Optimizing your virtual switch for VXLAN

Ron Fuller, VCP-NV, CCIE#5851 (R&S/Storage)
Staff Systems Engineer – NSBU
fuller@vmware.com

VXLAN Protocol Overview

Ethernet in IP overlay network

- Entire L2 frame encapsulated in UDP
- 50+ bytes of overhead

24 bit VXLAN Network Identifier

- 16 M logical networks

VXLAN can cross Layer 3 network boundaries

Overlay between ESXi hosts

- VMs do NOT see VXLAN ID

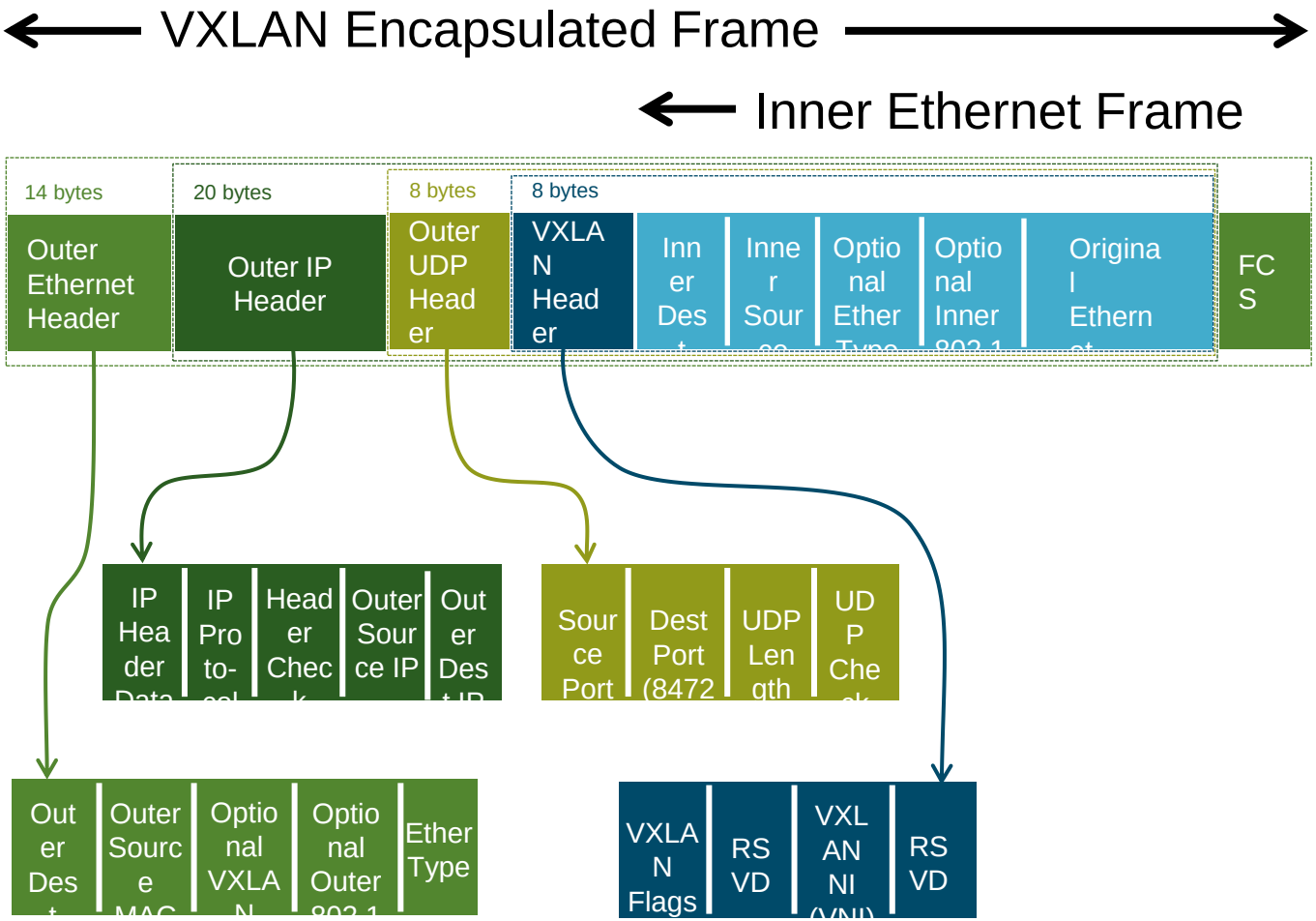
VTEP (VXLAN Tunnel End Point)

- VMkernel interface which serves as the endpoint for encapsulation/de-encapsulation of VXLAN traffic

Technology submitted to IETF for standardization

- With Cisco, Citrix, Red Hat, Broadcom, Arista and others

VXLAN Frame Format



*IP Header Data = Version, IHL, TOS, Length, ID

Network Adapter Offloads

TCP Checksum Offload	**- Software offloads TCP checksum calculation on send and checksum verification on receive**
Large Receive Offload	**- NIC aggregates packets to send large packet to software** **- Software sees fewer packets and interrupts**

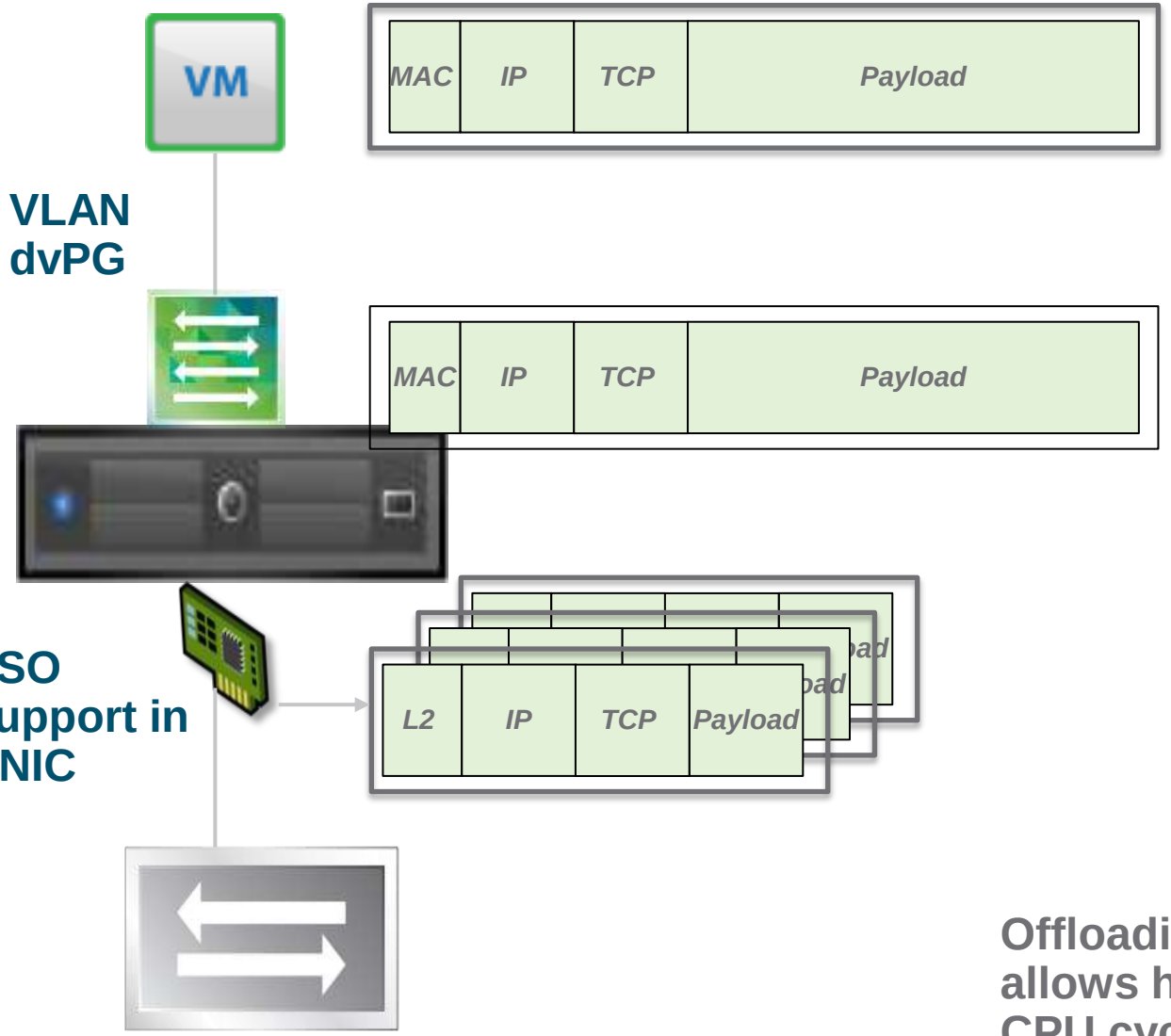
Network Adapter Offloads

TCP Segmentation Offload	**- Operating system sends large sized TCP packets to NIC** **- NIC segments packets as per physical MTU**
Receive Side Scaling	**- NIC distributes packets among queues** **- Unique receive thread per queue to drive multiple CPUs**

Important features for NSX vSwitch performance

TCP Segment Offload

Pre NSX - How is TCP forwarding optimized ?



1. VM on VLAN dvPG sends large TCP packet to Virtual NIC

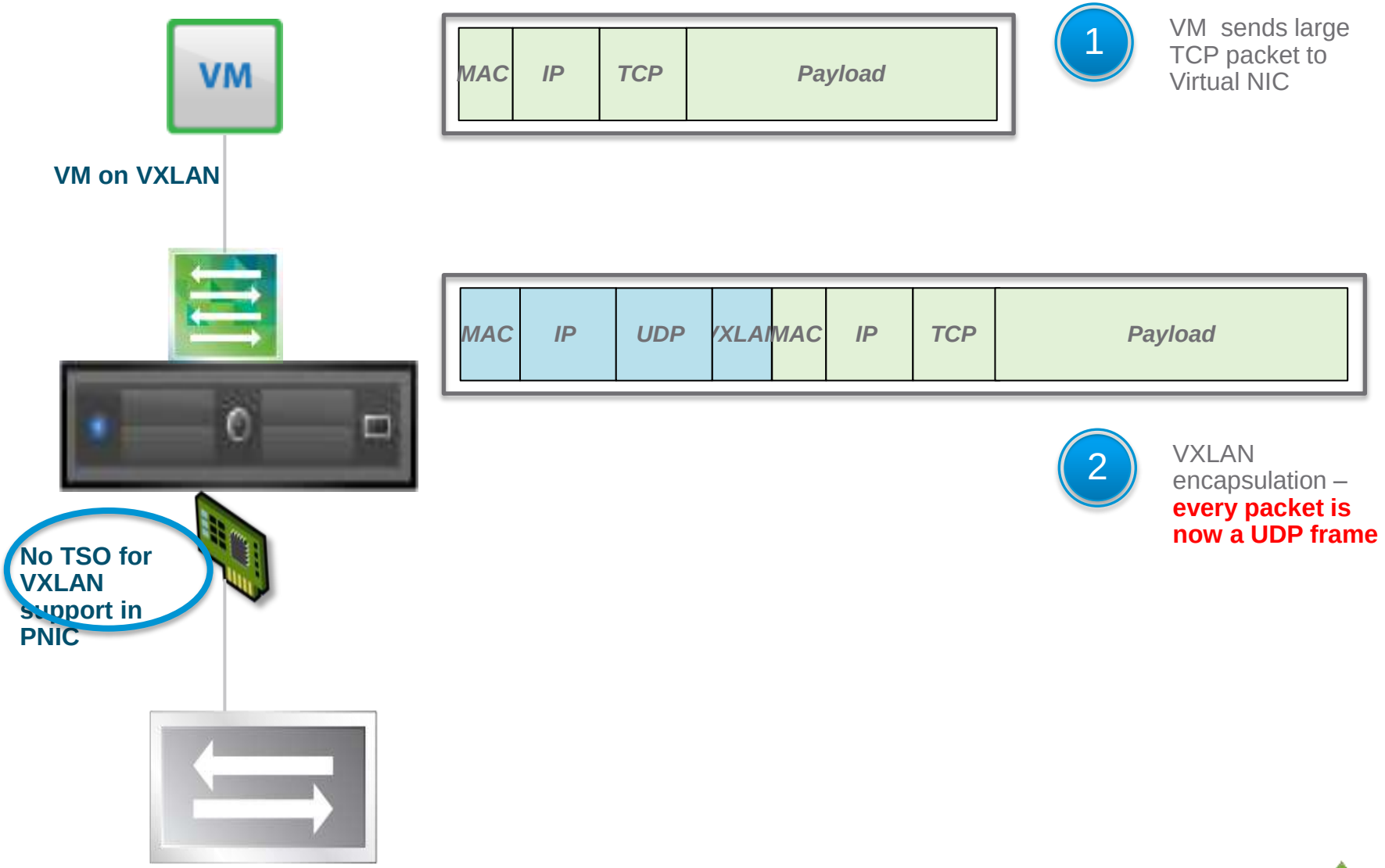
2. NSX vSwitch sends large packet to TSO enabled physical NIC

3. Physical NIC segments packet based on physical MTU

Offloading TSO to NIC allows host to conserve CPU cycles while sending large TCP messages. Almost all

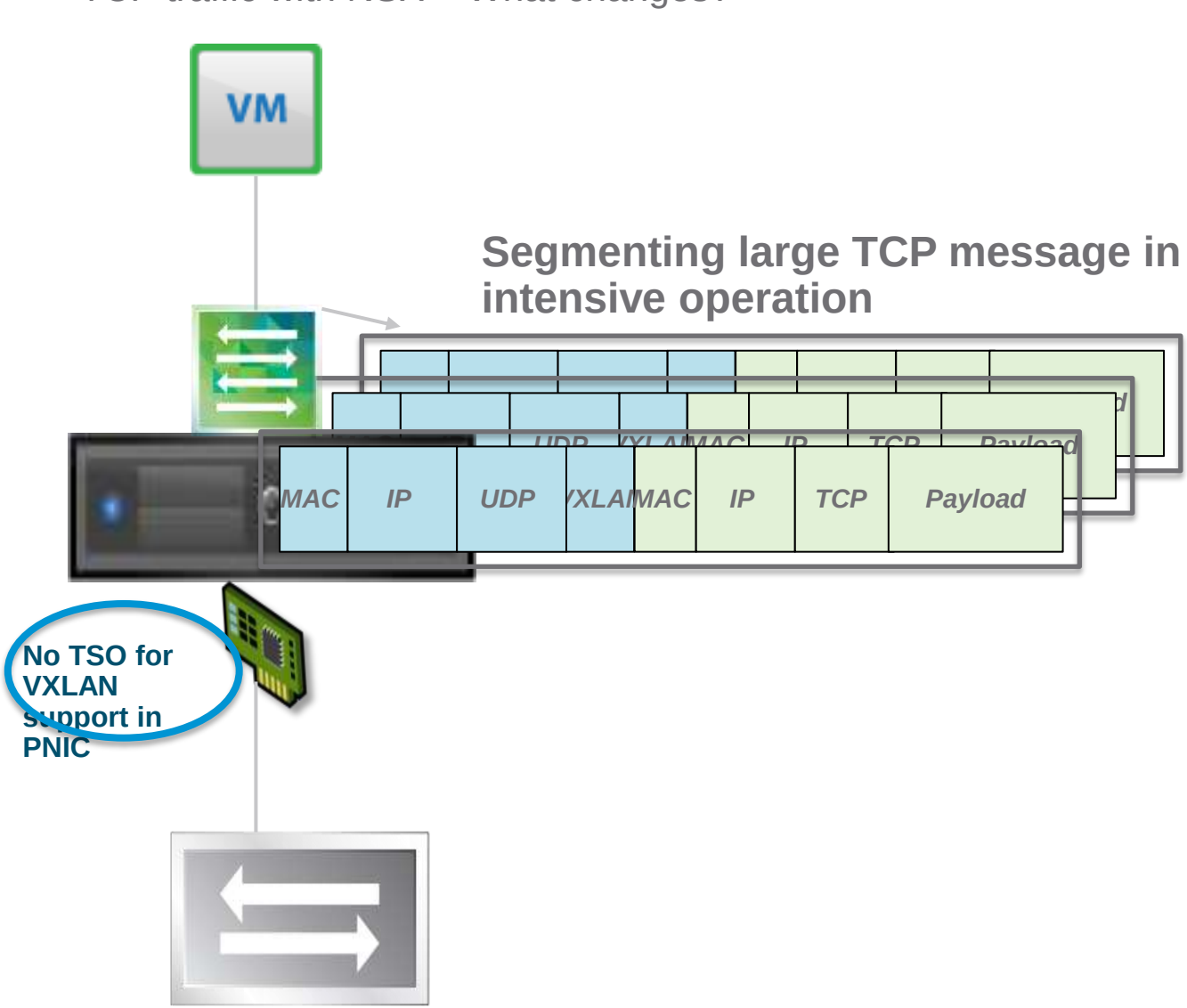
TCP Segment Offload

TCP traffic with NSX – What changes?



TCP Segment Offload – VM on VXLAN LS

TCP traffic with NSX – What changes?

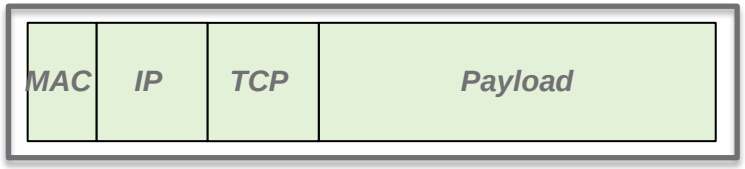


1 VM sends large TCP packet to Virtual NIC

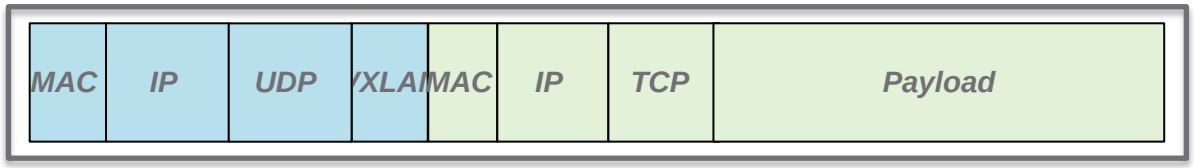
2 VXLAN encapsulation and TSO in software – **because the NIC can't do TSO for the inner TCP packet within a UDP frame**

TCP Segment Offload – VM on VXLAN LS

TSO Support for VXLAN



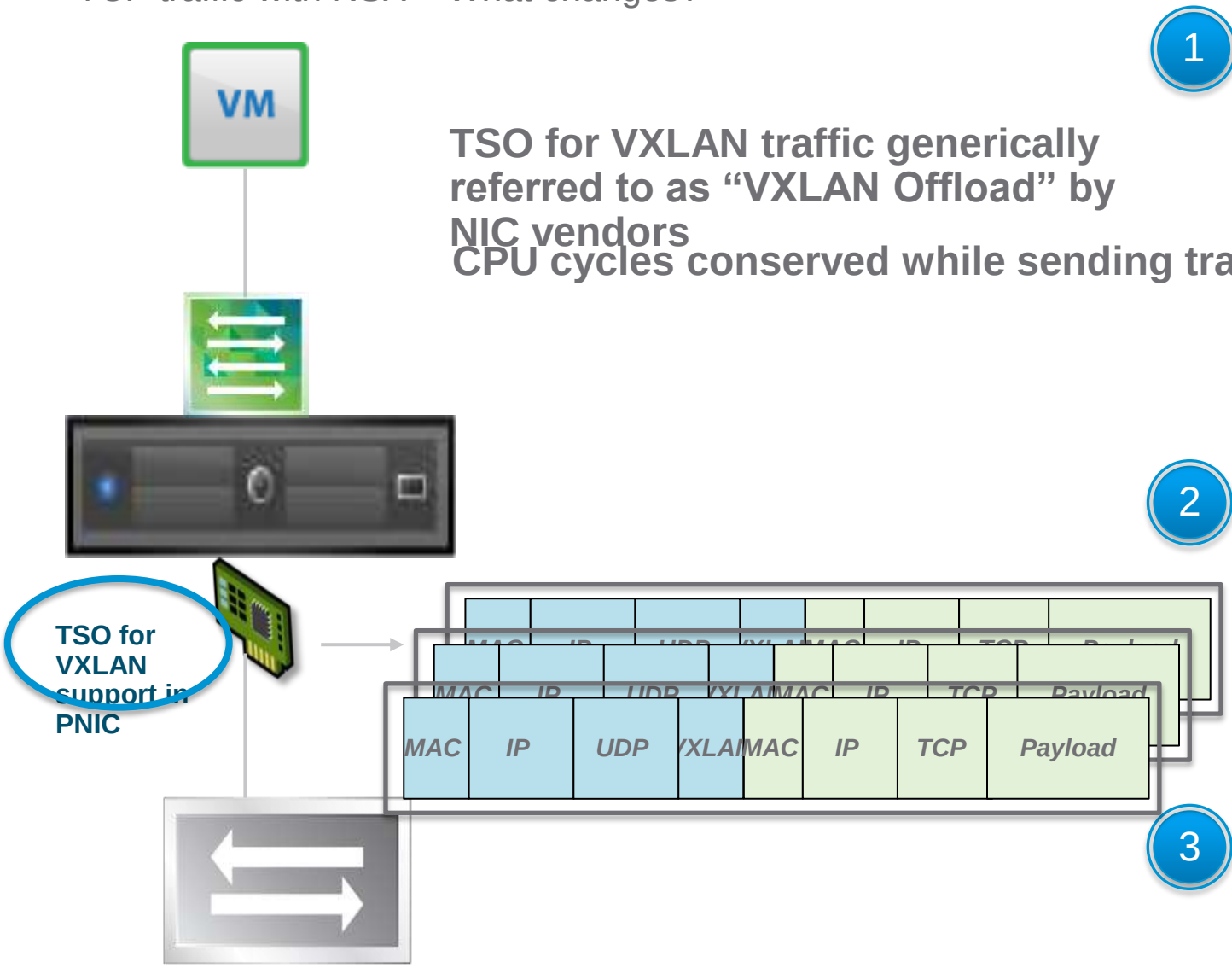
1 VM sends large TCP packet to Virtual NIC



2 VXLAN encapsulation – **every packet is now a UDP frame**

TCP Segment Offload – VM on VXLAN LS

TCP traffic with NSX – What changes?



1 VM sends large TCP packet to Virtual NIC

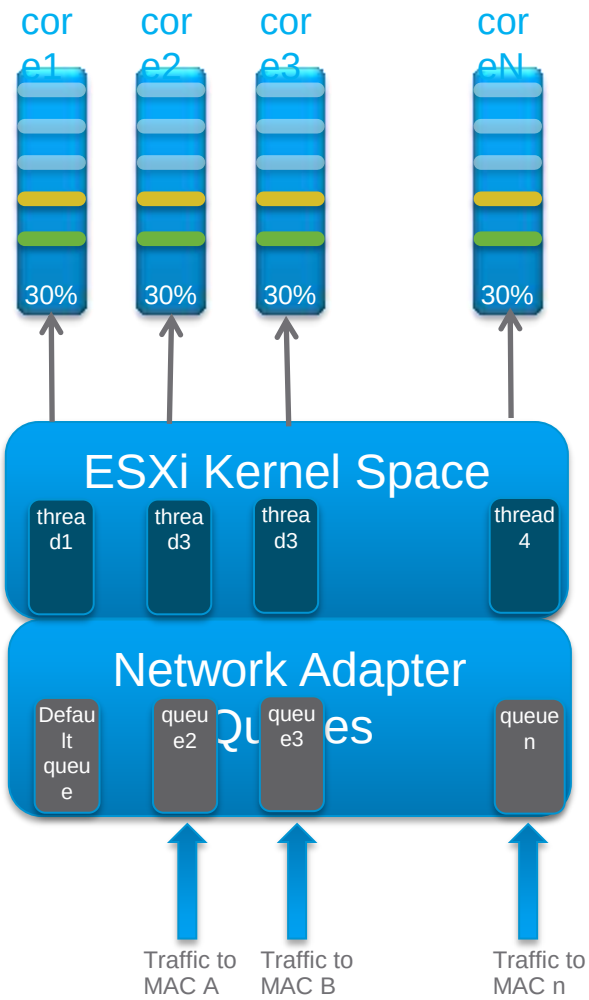
TSO for VXLAN traffic generically referred to as "VXLAN Offload" by NIC vendors

CPU cycles conserved while sending traffic

2 VXLAN encapsulation

3 TSO for VXLAN packets in PNIC

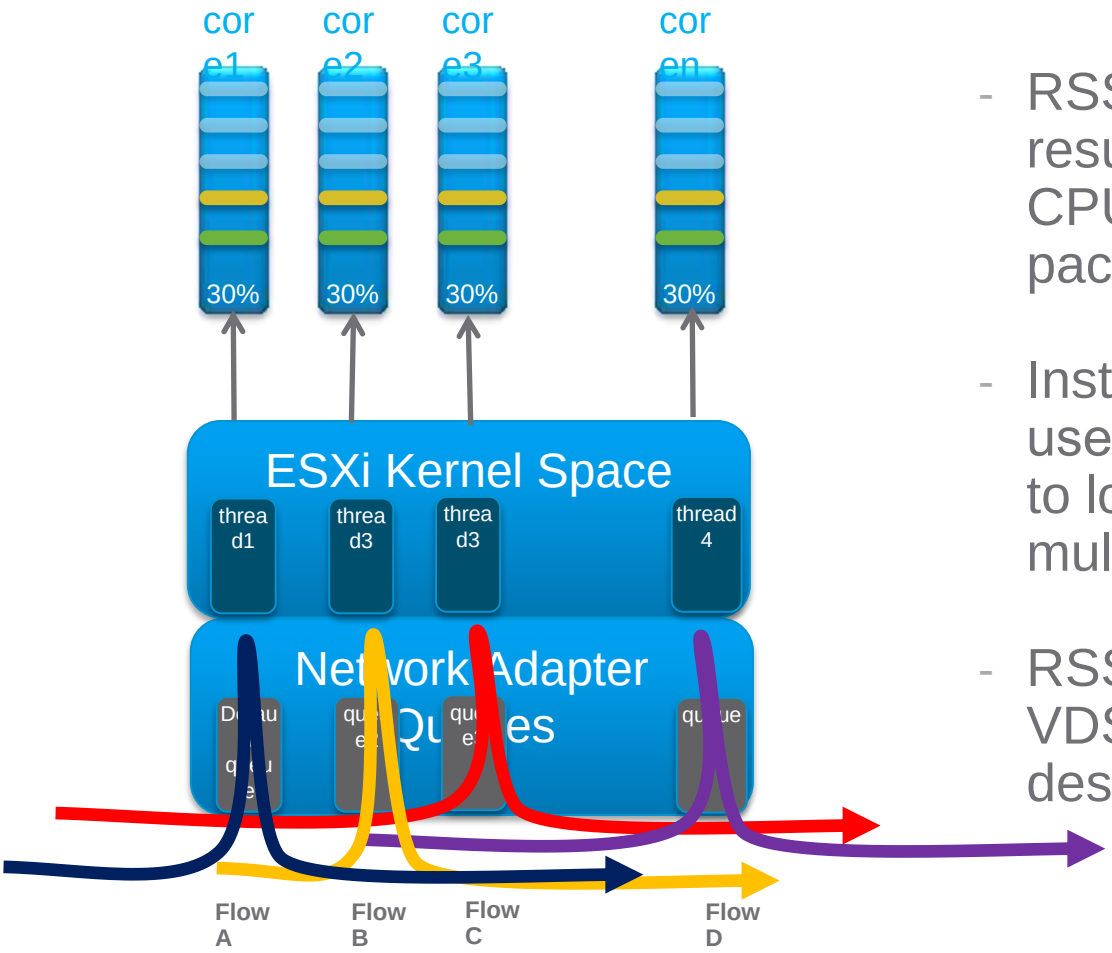
Netqueue



- Netqueue is a feature designed to enable a vSphere host to receive line rate traffic from the physical network
- Works by driving multiple CPUs to handle receive packet processing
- Before NSX – there is no traffic encapsulation and VDS receives traffic to multiple unique MAC addresses (MAC address assigned to VM vNICs)
- Netqueue works by steering traffic destined to each VM MAC to a unique NIC queue.
- Interrupts raised by NIC queues will drive multiple CPUs.

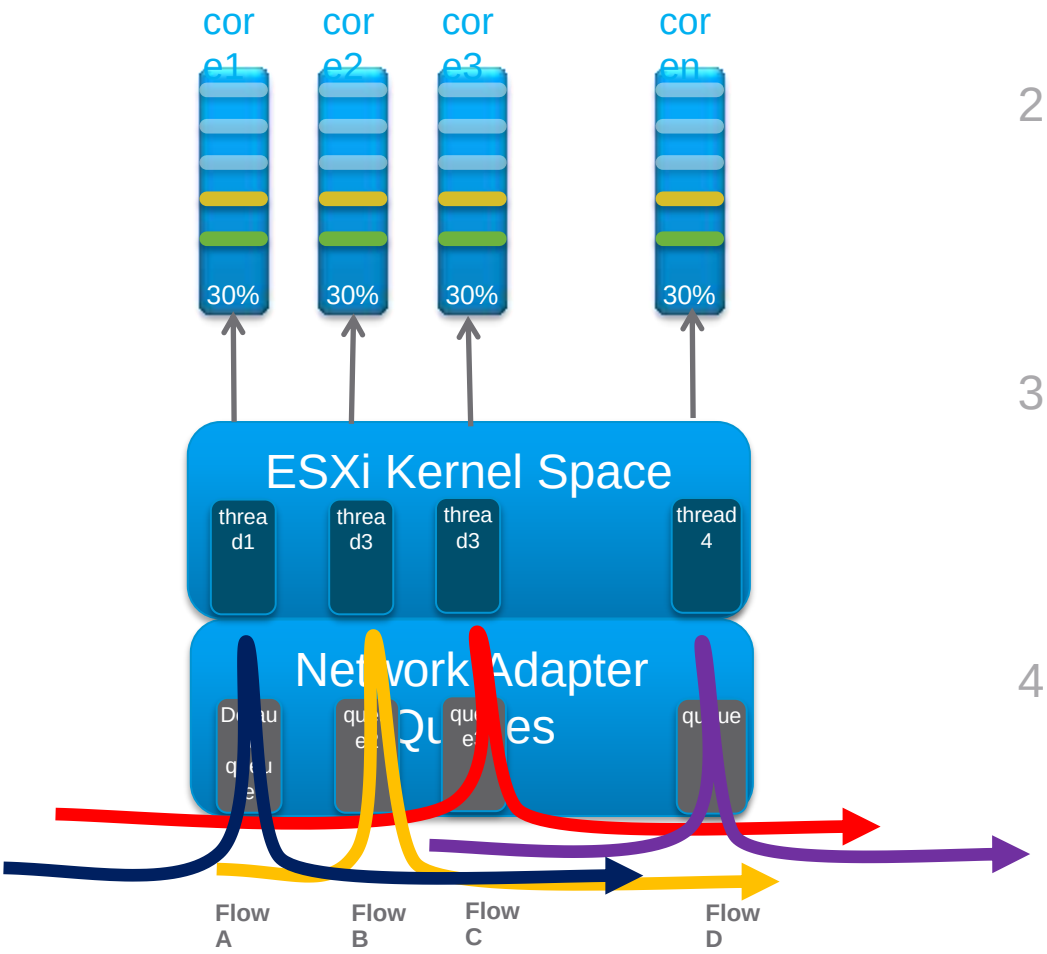
With NSX all traffic to a vSphere host is destined to VTEP MAC address.
Ability to drive multiple CPU with Netqueue is lost

Receive Side Scaling (RSS)



- RSS tries to achieve the same end result as Netqueue – use multiple CPUs to handle receiving of packets
- Instead of MAC addresses, RSS uses packet's Layer 2 – 4 headers to load balance traffic across multiple queues
- RSS is not enabled for all MACs. VDS enables RSS for traffic destined to VTEP MAC addresses

Receive Side Scaling (RSS)



1. Packet arrives at default queue
2. MAC filters attached to the queue are evaluated to see if RSS is required for packet dest MAC
3. If RSS is not enabled for packet dest MAC then default queue / thread processes traffic
4. If RSS is enabled for the dest MAC then packet is hashed using configured RSS algorithm to one of 4 queues reserved

vSphere Inbox v/s Async Drivers

- Inbox Drivers - vSphere packages NIC drivers into ESXi. Typically updated at vSphere major releases.

- Async Drivers – New NIC driver functionality and performance improvements released by NIC vendor independent of vSphere release. These drivers can be downloaded from https://my.vmware.com/web/vmware/downloads

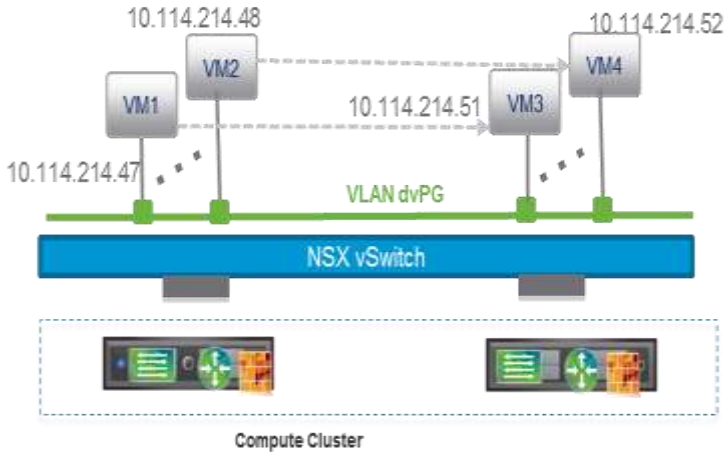
- Once the async driver matures that version is pulled into the next major vSphere release.

- With some NICs the vSphere inbox driver is optimal for VXLAN traffic (i.e. support VXLAN TSO and RSS) whereas with many NICs the inbox driver is not optimal and will require an upgrade (on compute and edge hosts) to the recommended async version

Test Topology and Tools

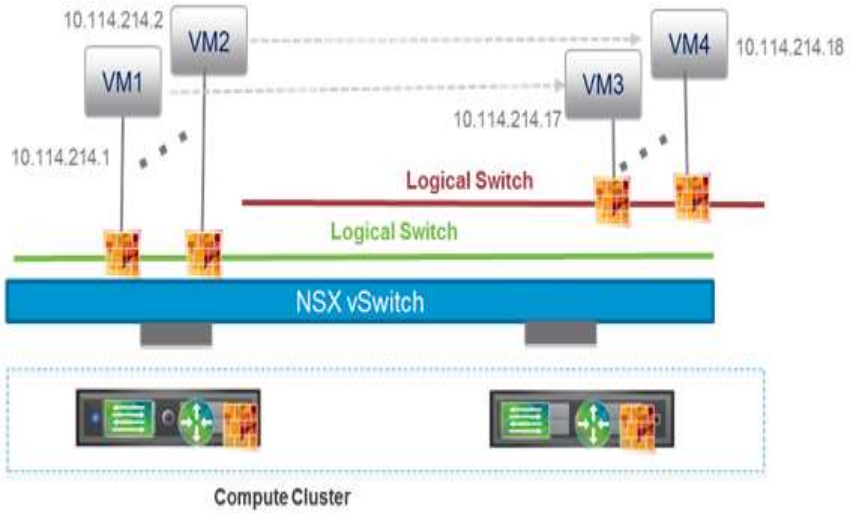
Test Topology

1



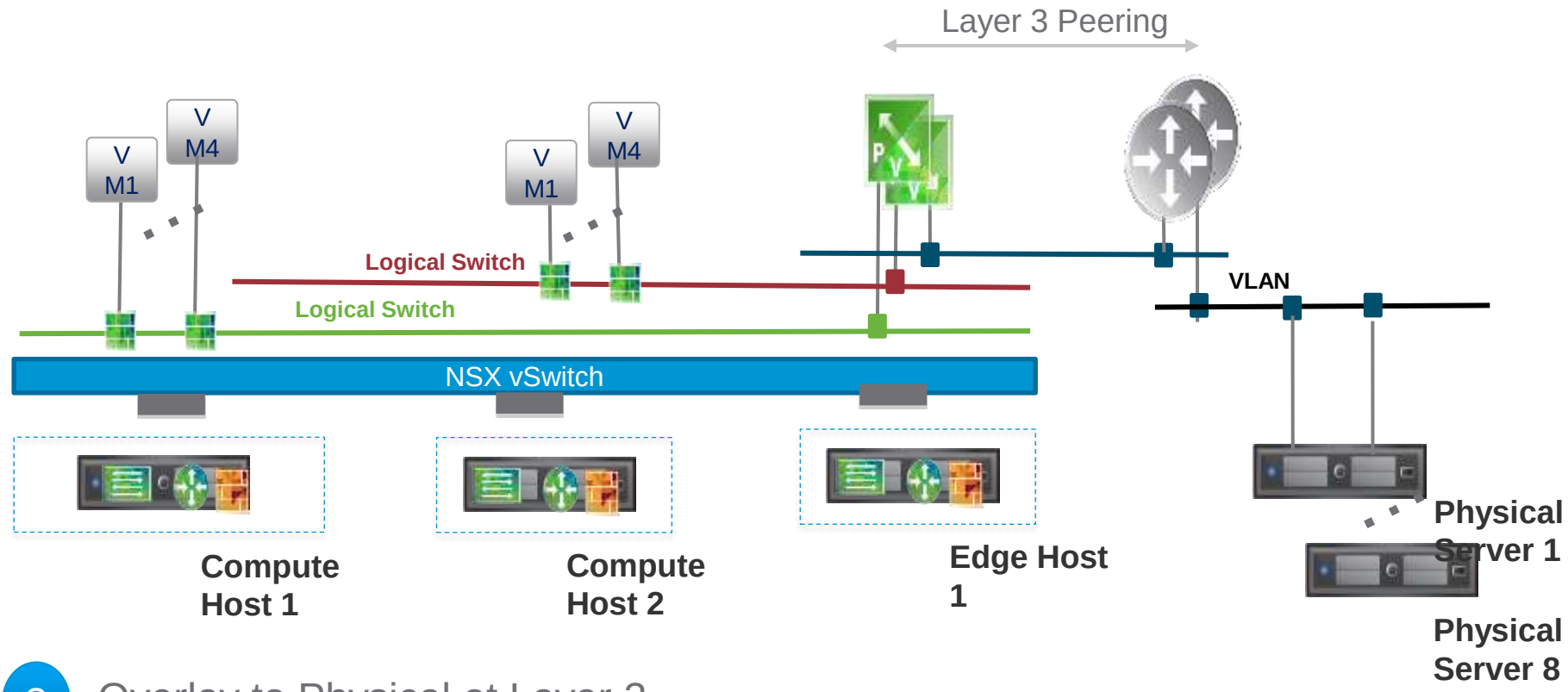
- East – West traffic before NSX / Network Virtualization
  - **Establish Performance baseline on VDS with VMs on VLANs**

2



- East – West traffic with NSX / Network Virtualization
  - Logical Switching

NSX Performance testing topology



3 Overlay to Physical at Layer 3

NSX Performance testing done in typical NSX topologies documented in design guide.

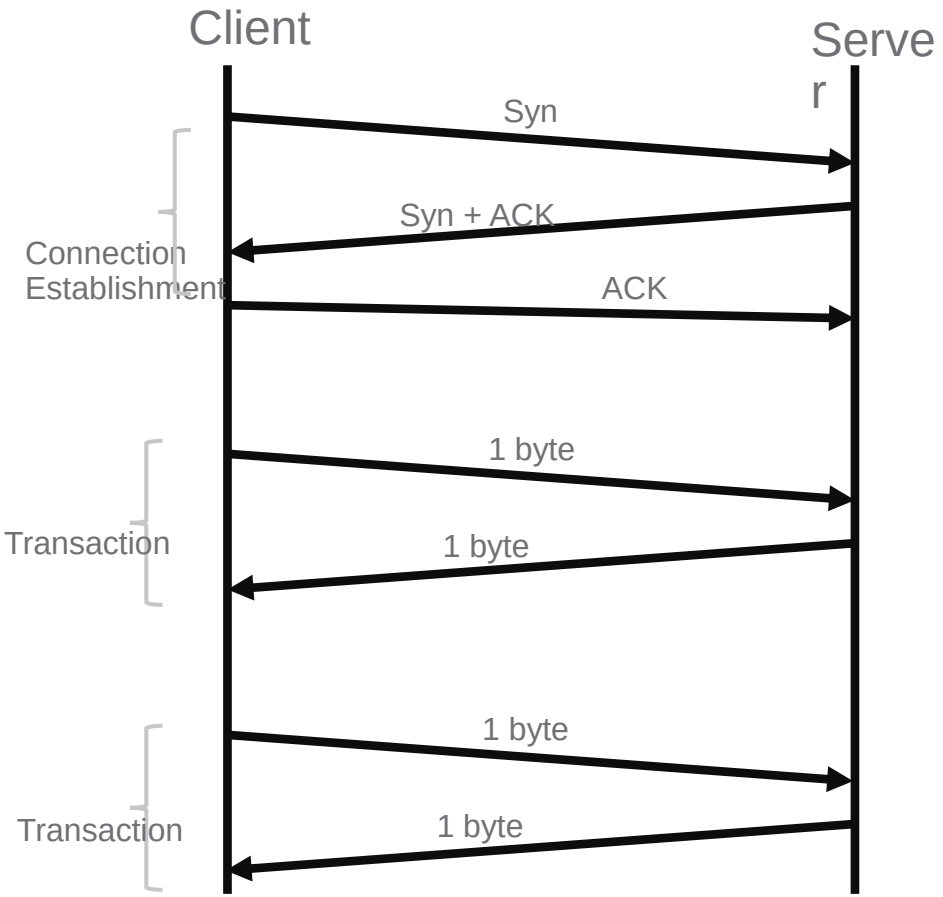
Test Tools and Methodology

- **iperf for bandwidth testing**
  - Supports running TCP and UDP traffic
  - Multiple TCP and UDP sessions support
  - UDP streams at defined sending rate to verify no loss UDP rate
  - On Server “iperf -s”.
  - On client iperf –c <server IP> -t <duration> -P <number of sessions>

Netperf for latency test

- TCP Latency using netperf tcp round robin test (tcp_rr)
- One transaction - 1 byte from client to server and then 1 byte from server to client
- Test reports # of transactions / sec.
- Time for one roundtrip = 1 / # of transactions per second
- On server run “netserver”

On client run “netperf –H <server_ip> -t TCP_RR –l <test duration>”



CPU overhead with NSX

"NSX introduced VXLAN, Routing and Firewall to the hypervisor. What does this cost in terms of CPU?"

The following sample methodology can be used to determine the CPU overhead for each feature

	Bandwidth in Gbps	Total CPU on host
1 Run a bandwidth test with VMs on VDS (VLAN dvPG) and record total bandwidth and CPU	18.426	364.46
2 Calculate CPU per Gbps over VLAN	1	19.74
3 Run a bandwidth test with VMs on Logical Switch (no DLR and no Firewall)	18.185	409.70
4 Calculate CPU per Gbps over VXLAN	1	22.53

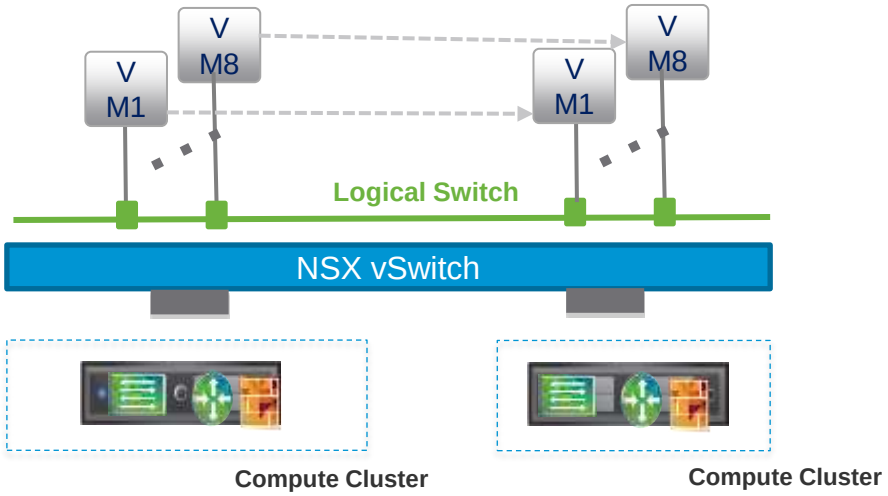
CPU overhead with NSX

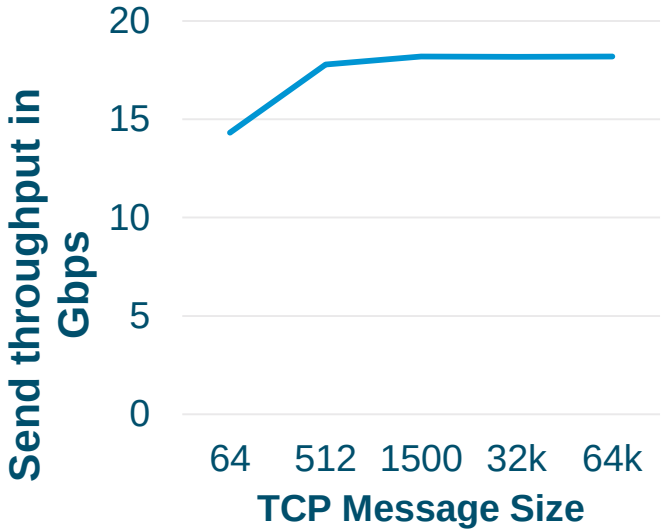
5	Additional CPU for 1 Gbps over VXLAN compared to VLAN = CPU per Gbps over VXLAN – CPU per Gbps over VLAN	22.53 - 19.74 = 2.79
6	Additional CPU for 10 Gbps over VXLAN	2.79 * 10 = 27.9
6	Additional CPU for 10 Gbps over VXLAN on a 12 core hypervisor	27.9 * 100 / 1200 = 2.35 %

Similar methodology can be followed to determine additional CPU for running Routing and Firewall on the host

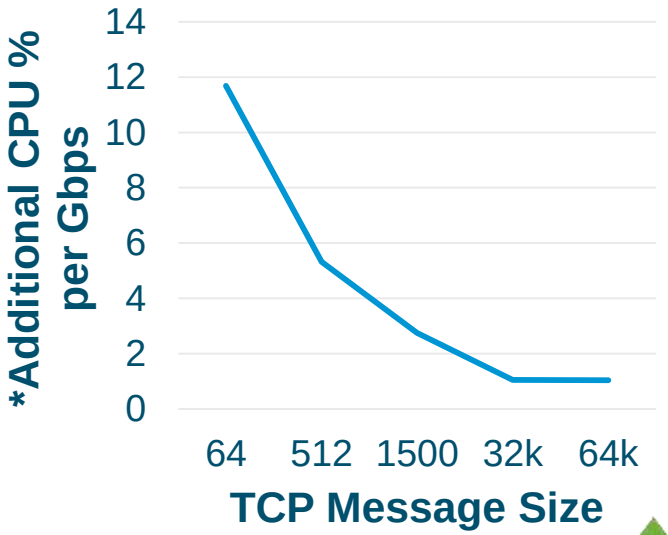
Test Results

Logical Switching





- Line rate traffic (~18Gbps) with 2 NICs per host with VXLAN
- Additional CPU for VXLAN traffic between hosts is ~3% of CPU.



*Y axis shows % of additional CPU overhead as compared to same baseline tests performed on VLAN based networks.

Summary

Summary

- RSS support is required in NIC and vSphere driver to receive line rate VXLAN traffic on a 10 Gbps NIC.

- TSO for VXLAN traffic support on NIC allows a hypervisor sending traffic to conserve CPU cycles by offloading TCP segmentation to the PNIC.

- Check with NIC vendor for TSO and RSS driver support – may need an async driver!