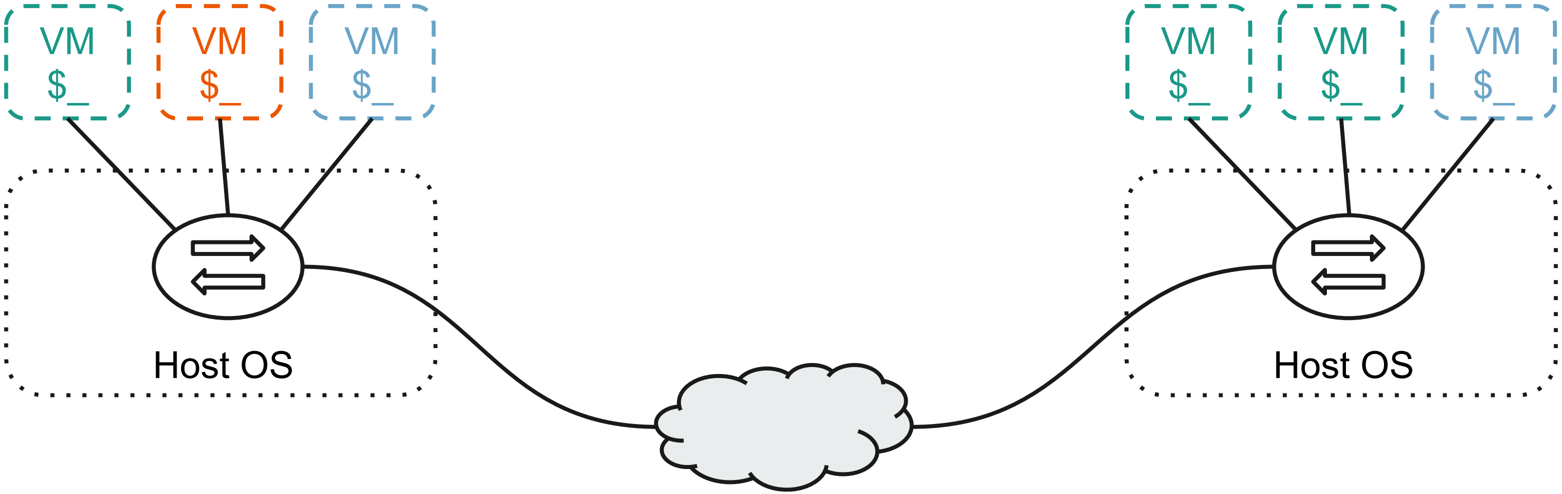
MTS: Bringing Multi-Tenancy to Virtual Networking

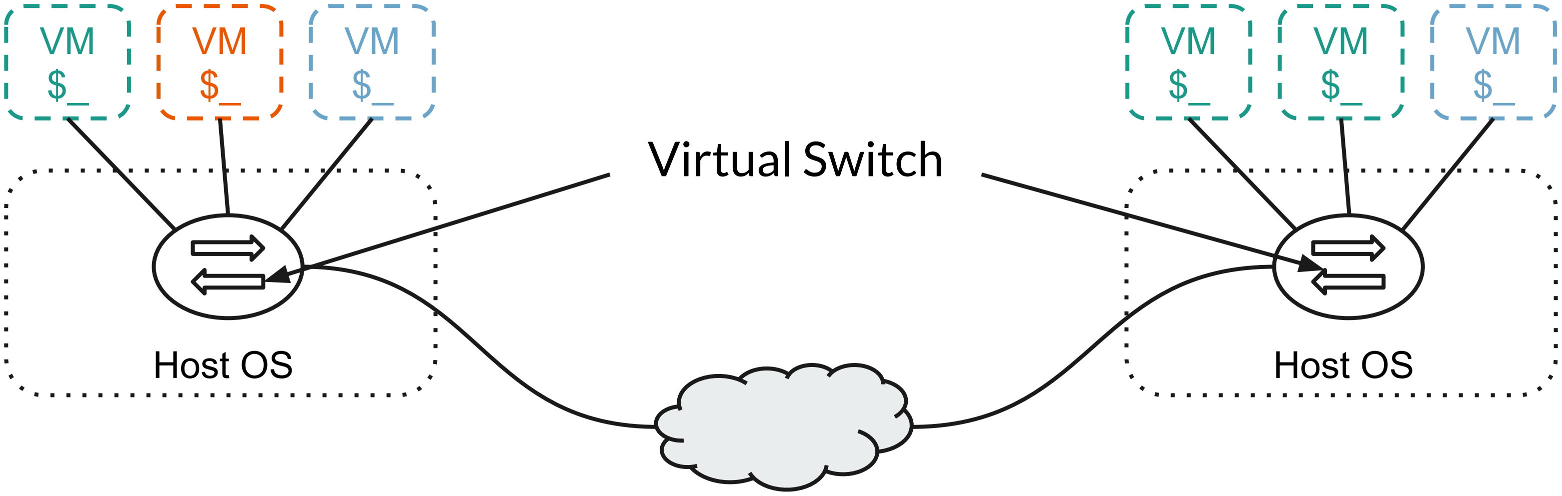
Kashyap Thimmaraju, Saad Hermak, Gábor Rétvári and Stefan Schmid

USENIX Annual Technical Conference 2019
July 11, Renton, Washington, USA

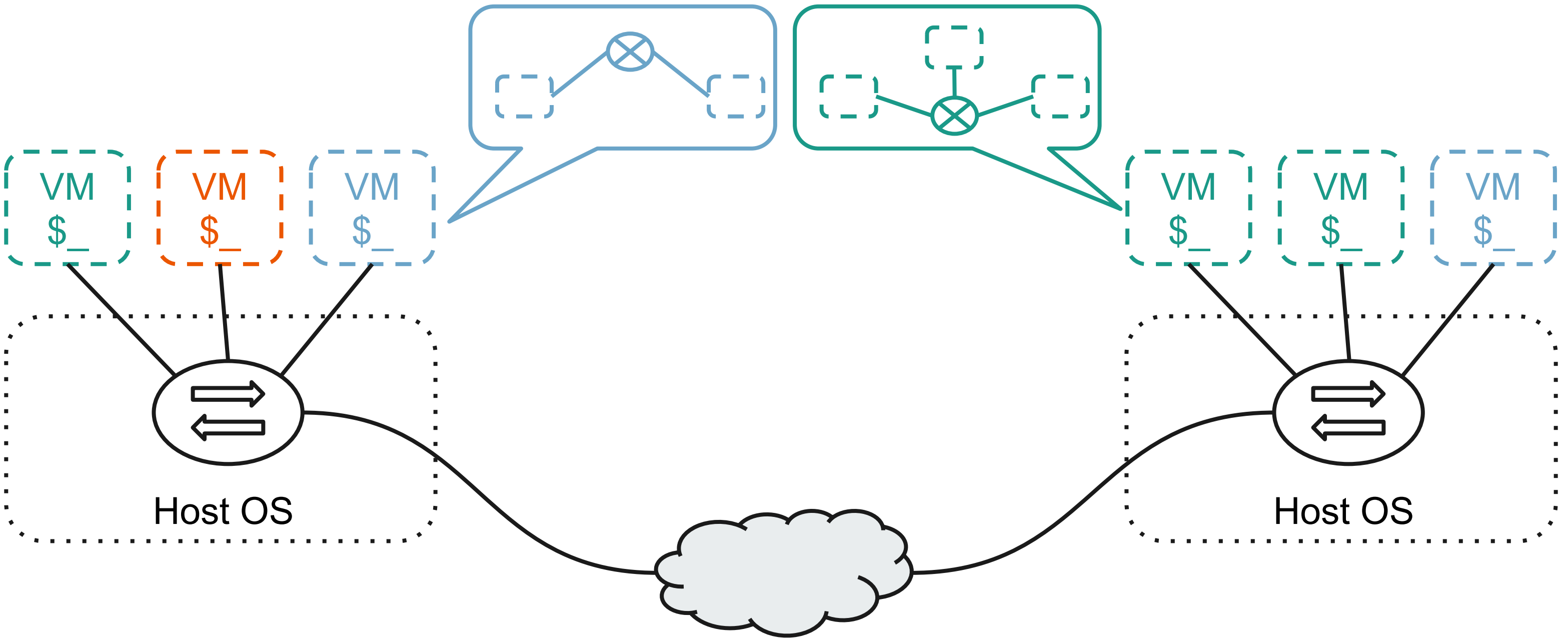
Virtual Networks Using Virtual Switches



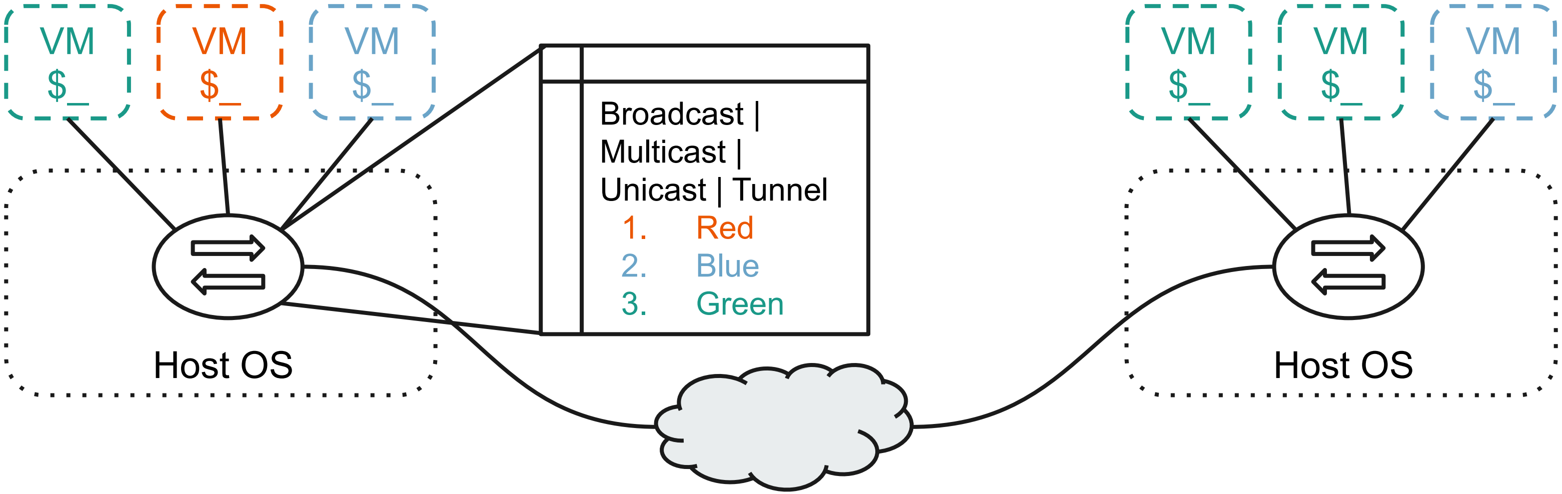
Virtual Networks Using Virtual Switches



Virtual Networks Using Virtual Switches



Virtual Networks Using Virtual Switches



More Than 20 Virtual Switches

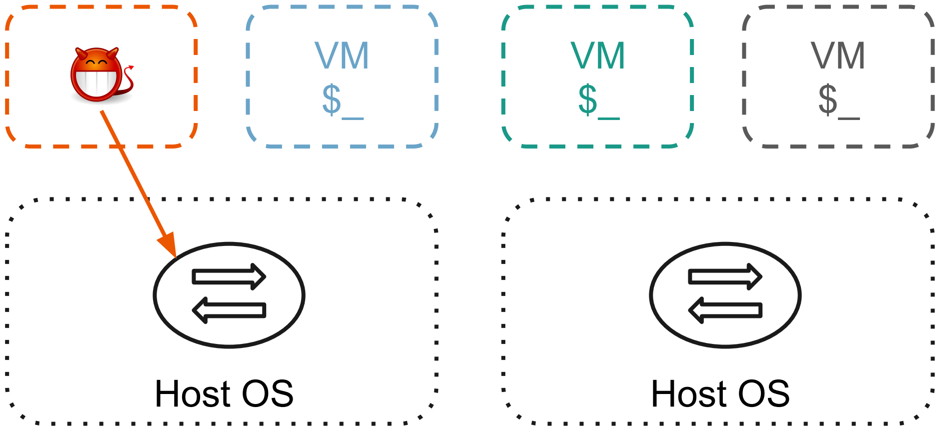
Most emphasis has been on performance and flexibility

Name	Year	Emphasis	Monolithic	Co-Location	Kernel	User
OvS	2009	Flexibility	✓	✓	✓	✓
Cisco NexusV	2009	Flexibility	✓	✓	✓	✗
VMware vSwitch	2009	Centralized control	✓	✓	✓	✗
Vale	2012	Performance	✓	✓	✓	✗
Research prototype	2012	Isolation	✓	✗	✓	✓
Hyper-Switch	2013	Performance	✓	✓	✓	✓
MS HyperV-Switch	2013	Centralized control	✓	✓	✓	✗
NetVM	2014	Performance, NFV	✓	✓	✗	✓
sv3	2014	Security	✗	✓	✗	✓
fd.io	2015	Performance	✓	✓	✗	✓
mSwitch	2015	Performance	✓	✓	✓	✗
BESS	2015	Programmability, NFV	✓	✓	✗	✓
PISCES	2016	Programmability	✓	✓	✓	✓
OvS with DPDK	2016	Performance	✓	✓	✗	✓
ESwitch	2016	Performance	✓	✓	✗	✓
MS VFP	2017	Performance, flexibility	✓	✓	✓	✗
Mellanox BlueField	2017	CPU offload	✓	✗	✓	✓
Liquid IO	2017	CPU offload	✓	✗	✓	✓
Stingray	2017	CPU offload	✓	✗	✓	✓
GPU-based OvS	2017	Acceleration	✓	✓	✓	✓
MS AccelNet	2018	Performance, flexibility	✓	✓	✓	✗
Google Andromeda	2018	Flexibility and performance	✓	✓	✗	✓

Security Weaknesses of Virtual Switches

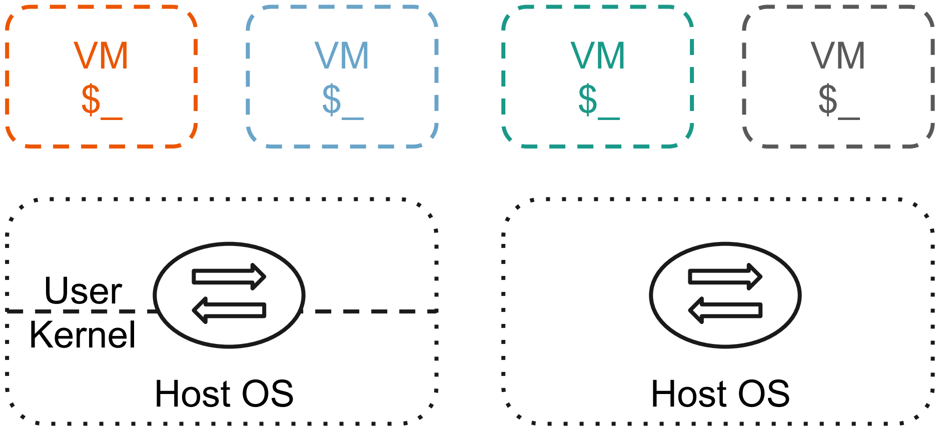
Processes Untrusted Data

A malicious VM can send arbitrary packets to the virtual switch



Privileged Packet Processing

Oftentimes runs in the kernel for performance



Single Point of Failure

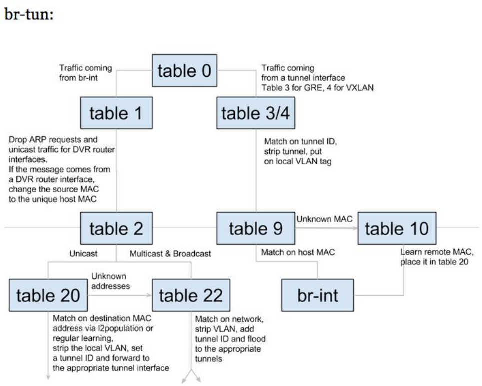
Virtual network configurations are complex

```
NXST_FLOW reply (xid=0x4):
 cookie=0x9e71d200b05c476d, duration=9483.262s, table=0, n_packets=68, n_bytes=5430, idle_age=768, priority=1,in_port=1 actions=resubmit(,2)
 cookie=0x9e71d200b05c476d, duration=9483.262s, table=0, n_packets=0, n_bytes=0, idle_age=9483, priority=0 actions=drop
 cookie=0x9e71d200b05c476d, duration=9483.261s, table=2, n_packets=21, n_bytes=1080, idle_age=768, priority=1,arp,dl_dst=ff:ff:ff:ff:ff:ff acti
ons=resubmit(,21)
 cookie=0x9e71d200b05c476d, duration=9483.260s, table=2, n_packets=12, n_bytes=1052, idle_age=768, priority=0,dl_dst=00:00:00:00:00:00/01:00:0
0:00:00:00 actions=resubmit(,20)
 cookie=0x9e71d200b05c476d, duration=9483.259s, table=2, n_packets=35, n_bytes=3298, idle_age=2458, priority=0,dl_dst=01:00:00:00:00:00/01:00:0
0:00:00:00 actions=resubmit(,22)
 cookie=0x9e71d200b05c476d, duration=9483.259s, table=3, n_packets=0, n_bytes=0, idle_age=9483, priority=0 actions=drop
 cookie=0x9e71d200b05c476d, duration=9282.478s, table=4, n_packets=0, n_bytes=0, idle_age=9282, priority=1,tun_id=0xf actions=mod_vlan_vid:1,re
submit(,10)
 cookie=0x9e71d200b05c476d, duration=9483.258s, table=4, n_packets=0, n_bytes=0, idle_age=9483, priority=0 actions=drop
 cookie=0x9e71d200b05c476d, duration=9483.258s, table=6, n_packets=0, n_bytes=0, idle_age=9483, priority=0 actions=drop
 cookie=0x9e71d200b05c476d, duration=9483.257s, table=10, n_packets=0, n_bytes=0, idle_age=9483, priority=1 actions=learn(table=20,hard_timeout
=300,priority=1,cookie=0x9e71d200b05c476d,NXM_OF_VLAN_TCI[0..11],NXM_OF_ETH_DST[]=NXM_OF_ETH_SRC[],load:0->NXM_OF_VLAN_TCI[],load:NXM_NX_TUN_ID
[]->NXM_NX_TUN_ID[],output:OXM_OF_IN_PORT[]),output:1
 cookie=0x9e71d200b05c476d, duration=9483.257s, table=20, n_packets=12, n_bytes=1052, idle_age=768, priority=0 actions=resubmit(,22)
 cookie=0x9e71d200b05c476d, duration=9483.256s, table=21, n_packets=21, n_bytes=1080, idle_age=768, priority=0 actions=resubmit(,22)
 cookie=0x9e71d200b05c476d, duration=9483.256s, table=22, n_packets=68, n_bytes=5430, idle_age=768, priority=0 actions=drop
```

I found a couple of sites that talk about the **Openflow** tables:

- Distributed Virtual Routing – Overview and East/West Routing
- Openstack Neutron using VXLAN
- Troubleshooting OpenStack Neutron Networking, Part One

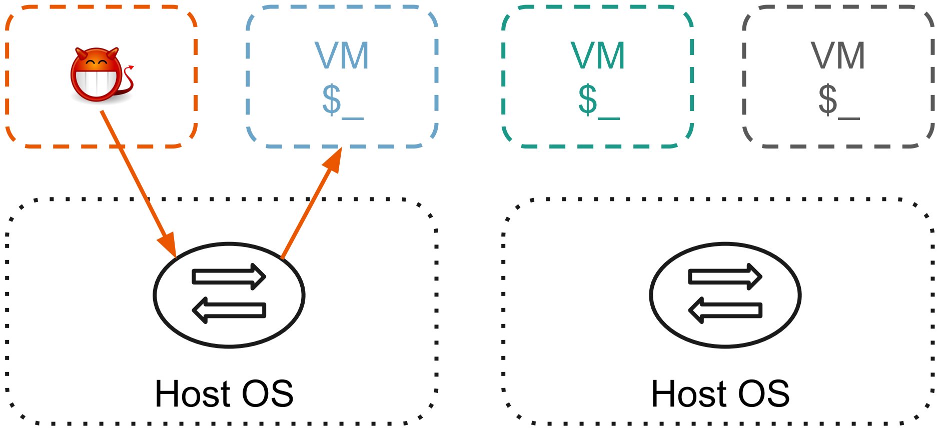
One of the above page goes over the tables in great detail and actually has a nice diagram:



Screenshot from Karim Elatov's blog:
https://elatov.github.io/2018/01/openstack-ansible-and-kolla-on-ubuntu-1604/#5-packet-goes-from-ovs-integration-bridge-br-int-to-ovs-tunnel-bridge-br-tun

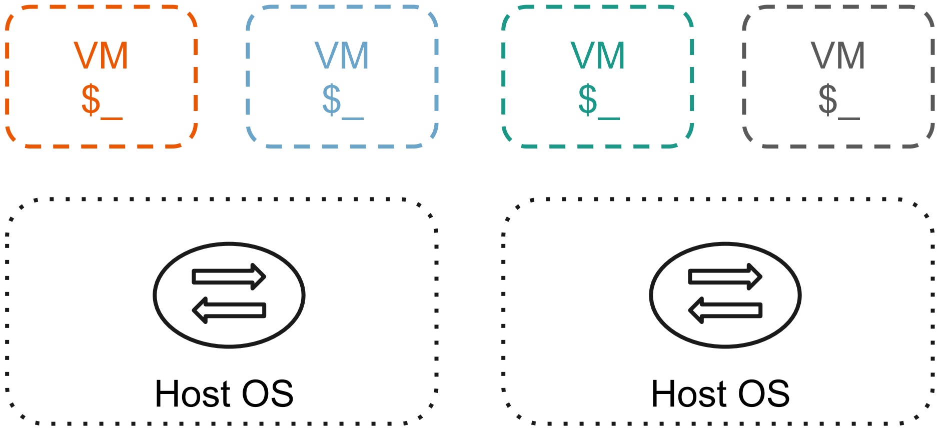
Single Point of Failure

Mis-configurations could lead to security issues



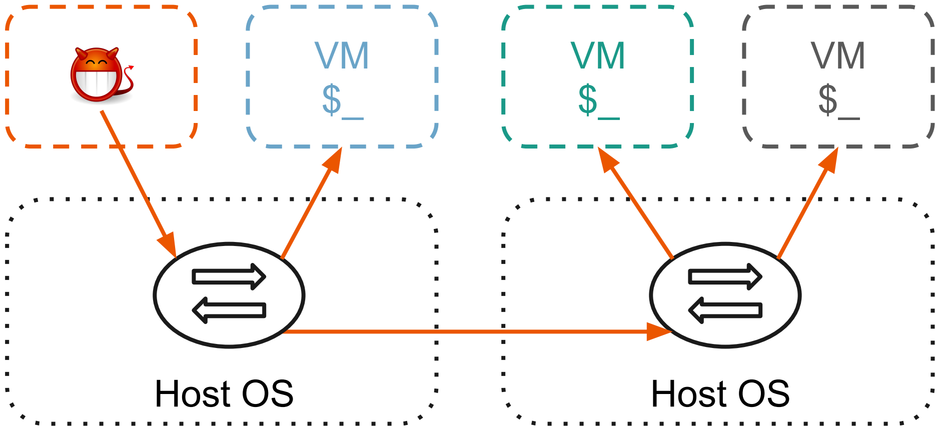
Co-Located with the Host OS

The consequence of a compromise can be severe, e.g., break out of VM isolation



Exploiting Virtual Switches in the Cloud

SOSR'18: Remote-Code Exection
OvS Con'19: Cross Tenant DoS

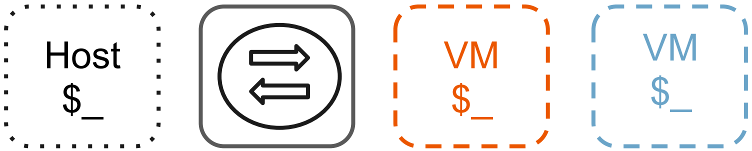


Outline

- ~~Motivation~~
- MTS
- Evaluation
- Scalability
- Pros and Cons
- Conclusion

MTS: Multi-Tenant Switch

Least Privilege Virtual Switch



1. Processes untrusted data
2. Privileged packet processing
3. Single point of failure
4. ~~Co-located with the Host OS~~

Least Common Mechanism

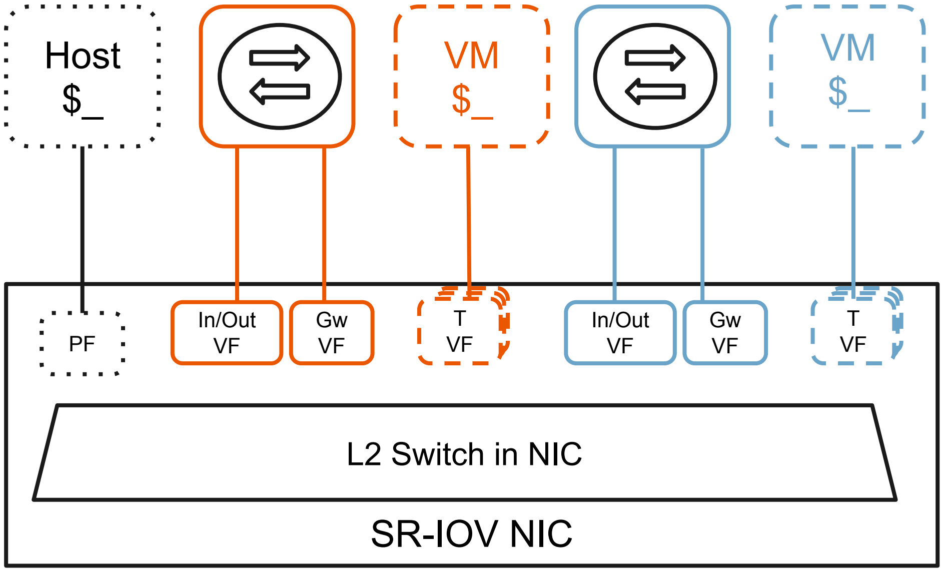
1. Processes untrusted data
2. Privileged packet processing
3. ~~Single point of failure~~
4. ~~Co-located with the Host OS~~

Extra Security Boundary

1. Processes untrusted data
2. ~~Privileged packet processing~~
3. ~~Single point of failure~~
4. ~~Co-located with the Host OS~~

Complete Mediation

1. Processes untrusted data
2. ~~Privileged packet processing~~
3. ~~Single point of failure~~
4. ~~Co-located with the Host OS~~



Evaluation

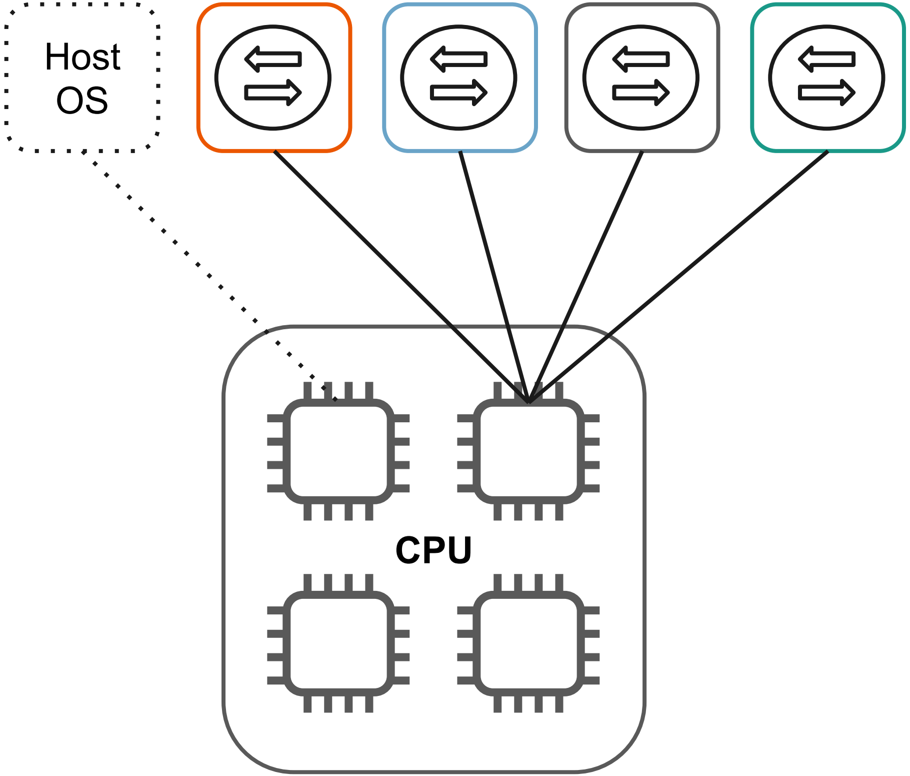
Experimental Setup & Factors

Mellanox ConnectX4, Open vSwitch, DPDK, QEMU, KVM
More details in the paper

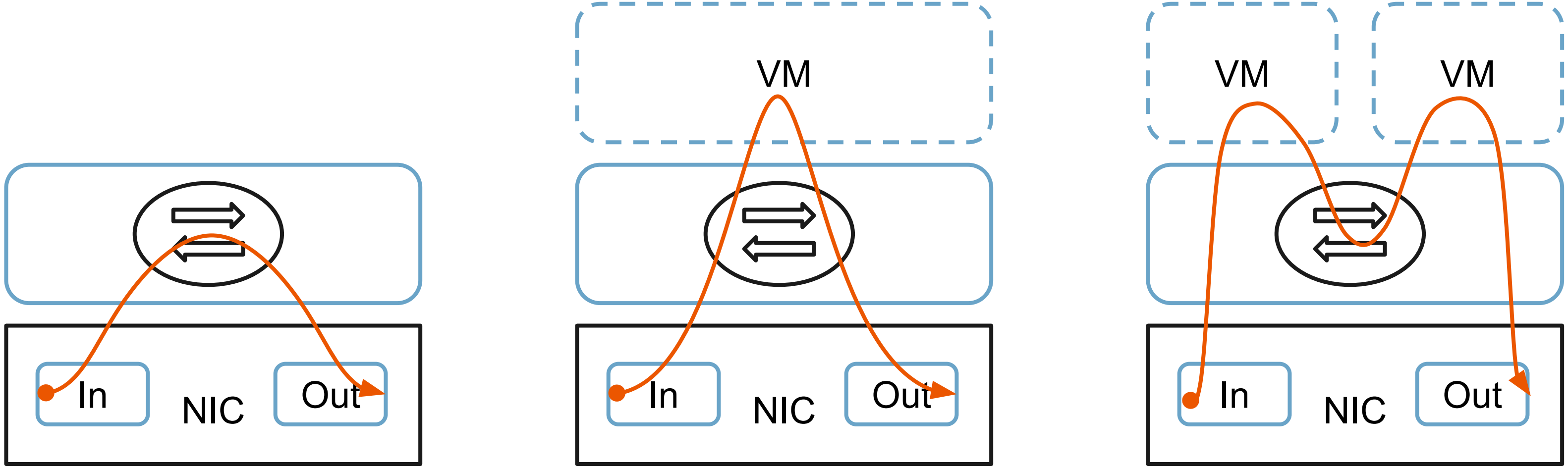
- Resources
- Traffic Patterns

Shared Resources

- Host OS pinned to 1 core
- All vswitch-VMs pinned to 1 core
- Each Tenant VM got dedicated cores (not shown here)



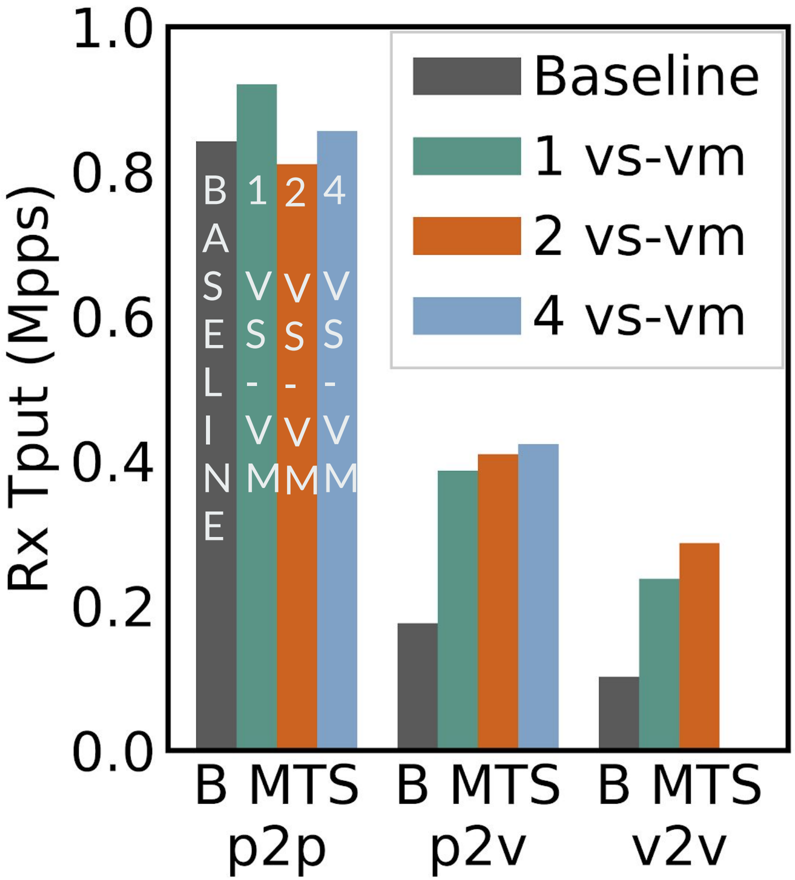
Traffic Patterns



p2p p2v v2v

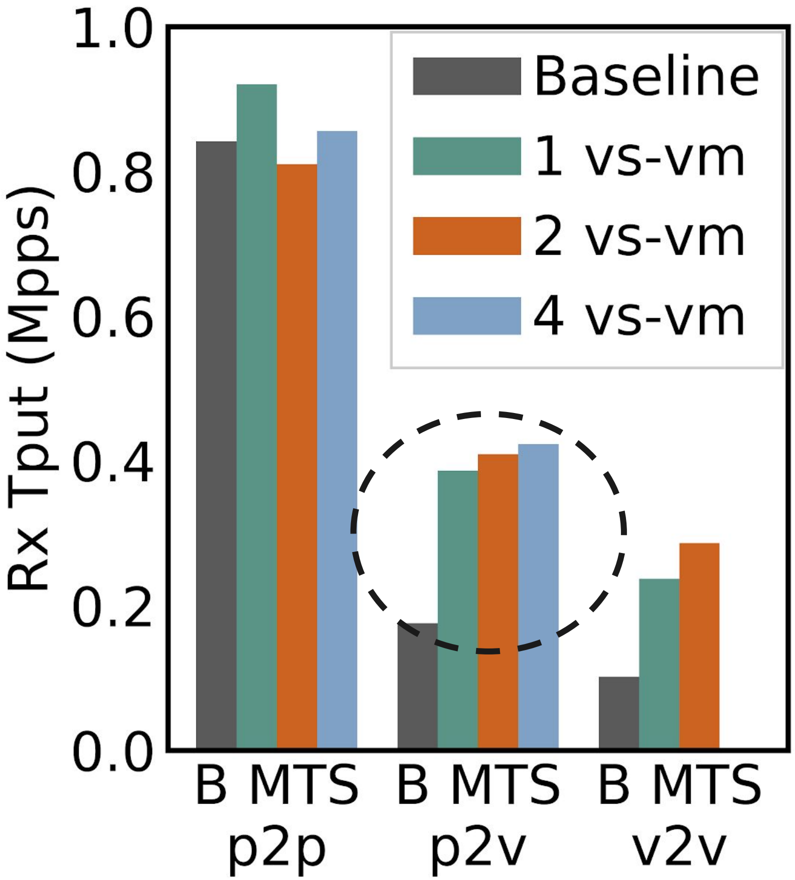
Baseline vs MTS Packet Processing Throughput Comparison

64 byte UDP packets
Roughly the same in p2p
MTS is ~2x Baseline in p2v and v2v



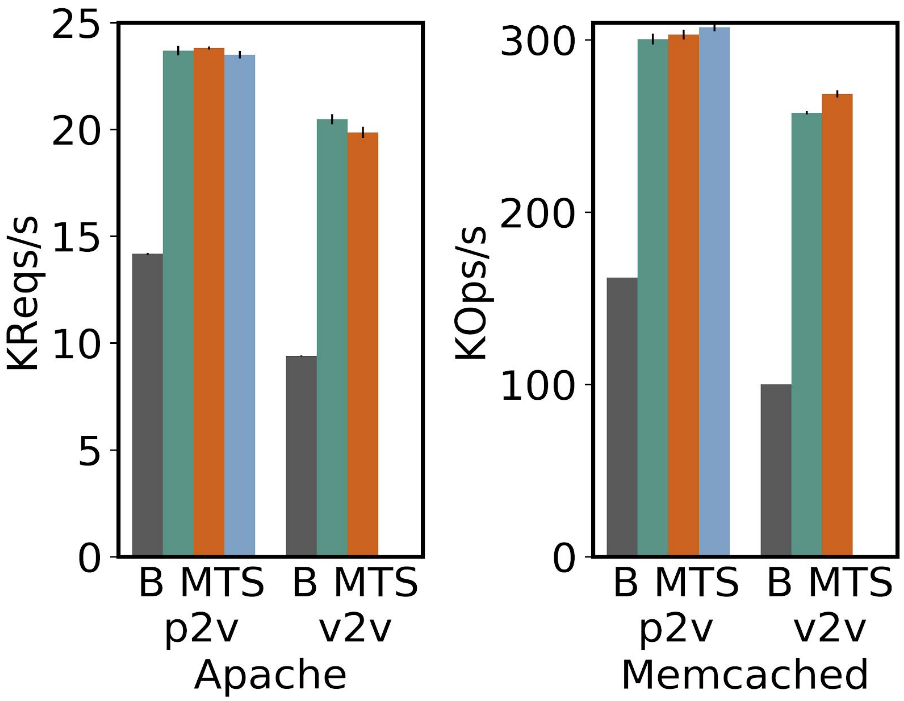
Baseline vs MTS Packet Processing Throughput Comparison

64 byte UDP packets
Roughly the same in p2p
MTS is ~2x Baseline in p2v and v2v



Baseline vs MTS Network Application Throughput

MTS beats Baseline in Apache and Memcached



1+ Physical Core
4x Network Isolation
1.5-2x Throughput

Scaling MTS

Containers in VMs

Real cloud systems can host more than just 4 tenants on a server

- Work in progress
- The packets per second throughput is the same as running it in a VM for 4 containers
- Can run 12 vswitches spread across 4 VMs
- Faced an issue with libvirt when adding 40 VFs to 16 vswitches spread across 4 VMs. The interfaces do not appear in the VM although the configuration is present.

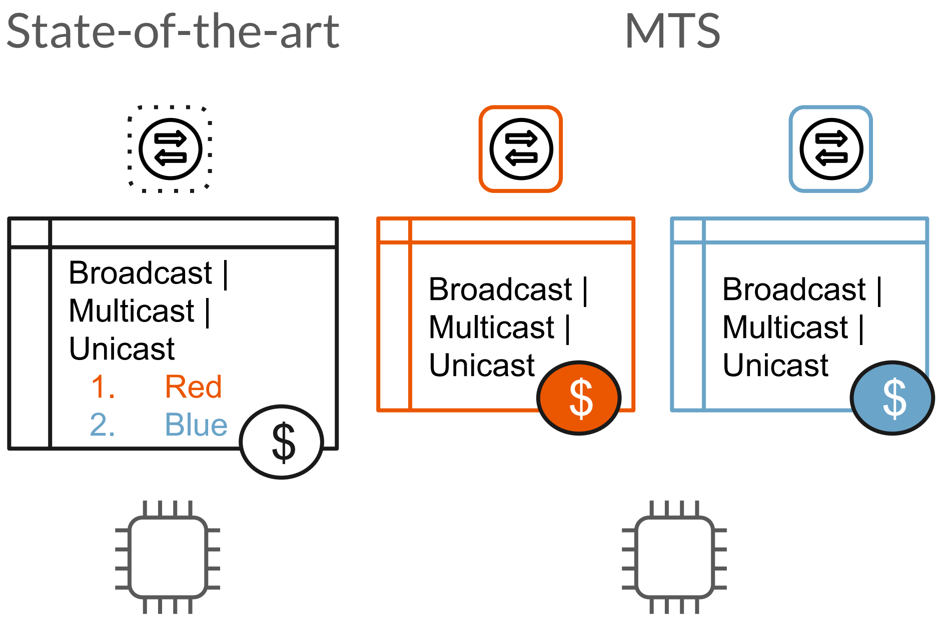
Pros and Cons

Limitations

- PCIe bus could become a bottleneck which our evaluation did not reveal
- The number of VFs on the NIC
- No clean solution for live migration of VMs with VFs

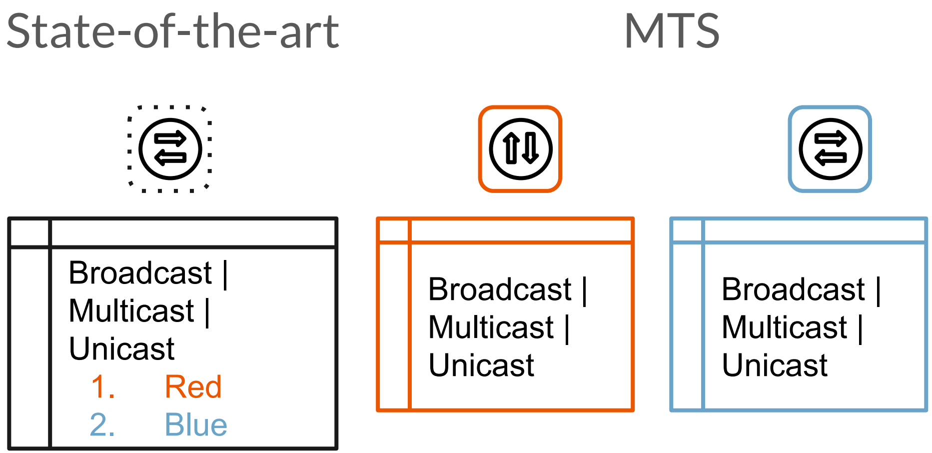
Pricing

Charge for CPU cycles used by the tenant-specific virtual switch



Tenant Specific Virtual Switch Software

1. Reduce parsing logic
2. Support tenant-specific features

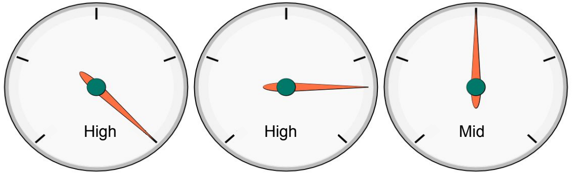


Conclusion

Key Takeaways

Our scripts and data are on github
www.github.com/securedataplane

1. Many virtual switches can be exploited to compromise Host and Network isolation
2. MTS is based on secure design principles that addresses security weakness of existing designs
3. MTS with SR-IOV offers security and performance for modest resources



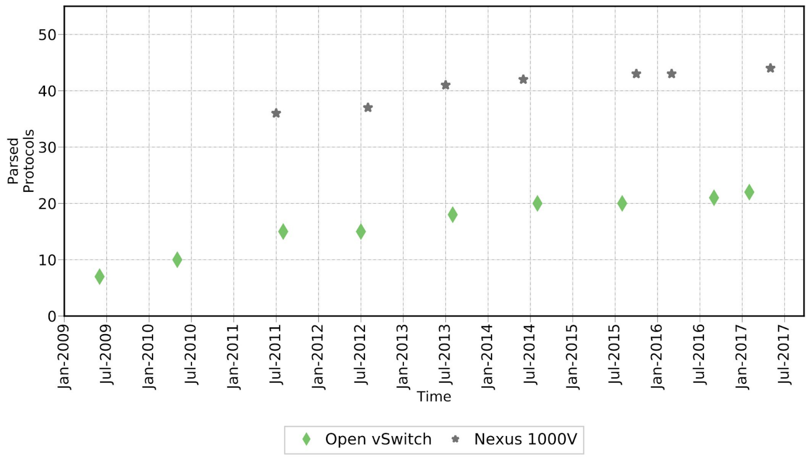
Security Performance Resource

Backup

Protocol Growth for OvS

Complex & Manual Protocol Parsers

Virtual switches have to support an increasing number of protocols over time



Vswitch Table Analysis

So Many Virtual Switches

More than 20

Name	Year	Emphasis	Monolithic	Co-Location	Kernel	User
OvS	2009	Flexibility	✓	✓	✓	✓
Cisco NexusV	2009	Flexibility	✓	✓	✓	✗
VMware vSwitch	2009	Centralized control	✓	✓	✓	✗
Vale	2012	Performance	✓	✓	✓	✗
Research prototype	2012	Isolation	✓	✗	✓	✓
Hyper-Switch	2013	Performance	✓	✓	✓	✓
MS HyperV-Switch	2013	Centralized control	✓	✓	✓	✗
NetVM	2014	Performance, NFV	✓	✓	✗	✓
sv3	2014	Security	✗	✓	✗	✓
fd.io	2015	Performance	✓	✓	✗	✓
mSwitch	2015	Performance	✓	✓	✓	✗
BESS	2015	Programmability, NFV	✓	✓	✗	✓
PISCES	2016	Programmability	✓	✓	✓	✓
OvS with DPDK	2016	Performance	✓	✓	✗	✓
ESwitch	2016	Performance	✓	✓	✗	✓
MS VFP	2017	Performance, flexibility	✓	✓	✓	✗
Mellanox BlueField	2017	CPU offload	✓	✗	✓	✓
Liquid IO	2017	CPU offload	✓	✗	✓	✓
Stingray	2017	CPU offload	✓	✗	✓	✓
GPU-based OvS	2017	Acceleration	✓	✓	✓	✓
MS AccelNet	2018	Performance, flexibility	✓	✓	✓	✗
Google Andromeda	2018	Flexibility and performance	✓	✓	✗	✓

So Many Virtual Switches

More than 20

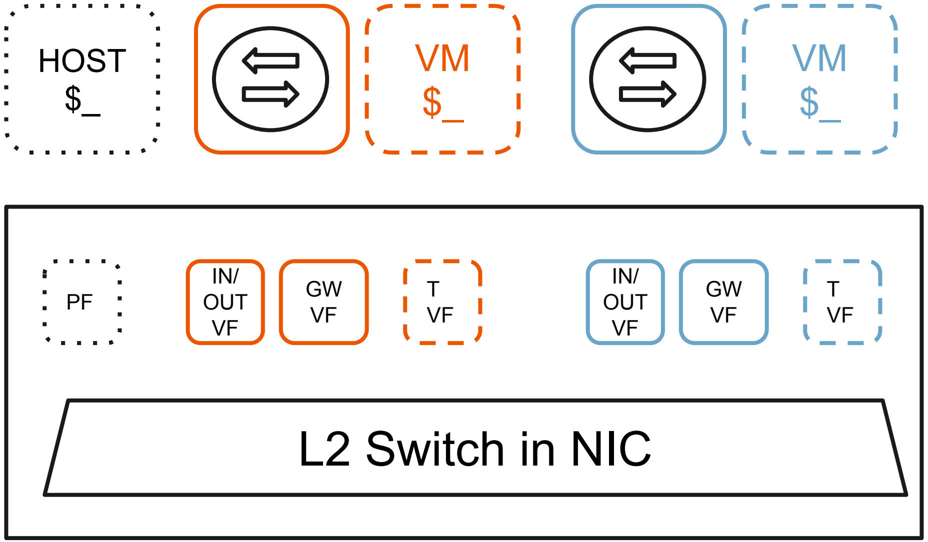
Name	Year	Emphasis	Monolithic	Co-Location	Kernel	User
OvS	2009	Flexibility	✓	✓	✓	✓
Cisco NexusV	2009	Flexibility	✓	✓	✓	✗
VMware vSwitch	2009	Centralized control	✓	✓	✓	✗
Vale	2012	Performance	✓	✓	✓	✗
Research prototype	2012	Isolation	✓	✗	✓	✓
Hyper-Switch	2013	Performance	✓	✓	✓	✓
MS HyperV-Switch	2013	Centralized control	✓	✓	✓	✗
NetVM	2014	Performance, NFV	✓	✓	✗	✓
sv3	2014	Security	✗	✓	✗	✓
fd.io	2015	Performance	✓	✓	✗	✓
mSwitch	2015	Performance	✓	✓	✓	✗
BESS	2015	Programmability, NFV	✓	✓	✗	✓
PISCES	2016	Programmability	✓	✓	✓	✓
OvS with DPDK	2016	Performance	✓	✓	✗	✓
ESwitch	2016	Performance	✓	✓	✗	✓
MS VFP	2017	Performance, flexibility	✓	✓	✓	✗
Mellanox BlueField	2017	CPU offload	✓	✗	✓	✓
Liquid IO	2017	CPU offload	✓	✗	✓	✓
Stingray	2017	CPU offload	✓	✗	✓	✓
GPU-based OvS	2017	Acceleration	✓	✓	✓	✓
MS AccelNet	2018	Performance, flexibility	✓	✓	✓	✗
Google Andromeda	2018	Flexibility and performance	✓	✓	✗	✓

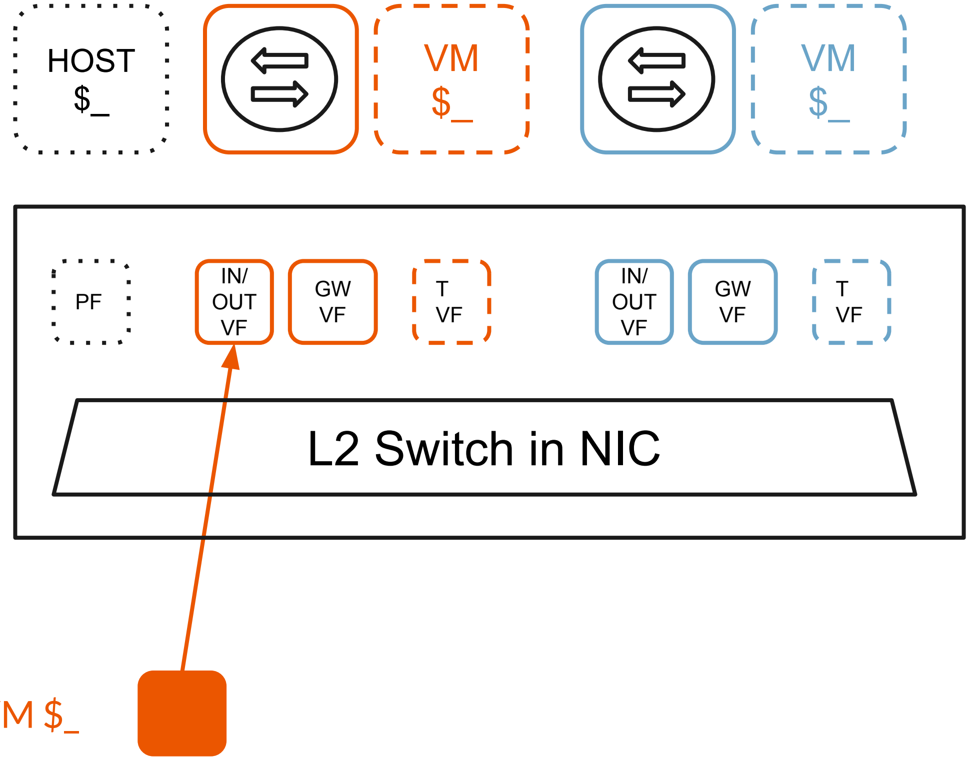
So Many Virtual Switches

More than 20

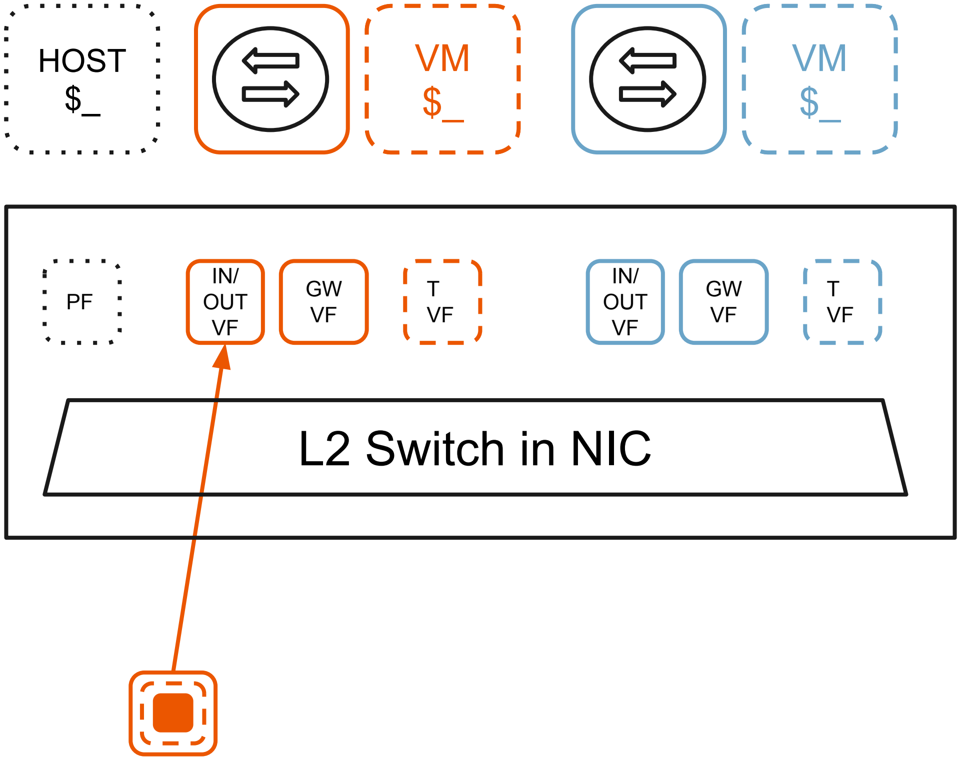
Name	Year	Emphasis	Monolithic	Co-Location	Kernel	User
OvS	2009	Flexibility	✓	✓	✓	✓
Cisco NexusV	2009	Flexibility	✓	✓	✓	✗
VMware vSwitch	2009	Centralized control	✓	✓	✓	✗
Vale	2012	Performance	✓	✓	✓	✗
Research prototype	2012	Isolation	✓	✗	✓	✓
Hyper-Switch	2013	Performance	✓	✓	✓	✓
MS HyperV-Switch	2013	Centralized control	✓	✓	✓	✗
NetVM	2014	Performance, NFV	✓	✓	✗	✓
sv3	2014	Security	✗	✓	✗	✓
fd.io	2015	Performance	✓	✓	✗	✓
mSwitch	2015	Performance	✓	✓	✓	✗
BESS	2015	Programmability, NFV	✓	✓	✗	✓
PISCES	2016	Programmability	✓	✓	✓	✓
OvS with DPDK	2016	Performance	✓	✓	✗	✓
ESwitch	2016	Performance	✓	✓	✗	✓
MS VFP	2017	Performance, flexibility	✓	✓	✓	✗
Mellanox BlueField	2017	CPU offload	✓	✗	[illegible]	✓
Liquid IO	2017	CPU offload	✓	✗	✓	✓
Stingray	2017	CPU offload	✓	✗	✓	✓
GPU-based OvS	2017	Acceleration	✓	✓	✓	✓
MS AccelNet	2018	Performance, flexibility	✓	✓	✓	✗
Google Andromeda	2018	Flexibility and performance	✓	✓	✗	[illegible]

Ingress Traffic Flow Example


HOST
$_
VM
$_
VM
$_
PF
IN/
OUT
VF
GW
VF
T
VF
IN/
OUT
VF
GW
VF
T
VF
L2 Switch in NIC

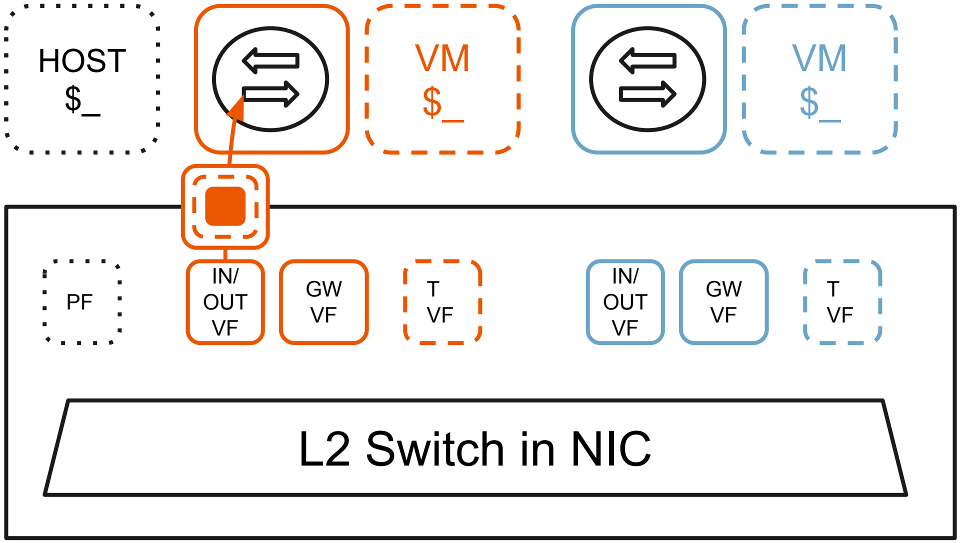


Packet destined to VM $_

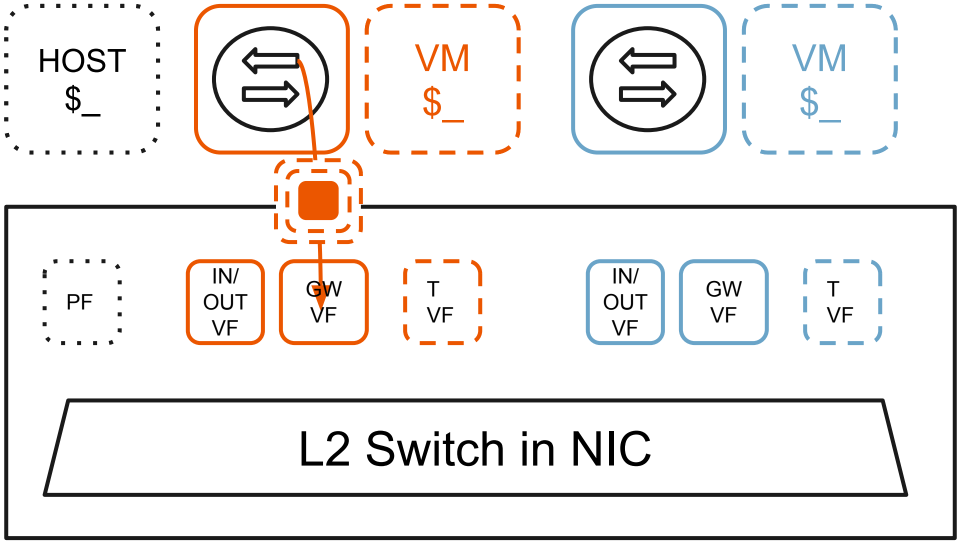


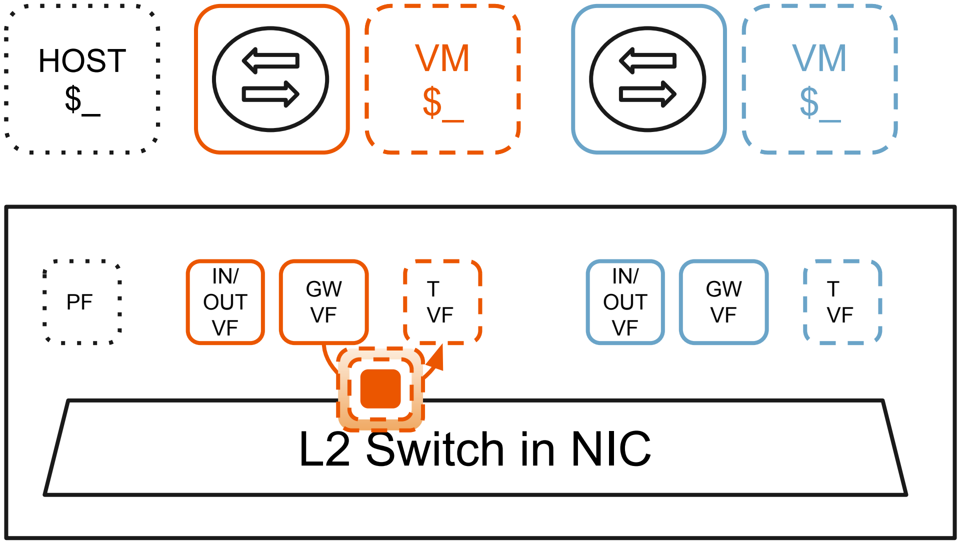
MAC address of the
vswitch VF

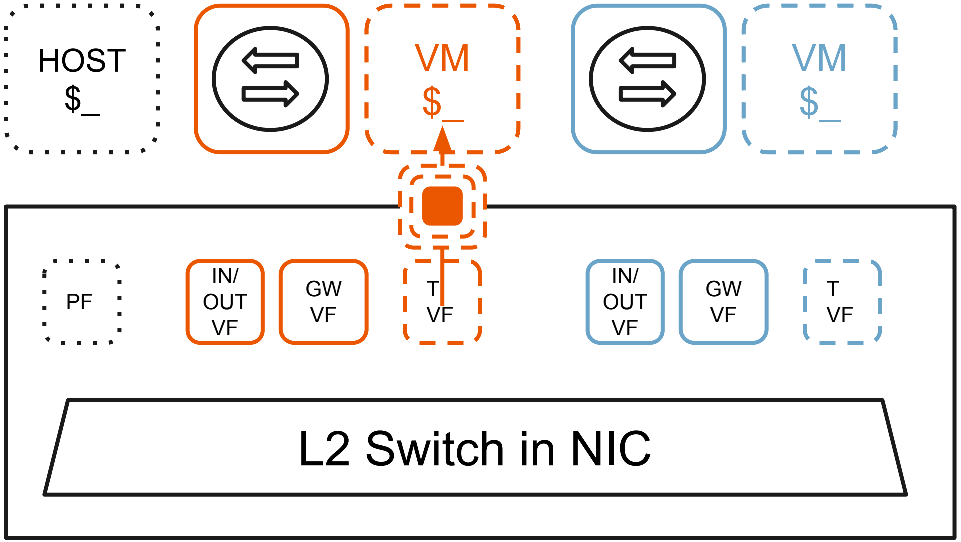
IP address of VM $_

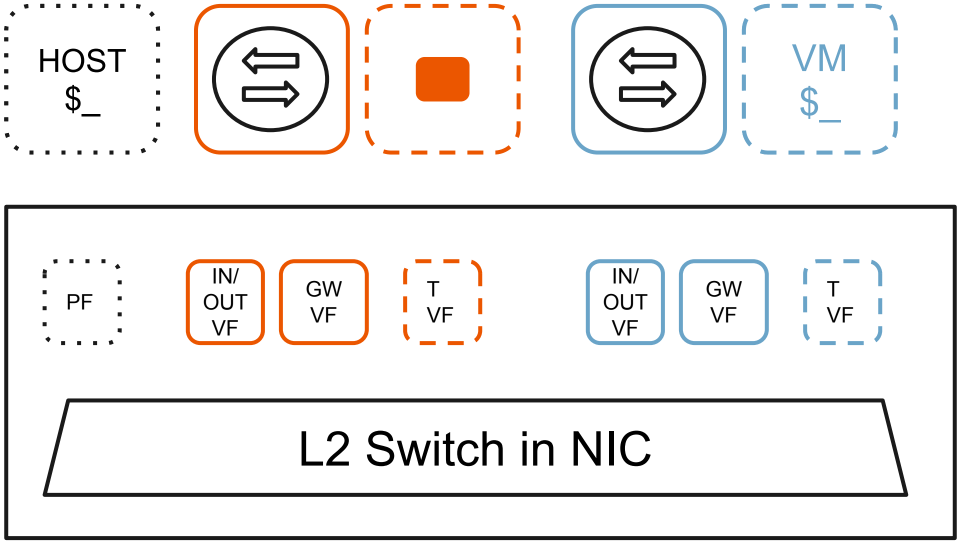


HOST
$_
VM
$_
VM
$_
PF
IN/
OUT
VF
GW
VF
T
VF
IN/
OUT
VF
GW
VF
T
VF
L2 Switch in NIC


HOST
$_
VM
$_
VM
$_
PF
IN/
OUT
VF
GW
VF
T
VF
IN/
OUT
VF
GW
VF
T
VF
L2 Switch in NIC


HOST
$_
VM
$_
VM
$_
PF
IN/
OUT
VF
GW
VF
T
VF
IN/
OUT
VF
GW
VF
T
VF
L2 Switch in NIC


HOST
$_
VM
$_
VM
$_
PF
IN/
OUT
VF
GW
VF
T
VF
IN/
OUT
VF
GW
VF
T
VF
L2 Switch in NIC


HOST
$_
VM
$_
PF
IN/
OUT
VF
GW
VF
T
VF
IN/
OUT
VF
GW
VF
T
VF
L2 Switch in NIC

Pricing

How it Helps Pricing

Can charge for compute and memory used by the vswitch

Google Cloud Why Google Solutions Products Pricing Getting started Docs Support Language Console

Compute Products Contact sales

Compute Engine
Product overview
Documentation

Quickstarts
All quickstarts
Using a Linux VM
Using a Windows VM

How-to guides
All how-to guides
Creating VM instances
Managing access to VM instances
Connecting to VM instances
Adding storage
Creating and managing instance templates
Creating and managing custom images
Managing your instances
Creating and managing groups of

Network pricing

General network pricing

Traffic type	Price
Ingress	No charge, unless there is a resource such as a load balancer that is processing ingress traffic. Responses to requests count as egress and are charged.
Egress[1] to the same zone	No charge
Egress to Google products (such as YouTube, Maps, Drive), whether from a VM in GCP with an external IP address or an internal IP address	No charge
Egress to DoubleClick in the same region	No charge
Egress to a different Google Cloud Platform service	No charge

Network pricing
General network pricing
Internet egress rates*
Load balancing and forwarding rules
Network Telemetry
Traffic through external IP addresses
VPN
Classic VPN pricing
HA VPN pricing at Beta
Cloud Router pricing
Unused IP address pricing
Disk pricing
Persistent disk pricing
Local SSD pricing
Image storage
Simulated maintenance event pricing

Load balancing and forwarding rules

The following applies to all types of load balancing and forwarding rules (protocol forwarding).

Iowa (us-central1)

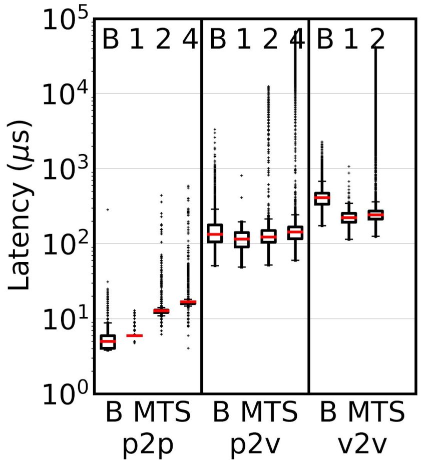
Item	Price per Unit (USD)	Pricing Unit
First 5 forwarding rules	$0.025	Per Hour
Per additional forwarding rule	$0.010	Per Hour
Ingress data processed by load balancer	$0.008	Per GB

If you pay in a currency other than USD, the prices listed in your currency on Cloud Platform SKUs apply.

Latency

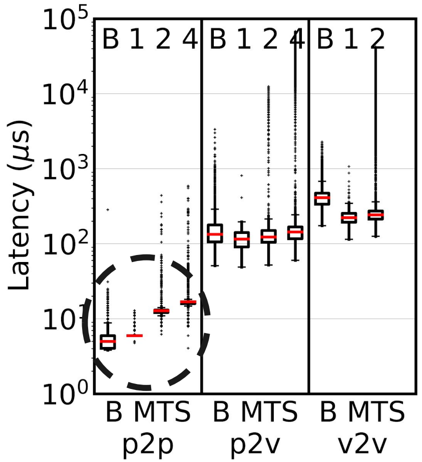
Baseline vs MTS Latency Comparison

64 byte UDP packets
Baseline is faster than MTS in p2p
MTS is faster than Baseline in p2v and v2v



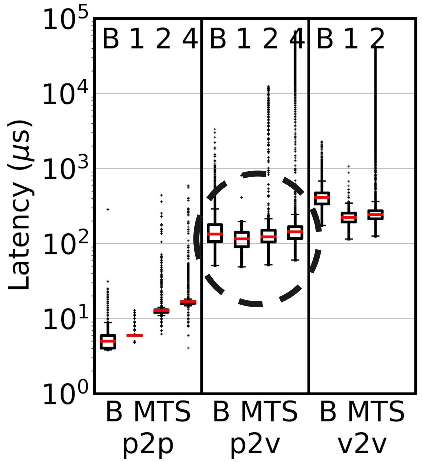
Baseline vs MTS Latency Comparison

Baseline is faster than MTS in p2p
MTS is faster than Baseline in p2v and v2v



Baseline vs MTS Latency Comparison

Baseline is faster than MTS in p2p
MTS is faster than Baseline in p2v and v2v



Baseline vs MTS Latency Comparison

Baseline is faster than MTS in p2p
MTS is faster than Baseline in p2v and v2v