# Considerations for Benchmarking Virtual Networks

draft-bmwg-nvp-03

Samuel Kommu, skommu@vmware.com

Jacob Rapp, jrapp@vmware.com

March 2019 IETF 104 – Prague

# Considerations for Benchmarking Network Virtualization Platforms - Overview

## draft-bmwg-nvp-03

### Scope

Network Virtualization Platforms (NVO3)

### Considerations

#### NVE Considerations

Co-located vs. Split-NVE

#### Server Hardware

Support for HW offloads (TSO / LRO / RSS)
Other Hardware offload benefits – Performance Related Tuning
Frame format sizes within Hypervisor

#### Documentation

System Under Test vs Device Under Test
Intra-Host (Source and destination on the same host)
Inter-Host (Source and Destination on different hosts – Physical Infra providing connectivity is part of SUT

#### Traffic Flow Optimizations

Fast-path vs. slow-path, Cores and co-processors

#### Control Plane Scale

Event handling (VM Create, Delete, etc)

# Scope clarifications

4. Scope

Focus of this document is the Network Virtualization Platform in two separate scenarios as outlined in RFC 8014 section 4, Network Virtualization Edge (NVE) and RFC 8394 section 1.1 Split-NVE and the associated learning phase:

4.1.1. Scenario 1

RFC 8014 Section 4.1 "NVE Co-located with server hypervisor": Where the entire NVE functionality will typically be implemented as part of the hypervisor and/or virtual switch on the server.

4.1.2. Scenario 2

RFC 8394 Section 1.1 "Split-NVE: A type of NVE (Network Virtualization Edge) where the functionalities are split across an end device supporting virtualization and an external network device."

4.1.3. Learning

Address learning rate is a key contributor to the overall performance of SUT specially in microservices type of use cases where a large amount of end-points are created and destroyed on demand.

4.1.4. Flow Optimization

There are several flow optimization algorithms that are designed to help improve latency or throughput. These optimizations MUST be documented.

# NVE Co-located vs. Split-NVE Review

## draft-bmwg-nvp-03

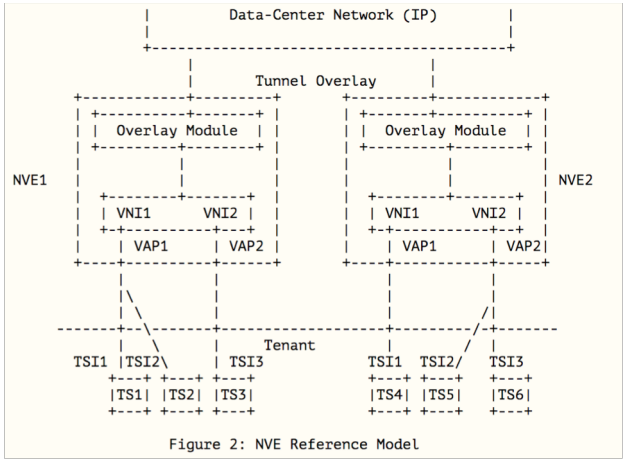
### Scope

Most of comments and questions were around clarifying scope

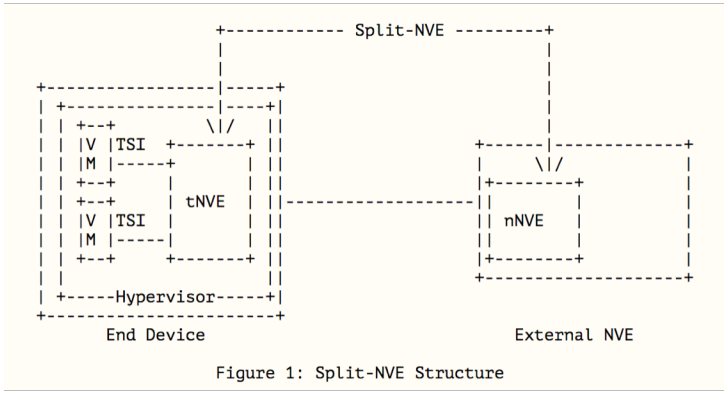
These benchmark considerations are specific to two scenarios of Network Virtualization Edge (NVE)

1. NVE Co-located with the server hypervisor (RFC 8014 Section 4.1 **An Architecture for Data-Center Network Virtualization over Layer 3 (NVO3)**) – "When server virtualization is used, the entire NVE functionality will typically be implemented as part of the hypervisor and/or virtual switch on the server. "

2. Split-NVE (RFC 8394 **Split Network Virtualization Edge (Split-NVE) Control-Plane Requirements** Section 1.1) – "Another possible scenario leads to the need for a split-NVE implementation. An NVE running on a server (e.g., within a hypervisor) could support NVO3 service towards the tenant but not perform all NVE functions (e.g., encapsulation) directly on the server; some of the actual NVO3 functionality could be implemented on (i.e., offloaded to) an adjacent switch to which the server is attached."




RFC8014 Section 3.2 Figure 2




RFC8394 Section 1 Figure 1

# Split co-located vs. not co-located

draft-bmwg-nvp-03

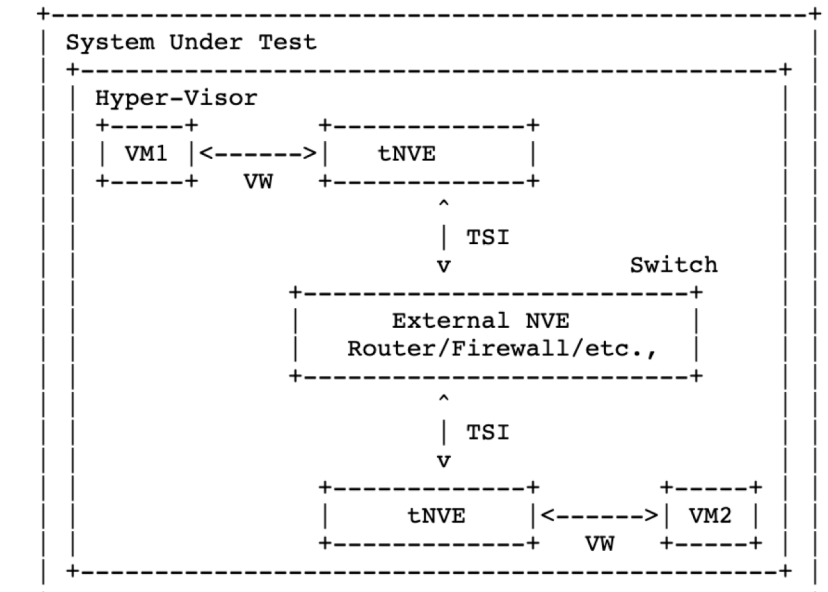



Figure 4 NVE Split collocated - System Under Test

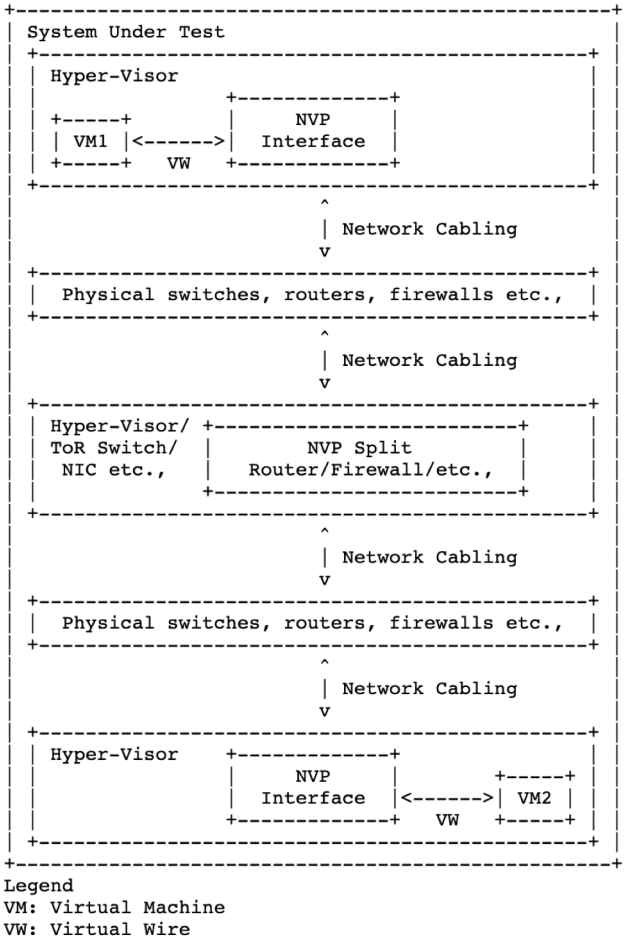



Figure 5 NVE Split not collocated - System Under Test

# Traffic Flow Optimizations

**5.2. Traffic Flow Optimizations**

Several mechanisms are employed to optimize traffic flows. Following are some examples:

**5.2.1. Fast Path**

A single flow may go through various switching, routing and firewalling decisions. While in the standard model, every single packet has to go through the entire process/pipeline, some optimizations help make this decision for the first packet, store the final state for that packet, and leverage it to skip the process for rest of the packets that are part of the same flow.

**5.2.2. Dedicated cores / Co-processors**

Packet processing is a CPU intensive workload. Some NVE's may use dedicated cores or a co-processor primarily for packet processing instead of sharing the cores used for the actual workloads. Such cases MUST be documented. Tests MUST be performed with both shared and dedicated cores. Results and differences in results MUST be documented.

**5.2.3. Prioritizing and de-prioritizing active flows**

Certain algorithms may prioritize or de-prioritize traffic flows based on purely their network characteristics such as the length of the flow. For example, de-prioritize a long-lived flow. This could result in changing the performance of a flow over a period of time. Such optimizations MUST be documented, and tests MUST consist of long-lived flows to help capture the change in performance for such flows. Tests MUST note the point at which performance changes.

# State Changes - WIP

6. Control Plane Scale Considerations

For a holistic approach to performance testing, control plane performance must also be considered. While the previous sections focused on performance tests after the SUT has come to a steady state, the following section focusses on tests to measure the time taken to bring the SUT to steady state.

In a physical network infrastructure world view, this could be various stages such as boot up time, time taken to apply configuration, BGP convergence time etc., In a virtual infrastructure world, this involves lot more components which may also be distributed across multiple hosts. Some of the components are:

- o VM Creation Event
- o VM Migration Event
- i How many total VMs can the SUT support
- o What is the rate at which the SUT would allow creation of VMs

Please refer to section 2 of RFC 8394 for various VM events and their definitions. In the following section we further clarify some of the terms used in the above RFC.

# State Changes – WIP Cont.

6.1.1. VM Events

Performance of various control plane activities which are associated with the System Under Test, MUST BE documented.

- VM Creation: Time taken to join the VMs to the SUT provided network
- Policy Realization: Time taken for policy realization on the VM
- VM Migration: Time taken to migrate a VM from one SUT provided network to another SUT provided network

For the test itself, the following process could be use:

1 API to call to join VM on the SUT provided network

2 Loop while incrementing a timer - till the VM comes online on the SUT provided network

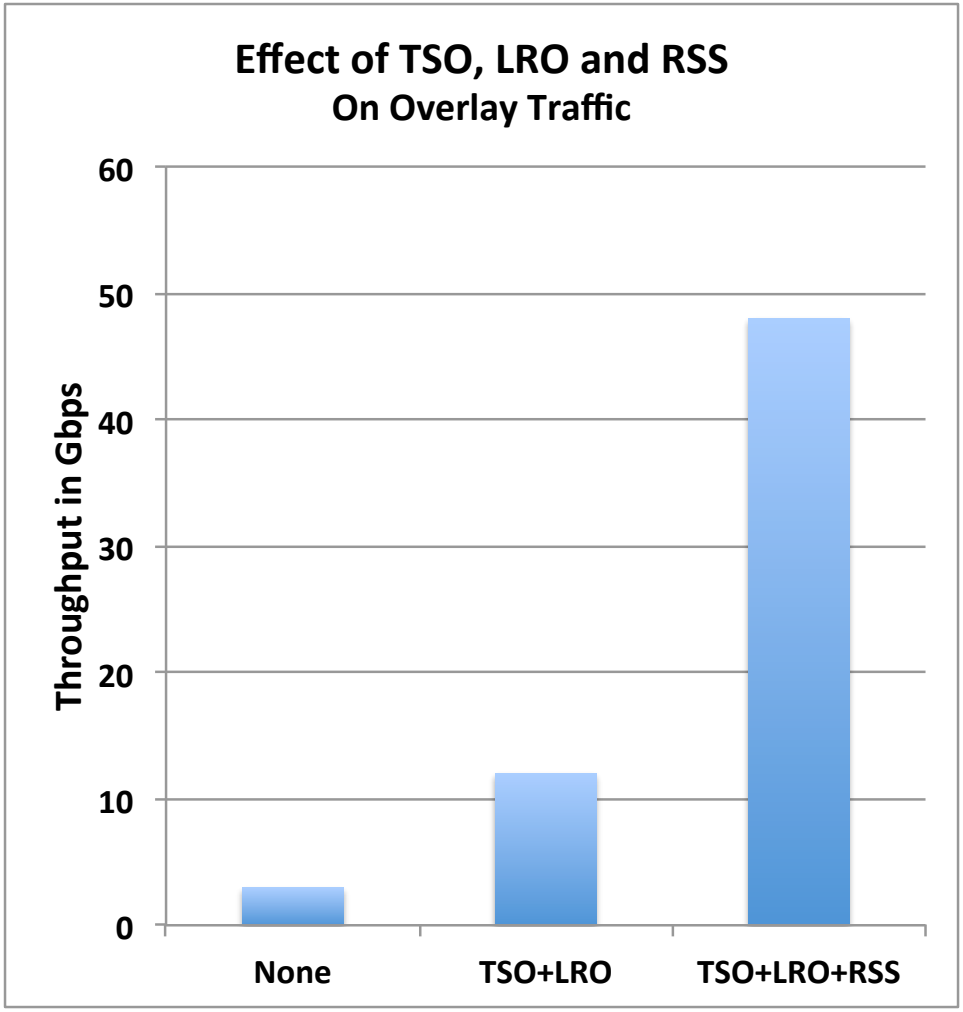
Similarly, policy realization and VM migration may also be tested with a check on whether the VM is available or not available based on the type of policy that is applied.

# Test Results

# Example Test Methodology

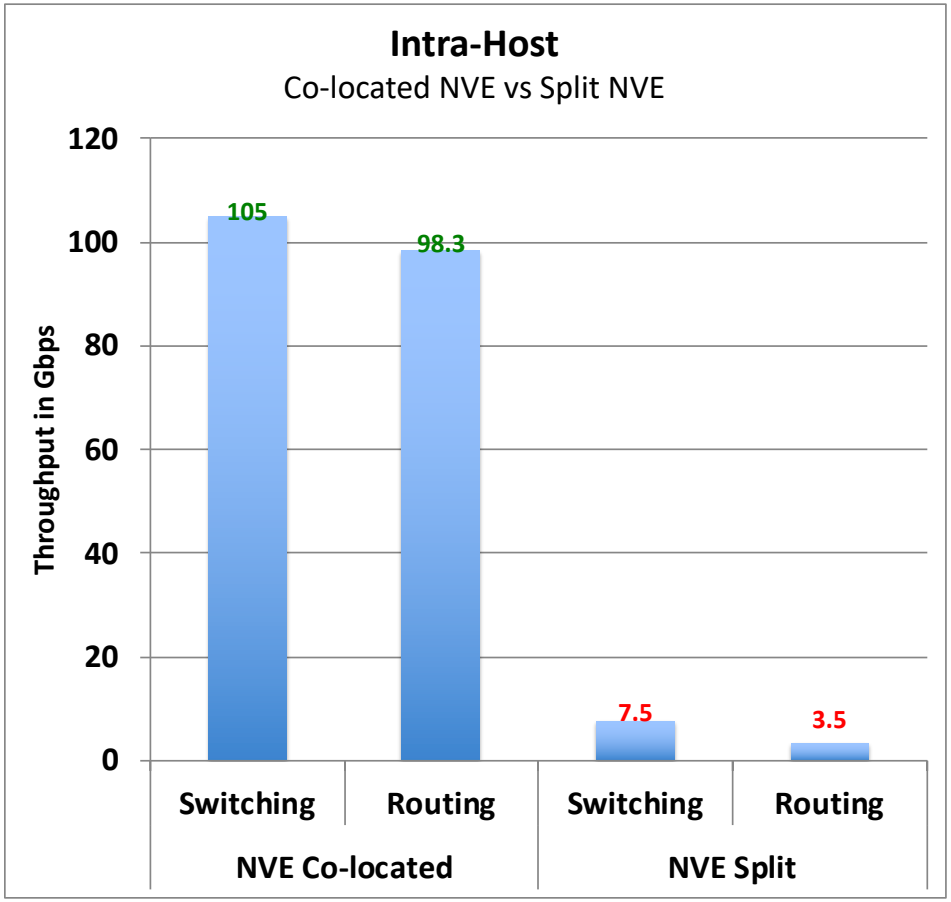
- Testing with iPerf
- Options
- -P 4 -t 90
  - P No of threads
  - t Time in seconds
  - We use about 4 VM pairs. So thats 4 VMs x 4 Threads each 16 Threads total.
- Notes: Apart from the above - on the server we use "iperf -s" to start the server side thread and "iperf -c" for the client side. On the client side the full iperf command with options would be: "iperf -c <Server IP> -P 4 -t 90"

# Example Results - Offloads



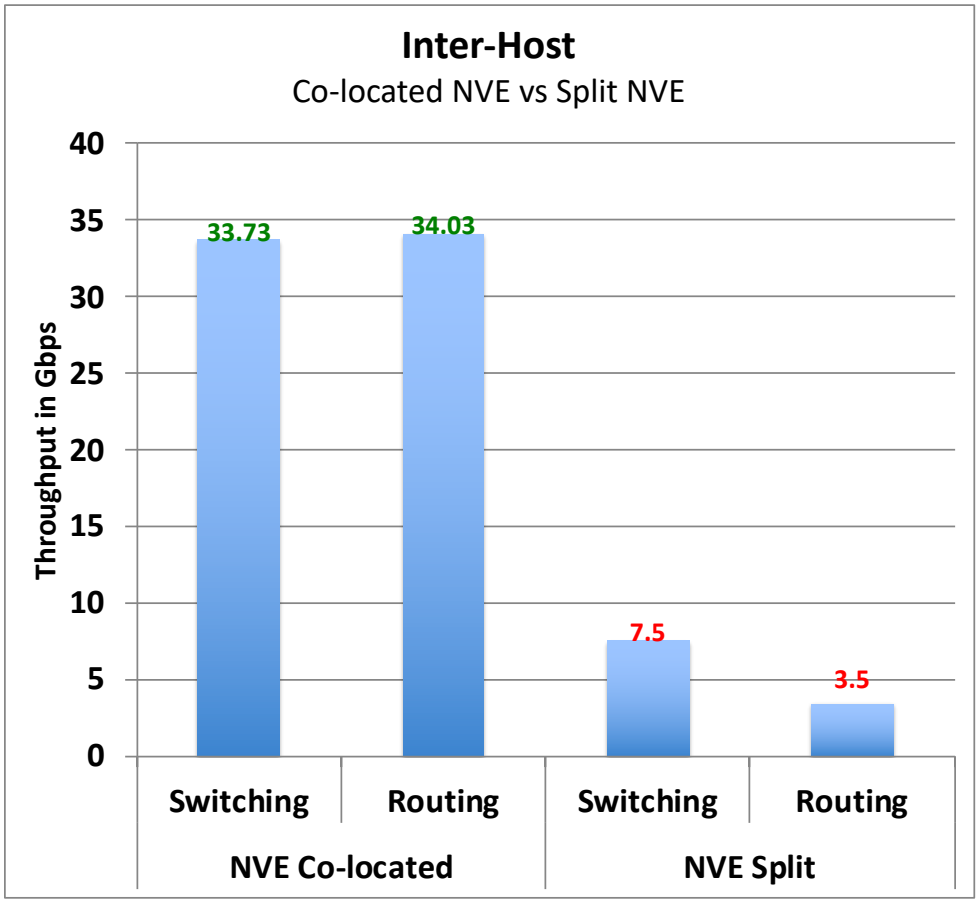

- > 10 times difference in throughput
- Throughput is a function of not just CPU but NIC card capabilities
- Other offload capabilities also have impact on performance – not profiled here
- Virtual ports don't have a rigid bandwidth profile

# Example Results – Intra-Host



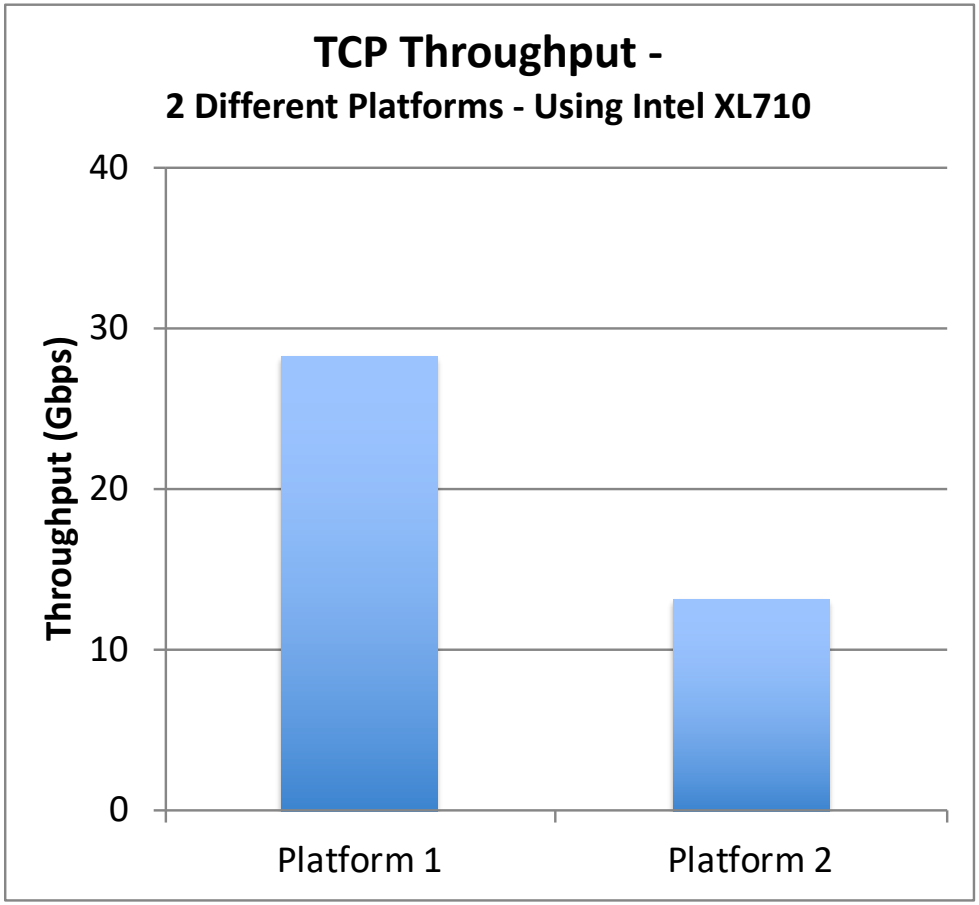

- 14 – 28 times difference in throughput
- Inline datapath takes advantage of TCP based offloads resulting in better throughput
- Less CPU cycles spent for the same amount of payload
  - 1x64K Segment vs 21xPackets (TSO)
- Virtual ports don't have a rigid bandwidth profile

# Example Results – Inter-Host




- 4 - 9 times difference in throughput
  - May be more with more ports of 40G
- Inline Datapath that takes advantage of TCP based offloads resulting in better throughput
- Less CPU cycles spent for the same amount of payload
  - 1x64K Segment vs 21xPackets (TSO)
- NVE-Co-located: Limited by Physical NIC port speed/Queuing capabilities – compared to Intra-host

# Example Results – Platform Differences



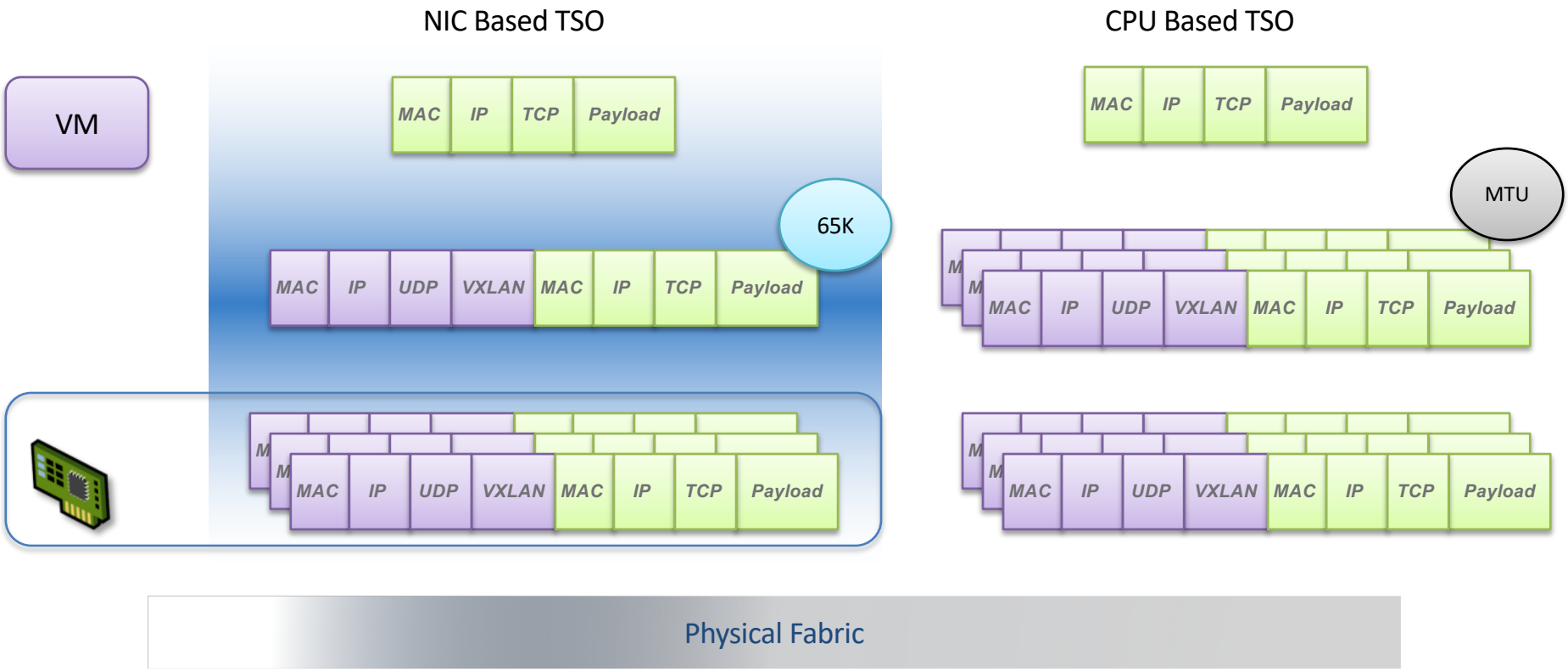

- Using multiple queues multiplies the throughput achieved
- Queuing algorithms have an impact on throughput
- NIC based queuing
  - RSS – brute force
- HV dictated queuing
  - Finer control on flows and the queues used

# Backup Slides

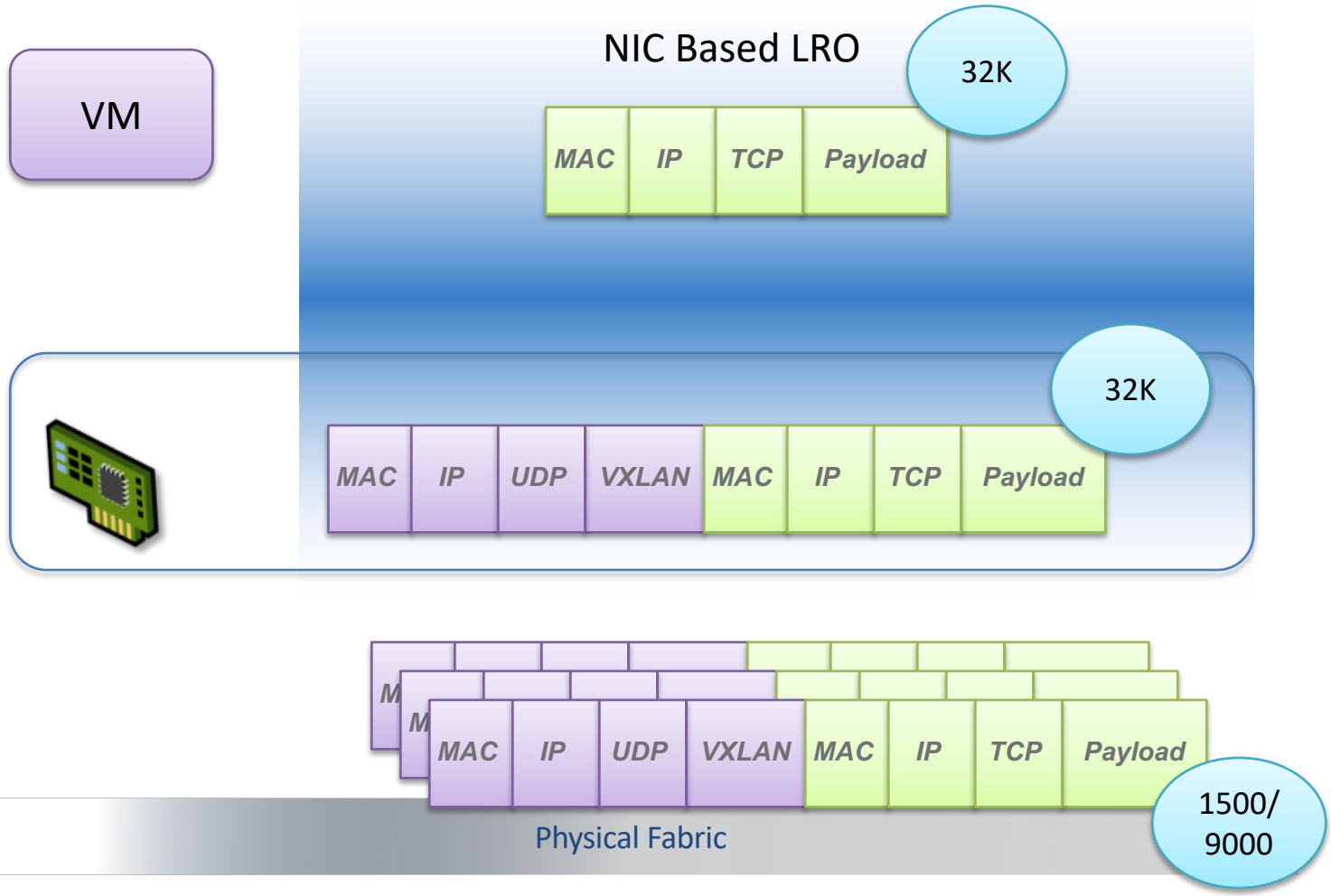
# Hardware Switch vs Software Switch

| Hardware Switching | Logical Switch/Logical Router etc., |
|---|---|
| Works at lower layer packets | Works closer to application layer segments |
| Limited by ASIC/SoC | Limited mostly by CPU and Memory (only LB)<br>• which is not really a limit with today's processor capabilities and memory capacity/speeds |
| Packet size limited by supported MTU<br>• General Max supported is 9K | Packet size a function of RSS, TSO & LRO etc.,<br>• By default 65K |
| Multiport – often 48 or more | Generally 2 Ports/Server |
| Extending functionality through additional ASIC / FPGAs and Hardware | NIC Offloads<br>Intel DPDK / Latest Drivers etc.,<br>SSL Offload with AES-NI (Intel and AMD) |

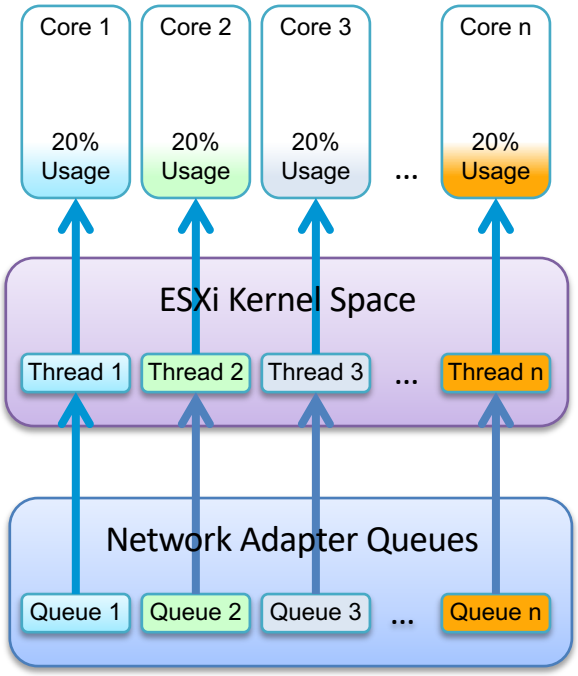
# TSO for Overlay Traffic



# LRO for Overlay Traffic



# Receive Side Scaling (RSS)




- With Receive Side Scaling Enabled
  - Network adapter has multiple queues to handle receive traffic
  - 5 tuple based hash (Src/Dest IP, Src/Dest MAC and Src Port) for optimal distribution to queues
  - Kernel thread per receive queue helps leverage multiple CPU cores

# Page Size and Response Times

| | | |
|---|---|---|
| Average Page Size | 2MB | http://httparchive.org/trends.php |
| Average HTML Content | 56KB | |
| Web Response Times | 200ms | https://developers.google.com/speed/docs/insights/Server |
| Memcached Response Time | Sub 1ms | https://code.google.com/p/memcached/wiki/NewPerformance |

Documentation

# Example Test Methodology

- Application level throughput using Apache Benchmark
  - ~2m file sizes based on http://httparchive.org/trends.php
    - Images tend to be larger
    - Page content tends to be smaller
- Application latency with Memslap
  - Standard settings
- iPerf
- Avalanche

Application Layer Benchmarks