Introduction to Computer Networks

Data Centers

Computer Science & Engineering
UNIVERSITY of WASHINGTON

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
- Flexibility service management
  - Resiliency: isolate failure of servers and storage
  - Workload movement: move work to other locations

2

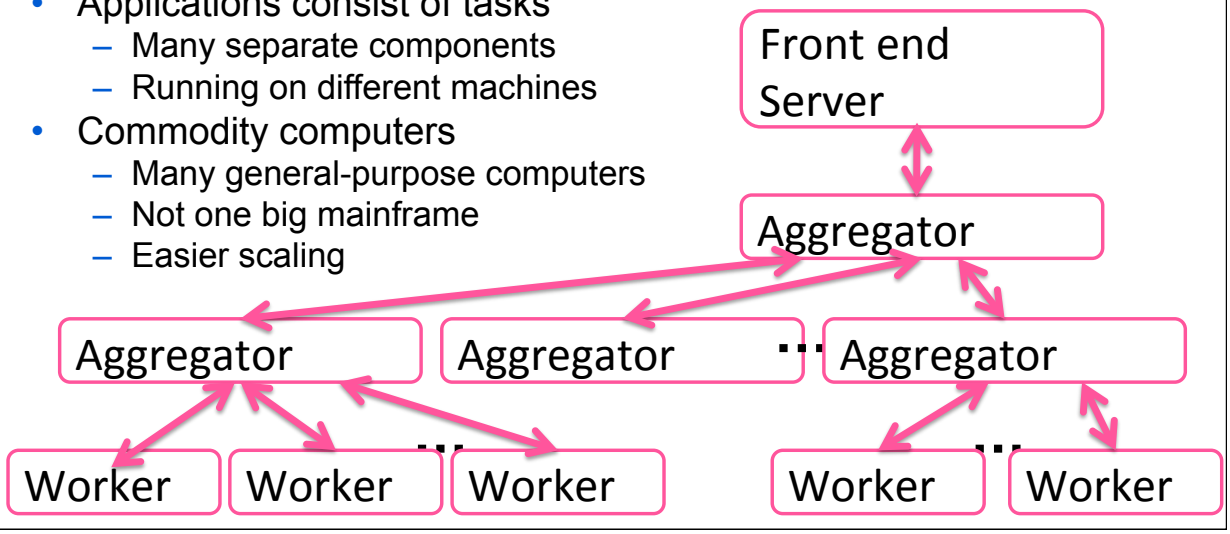
Cloud Service Models

- Software as a Service
  - Provider licensed 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 software platform for building applications
  - E.g., Google's App-Engine
  - Avoid worrying about scalability of platform
- 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

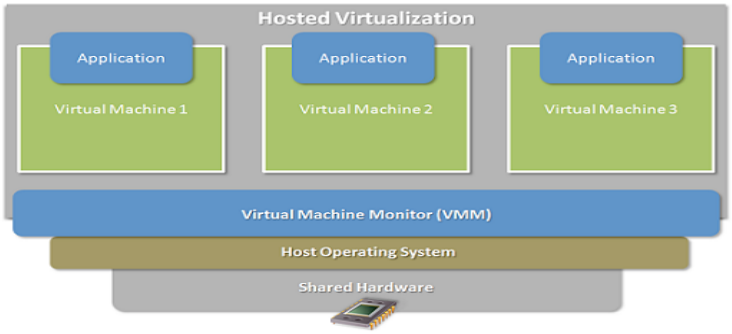
3

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



Enabling Technology: Virtualization



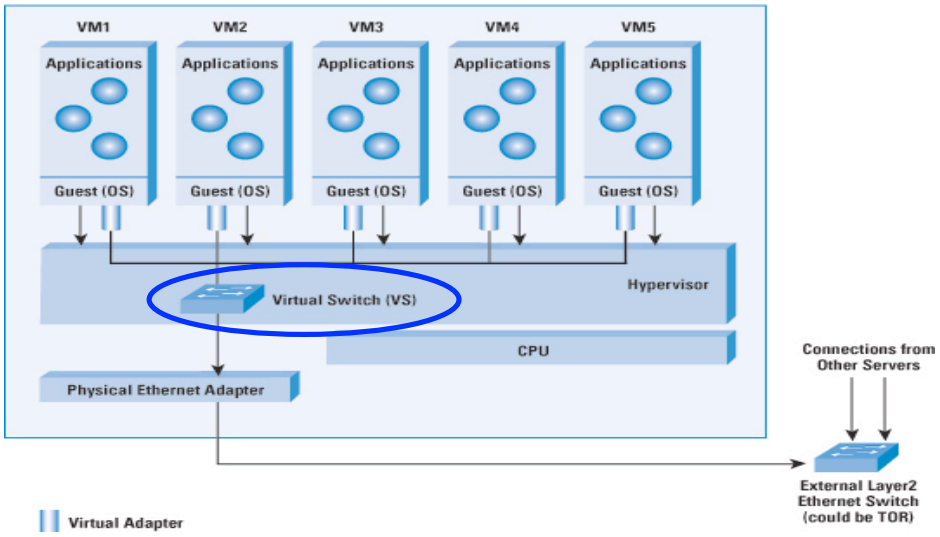
- Multiple virtual machines on one physical machine
- Applications run unmodified as on real machine
- VM can migrate from one computer to another

5

What are the design requirements in building datacenter networks?

6

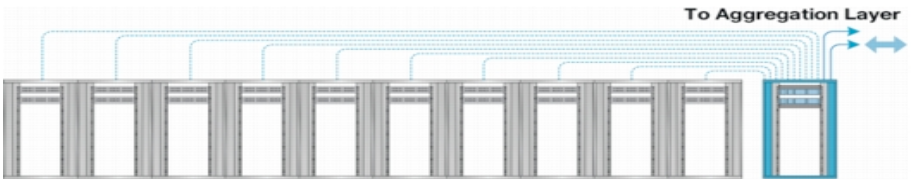
Status Quo: Virtual Switch in Server



7

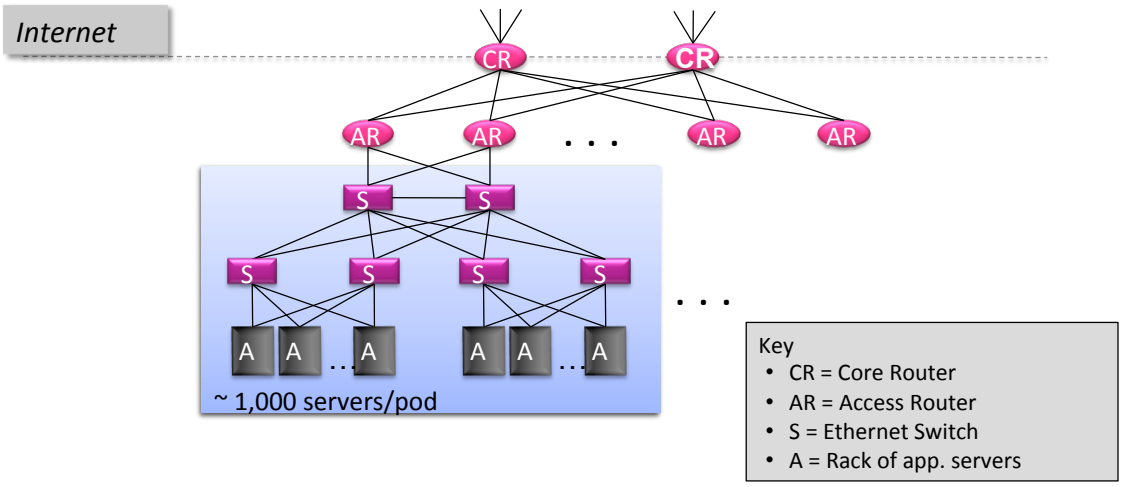
Top-of-Rack Architecture

- Rack of servers
  - Commodity servers
  - And top-of-rack switch
- Modular design
  - Preconfigured racks
  - Power, network, and storage cabling
- Aggregate to the next level



8

Data Center Network Topology

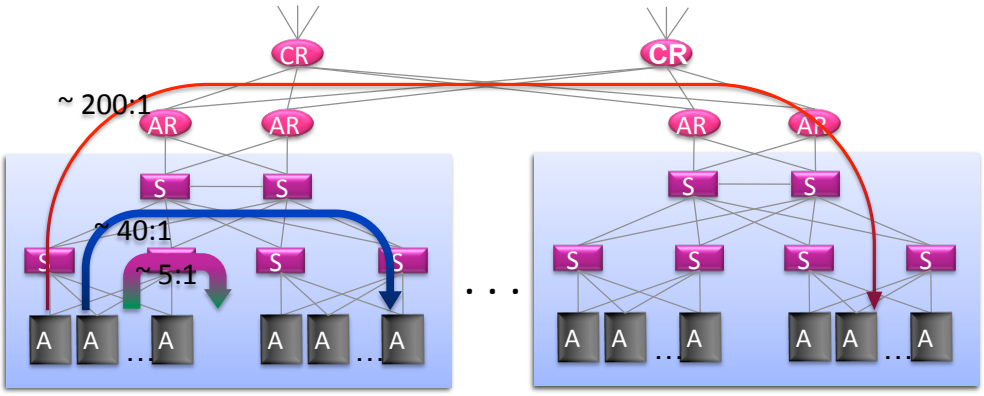


9

What do you think are the issues with these types of topologies?

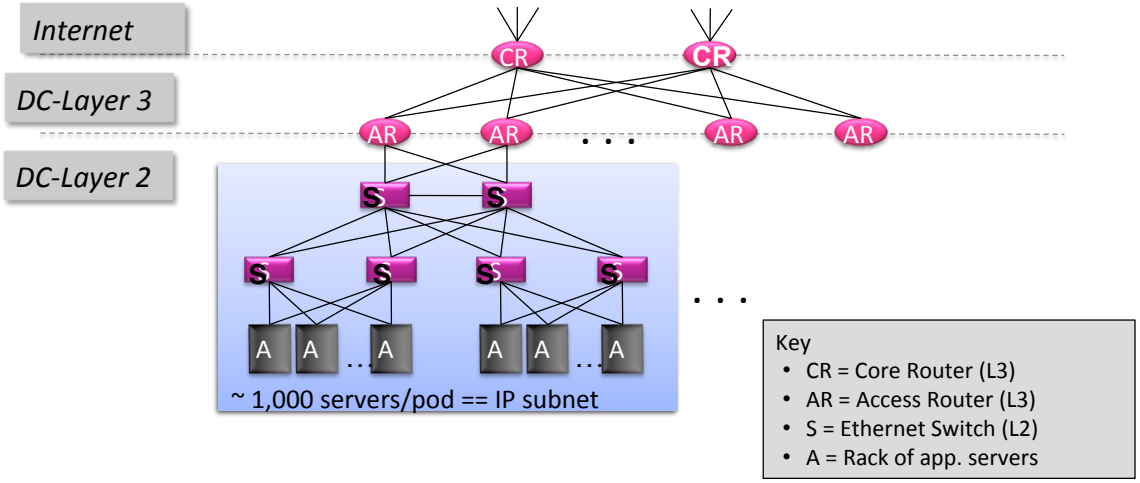
10

Capacity Mismatch



11

Data-Center Routing



12

Reminder: Layer 2 vs. Layer 3

- Ethernet switching (layer 2)
  - Cheaper switch equipment
  - Fixed addresses and auto-configuration
  - Seamless mobility, migration, and failover
- IP routing (layer 3)
  - Scalability through hierarchical addressing
  - Efficiency through shortest-path routing
  - Multipath routing through equal-cost multipath
- So, like in enterprises…
  - Data centers often connect layer-2 islands by IP routers

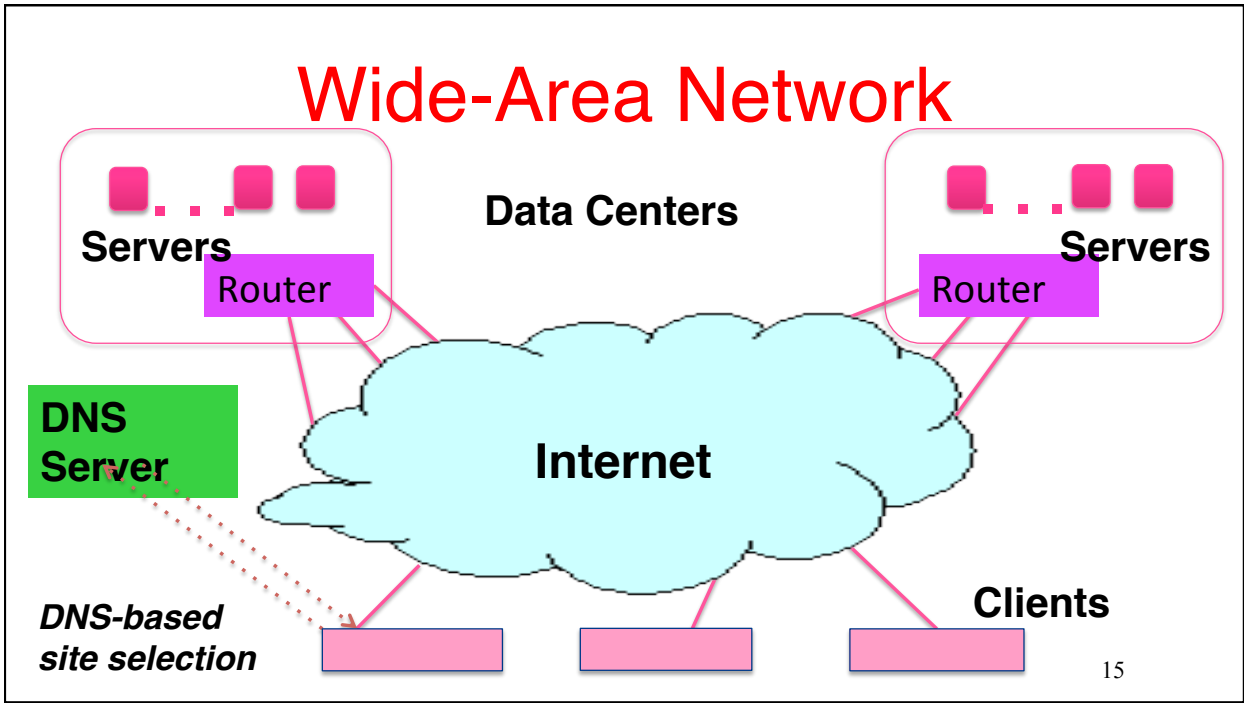
13

Data Center Costs (Monthly Costs)

- Servers: 45%
  - CPU, memory, disk
- Infrastructure: 25%
  - UPS, cooling, power distribution
- Power draw: 15%
  - Electrical utility costs
- Network: 15%
  - Switches, links, transit

http://perspectives.mvdirona.com/2008/11/28/CostOfPowerInLargeScaleDataCenters.aspx

14



Data Center Traffic Engineering

Challenges and Opportunities

16

Traffic Engineering Challenges

- Scale
  - Many switches, hosts, and virtual machines
- Churn
  - Large number of component failures
  - Virtual Machine (VM) migration
- Traffic characteristics
  - High traffic volume and dense traffic matrix
  - Volatile, unpredictable traffic patterns
- Performance requirements
  - Delay-sensitive applications
  - Resource isolation between tenants

17

Traffic Engineering Opportunities

- Efficient network
  - Low propagation delay and high capacity
- Specialized topology
  - Fat tree, Clos network, etc.
  - Opportunities for hierarchical addressing
- Control over both network and hosts
  - Joint optimization of routing and server placement
  - Can move network functionality into the end host
- Flexible movement of workload
  - Services replicated at multiple servers and data centers
  - Virtual Machine (VM) migration

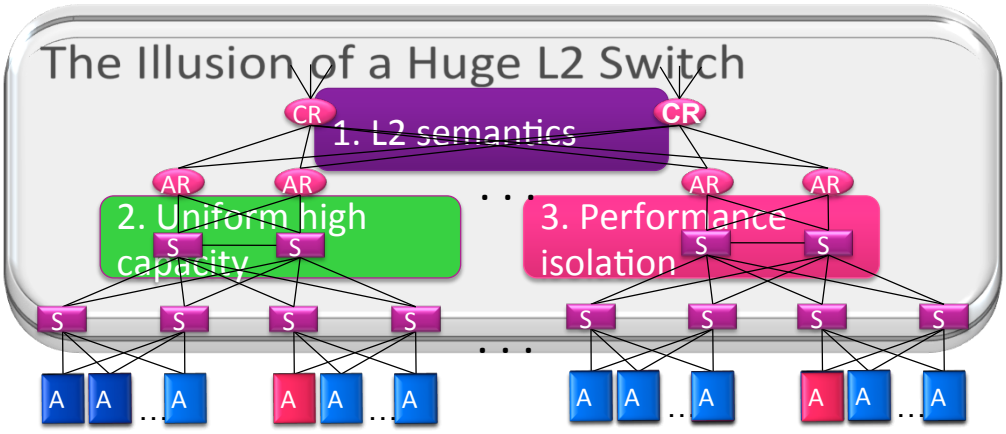
18

VL2 Paper

Slides from Changhoon Kim (now at Microsoft)

19

Virtual Layer 2 Switch



20

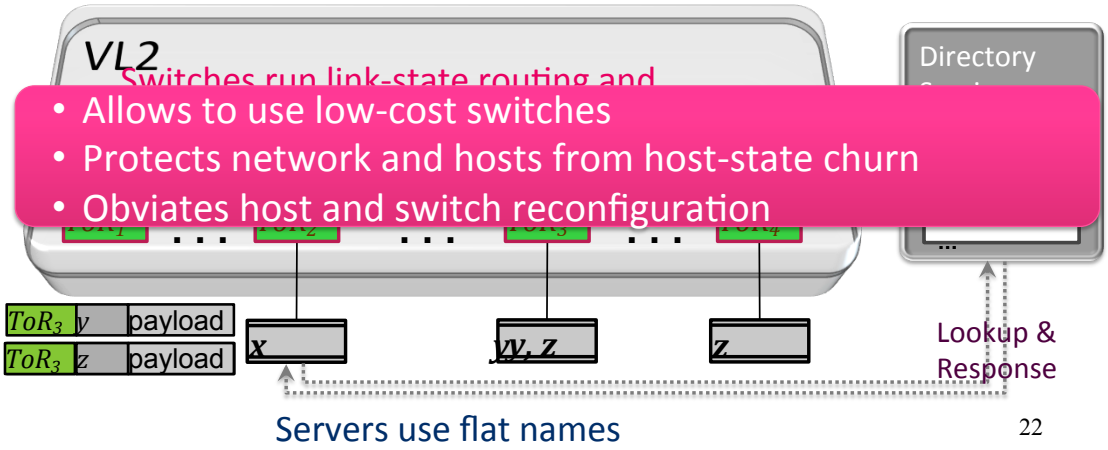
VL2 Goals and Solutions

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

"Hose": each node has ingress/egress bandwidth constraints

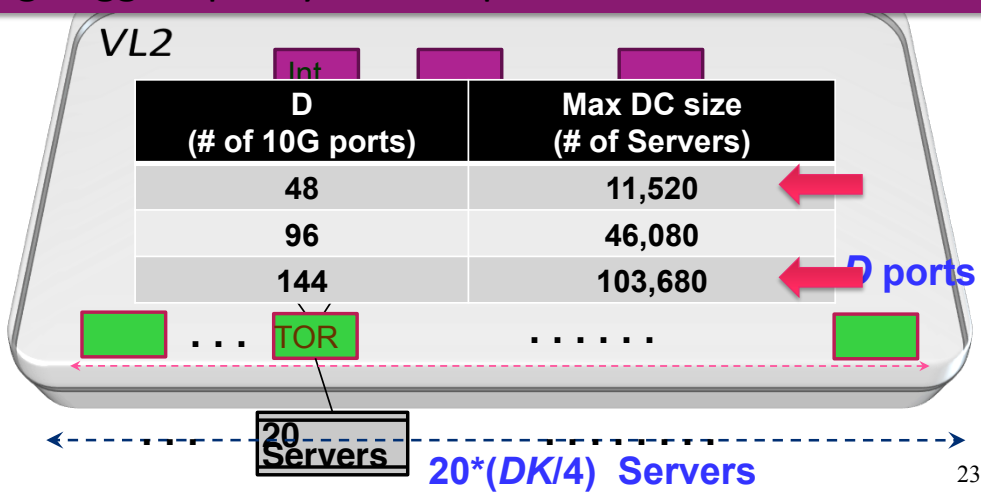
Name/Location Separation

Cope with host churns with very little overhead



Clos Network Topology

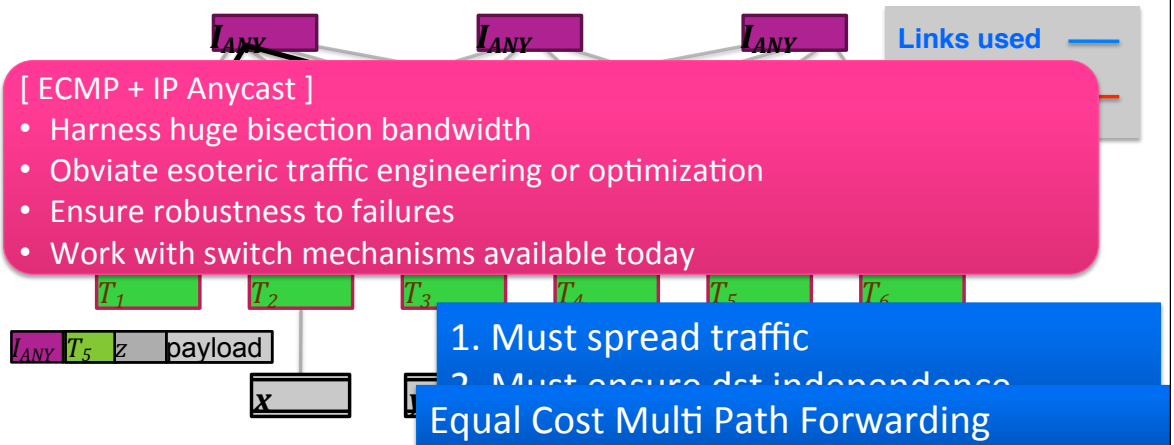
Offer huge aggr capacity & multi paths at modest cost



23

Valiant Load Balancing: Indirection

Cope with arbitrary TMs with very little overhead



24

Research Questions

- What topology to use in data centers?
  - Reducing wiring complexity
  - Achieving high bisection bandwidth
  - Exploiting capabilities of optics and wireless
- Routing architecture?
  - Flat layer-2 network vs. hybrid switch/router
  - Flat vs. hierarchical addressing
- How to perform traffic engineering?
  - Over-engineering vs. adapting to load
  - Server selection, VM placement, or optimizing routing
- Virtualization of NICs, servers, switches, …

25

Research Questions

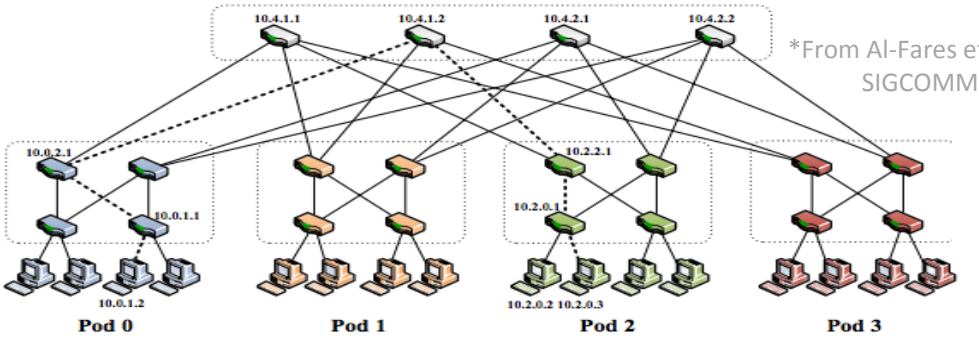
- Rethinking TCP congestion control?
  - Low propagation delay and high bandwidth
  - "Incast" problem leading to bursty packet loss
- Division of labor for TE, access control, …
  - VM, hypervisor, ToR, and core switches/routers
- Reducing energy consumption
  - Better load balancing vs. selective shutting down
- Wide-area traffic engineering
  - Selecting the least-loaded or closest data center
- Security
  - Preventing information leakage and attacks

26

Datacenter Topologies and Fault-Tolerance

27

FatTree Datacenters

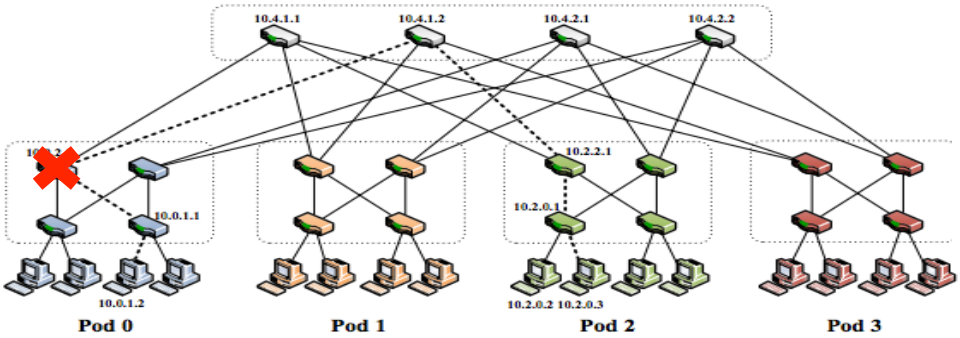


*From Al-Fares et al. SIGCOMM '08

- Alternate topologies that are less interconnected than VL2
- Commodity switches for cost, bisection bandwidth, and resilience to failures

28

FatTree Example: PortLand



- Heartbeats to detect failures
- Centralized controller installs updated routes
- Exploits path redundancy

29

Resilience to Failures is Important

- **Frequent**
  - The most failure-prone devices have a median 8.6 min TTF
- **Often Simultaneous**
  - 41% of link failure events affect multiple devices
- **Not Fail-Stop**
  - Link flapping
  - Grey/partial failures

*Statistics from Gill et al. SIGCOMM '11

30

Issue 1: Slow Recovery

- Goes to a centralized controller, which then installs state in multiple switches
- Restoration of connectivity is not local
- Topology does not support local reroutes

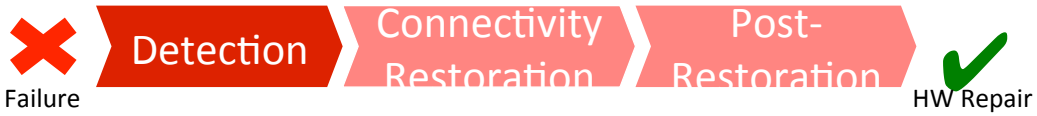
31

Issue 2: Suboptimal Flow Assignment

- Failures result in an unbalanced tree
- Loses load balancing properties
  - Spreading equally over servers/paths doesn't work (ECMP, VL2, etc)
- The entire network may need to be rebalanced

32

Issue 3: Slow Detection



- Commodity switches fail often
- Not always sure they failed
  - Gray/partial failures
  - Try to minimize global costs

33

F10

- Co-design of topology, routing protocols and failure detector
  - Novel topology that enables local, fast recovery
  - Cascading protocols for optimal use of the network
  - Fine-grained failure detector for fast detection
- Same # of switches/links as FatTrees

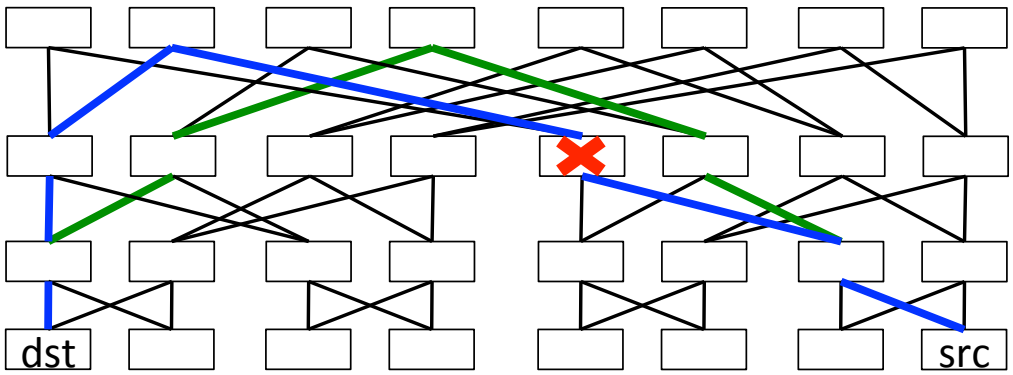
34

Outline

- Motivation & Approach
- Topology: AB FatTree
- Cascaded Failover Protocols
- Failure Detection
- Evaluation
- Conclusion

35

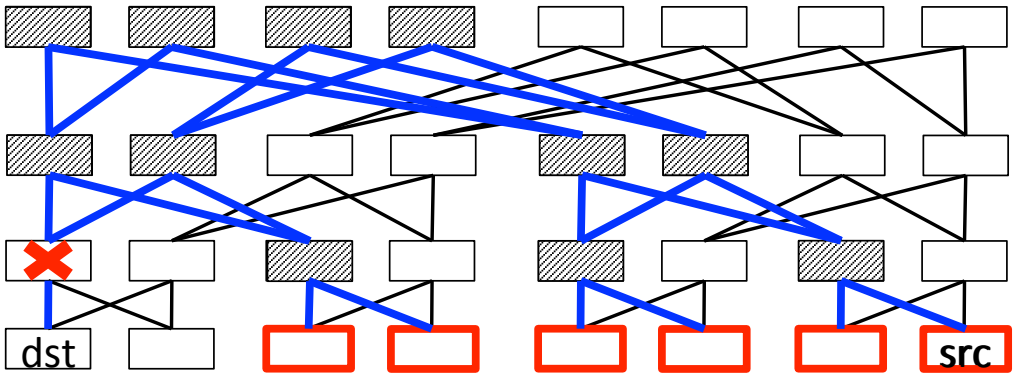
Why is FatTree Recovery Slow?



- Lots of redundancy on the upward path
- Immediately restore connectivity at the point of failure

36

Why is FatTree Recovery Slow?



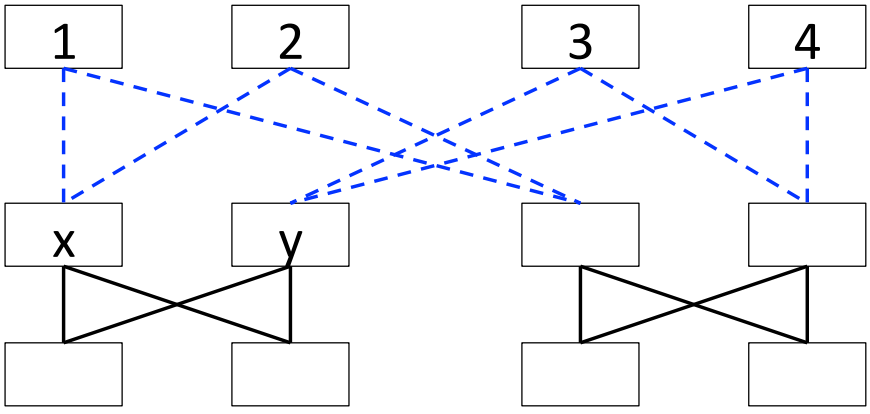
- No redundancy on the way down
- Alternatives are many hops away

No direct path
Has alternate path

37

Type A Subtree

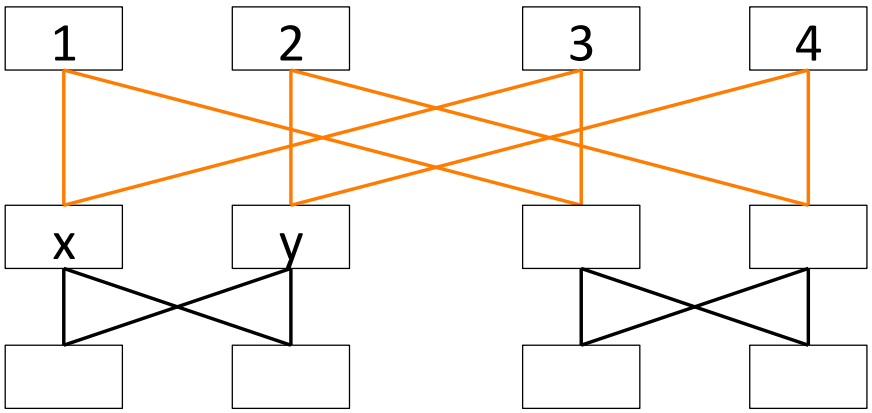
Consecutive Parents



38

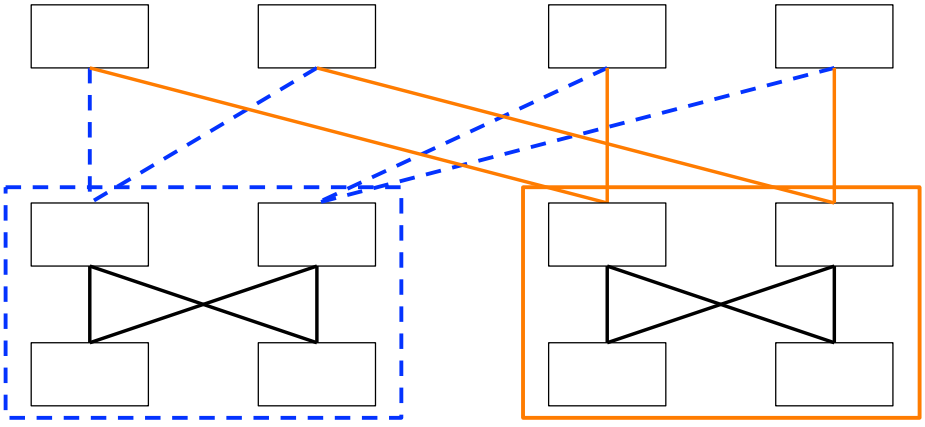
Type B Subtree

Strided Parents



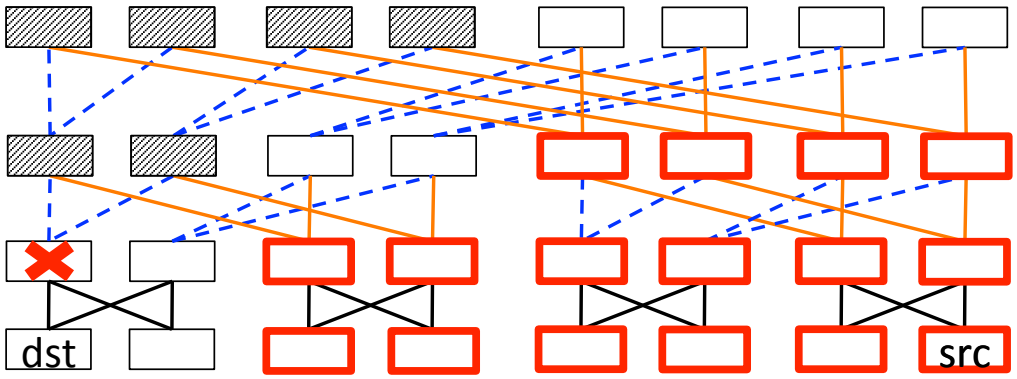
39

AB FatTree



40

Alternatives in AB FatTrees



- More nodes have alternative, direct paths
- One hop away from node with an alternative

No direct path
Has alternate path

41

Cascaded Failover Protocols

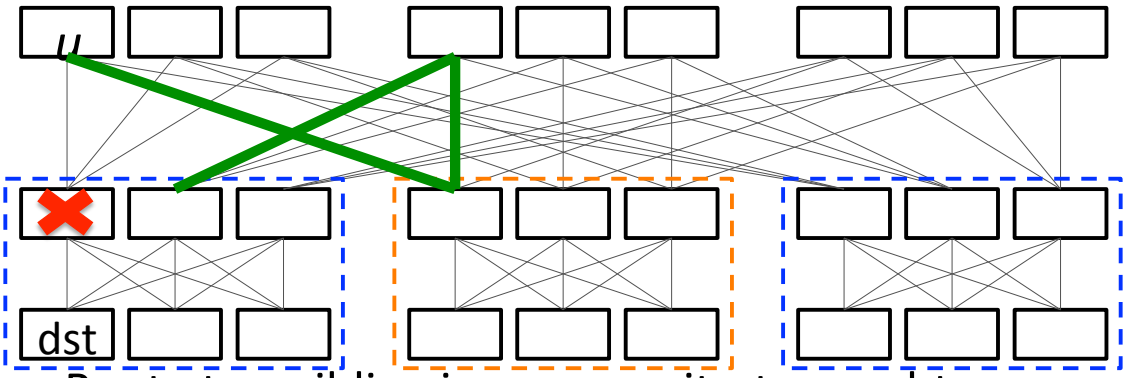
µs
- A local rerouting mechanism
  - Immediate restoration

ms
- A pushback notification scheme
  - Restore direct paths

s
- An epoch-based centralized scheduler
  - globally re-optimizes traffic

42

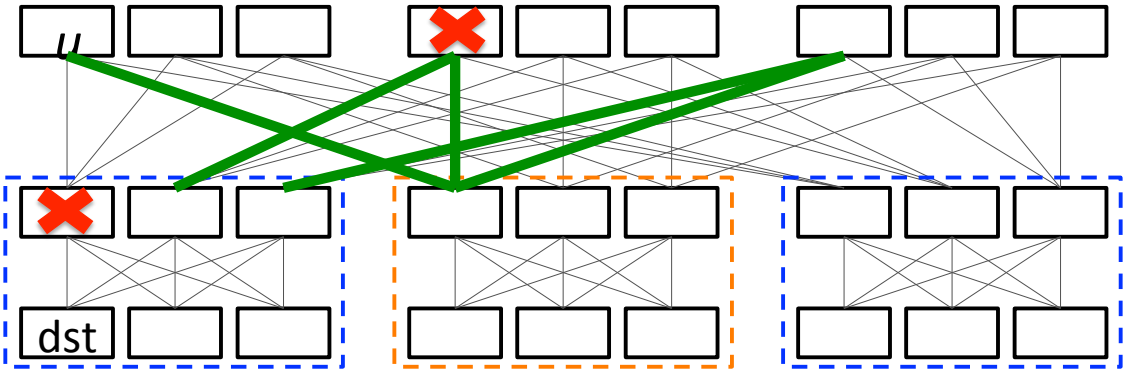
Local Rerouting



- Route to a sibling in an opposite-type subtree
- Immediate, local rerouting around the failure

43

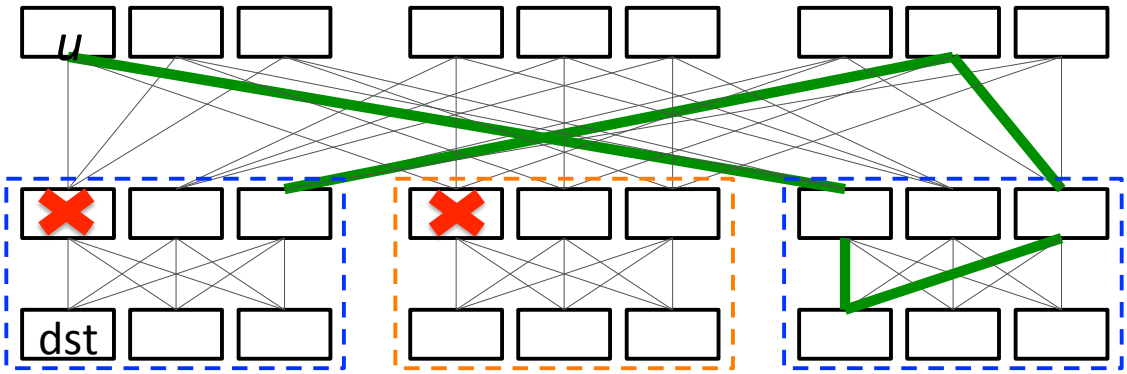
Local Rerouting – Multiple Failures



- Resilient to multiple failures
- Guaranteed to find a 2-hop reroute until k/2 failures

44

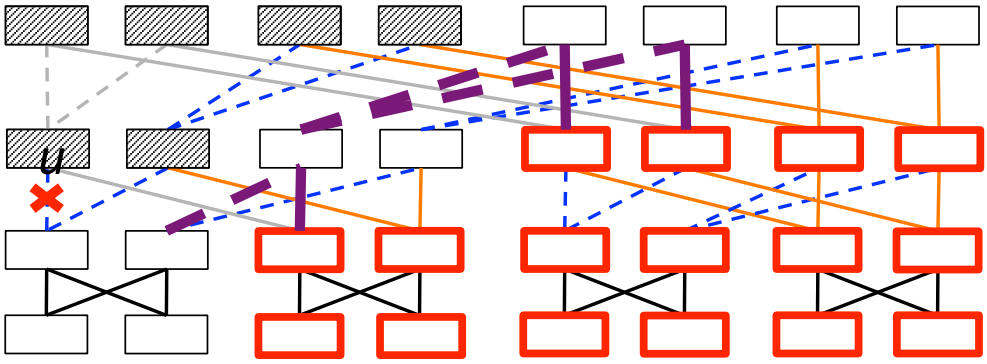
Local Rerouting – Scheme 2



- Can reroute until there k failures
- Increased load and path dilation

45

Pushback Notification



- Detecting switch broadcasts notification
- Restores direct paths, but not finished yet

No direct path
Has alternate path

46

Centralized Scheduler

- Based on existing work (Hedera, MicroTE)
- Gather traffic matrices and store the last 20
- Find stable, large flows and place them using a greedy algorithm

47

Load Balancing

- We can use the same mechanisms for load balancing
  - Reroute locally using weighted ECMP
  - Pushback notifications for backpressure congestion control
  - Global rearrangement by a centralized scheduler

48

Outline

- Motivation & Approach
- Topology: AB FatTree
- Cascaded Failover Protocols
- Failure Detection
- Evaluation

49

Why are Today's Detectors Slow?

- Based on loss of multiple heartbeats
  - Detector is separated from failure
- Slow because:
  - Congestion
  - Gray failures
  - Don't want to waste too many resources

50

F10 Failure Detector

- Look at the link itself
  - Send traffic to physical neighbors when idle
  - Monitor incoming bit transitions and packets
  - Stop sending and reroute the very next packet
- Can be fast because rerouting is cheap

51

Outline

- Motivation & Approach
- Topology: AB FatTree
- Cascaded Failover Protocols
- Failure Detection
- Evaluation

52

Evaluation

1. Can F10 reroute quickly?
2. How much overhead does rerouting add?
3. Can F10 avoid congestion loss that results from failures?
4. How much does this effect application performance?

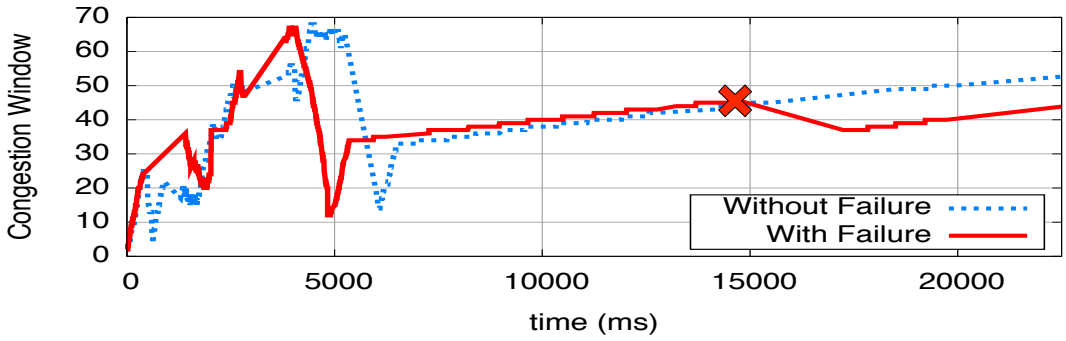
53

Methodology

- Testbed
  - Emulab w/ Click implementation
  - Used smaller packets to account for slower speed
- Packet-level simulator
  - 24-port 10GbE switches, 3 levels
  - Traffic model from Benson et al. IMC 2010
  - Failure model from Gill et al. SIGCOMM 2011
  - Validated using testbed

54

F10 Can Reroute Quickly

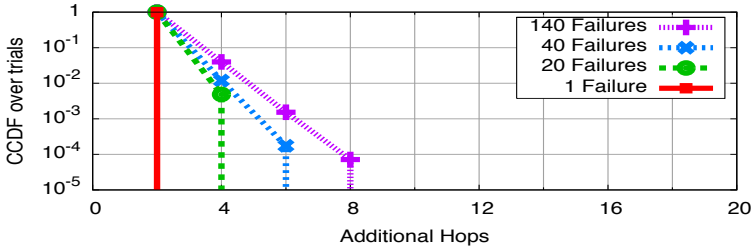


- F10 can recover from failures in under a millisecond
- Much less time than a TCP timeout

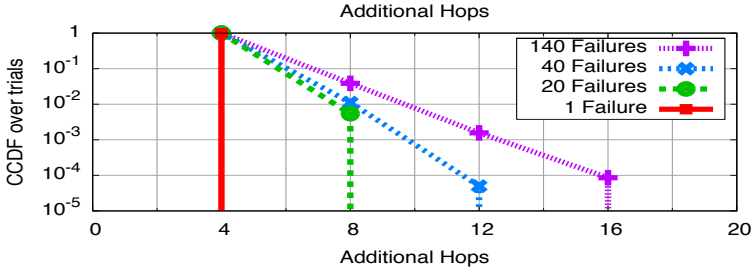
55

Path Dilation From Link Failures

AB FatTree

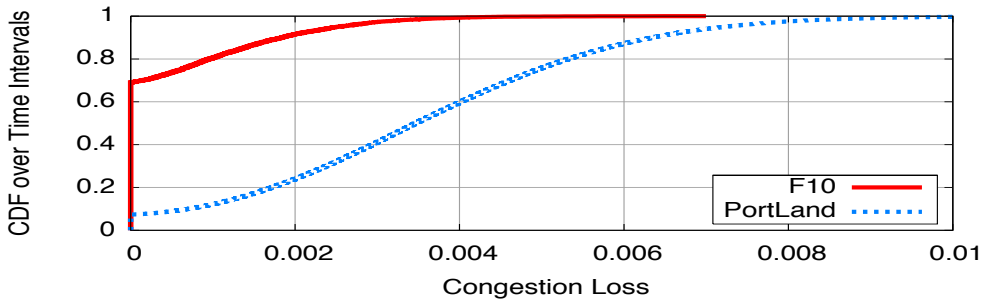


Standard FatTree



56

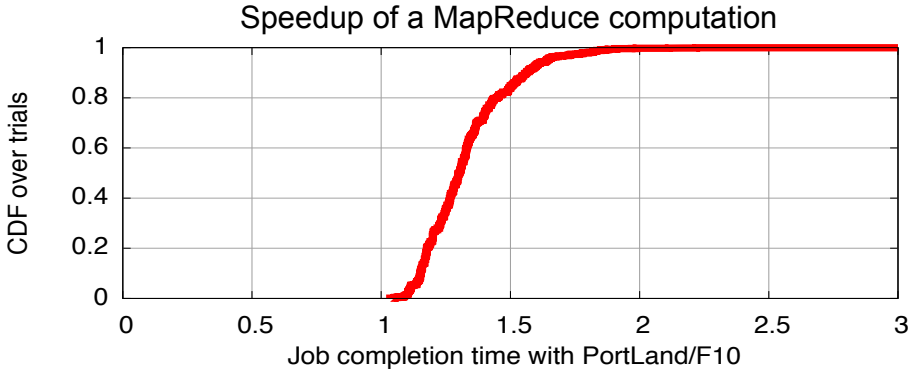
F10 Can Avoid Congestion Loss



PortLand has 7.6x the congestion loss of F10 under realistic traffic and failure conditions

57

F10 Improves App Performance



Median speedup is 1.3x

58

Summary

- F10 is a co-design of topology, routing protocols, and failure detector:
  - AB FatTrees to allow local recovery and increase path diversity
  - Pushback and global re-optimization restore congestion-free operation
- Significant benefit to application performance on typical workloads and failure conditions

59

Introduction to Computer Networks

TCP in Datacenters

Computer Science & Engineering

UNIVERSITY of WASHINGTON

TCP in the Data Center

- We'll see TCP does not meet demands of apps.
  - Suffers from bursty packet drops, Incast [SIGCOMM '09], ...
  - 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
  - No solid understanding of consequences, tradeoffs

61

Roadmap

- What's really going on?
  - Interviews with developers and operators
  - Analysis of applications
  - Switches: shallow-buffered vs deep-buffered
  - Measurements
- A systematic study of transport in Microsoft's DCs
  - Identify **impairments**
  - Identify **requirements**
- Our solution: **Data Center TCP**

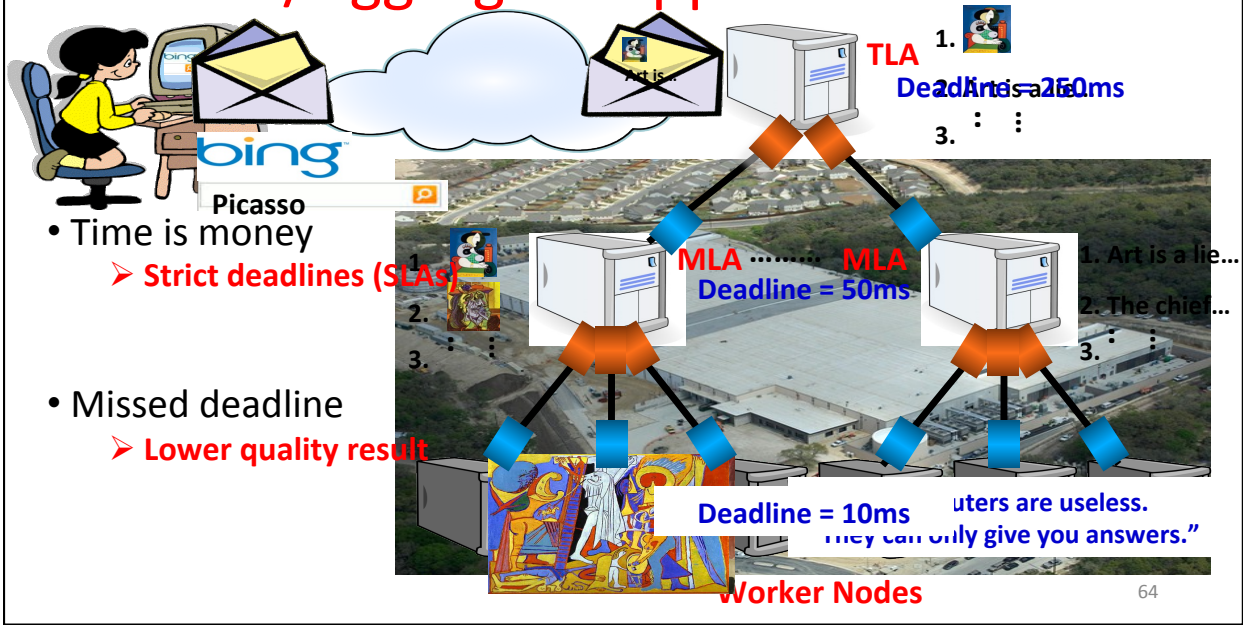
62

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

63

Partition/Aggregate Application Structure



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

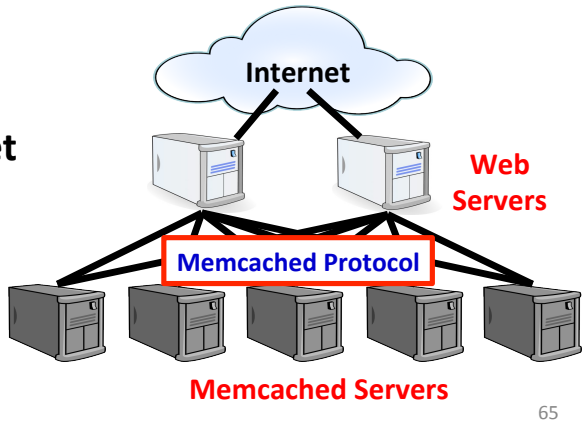
64

Generality of Partition/Aggregate

- The foundation for many large-scale web applications.
  - Web search, Social network composition, Ad selection, etc.
- Example: **Facebook**

Partition/Aggregate ~ Multiget

- Aggregators: **Web Servers**
- Workers: **Memcached Servers**



65

Workloads

- Partition/Aggregate **(Query)** → **Delay-sensitive**
- Short messages [50KB-1MB] **(Coordination, Control state)** → **Delay-sensitive**
- Large flows [1MB-50MB] **(Data update)** → **Throughput-sensitive**

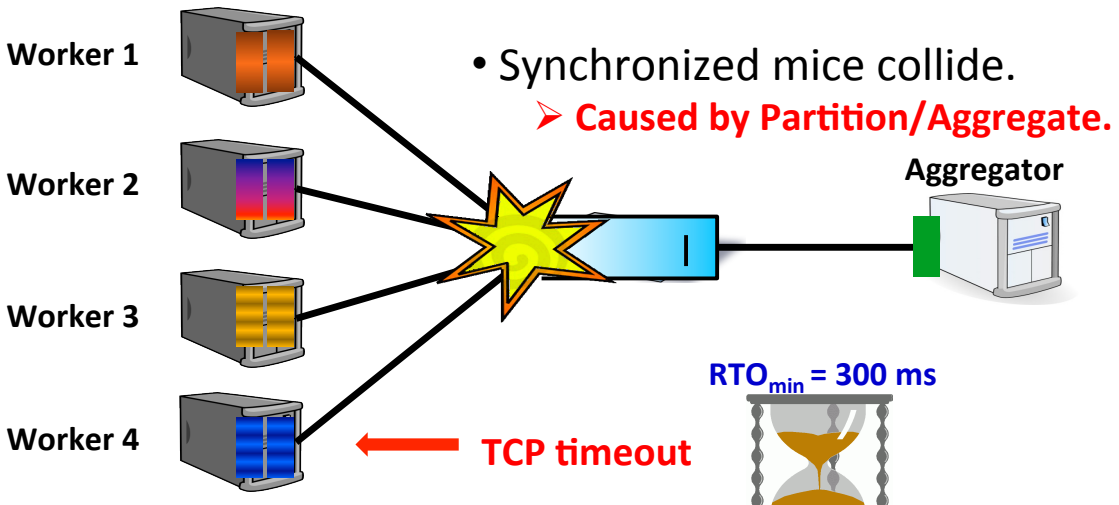
66

Impairments

- Incast
- Queue Buildup
- Buffer Pressure

67

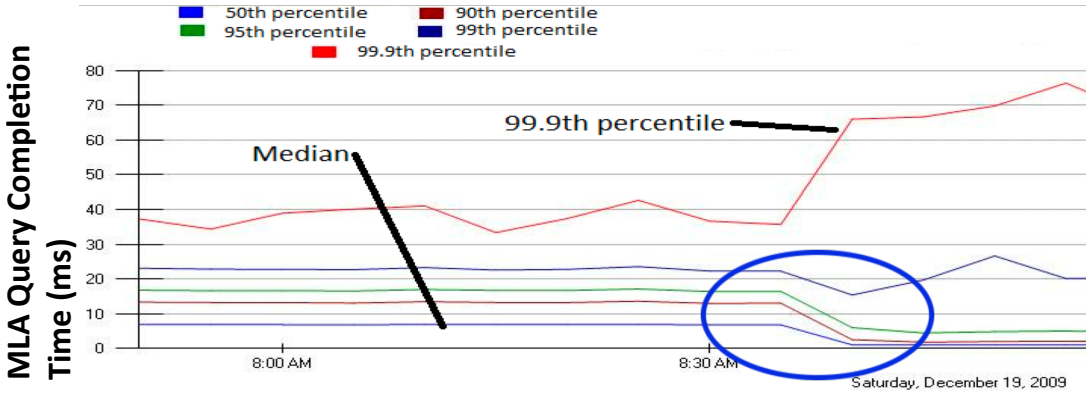
Incast



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

68

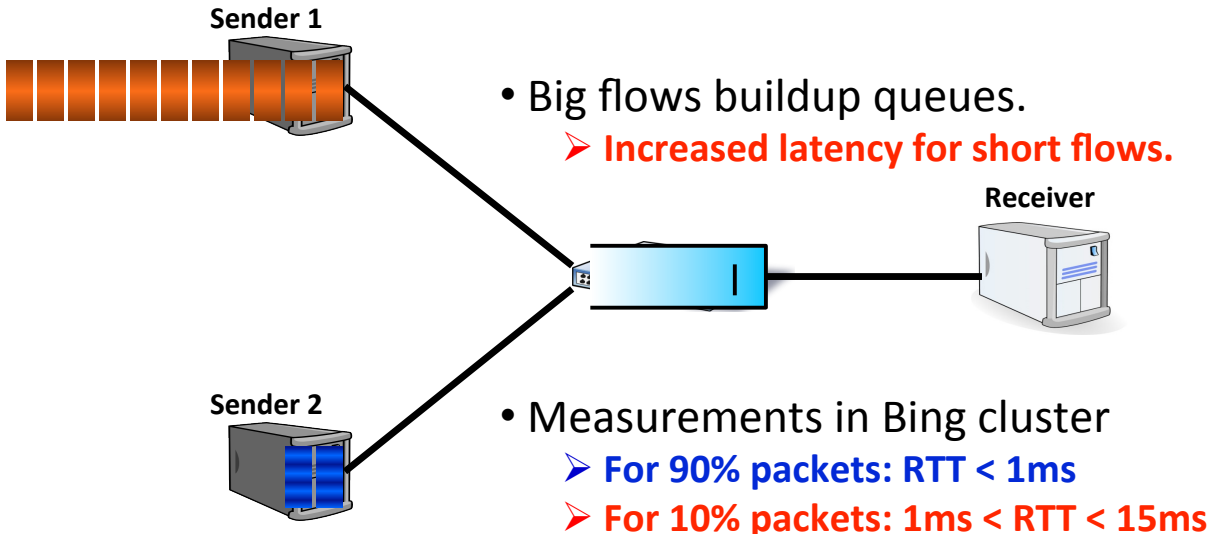
Incast Really Happens



Jittering **99.9th percentile is being tracked.** **ntiles.**

69

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**

70

Data Center Transport Requirements

1. High Burst Tolerance

– Incast due to Partition/Aggregate is common.

2. Low Latency

– Short flows, queries

3. High Throughput

The challenge is to achieve these three together.

71

Tension Between Requirements

High Throughput
High Burst Tolerance

Low Latency

Deep
➢ Qu
Inc

Objective:
Low Queue Occupancy & High Throughput

&

Reduced RTO$_{min}$ (SIGCOMM '09)
➢ Doesn't Help Latency

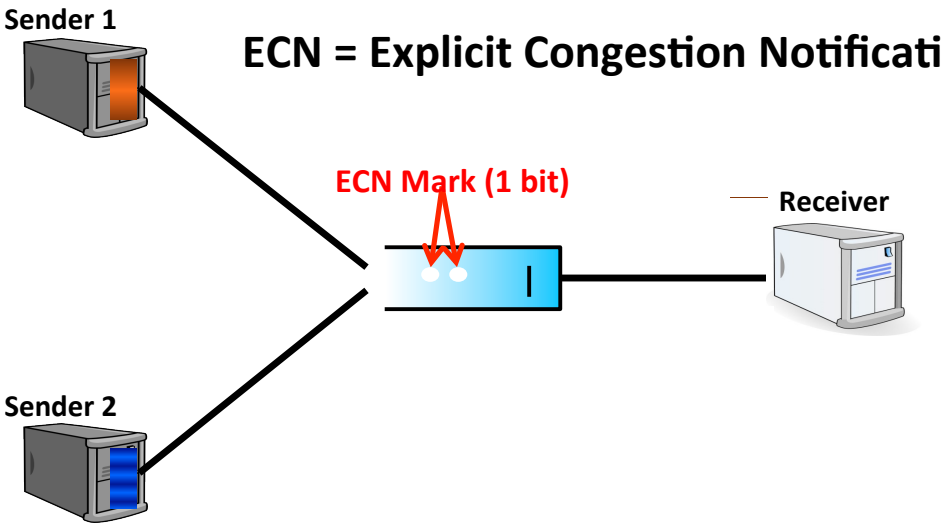
AQM – RED:
➢ Avg Queue Not Fast Enough for Incast

72

The DCTCP Algorithm

73

Review: The TCP/ECN Control Loop

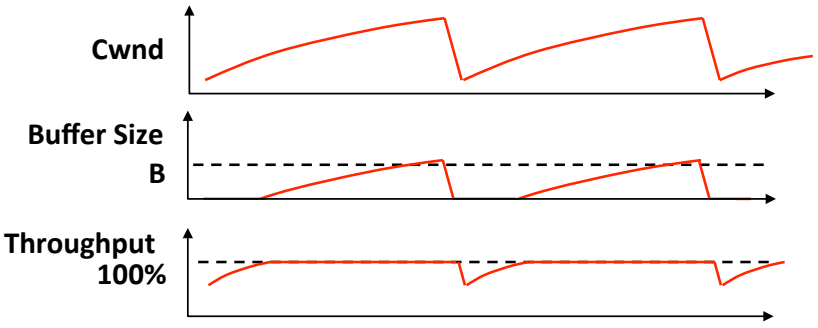


74

Small Queues & TCP Throughput:

The Buffer Sizing Story

- Bandwidth-delay product rule of thumb:
  - A single flow needs $C \times RTT$ buffers for **100% Throughput.**

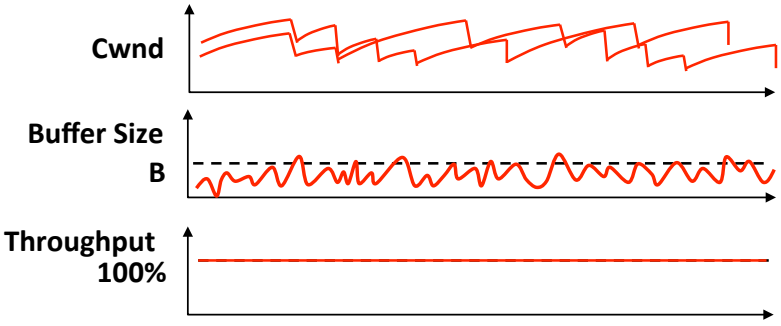


17

Small Queues & TCP Throughput:

The Buffer Sizing Story

- Bandwidth-delay product rule of thumb:
  - A single flow needs $C \times RTT$ buffers for **100% Throughput.**
- Appenzeller rule of thumb (SIGCOMM '04):
  - Large # of flows: $C \times RTT/\sqrt{N}$ is enough.



17

Small Queues & TCP Throughput:

The Buffer Sizing Story

- Bandwidth-delay product rule of thumb:
  - A single flow needs $C \times RTT$ buffers for **100% Throughput.**
- Appenzeller rule of thumb (SIGCOMM '04):
  - Large # of flows: $C \times RTT/\sqrt{N}$ is enough.
- Can't rely on stat-mux benefit in the DC.
  - Measurements show **typically 1-2 big flows** at each server**, at most 4.**

17

Small Queues & TCP Throughput:

The Buffer Sizing Story

- Bandwidth-delay product rule of thumb:
  - A single flow needs $C \times RTT$ buffers for **100% Throughput.**
- Appenzeller rule of thumb (SIGCOMM '04):
  - Large # of flows: $C \times RTT/\sqrt{N}$ is enough.
- Can't rely on stat-mux benefit in the DC.
  - Measurements show **typically 1-2 big flows** at each server**, at most 4.**

Real Rule of Thumb:
Low Variance in Sending Rate → Small Buffers Suffice

17

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%

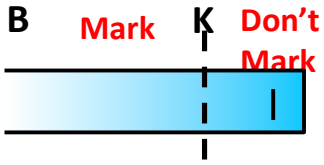
2. Mark based on **instantaneous** queue length.
   - ✓ Fast feedback to better deal with bursts.

18

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:

$$F = \frac{\#\ of\ marked\ ACKs}{Total\ \#\ of\ ACKs}$$

$$\alpha \leftarrow (1-g)\alpha + gF$$

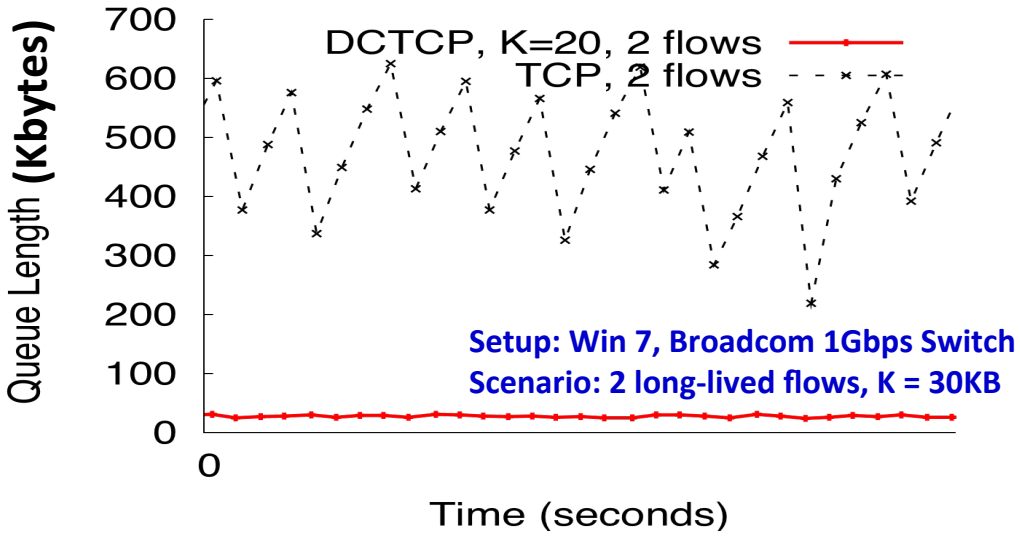
➢ **Adaptive window decreases:**

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

– Note: decrease factor between 1 and 2.

19

DCTCP in Action



20

Why it Works

1.High Burst Tolerance

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

2. Low Latency

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

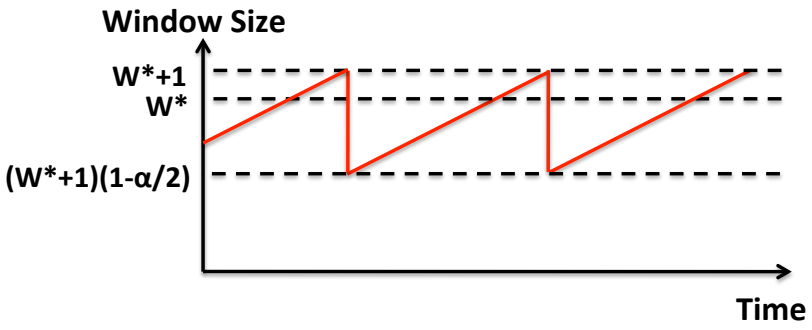
3. High Throughput

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

21

Analysis

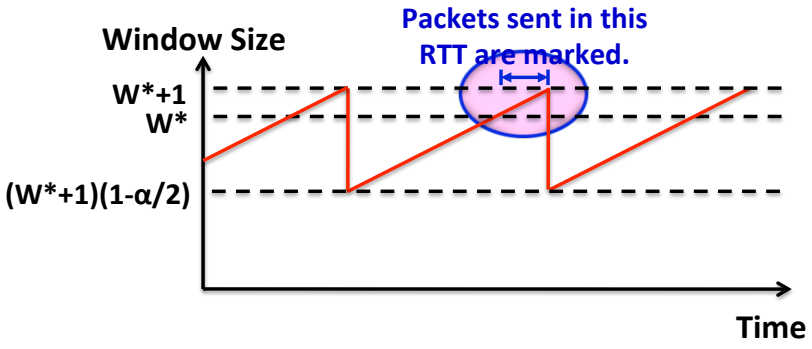
- How low can DCTCP maintain queues without loss of throughput?
- How do we set the DCTCP parameters?
  - ➢ **Need to quantify queue size oscillations (Stability).**



22

Analysis

- How low can DCTCP maintain queues without loss of throughput?
- How do we set the DCTCP parameters?
  - ➢ **Need to quantify queue size oscillations (Stability).**



22

Analysis

- How low can DCTCP maintain queues without loss of throughput?
- How do we set the DCTCP parameters?
  - ➢ **Need to quantify queue size oscillations (Stability).**

$$K > \frac{1}{7} C \times RTT$$

85% Less Buffer than TCP

22

Evaluation

- Implemented in Windows stack.
- Real hardware, **1Gbps and 10Gbps** experiments
  - **90 server testbed**
  - **Broadcom Triumph** 48 1G ports – 4MB shared memory
  - **Cisco Cat4948** 48 1G ports – 16MB shared memory
  - **Broadcom Scorpion** 24 10G ports – 4MB shared memory
- Numerous micro-benchmarks
  - **Throughput and Queue Length**
  - **Multi-hop**
  - **Queue Buildup**
  - **Buffer Pressure**
  - **Fairness and Convergence**
  - **Incast**
  - **Static vs Dynamic Buffer Mgmt**
- **Cluster traffic benchmark**

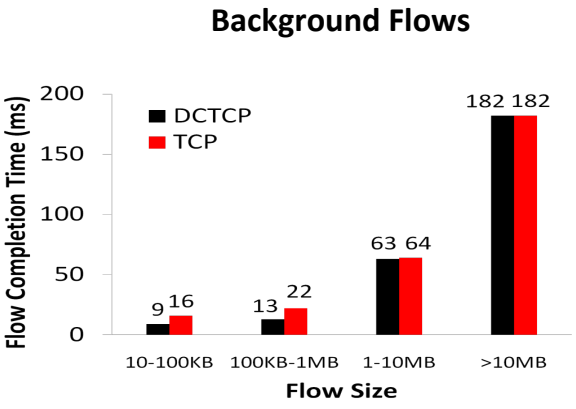
23

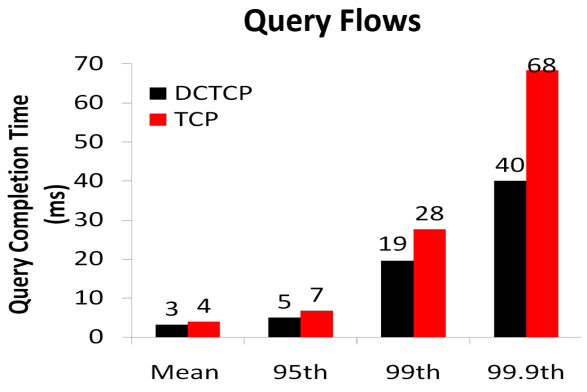
Cluster Traffic Benchmark

- Emulate traffic within 1 Rack of Bing cluster
  - 45 1G servers, 10G server for external traffic
- Generate query, and background traffic
  - Flow sizes and arrival times follow distributions seen in Bing
- Metric:
  - Flow completion time for queries and background flows.

24

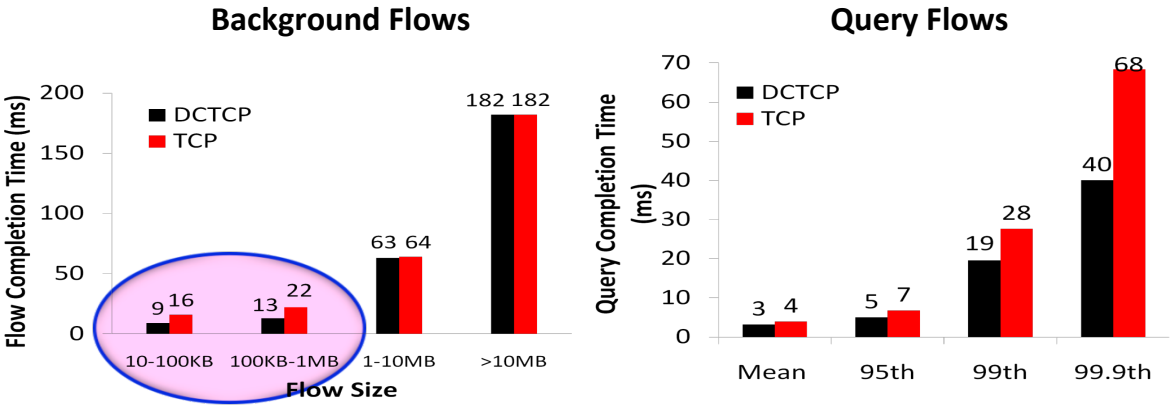
Baseline





25

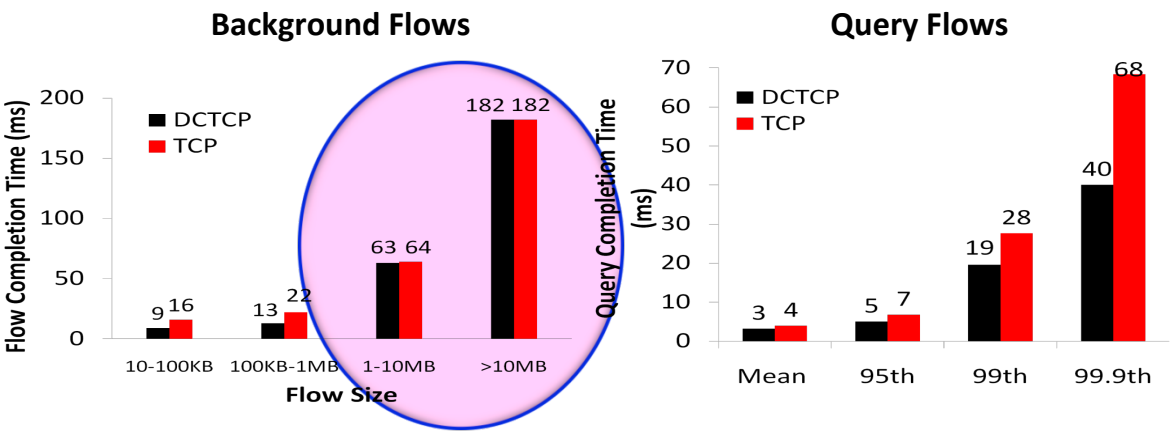
Baseline



✓ Low latency for short flows.

25

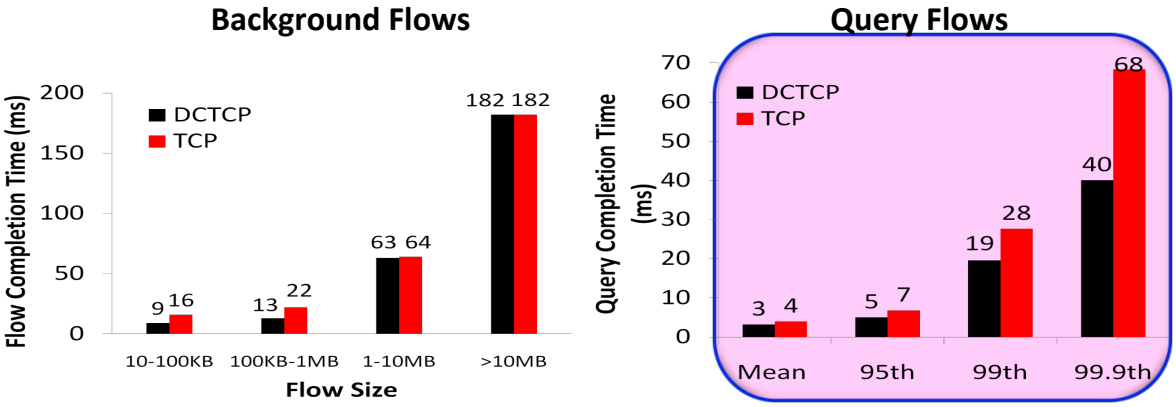
Baseline



✓ Low latency for short flows.
✓ High throughput for long flows.

25

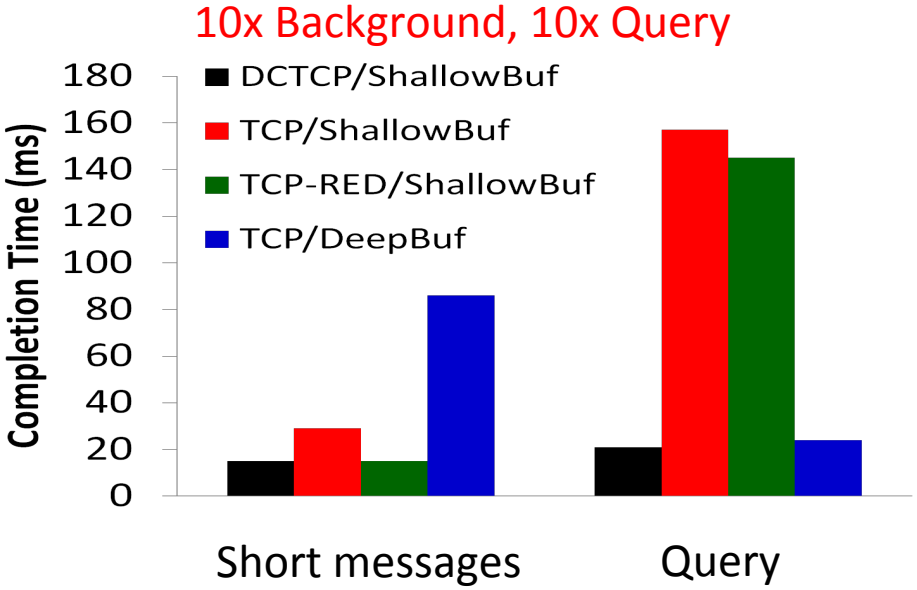
Baseline



- ✓ Low latency for short flows.
- ✓ High throughput for long flows.
- ✓ High burst tolerance for query flows.

25

Scaled Background & Query



26

Conclusions

- DCTCP satisfies all the requirements for Data Center packet transport.
  - ✓ **Handles bursts well**
  - ✓ **Keeps queuing delays low**
  - ✓ **Achieves high throughput**
- Features:
  - ✓ **Very simple change to TCP and a single switch parameter.**
  - ✓ **Based on mechanisms already available in Silicon.**

27

Introduction to Computer Networks

Distributed Systems inside Datacenters

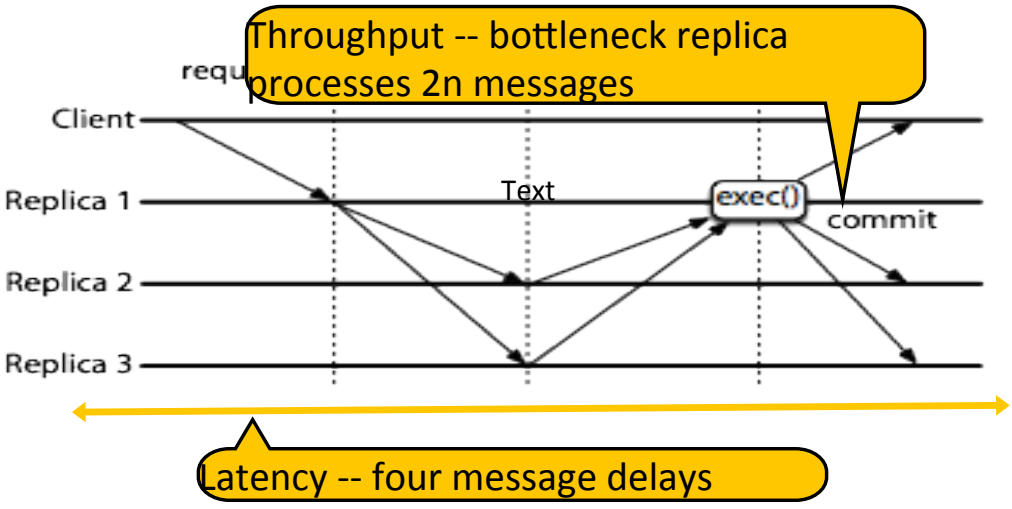
Computer Science & Engineering

UNIVERSITY of WASHINGTON

Coordination Systems

- Replicated, consistent, and highly available services
- Many names and variations:
  - consensus, atomic broadcast, Paxos, Viewstamped replication, Virtual synchrony
- Widely used inside the datacenter
  - distributed synchronization, cluster management (e.g., Chubby, Zookeeper)
  - persistent storage in distributed databases (e.g., Spanner, H-Store)

Paxos



Protocol Complexity

- *Asynchronous model* of distributed computation
- *Nodes* have arbitrary speeds (fail-stop-restart)
- *Network* is arbitrarily asynchronous:
  - messages have no bounded latency
  - messages can be lost, reordered
  - arbitrary interleaving between concurrent group communications

Datacenter Setting

- Datacenter networks are more *predictable*
  - structured topology, known routes $\Rightarrow$ predictable latencies
- Datacenter networks are more *reliable*
  - failures less common, can avoid congestion loss
- Datacenter networks are *extensible*
  - switches capable of complex line-rate processing/forwarding
  - exposed through SDNs / OpenFlow
  - single administrative domain makes changes practical

Speculative Paxos

- Based on *approximate asynchrony*
- Faster state machine replication with network support
  - Mostly-Ordered Multicast (MOM) primitive
  - Optimized speculative consensus protocol

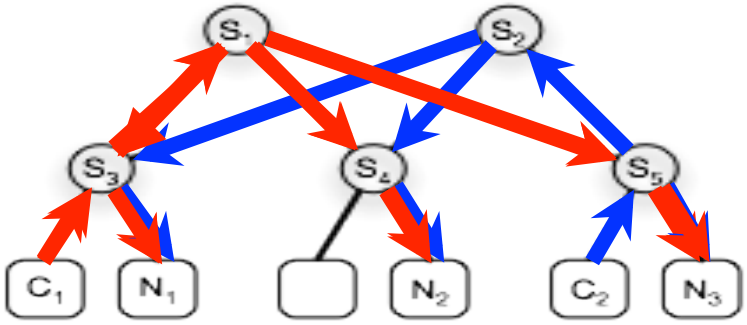
Ordered Multicasts

- Guaranteed ordering of concurrent messages:
  - If any node receives m_1 then m_2, then all other receivers process both in the same order
  - violated by reordering *or packet drops*!
- Too strong:
  - essentially eliminates need for consensus
  - how to deal with node or network failures?

Mostly-Ordered Multicast (MOMs)

- *Best-effort* ordering of concurrent messages:
  - If any node receives m_1 then m_2, then all other receivers process them in the same order *with high probability*
- More practical to implement; but not satisfied by existing multicast protocols

Mostly-Ordered Multicast

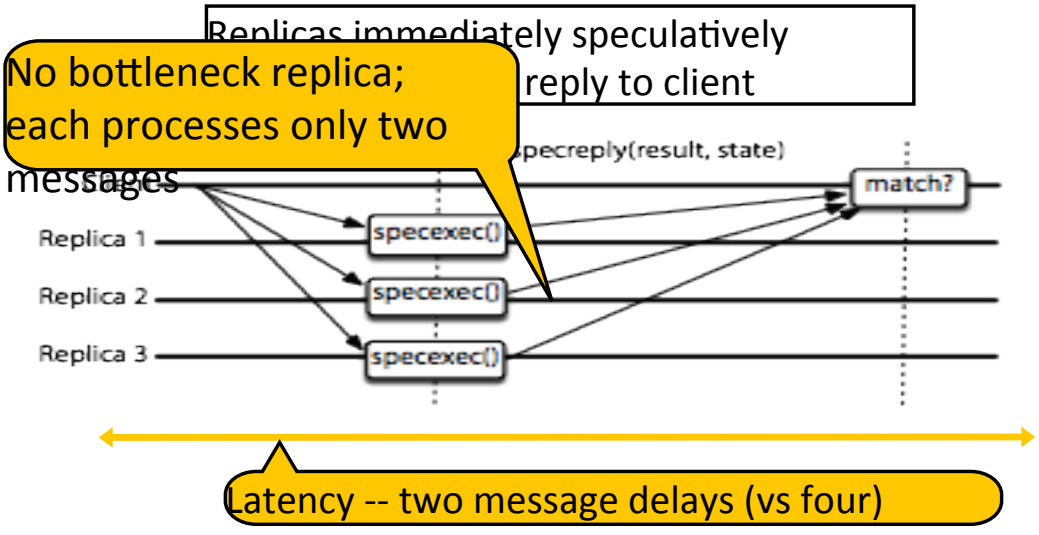


- Different path lengths, congestion cause reordering
- Route multicast messages to a root switch equidistant from receivers
- QoS prioritization to minimize queuing delay
- Additional hardware support: timestamping at linerate

Speculative Paxos

- Relies on MOMs to order requests in the normal case
- But not required:
  - remains correct even with reorderings: safety + liveness under usual conditions
- Design pattern: separate mechanisms for *approximately synchronous* and *asynchronous* execution modes

Speculative Paxos



Speculative Execution

- Replicas execute requests speculatively
  - but clients know their requests succeeded; don't need to speculate
- Replicas periodically run *synchronization* protocol
- If divergence detected: *reconciliation*
  - replicas pause execution, select leader, send logs
  - leader decides ordering for operations and notifies replicas
  - replicas rollback and re-execute requests in proper order

Summary of Results

- Speculative Paxos outperforms Paxos when reorder rates are low
  - *2.3x* higher throughput, *40%* lower latency
  - effective up to reorder rates of *0.5%*
- MOMs provide the necessary network support
  - reorder rates (without timestamping support):
    - *0.01-0.02%* in small testbed
    - *0.02-0.10%* in simulated 160-switch datacenter