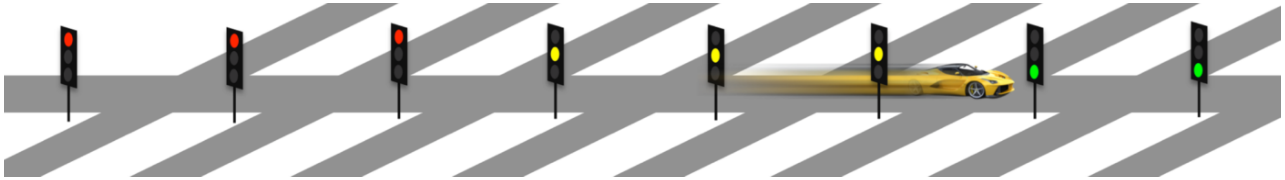
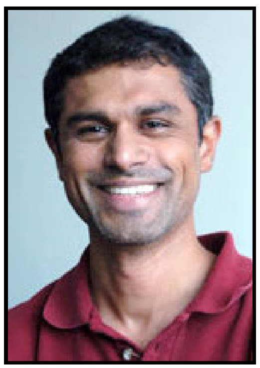


Fastpass

A Centralized "Zero-Queue" Datacenter Network

Jonathan Perry

Amy Ousterhout



Hari Balakrishnan

Devavrat Shah

Hans Fugal

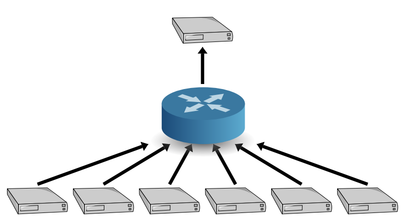
Ideal datacenter network properties

No current design satisfies all these properties simultaneously

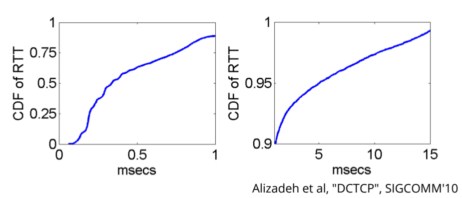
Scaling Memcache at Facebook, Fine-grained TCP retransmissions

Datacenter TDMA, Tail at scale, pFabric, PDQ, DCTCP, D3, Orchestra

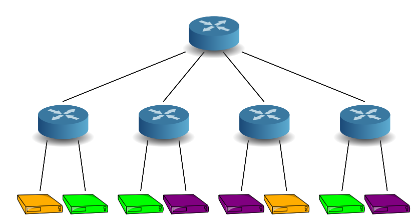
EyeQ, Seawall, Oktopus, Hedera, VL2, Mordia, SWAN, MATE, DARD



Burst Control



Low Tail Latency

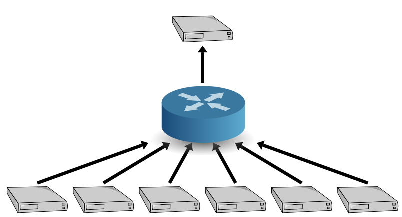


Multiple Objectives

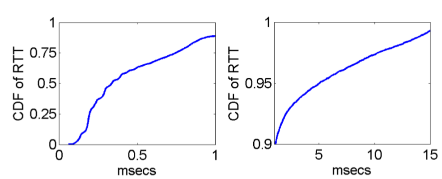
Fastpass goals

Is it possible to design a network that provides

1. <u>Zero</u> network queues
2. High Utilization
3. Multiple app and user objectives

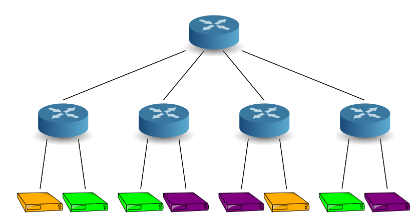


Burst Control



Alizadeh et al, "DCTCP", SIGCOMM'10

Low Tail Latency



Multiple Objectives

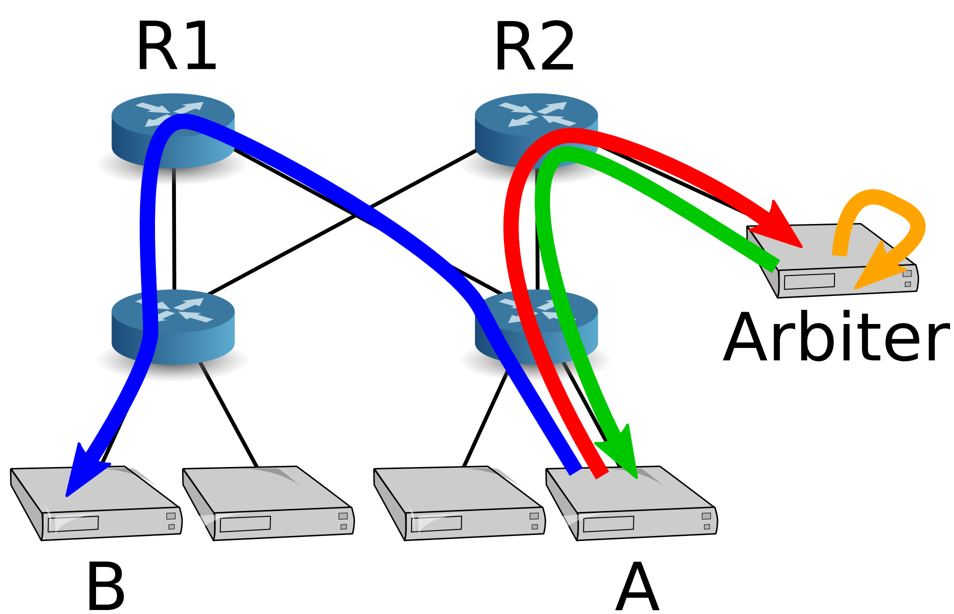
Centralized *arbiter* schedules
and assigns paths to all packets

Concerns with centralization:

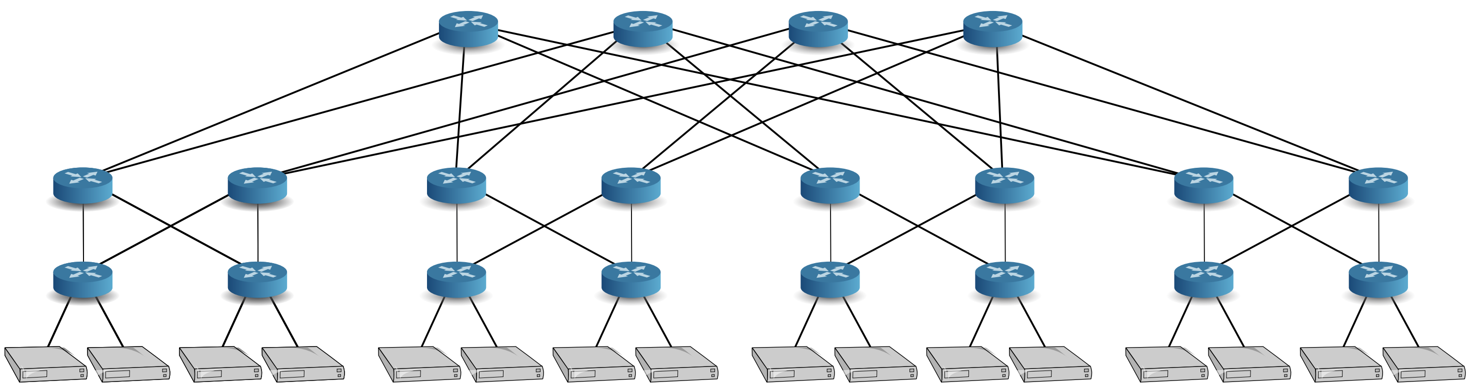
- Latency
- Scaling
- Fault tolerance

Example: Packet from A to B

5μs	A → Arbiter	"A has 1 packet for B"
1-20μs	Arbiter	timeslot allocation & path selection
15μs	Arbiter → A	"@t=107: A → B through R1"
no queuing	A → B	sends data



Scheduling and selecting paths



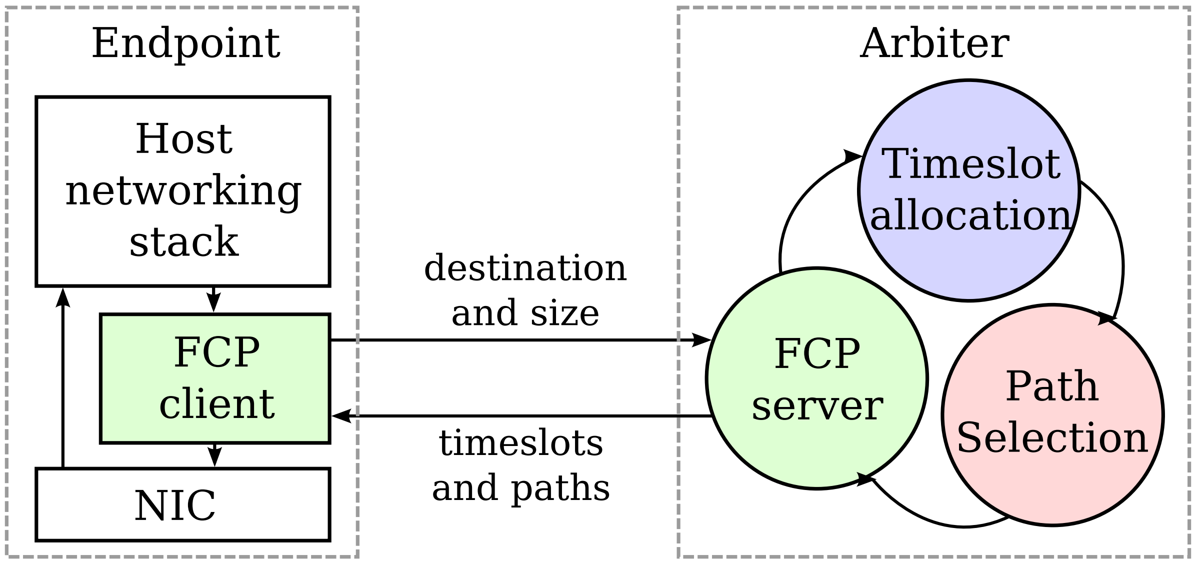
Timeslot = 1.2 μs

Step 1: Timeslot Allocation
Choose a matching

Step 2: Path selection
Map matching onto paths

Arbiter treats network as a big switch

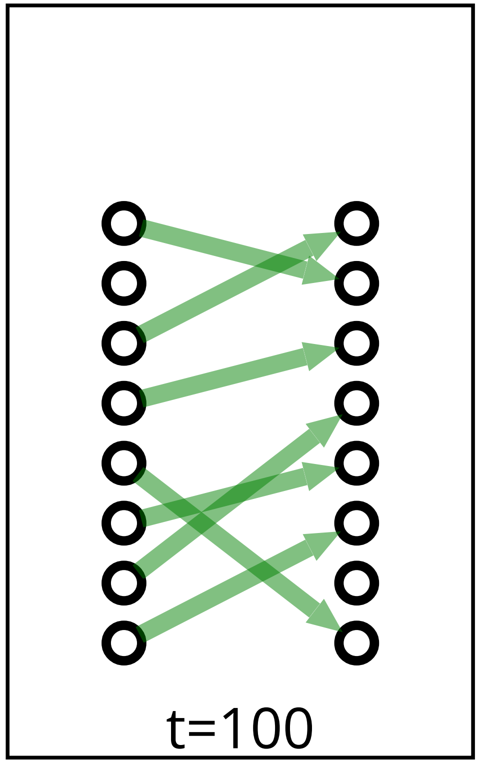
System structure



Challenges:

- Latency
- Scaling
- Fault tolerance

Timeslot allocation = maximal matching



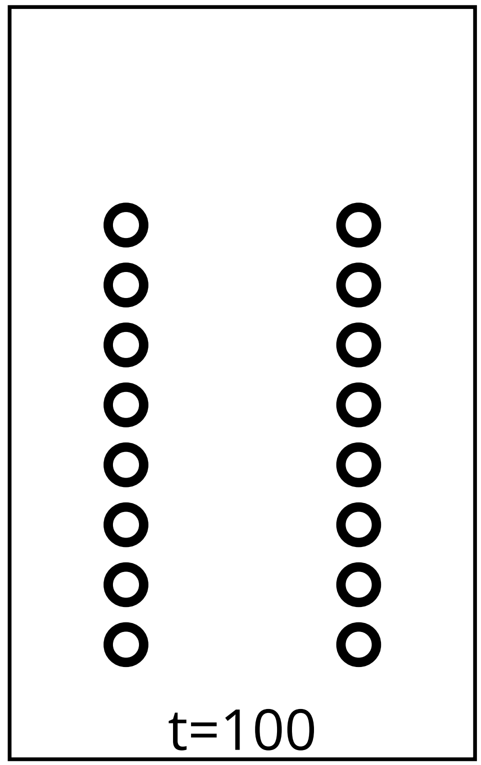
$$\left(\overset{\text{src}}{1} \rightarrow \overset{\text{dst}}{2}\ ,\ \overset{\text{pkts}}{3}\right)$$
$$\left(\overset{\text{src}}{3} \rightarrow \overset{\text{dst}}{1}\ ,\ \overset{\text{pkts}}{3}\right)$$
$$\left(\overset{\text{src}}{7} \rightarrow \overset{\text{dst}}{4}\ ,\ \overset{\text{pkts}}{1}\right)$$
$$\left(\overset{\text{src}}{5} \rightarrow \overset{\text{dst}}{8}\ ,\ \overset{\text{pkts}}{2}\right)$$
$$\left(\overset{\text{src}}{4} \rightarrow \overset{\text{dst}}{3}\ ,\ \overset{\text{pkts}}{4}\right)$$
$$\left(\overset{\text{src}}{1} \rightarrow \overset{\text{dst}}{3}\ ,\ \overset{\text{pkts}}{1}\right)$$
$$\left(\overset{\text{src}}{8} \rightarrow \overset{\text{dst}}{6}\ ,\ \overset{\text{pkts}}{3}\right)$$

~10ns per demand

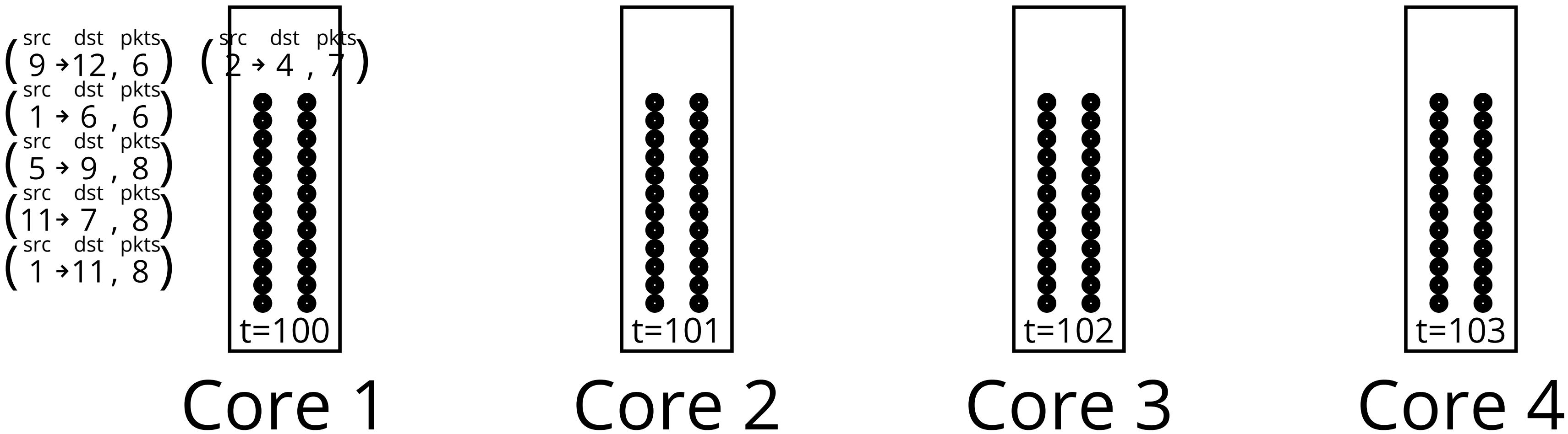
How to support different objectives?

Order matters!

src		dst	pkts
1	→	4	1
6	→	2	2
4	→	3	2
1	→	7	3
8	→	5	4
6	→	7	5
6	→	1	5
2	→	5	6



How to scale timeslot allocation?



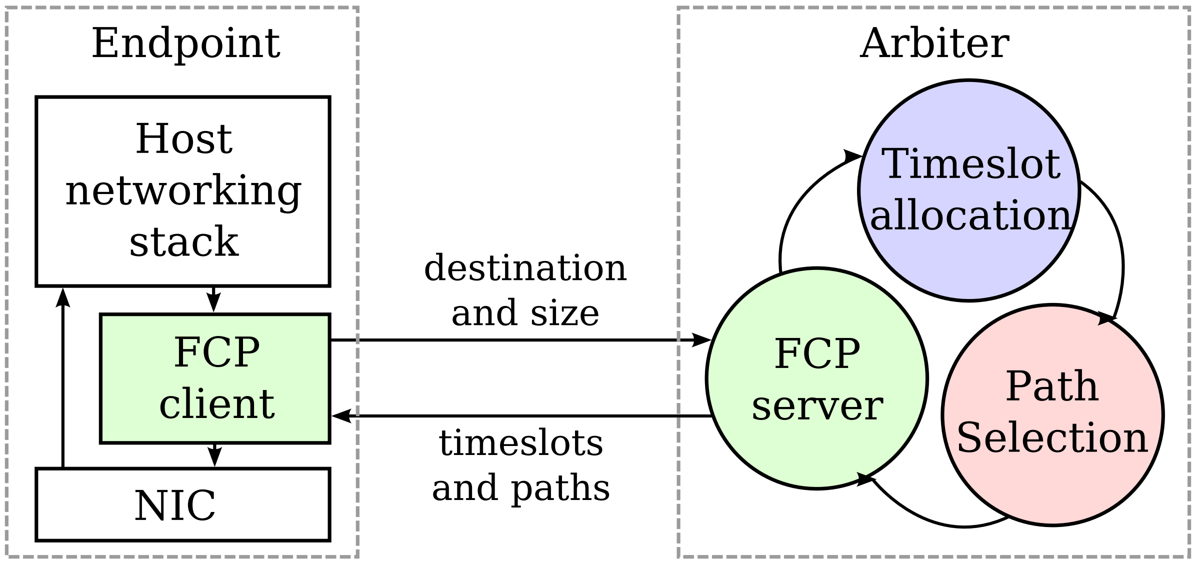
Can pipeline timeslot allocation

2211.8 Gbits/s on 8 cores

Are maximal matchings good matchings?

Maximal Matching 2C network capacity		Optimal scheduler C network capacity
Dai-Prabhakar '00: Finite average latency	$\Leftarrow$	Finite average latency
Our theorem: Average latency	$\leq 2 \times$	Average latency

System structure



Challenges:

- Latency
- Scaling
- Fault tolerance

Fault-tolerance

Arbiter failures

Hot backups , TCP as last resort

Switch failures

Packet loss to arbiter

Experimental results

Timeslot allocation	2.21 Terabits/s with 8 cores
Path selection	>5 Terabits/s with 10 cores

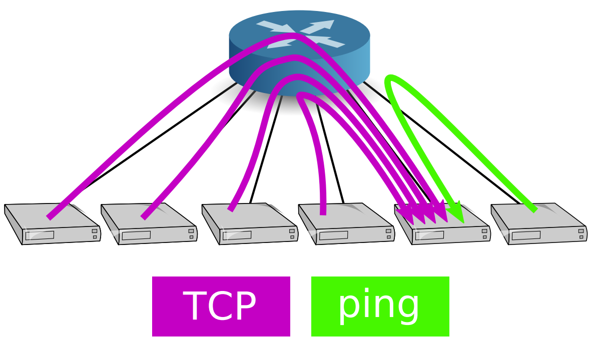
Facebook experiments:

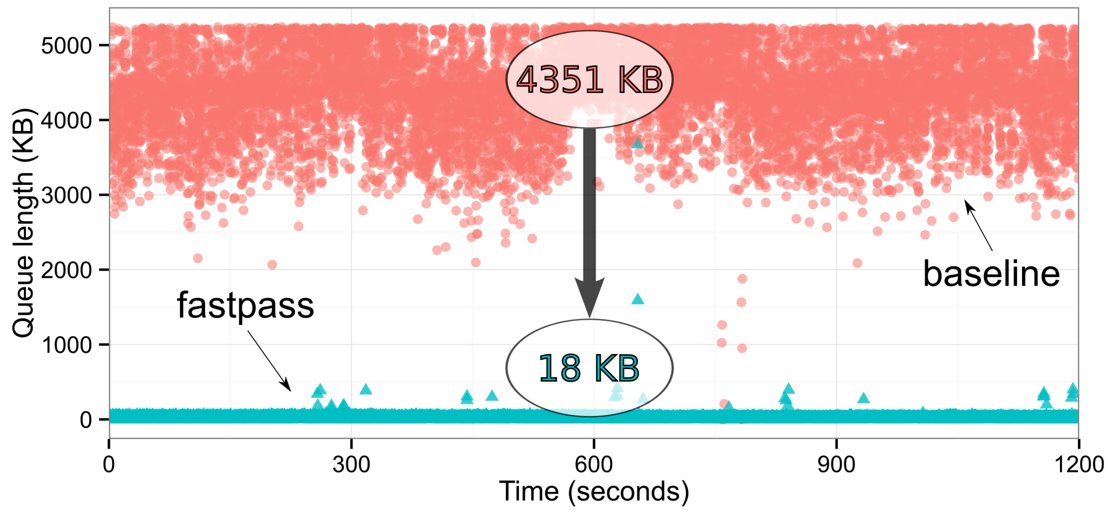
Switch queue length, RTT

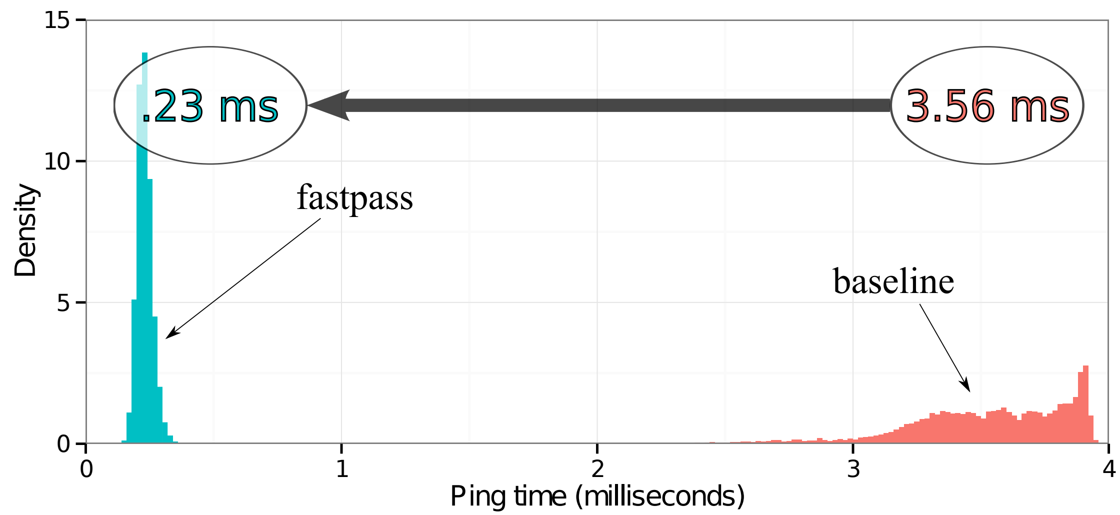
Convergence to network share

Reducing retransmission in production

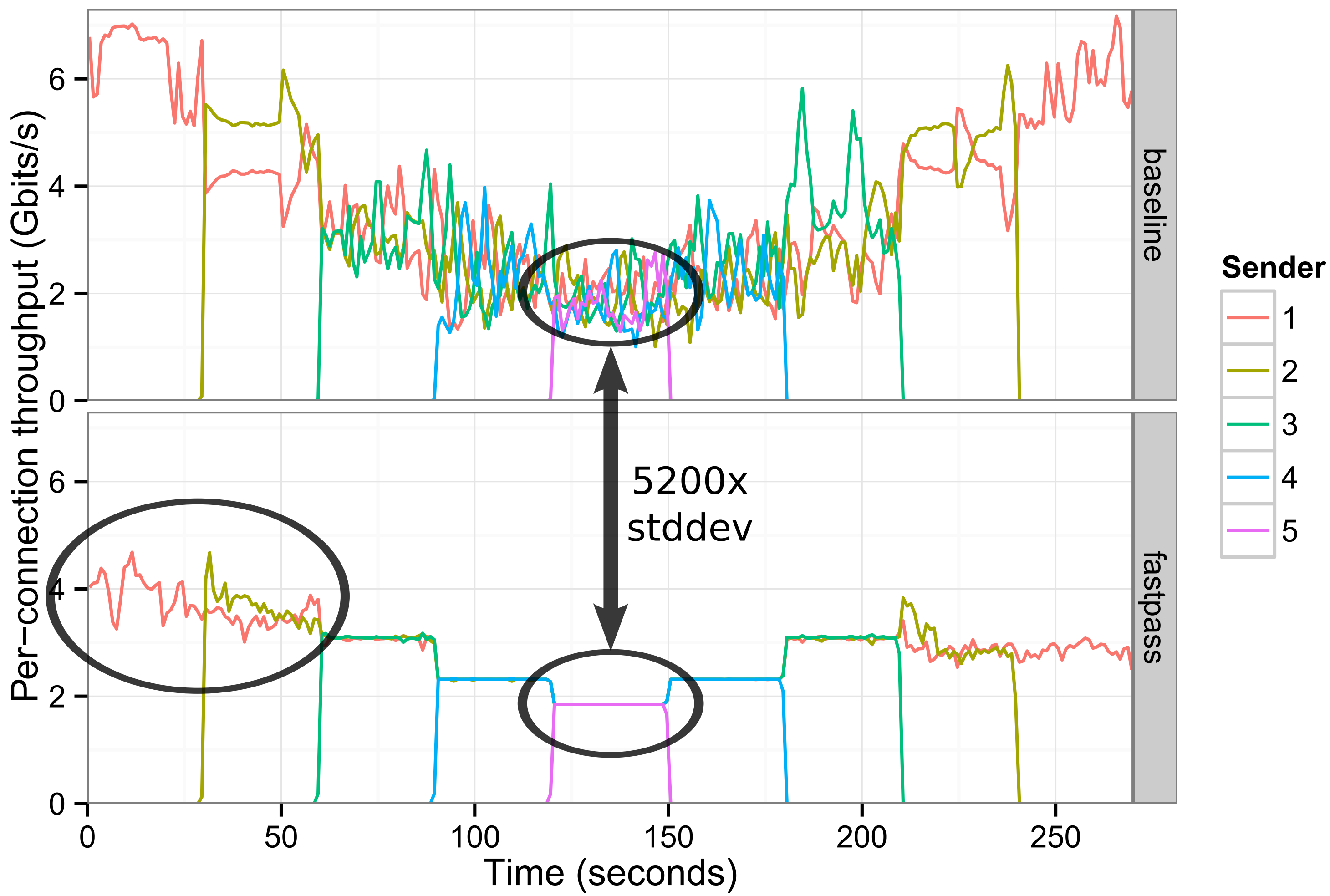
Queues & RTT


TCP
ping

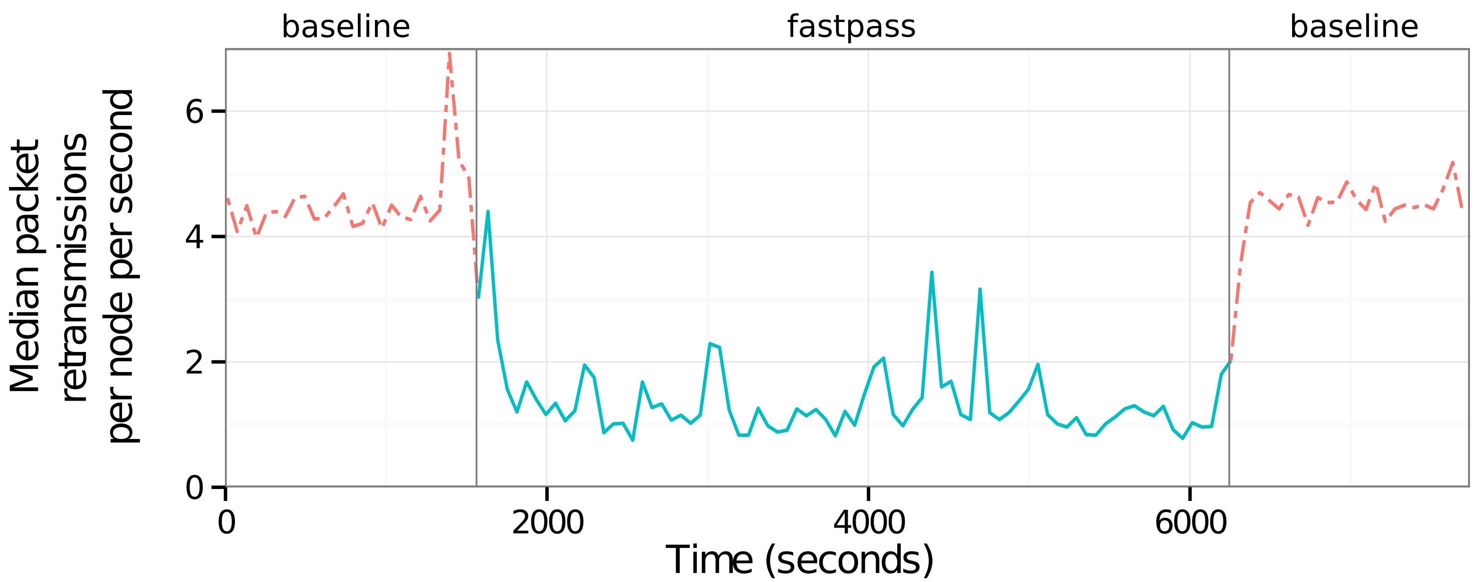

Queue length (KB)
5000
4000
3000
2000
1000
0
0
300
600
900
1200
Time (seconds)
4351 KB
18 KB
fastpass
baseline


Density
15
10
5
0
0
1
2
3
4
Ping time (milliseconds)
.23 ms
3.56 ms
fastpass
baseline

Convergence to network share



Reducing retransmissions in production



Each server: ~50k QPS

Benefits

A: "Now I can see pictures of other's people's food and children so much more quickly...can't wait..>.>"

B: "You forgot about [...] cats. I will say, faster pics of cats is probably worth some merit."

Benefits

Low user latency

Stronger network semantics

No packet drops, predictable latency, deadlines, SLAs

Developer productivity

Less dealing with bursts, tail latency, hotspots
Simplify building complex systems

Lower infrastructure cost

Less over-provisioning

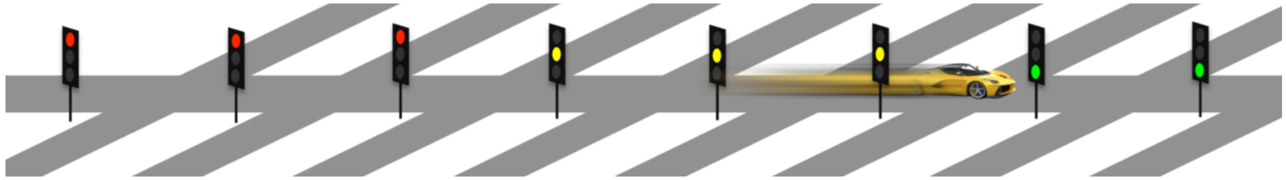
Fastpass enables new network designs

Traditional	Flow control	Congestion control	Update routing tables	Scheduling & Queue management	Packet forwarding
SDN	Flow control	Congestion control	Update routing tables	Scheduling & Queue management	Packet forwarding
Fastpass	Flow control	Congestion control	Per-packet path selection	Scheduling & Queue management	Packet forwarding

Endpoint | Centralized | Switch

Fastpass: centralizes control at packet granularity

Switches can become even simpler and faster



Conclusion

Zero network queues
High Utilization
Multiple app and user objectives

Pushes centralization to a logical extreme

Opens up new possibilities for even faster networks

Code (MIT licensed): http://fastpass.mit.edu