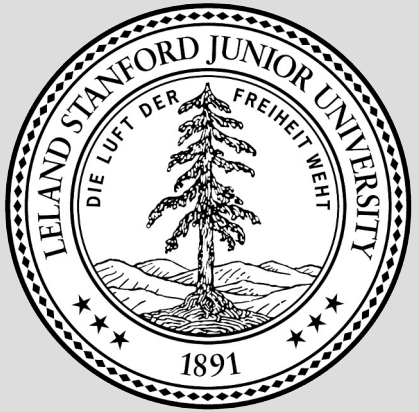
CS244 Lecture 6

Datacenter Networking

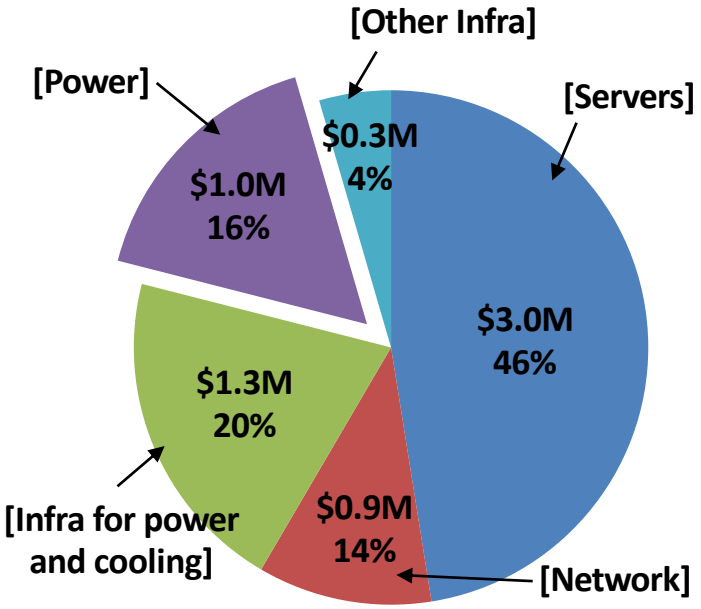
Chang Kim



VL2: A Scalable and Flexible Data Center Network
EyeQ: Practical Network Performance Isolation at the Edge

Key Tenet of Data Centers: Agility at Scale

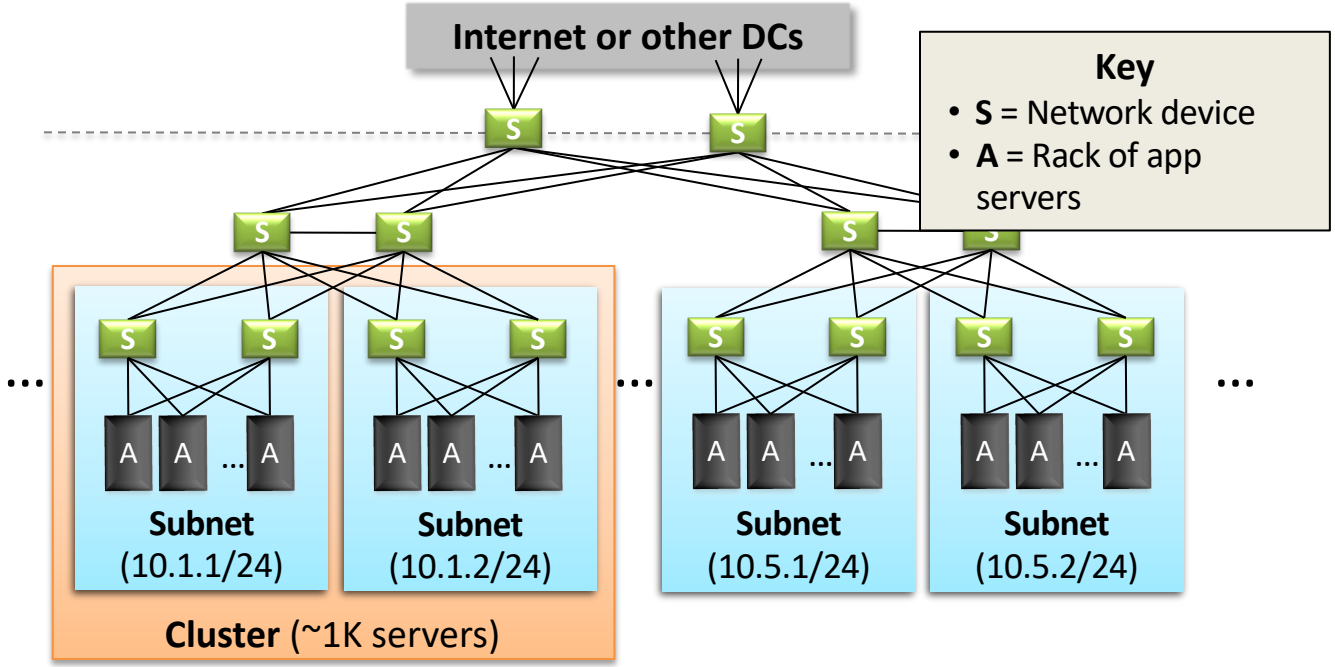
- DCs are digital-era factories, requiring huge up-front investment
- Golden rule: ***Maximize the amount of useful work per dollar spent***
- Best operating principles: **Multi-tenancy** & **Dynamic resizing**



Monthly bill for a 50,000-server DC

Agility **– Capability to assign any servers to any tenants any time**

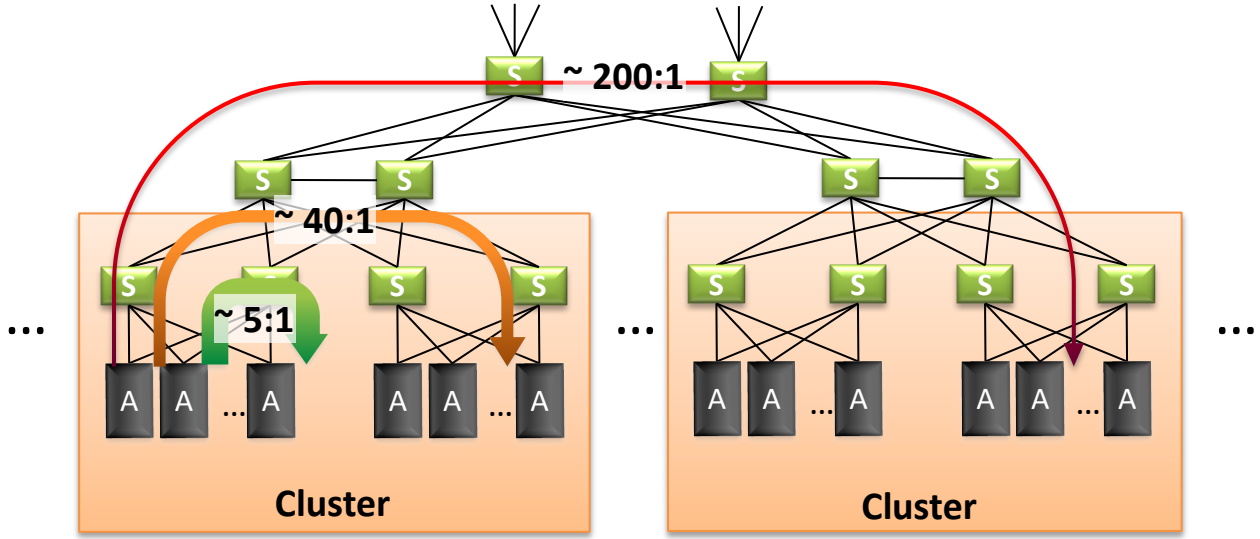
Status Quo Ante: DC Networks circa ~2010



Reference – "Data Center: Load-balancing Data Center Services", Cisco 2004

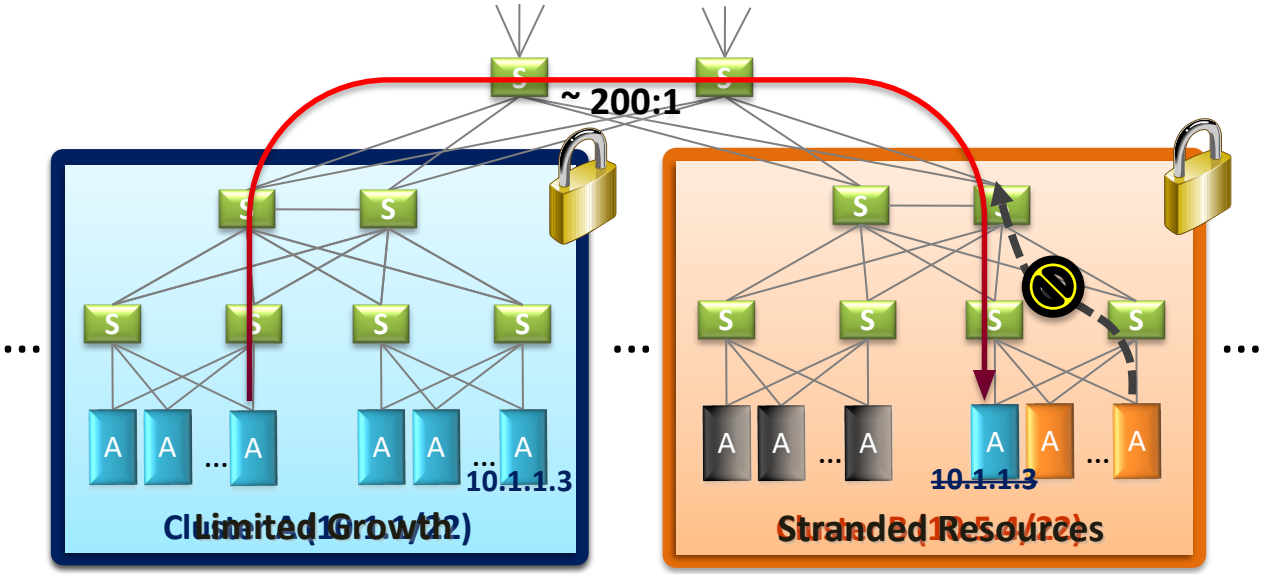
Designed mainly for pre-cloud web-hosting services

Modern Workload On Yesterday's Network



- Depends on high-cost mainframe-style network devices
- Extremely limited server-to-server capacity
- Highly-distributed apps suffer from poor capacity

Agility Was Very Hard To Achieve



- Cause waste of resources, lowering DC utilization

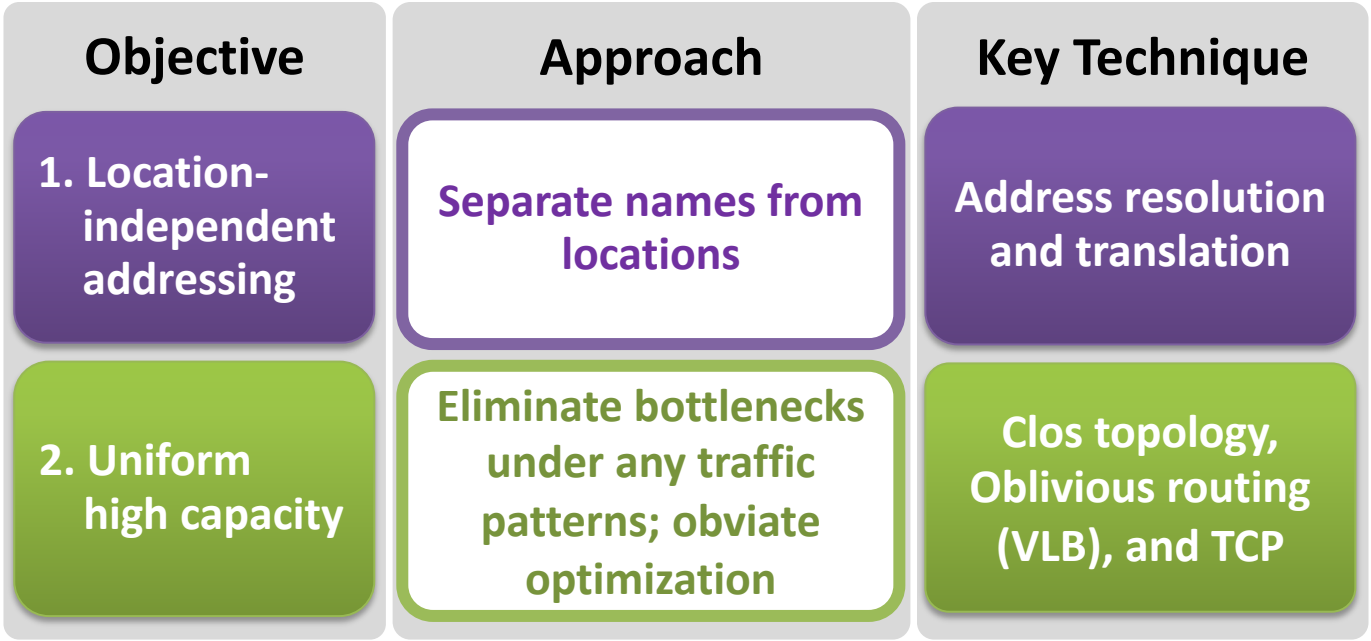
What The Authors of VL2 Desired Concretely

- To network, **“*Support for* Agility”** means

Help tenants stop caring about the placement of their servers

- ✓ **Assign any IP addresses to any servers**
- ✓ **Offer Consistently high networking performance between any servers**
- ✓ **Protect tenants from one another**

Key Objectives and Techniques of VL2



Achieve both even when tenants do *not* cooperate with or trust one another

Part I: Any Address to Any Servers

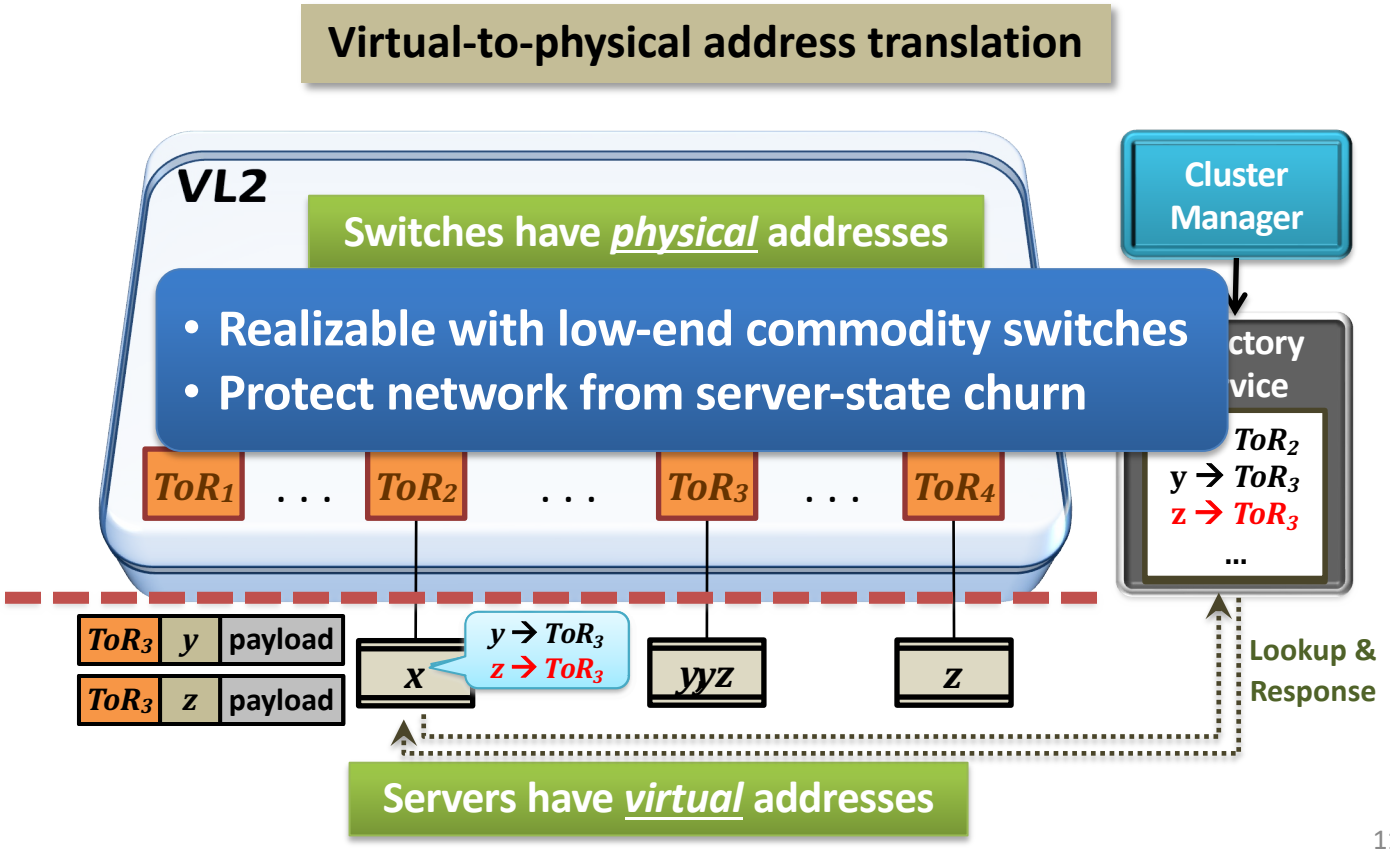
- **Flat addressing**
- **Bring your own address space**
- **Reachability isolation**
- Uniform high capacity
- Performance isolation

Challenges and Opportunities

- Challenge
  - Huge amount of **server state** and **churn** to it
- Opportunity
  - **Cluster manager** premeditates and coordinate any server-state changes
  - **Eventual consistency** is fine

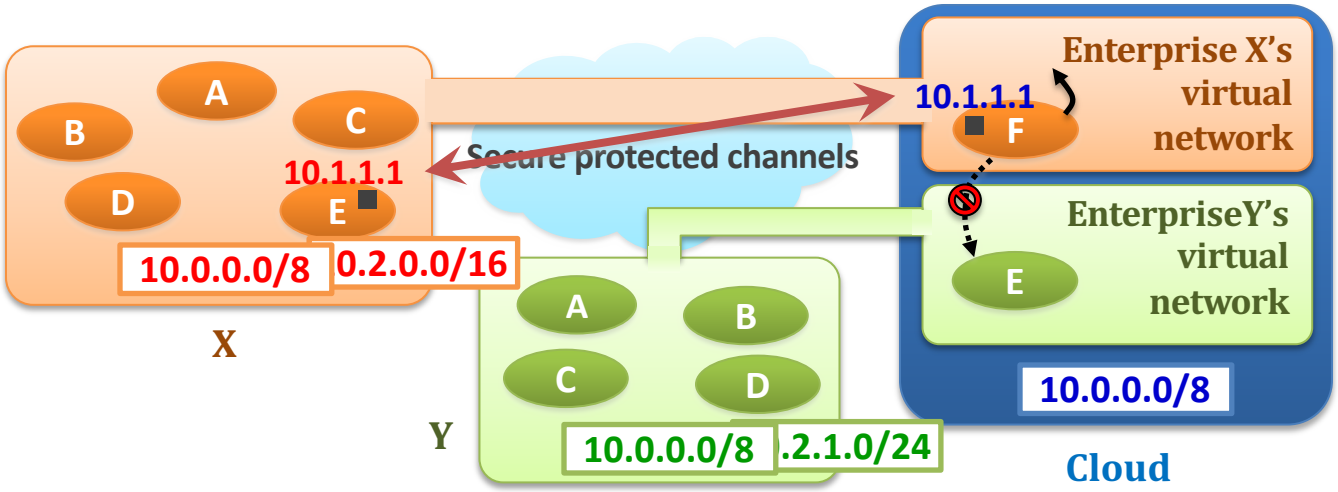
Huge amount of server-state and churn might be manageable

Flat Addressing:
Virtual Memory Technology for Network



Cloud DC Needs More Than Flat Addressing

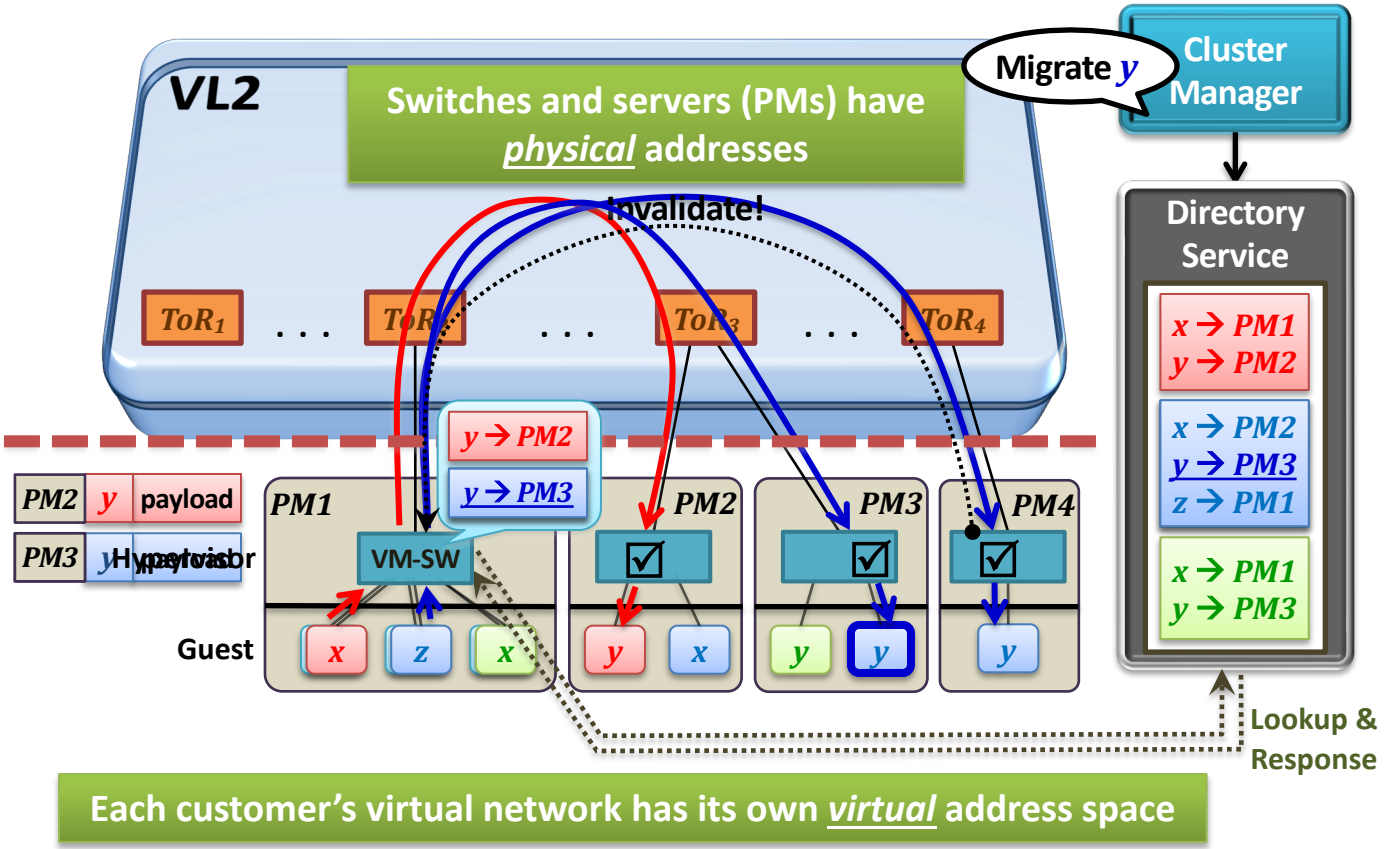
- Partially cloud-based service deployment
- Corporate sites in cloud



BYO Address Space

Reachability Isolation

BYOAS and Reachability Isolation



Part II: Predictable and Uniform High Capacity

- Flat addressing
- Bring your own address space
- Reachability isolation
- **Uniform high capacity**
- **Performance isolation**

Challenges

- Instrumented a large data-mining cluster and derived distinctive traffic patterns
- **Traffic patterns are highly volatile**
  - A large number of distinctive patterns even in a day
- **Traffic patterns are unpredictable**
  - Correlation between patterns very weak

If you are to optimize routing to avoid hot spots, you should do that very frequently and rapidly

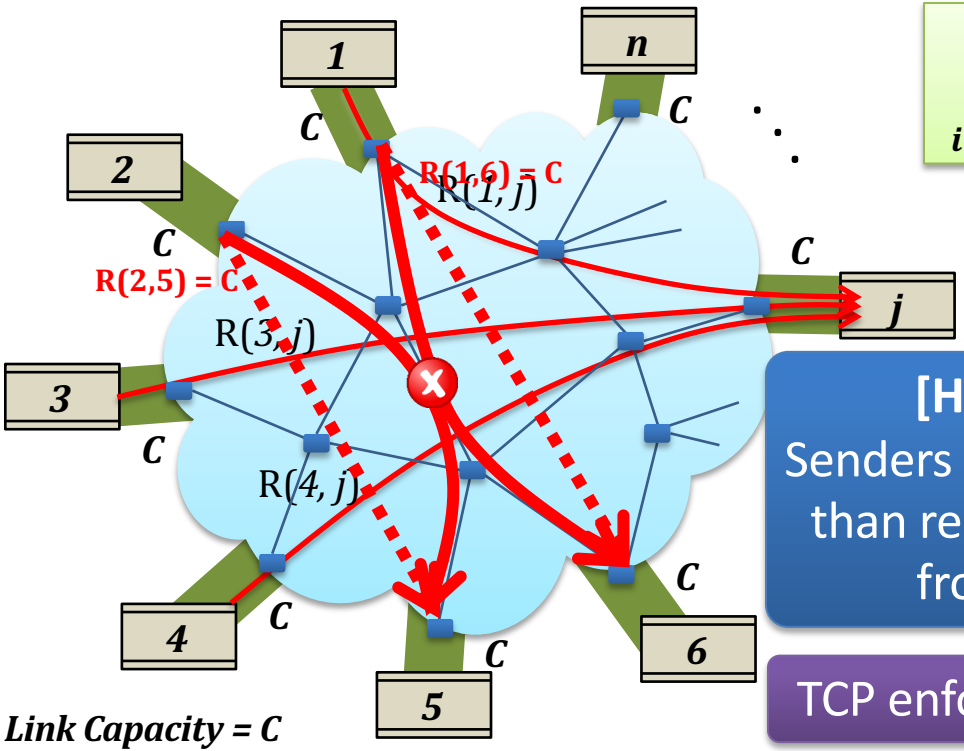
Opportunities

- **Very few elephant flows**
  - Traffic flows are numerous and not huge
  - Agree with observations in other DC-measurement studies [Kandula et al., IMC'09 & Benson et al., IMC'10]
- **Links substantially thicker than max-sized flows**
  - Maximum network I/O capacity of a single CPU core is limited to 2 ~ 3 Gbps

Simple probabilistic traffic spreading might work well enough

Hose Model: The Most Lenient Traffic Model That is Admissible

R(i, j) = Node i's transmission rate to j



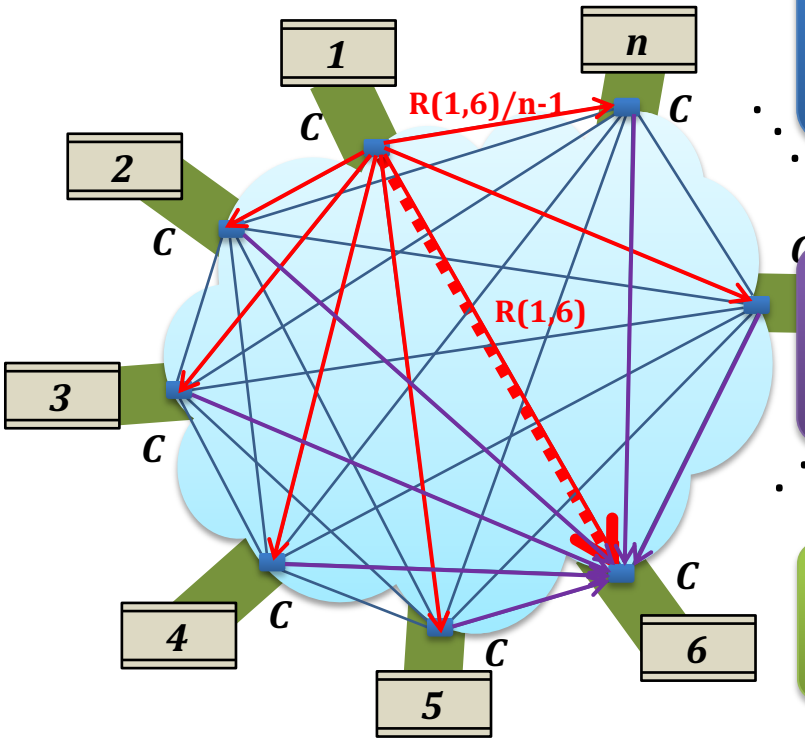
$$\sum_{i=1\,..\,n} R(i,j) \le C$$

[Hose Model]
Senders do *not* send more than receivers can draw from network

TCP enforces hose model

Hose Model and VLB*

R(i, j) = Node i's transmission rate to j



[Hose Model]
Senders do *not* send more than receivers can draw from network

+

[VLB]
Detouring on full mesh & Even traffic spreading

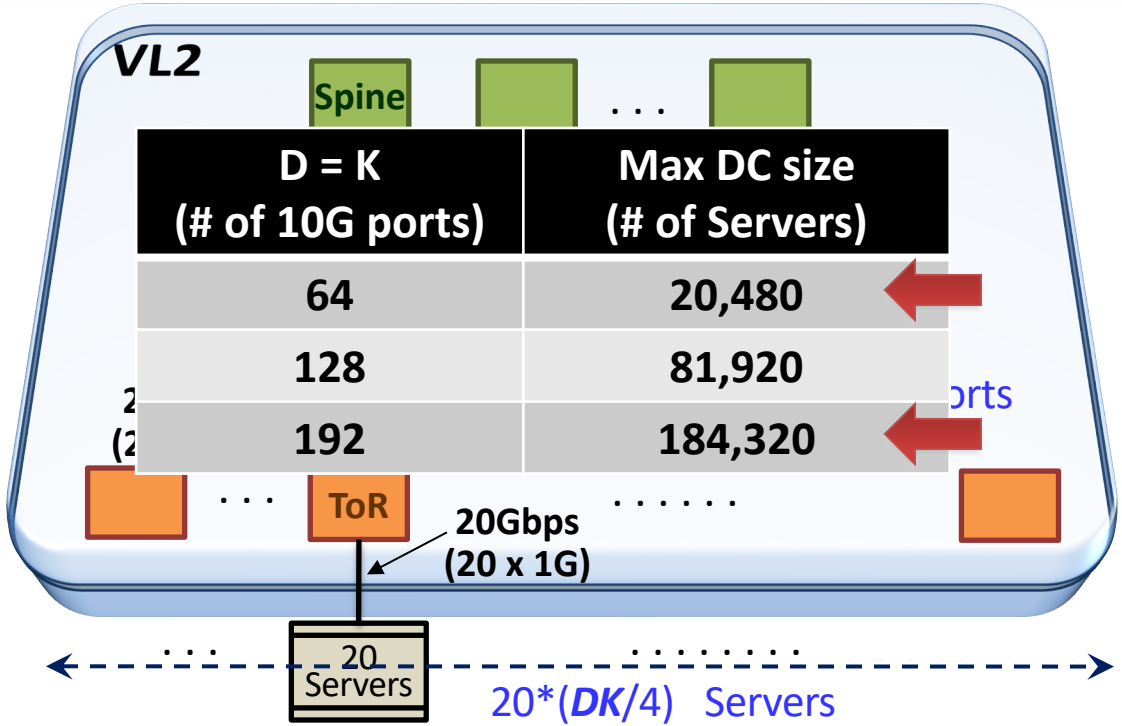
=

Free of bottleneck with minimal link capacity (2C/n)

* L. Valiant, "A scheme for fast parallel communication," *SIAM J. on Comp.,*

The VL2 Topology: Adaptation of Clos Network*

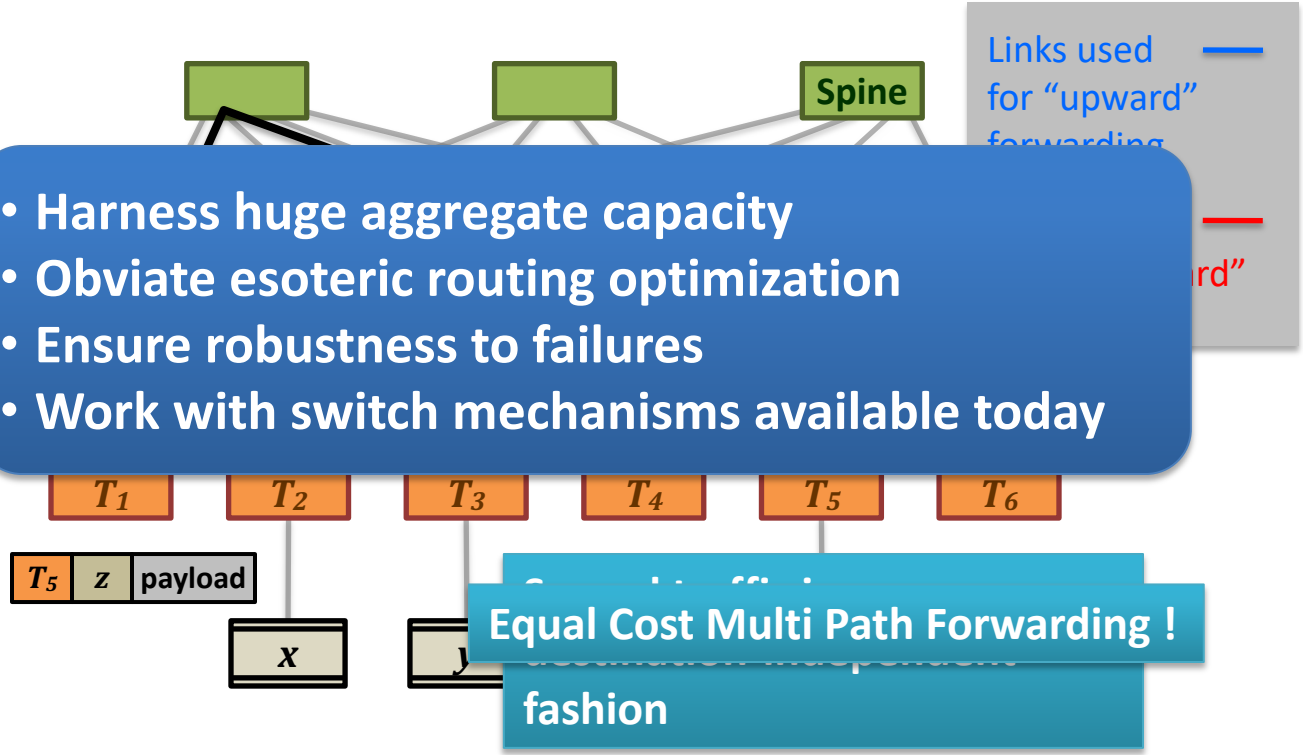
Ensure huge aggregate capacity and robustness at low cost



* C. Clos, "A Study of Non-blocking Switching Networks," *Bell Sys. Tech. J.*, 1953

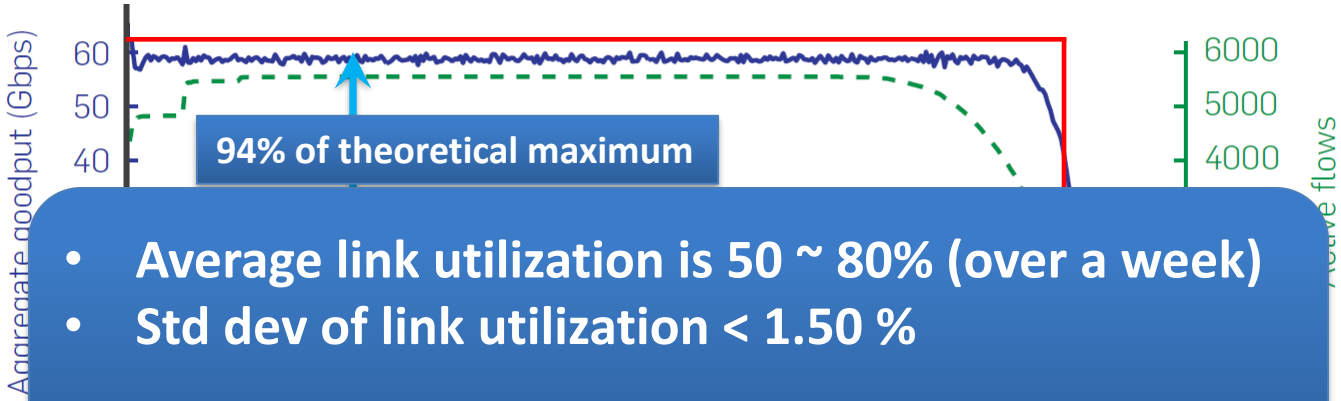
Practical VLB: Folded Clos + ECMP

Uniform high bandwidth under arbitrary traffic patterns



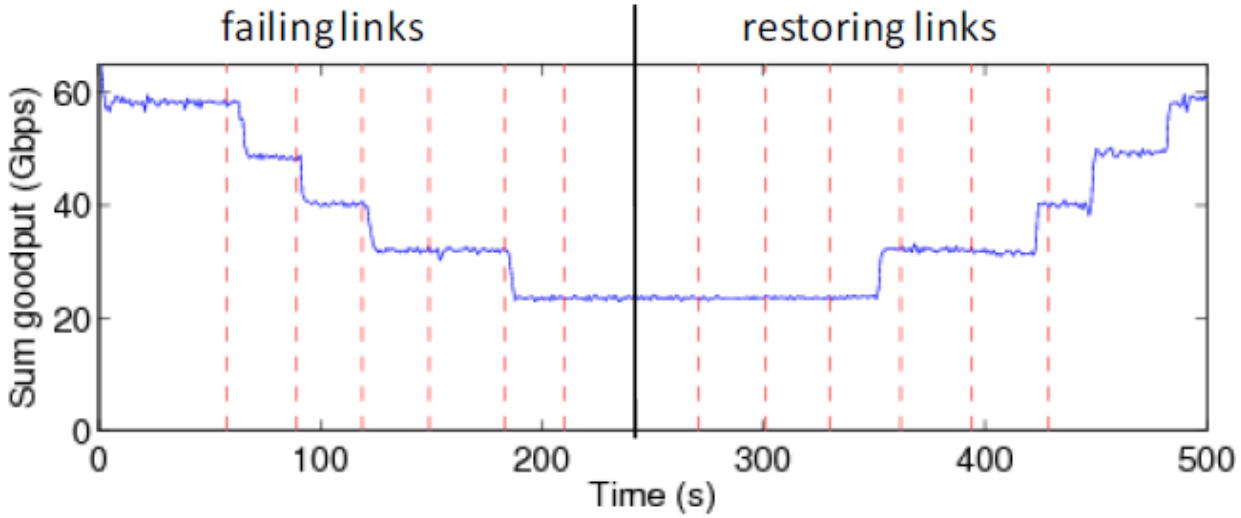
Reality Check

- How well does the theory of VLB hold in practice?



- Cause of sub-optimality
  - **Random flow spreading** (vs. Round-robin byte spreading)
  - TCP congestion-control dynamics
  - Retransmission

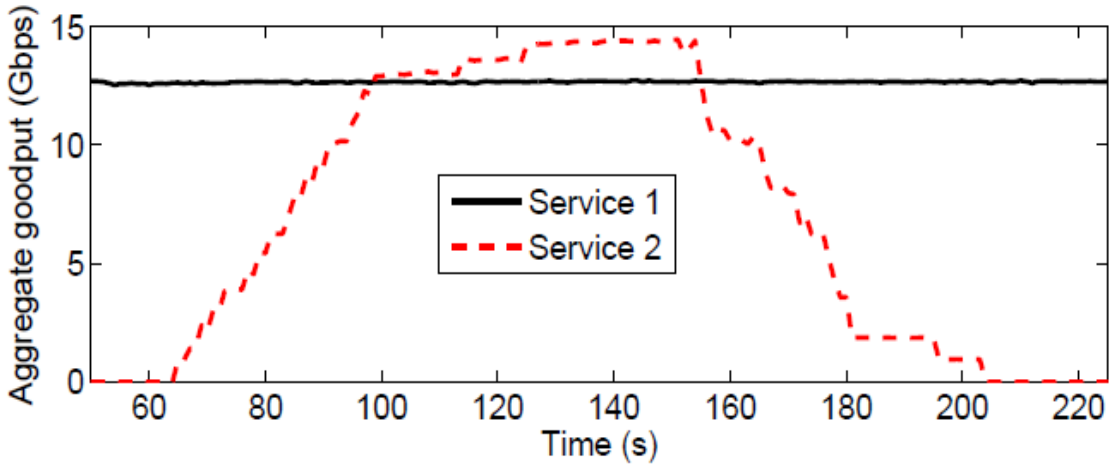
Resilience to Failures



Performance degrades and recovers gracefully as links are failed and restored

Does VL2 Ensure Performance Isolation?

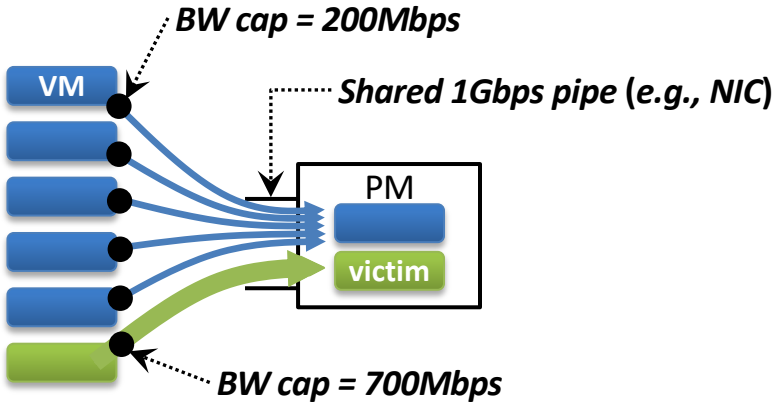
- In theory TCP is not perfect at enforcing hose model
  - Adjusting sending rate takes a few RTTs
- Is TCP "fast enough" in practice?



So, Are we all done here?

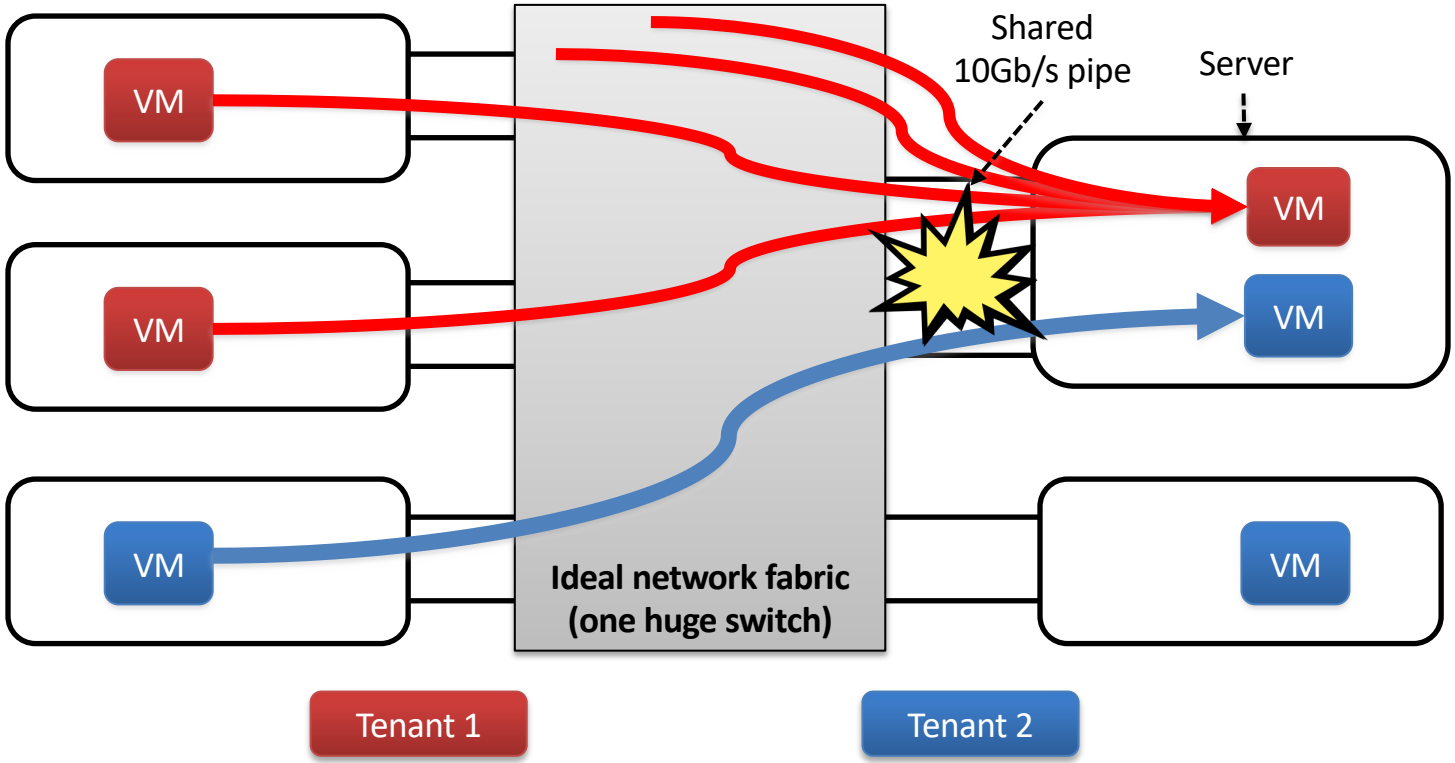
Why is VL2's performance isolation insufficient?

- TCP isn't helpful in cloud
  - Provider can't force customers to use only TCP
  - Provider can't trust networking stack in VM
- Connection-level fairness is irrelevant
- Static rate-limiting doesn't help



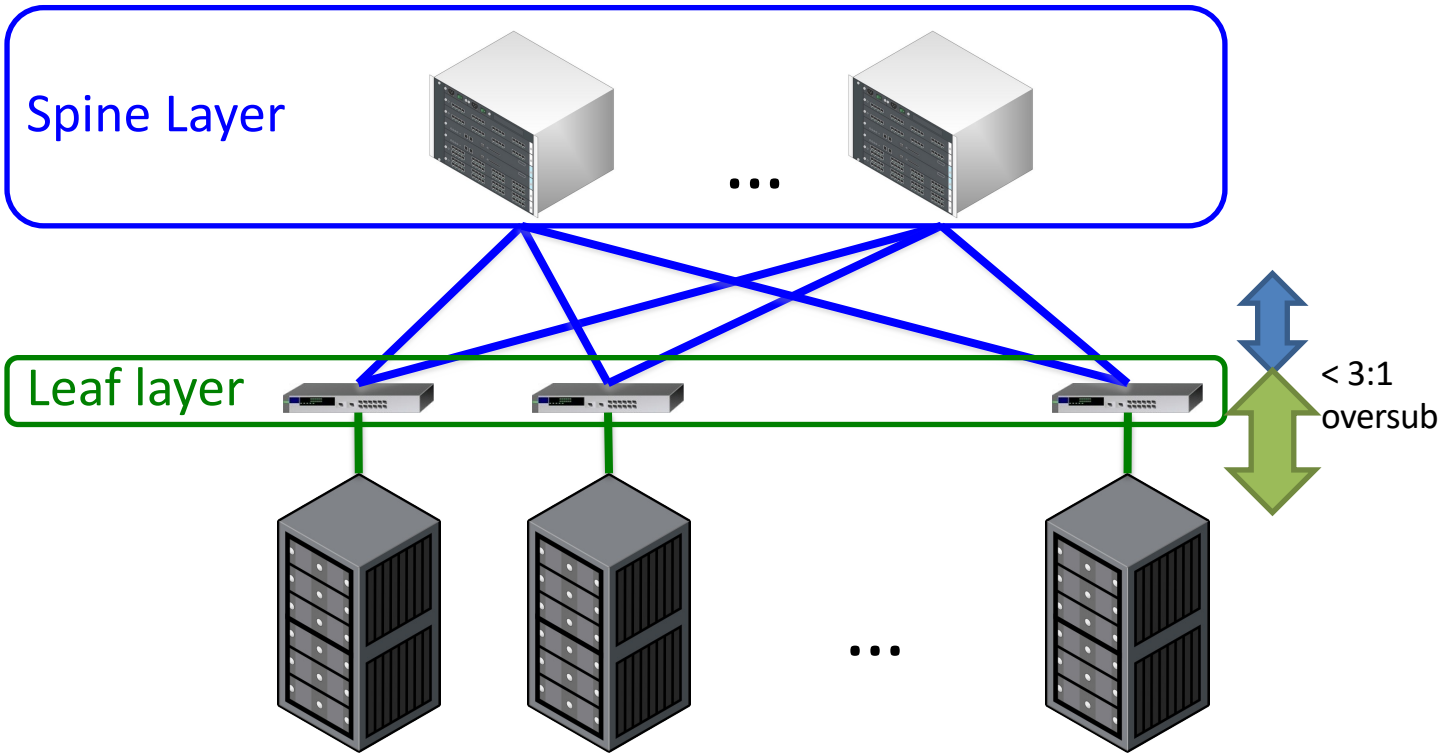
Existing Solutions Fall Short

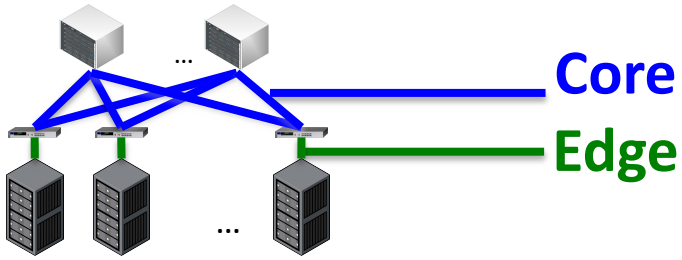
- Even a full-bisection BW network isn't enough



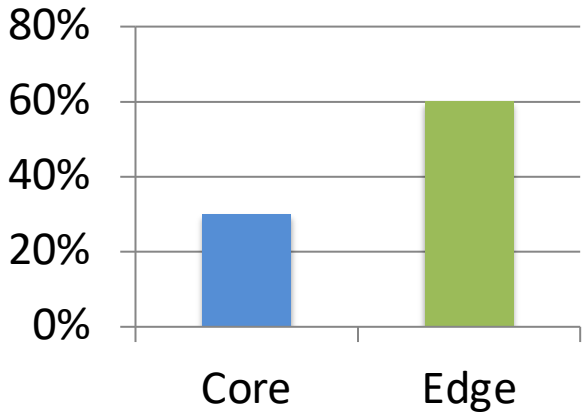

FOR
SALE
Brown.

Where Does Congestion Happen? Measurement Study on Microsoft Azure





99.9th percentile utilization (%)



Timescales: over 2 weeks, 99.9th percentile = several minutes

Hottest storage cluster: 1000x more drops at the Edge, than Core.

16 of 17 clusters: 0 drops in the Core.

So, What Do We Desire?

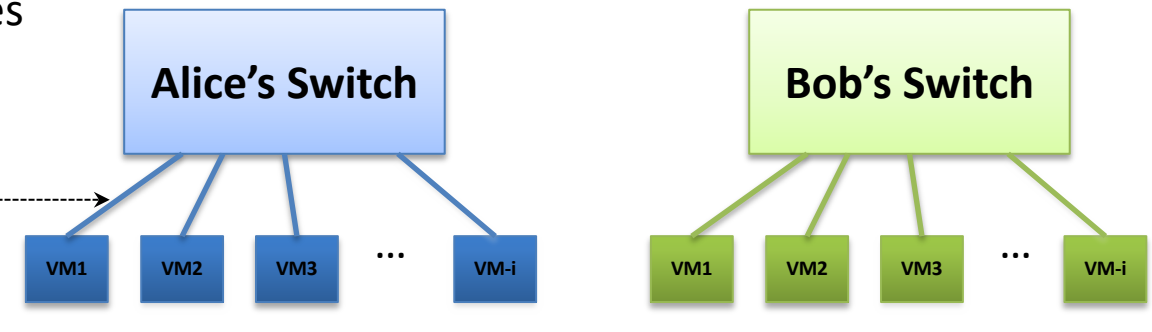
- Lenient to customers
- Secure
- Capable of dealing with micro contention
- Work-conserving and efficient
- Scalable
- Work with off-the-shelf network devices

[Problem Formulation]

- Each VM is given a certain virtual NIC capacity
- Given a physical NIC serving multiple VMs, ensure fair allotment of the physical NIC capacity for every VM

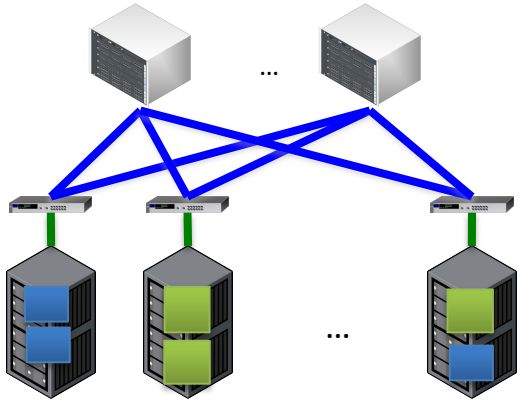
EyeQ: Predictable Bandwidth Partitioning at the Edge

Customer specifies capacity of the virtual NIC.
No traffic matrix. (Hose Model)



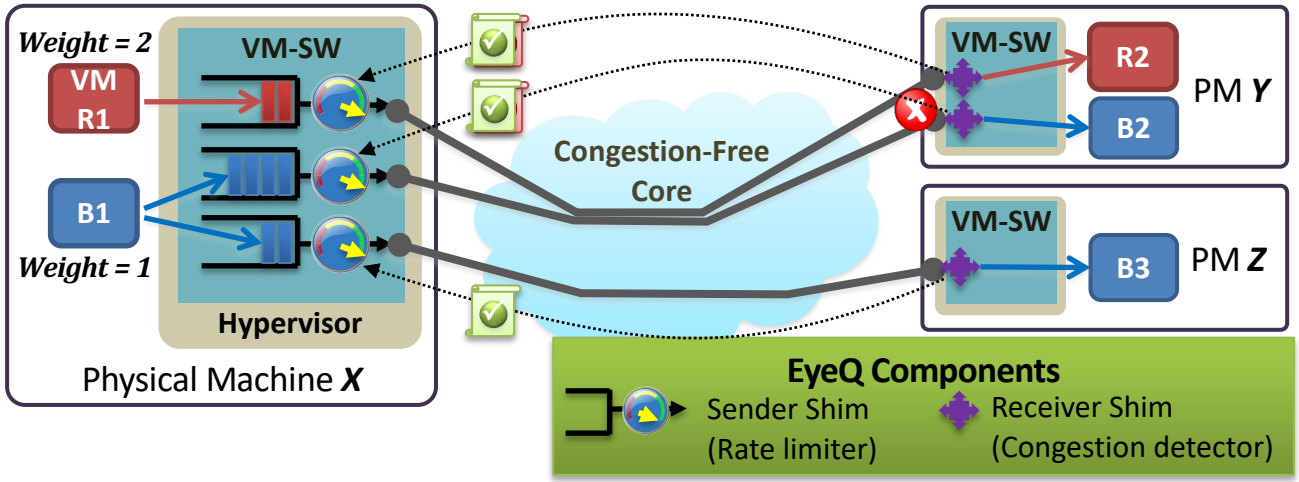
Provider: assures near dedicated performance.

Deployable ***today*** in a cloud datacenter network.



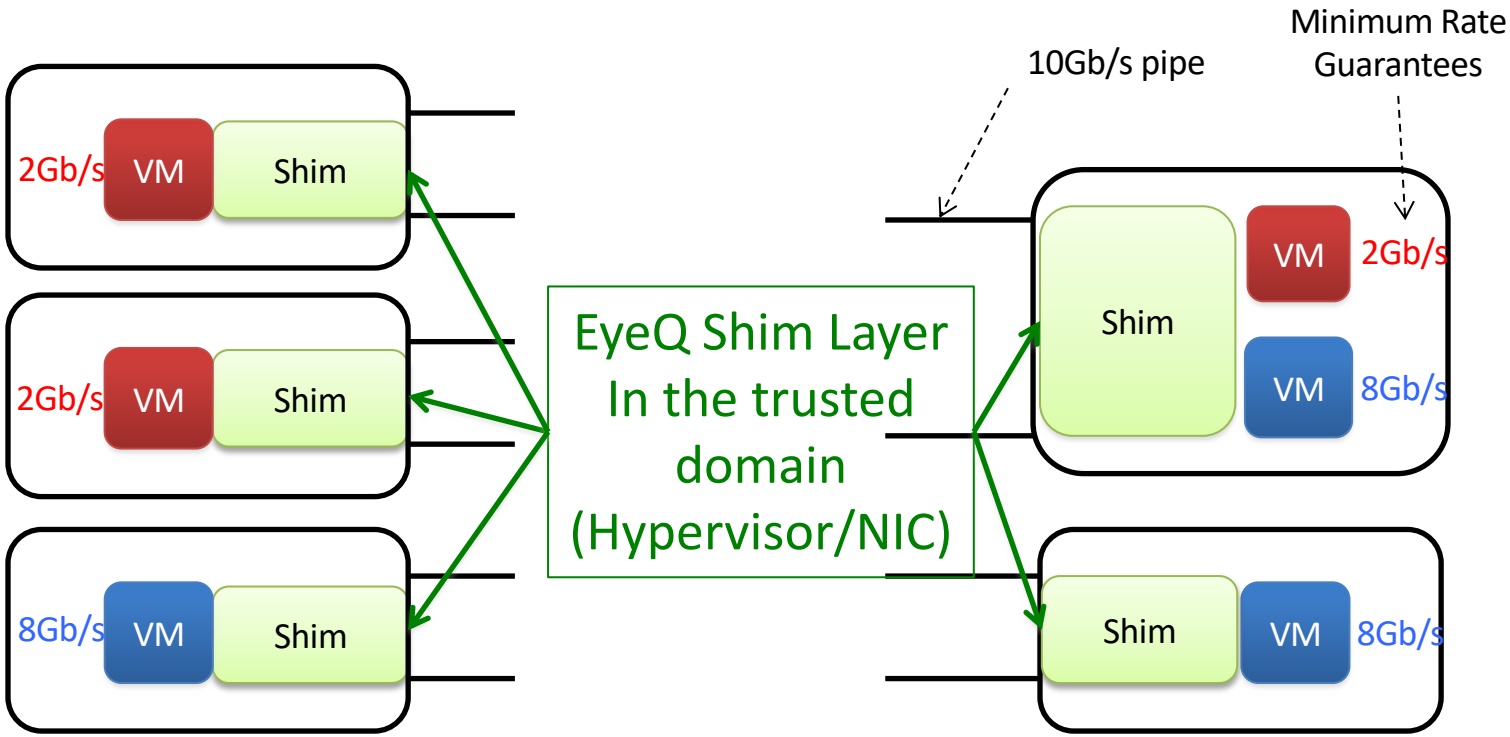
Basic Idea of EyeQ

- **Congestion-controlled hypervisor-to-hypervisor tunnel**
- Tunnel: A logical bundle of all flows between a VM pair
- Rate-limit tunnels weight-proportionally (a la TCP)

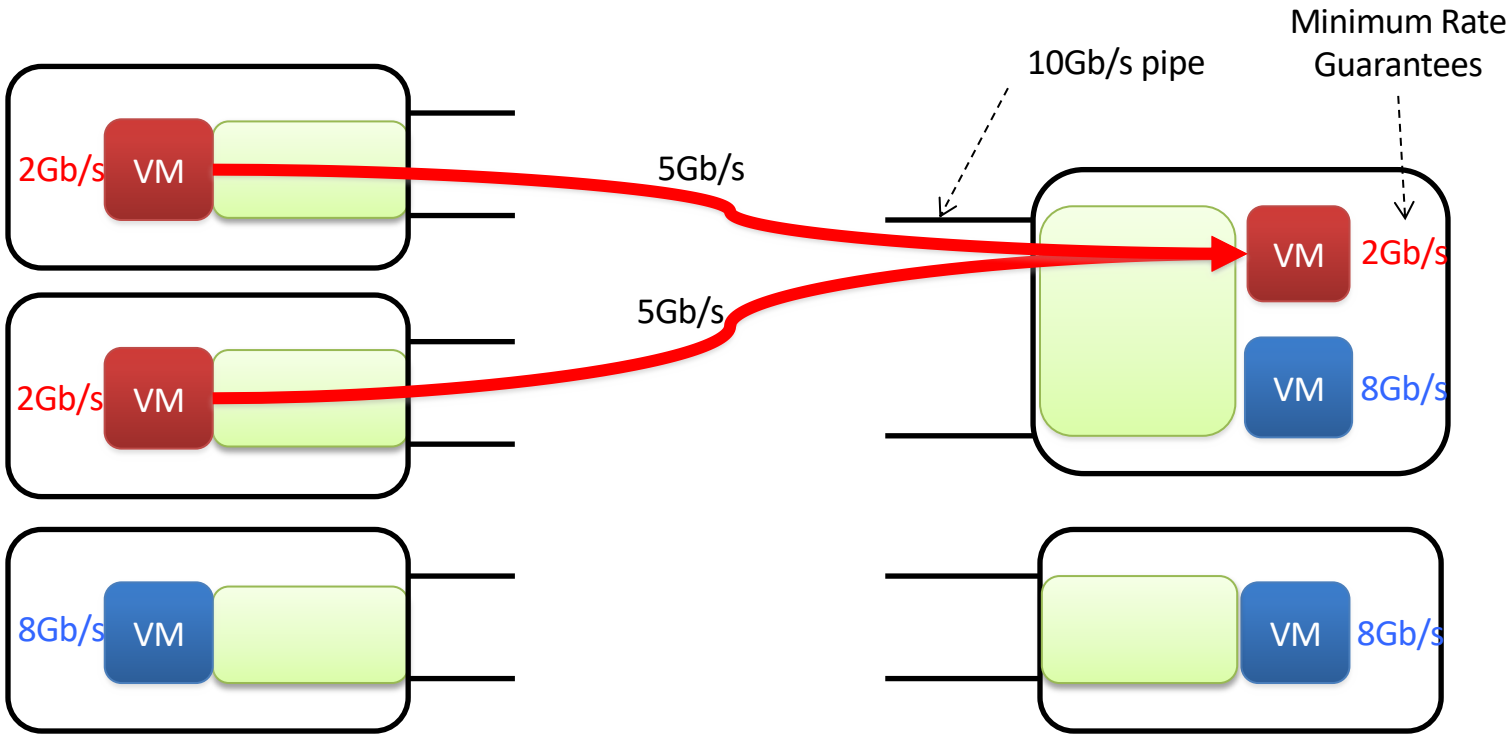


- **Distributed congestion control**: Efficiency, scalability, fine time scale
- **Hypervisor-based**: Isolation from tenants, no new H/W mechanism

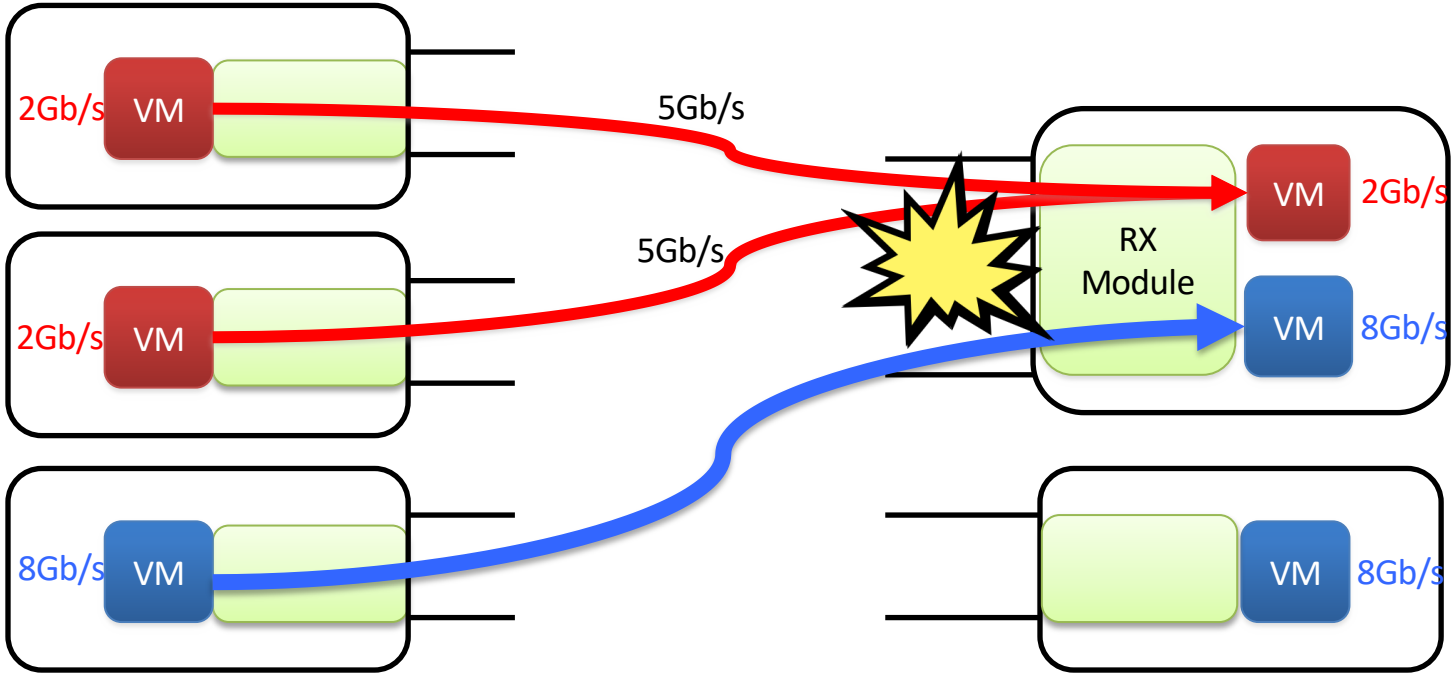
Decentralized Scheduling



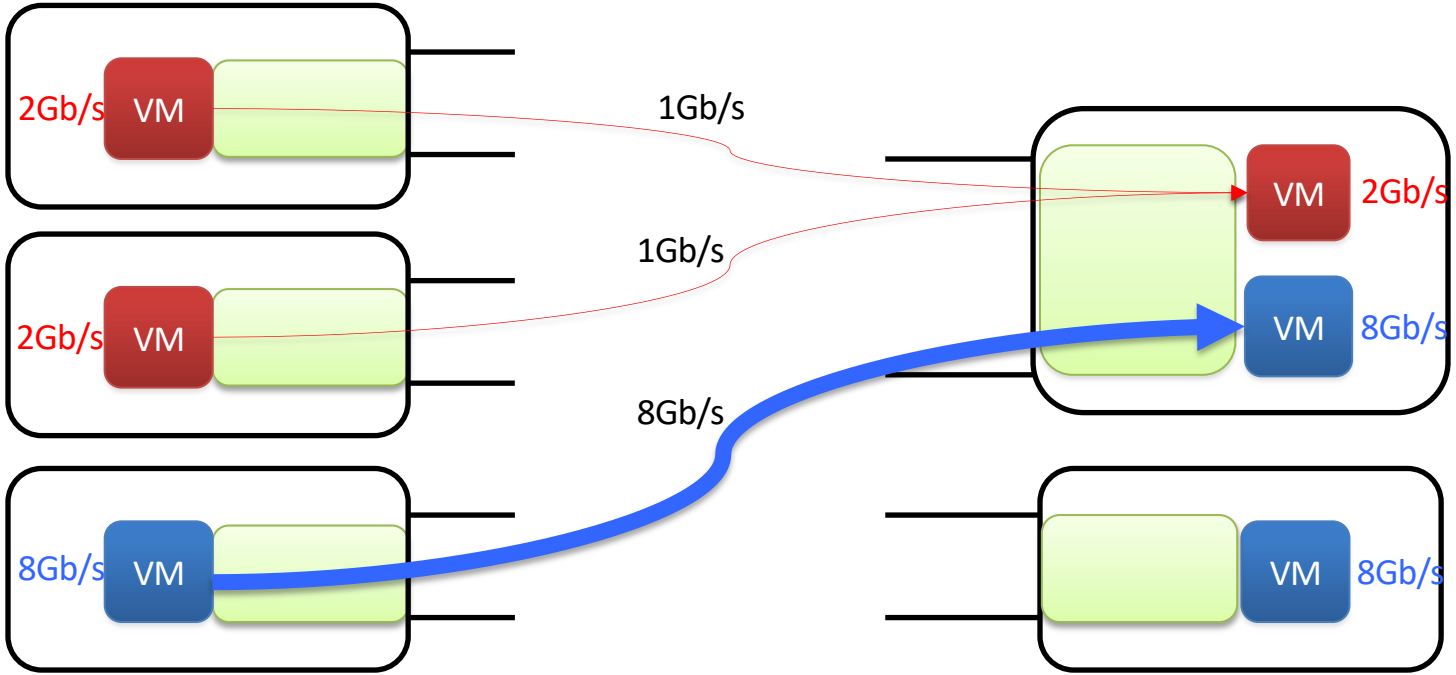
Decentralized Scheduling



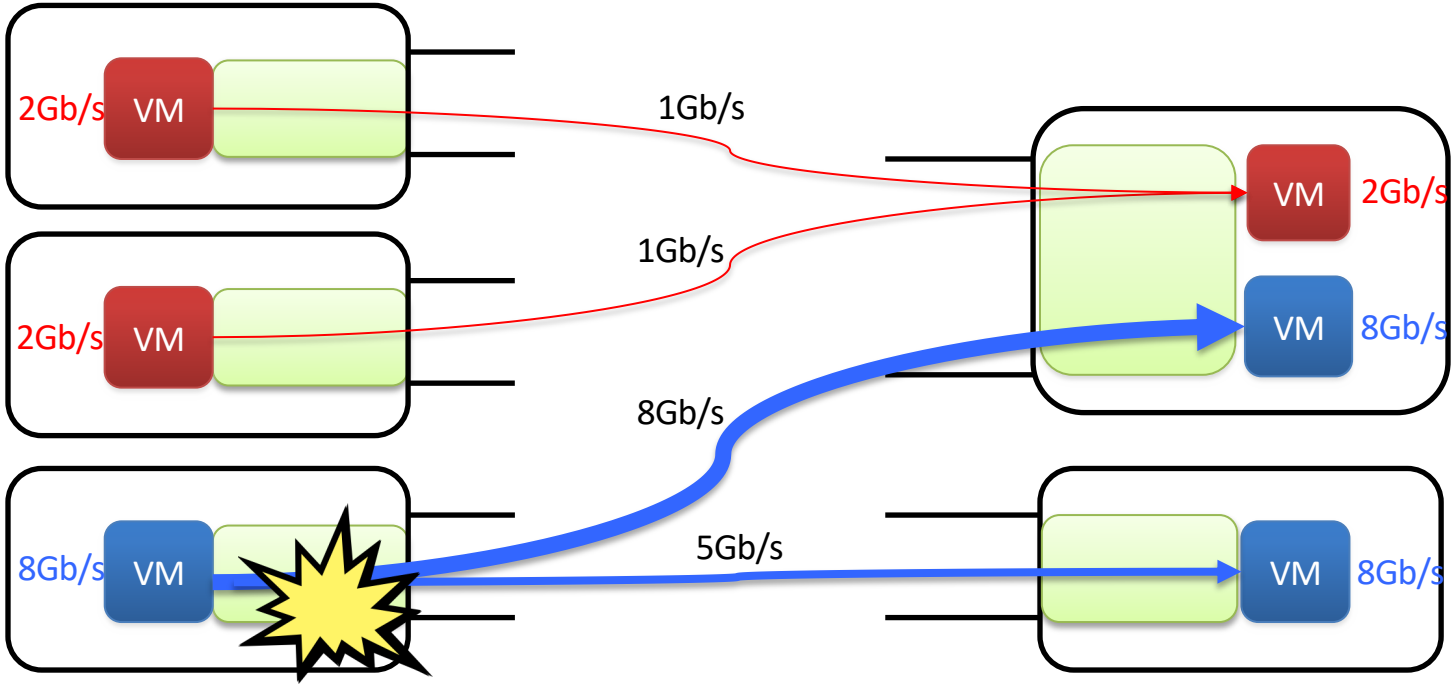
Decentralized Scheduling



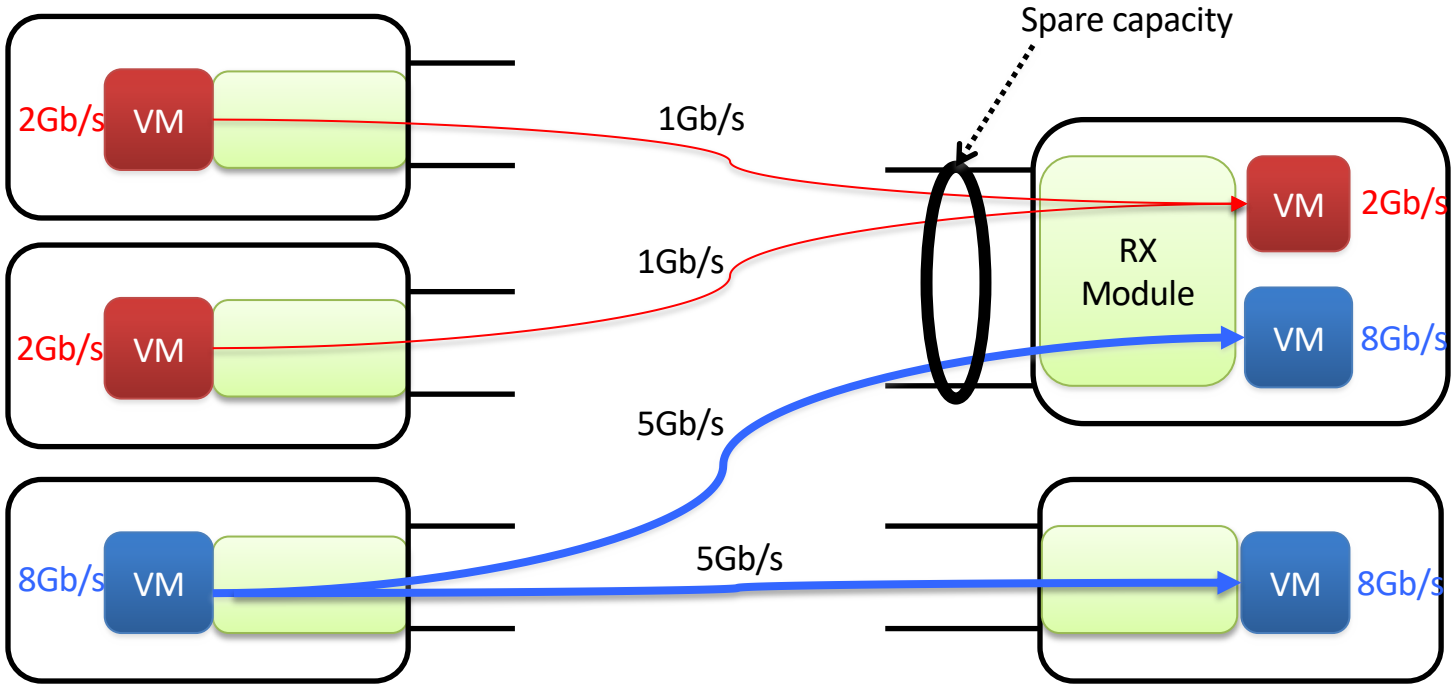
Decentralized Scheduling



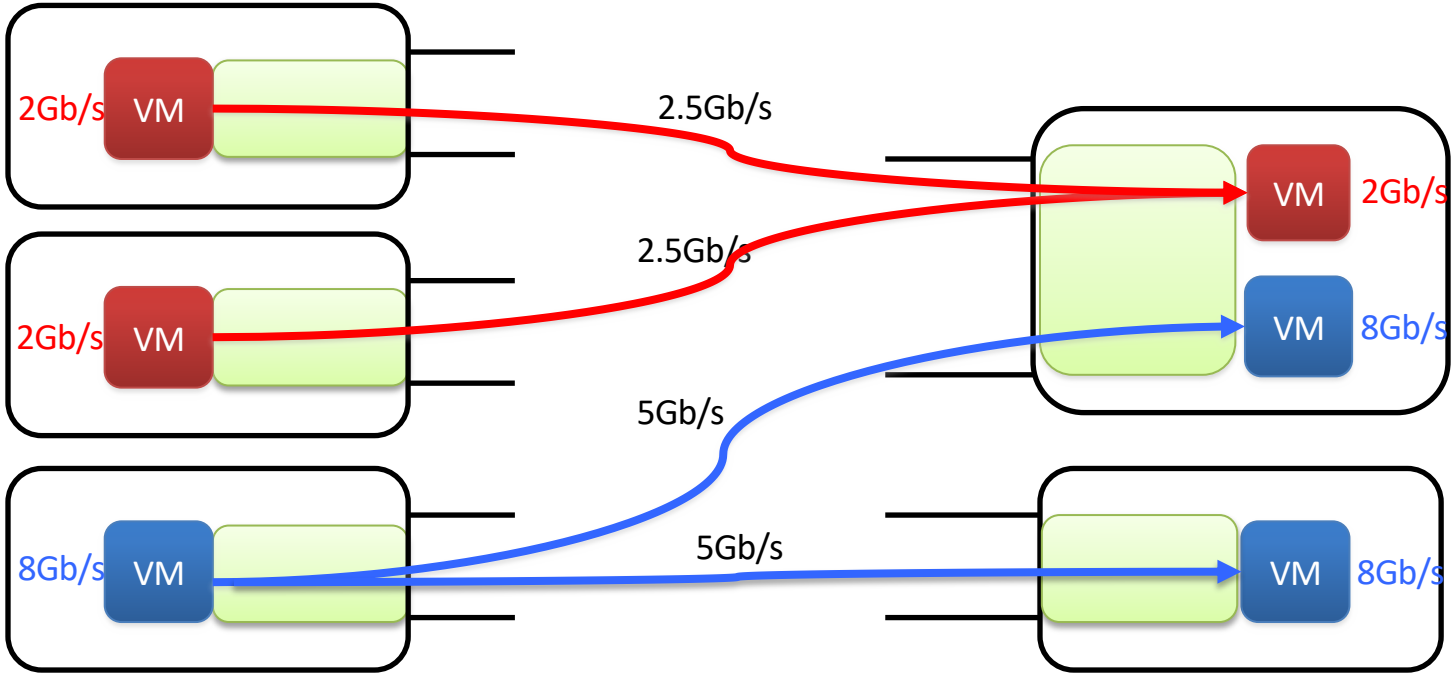
Decentralized Scheduling



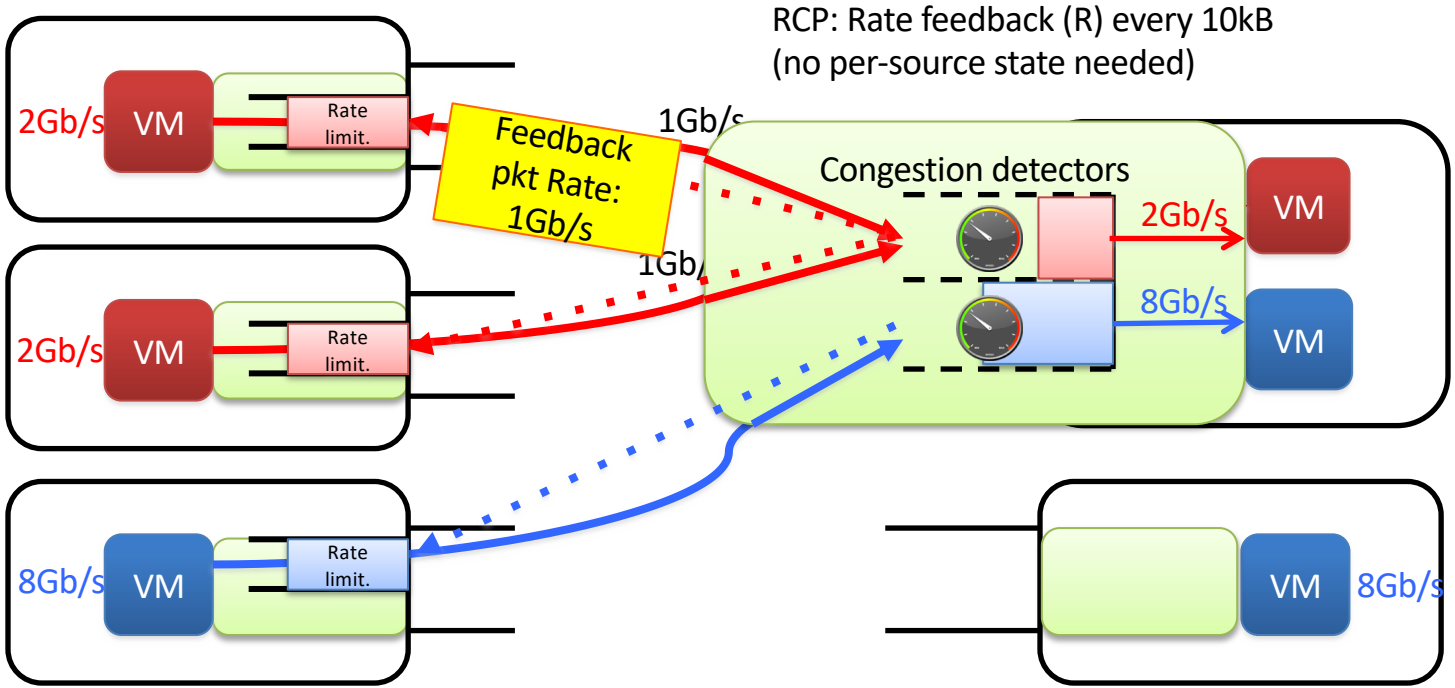
Work Conserving Allocations



Work Conserving Allocations



Transmit/Receive Modules



Per-destination rate limiters:
only if dest. is congested… bypass otherwise

EyeQ's Key Contribution: Simplicity

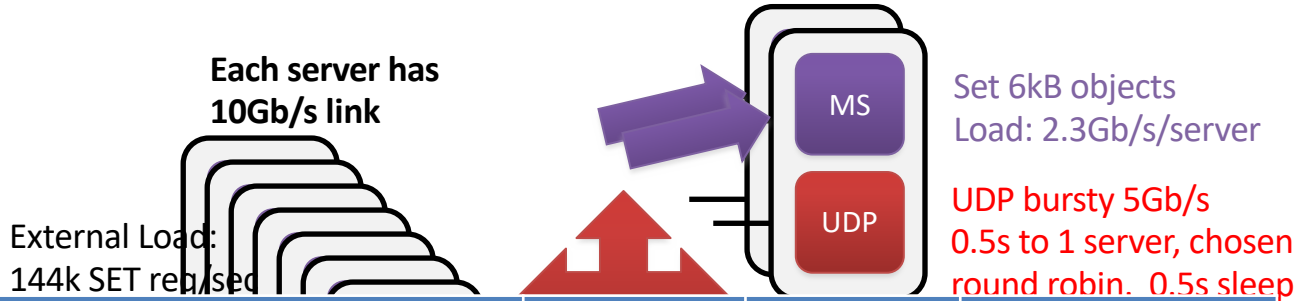
- Observation
  - Network Congestion predominantly occurs at the Edge (Hypervisor / Top of Rack)
- Consequences: Simplicity
  - Distributed, end-to-end bandwidth allocation
    - Amenable to NIC-based implementation
  - Network need not be tenant aware
- Implementation
  - High speed in software at 10Gb/s

Timescales Matter

- Fast convergence important
  - Switches only have few MB (milliseconds) worth of buffering before they drop packets

- RCP's worst-case convergence time
  - N long lived flows competing for a single bottleneck: few milliseconds.
  - Usually few 100 microseconds.

Macro Evaluation: Memcached Latency



Scenario	50th	99.9th	Throughput
Baseline (Linux 3.4)	98us	666us	144kreq/s
Without Interference + EyeQ	100us	**630**us	144kreq/s
With Interference	**4127**us	**>10^6**us	144kreq/s
With Interference + EyeQ	102us	**750**us	144kreq/s

What Is Network Virtualization? How Does It Enable Agility?