Fastpass
A Centralized “Zero-Queue” Datacenter Network
Ideal datacenter properties

• Burst control
  — memcached, Fine-grained RTO

• Low tail latency
  — pFabric, HULL, DCTCP, D3, Orchestra

• Multiple app/user objectives
  — Hedera, SWAN, MATE, DARD, VL2, …

How to satisfy all these properties simultaneously?
How to design a network with...

- “Zero” network queues
- High utilization
- Multiple resource allocation objectives
Control each packet’s ** timing and path using a centralized arbiter**

<table>
<thead>
<tr>
<th></th>
<th>Flow control</th>
<th>Congestion control</th>
<th>Update routing tables</th>
<th>Scheduling and queue management</th>
<th>Packet forwarding</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Traditional</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SDN</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fastpass</strong></td>
<td></td>
<td></td>
<td>Per-packet path selection</td>
<td>Scheduling and queue management</td>
<td></td>
</tr>
</tbody>
</table>

**Endpoint**

**Centralized**

**Switch**
Architecture
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
## Timeslot Allocation

Ordering of requests used to implement policies. E.g. LRU for max-min fairness, lowest remaining MTUs for min-FCT.

<table>
<thead>
<tr>
<th>active flows</th>
<th>src</th>
<th>dst</th>
<th>last allocation</th>
<th>allocated srcs &amp; dsts</th>
</tr>
</thead>
<tbody>
<tr>
<td>src</td>
<td>dst</td>
<td>src</td>
<td>dst</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>45</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>47</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>48</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>48</td>
</tr>
</tbody>
</table>

Example:
- 1-3: LRU allocation
- 4-2: LRU allocation
- 2-3: LRU allocation
- 3-4: Lowest remaining MTUs
Path selection

Use edge coloring, each color denotes a path
Implementation

- Pipelined execution of tasks over multiple cores
- Clock synchronization using PTP (achieves sub microsecond synchronization)
- Client timing using hrtimers (microsecond scale precision)
Fault tolerance

- Arbiter failures
  - hot backups

- Network failures
  - packet loss reports
Results:

Smaller queues, lower RTTs with iperf and pings
Results:

Lesser retransmissions in production...

Each server: ~50k QPS
Results:

...But latency and throughput profiles were barely different
Issues

• Not really zero-queue — simply relocated to the endpoints and arbiter

• How to scale?
  — Several arbiters would need to cooperate
  — Precise time synchronization required

• How useful is Fastpass in practice?
  — End-to-end delay at varying load
  — Experimental setup had only single ToR