirror
- Very low latency

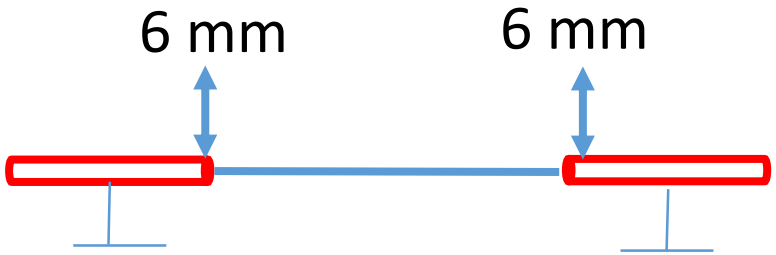


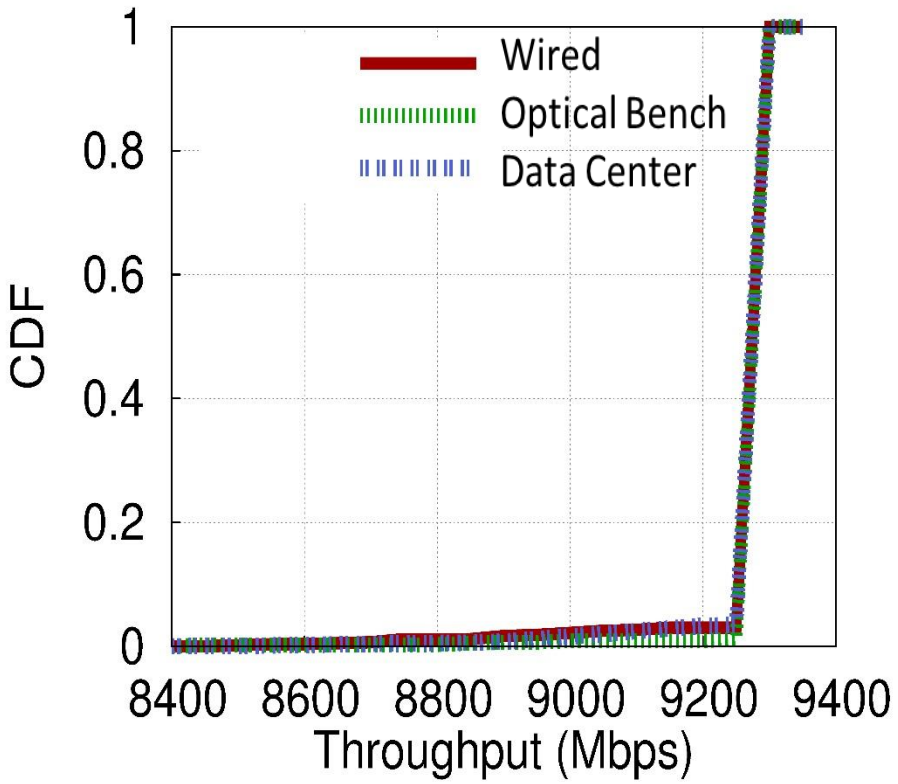
FSO Prototype in Data center



FSO Link Performance

- Effect of vibrations, etc.
- 6mm movement tolerance
- Range up to 24m tested





FSO link is as robust as a wired link

A different perspective - protocols

- Cloud computing service provider
  - Amazon,Microsoft,Google
- Transport **inside** the DC
  - TCP rules (99.9% of traffic)
- How is TCP doing?

TCP in the Data Center

- TCP does not meet demands of apps.
  - Incast
    - Suffers from bursty packet drops
    - Not fast enough utilize spare bandwidth
  - Builds up large queues:
    - Adds significant latency.
    - Wastes precious buffers, esp. bad with shallow-buffered switches.
- Operators work around TCP problems.
  - Ad-hoc, inefficient, often expensive solutions
- A solution: **Data Center TCP [1]**

[1] Alizadeh et al. "Data center tcp (dctcp)."
ACM SIGCOMM computer communication review 41.4 (2011): 63-74.

Case Study: Microsoft Bing

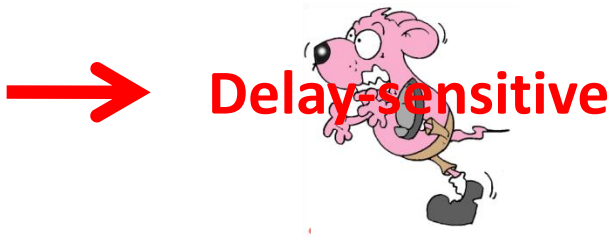
- Measurements from 6000 server production cluster
- Instrumentation passively collects logs
  - Application-level
  - Socket-level
  - Selected packet-level
- More than **150TB** of compressed data over a month

Partition/Aggregate Application Structure

- Time is money
  - **Strict deadlines (SLAs)**
- Missed deadline
  - **Lower quality result**

Workloads

- Partition/Aggregate
  (Query)



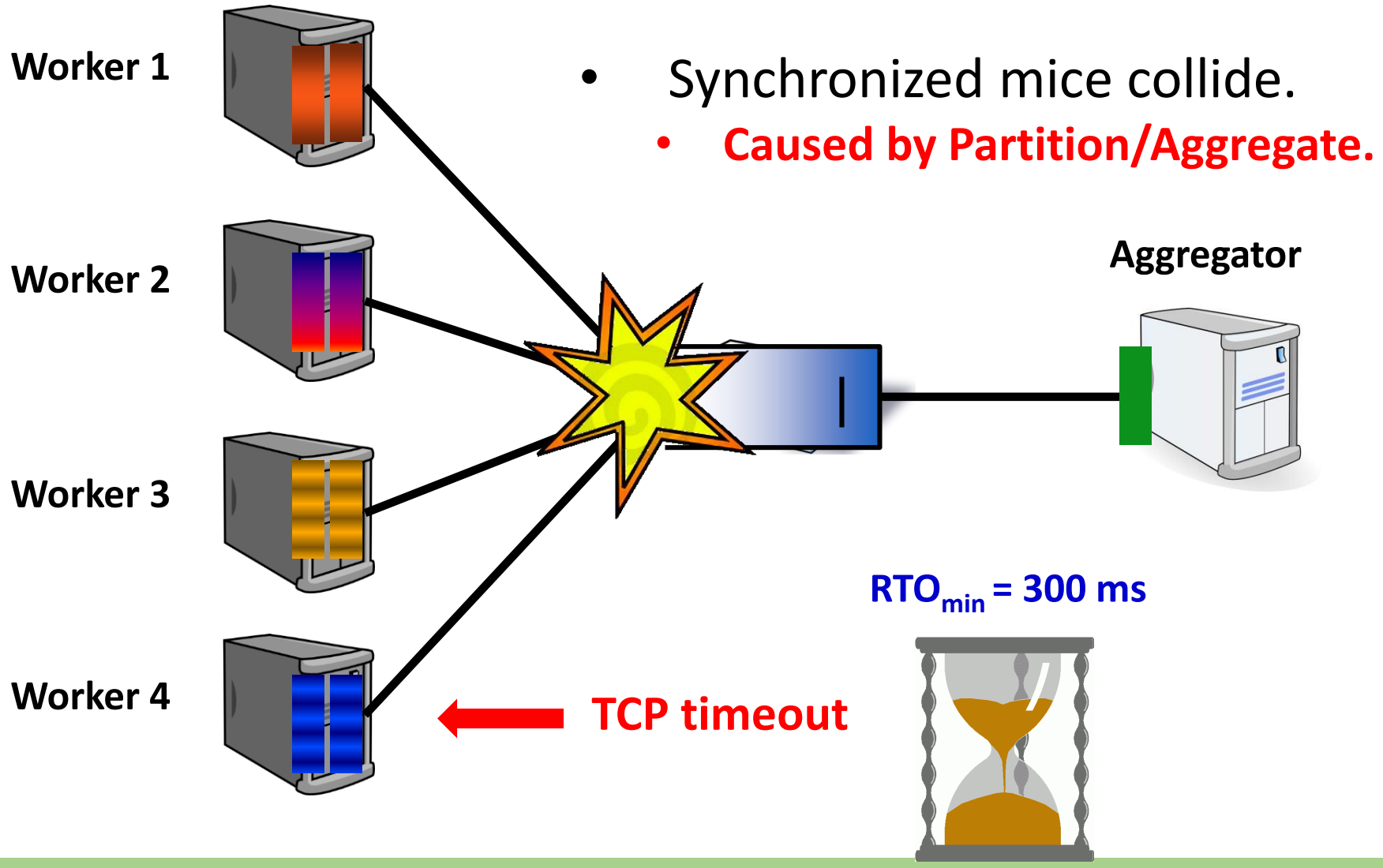
- Short messages [50KB-1MB]
  (Coordination, Control state)

- Large flows [1MB-50MB]
  (Data update)

Impairments

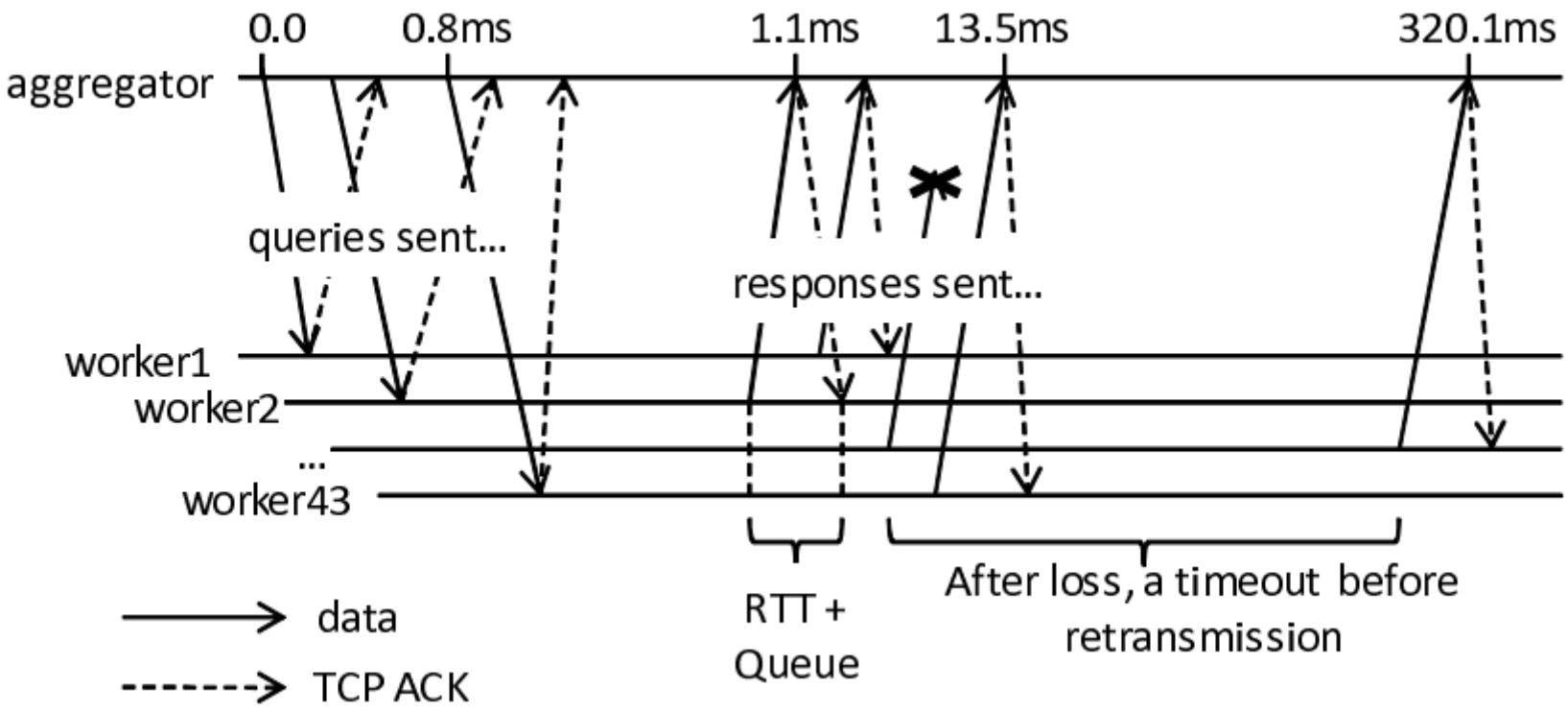
- Incast
- Queue Buildup

Incast

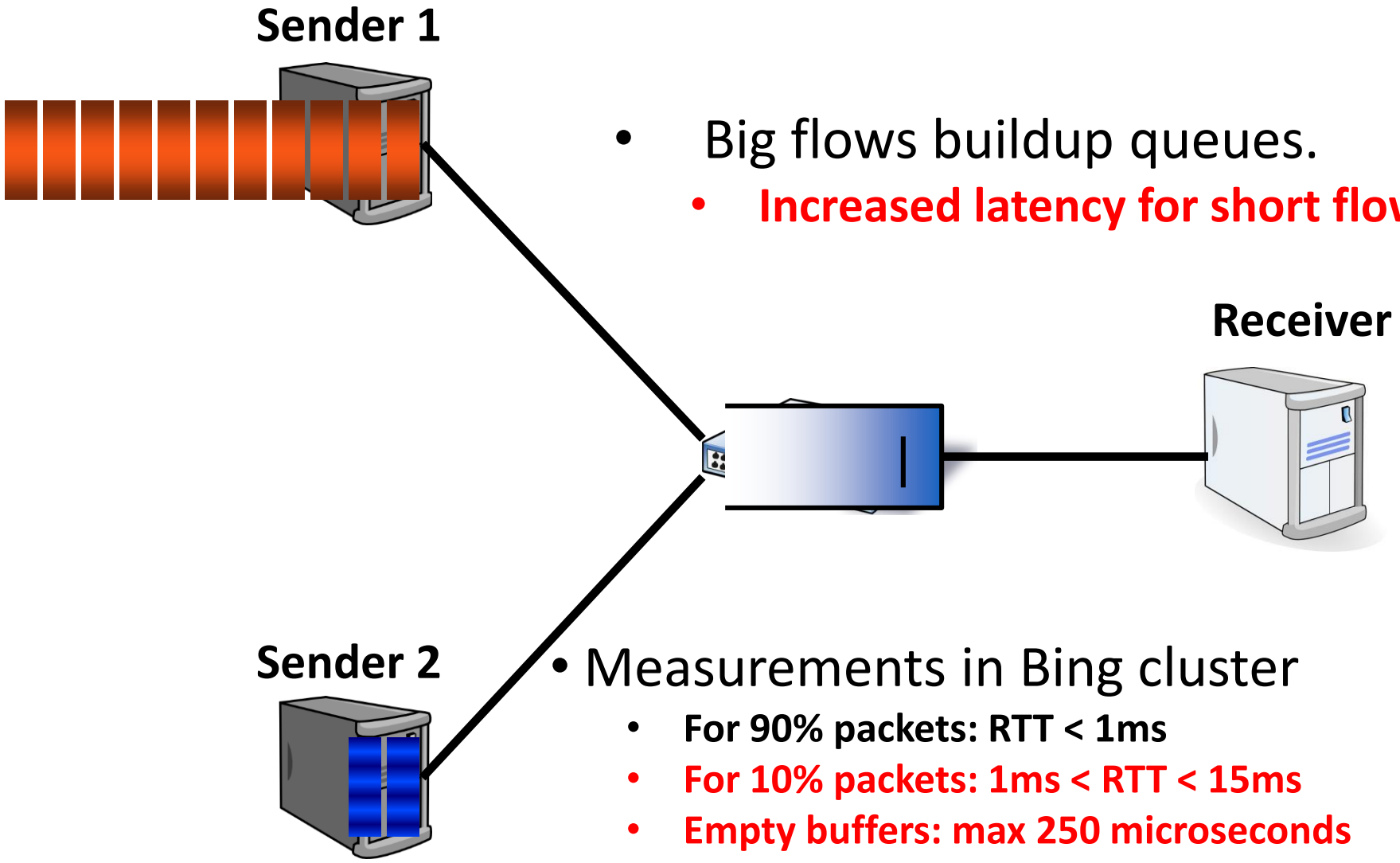


- Synchronized mice collide.
  - **Caused by Partition/Aggregate.**

Incast


0.0
0.8ms
1.1ms
13.5ms
320.1ms
aggregator
queries sent...
responses sent...
worker1
worker2
...
worker43
data
TCP ACK
RTT +
Queue
After loss, a timeout before
retransmission

Queue Buildup



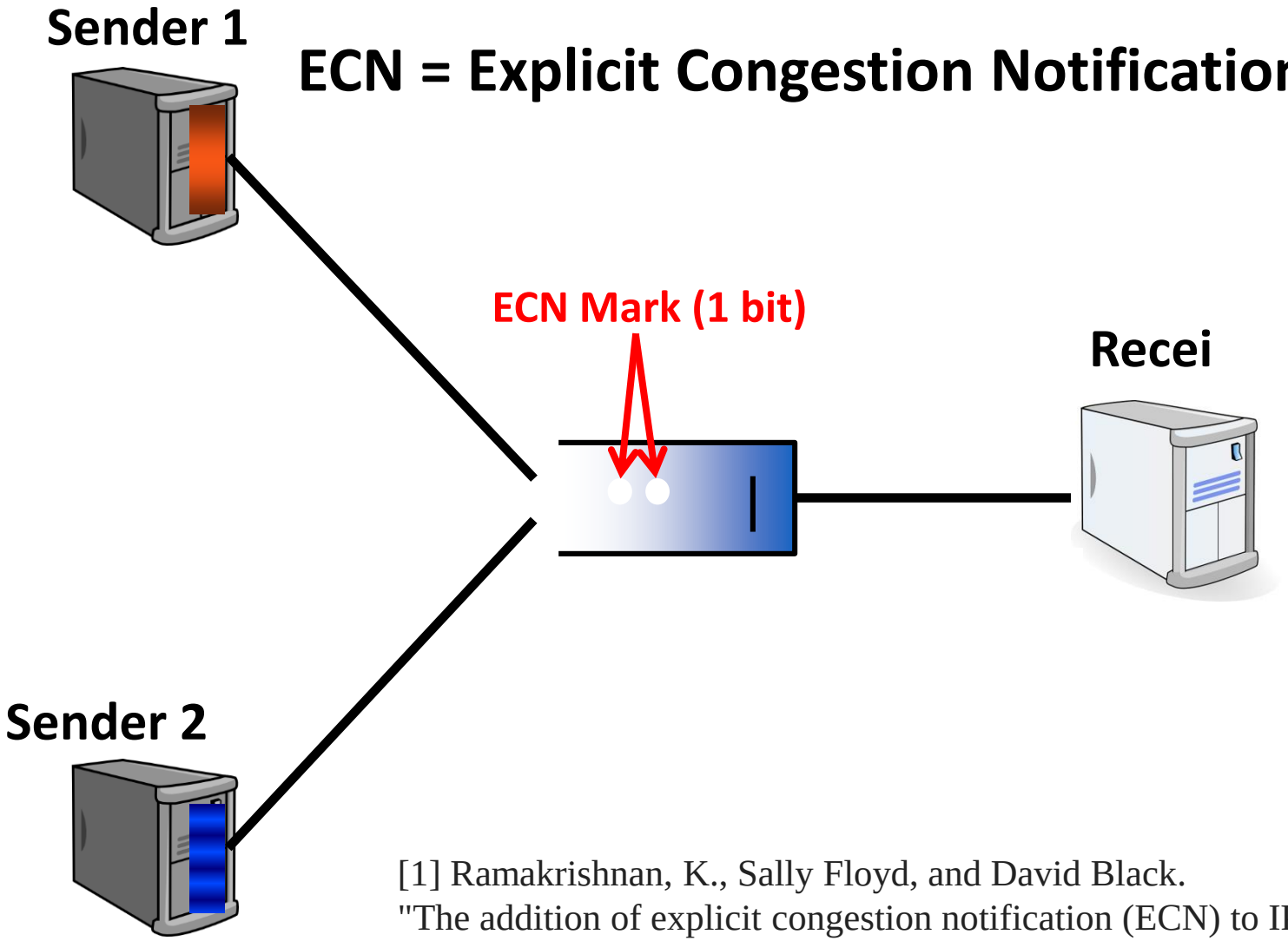
- Big flows buildup queues.
  - **Increased latency for short flows.**
- Measurements in Bing cluster
  - **For 90% packets: RTT < 1ms**
  - **For 10% packets: 1ms < RTT < 15ms**
  - **Empty buffers: max 250 microseconds**

Data Center Transport Requirements

- **High Burst Tolerance**
  - Incast due to Partition/Aggregate is common.
- **Low Latency**
  - Short flows, queries
- **High Throughput**
  - Large file transfers

The challenge is to achieve these three together.

The TCP/ECN Control Loop



[1] Ramakrishnan, K., Sally Floyd, and David Black.
"The addition of explicit congestion notification (ECN) to IP." (2001).

Two Key Ideas

1. React in proportion to the **extent** of congestion, not its **presence**.
   - Reduces **variance** in sending rates, lowering queuing requirements.

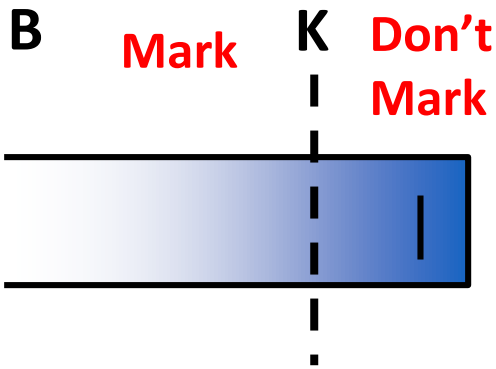
ECN Marks	TCP	DCTCP
1 0 1 1 1 1 0 1 1 1	Cut window by 50%	Cut window by 40%
0 0 0 0 0 0 0 0 0 1	Cut window by 50%	Cut window by 5%

2. Mark based on **instantaneous** queue length.
   - Fast feedback to better deal with bursts.

Data Center TCP Algorithm

Switch side:

- Mark packets when **Queue Length > K.**



Sender side:

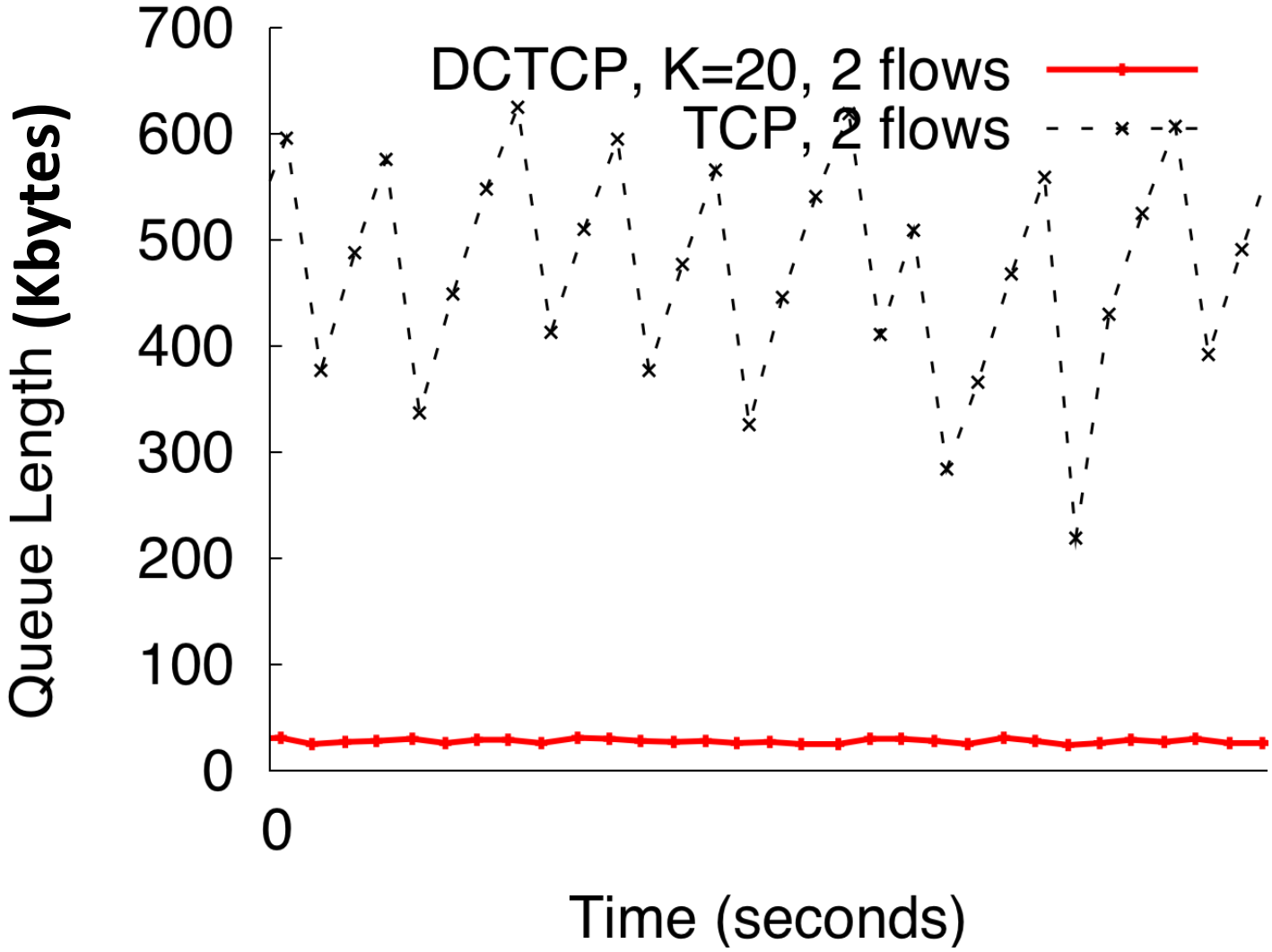
- Maintain running average of ***fraction*** of packets marked **(α)**.

In each RTT:

$$F = \frac{\#\ of\ marked\ ACKs}{Total\ \#\ of\ ACKs} \qquad \alpha \leftarrow (1-g)\alpha + gF$$

- **Adaptive window decreases:** $Cwnd \leftarrow (1 - \frac{\alpha}{2})Cwnd$

DCTCP in Action



Why it Works

- **High Burst Tolerance**
  - **Large buffer headroom →** bursts fit.
  - **Aggressive marking →** sources react before packets are dropped.
- **Low Latency**
  - **Small buffer occupancies →** low queuing delay.
- **High Throughput**
  - **ECN averaging →** smooth rate adjustments, cwind low variance.