FairCloud

Sharing Cloud Networks across Multiple Entities

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

amplab UC Berkeley

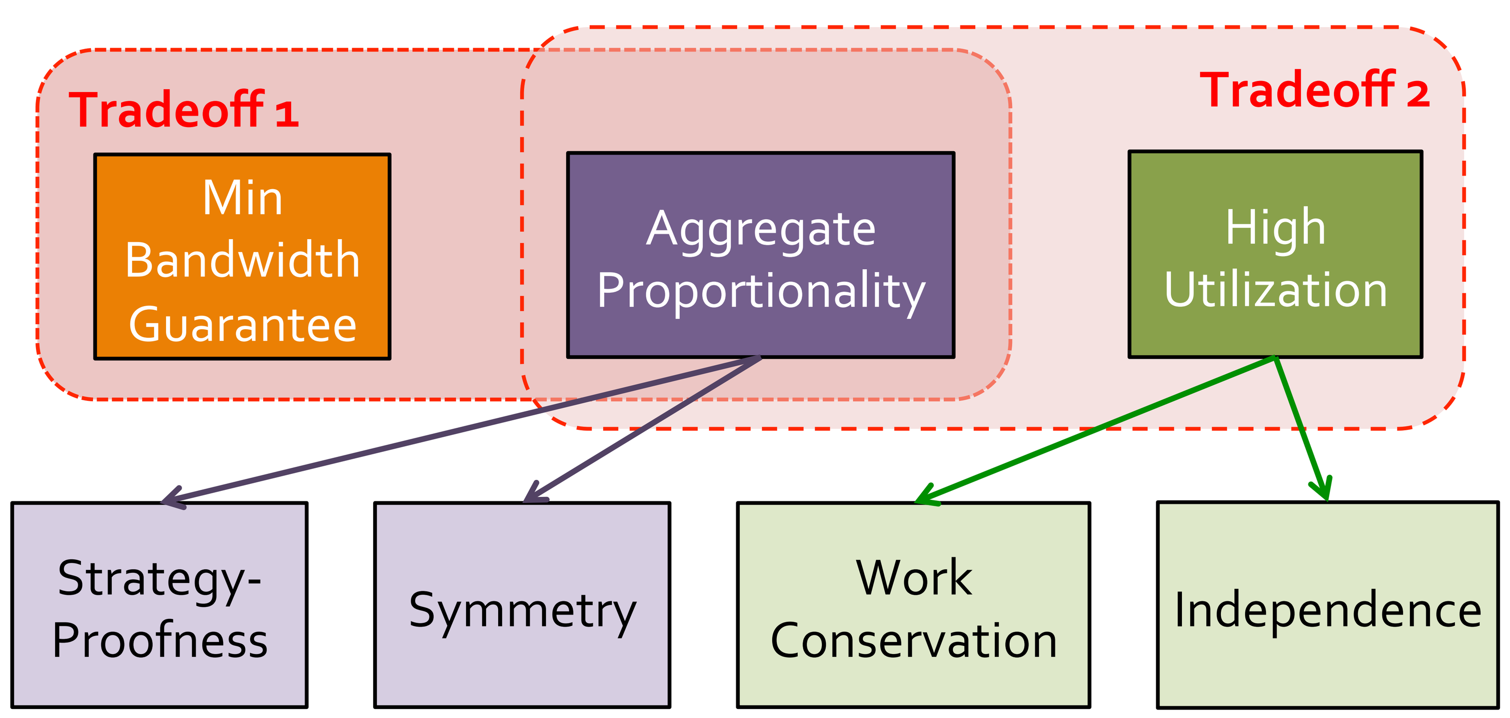
The Problem

- Unlike CPU or memory, network is shared in a **best-effort** manner in the cloud
  - » Lack of predictability hurts applications
- Network sharing is difficult
  - » Usage Attribution
  - » Distributed Resource
- **Objective**
  - » Meaningful sharing of cloud networks

Requirements

1. **Minimum Bandwidth Guarantee**
   - » Captures the desire of tenants to get performance isolation for their applications
2. **Aggregate Proportionality**
   - » Captures payment-proportionality
3. **High Utilization**
   - » Provides incentives such that throughput is only constrained by the network capacity

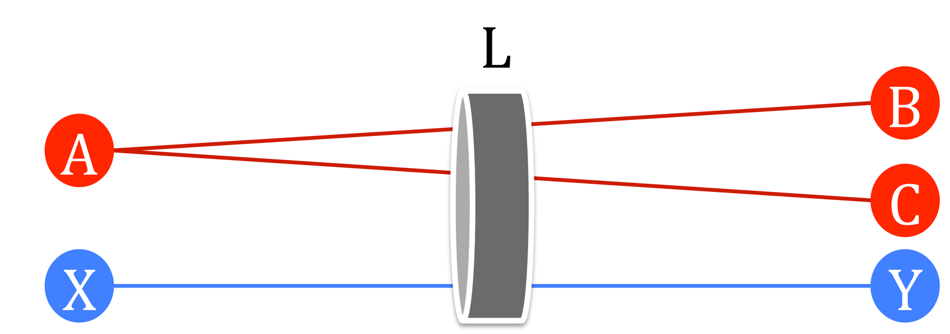
Design Space



- **Strategy-Proofness:** Tenants cannot game allocations
- **Symmetry:** Allocation in any direction of a link is same
- **Work Conservation:** 100% utilization of bottlenecks
- **Independence:** Allocation on any link does not depend on allocation in another

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. **Link 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. **Network 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	✔	✔	✖	✔
Link PES	✖	✔	✔	✔
Network PES	✔	✔	✔	✖

Implementation

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

Results

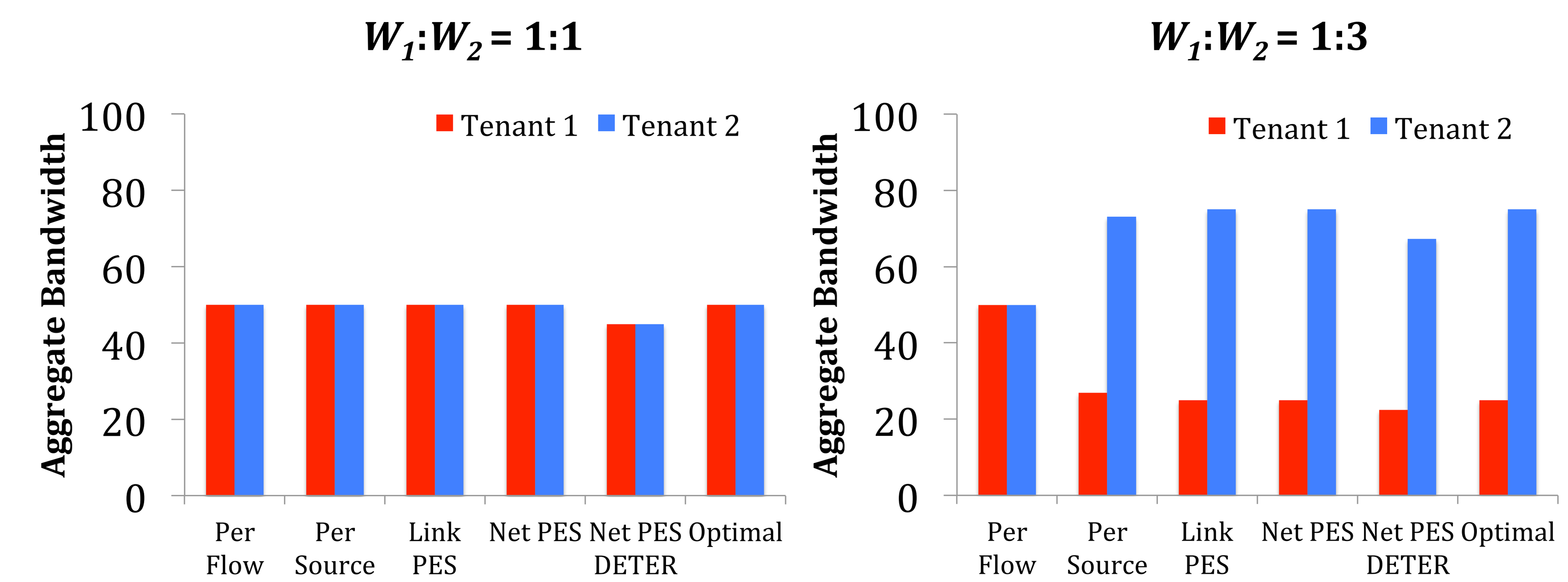


Figure: *Simulation & DETER results showing tenant shares of the aggregate bandwidth in an eight-node three-level oversubscribed tree. Each tenant has four unicast flows through the core.*

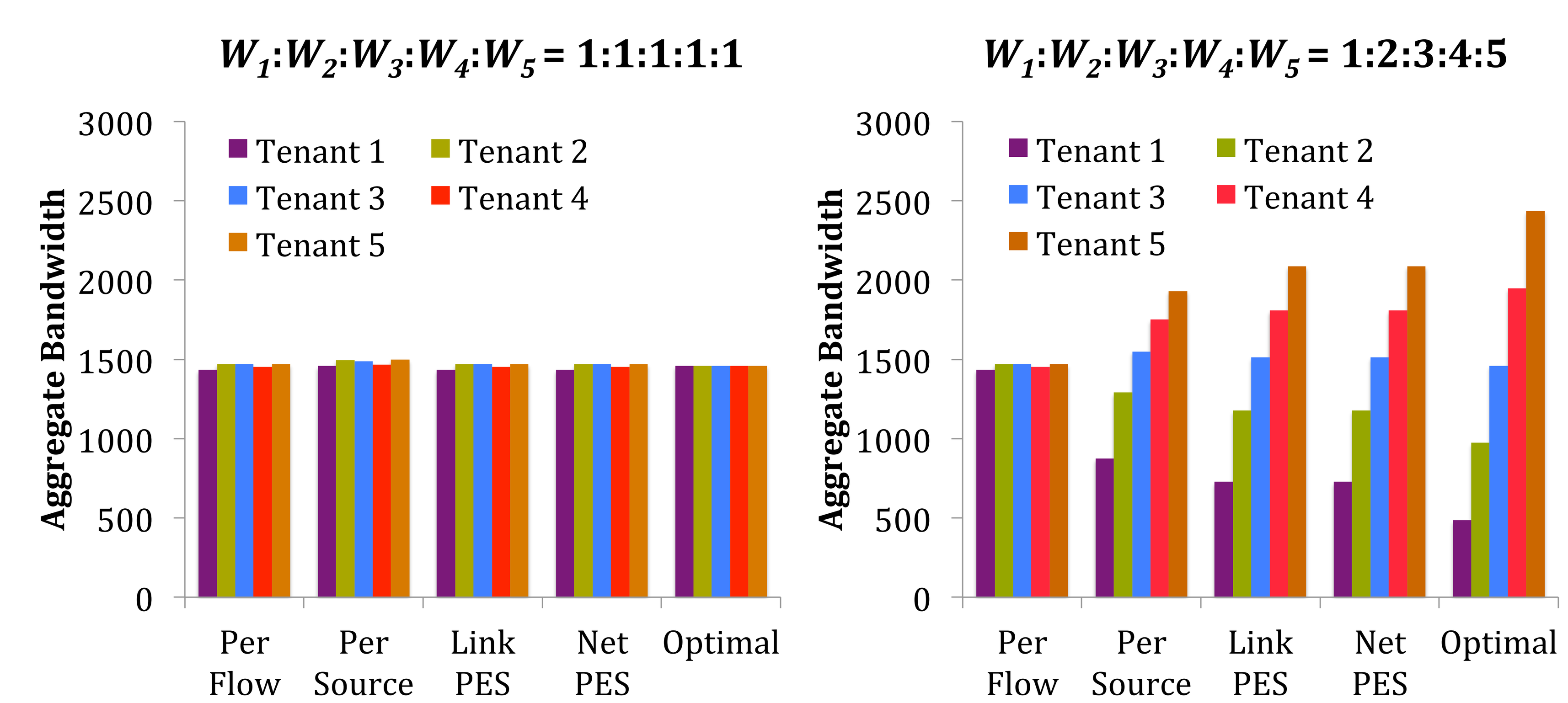


Figure: *Simulation results showing tenant shares of the aggregate bandwidth in an eight-node three-level full bisection bandwidth tree. Each tenant is performing MapReduce shuffle with four mappers and four reducers.*

Future Work

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