Lecture 14:
Data Center Networking (review+D^3) + Data-driven Networking Design

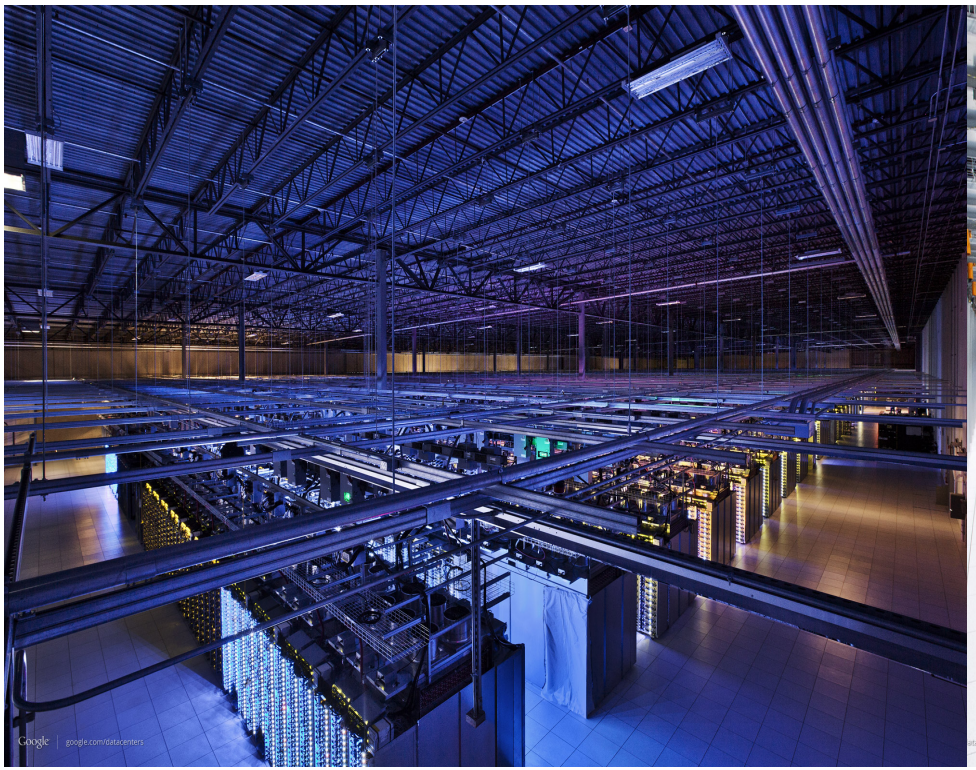
Schedule

- Today: Data driving network design
- Wed: P2P networks
- Mon 12/3: Final review, Q&A
- Wed 12/5: Second midterm

- Project 3: Due 12/11, 11:59pm

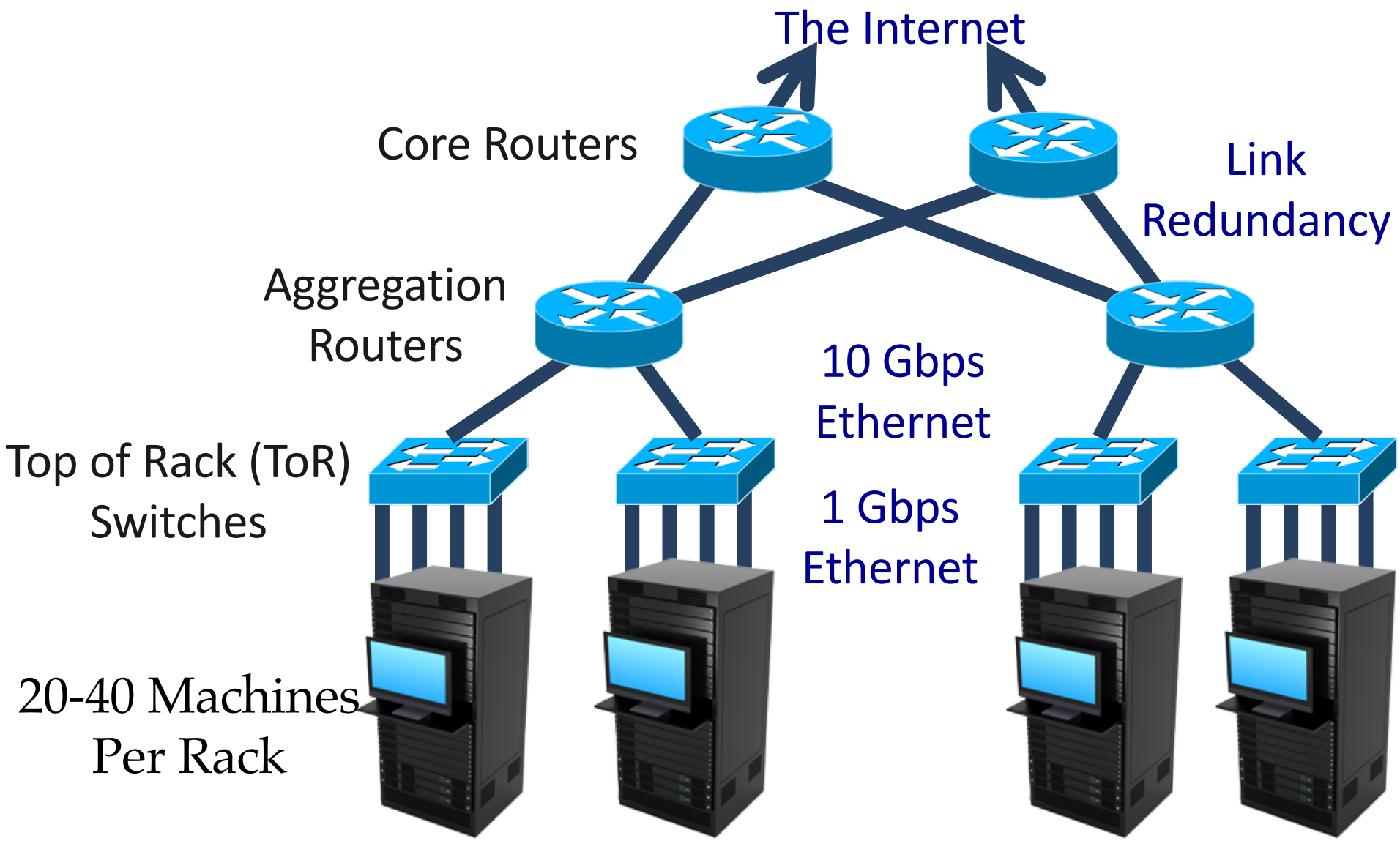
Today's Data Centers

- 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

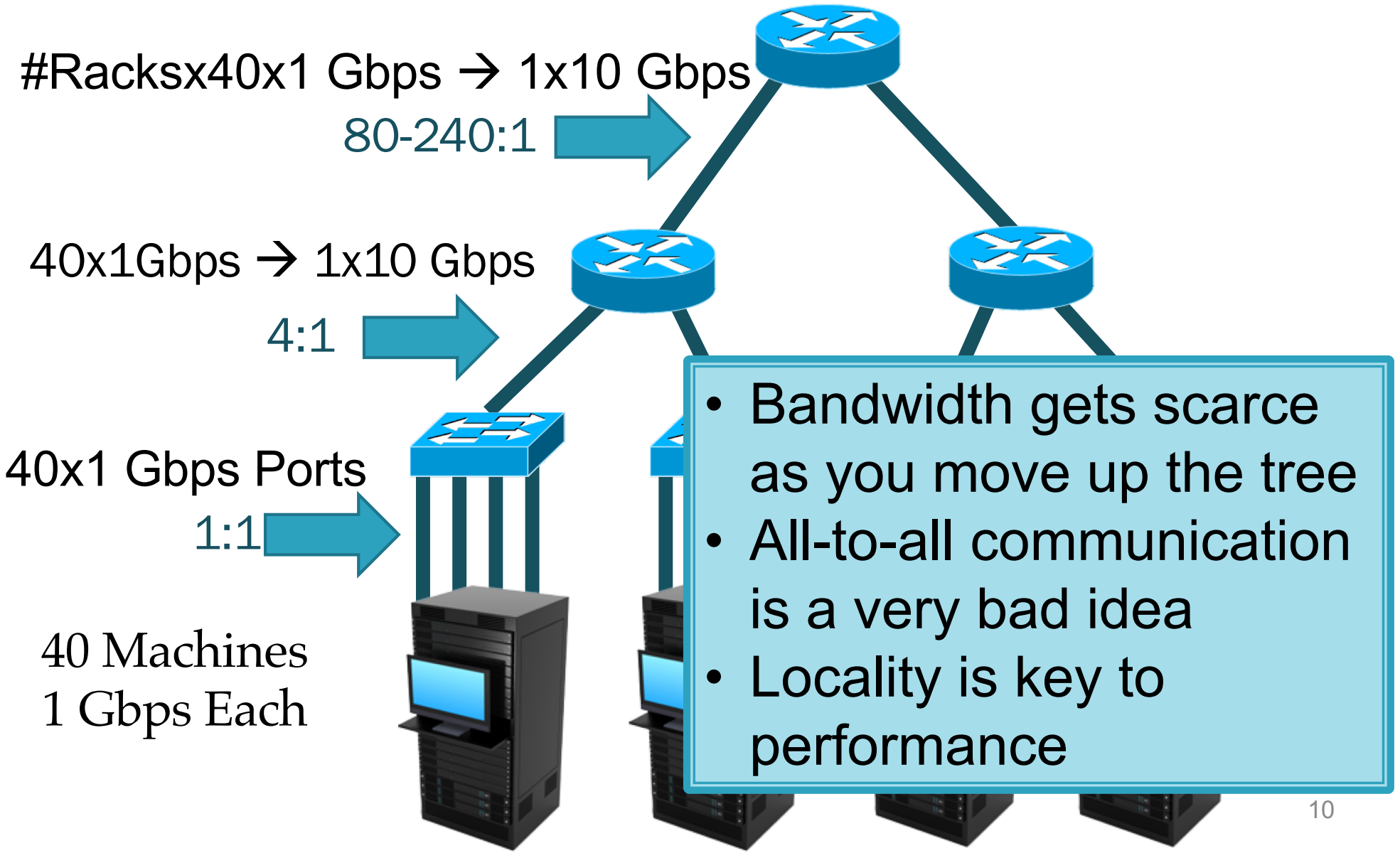
Today's Topic: Network Problems

- 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

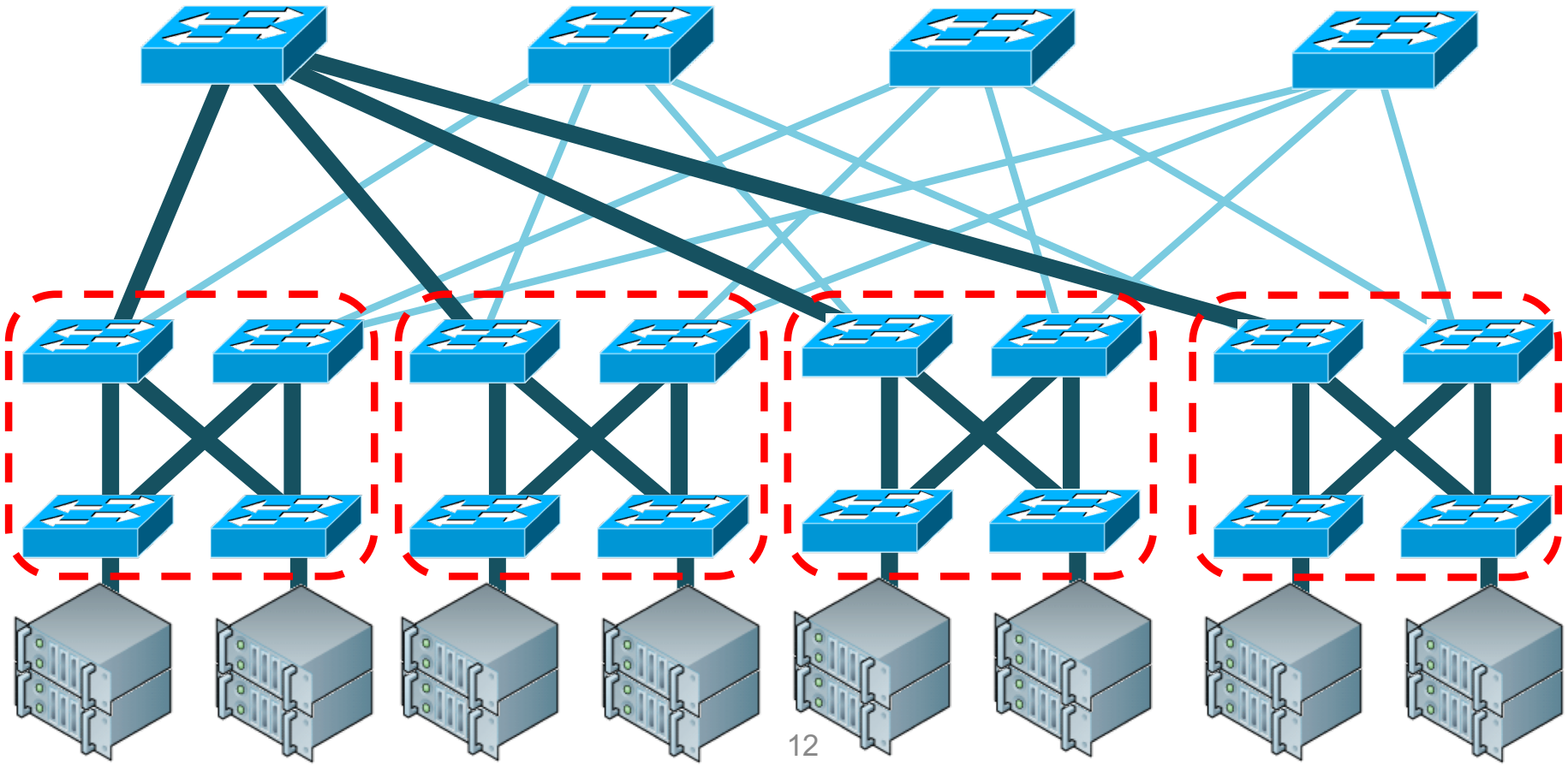
Oversubscription can be Harmful

- 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

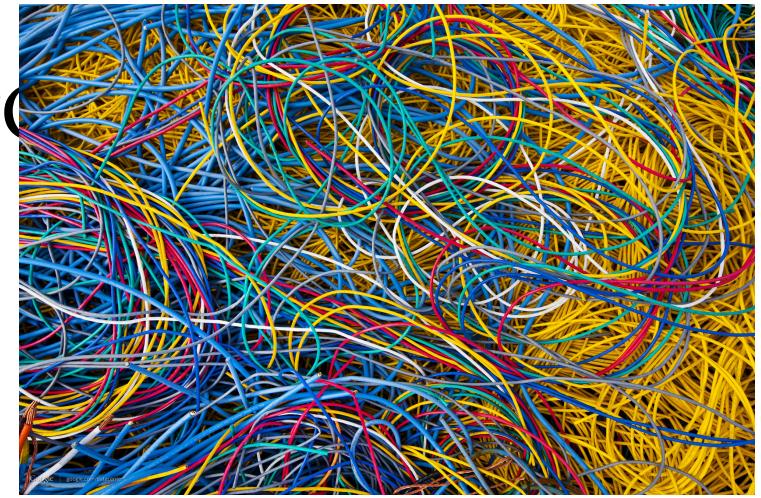
To build a K-ary fat tree
- K-port switches
- K pods, each with K switches
- $K^3/4$ hosts
- $(K/2)^2$ core switches

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 (NetFP
  - Wiring is a nightmare
    # of wires: → $3K^3/4$
    - 48 port switches = 82944

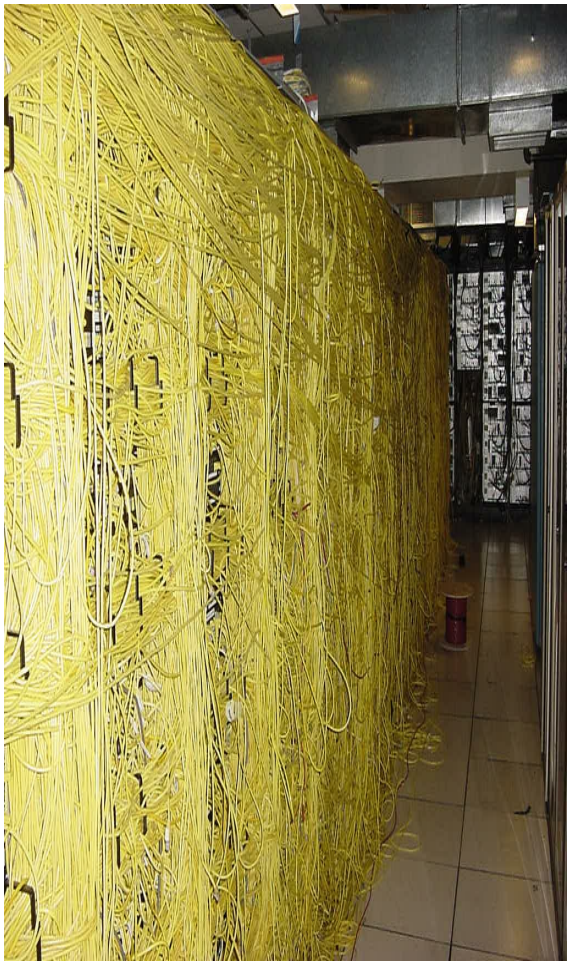


Facebook Data Center Fabric

- 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

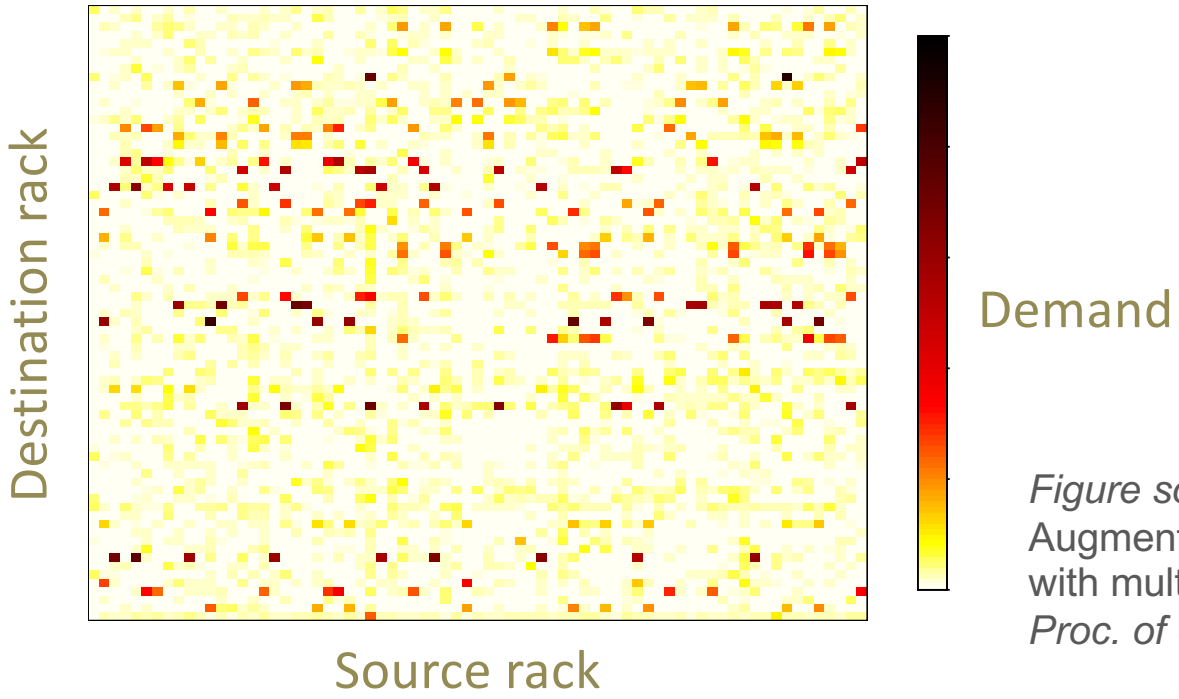


Figure source: Halperin, D., et al. Augmenting data center networks with multi-gigabit wireless links. In *Proc. of SIGCOMM* (2011)

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

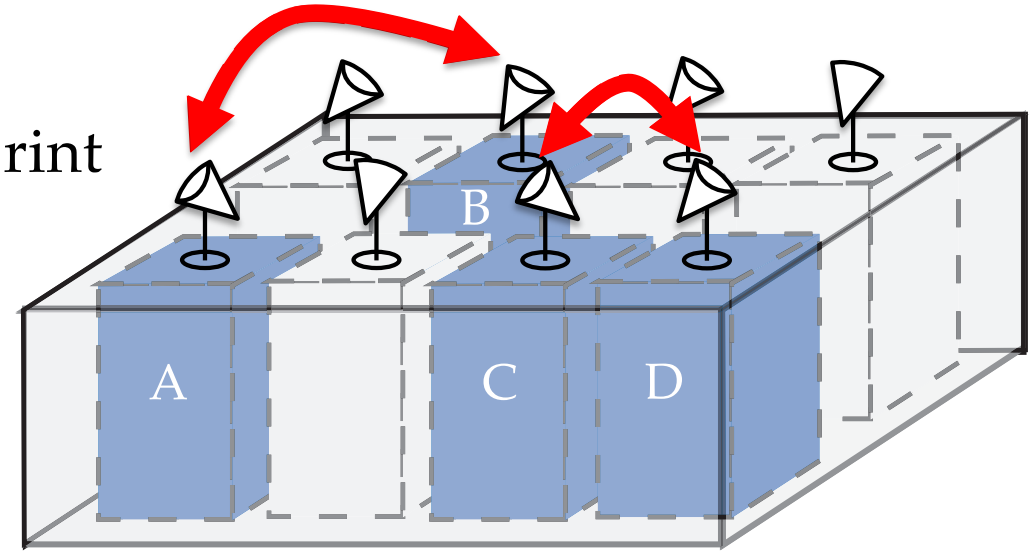
Need alternative solutions!

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**

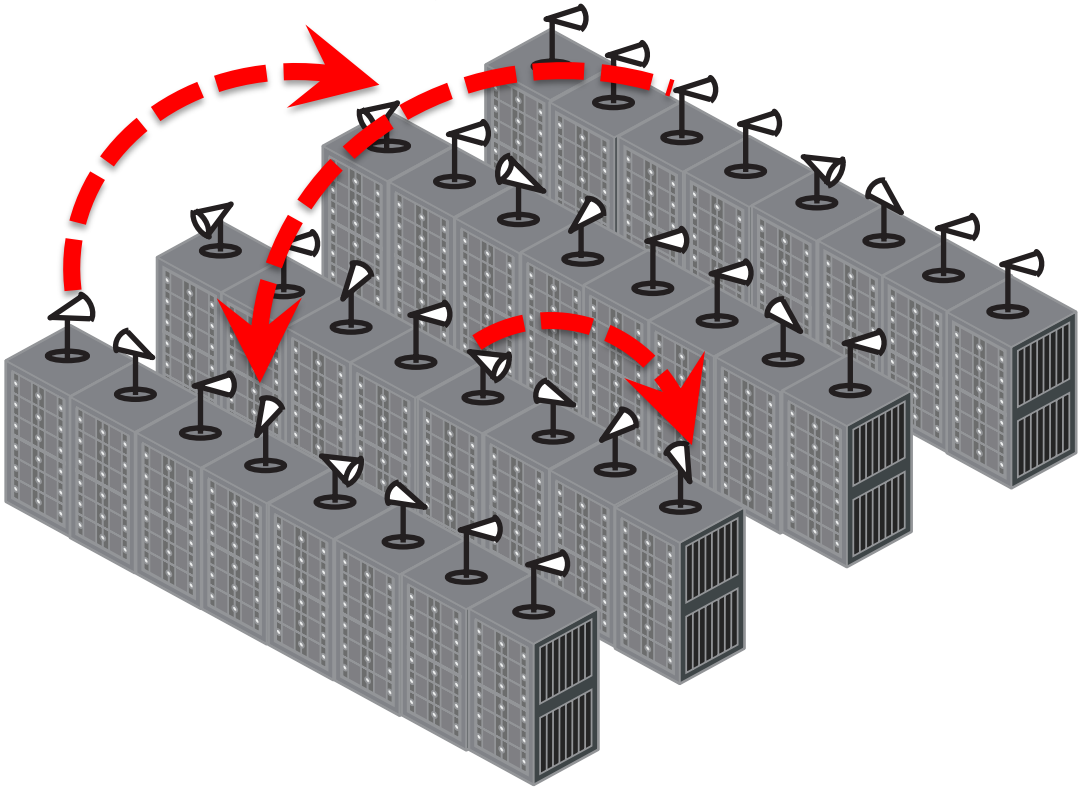
Augmenting via Wireless Links

- 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

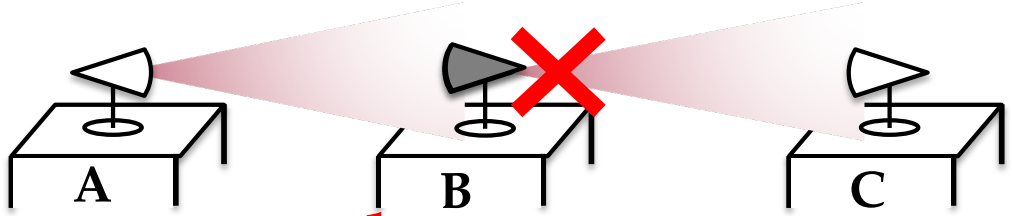
- Connect any rack pair wirelessly to address dynamic traffic hotspots



Hard to do so using 60GHz transmissions

Key Challenges

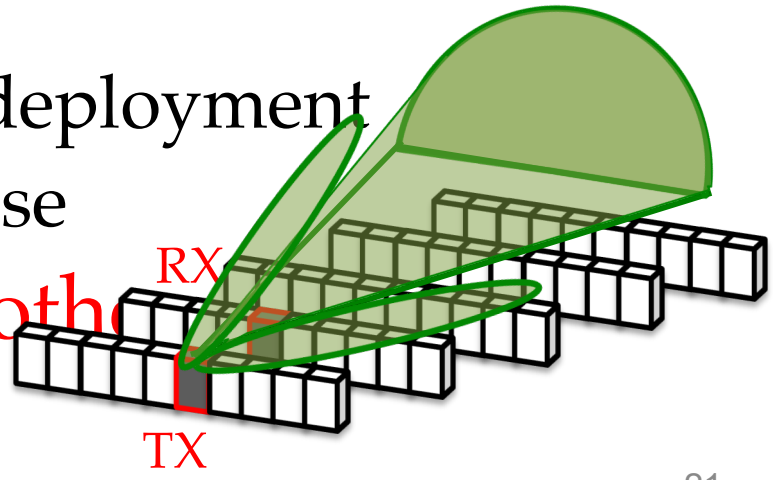
- **Link blockage**: small obstacles block the link



→ Must use multi-hop forwarding

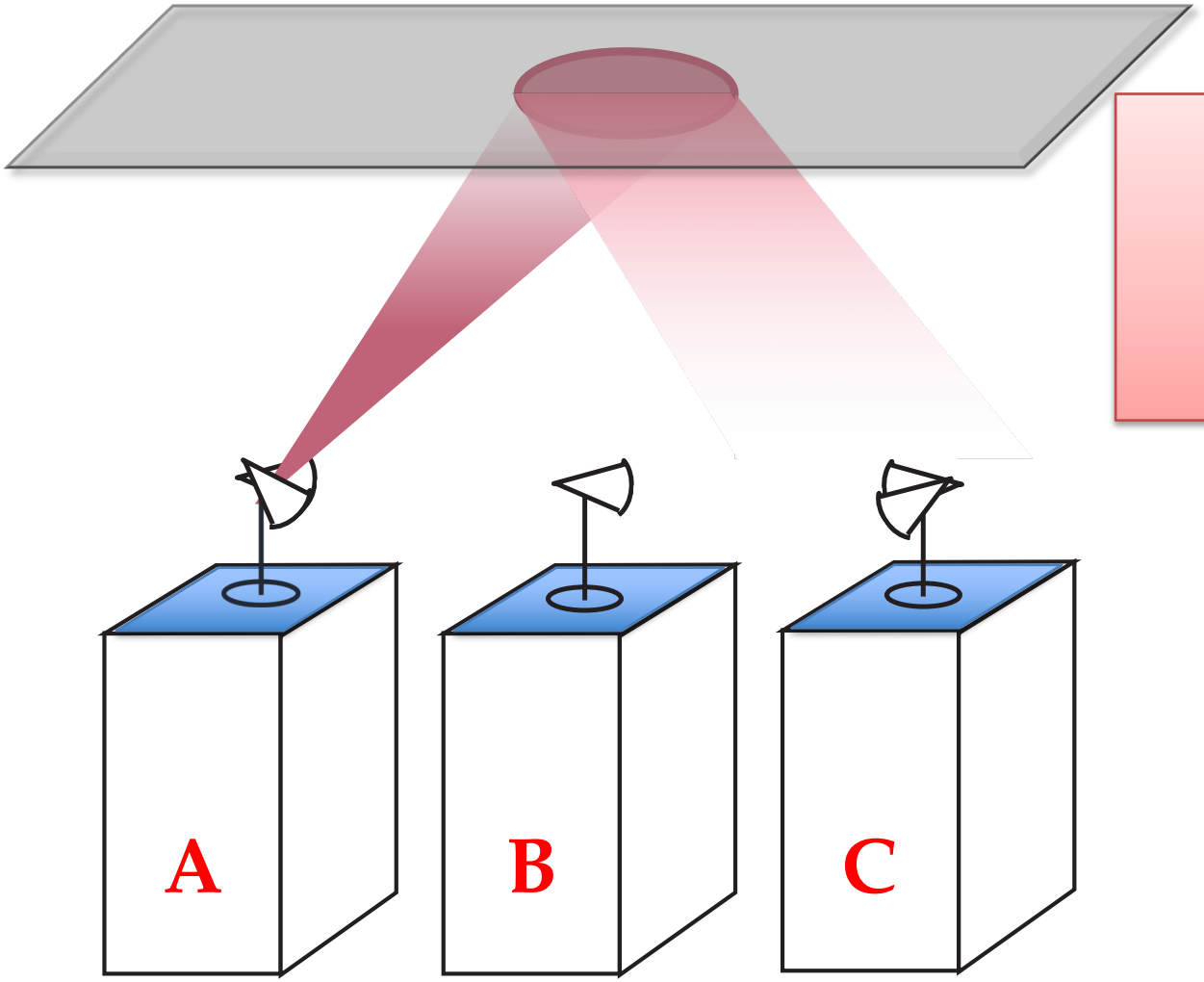
- **Radio interference**: beam interferes with racks in its direction
  - Exacerbated by dense rack deployment
  - Signal leakage makes it worse

→ Links interfere with each oth



Wireless 3D Beamforming

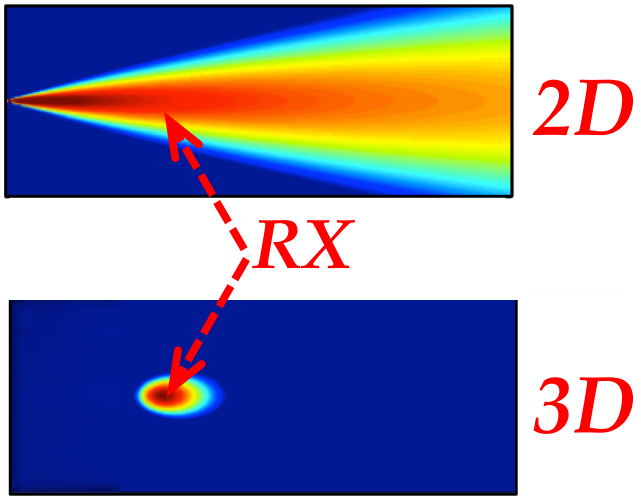
Connect racks by reflecting signal off the ceiling!



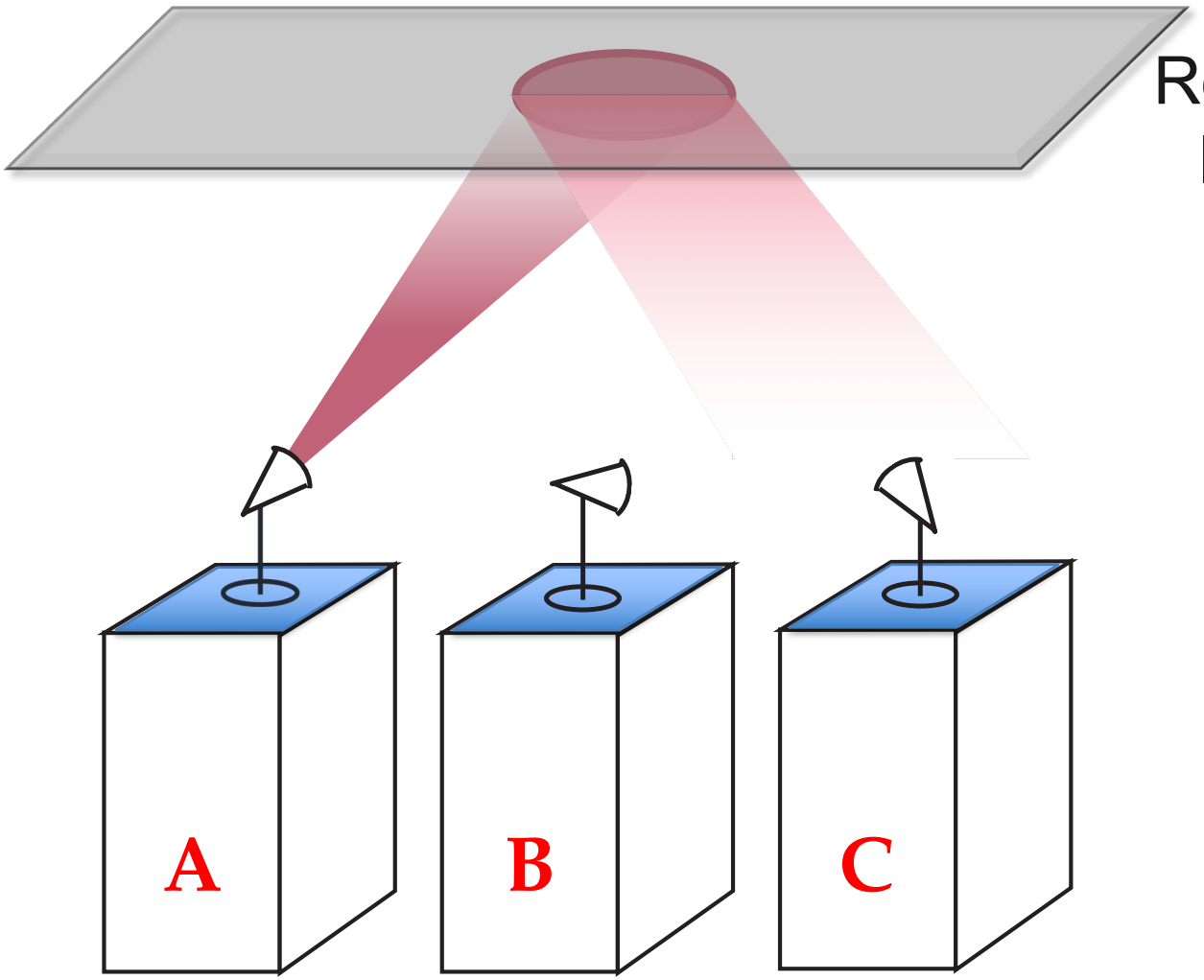
Key Benefits

No more link blockage

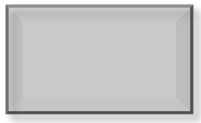
Much smaller interference



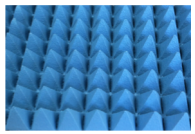
Easy Setup



Reuse existing hardware, low maintenance cost!

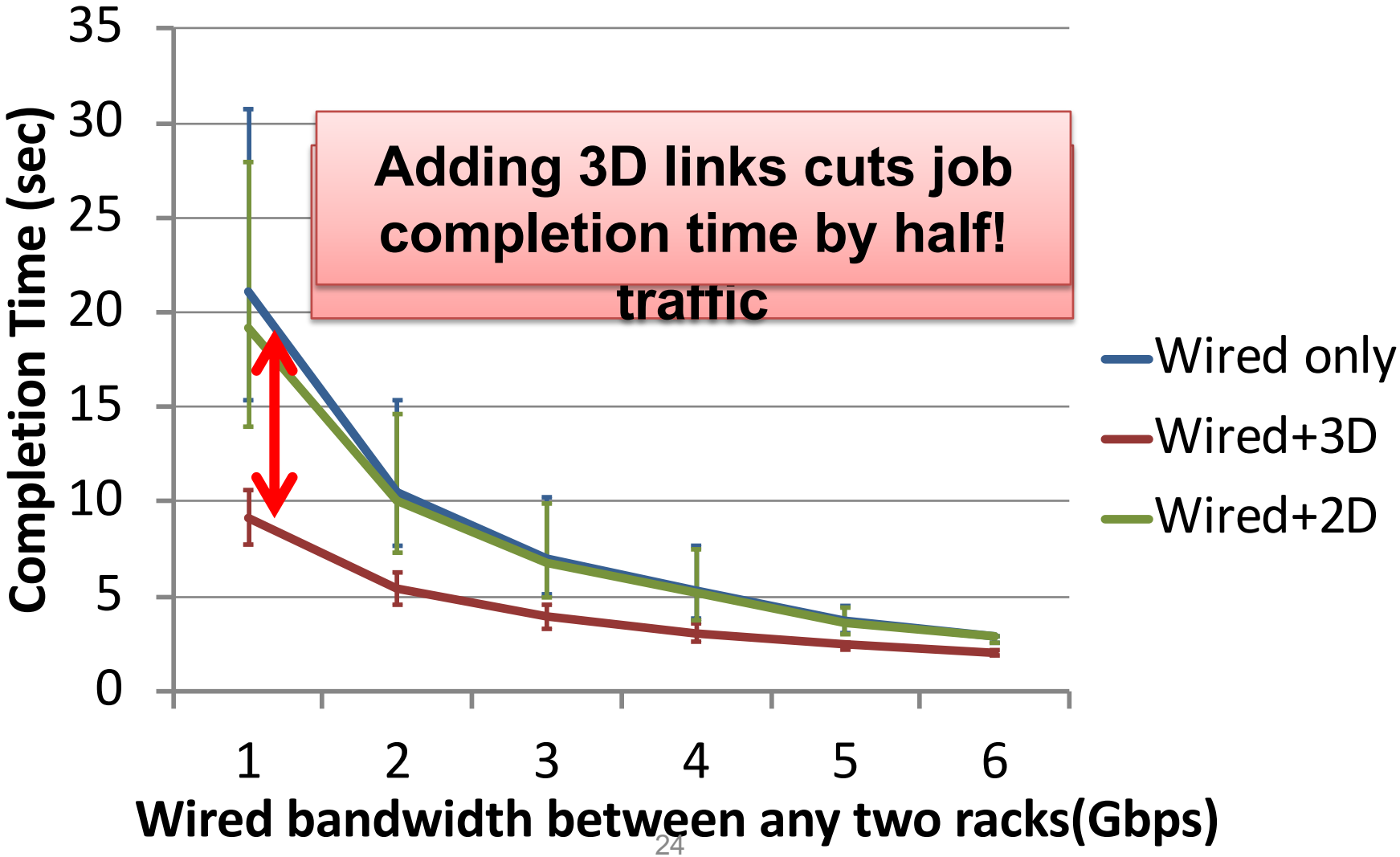


Reflector



Absorber

Results of Addressing Traffic Hotpots

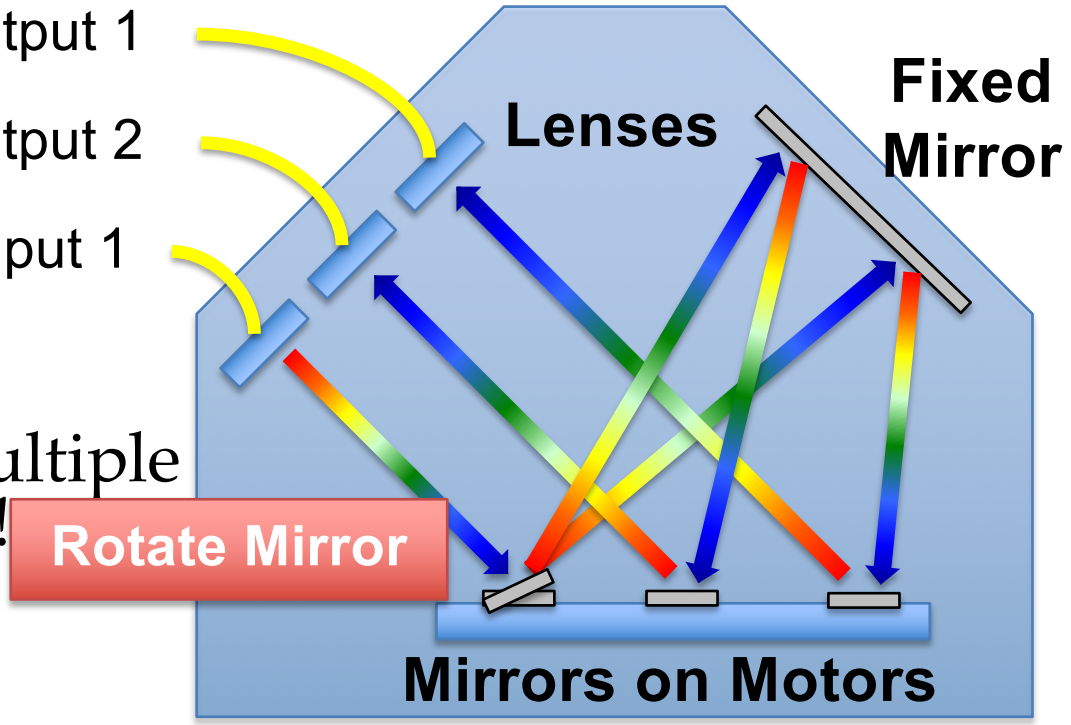


3D Beamforming Summary

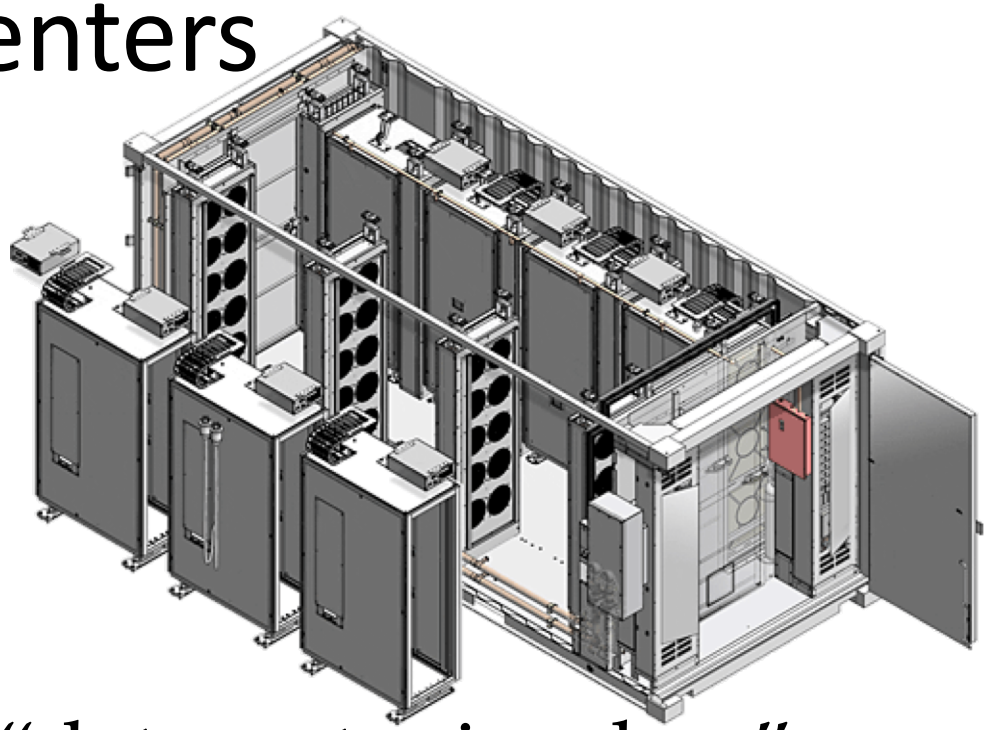
- New wireless primitive added into data center toolbox
  - Overcomes key challenges of using wireless in data centers
  - Opens up more opportunities of using wireless in data centers
- Practical issues
  - Traffic scheduling
  - Reflector placement, impact on data center cooling

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)

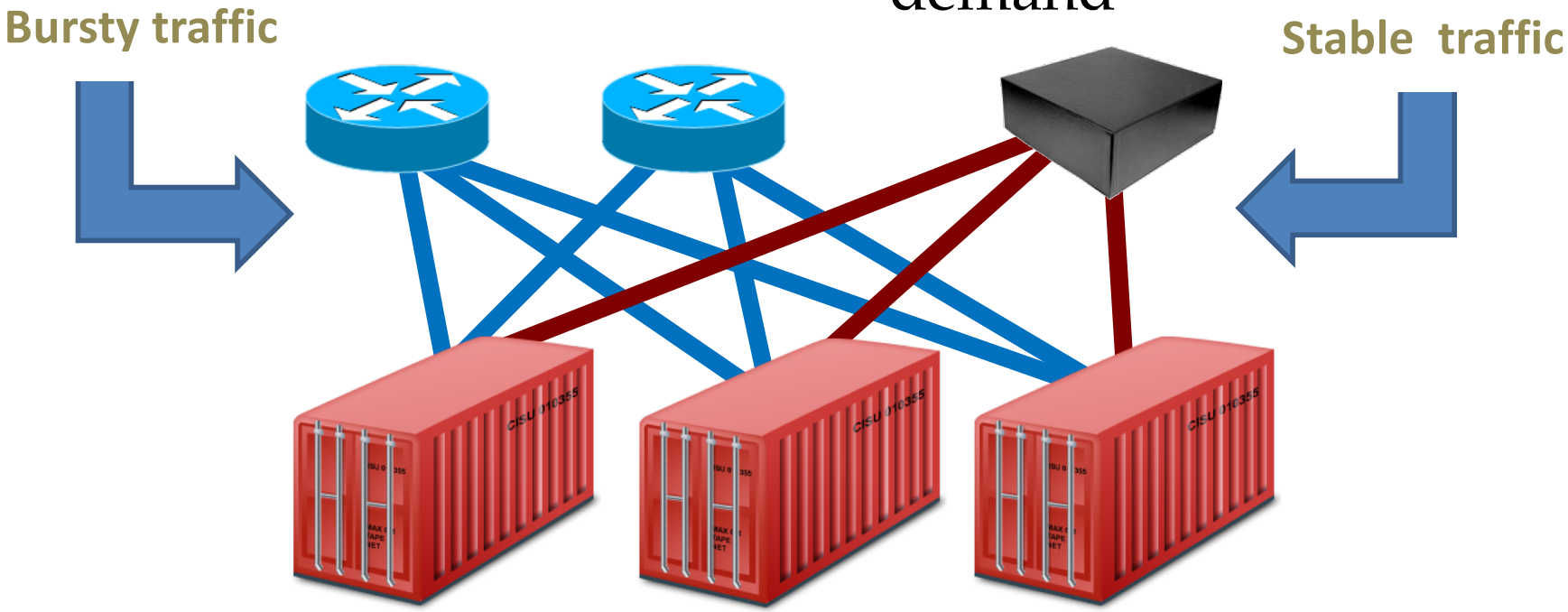
Helios: Datacenters at Light Speed

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
  - Fat tree
  - Wireless in data centers
  - Optical in data centers
- Transport protocols

Transport on the Internet

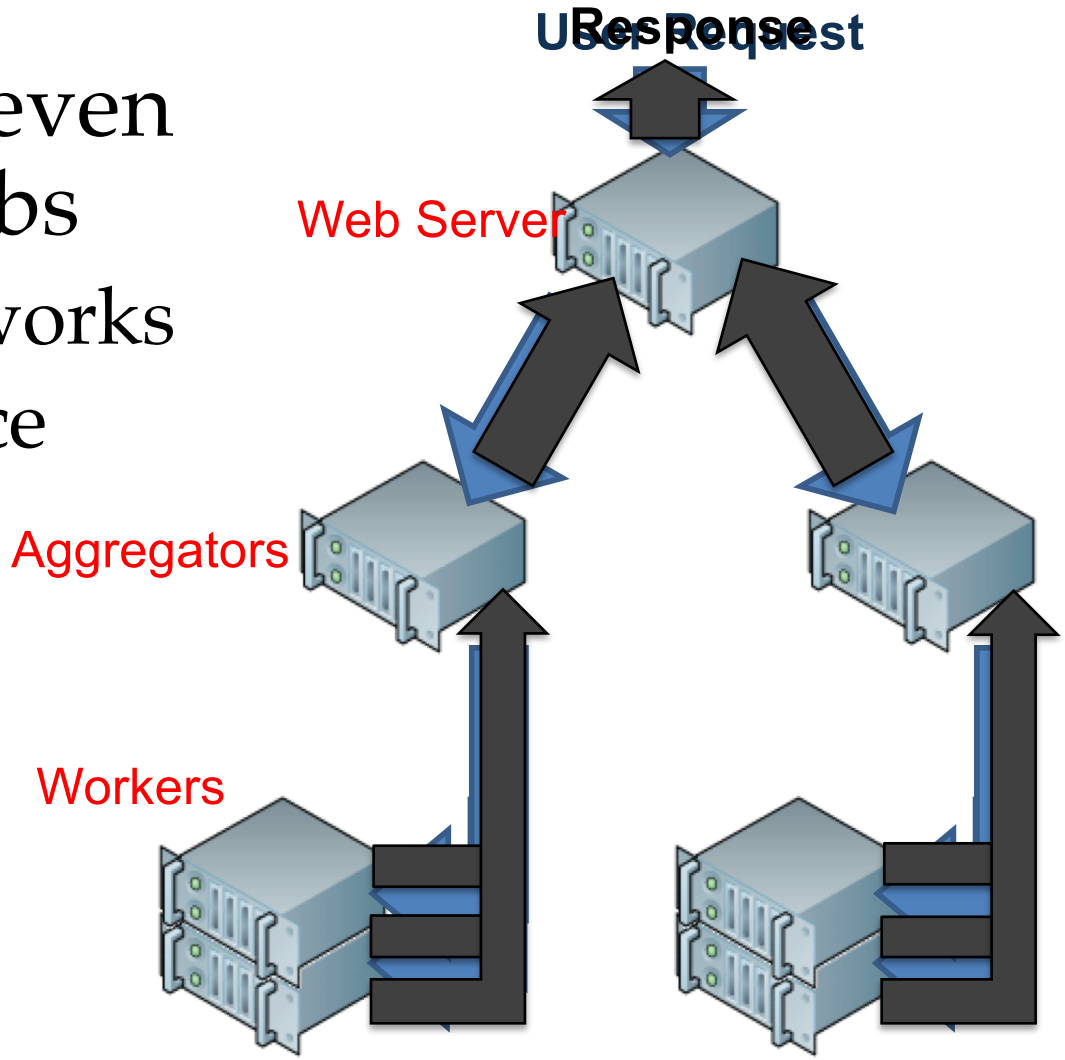
- TCP optimized for the WAN
  - Fairness
    - Slow-start, AIMD convergence
  - Defense against network failures
    - 3-way handshake, reordering
  - Zero knowledge congestion control
    - Self-induces congestion, loss equals congestion

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

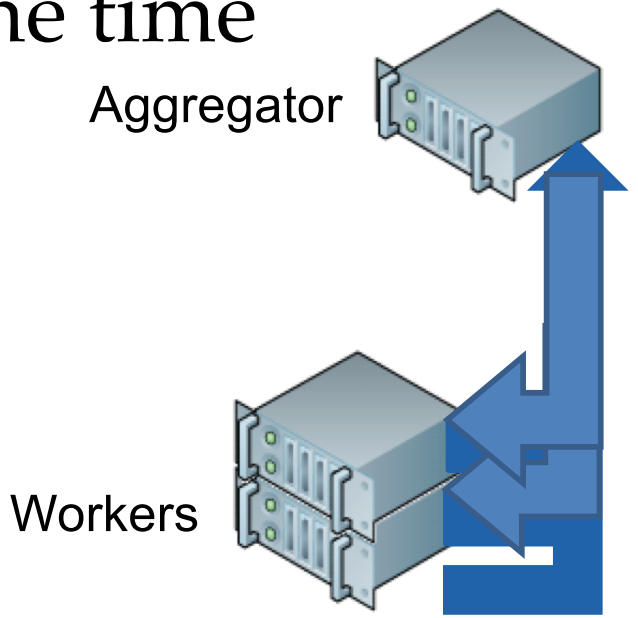
Partition/Aggregate Pattern

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



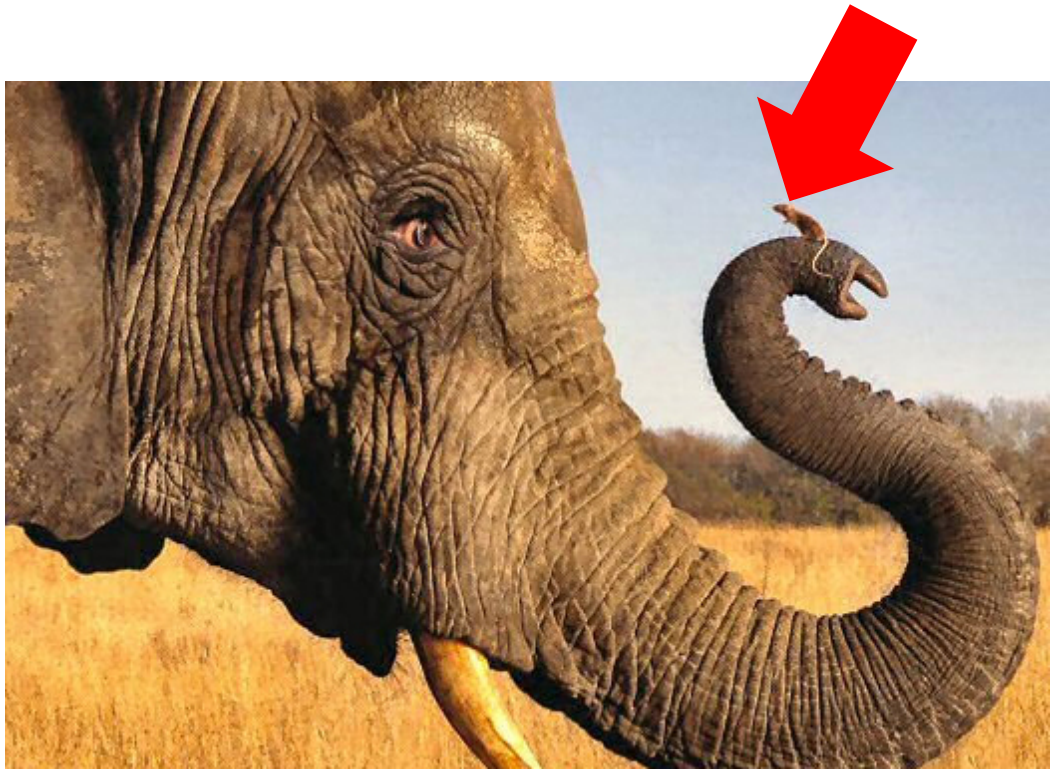
Problem: Incast

- 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 ☹



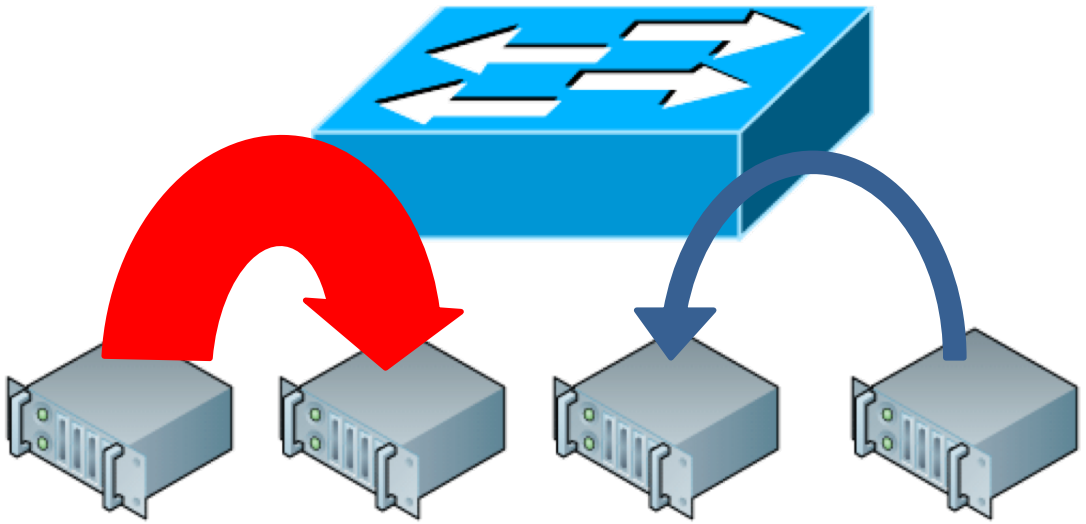
Problem: Queue Buildup

- 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!



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!

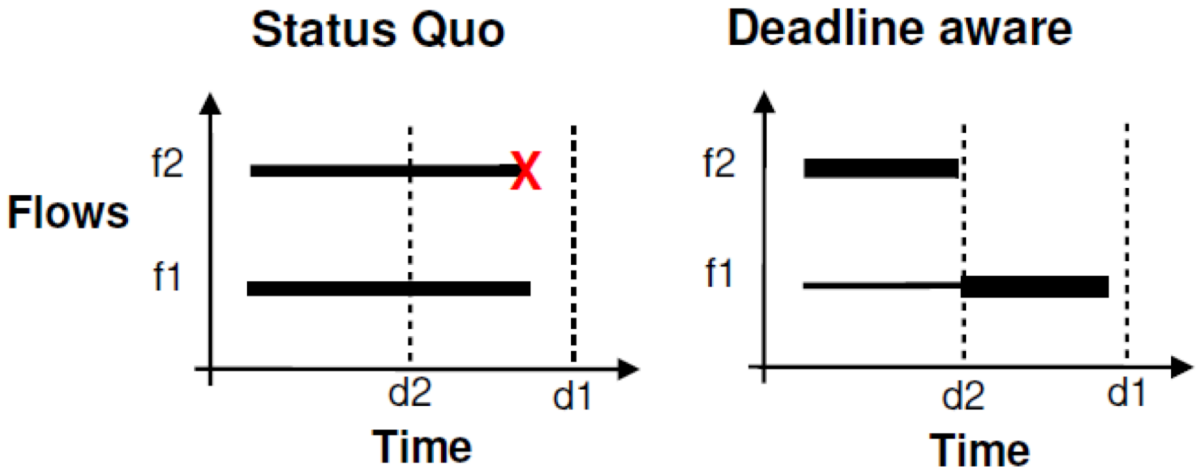


Dirty Slate Approach: DCTCP

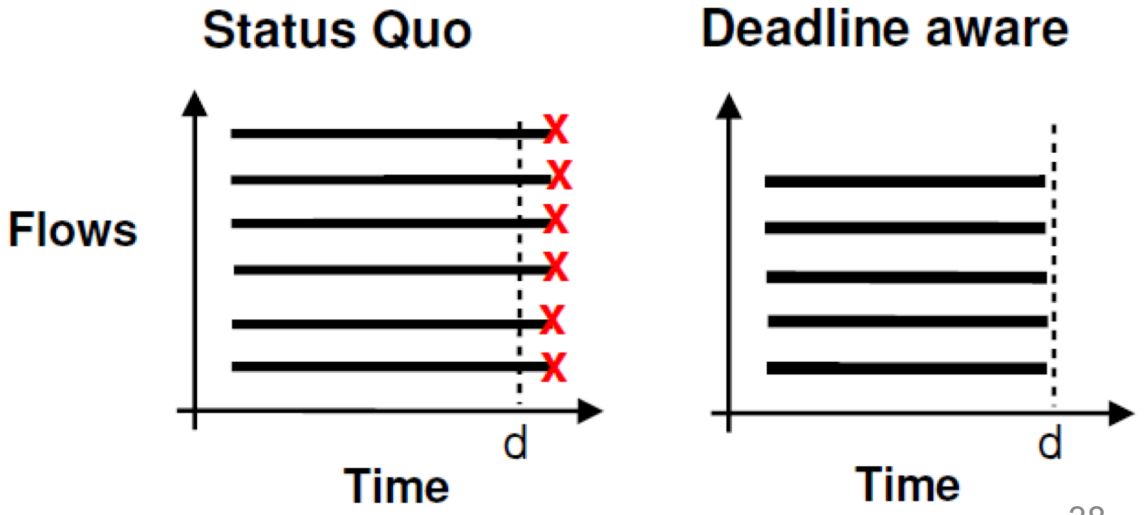
- Goals
  - Alter TCP to achieve low latency
  - Work with shallow buffered switches
  - Do not modify apps, switches, or routers
- Idea
  - Scale window in proportion to congestion
  - Use existing ECN functionality
  - Turn single-bit congestion info to multi-bit

The Need of Deadline-Awareness

- Case #1
  - Fair share causes both to fail
  - Unfair share enable 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

Conclusion

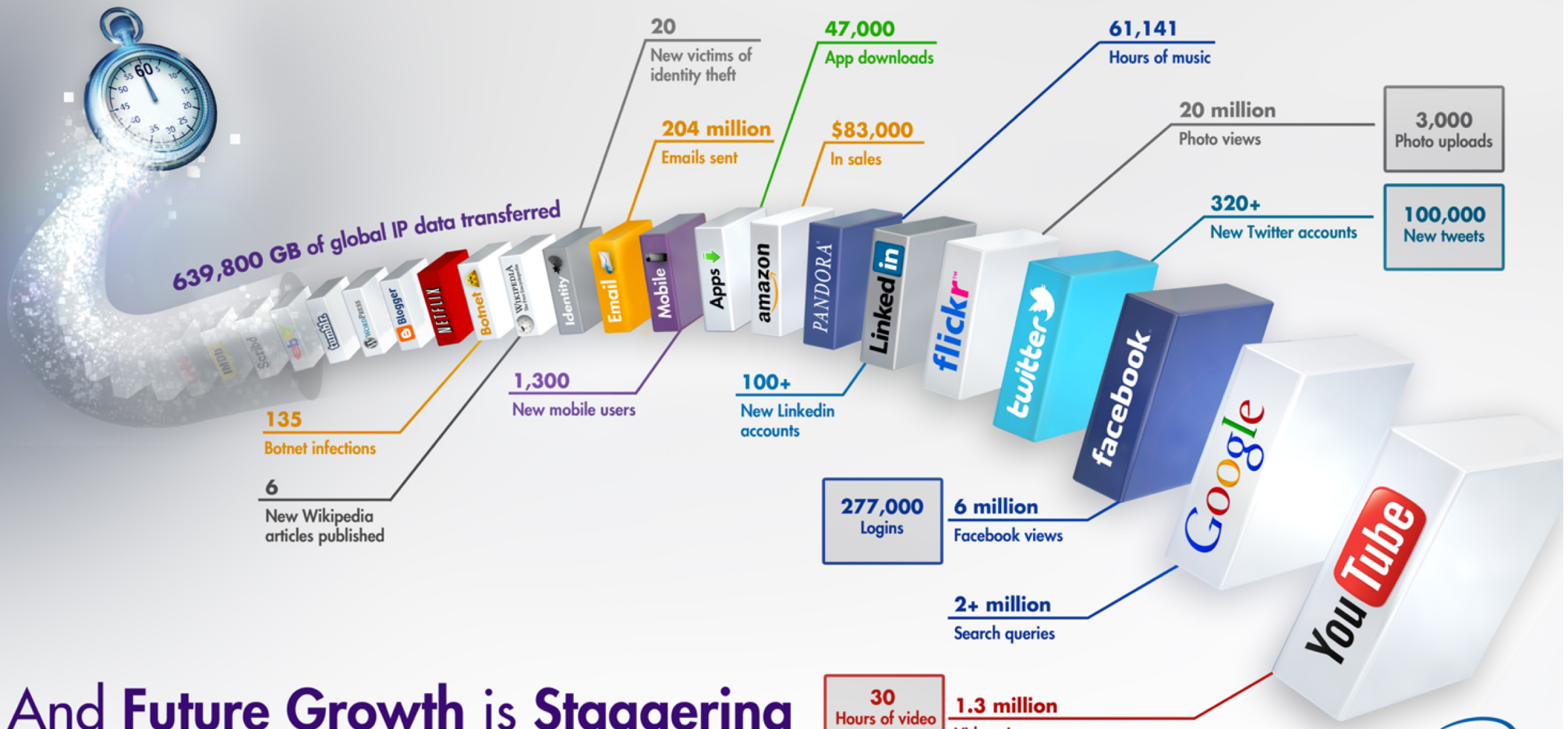
- Data center is a super hot topic right now
  - Topology, routing
  - Network stack
  - Heat, power
  - Management
  - Applications: Hadoop, Dynamo, Cassandra, NoSQL
- Tough for academic research
  - No real data centers to test around

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

DATA-DRIVEN NETWORK DESIGN

What Happens in an **Internet Minute?**



And **Future Growth** is **Staggering**

Today, the number of networked devices = the global population

By **2015**, the number of networked devices = **2x** the global population

In **2015**, it would take you **5 years** to view all video crossing IP networks each **second**

IP

intel

THE LANDSCAPE OF BIG DATA

90% of the data in the world today has been created in the last two years alone

Big data is projected to grow into a **$53.4 BILLION** market by **2017**, up from **$10.2 BILLION** in **2013**

All of the world's digital data equals about **900 exabytes**, **70%** of which is created by individuals

China will account for more than **1/5** of the world's data by 2020

1 terabyte = 1000 gigabytes	1 petabyte = 1000 terabytes	1 exabyte = 1000 petabytes	1 zetabyte = 1000 exabytes
1TB	1PB	1EB	1ZB

1 EB = 1 billion gigabytes or **250 billion DVDs**

1 EB = is nearly **2 times** as large as the web archive at the **US Library of Congress**

MORE THAN 570 NEW WEBSITES are created every minute of the day

Average online cart sizes increase **10%-15%** as a result of personalized advertising using customer information obtained through big data

MOBILE

Global mobile data traffic grew **81% IN 2013**

OVER 500 MILLION mobile devices and connections were added in **2013**

Mobile traffic in 2013 was nearly **18 TIMES** the size of the entire global Internet in **2000**

IMPACT OF BIG DATA

Poor data across businesses and the government costs the U.S. economy **$3.1TRILLION/YEAR**

Accessing 10% more data equates to an additional income of **$65.7 MILLION** for the average **Fortune 1000** company

VIDEO

Percentage of all online traffic that was mobile video

- 53% IN 2013
- 69% IN 2018

100 HOURS of video is uploaded to **YouTube** every minute

WHICH IS WHY

34% of companies implemented big data initiatives in **2013**. Up from **27%** in **2012**

68% of companies are running **two or more** big data projects as part of their big data initiatives

15% of companies with Big Data initiatives spent at least **$100 million** each on those initiatives in **2012**, **7%** invested at least **$500 million** and **24%** invested less than **$2.5 million** a piece.

The White House administration is investing **$200 MILLION** into big data research projects.

BY 2015

4.4 MILLION IT JOBS globally will be created to support big data, generating **1.9 MILLION** IT jobs in the United States

BY 2018

The **United States** alone could face a shortage of **140,000 TO 190,000 PEOPLE** with deep analytical skills as well as **1.5 MILLION** managers and analysts with the know-how to use the analysis of big data to make effective decisions.

Data Driven Design

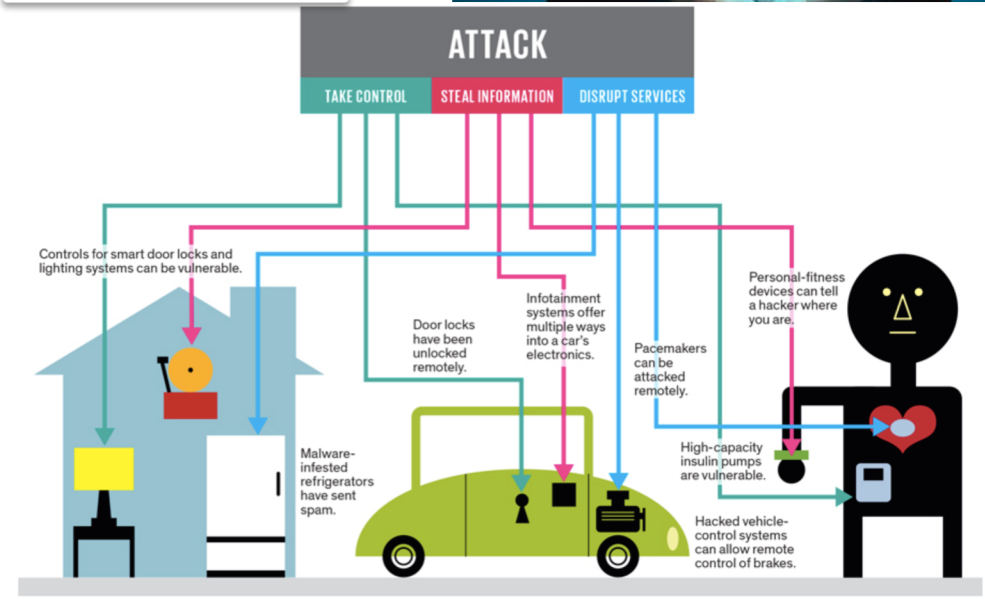


Illustration: J. D. King

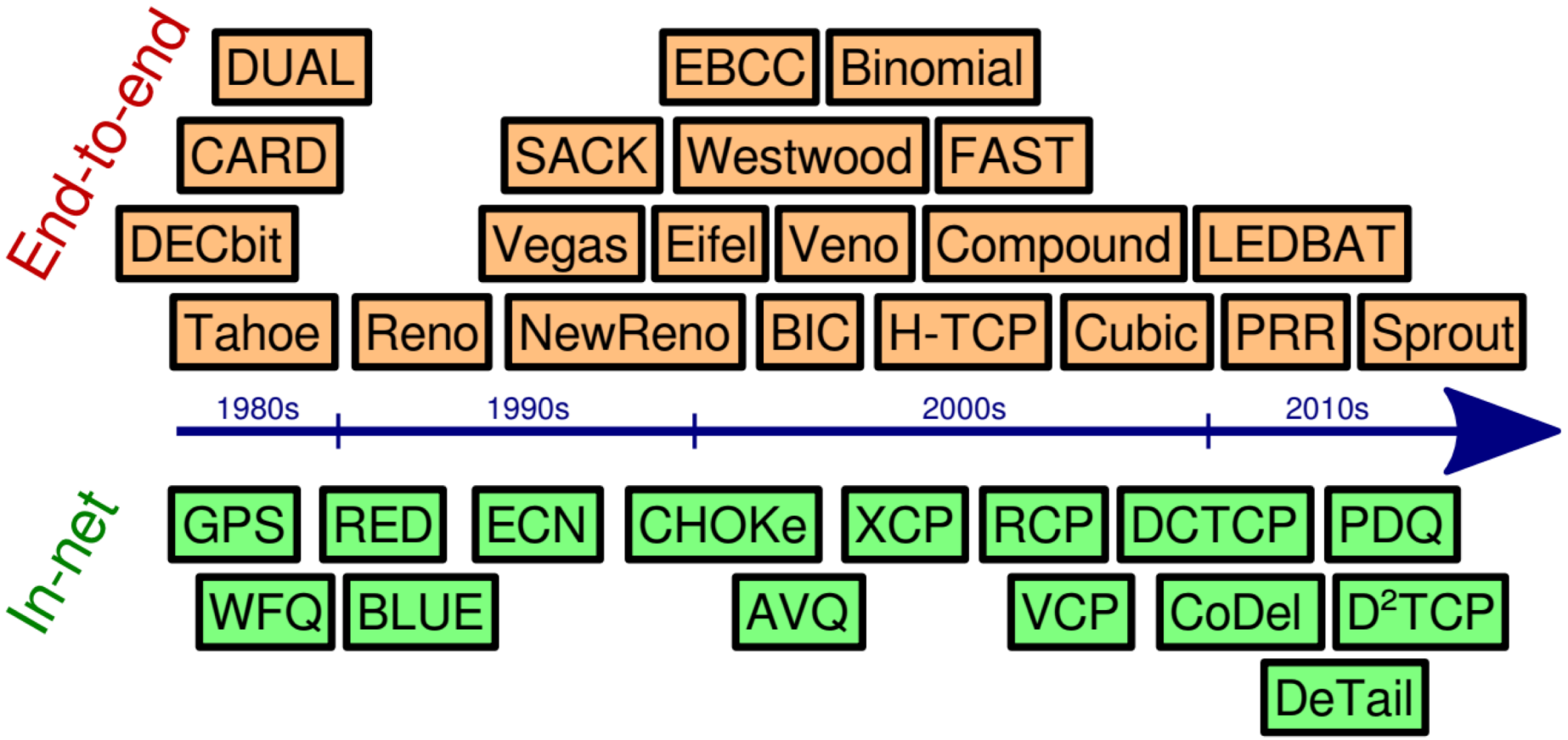
Data-Driven Network Design

- Type 1: Understand “real” network behaviors from ”in the wild” measurements
  - Many existing network protocol designs were driven by insights (which might be obsolete)
- Type 2: Use machine learning, especially reinforcement learning to identify rules for network functionalities

Data-Driven Network Design

- Type 1: Understand “real” network behaviors from ”in the wild” measurements
  - Many existing network protocol designs were driven by insights (which might be obsolete)
- Type 2: Use machine learning, especially reinforcement learning to identify rules for network functionalities

EXAMPLE 1:
MACHINE GENERATED TCP
SIGCOMM 2013

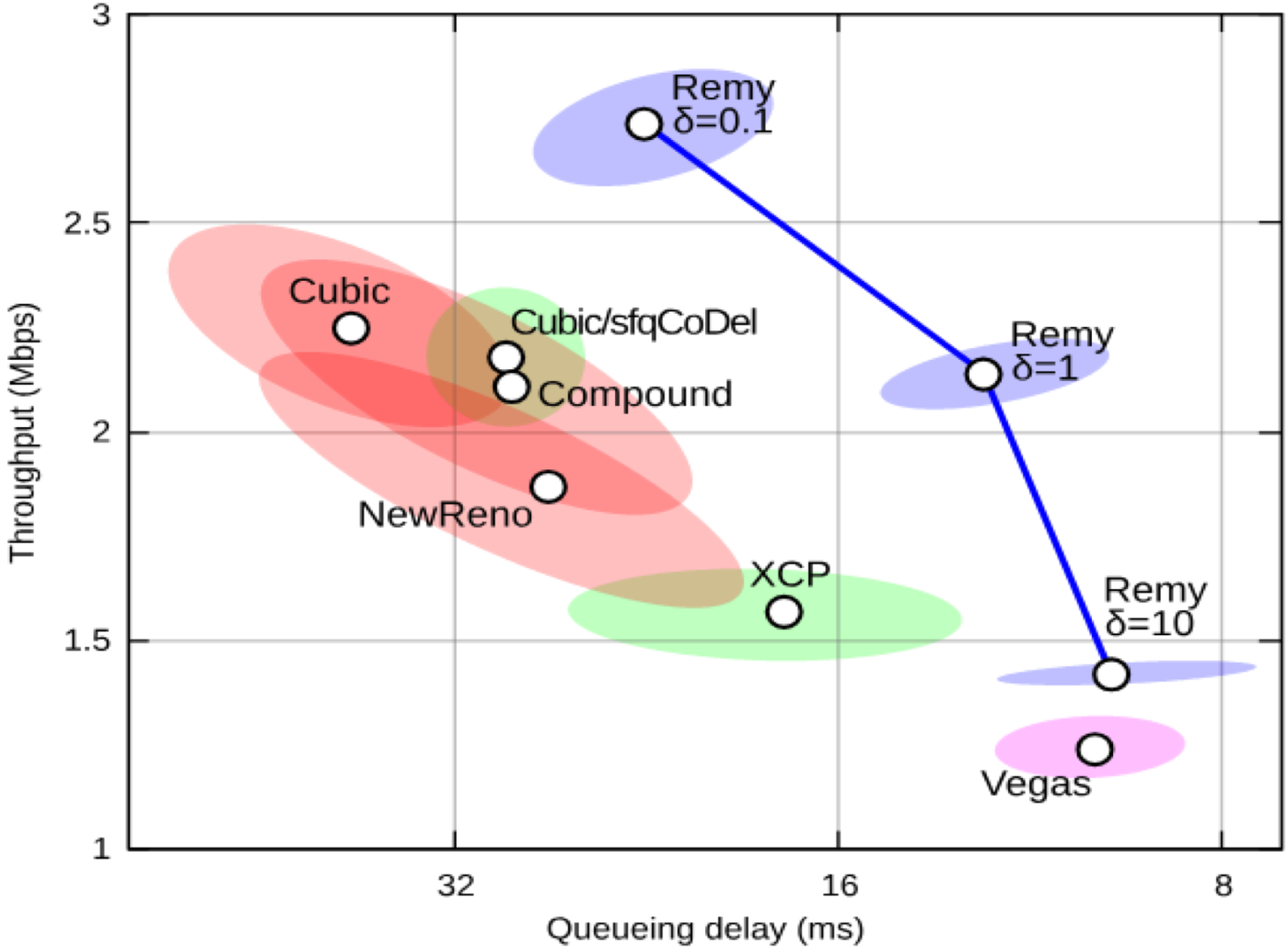

End-to-end
DUAL
EBCC
Binomial
CARD
SACK
Westwood
FAST
DECbit
Vegas
Eifel
Veno
Compound
LEDBAT
Tahoe
Reno
NewReno
BIC
H-TCP
Cubic
PRR
Sprout
1980s
1990s
2000s
2010s
In-net
GPS
RED
ECN
CHOKe
XCP
RCP
DCTCP
PDQ
WFQ
BLUE
AVQ
VCP
CoDel
D²TCP
DeTail

TCP Remy

- So far, human designers have created TCP's congestion-control algorithms **by hand**
  - Making implicit assumptions
- The **emergent behavior** of a flock of machines running TCP is complicated and difficult to characterize
- Remy: machine generated TCP algorithm based on observations of the current network state

http://web.mit.edu/remy/

Remy vs. existing TCP designs



Example throughput-delay plot of each scheme running over a trace of the Verizon LTE downlink, four flows contending

Free the Network to Evolve

- Transport layer should adapt to whatever:
  - Network does
  - Application wants

Remy: a program that generates congestion-control schemes offline

Input:

- Prior assumptions (what network may do)
- Goal (what app wants)

Output: CC algorithm for a TCP sender (RemyCC)

Time: a few hours

Cost: \$5–\$10 on Amazon EC^2

Back to Congestion Control

- Rule: At this moment, do I,
  - Send a packet
  - Not send a packet

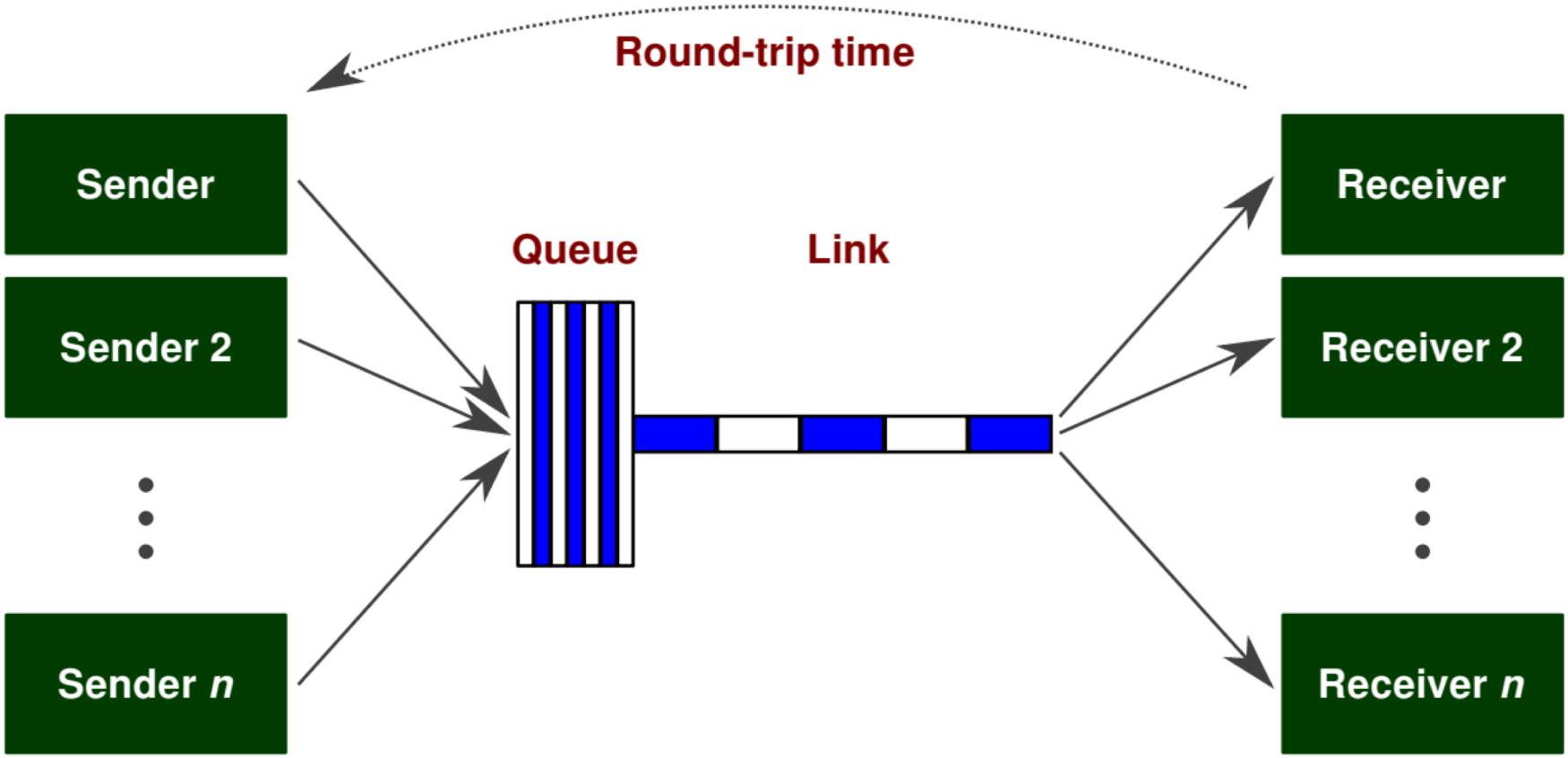
- Objective:
  - Maximize Proportional Fair
    - Sum (log (thpt_i))
    - Sum (log (thpt_i / (delay_i)$^{\delta}$)
  - Minimize
    - Avg flow completion time
    - Page load time
    - Tail completion time

Assumptions

- Model of network uncertainty
  - Link speed distribution
  - Delay distribution
- Traffic model
  - Web browsing, MapReduce, videoconferencing

- Assuming everyone is running the same alg.

Dumbbell Network



Remy Structure

- Remy searches for the best congestion-control algorithm
- Optimizes expected objective over prior assumptions
- Makes tractable by limiting available state
- Remy tracks three congestion signals

r_ewma: moving average of interval between acks

s_ewma: . . . between sender timestamps echoed in acks

rtt_ratio: ratio of last RTT to smallest RTT so far

Remy maps each state to an action

$$\textsc{Rule}(r\_ewma, s\_ewma, rtt\_ratio) \rightarrow \langle m, b, \tau \rangle$$

m	Multiple to congestion window
b	Increment to congestion window
τ	Minimum interval between two outgoing packets

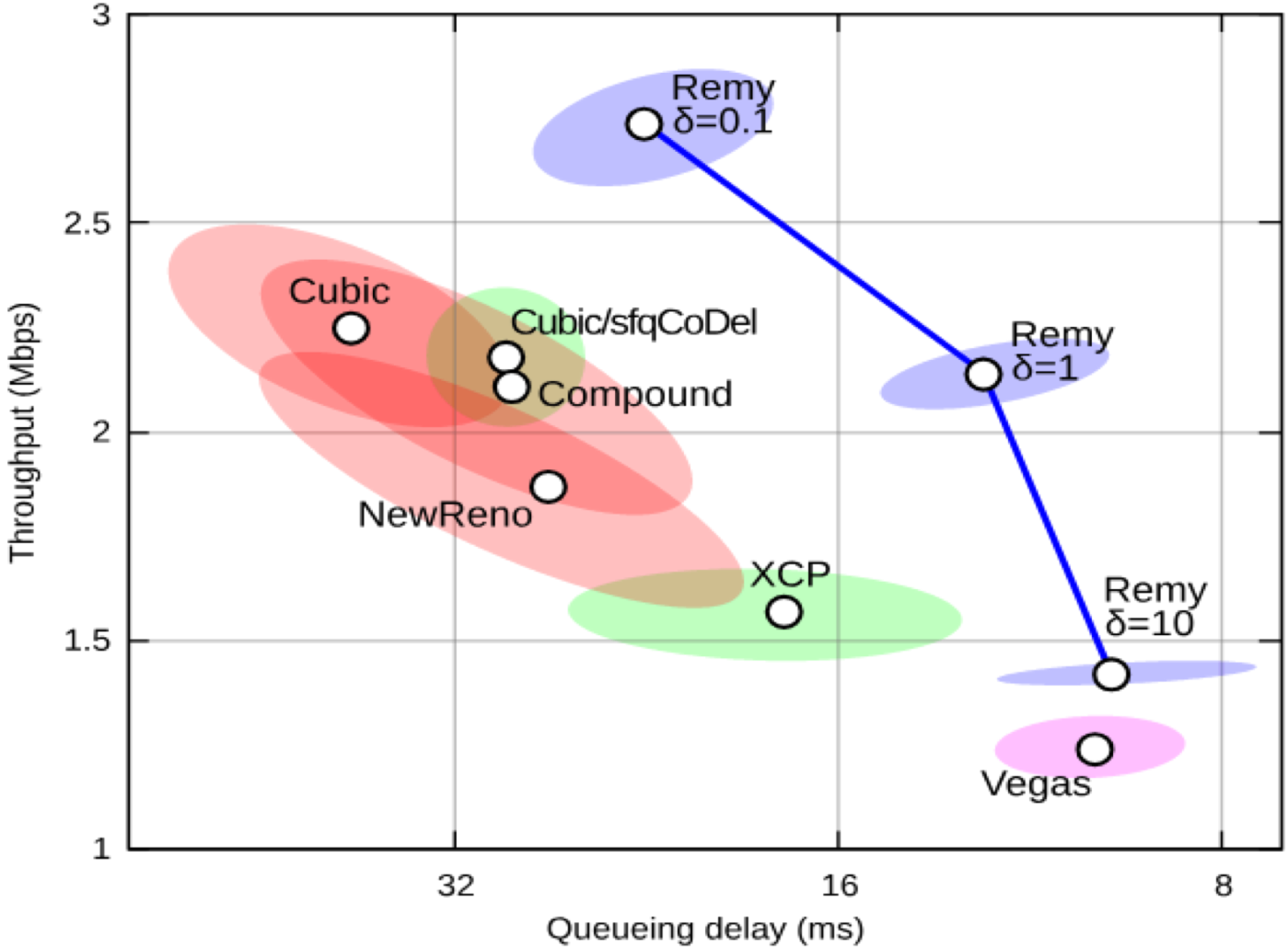
On ack:

- $\langle m, b, \tau \rangle \leftarrow \textsc{Rule}(r\_ewma, s\_ewma, rtt\_ratio)$
- $\texttt{cwnd} \leftarrow m \cdot \texttt{cwnd} + b$

Send packet if:

- $\texttt{cwnd} > \texttt{FlightSize}$, and
- last packet sent $> \tau$ ago

Remy vs. existing TCP designs



Example throughput-delay plot of each scheme running over a trace of the Verizon LTE downlink, four flows contending

Conclusions

- Traditionally: simple rules, complex behavior
- With Remy: complex rules, consistent behavior
- Computer-designed > human-designed
- Follow-up work:
  - How easy is it to learn a network protocol to achieve a desired goal, despite a mismatched set of assumptions?
  - Can tolerate mismatched link-rate assumption, but need precision about the number of senders

Project 3: Wed Crawler

- (40) Basic crawler:
  - Start from index.html, get links to objects and other pages, crawl everything (including images) into a local directory
- (40) Parallel crawler:
  - Parallel connections via multi-threads
  - Threads use the same cookie
- (20) Advanced Parallel crawler:
  - Threads use different cookies to avoid per-user rate limit

EXAMPLE 2:

UNCOVERING SOCIAL NETWORK SYBILS IN THE WILD (IMC 2011)

Sybils on OSNs

- Large OSNs are attractive targets for…
  - Spam dissemination
  - Theft of personal information

- Sybil, *sɪbəl*, Noun: a fake account that attempts to create many friendships with honest users
  - Friendships are precursor to other malicious activity

- Research has identified malicious Sybils on OSNs
  - Twitter [CCS 2010]
  - Facebook [IMC 2010]

Understanding Sybil Behavior

- Prior work has focused on spam
  - Content, dynamics, campaigns
  - Includes compromised accounts

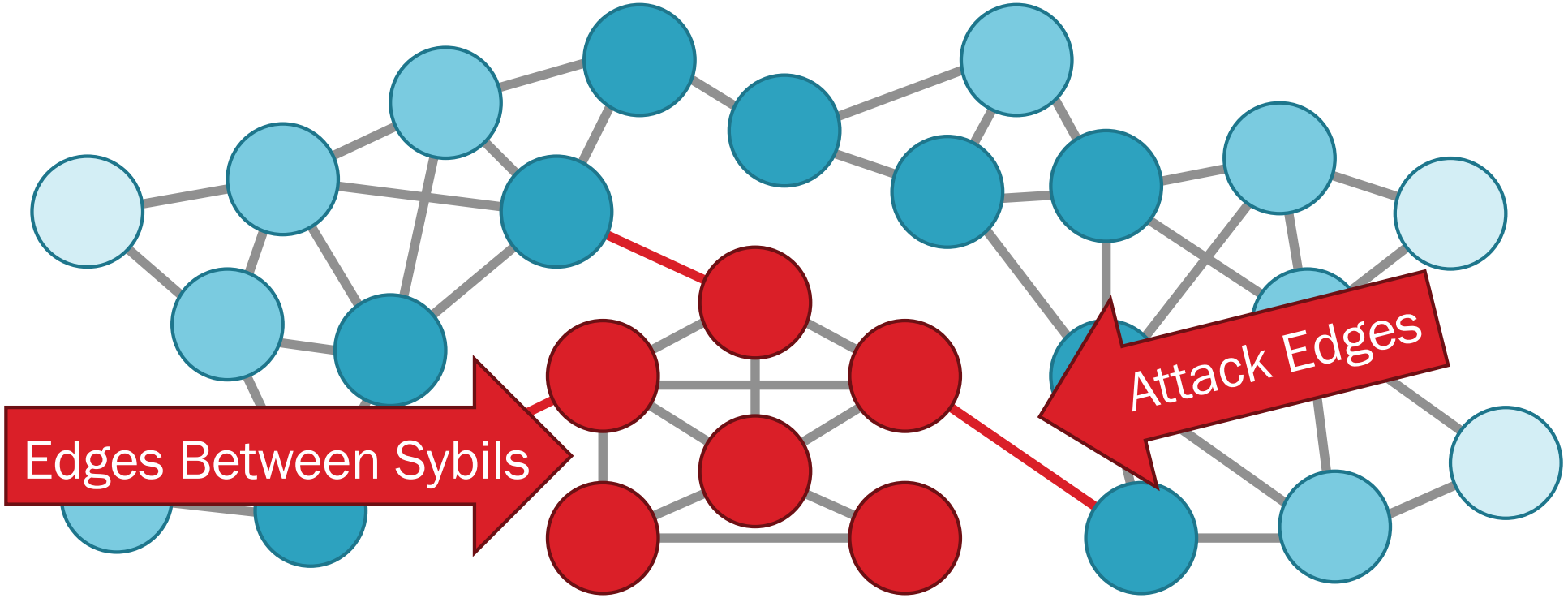
- Open question: What is the behavior of Sybils in the wild?
  - ➢ *Important for evaluating Sybil detectors*

- Partnership with largest OSN in China: Renren
  - Leverage ground-truth data on 560K Sybils
  - Develop measurement-based, real-time Sybil detector
  - Deployed, caught additional 100K Sybils in 6 months

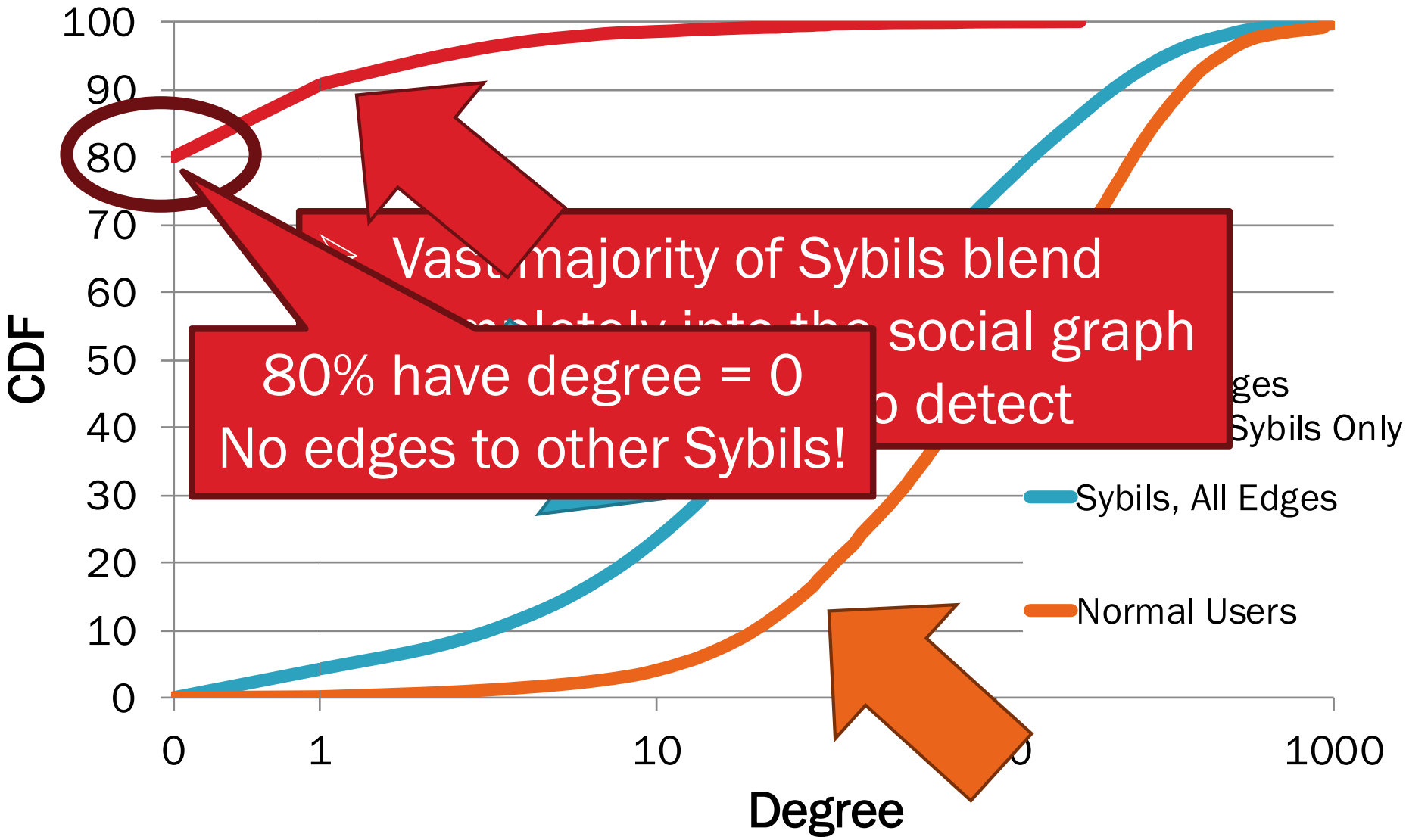
Community-based Sybil Detectors

- Prior work on decentralized OSN Sybil detectors
  - SybilGuard, SybilLimit, SybilInfer, Sumup
  - Key assumption:

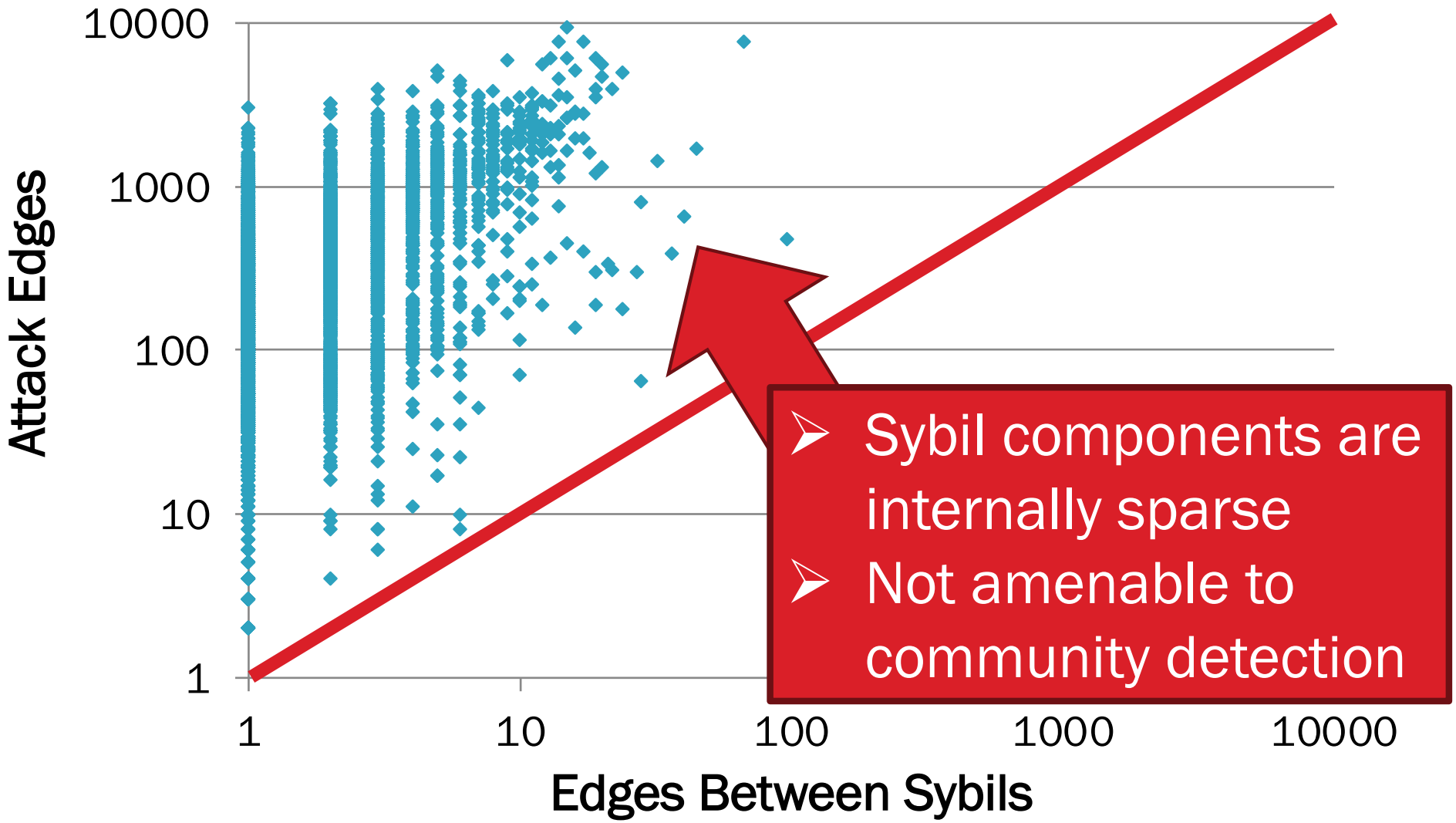
Sybils form tight-knit communities



Do Sybils Form Connected Components?



Can Sybil Components be Detected?



Conclusion

- First look at Sybils in the wild
  - Ground-truth from inside a large OSN
  - Deployed detector is still active
- Sybils are quite sophisticated
  - Cheap labor → very realistic fakes
  - Created and managed by-hand
- Need for new, decentralized Sybil detectors
  - Results may not generalize beyond Renren
  - Evaluation on other large OSNs