Benchmarking Working Group
Internet-Draft
Intended status: Informational
Expires: September 12, 2019

M. Konstantynowicz, Ed.
P. Mikus, Ed.
Cisco Systems
March 11, 2019

NFV Service Density Benchmarking
draft-mkonstan-nf-service-density-00

Abstract

Network Function Virtualization (NFV) system designers and operators continuously grapple with the problem of qualifying performance of network services realised with software Network Functions (NF) running on Commercial-Off-The-Shelf (COTS) servers. One of the main challenges is getting repeatable and portable benchmarking results and using them to derive deterministic operating range that is production deployment worthy.

This document specifies benchmarking methodology for NFV services that aims to address this problem space. It defines a way for measuring performance of multiple NFV service instances, each composed of multiple software NFs, and running them at a varied service "packing" density on a single server.

The aim is to discover deterministic usage range of NFV system. In addition specified methodology can be used to compare and contrast different NFV virtualization technologies.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 12, 2019.

Table of Contents

1. Terminology

- NFV - Network Function Virtualization, a general industry term describing network functionality implemented in software.
- NFV service - a software based network service realized by a topology of interconnected constituent software network function applications.
- NFV service instance - a single instantiation of NFV service.
- Data-plane optimized software - any software with dedicated threads handling data-plane packet processing e.g. FD.io VPP (Vector Packet Processor), OVS-DPDK.

2. Motivation

2.1. Problem Description

Network Function Virtualization (NFV) system designers and operators continuously grapple with the problem of qualifying performance of network services realised with software Network Functions (NF) running on Commercial-Off-The-Shelf (COTS) servers. One of the main challenges is getting repeatable and portable benchmarking results and using them to derive deterministic operating range that is production deployment worthy.

Lack of well defined and standardised NFV centric performance methodology and metrics makes it hard to address fundamental questions that underpin NFV production deployments:

1. What NFV service and how many instances can run on a single compute node?
2. How to choose the best compute resource allocation scheme to maximise service yield per node?

3. How do different NF applications compare from the service density perspective?

4. How do the virtualisation technologies compare e.g. Virtual Machines, Containers?

Getting answers to these points should allow designers to make a data based decision about the NFV technology and service design best suited to meet requirements of their use cases. Equally, obtaining the benchmarking data underpinning those answers should make it

easier for operators to work out expected deterministic operating range of chosen design.

2.2. Proposed Solution

The primary goal of the proposed benchmarking methodology is to focus on NFV technologies used to construct NFV services. More specifically to i) measure packet data-plane performance of multiple NFV service instances while running them at varied service "packing" densities on a single server and ii) quantify the impact of using multiple NFs to construct each NFV service instance and introducing multiple packet processing hops and links on each packet path.

The overarching aim is to discover a set of deterministic usage ranges that are of interest to NFV system designers and operators. In addition, specified methodology can be used to compare and contrast different NFV virtualisation technologies.

In order to ensure wide applicability of the benchmarking methodology, the approach is to separate NFV service packet processing from the shared virtualisation infrastructure by decomposing the software technology stack into three building blocks:

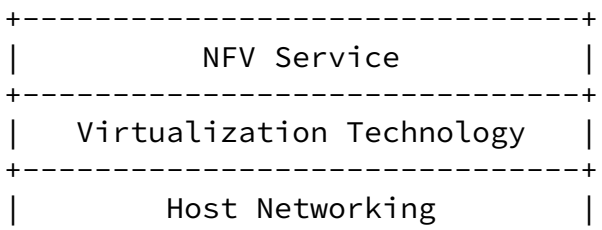


Figure 1. NFV software technology stack.

Proposed methodology is complementary to existing NFV benchmarking industry efforts focusing on vSwitch benchmarking [RFC8204], [TST009] and extends the benchmarking scope to NFV services.

This document does not describe a complete benchmarking methodology, instead it is focusing on system under test configuration part. Each of the compute node configurations identified by (RowIndex, ColumnIndex) is to be evaluated for NFV service data-plane performance using existing and/or emerging network benchmarking standards. This may include methodologies specified in [RFC2544], [TST009], [draft-vpolak-mkonstan-bmwg-mlrsearch] and/or [draft-vpolak-bmwg-plrsearch].

3. NFV Service

It is assumed that each NFV service instance is built of one or more constituent NFs and is described by: topology, configuration and resulting packet path(s).

Each set of NFs forms an independent NFV service instance, with multiple sets present in the host.

3.1. Topology

NFV topology describes the number of network functions per service instance, and their inter-connections over packet interfaces. It includes all point-to-point virtual packet links within the compute node, Layer-2 Ethernet or Layer-3 IP, including the ones to host networking data-plane.

Theoretically, a large set of possible NFV topologies can be realised using software virtualisation topologies, e.g. ring, partial -/full-mesh, star, line, tree, ladder. In practice however, only a few topologies are in the actual use as NFV services mostly perform either bumps-in-a-wire packet operations (e.g. security filtering/

inspection, monitoring/telemetry) and/or inter-site forwarding decisions (e.g. routing, switching).

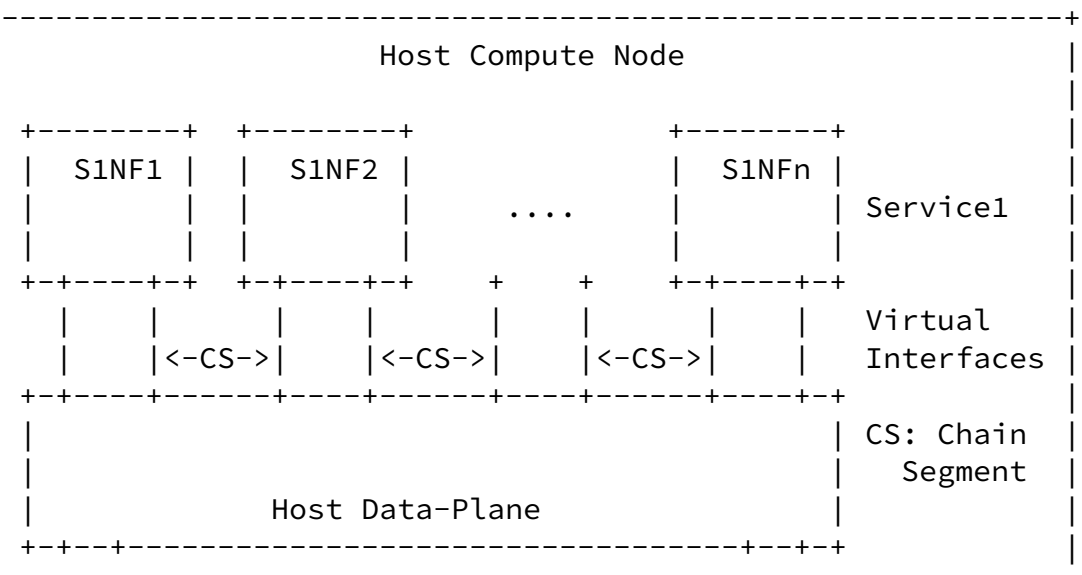
Two main NFV topologies have been identified so far for NFV service density benchmarking:

1. Chain topology: a set of NFs connect to host data-plane with minimum of two virtual interfaces each, enabling host data-plane to facilitate NF to NF service chain forwarding and provide connectivity with external network.

2. Pipeline topology: a set of NFs connect to each other in a line fashion with edge NFs homed to host data-plane. Host data-plane provides connectivity with external network.

Both topologies are shown in figures below.

NF chain topology:



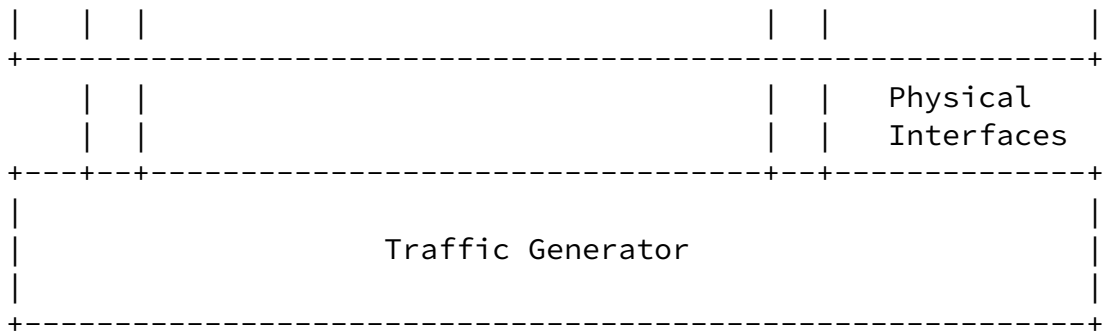
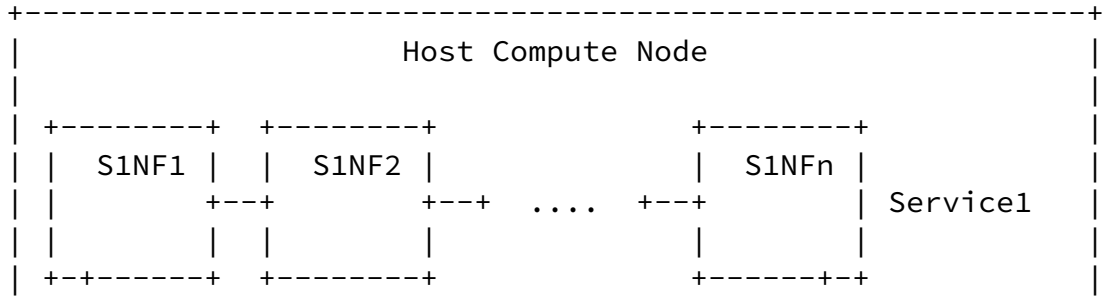


Figure 2. NF chain topology forming a service instance.

NF pipeline topology:



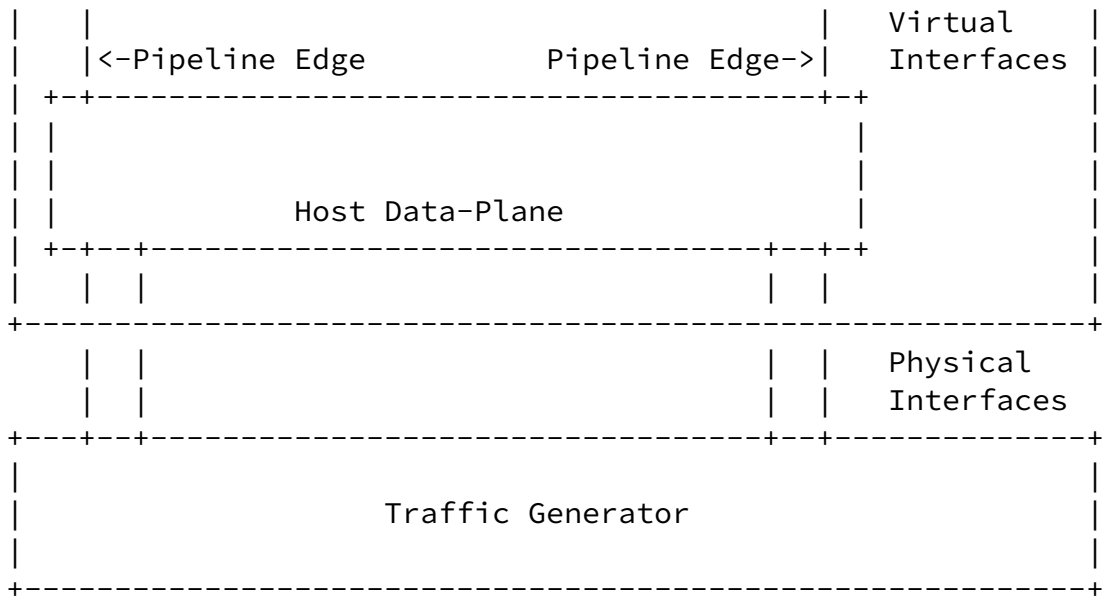


Figure 3. NF pipeline topology forming a service instance.

3.2. Configuration

NFV configuration includes all packet processing functions in NFs including Layer-2, Layer-3 and/or Layer-4-to-7 processing as appropriate to specific NF and NFV service design. L2 sub- interface encapsulations (e.g. 802.1q, 802.1ad) and IP overlay encapsulation (e.g. VXLAN, IPSec, GRE) may be represented here too as appropriate, although in most cases they are used as external encapsulation and handled by host networking data-plane.

NFV configuration determines logical network connectivity that is Layer-2 and/or IPv4/IPv6 switching/routing modes, as well as NFV service specific aspects. In the context of NFV density benchmarking methodology the initial focus is on the former.

Building on the two identified NFV topologies, two common NFV configurations are considered:

1. Chain configuration:

    * Relies on chain topology to form NFV service chains.

    * NF packet forwarding designs:

        + IPv4/IPv6 routing.

    * Requirements for host data-plane:

        + L2 switching with L2 forwarding context per each NF chain segment, or

        + IPv4/IPv6 routing with IP forwarding context per each NF chain segment or per NF chain.

2. Pipeline configuration:

    * Relies on pipeline topology to form NFV service pipelines.

    * Packet forwarding designs:

        + IPv4/IPv6 routing.

    * Requirements for host data-plane:

        + L2 switching with L2 forwarding context per each NF pipeline edge link, or

        + IPv4/IPv6 routing with IP forwarding context per each NF pipeline edge link or per NF pipeline.

3.3. Packet Path(s)

NFV packet path(s) describe the actual packet forwarding path(s) used for benchmarking, resulting from NFV topology and configuration. They are aimed to resemble true packet forwarding actions during the NFV service lifecycle.

Based on the specified NFV topologies and configurations two NFV packet paths are taken for benchmarking:

1. Snake packet path

    * Requires chain topology and configuration.

    * Packets enter the NFV chain through one edge NF and progress to the other edge NF of the chain.

    * Within the chain, packets follow a zigzagging "snake" path entering and leaving host data-plane as they progress through the NF chain.

* Host data-plane is involved in packet forwarding operations between NIC interfaces and edge NFs, as well as between NFs in the chain.

2. Pipeline packet path

* Requires pipeline topology and configuration.

* Packets enter the NFV chain through one edge NF and progress to the other edge NF of the pipeline.

* Within the chain, packets follow a straight path entering and leaving subsequent NFs as they progress through the NF pipeline.

* Host data-plane is involved in packet forwarding operations between NIC interfaces and edge NFs only.

Both packet paths are shown in figures below.

Snake packet path:

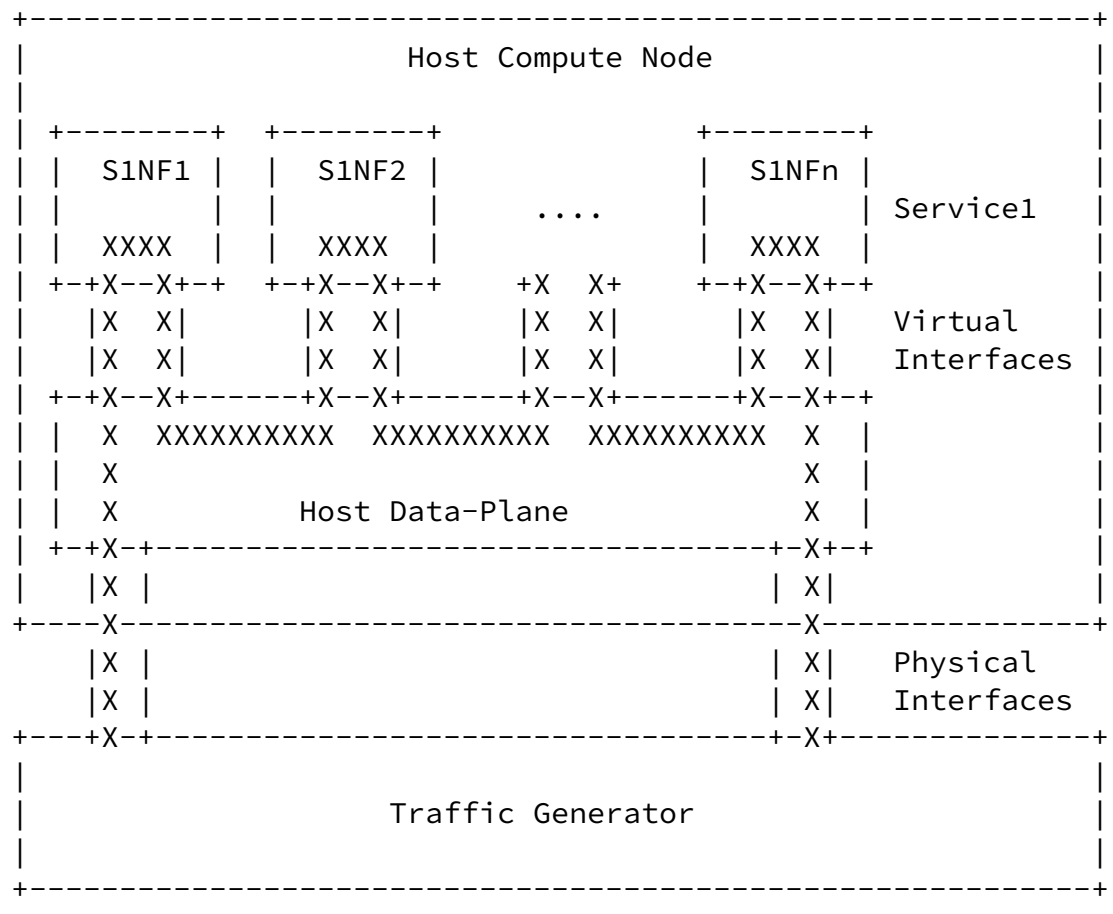


Figure 4. Snake packet path thru NF chain topology.

Pipeline packet path:

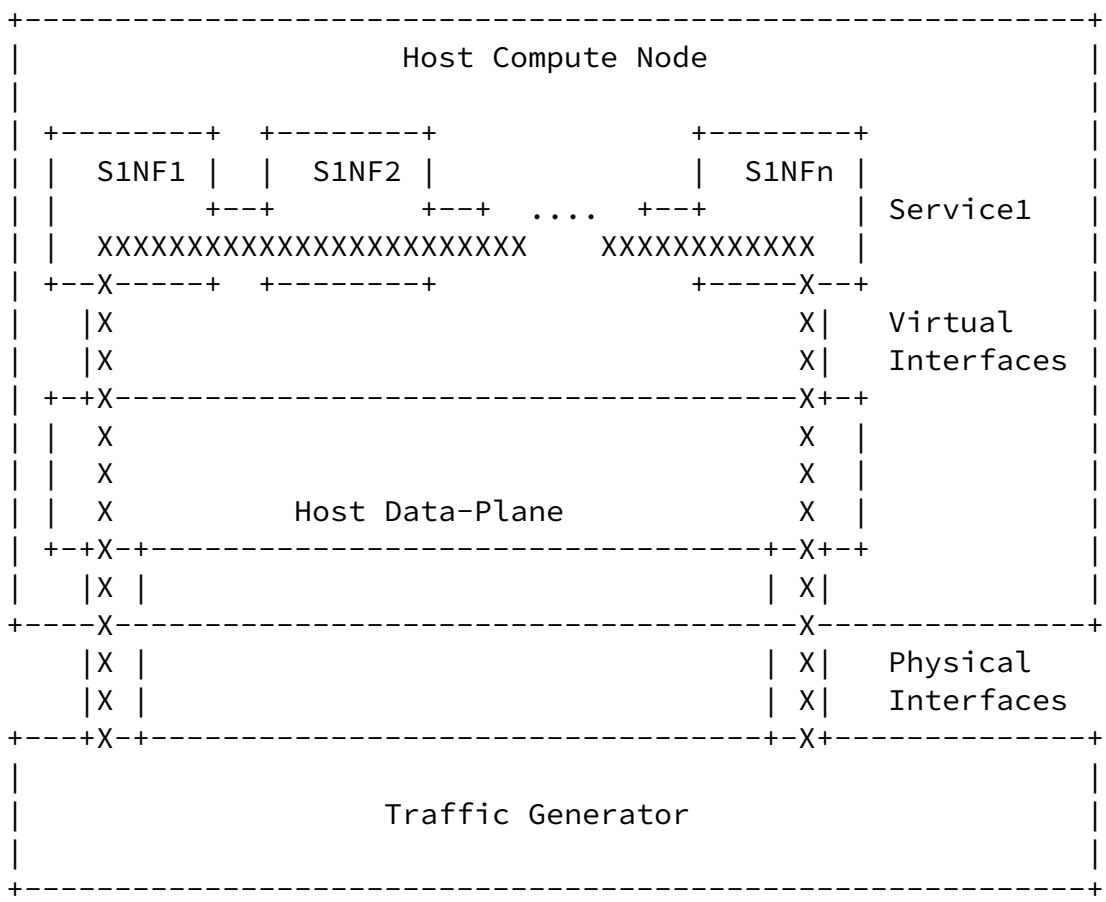


Figure 5. Pipeline packet path thru NF pipeline topology.

In all cases packets enter NFV system via shared physical NIC interfaces controlled by shared host data-plane, are then associated with specific NFV service (based on service discriminator) and subsequently are cross- connected/switched/routed by host data-plane to and through NF topologies per one of above listed schemes.

4. Virtualization Technology

NFV services are built of composite isolated NFs, with virtualisation technology providing the workload isolation. Following

virtualisation technology types are considered for NFV service density benchmarking:

1. Virtual Machines (VMs)

    * Relying on host hypervisor technology e.g. KVM, ESXi, Xen.

    * NFs running in VMs are referred to as VNFs.

2. Containers

    * Relying on Linux container technology e.g. LXC, Docker.

    * NFs running in Containers are referred to as CNFs.

Different virtual interface types are available to VNFs and CNFs:

1. VNF

    * virtio-vhostuser: fully user-mode based virtual interface.

    * virtio-vhostnet: involves kernel-mode based backend.

2. CNF

    * memif: fully user-mode based virtual interface.

    * af_packet: involves kernel-mode based backend.

    * (add more common ones)

5. Host Networking

Host networking data-plane is the central shared resource that underpins creation of NFV services. It handles all of the connectivity to external physical network devices through physical network connections using NICs, through which the benchmarking is done.

Assuming that NIC interface resources are shared, here is the list of widely available host data-plane options for providing packet

connectivity to/from NICs and constructing NFV chain and pipeline topologies and configurations:

- Linux Kernel-Mode Networking.
- Linux User-Mode vSwitch.
- Virtual Machine vSwitch.
- Linux Container vSwitch.
- SRIOV NIC Virtual Function - note: restricted support for chain and pipeline topologies, as it requires hair-pinning through the NIC and oftentimes also through external physical switch.

Analysing properties of each of these options and their Pros/Cons for specified NFV topologies and configurations is outside the scope of this document.

From all listed options, performance optimised Linux user-mode vswitch deserves special attention. Linux user-mode switch decouples NFV service from the underlying NIC hardware, offers rich multi-tenant functionality and most flexibility for supporting NFV services. But in the same time it is consuming compute resources and is harder to benchmark in NFV service density scenarios.

Following sections focus on using Linux user-mode vSwitch, focusing on its performance benchmarking at increasing levels of NFV service density.

6. NFV Service Density Matrix

In order to evaluate performance of multiple NFV services running on a compute node, NFV service instances are benchmarked at increasing density, allowing to construct an NFV Service Density Matrix. Table below shows an example of such a matrix, capturing number of NFV service instances (row indices), number of NFs per service instance (column indices) and resulting total number of NFs (values).

NFV Service Density - NF Count View

SVC	001	002	004	006	008	00N
001	1	2	4	6	8	1*N
002	2	4	8	12	16	2*N
004	4	8	16	24	32	4*N
006	6	12	24	36	48	6*N
008	8	16	32	48	64	8*N
00M	M*1	M*2	M*4	M*6	M*8	M*N

RowIndex: Number of NFV Service Instances, 1..M.
ColumnIndex: Number of NFs per NFV Service Instance, 1..N.
Value: Total number of NFs running in the system.

In order to deliver good and repeatable network data-plane performance, NFs and host data-plane software require direct access to critical compute resources. Due to a shared nature of all resources on a compute node, a clearly defined resource allocation scheme is defined in the next section to address this.

In each tested configuration host data-plane is a gateway between the external network and the internal NFV network topologies. Offered packet load is generated and received by an external traffic generator per usual benchmarking practice.

It is proposed that initial benchmarks are done with the offered packet load distributed equally across all configured NFV service instances. This could be followed by various per NFV service instance load ratios mimicking expected production deployment scenario(s).

Following sections specify compute resource allocation, followed by examples of applying NFV service density methodology to VNF and CNF benchmarking use cases.

7. Compute Resource Allocation

Performance optimized NF and host data-plane software threads require timely execution of packet processing instructions and are very sensitive to any interruptions (or stalls) to this execution e.g. cpu core context switching, or cpu jitter. To that end, NFV service density methodology treats controlled mapping ratios of data plane software threads to physical processor cores with directly allocated

cache hierarchies as the first order requirement.

Other compute resources including memory bandwidth and PCIe bandwidth have lesser impact and as such are subject for further study. For more detail and deep-dive analysis of software data plane performance and impact on different shared compute resources is available in [BSDP].

It is assumed that NFs as well as host data-plane (e.g. vswitch) are performance optimized, with their tasks executed in two types of software threads:

- data-plane - handling data-plane packet processing and forwarding, time critical, requires dedicated cores. To scale data-plane performance, most NF apps use multiple data-plane threads and rely on NIC RSS (Receive Side Scaling), virtual interface multi-queue and/or integrated software hashing to distribute packets across the data threads.

- main-control - handling application management, statistics and control-planes, less time critical, allows for core sharing. For most NF apps this is a single main thread, but often statistics (counters) and various control protocol software are run in separate threads.

Core mapping scheme described below allocates cores for all threads of specified type belonging to each NF app instance, and separately lists number of threads to a number of logical/physical core mappings for processor configurations with enabled/disabled Symmetric Multi-Threading (SMT) (e.g. AMD SMT, Intel Hyper-Threading).

If NFV service density benchmarking is run on server nodes with Symmetric Multi-Threading (SMT) (e.g. AMD SMT, Intel Hyper-Threading) for higher performance and efficiency, logical cores allocated to data- plane threads should be allocated as pairs of sibling logical cores corresponding to the hyper-threads running on the same physical core.

Separate core ratios are defined for mapping threads of vSwitch and NFs. In order to get consistent benchmarking results, the mapping ratios are enforced using Linux core pinning.

application	thread type	app:core ratio	threads/pcores (SMT disabled)	threads/lcores map (SMT enabled)

vSwitch-1c	data	1:1	1DT/1PC	2DT/2LC
	main	1:S2	1MT/S2PC	1MT/1LC
vSwitch-2c	data	1:2	2DT/2PC	4DT/4LC
	main	1:S2	1MT/S2PC	1MT/1LC
vSwitch-4c	data	1:4	4DT/4PC	8DT/8LC
	main	1:S2	1MT/S2PC	1MT/1LC
NF-0.5c	data	1:S2	1DT/S2PC	1DT/1LC
	main	1:S2	1MT/S2PC	1MT/1LC
NF-1c	data	1:1	1DT/1PC	2DT/2LC
	main	1:S2	1MT/S2PC	1MT/1LC
NF-2c	data	1:2	2DT/2PC	4DT/4LC
	main	1:S2	1MT/S2PC	1MT/1LC

- Legend to table
  - Header row
    - application - network application with optimized data-plane, a vSwitch or Network Function (NF) application.

  + thread type - either "data", short for data-plane; or "main", short for all main-control threads.

  + app:core ratio - ratio of per application instance threads of specific thread type to physical cores.

  + threads/pcores (SMT disabled) - number of threads of specific type (DT for data-plane thread, MT for main thread) running on a number of physical cores, with SMT disabled.

  + threads/lcores map (SMT enabled) - number of threads of specific type (DT, MT) running on a number of logical cores, with SMT enabled. Two logical cores per one physical core.

* Content rows

  + vSwitch-(1c|2c|4c) - vSwitch with 1 physical core (or 2, or 4) allocated to its data-plane software worker threads.

  + NF-(0.5c|1c|2c) - NF application with half of a physical core (or 1, or 2) allocated to its data-plane software worker threads.

  + Sn - shared core, sharing ratio of (n).

  + DT - data-plane thread.

  + MT - main-control thread.

  + PC - physical core, with SMT/HT enabled has many (mostly 2 today) logical cores associated with it.

  + LC - logical core, if more than one lc get allocated in sets of two sibling logical cores running on the same physical core.

  + SnPC - shared physical core, sharing ratio of (n).

  + SnLC - shared logical core, sharing ratio of (n).

Maximum benchmarked NFV service densities are limited by a number of physical cores on a compute node.

A sample physical core usage view is shown in the matrix below.

NFV Service Density - Core Usage View
vSwitch-1c, NF-1c

SVC	001	002	004	006	008	010
001	2	3	6	9	12	15
002	3	6	12	18	24	30
004	6	12	24	36	48	60
006	9	18	36	54	72	90
008	12	24	48	72	96	120
010	15	30	60	90	120	150

RowIndex: Number of NFV Service Instances, 1..10.
ColumnIndex: Number of NFs per NFV Service Instance, 1..10.
Value: Total number of physical processor cores used for NFs.

8. NFV Service Density Benchmarks

To illustrate defined NFV service density applicability, following sections describe three sets of NFV service topologies and configurations that have been benchmarked in open-source: i) in [LFN-FDio-CSIT], a continuous testing and data-plane benchmarking project, and ii) as part of CNCF CNF Testbed initiative [CNCF-CNF-Testbed].

In both cases each NFV service instance definition is based on the same set of NF applications, and varies only by network addressing configuration to emulate multi-tenant operating environment.

8.1. Test Methodology - MRR Throughput

Initial NFV density throughput benchmarks have been performed using Maximum Receive Rate (MRR) test methodology defined and used in FD.io CSIT.

MRR tests measure the packet forwarding rate under the maximum load offered by traffic generator over a set trial duration, regardless of packet loss. Maximum load for specified Ethernet frame size is set to the bi-directional link rate (2x 10GbE in referred results).

Tests were conducted with two traffic profiles: i) continuous stream

of 64B frames, ii) continuous stream of IMIX sequence of (7x 64B, 4x 570B, 1x 1518B), all sizes are L2 untagged Ethernet.

NFV service topologies tested include: VNF service chains, CNF service chains and CNF service pipelines.

8.2. VNF Service Chain

VNF Service Chain (VSC) topology is tested with KVM hypervisor (Ubuntu 18.04-LTS), with NFV service instances consisting of NFs running in VMs (VNFs). Host data-plane is provided by FD.io VPP vswitch. Virtual interfaces are virtio-vhostuser. Snake forwarding packet path is tested using [TRex] traffic generator, see figure.

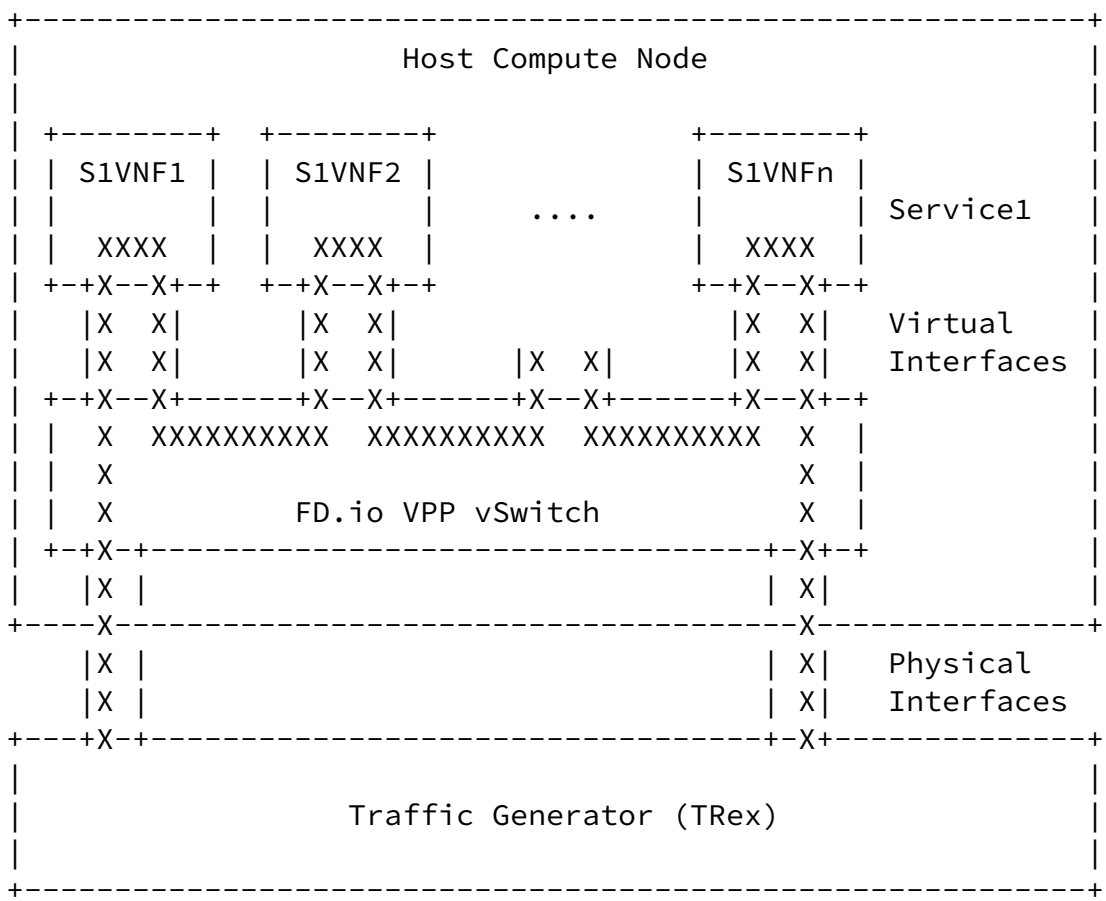


Figure 6. VNF service chain test setup.

8.3. CNF Service Chain

CNF Service Chain (CSC) topology is tested with Docker containers (Ubuntu 18.04-LTS), with NFV service instances consisting of NFs running in Containers (CNFs). Host data-plane is provided by FD.io VPP vswitch. Virtual interfaces are memif. Snake forwarding packet path is tested using [TRex] traffic generator, see figure.

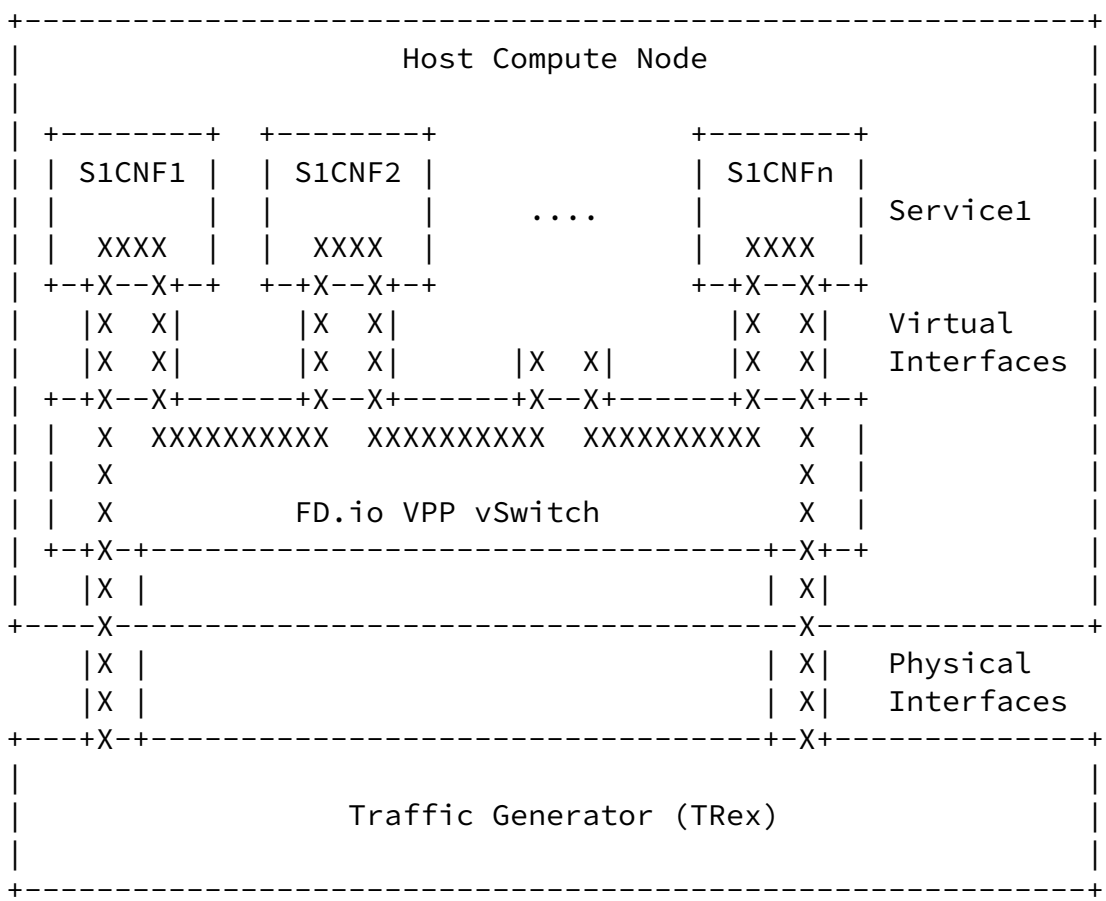
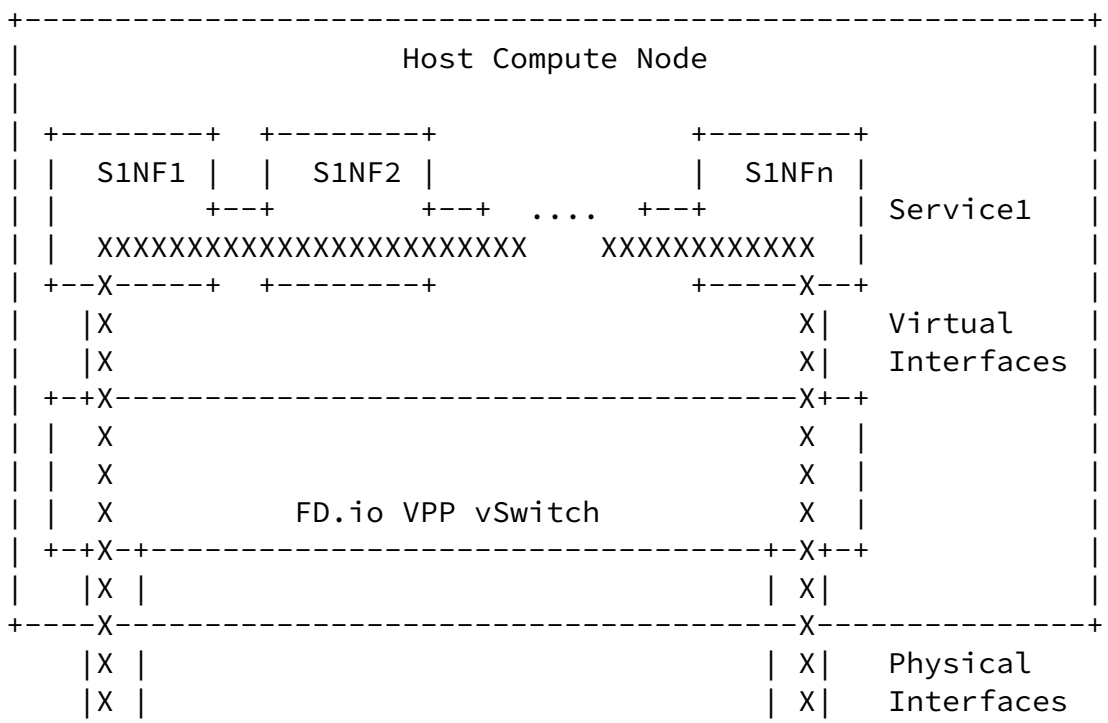


Figure 7. CNF service chain test setup.

8.4. CNF Service Pipeline

CNF Service Pipeline (CSP) topology is tested with Docker containers (Ubuntu 18.04-LTS), with NFV service instances consisting of NFs running in Containers (CNFs). Host data-plane is provided by FD.io VPP vswitch. Virtual interfaces are memif. Pipeline forwarding packet path is tested using [TRex] traffic generator, see figure.



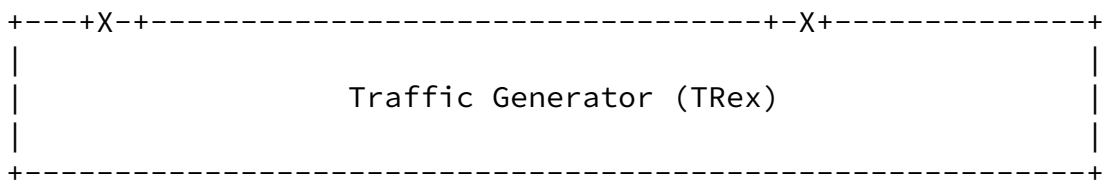


Figure 8. CNF service chain test setup.

8.5. Sample Results: FD.io CSIT

FD.io CSIT project introduced NFV density benchmarking in release CSIT-1901 and published results for the following NFV service topologies and configurations:

1. VNF Service Chains
    * VNF: DPDK-L3FWD v18.10
        + IPv4 forwarding
        + NF-1c
    * vSwitch: VPP v19.01-release
        + L2 MAC switching
        + vSwitch-1c, vSwitch-2c
    * frame sizes: 64B, IMIX

2. CNF Service Chains
    * CNF: VPP v19.01-release
        + IPv4 routing
        + NF-1c
    * vSwitch: VPP v19.01-release
        + L2 MAC switching

        + vSwitch-1c, vSwitch-2c

    * frame sizes: 64B, IMIX

3. CNF Service Pipelines

    * CNF: VPP v19.01-release

        + IPv4 routing

        + NF-1c

    * vSwitch: VPP v19.01-release

        + L2 MAC switching

        + vSwitch-1c, vSwitch-2c

    * frame sizes: 64B, IMIX

More information is available in FD.io CSIT-1901 report, with specific references listed below:

- Testbed: [CSIT-1901-testbed-2n-skx]
- Test environment: [CSIT-1901-test-enviroment]
- Methodology: [CSIT-1901-nfv-density-methodology]
- Results: [CSIT-1901-nfv-density-results]

8.6. Sample Results: CNCF/CNFs

CNCF CI team introduced a CNF testbed initiative focusing on benchmaring NFV density with open-source network applications running

as VNFs and CNFs. Following NFV service topologies and configurations have been tested to date:

1. VNF Service Chains

    * VNF: VPP v18.10-release
        + IPv4 routing
        + NF-1c
    * vSwitch: VPP v18.10-release
        + L2 MAC switching
        + vSwitch-1c, vSwitch-2c
    * frame sizes: 64B, IMIX

2. CNF Service Chains
    * CNF: VPP v18.10-release
        + IPv4 routing
        + NF-1c
    * vSwitch: VPP v18.10-release
        + L2 MAC switching
        + vSwitch-1c, vSwitch-2c
    * frame sizes: 64B, IMIX

3. CNF Service Pipelines
    * CNF: VPP v18.10-release
        + IPv4 routing
        + NF-1c
    * vSwitch: VPP v18.10-release
        + L2 MAC switching
        + vSwitch-1c, vSwitch-2c

    * frame sizes: 64B, IMIX

More information is available in CNCF CNF Testbed github, with summary test results presented in summary markdown file, references listed below:

o Results: [CNCF-CNF-Testbed-Results]

9. IANA Considerations

No requests of IANA

10. Security Considerations

..

11. Acknowledgements

Thanks to Vratko Polak of FD.io CSIT project and Michael Pedersen of the CNCF Testbed initiative for their contributions and useful suggestions.

12. References

12.1. Normative References

[RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <https://www.rfc-editor.org/info/rfc2544>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

12.2. Informative References

[BSDP] "Benchmarking Software Data Planes Intel(R) Xeon(R) Skylake vs. Broadwell", March 2019, <https://fd.io/wp-content/uploads/sites/34/2019/03/benchmarking_sw_data_planes_skx_bdx_mar07_2019.pdf>.

[CNCF-CNF-Testbed] "Cloud native Network Function (CNF) Testbed", March 2019, <https://github.com/cncf/cnf-testbed/>.

[CNCF-CNF-Testbed-Results]
"CNCF CNF Testbed: NFV Service Density Benchmarking", December 2018, <https://github.com/cncf/cnf-testbed/blob/master/comparison/doc/cncf-cnfs-results-summary.md>.

[CSIT-1901-nfv-density-methodology]
"FD.io CSIT Test Methodology: NFV Service Density", March 2019, <https://docs.fd.io/csit/rls1901/report/introduction/methodology_nfv_service_density.html>.

[CSIT-1901-nfv-density-results]
"FD.io CSIT Test Results: NFV Service Density", March 2019, <https://docs.fd.io/csit/rls1901/report/vpp_performance_tests/nf_service_density/index.html>.

[CSIT-1901-test-enviroment]
"FD.io CSIT Test Environment", March 2019, <https://docs.fd.io/csit/rls1901/report/vpp_performance_tests/test_environment.html>.

[CSIT-1901-testbed-2n-skx]
"FD.io CSIT Test Bed", March 2019, <https://docs.fd.io/csit/rls1901/report/introduction/physical_testbeds.html#node-xeon-skylake-2n-skx>.

[draft-vpolak-bmwg-plrsearch]
"Probabilistic Loss Ratio Search for Packet Throughput (PLRsearch)", November 2018, <https://tools.ietf.org/html/draft-vpolak-bmwg-plrsearch-00>.

[draft-vpolak-mkonstan-bmwg-mlrsearch]
"Multiple Loss Ratio Search for Packet Throughput (MLRsearch)", November 2018, <https://tools.ietf.org/html/draft-vpolak-mkonstan-bmwg-mlrsearch-00>.

[LFN-FDio-CSIT]
"Fast Data io, Continuous System Integration and Testing Project", March 2019, <https://wiki.fd.io/view/CSIT>.

[RFC8204] Tahhan, M., O'Mahony, B., and A. Morton, "Benchmarking

Virtual Switches in the Open Platform for NFV (OPNFV)", RFC 8204, DOI 10.17487/RFC8204, September 2017, <https://www.rfc-editor.org/info/rfc8204>.

[TRex] "TRex Low-Cost, High-Speed Stateful Traffic Generator", March 2019, <https://github.com/cisco-system-traffic-generator/trex-core>.

[TST009] "ETSI GS NFV-TST 009 V3.1.1 (2018-10), Network Functions Virtualisation (NFV) Release 3; Testing; Specification of Networking Benchmarks and Measurement Methods for NFVI", October 2018, <https://www.etsi.org/deliver/etsi_gs/NFV-TST/001_099/009/03.01.01_60/gs_NFV-TST009v030101p.pdf>.

Authors' Addresses

Maciek Konstantynowicz (editor)
Cisco Systems

Email: mkonstan@cisco.com

Peter Mikus (editor)
Cisco Systems

Email: pmikus@cisco.com