FairCloud
Sharing Cloud Networks across Multiple Entities
Gautam Kumar, Mosharaf Chowdhury with Lucian Popa, Arvind Krishnamurthy, Sylvia Ratnasamy, Ion Stoica

The Problem
• Unlike CPU or memory, network is shared in a best-effort manner in the cloud
  » Lack of predictability hurts applications
• Network share of a VM V depends on
  » Collocated VMs,
  » Destination VMs, and
  » Cross-traffic on each link used by V.

Objectives
1. Allocate network resources to VMs to reflect payments
2. Provide bandwidth guarantees to enable predictable performance

Core Properties
1. Strategy-Proofness
   » No cheating using application-level tricks
2. Symmetry
   » Allocation is same in both directions for same demands
3. Work Conservation
4. Independence
   » Allocation in one link does not depend on allocation in other links

Tradeoff
1. Payment proportionality
   » How closely is the allocation correlated with payment?

Per Endpoint Sharing (PES)
• Associate weights to VM-VM communications
• Proportional allocation
• Weight of link A-B is
  \[ W_{A:B} = \frac{W_A}{N_A} + \frac{W_B}{N_B} \]

PES Variants
1. Local PES
   » \( N_A \): Number of VMs A communicates with on link L
   » Can be implemented in local switch
   » Weights differ on a link to link basis
2. Global PES
   » \( N_A \): Number of VMs A communicates with in the entire network
   » Constant weights across all links, can be implemented using CSFQ

Comparison

<table>
<thead>
<tr>
<th></th>
<th>Strategy-Proofness</th>
<th>Symmetry</th>
<th>Work Conservation</th>
<th>Independence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Per-Flow</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Per-Source</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Oktopus</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Local PES</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Global PES</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>

Implementation
• Flow-level simulator
  » Approximates TCP flow/congestion control
  » Initially compared 13 different strategies
  » ~4000 SLOC in Java
• Click implementation of Global PES using WFQ
  » 15 node DETERlab tested
  » ~1000 SLOC in C++/Python

Results

Future Work
• Implement Global PES using CSFQ to avoid per-router state
• Simulate MapReduce trace from Facebook to compare different strategies