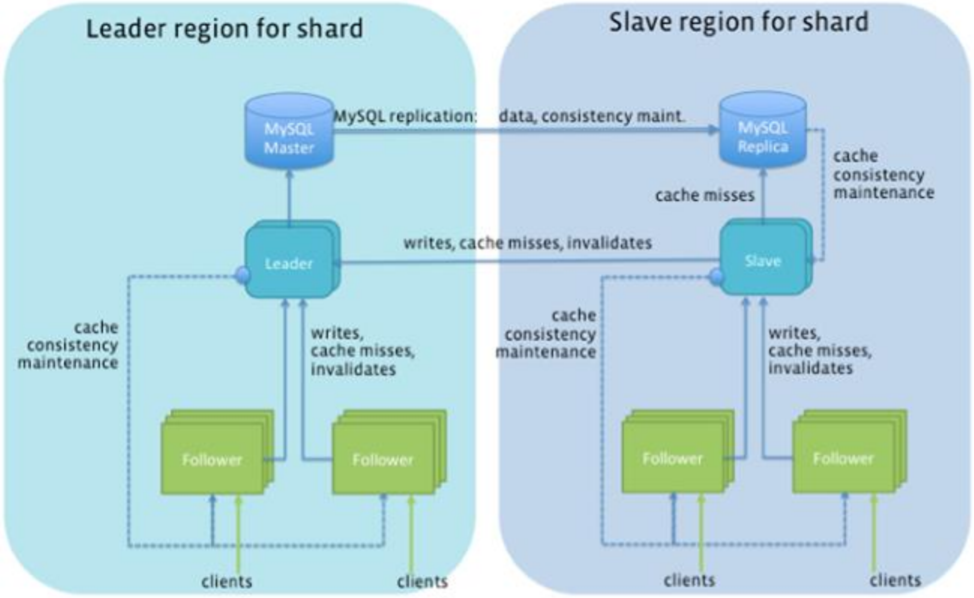
CS 6453
Network Fabric

Presented by Ayush Dubey

Based on:

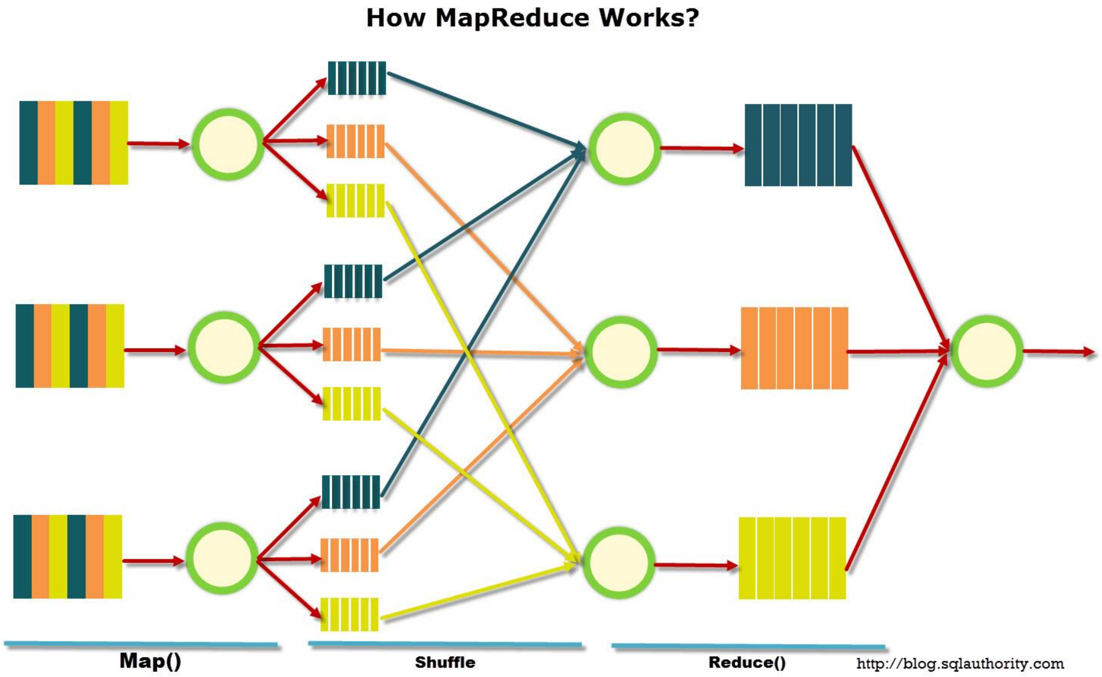
1. Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. Singh et al. SIGCOMM15.
2. Network Traffic Characteristics of Data Centers in the Wild. Benson et al. IMC10.
3. Benson's original slide deck from IMC10.

Example – Facebook’s Graph Store Stack



Source: https://www.facebook.com/notes/facebook-engineering/tao-the-power-of-the-graph/10151525983993920/

Example - MapReduce



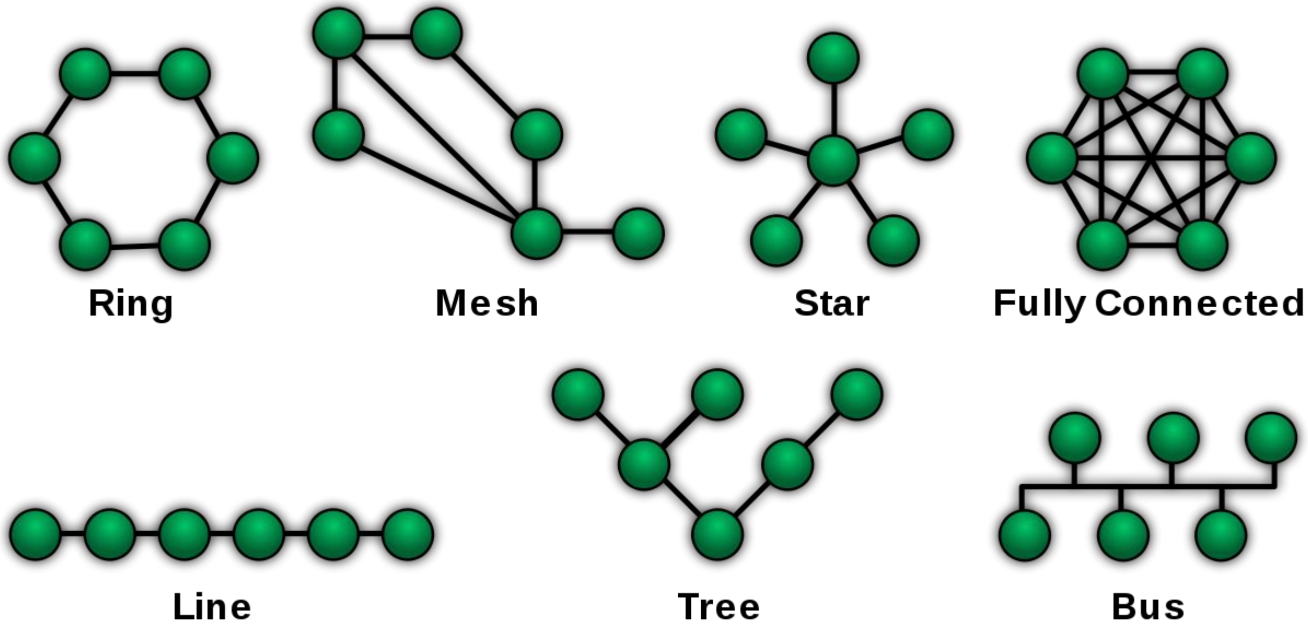
Source: https://blog.sqlauthority.com/2013/10/09/big-data-buzz-words-what-is-mapreduce-day-7-of-21/

Performance of distributed systems depends heavily on the datacenter interconnect

Evaluation Metrics for Datacenter Topologies

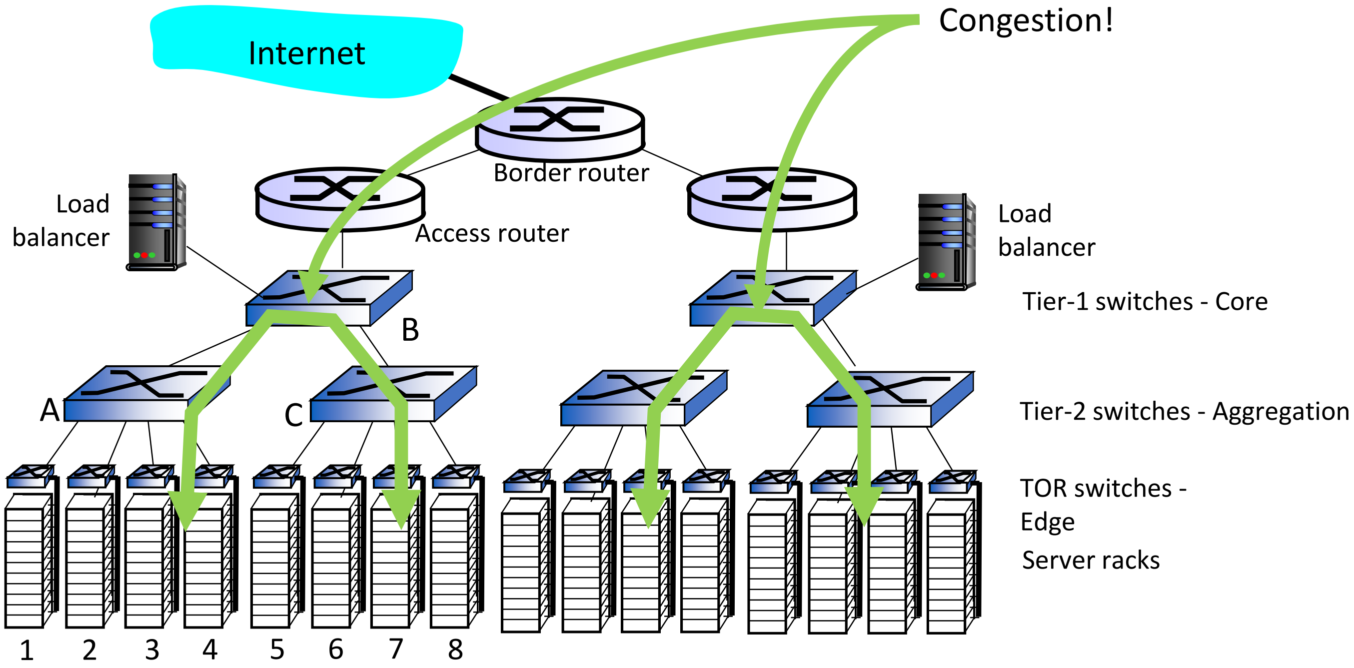
- Diameter – max #hops between any 2 nodes
  - Worst case latency
- Bisection Width – min #links cut to partition network into 2 equal halves
  - Fault tolerance
- Bisection Bandwidth – min bandwidth between any 2 equal halves of the network
  - Bottleneck
- Oversubscription – ratio of worst-case achievable aggregate bandwidth between end-hosts to total bisection bandwidth

Legacy Topologies



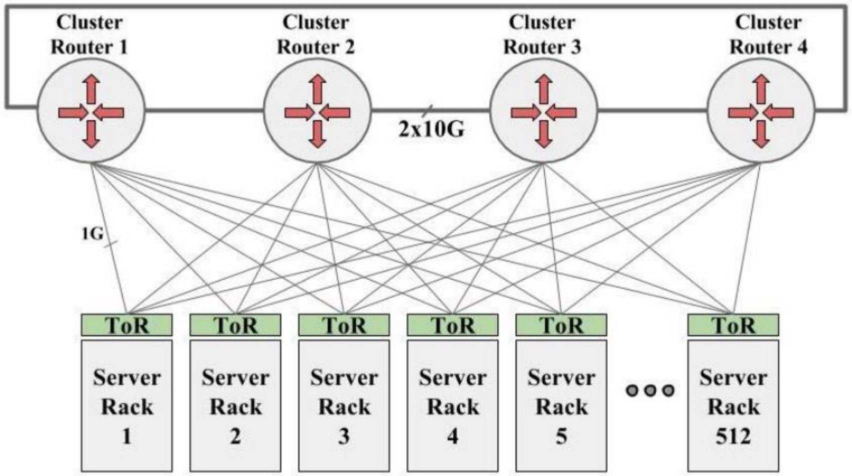
Source: http://pseudobit.blogspot.com/2014/07/network-classification-by-network.html

3-Tier Architecture



Source: CS 5413, Hakim Weatherspoon, Cornell University

Big-Switch Architecture



Cost $O(100,000)!

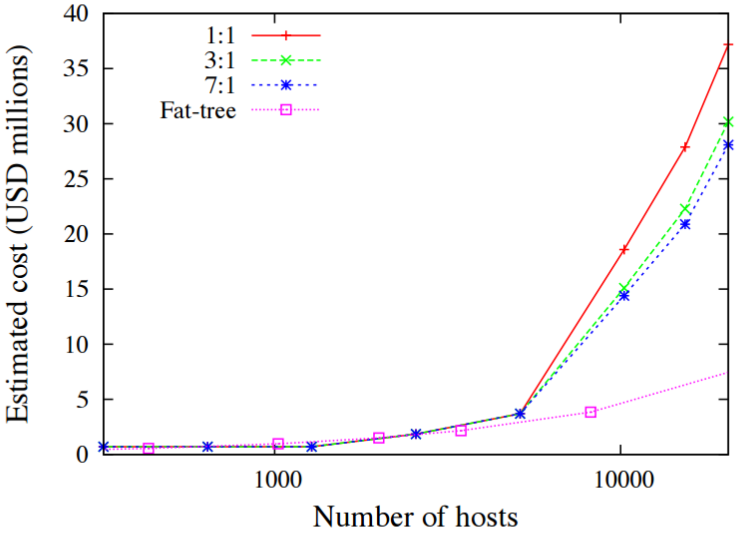
Cost $O(1,000)!

Figure 2: A traditional 2Tbps four-post cluster (2004). Top of Rack (ToR) switches serving 40 1G-connected servers were connected via 1G links to four 512 1G port Cluster Routers (CRs) connected with 10G sidelinks.

Source: Jupiter Rising, Google

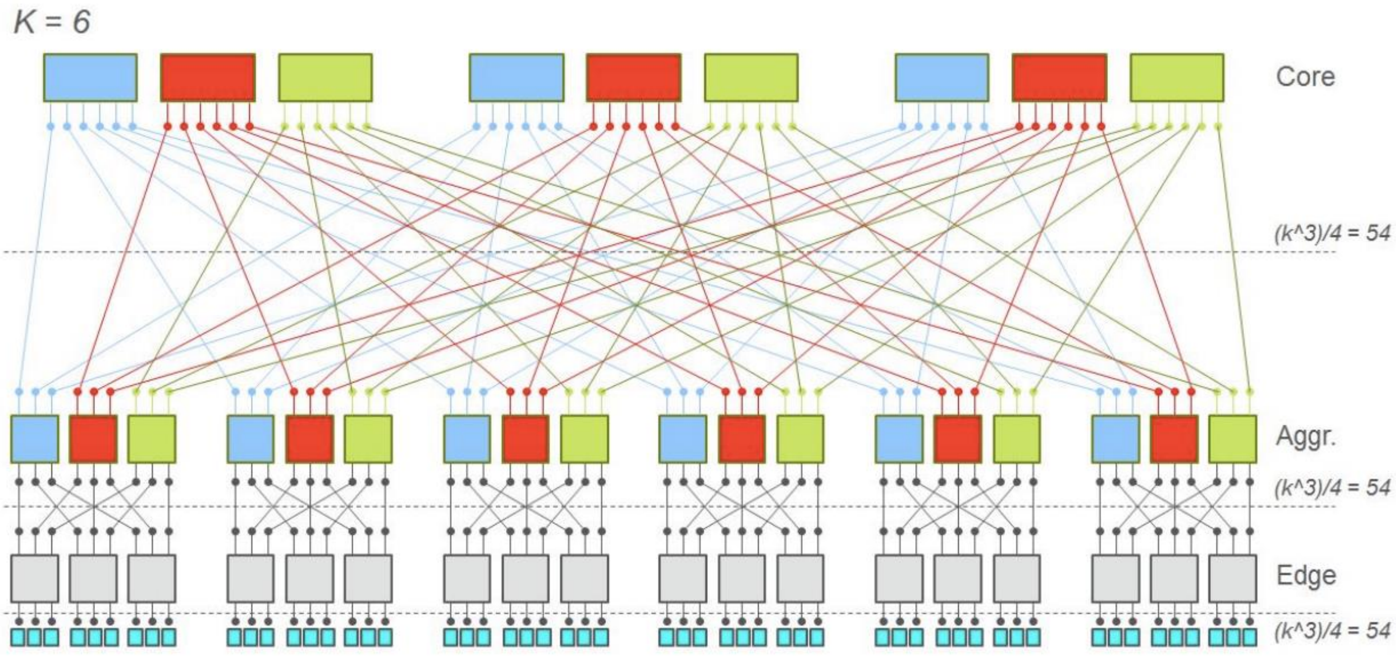
Goals for Datacenter Networks (circa 2008)

- 1:1 oversubscription ratio – all hosts can communicate with arbitrary other hosts at full bandwidth of their network interface
    - Google's Four-Post CRs offered only about 100Mbps
- Low cost – cheap off-the-shelf switches



Source: A Scalable, Commodity Data Center Network Architecture. Al-Fares et al.

Fat-Trees



Source: Francesco Celestino, https://www.systems.ethz.ch/sites/default/files/file/acn2016/slides/04-topology.pdf

Advantages of Fat-Tree Design

- Increased throughput between racks
- Low cost because of commodity switches
- Increased redundancy

Case Study: The Evolution of Google’s Datacenter Network

(Figures from original paper)

Google Datacenter Principles

- High bisection bandwidth and graceful fault tolerance
  - Clos/Fat-Tree topologies
- Low Cost
  - Commodity silicon
- Centralized control

Firehose 1.0

- Goal – 1Gbps bisection bandwidth to each 10K servers in datacenter

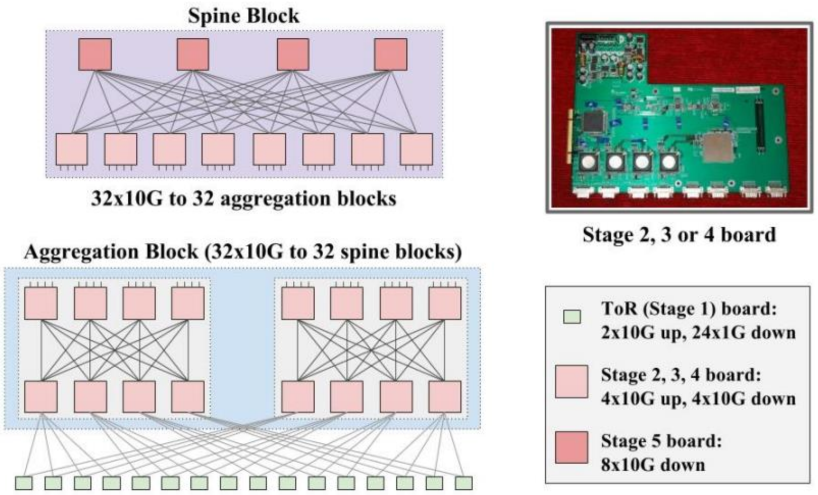


Figure 5: Firehose 1.0 topology. Top right shows a sample 8x10G port fabric board in Firehose 1.0, which formed Stages 2, 3 or 4 of the topology.

Firehose 1.0 – Limitations

- Low radix (#ports) ToR switch easily partitions the network on failures
- Attempted to integrate switching fabric into commodity servers using PCI
  - No go, servers fail frequently
- Server to server wiring complexity
- Electrical reliability

Firehose 1.1 – First Production Fat-Tree

- Custom enclosures with dedicated single-board computers
  - Improve reliability compared to regular servers
- Buddy two ToR switches by interconnecting
  - At most 2:1 oversubscription
  - Scales up to 20K machines
- Use fiber rather than Ethernet for longest distances (ToR to above)
  - Workaround 14m CX4 cable limit improves deployability
- Deployed on the side with legacy four-post CR

Watchtower

- Goal – leverage next-gen 16X10G merchant silicon switch chips
- Support larger fabrics with more bandwidth
- Fiber bundling reduces cable complexity and cost

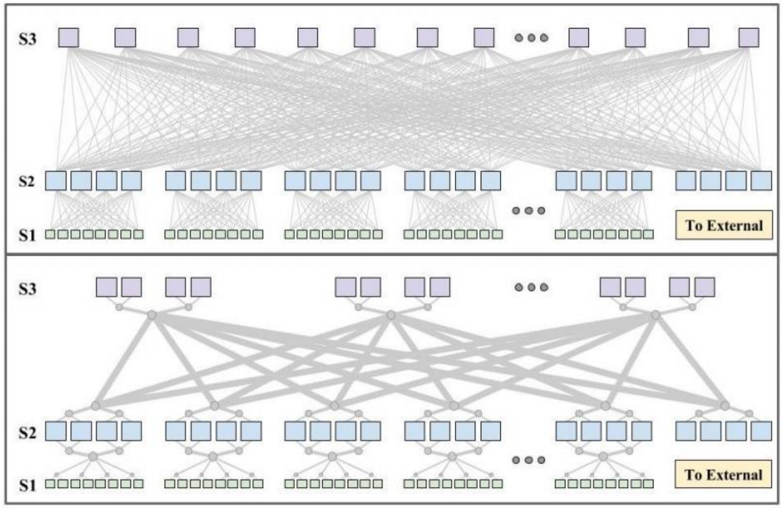


Figure 10: Reducing deployment complexity by bundling cables. Stages 1, 2 and 3 in the fabric are labeled S1, S2 and S3, respectively.

Watchtower – Depopulated Clusters

- Natural variation in bandwidth demands across clusters
- Dominant fabric cost is optics and associated fiber
- A is twice as cost-effective as B

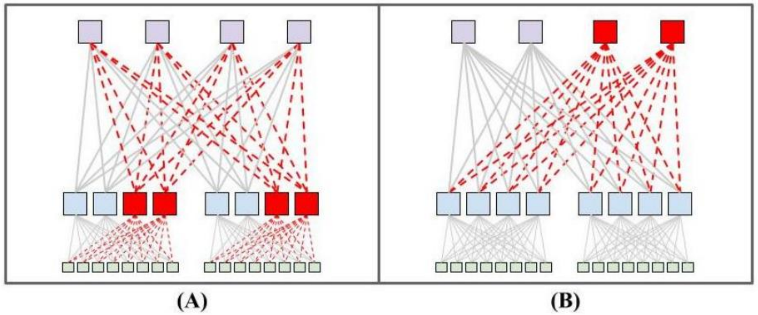


Figure 11: Two ways to depopulate the fabric for 50% capacity.

Saturn and Jupiter

- Better silicon gives higher bandwidth
- Lots of engineering challenges detailed in the paper

Software Control

- Custom control plane
  - Existing protocols did not support multipath, equal-cost forwarding
  - Lack of high quality open source routing stacks
  - Protocol overhead of running broadcast-based algorithms on such large scale
  - Easier network manageability
- Treat the network as a single fabric with O(10,000) ports
- Anticipated some of the principles of Software Defined Networking

Issues – Congestion

High congestion as utilization approached 25%

- Bursty flows
- Limited buffer on commodity switches
- Intentional oversubscription for cost saving
- Imperfect flow hashing

Congestion – Solutions

- Configure switch hardware schedulers to drop packets based on QoS
- Tune host congestion window
- Link-level pause reduces over-running oversubscribed links
- Explicit Congestion Notification
- Provision bandwidth on-the-fly by repopulating
- Dynamic buffer sharing on merchant silicon to absorb bursts
- Carefully configure switch hashing to support ECMP load balancing

Issues – Control at Large Scale

- Liveness and routing protocols interact badly
  - Large-scale disruptions
  - Required manual interventions
- We can now leverage many years of SDN research to mitigate this!
  - E.g. consistent network updates addressed in “Abstractions for Network Update” by Reitblatt et al.

Google Datacenter Principles – Revisited

- High bisection bandwidth and graceful fault tolerance
  - Clos/Fat-Tree topologies
- Low Cost
  - Commodity silicon
- Centralized control

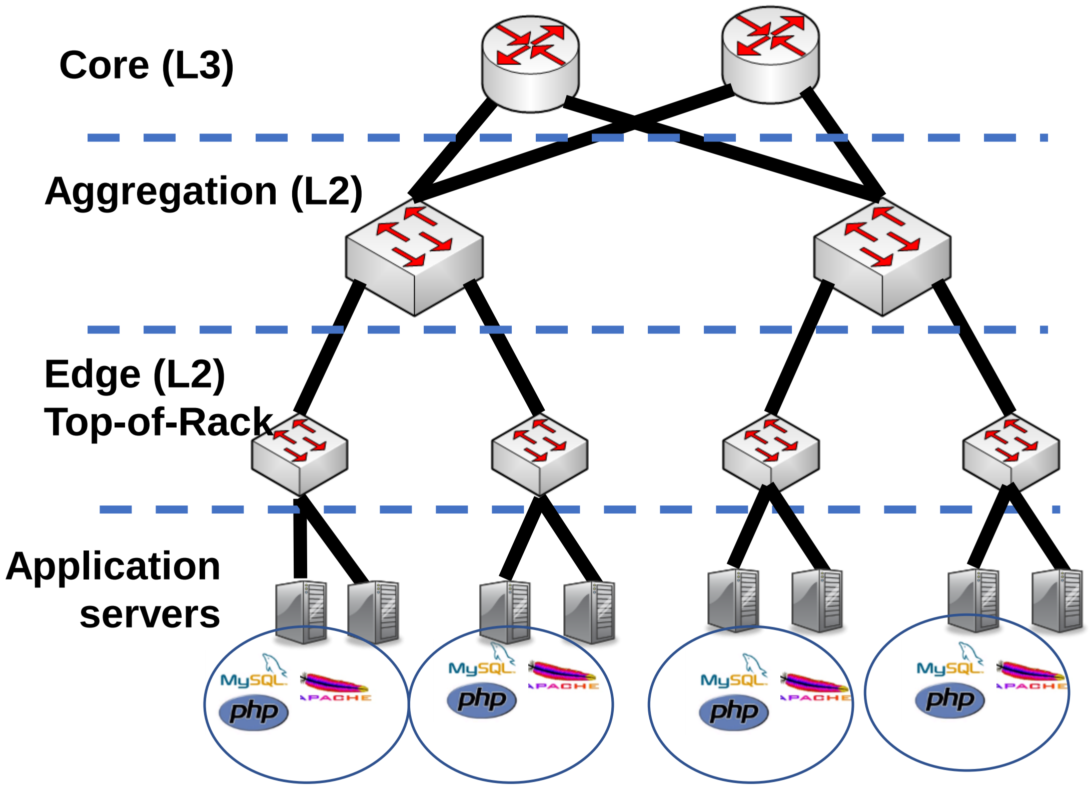
Do real datacenter workloads match these goals?

(Disclaimer: following slides are adapted from Benson's slide deck)

The Case for Understanding Data Center Traffic

- Better understanding → better techniques
- Better traffic engineering techniques
  - Avoid data losses
  - Improve app performance
- Better Quality of Service techniques
  - Better control over jitter
  - Allow multimedia apps
- Better energy saving techniques
  - Reduce data center's energy footprint
  - Reduce operating expenditures
- Initial stab→ network level traffic + app relationships

Canonical Data Center Architecture



Core (L3)

Aggregation (L2)

Edge (L2)
Top-of-Rack

Application
servers

Dataset: Data Centers Studied

- 10 data centers
- 3 classes
  - Universities
  - Private enterprise
  - Clouds
- Internal users
  - Univ/priv
  - Small
  - Local to campus
- External users
  - Clouds
  - Large
  - Globally diverse

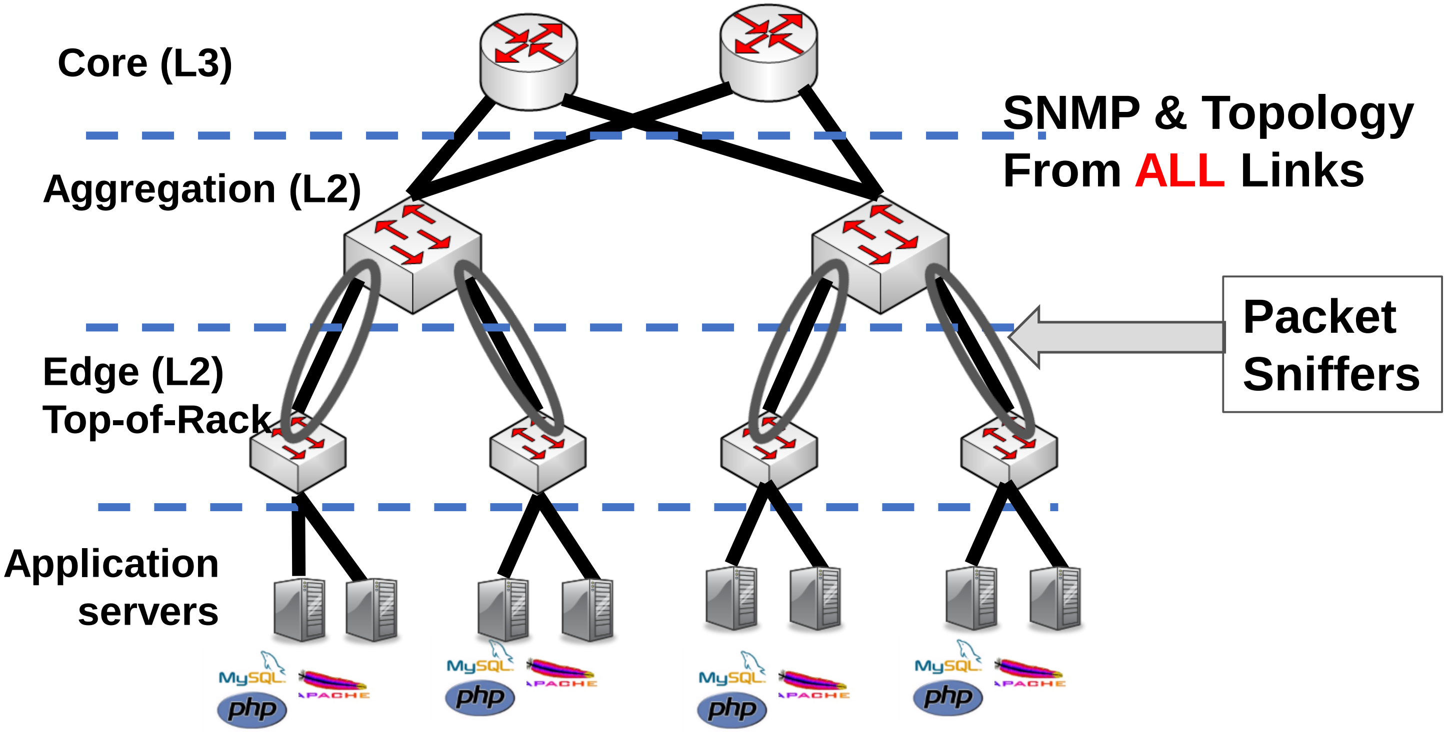
DC Role	DC Name	Location	Number Devices
Universities	EDU1	US-Mid	22
	EDU2	US-Mid	36
	EDU3	US-Mid	11
Private Enterprise	PRV1	US-Mid	97
	PRV2	US-West	100
Commercial Clouds	CLD1	US-West	562
	CLD2	US-West	763
	CLD3	US-East	612
	CLD4	S. America	427
	CLD5	S. America	427

Dataset: Collection

- SNMP
  - Poll SNMP MIBs
  - Bytes-in/bytes-out/discards
  - > 10 Days
  - Averaged over 5 mins
- Packet Traces
  - Cisco port span
  - 12 hours
- Topology
  - Cisco Discovery Protocol

DC Name	SNMP	Packet Traces	Topology
EDU1	Yes	Yes	Yes
EDU2	Yes	Yes	Yes
EDU3	Yes	Yes	Yes
PRV1	Yes	Yes	Yes
PRV2	Yes	Yes	Yes
CLD1	Yes	No	No
CLD2	Yes	No	No
CLD3	Yes	No	No
CLD4	Yes	No	No
CLD5	Yes	No	No

Canonical Data Center Architecture



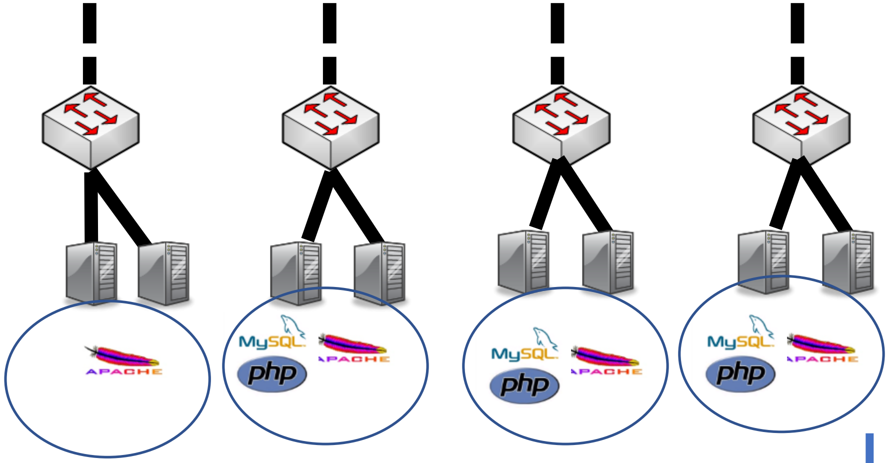
SNMP & Topology From ALL Links

Packet Sniffers

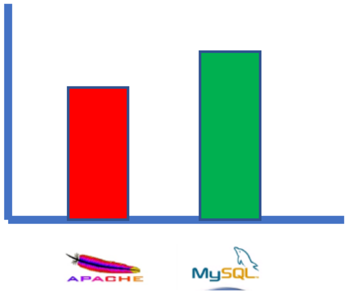
Topologies

Datacenter	Topology	Comments
EDU1	2-Tier	Middle-of-Rack switches instead of ToR
EDU2	2-Tier	
EDU3	Star	High capacity central switch connecting racks
PRV1	2-Tier	
PRV2	3-Tier	
CLD	Unknown	

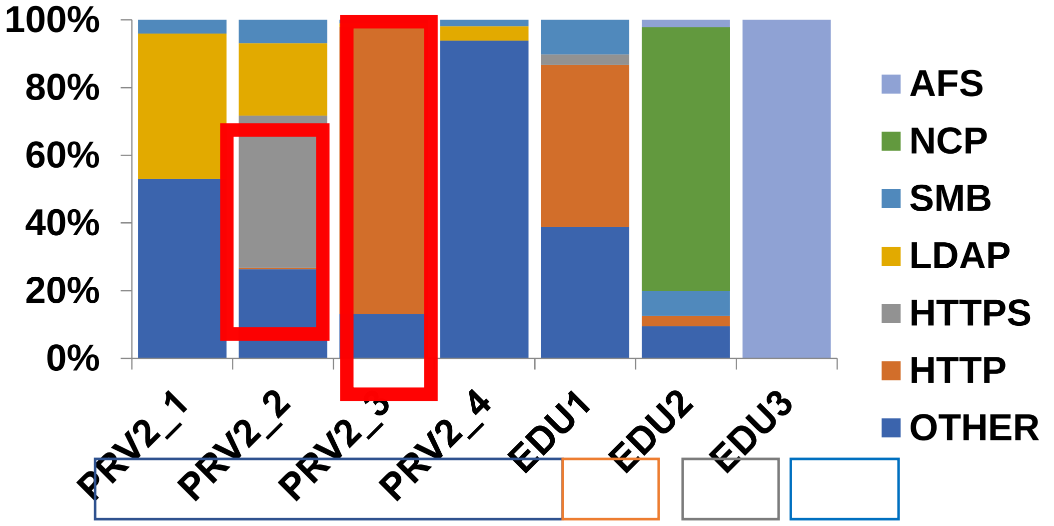
Applications



- Start at bottom
  - Analyze running applications
  - Use packet traces
- BroID tool for identification
  - Quantify amount of traffic from each app



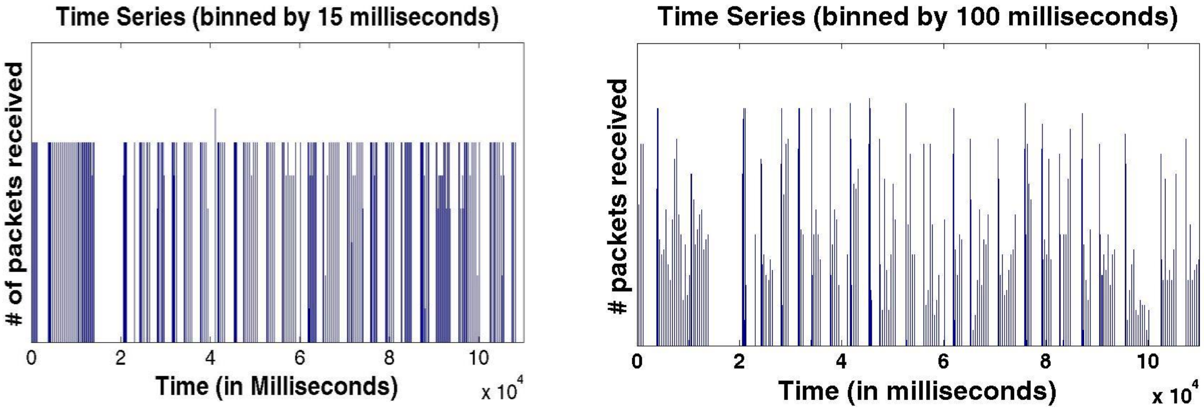
Applications



- Cannot assume uniform distribution of applications
- Clustering of applications
  - PRV2_2 hosts secured portions of applications
  - PRV2_3 hosts unsecure portions of applications

Analyzing Packet Traces

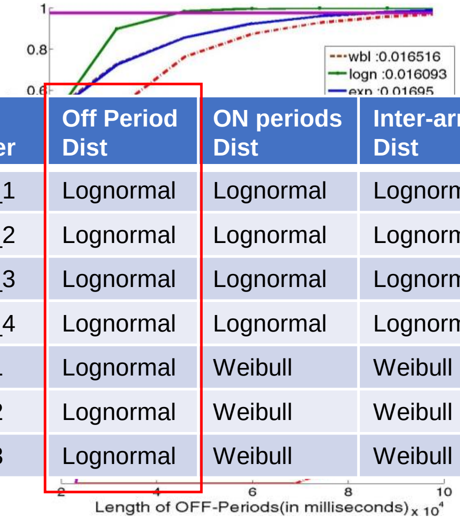
- Transmission patterns of the applications
- Properties of packet crucial for
  - Understanding effectiveness of techniques



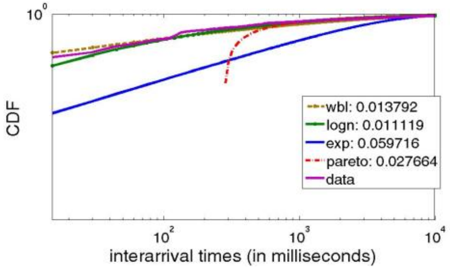
- ON-OFF traffic at edges
  - Binned in 15 and 100 m. secs
  - We observe that ON-OFF persists

Data-Center Traffic is Bursty

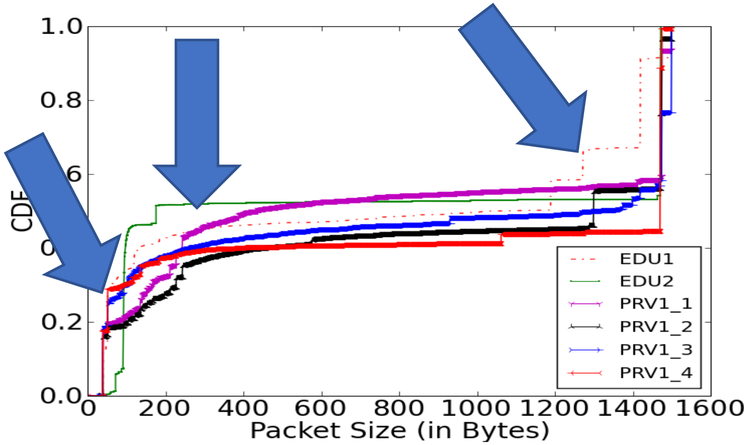
- Understanding arrival process
  - Range of acceptable models
- What is the arrival process?
  - **Heavy-tail** for the 3 distributions
    - ON, OFF times, Inter-arrival,
  - **Lognormal** across all data centers
- Different from Pareto of WAN
  - Need new models



Data Center	Off Period Dist	ON periods Dist	Inter-arrival Dist
Prv2_1	Lognormal	Lognormal	Lognormal
Prv2_2	Lognormal	Lognormal	Lognormal
Prv2_3	Lognormal	Lognormal	Lognormal
Prv2_4	Lognormal	Lognormal	Lognormal
EDU1	Lognormal	Weibull	Weibull
EDU2	Lognormal	Weibull	Weibull
EDU3	Lognormal	Weibull	Weibull



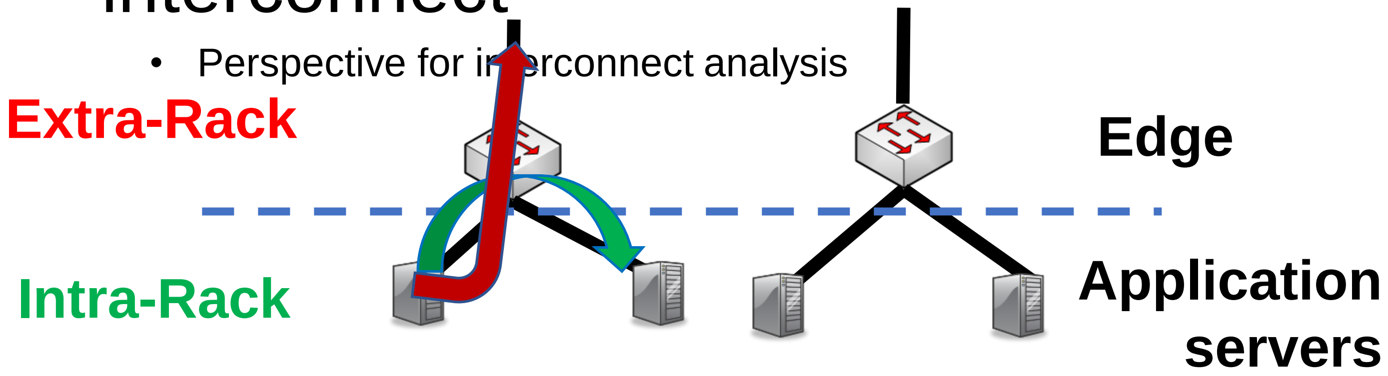
Packet Size Distribution



- Bimodal (200B and 1400B)
- Small packets
  - TCP acknowledgements
  - Keep alive packets
- Persistent connections → important to apps

Intra-Rack Versus Extra-Rack

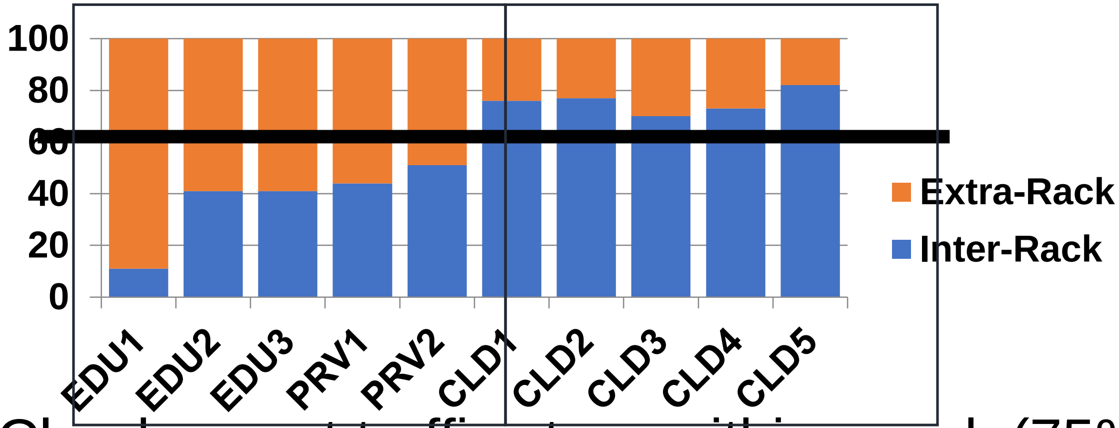
- Quantify amount of traffic using interconnect
  - Perspective for interconnect analysis



Extra-Rack = Sum of Uplinks

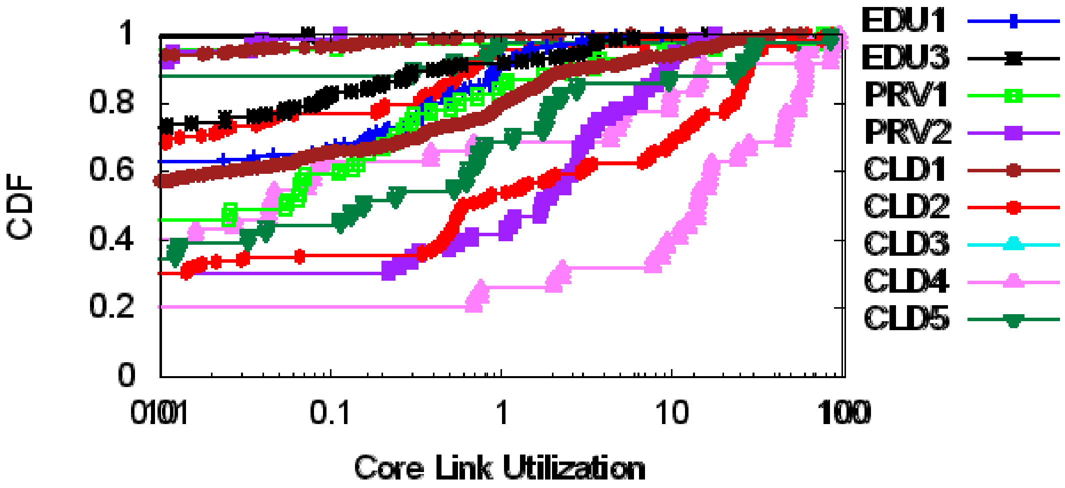
Intra-Rack = Sum of Server Links – Extra-Rack

Intra-Rack Versus Extra-Rack Results



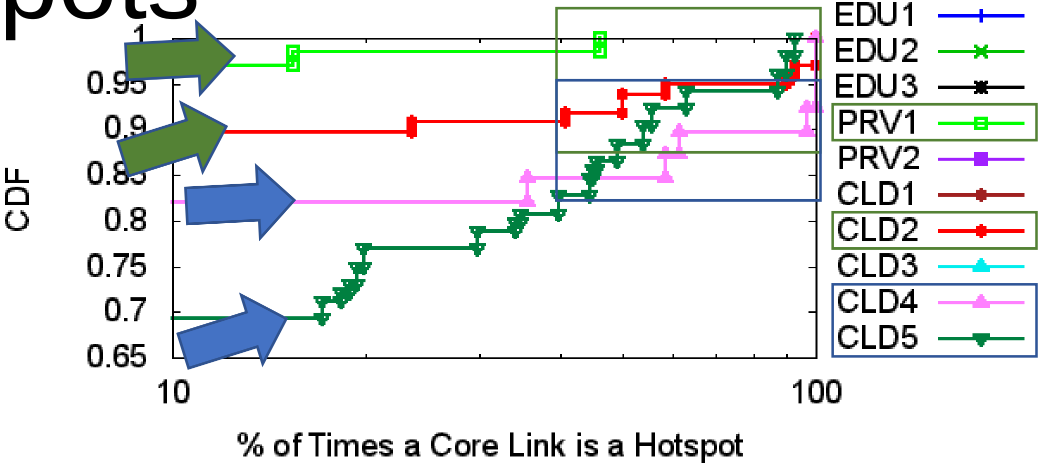
- Clouds: most traffic stays within a rack (75%)
  - Colocation of apps and dependent components
- Other DCs: > 50% leaves the rack
  - Un-optimized placement

Extra-Rack Traffic on DC Interconnect



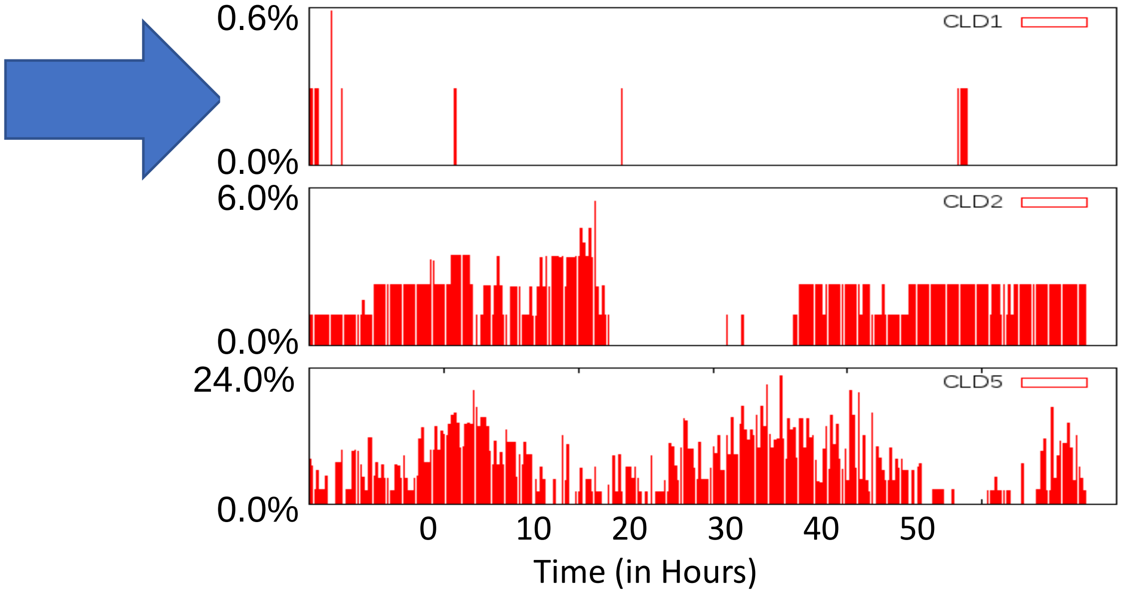
- Utilization: core > agg > edge
  - Aggregation of many unto few
- Tail of core utilization differs
  - Hot-spots → links with > 70% util
  - Prevalence of hot-spots differs across data centers

Persistence of Core Hot-Spots



- Low persistence: PRV2, EDU1, EDU2, EDU3, CLD1, CLD3
- High persistence/low prevalence: PRV1, CLD2
  - 2-8% are hotspots > 50%
- High persistence/high prevalence: CLD4, CLD5
  - 15% are hotspots > 50%

Prevalence of Core Hot-Spots

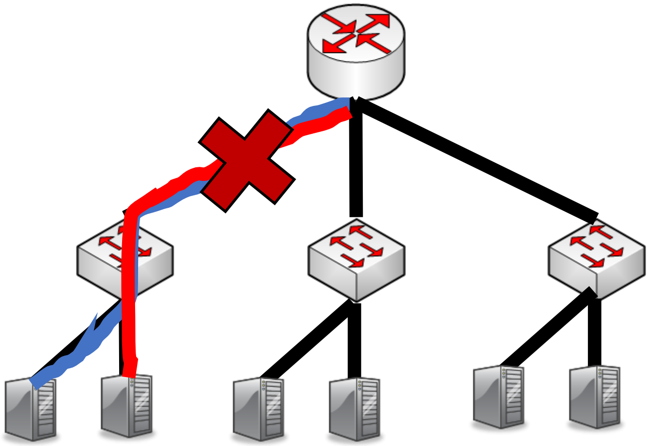


- Low persistence: very few concurrent hotspots
- High persistence: few concurrent hotspots
- High prevalence: < 25% are hotspots at any time

Observations from Interconnect

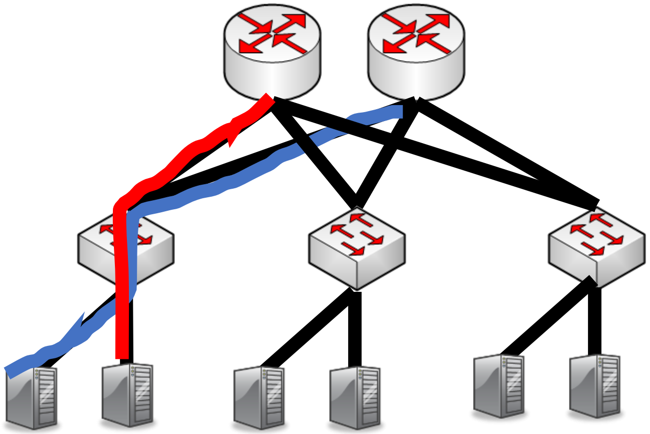
- Links utils low at edge and agg
- Core most utilized
  - Hot-spots exists (> 70% utilization)
  - < 25% links are hotspots
  - Loss occurs on less utilized links (< 70%)
    - Implicating momentary bursts
- Time-of-Day variations exists
  - Variation an order of magnitude larger at core
- Apply these results to evaluate DC design requirements

Assumption 1: Larger Bisection



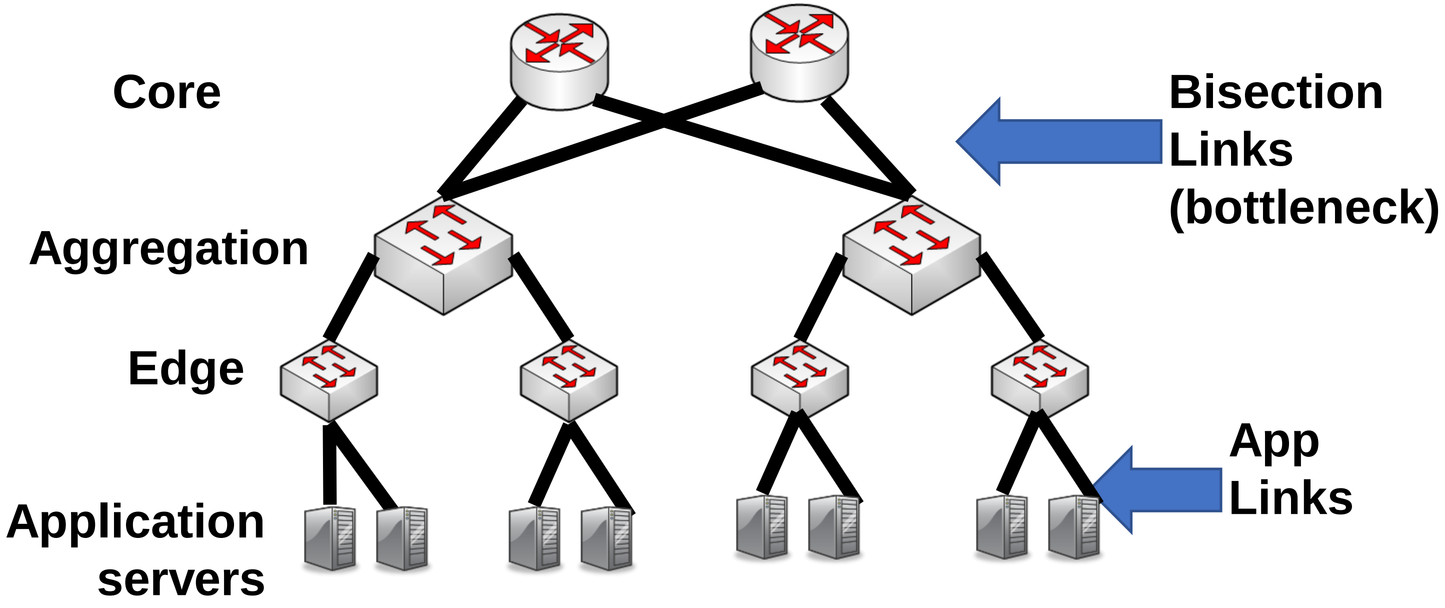
- Need for larger bisection
  - VL2 [Sigcomm ‘09], Monsoon [Presto ‘08],Fat-Tree [Sigcomm ‘08], Portland [Sigcomm ‘09], Hedera [NSDI ’10]
  - Congestion at oversubscribed core links

Argument for Larger Bisection



- Need for larger bisection
  - VL2 [Sigcomm ‘09], Monsoon [Presto ‘08],Fat-Tree [Sigcomm ‘08], Portland [Sigcomm ‘09], Hedera [NSDI ’10]
  - Congestion at oversubscribed core links
  - Increase core links and eliminate congestion

Calculating Bisection Demand



Core

Aggregation

Edge

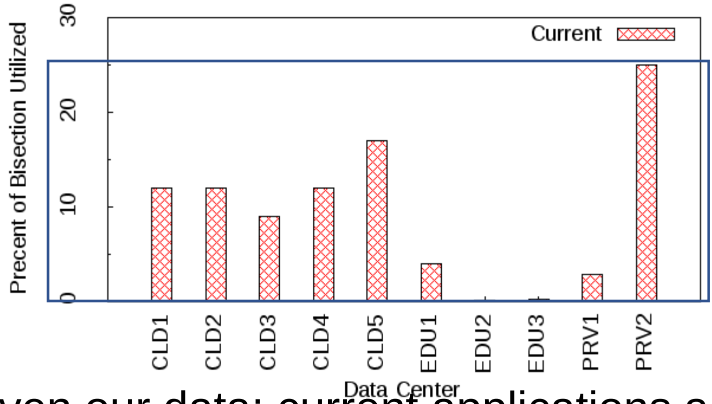
Application servers

Bisection Links (bottleneck)

App Links

If $\left(\frac{\Sigma\ \text{traffic (App)}}{\Sigma\ \text{capacity(Bisection}} \right) > 1$ then more device are needed at the bisection

Bisection Demand



- Given our data: current applications and DC design
  - NO, more bisection is not required
  - Aggregate bisection is only 30% utilized
- Need to better utilize existing network
  - Load balance across paths
  - Migrate VMs across racks

Related Works

- IMC ‘09 [Kandula`09]
  - Traffic is unpredictable
  - Most traffic stays within a rack
- Cloud measurements [Wang’10,Li’10]
  - Study application performance
  - End-2-End measurements

Insights Gained

- 75% of traffic stays within a rack (Clouds)
  - Applications are not uniformly placed
- Half packets are small (< 200B)
  - Keep alive integral in application design
- At most 25% of core links highly utilized
  - Effective routing algorithm to reduce utilization
  - Load balance across paths and migrate VMs
- Questioned popular assumptions
  - Do we need more bisection? No
  - Is centralization feasible? Yes

Are Fat-Trees the last word in datacenter topologies?

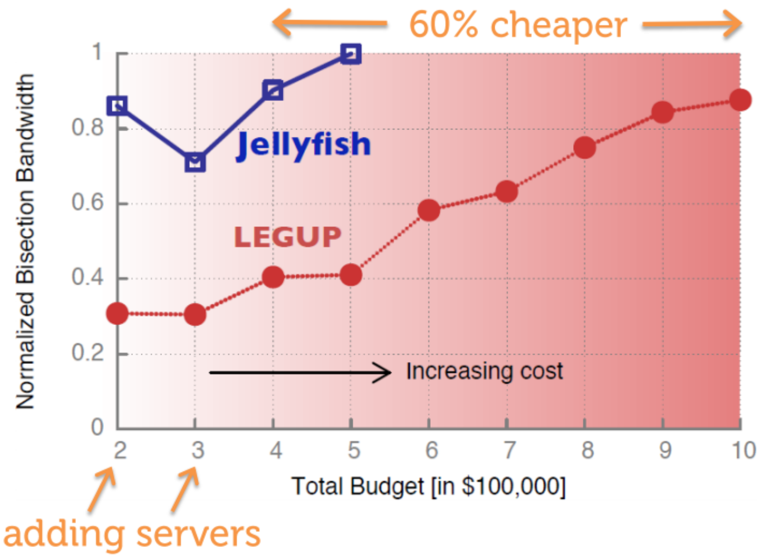
(Figures from original papers/slide decks)

Fat-Tree – Limitations

- Incremental expansion hard
- Structure in networks constrains expansion
  - 3-level Fat-Tree: $5k^2/4$ switches
  - 24 port switches → 3,456 servers
  - 48 port switches → 27,648 servers

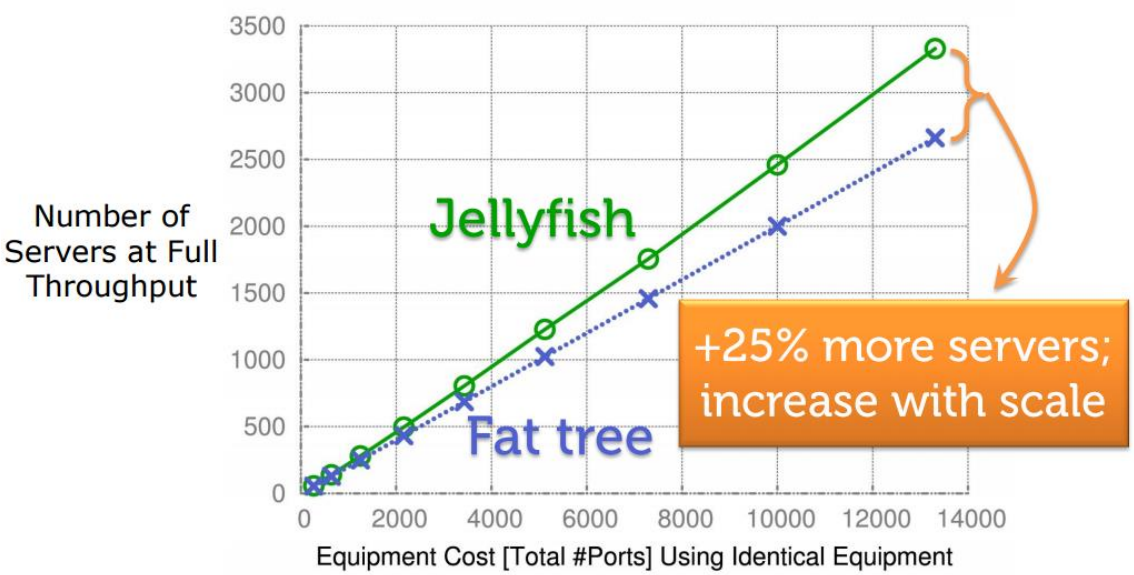
Jellyfish – Randomly Connect ToR Switches

- Same procedure for construction and expansion



LEGUP: [Curtis, Keshav, Lopez-Ortiz, CoNEXT'10]

Jellyfish – Higher Bandwidth than Fat-Trees



+25% more servers; increase with scale

Packet level simulation; random permutation traffic

Jellyfish – Higher Bandwidth than Fat-Trees

If we fully utilize all available capacity …

$$\sum_{\forall \text{links}} \text{capacity}(link)$$

//

$$\text{Number of flows at full throughput (1 Gbps)} = \frac{\cancel{\text{total network capacity}}}{\cancel{\text{capacity used per flow}}}$$

//

1 Gbps • mean path length

Mission:
minimize average path length

Fat-Trees – Limitations

- Perform well in average case
- Core layer can have high-persistence, high-prevalence hotspots

Flyways – Dynamic High Bandwidth Links

- 60GHz low cost wireless technology
- Dynamically inject links where needed

Fat-Trees – Limitations

- High maintenance and cabling costs
- Static topology has low flexibility

Completely Wireless Datacenters

- Cayley (Ji-Yong, Hakim, EGS, Darko Kirovski, ANCS12) uses 60GHz wireless
- Firefly (Hamedazimi et al., SIGCOMM14) and ProjecToR (Ghobadi et al., SIGCOMM16) use free-space optics

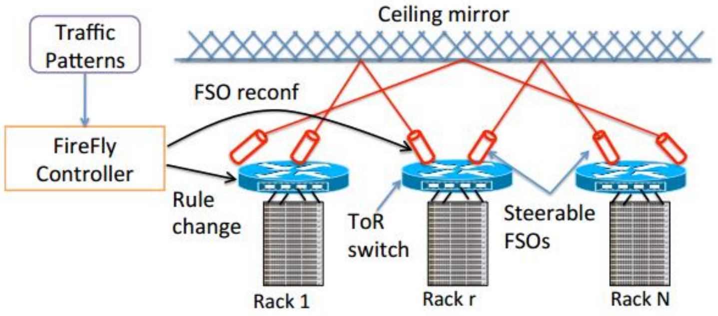


Figure 1: High-level view of the FireFly architecture. The only switches are the Top-of-Rack (ToR) switches.

Source: Hamedazimi et al., SIGCOMM14