Seawall: Performance Isolation for Cloud Datacenter Networks

Alan Shieh
Cornell University

Srikanth Kandula
Albert Greenberg
Changhoon Kim
Microsoft Research

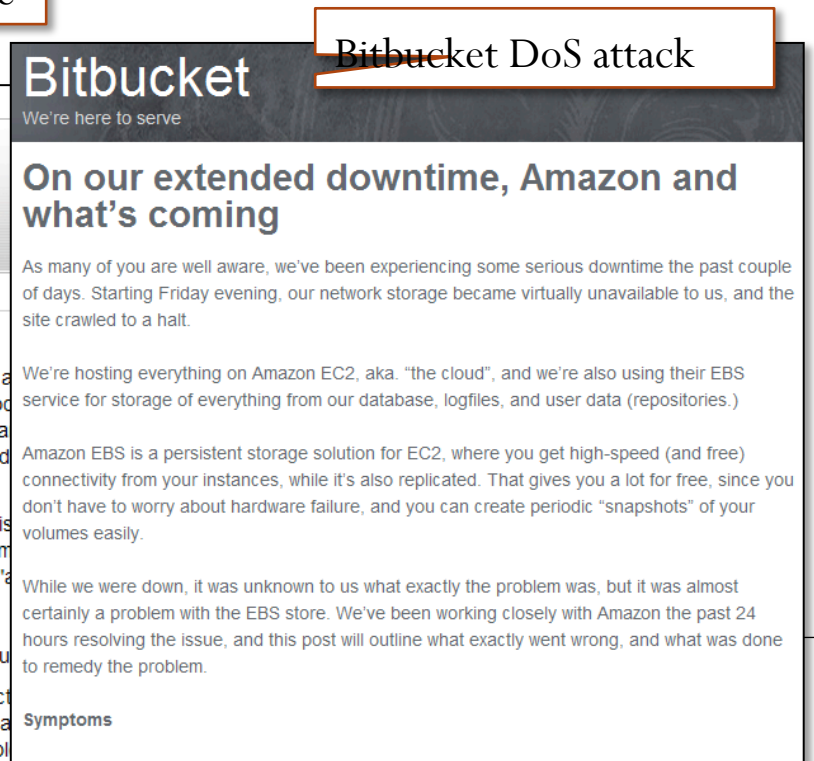
Cloud datacenters: Benefits and obstacles

- Moving to the cloud has manageability, costs & elasticity benefits
- Selfish tenants can monopolize resources
- Compromised & malicious tenants can degrade system performance
- Problems already occur

Runaway client overloads storage

Spammers on AWS

Bitbucket DoS attack



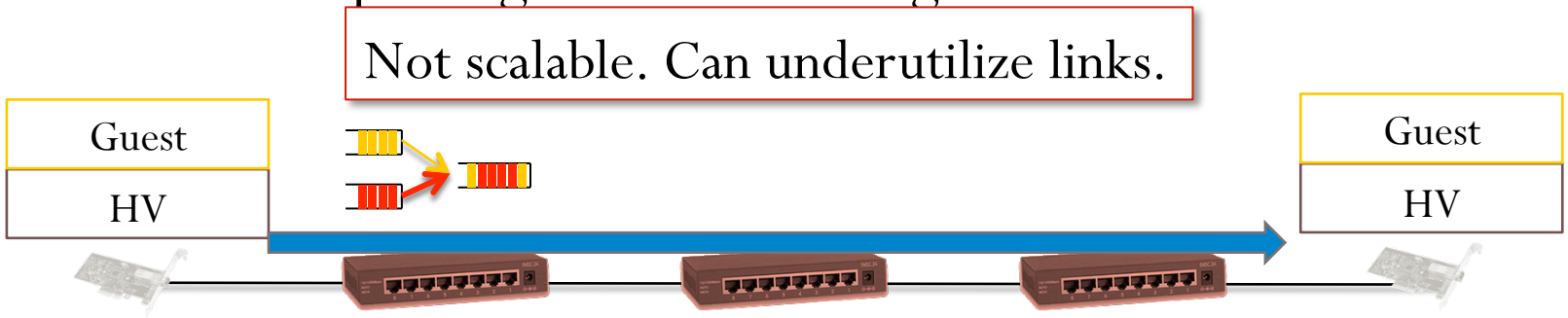
Goals

- Isolate tenants to avoid collateral damage
- Control each tenant's share of network
- Utilize all network capacity
- Constraints
  - Cannot trust tenant code
  - Minimize network reconfiguration during VM churn
  - Minimize end host and network cost

Existing mechanisms are insufficient for cloud

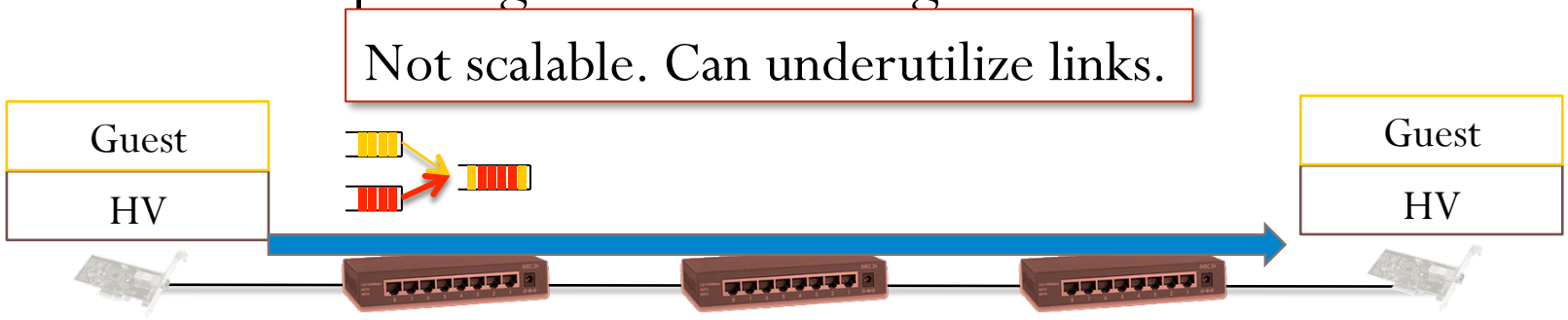
Existing mechanisms are insufficient

- In-network queuing and rate limiting

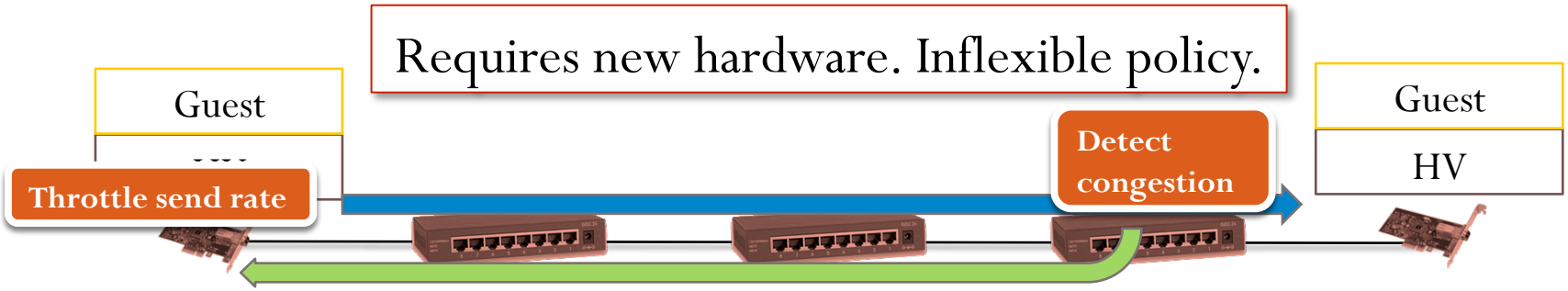


Existing mechanisms are insufficient

- In-network queuing and rate limiting



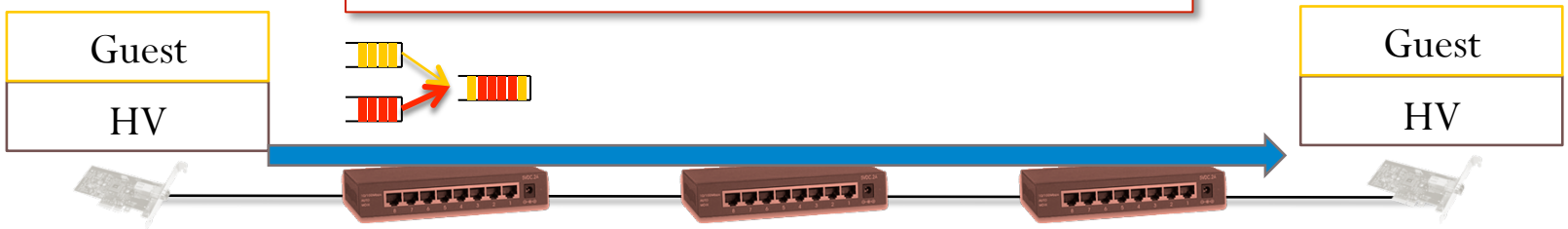
- Network-to-source congestion control (Ethernet QCN)



Existing mechanisms are insufficient

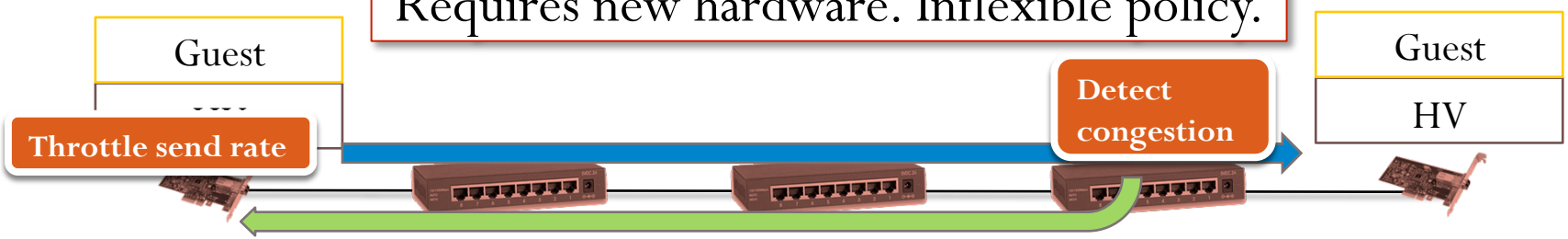
- In-network queuing and rate limiting

Not scalable. Can underutilize links.



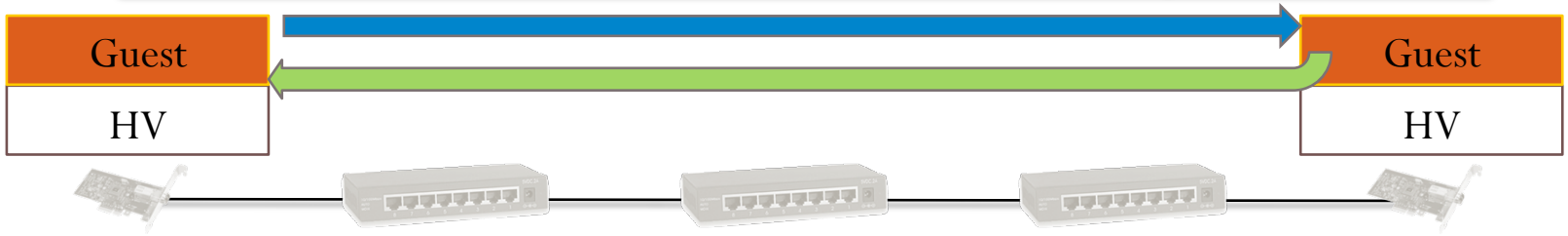
- Network-to-source congestion control (Ethernet QCN)

Requires new hardware. Inflexible policy.

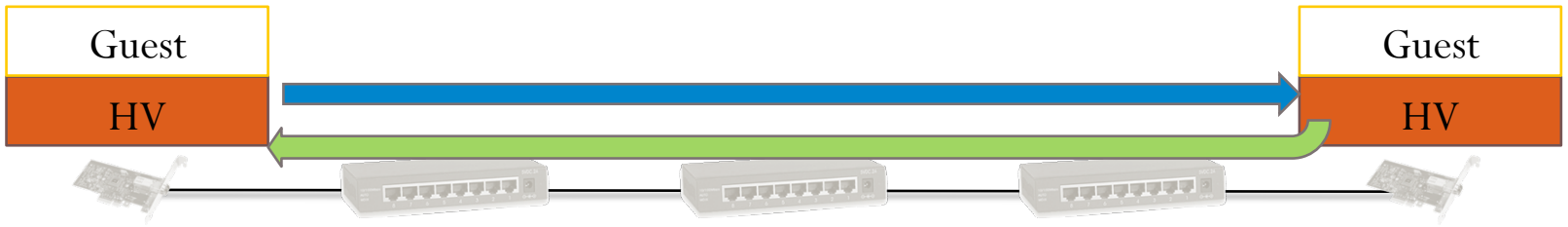


- End-to-end congestion control (TCP)

Poor control over allocation. Guests can change TCP stack.



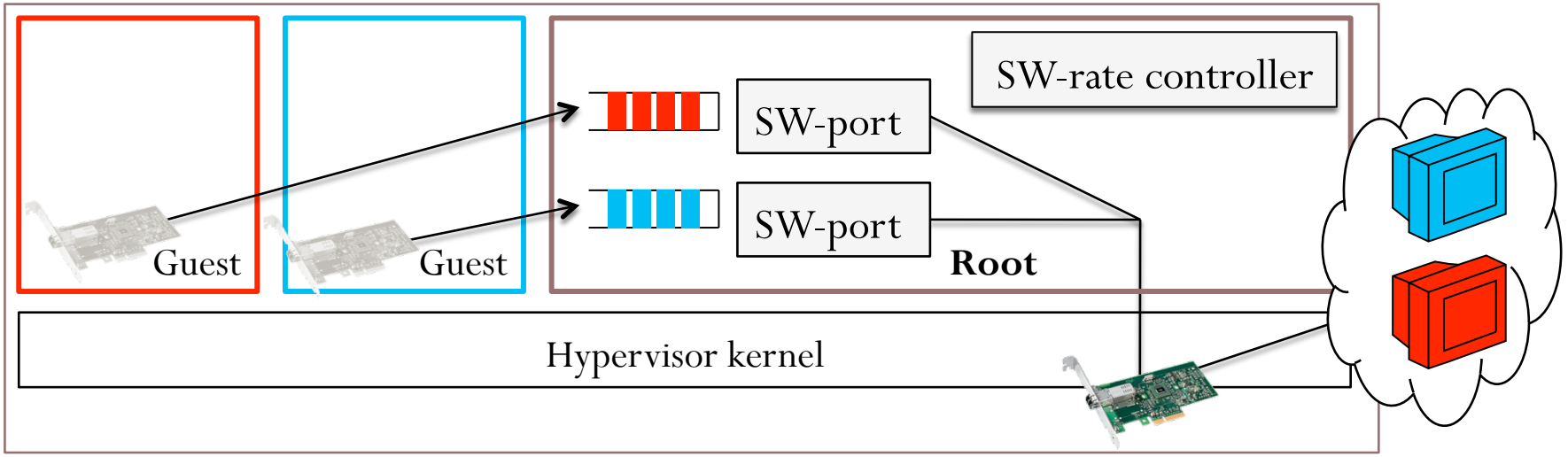
Seawall = Congestion controlled, hypervisor-to-hypervisor tunnels



Benefits

- Scales to # of tenants, flows, and churn
- Don't need to trust tenant
- Works on commodity hardware
- Utilizes network links efficiently
- Achieves good performance
  (1 Gb/s line rate & low CPU overhead)

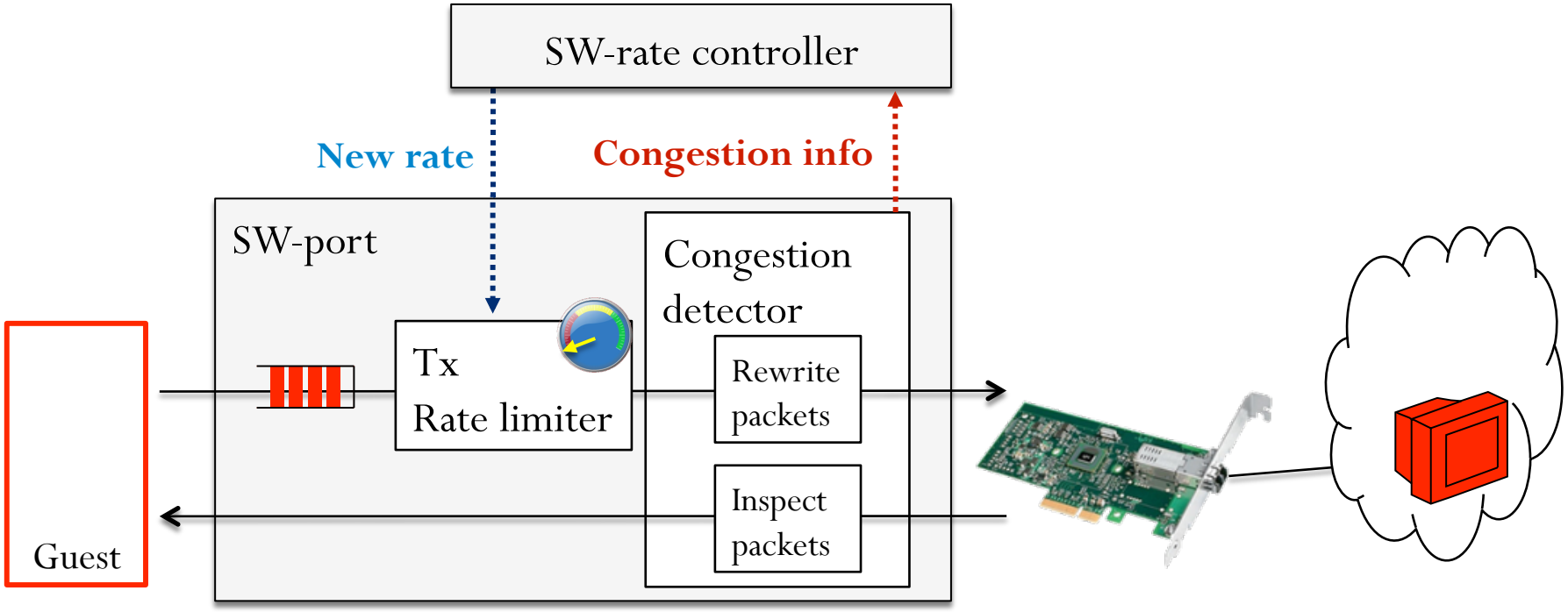
Components of Seawall



- Seawall rate controller allocates network resources for each output flow
  - Goal: achieve utilization and division
- Seawall ports enforce decisions of rate controller
  - Lie on forwarding path
  - One per VM source/destination pair

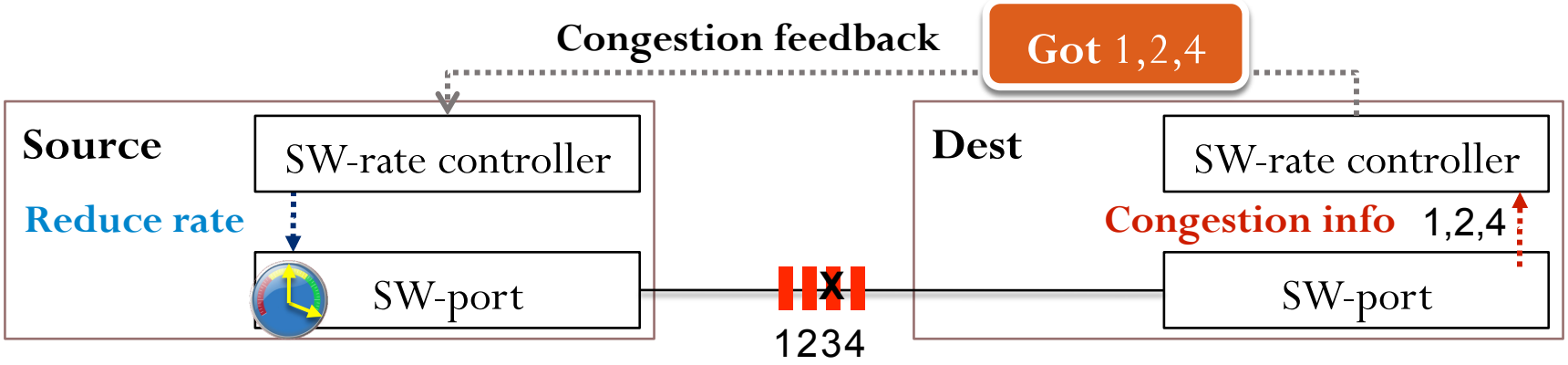
Seawall port

- Rate limit transmit traffic
- Rewrite and monitor traffic to support congestion control
- Exchanges congestion feedback and rate info with controller

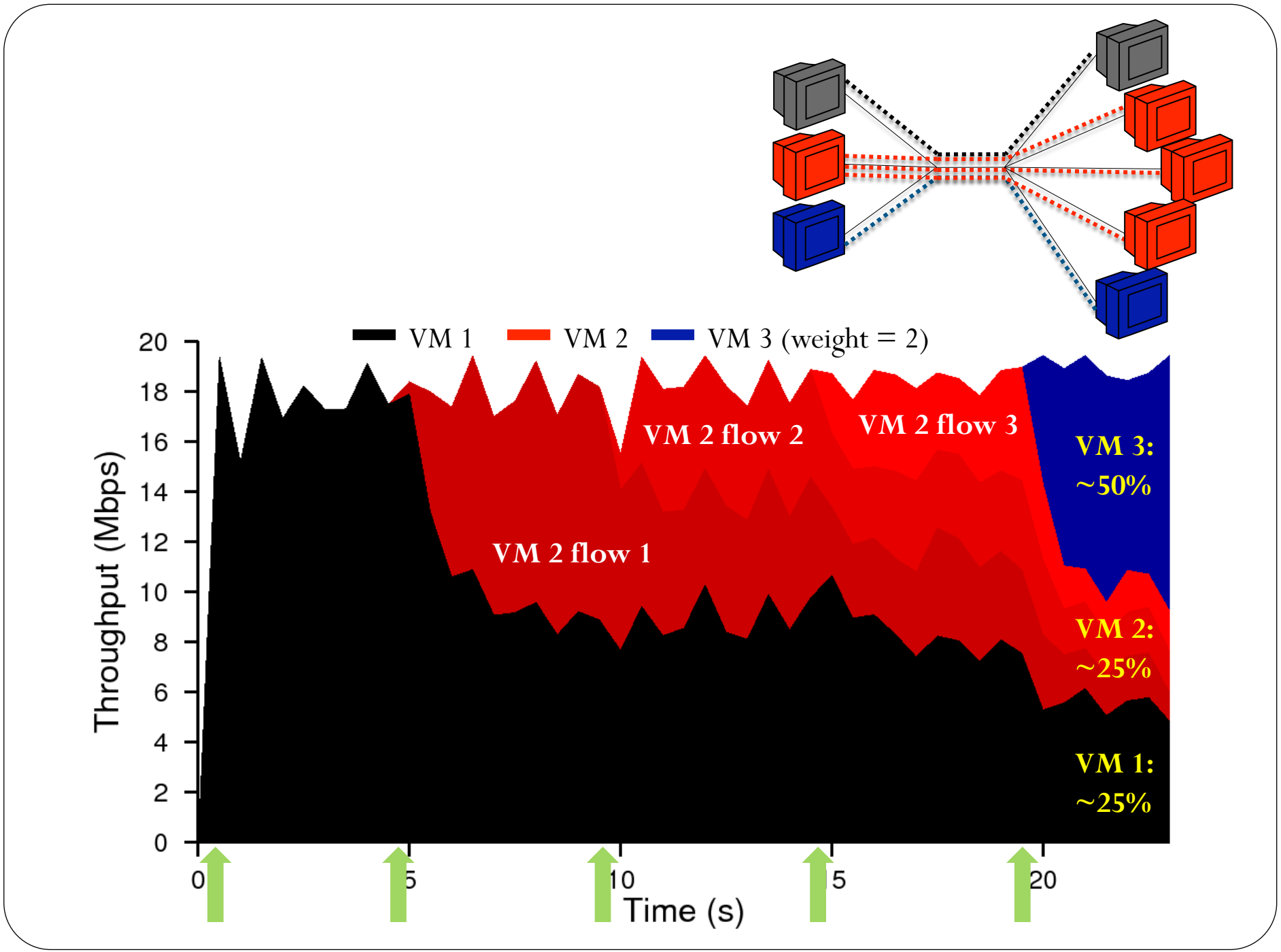


Rate controller:
Operation and control loop

- Rate controller adjusts rate limit based on presence and absence of loss

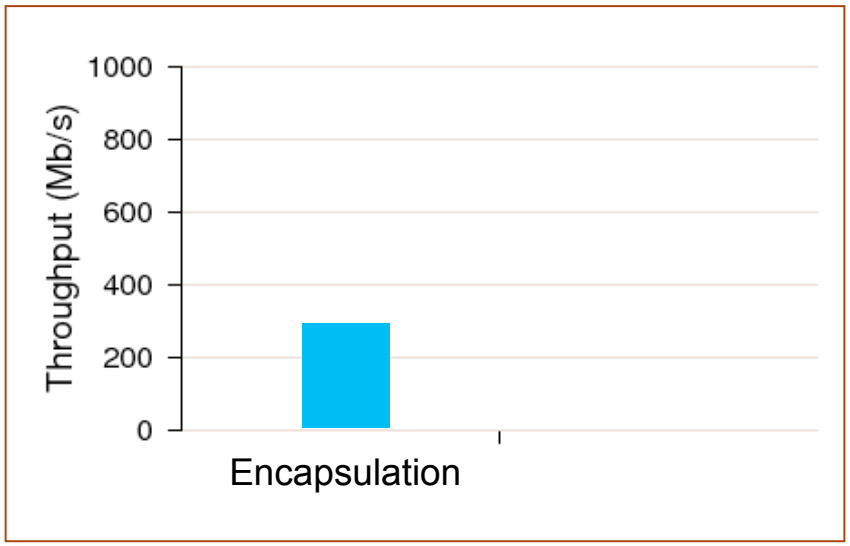


- Algorithm divides network proportional to weights & is max/min fair
  - Efficiency: AIMD with faster increase
  - Traffic-agnostic allocation:
    Per-link share is same regardless of # of flows & destinations


VM 1
VM 2
VM 3 (weight = 2)
VM 2 flow 1
VM 2 flow 2
VM 2 flow 3
VM 3:
~50%
VM 2:
~25%
VM 1:
~25%
Throughput (Mbps)
Time (s)
0
2
4
6
8
10
12
14
16
18
20
0
5
10
15
20

Improving SW-port performance

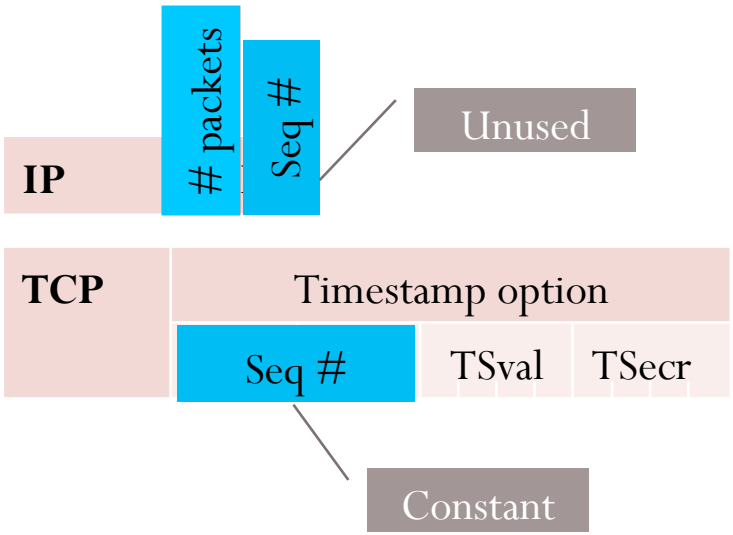
- How to add congestion control header to packets?
- Naïve approach: Use encapsulation, but poses problems
  - More code in SW-Port
  - Breaks hardware optimizations that depend on header format
    - Packet ACLs: Filter on TCP 5-tuple
    - Segmentation offload: Parse TCP header to split packets
    - Load balancing: Hash on TCP 5-tuple to spray packets (e.g. RSS)



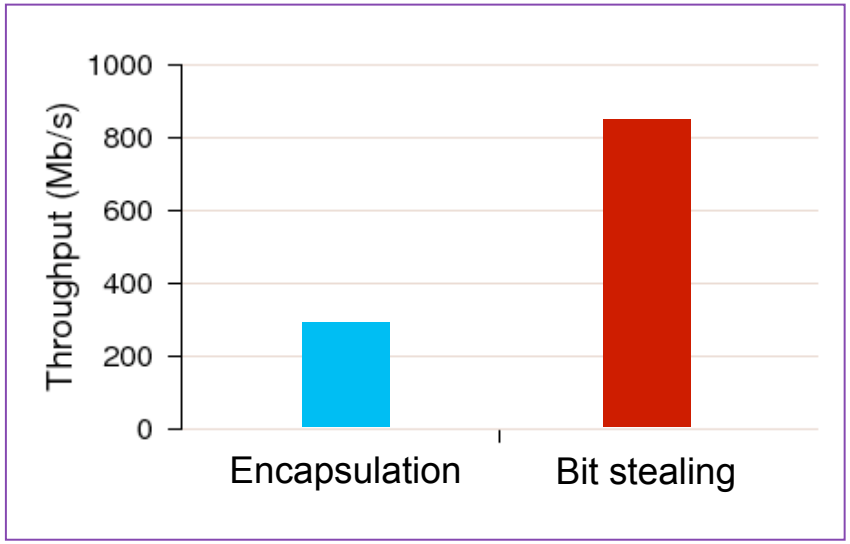
"Bit stealing" solution:

Use spare bits from existing headers

- Constraints on header modifications
  - Network can route & process packet
  - Receiver can reconstruct for guest
- Other protocols: might need paravirtualization.



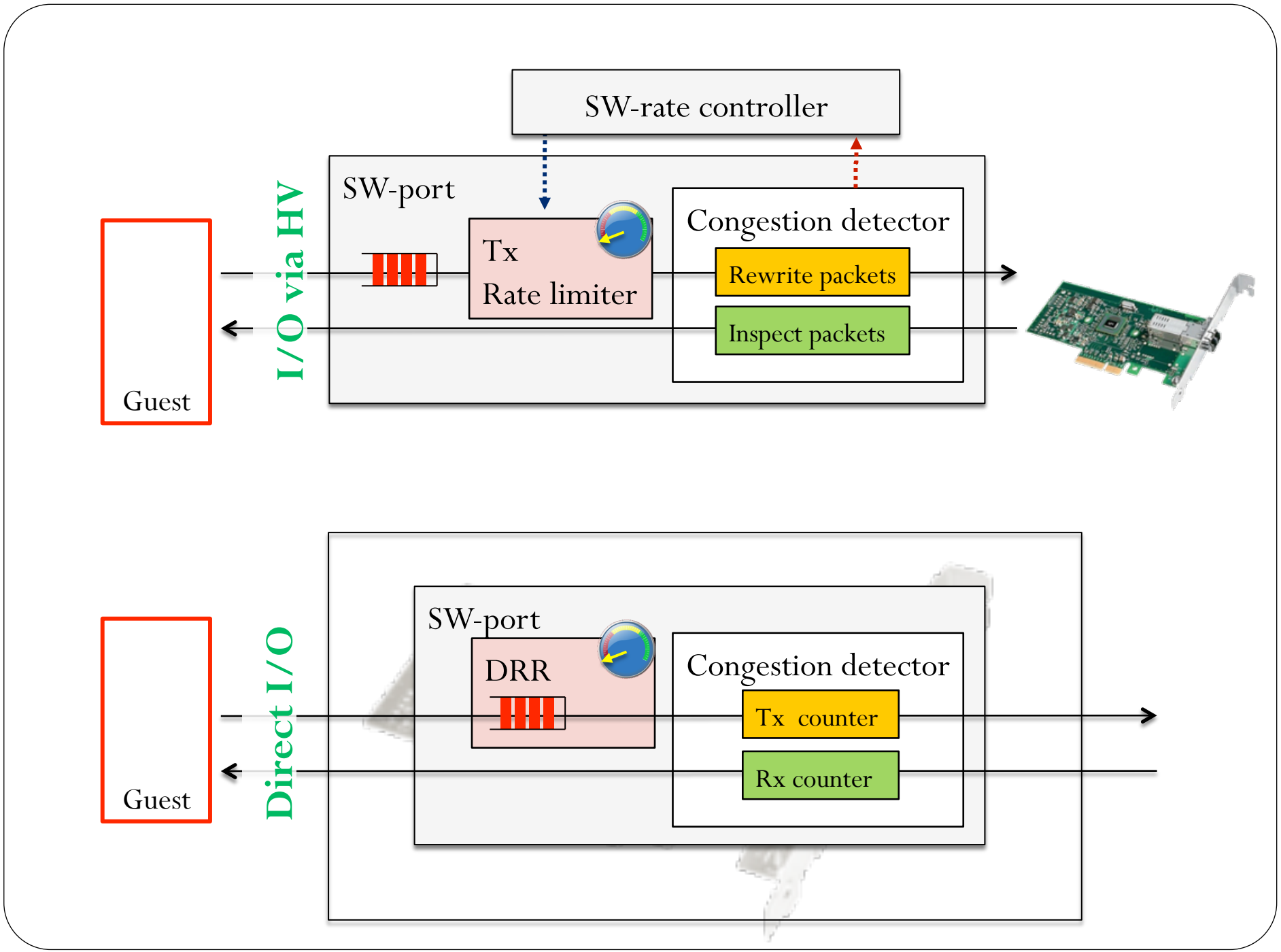
“Bit stealing” solution:
Performance improvement



Throughput: 280 Mb/s => 843 Mb/s

Supporting future networks

- Hypervisor vSwitch scales to 1 Gbps, but may be bottleneck for 10 Gbps
- Multiple approaches to scale to 10 Gbps
  - Hypervisor & multi-core optimizations
  - Bypass hypervisor with direct I/O (e.g. SR-IOV)
  - Virtualization-aware physical switch (e.g. NIV, VEPA)
- While efficient, currently direct I/O loses policy control
- Future SR-IOV NICs support classifiers, filters, rate limiters

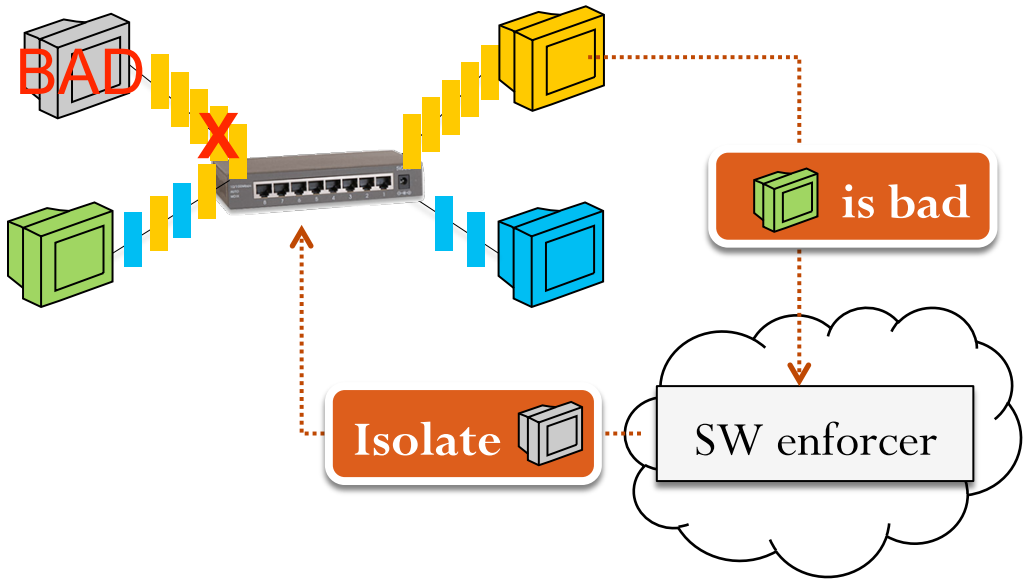

SW-rate controller
SW-port
Tx
Rate limiter
Congestion detector
Rewrite packets
Inspect packets
I/O via HV
Guest
SW-port
DRR
Congestion detector
Tx counter
Rx counter
Direct I/O
Guest

Summary

- Without performance isolation, no protection in cloud against selfish, compromised & malicious tenants
- Hypervisor rate limiters + end-to-end rate controller provide isolation, control, and efficiency
- Prototype achieves performance and security on commodity hardware

Preserving performance isolation after hypervisor compromise

- Compromised hypervisor at source can flood network
- Solution:
  Use network filtering to isolate sources that violate congestion control
  - Destinations act as detector



Preserving performance isolation after hypervisor compromise

- Pitfall: If destination is compromised, danger of DoS from false accusations
- Refinement: Apply least privilege (i.e. fine-grained filtering)