Chatty Tenants and the Cloud Network Sharing Problem

Hitesh Ballani†, **Keon Jang**†, Thomas Karagiannis†

Changhoon Kim‡, Dinan Gunawardena†, Greg O'Shea†

MSR Cambridge†, Windows Azure‡

This talk is about . . .

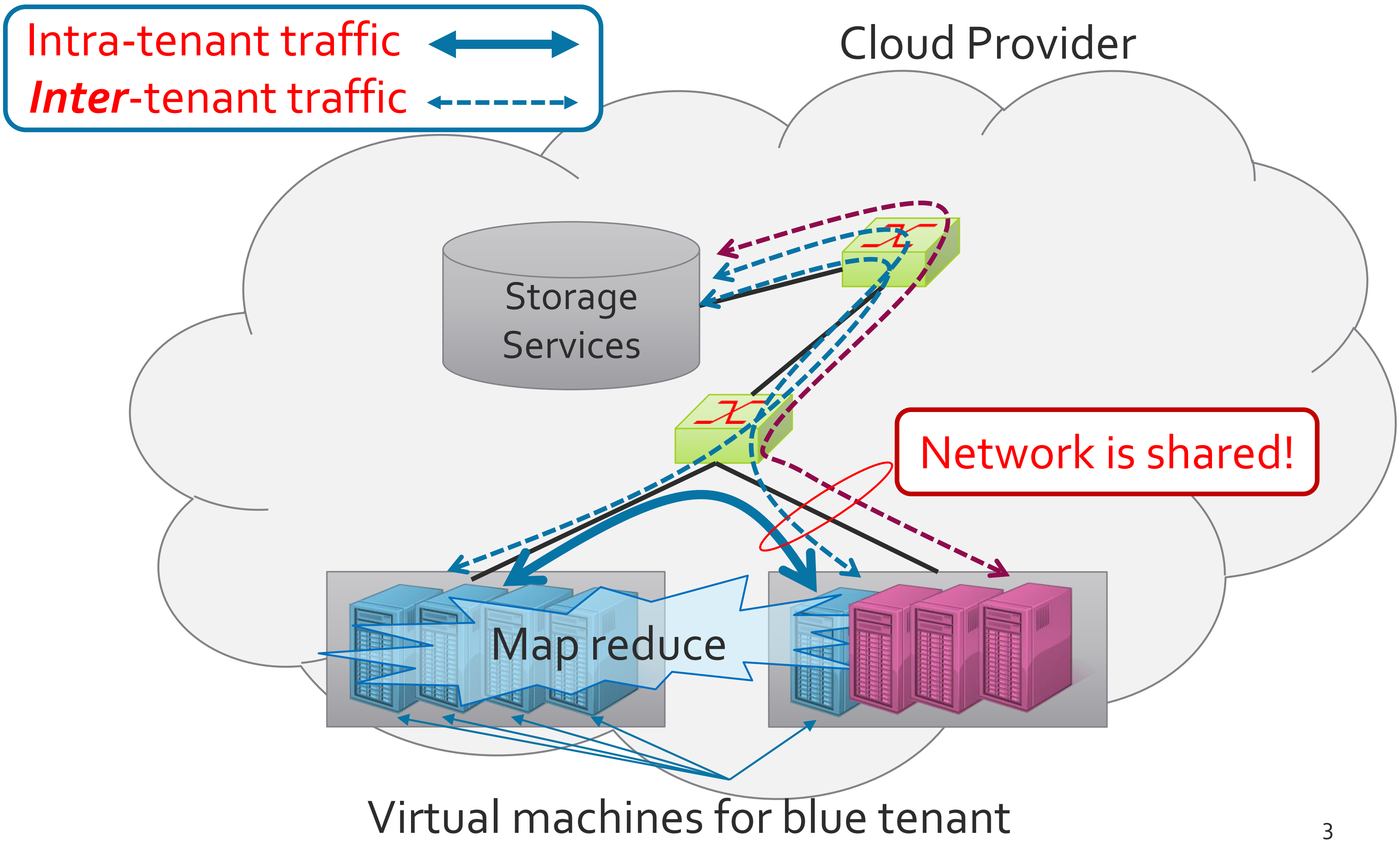
How to share the network in *multi-tenant datacenters?*

Multi-tenant datacenters

- Public cloud datacenters
  - Windows Azure, Amazon EC2, Rackspace, …
  - **Tenants**: users renting virtual machines

- Private cloud datacenters

A use-case of cloud datacenters



Requirements for network sharing

[FairCloud]

Tenants want predictable performance / cost

→ Req 1. **Minimum bandwidth guarantee**

Not all flows are equal: some tenants pay more

→ Req 2. **Proportionality**

Utilize spare resources as much as possible

→ Req 3. **High utilization**

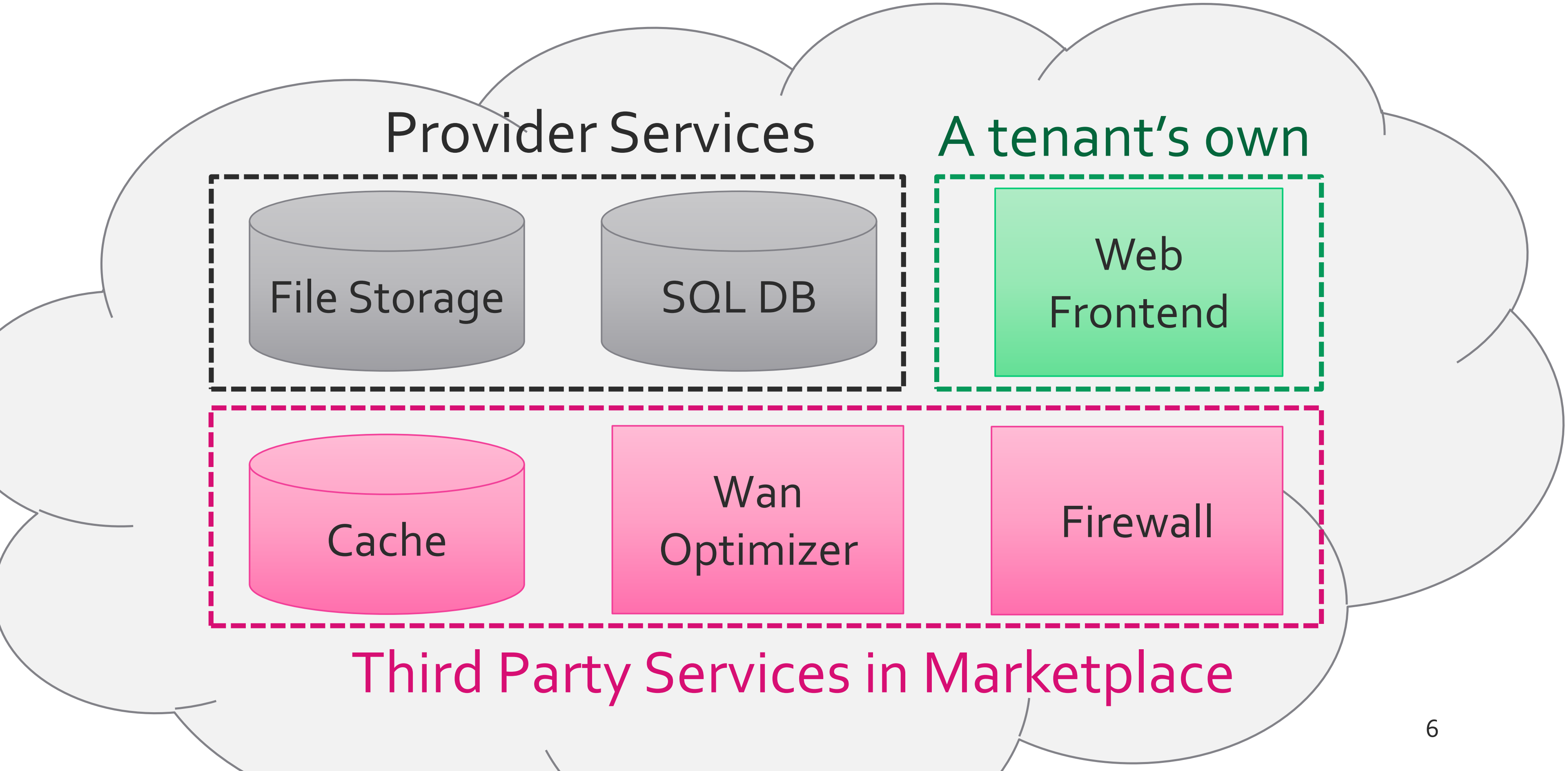
Existing solutions for network sharing

	Today
Min-guarantee	⊘
Proportionality	⊘
High utilization	✓

Prior work focuses on **intra tenant traffic**

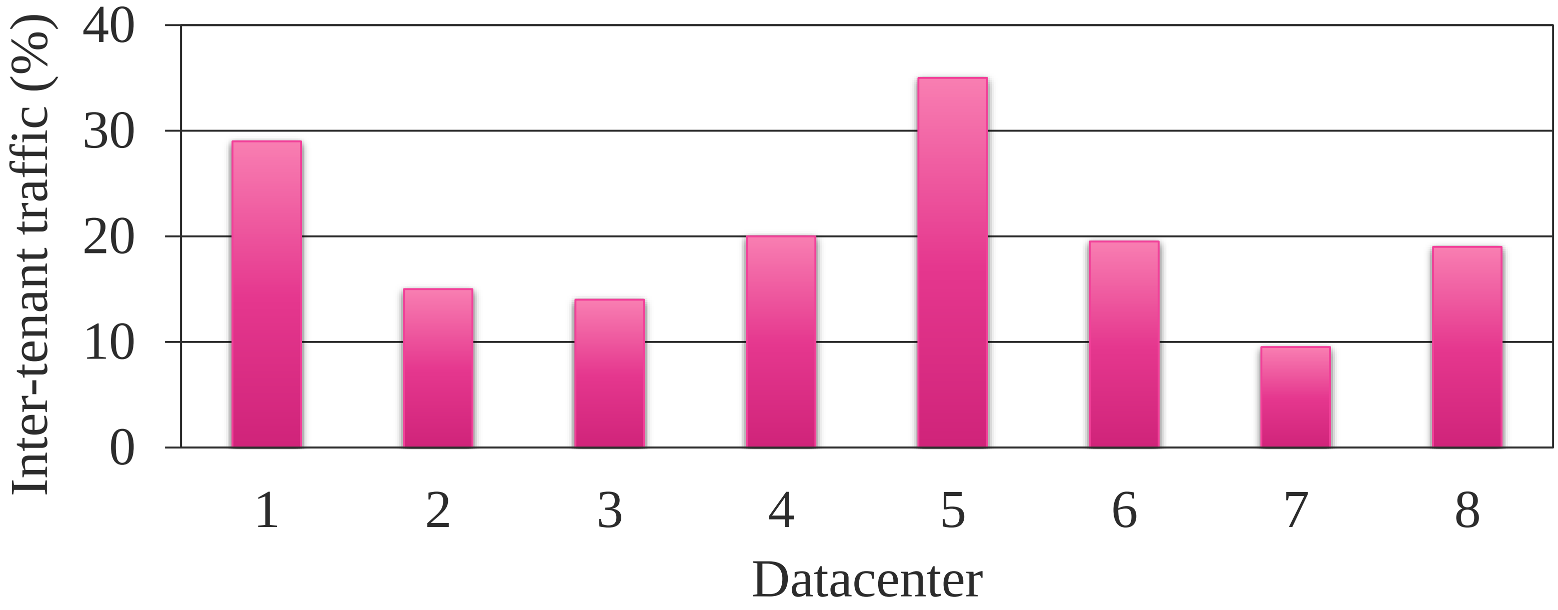
Chatty tenants in the cloud

Typical cloud applications have many dependency



Prevalence of inter-tenant traffic

Measurement from 8 datacenters of a public cloud service provider



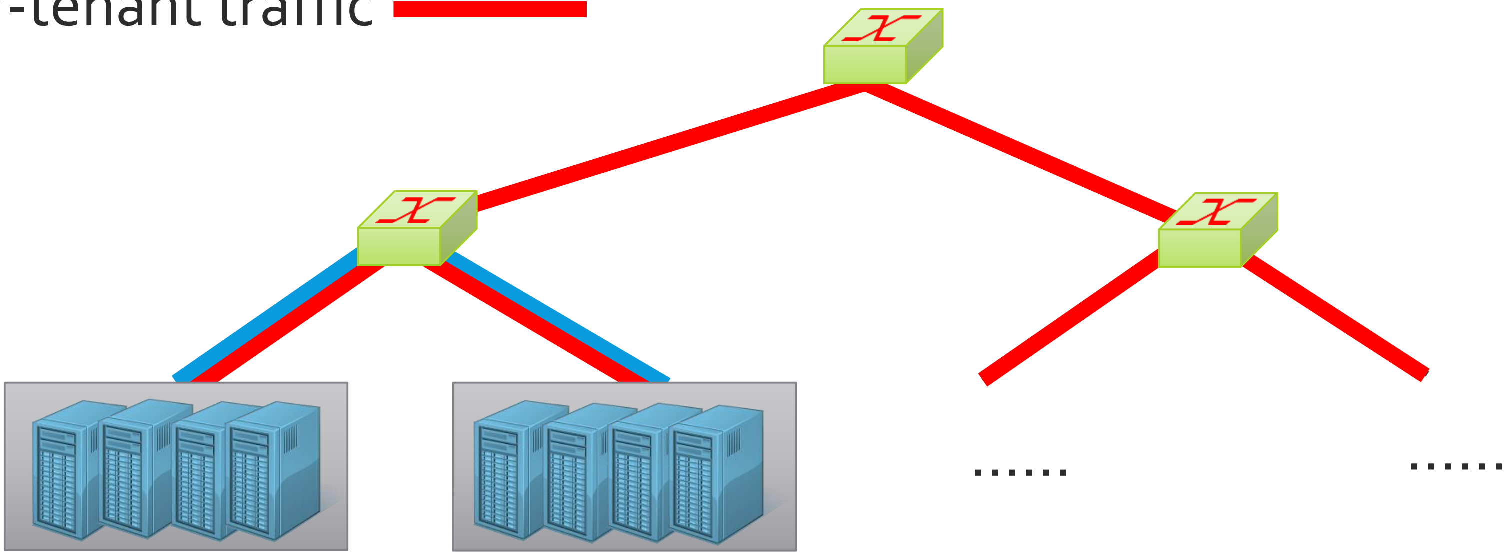
Inter-tenant traffic accounts for **10-35%** of traffic!

Min bandwidth guarantee is harder

On what links bandwidth guarantee is required?

Intra-tenant traffic

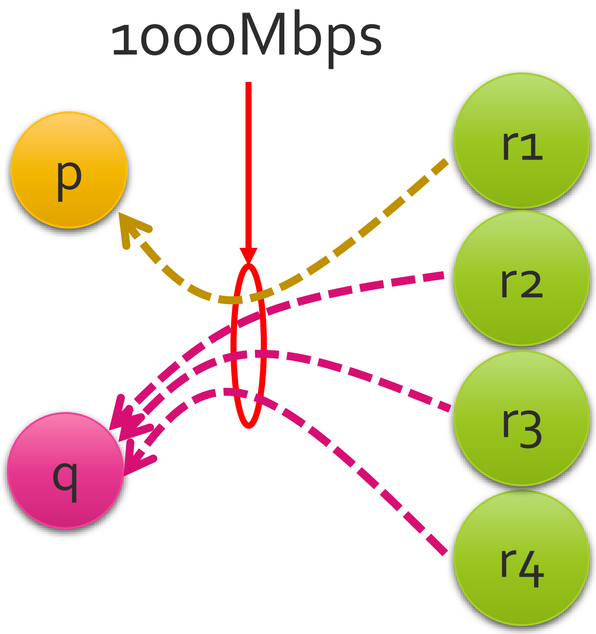
Inter-tenant traffic



Inter-tenant traffic leads to richer communication pattern and makes minimum bandwidth guarantee harder!

How to define proportionality?

P and Q are paying same amount



Allocation	P (Mbps)	Q (Mbps)
Per flow	250	750
Seawall	250	750
FairCloud	333	666

Q: Whose payment should dictate the flow bandwidth?

Hadrian Overview

- What semantics should we provide to tenants?
  - Virtual network abstraction
- How to allocate bandwidth?
  - Hose-compliant bandwidth allocation

} This talk

- How to place virtual machines?
  - Greedy heuristic that guarantees min bandwidth

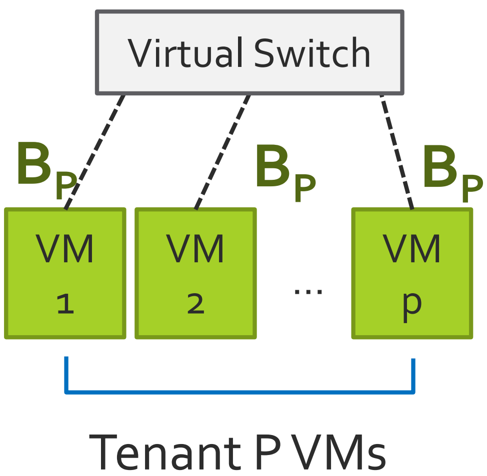
Hadrian Overview

- What semantics should we provide to tenants?
  - Virtual network abstraction
- How to allocate bandwidth?
  - Hose-compliant bandwidth allocation
- How to place virtual machines?
  - Greedy heuristic that guarantees min bandwidth

State of the art: Hose-model

Tenant Request: <N, B>

Each VM is guaranteed to send/receive at B bps *minimum* of B bps

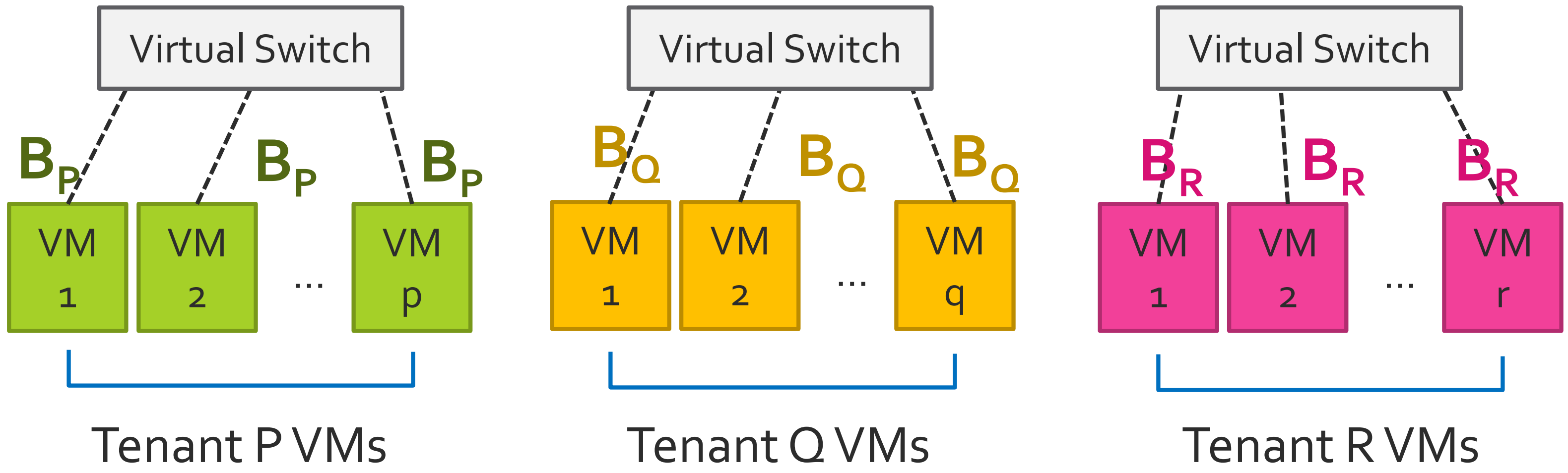


- ✓ Minimum bandwidth guarantee
- ✓ High-utilization

State of the art: Hose-model

Tenant Request: <N, B>

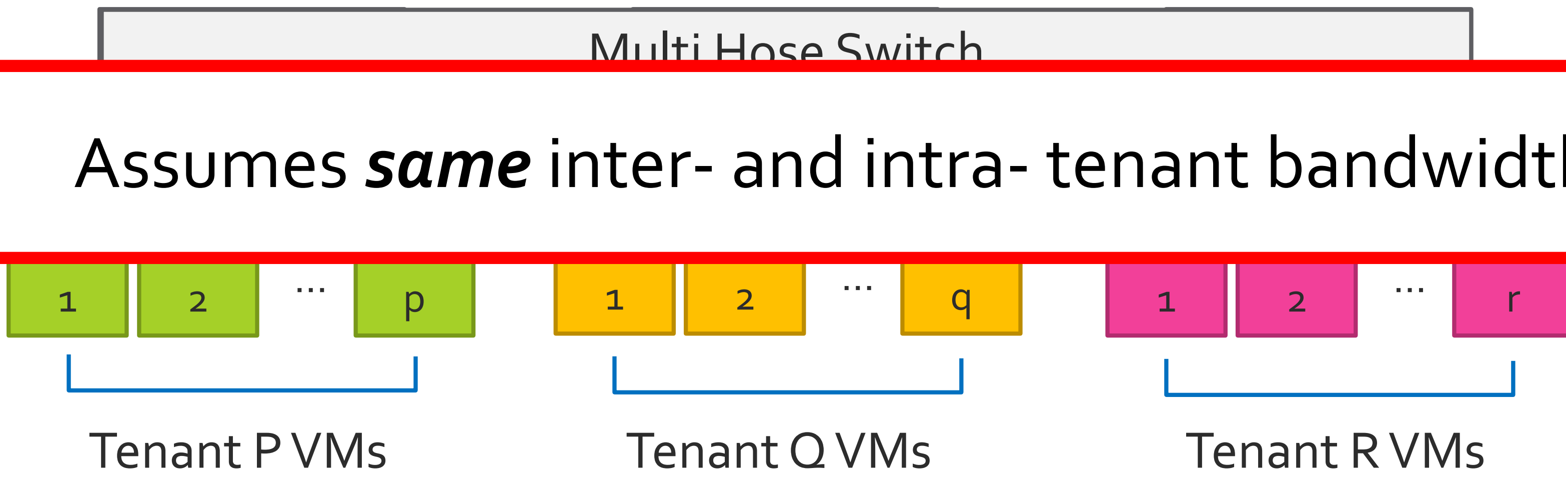
Each VM is guaranteed to send/receive at ***minimum*** of B bps



Multi-hose model

Tenant Request: <N, B>

VMs in different tenants communicates with each other at a rate of *min(Bp, Bq)*

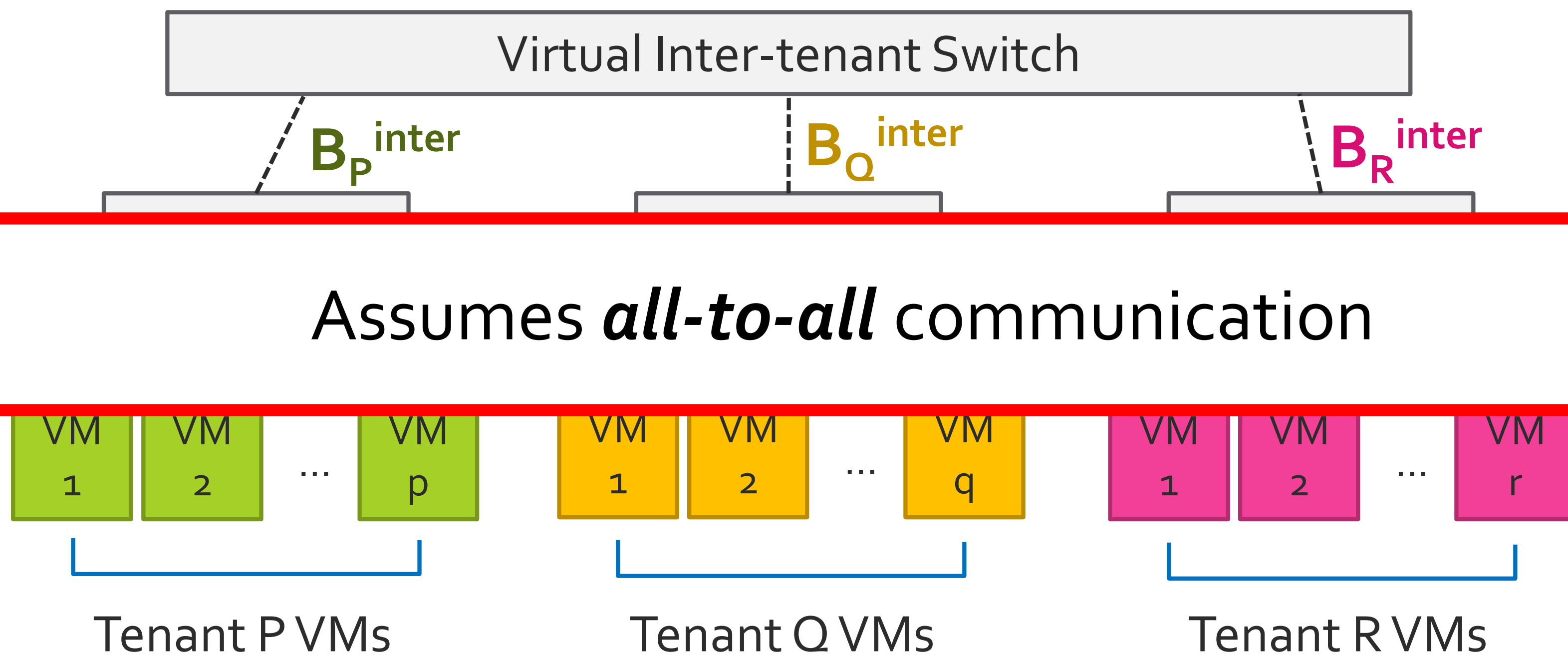


Assumes ***same*** inter- and intra- tenant bandwidth

✓ Allows Inter-tenant communication

Hierarchical hose model

Tenant Request: <N, B, B^{inter}>



Assumes ***all-to-all*** communication

✓ Separate inter-tenant bandwidth requirement

Communication dependency

Most tenants communicate with only few other tenants

Tenant Request: <N, B, B^{inter}, list of *dependent tenants*>

→ Reduces possible communication patterns

→ Helps place dependent tenants closer

Q:How about service tenants (e.g., storage)?

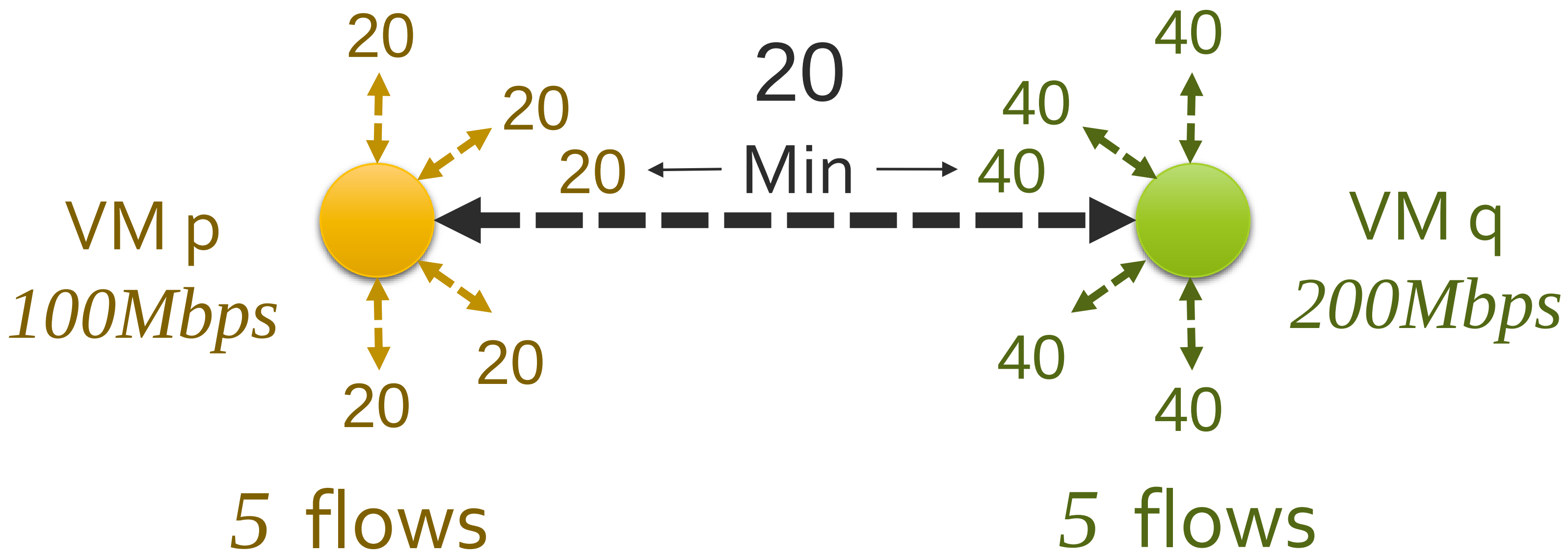
Tenant Request: <N, B, B^{inter}, *>

Hadrian Overview

- What semantics should we provide to tenants?
  - Virtual network abstraction
- How to allocate bandwidth?
  - Hose-compliant bandwidth allocation
- How to place virtual machines?
  - Greedy heuristic that guarantees min bandwidth

Hose-compliant bandwidth allocation

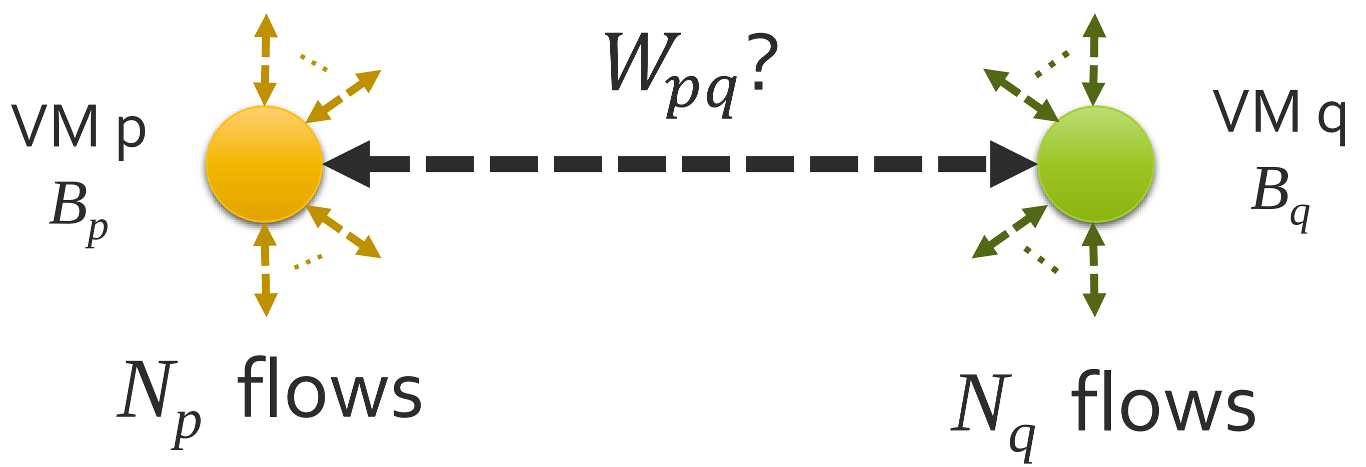
Whose payment should dictate the bandwidth of the flow?



Our approach : take ***minimum*** from two sides

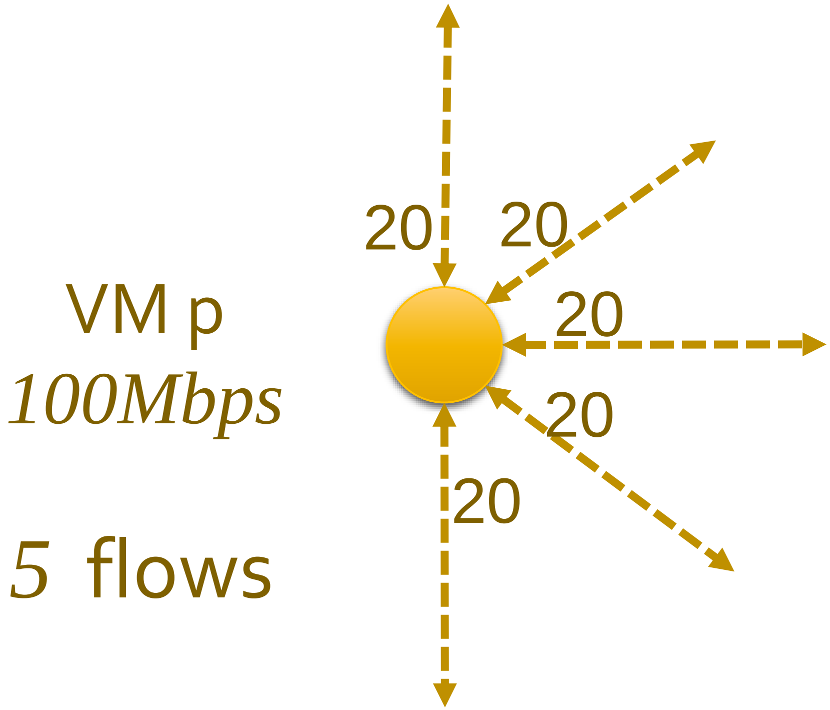
Hose-compliant bandwidth allocation

Weighted fair-share at the bottleneck



$$W_{pq} = \color{red}{min}\left(\frac{B_p}{N_p}, \frac{B_q}{N_q}\right)$$

Upper bound proportionality



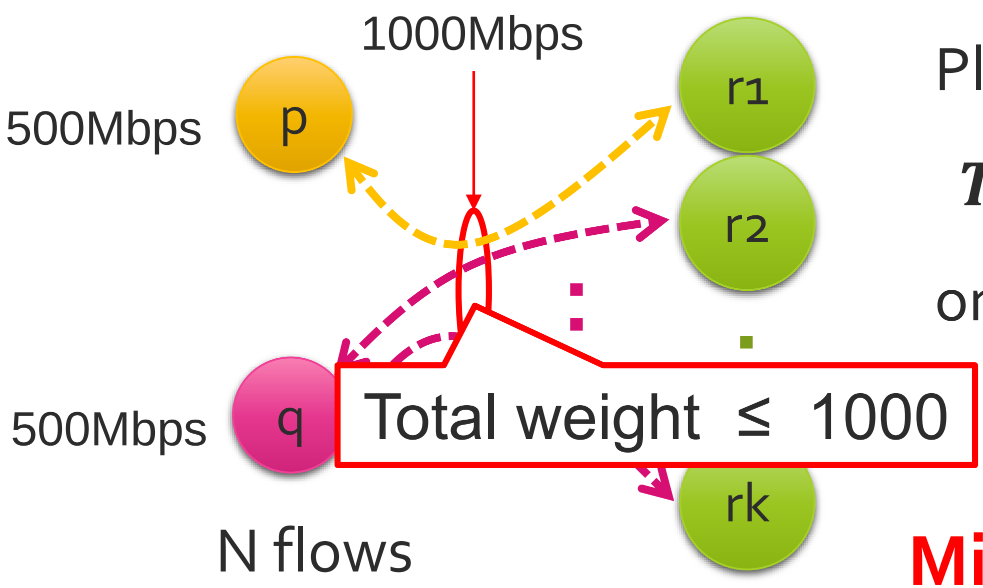
Min (20, ?) x 5 ≤ 100

Max aggregate weight for p's flow = 100

Upper bound of total weight of VM's flows is proportional to the VM's payment

Minimum bandwidth guarantee

Total weight for all flows of a given VM is **bounded**



Placement algorithm enforces

$$\boldsymbol{Total\ Weight\ \leq\ Capacity}$$

on any link in the network

↓

Min bandwidth guarantee

The verification can be formulated as max flow network problem

Evaluation

Synthetic cloud workload benchmark

- Tenants submit requests for VMs and execute jobs
- A job has

  CPU Processing, Inter-tenant traffic**, **Intra-tenant traffic
- Inter-tenant traffic ratio: 10 - 40%
- Fraction of tenant w/ inter-tenant : 20%

Environments

- **Testbed**: 16 end hosts
- **Large scale simulation**: 16,000 end hosts

Evaluation criteria

Network sharing requirements

- Minimum bandwidth guarantee
- Upper-bound proportionality
- High-utilization

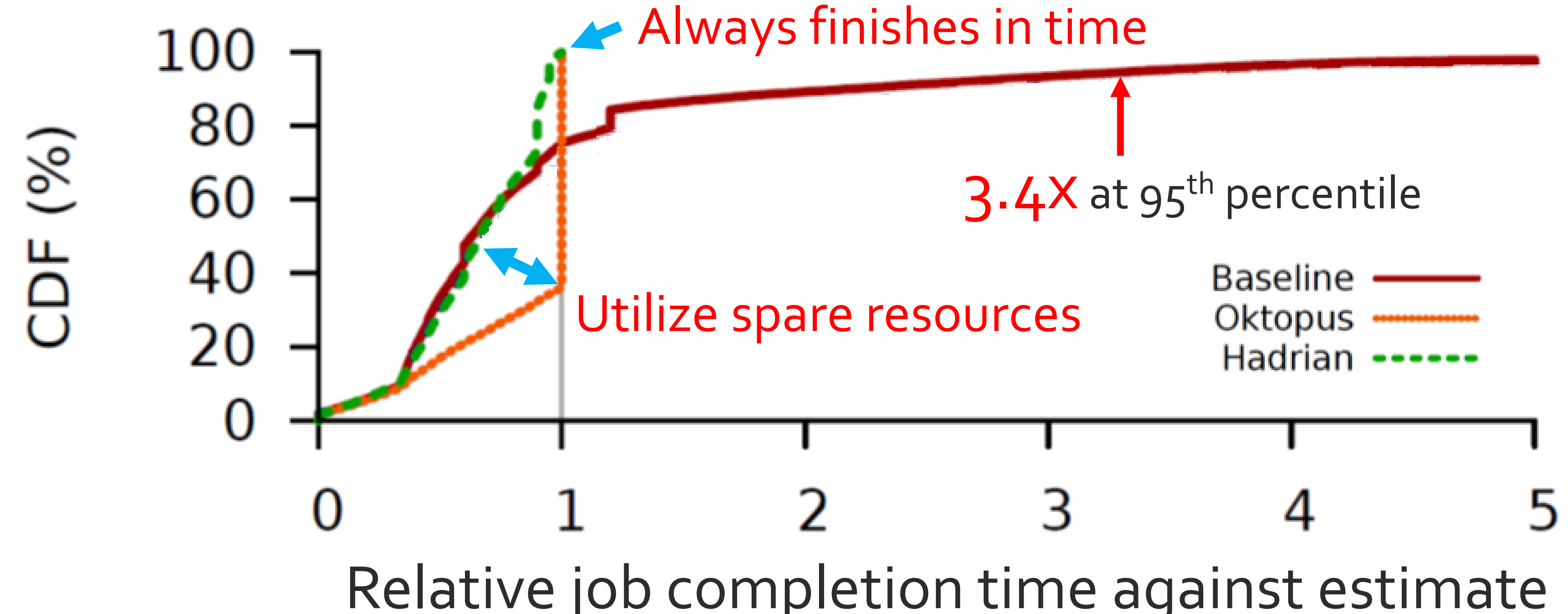
Benefits of *Hadrian*

- Metric: acceptance ratio

Comparison with

- Baseline: per flow sharing
- Existing approaches: Oktopus , FairCloud

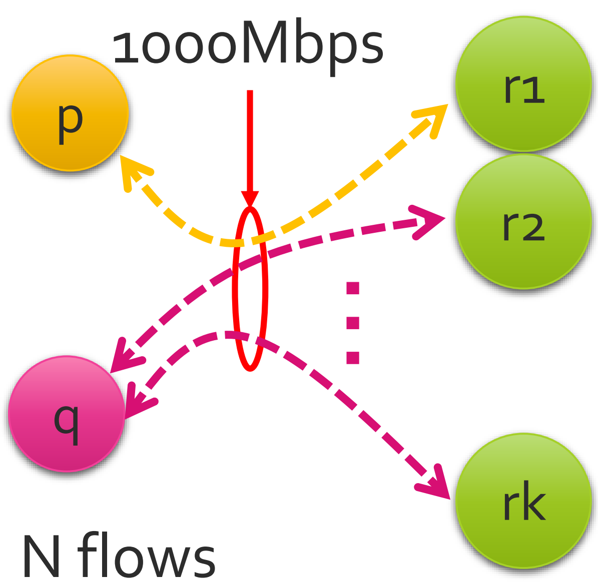
Job completion time

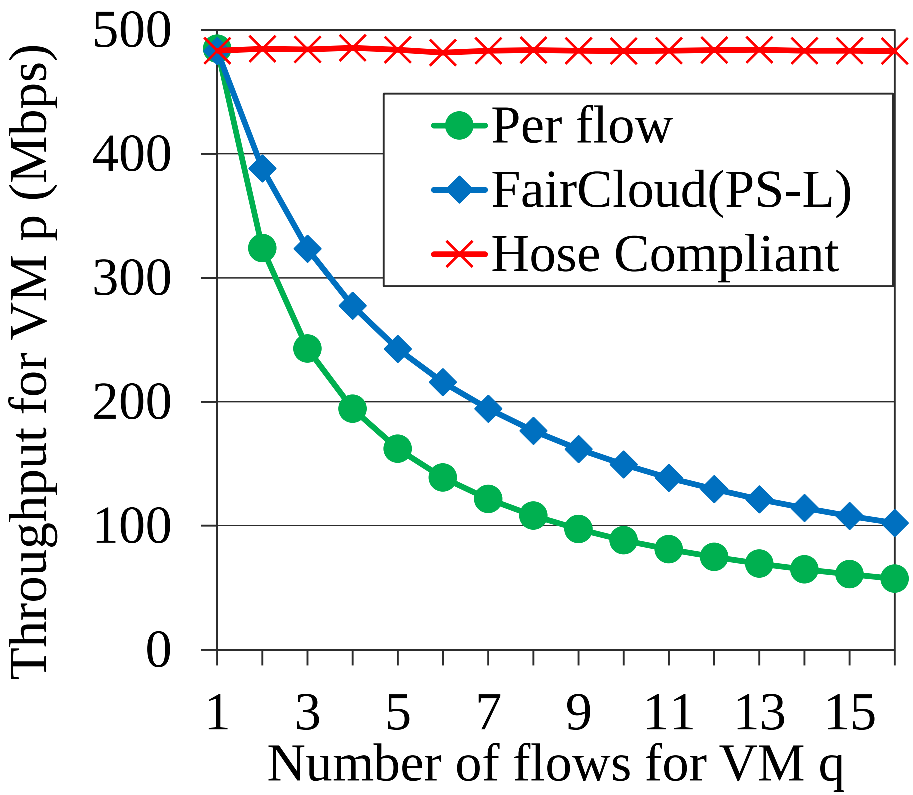


Estima uirement

- ✓ Minimum bandwidth guarantee
- ✓ High utilization

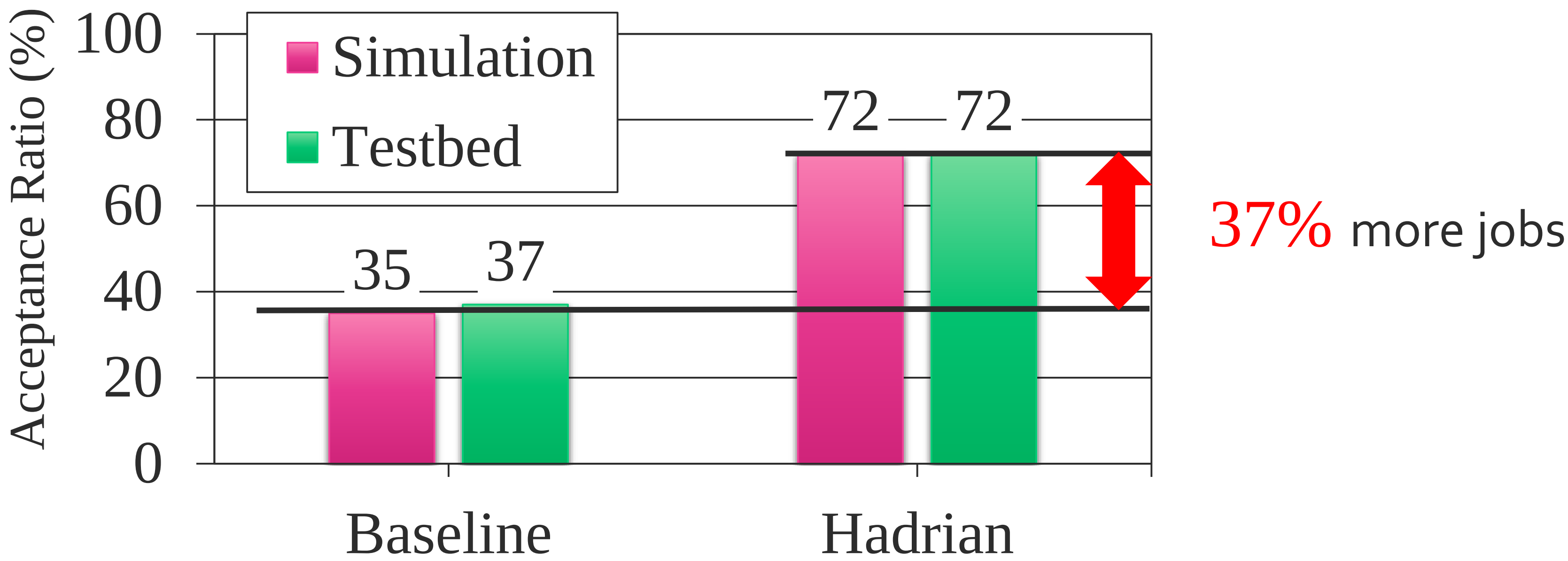
Bandwidth allocation





✓ Upper bound proportionality

Request acceptance ratio – testbed



Similar benefits in large scale simulations
Testbed and Simulation shows ***consistent*** result

Summary

We show that Inter-tenant traffic is prevalent

- 10~35% from a major public cloud provider

We propose **Hadrian**

- *Virtual network abstraction*: inter-tenant, dependency
- *Bandwidth allocation strategy*: upper-bound proportionality
- *Placement algorithm*: greedy dependency aware packing

Our evaluation show that

- Hadrian meets three network sharing requirements
- Hadrian delivers **predictability** and higher **efficiency**

Thank you