Computer Networks

Datacenter Networks

Material based on courses at Princeton, MIT

What are Data Centers?

Large facilities with 10s of thousands of networked servers

- Compute, storage, and networking working in concert
- "Warehouse-Scale Computers"
- Huge investment: ~ 0.5 billion for large datacenter

Data Center Costs

Amortized Cost*	Component	Sub-Components
~45%	Servers	CPU, memory, disk
~25%	Power infrastructure	UPS, cooling, power distribution
~15%	Power draw	Electrical utility costs
~15%	Network	Switches, links, transit

The Cost of a Cloud: Research Problems in Data Center Networks. Sigcomm CCR 2009. Greenberg, Hamilton, Maltz, Patel.

*3 yr amortization for servers, 15 yr for infrastructure; 5% cost of money

Server Costs

30% utilization considered “good” in most data centers!

Uneven application fit

- Each server has CPU, memory, disk: most applications exhaust one resource, stranding the others

Uncertainty in demand

- Demand for a new service can spike quickly

Risk management

- Not having spare servers to meet demand brings failure just when success is at hand

Goal: Agility – Any service, Any Server

Turn the servers into a single large fungible pool

- Dynamically expand and contract service footprint as needed

Benefits

- Lower cost (higher utilization)
- Increase developer productivity
- Achieve high performance and reliability

Achieving Agility

Workload management

- Means for rapidly installing a service's code on a server
- *Virtual machines, disk images, containers*

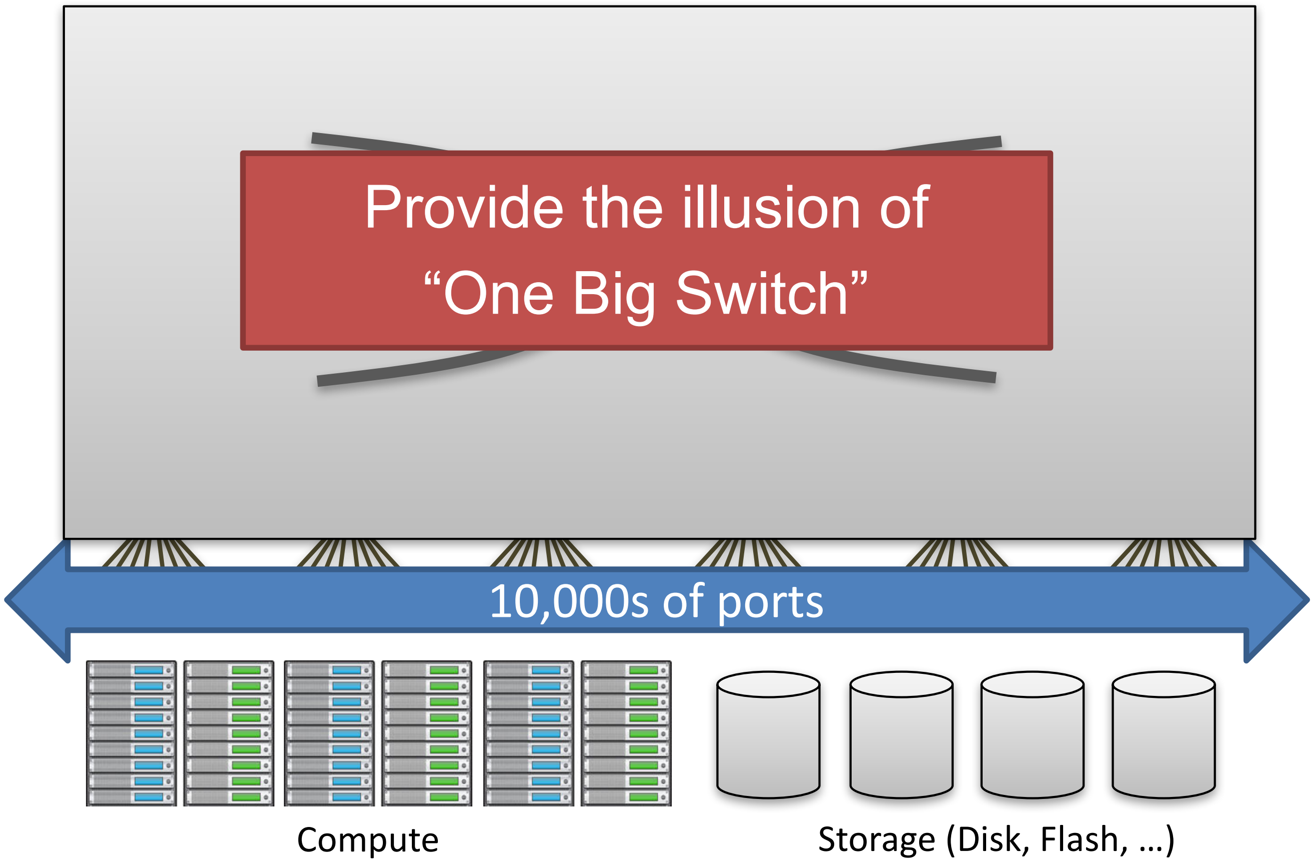
Storage Management

- Means for a server to access persistent data
- *Distributed filesystems (e.g., HDFS, blob stores)*

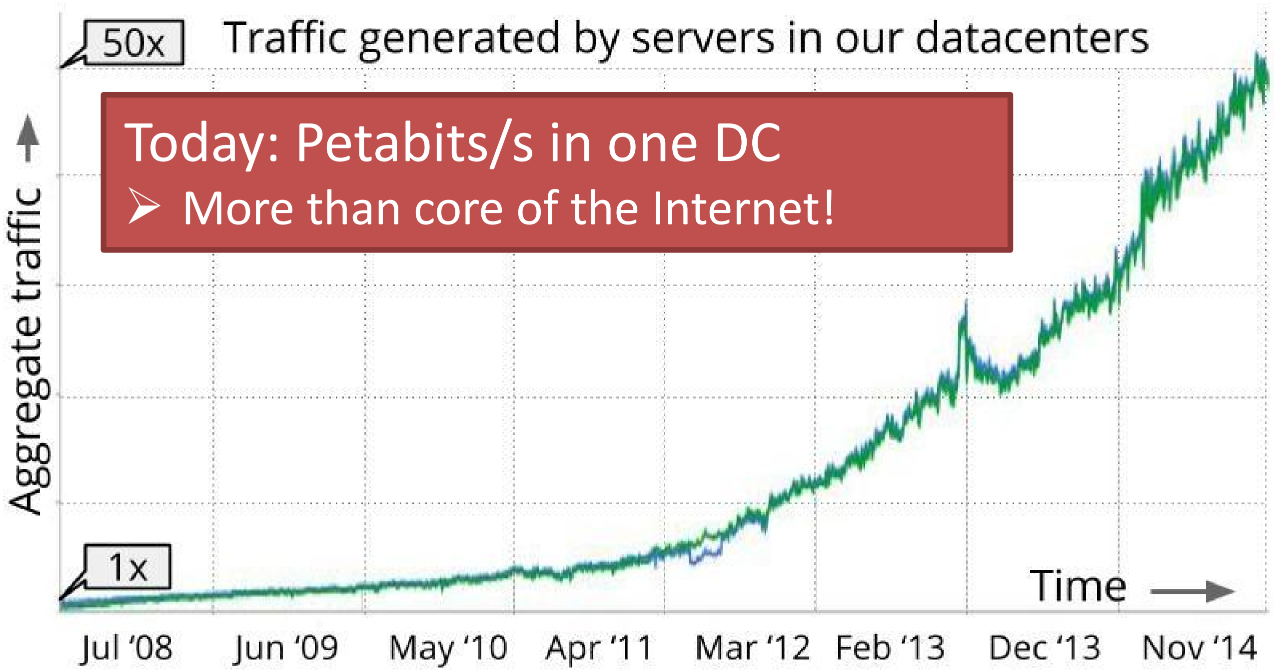
Network

- Means for communicating with other servers, regardless of where they are in the data center

Datacenter Networks



Datacenter Traffic Growth



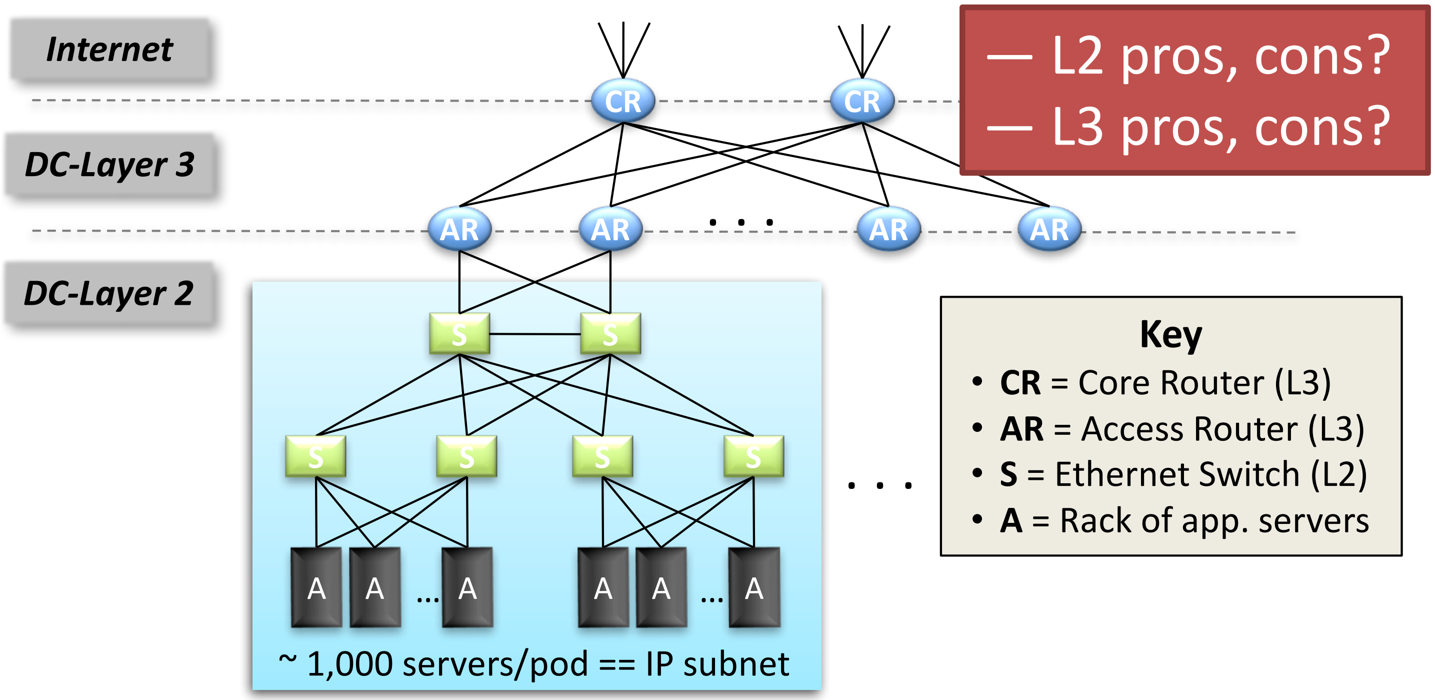
Today: Petabits/s in one DC

- More than core of the Internet!

✧ Source: "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network", SIGCOMM 2015.

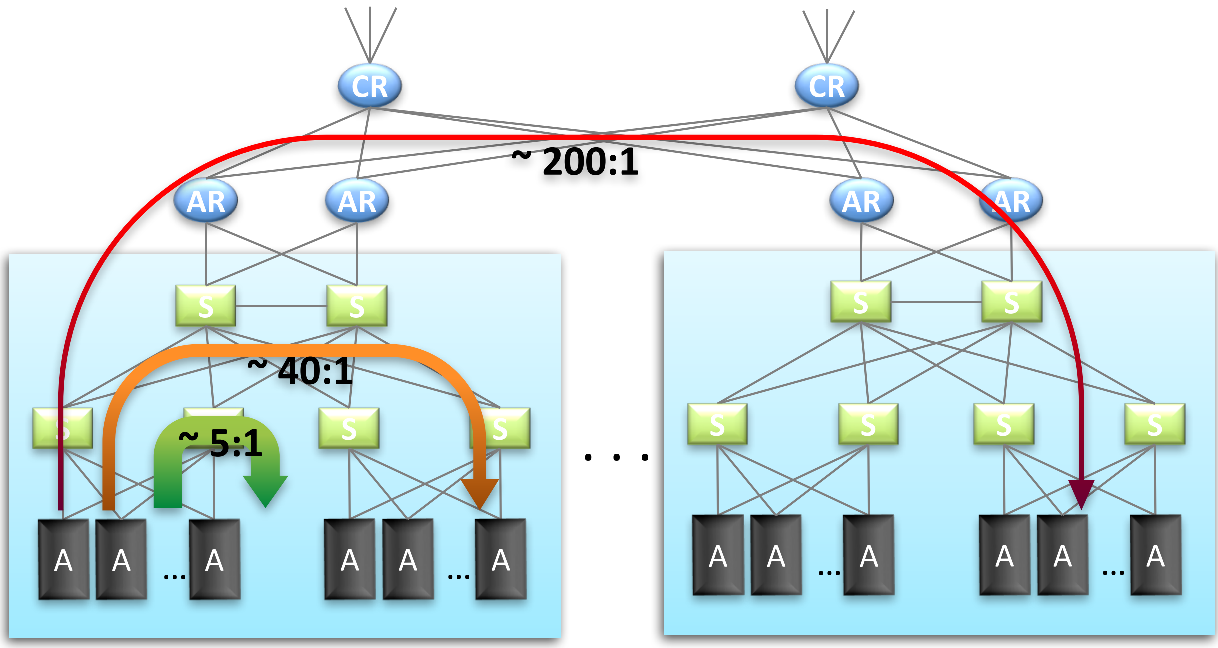
Conventional DC Network Problems

Conventional DC Network



Reference – "Data Center: Load balancing Data Center Services", Cisco 2004

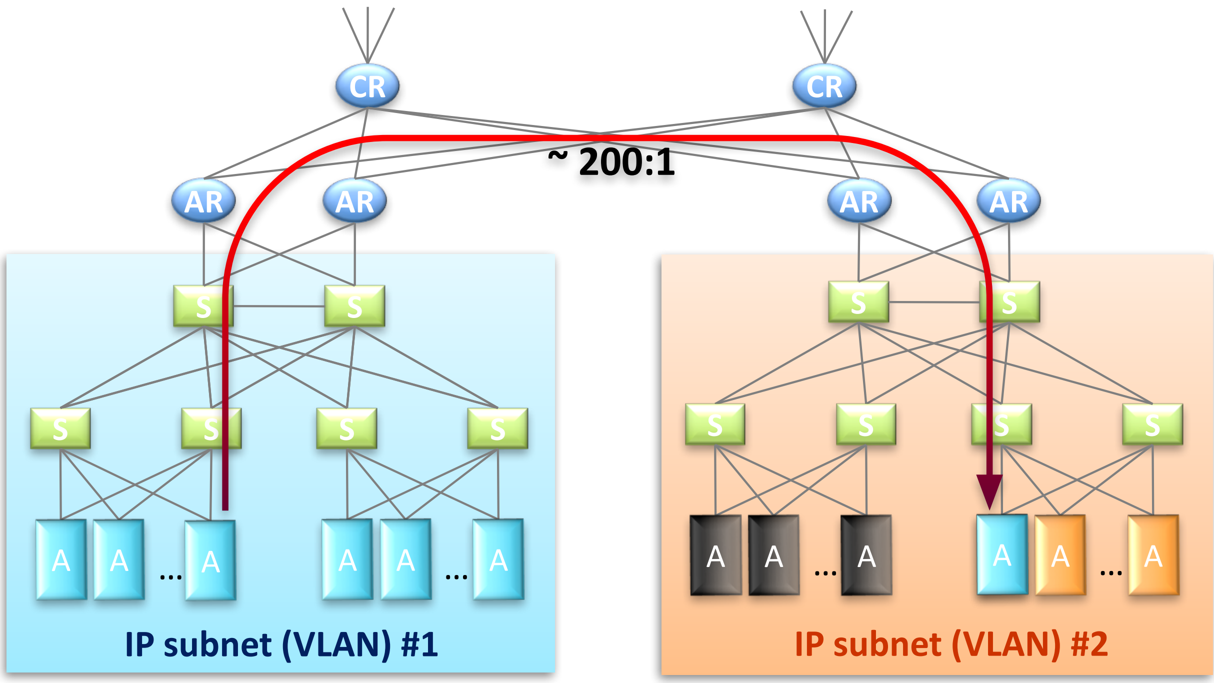
Conventional DC Network Problems



Dependence on high-cost proprietary routers

Extremely limited server-to-server capacity

Conventional DC Network Problems

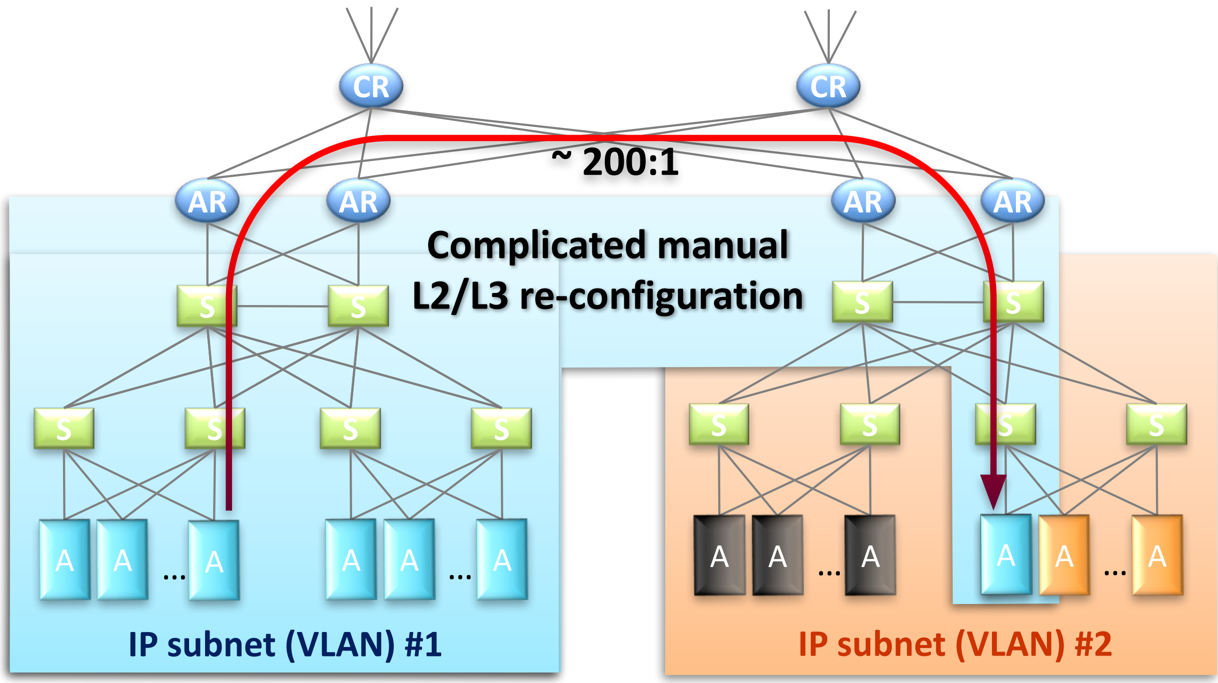


Dependence on high-cost proprietary routers

Extremely limited server-to-server capacity

Resource fragmentation

And More Problems …

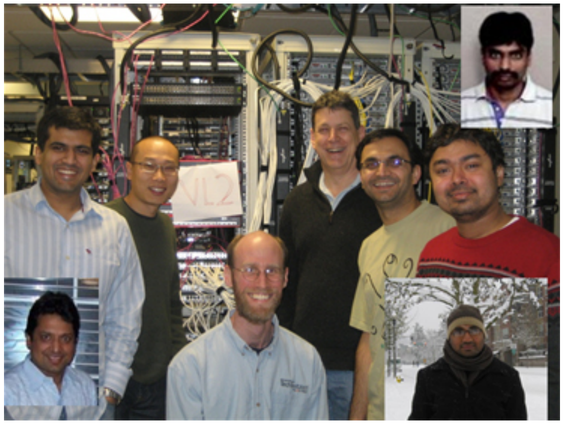


Poor reliability

Lack of performance isolation

VL2 Paper

Measurements



http://research.microsoft.com/en-US/news/features/datacenternetworking-081909.aspx

VL2 Design

- Clos topology
- Valiant LB
- Name/location separation
  (precursor to network virtualization)

Measurements

DC Traffic Characteristics

Instrumented a large cluster used for data mining and identified distinctive traffic patterns

Traffic patterns are **highly volatile**

- A large number of distinctive patterns even in a day

Traffic patterns are **unpredictable**

- Correlation between patterns very weak

Traffic-aware optimization needs to be done frequently and rapidly

DC Opportunities

DC controller knows **everything** about **hosts**

Host OS's are easily **customizable**

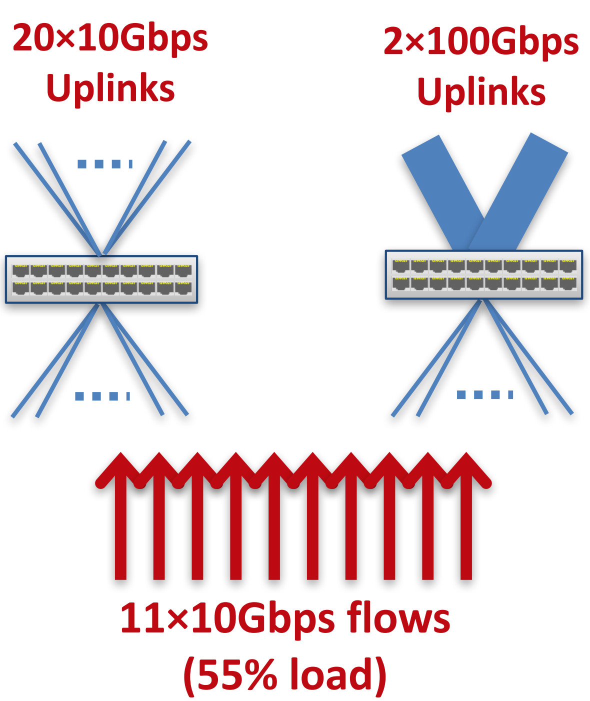
Probabilistic flow distribution would work well enough, because …

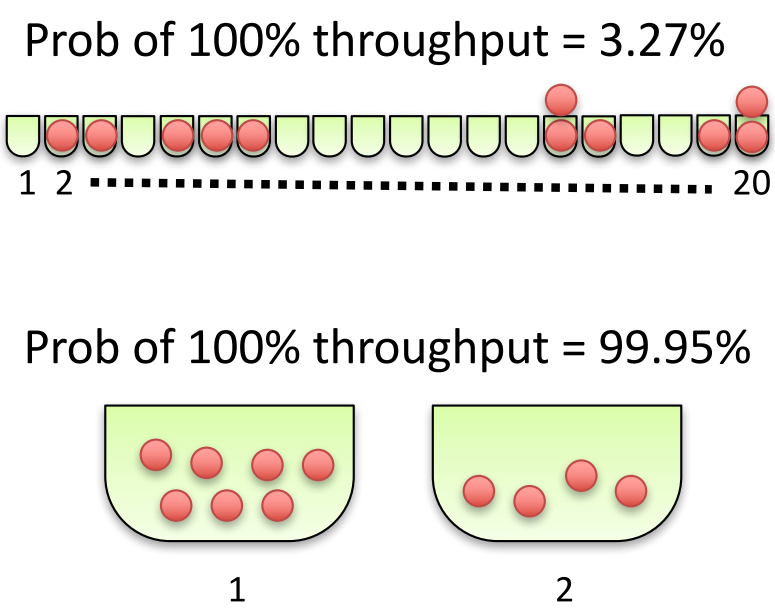
- Flows are numerous and not huge – few elephants
- Commodity switch-to-switch links are substantially thicker (~ 10x) than the maximum thickness of a flow

DC network can be made simple

Intuition

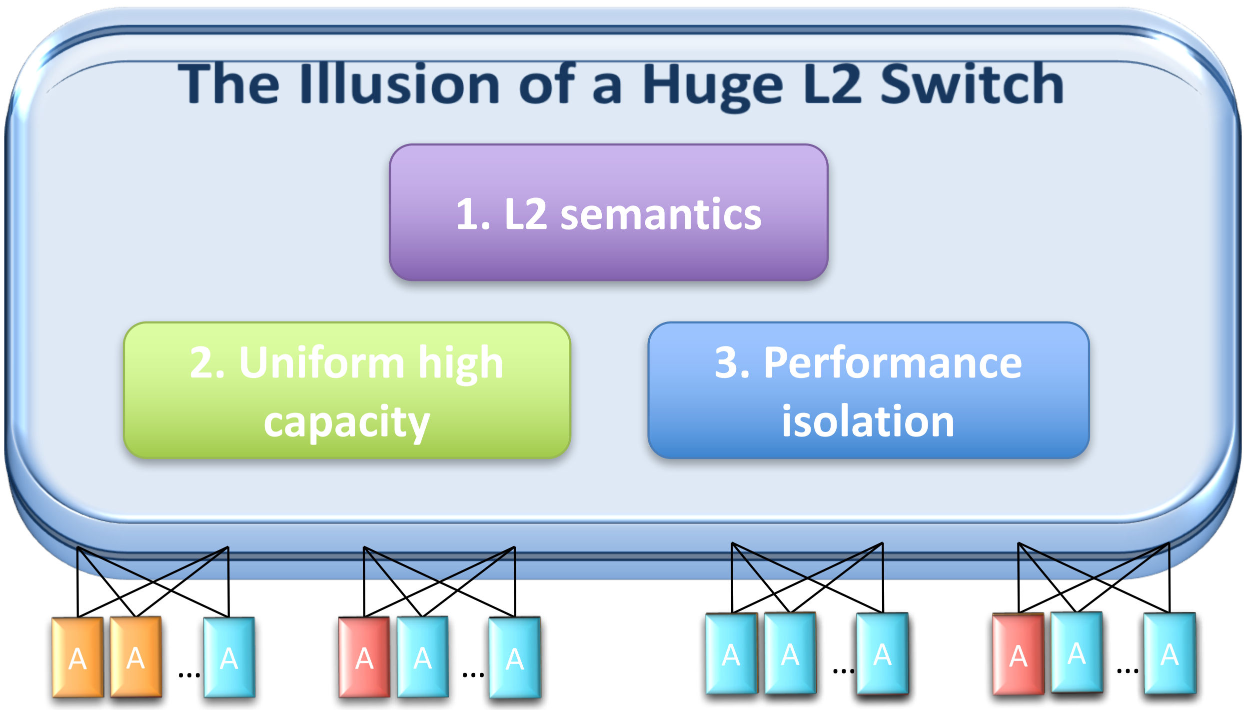
Higher speed links improve *flow-level* load balancing (ECMP)





Virtual Layer 2

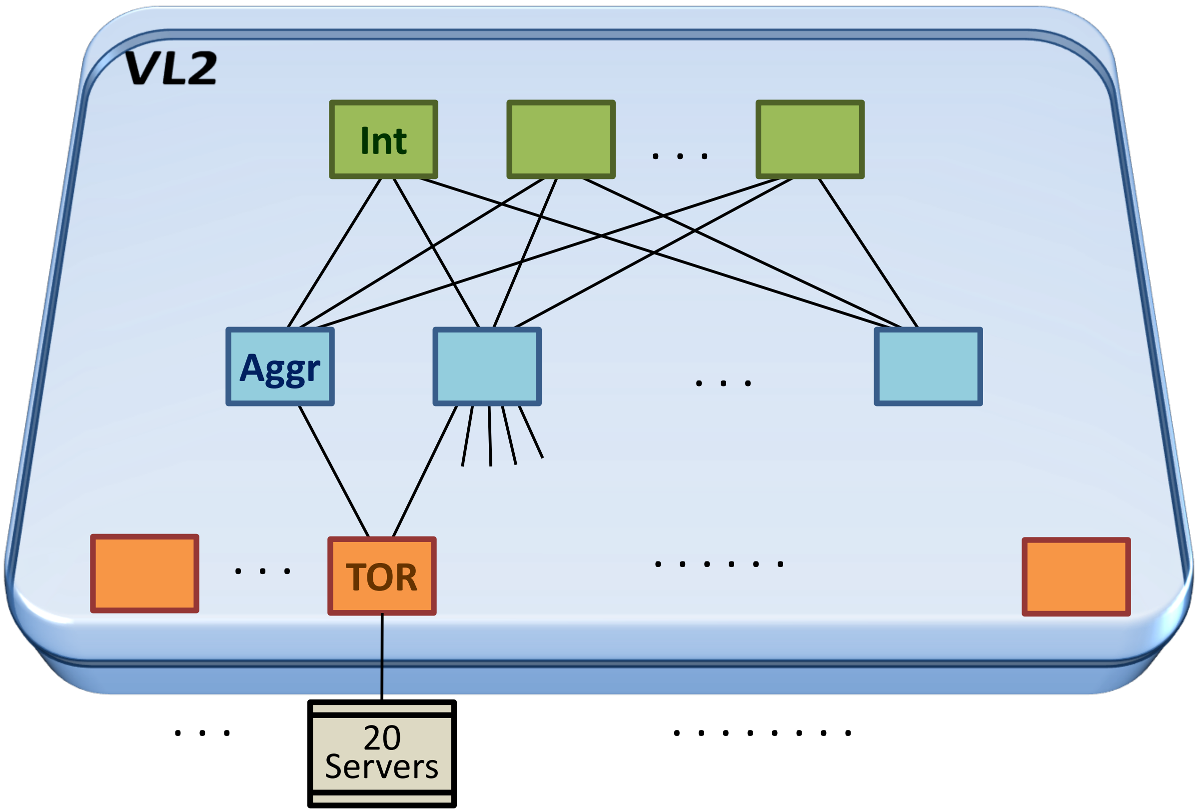
VL2 Goals



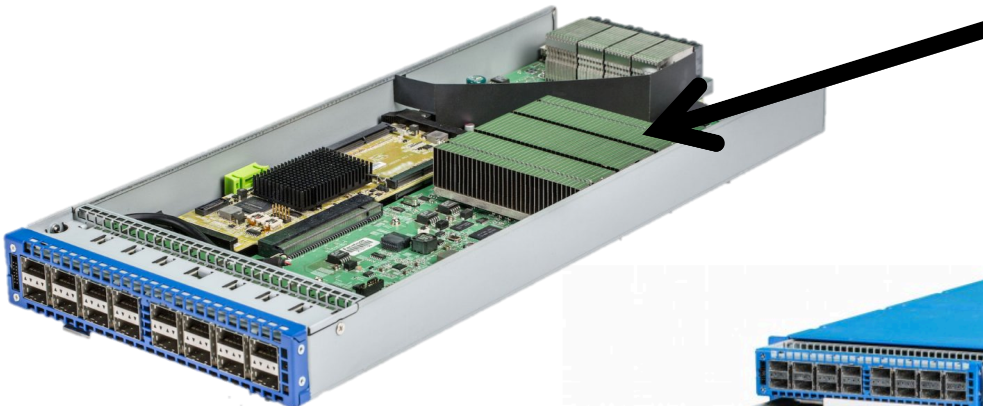
The Illusion of a Huge L2 Switch
1. L2 semantics
2. Uniform high capacity
3. Performance isolation
A
A
…
A
A
A
…
A
A
A
…
A
A
A
…
A

Clos Topology

Offer huge capacity via multiple paths (scale out)



Building Block:
Merchant Silicon Switching Chips



Switch ASIC

Facebook Wedge

6 pack

✧ Image courtesy of Facebook

VL2 Design Principles

Randomizing to Cope with Volatility

- Tremendous variability in traffic matrices

Separating Names from Locations

- Any server, any service

Embracing End Systems

- Leverage the programmability & resources of servers
- Avoid changes to switches

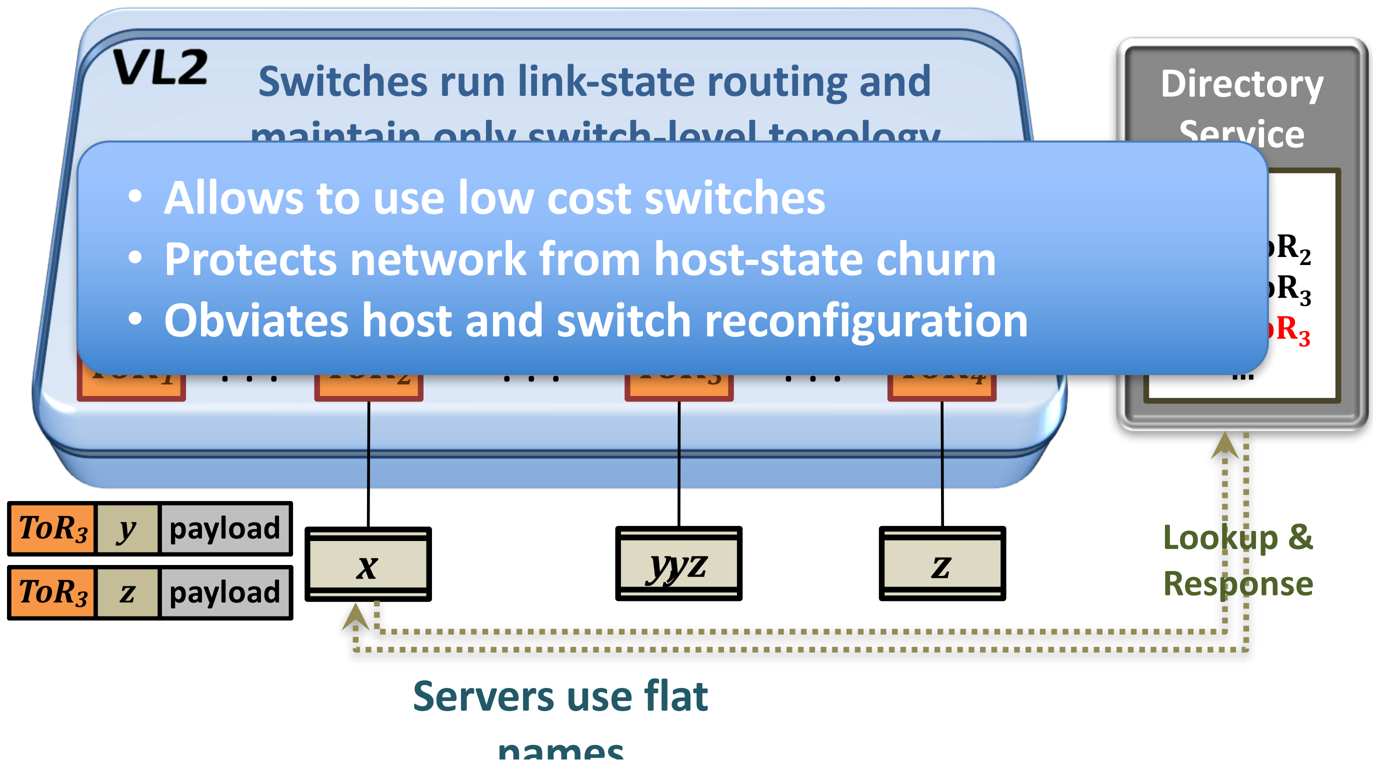
Building on Proven Networking Technology

- Build with parts shipping today
- Leverage low cost, powerful merchant silicon ASICs

VL2 Goals and Solutions

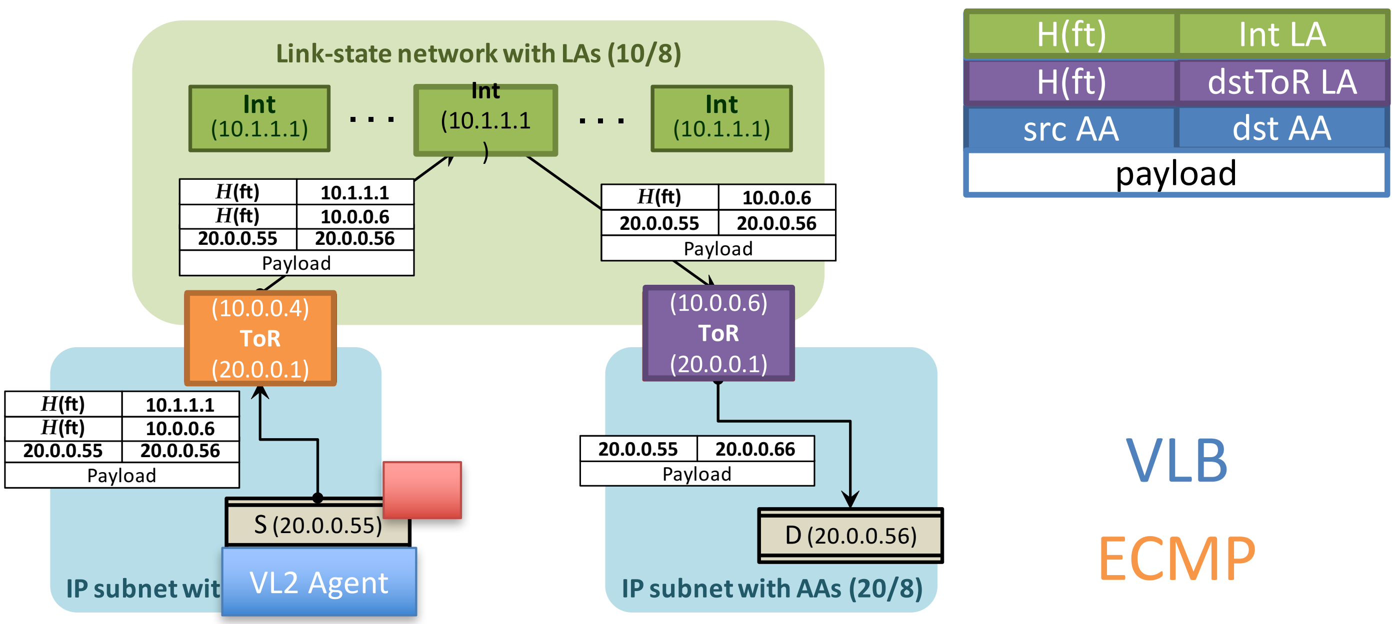
Objective	Approach	Solution
1. Layer-2 semantics	Employ flat addressing	Name-location separation & resolution service
2. Uniform high capacity between servers	Guarantee bandwidth for hose-model traffic	Flow-based random traffic indirection (Valiant LB)
3. Performance Isolation	Enforce hose model using existing mechanisms only	TCP

Addressing and Routing: Name-Location Separation



- Allows to use low cost switches
- Protects network from host-state churn
- Obviates host and switch reconfiguration

VL2 Agent in Action



Why use hash for Src IP?
Why anycast & double encap?

Other details

How does L2 broadcast work?

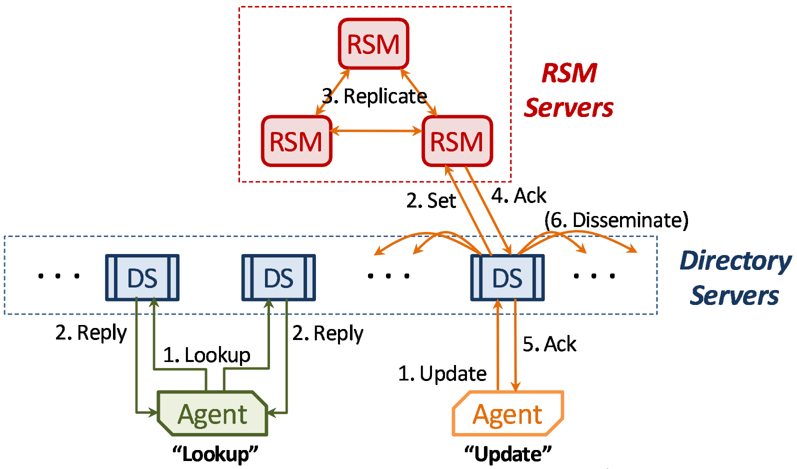
How does Internet communication work?

VL2 Directory System

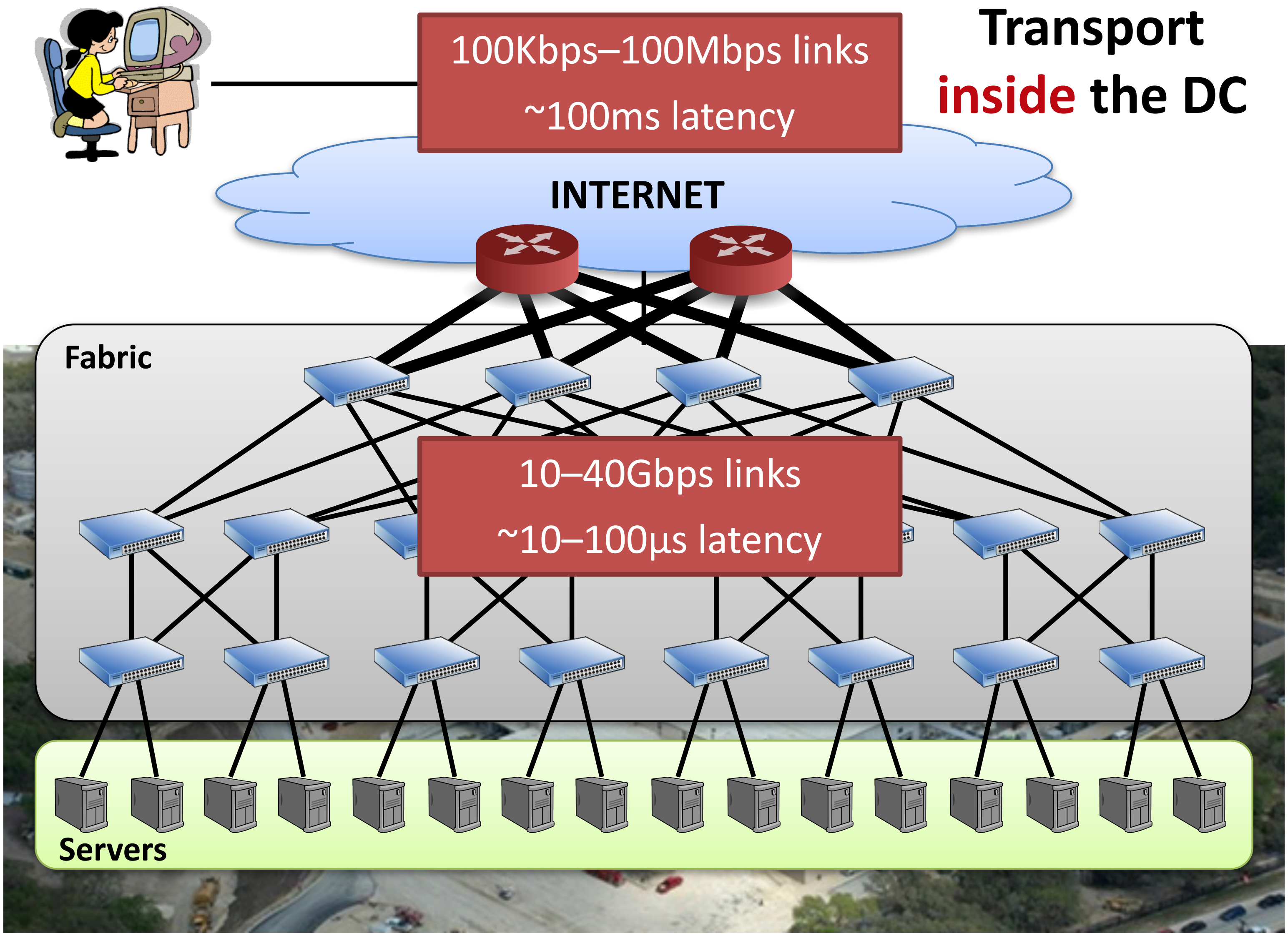
Read-optimized Directory Servers for lookups

Write-optimized
Replicated State
Machines for updates

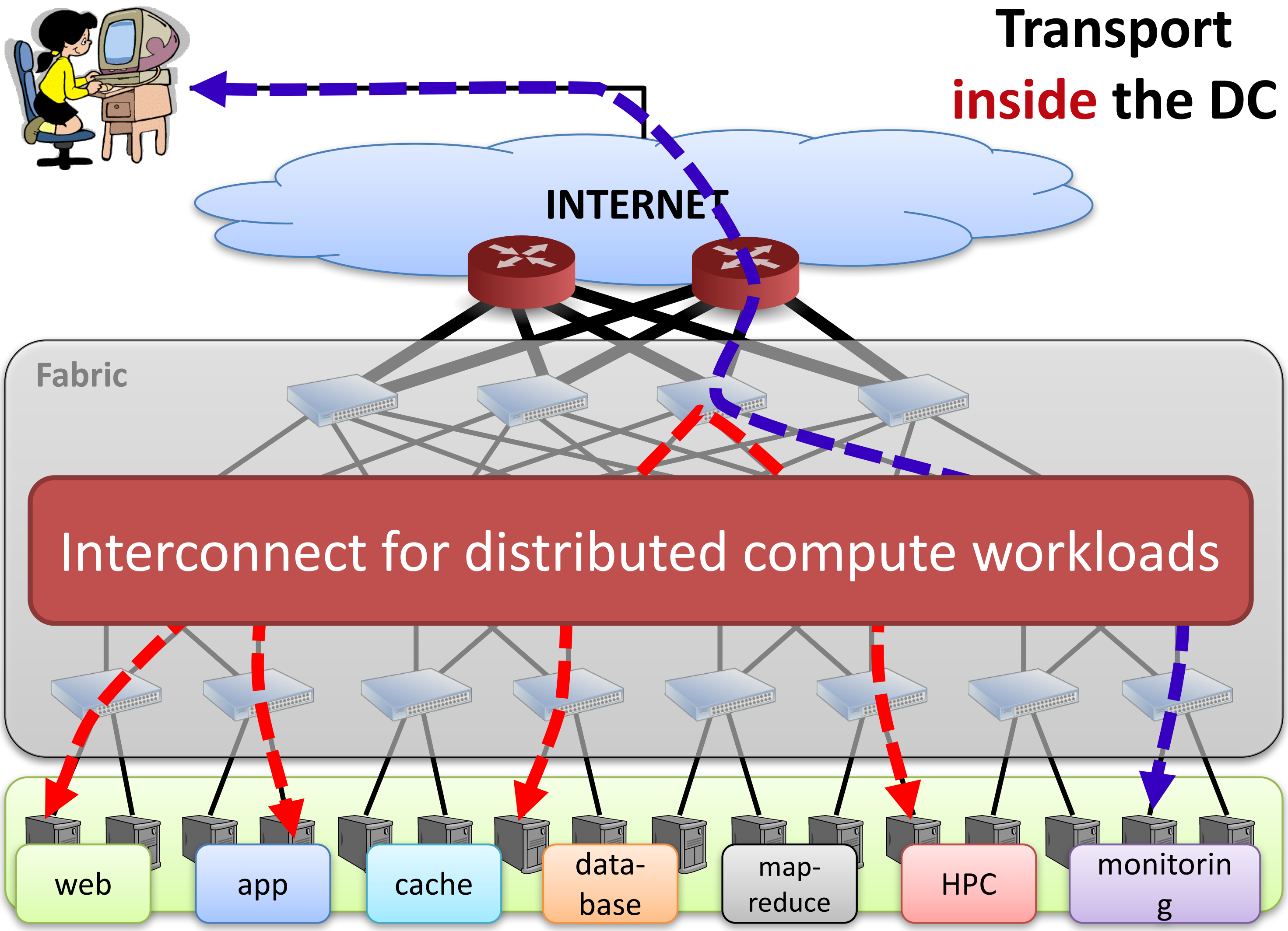
Stale mappings?



Data Center Congestion Control


100Kbps–100Mbps links
~100ms latency
Transport
inside the DC
INTERNET
Fabric
10–40Gbps links
~10–100μs latency
Servers

Transport inside the DC



What's Different About DC Transport?

Network characteristics
- Very high link speeds (Gb/s); very low latency (microseconds)

Application characteristics
- Large-scale distributed computation

Challenging traffic patterns
- Diverse mix of mice & elephants
- Incast

Cheap switches
- Single-chip shared-memory devices; shallow buffers

Data Center Workloads

Mice & Elephants

Short messages

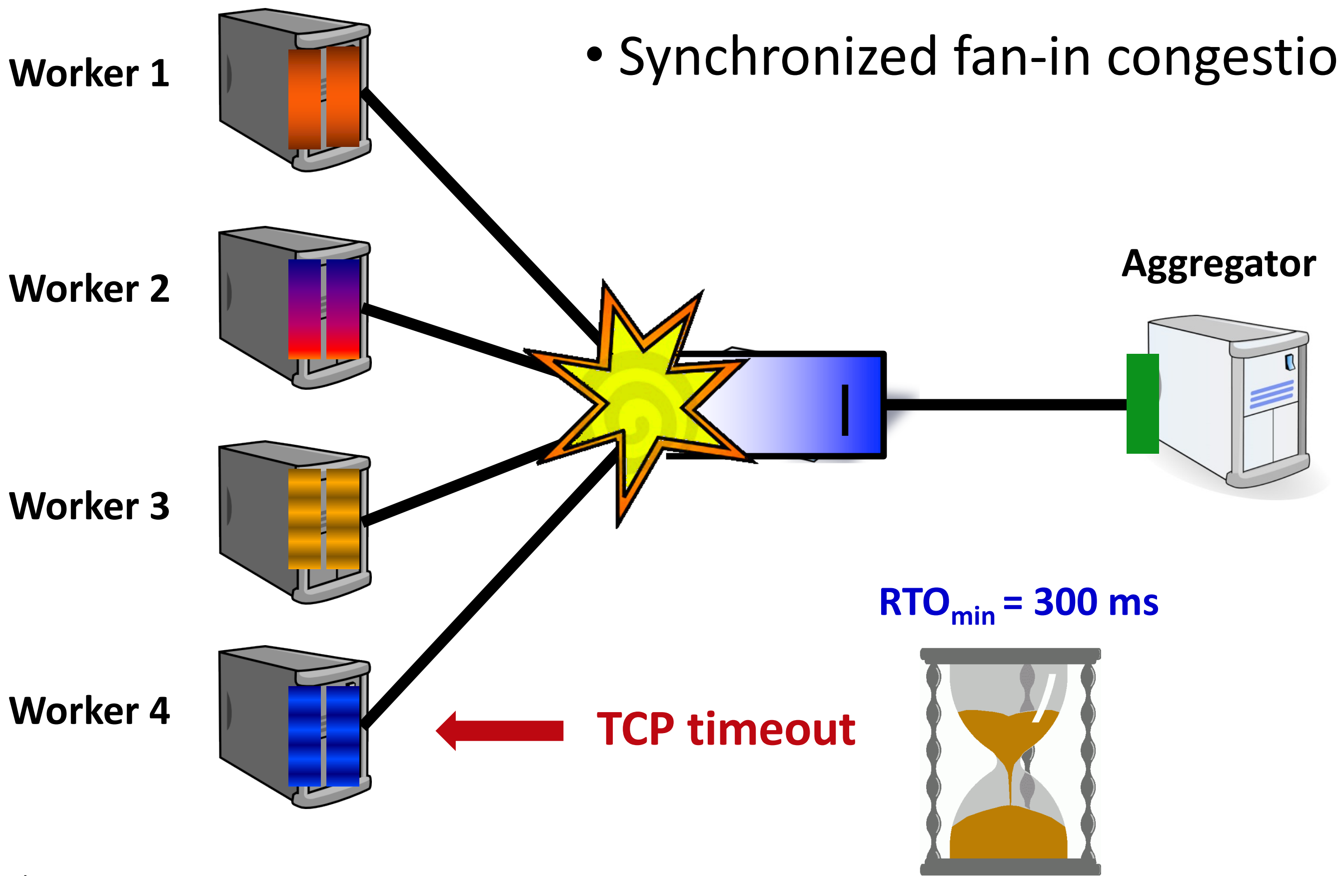
(e.g., query, coordination) → **Low Latency**

Large flows

(e.g., data update, backup) → **High Throughput**

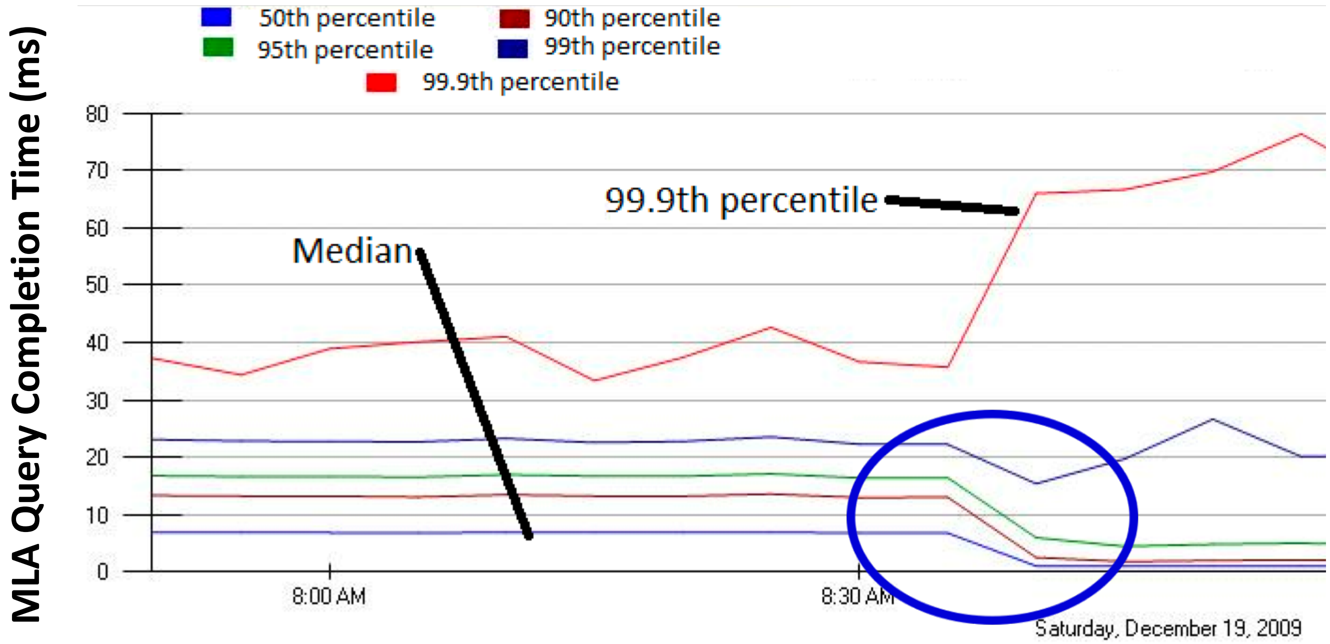
Incast

- Synchronized fan-in congestion



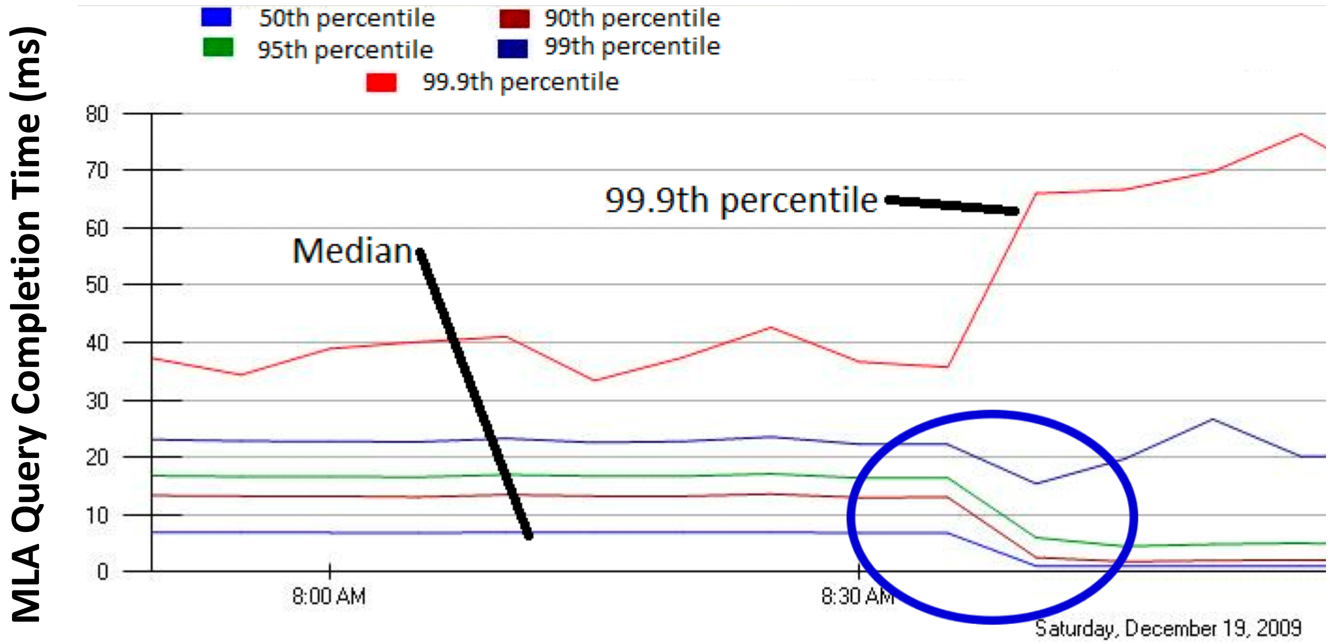
✧ Vasudevan et al. (SIGCOMM'09)

Incast in Bing



Jittering trades of median for high percentiles

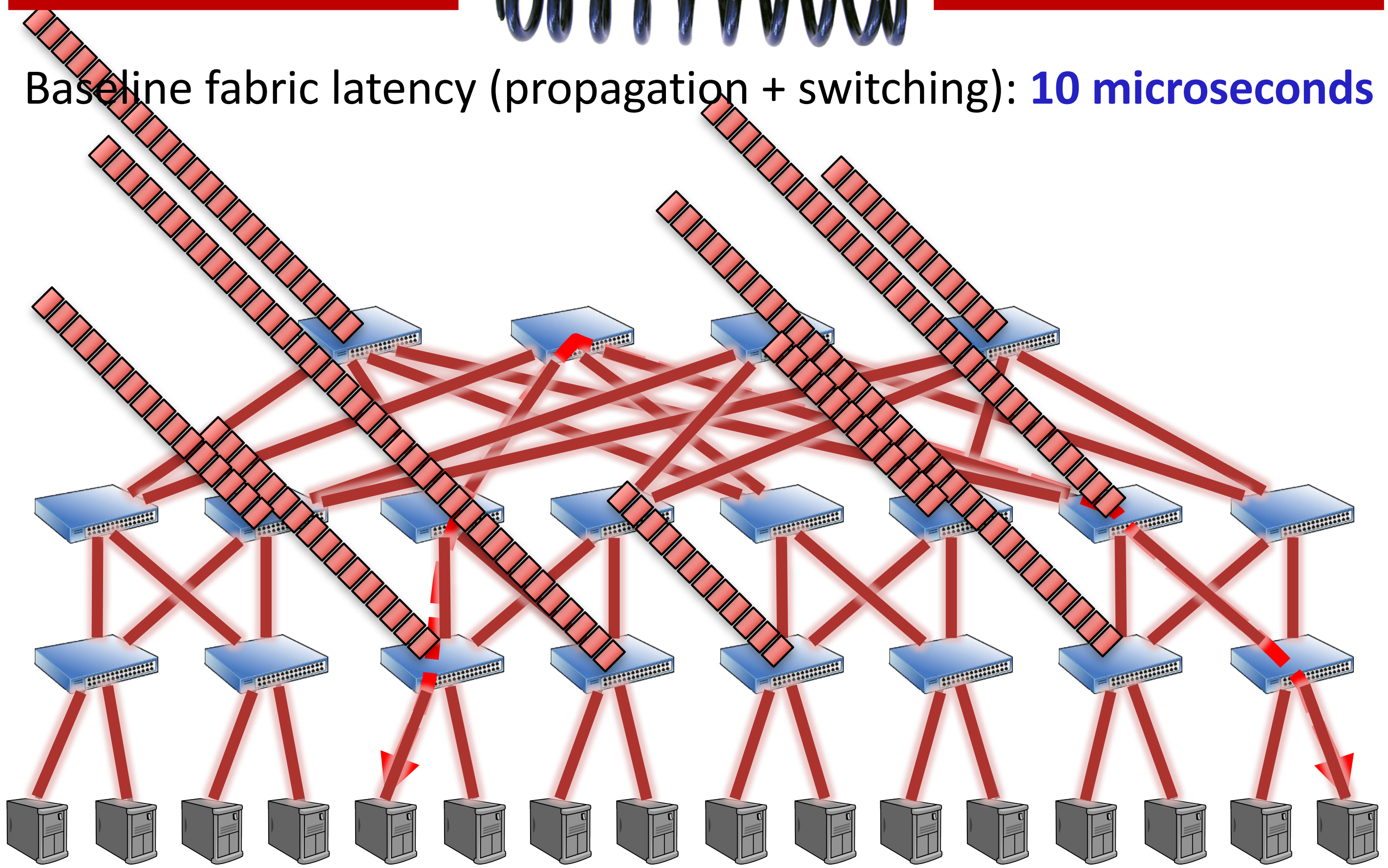
Incast in Bing



Jittering trades of median for high percentiles

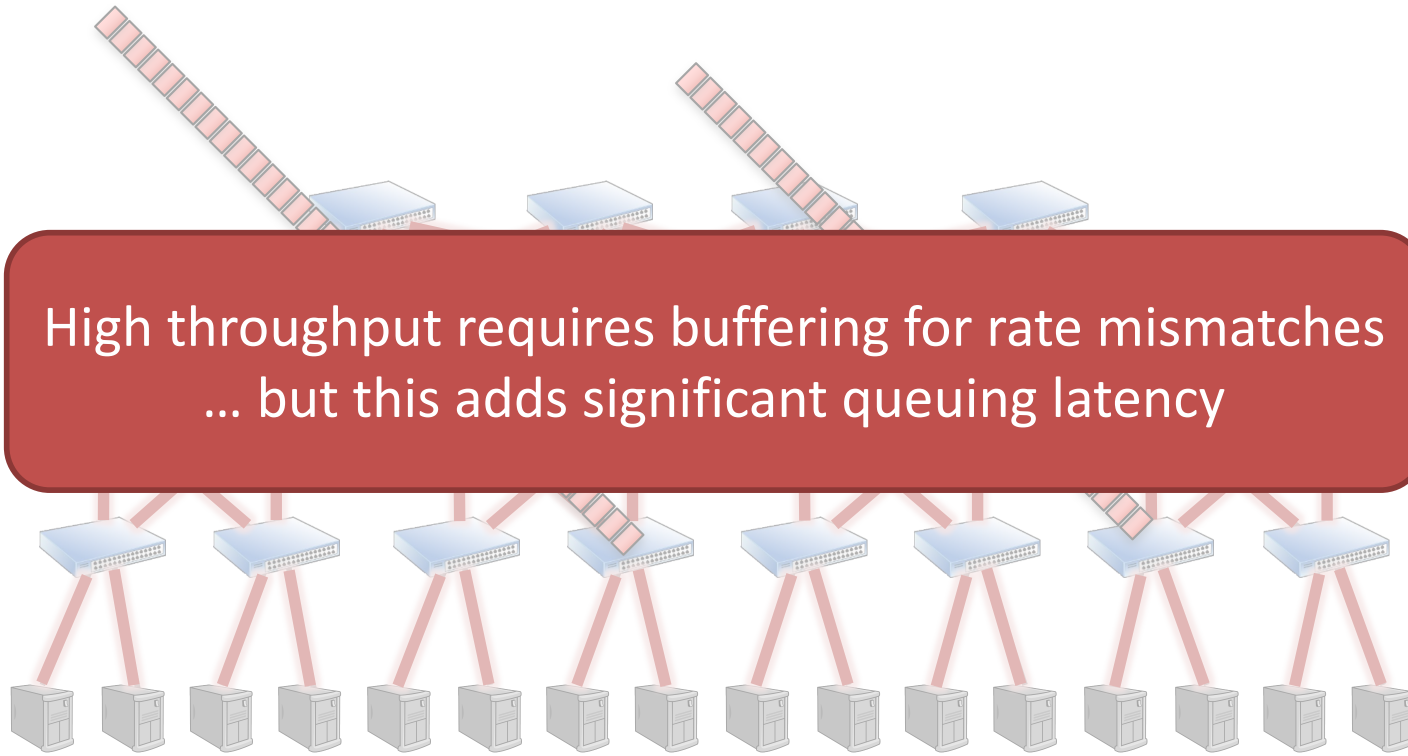
High Throughput  **Low Latency**

Baseline fabric latency (propagation + switching): **10 microseconds**



High Throughput  **Low Latency**

Baseline fabric latency (propagation + switching): **10 microseconds**



High throughput requires buffering for rate mismatches
… but this adds significant queuing latency

Data Center TCP

TCP in the Data Center

TCP [Jacobsen et al.'88] is widely used in the data center

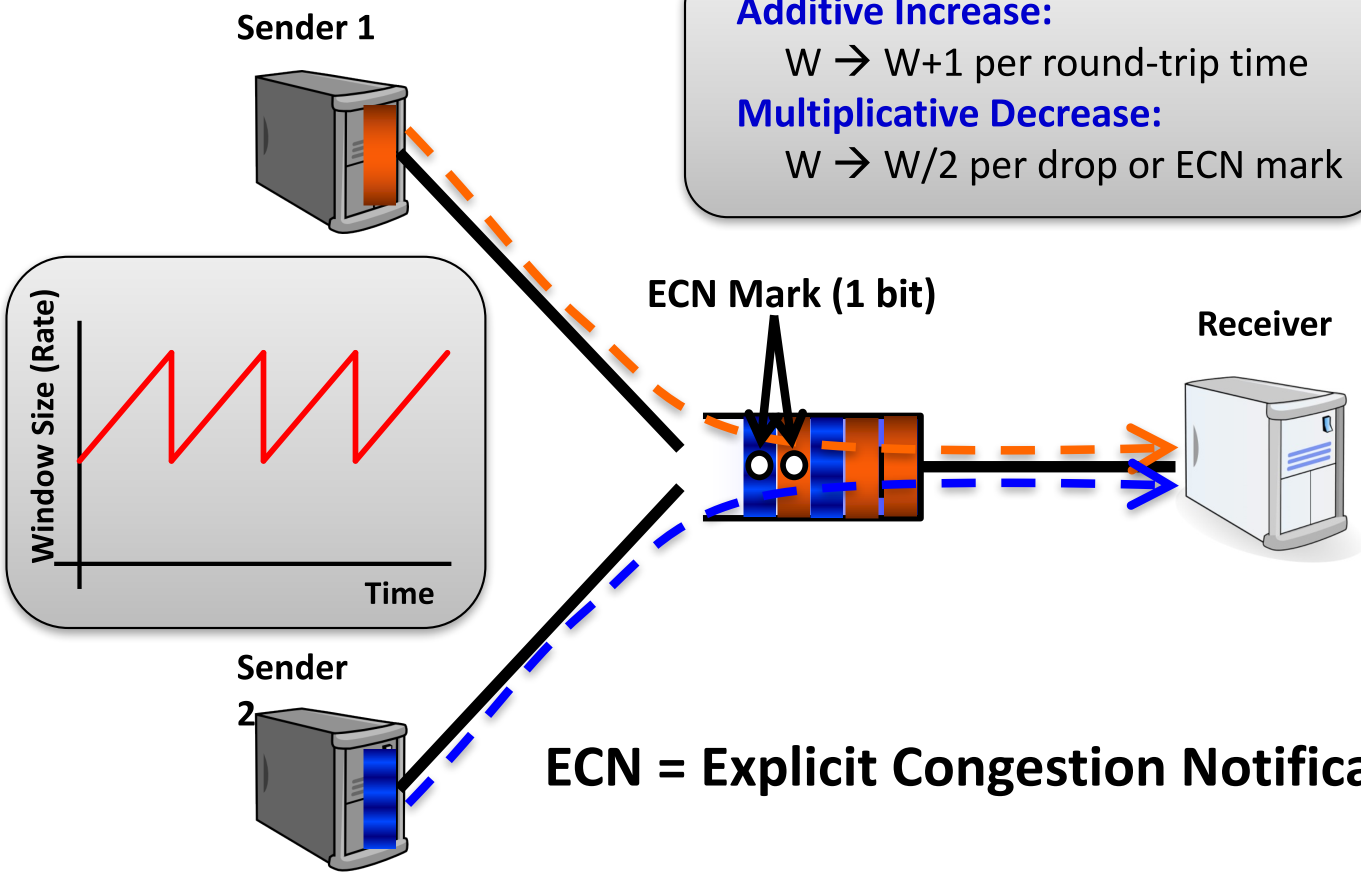
- More than **99%** of the traffic

Operators work around TCP problems

- Ad-hoc, inefficient, often expensive solutions
- TCP is deeply ingrained in applications

Practical deployment is hard
→ keep it simple!

Review: The TCP Algorithm



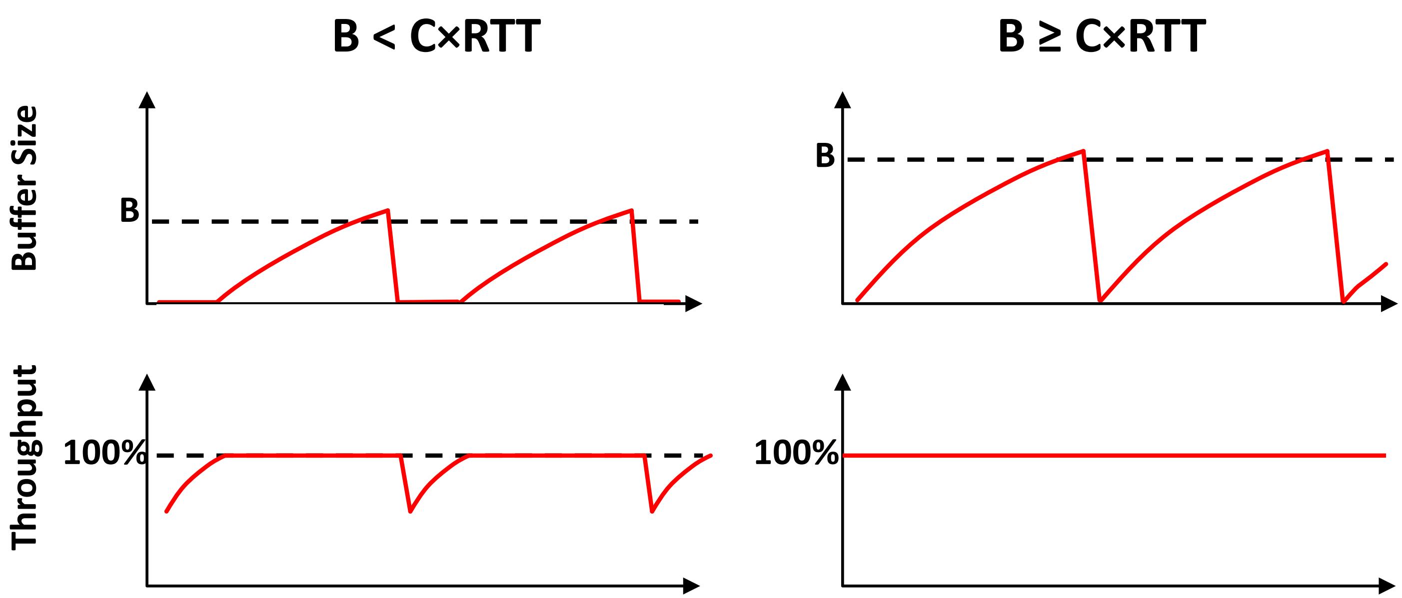
Additive Increase:
W → W+1 per round-trip time
Multiplicative Decrease:
W → W/2 per drop or ECN mark

ECN = Explicit Congestion Notification

TCP Buffer Requirement

Bandwidth-delay product rule of thumb:

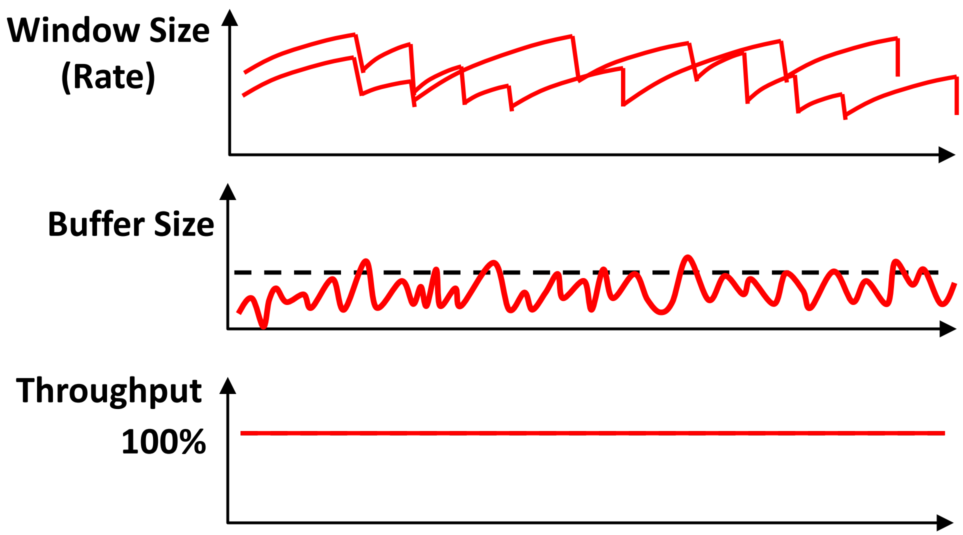
- A single flow needs **C×RTT** buffers for **100% Throughput.**



Reducing Buffer Requirements

Appenzeller et al. (SIGCOMM '04):

– Large # of flows: $C \times RTT/\sqrt{N}$ is enough.



Reducing Buffer Requirements

Appenzeller et al. (SIGCOMM '04):

– Large # of flows: $\boldsymbol{C} \times \boldsymbol{RTT}/\sqrt{\boldsymbol{N}}$ is enough

Can't rely on stat-mux benefit in the DC.

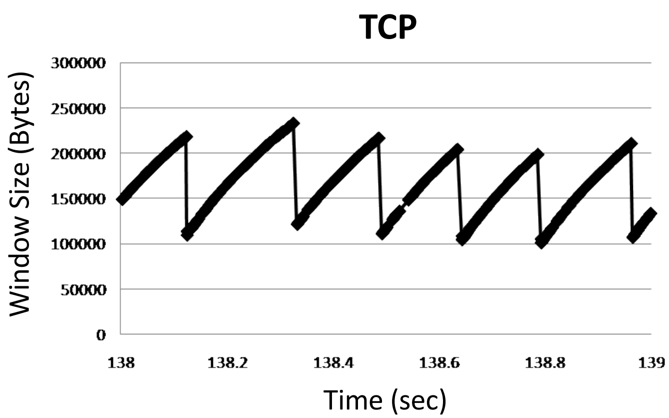
– Measurements show typically **only 1-2 large flows** at each server

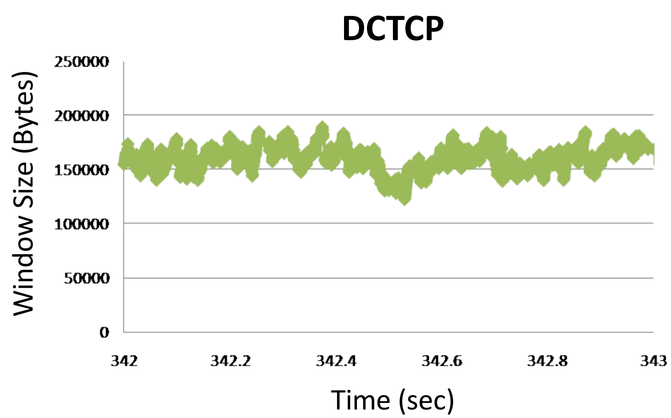
Key Observation:
Low variance in sending rate → Small buffers suffice

DCTCP: Main Idea

- Extract multi-bit feedback from single-bit stream of ECN marks
  - Reduce window size based on **fraction** of marked packets.

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%

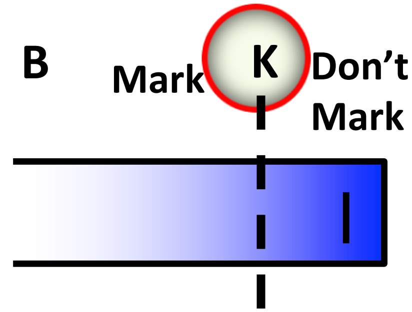




DCTCP: Algorithm

Switch side:

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



Sender side:

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

$$\text{each RTT}: \ F = \frac{\#\ \text{of marked ACKs}}{\text{Total}\ \#\ \text{of ACKs}} \ \Rightarrow \ \alpha \leftarrow (1-g)\alpha + gF$$

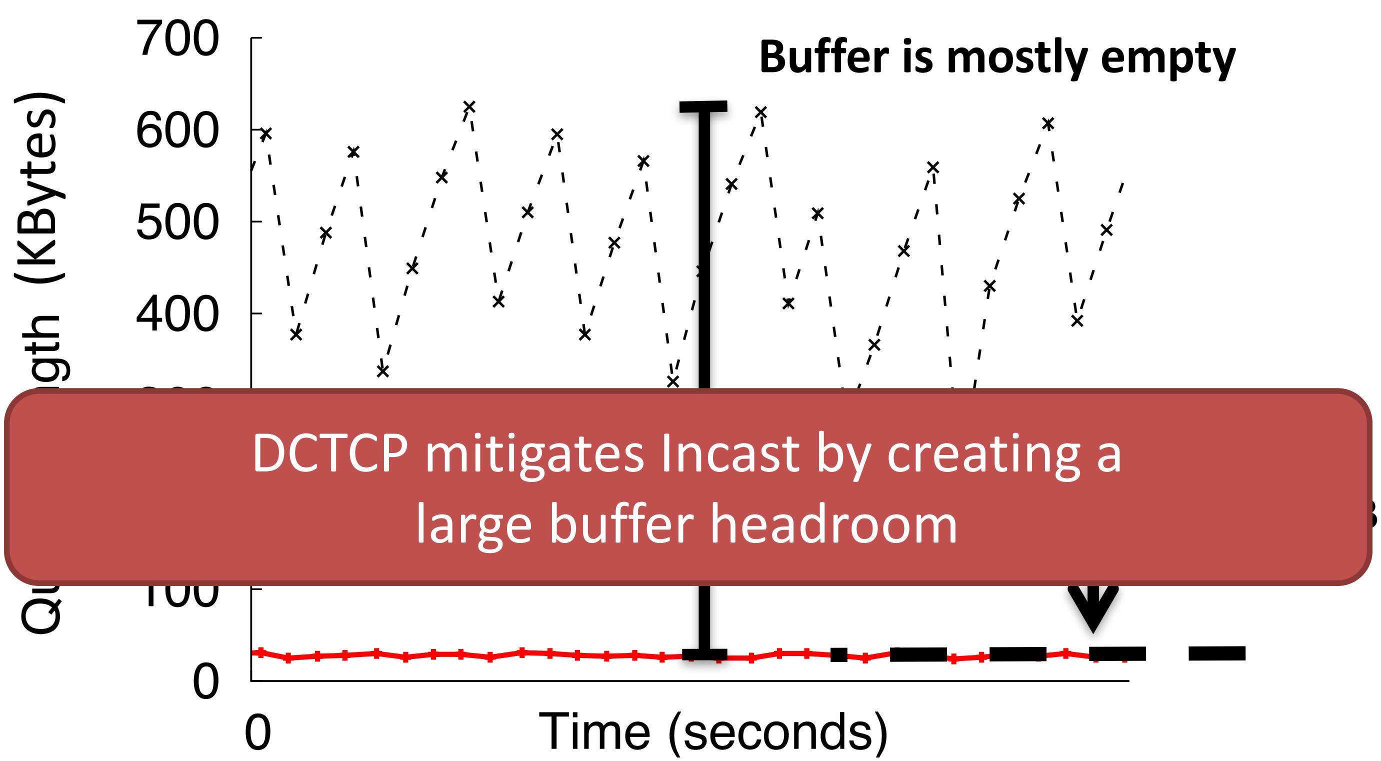
$$W \leftarrow (1 - \frac{\alpha}{2})W$$

➢ **Adaptive window decreases:**

- Note: decrease factor between 1 and 2.

DCTCP vs TCP

Experiment: 2 flows (Win 7 stack), Broadcom 1Gbps Switch



DCTCP mitigates Incast by creating a large buffer headroom

Why it Works

1. Low Latency

- ✓ **Small buffer occupancies** → low queuing delay

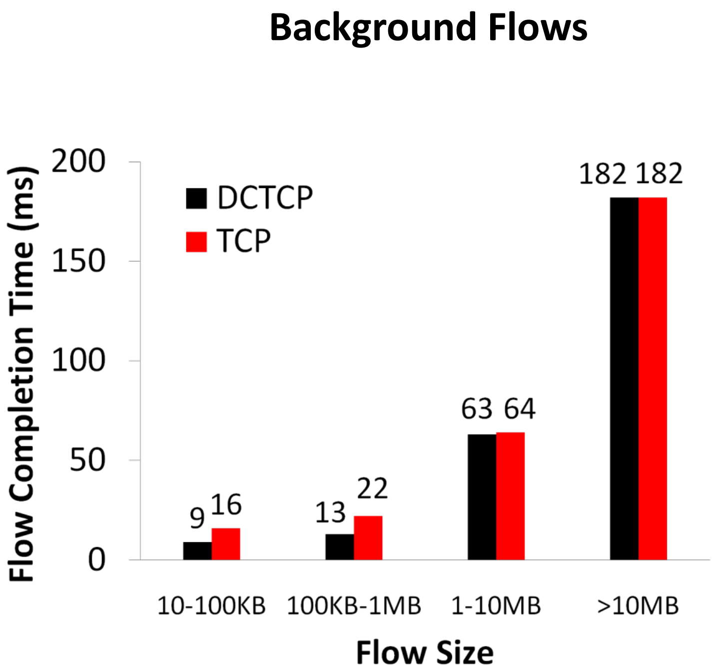
2. High Throughput

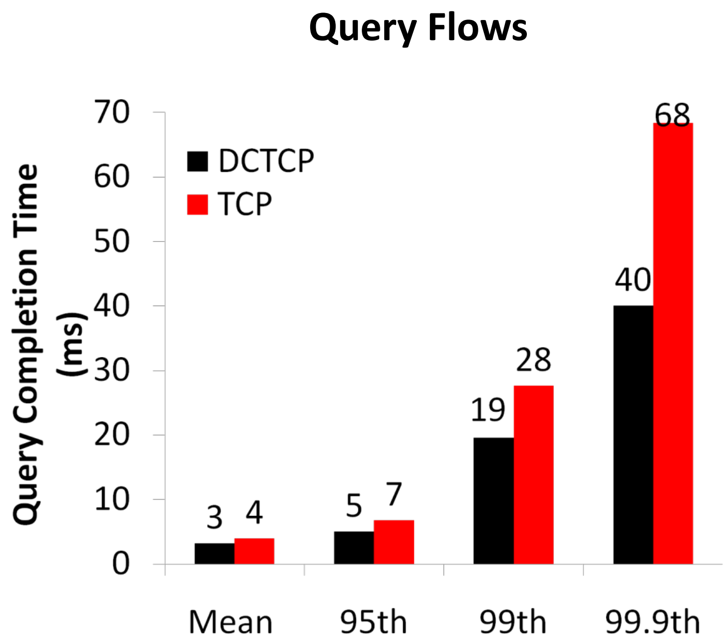
- ✓ **ECN averaging** → smooth rate adjustments, low variance

3. High Burst Tolerance

- ✓ **Large buffer headroom** → bursts fit
- ✓ **Aggressive marking** → sources react before packets are dropped

Bing Benchmark (baseline)





Bing Benchmark (scaled 10x)