Flexplane: An Experimentation Platform for Resource Management in Datacenters

Amy Ousterhout, Jonathan Perry, Hari Balakrishnan, Petr Lapukhov
Resource Management

• Dozens of new resource management schemes
  – DCTCP, PDQ, RCP, HULL, pFabric, LSTF, D³, etc.

• Difficult to experiment with in real networks
  – Schemes require changes to hardware routers
Experimentation with Resource Management

• Experimentation in real networks
  – Software routers - limited throughput
  – Programmable hardware - limited flexibility

• Experimentation in simulation (e.g., ns2)
  – Does not accurately model real network stacks, NICs, and applications
Flexplane: an Experimentation Platform

• Goal: faithfully evaluate resource management schemes

• Flexplane provides:
  – Accuracy – predict behavior of hardware
  – Flexibility – express schemes in C++
  – High throughput – run at hardware rates
Approach: Whole-Network Emulation

- Emulator maintains a model of the real network
- Users implement schemes in emulated routers
- Packets experience same behavior in emulator as in hardware network running same scheme
Three Steps for Each Packet

• *Convey* abstract packet to emulator
• *Emulate* the network behavior
  – Time divided into timeslots
• *Reflect* behavior onto real network
Accuracy

• Goal: predict behavior of a hardware network

\[ l = u + q \]

**Hardware:**

\[ l' = r + t_e + q_e + u + q' \leq 4u + q' + q_e \]

- \( l \): latency
- \( u \): unloaded delay
- \( q \): queuing delay
- \( r \): RTT to the emulator
- \( t_e \): transmission delay
- \( q_e \): switch delay
- \( u = t + s \)
Flexplane API

• Decouples framework from schemes

Emulator

- int route(AbstractPkt *pkt)
- int classify(AbstractPkt *pkt, int port)
- enqueue(AbstractPkt *pkt, int port, int queue)
- AbstractPkt *schedule(int output_port)

incoming packets → route → classify → enqueue → schedule → outgoing packets
Multicore Emulator Architecture

- Pin network components (routers, endpoints) to cores
  - Router state not shared across cores
- Communication via FIFO queues
- Achieves 761 Gbits/s with 8 cores
Flexplane is Easy to Use

- Implemented several resource management schemes in dozens of lines of code

<table>
<thead>
<tr>
<th>scheme</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>drop tail queue manager</td>
<td>39</td>
</tr>
<tr>
<td>RED queue manager</td>
<td>125</td>
</tr>
<tr>
<td>DCTCP queue manager</td>
<td>43</td>
</tr>
<tr>
<td>priority queueing scheduler</td>
<td>29</td>
</tr>
<tr>
<td>round robin scheduler</td>
<td>40</td>
</tr>
<tr>
<td>HULL scheduler</td>
<td>60</td>
</tr>
<tr>
<td>pFabric QM, queues, scheduler</td>
<td>251</td>
</tr>
</tbody>
</table>
Flexplane is Accurate

- Bulk TCP: 5 senders, 1 receiver
- Throughput 9.2-9.3 Gbits/s vs. 9.4 Gbits/s in hardware
- Similar queue occupancies

<table>
<thead>
<tr>
<th>Queue Occupancies (MTUs)</th>
<th>Hardware</th>
<th>Flexplane</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>median</td>
<td>σ</td>
</tr>
<tr>
<td>DropTail</td>
<td>931</td>
<td>73.7</td>
</tr>
<tr>
<td>RED</td>
<td>138</td>
<td>12.9</td>
</tr>
<tr>
<td>DCTCP</td>
<td>61</td>
<td>4.9</td>
</tr>
</tbody>
</table>
Flexplane Enables Experimentation

- Reproducible research in real networks

(a) (0, 100KB): Average

(b) (0, 100KB): 99th percentile

(c) (10MB, ∞): Average

- Experiment with Spark
  - Results depend on resource management scheme and application