Fastpass: A Centralized "Zero-Queue" Datacenter Network

SIGCOMM2014

Why Design Fastpass

- Large queue is presented in datacenter network. In a facebook cluster, the total queue length 4.35Mbytes.

- Large queue brings significant queuing delays and a lot of packet retransmissions.

- Want to decrease or even eliminate the queues in the switch, but:
  - A pure server based implementation does not have a network overview, can not make correct decision.
  - A switch based implementation is hard to deploy in a large production network because additional new hardware features are needed.

- A Cetralized controller plus fine grained control may eliminate the network queue.

Why Fastpass

- At what granularity should the control be:
  - At the granularity of a timeslot that a 1500 MTU packet uses to traverse a 10Gbps link, which is around 1us.
  - Seemingly impossible, but if a transmission is carefully planned ahead and using some batch processing and parallelism it is possible, even inside a datacenter network with thousands of nodes.

- Timeslot: The time that 1500 MTU packet uses to traverse a 10Gbps link.

- At any given timeslot, a most one packet is scheduled to run on each link, eliminating queue at the switch.

About Fastpass

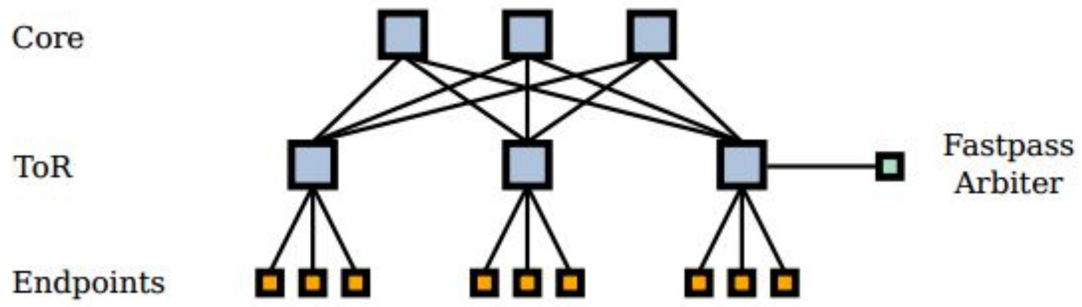


Figure 1: Fastpass arbiter in a two-tier network topology.

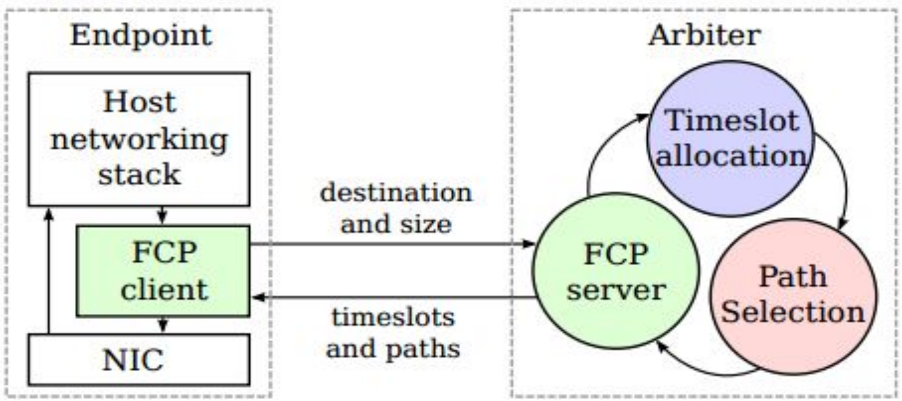
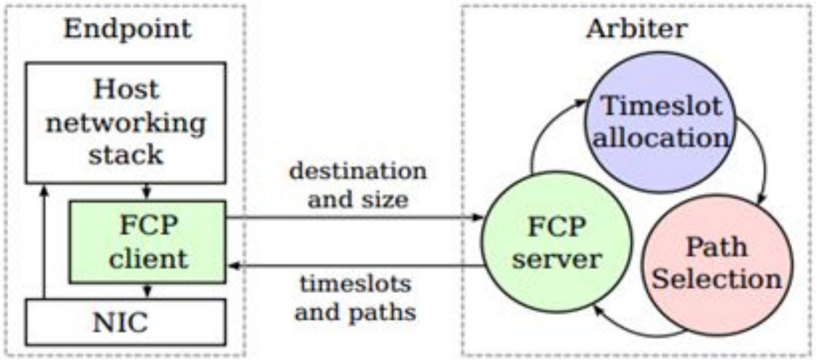


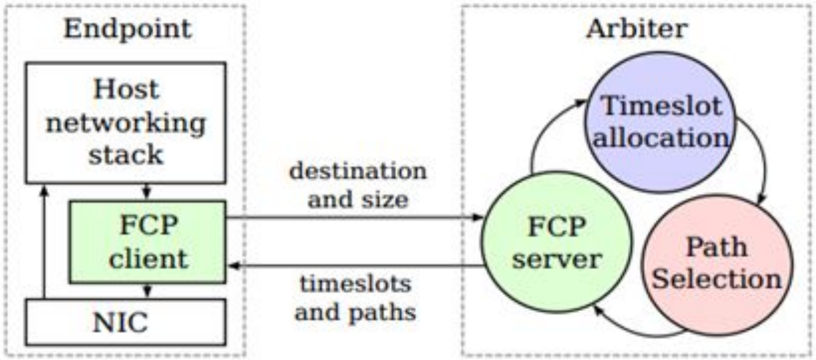
Figure 2: Structure of the arbiter, showing the timeslot allocator, path selector, and the client-arbiter communication.

At End-point



- When application calls send or sendto system calls, packets will first get delivered to FCP clients, queued there.
- FCP clients aggregate the demand of transmission during a few microseconds and transmit the demand to the arbiter.
- Demand is actually defined as the number of timeslots needed to fully transmit all the packets.

At Arbiter

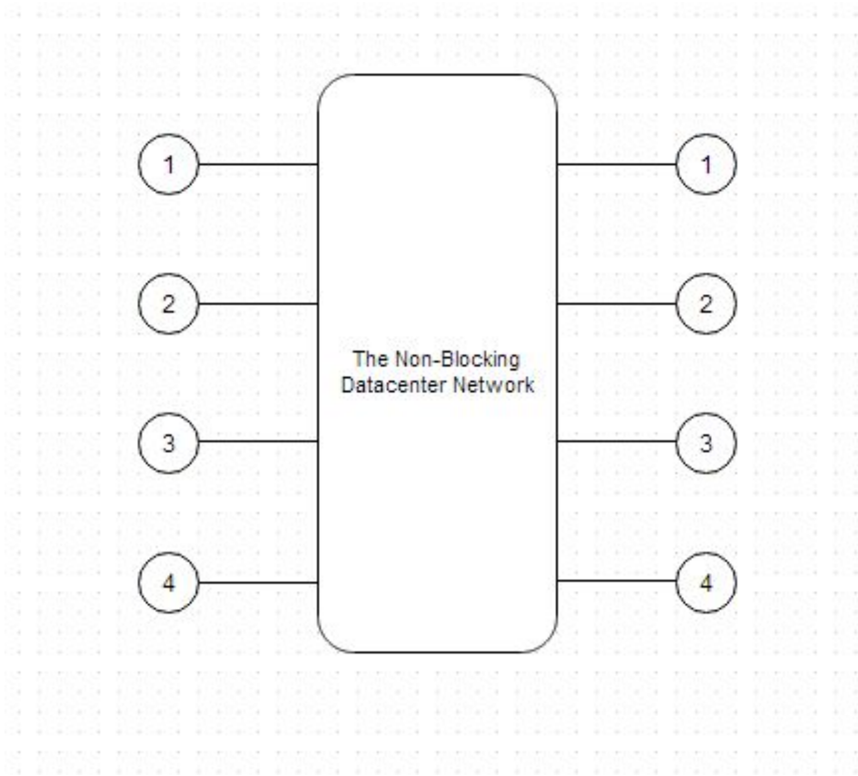


- Three major componants.

- Each componant works in parallel and takes several cores to speed up execution.

Timeslot Allocation at Arbiter

- A preliminary condition for the network: rearrangebly non blocking.
- Only access links are bottleneck links for a traffic matrix. Any traffic matrix that satisfies the access link constraint could be routed in the network.
- This condition makes thing easy. Just do a maximal matching for all source-destination pair.



Timeslot Allocation at Arbiter

Figure 4: Timeslot allocation for the max-min fairness allocation objective.

Timeslot Allocation in Fastpass

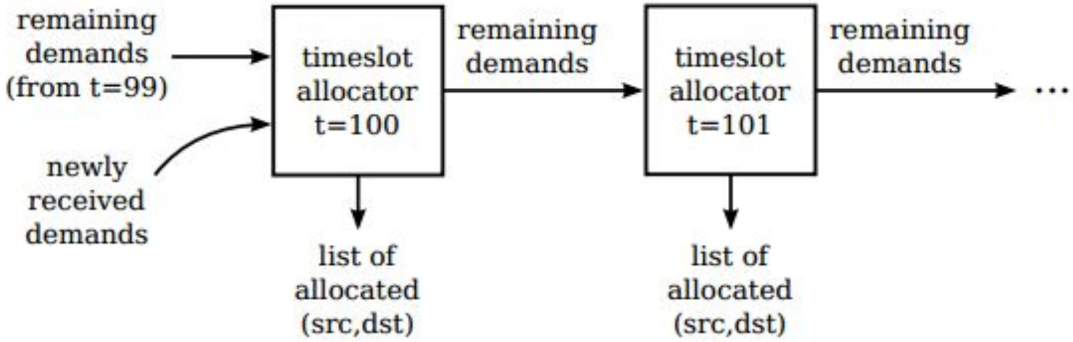
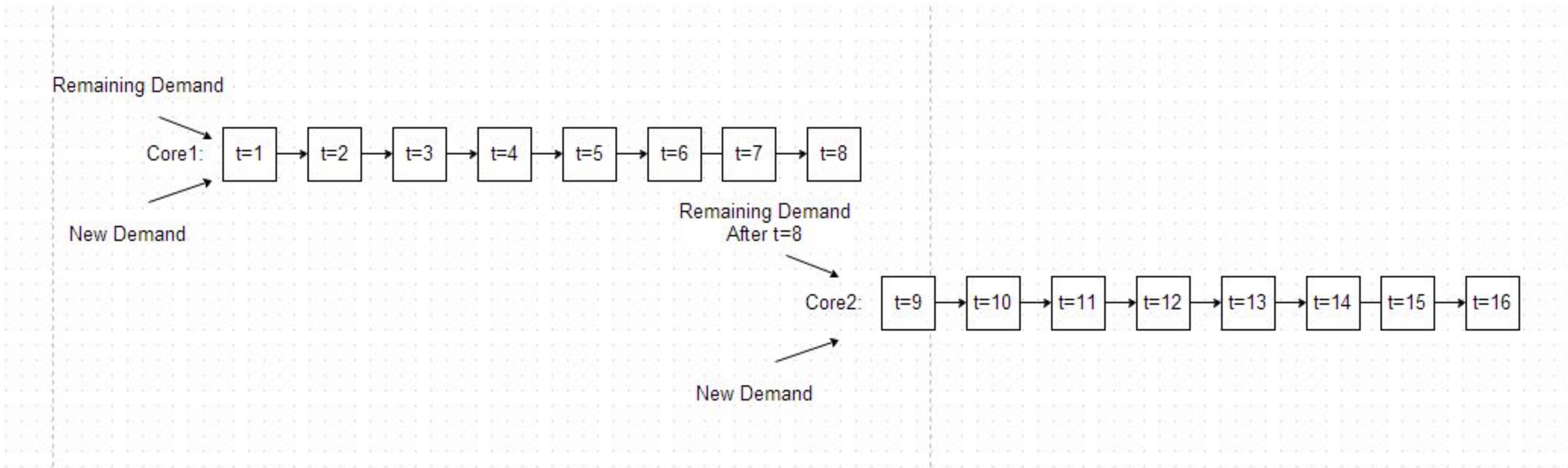


Figure 3: Pipelined timeslot allocation. The allocator for timeslot t processes the demands not fully satisfied by the allocator for $t-1$



Path Selection

- Assign Path to packets such that no link is assigned multiple packets in a sing time slot.
- Can be achieved using edge-coloring.
- The algorithm has time complexity O(nd logd) using exsiting algorithm.

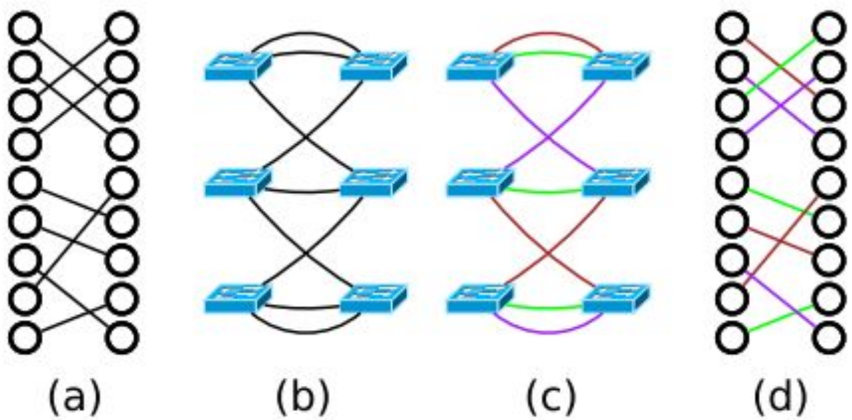


Figure 5: Path selection. (a) input matching (b) ToR graph (c) edge-colored ToR graph (d) edge-colored matching.

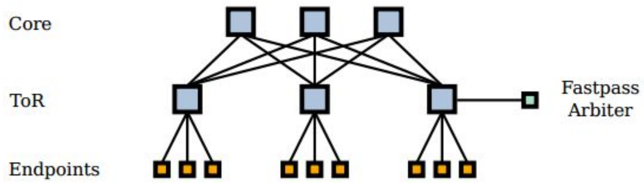


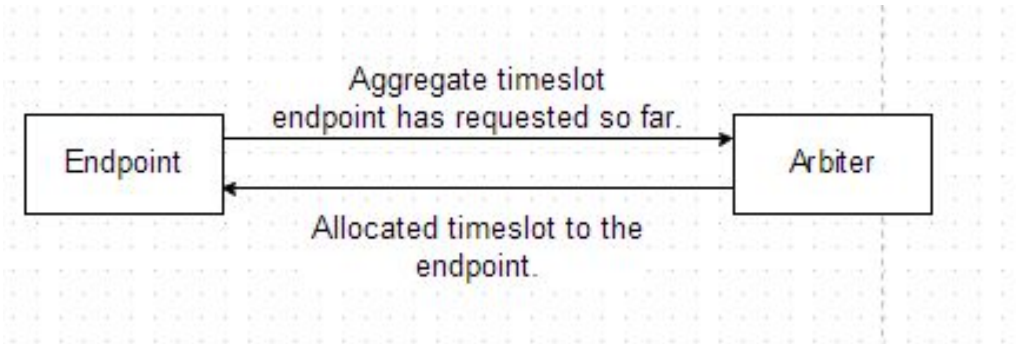
Figure 1: Fastpass arbiter in a two-tier network topology.

Handling Failure

- Arbiter: One primary, multiple secondary.
- Requests are multicasted to both primary and secondary Arbiters.
- Only primary arbiter responds the requests. Secondary Arbiter discards requests.
- Primary Arbiter sends out watchdog packet every T_w seconds.
- Sedonary Arbiter does not receive warchdog packet for T_d seconds, secondary Arbiter will start to take place.

Handling Failure

- Arbiter only maintains soft state, when primary Arbiter fails, secondary Arbiter needs to resynchronize with endpoints.



About Implementation

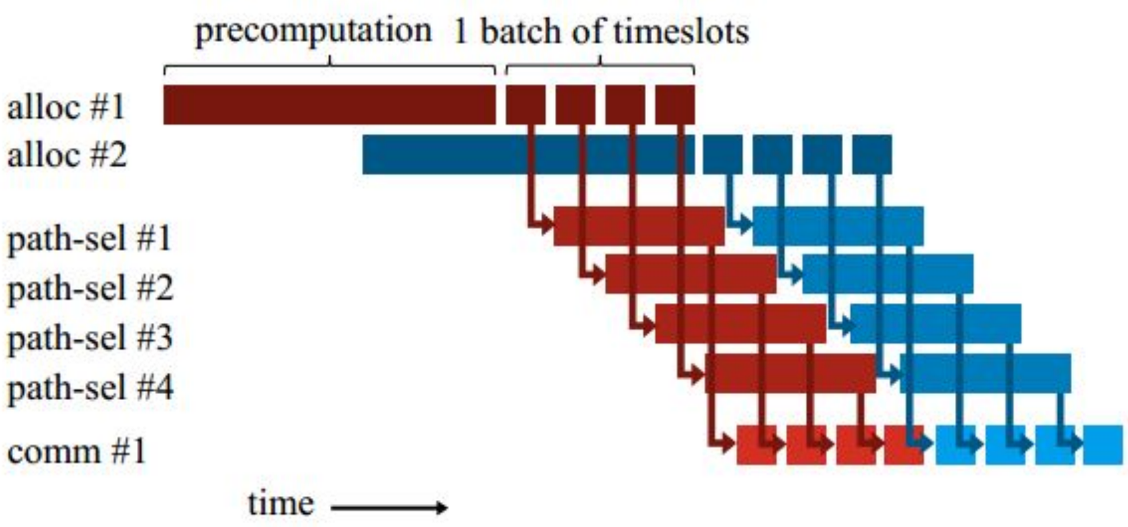


Figure 6: Multicore allocation: (1) allocation cores assign packets to timeslots, (2) path selection cores assign paths, and (3) communication cores send allocations to endpoints.

- Clock is synchoronizes in the system using IEEE1588 Precision Time Protocol.

Experiment A

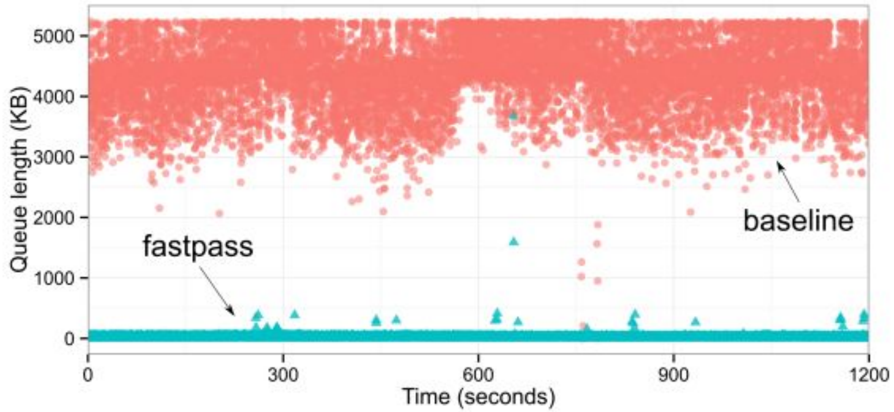


Figure 7: Switch queue lengths sampled at 100ms intervals on the top-of-rack switch. The diagram shows measurements from two different 20 minute experiments: baseline (red) and Fastpass (blue). Baseline TCP tends to fill switch queues, whereas Fastpass keeps queue occupancy low.

- On a rack with 32 servers, four server generate traffic to a single receiver.
- Throughput: 9.43Gbps in baseline, 9.28Gbps in Fastpass. This is because 1% of bandwidth in Fastpass is reserved for tranmitting control messages.

Experiment B

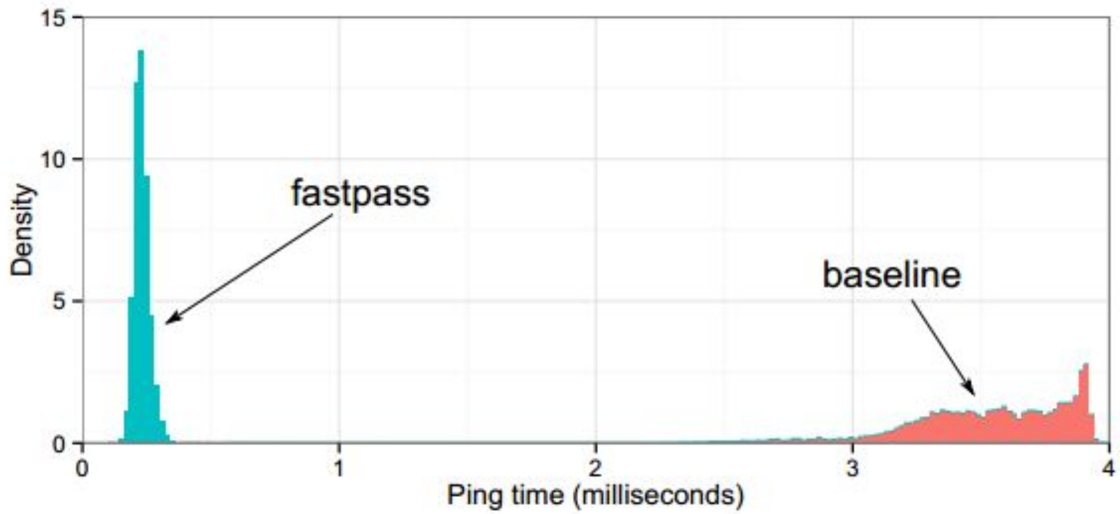


Figure 8: Histogram of ping RTTs with background load using Fastpass (blue) and baseline (red). Fastpass's RTT is 15.5× smaller, even with the added overhead of contacting the arbiter.

- Same setting as A, with a fifth machine sends a small request to receiver every 10ms, using Ping.

Experiment C

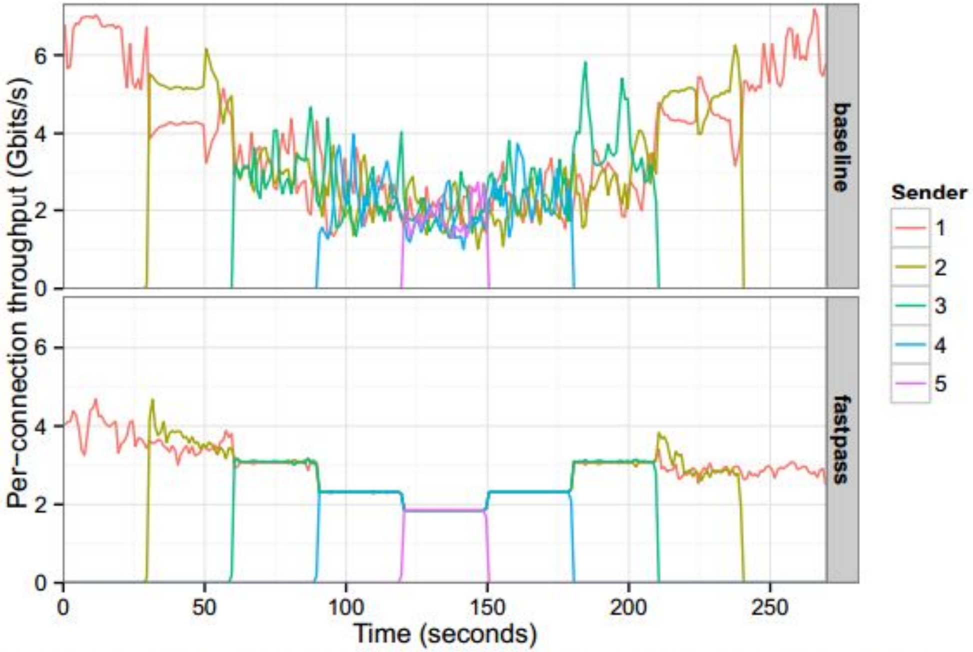


Figure 9: Each connection's throughput, with a varying number of senders. Even with 1s averaging intervals, baseline TCP flows achieve widely varying rates. In contrast, for Fastpass (bottom), with 3, 4, or 5 connections, the throughput curves are on top of one another. The Fastpass max-min fair timeslot allocator maintains fairness at fine granularity. The lower one- and two-sender Fastpass throughput is due to Fastpass qdisc overheads (§7.2).

- Five servers sends to the sixth server. The connections are gradually increased until all five connections are presented in the network.

Experiment E

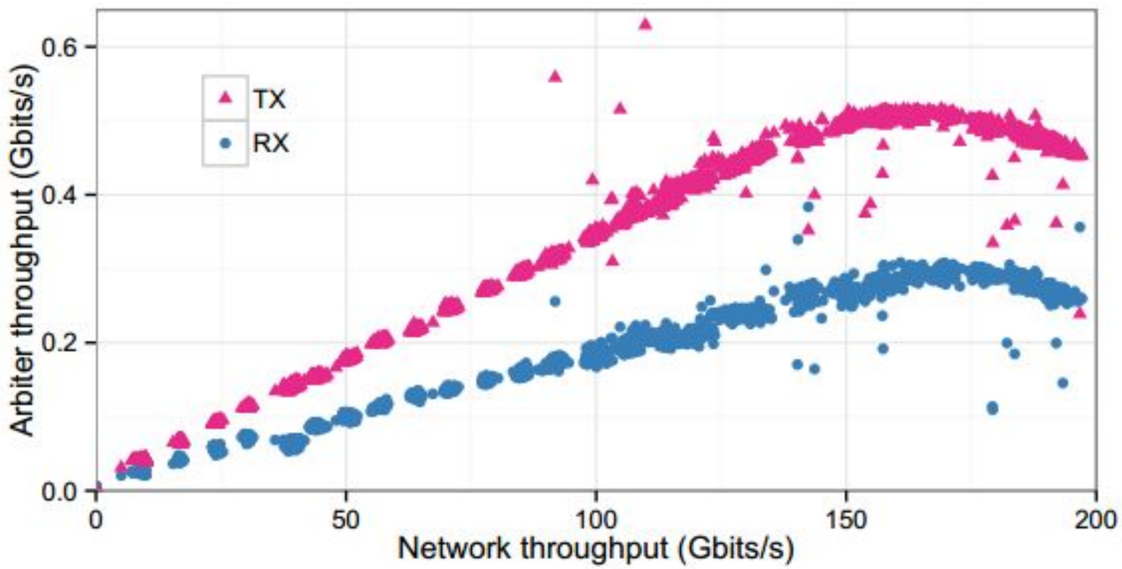


Figure 11: The arbiter requires 0.5 Gbits/s TX and 0.3 Gbits/s RX bandwidth to schedule 150 Gbits/s: around 0.3% of network traffic.

Experiment F: timeslot allocation cores

	2 cores	4 cores	6 cores	8 cores
Throughput (Gbits/s)	825.6	1545.1	1966.4	2211.8

- Traffic are generated using workload cores, not from network.

Experiment G: path selection cores

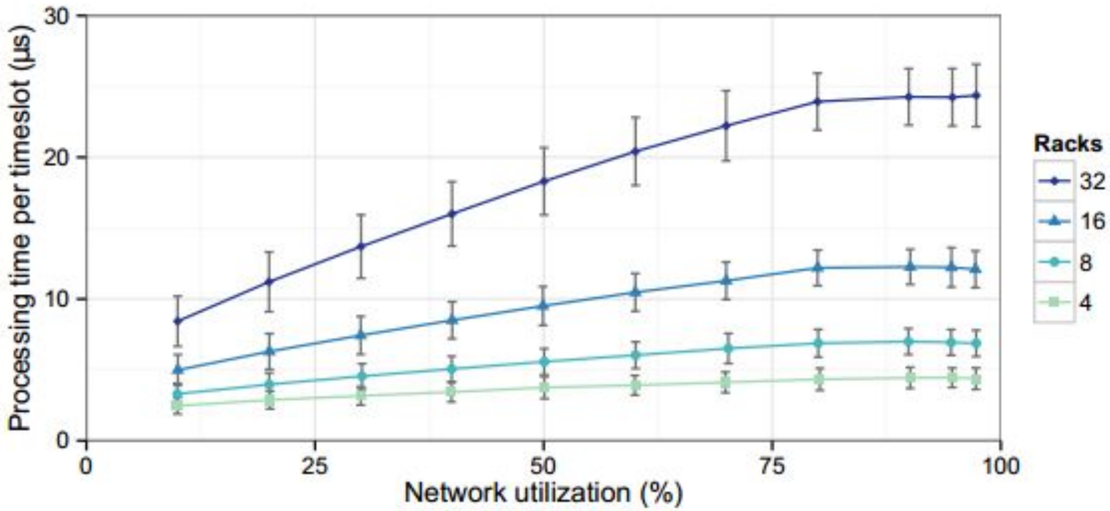


Figure 12: Path selection routes traffic from 16 racks of 32 endpoints in <12 µs. Consequently, 10 pathsel-cores would output a routing every <1.2 µs, fast enough to support 10 Gbits/s endpoint links. Error bars show one standard deviation above and below the mean.

Experiment H: production results

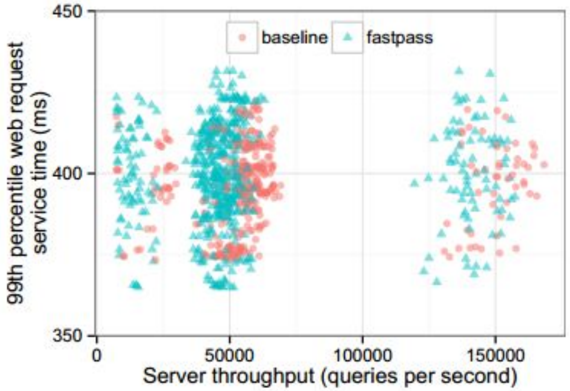


Figure 14: 99th percentile web request service time vs. server load in production traffic. Fastpass shows a similar latency profile as baseline.

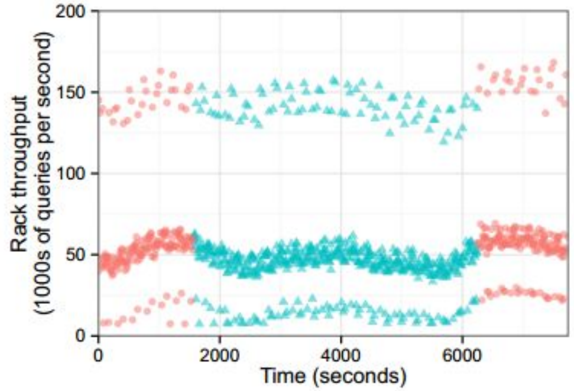


Figure 15: Live traffic server load as a function of time. Fastpass is shown in the middle with baseline before and after. The offered load oscillates gently with time.

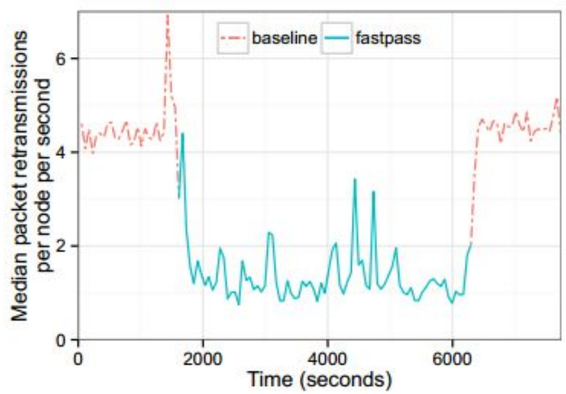


Figure 16: Median server TCP retransmission rate during the live experiment. Fastpass (middle) maintains a 2.5× lower rate of retransmissions than baseline (left and right).