Securing Self-Virtualizing Ethernet Devices

Igor Smolyar, Muli Ben-Yehuda, Dan Tsafrir

In a nutshell

- We show an attack where an untrusted virtual machine completely controls the network bandwidth of other, unrelated virtual machines
- This attack exploits a vulnerability in self-virtualizing Ethernet NICs
- To defend against the attack, you have to either:
  - Modify device hardware/firmware, or
  - Give up on flow control functionality and lose performance, or
  - Trust your virtual machines
- We show how to build an attack-resistant NIC

In a nutshell

- We show an attack where an untrusted virtual machine completely controls the network bandwidth of other, unrelated virtual machines
- This attack exploits a vulnerability in self-virtualizing Ethernet NICs
- To defend against the attack, you have to either:
  - Modify device hardware/firmware, or
  - Give up on flow control functionality and lose performance, or
  - Trust your virtual machines
- We show how to build an attack-resistant NIC

In a nutshell

- We show an attack where an untrusted virtual machine completely controls the network bandwidth of other, unrelated virtual machines
- This attack exploits a vulnerability in self-virtualizing Ethernet NICs
- To defend against the attack, you have to either:
    - Modify device hardware/firmware, or
    - Give up on flow control functionality and lose performance, or
    - Trust your virtual machines
- We show how to build an attack-resistant NIC

In a nutshell

- We show an attack where an untrusted virtual machine completely controls the network bandwidth of other, unrelated virtual machines
- This attack exploits a vulnerability in self-virtualizing Ethernet NICs
- To defend against the attack, you have to either:
  - Modify device hardware/firmware, or
  - Give up on flow control functionality and lose performance, or
  - Trust your virtual machines
- We show how to build an attack-resistant NIC

In a nutshell

- We show an attack where an untrusted virtual machine completely controls the network bandwidth of other, unrelated virtual machines
- This attack exploits a vulnerability in self-virtualizing Ethernet NICs
- To defend against the attack, you have to either:
  - Modify device hardware/firmware, or
  - Give up on flow control functionality and lose performance, or
  - Trust your virtual machines
- We show how to build an attack-resistant NIC

In a nutshell

- We show an attack where an untrusted virtual machine completely controls the network bandwidth of other, unrelated virtual machines
- This attack exploits a vulnerability in self-virtualizing Ethernet NICs
- To defend against the attack, you have to either:
  - Modify device hardware/firmware, or
  - Give up on flow control functionality and lose performance, or
  - Trust your virtual machines
- We show how to build an attack-resistant NIC

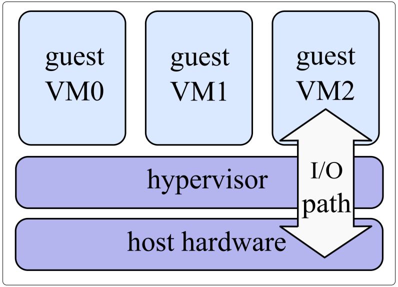
In a nutshell

- We show an attack where an untrusted virtual machine completely controls the network bandwidth of other, unrelated virtual machines
- This attack exploits a vulnerability in self-virtualizing Ethernet NICs
- To defend against the attack, you have to either:
  - Modify device hardware/firmware, or
  - Give up on flow control functionality and lose performance, or
  - Trust your virtual machines
- We show how to build an attack-resistant NIC

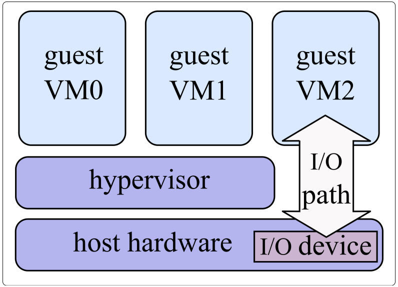
In a nutshell

- We show an attack where an untrusted virtual machine completely controls the network bandwidth of other, unrelated virtual machines
- This attack exploits a vulnerability in self-virtualizing Ethernet NICs
- To defend against the attack, you have to either:
  - Modify device hardware/firmware, or
  - Give up on flow control functionality and lose performance, or
  - Trust your virtual machines
- We show how to build an attack-resistant NIC

Types of I/O Virtualization



Emulation & Para-virtualization



Direct I/O Device Assignment

Direct Device Assignment

- **Great performance** minimizes the number of I/O-related world switches between the guest and the host
- **Problem: not scalable** ~5-10 I/O devices per host, **but** 50-100 virtual machines per host
- **Solution: self-virtualizing devices** PCI-SIG proposed the Single Root I/O Virtualization (**SRIOV**) standard for scalable device assignment
  - PCI device presents itself as multiple virtual interfaces
  - SRIOV spec supports up to 64K virtual devices
  - Intel XL710 40GbE NIC implements 128 virtual interfaces

Direct Device Assignment

- **Great performance** minimizes the number of I/O-related world switches between the guest and the host
- **Problem: not scalable** ~5-10 I/O devices per host, **but** 50-100 virtual machines per host
- **Solution: self-virtualizing devices** PCI-SIG proposed the Single Root I/O Virtualization (**SRIOV**) standard for scalable device assignment
  - PCI device presents itself as multiple virtual interfaces
  - SRIOV spec supports up to 64K virtual devices
  - Intel XL710 40GbE NIC implements 128 virtual interfaces

Direct Device Assignment

- **Great performance** minimizes the number of I/O-related world switches between the guest and the host
- **Problem: not scalable** ~5-10 I/O devices per host, **but** 50-100 virtual machines per host
- **Solution: self-virtualizing devices** PCI-SIG proposed the Single Root I/O Virtualization (**SRIOV**) standard for scalable device assignment
  - PCI device presents itself as multiple virtual interfaces
  - SRIOV spec supports up to 64K virtual devices
  - Intel XL710 40GbE NIC implements 128 virtual interfaces

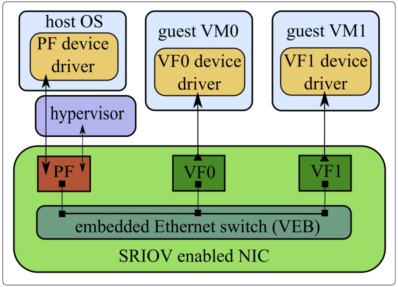
Direct Device Assignment

- **Great performance** minimizes the number of I/O-related world switches between the guest and the host
- **Problem: not scalable** ~5-10 I/O devices per host, **but** 50-100 virtual machines per host
- **Solution: self-virtualizing devices** PCI-SIG proposed the Single Root I/O Virtualization (**SRIOV**) standard for scalable device assignment
  - PCI device presents itself as multiple virtual interfaces
  - SRIOV spec supports up to 64K virtual devices
  - Intel XL710 40GbE NIC implements 128 virtual interfaces

Single Root I/O Virtualization (SRIOV)

Each SRIOV capable device consists of at least one Physical Function (PF) and multiple virtual partitions called Virtual Functions (VF)

- **PF** is a standard PCIe function with full configuration space. Can control entire PCI device and perform I/O operations
- **VF** is a "lightweight" PCI function that implements only only a subset of standard PCI functionality, mostly performs I/O



SRIOV NIC in a virtualized environment

Single Root I/O Virtualization (SRIOV)

- **HPC** with SRIOV it is possible to virtualize HPC setups. Without SRIOV, many use cases in cloud computing, HPC and enterprise data centers would be infeasible
- **Cloud Service Providers** such as Amazon Elastic Compute Cloud (EC2) use SRIOV as the underlying technology in EC2 HPC services
- **Data Centers** Oracle Exalogic Elastic Cloud uses SRIOV technology to share the internal network

Single Root I/O Virtualization (SRIOV)

- **HPC** with SRIOV it is possible to virtualize HPC setups. Without SRIOV, many use cases in cloud computing, HPC and enterprise data centers would be infeasible
- **Cloud Service Providers** such as Amazon Elastic Compute Cloud (EC2) use SRIOV as the underlying technology in EC2 HPC services
- **Data Centers** Oracle Exalogic Elastic Cloud uses SRIOV technology to share the internal network

Single Root I/O Virtualization (SRIOV)

- **HPC** with SRIOV it is possible to virtualize HPC setups. Without SRIOV, many use cases in cloud computing, HPC and enterprise data centers would be infeasible
- **Cloud Service Providers** such as Amazon Elastic Compute Cloud (EC2) use SRIOV as the underlying technology in EC2 HPC services
- **Data Centers** Oracle Exalogic Elastic Cloud uses SRIOV technology to share the internal network

Single Root I/O Virtualization (SRIOV)

- **HPC** with SRIOV it is possible to virtualize HPC setups. Without SRIOV, many use cases in cloud computing, HPC and enterprise data centers would be infeasible
- **Cloud Service Providers** such as Amazon Elastic Compute Cloud (EC2) use SRIOV as the underlying technology in EC2 HPC services
- **Data Centers** Oracle Exalogic Elastic Cloud uses SRIOV technology to share the internal network

Ethernet Flow Control

- **Traditional Ethernet is lossy** with no guarantee of delivery of Ethernet frames
  - Most data frame drops happen when the receiver's buffers are full and has no memory available to store incoming data frames
  - Assumes that reliability provided by upper-level protocols (e.g. TCP) or applications
- **Ethernet Flow Control (FC)** proposed to create a lossless data link medium
- **Priority Flow Control (PFC)** extends FC for data centers, part of Data Center Bridging (DCB) or Converged Enhanced Ethernet (CEE)

Ethernet Flow Control

- **Traditional Ethernet is lossy** with no guarantee of delivery of Ethernet frames
  - Most data frame drops happen when the receiver's buffers are full and has no memory available to store incoming data frames
  - Assumes that reliability provided by upper-level protocols (e.g. TCP) or applications
- **Ethernet Flow Control (FC)** proposed to create a lossless data link medium
- **Priority Flow Control (PFC)** extends FC for data centers, part of Data Center Bridging (DCB) or Converged Enhanced Ethernet (CEE)

Ethernet Flow Control

- **Traditional Ethernet is lossy** with no guarantee of delivery of Ethernet frames
  - Most data frame drops happen when the receiver's buffers are full and has no memory available to store incoming data frames
  - Assumes that reliability provided by upper-level protocols (e.g. TCP) or applications
- **Ethernet Flow Control (FC)** proposed to create a lossless data link medium
- **Priority Flow Control (PFC)** extends FC for data centers, part of Data Center Bridging (DCB) or Converged Enhanced Ethernet (CEE)

Ethernet Flow Control

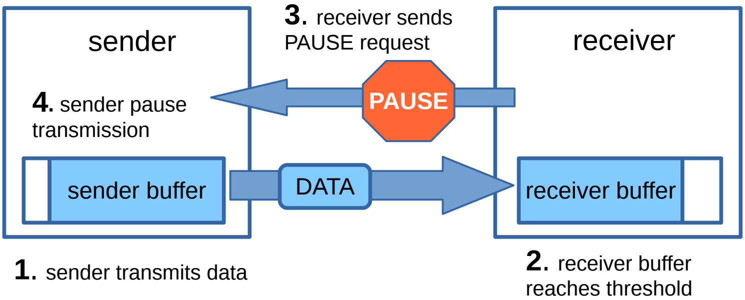
- **Traditional Ethernet is lossy** with no guarantee of delivery of Ethernet frames
  - Most data frame drops happen when the receiver's buffers are full and has no memory available to store incoming data frames
  - Assumes that reliability provided by upper-level protocols (e.g. TCP) or applications
- **Ethernet Flow Control (FC)** proposed to create a lossless data link medium
- **Priority Flow Control (PFC)** extends FC for data centers, part of Data Center Bridging (DCB) or Converged Enhanced Ethernet (CEE)

Ethernet Flow Control

1. The sender (e.g. Ethernet switch) transmits data faster than the receiver can process
2. The receiver (e.g. host's Ethernet NIC) runs out of space
3. The receiver sends the sender a MAC control frame with a pause request
4. The sender stops transmitting data for requested period of time

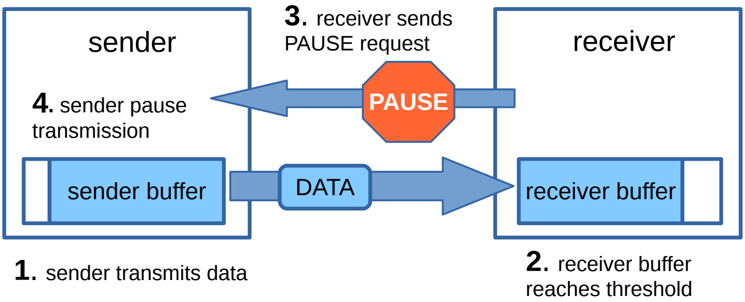
Ethernet Flow Control

1. The sender (e.g. Ethernet switch) transmits data faster than the receiver can process
2. The receiver (e.g. host's Ethernet NIC) runs out of space
3. The receiver sends the sender a MAC control frame with a pause request
4. The sender stops transmitting data for requested period of time



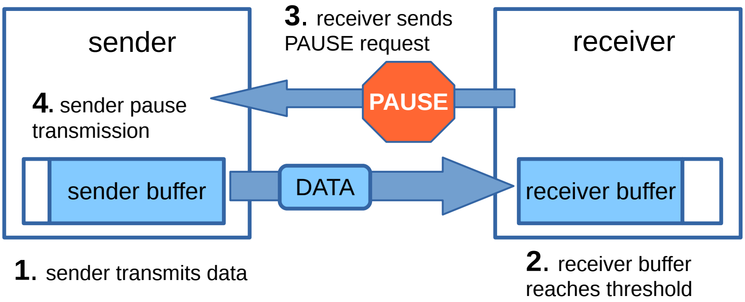
Ethernet Flow Control

1. The sender (e.g. Ethernet switch) transmits data faster than the receiver can process
2. The receiver (e.g. host's Ethernet NIC) runs out of space
3. The receiver sends the sender a MAC control frame with a pause request
4. The sender stops transmitting data for requested period of time



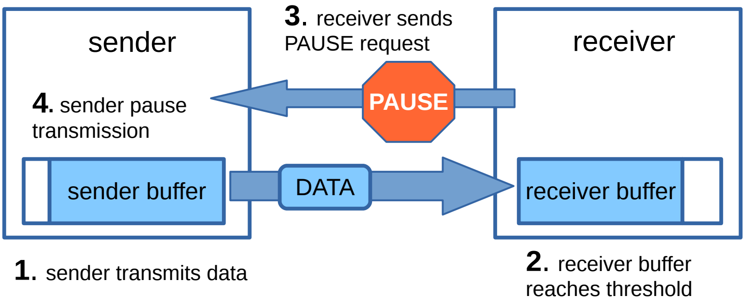
Ethernet Flow Control

1. The sender (e.g. Ethernet switch) transmits data faster than the receiver can process
2. The receiver (e.g. host's Ethernet NIC) runs out of space
3. The receiver sends the sender a MAC control frame with a pause request
4. The sender stops transmitting data for requested period of time



Ethernet Flow Control

1. The sender (e.g. Ethernet switch) transmits data faster than the receiver can process
2. The receiver (e.g. host's Ethernet NIC) runs out of space
3. The receiver sends the sender a MAC control frame with a pause request
4. The sender stops transmitting data for requested period of time



Ethernet Flow Control

link speed, Gbps	single frame pause time, ms	frame rate required to stop transmission, frames/second
1	33.6	30
10	3.36	299
40	0.85	1193

Table : Pause frame rate for stopping traffic completely

Attacking VMs via Flow Control

- **Flow Control** works on link-level
- **Link is shared** between VMs; all VMs with direct access to the VFs of the same PF share the same physical link to the edge switch
- **Each FC Pause Frame** halts traffic on the *entire* link
- **All VFs** associated with this PF are affected

Attacking VMs via Flow Control

- **Flow Control** works on link-level
- **Link is shared** between VMs; all VMs with direct access to the VFs of the same PF share the same physical link to the edge switch
- **Each FC Pause Frame** halts traffic on the *entire* link
- **All VFs** associated with this PF are affected

Attacking VMs via Flow Control

- **Flow Control** works on link-level
- **Link is shared** between VMs; all VMs with direct access to the VFs of the same PF share the same physical link to the edge switch
- **Each FC Pause Frame** halts traffic on the *entire* link
- **All VFs** associated with this PF are affected

Attacking VMs via Flow Control

- **Flow Control** works on link-level
- **Link is shared** between VMs; all VMs with direct access to the VFs of the same PF share the same physical link to the edge switch
- **Each FC Pause Frame** halts traffic on the *entire* link
- **All VFs** associated with this PF are affected

Attacking VMs via Flow Control

- **Flow Control** works on link-level
- **Link is shared** between VMs; all VMs with direct access to the VFs of the same PF share the same physical link to the edge switch
- **Each FC Pause Frame** halts traffic on the *entire* link
- **All VFs** associated with this PF are affected

The Attack

- The malicious VM sends a pause frame
- **All traffic on the shared link pauses**
- And then continues. . .
- **Until the malicious VM sends the next pause frame**

The Attack

- The malicious VM sends a pause frame
- **All traffic on the shared link pauses**
- And then continues. . .
- **Until the malicious VM sends the next pause frame**

The Attack

- The malicious VM sends a pause frame
- **All traffic on the shared link pauses**
- And then continues. . .
- **Until the malicious VM sends the next pause frame**

The Attack

- The malicious VM sends a pause frame
- **All traffic on the shared link pauses**
- And then continues. . .
- **Until the malicious VM sends the next pause frame**

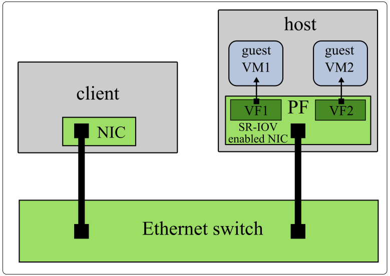
The Attack

- The malicious VM sends a pause frame
- **All traffic on the shared link pauses**
- And then continues. . .
- **Until the malicious VM sends the next pause frame**

Attack Evaluation—Setup

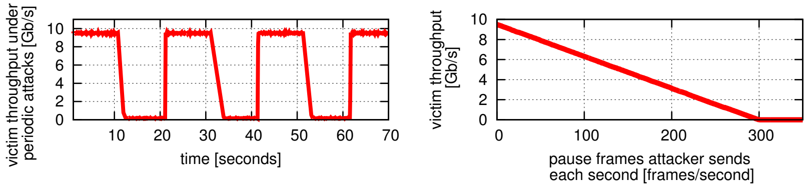
Our testbed consists of two identical servers: one acting as client and the other as the host with SRIOV capable NIC

- On host VF1 assigned to guest VM1 and VF2 to guest VM2
- traffic generated between VM1 and the client using `iperf` and `netperf`
- VM2 is the attacking VM1 sending generated PAUSE frames with `tcpreplay`

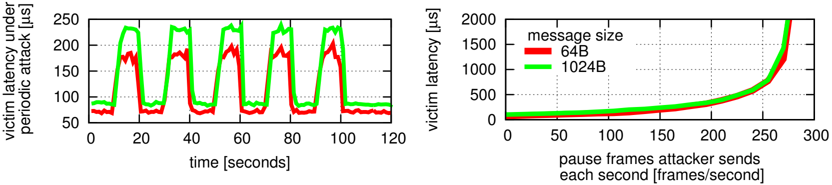


Setup scheme

Attack Results using Intel 10GbE NIC



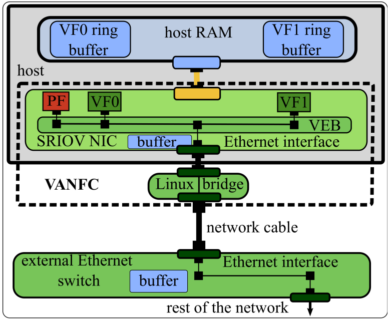
Pause frame attack: victim throughput in 10GbE environment



Pause frame attack: victim latency in 10GbE environment

The Virtualization-Aware Network Flow Controller design

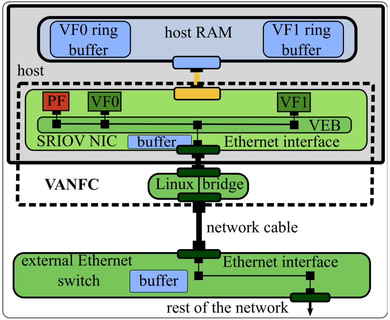
- **Filter** outbound traffic transmitted by a VF
- **Internal switch** replicates Ethernet switch
- **All valid** pause frames are generated by the NIC's hardware and have the PF's source MAC address
- **All malicious** pause frames are sent with source address of a VF



Schema of VANFC

The Virtualization-Aware Network Flow Controller design

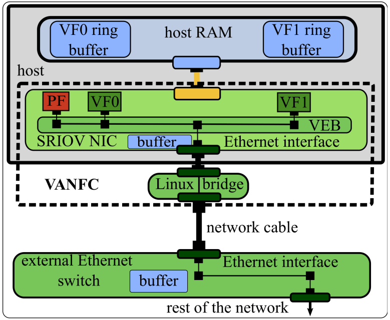
- **Filter** outbound traffic transmitted by a VF
- **Internal switch** replicates Ethernet switch
- **All valid** pause frames are generated by the NIC's hardware and have the PF's source MAC address
- **All malicious** pause frames are sent with source address of a VF



Schema of VANFC

The Virtualization-Aware Network Flow Controller design

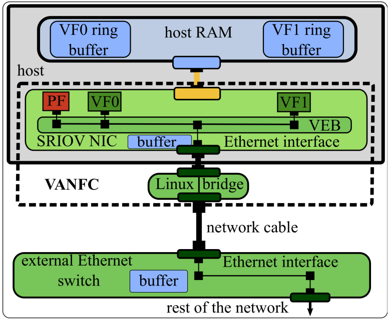
- **Filter** outbound traffic transmitted by a VF
- **Internal switch** replicates Ethernet switch
- **All valid** pause frames are generated by the NIC's hardware and have the PF's source MAC address
- **All malicious** pause frames are sent with source address of a VF



Schema of VANFC

The Virtualization-Aware Network Flow Controller design

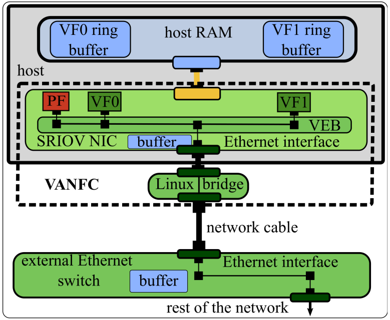
- **Filter** outbound traffic transmitted by a VF
- **Internal switch** replicates Ethernet switch
- **All valid** pause frames are generated by the NIC's hardware and have the PF's source MAC address
- **All malicious** pause frames are sent with source address of a VF



Schema of VANFC

The Virtualization-Aware Network Flow Controller design

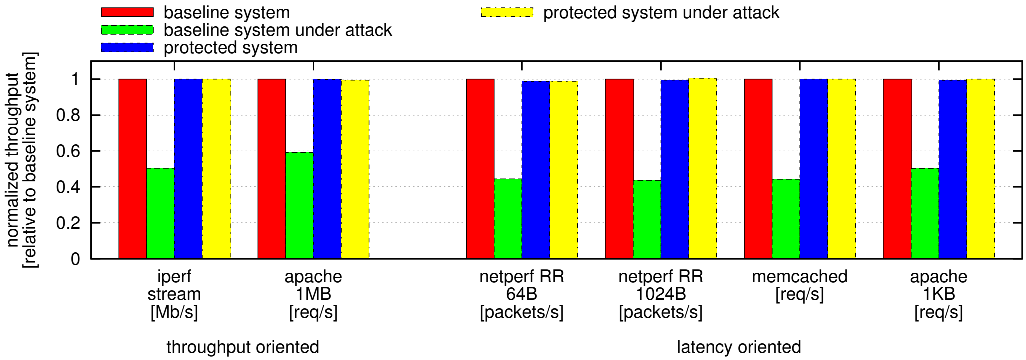
- **Filter** outbound traffic transmitted by a VF
- **Internal switch** replicates Ethernet switch
- **All valid** pause frames are generated by the NIC's hardware and have the PF's source MAC address
- **All malicious** pause frames are sent with source address of a VF



Schema of VANFC

Evaluating VANFC

VANFC completely blocks VM2's attack and introduces no performance penalty



VANFC performance evaluation results

Conclusions

- SRIOV, as currently deployed on current Ethernet networks, is incompatible with flow control
- Removing host from the I/O path requires adding functionality to the hardware
- VANFC 100% effective in securing SRIOV against this flaw while imposing no overhead on throughput or latency
- Future work:
  - Extend to SRIOV InfiniBand and Fiber Channel, NVMe SSD and GPGU
  - Develop VF co-residency detection techniques
  - Use the hypervisor to solve the problem of VM ring buffer exhaustion

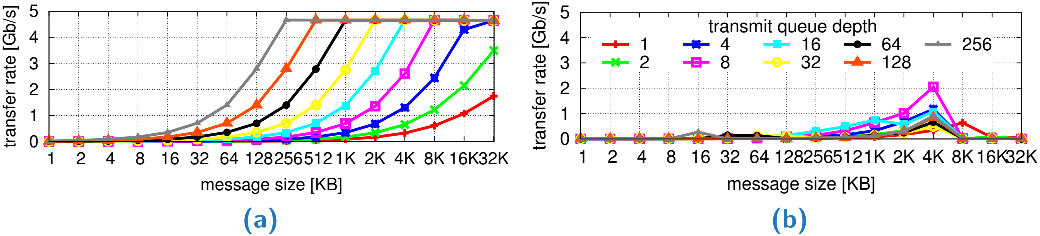
Thank You

Questions?

Can SRIOV be "secured" by disabling FC?

- TCP has its own flow control; however
  - relying on TCP alone for flow control leads to increased resource utilization
  - higher CPU utilization results in higher charges
  - TCP `incast` problem requires flow control
- Remote DMA over Converged Ethernet (RoCE) significantly reduces CPU utilization when compared with TCP
  - Kissel et al. show that on 40 GbE link, sender CPU utilization reduced from 100% using TCP to 2% using RoCE
  - Kissel et al. also show that the same problem is relevant not only to RoCE but can be generalized to TCP as well

Performance of a single RoCE flow in the system with two competing RoCE flows[1]



Graph (a) shows performance with enabled flow control; graph (b) shows performance with disabled flow control.

[1] Taken from Kissel et al. with the authors' explicit permission