Centralized Arbitration for Data Centers

Lecture 18, Computer Networks (198:552)
Datacenter transport

• Goals: Complete flows and jobs quickly

  Short flows  (e.g., query, coordination)  ➔ Low Latency
  Large flows  (e.g., data update, backup)  ➔ High Throughput
  Big-data jobs  (e.g., map-reduce, ML training)  ➔ Complete fast
Low latency approaches

Endpoint algorithms keep queues small (e.g., DCTCP)

Prioritize mice over elephants in the fabric (e.g., pFabric)
Continuing problems

- Flows need not be the most semantically meaningful entities!
  - **Co-flows**: Big data jobs, partition/aggregate queries
  - Result isn’t meaningful until a set of related flows complete

- Multiple competing objectives: provider & apps
  - Fair sharing vs. efficient usage vs. app objectives

- Change all switches? Endpoints?
Opportunity

• Many DC apps/platforms know flow size or deadlines in advance
  - Key/value stores
  - Data processing
  - Web search
Objective?
- App: Min. avg FCT, co-flow completion time, delays, retransmits
- Net: Share the network fairly

DC transport = Flow scheduling on giant switch

ingress & egress capacity constraints
Building such a network is hard!

• “One big switch” no longer
  • Failures
  • Heterogeneous links
  • Respect *internal link rate* constraints
DC transport = Flow scheduling on giant switch

Objective?
- App: Min. avg FCT, co-flow completion time, delays, retransmits
- Net: Share the network fairly

Ingress, egress, and core capacity constraints
Building such a network is hard!

- “One big switch” no longer
  - Failures
  - Heterogeneous links
  - Respect internal link rate constraints

- Switch modifications to prioritize at high speeds

Can we do better?
Objective?
- App: Min. avg FCT, co-flow completion time, delays, retransmits
- Net: Share the network fairly

DC transport = A controller scheduling packets at fine granularity
FastPass

Jonathan Perry et al., SIGCOMM’14
FastPass Goals

• Is it possible to design a network that provides

  Zero network queues

  High utilization

  Multiple app and provider objectives
Logically centralized arbiter

Processing times

(1) 10 us A→arbiter
(2) 1—20 us Arbiter
(3) 10 us Arbiter→A
(4) No queuing A→B

(4) Packet A → B with 0 queues

(1) A has 1 packet for B
(2) Arbiter allocates time slot & path for pkt
(3) At t=95, send A→B through R1
Logically centralized arbiter

• Schedule and assign paths to all packets

• Will the arbiter scale to schedule packets over a large network?
• Will the latency to start flows be a problem?
• How will the system deal with faults?

• Can endpoints send at the right time?
• Clocks must be synchronized!

Concerns with centralization

Concerns with endpoint behavior
Arbiter problem (1): Time slot allocation

• Choose which hosts get to send
  • Send as much as possible without endpoint port contention

• Allocate packet transmissions per *time slot*
  • 1.2 us: 1500 byte packet at 10 Gbit/s
Arbiter problem (2): Path selection

• Select paths for allocated host transmissions

• Ensure no two packets conflict on *any* network port
  • At all levels of the topology
Can you solve (1) and (2) separately, or should we solve them together?

Solve separately if the network has **full bisection bandwidth**

i.e.: **one big switch**: any port to any other without contention
Time slot allocation == Matching problem!

• Want: optimal matching
  • Maximize # packets transmitted in a timeslot
  • Expensive to implement!

• Can do: online maximal matching

• Greedy approach: allocate to current time slot if source and destination free; else push to next time slot

• Fastpass: allocates in 10 ns per demand
Pipelined time slot allocation

Allocate traffic for 2 Tbits/s on 8 cores!
Time slot allocation: Meeting app goals

- Ordering the demands changes achieved allocation goal
  - Max-min fairness: least recently allocated first
  - FCT minimization: fewest remaining MTUs first

- Allocation in batches of 8 timeslots

<table>
<thead>
<tr>
<th>src</th>
<th>dst</th>
<th>last allocation</th>
<th>allocated srcs &amp; dsts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>45</td>
<td>✓</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>47</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>48</td>
<td>×</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>48</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>51</td>
<td>✓</td>
</tr>
</tbody>
</table>
Path selection

• Assume a 2-tier topology: ToR $\rightarrow$ aggregate switch $\rightarrow$ ToR
• Given pair of ToRs, find non-overlapping agg switch
• **Edge coloring problem!**
Will the delay to start flows be an issue?

• Arbiter must respond *before* ideal zero-queue transmission occurs

• Light load: independent transmissions may win 😊
Can endpoints transmit at the right time?

• What happens if not?

• (1) Need to ensure that clocks aren’t *too out of sync*
• (2) Endpoints transmit when told

• Use PTP to sync within few us

• Fastpass \texttt{qdisc} to shape traffic as told
  • \texttt{hrtimers} for microsecond resolution
Experimental results on Facebook rack
Convergence to network share
Discussion

- Ordering demands: an indirect way to set allocation goals?
- Online time slot allocation without preemption: what about priorities?
- Comparison with delay-based congestion control? Fine-grained network scheduling approaches (ex: pFabric)?
- How to scale to much larger networks?
- Can you completely eliminate retransmissions and reordering?
  - What is the impact on the host networking stack?
- Do you need to arbitrate the arbitration packets?
Acknowledgment

• Slides heavily adapted from material by Mohammad Alizadeh and Jonathan Perry