FairCloud

Sharing Cloud Networks across Multiple Entities

Gautam Kumar, Mosharaf Chowdhury *with* Lucian Popa, Arvind Krishnamurthy, Sylvia Ratnasamy, Ion Stoica

amplab UC Berkeley

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

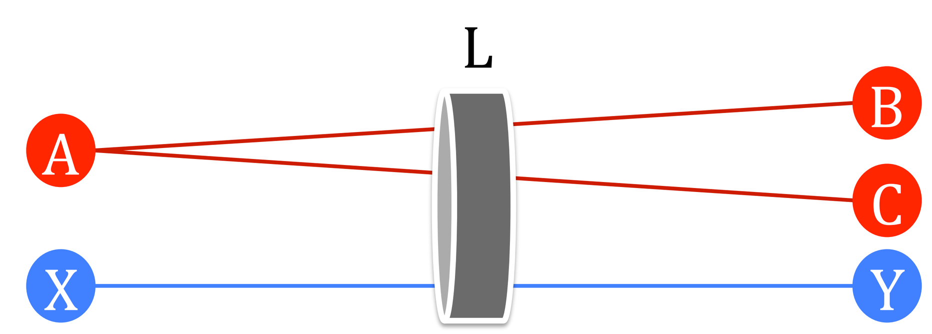
1. Payment proportionality
- » How closely is the allocation *correlated* with payment?

VS

2. Guaranteed Bandwidth
- » The *minimum* bandwidth that a VM can send/receive at

Per Endpoint Sharing (PES)

- Associate weights to VM-VM communications
- Proportional allocation
- Weight of link A-B is $W_{A\text{-}B} = \frac{W_A}{N_A} + \frac{W_B}{N_B}$



$W_{A\text{-}B} = 1.5/5$, $W_{A\text{-}C} = 1.5/5$ $W_{A\text{-}B\text{-}C} = 3/5$
All VMs have W=1 $W_{X\text{-}Y} = 2/5$

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

	Strategy-Proofness	Symmetry	Work Conservation	Independence
Per-Flow	✖	✔	✔	✔
Per-Source	✖	✖	✔	✔
Oktopus	✔	✔	✖	✔
Local PES	✖	✔	✔	✔
Global PES	✔	✔	✔	✖

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 testbed
  - » ~1000 SLOC in C++/Python

Results

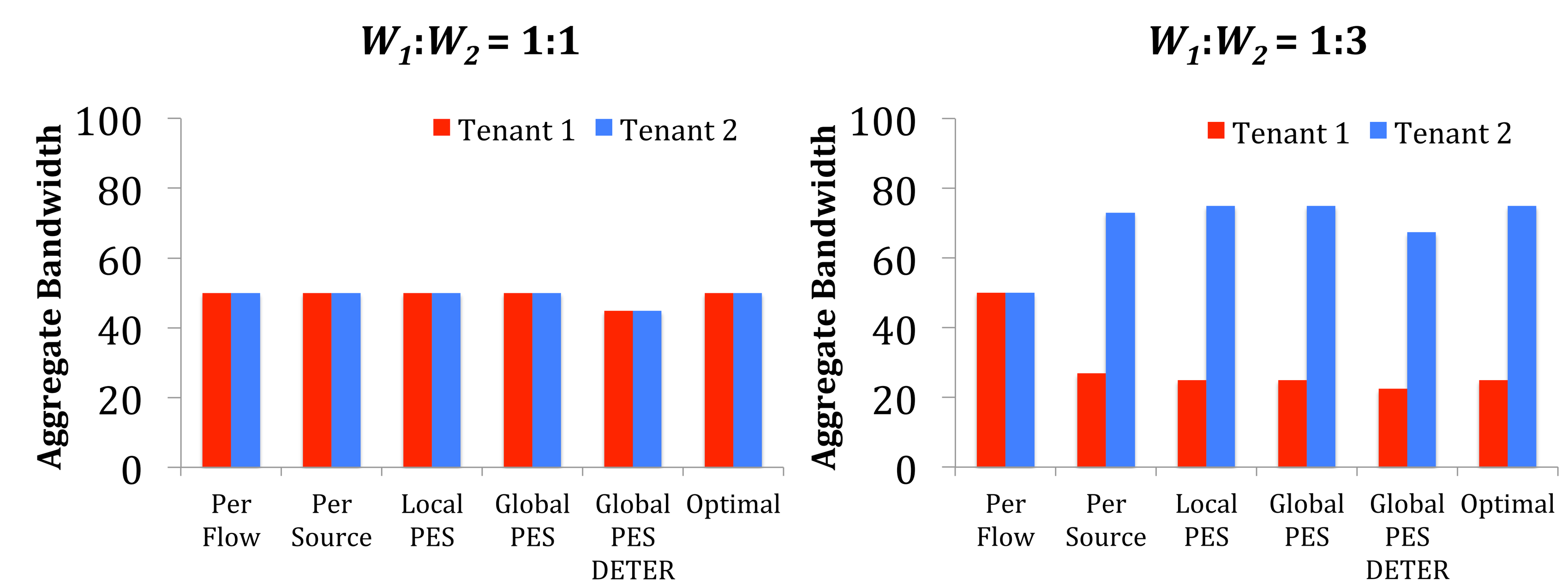


Figure: *Simulation & DETER results showing tenant shares of the aggregate bandwidth in an eight-node three-level oversubscribed tree. Each tenant has four unicast flows through the core.*

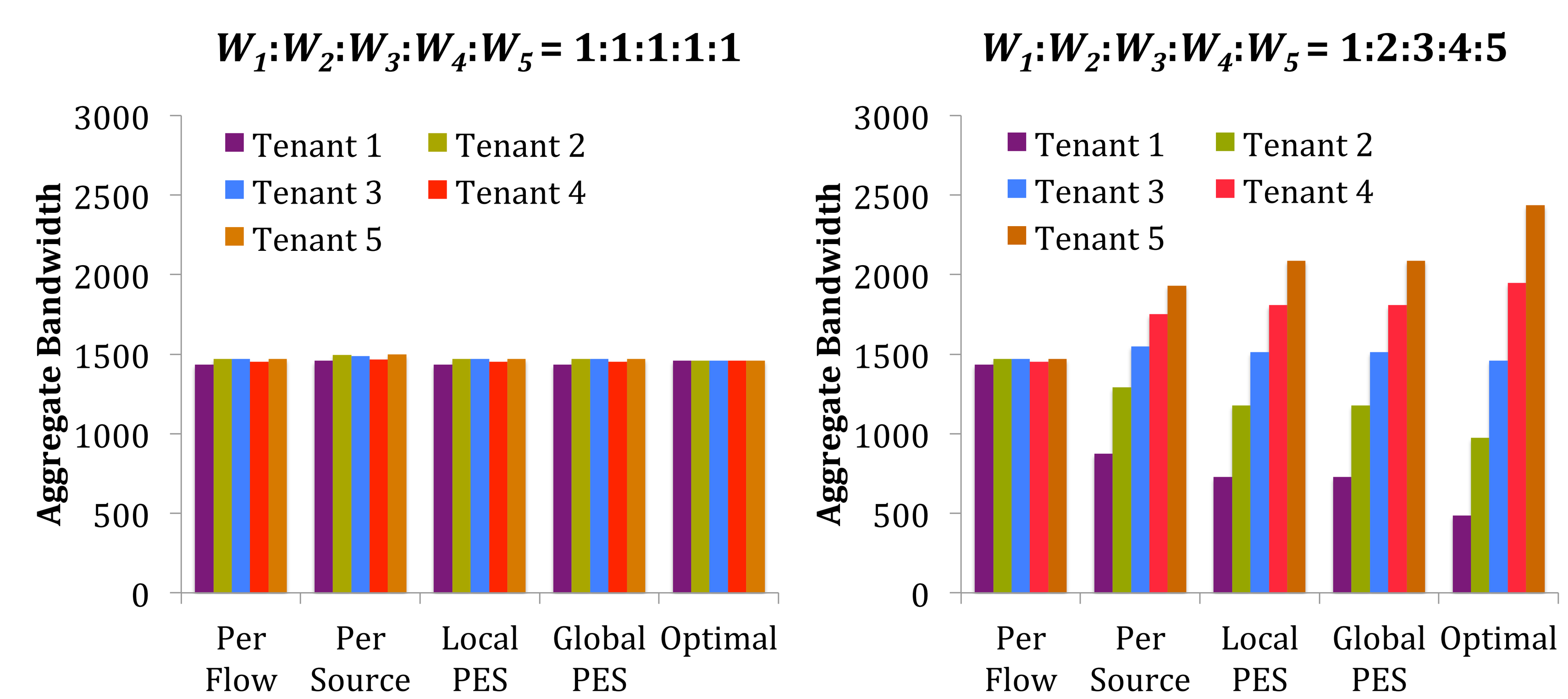


Figure: *Simulation results showing tenant shares of the aggregate bandwidth in an eight-node three-level full bisection bandwidth tree. Each tenant is performing MapReduce shuffle with four mappers and four reducers.*

Future Work

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