Data Center Networks

Lecture 13

Network as a Computer

- Network/cluster computing has been around forever
  - Grid computing
  - Cluster computing
- Highly specialized
  - Scientific computing (eg. nuclear simulation)
  - Stock transactions
- All of a sudden, data centers are extremely hot
  - **Why?**

The Internet Made it Happen

- Everyone wants to operate at Internet scale
  - Millions of users
    - Can you website survive a flash of mob?
  - Zetabytes of data to analyze
    - Web server logs
    - Ads reviews/clicks
    - Social networks, blogs, Twitter, video…
- But not everyone has the expertise to build a cluster
  - **Let someone else do it for you!**

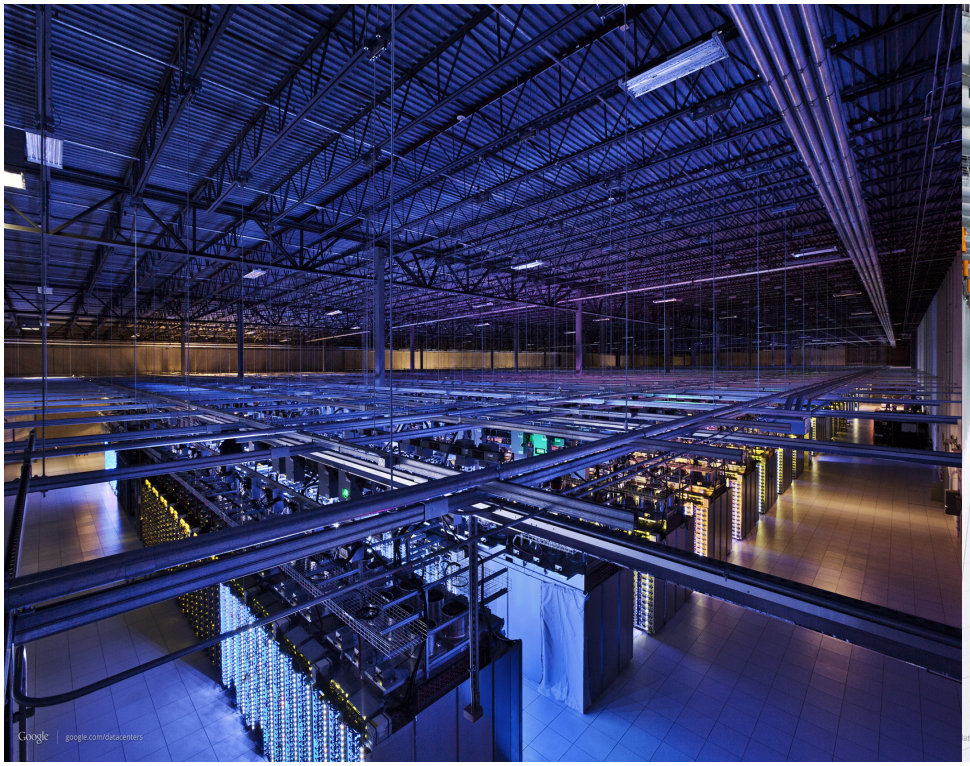
Cloud Computing

- Everything is a service
  - Infrastructure
  - Platform
  - Software
  - Storage

- What actually powers the cloud?

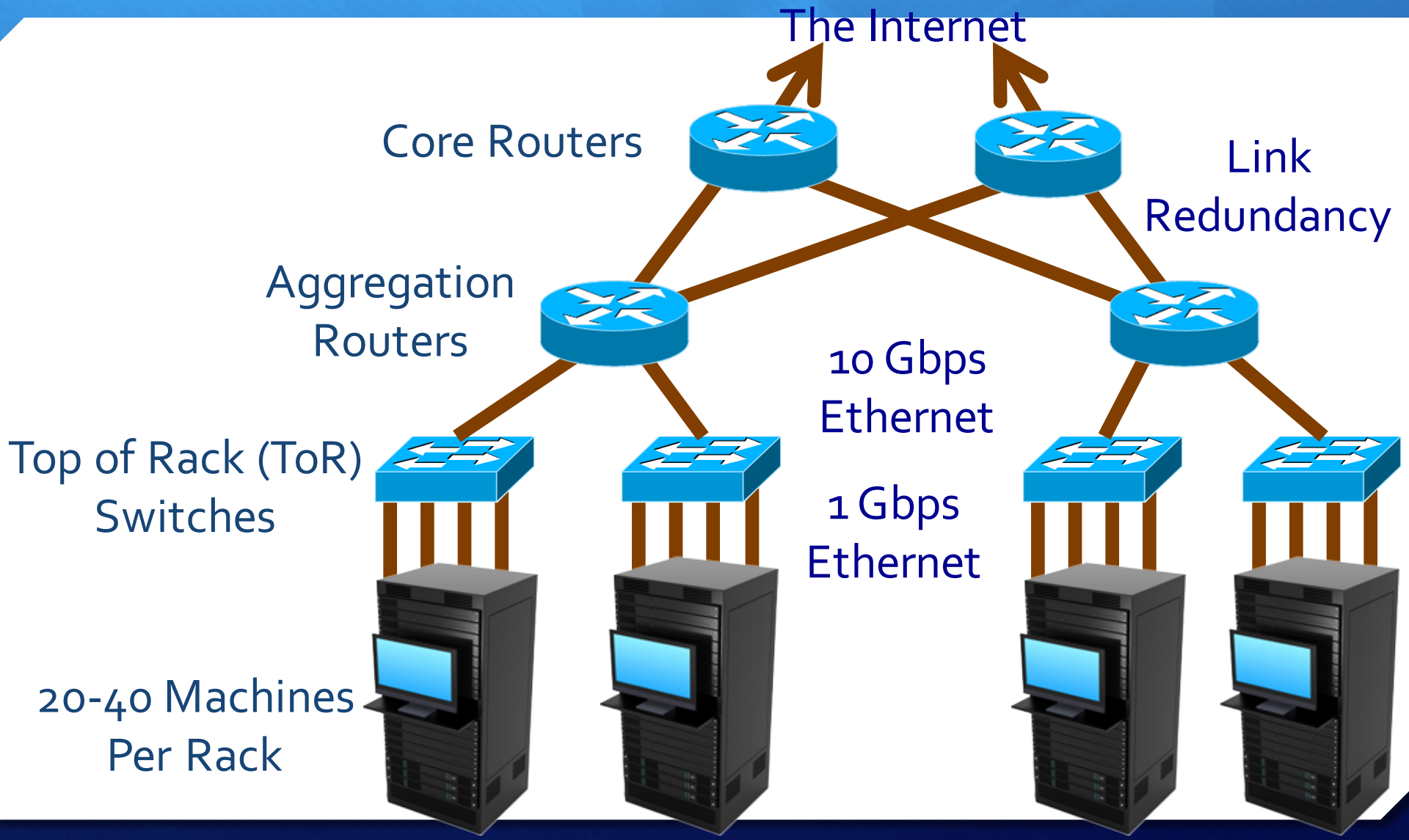
Today's Data Centers

- Warehouse of servers



https://www.youtube.com/watch?v=zDAYZU4A3w0

Typical Data Center Topology



Advantages of Today's Designs

- Cheap, off-the-shelf commodity hardware
  - No more specialized hardware or networking kit
  - Easier to scale out horizontally
- Use standard software
  - No need for cluster or grid OSs
  - Stock networking protocols
- Ideal for VMs
  - Redundant
  - Homogeneous

Lots of Open Problems

- Diverse applications
  - Heterogeneous, unpredictable traffic patterns
  - Competition over resources
  - Isolation
  - Reliability issues
  - Privacy
- Management, diagnosis and debugging at scale
- Heat and power

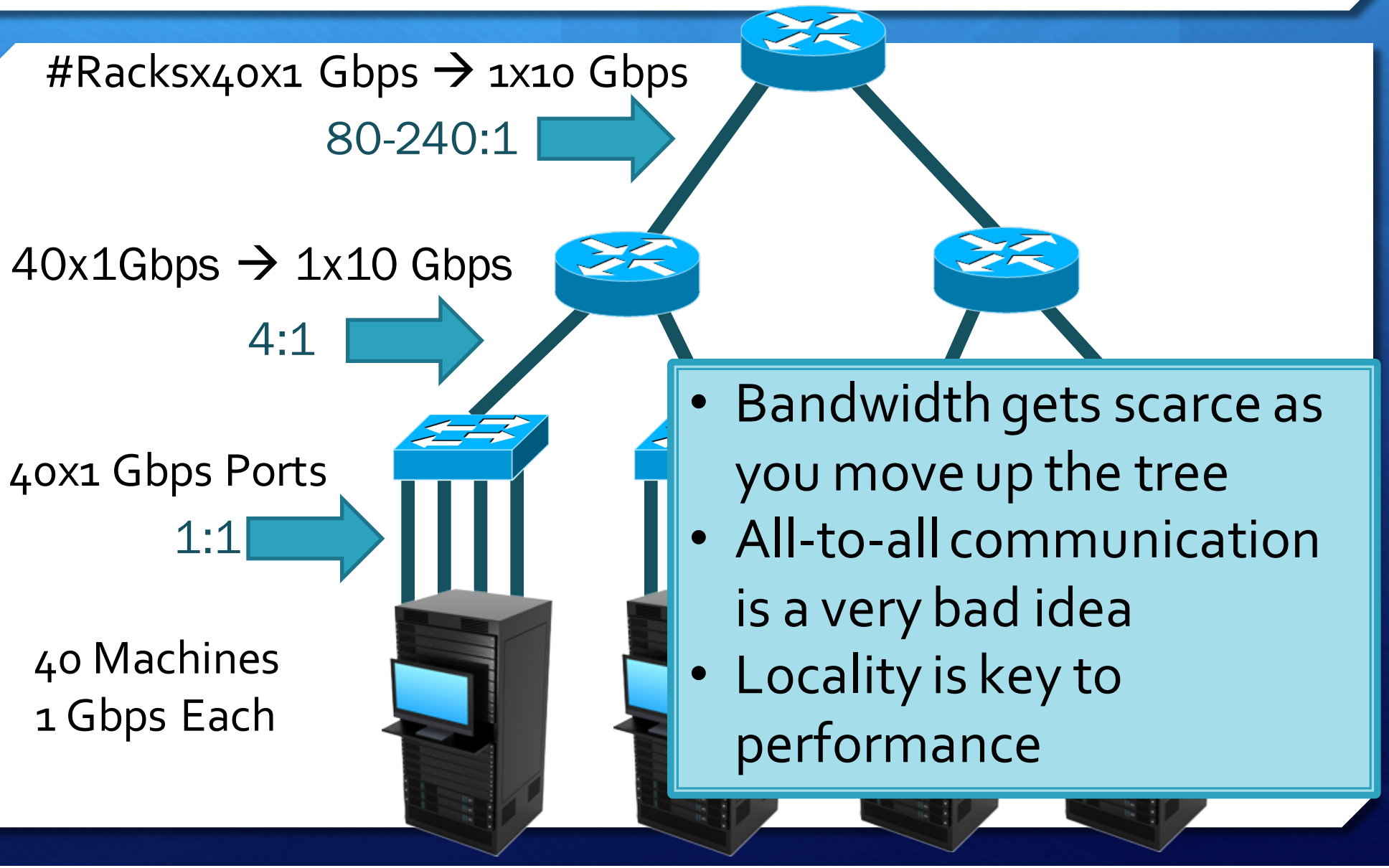
Today's Topic: Network Problems

- Data centers are data-intensive
- Hardware can handle it
  - CPUs scale with Moore's Law
  - RAM is almost as fast as CPU
  - RAID and SSDs are pretty fast
- Current network cannot handle it
  - Slow, not keeping pace over time
  - Wiring is a nightmare
  - Expensive
  - Hard to manage
  - Non-optimal protocols

Outline

- Introduction
- Network topology and routing
  - Fat tree
  - Wireless in data centers
  - Optical in data centers
- Transport protocols

Problem: Oversubscription



- Bandwidth gets scarce as you move up the tree
- All-to-all communication is a very bad idea
- Locality is key to performance

Oversubscription can be Harmful

- Ruin your network
  - Limits application scalability
- Problem is about to get worse
  - 10 GigE servers are more affordable
  - 128 port 10 GigE routers are not
- A issue of the core routers
- Get rid of the core routers by using cheap switches?
  - Maintain 1:1 subscription ratio

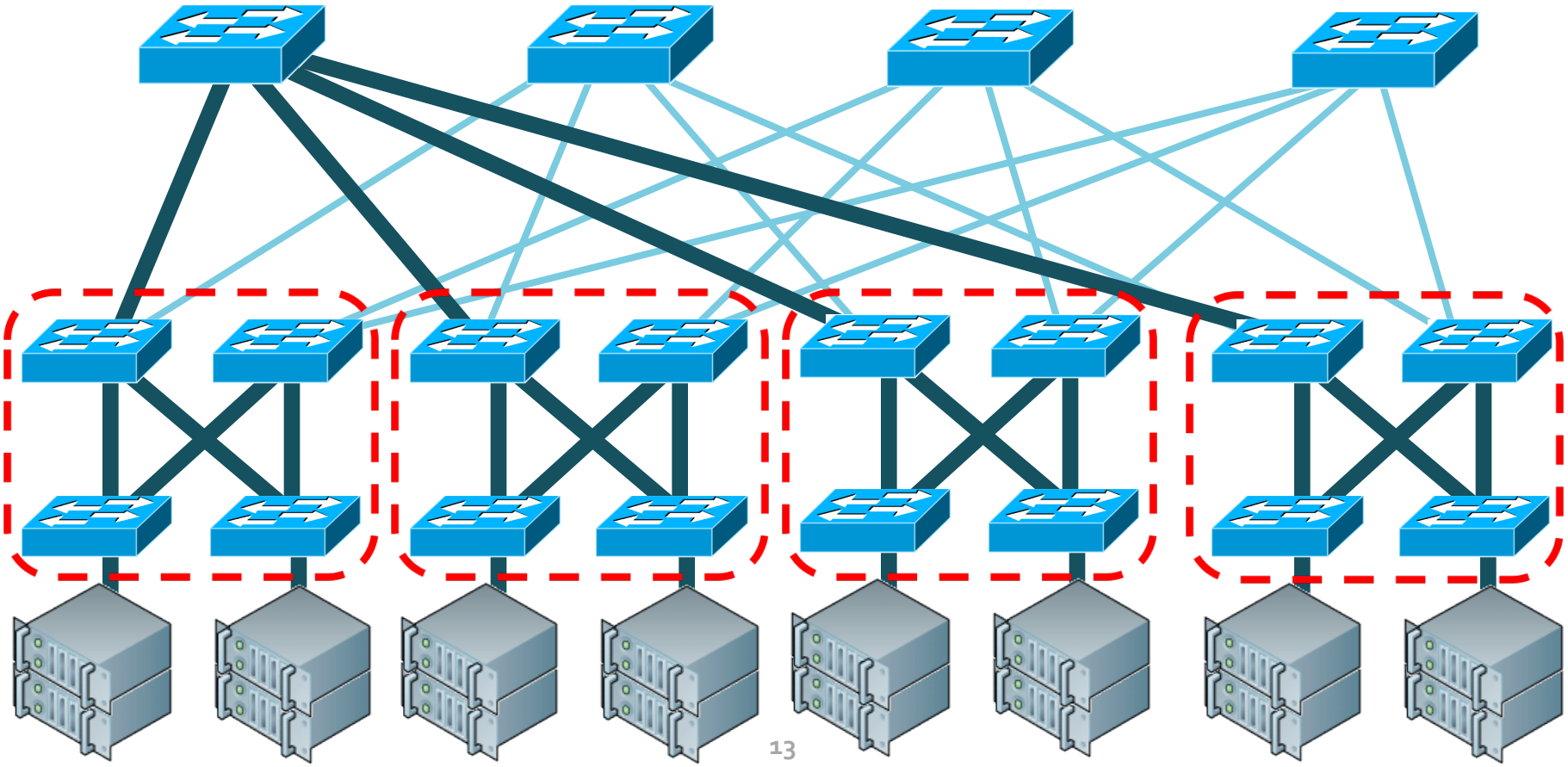
Fat-tree

To build a K-ary fat tree

- K-port switches
- K pods, each with K switches
- $K^3/4$ hosts
- $(K/2)^2$ core switches

In this example K=4

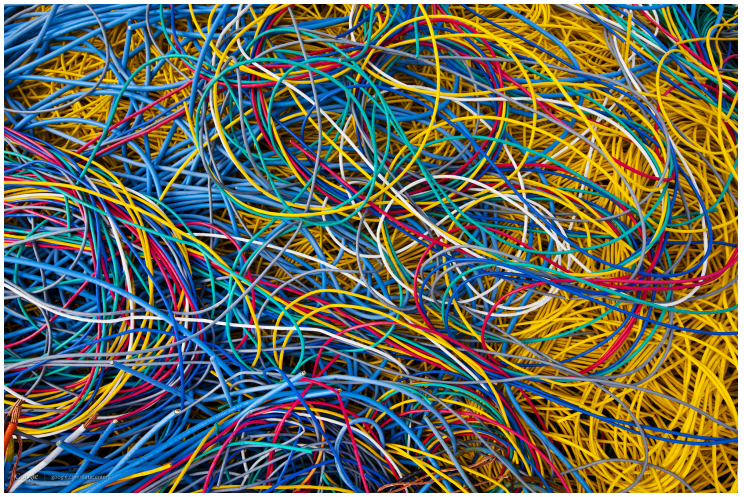
- 4-port switches
- 4 pods, each with 4 switches
- $K^3/4 = 16$ hosts
- $(K/2)^2 = 4$ core switches



Fat-tree

- The good
  - Full bisection bandwidth
  - Low-cost, commodity hardware
  - Redundancy for failover
- The bad
  - Custom routing (NetFPGA)
  - Wiring is a nightmare

    # of wires: → $3K^3/4$
    - 48 port switches = 82944

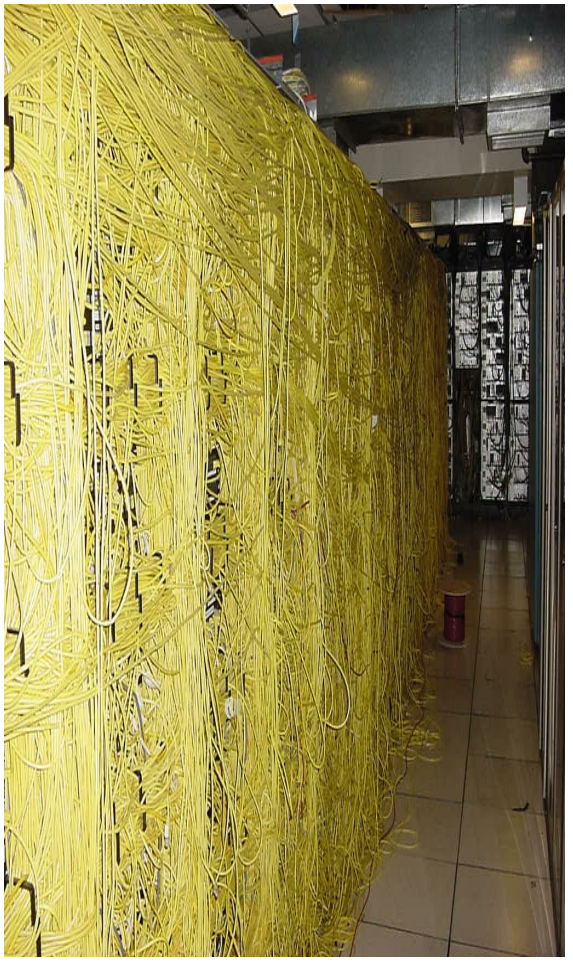


Facebook Data Center Fabric

- https://www.youtube.com/watch?v=mLEawo6OzFM

Limitations of Wired Interconnects

- Wiring is complex and costly
  - Planning, deploying, testing 10K+ fibers
  - Takes several weeks or even months
- Difficult to change wiring
  - High labor cost
  - Significant interruptions to operations
- Overprovisioning is difficult
  - Traffic demands unpredictable
  - Limited by hardware costs



Dealing with Traffic Hotspots

- Measurements show sporadic congestion losses caused by traffic hotspots
  - Traffic hotspots are unpredictable, can appear anywhere
  - Can double failure rate for some jobs

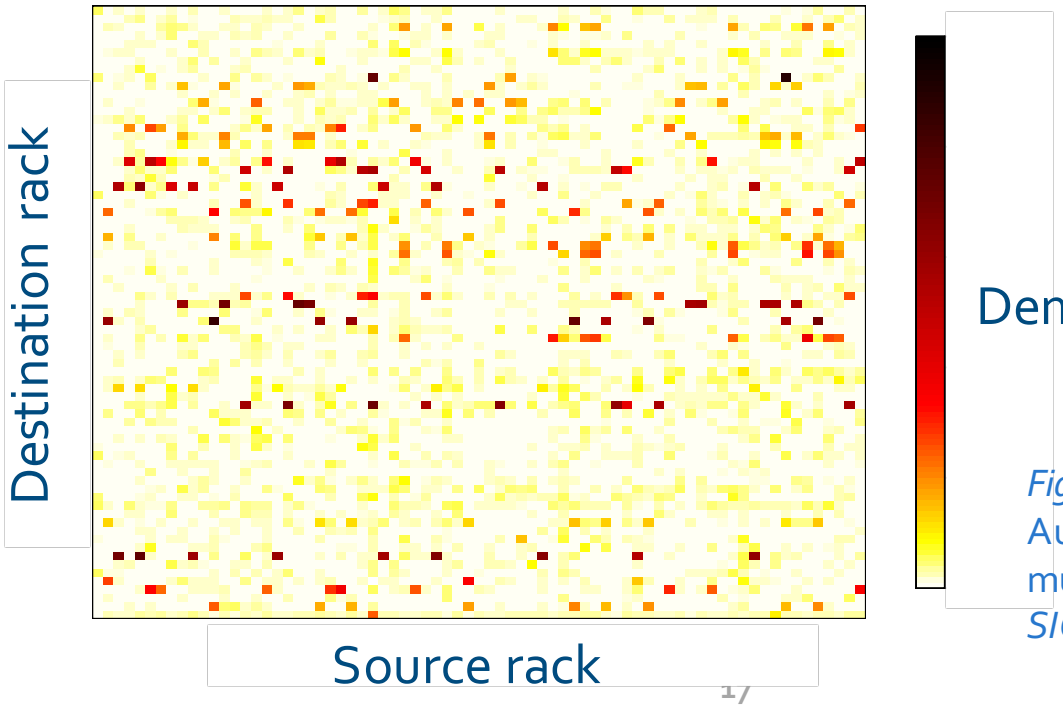


Figure source: Halperin, D., et al. Augmenting data center networks with multi-gigabit wireless links. In *Proc. of SIGCOMM* (2011)

Dealing with Traffic Hotspots

- Measurements show sporadic congestion losses caused by traffic hotspots
  - Traffic hotspots are unpredictable, can appear anywhere
  - Can double failure rate for some jobs
- Hard to add bandwidth using wires
  - Do not know where and when to add capacity
  - Rewiring is complex, high labor cost, interrupts current operation

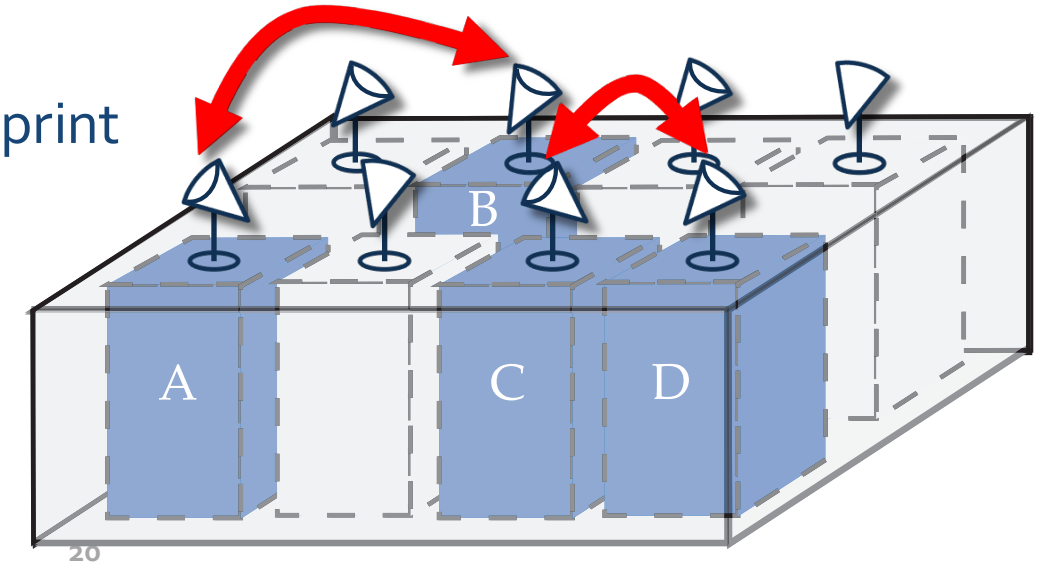
Need alternative solutions!

The Need of Flexible Interconnects

- Not always need continuous communication
  → Full bisection bandwidth is not always necessary
- Traffic bursts do exist
  → Need enough bandwidth to support
- The trend: flexible network interconnects to add bandwidth on demand
  - **Wireless**
  - **Optical**

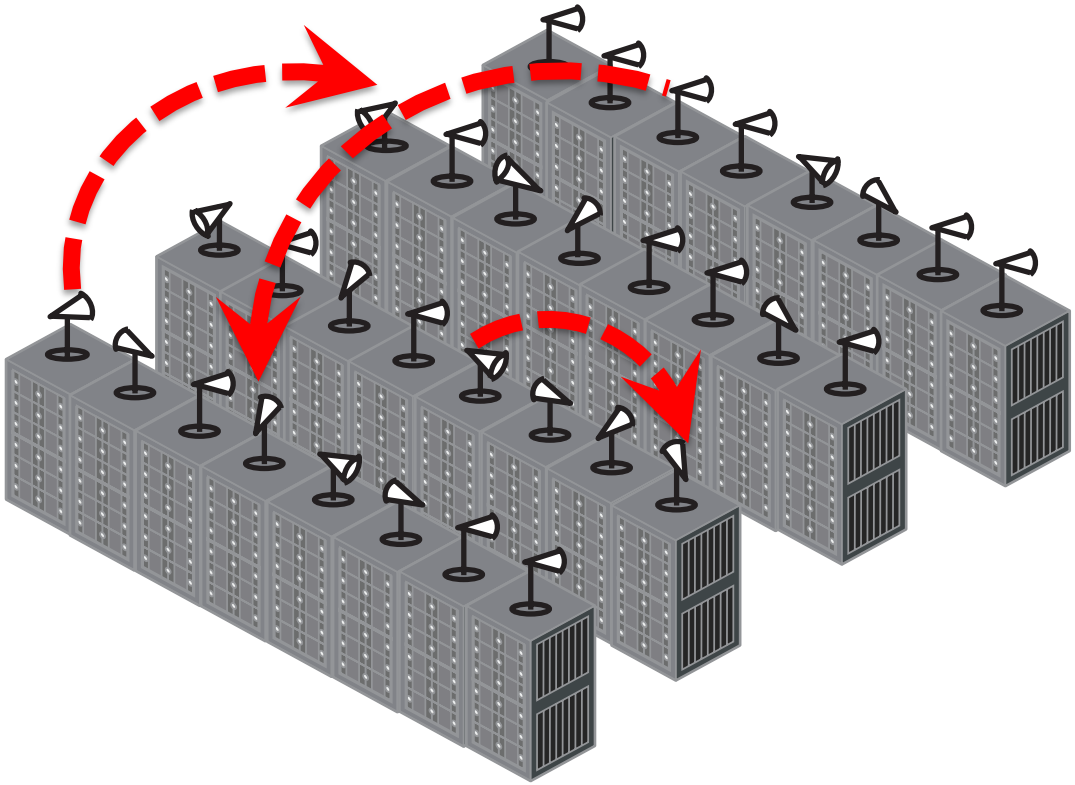
Augmenting via Wireless Links

- Key benefit: on-demand links
  - Create links on-the-fly at congestion hotspots
  - Adapt to traffic dynamics
- New wireless technology: 60 GHz beamforming
  - Multi-Gbps data rate
  - Small interference footprint



Flexible Wireless Links in Data Centers

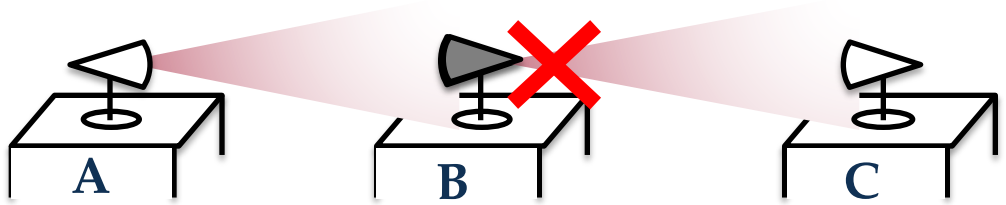
- Connect any rack pair wirelessly to address dynamic traffic hotspots



Hard to do so using 60GHz transmissions

Key Challenges

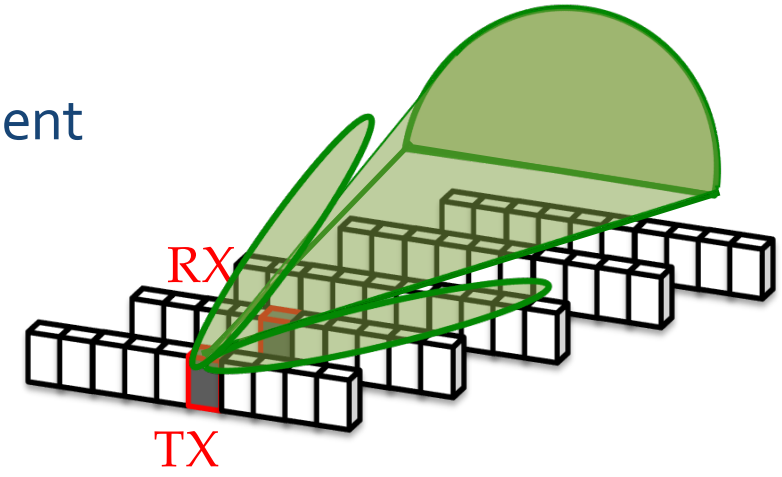
- **Link blockage**: small obstacles block the link



→ Must use multi-hop forwarding

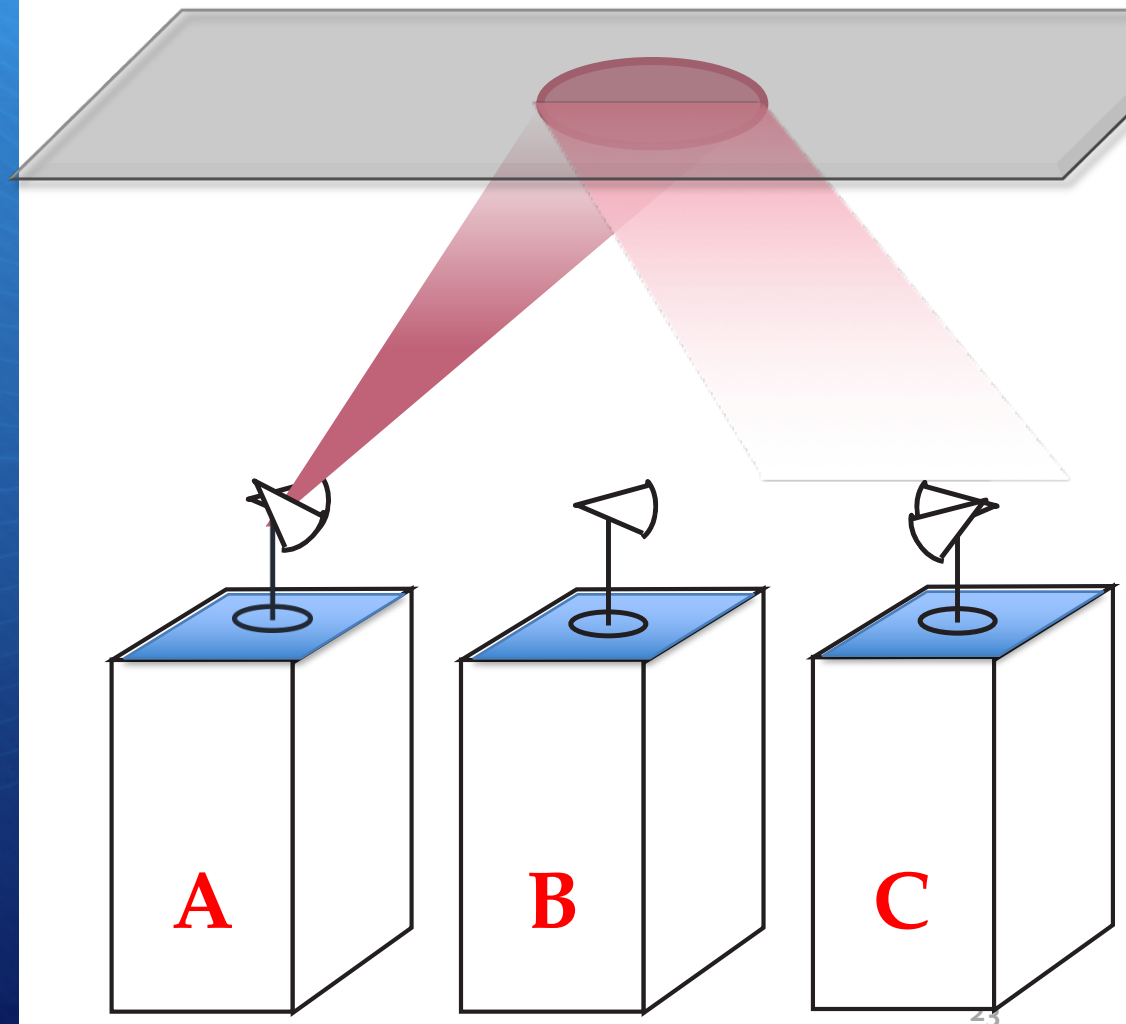
- **Radio interference**: beam interferes with racks in its direction
  - Exacerbated by dense rack deployment
  - Signal leakage makes it worse

→ Links interfere with each other



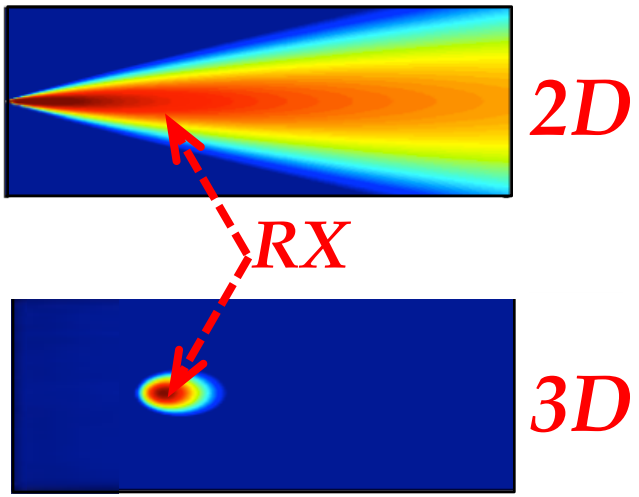
Wireless 3D Beamforming

Connect racks by reflecting signal off the ceiling!

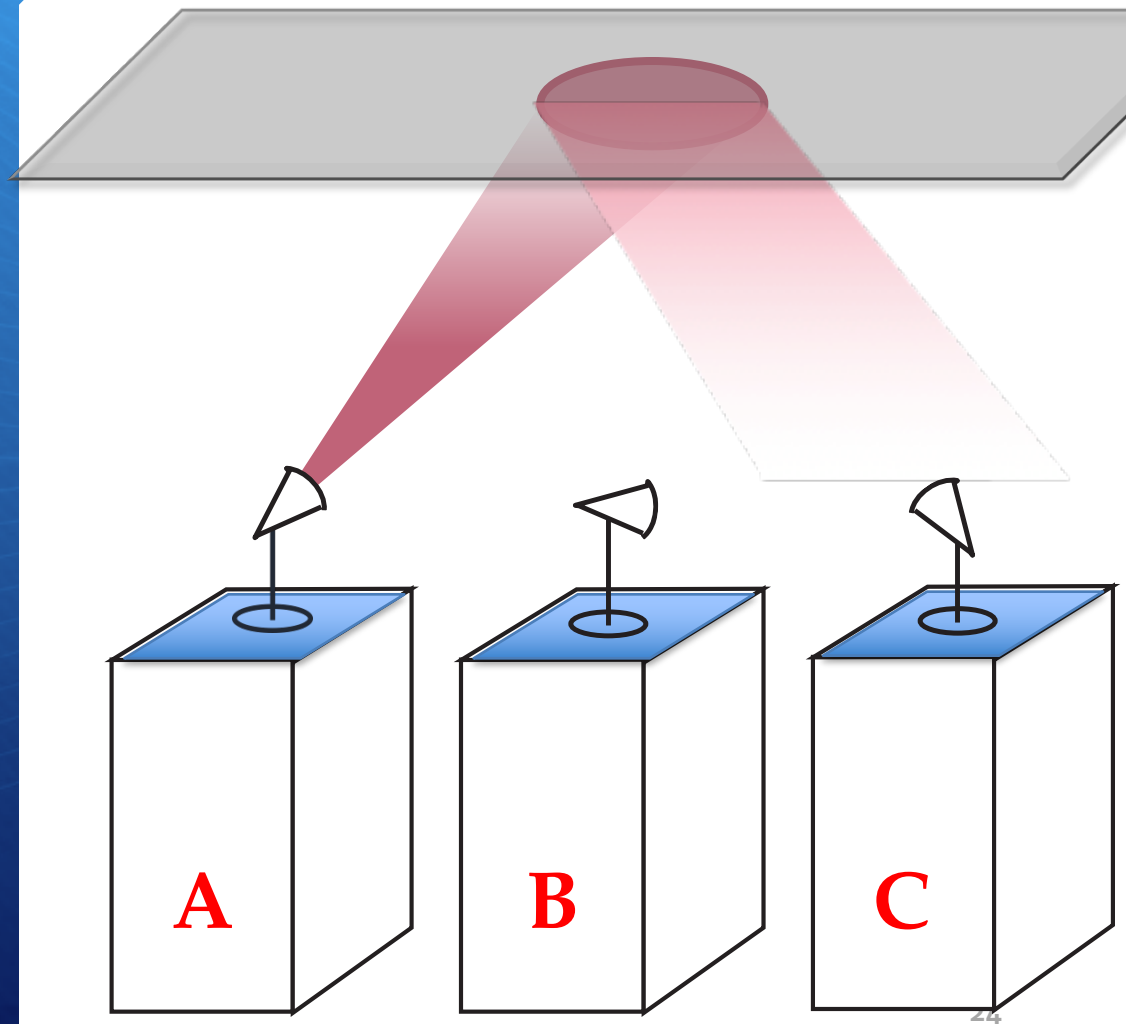


Key Benefits

- ✔ No more link blockage
- ✔ Much smaller interference

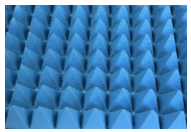


Easy Setup



Reuse existing hardware, low maintenance cost!

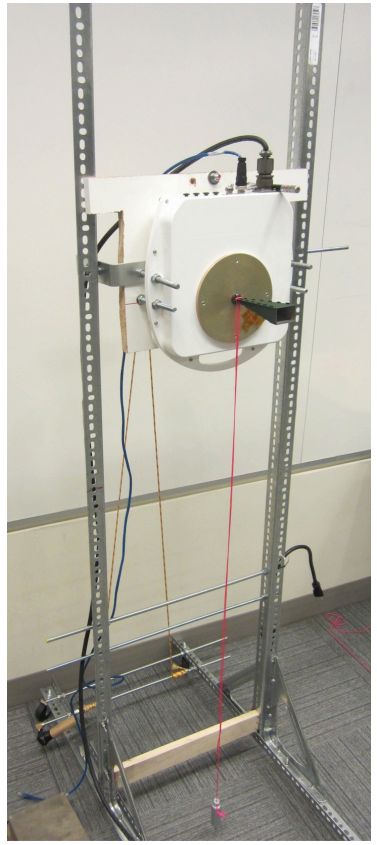
Reflector



Absorber

Testbed Measurements

- Off-the-shelf 60GHz radio and horn antenna
- Result summary
  - Extended link connectivity
    - No energy loss in reflection, negligible impact on data rate
  - Smaller interference footprint
    - Match simulation results
  - Robustness to alignment errors
    - 1° alignment error has negligible impact



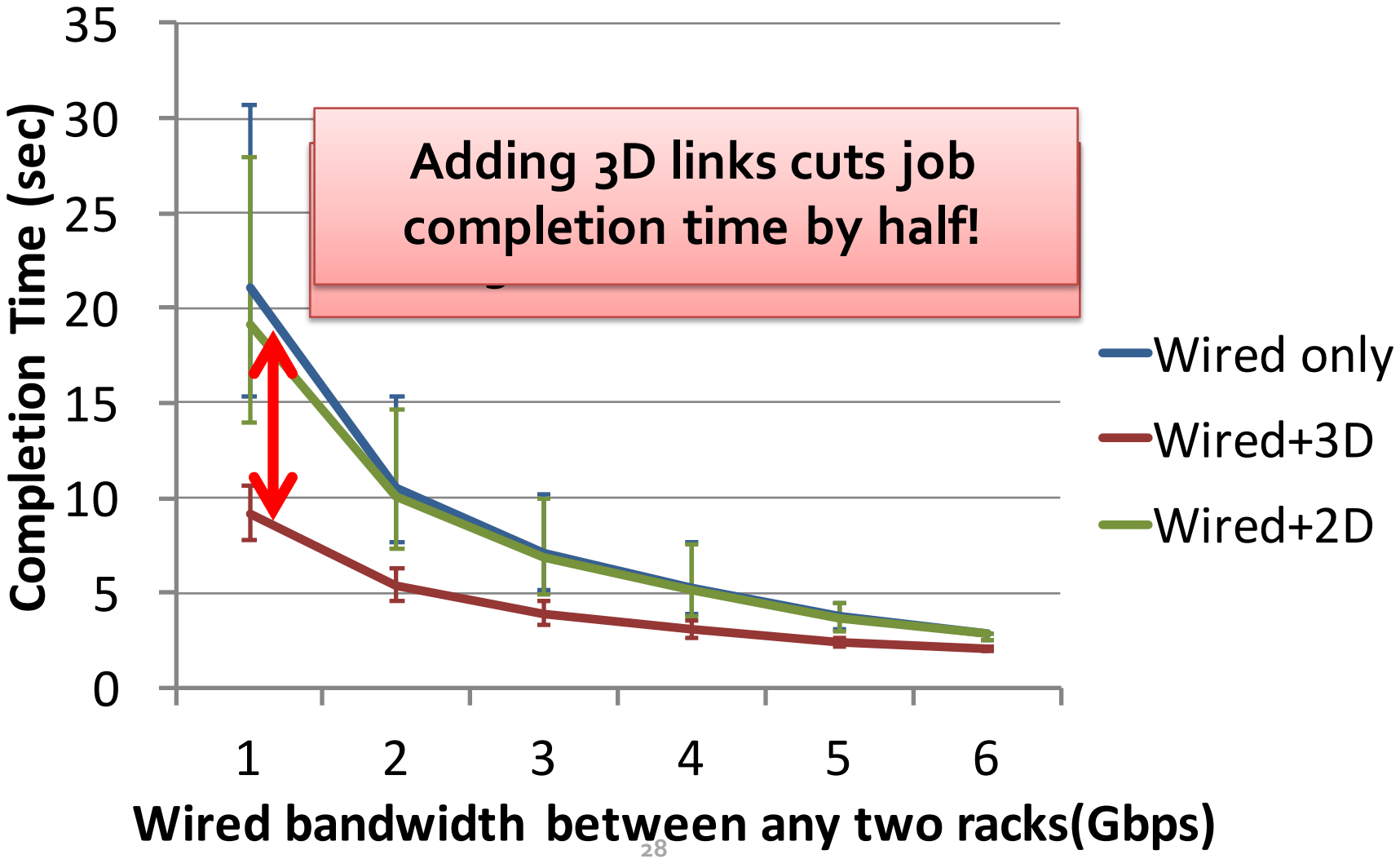
Addressing Traffic Hotspots

- Challenges of scheduling 60GHz links
  - Minimize interference
  - Respond to short-lived traffic hotspots
  - Antenna rotation delay comparable to transmission time
- Scheduler design
  - Online scheduling accounting for accumulative interference
  - Rotation-aware radio selection

Results of Addressing Traffic Hotpots

- Data center setting
  - 250 racks (5000 servers), 8 radios/rack
- Synthetic hotspot traffic based on popular workloads
- Create wireless links for hotspots
  - Assume perfect traffic allocation so that wired and wireless links finish at the same time

Results of Addressing Traffic Hotpots

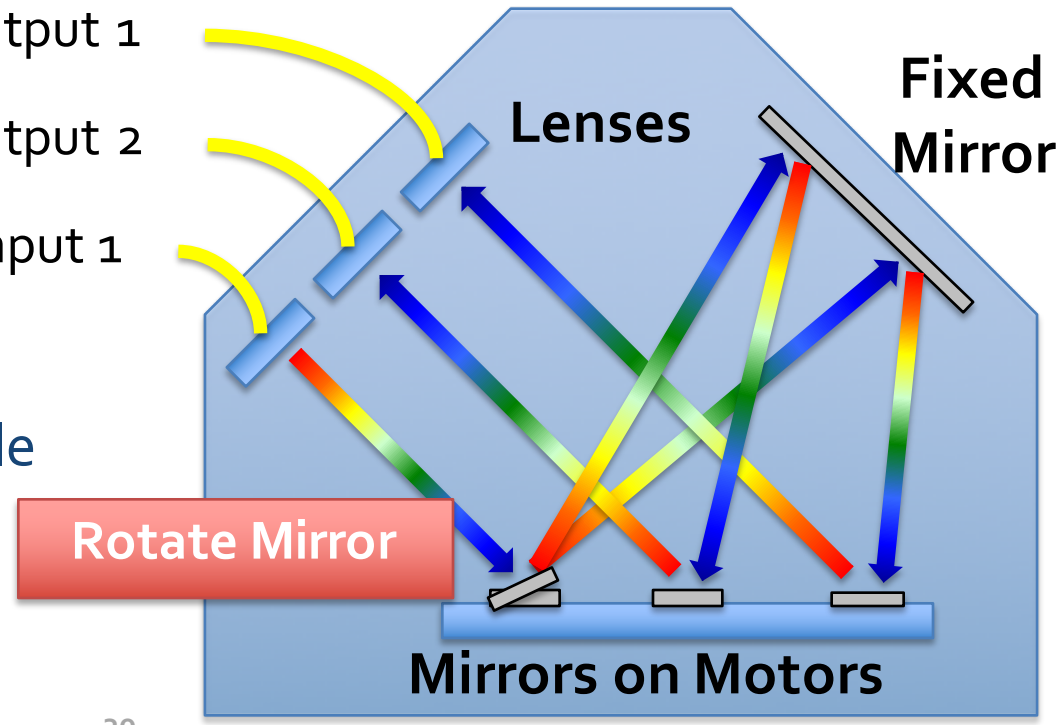


3D Beamforming Summary

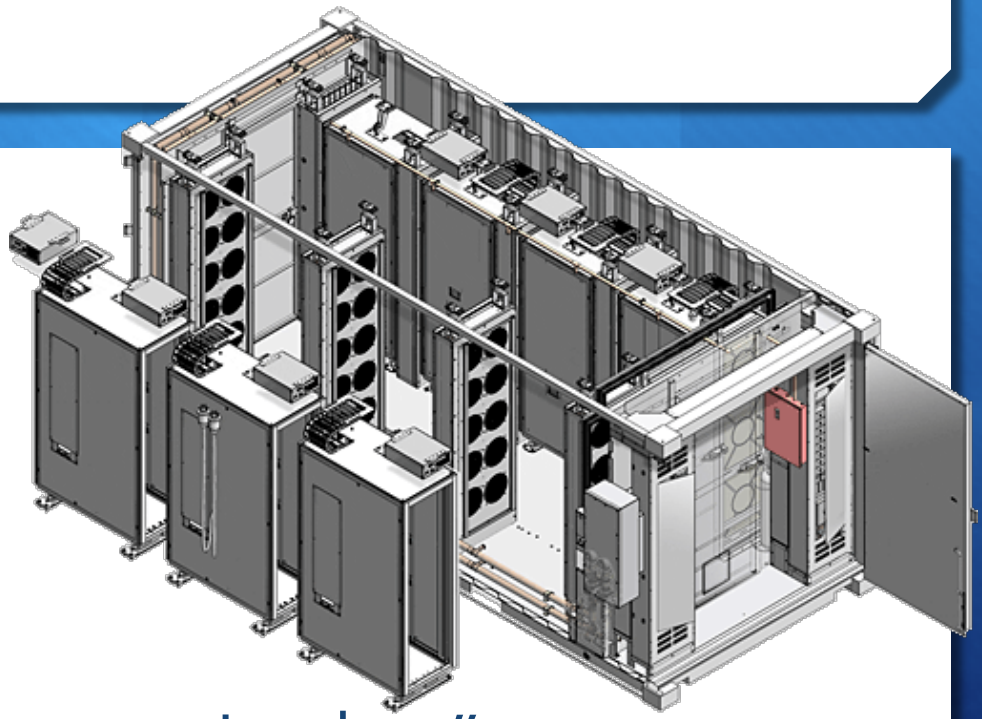
- New wireless primitive added into data center toolbox
  - Overcomes key challenges of using wireless in data centers
  - Opens up more opportunities of using wireless in data centers
- Practical issues
  - Traffic scheduling
  - Reflector placement, impact on data center cooling

Augmenting via Optical Links

- Optical circuit switch
  - Uses mirrors to bounce light from port to port
  - No decoding
  - Mechanically adjust mirrors in milliseconds
- Wavelength division Multiplexing
  - Single port carries multiple streams concurrently!



Example: Connecting Modular Datacenters



- Shipping container: "datacenter in a box"
  - 250-1K servers per container
- How do you connect the containers?
  - Copper cable constraint: physical distance matters (10GigE -> 10 meters)

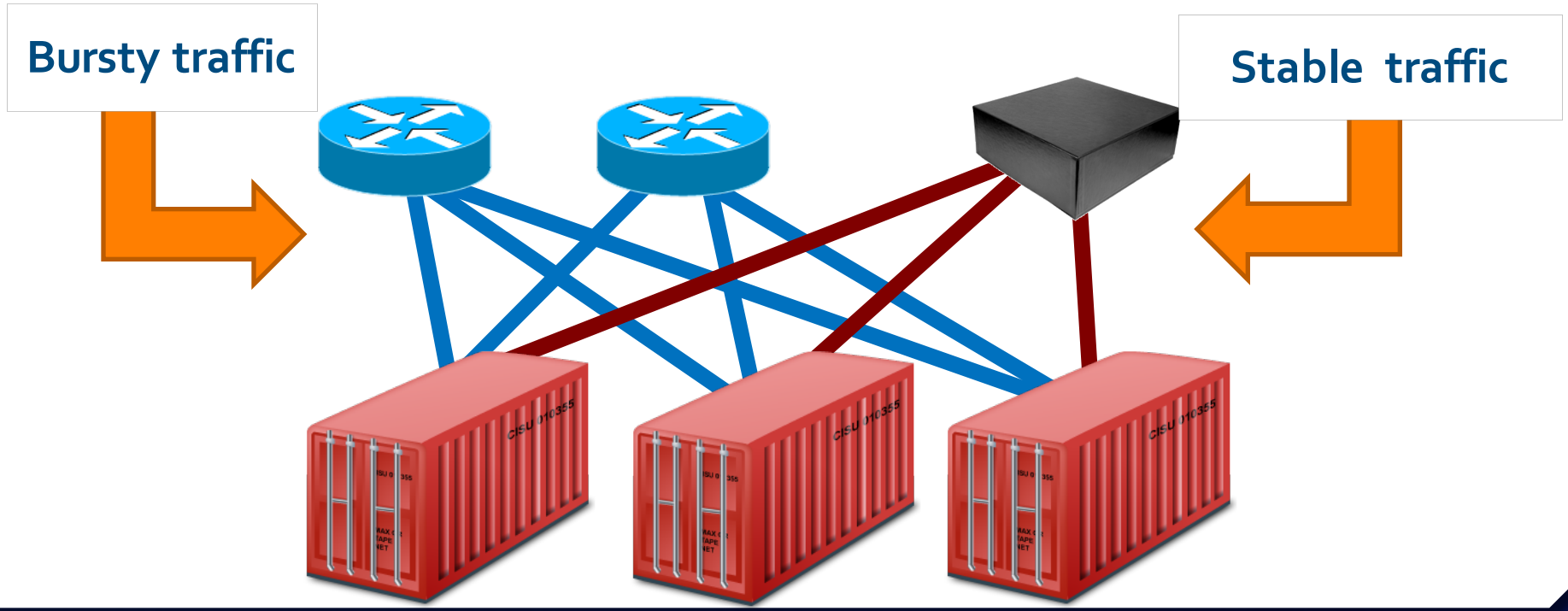
Helios: Datacenters at Light Speed

Packet switch network

+ Electrical packet switches
+ Connect all containers

Optical circuit network

+ Optical circuit switches
+ Direct container-to-container links on demand



Helios Overview

- Complete system design
  - Traffic measurements from container switches
  - Traffic estimation and partition
  - Optical link scheduling
- Real hardware implementation
  - One 64-port optical circuit switch, 24 servers
- Summary
  - Interesting direction
  - Not ready for prime-time

Outline

- Introduction
- Network topology and routing
- Transport protocols
  - Google and Facebook
  - DCTCP
  - D3

Transport on the Internet

- TCP optimized for the WAN
  - Fairness
    - Slow-start, AIMD convergence
  - Defense against network failures
    - 3-way handshake, reordering
  - Zero knowledge congestion control
    - Self-induces congestion, loss equals congestion

Data Center is not the Internet

- The good
  - Possibility to make unilateral changes
    - Homogeneous hardware/software
    - Single administrative domain
  - Low error rates
- The bad
  - Latency is tiny (250µs in absence of queuing)
  - Little statistical multiplexing
    - One long flow may dominate a path
    - Cheap switches have queuing issues
  - Incast

Problem: Queue Buildup

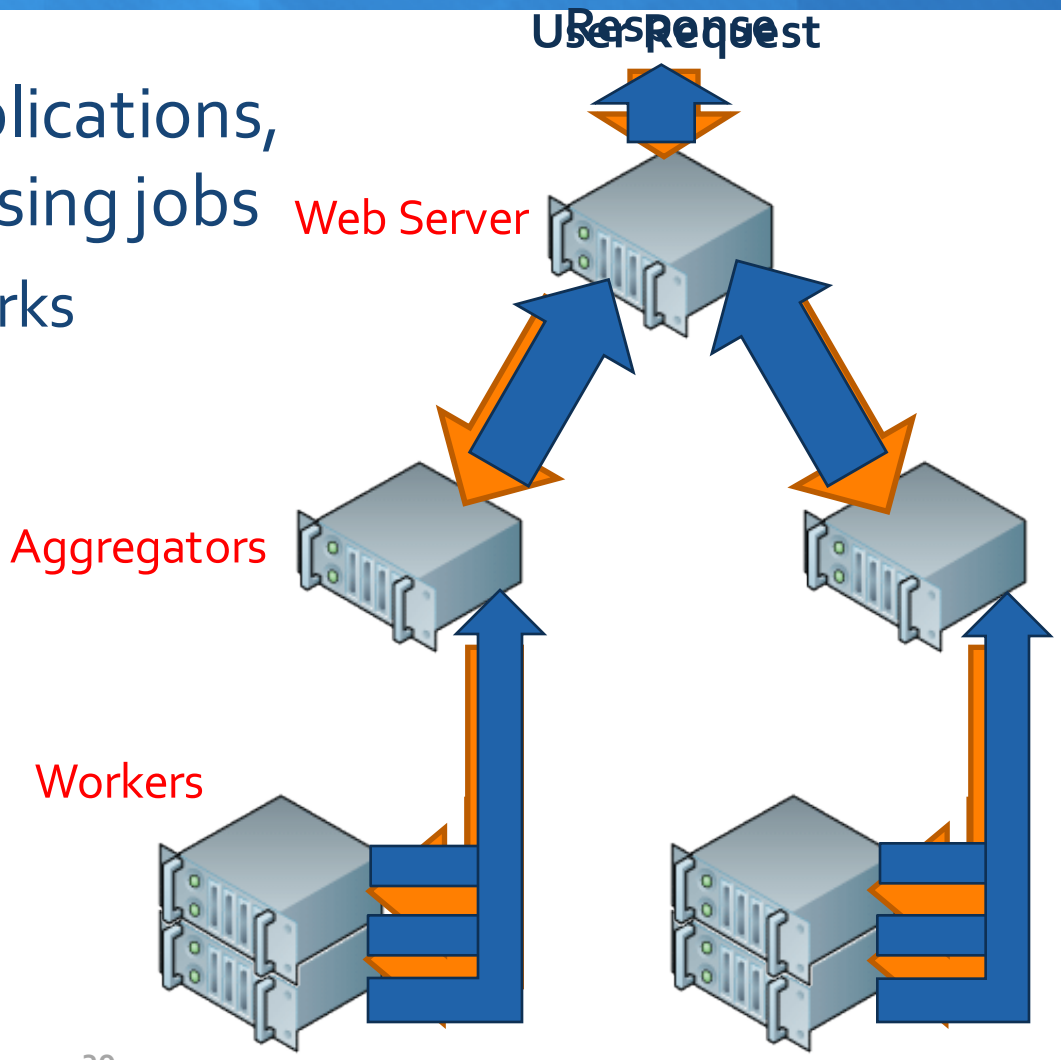
- Long TCP flows congest the network
  - Ramp up, past slow start
  - Don't stop until they induce queuing + loss
  - Oscillate around max utilization
- Short flows can't compete
  - Never get out of slow start
  - Deadline sensitive!

Conclusion

- Data center is a super hot topic right now
  - Topology, routing
  - Network stack
  - Heat, power
  - Management
  - Applications: Hadoop, Dynamo, Cassandra, NoSQL
- Tough for academic research
  - No real data centers to test around

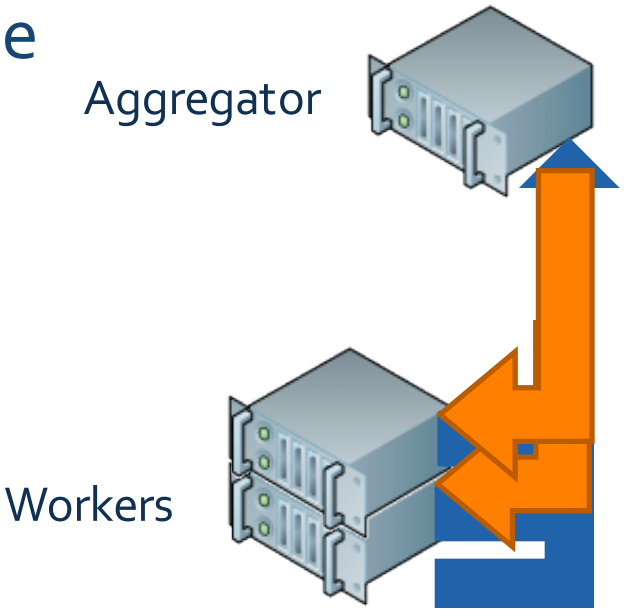
Partition/Aggregate Pattern

- Common for web applications, and even data processing jobs
  - Search, social networks
  - Dryad, MapReduce
- Responses under deadline
  - 230~300ms



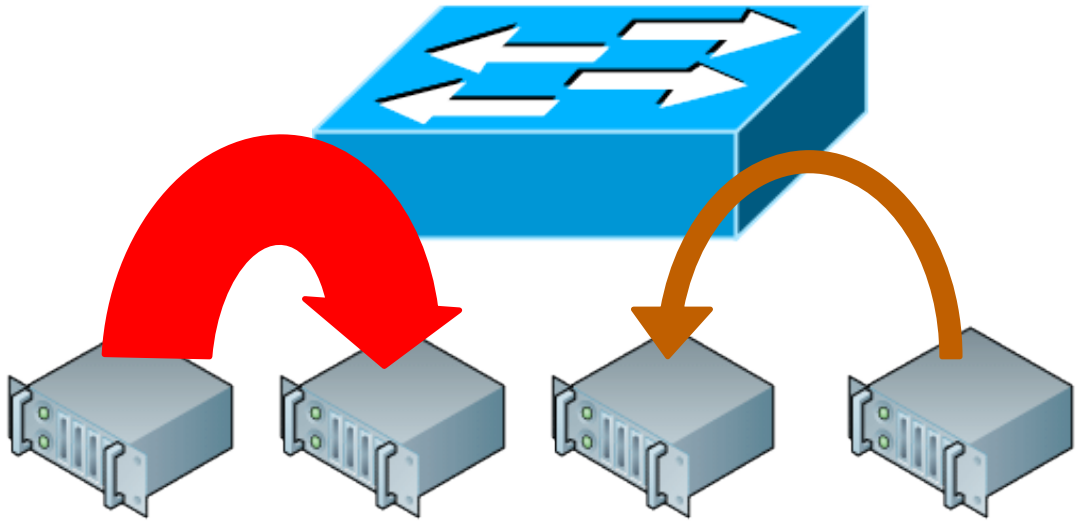
Problem: Incast

- Aggregator sends out queries to a rack of workers
  - 1 aggregator, 39 workers
- Each query takes the same time to complete
- All workers answer at the same time
  - 39 flows → 1 port
  - Limited switch memory
  - Limited buffer at aggregator
- Result: packet losses ☹



Problem: Buffer Pressure

- In theory, each port on a switch should have its own buffer
- Cheap switches share buffer memory across ports
  - Fat flows can congest the thin flow!



Industry Hacks

- Limits search worker response to one TCP packet
- Use heavy compression to maximize data

- Custom engineer to use UDP
- Connection pooling: share buffer pool per thread

Dirty Slate Approach: DCTCP

- Goals
  - Alter TCP to achieve low latency
  - Work with shallow buffered switches
  - Do not modify apps, switches, or routers
- Idea
  - Scale window in proportion to congestion
  - Use existing ECN functionality
  - Turn single-bit congestion info to multi-bit

ECN and ECN++

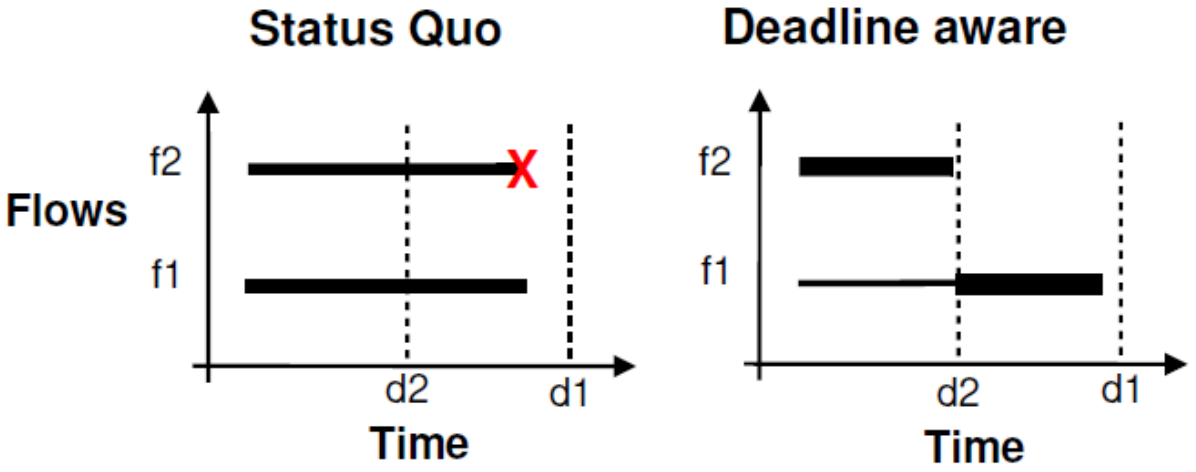
- Original ECN
  - Switches mark EC bit of packet if there is congestion
  - Receivers echo the EC bit in ACK
  - EC in ACK stays set until sender clears with CWR
- Problem: feedback is binary
- DCTCP:
  - Receiver echoes the actual EC bits
  - Sender estimate congestion ($0 \leq \alpha \leq 1$) each RTT based on the fraction of marked packets
  - cwnd = cwnd * $(1 - \alpha/2)$

Shortcomings of DCTCP

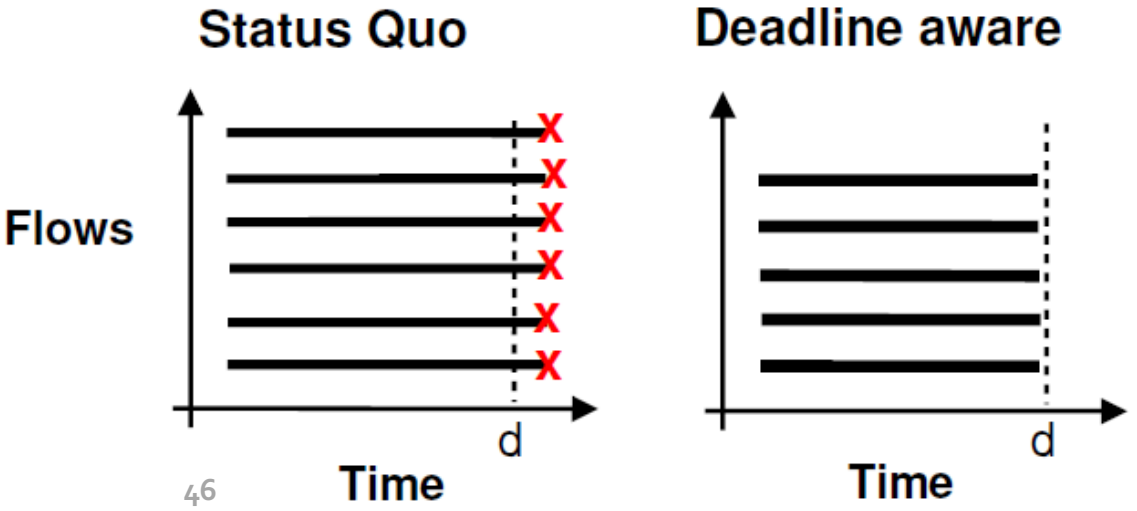
- No scheduling, cannot solve incast
- Queries may still miss deadlines
  - Flows do not help if they miss the deadline
- Network throughput is not the right metric
  - Application goodput is
- TCP/DCTCP is oblivious to deadline

The Need of Deadline-Awareness

- Case #1
  - Fair share causes bot to fail
  - Unfair share enables both to succeed



- Case #2
  - Allowing all makes all fail
  - Quenching one leads to better goodput



Clean Slate Approach: D^3

- **Key insight**: ask for the bandwidth required to meet the deadline
  - Hosts use flow size and deadline to request bandwidth
  - Routers measure utilization and make soft-reservations
- End-hosts:
  - Estimate rate = flow_size / deadline
  - Send request in the header
- Routers:
  - Greedily assign bandwidth to maximize satisfied requests
  - Signs allocated rate fed back to sender via ACK

D3 Overview

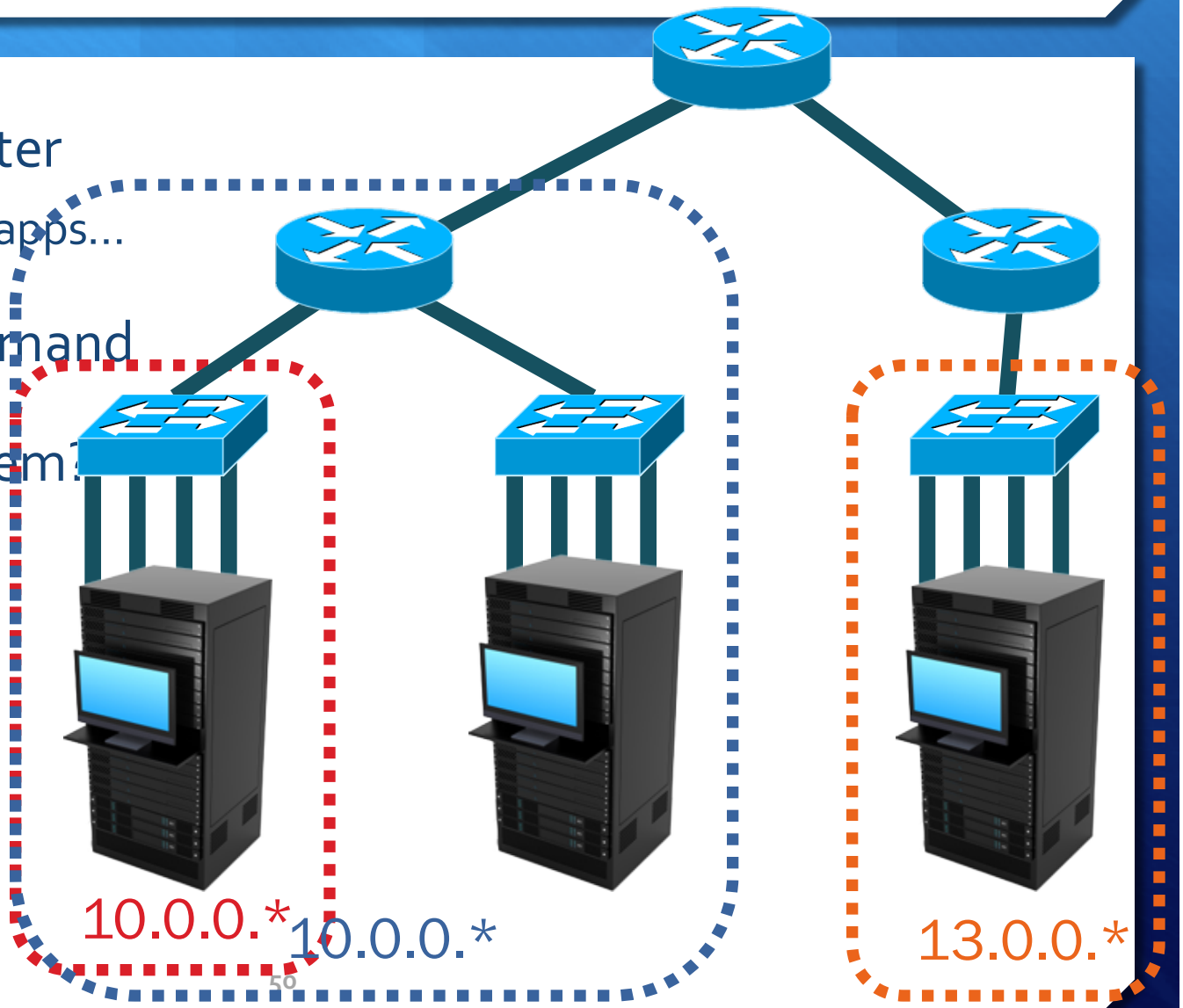
- Benefits
  - Higher goodput under heavy load
  - Better support deadline-bound apps
- Challenges ahead
  - All-or-nothing deployment
    - Not for incremental deployment, may not play nice with TCP
  - Complexity in the switch
    - XCP ran in an FPGA
  - Application level changes
    - Deadline changes

Open Problems

- Data center measurement data
  - Real traces are really hard to get
  - How representative are they?
    - Application-dependent
- Hard to quantify research results
  - Cross-paper comparisons
  - Reproducibility

Problem: Routing

- In a typical data center
  - Multiple customers, apps…
- VM allocation on demand
- How do we place them?
  - Performance
  - Scalability
  - Load-balancing
  - Fragmentation



Virtual Layer-2 (VL2)

- Traffic-oblivious routing: insert a layer 2.5 into the network stack
  - Translate virtual IPs to actual IPs
  - Directory servers maintain the mapping
- Benefits
  - No more VLANs, easy VM migration, multi-path load balancing
  - No modifications to apps
- Issues
  - Must modify host Oss
  - Directory servers need to scale