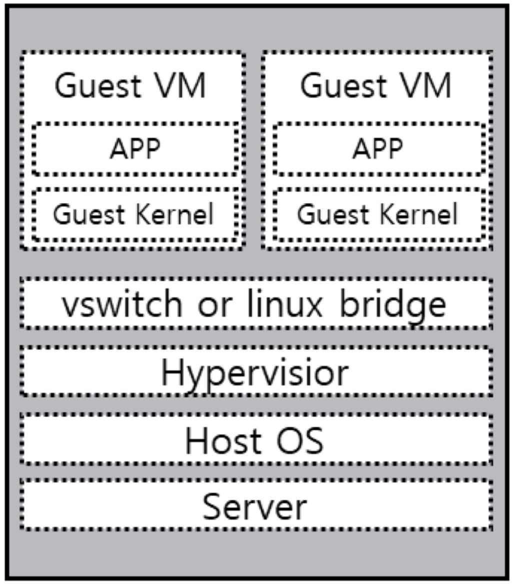
Considerations for Benchmarking Network Performance in Containerized Infrastructure

draft-dcn-bmwg-containerized-infra-00

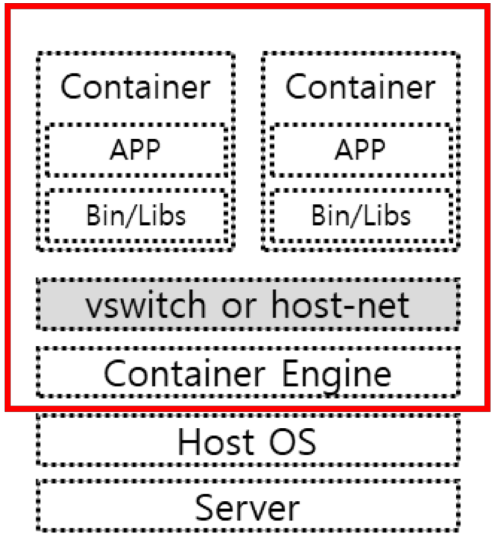
Kyoungjae Sun (gomjae@dcn.ssu.ac.kr),
Hyunsik Yang, Youngki Park, Younghan Kim
IISTRC, Soong-Sil University
Wangbong Lee
ETRI

Containerized Infrastructure

- Virtualized Network Functions(VNFs) are running on container
  - Sharing same host OS
    - isolated by using different namespace
  - It can reduce
    - Processing load by hypervisor
    - Resource for Guest OS
  - Suitable for micro-service and cloud-native environment



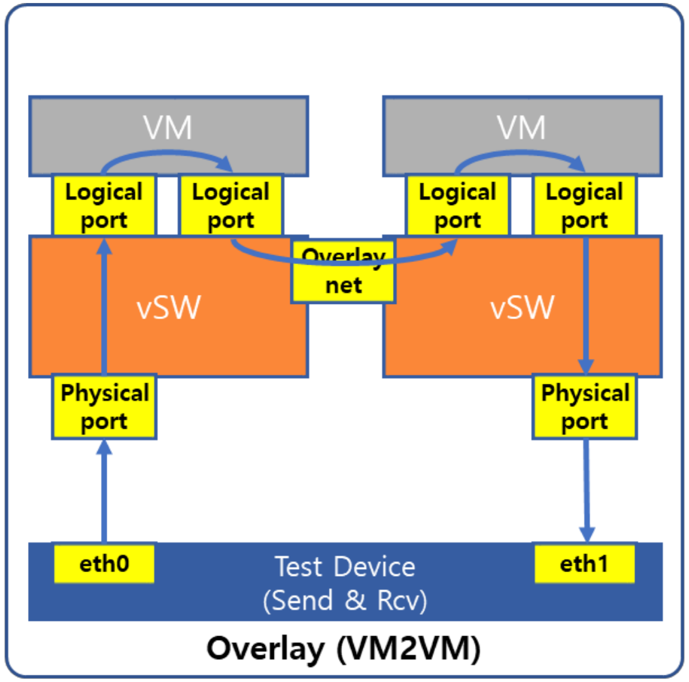
VM(Virtual Machine) based NFV Infra



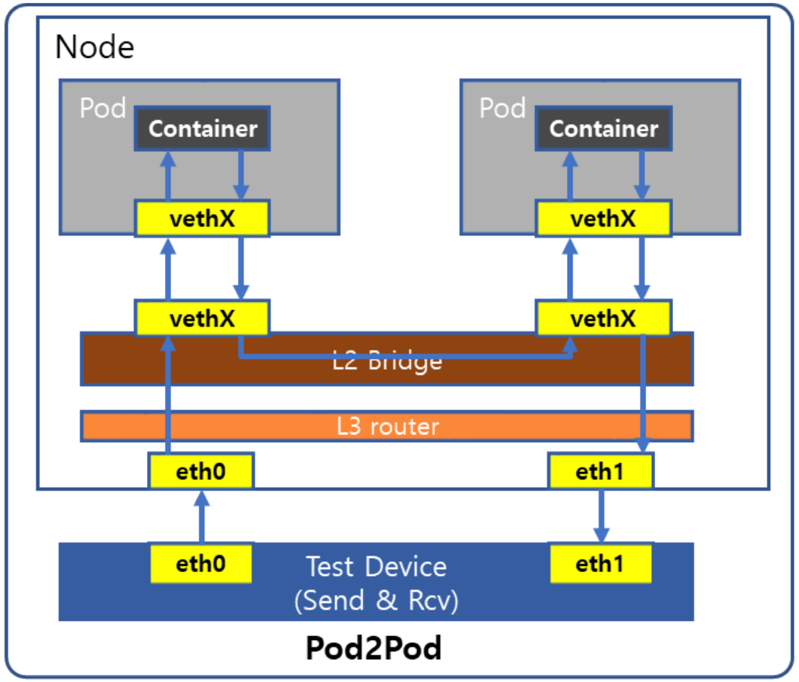
Container based VNF Infra

NFV Infrastructure Model

- ETSI GS NFV-TST 009
  - For container networking, ETSI already described their network test architecture
    - host system may use OVS, but there are many other options
    - Network Plug-ins (CNI, CNM, ..)



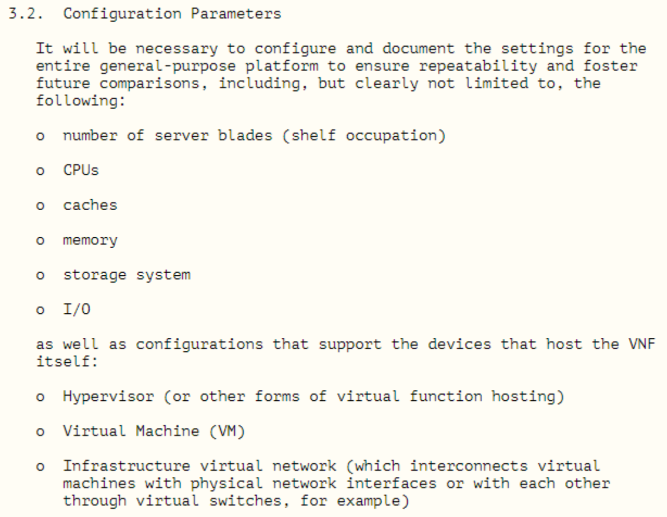
Overlay (VM2VM)



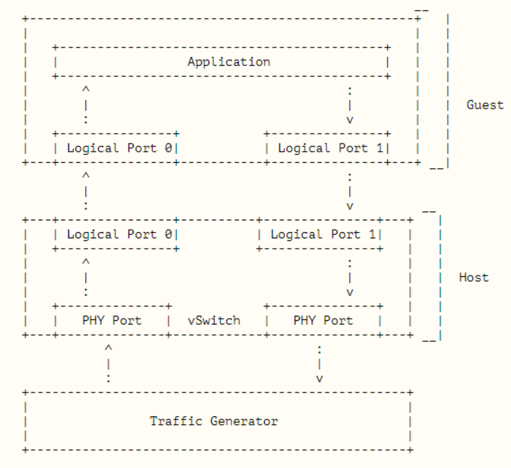
Pod2Pod

Benchmarking Considerations

- There are two RFCs about NFV benchmarking
- RFC 8172 : Considerations for Benchmarking Virtual Network Functions and Their Infrastructure
  - Define general-purpose platform as VM-based infra
- RFC 8204 : Benchmarking Virtual Switches in the Open Platform for NFV (OPNFV)
  - Describe deployment scenarios for testing vswitch benchmarking based on VM-based infra

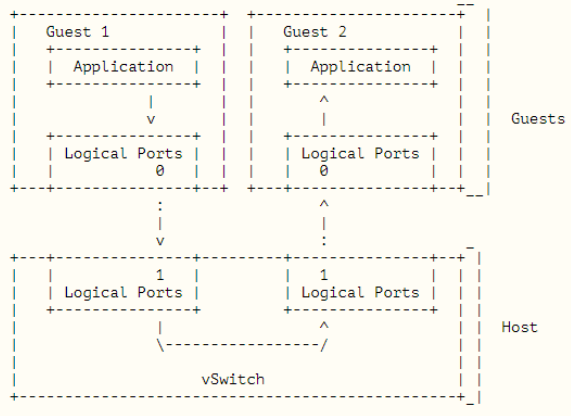


```
3.2.  Configuration Parameters

   It will be necessary to configure and document the settings for the
   entire general-purpose platform to ensure repeatability and foster
   future comparisons, including, but clearly not limited to, the
   following:

   o  number of server blades (shelf occupation)

   o  CPUs

   o  caches

   o  memory

   o  storage system

   o  I/O

   as well as configurations that support the devices that host the VNF
   itself:

   o  Hypervisor (or other forms of virtual function hosting)

   o  Virtual Machine (VM)

   o  Infrastructure virtual network (which interconnects virtual
      machines with physical network interfaces or with each other
      through virtual switches, for example)
```



```
                                                         __
+--------------------------------------------------+   |
|                                                  |   |
|   +------------------------------------------+   |   |
|   |              Application                 |   |   |
|   +------------------------------------------+   |   |
|       ^                                  :       |   |
|       |                                  |       |   |  Guest
|       :                                  v       |   |
|   +---------------+           +---------------+  |   |
|   | Logical Port 0|           | Logical Port 1|  |   |
+---+---------------+-----------+---------------+--+ __|
        ^                                  :
        |                                  |
        :                                  v         __
+---+---------------+----------+---------------+---+   |
|   | Logical Port 0|          | Logical Port 1|   |   |
|   +---------------+          +---------------+   |   |
|       ^                                  :       |   |
|       |                                  |       |   |  Host
|       :                                  v       |   |
|   +--------------+            +--------------+   |   |
|   |   PHY Port   |  vSwitch   |   PHY Port   |   |   |
+---+--------------+------------+--------------+---+ __|
           ^                           :
           |                           |
           :                           v
+--------------------------------------------------+
|                                                  |
|                Traffic Generator                 |
|                                                  |
+--------------------------------------------------+
```

Figure 4: Physical Port to Virtual Switch to VNF to Virtual Switch Physical Port



```
                                                      __
+----------------------+  +----------------------+   |
|   Guest 1            |  |   Guest 2            |   |
|   +---------------+  |  |   +---------------+  |   |
|   |  Application  |  |  |   |  Application  |  |   |
|   +---------------+  |  |   +---------------+  |   |
|          |           |  |       ^              |   |
|          v           |  |       |              |   |  Guests
|   +---------------+  |  |   +---------------+  |   |
|   | Logical Ports |  |  |   | Logical Ports |  |   |
|   |           0   |  |  |   |   0           |  |   |
+---+---------------+--+  +---+---------------+--+ __|
            :                     ^
            |                     |
            v                     :                  _
+---+---------------+---------+---------------+--+ |
|   |           1   |         |   1           |  | |
|   | Logical Ports |         | Logical Ports |  | |
|   +---------------+         +---------------+  | |
|               |                   ^             | |  Host
|               \-------------------/             | |
|                                                 | |
|                    vSwitch                      | |
+-------------------------------------------------+_|
```

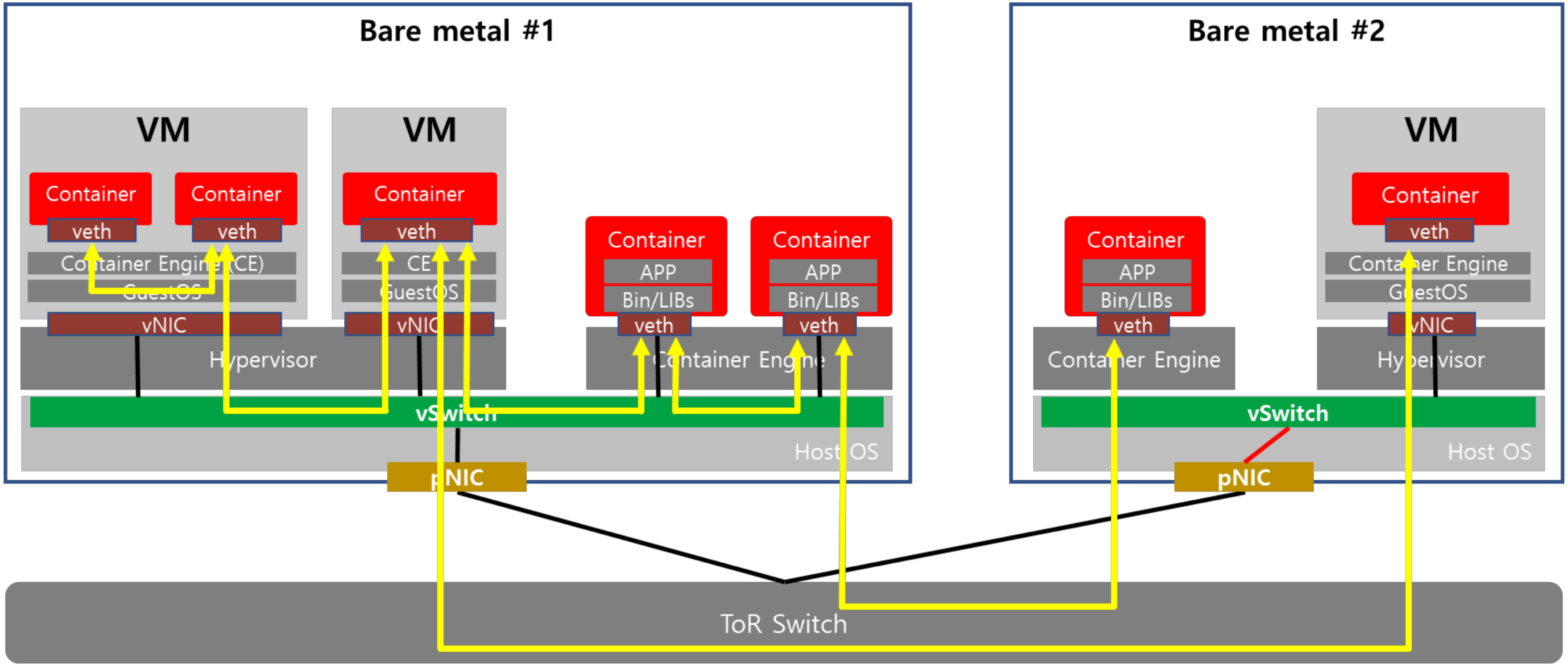
Figure 8: VNF to Virtual Switch to VNF

- Does it applicable for containerized infrastructure?
- Do test scenarios are covered also for containerized infrastructure?

Our Experience

- Network performance testing in containerized infrastructure
  - Deployment Environment
    - Deploy the container on Baremetal
    - Deploy the container on VM
  - OpenStack + Kubernetes Hybrid Environment
    - Creates POD using Kubernetes (baremetal & VM)
  - Network Feature
    - CNI – Flannel, Kuryr Networking, ..
    - Network Acceleration Feature(SR-IOV)
  - Network Service Type
    - VxLAN, VLAN, SR-IOV, offloading VxLAN

Test-bed Environment #1



Test-bed Environment #2

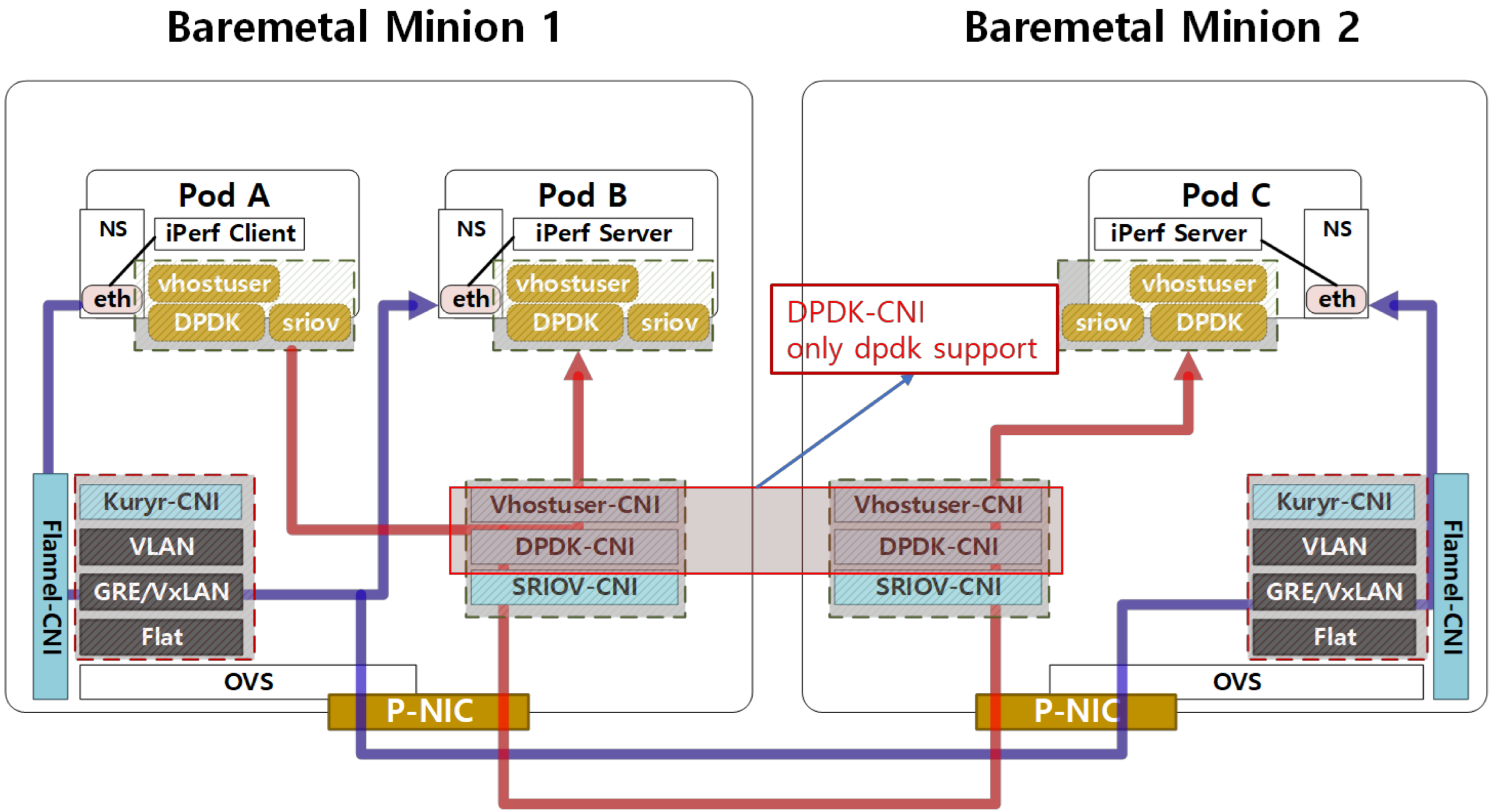
NODE	Classification	Specification
Baremetal (Master / Minion1 / Minion2)	CPU	Intel(R) Xeon(R) Gold 6148 2.40GHz * 2
	MEMORY	DDR4 2400 MHz 32GB * 6
	SR-IOV NIC	Mellanox ConnectX-5 (40G SFP+)
VM (Minion3 / Minion4)	CPU	Virtualized CPU * 8 (apply host-model)
	MEMORY	Virtualized MEM * 32GB
	NIC	vhost-net and sr-iov vf, vhost-user
System Software	OS	Ubuntu 16.04 Server LTS
	Cloud OS	Openstack queens by Devstack
	COE	kubernetes v1.9.0 and docker 18.06
	CNI	default cni plugin driver and kuryr, flannel, sr-iov, vshot-user, multus

Testing Scenarios

- BMP2BMP
  - Baremetal POD to Baremetal POD (local or remote)
- BMP2VMP
  - Baremetal POD to VM POD (local or remote)
- VMP2VMP
  - VM POD to VM POD (local or remote)
- Common Configuration
  - container image : ubuntu 16.04 (modified)
  - bandwidth tool : iperf or iperf3 (https://iperf.fr)
  - latency tool : sockperf (https://github.com/Mellanox/sockperf)

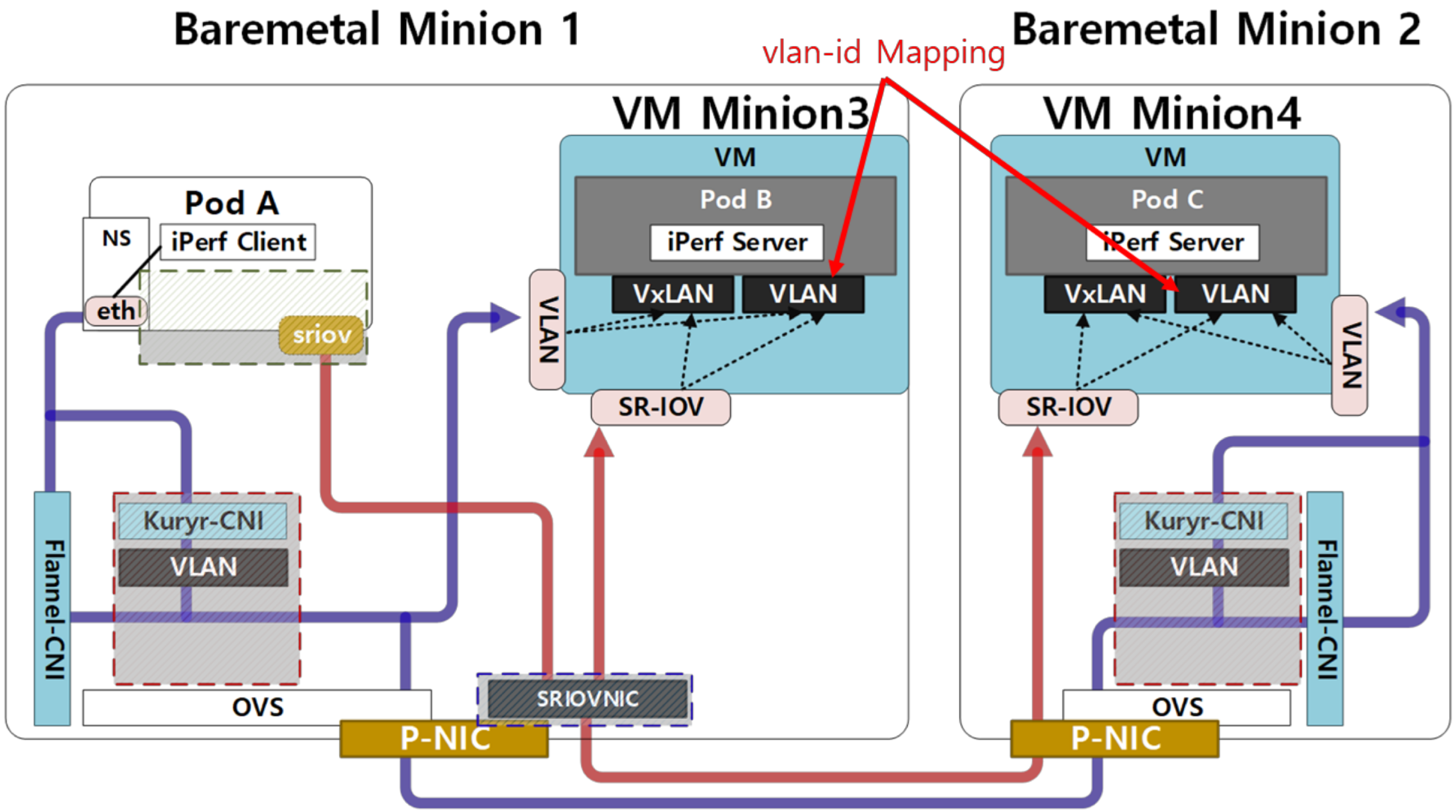
Scenario – BMP2BMP

- Networking Scenario
  - OpenStack-Kuryr (OVS bridge)
  - Flannel-CNI (docker bridge-Flannel bridge)
  - MACVLAN, IPVLAN / Data acceleration(SR-IOV)

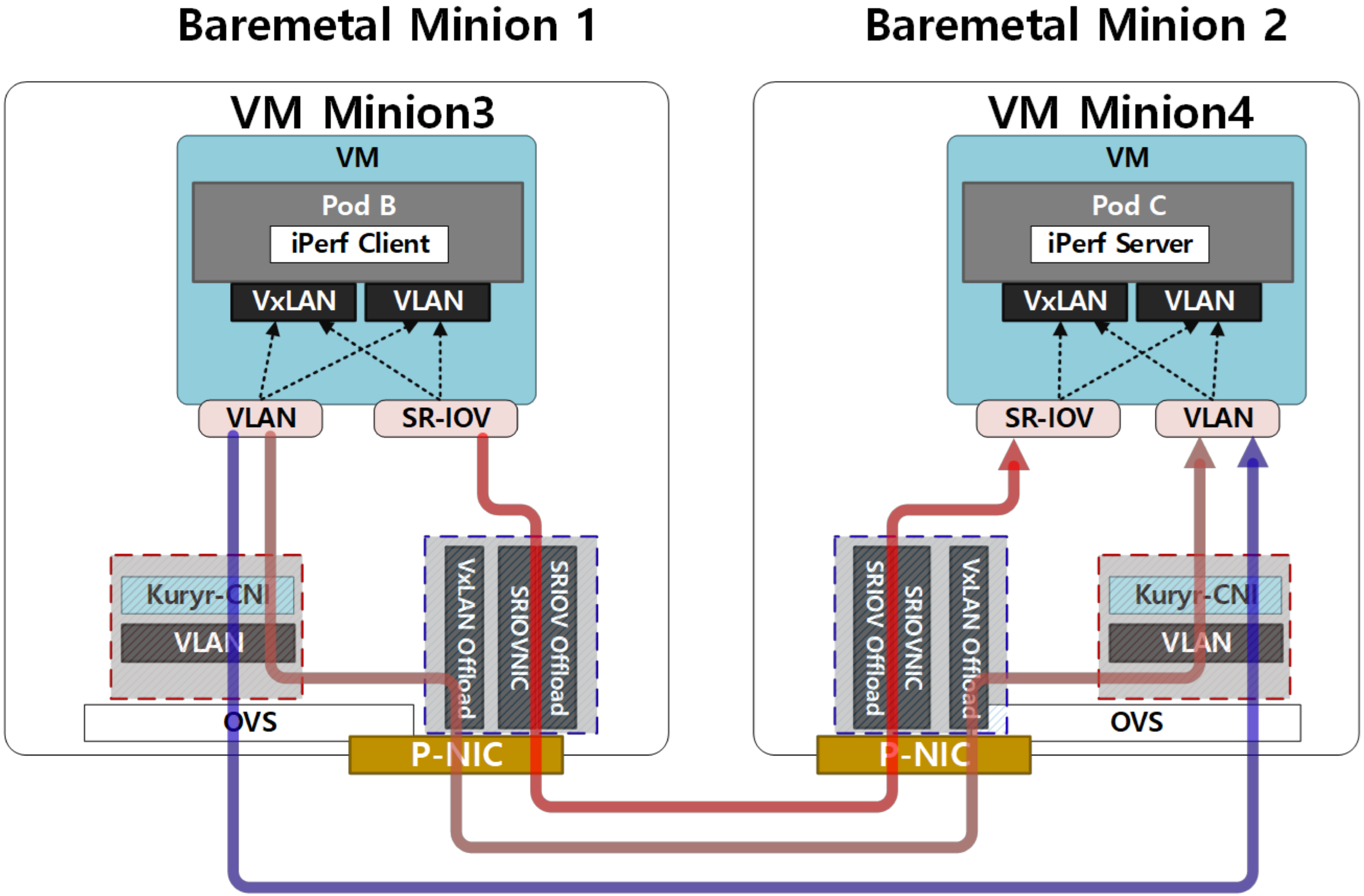


Scenario – BMP2VMP

- VM based Container Network
  - VxLAN and VLAN modules are running in guest VM (ovs bridge)
  - VM network port supports VLAN and SR-IOV

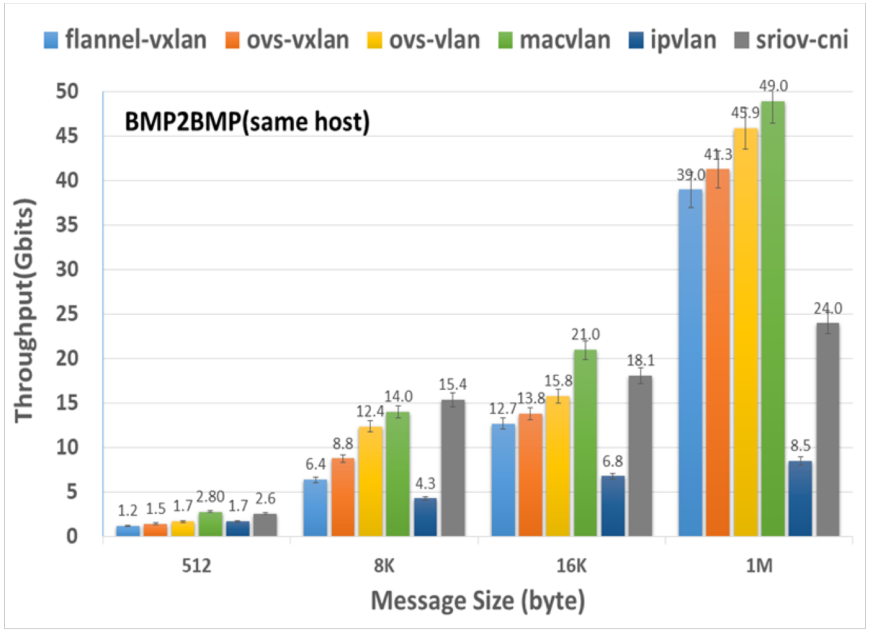


Scenario – VMP2VMP

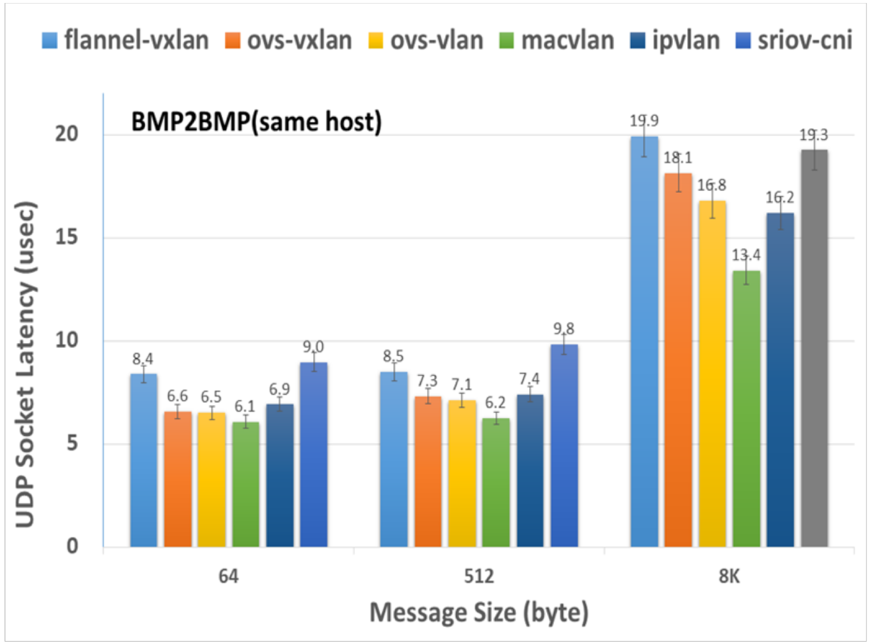


Result – BMP2BMP (local)

- VxLAN results
  - Ovs-vxlan > flannel-vxlan up to 10%
  - Overhead due to software processing of VxLAN packets
- VLAN results
  - Throughput : macvlan > ovs-vlan (20% lower) > SR-IOV > ipvlan
  - Latency : SRIOV(up to 16K) > ovs-vlan > ipvlan > macvlan



(a) Network traffic throughput of TCP by message size

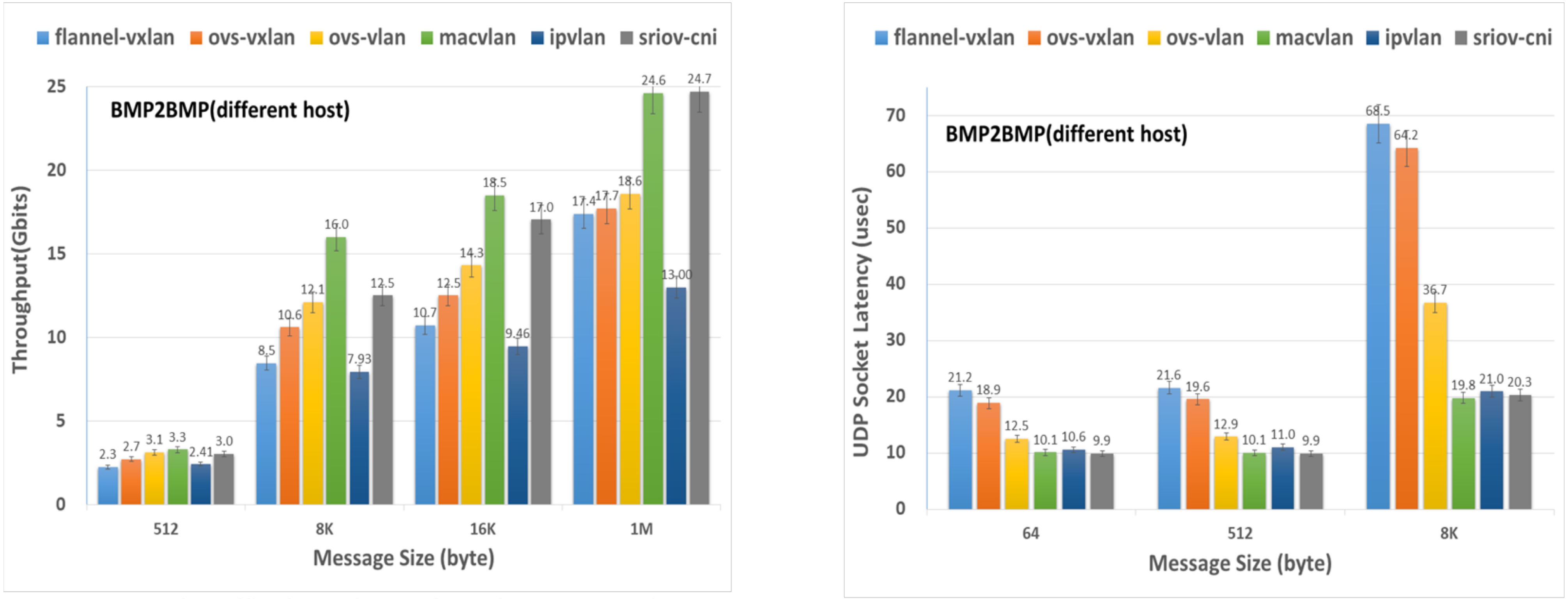


(b) UDP Socket latency by message size

Network Performance Evaluation between POD at the Same Bare Metal

Result - BMP2BMP (Remote)

- VxLAN results: ovs-vxlan > flannel-vxlan
- VLAN results: MACVLAN > ovs-vlan > ipvlan
  - SR-IOV cannot support RDMA (remote direct memory access)



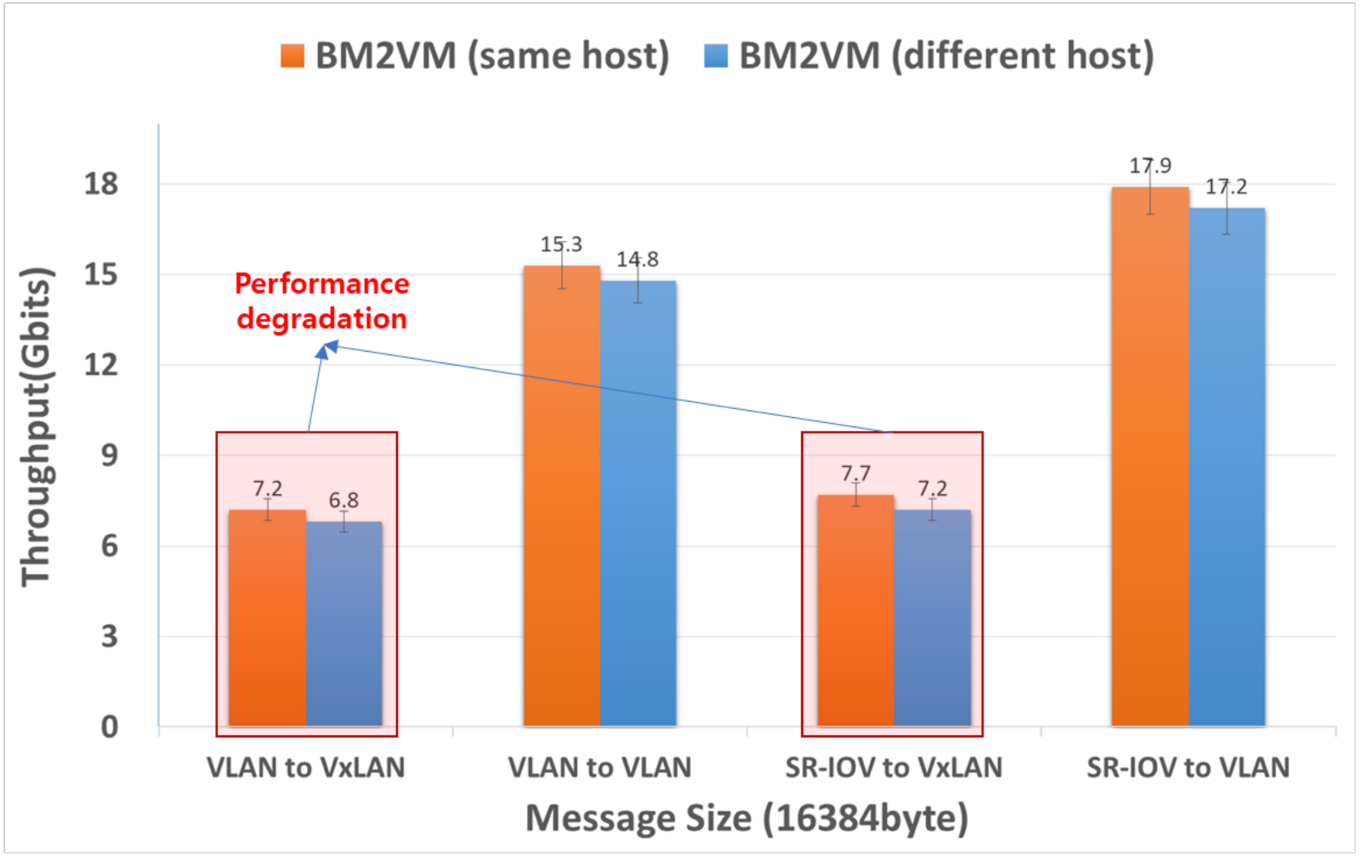
(a) Network traffic throughput of TCP by message size

(b) UDP Socket latency by message size

Network Performance Evaluation between POD at the Different Bare Metal

Result – BMP2VMP

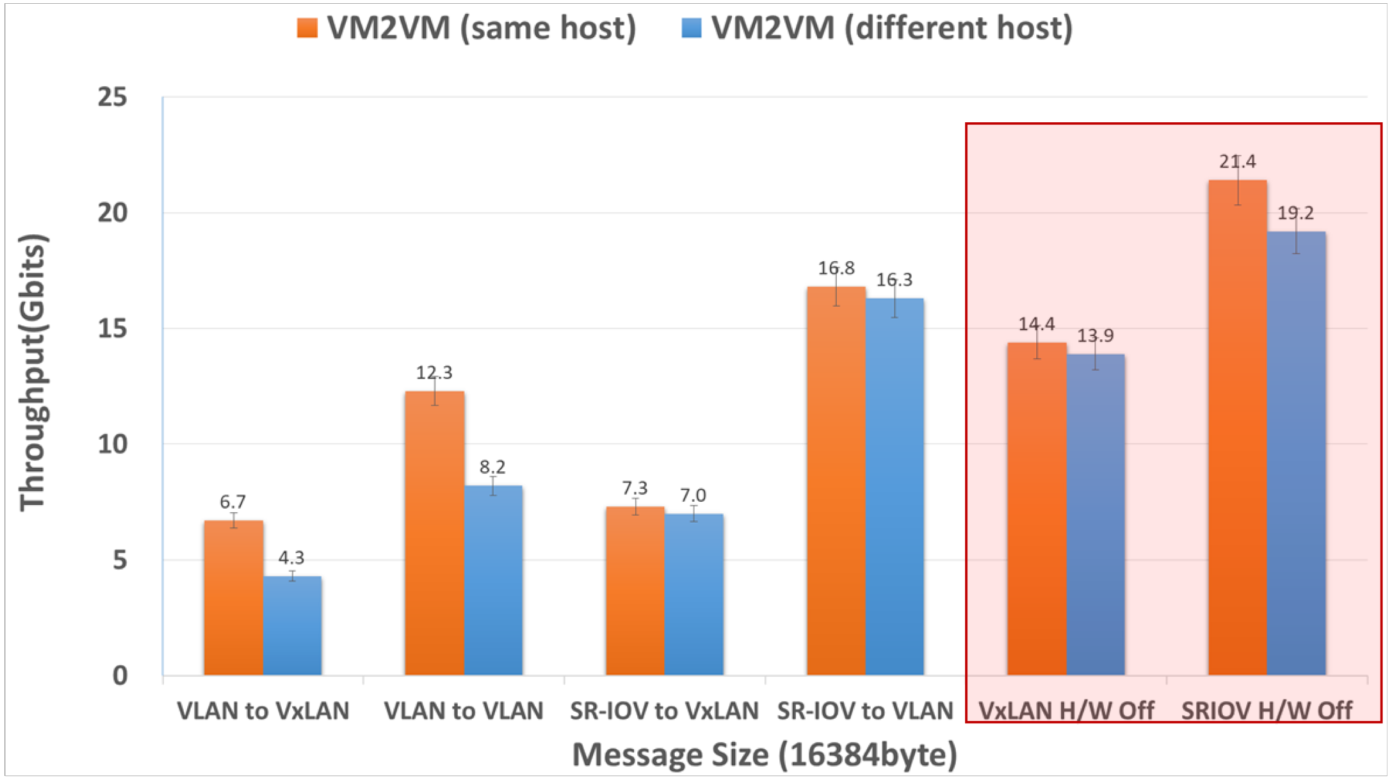
- Performance degradation by software processing of Vxlan in VM
  - Encap/Decap processing of VxLAN (for internal network)



Network Throughput between Baremetal and VM POD

Result – VMP2VMP

- In the case of VM, Best performance by applying hardware offload to SR-IOV and VxLAN.
  - Using H/W offloading, Encap/Decap process is done by hardware



VM based Container Network Throughput

Conclusion

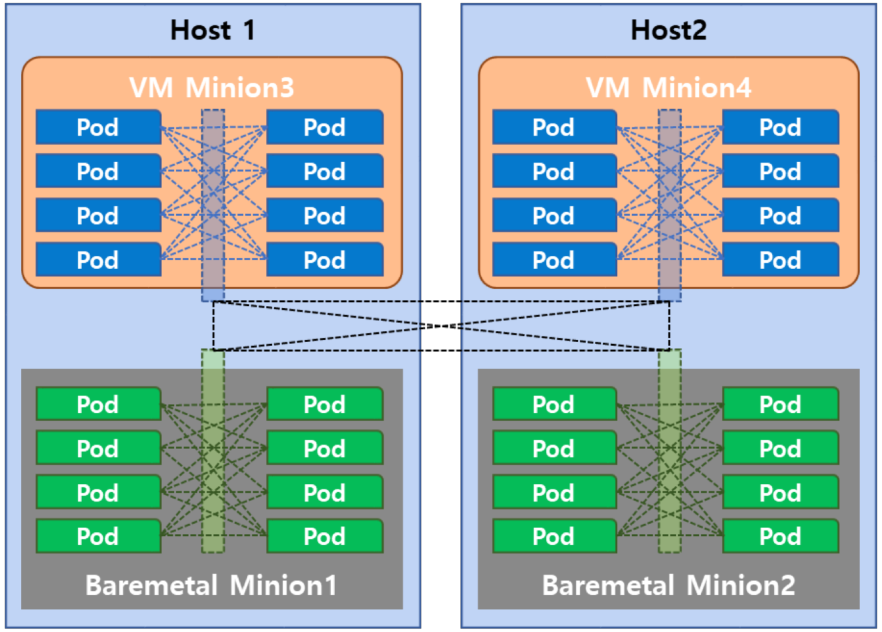
- What we learned
  - Containerized infrastructure have different isolation method
    - It may impact performance of VNF lifecycle management
  - Containerized infrastructures have several deployment options
    - POD / individual container (depends on container engine)
    - Running on VM / Baremetal
    - Testing scenarios will be different for each deployment models
- Our initial draft based on learning
  - But, we need more work to go forward
    - Including Test scenario, specific technologies, …
  - Feedbacks and reviews are always welcome
    - Thanks Al and Maciek for review before meeting!

Thankyou!

Backup slides

Parallel Paths Test

- Using Message Passing Interface(MPI)
  - Apply Collective communication (MPI_ALLTOALL)
  - 8 PODs in each host server
  - Measure latency of 2 socket processing on each POD (packet size=16KB)



- 1 pod = 1 Container
- Includes MPI library in Pod

Test Scenario

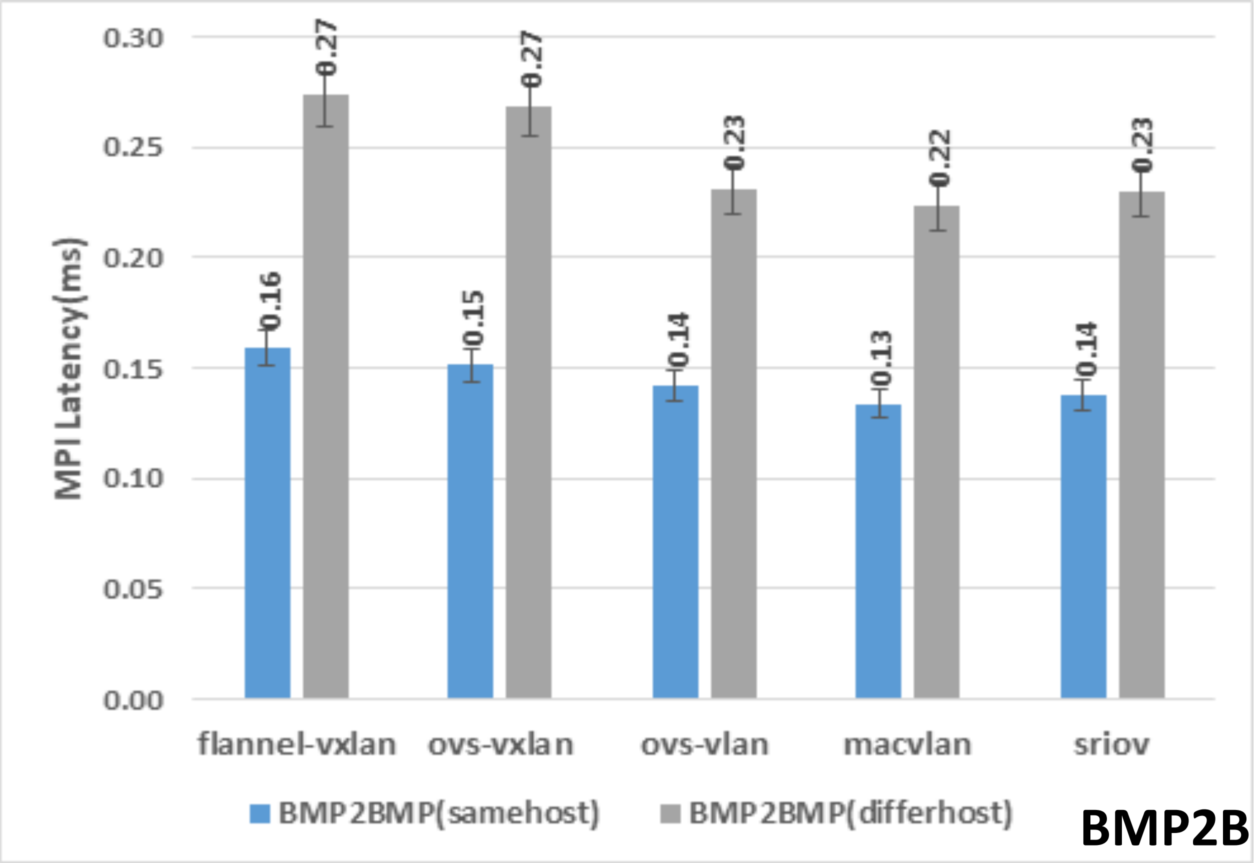
BMP2BMP

BMP2VMP

VMP2VMP

Testing Results (1)

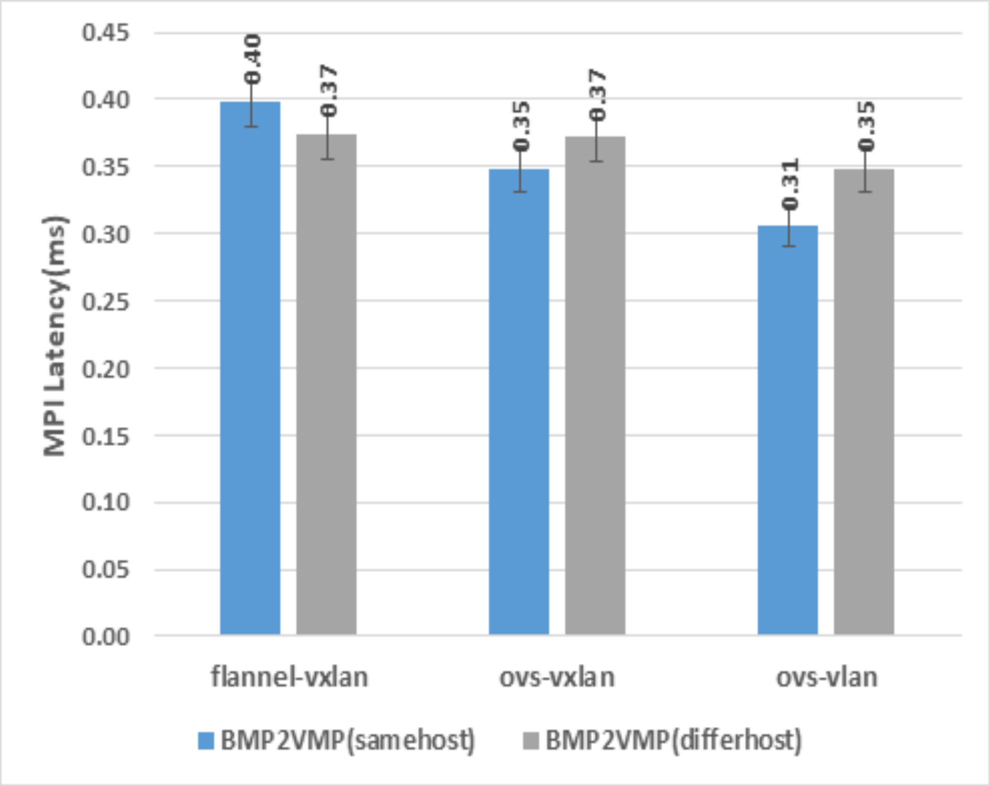
- VLAN technologies(ovs-vlan, macvlan, sriov) are shown better performance up to 10% than overlay network (vxlan) for all test scenarios.



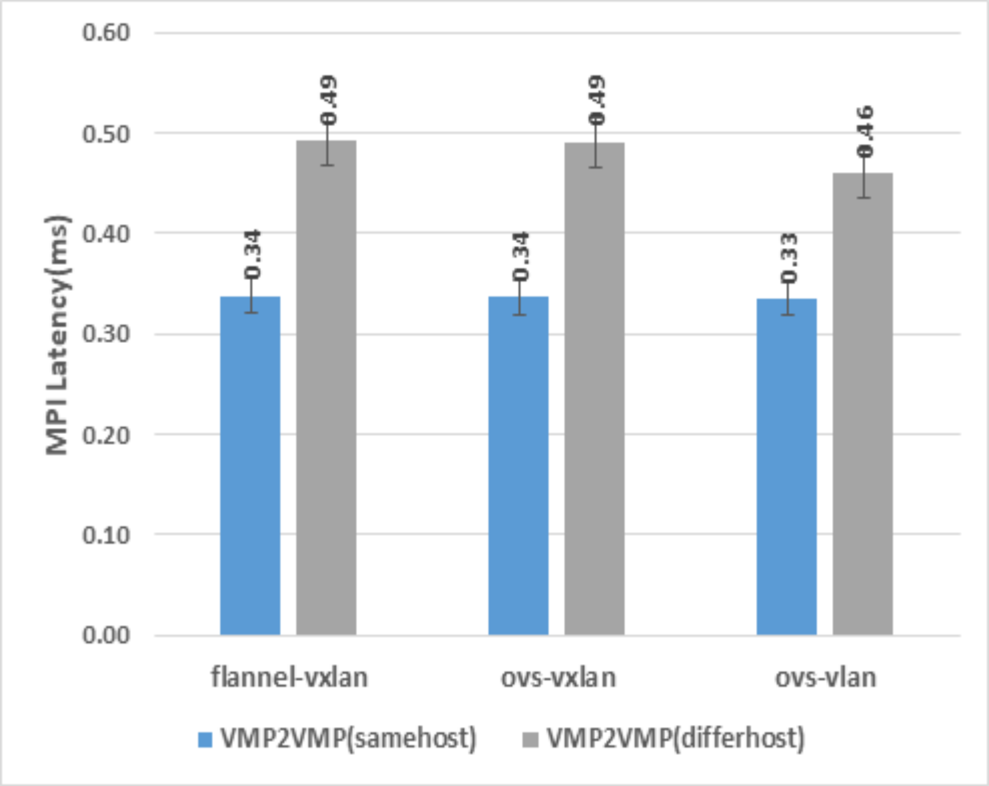
BMP2BMP

Testing Results (2)

BMP2VMP

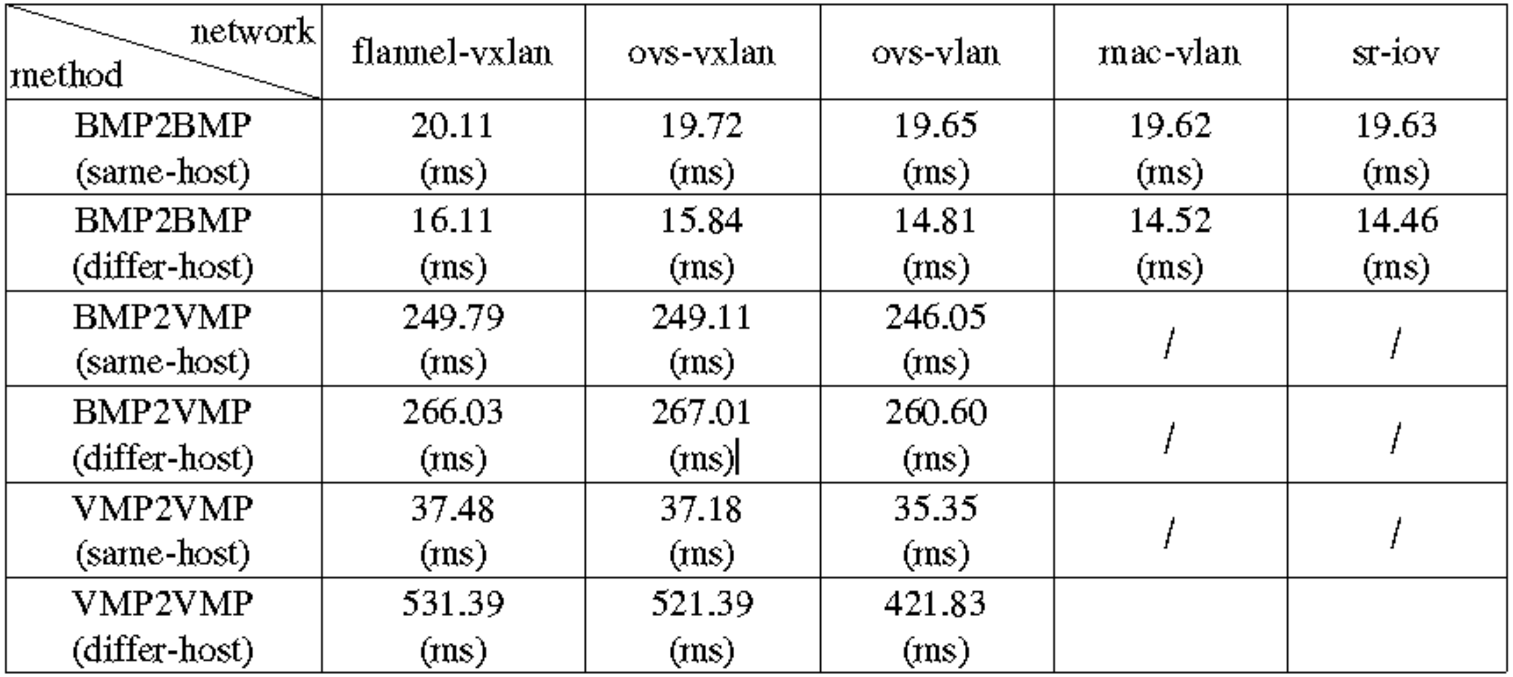


VMP2VMP



Results - Increase the process to four

- BMP2BMP – same host case results higher latency for increasing process load
- BMP2VMP – Parallel path created in BMP impacts latency for both case (same & different host case)
- VMP2VMP
  - In case of same-host, low latency since that parallel path are processed in host kernel via single interface



network method	flannel-vxlan	ovs-vxlan	ovs-vlan	mac-vlan	sr-iov
BMP2BMP (same-host)	20.11 (ms)	19.72 (ms)	19.65 (ms)	19.62 (ms)	19.63 (ms)
BMP2BMP (differ-host)	16.11 (ms)	15.84 (ms)	14.81 (ms)	14.52 (ms)	14.46 (ms)
BMP2VMP (same-host)	249.79 (ms)	249.11 (ms)	246.05 (ms)	/	/
BMP2VMP (differ-host)	266.03 (ms)	267.01 (ms)	260.60 (ms)	/	/
VMP2VMP (same-host)	37.48 (ms)	37.18 (ms)	35.35 (ms)	/	/
VMP2VMP (differ-host)	531.39 (ms)	521.39 (ms)	421.83 (ms)		