Considerations for Benchmarking Network Performance in Containerized Infrastructures
draft-dcn-bmwg-containerized-infra-01

Abstract

   This draft describes benchmarking considerations for the
   containerized infrastructure. In the containerized infrastructure,
   Virtualized Network Functions(VNFs) are deployed on operating-system-
   level virtualization platform by abstracting the user namespace as
   opposed to virtualization using a hypervisor. Leveraging this, the
   system configurations and networking scenarios for benchmarking will
   be partially changed by the way in which the resource allocation and
   network technologies speicifed for containerized VNFs. In this draft
   we compare the state of the art in a container networking
   architecture with networking on VM-based virtualized systems, and
   provide several test scenarios in the containerized infrastructure.

Status of This Memo

Table of Contents

1. Introduction

   The Benchmarking Methodology Working Group(BMWG) has recently
   expanded its benchmarking scope from Physical Network Function(PNF)
   running on dedicated hardware system to Network Function
   Virtualization(NFV) infrastructure and Virtualized Network
   Function(VNF). [RFC8172] described considerations for configuring
   NFV infrastructure and benchmarking metrics, and [RFC8204] gives
   guidelines for benchmarking virtual switch which connects VNFs in
   Open Platform for NFV(OPNFV).

   Recently NFV infrastructure has evolved to include a lightweight
   virtualized platform called the containerized infrastructure, where
   VNFs share the same host Operating System(OS) and they are logically
   isolated by using a different namespace. While previous NFV
   infrastructure uses a hypervisor to allocate resources for Virtual
   Machine(VMs) and instantiate VNFs on it, the containerized
   infrastructure virtualizes resources without a hypervisor, therefore

making containers very lightweight and more efficient in
infrastructure resource utilization compared to the VM-based NFV
infrastructure. When we consider benchmarking for VNFs in the
containerized infrastructure, it may have a different System Under
Test(SUT) and Device Under Test(DUT) configuration compared with both
black-box benchmarking and VM-based NFV infrastructure as described
in [RFC8172]. Accordingly, additional configuration parameters and
testing strategies may be required.

In the containerized infrastructure, a VNF network is implemented by
running both switch and router functions in the host system. For
example, the internal communication between VNFs in the same host
uses the L2 bridge function, while communication with external
node(s) uses the L3 router function. For container networking, the
host system may use a virtual switch(vSwitch), but other options
exist. In the [ETSI-TST-009], they describe differences in
networking structure between the VM-based and the containerized
infrastructure. Occasioned by these differences, deployment
scenarios for testing network performance described in [RFC8204] may
be partially applied to the containerized infrastructure, but other
scenarios may be required.

In this draft, we describe differences and additional considerations
for benchmarking containerized infrastructure based on [RFC8172] and
[RFC8204]. In particular, we focus on differences in system
configuration parameters and networking configurations of the
containerized infrastructure compared with VM-based NFV
infrastructure. Note that, although the detailed configurations of
both infrastructures differ, the new benchmarks and metrics defined
in [RFC8172] can be equally applied in containerized infrastructure
from a generic-NFV point of view, and therefore defining additional
metrics or methodologies is out of scope.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document is to be interpreted as described in [RFC2119]. This
document uses the terminology described in [RFC8172], [RFC8204],
[ETSI-TST-009].

3. Benchmarking Considerations

3.1. Comparison with the VM-based Infrastructure

For the benchmarking of the containerized infrastructure, as
mentioned in [RFC8172], the basic approach is to reuse existing
benchmarking methods developed within the BMWG. Various network

function specifications defined in BMWG should still be applied to
containerized VNF(C-VNF)s for the performance comparison with
physical network functions and VM-based VNFs.

```
+-------------------------------+   +--------------------------------+
|+--------------+ +--------------+|  |+------------+    +------------+|
||  Guest VM   | |  Guest VM   ||  || Container  |    | Container  ||
||+------------+| |+------------+||  ||+----------+|    |+----------+||
|||   APP    || ||   APP    |||  |||  APP    ||    ||   APP   |||
||+------------+| |+------------+||  ||+----------+|    |+----------+||
||+------------+| |+------------+||  ||+----------+|    |+----------+||
|||Guest Kernel|| ||Guest Kernel|||  ||| Bin/Libs ||    || Bin/Libs |||
||+------------+| |+------------+||  ||+----------+|    |+----------+||
|+------^-------+ +-------^------+|  |+-----^------+    +------^-----+|
|+------|----------------|------+|  |+-----|-----------------|-----+|
||      |   Hypervisor   |      ||  ||     |+---------------+|     ||
|+------|----------------|------+|  ||     ||Container Engine||     ||
|+------|----------------|------+|  ||     |+---------------+|     ||
||      | Host OS Kernel |      ||  ||     |  Host OS Kernel |     ||
|+------|----------------|-----+||  |+-----|-----------------|-----+|
|    +--v---------------v--+   |   |  +---v-----------------v---+  |
+----|   physical network  |----+  +--|   physical network     |--+
     +---------------------+            +------------------------+
   (a) VM-Based Infrastructure    (b) Containerized Infrastructure
```
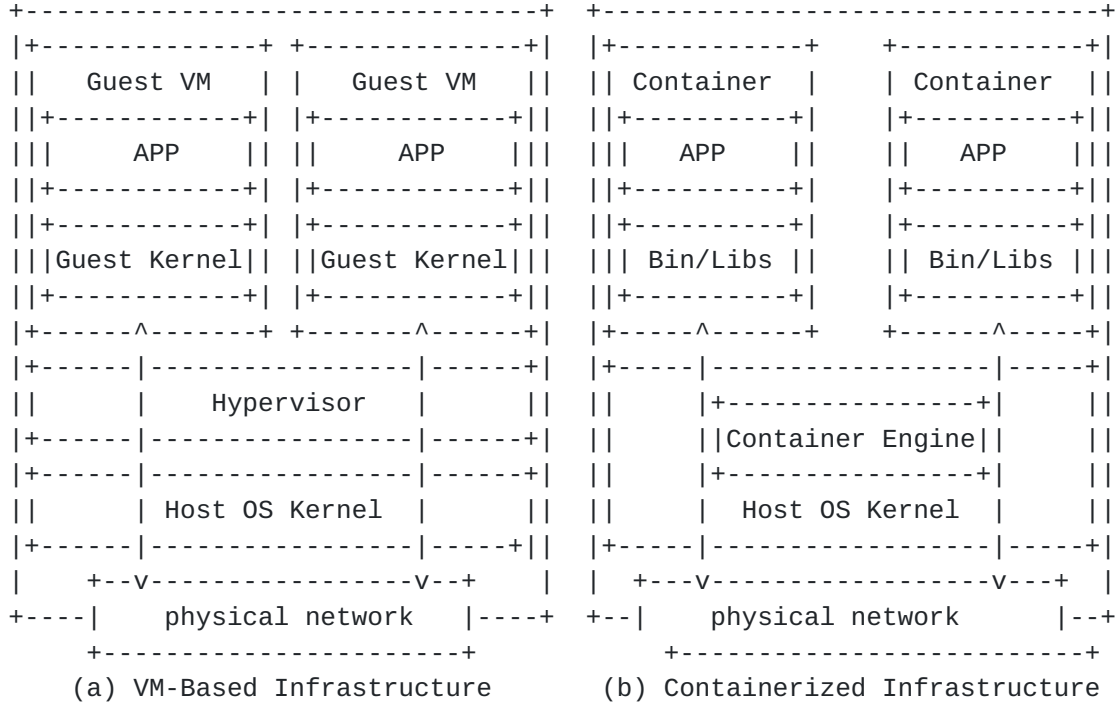
               Figure 1: Comparison of NFV Infrastructures

   In Figure 1, we describe two different NFV architectures: VM-based
   and Containerized. A major distinction between the containerized and
   the VM-based infrastructure is that with the former, all VNFs share
   same host resources including but not limited to computing, storage
   and networking resources, as well as the host Operating System(OS),
   kernel and libraries. The absence of the guest OS and the
   hypervisor, necessitates the following considerations that occur in
   the test environment:

   o When we consider hardware configurations for the containerized
   infrastructure, all components described in [RFC8172] can be part of
   the test setup. While capabilities of servers and storages should
   meet the minimum requirements for testing, it is possible to deploy a
   test environment with less capabilities than in the VM-based
   infrastructure.

   o About configuration parameters, the containerized infrastructure
   needs specified management system instead of a hypervisor(e.g. Linux
   Container, Docker Engine).

o In the VM-based infrastructure, each VM manipulates packets in the
kernel of the guest OS through its own CPU threads, virtualized and
assigned by the hypervisor. On the other hand, C-VNFs use the host
CPU without virtualization. Different CPU resource assignment
methods may have different CPU utilization perspectives for the
performance benchmarking.

o From a Memory Management Unit(MMU) point of view, there are
differences in how the paging process is conducted between two
environments. The main difference lies in the isolated nature of the
OS for VM-based VNFs. In the containerized infrastructure, memory
paging which processes conversion between physical address and
virtual address is affected by the host resource directly. Thus,
memory usage of each C-VNFs is more dependent on the host resource
capabilities than in VM-based VNFs.

3.2. Container Networking Classification

Container networking services are provided as network plugins.
Basically, using them, network services are deployed by using
isolation environment from container runtime through the host
namespace, creating virtual interface, allocating interface and IP
address to C-VNF. Since the containerized infrastructure has
defferent network architecture depending on its using plugins, it is
necessary to specify the plugin used in the infrastructure. There
are two proposed models for configuring network interfaces for
containers as belows;

o CNM(Container Networking Model) proposed by Docker, using
libnetwork which provides an interface between the Docker daemon and
network drivers.

o CNI(Conainer Network Interface) proposed by CoreOS, describing
network configuration files in JSON format and plugins are
instantiated as new namespaces. Kubernetes uses CNI for providing
network service.

Regardless of both CNM and CNI, container network model can be
classified into kernel space network model and user space network
model according to the location of network service creation. In case
of kernel-based network model, network interfaces are created in
kernel space so that data packets should be processed in network
stack of host kernel before transferring packets to the C-VNF running
in user space. On the other hand, using user-based network model,
data packets from physical network port are bypassed kernel
processing and delivered directly to user space. Specific
technologies for each network model and example of network
architecture are written as follows:

o Kernel space network model: Docker Network[Docker-network], Flannel
  Network[Flannel], Calico[Calico], OVS(OpenvSwitch)[OVS], OVN(Open
  Virtual Network)[OVN], eBPF[eBPF]

```
+------------------------------------------------------------------+
| User Space                                                       |
|    +----------+                          +----------+   |
|    | Container |                          | Container |   |
|    | +-------+ |                          | +-------+ |   |
|    +-| eth   |-+                          +-| eth   |-+   |
|      +--^----+                             +----^--+      |
|         |   +------------------------------------------+  |      |
|         |   |    vSwitch                               |  |      |
|         |   | +------------------------------------+ |  |      |
|         |   | |              +--v---v---v--+        | |  |      |
|         |   | |bridge        |    tag[n]   |        | |  |      |
|         |   | |              +--^-------^--+        | |  |      |
|         |   | +--^------------|-------|-----------^-+ |  |      |
|         |   | |    |         +---+      +---+      |   |  |      |
|         |   | |    | +------ v-----+ +-------v----+ |   |  |      |
|         |   | |    | |tunnel bridge| | flat bridge | |   |  |      |
|         |   | |    | +------^------+ +-------^-----+ |   |  |      |
|         |   +--- |--------|---------------|-------|---+   |  |      |
---------|------ |--------|---------------|-------|------|---------
|    +----|-------|--------|---------------|-------|------|----+   |
|    | +--v-------v--+     |               |     +--v------v--+ |   |
|    | |    veth     |     |               |     |    veth    | |   |
|    | +---^---------+     |               |     +---^--------+ |   |
|    | Kernel Datapath     |               |                   |   |
|    +--------------------|---------------|-----------------+   |
|                         |               |                     |
| Kernel Space       +--v----------------v--+                   |
+--------------------|          NIC         |-------------------+
                     +----------------------+
```
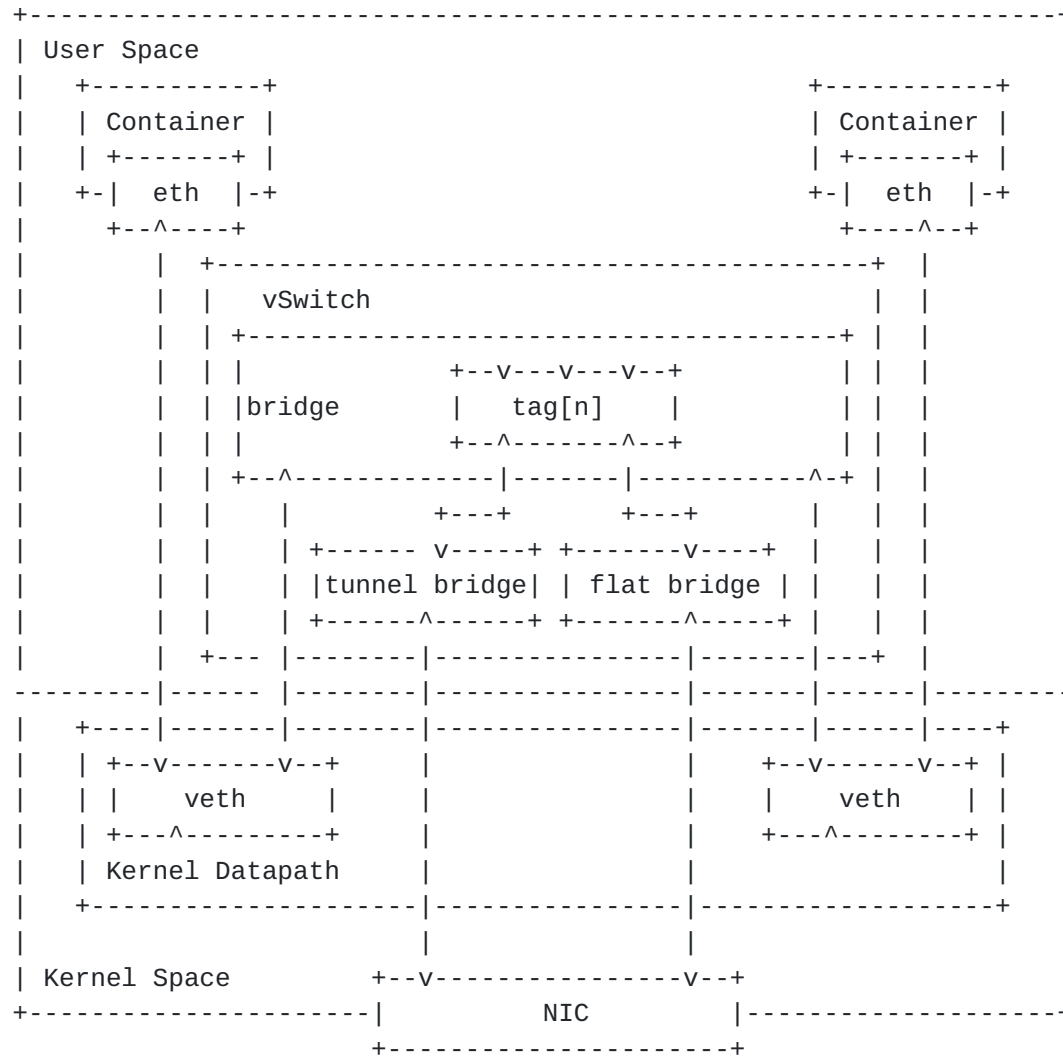
                Figure 2: Examples of Kernel Space Network Model

o User space network model - Device pass-through model: SR-
  IOV[SR-IOV]

```
   +-------------------------------------------------------------------+
   | User Space                                                        |
   |     +----------------+                   +----------------+    |
   |     |    Container    |                   |    Container   |    |
   |     | +------------+ |                   | +------------+ |    |
   |     +-|  vf driver  |-+                   +-|  vf driver  |-+    |
   |       +-----^-------+                     +------^------+       |
   |             |                                   |              |
   -------------|-----------------------------------|--------------
   |           +---------+                   +---------+            |
   |              +------|------------------|------+                |
   |              | +----v-----+       +-----v----+ |               |
   |              | | virtual  |       | virtual  | |               |
   |              | | function |       | function | |               |
   | Kernel Space | +----^-----+  NIC  +-----^----+ |               |
   +--------------|      |                   |      |---------------+
                  | +----v-------------------v----+ |
                  | |        Classify and Queue    | |
                  | +----------------------------+ |
                  +------------------------------+
```
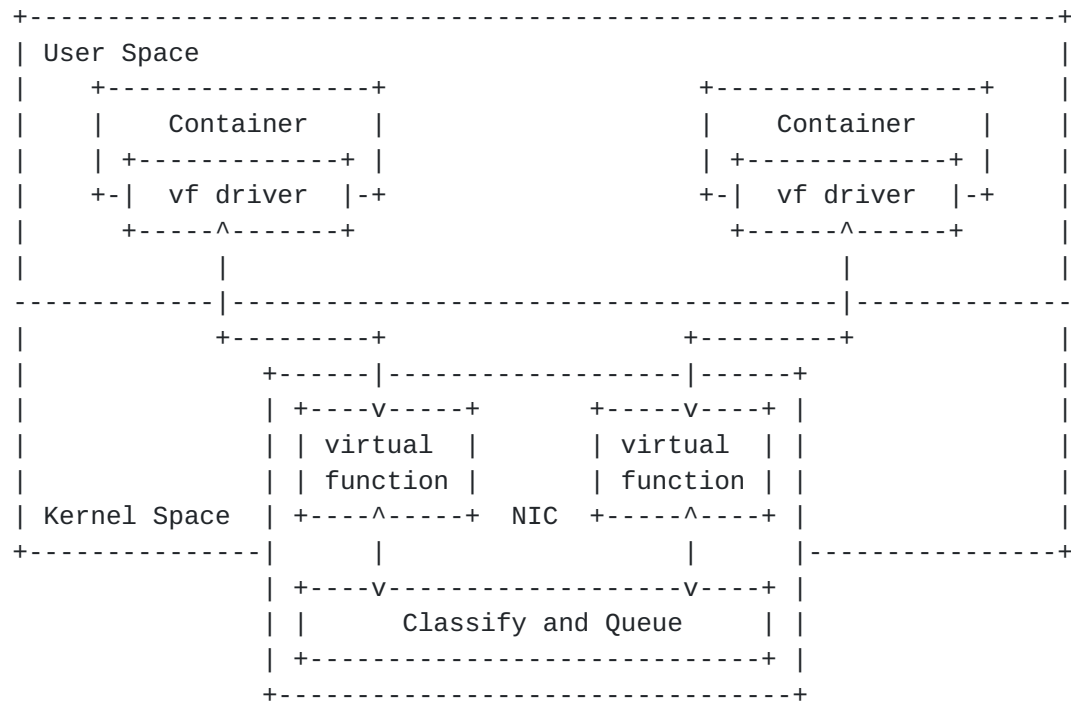
Figure 3: Examples of User Space Netowrk Model - Device Pass-through

- vSwitch model: ovs-dpdk[ovs-dpdk], vpp[vpp], netmap[netmap]

```
+-----------------------------------------------------------------+
| User Space                                                      |
|     +-----------------+                   +-----------------+   |
|     |    Container    |                   |    Container    |   |
|     | +-------------+ |                   | +-------------+ |   |
|     +-| virtio-user |-+                   +-| virtio-user |-+   |
|       +-----^-------+                       +-------^-----+     |
|             |                                       |          |
|           +---------+                     +---------+          |
|     +----------------|-------------------|-----------------+   |
|     |   vSwitch      |                   |                 |   |
|     |         +-------v-----+       +-----v-------+         |   |
|     |         | virtio-user |       | virtio-user |         |   |
|     |         +-------^-----+       +-----^-------+         |   |
|     |     +-----------|-------------------|-------------+   |   |
|     |     |     +--v--------------------v---+           |   |   |
|     |     |bridge   |        tag[n]         |           |   |   |
|     |     |     |     +-----------^-------------+       |   |   |
|     |     +---------------------|-----------------------+   |   |
|     |                  +-------v--------+                   |   |
|     |                  | dpdk0 bridge  |                   |   |
|     |                  +-------^--------+                   |   |
|     +----------------------------|----------------------------+ |
|                        +-------v--------+                       |
|                        |   DPDK PMD     |                       |
|                        +-------^--------+                       |
-------------------------------|---------------------------------
| Kernel Space                 +-----v------+                    |
+------------------------------|    NIC     |--------------------+
                               +------------+
```
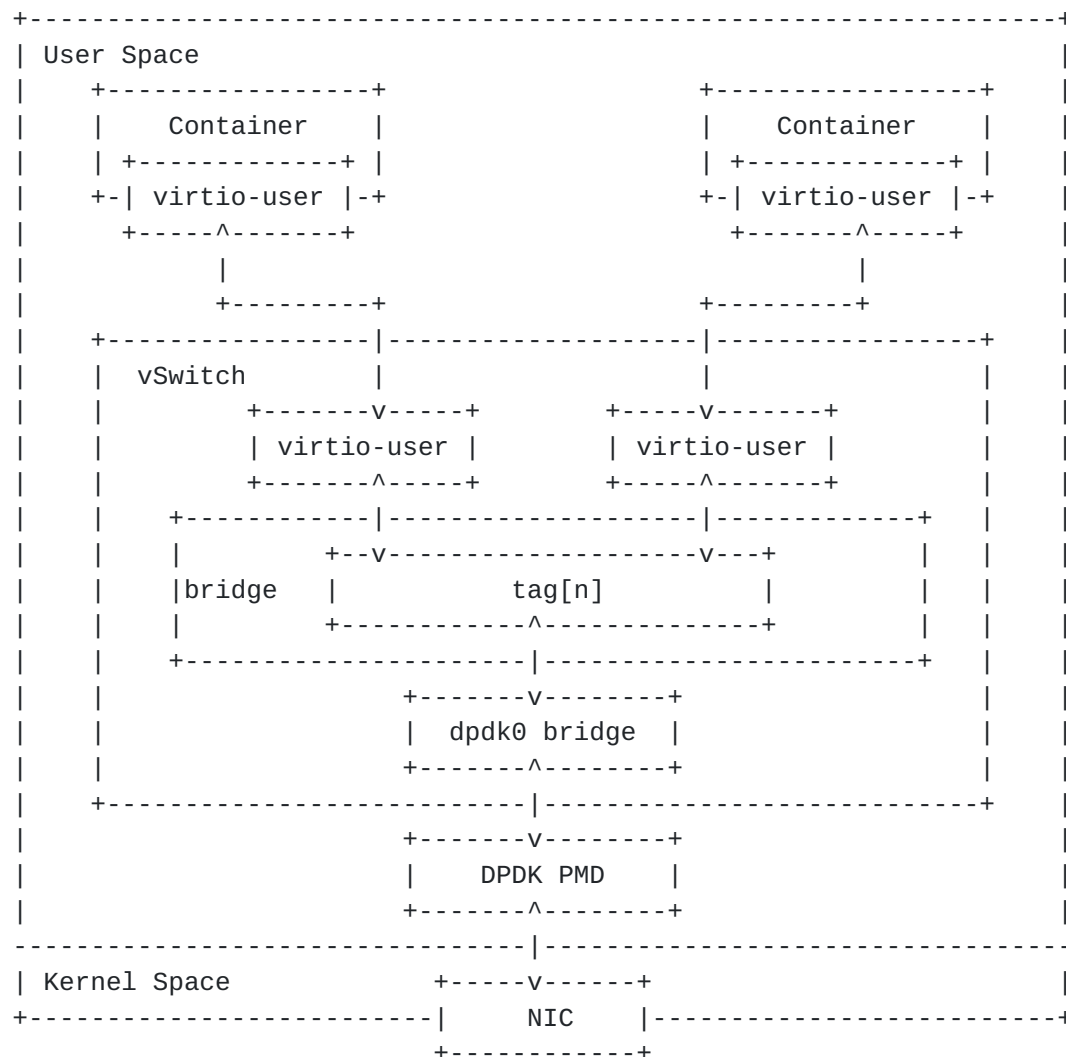
Figure 4: Examples of User Space Netowrk Model - vSwitch Model using
DPDK

3.3. Resource Considerations

In the containerized infrastructure, resource utilization and
isolation may have different charateristics compared with the VM-
based infrastructure. Some details are listed as follows:

o Hugepage: When using Cent OS or RedHat OS in the VM-based
infrastructure, Hugepage should be set to at least 1G byte. In the
containerized infrastructure, container is isolated in the
application level so that administrators can set Hugepage more
granular level(e.g 2M, 4M, ...). In addition, since the increase of
the Hugepage can affect the Translation Lookaside Buffer (TLB) miss,
the value of the Hugepage should be taken into account in the
performance measurement. Moreover, benchmarking results may vary

according to Hugepage set value of kernel space model and user space
model in the containerized infrastructure so that Hugepage values
should be considered when we configure test environment.

o NUMA: NUMA technology can be used both in the VM-based and
containerized infrastructure. However, the containerized
infrastructure provides more variable options than the VM-based
infrastructure such as kernel memory, user memory, and CPU setting.
Instantiation of C-VNFs is somewhat non-deterministic and apparently
NUMA-Node agnostic, which is one way of saying that performance will
likely vary whenever this instantiation is performed. So, when we
use NUMA in the containerized infratructure, repeated instantiation
and testing to quantify the performance variation is required.

o RX/TX Multiple-Queue: RX/TX Multiple-Queue technology[Multique],
which enables packet sending/receiving processing to scale with
number of available vcpus of guest VM, may be used to enhance network
performance in the VM-based infrastructure. However, RX/TX Multiple-
Queue technology is not supported in the containerized infrastructure
yet.

[4](#). Benchmarking Scenarios for the Containerized Infrastructure

Figure 5 shows briefly differences of network architectures based on
deployment models. Basically, on baremetal, C-VNFs can be deployed
as a cluster called POD by Kubernetes, otherwise each C-VNF can be
deployed separately using Docker. In former case, there is only one
external network interface even a POD contains more than one C-VNF.
An additional deployment model considers a scenario in which C-VNFs
or PODs are running on VM. In our draft, we define new
terminologies; BMP which is Pod on baremetal and VMP which is Pod on
VM.

```
+------------------------------------------------------------------+
|                        Baremetal Node                            |
|                                                                  |
| +--------------+ +--------------+ +------------- + +------------+ |
| |              | |     POD      | |     VM       | |     VM     | |
| |              | |+------------+| |+------------+| |  +-------+  | |
| |   C-VNF(A)   | || C-VNFs(B)  || || C-VNFs(C)  || |  |PODs(D)|  | |
| |              | |+------------+| |+-----^------+| |  +---^---+  | |
| |              | |  |           | |     |        | |      |     | |
| |    +------+  | |    +------+   | |   +--v---+   | |  +---v--+  | |
| +---| veth |---+ +---| veth |---+ +---|virtio|----+ +--|virtio|---+ |
|     +--^---+         +---^--+        +--^---+          +---^--+     |
|        |               |               |                 |        |
|        |               |             +--v---+          +---v--+    |
| +------|---------------|-----------|vhost |---------|vhost |---+ |
| | |    |               |             +--^---+          +---^--+ | |
| | |    |               |               |                 |     | |
| | |  +--v---+       +---v--+         +--v---+          +---v--+ | |
| | +-| veth |---------| veth |---------| Tap  |---------| Tap  |-+ | |
| | | | +--^---+         +---^--+         +--^---+          +---^--+ | | |
| | | |   |               |  vSwitch    |               |     | | | |
| | | | +--|---------------|---------------|---------------|--+ | | |
| | | +-| |               |  Bridge      |               |  |-+ | | |
| | |   +--|---------------|---------------|---------------|--+   | |
| | |      |  +---------+   |              +--|---------------|---+  | |
| | |      |  |Container|   |              |  |   Hypervisor  |   |  | |
| | |      |  | Engine  |   |              |  |               |   |  | |
| | |      |  +---------+   |              +--|---------------|---+  | |
| | |      |               |  Host Kernel |               |      | |
| +------|---------------|--------------|---------------|------+ |
|     +--v----------------v--------------v---------------v--+      |
+-----|                  physical network                   |-----+
      +---------------------------------------------------------+
```
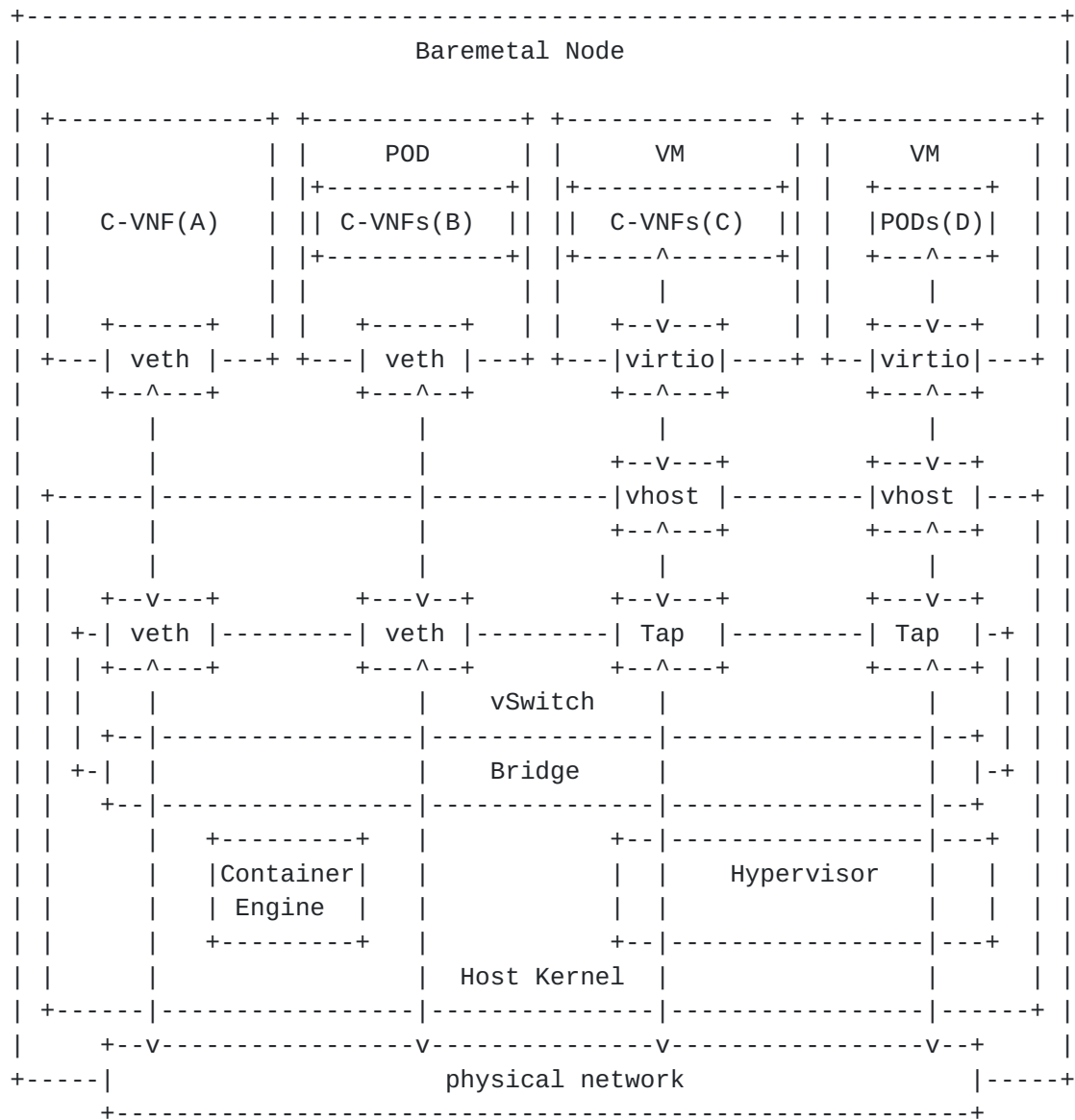
    Figure 5: Examples of Networking Architecture based on Deployment
   Models - (A)C-VNF on Baremetal (B)Pod on Baremetal(BMP) (C)C-VNF on
                      VM (D)Pod on VM(VMP)

  In [ETSI-TST-009], they described data plane test scenarios in a
  single host. In that document, there are two scenario for
  containerized infrastucture; Container2Container which is internal
  communication between two containers in the same Pod, and Pod2Pod
  model which is communication between two containers runnung in
  different Pods. According to our new terminologies, we can call
  Pod2Pod model as BMP2BMP scenario. When we consider container
  running on VM as an additional deployment option, there can be more
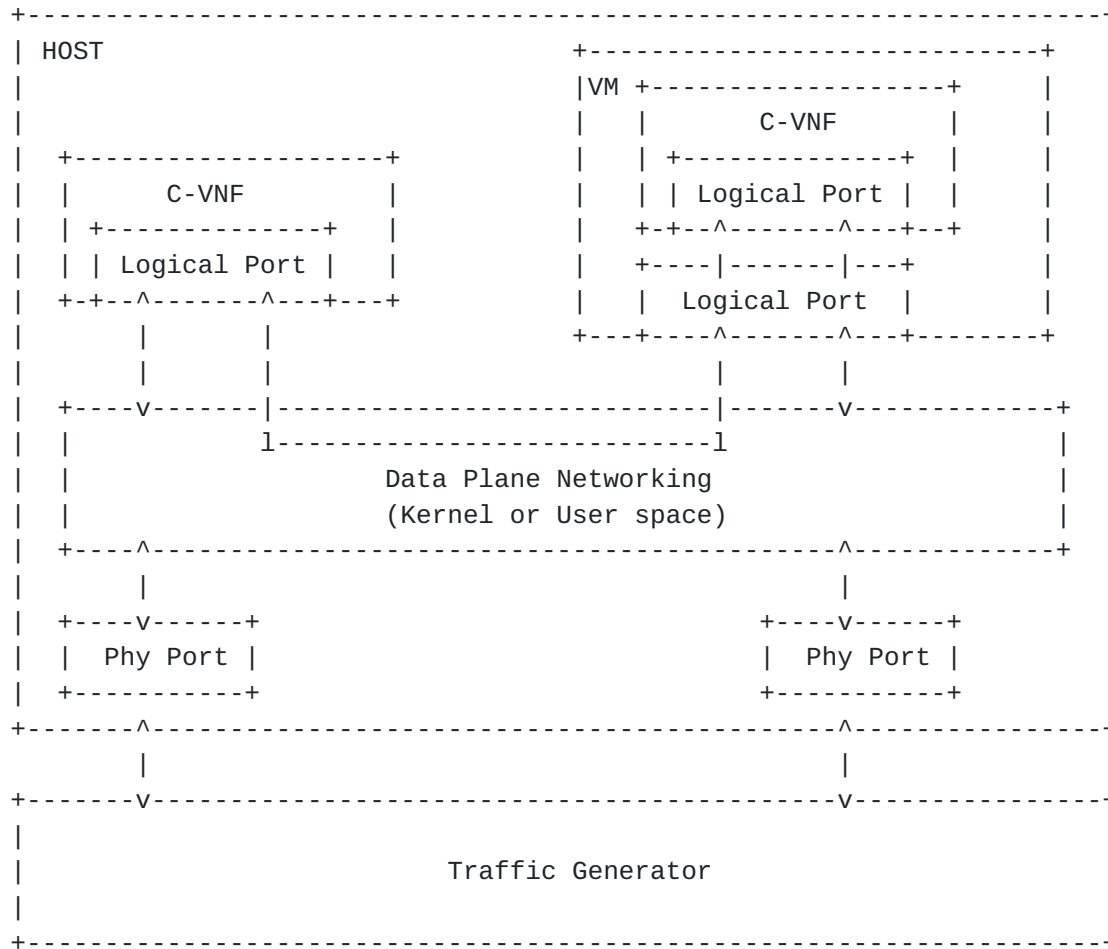 single host test scenarios as follows;

o BMP2VMP scenario

```
+----------------------------------------------------------------------+
| HOST                               +------------------------------+  |
|                                    |VM +-------------------+      |  |
|                                    |   |       C-VNF       |      |  |
|   +-------------------+            |   | +-------------+   |      |  |
|   |      C-VNF        |            |   | | Logical Port |  |      |  |
|   | +-------------+   |            |   +-+--^-------^---+--+      |  |
|   | | Logical Port |  |            |   +----|-------|---+        |  |
|   +-+--^-------^---+---+            |   | Logical Port  |        |  |
|        |       |                   +---+----^-------^---+--------+  |
|        |       |                           |       |               |
|   +----v-------|-------------------------- |-------v------------+  |
|   |            l----------------------------l                   |  |
|   |                    Data Plane Networking                    |  |
|   |                    (Kernel or User space)                   |  |
|   +----^-----------------------------------------^-------------+  |
|        |                                         |               |
|   +----v------+                        +----v------+             |
|   | Phy Port |                         | Phy Port |             |
|   +-----------+                        +-----------+             |
+--------^--------------------------------------------^------------+
         |                                            |
+--------v--------------------------------------------v------------+
|                                                                 |
|                     Traffic Generator                           |
|                                                                 |
+-----------------------------------------------------------------+
```

Figure 6: Single Host Test Scenario - BMP2VMP

o VMP2VMP scenario

```
+------------------------------------------------------------------+
|  HOST                                                            |
|  +-----------------------------+  +-----------------------------+ |
|  |VM +-------------------+     |  |VM +-------------------+    | |
|  |   |     C-VNF         |     |  |   |     C-VNF         |    | |
|  |   | +--------------+  |     |  |   | +--------------+  |    | |
|  |   | | Logical Port |  |     |  |   | | Logical Port |  |    | |
|  |   +-+--^-------^---+--+     |  |   +-+--^-------^---+--+    | |
|  |   +----|-------|---+        |  |   +----|-------|---+      | |
|  |   | Logical Port  |        |  |   | Logical Port  |      | |
|  +---+----^-------^---+-------+  +---+----^-------^---+-------+ |
|          |       |              |       |              |
|  +--------v-------v---------------------|-------v-------------+ |
|  |                l----------------------l                   | |
|  |                 Data Plane Networking                     | |
|  |                 (Kernel or User space)                    | |
|  +----^-------------------------------------------^----------+ |
|       |                                           |            |
|  +----v------+                         +----v------+           |
|  | Phy Port  |                         | Phy Port  |           |
|  +-----------+                         +-----------+           |
+-------^-------------------------------------------^------------+
        |                                           |
+-------v-------------------------------------------v------------+
|                                                               |
|                    Traffic Generator                          |
|                                                               |
+---------------------------------------------------------------+
```
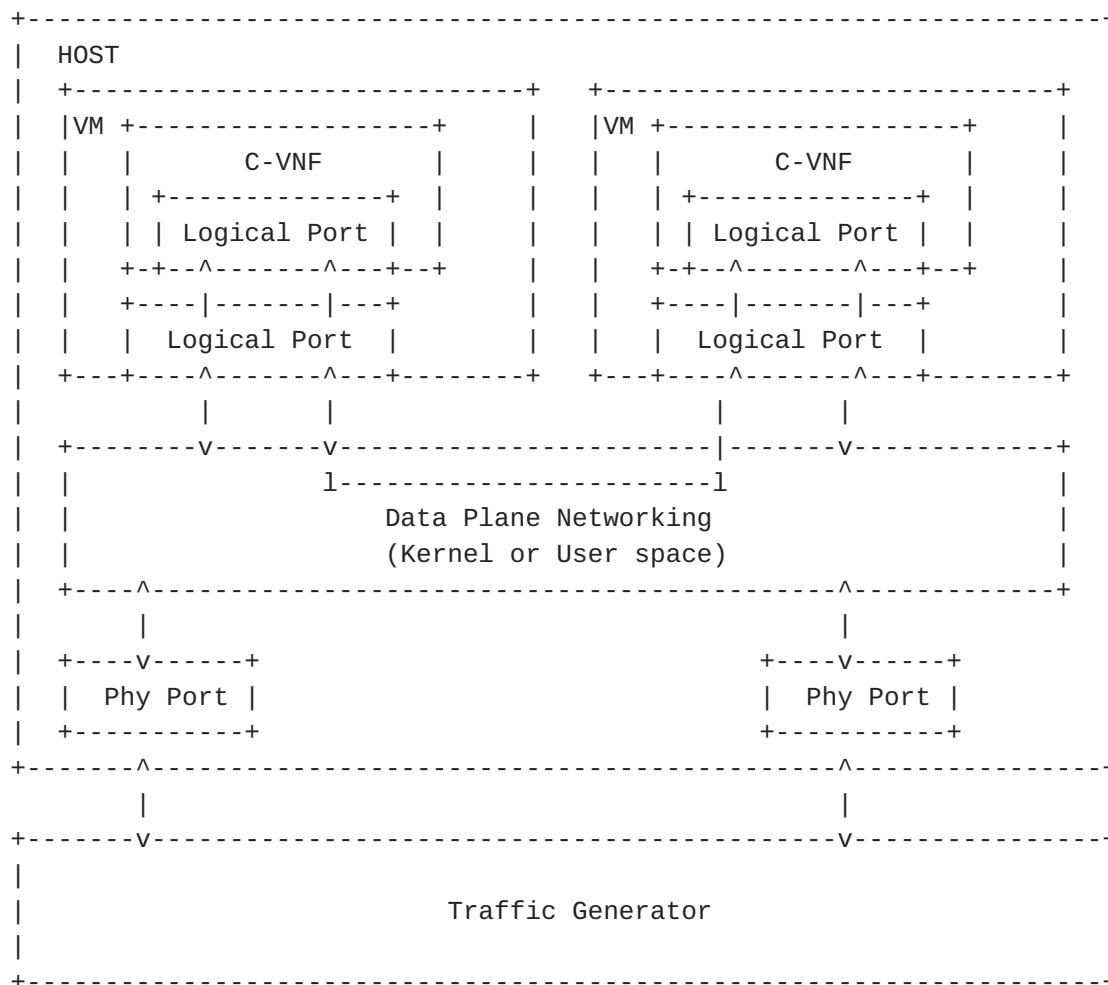
Figure 7: Single Host Test Scenario - VMP2VMP

5. Additional Considerations

When we consider benchmarking for not only containerized but also VM-
based infrastucture and network functions, benchmarking scenarios may
contain various operational use cases. Traditional black-box
benchmarking is focused to measure in-out performance of packet from
physical network ports, since hardware is tightly coupled with its
function and only single function is running on its dedicated
hardware. However, in the NFV environment, the physical network port
commonly will be connected to multiple VNFs(i.e. Multiple PVP test
setup architecture was described in [ETSI-TST-009]) rather than
dedicated to a single VNF. Therefore, benchmarking scenarios should
reflect operational considerations such as number of VNFs or network
services defined by a set of VNFs in a single host.
[service-density], which proposed a way for measuring performance of
multiple NFV service instances at a varied service density on a

single host, is one example of these operational benchmarking
aspects.

6. Security Considerations

TBD

7. Acknkowledgement

We would like to thanks people Al, Maciek and Luis who reviewed and
gave comments of previous draft.

8. Informative References

[Calico]   "Project Calico", July 2019,
           <https://docs.projectcalico.org/>.

[Docker-network]
           "Docker, Libnetwork design", July 2019,
           <https://github.com/docker/libnetwork/>.

[eBPF]   "eBPF, extended Berkeley Packet Filter", July 2019,
           <https://www.iovisor.org/technology/ebpf>.

[ETSI-TST-009]
           "Network Functions Virtualisation (NFV) Release 3;
           Testing; Specification of Networking Benchmarks and
           Measurement Methods for NFVI", October 2018.

[Flannel] "flannel 0.10.0 Documentation", July 2019,
           <https://coreos.com/flannel/>.

[Multique]
           "Multiqueue virtio-net", July 2019,
           <https://www.linux-kvm.org/page/Multiqueue>.

[netmap]   "Netmap: a framework for fast packet I/O", July 2019,
           <https://github.com/luigirizzo/netmap>.

[OVN]   "How to use Open Virtual Networking with Kubernetes", July
           2019, <https://github.com/ovn-org/ovn-kubernetes>.

[OVS]   "Open Virtual Switch", July 2019,
           <https://www.openvswitch.org/>.

[ovs-dpdk]
          "Open vSwitch with DPDK", July 2019,
          <http://docs.openvswitch.org/en/latest/intro/install/
          dpdk/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
          Requirement Levels", RFC 2119, March 1997.

[RFC8172] Morton, A., "Considerations for Benchmarking Virtual
          Network Functions and Their Infrastructure", RFC 8172,
          July 2017.

[RFC8204] Tahhan, M., O'Mahony, B., and A. Morton, "Benchmarking
          Virtual Switches in the Open Platform for NFV (OPNFV)",
          RFC 8204, September 2017.

[service-density]
          Konstantynowicz, M. and P. Mikus, "NFV Service Density
          Benchmarking", March 2019, <https://tools.ietf.org/html/
          draft-mkonstan-nf-service-density-00>.

[SR-IOV]   "SRIOV for Container-networking", July 2019,
          <https://github.com/intel/sriov-cni>.

[vpp]   "VPP with Containers", July 2019, <https://fdio-
          vpp.readthedocs.io/en/latest/usecases/containers.html>.

Authors' Addresses

   Kyoungjae Sun
   School of Electronic Engineering
   Soongsil University
   369, Sangdo-ro, Dongjak-gu
   Seoul, Seoul 06978
   Republic of Korea

   Phone: +82 10 3643 5627
   EMail: gomjae@dcn.ssu.ac.kr

Hyunsik Yang
School of Electronic Engineering
Soongsil University
369, Sangdo-ro, Dongjak-gu
Seoul, Seoul 06978
Republic of Korea

Phone: +82 10 9005 7439
EMail: yangun@dcn.ssu.ac.kr

Youngki Park
School of Electronic Engineering
Soongsil University
369, Sangdo-ro, Dongjak-gu
Seoul, Seoul 06978
Republic of Korea

Phone: +82 10 4281 0720
EMail: ykpark@dcn.ssu.ac.kr

Younghan Kim
School of Electronic Engineering
Soongsil University
369, Sangdo-ro, Dongjak-gu
Seoul, Seoul 06978
Republic of Korea

Phone: +82 10 2691 0904
EMail: younghak@ssu.ac.kr

Wangbong Lee
ETRI
ETRI
161, Gajeong-ro, Yoosung-gu
Dajeon, Dajeon 34129
Republic of Korea

Phone: +82 10 5336 2323
EMail: leewb@etri.re.kr