

Benchmarking Methodology Working Group
Internet-Draft
Intended status: Experimental
Expires: January 9, 2020

LM. Contreras
J. Rodriguez
L. Luque
Telefonica
July 8, 2019

5G transport network benchmarking
draft-contreras-bmwg-5g-00

Abstract

New 5G services are starting to be deployed in operational networks, leveraging in a number of novel technologies and architectural concepts. The purpose of this document is to overview the implications of 5G services in transport networks and to provide guidance on benchmarking of the infrastructures supporting those services.

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 January 9, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Conventions used in this document	2
3. 5G services	2
4. Benchmarking aspects of transport networks in 5G	3
5. Guidance on 5G transport benchmarking	4
6. Security Considerations	4
7. IANA Considerations	4
8. Acknowledgements	4
9. References	4
9.1. Normative References	4
9.2. Informative References	4
Authors' Addresses	5

1. Introduction

5G services are starting to be introduced in real operational networks. The challenges of 5G are multiple, impacting in different technological areas such as radio access, mobile core and transport network. From all those technological areas, the transport network is the focus of this document.

It is important for operators to have a good basis of benchmarking solutions, technologies and architectures before moving them into production. With such aim, this document intends to overview available guidelines to assist on the benchmarking of 5G transport networks, identifying gaps that could require further work and details.

As result, it is expected to provide guidance on benchmarking of 5G transport network infrastructures ready for experimentation in lab environments or real deployment in operational networks.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [RFC2119].

3. 5G services

5G transport networks will need to accommodate different kind of services with very distinct needs and requirements leveraging on the same infrastructure. 5G services can be grouped in three main

categories, namely enhanced Mobile Broadband (eMBB), ultra-Reliable and Low Latency Communications (URLLC), and massive Machine Type Communications (mMTC). Each of them presents different inherent characteristics spanning from ultra-low latency to high bandwidth and high reliability. For instance, eMBB services are expected to provide peak bit rates of up to 1 Gbps, uRRLC services will require latencies as low as below microsecond delays, and mMTC will demand to support up to 100 times the number of current sessions. All these features impose great constraints to the networks deployed today in backhaul and aggregation, in terms of not only network capacity but also in terms of data processing, especially for guaranteeing very low latencies.

The impact in the transport network of those challenges is increased by some other additional challenges introduced by the emergence of two new technological paradigms: the network virtualization and the network programmability.

In one hand, virtualization will introduce uncertainty on the traffic patterns due to the flexibility and scalability in the deployment traffic sources in the transport network. On the other hand, programmability will potentially enable automated reconfiguration of the transport network which requires coordination mechanisms to avoid misconfigurations.

A final consideration is the introduction of the network slicing concept in 5G networks. According to that, the objective is to provide customized and tailored logical networks to different customers, allocating resources for the specific customer service request.

4. Benchmarking aspects of transport networks in 5G

The benchmarking aspects of 5G transport networks can be then structured in the following manner:

Data plane benchmarking: aspects to consider in data plane benchmarking refer to both hardware capabilities as well as to transport encapsulations. Examples of hardware capabilities are recent developments such as IEEE TSN, and example of encapsulation is SRv6 [I-D.ietf-spring-srv6-network-programming].

Control plane benchmarking: aspects to consider for control plane relates to transport infrastructure programmability. In this case some previous works exists such as RFC8456 [RFC8456].

Management plane benchmarking: one specific aspect of management benchmarking in 5G refers to the capability of managing the transport network slice lifecycle.

Architecture benchmarking: new architectural frameworks are being conceived to support advanced services like 5G. An example of these architectures is [I-D.ietf-detnet-architecture].

5. Guidance on 5G transport benchmarking

To be completed.

6. Security Considerations

This draft does not include any security considerations.

7. IANA Considerations

This draft does not include any IANA considerations

8. Acknowledgements

This work has been partly funded by the European Commission through the H2020 project 5G-EVE (Grant Agreement no. 815074).

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

9.2. Informative References

[I-D.ietf-detnet-architecture]
Finn, N., Thubert, P., Varga, B., and J. Farkas,
"Deterministic Networking Architecture", draft-ietf-detnet-architecture-13 (work in progress), May 2019.

[I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J.,
daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-01 (work in progress), July 2019.

[RFC8456] Bhuvaneshwaran, V., Basil, A., Tassinari, M., Manral, V., and S. Banks, "Benchmarking Methodology for Software-Defined Networking (SDN) Controller Performance", RFC 8456, DOI 10.17487/RFC8456, October 2018, <<https://www.rfc-editor.org/info/rfc8456>>.

Authors' Addresses

Luis M. Contreras
Telefonica
Ronda de la Comunicacion, s/n
Sur-3 building, 3rd floor
Madrid 28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com
URI: <http://lmcontreras.com/>

Juan Rodriguez
Telefonica
Zurbaran, 12
Madrid 28010
Spain

Email: juan.rodriquezmartinez@telefonica.com

Lourdes Luque
Telefonica
Zurbaran, 12
Madrid 28010
Spain

Email: lourdes.luquecanto@telefonica.com

Benchmarking Methodology Working Group
Internet-Draft
Intended status: Informational
Expires: May 7, 2020

K. Sun
H. Yang
Y. Park
Y. Kim
Soongsil University
W. Lee
ETRI
November 04, 2019

Considerations for Benchmarking Network Performance in Containerized
Infrastructures
draft-dcn-bmwg-containerized-infra-03

Abstract

This draft describes considerations for benchmarking network performance in containerized infrastructures. 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 specified 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 for benchmarking network performance in containerized infrastructures.

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 May 7, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Benchmarking Considerations	3
3.1. Comparison with the VM-based Infrastructure	3
3.2. Container Networking Classification	5
3.3. Resource Considerations	8
4. Benchmarking Scenarios for the Containerized Infrastructure .	10
5. Additional Considerations	13
6. Security Considerations	14
7. Acknowledgement	14
8. Informative References	14
Authors' Addresses	15

1. Introduction

The Benchmarking Methodology Working Group (BMWG) has recently expanded its benchmarking scope from Physical Network Function (PNF) running on a 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.

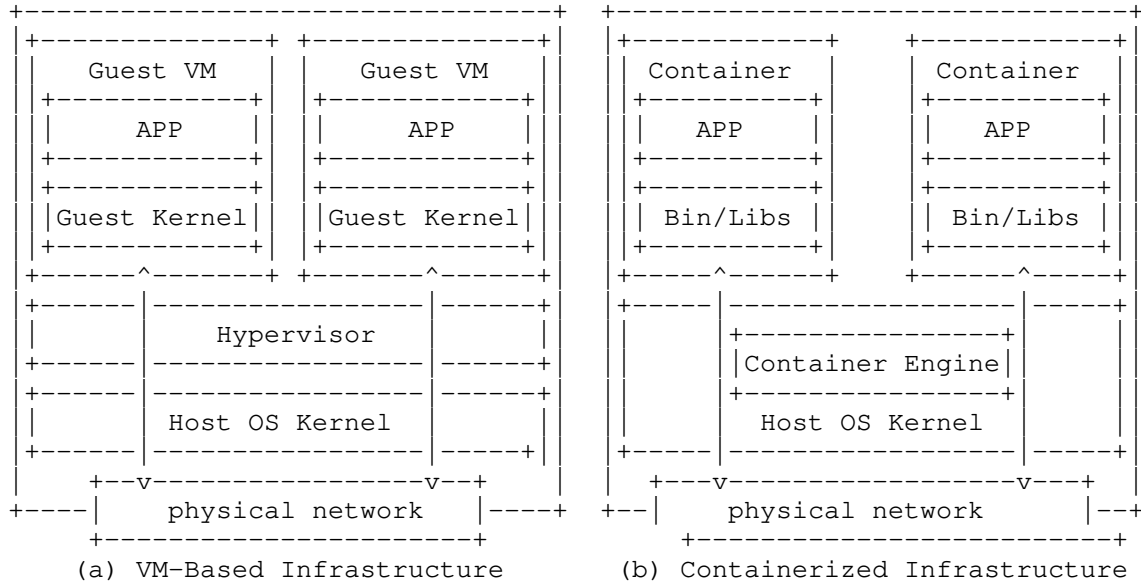


Figure 1: Comparison of NFV Infrastructures

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 the 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 the capabilities of servers and storage should meet the minimum requirements for testing, it is possible to deploy a test environment with fewer capabilities than in the VM-based infrastructure.
- o About configuration parameters, the containerized infrastructure needs a 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 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 a physical address and the 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 an isolation environment from container runtime through the host namespace, creating a virtual interface, allocating interface and IP address to C-VNF. Since the containerized infrastructure has different 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 below;

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

- o CNI(Container 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, the container network model can be classified into the kernel-space network model and user-space network model according to the location of network service creation. In the case of the 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]

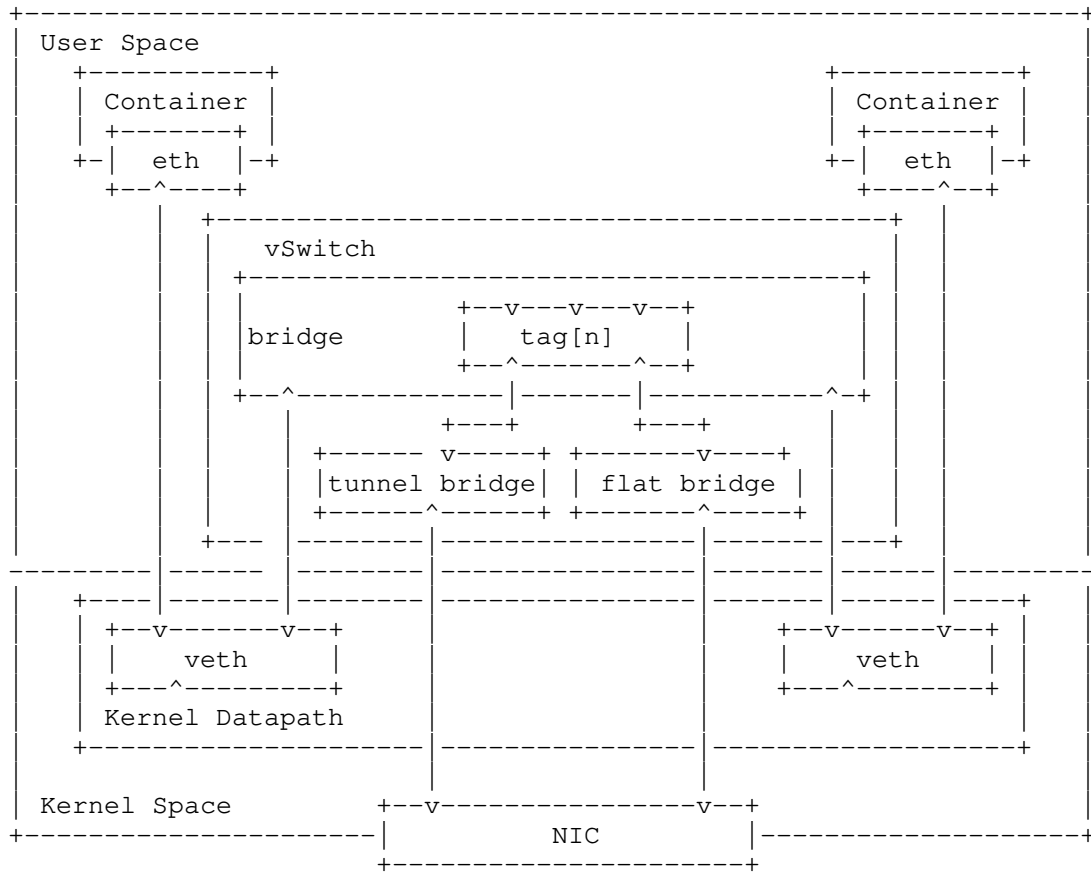


Figure 2: Examples of Kernel Space Network Model

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

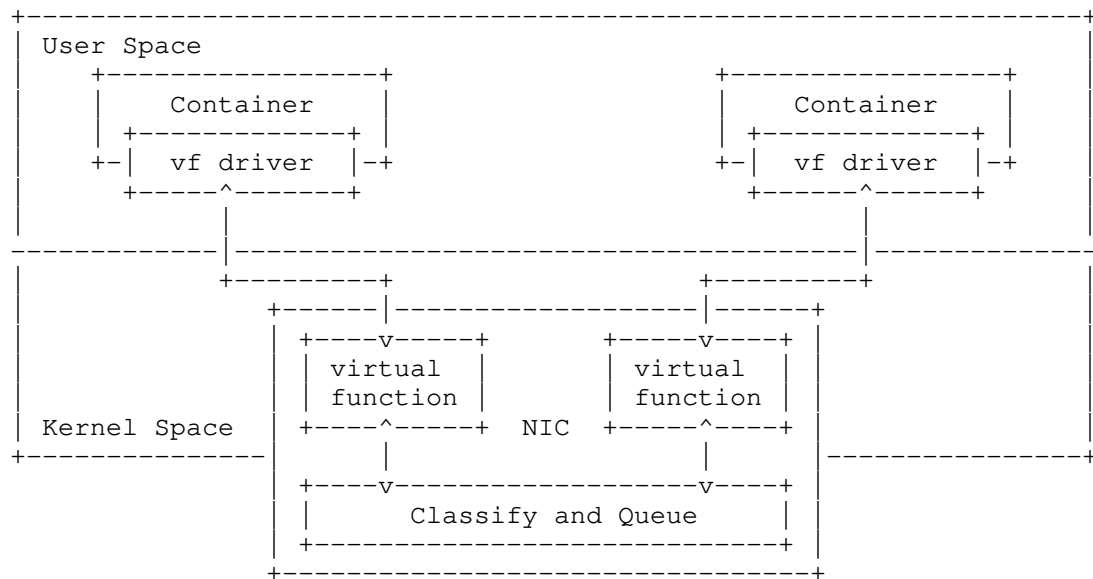


Figure 3: Examples of User Space Network Model - Device Pass-through

o User space network model / vSwitch model: ovs-dpdk[ovs-dpdk], vpp[vpp], netmap[netmap]

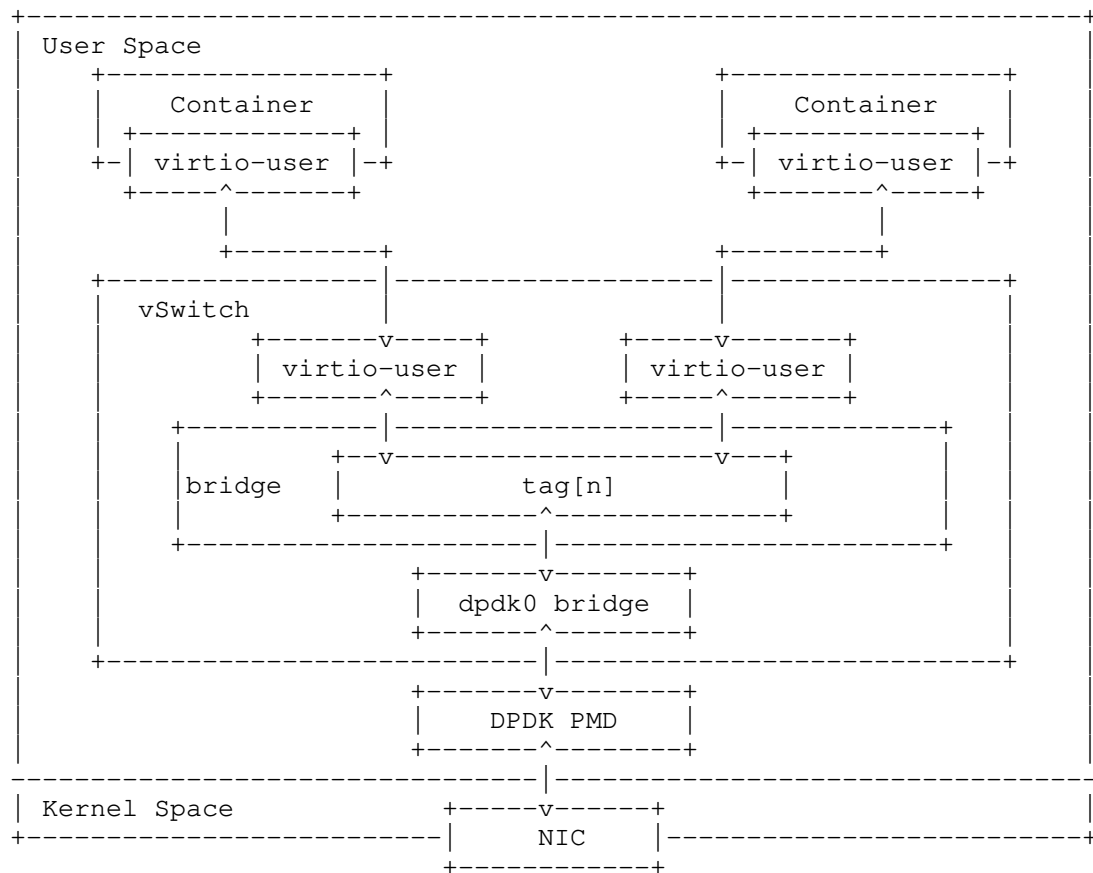


Figure 4: Examples of User Space Network Model - vSwitch Model using DPDK

3.3. Resource Considerations

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

o Hugepage

The huge page is that configuring a large page size of memory to reduce Translation Lookaside Buffer (TLB) miss rate and increase the application performance. This increases the performance of logical/virtual to physical address lookups performed by a CPU's memory management unit, and generally overall system performance. When using Cent OS or RedHat OS in the VM-based infrastructure, the huge

page should be set to at least 1G byte. In the VM-based infrastructure, the host OS and the hypervisor can configure a huge page depending on the guest OS. For example, guest VMs with the Linux OS requires to set huge pages at least 1G bytes. Even though it is a huge size, since this memory page is for not only its running application but also guest OS operation processes, actual memory pages for application is smaller.

In the containerized infrastructure, the container is isolated in the application level and administrators can set huge pages more granular level (e.g. Kubernetes allows to use of 512M bytes huge pages for the container as default values). Moreover, this page is dedicated to the application but another process so application use page more efficient way. Therefore, even if the page size is smaller than the VM, the effect of the huge page is large, which leads to the utilization of physical memory and the increasing number of functions in the host.

- o NUMA

NUMA technology can be used both in the VM-based and containerized infrastructure. Using NUMA, performance will be increasing not CPU and memory but also network since that network interface connected PCIe slot of specific NUMA node have locality. Using NUMA, it requires a strong understanding of VNF's memory requirements. If VNF uses more memory than a single NUMA node contains, the overhead will be occurred due to being spilled to another NUMA node.

In the VM-based infrastructure, the hypervisor can perform extracting NUMA topology and schedules VM workloads. In containerized infrastructure, however, it is more difficult to expose the NUMA topology to the container and currently, it is hard to guarantee the locality of memory when the container is deployed to host that has multiple NUMA nodes. For that reason, the 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 infrastructure, 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 the 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 bare metal, C-VNFs can be deployed as a cluster called POD by Kubernetes. Otherwise each C-VNF can be deployed separately using Docker. In the 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 bare metal and VMP which is Pod on VM.

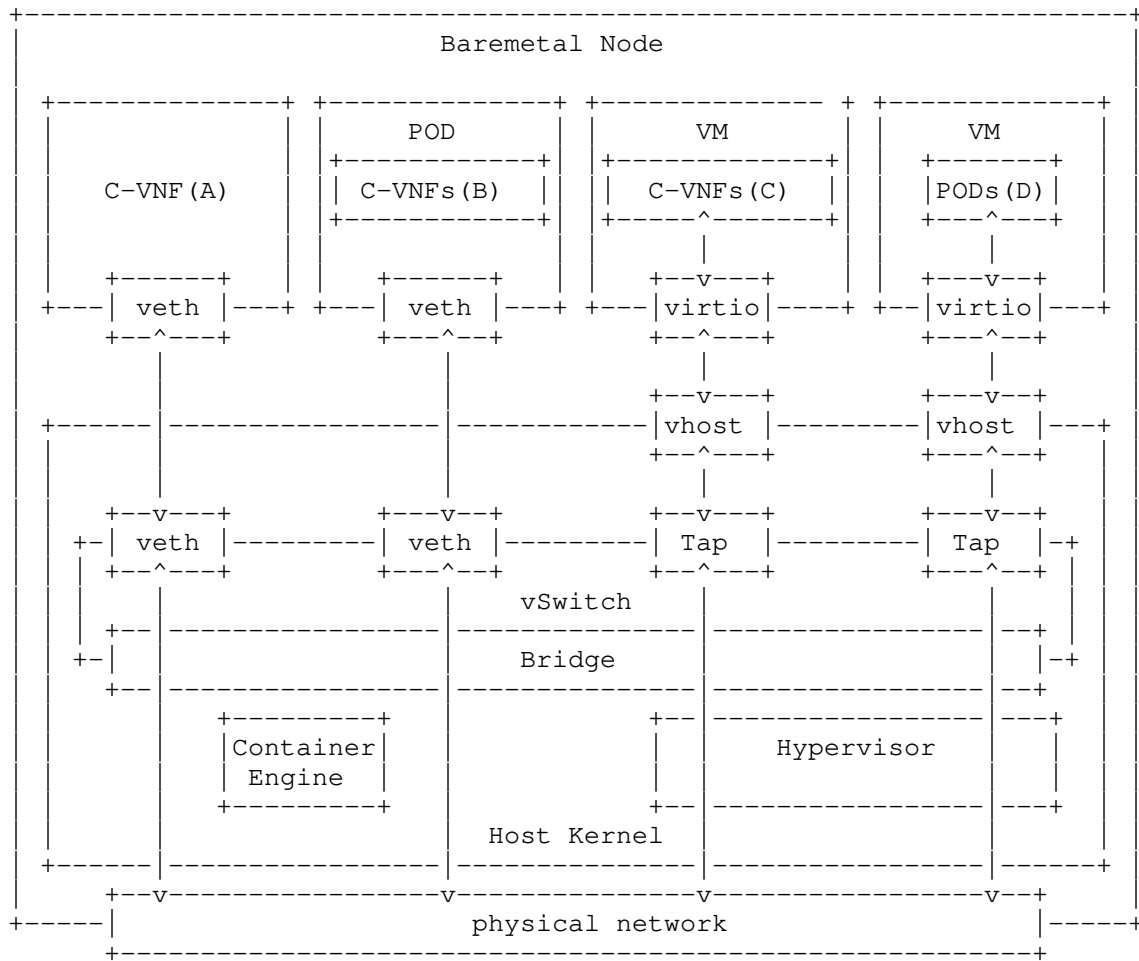


Figure 5: Examples of Networking Architecture based on Deployment Models - (A) C-VNF on Baremetal (B) Pod on Baremetal (BMP) (C) C-VNF on VM (D) Pod on VM (VMP)

In [ETSI-TST-009], they described data plane test scenarios in a single host. In that document, there are two scenarios for containerized infrastructure; Container2Container which is internal communication between two containers in the same Pod, and the Pod2Pod model which is communication between two containers running in different Pods. According to our new terminologies, we can call the Pod2Pod model as the 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

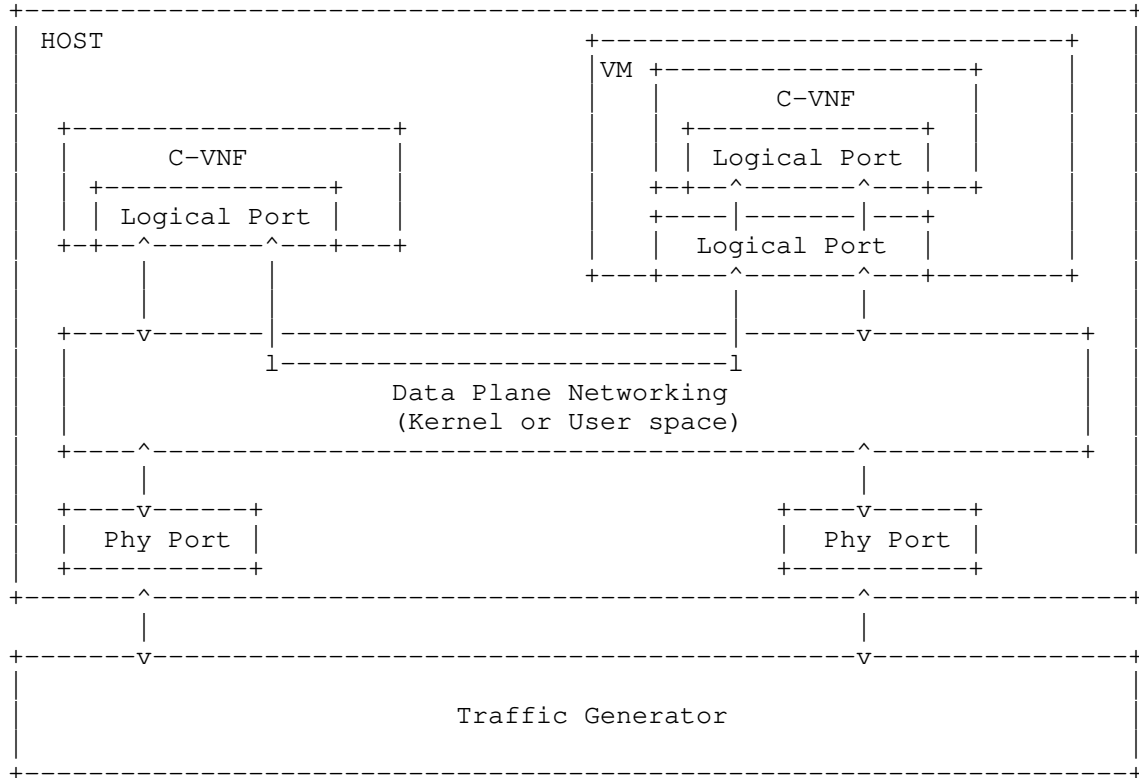


Figure 6: Single Host Test Scenario - BMP2VMP

- o VMP2VMP scenario

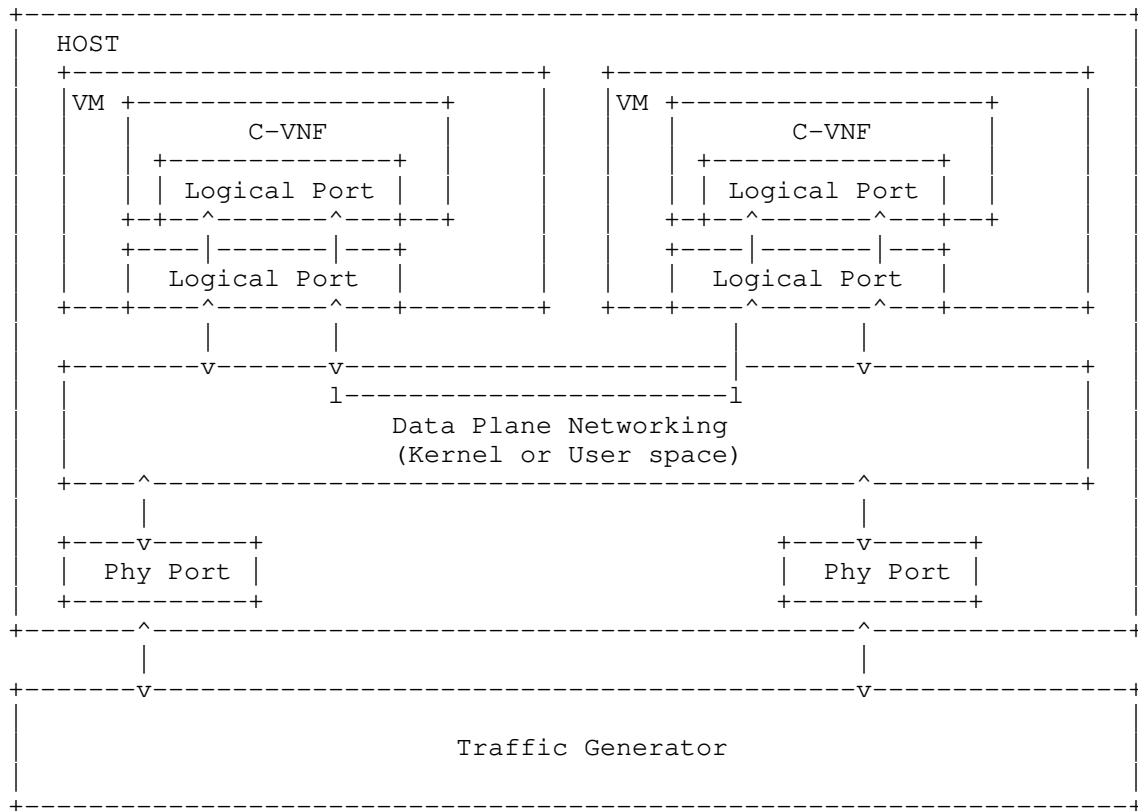


Figure 7: Single Host Test Scenario - VMP2VMP

5. Additional Considerations

When we consider benchmarking for not only containerized but also VM-based infrastructure 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 the hardware is tightly coupled with its function and only a 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 architectures were 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 the 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. Acknowledgement

We would like to thank 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/>>.
- [Multiqueue] "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
161, Gajeong-ro, Yoosung-gu
Dajeon, Dajeon 34129
Republic of Korea

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

Benchmarking Methodology Working Group
Internet-Draft
Intended status: Informational
Expires: 3 September 2022

K. Sun
ETRI
H. Yang
KT
J. Lee
T. Ngoc
Y. Kim
Soongsil University
March 2022

Considerations for Benchmarking Network Performance in Containerized
Infrastructures
draft-dcn-bmwg-containerized-infra-08

Abstract

This draft describes considerations for benchmarking network performance in containerized infrastructures. In the containerized infrastructure, Virtualized Network Functions (VNFs) are deployed on an operating-system-level virtualization platform by abstracting the user namespace as opposed to virtualization using a hypervisor. Hence, the system configurations and networking scenarios for benchmarking will be partially changed by how the resource allocation and network technologies are specified for containerized VNFs. This draft compares the state of the art in the container networking architecture with VM-based virtualized systems networking architecture and provides several test scenarios for benchmarking network performance in containerized infrastructures.

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 2 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
2. Terminology	4
3. Containerized Infrastructure Overview	4
4. Networking Models in Containerized Infrastructure	8
4.1. Kernel-space vSwitch Model	9
4.2. User-space vSwitch Model	10
4.3. eBPF Acceleration Model	10
4.4. Smart-NIC Acceleration Model	12
4.5. Model Combination	13
5. Performance Impacts	14
5.1. CPU Isolation / NUMA Affinity	14
5.2. Hugepages	15
5.3. Service Function Chaining	15
5.4. Additional Considerations	16
6. Security Considerations	16
7. References	16
7.1. Informative References	16
Appendix A. Benchmarking Experience(Contiv-VPP)	18
A.1. Benchmarking Environment	18
A.2. Trouble shooting and Result	22
Appendix B. Benchmarking Experience(SR-IOV with DPDK)	23
B.1. Benchmarking Environment	24
B.2. Trouble shooting and Results	27
Appendix C. Benchmarking Experience(Multi-pod Test)	27
C.1. Benchmarking Overview	27
C.2. Hardware Configurations	28
C.3. NUMA Allocation Scenario	30
C.4. Traffic Generator Configurations	30
C.5. Benchmark Results and Trouble-shootings	30
Authors' Addresses	31

1. Introduction

The Benchmarking Methodology Working Group (BMWG) has recently expanded its benchmarking scope from Physical Network Function (PNF) running on a 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 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, the containerized infrastructure virtualizes resources without a hypervisor, 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.

This draft aims to distinguish benchmarking of containerized infrastructure from the previous benchmarking methodology of common NFV infrastructure. Considering the point in [RFC8204] that virtual switch (vSwitch) is the networking principle of containerized infrastructure, this draft investigates different network models based on vSwitch location and acceleration technologies. At the same time, it is essential to uncover the impact of different deployment configurations on containerized infrastructure, such as resource

isolation, hugepages, service function chaining. The benchmark experiences of various combinations of these mentioned configurations and networking models are also presented in this draft as the references to set up and benchmark containerized 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 are 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. Containerized Infrastructure Overview

For 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. A major distinction of the containerized infrastructure from the VM-based infrastructure is the absence of a hypervisor. Without hypervisor, all C- VNFs share the same host resources, including but not limited to computing, storage, and networking resources, as well as the host Operating System(OS), kernel, and libraries. These architectural differences bring additional considerations of resource management impacts for benchmarking.

In a common containerized infrastructure, thanks to the proliferation of Kubernetes, the pod is defined as a basic unit for orchestration and management that can host multiple containers. Based on that, [ETSI-TST-009] defined two test scenario for container infrastructure as follows.

- o Container2Container: Communication between containers running in the same pod. it can be done by shared volumes or Inter-process communication (IPC).

- o Pod2Pod: Communication between containers running in the different pods.

As mentioned in [RFC8204], vSwitch is also an important aspect of the containerized infrastructure. For Pod2Pod communication, every pod has only one virtual Ethernet (vETH) interface. This interface is connected to the vSwitch via vETH pair for each container. Not only Pod2Pod but also Pod2External scenario that communicates with an external node is also required. In this case, vSwitch SHOULD support gateway and Network Address Translation (NAT) functionalities.

Figure 1 shows briefly differences of network architectures based on container deployment models. Basically, on bare metal, C-VNFs can be deployed as a cluster called POD by Kubernetes. Otherwise, each C-VNF can be deployed separately using Docker. In the former case, there is only one external network interface, even a POD containing more than one C-VNF. An additional deployment model considers a scenario where C-VNFs or PODs are running on VM. In our draft, we define new terminologies; BMP, which is Pod on bare metal, and VMP, which is Pod on VM.

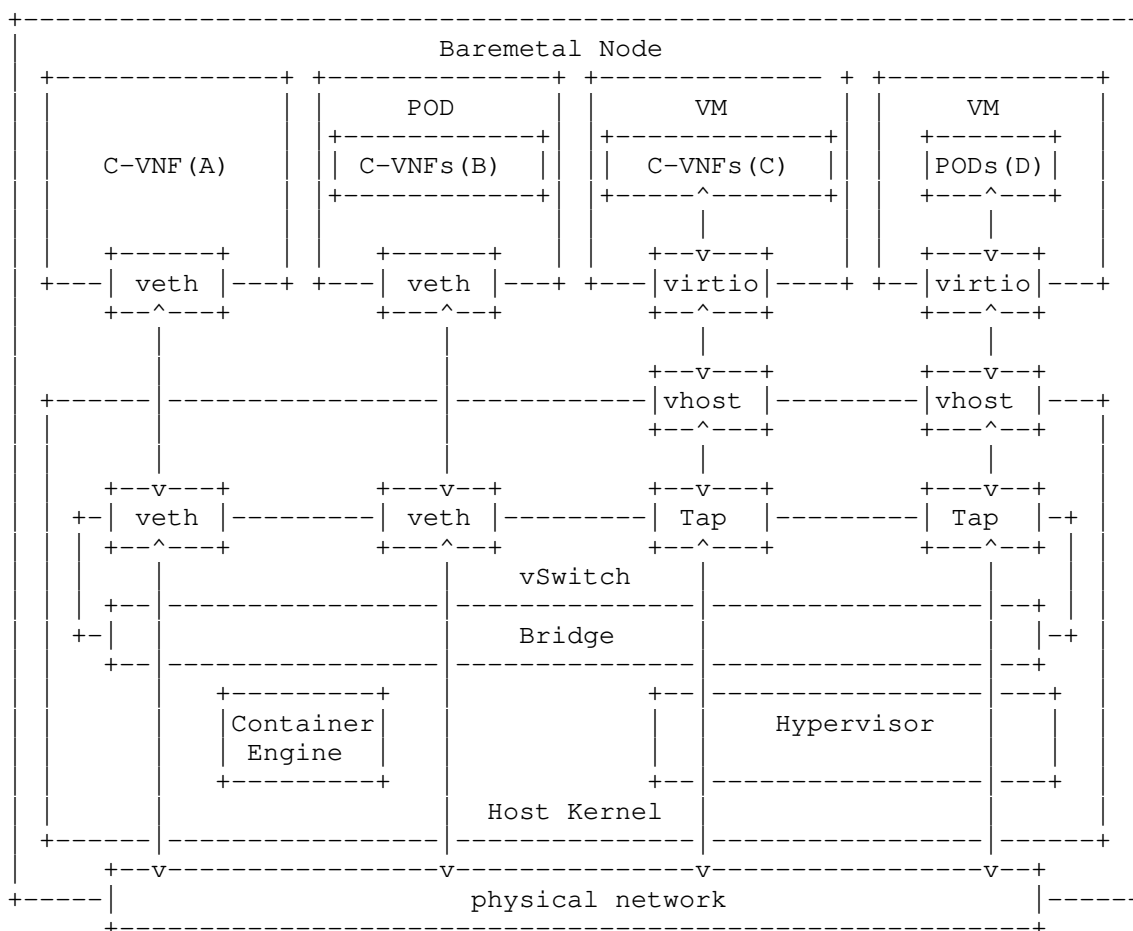


Figure 1: Examples of Networking Architecture based on Deployment Models - (A) C-VNF on Baremetal (B) Pod on Baremetal (BMP) (C) C-VNF on VM (D) Pod on VM (VMP)

In [ETSI-TST-009], they described data plane test scenarios in a single host. In that document, there are two scenarios for containerized infrastructure; Container2Container, which is internal communication between two containers in the same Pod, and the Pod2Pod model, which is communication between two containers running in different Pods. According to our new terminologies, we can call the Pod2Pod model the 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

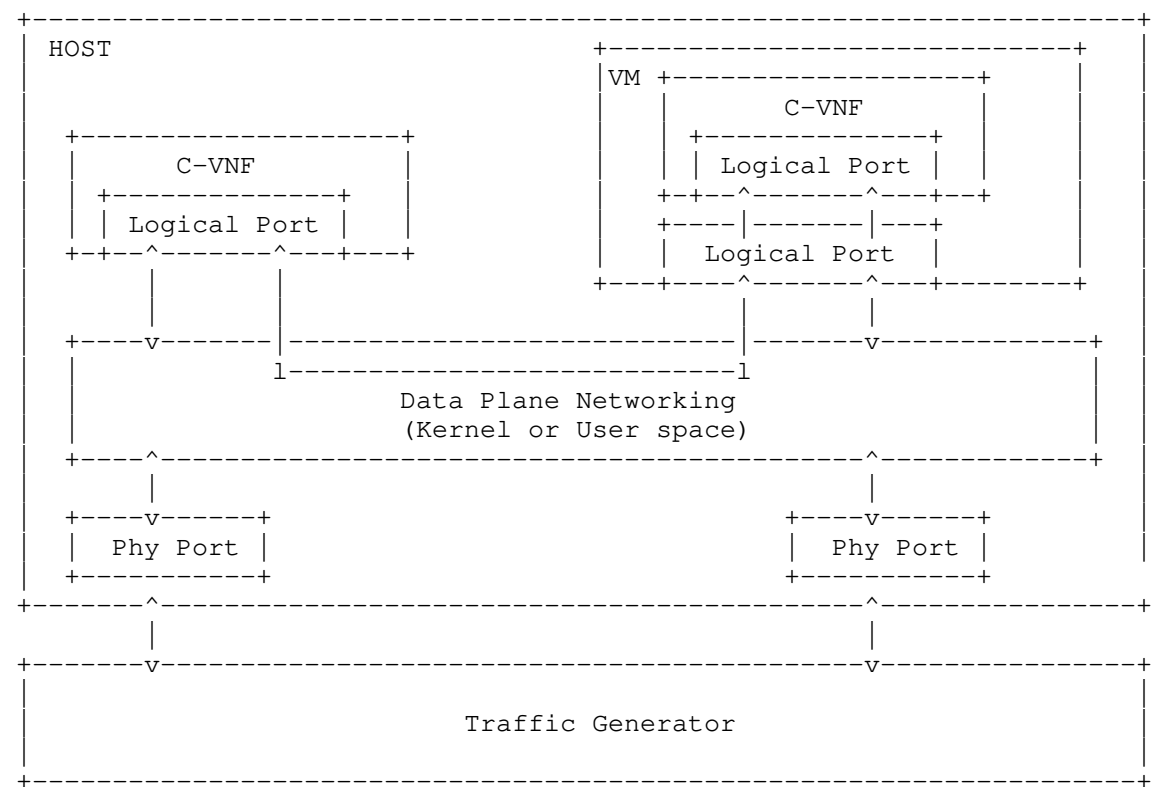


Figure 2: Single Host Test Scenario - BMP2VMP

o VMP2VMP scenario

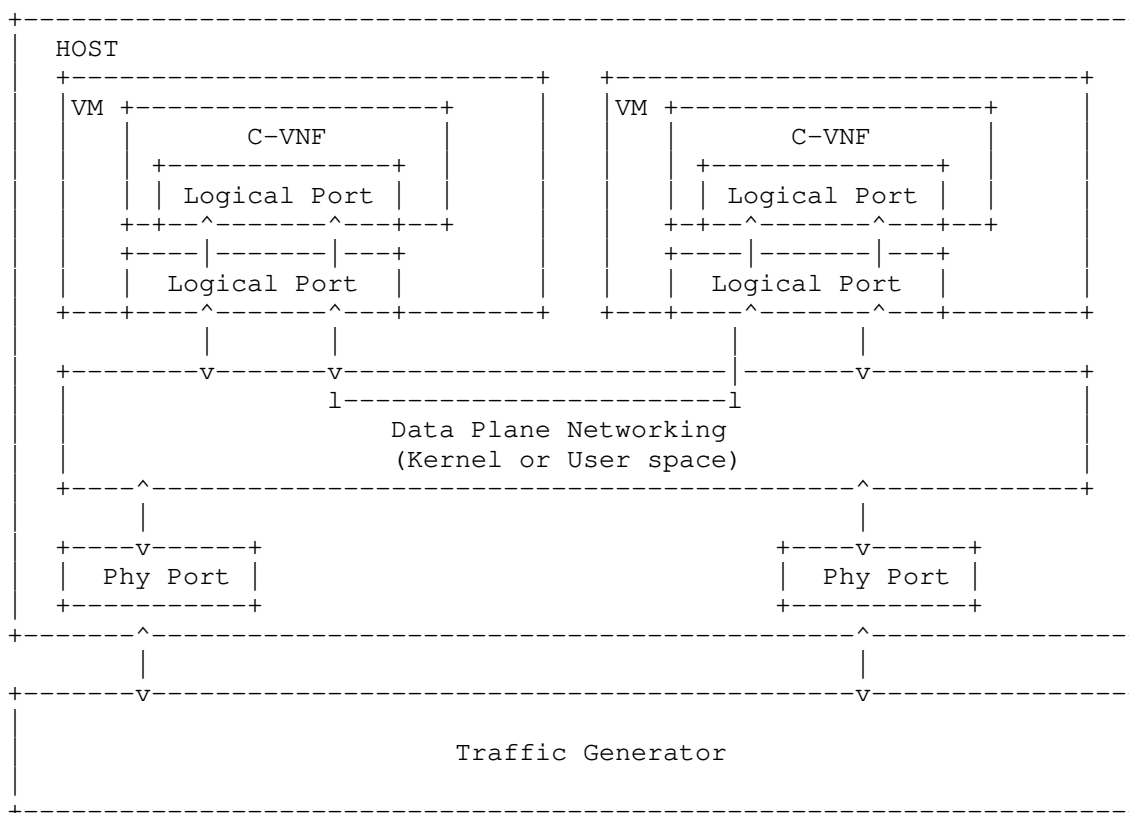


Figure 3: Single Host Test Scenario - VMP2VMP

4. Networking Models in Containerized Infrastructure

Container networking services are provided as network plugins. Basically, by using them, network services are deployed as an isolation environment from container runtime through the host namespace, creating a virtual interface, allocating interface and IP address to C-VNF. Since the containerized infrastructure has different network architecture depending on its using plugins, it is necessary to specify the plugin used in the infrastructure. Especially for Kubernetes infrastructure, several Container Networking Interface (CNI) plugins are developed, which describes network configuration files in JSON format, and plugins are instantiated as new namespaces. When the CNI plugin is initiated, it pushes forwarding rules and networking policies to the existing vSwitch (i.e., Linux bridge, Open vSwitch) or creates its own switch functions to provide networking service.

The container network model can be classified according to the location of the vSwitch component. There are some CNI plugins that provide networking without the vSwitch components; however, this draft focuses on plugins using vSwitch components.

4.1. Kernel-space vSwitch Model

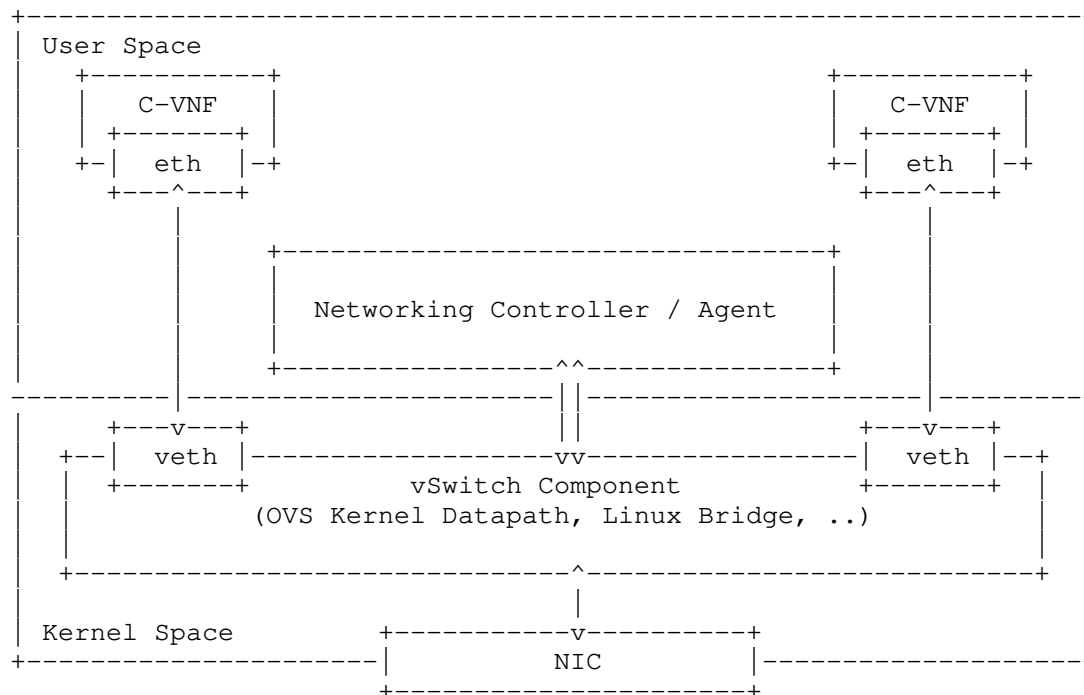


Figure 4: Examples of Kernel-Space vSwitch Model

Figure 4 shows kernel-space vSwitch model. In this model, because the vSwitch component is running on kernel space, data packets should be processed in-network stack of host kernel before transferring packets to the C-VNF running in user-space. Not only pod2External but also pod2pod traffic should be processed in the kernel space. For dynamic networking configuration, the Forwarding policy can be pushed by the controller/agent located in the user-space. In the case of Open vSwitch (OVS) [OVS], the first packet of flow can be sent to the user space agent (ovs-switchd) for forwarding decision. Kernel-space vSwitch models are listed below;

- o Docker Network[Docker-network], Flannel Network[Flannel],
- OVS(OpenvSwitch) [OVS], OVN(Open Virtual Network) [OVN]

4.2. User-space vSwitch Model

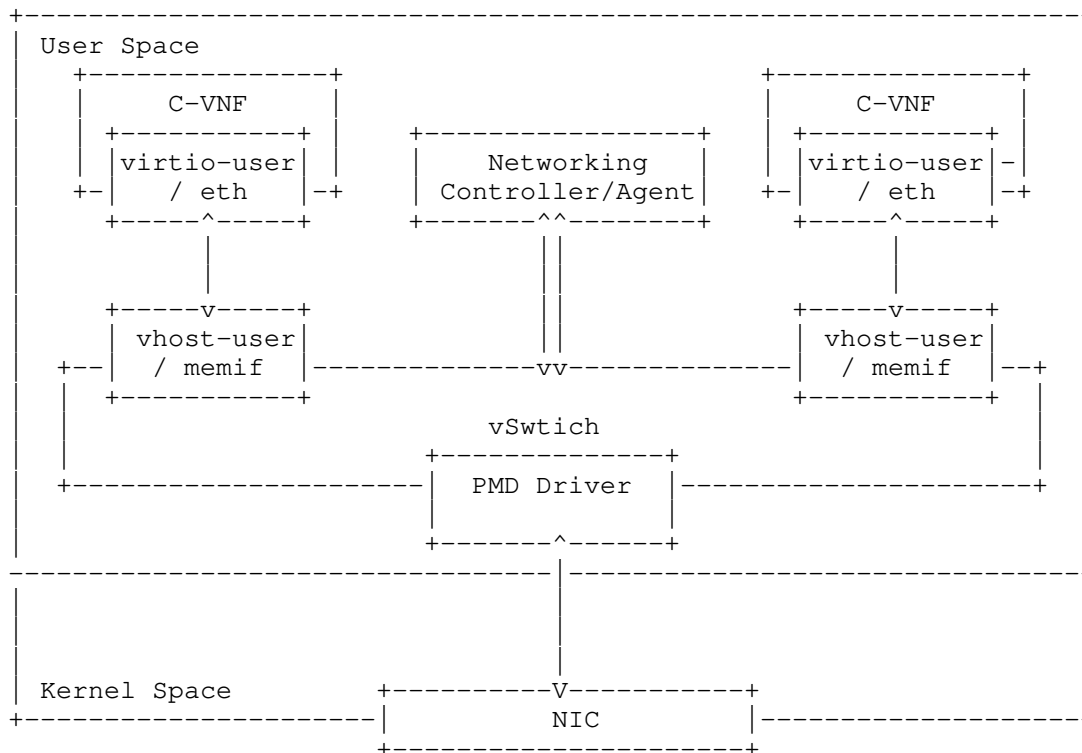


Figure 5: Examples of User-Space vSwitch Model

Figure 5 shows user-space vSwitch model, in which data packets from physical network port are bypassed kernel processing and delivered directly to the vSwitch running on user-space. This model is commonly considered as Data Plane Acceleration (DPA) technology since it can achieve high-rate packet processing than a kernel-space network with limited packet throughput. For bypassing kernel and directly transferring the packet to vSwitch, Data Plane Development Kit (DPDK) is essentially required. With DPDK, an additional driver called Pull-Mode Driver (PMD) is created on vSwitch. PMD driver must be created for each NIC separately. User-space vSwitch models are listed below;

- o ovs-dpdk[ovs-dpdk], vpp[vpp]

4.3. eBPF Acceleration Model

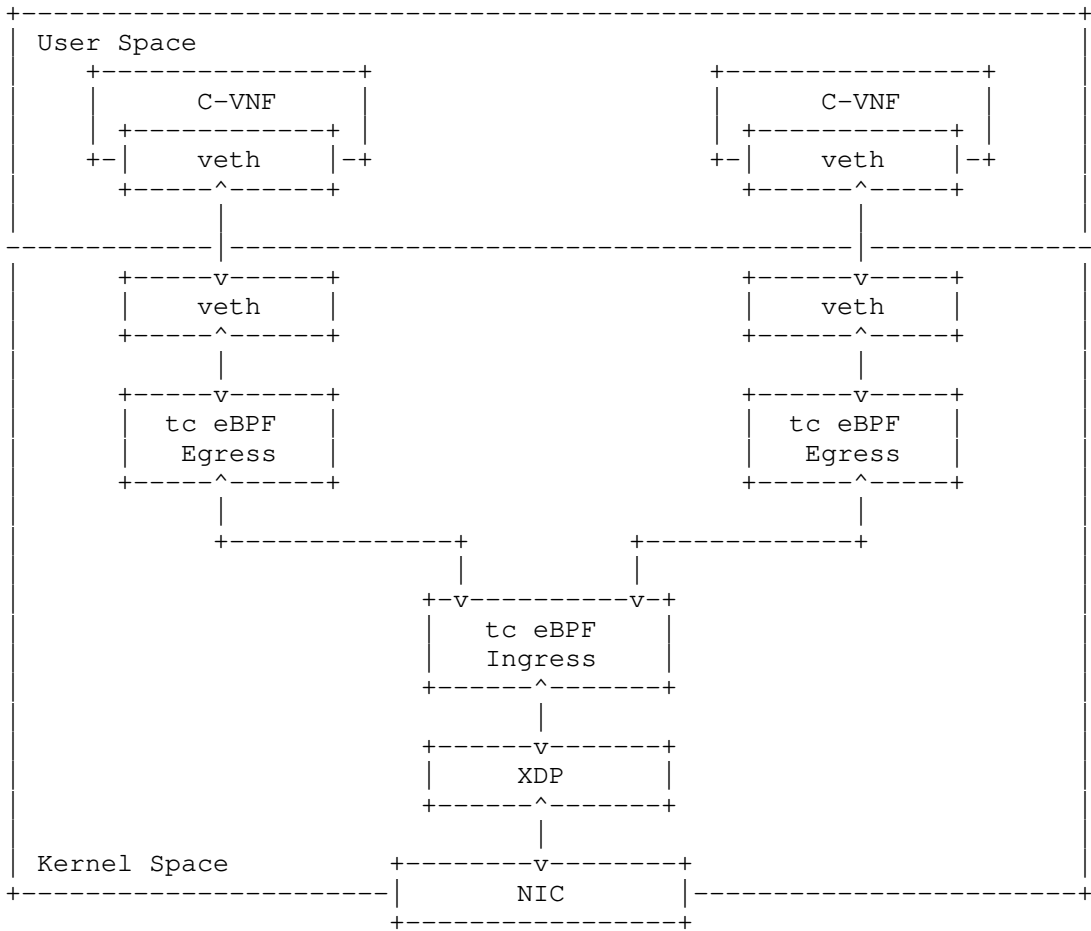


Figure 6: Examples of eBPF Acceleration Model

Figure 6 shows eBPF Acceleration model, which leverages extended Berkeley Packet Filter (eBPF) technology [eBPF] to achieve high-performance packet processing. It enables execution of sandboxed programs inside abstract virtual machines within the Linux kernel without changing the kernel source code or loading the kernel module. To accelerate data plane performance, eBPF programs are attached to different BPF hooks inside the linux kernel stack.

One type of BPF hook is the eXpress Data Path (XDP) at the networking driver. It is the first hook that triggers eBPF program upon packet reception from external network. The other type of BPF hook is Traffic Control Ingress/Egress eBPF hook (tc eBPF). These hooks are attached to the vETH pair of the pod and the XDP hook. The tc Egress

eBPF hooks at the vETH pair enforce policy on all traffic exit the pod, while the tc Ingress eBPF hook at the end of the kernel networking runs after initial packet processing from XDP hook.

On the egress datapath side, whenever a packet exits the pod, it goes through vETH pair then is picked up by the tc egress eBPF hook. These hooks trigger eBPF programs to forward the packet directly to the external facing network interface, bypassing all of the kernel network layer processing such as iptables. On the ingress datapath side, eBPF programs at the XDP and tc ingress eBPF hook pick up packets from the network device and directly deliver it to the vETH interface pair, or bypassing context-switching process to the pod network namespace in the case of Cilium project [Cilium].

Notable eBPF Acceleration models are 2 CNI plugin projects: Calico[Calico], Cilium[Cilium]. In the case of Cilium, eBPF/XDP program can be offloaded directly on the smart NIC card, which allows data plane acceleration without using the CPU. Container network performance of these eBPF-based project is reported in [cilium-benchmark].

4.4. Smart-NIC Acceleration Model

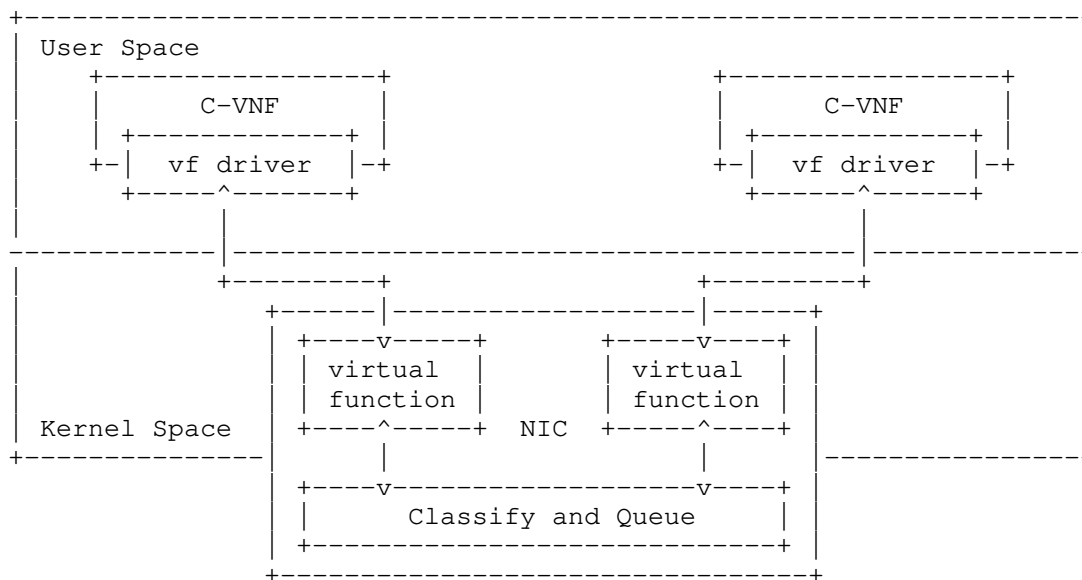


Figure 7: Examples of Smart-NIC Acceleration Model

Figure 7 shows Smart-NIC acceleration model, which does not use vSwitch component. This model can be separated into two technologies.

One is Single-Root I/O Virtualization (SR-IOV) [SR-IOV], which is an extension of PCIe specifications to enable multiple partitions running simultaneously within a system to share PCIe devices. In the NIC, there are virtual replicas of PCI functions known as virtual functions (VF), and each of them is directly connected to each container's network interfaces. Using SR-IOV, data packets from external bypass both kernel and user space and are directly forwarded to container's virtual network interface.

The other technology is eBPF/XDP programs offloading to Smart-NIC card as mentioned in the previous section. It enables general acceleration of eBPF. eBPF programs are attached to XDP and run at the Smart-NIC card, which allows server CPUs to perform more application-level work. However, not all Smart-NIC cards provide eBPF/XDP offloading support.

4.5. Model Combination

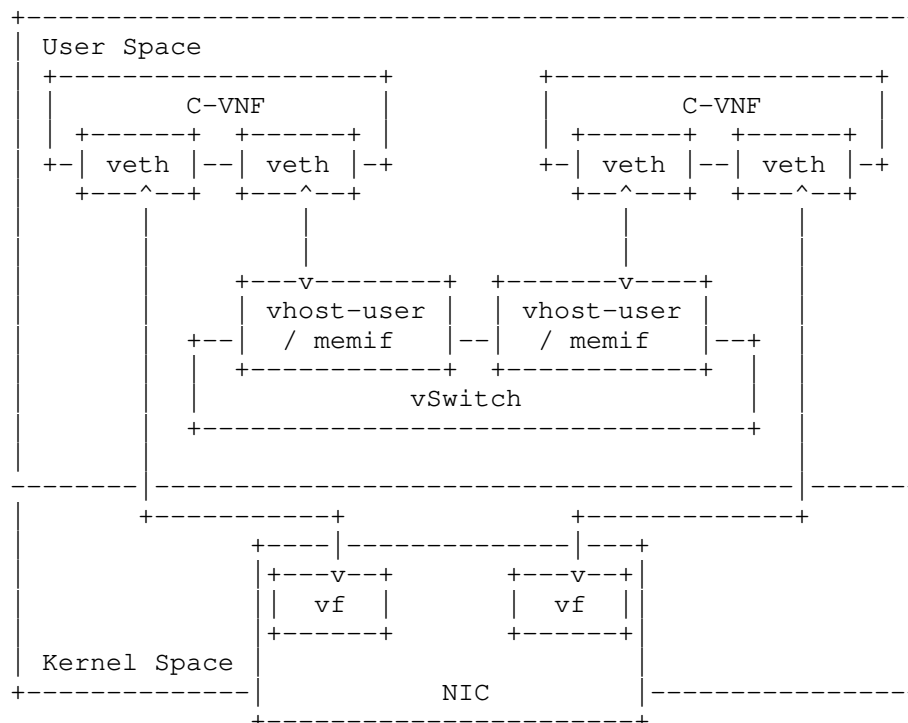


Figure 8: Examples of Model Combination deployment

Figure 8 shows the networking model when combining user-space vSwitch model and Smart-NIC acceleration model. This model is frequently considered in service function chain scenarios when two different types of traffic flows are present. These two types are North/South traffic and East/West traffic.

North/South traffic is the type that packets are received from other servers and routed through VNF. For this traffic type, Smart-NIC model such as SR-IOV is preferred because packets always have to pass the NIC. User-space vSwitch involvement in north-south traffic will create more bottlenecks. On the other hand, East/West traffic is a form of sending and receiving data between containers deployed in the same server and can pass through multiple containers. For this type, user-space vSwitch models such as OVS-DPDK and VPP are preferred because packets are routed within the user space only and not through the NIC.

The throughput advantages of these different networking models with different traffic direction cases are reported in [Intel-SRIOV-NFV].

5. Performance Impacts

5.1. CPU Isolation / NUMA Affinity

CPU pinning enables benefits such as maximizing cache utilization, eliminating operating system thread scheduling overhead as well as coordinating network I/O by guaranteeing resources. This technology is very effective in avoiding the "noisy neighbor" problem, and it is already proved in existing experience [Intel-EPA].

Using NUMA, performance will be increasing not CPU and memory but also network since that network interface connected PCIe slot of specific NUMA node have locality. Using NUMA requires a strong understanding of VNF's memory requirements. If VNF uses more memory than a single NUMA node contains, the overhead will occur due to being spilled to another NUMA node. Network performance can be changed depending on the location of the NUMA node whether it is the same NUMA node where the physical network interface and CNF are attached to. There is benchmarking experience for cross-NUMA performance impacts [ViNePERF]. In that tests, they consist of cross-NUMA performance with 3 scenarios depending on the location of the traffic generator and traffic endpoint. As the results, it was verified as below:

- o A single NUMA Node serving multiple interfaces is worse than Cross-NUMA Node performance degradation

- o Worse performance with VNF sharing CPUs across NUMA

5.2. Hugepages

Hugepage configures a large page size of memory to reduce Translation Lookaside Buffer (TLB) miss rate and increase the application performance. This increases the performance of logical/virtual to physical address lookups performed by a CPU's memory management unit, and overall system performance. In the containerized infrastructure, the container is isolated at the application level, and administrators can set huge pages more granular level (e.g., Kubernetes allows to use of 512M bytes huge pages for the container as default values). Moreover, this page is dedicated to the application but another process, so the application uses the page more efficiently way. From a network benchmark point of view, however, the impact on general packet processing can be relatively negligible, and it may be necessary to consider the application level to measure the impact together. In the case of using the DPDK application, as reported in [Intel-EPA], it was verified to improve network performance because packet handling processes are running in the application together.

5.3. Service Function Chaining

When we consider benchmarking for containerized and VM-based infrastructure and network functions, benchmarking scenarios may contain various operational use cases. Traditional black-box benchmarking focuses on measuring the in-out performance of packets from physical network ports since the hardware is tightly coupled with its function and only a 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 architectures were described in [ETSI-TST-009]) rather than dedicated to a single VNF. This scenario is called Service Function Chaining. Therefore, benchmarking scenarios should reflect operational considerations such as the number of VNFs or network services defined by a set of VNFs in a single host. [service-density] proposed a way for measuring the performance of multiple NFV service instances at a varied service density on a single host, which is one example of these operational benchmarking aspects. Another aspect in benchmarking service function chaining scenario should be considered is different network acceleration technologies. Network performance differences may occur because of different traffic patterns based on the provided acceleration method.

5.4. Additional Considerations

Apart from the single-host test scenario, the multi-hosts scenario should also be considered in container network benchmarking, where container services are deployed across different servers. To provide network connectivity for container-based VNFs between different server nodes, inter-node networking is required. According to [ETSI-NFV-IFA-038], there are several technologies to enable inter-node network: overlay technologies using a tunnel endpoint (e.g. VXLAN, IP in IP), routing using Border Gateway Protocol (BGP), layer 2 underlay, direct network using dedicated NIC for each pod, or load balancer using LoadBalancer service type in Kubernetes. Different protocols from these technologies may cause performance differences in container networking.

6. Security Considerations

TBD

7. References

7.1. Informative References

- [Calico] "Project Calico", July 2019,
<<https://docs.projectcalico.org/>>.
- [Cilium] "Cilium Documentation", March 2022,
<<https://docs.cilium.io/en/stable//>>.
- [cilium-benchmark]
Cilium, "CNI Benchmark: Understanding Cilium Network Performance", May 2021,
<<https://cilium.io/blog/2021/05/11/cni-benchmark>>.
- [Docker-network]
"Docker, Libnetwork design", July 2019,
<<https://github.com/docker/libnetwork/>>.
- [DPDK_eBPF]
"DPDK-Berkeley Packet Filter Library", August 2021,
<https://doc.dpdk.org/guides/prog_guide/bpf_lib.html>.
- [eBPF] "eBPF, extended Berkeley Packet Filter", July 2019,
<<https://www.iovisor.org/technology/ebpf>>.

- [ETSI-NFV-IFA-038] "Network Functions Virtualisation (NFV) Release 4; Architectural Framework; Report on network connectivity for container-based VNF", November 2021.
- [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/>>.
- [Intel-EPA] Intel, "Enhanced Platform Awareness in Kubernetes", 2018, <<https://builders.intel.com/docs/networkbuilders/enhanced-platform-awareness-feature-brief.pdf>>.
- [Intel-SRIOV-NFV] Patrick, K. and J. Brian, "SR-IOV for NFV Solutions Practical Considerations and Thoughts", February 2017.
- [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, <<https://www.rfc-editor.org/rfc/rfc2119>>.
- [RFC8172] Morton, A., "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure", RFC 8172, July 2017, <<https://www.rfc-editor.org/rfc/rfc8172>>.
- [RFC8204] Tahhan, M., O'Mahony, B., and A. Morton, "Benchmarking Virtual Switches in the Open Platform for NFV (OPNFV)", RFC 8204, September 2017, <<https://www.rfc-editor.org/rfc/rfc8204>>.

- [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>>.
- [userspace-cni] "Userspace CNI Plugin", August 2021, <<https://github.com/intel/userspace-cni-network-plugin>>.
- [ViNePERF] Anuket Project, "Cross-NUMA performance measurements with VSPERF", March 2019, <<https://wiki.anuket.io/display/HOME/Cross-NUMA+performance+measurements+with+VSPERF>>.
- [vpp] "VPP with Containers", July 2019, <<https://fdio-vpp.readthedocs.io/en/latest/usecases/containers.html>>.

Appendix A. Benchmarking Experience (Contiv-VPP)

A.1. Benchmarking Environment

In this test, our purpose is to test the performance of user-space based model for container infrastructure and figure out the relationship between resource allocation and network performance. With respect to this, we set up Contiv-VPP, one of the user-space based network solutions in container infrastructure and tested like below.

- o Three physical server for benchmarking

Node Name	Specification	Description
Conatiner Control for Master	<ul style="list-style-type: none"> - Intel(R) Xeon(R) CPU E5-2690 (2Socket X 12Core) - MEM 128G - DISK 2T - Control plane : 1G 	Container Deployment and Network Allocation <ul style="list-style-type: none"> - ubuntu 18.04 - Kubernetes Master - CNI Conterller .. Contive VPP Controller .. Contive VPP Agent
Conatiner Service for Worker	<ul style="list-style-type: none"> - Intel(R) Xeon(R) Gold 6148 (2socket X 20Core) - MEM 128G - DISK 2T - Control plane : 1G - Data plane : MLX 10G (1NIC 2PORT) 	Container Service <ul style="list-style-type: none"> - ubuntu 18.04 - Kubernetes Worker - CNI Agent .. Contive VPP Agent
Packet Generator	<ul style="list-style-type: none"> - Intel(R) Xeon(R) CPU E5-2690 (2Socket X 12Core) - MEM 128G - DISK 2T - Control plane : 1G - Data plane : MLX 10G (1NIC 2PORT) 	Packet Generator <ul style="list-style-type: none"> - CentOS 7 - installed Trex 2.4

Figure 9: Test Environment-Server Specification

- o The architecture of benchmarking

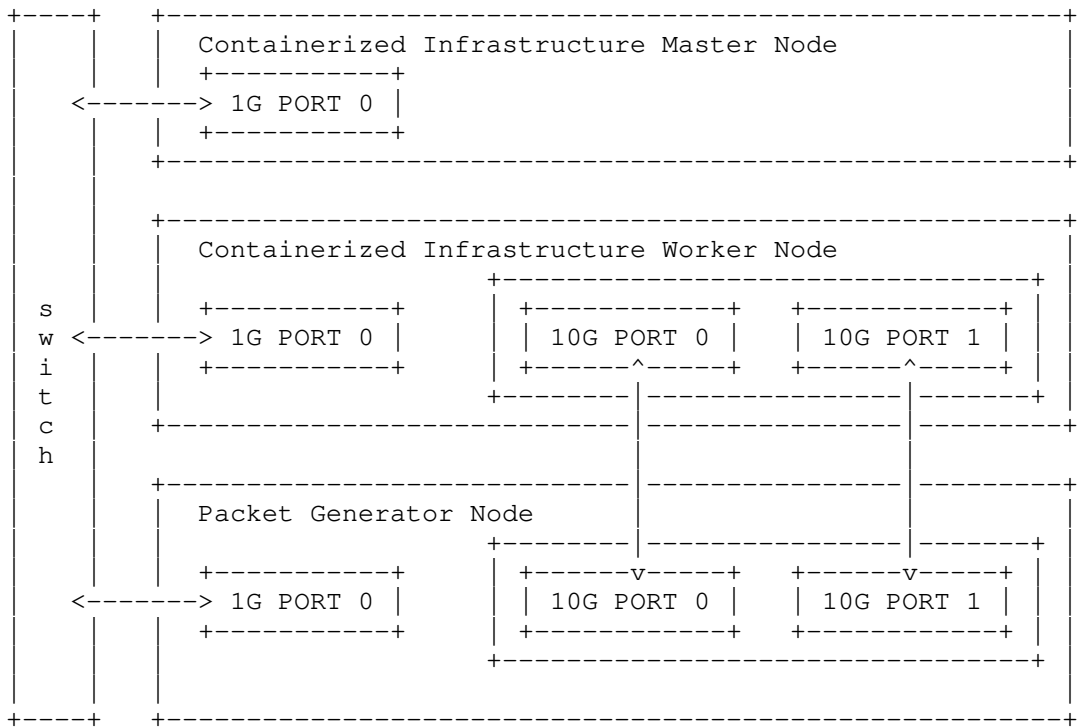


Figure 10: Test Environment-Architecture

o Network model of Containerized Infrastructure(User space Model)

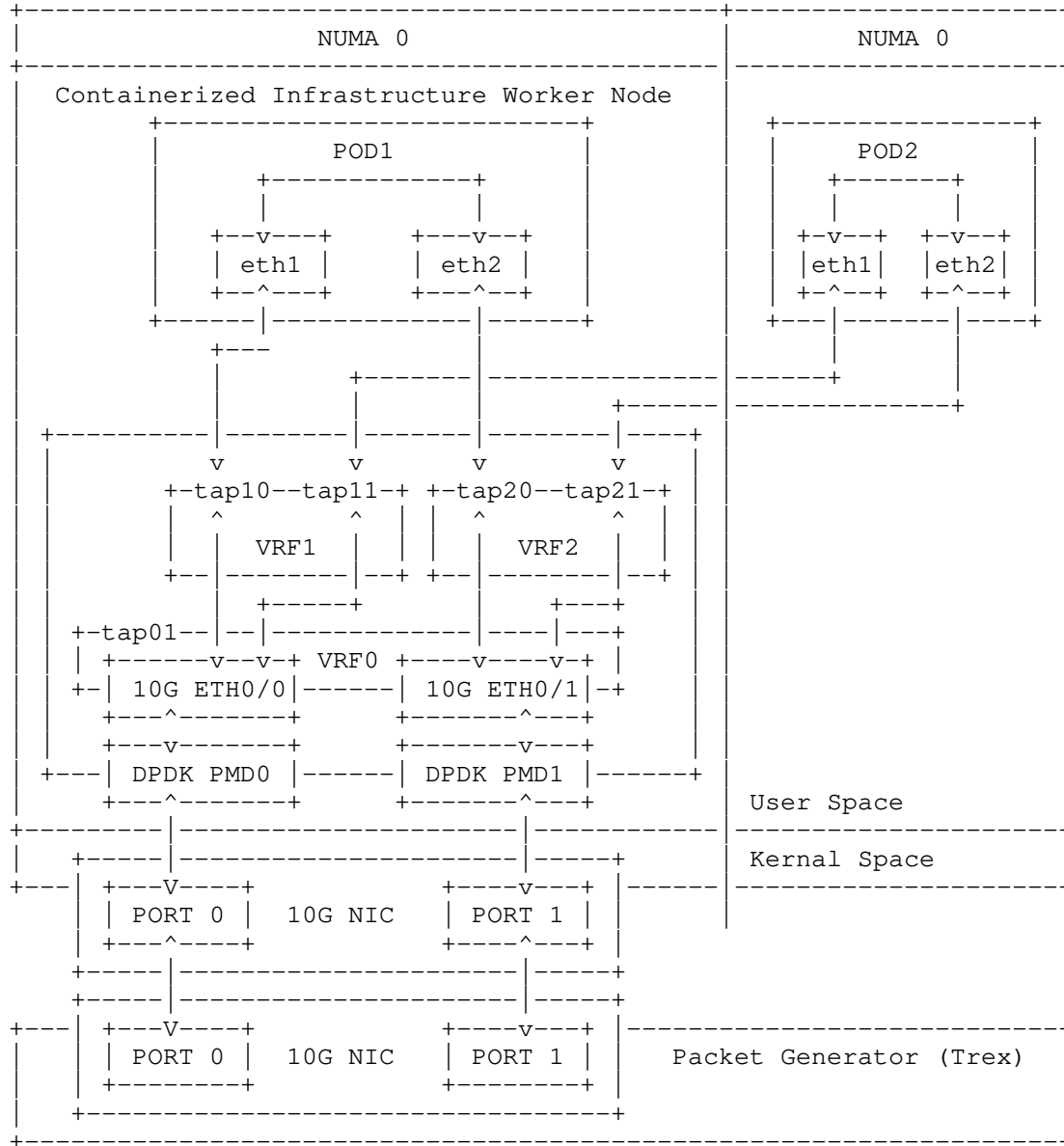


Figure 11: Test Environment-Network Architecture

We set up a Contive-VPP network to benchmark the user space container network model in the containerized infrastructure worker node. We set up network interface at NUMA0, and we created different network subnets VRF1, VRF2 to classify input and output data traffic,

respectively. And then, we assigned two interfaces which connected to VRF1, VRF2 and, we setup routing table to route Trex packet from eth1 interface to eth2 interface in POD.

A.2. Trouble shooting and Result

In this environment, we confirmed that the routing table doesn't work when we send packets using Trex packet generator. The reason is that when kernel space based network configured, ip forwarding rule is processed to kernel stack level while 'ip packet forwarding rule' is processed only in vrf0, which is the default virtual routing and forwarding (VRF0) in VPP. The above testing architecture makes problem since vrf1 and vrf2 interface couldn't route packet. According to above result, we assigned vrf0 and vrf1 to POD and, data flow is like below.

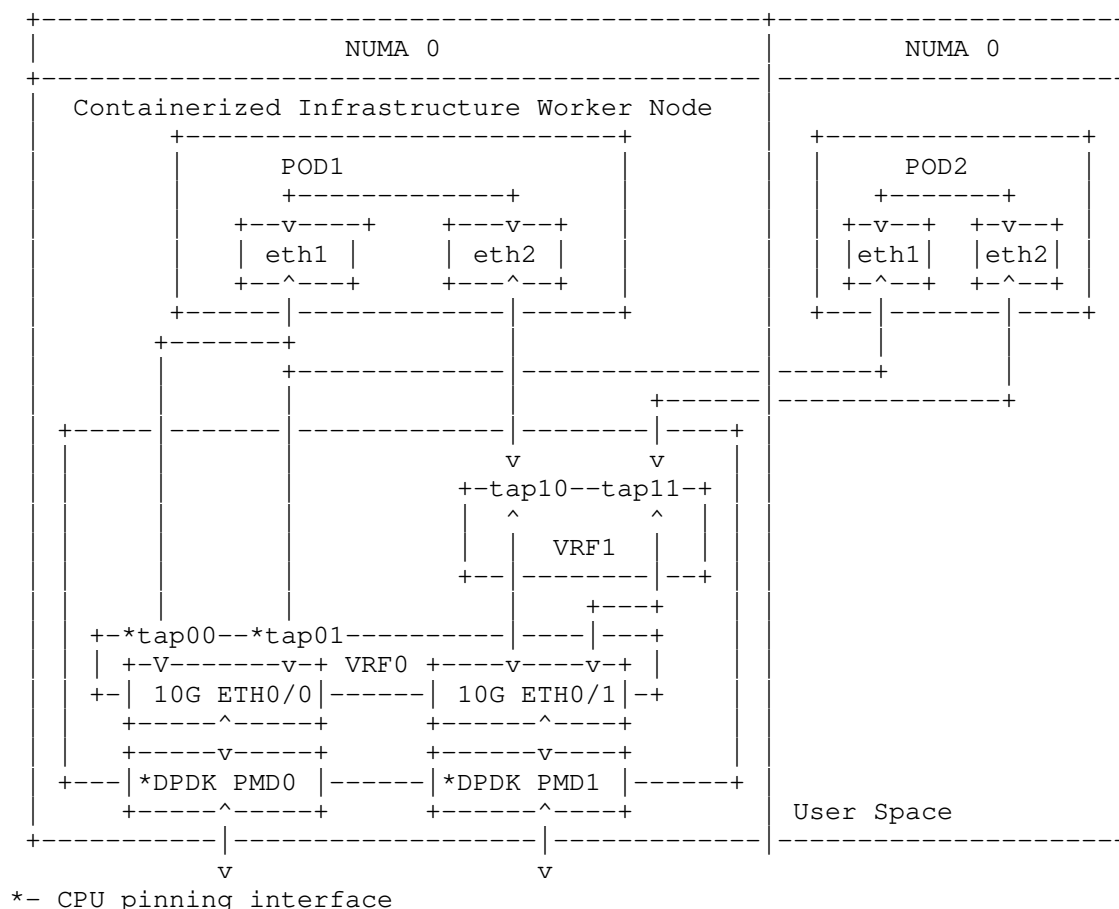


Figure 12: Test Environment-Network Architecture(CPU Pinning)

We conducted benchmarking with three conditions. The test environments are as follows. - Basic VPP switch - General kubernetes (No CPU Pinning) - Shared Mode / Exclusive mode. In the basic Kubernetes environment, all PODs share a host's CPU. Shared mode is that some POD share a pool of CPU assigned to specific PODs. Exclusive mode is that a specific POD dedicates a specific CPU to use. In shared mode, we assigned two CPUs for several PODs, in exclusive mode, we dedicated one CPU for one POD, independently. The result is like Figure 13. First, the test was conducted to figure out the line rate of the VPP switch, and the basic Kubernetes performance. After that, we applied NUMA to the network interface using Shared Mode and Exclusive Mode in the same node and different node. In Exclusive and Shared mode tests, we confirmed that Exclusive mode showed better performance than Shared mode when same NUMA CPU was assigned, respectively. However, we confirmed that performance is reduced at the section between the vpp switch and the POD, affecting the total result.

Model	NUMA Mode (pinning)	Result (Gbps)
Maximum Line Rate	N/A	3.1
	same NUMA	9.8
Without CMK	N/A	1.5
CMK-Exclusive Mode	same NUMA	4.7
	Different NUMA	3.1
CMK-shared Mode	same NUMA	3.5
	Different NUMA	2.3

Figure 13: Test Results

Appendix B. Benchmarking Experience(SR-IOV with DPDK)

B.1. Benchmarking Environment

In this test, our purpose is to test the performance of Smart-NIC acceleration model for container infrastructure and figure out relationship between resource allocation and network performance. With respect to this, we setup SRIOV combining with DPDK to bypass the Kernel space in container infrastructure and tested based on that.

- o Three physical server for benchmarking

Node Name	Specification	Description
Conatiner Control for Master	<ul style="list-style-type: none"> - Intel(R) Core(TM) i5-6200U CPU (1socket x 4Core) - MEM 8G - DISK 500GB - Control plane : 1G 	Container Deployment and Network Allocation <ul style="list-style-type: none"> - ubuntu 18.04 - Kubernetes Master - CNI Conterller MULTUS CNI SRIOV plugin with DPDK
Conatiner Service for Worker	<ul style="list-style-type: none"> - Intel(R) Xeon(R) E5-2620 v3 @ 2.4Ghz (1socket X 6Core) - MEM 128G - DISK 2T - Control plane : 1G - Data plane : XL710-qda2 (1NIC 2PORT- 40Gb) 	Container Service <ul style="list-style-type: none"> - Centos 7.7 - Kubernetes Worker - CNI Agent MULTUS CNI SRIOV plugin with DPDK
Packet Generator	<ul style="list-style-type: none"> - Intel(R) Xeon(R) Gold 6148 @ 2.4Ghz (2Socket X 20Core) - MEM 128G - DISK 2T - Control plane : 1G - Data plane : XL710-qda2 (1NIC 2PORT- 40Gb) 	Packet Generator <ul style="list-style-type: none"> - CentOS 7.7 - installed Trex 2.4

Figure 14: Test Environment-Server Specification

- o The architecture of benchmarking

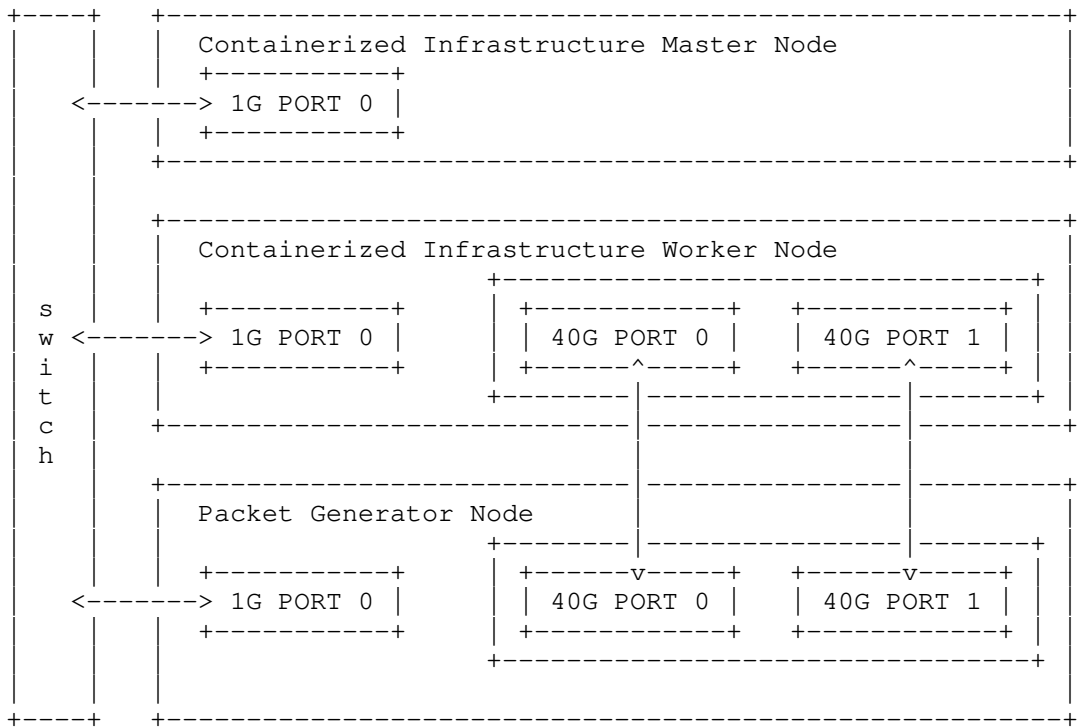


Figure 15: Test Environment-Architecture

o Network model of Containerized Infrastructure(User space Model)

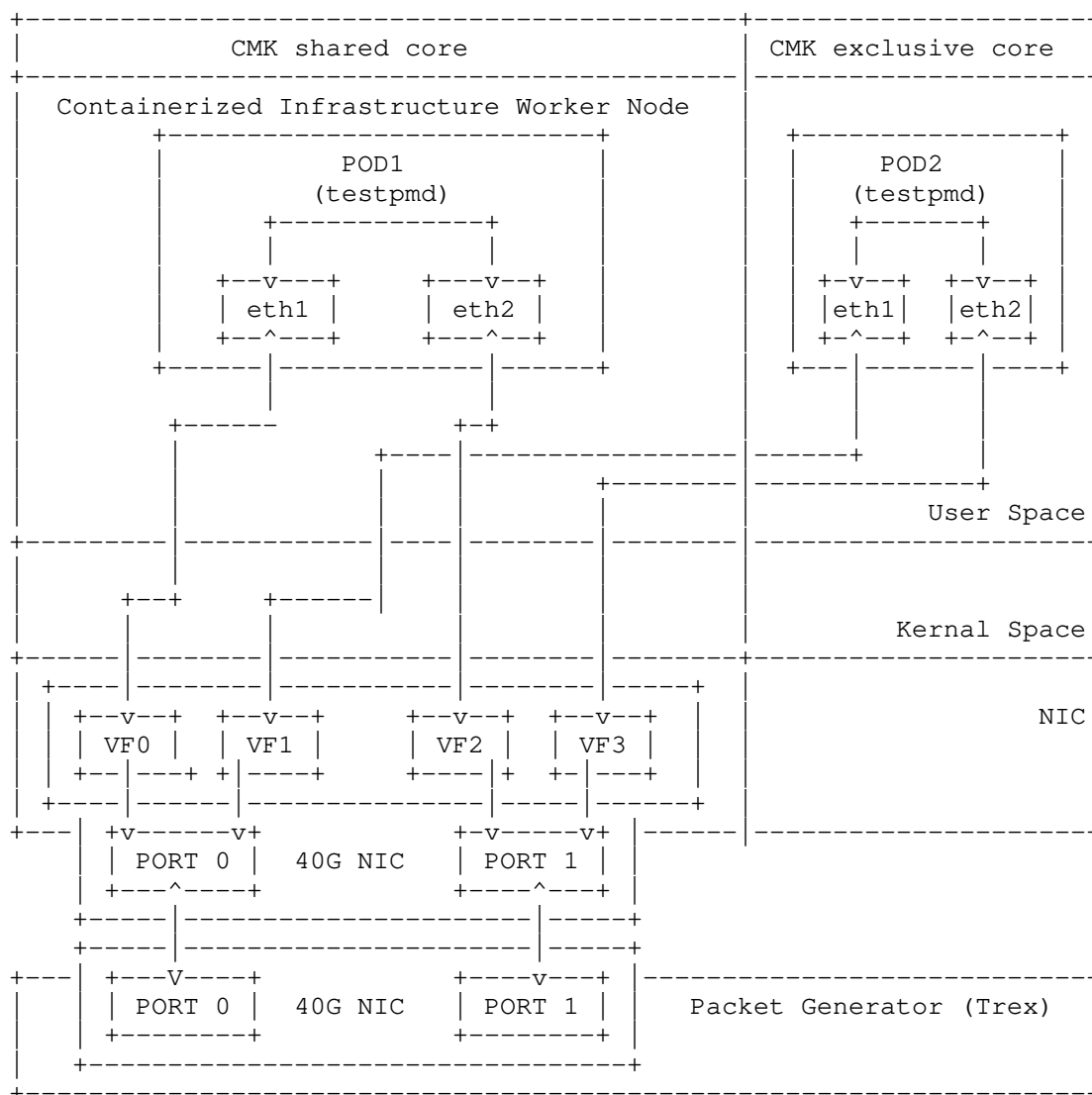


Figure 16: Test Environment-Network Architecture

We set up a Multus CNI, SRIOV CNI with DPDK to benchmark the user-space container network model in the containerized infrastructure worker node. The Multus CNI support creates multiple interfaces for a container. The traffic is bypassed the Kernel space by SRIOV with DPDK. We established two modes of CMK: shared core and exclusive core. We created VFs for each network interface of a container. Then, we set up TREX to route packet from eth1 to eth2 in a POD.

B.2. Trouble shooting and Results

Figure 17 shows the test results when using 1518 bytes packet traffic from the T-Rex traffic generator. First, we get the maximum line rate of the system using SR-IOV as the packet acceleration technique. Then we measured throughput when applying the CMK feature. We observed similar results as VPP CPU Pinning test. The default Kubernetes system without CMK feature enabled had the worst performance as the CPU resources are shared without any isolation. When the CMK feature is enabled, Exclusive Mode performed better than Shared Mode because each pod had its own dedicated CPU.

Model	Result (Gbps)
Maximum Line Rate	39.3
Without CMK	11.5
CMK-Exclusive Mode	39.2
CMK-shared Mode	29.6

Figure 17: SR-IOV CPU Pinning Test Results

Appendix C. Benchmarking Experience (Multi-pod Test)

C.1. Benchmarking Overview

The main goal of this experience was to benchmark the multi-pod scenario, in which packets are traversed through two pods. To create additional interfaces for forwarding packets between two pods, Multus CNI was used. We compared two userspace-vSwitch model network technologies: OVS/DPDK and VPP-memif. Since that vpp-memif has a different packet forwarding mechanism by using shared memory interface, it is expected that vpp-memif may provide higher performance than OVS-DPDK. Also, we consider NUMA impact for both cases, and made 6 scenarios depending on CPU location of vSwitch and two pods. Figure 18 is packet forwarding scenario in this test, where two pods run on the same host and vSwitch delivers packets between two pods.

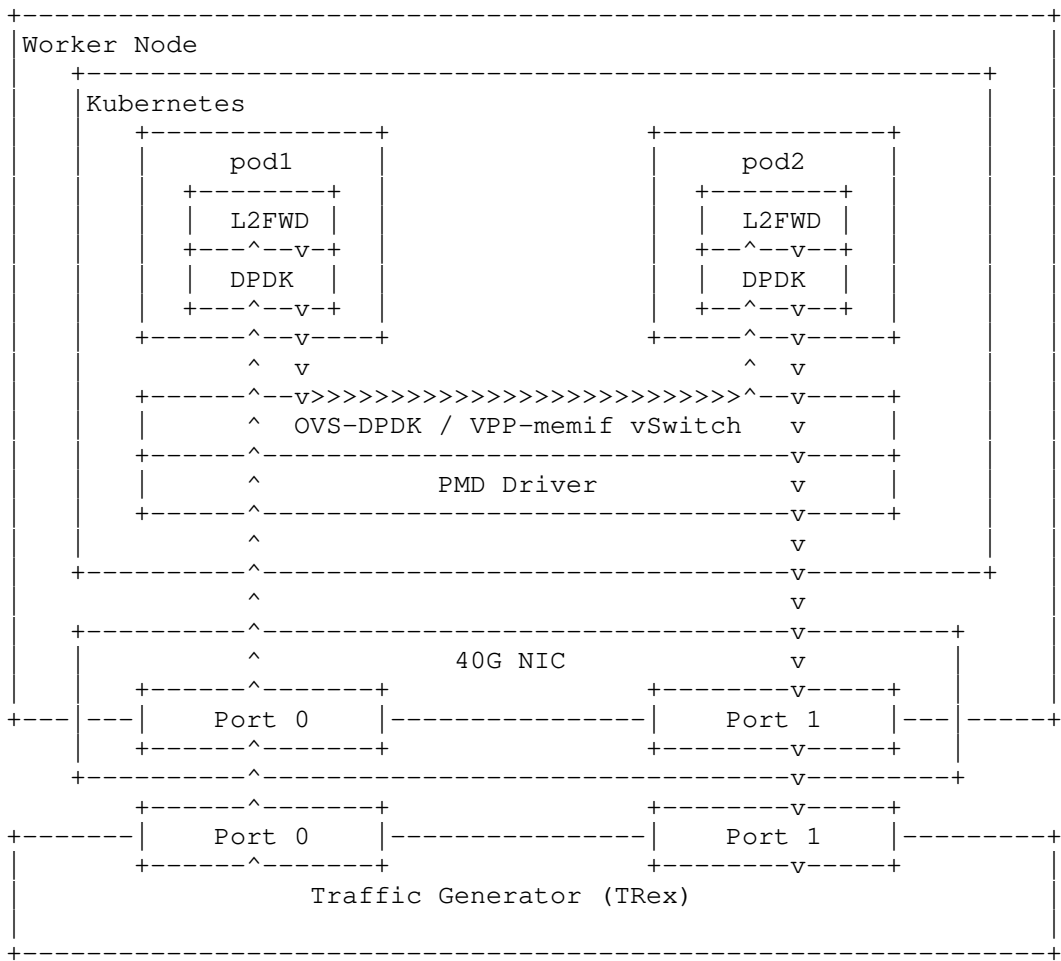


Figure 18: Multi-pod Benchmarking Scenario

C.2. Hardware Configurations

Node Name	Specification	Description
Conatiner Control for Master	<ul style="list-style-type: none"> - Intel(R) Core(TM) E5-2620v3 @ 2.40GHz (1socket x 12Cores) - MEM 32GB - DISK 1TB - NIC: Control plane: 1G - OS: CentOS Linux7.9 	<ul style="list-style-type: none"> - Container Deployment and Network Allocation - ubuntu 18.04 - Kubernetes Master - CNI Controller - MULTUS CNI - DPDK-OVS/VPP-memif
Conatiner Service for Worker	<ul style="list-style-type: none"> - Intel(R) Xeon(R) Gold 6148 @ 2.40GHz (2socket X 40Cores) - MEM 256GB - DISK 2TB - NIC - Control plane: 1G - Data plane: XL710-qda2 (1NIC 2PORT- 40Gb) - OS: CentOS Linux 7.9 	<ul style="list-style-type: none"> - Container dpdk-L2fwd - Kubernetes Worker - CNI Agent - Multus CNI - DPDK-OVS/VPP-memif
Packet Generator	<ul style="list-style-type: none"> - Intel(R) Xeon(R) Gold 6148 @ 2.4Ghz (2Socket X 40Core) - MEM 256GB - DISK 2TB - NIC - Data plane: XL710-qda2 (1NIC 2PORT - 40Gb) - OS: CentOS Lunix 7.9 	<ul style="list-style-type: none"> - Packet Generator - Installed Trex v2.92

Figure 19: Hardware Configurations for Multi-pod Benchmarking

For installations and configurations of CNIs, we used userspace-cni network plugin. Among this CNI, multus provides to create multiple interfaces for each pod. Both OVS-DPDK and VPP-memif bypass kernel with DPDK PMD driver. For CPU isolation and NUMA allocation, we used Intel CMK with exclusive mode. Since Trex generator is upgraded to the new version, we used the latest version of Trex.

C.3. NUMA Allocation Scenario

To analyze benchmarking impacts of different NUMA allocation, we set 6 scenarios depending on CPU location allocating to two pods and vSwich. For this scenario, we did not consider cross-NUMA case, which allocates CPUs to pod or switch in a manner that two cores are located in different NUMA nodes. 6 scenarios we considered are listed in Table 1. Note that, NIC is attached to the NUMA1.

Scenario #	vSwitich	pod1	pod2
S1	NUMA1	NUMA0	NUMA0
S2	NUMA1	NUMA1	NUMA1
S3	NUMA0	NUMA0	NUMA0
S4	NUMA0	NUMA1	NUMA1
S5	NUMA1	NUMA1	NUMA0
S6	NUMA0	NUMA0	NUMA1

Table 1: NUMA Allocation Scenarios

C.4. Traffic Generator Configurations

For multi-pod benchmarking, we discovered Non Drop Rate (NDR) with binary search algorithm. In Trex, it supports command to discover NDR for each testing. Also, we test for different ethernet frame sizes from 64bytes to 1518bytes. For running Trex, we used command as follows;

```
./ndr --stl --port 0 1 -v --profile stl/bench.py --prof-tun size=x --opt-bin-search
```

C.5. Benchmark Results and Trouble-shootings

As the benchmarking results, Table 2 shows packet loss ratio using 1518 bytes packet in OVS-DPDK/vpp-memif. From that result, we can say that the vpp-memif has better performance than OVS-DPDK, which is came from the difference in the way to forward packets between vswitch and pod. Also, the impact of NUMA is bigger when vswitch and both pods are located in the same node than when allocating CPU to the node where NIC is attached.

Networking Model	S1	S2	S3	S4	S5	S6
OVS-DPDK	21.29	13.17	6.32	19.76	12.43	6.38
vpp-memif	59.96	34.17	45.13	57.1	33.47	44.92

Table 2: Multi-pod Benchmarking Results (% of Line Rate)

Authors' Addresses

Kyoungjae Sun
ETRI
218, Gajeong-ro, Yuseung-gu
Dajeon
34065
Republic of Korea
Phone: +82 10 3643 5627
Email: kjsun@etri.re.kr

Hyunsik Yang
KT
KT Research Center 151
Taebong-ro, Seocho-gu
Seoul
06763
Republic of Korea
Phone: +82 10 9005 7439
Email: yangun@dcn.ssu.ac.kr

Jangwon Lee
Soongsil University
369, Sangdo-ro, Dongjak-gu
Seoul
06978
Republic of Korea
Phone: +82 10 7448 4664
Email: jangwon.lee@dcn.ssu.ac.kr

Tran Minh Ngoc
Soongsil University
369, Sangdo-ro, Dongjak-gu
Seoul
06978
Republic of Korea
Phone: +82 2 820 0841
Email: mipearlska1307@dcn.ssu.ac.kr

Younghan Kim
Soongsil University
369, Sangdo-ro, Dongjak-gu
Seoul
06978
Republic of Korea
Phone: +82 10 2691 0904
Email: younghak@ssu.ac.kr

Network Working Group
Internet-Draft
Updates: 2544 (if approved)
Intended status: Informational
Expires: May 21, 2020

A. Morton
AT&T Labs
November 18, 2019

Updates for the Back-to-back Frame Benchmark in RFC 2544
draft-ietf-bmwg-b2b-frame-01

Abstract

Fundamental Benchmarking Methodologies for Network Interconnect Devices of interest to the IETF are defined in RFC 2544. This memo updates the procedures of the test to measure the Back-to-back frames Benchmark of RFC 2544, based on further experience.

This memo updates Section 26.4 of RFC 2544.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14[RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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 May 21, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Scope and Goals	3
3. Motivation	4
4. Prerequisites	6
5. Back-to-back Frames	7
5.1. Preparing the list of Frame sizes	7
5.2. Test for a Single Frame Size	7
5.3. Test Repetition	8
5.4. Benchmark Calculations	8
6. Reporting	9
7. Security Considerations	10
8. IANA Considerations	11
9. Acknowledgements	11
10. References	11
10.1. Normative References	11
10.2. Informative References	12
Author's Address	13

1. Introduction

The IETF's fundamental Benchmarking Methodologies are defined in [RFC2544], supported by the terms and definitions in [RFC1242], and [RFC2544] actually obsoletes an earlier specification, [RFC1944]. Over time, the benchmarking community has updated [RFC2544] several times, including the Device Reset Benchmark [RFC6201], and the important Applicability Statement [RFC6815] concerning use outside the Isolated Test Environment (ITE) required for accurate benchmarking. Other specifications implicitly update [RFC2544], such as the IPv6 Benchmarking Methodologies in [RFC5180].

Recent testing experience with the Back-to-back Frame test and Benchmark in Section 26.4 of [RFC2544] indicates that an update is warranted [OPNFV-2017] [VSPERF-b2b]. In particular, analysis of the results indicates that buffers size matters when compensating for disruptions in the software packet processor, and this finding increases the importance of the Back-to-back frame characterization described here. This memo describes additional rationale and provides the updated method.

[RFC2544] provides its own Requirements Language consistent with [RFC2119], since [RFC1944] predates [RFC2119]. Thus, the requirements presented in this memo are expressed in [RFC2119] terms, and intended for those performing/reporting laboratory tests to improve clarity and repeatability, and for those designing devices that facilitate these tests.

2. Scope and Goals

The scope of this memo is to define an updated method to unambiguously perform tests, measure the benchmark(s), and report the results for Back-to-back Frames (presently described Section 26.4 of [RFC2544]).

The goal is to provide more efficient test procedures where possible, and to expand reporting with additional interpretation of the results. The tests described in this memo address the cases where the maximum frame rate of a single ingress port cannot be transferred to an egress port loss-free (for some frame sizes of interest).

[RFC2544] Benchmarks rely on test conditions with constant frame sizes, with the goal of understanding what network device capability has been tested. Tests with the smallest size stress the header processing capacity, and tests with the largest size stress the overall bit processing capacity. Tests with sizes in-between may determine the transition between these two capacities. However, conditions simultaneously sending multiple frame sizes, such as those described in [RFC6985], MUST NOT be used in Back-to-back Frame testing.

Section 3 of [RFC8239] describes buffer size testing for physical networking devices in a Data Center. The [RFC8239] methods measure buffer latency directly with traffic on multiple ingress ports that overload an egress port on the Device Under Test (DUT), and are not subject to the revised calculations presented in this memo. Likewise, the methods of [RFC8239] SHOULD be used for test cases where the egress port buffer is the known point of overload.

3. Motivation

Section 3.1 of [RFC1242] describes the rationale for the Back-to-back Frames Benchmark. To summarize, there are several reasons that devices on a network produce bursts of frames at the minimum allowed spacing, and it is therefore worthwhile to understand the Device Under Test (DUT) limit on the length of such bursts in practice. Also, [RFC1242] states:

"Tests of this parameter are intended to determine the extent of data buffering in the device."

After this test was defined, there have been occasional discussions of the stability and repeatability of the results, both over time and across labs. Fortunately, the Open Platform for Network Function Virtualization (OPNFV) VSPERF project's Continuous Integration (CI) testing routinely repeats Back-to-back Frame tests to verify that test functionality has been maintained through development of the test control programs. These tests were used as a basis to evaluate stability and repeatability, even across labset-ups when the test platform was migrated to new DUT hardware at the end of 2016.

When the VSPERF CI results were examined [VSPERF-b2b], several aspects of the results were considered notable:

1. Back-to-back Frame Benchmark was very consistent for some fixed frame sizes, and somewhat variable for others.
2. The number of Back-to-back Frames with zero loss reported for large frame sizes was unexpectedly long (translating to 30 seconds of buffer time), and no explanation or measurement limit condition was indicated.
3. Calculation of the extent of buffer time in the DUT helped to explain the results observed with all frame sizes (for example, some frame sizes cannot exceed the frame header processing rate of the DUT and therefore no buffering occurs, therefore the results depended on the test equipment and not the DUT).
4. It was found that the actual buffer time in the DUT could be estimated using results from the Throughput tests conducted according to Section 26.1 of [RFC2544]. It is apparent that the DUT's frame processing rate tends to increase the "implied" estimate (measured according to Section 26.4 of [RFC2544]), and a calculation using the Throughput measurement can reveal a "corrected" estimate.

Further, if the Throughput tests of Section 26.1 of [RFC2544] are conducted as a prerequisite test, the number of frame sizes required for Back-to-back Frame Benchmarking can be reduced to one or more of the small frame sizes, or the results for large frame sizes can be noted as invalid in the results if tested anyway (these are the frame sizes for which the back-to-back frame rate cannot exceed the frame header processing rate of the DUT and no buffering occurs).

[VSPERF-b2b] provides the details of the calculation to estimate the actual buffer storage available in the DUT, using results from the Throughput tests for each frame size, and the maximum theoretical frame rate for the DUT links (which constrain the minimum frame spacing).

The simplified model used in these calculations for the DUT includes a packet header processing function with limited rate of operation, as shown below:

```

|----- DUT -----|
Generator -> Ingress -> Buffer -> HeaderProc -> Egress -> Receiver
```

So, in the back2back frame testing:

1. The Ingress burst arrives at Max Theoretical Frame Rate, and initially the frames are buffered
2. The packet header processing function (HeaderProc) operates at approximately the "Measured Throughput", removing frames from the buffer
3. Frames that have been processed are clearly not in the buffer, so the Corrected DUT buffer time equation (Section 5.4) estimates and removes the frames that the DUT forwarded on Egress during the burst.

Knowledge of approximate buffer storage size (in time or bytes) may be useful to estimate whether frame losses will occur if DUT forwarding is temporarily suspended in a production deployment, due to an unexpected interruption of frame processing (an interruption of duration greater than the estimated buffer would certainly cause lost frames).

The presentation of OPNFV VSPERF evaluation and development of enhanced search algorithms [VSPERF-BSLV] was discussed at IETF-102. The enhancements are intended to compensate for transient interrupts that may cause loss at near-Throughput levels of offered load. Subsequent analysis of the results indicates that buffers within the DUT can compensate for some interrupts, and this finding increases

the importance of the Back-to-back frame characterization described here.

4. Prerequisites

The Test Setup MUST be consistent with Figure 1 of [RFC2544], or Figure 2 when the tester's sender and receiver are different devices. Other mandatory testing aspects described in [RFC2544] MUST be included, unless explicitly modified in the next section.

The ingress and egress link speeds and link layer protocols MUST be specified and used to compute the maximum theoretical frame rate when respecting the minimum inter-frame gap.

The test results for the Throughput Benchmark conducted according to Section 26.1 of [RFC2544] for all [RFC2544]-RECOMMENDED frame sizes MUST be available to reduce the tested frame size list, or to note invalid results for individual frame sizes (because the burst length may be essentially infinite for large frame sizes).

Note that:

- o the Throughput and the Back-to-back Frame measurement configuration traffic characteristics (unidirectional or bi-directional) MUST match.
- o the Throughput measurement MUST be under zero-loss conditions, according to Section 26.1 of [RFC2544].

The Back-to-back Benchmark described in Section 3.1 of [RFC1242] MUST be measured directly by the tester, where buffer size is inferred from packet loss measurements. Therefore, sources of packet loss that are un-related to consistent evaluation of buffer size SHOULD be identified and removed or mitigated. Example sources include:

- o On-path active components that are external to the DUT
- o Operating system environment interrupting DUT operation
- o Shared resource contention between the DUT and other off-path component(s), impacting DUT's behaviour, sometimes called the "noisy neighbour" problem.

Mitigations applicable to some of the sources above are discussed in Section 5.2, with the other measurement requirements described below in Section 5.

5. Back-to-back Frames

Objective: To characterize the ability of a DUT to process back-to-back frames as defined in [RFC1242].

The Procedure follows.

5.1. Preparing the list of Frame sizes

From the list of RECOMMENDED Frame sizes (Section 9 of [RFC2544]), select the subset of Frame sizes whose measured Throughput was less than the maximum theoretical Frame Rate. These are the only Frame sizes where it is possible to produce a burst of frames that cause the DUT buffers to fill and eventually overflow, producing one or more discarded frames.

5.2. Test for a Single Frame Size

Each trial in the test requires the tester to send a burst of frames (after idle time) with the minimum inter-frame gap, and to count the corresponding frames forwarded by the DUT.

The duration of the trial MUST be at least 2 seconds, to allow DUT buffers to deplete.

If all frames have been received, the tester increases the length of the burst according to the search algorithm and performs another trial.

If the received frame count is less than the number of frames in the burst, then the limit of DUT processing and buffering may have been exceeded, and the burst length is determined by the search algorithm for the next trial.

Classic search algorithms have been adapted for use in benchmarking, where the search requires discovery of a pair of outcomes, one with no loss and another with loss, at load conditions within the acceptable tolerance. Also for conditions encountered when benchmarking the Infrastructure for Network Function Virtualization require algorithm enhancement. Fortunately, the adaptation of Binary Search, and an enhanced Binary Search with Loss Verification have been specified in [TST009]. These algorithms (see clause 12.3) can easily be used for Back-to-back Frame benchmarking by replacing the Offered Load level with burst length in frames. [TST009] Annex B describes the theory behind the enhanced Binary Search algorithm.

There is also promising work-in-progress that may prove useful in for Back-to-back Frame benchmarking.

[I-D.vpolak-mkonstan-bmwg-mlrsearch] and [I-D.vpolak-bmwg-plrsearch] are two such examples.

Either the [TST009] Binary Search or Binary Search with Loss Verification algorithms MUST be used, and input parameters to the algorithm(s) MUST be reported.

The Back-to-back Frame value is the longest burst of frames that the DUT can successfully process and buffer without frame loss, as determined from the series of trials. The tester may impose a (configurable) minimum step size for burst length, and the step size MUST be reported with the results (as this influences the accuracy and variation of test results).

5.3. Test Repetition

The test MUST be repeated N times for each frame size in the subset list, and each Back-to-back Frame value made available for further processing (below).

5.4. Benchmark Calculations

For each Frame size, calculate the following summary statistics for Back-to-back Frame values over the N tests:

- o Average (Benchmark)
- o Minimum
- o Maximum
- o Standard Deviation

Further, calculate the Implied DUT Buffer Time and the Corrected DUT Buffer Time in seconds, as follows:

Implied DUT Buffer Time =

Average num of Back-to-back Frames / Max Theoretical Frame Rate

The formula above is simply expressing the Burst of Frames in units of time.

The next step is to apply a correction factor that accounts for the DUT's frame forwarding operation during the test (assuming the simple model of the DUT composed of a buffer and a forwarding function, described in Section 3).

Corrected DUT Buffer Time =

$$= \text{Implied DUT Buffer Time} * \frac{\text{Measured Throughput}}{\text{Max Theoretical Frame Rate}}$$

where:

1. The "Measured Throughput" is the [RFC2544] Throughput Benchmark for the frame size tested, as augmented by methods including the Binary Search with Loss Verification algorithm in [TST009] where applicable, and MUST be expressed in Frames per second in this equation.
2. The "Max Theoretical Frame Rate" is a calculated value for the interface speed and link layer technology used, and MUST be expressed in Frames per second in this equation.

The term on the far right in the formula for Corrected DUT Buffer Time accounts for all the frames in the Burst that were transmitted by the DUT *while the Burst of frames were sent in*. So, these frames are not in the Buffer and the Buffer size is more accurately estimated by excluding them.

6. Reporting

The back-to-back results SHOULD be reported in the format of a table with a row for each of the tested frame sizes. There SHOULD be columns for the frame size and for the resultant average frame count for each type of data stream tested.

The number of tests Averaged for the Benchmark, N, MUST be reported.

The Minimum, Maximum, and Standard Deviation across all complete tests SHOULD also be reported (they are referred to as "Min,Max,StdDev" in the table below).

The Corrected DUT Buffer Time SHOULD also be reported.

If the tester operates using a maximum burst length in frames, then this maximum length SHOULD be reported.

Frame Size, octets	Ave B2B Length, frames	Min,Max,StdDev	Corrected Buff Time, Sec
64	26000	25500,27000,20	0.00004

Back-to-Back Frame Results

Static and configuration parameters:

Number of test repetitions, N

Minimum Step Size (during searches), in frames.

If the tester has an actual frame rate of interest (less than the Throughput rate), it is useful to estimate the buffer time at that frame rate:

$$\begin{aligned} \text{Actual Buffer Time} = \\ = \text{Corrected DUT Buffer Time} * \frac{\text{Measured Throughput}}{\text{Actual Frame Rate}} \end{aligned}$$

and report this value, properly labeled.

7. Security Considerations

Benchmarking activities as described in this memo are limited to technology characterization using controlled stimuli in a laboratory environment, with dedicated address space and the other constraints of[RFC2544].

The benchmarking network topology will be an independent test setup and MUST NOT be connected to devices that may forward the test traffic into a production network, or misroute traffic to the test management network. See [RFC6815].

Further, benchmarking is performed on a "black-box" basis, relying solely on measurements observable external to the DUT/SUT.

Special capabilities SHOULD NOT exist in the DUT/SUT specifically for benchmarking purposes. Any implications for network security arising from the DUT/SUT SHOULD be identical in the lab and in production networks.

8. IANA Considerations

This memo makes no requests of IANA.

9. Acknowledgements

Thanks to Trevor Cooper, Sridhar Rao, and Martin Klokik of the VSPERF project for many contributions to the testing [VSPERF-b2b]. Yoshiaki Itou has also investigated the topic, and made useful suggestions. Maciek Konstantyowicz and Vratko Polak also provided many comments and suggestions based on extensive integration testing and resulting search algorithm proposals - the most up-to-date feedback possible. Tim Carlin also provided comments and support for the draft.

10. References

10.1. Normative References

- [RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <<https://www.rfc-editor.org/info/rfc1242>>.
- [RFC1944] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 1944, DOI 10.17487/RFC1944, May 1996, <<https://www.rfc-editor.org/info/rfc1944>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [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>>.
- [RFC5180] Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, DOI 10.17487/RFC5180, May 2008, <<https://www.rfc-editor.org/info/rfc5180>>.
- [RFC6201] Asati, R., Pignataro, C., Calabria, F., and C. Olvera, "Device Reset Characterization", RFC 6201, DOI 10.17487/RFC6201, March 2011, <<https://www.rfc-editor.org/info/rfc6201>>.

- [RFC6815] Bradner, S., Dubray, K., McQuaid, J., and A. Morton, "Applicability Statement for RFC 2544: Use on Production Networks Considered Harmful", RFC 6815, DOI 10.17487/RFC6815, November 2012, <<https://www.rfc-editor.org/info/rfc6815>>.
- [RFC6985] Morton, A., "IMIX Genome: Specification of Variable Packet Sizes for Additional Testing", RFC 6985, DOI 10.17487/RFC6985, July 2013, <<https://www.rfc-editor.org/info/rfc6985>>.
- [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>>.

10.2. Informative References

- [I-D.vpolak-bmwg-plrsearch]
Konstantynowicz, M. and V. Polak, "Probabilistic Loss Ratio Search for Packet Throughput (PLRsearch)", draft-vpolak-bmwg-plrsearch-02 (work in progress), July 2019.
- [I-D.vpolak-mkonstan-bmwg-mlrsearch]
Konstantynowicz, M. and V. Polak, "Multiple Loss Ratio Search for Packet Throughput (MLRsearch)", draft-vpolak-mkonstan-bmwg-mlrsearch-02 (work in progress), July 2019.
- [OPNFV-2017]
Cooper, T., Morton, A., and S. Rao, "Dataplane Performance, Capacity, and Benchmarking in OPNFV", June 2017, <<https://wiki.opnfv.org/download/attachments/10293193/VSPERF-Dataplane-Perf-Cap-Bench.pptx?api=v2>>.
- [RFC8239] Avramov, L. and J. Rapp, "Data Center Benchmarking Methodology", RFC 8239, DOI 10.17487/RFC8239, August 2017, <<https://www.rfc-editor.org/info/rfc8239>>.
- [TST009] Morton, R. A., "ETSI GS NFV-TST 009 V3.2.1 (2019-06), "Network Functions Virtualisation (NFV) Release 3; Testing; Specification of Networking Benchmarks and Measurement Methods for NFVI", June 2019, <https://www.etsi.org/deliver/etsi_gs/NFV-TST/001_099/009/03.01.01_60/gs_NFV-TST009v030101p.pdf>.

[VSPERF-b2b]

Morton, A., "Back2Back Testing Time Series (from CI)",
June 2017, <[https://wiki.opnfv.org/display/vsperf/
Traffic+Generator+Testing#TrafficGeneratorTesting-
AppendixB:Back2BackTestingTimeSeries\(fromCI\)](https://wiki.opnfv.org/display/vsperf/Traffic+Generator+Testing#TrafficGeneratorTesting-AppendixB:Back2BackTestingTimeSeries(fromCI))>.

[VSPERF-BSLV]

Morton, A. and S. Rao, "Evolution of Repeatability in
Benchmarking: Fraser Plugfest (Summary for IETF BMWG)",
July 2018,
<[https://datatracker.ietf.org/meeting/102/materials/
slides-102-bmwg-evolution-of-repeatability-in-
benchmarking-fraser-plugfest-summary-for-ietf-bmwg-00](https://datatracker.ietf.org/meeting/102/materials/slides-102-bmwg-evolution-of-repeatability-in-benchmarking-fraser-plugfest-summary-for-ietf-bmwg-00)>.

Author's Address

Al Morton
AT&T Labs
200 Laurel Avenue South
Middletown,, NJ 07748
USA

Phone: +1 732 420 1571
Fax: +1 732 368 1192
Email: acmorton@att.com

Network Working Group
Internet-Draft
Updates: 2544 (if approved)
Intended status: Informational
Expires: June 21, 2021

A. Morton
AT&T Labs
December 18, 2020

Updates for the Back-to-back Frame Benchmark in RFC 2544
draft-ietf-bmwg-b2b-frame-04

Abstract

Fundamental Benchmarking Methodologies for Network Interconnect Devices of interest to the IETF are defined in RFC 2544. This memo updates the procedures of the test to measure the Back-to-back frames Benchmark of RFC 2544, based on further experience.

This memo updates Section 26.4 of RFC 2544.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14[RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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 June 21, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Scope and Goals	3
3. Motivation	4
4. Prerequisites	6
5. Back-to-back Frames	7
5.1. Preparing the list of Frame sizes	7
5.2. Test for a Single Frame Size	8
5.3. Test Repetition and Benchmark	9
5.4. Benchmark Calculations	9
6. Reporting	11
7. Security Considerations	12
8. IANA Considerations	12
9. Acknowledgments	12
10. References	13
10.1. Normative References	13
10.2. Informative References	13
Author's Address	15

1. Introduction

The IETF's fundamental Benchmarking Methodologies are defined in [RFC2544], supported by the terms and definitions in [RFC1242], and [RFC2544] actually obsoletes an earlier specification, [RFC1944]. Over time, the benchmarking community has updated [RFC2544] several times, including the Device Reset Benchmark [RFC6201], and the important Applicability Statement [RFC6815] concerning use outside the Isolated Test Environment (ITE) required for accurate benchmarking. Other specifications implicitly update [RFC2544], such as the IPv6 Benchmarking Methodologies in [RFC5180].

Recent testing experience with the Back-to-back Frame test and Benchmark in Section 26.4 of [RFC2544] indicates that an update is warranted [OPNFV-2017] [VSPERF-b2b]. In particular, analysis of the results indicates that buffer size matters when compensating for interruptions of software packet processing, and this finding increases the importance of the Back-to-back frame characterization described here. This memo describes additional rationale and provides the updated method.

[RFC2544] (which obsoletes [RFC1944]) provides its own Requirements Language consistent with [RFC2119], since [RFC1944] pre-dates [RFC2119] and all three memos share common authorship. Today, [RFC8174] clarifies the usage of Requirements Language, so the requirements presented in this memo are expressed in [RFC8174] terms, and intended for those performing/reporting laboratory tests to improve clarity and repeatability, and for those designing devices that facilitate these tests.

2. Scope and Goals

The scope of this memo is to define an updated method to unambiguously perform tests, measure the benchmark(s), and report the results for Back-to-back Frames (presently described in Section 26.4 of [RFC2544]).

The goal is to provide more efficient test procedures where possible, and to expand reporting with additional interpretation of the results. The tests described in this memo address the cases in which the maximum frame rate of a single ingress port cannot be transferred loss-free to an egress port (for some frame sizes of interest).

[RFC2544] Benchmarks rely on test conditions with constant frame sizes, with the goal of understanding what network device capability has been tested. Tests with the smallest size stress the header processing capacity, and tests with the largest size stress the overall bit processing capacity. Tests with sizes in-between may determine the transition between these two capacities. However, conditions simultaneously sending a mixture of Internet frame sizes (IMIX), such as those described in [RFC6985], MUST NOT be used in Back-to-back Frame testing.

Section 3 of [RFC8239] describes buffer size testing for physical networking devices in a data center. The [RFC8239] methods measure buffer latency directly with traffic on multiple ingress ports that overload an egress port on the Device Under Test (DUT) and are not subject to the revised calculations presented in this memo. Likewise, the methods of [RFC8239] SHOULD be used for test cases where the egress port buffer is the known point of overload.

3. Motivation

Section 3.1 of [RFC1242] describes the rationale for the Back-to-back Frames Benchmark. To summarize, there are several reasons that devices on a network produce bursts of frames at the minimum allowed spacing; and it is, therefore, worthwhile to understand the Device Under Test (DUT) limit on the length of such bursts in practice. Also, [RFC1242] states:

"Tests of this parameter are intended to determine the extent of data buffering in the device."

After this test was defined, there have been occasional discussions of the stability and repeatability of the results, both over time and across labs. Fortunately, the Open Platform for Network Function Virtualization (OPNFV) VSPERF project's Continuous Integration (CI) [VSPERF-CI] testing routinely repeats Back-to-back Frame tests to verify that test functionality has been maintained through development of the test control programs. These tests were used as a basis to evaluate stability and repeatability, even across lab set-ups when the test platform was migrated to new DUT hardware at the end of 2016.

When the VSPERF CI results were examined [VSPERF-b2b], several aspects of the results were considered notable:

1. Back-to-back Frame Benchmark was very consistent for some fixed frame sizes, and somewhat variable for other frame sizes.
2. The number of Back-to-back Frames with zero loss reported for large frame sizes was unexpectedly long (translating to 30 seconds of buffer time), and no explanation or measurement limit condition was indicated. It was important that the buffering time calculations were part of the referenced testing and analysis[VSPERF-b2b], because the calculated buffer times of 30 seconds for some frame sizes were clearly wrong or highly suspect. On the other hand, a result expressed only as a large number of Back-to-back Frames does not permit such an easy comparison with reality.
3. Calculation of the extent of buffer time in the DUT helped to explain the results observed with all frame sizes (for example, tests with some frame sizes cannot exceed the frame header processing rate of the DUT and thus no buffering occurs; therefore, the results depended on the test equipment and not the DUT).

4. It was found that a better estimate of the DUT buffer time could be calculated using measurements of both the longest burst in frames without loss and results from the Throughput tests conducted according to Section 26.1 of [RFC2544]. It is apparent that the DUT's frame processing rate empties the buffer during a trial and tends to increase the "implied" buffer size estimate (measured according to Section 26.4 of [RFC2544] because many frames have departed the buffer when the burst of frames ends). A calculation using the Throughput measurement can reveal a "corrected" buffer size estimate.

Further, if the Throughput tests of Section 26.1 of [RFC2544] are conducted as a prerequisite test, the number of frame sizes required for Back-to-back Frame Benchmarking can be reduced to one or more of the small frame sizes, or the results for large frame sizes can be noted as invalid in the results if tested anyway (these are the larger frame sizes for which the back-to-back frame rate cannot exceed the frame header processing rate of the DUT and little or no buffering occurs).

The material below provides the details of the calculation to estimate the actual buffer storage available in the DUT, using results from the Throughput tests for each frame size, and the maximum theoretical frame rate for the DUT links (which constrain the minimum frame spacing).

In reality, there are many buffers and packet header processing steps in a typical DUT. The simplified model used in these calculations for the DUT includes a packet header processing function with limited rate of operation, as shown below:

|----- DUT -----|
Generator -> Ingress -> Buffer -> HeaderProc -> Egress -> Receiver

So, in the Back-to-back Frame testing:

1. The ingress burst arrives at Max Theoretical Frame Rate, and initially the frames are buffered.
2. The packet header processing function (HeaderProc) operates at the "Measured Throughput" (Section 26.1 of [RFC2544]), removing frames from the buffer (this is the best approximation we have).
3. Frames that have been processed are clearly not in the buffer, so the Corrected DUT buffer time equation (Section 5.4) estimates and removes the frames that the DUT forwarded on egress during the burst. We define buffer time as the number of frames

occupying the buffer divided by the Maximum Theoretical Frame Rate (on ingress) for the frame size under test.

4. A helpful concept is the buffer filling rate, which is the difference between the Max Theoretical Frame Rate (ingress) and the Measured Throughput (HeaderProc on egress). If the actual buffer size in frames was known, the time to fill the buffer during a measurement can be calculated using the filling rate as a check on measurements. However, the buffer in the model represents many buffers of different sizes in the DUT data path.

Knowledge of approximate buffer storage size (in time or bytes) may be useful to estimate whether frame losses will occur if DUT forwarding is temporarily suspended in a production deployment, due to an unexpected interruption of frame processing (an interruption of duration greater than the estimated buffer would certainly cause lost frames). In Section 5, the calculations for the correct buffer time use the combination of offered load at Max Theoretical Frame Rate and header processing speed at 100% of Measured Throughput. Other combinations are possible, such as changing the percent of measured Throughput to account for other processes reducing the header processing rate.

The presentation of OPNFV VSPERF evaluation and development of enhanced search algorithms [VSPERF-BSLV] was discussed at IETF-102. The enhancements are intended to compensate for transient interrupts that may cause loss at near-Throughput levels of offered load. Subsequent analysis of the results indicates that buffers within the DUT can compensate for some interrupts, and this finding increases the importance of the Back-to-back frame characterization described here.

4. Prerequisites

The Test Setup MUST be consistent with Figure 1 of [RFC2544], or Figure 2 when the tester's sender and receiver are different devices. Other mandatory testing aspects described in [RFC2544] MUST be included, unless explicitly modified in the next section.

The ingress and egress link speeds and link layer protocols MUST be specified and used to compute the maximum theoretical frame rate when respecting the minimum inter-frame gap.

The test results for the Throughput Benchmark conducted according to Section 26.1 of [RFC2544] for all [RFC2544]-RECOMMENDED frame sizes MUST be available to reduce the tested frame size list, or to note invalid results for individual frame sizes (because the burst length may be essentially infinite for large frame sizes).

Note that:

- o the Throughput and the Back-to-back Frame measurement configuration traffic characteristics (unidirectional or bi-directional, and number of flows generated) MUST match.
- o the Throughput measurement MUST be under zero-loss conditions, according to Section 26.1 of [RFC2544].

The Back-to-back Benchmark described in Section 3.1 of [RFC1242] MUST be measured directly by the tester, where buffer size is inferred from Back-to-back Frame bursts and associated packet loss measurements. Therefore, sources of packet loss that are unrelated to consistent evaluation of buffer size SHOULD be identified and removed or mitigated. Example sources include:

- o On-path active components that are external to the DUT
- o Operating system environment interrupting DUT operation
- o Shared resource contention between the DUT and other off-path component(s) impacting DUT's behaviour, sometimes called the "noisy neighbour" problem with virtualized network functions.

Mitigations applicable to some of the sources above are discussed in Section 5.2, with the other measurement requirements described below in Section 5.

5. Back-to-back Frames

Objective: To characterize the ability of a DUT to process back-to-back frames as defined in [RFC1242].

The Procedure follows.

5.1. Preparing the list of Frame sizes

From the list of RECOMMENDED frame sizes (Section 9 of [RFC2544]), select the subset of frame sizes whose measured Throughput (during prerequisite testing) was less than the Maximum Theoretical Frame Rate of the DUT/test-set-up. These are the only frame sizes where it is possible to produce a burst of frames that cause the DUT buffers to fill and eventually overflow, producing one or more discarded frames.

5.2. Test for a Single Frame Size

Each trial in the test requires the tester to send a burst of frames (after idle time) with the minimum inter-frame gap, and to count the corresponding frames forwarded by the DUT.

The duration of the trial includes three REQUIRED components:

1. The time to send the burst of frames (at the back-to-back rate), determined by the search algorithm.
2. The time to receive the transferred burst of frames (at the [RFC2544] Throughput rate), possibly truncated by buffer overflow, and certainly including the latency of the DUT.
3. At least 2 seconds not overlapping the time to receive the burst (2.), to ensure that DUT buffers have depleted. Longer times MUST be used when conditions warrant, such as when buffer times >2 seconds are measured or when burst sending times are >2 seconds, but care is needed since this time component directly increases trial duration and many trials and tests comprise a complete benchmarking study.

The upper search limit for the time to send each burst MUST be configurable, to values as high as 30 seconds (buffer time results reported at or near the configured upper limit are likely invalid, and the test MUST be repeated with a higher search limit).

If all frames have been received, the tester increases the length of the burst according to the search algorithm and performs another trial.

If the received frame count is less than the number of frames in the burst, then the limit of DUT processing and buffering may have been exceeded, and the burst length is determined by the search algorithm for the next trial (the burst length is typically reduced, but see below).

Classic search algorithms have been adapted for use in benchmarking, where the search requires discovery of a pair of outcomes, one with no loss and another with loss, at load conditions within the acceptable tolerance or accuracy. Conditions encountered when benchmarking the Infrastructure for Network Function Virtualization require algorithm enhancement. Fortunately, the adaptation of Binary Search, and an enhanced Binary Search with Loss Verification have been specified in clause 12.3 of [TST009]. These algorithms can easily be used for Back-to-back Frame benchmarking by replacing the Offered Load level with burst length in frames. [TST009] Annex B

describes the theory behind the enhanced Binary Search with Loss Verification algorithm.

There is also promising work-in-progress that may prove useful in Back-to-back Frame benchmarking.
[I-D.vpolak-mkonstan-bmwg-mlrsearch] and [I-D.vpolak-bmwg-plrsearch] are two such examples.

Either the [TST009] Binary Search or Binary Search with Loss Verification algorithms MUST be used, and input parameters to the algorithm(s) MUST be reported.

The tester usually imposes a (configurable) minimum step size for burst length, and the step size MUST be reported with the results (as this influences the accuracy and variation of test results).

The original Section 26.4 of [RFC2544] definition is stated below:

The Back-to-back Frame value is the longest burst of frames that the DUT can successfully process and buffer without frame loss, as determined from the series of trials.

5.3. Test Repetition and Benchmark

On this topic, Section 26.4 of [RFC2544] requires:

The trial length MUST be at least 2 seconds and SHOULD be repeated at least 50 times with the average of the recorded values being reported.

Therefore, the Back-to-back Frame Benchmark is the average of burst length values over repeated tests to determine the longest burst of frames that the DUT can successfully process and buffer without frame loss. Each of the repeated tests completes an independent search process.

In this update, the test MUST be repeated N times (the number of repetitions is now a variable that must be reported), for each frame size in the subset list, and each Back-to-back Frame value made available for further processing (below).

5.4. Benchmark Calculations

For each frame size, calculate the following summary statistics for longest Back-to-back Frame values over the N tests:

- o Average (Benchmark)

- o Minimum
- o Maximum
- o Standard Deviation

Further, calculate the Implied DUT Buffer Time and the Corrected DUT Buffer Time in seconds, as follows:

Implied DUT Buffer Time =

Average num of Back-to-back Frames / Max Theoretical Frame Rate

The formula above is simply expressing the burst of frames in units of time.

The next step is to apply a correction factor that accounts for the DUT's frame forwarding operation during the test (assuming the simple model of the DUT composed of a buffer and a forwarding function, described in Section 3).

Corrected DUT Buffer Time =

$$= \text{Implied DUT Buffer Time} - \left(\text{Implied DUT Buffer Time} * \frac{\text{Measured Throughput}}{\text{Max Theoretical Frame Rate}} \right)$$

where:

1. The "Measured Throughput" is the [RFC2544] Throughput Benchmark for the frame size tested, as augmented by methods including the Binary Search with Loss Verification algorithm in [TST009] where applicable, and MUST be expressed in frames per second in this equation.
2. The "Max Theoretical Frame Rate" is a calculated value for the interface speed and link layer technology used, and MUST be expressed in frames per second in this equation.

The term on the far right in the formula for Corrected DUT Buffer Time accounts for all the frames in the Burst that were transmitted by the DUT *while the Burst of frames were sent in*. So, these frames are not in the buffer and the buffer size is more accurately estimated by excluding them.

6. Reporting

The back-to-back frame results SHOULD be reported in the format of a table with a row for each of the tested frame sizes. There SHOULD be columns for the frame size and for the resultant average frame count for each type of data stream tested.

The number of tests Averaged for the Benchmark, N, MUST be reported.

The Minimum, Maximum, and Standard Deviation across all complete tests SHOULD also be reported (they are referred to as "Min,Max,StdDev" in the table below).

The Corrected DUT Buffer Time SHOULD also be reported.

If the tester operates using a limited maximum burst length in frames, then this maximum length SHOULD be reported.

Frame Size, octets	Ave B2B Length, frames	Min,Max,StdDev	Corrected Buff Time, Sec
64	26000	25500,27000,20	0.00004

Back-to-Back Frame Results

Static and configuration parameters (reported with the table above):

Number of test repetitions, N

Minimum Step Size (during searches), in frames.

If the tester has a specific (actual) frame rate of interest (less than the Throughput rate), it is useful to estimate the buffer time at that actual frame rate:

$$\begin{aligned} \text{Actual Buffer Time} &= \\ &= \text{Corrected DUT Buffer Time} * \frac{\text{Max Theoretical Frame Rate}}{\text{Actual Frame Rate}} \end{aligned}$$

and report this value, properly labeled.

7. Security Considerations

Benchmarking activities as described in this memo are limited to technology characterization using controlled stimuli in a laboratory environment, with dedicated address space and the other constraints of[RFC2544].

The benchmarking network topology will be an independent test setup and MUST NOT be connected to devices that may forward the test traffic into a production network, or misroute traffic to the test management network. See [RFC6815].

Further, benchmarking is performed on an "opaque-box" (a.k.a. "black-box") basis, relying solely on measurements observable external to the DUT/SUT.

The DUT developers are commonly independent from the personnel and institutions conducting benchmarking studies. DUT developers might have incentives to alter the performance of the DUT if the test conditions can be detected. Special capabilities SHOULD NOT exist in the DUT/SUT specifically for benchmarking purposes. Procedures described in this document are not designed to detect such activity. Additional testing outside of the scope of this document would be needed and has been used successfully in the past to discover such malpractices.

Any implications for network security arising from the DUT/SUT SHOULD be identical in the lab and in production networks.

8. IANA Considerations

This memo makes no requests of IANA.

9. Acknowledgements

Thanks to Trevor Cooper, Sridhar Rao, and Martin Klozik of the VSPERF project for many contributions to the early testing [VSPERF-b2b]. Yoshiaki Itou has also investigated the topic, and made useful suggestions. Maciek Konstantyowicz and Vratko Polak also provided many comments and suggestions based on extensive integration testing and resulting search algorithm proposals - the most up-to-date feedback possible. Tim Carlin also provided comments and support for the draft. Warren Kumari's review improved readability in several key passages. David Black, Martin Duke, and Scott Bradner's comments improved the clarity and configuration advice on trial duration. Malisa Vucinic suggested additional text on DUT design cautions in the Security Considerations section.

10. References

10.1. Normative References

- [RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <<https://www.rfc-editor.org/info/rfc1242>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [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>>.
- [RFC6985] Morton, A., "IMIX Genome: Specification of Variable Packet Sizes for Additional Testing", RFC 6985, DOI 10.17487/RFC6985, July 2013, <<https://www.rfc-editor.org/info/rfc6985>>.
- [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>>.
- [RFC8239] Avramov, L. and J. Rapp, "Data Center Benchmarking Methodology", RFC 8239, DOI 10.17487/RFC8239, August 2017, <<https://www.rfc-editor.org/info/rfc8239>>.
- [TST009] Morton, A., "ETSI GS NFV-TST 009 V3.4.1 (2020-12), "Network Functions Virtualisation (NFV) Release 3; Testing; Specification of Networking Benchmarks and Measurement Methods for NFVI"", December 2020, <https://www.etsi.org/deliver/etsi_gs/NFV-TST/001_099/009/03.04.01_60/gs_NFV-TST009v030401p.pdf>.

10.2. Informative References

- [I-D.vpolak-bmwg-plrsearch] Konstantynowicz, M. and V. Polak, "Probabilistic Loss Ratio Search for Packet Throughput (PLRsearch)", draft-vpolak-bmwg-plrsearch-03 (work in progress), March 2020.

- [I-D.vpolak-mkonstan-bmwg-mlrsearch]
Konstantynowicz, M. and V. Polak, "Multiple Loss Ratio Search for Packet Throughput (MLRsearch)", draft-vpolak-mkonstan-bmwg-mlrsearch-03 (work in progress), March 2020.
- [OPNFV-2017]
Cooper, T., Morton, A., and S. Rao, "Dataplane Performance, Capacity, and Benchmarking in OPNFV", June 2017,
<<https://wiki.opnfv.org/download/attachments/10293193/VSPERF-Dataplane-Perf-Cap-Bench.pptx?api=v2>>.
- [RFC1944] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 1944, DOI 10.17487/RFC1944, May 1996,
<<https://www.rfc-editor.org/info/rfc1944>>.
- [RFC5180] Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, DOI 10.17487/RFC5180, May 2008, <<https://www.rfc-editor.org/info/rfc5180>>.
- [RFC6201] Asati, R., Pignataro, C., Calabria, F., and C. Olvera, "Device Reset Characterization", RFC 6201, DOI 10.17487/RFC6201, March 2011,
<<https://www.rfc-editor.org/info/rfc6201>>.
- [RFC6815] Bradner, S., Dubray, K., McQuaid, J., and A. Morton, "Applicability Statement for RFC 2544: Use on Production Networks Considered Harmful", RFC 6815, DOI 10.17487/RFC6815, November 2012,
<<https://www.rfc-editor.org/info/rfc6815>>.
- [VSPERF-b2b]
Morton, A., "Back2Back Testing Time Series (from CI)", June 2017, <[https://wiki.opnfv.org/display/vsperf/Traffic+Generator+Testing#TrafficGeneratorTesting-AppendixB:Back2BackTestingTimeSeries\(fromCI\)](https://wiki.opnfv.org/display/vsperf/Traffic+Generator+Testing#TrafficGeneratorTesting-AppendixB:Back2BackTestingTimeSeries(fromCI))>.
- [VSPERF-BSLV]
Morton, A. and S. Rao, "Evolution of Repeatability in Benchmarking: Fraser Plugfest (Summary for IETF BMWG)", July 2018,
<<https://datatracker.ietf.org/meeting/102/materials/slides-102-bmwg-evolution-of-repeatability-in-benchmarking-fraser-plugfest-summary-for-ietf-bmwg-00>>.

[VSPERF-CI]

Tahhan, M., "OPNFV VSPERF CI", June 2019,
<<https://wiki.opnfv.org/display/vsperf/VSPERF+CI>>.

Author's Address

Al Morton
AT&T Labs
200 Laurel Avenue South
Middletown,, NJ 07748
USA

Phone: +1 732 420 1571
Fax: +1 732 368 1192
Email: acmorton@att.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: February 22, 2020

S. Jacob, Ed.
K. Tiruveedhula
Juniper Networks
August 21, 2019

Benchmarking Methodology for EVPN and PBB-EVPN
draft-ietf-bmwg-evpntest-03

Abstract

This document defines methodologies for benchmarking EVPN and PBB-EVPN performance. EVPN is defined in RFC 7432, and is being deployed in Service Provider networks. Specifically this document defines the methodologies for benchmarking EVPN/PBB-EVPN convergence, data plane performance, and control plane performance.

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 February 22, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Language	3
1.2. Terminologies	3
2. Test Topology	3
3. Test Cases for EVPN Benchmarking	6
3.1. Local MAC Learning	6
3.2. Remote MAC Learning	7
3.3. MAC Flush due to local link failure and Relearning	7
3.4. MAC Flush due to remote link failure.	8
3.5. MAC Aging	8
3.6. Remote Mac Aging	9
3.7. Local and Remote MAC Learning	9
3.8. High Availability.	10
3.9. ARP/ND Scale	11
3.10. Scaling of Services	11
3.11. Scale Convergence	12
3.12. SOAK Test.	12
4. Test Cases for PBB-EVPN Benchmarking	13
4.1. Local MAC Learning	13
4.2. Remote Mac Learning	13
4.3. MAC Flush due to link failure	14
4.4. MAC Flush due to remote Failure	14
4.5. MAC aging	15
4.6. Remote MAC Aging.	16
4.7. Local and Remote MAC Learning	16
4.8. High Availability	17
4.9. Scale	17
4.10. Scale Convergence	18
4.11. Soak Test	18
5. Acknowledgements	19
6. IANA Considerations	19
7. Security Considerations	19
8. References	19
8.1. Normative References	19
8.2. Informative References	20
Appendix A. Appendix	20
Authors' Addresses	20

1. Introduction

EVPN is defined in RFC 7432, and describes BGP MPLS- based Ethernet VPNs (EVPN). PBB-EVPN is defined in RFC 7623, discusses how Ethernet Provider backbone Bridging can be combined with EVPNs to provide a new/combined solution. This draft defines methodologies that can be used to benchmark both RFC 7432 and RFC 7623 solutions. Further, this draft provides methodologies for benchmarking the performance of

EVPN data and control planes, MAC learning, MAC flushing, MAC ageing, convergence, high availability, and scale.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2. Terminologies

MHPE Multi homed Provide Edge router.

RR Route Reflector.

P Provider Router.

CE Customer Router/Devices/Switch.

MHPE2 Multi homed Provider Edge router 2.

MHPE1 Multi homed Provider Edge router 1.

SHPE3 Single homed Provider Edge Router 3.

AA EVPN Terminologies AA All-Active.

SA EVPN Terminologies SA Single-Active.

RT Router Tester.

Sub Interface Each physical Interfaces is subdivided in to Logical units.

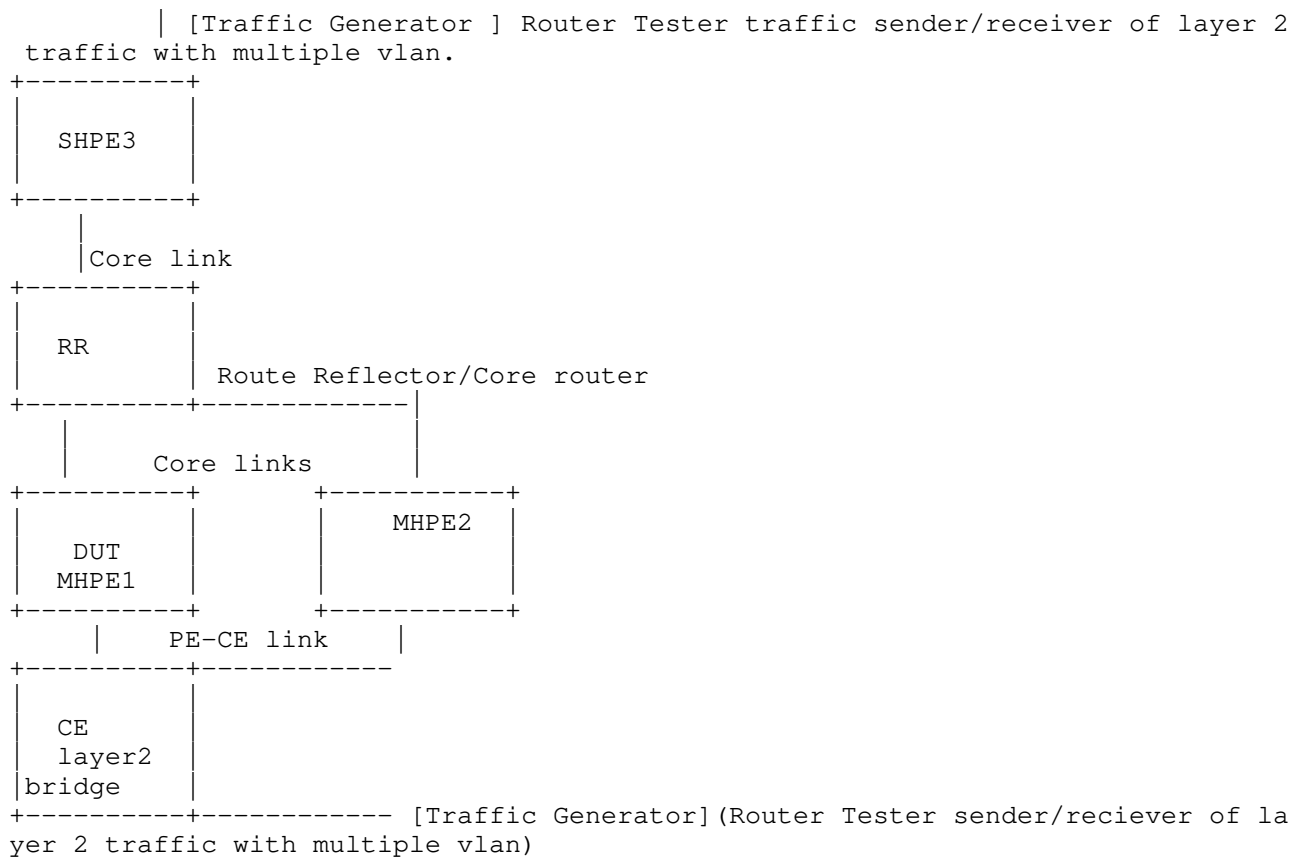
EVI EVPN Instances which will be running on sub interface or physical port of the provider Edge routers.

DF Designated Forwarder.

ESI Ethernet Segment Identifier.

2. Test Topology

EVPN/PBB-EVPN Services running on SHPE3, MHPE1 and MHPE2 in Single Active Mode:



Topology 1

Test Setup

Figure 1

Mode	Receiver	Test	Traffic Direction	Sender
Single Active	SHPE3	Local Mac Learning	Layer 2 traffic Uni multiple MAC	CE
Single Active CE	Remote MAC Learning	Layer 2 traffic uni multiple MAC	SHPE3	
Single Active CE/SHPE3	Scale Convergence	Bi	Layer 2 traffic Local & Remote Learning multiple mac & vlans	CE/SHPE3

++

Table showing Traffic pattern for various test

Figure 2

Test Setup Configurations:

There are five routers in the Test setup. SHPE3, RR/P, MHPE1 and MHPE2 emulating a service provider network. CE is a customer device connected to MHPE1 and MHPE2, it is configured with bridge domains in multiple vlans. The router tester is connected to CE and SHPE3. The MHPE1 acts as DUT. The RT will be used as sender and receiver of traffic. The measurement will be taken in DUT.

All routers except CE is configured with OSPF/IS-IS,LDP,MPLS,BGP with EVPN address family.

All routers except CE must have IBGP configured with RR acting as route reflector.

MHPE1,MHPE2,SHPE3 must be configured with "N" EVPN/PBB-EVPN instances depends up on the cases.

MHPE1 and MHEPE2 must be configured with ESI per vlan or ESI on IFD.

MHPE1 and MHEPE2 are running Single Active mode of EVPN.

CE is acting as bridge configured with vlans that is configured on MHPE1,MHPE2,SHPE3.

Depends up on the test traffic will be flowing uni directional or bi directional depends on the test performed.

The above configuration will be serving as the base configuration for all test cases.

3. Test Cases for EVPN Benchmarking

3.1. Local MAC Learning

Objective:

To Record the time taken to learn the MAC address locally in DUT.

Topology : Topology 1

Procedure:

The data plane MAC learning can be measured using the parameters defined in RFC 2889 section 5.8. Send "X" unicast frames from CE to MHPE1(DUT) working in SA mode with "X" different source and destination address from RT. The DUT must learn these "X" macs in data plane.

Measurement :

Measure the time taken to learn "X" MACs in DUT evpn mac table. The data plane measurement is taken by considering DUT as black box the range of X MAC is known from RT and the same must be learned in DUT, the time taken to learn "X" macs is measured.

Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The mac learning time is calculated by averaging the values obtained from "N" samples.

Mac learning in sec = $(T1+T2+..Tn/N)$

3.2. Remote MAC Learning

Objective:

To Record the time taken to learn the remote macs.

Topology : Topology 1

Procedure:

Send X frames with X different SA and DA to SHPE3 from RT. SHPE3 will advertise these locally learned macs to MHPE1 and MHPE2 via control plane. Measure the time taken to learn these X MACs from remote peer in DUT EVPN MAC address table. The DUT and MHPE2 are running SA mode.

Measurement :

Measure the time taken by the DUT to learn the "X" MACs in the data plane. Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The mac learning time is calculated by averaging the values obtained from "N" samples.

Mac learning in sec = $(T1+T2+..Tn/N)$

3.3. MAC Flush due to local link failure and Relearning

Objective:

To record the time taken to flush the mac learned locally and the time taken to relearn the same amount of macs.

Topology : Topology 1

Procedure:

Send X frames with X different SA and DA to DUT from CE using traffic generator. Wait till the MHPE1 learns all X MAC address. Then fail the MHPE1 CE link and measure the time taken to flush these X MACs from the EVPN MAC table. Bring up the link which was made Down (the

link between MHPE1 and CE). Measure time taken to relearn it. The DUT and MHPE2 are running SA mode.

Measurement :

Measure the time taken for flushing these X MAC address. Measure the time taken to relearn the X MACs in DUT. Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The flush and the relearning time is calculated by averaging the values obtained by "N" samples.

Flush time for X Macs in sec = $(T1+T2+..Tn/N)$

Relearning time for X macs in sec = $(T1+T2+..Tn/N)$

3.4. MAC Flush due to remote link failure.

Objective:

To record the time taken to flush the remote mac learned in DUT during remote link failure.

Topology : Topology 1

Procedure:

Send X frames with X different SA and DA to DUT from SHPE3 using traffic generator. Bring down the link between SHPE3 and traffic generator. Then measure the time taken to flush the DUT EVPN MAC table. The DUT and MHPE2 are running SA mode.

Measurement :

Measure the time taken to flush X remote MACs from EVPN MAC table of DUT. Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The flush rate is calculated averaging the values obtained by "N" samples.

Flush time for X Macs in sec = $(T1+T2+..Tn/N)$

3.5. MAC Aging

Objective:

To measure the mac aging time.

Topology : Topology 1

Procedure:

Send X frames with X different SA and DA to DUT from CE using traffic generator. Wait till X MAC address are learned. Then stop the traffic. Record the time taken to flush X MACS from DUT EVPN MAC table due to aging. The DUT and MHPE2 are running SA mode

Measurement :

Measure the time taken to flush X MAC address due to aging. Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The aging is calculated averaging the values obtained by "N" samples.

Aging time for X Macs in sec = $(T1+T2+..Tn/N)$

3.6. Remote Mac Aging**Objective:**

To measure the remote mac aging time.

Topology : Topology 1

Procedure:

Send X frames with X different SA and DA to DUT from SHPE3 using traffic generator. Stop the traffic at remote PE SHPE3. Due to MAC aging SHPE3 will withdraw its routes from DUT and MHPE2. Measure the time taken to remove these MACs from DUT EVPN MAC table. DUT and MHPE2 are running in SA mode

Measurement :

Measure the time taken to flush X remote MACs learned in DUT EVPN MAC table due to aging. Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The aging is calculated by averaging the values obtained by "N" samples.

Aging time for X Macs in sec = $(T1+T2+..Tn/N)$

3.7. Local and Remote MAC Learning**Objective:**

To record the time taken to learn both local and remote macs.

Topology : Topology 1

Procedure:

Send X frames with X different SA and DA to DUT from SHPE3 using traffic generator. Send X frames with different SA and DA from traffic generator connected to CE. The SA and DA of flows must be complimentary to have unicast flows. Measure the time taken by the DUT to learn 2X in EVPN MAC. DUT and MHPE2 are running in SA mode.

Measurement :

Measure the time taken to learn 2X MAC address in DUT EVPN MAC table. Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The mac learning time is calculated by averaging the values obtained by "N" samples.

Time to learn 2X Macs in sec = $(T1+T2+..Tn/N)$

3.8. High Availability.

Objective:

To record traffic loss during routing engine fail over.

Topology : Topology 1

Procedure:

Send X frames from CE to DUT from traffic generator with X different SA and DA. Send X frames from traffic generator to SHPE3 with X different SA and DA so that 2X MAC address will be learned in DUT. There is a bi directional traffic flow with X pps in each direction. Then do a routing engine fail-over.

Measurement :

There should be 0 traffic loss which is the ideal case, No change in the DF role. DUT should not withdraw any routes. Repeat the test "N" times and plot the data. The packet loss is calculated by averaging the values obtained from "N" samples.

Packet loss in sec = $(T1+T2+..Tn/N)$

3.9. ARP/ND Scale

These tests are conducted to Record the scaling parameter of ARP/ND of the DUT.

Objective:

To Record the ARP/ND scale of the DUT.

Topology : Topology 1

Procedure:

Send X arp/icmpv6 request from RT to DUT with different sender ip/ipv6 address to the same target gateway ip address. Measure whether X MAC+IPv4 address/MAC+IPv6 address of the hosts are learned in DUT.

Measurement :

The DUT must learn X MAC+IPv4/MAC+IPv6 and it must advertise the X MAC+IPv4/MAC+IPv6 to the remote router.

3.10. Scaling of Services

Objective:

To measure the scale limit of DUT for EVPN. This is to measure the performance of DUT in scaling to "X" EVPN instances.

Topology : Topology 1

Procedure:

The DUT, MHPE2 and SHPE3 are scaled to "N" EVI. Clear BGP neighbors of the DUT. Once adjacency is established in the DUT. Measure the routes received from MHPE2 and SHPE3 for "N" EVI in the DUT.

Measurement :

There should not be any loss of route types 1,2,3 and 4 in DUT. DUT must relearn all type 1,2,3 and 4 from remote routers. The DUT must be subjected to various values of N to find the optimal scale limit

3.11. Scale Convergence

Objective:

To measure the convergence time of DUT when the DUT is scaled with EVPN instance along with traffic.

Topology : Topology 1

Procedure:

Scale N EVIs in DUT, SHPE3 and MHPE2. Send F frames to DUT from CE using traffic generator with X different SA and DA for N EVI's. Send F frames from traffic generator to SHPE3 with X different SA and DA. There will be 2X number of MAC address will be learned in DUT EVPN MAC table. There is a bi directional traffic flow with F pps in each direction. Then clear the BGP neighbors in the DUT. Once the adjacency is restored in DUT. Measure the time taken to learn 2X MAC address in DUT MAC table.

Measurement :

The DUT must learn 2X MAC address. Measure the time taken to learn 2X MAC in DUT. Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The convergence time is calculated by averaging the values obtained by "N" samples.

Convergence time in sec = $(T1+T2+..Tn/N)$

3.12. SOAK Test.

Objective:

This test is carried out to measure the stability of the DUT in a scaled environment with traffic over a period of time "T' ". In each interval "t1" the DUT CPU usage, memory usage are measured. The DUT is checked for any crashes during this time period.

Topology : Topology 1

Procedure:

Scale N EVI's in DUT, SHPE3 and MHPE2. Send F frames to DUT from CE using traffic generator with different X SA and DA for N EVI's. Send F frames from traffic generator to SHPE3 with X different SA and DA. There will be 2X number of MAC address will be learned in DUT EVPN

MAC table. There is a bi directional traffic flow with F pps in each direction. The DUT must run with traffic for 24 hours, every hour check for memory leak, CPU usage and crash.

Measurement :

Take the hourly reading of CPU, process memory. There should not be any leak, crashes, CPU spikes.

4. Test Cases for PBB-EVPN Benchmarking

4.1. Local MAC Learning

Objective:

To Record the time taken to learn the MAC address locally.

Topology : Topology 1

Procedure:

Send "X" unicast frames from CE to MHPE1(DUT) working in SA mode with "X" different source and destination address from RT. The DUT must learn "X" macs in data plane.

Measurement :

Measure the time taken by the DUT to learn the "X" MACs in the data plane. The data plane measurement is taken by considering DUT as black box the range of "X" MAC is known from RT and the same must be learned in DUT, the time taken to learn "X" MAC is measured. Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The mac learning time is calculated by averaging the values obtained from "N" samples.

Mac learning in sec = $(T1+T2+..Tn/N)$

4.2. Remote Mac Learning

Objective:

To Record the time taken to learn the remote macs.

Topology : Topology 1

Procedure:

Send X frames with X different SA and DA to SHPE3 from RT. These macs will be flooded to MHPE1 and MHPE2 by SHPE3. The DUT and MHPE2 are running SA mode.

Measurement :

Measure the time taken to learn X mac address in DUT mac table. Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The mac learning time is calculated by averaging the values obtained by "N" samples.

Mac learning in sec = $(T1+T2+..Tn/N)$

4.3. MAC Flush due to link failure

Objective:

To record the time taken to flush the mac learned locally and the time taken to relearn the same amount of macs.

Topology : Topology 1

Procedure:

Send X frames with X different SA and DA to DUT from CE using traffic generator. Wait till the MHPE1 learn all X MAC address. Then fail the MHPE1 CE link and measure the time taken to flush these X MACs from the PBB-EVPN MAC table. Then bring up the link. Measure the time taken to relearn X MACS. The DUT and MHPE2 are running SA mode.

Measurement :

Measure the time taken for flushing these X MAC address. Measure the time taken to relearn the X MACs in DUT. Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The flush and the relearning time is calculated by averaging the values obtained by "N" samples.

Flush time for X Macs in sec = $(T1+T2+..Tn/N)$

Relearning time for X macs in sec = $(T1+T2+..Tn/N)$

4.4. MAC Flush due to remote Failure

Objective:

To record the time taken to flush the remote mac learned in DUT during remote link failure.

Topology : Topology 1

Procedure:

Send X frames with X different SA and DA to DUT from SHPE3 using traffic generator. Bring down the link between SHPE3 and traffic generator. Then measure the time taken to flush the DUT PBB-EVPN MAC address table. The remote MACs will be learned by Data plane, but the B-MAC will be learned by control plane. The DUT and MHPE2 are running SA mode.

Measurement :

Measure the time taken to flush X remote MACs from PBB-EVPN MAC table of DUT. Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The flush rate is calculated by averaging the values obtained by "N" samples.

Flush time for X Macs in sec = $(T1+T2+..Tn/N)$

4.5. MAC aging

Objective:

To measure the mac aging time.

Topology : Topology 1

Procedure:

Send X frames with X different SA and DA to DUT from CE using traffic generator. Wait till X MAC address are learned in DUT PBB- EVPN MAC table. Then stop the traffic. Record the time taken to flush X MAC entries due to aging. The DUT and MHPE2 running in SA mode

Measurement :

Measure the time taken to flush X MAC address due to aging. Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The aging is calculated by averaging the values obtained by "N" samples.

Aging time for X Macs in sec = $(T1+T2+..Tn/N)$

4.6. Remote MAC Aging.

Objective:

To measure the remote mac aging time.

Topology : Topology 1

Procedure:

Send X frames with X different SA and DA to DUT from SHPE3 using traffic generator. Stop the traffic at remote PE(SHPE3). Measure the time taken to remove these remote MACs from DUT PBB-EVPN MAC table. The DUT and MHPE2 are running in SA mode.

Measurement :

Measure the time taken to flush the X remote MACs from DUT PBB-EVPN MAC table due to aging Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The aging is calculated by averaging the values obtained by "N" samples.

Aging time for X Macs in sec = $(T1+T2+..Tn/N)$

4.7. Local and Remote MAC Learning

Objective:

To record the time taken to learn both local and remote macs.

Topology : Topology 1

Procedure:

Send X frames with X different SA and DA to DUT from SHPE3 using traffic generator. Send X frames with different SA and DA from traffic generator connected to CE. The SA and DA of flows must be complimentary to have unicast flows. Measure the time taken by the DUT to learn 2X in MAC table. DUT and MHPE2 are running in SA mode.

Measurement :

Measure the time taken to learn 2X MAC address table in DUT PBB-EVPN MAC table. Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The mac

learning time is calculated by averaging the values obtained by "N" samples.

Time to learn 2X Macs in sec = $(T1+T2+..Tn/N)$

4.8. High Availability

Objective:

To record traffic loss during routing engine failover.

Topology : Topology 1

Procedure:

Send X frames to DUT with X different SA and DA from CE using the traffic generator. Send X frames from traffic generator to SHPE3 with X different SA and DA so that 2X MAC address will be Learned in DUT. There is a bi directional traffic flow with X pps in each direction. Then do a routing engine fail-over.

Measurement :

There should be 0 traffic loss which is the ideal case, No change in the DF role. DUT should not withdraw any routes. Repeat the test "N" times and plot the data. The packet loss is calculated by averaging the values obtained from "N" samples.

Packet loss in sec = $(T1+T2+..Tn/N)$

4.9. Scale

Objective:

To measure the scale limit of DUT for PBB-EVPN.

Topology : Topology 1

Procedure:

The DUT, MHPE2 and SHPE3 are scaled to "N" PBB-EVPN instances. Clear BGP neighbors in the DUT Once adjacency is established in DUT, check routes received from SHPE3 and MHPE2.

Measurement :

There should not be any loss of route types 2,3 and 4 in DUT. The DUT must relearn all type 2,3 and 4 routes from remote routers. The DUT must be subjected to various values of N to find the optimal scale limit.

4.10. Scale Convergence

Objective:

To measure the convergence time of DUT when the DUT is scaled with EVPN instance along with traffic.

Topology : Topology 1

Procedure:

Scale N PBB-EVIs in DUT, SHPE3 and MHPE2. Send F frames to DUT from CE using traffic generator with X different SA and DA for N PBB-EVI's. Send F frames from traffic generator to SHPE3 with X different SA and DA. There will be 2X number of MAC address will be learned in DUT PBB-EVPN MAC table. There is a bi directional traffic flow with F pps in each direction. Then clear the BGP neighbors in the DUT. Once the adjacency is restored in DUT. Measure the time taken to learn 2X MAC address in DUT MAC table.

Measurement :

The DUT must learn 2X MAC address. Measure the time taken to learn 2X MAC in DUT. Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The convergence time is calculated by averaging the values obtained by "N" samples.

Convergence time in sec = $(T1+T2+..Tn/N)$

4.11. Soak Test

Objective:

To measure the stability of the DUT in a scaled environment with traffic.

Topology : Topology 1

Procedure:

Scale N PBB-EVI's in DUT, SHPE3 and MHPE2. Send F frames to DUT from CE using traffic generator with X different SA and DA for N EVI's. Send F frames from traffic generator to SHPE3 with X different SA and DA. There will be 2X number of MAC address will be learned in DUT PBB-EVPN MAC table. There is a bi directional traffic flow with F pps in Each direction. The DUT must run with traffic for 24 hours, every hour check the memory leak, crashes.

Measurement :

Take the hourly reading of CPU process, memory usages. There should not be any memory leak, crashes, CPU spikes.

5. Acknowledgements

We would like to thank Fioccola Giuseppe of Telecom Italia reviewing our draft and commenting it. We would like to thank Sarah Banks for guiding and mentoring us.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

There is no additional consideration from RFC 6192.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [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>>.
- [RFC2899] Ginoza, S., "Request for Comments Summary RFC Numbers 2800-2899", RFC 2899, DOI 10.17487/RFC2899, May 2001, <<https://www.rfc-editor.org/info/rfc2899>>.

8.2. Informative References

- [RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <<https://www.rfc-editor.org/info/rfc7432>>.
- [RFC7623] Sajassi, A., Ed., Salam, S., Bitar, N., Isaac, A., and W. Henderickx, "Provider Backbone Bridging Combined with Ethernet VPN (PBB-EVPN)", RFC 7623, DOI 10.17487/RFC7623, September 2015, <<https://www.rfc-editor.org/info/rfc7623>>.

Appendix A. Appendix

Authors' Addresses

Sudhin Jacob (editor)
Juniper Networks
Bangalore
India

Phone: +91 8061212543
Email: sjacob@juniper.net

Kishore Tiruveedhula
Juniper Networks
10 Technology Park Dr
Westford, MA 01886
USA

Phone: +1 9785898861
Email: kishoret@juniper.net

Benchmarking Methodology Working Group
Internet-Draft
Intended status: Informational
Expires: March 6, 2020

B. Balarajah
C. Rossenhoevel
EANTC AG
B. Monkman
NetSecOPEN
September 3, 2019

Benchmarking Methodology for Network Security Device Performance
draft-ietf-bmwg-ngfw-performance-01

Abstract

This document provides benchmarking terminology and methodology for next-generation network security devices including next-generation firewalls (NGFW), intrusion detection and prevention solutions (IDS/IPS) and unified threat management (UTM) implementations. This document aims to strongly improve the applicability, reproducibility, and transparency of benchmarks and to align the test methodology with today's increasingly complex layer 7 application use cases. The main areas covered in this document are test terminology, traffic profiles and benchmarking methodology for NGFWs to start with.

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 March 6, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Requirements	4
3. Scope	4
4. Test Setup	4
4.1. Testbed Configuration	4
4.2. DUT/SUT Configuration	5
4.3. Test Equipment Configuration	9
4.3.1. Client Configuration	9
4.3.2. Backend Server Configuration	11
4.3.3. Traffic Flow Definition	11
4.3.4. Traffic Load Profile	12
5. Test Bed Considerations	13
6. Reporting	14
6.1. Key Performance Indicators	15
7. Benchmarking Tests	16
7.1. Throughput Performance With NetSecOPEN Traffic Mix	17
7.1.1. Objective	17
7.1.2. Test Setup	17
7.1.3. Test Parameters	17
7.1.4. Test Procedures and expected Results	19
7.2. TCP/HTTP Connections Per Second	20
7.2.1. Objective	20
7.2.2. Test Setup	20
7.2.3. Test Parameters	20
7.2.4. Test Procedures and Expected Results	22
7.3. HTTP Throughput	23
7.3.1. Objective	23
7.3.2. Test Setup	23
7.3.3. Test Parameters	23
7.3.4. Test Procedures and Expected Results	25
7.4. TCP/HTTP Transaction Latency	26
7.4.1. Objective	26
7.4.2. Test Setup	26
7.4.3. Test Parameters	26
7.4.4. Test Procedures and Expected Results	28
7.5. Concurrent TCP/HTTP Connection Capacity	29
7.5.1. Objective	29
7.5.2. Test Setup	30

7.5.3. Test Parameters	30
7.5.4. Test Procedures and expected Results	31
7.6. TCP/HTTPS Connections per second	33
7.6.1. Objective	33
7.6.2. Test Setup	33
7.6.3. Test Parameters	33
7.6.4. Test Procedures and expected Results	35
7.7. HTTPS Throughput	36
7.7.1. Objective	36
7.7.2. Test Setup	36
7.7.3. Test Parameters	36
7.7.4. Test Procedures and Expected Results	39
7.8. HTTPS Transaction Latency	40
7.8.1. Objective	40
7.8.2. Test Setup	40
7.8.3. Test Parameters	40
7.8.4. Test Procedures and Expected Results	42
7.9. Concurrent TCP/HTTPS Connection Capacity	43
7.9.1. Objective	43
7.9.2. Test Setup	43
7.9.3. Test Parameters	43
7.9.4. Test Procedures and expected Results	45
8. Formal Syntax	46
9. IANA Considerations	46
10. Security Considerations	46
11. Acknowledgements	47
12. Contributors	47
13. References	47
13.1. Normative References	47
13.2. Informative References	47
Appendix A. NetSecOPEN Basic Traffic Mix	48
Authors' Addresses	56

1. Introduction

15 years have passed since IETF recommended test methodology and terminology for firewalls initially ([RFC2647], [RFC3511]). The requirements for network security element performance and effectiveness have increased tremendously since then. Security function implementations have evolved to more advanced areas and have diversified into intrusion detection and prevention, threat management, analysis of encrypted traffic, etc. In an industry of growing importance, well-defined and reproducible key performance indicators (KPIs) are increasingly needed: They enable fair and reasonable comparison of network security functions. All these reasons have led to the creation of a new next-generation firewall benchmarking document.

2. Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119], [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Scope

This document provides testing terminology and testing methodology for next-generation firewalls and related security functions. It covers two main areas: security effectiveness configurations, followed by performance benchmark testing. This document focuses on advanced, realistic, and reproducible testing methods. Additionally, it describes test bed environments, test tool requirements and test result formats.

4. Test Setup

Test setup defined in this document is applicable to all benchmarking test scenarios described in Section 7.

4.1. Testbed Configuration

Testbed configuration MUST ensure that any performance implications that are discovered during the benchmark testing aren't due to the inherent physical network limitations such as number of physical links and forwarding performance capabilities (throughput and latency) of the network device in the testbed. For this reason, this document recommends avoiding external devices such as switches and routers in the testbed wherever possible.

However, in the typical deployment, the security devices (Device Under Test/System Under Test) are connected to routers and switches which will reduce the number of entries in MAC or ARP tables of the Device Under Test/System Under Test (DUT/SUT). If MAC or ARP tables have many entries, this may impact the actual DUT/SUT performance due to MAC and ARP/ND table lookup processes. Therefore, it is RECOMMENDED to connect aggregation switches or routers between test equipment and DUT/SUT as shown in Figure 1. The aggregation switches or routers can be also used to aggregate the test equipment or DUT/SUT ports, if the numbers of used ports are mismatched between test equipment and DUT/SUT.

If the test equipment is capable of emulating layer 3 routing functionality and there is no need for test equipment port

aggregation, it is RECOMMENDED to configure the test setup as shown in Figure 2.

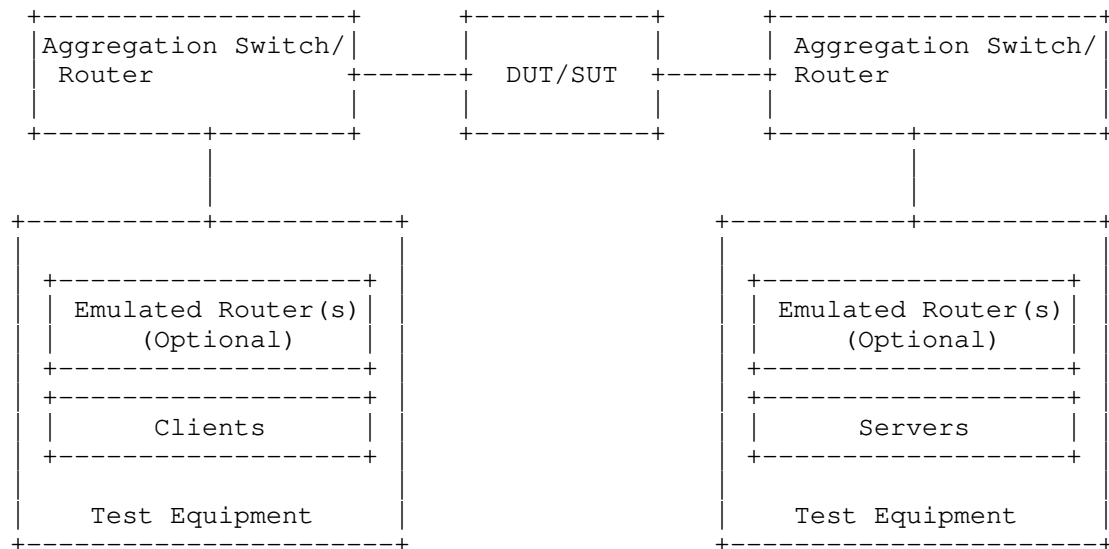


Figure 1: Testbed Setup - Option 1

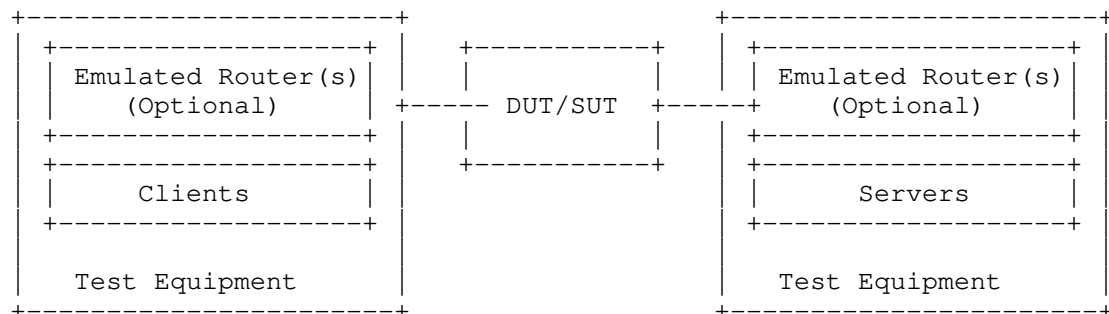


Figure 2: Testbed Setup - Option 2

4.2. DUT/SUT Configuration

A unique DUT/SUT configuration MUST be used for all benchmarking tests described in Section 7. Since each DUT/SUT will have their own unique configuration, users SHOULD configure their device with the same parameters and security features that would be used in the actual deployment of the device or a typical deployment in order to achieve maximum security coverage.

This document attempts to define the recommended security features which SHOULD be consistently enabled for all the benchmarking tests described in Section 7. Table 1 below describes the RECOMMENDED sets of feature list which SHOULD be configured on the DUT/SUT.

Based on customer use case, users MAY enable or disable SSL inspection feature for "Throughput Performance with NetSecOPEN Traffic Mix" test scenario described in Section 7.1

To improve repeatability, a summary of the DUT configuration including description of all enabled DUT/SUT features MUST be published with the benchmarking results.

DUT Features	NGFW	
	RECOMMENDED	OPTIONAL
SSL Inspection	x	
IDS/IPS	x	
Web Filtering		x
Antivirus	x	
Anti Spyware	x	
Anti Botnet	x	
DLP		x
DDoS		x
Certificate Validation		x
Logging and Reporting	x	
Application Identification	x	

Table 1: DUT/SUT Feature List

In summary, DUT/SUT SHOULD be configured as follows:

- o All security inspection enabled
- o Disposition of all traffic is logged - Logging to an external device is permissible
- o Detection of Common Vulnerabilities and Exposures (CVE) matching the following characteristics when searching the National Vulnerability Database (NVD)
 - * Common Vulnerability Scoring System (CVSS) Version: 2
 - * CVSS V2 Metrics: AV:N/Au:N/I:C/A:C
 - * AV=Attack Vector, Au=Authentication, I=Integrity and A=Availability
 - * CVSS V2 Severity: High (7-10)
 - * If doing a group test the published start date and published end date SHOULD be the same
- o Geographical location filtering and Application Identification and Control configured to be triggered based on a site or application from the defined traffic mix

In addition, it is also RECOMMENDED to configure a realistic number of access policy rules on the DUT/SUT. This document determines the number of access policy rules for three different classes of DUT/SUT. The classification of the DUT/SUT MAY be based on its maximum supported firewall throughput performance number defined in the vendor data sheet. This document classifies the DUT/SUT in four different categories; namely Extra Small, Small, Medium, and Large.

The RECOMMENDED throughput values for the following classes are:

Extra Small (XS) - supported throughput less than 1Gbit/s

Small (S) - supported throughput less than 5Gbit/s

Medium (M) - supported throughput greater than 5Gbit/s and less than 10Gbit/s

Large (L) - supported throughput greater than 10Gbit/s

The Access Control Rules (ACL) defined in Table 2 SHOULD be configured from top to bottom in the correct order as shown in the table.

(Note: There will be differences between how security vendors implement ACL decision making.) The configured ACL MUST NOT block the test traffic used for the benchmarking test scenarios.

				DUT/SUT Classification #rules			
Rules Type	Match Criteria	Description	Action	XS	S	M	L
Application layer	Application	Any application traffic NOT included in the test traffic	block	5	10	20	50
Transport layer	Src IP and TCP/UDP Dst ports	Any src IP subnet used in the test AND any dst ports NOT used in the test traffic	block	25	50	100	250
IP layer	Src/Dst IP	Any src/dst IP subnet NOT used in the test	block	25	50	100	250
Application layer	Application	Applications included in the test traffic	allow	10	10	10	10
Transport layer	Src IP and TCP/UDP Dst ports	Half of the src IP used in the test AND any dst ports used in the test traffic. One rule per subnet	allow	1	1	1	1
IP layer	Src IP	The rest of the src IP subnet range used in the test. One rule per subnet	allow	1	1	1	1

Table 2: DUT/SUT Access List

4.3. Test Equipment Configuration

In general, test equipment allows configuring parameters in different protocol layers. These parameters thereby influence the traffic flows which will be offered and impact performance measurements.

This document specifies common test equipment configuration parameters applicable for all test scenarios defined in Section 7. Any test scenario specific parameters are described under the test setup section of each test scenario individually.

4.3.1. Client Configuration

This section specifies which parameters SHOULD be considered while configuring clients using test equipment. Also, this section specifies the recommended values for certain parameters.

4.3.1.1. TCP Stack Attributes

The TCP stack SHOULD use a TCP Reno [RFC5681] variant, which include congestion avoidance, back off and windowing, fast retransmission, and fast recovery on every TCP connection between client and server endpoints. The default IPv4 and IPv6 MSS segments size MUST be set to 1460 bytes and 1440 bytes respectively and a TX and RX receive windows of 64 KByte. Client initial congestion window MUST NOT exceed 10 times the MSS. Delayed ACKs are permitted and the maximum client delayed Ack MUST NOT exceed 10 times the MSS before a forced ACK. Up to 3 retries SHOULD be allowed before a timeout event is declared. All traffic MUST set the TCP PSH flag to high. The source port range SHOULD be in the range of 1024 - 65535. Internal timeout SHOULD be dynamically scalable per RFC 793. Client SHOULD initiate and close TCP connections. TCP connections MUST be closed via FIN.

4.3.1.2. Client IP Address Space

The sum of the client IP space SHOULD contain the following attributes. The traffic blocks SHOULD consist of multiple unique, discontinuous static address blocks. A default gateway is permitted. The IPv4 ToS byte or IPv6 traffic class should be set to '00' or '000000' respectively.

The following equation can be used to determine the required total number of client IP addresses.

$$\text{Desired total number of client IP} = \frac{\text{Target throughput [Mbit/s]}}{\text{Throughput per IP address [Mbit/s]}}$$

Based on deployment and use case scenario, the value for "Throughput per IP address" can be varied.

(Option 1) DUT/SUT deployment scenario 1 : 6-7 Mbit/s per IP (e.g. 1,400-1,700 IPs per 10Gbit/s throughput)

(Option 2) DUT/SUT deployment scenario 2 : 0.1-0.2 Mbit/s per IP (e.g. 50,000-100,000 IPs per 10Gbit/s throughput)

Based on deployment and use case scenario, client IP addresses SHOULD be distributed between IPv4 and IPv6 type. The Following options can be considered for a selection of traffic mix ratio.

(Option 1) 100 % IPv4, no IPv6

(Option 2) 80 % IPv4, 20% IPv6

(Option 3) 50 % IPv4, 50% IPv6

(Option 4) 20 % IPv4, 80% IPv6

(Option 5) no IPv4, 100% IPv6

4.3.1.3. Emulated Web Browser Attributes

The emulated web browser contains attributes that will materially affect how traffic is loaded. The objective is to emulate modern, typical browser attributes to improve realism of the result set.

For HTTP traffic emulation, the emulated browser MUST negotiate HTTP 1.1. HTTP persistency MAY be enabled depending on test scenario. The browser MAY open multiple TCP connections per Server endpoint IP at any time depending on how many sequential transactions are needed to be processed. Within the TCP connection multiple transactions MAY be processed if the emulated browser has available connections. The browser SHOULD advertise a User-Agent header. Headers MUST be sent uncompressed. The browser SHOULD enforce content length validation.

For encrypted traffic, the following attributes SHALL define the negotiated encryption parameters. The test clients MUST use TLSv1.2 or higher. TLS record size MAY be optimized for the HTTPS response object size up to a record size of 16 KByte. The client endpoint MUST send TLS Extension Server Name Indication (SNI) information when opening a security tunnel. Each client connection MUST perform a full handshake with server certificate and MUST NOT use session reuse or resumption. Cipher suite and key size should be defined in the parameter session of each test scenario.

4.3.2. Backend Server Configuration

This document specifies which parameters should be considered while configuring emulated backend servers using test equipment.

4.3.2.1. TCP Stack Attributes

The TCP stack on the server side SHOULD be configured similar to the client side configuration described in Section 4.3.1.1. In addition, server initial congestion window MUST NOT exceed 10 times the MSS. Delayed ACKs are permitted and the maximum server delayed ACK MUST NOT exceed 10 times the MSS before a forced ACK.

4.3.2.2. Server Endpoint IP Addressing

The server IP blocks SHOULD consist of unique, discontinuous static address blocks with one IP per Server Fully Qualified Domain Name (FQDN) endpoint per test port. The IPv4 ToS byte and IPv6 traffic class bytes should be set to '00' and '000000' respectively.

4.3.2.3. HTTP / HTTPS Server Pool Endpoint Attributes

The server pool for HTTP SHOULD listen on TCP port 80 and emulate HTTP version 1.1 with persistence. The Server MUST advertise server type in the Server response header [RFC2616]. For HTTPS server, TLS 1.2 or higher MUST be used with a maximum record size of 16 KByte and MUST NOT use ticket resumption or Session ID reuse. The server MUST listen on port TCP 443. The server SHALL serve a certificate to the client. It is REQUIRED that the HTTPS server also check Host SNI information with the FQDN. Cipher suite and key size should be defined in the parameter section of each test scenario.

4.3.3. Traffic Flow Definition

This section describes the traffic pattern between client and server endpoints. At the beginning of the test, the server endpoint initializes and will be ready to accept connection states including initialization of the TCP stack as well as bound HTTP and HTTPS servers. When a client endpoint is needed, it will initialize and be given attributes such as a MAC and IP address. The behavior of the client is to sweep through the given server IP space, sequentially generating a recognizable service by the DUT. Thus, a balanced, mesh between client endpoints and server endpoints will be generated in a client port server port combination. Each client endpoint performs the same actions as other endpoints, with the difference being the source IP of the client endpoint and the target server IP pool. The client SHALL use Fully Qualified Domain Names (FQDN) in Host Headers and for TLS Server Name Indication (SNI).

4.3.3.1. Description of Intra-Client Behavior

Client endpoints are independent of other clients that are concurrently executing. When a client endpoint initiates traffic, this section describes how the client steps through different services. Once the test is initialized, the client endpoints SHOULD randomly hold (perform no operation) for a few milliseconds to allow for better randomization of start of client traffic. Each client will either open a new TCP connection or connect to a TCP persistence stack still open to that specific server. At any point that the service profile may require encryption, a TLS encryption tunnel will form presenting the URL request to the server. The server will then perform an SNI name check with the proposed FQDN compared to the domain embedded in the certificate. Only when correct, will the server process the HTTPS response object. The initial response object to the server MUST NOT have a fixed size; its size is based on benchmarking tests described in Section 7. Multiple additional sub-URLs (response objects on the service page) MAY be requested simultaneously. This MAY be to the same server IP as the initial URL. Each sub-object will also use a conical FQDN and URL path, as observed in the traffic mix used.

4.3.4. Traffic Load Profile

The loading of traffic is described in this section. The loading of a traffic load profile has five distinct phases: Init, ramp up, sustain, ramp down, and collection.

1. During the Init phase, test bed devices including the client and server endpoints should negotiate layer 2-3 connectivity such as MAC learning and ARP. Only after successful MAC learning or ARP/ND resolution SHALL the test iteration move to the next phase. No measurements are made in this phase. The minimum RECOMMEND time for Init phase is 5 seconds. During this phase, the emulated clients SHOULD NOT initiate any sessions with the DUT/SUT, in contrast, the emulated servers should be ready to accept requests from DUT/SUT or from emulated clients.
2. In the ramp up phase, the test equipment SHOULD start to generate the test traffic. It SHOULD use a set approximate number of unique client IP addresses actively to generate traffic. The traffic SHOULD ramp from zero to desired target objective. The target objective will be defined for each benchmarking test. The duration for the ramp up phase MUST be configured long enough, so that the test equipment does not overwhelm DUT/SUT's supported performance metrics namely; connections per second, concurrent TCP connections, and application transactions per second. The

RECOMMENDED time duration for the ramp up phase is 180-300 seconds. No measurements are made in this phase.

3. In the sustain phase, the test equipment SHOULD continue generating traffic to constant target value for a constant number of active client IPs. The RECOMMENDED time duration for sustain phase is 600 seconds. This is the phase where measurements occur.
4. In the ramp down/close phase, no new connections are established, and no measurements are made. The time duration for ramp up and ramp down phase SHOULD be same. The RECOMMENDED duration of this phase is between 180 to 300 seconds.
5. The last phase is administrative and will occur when the test equipment merges and collates the report data.

5. Test Bed Considerations

This section recommends steps to control the test environment and test equipment, specifically focusing on virtualized environments and virtualized test equipment.

1. Ensure that any ancillary switching or routing functions between the system under test and the test equipment do not limit the performance of the traffic generator. This is specifically important for virtualized components (vSwitches, vRouters).
2. Verify that the performance of the test equipment matches and reasonably exceeds the expected maximum performance of the system under test.
3. Assert that the test bed characteristics are stable during the entire test session. Several factors might influence stability specifically for virtualized test beds, for example additional workloads in a virtualized system, load balancing and movement of virtual machines during the test, or simple issues such as additional heat created by high workloads leading to an emergency CPU performance reduction.

Test bed reference pre-tests help to ensure that the maximum desired traffic generator aspects such as throughput, transaction per second, connection per second, concurrent connection and latency.

Once the desired maximum performance goals for the system under test have been identified, a safety margin of 10% SHOULD be added for throughput and subtracted for maximum latency and maximum packet loss.

Test bed preparation may be performed either by configuring the DUT in the most trivial setup (fast forwarding) or without presence of DUT.

6. Reporting

This section describes how the final report should be formatted and presented. The final test report MAY have two major sections; Introduction and result sections. The following attributes SHOULD be present in the introduction section of the test report.

1. The name of the NetSecOPEN traffic mix (see Appendix A) MUST be prominent.
2. The time and date of the execution of the test MUST be prominent.
3. Summary of testbed software and Hardware details

A. DUT Hardware/Virtual Configuration

- + This section SHOULD clearly identify the make and model of the DUT
- + The port interfaces, including speed and link information MUST be documented.
- + If the DUT is a virtual VNF, interface acceleration such as DPDK and SR-IOV MUST be documented as well as cores used, RAM used, and the pinning / resource sharing configuration. The Hypervisor and version MUST be documented.
- + Any additional hardware relevant to the DUT such as controllers MUST be documented

B. DUT Software

- + The operating system name MUST be documented
- + The version MUST be documented
- + The specific configuration MUST be documented

C. DUT Enabled Features

- + Configured DUT/SUT features (see Table 1) MUST be documented

- + Attributes of those featured MUST be documented
- + Any additional relevant information about features MUST be documented

D. Test equipment hardware and software

- + Test equipment vendor name
- + Hardware details including model number, interface type
- + Test equipment firmware and test application software version

4. Results Summary / Executive Summary

1. Results SHOULD resemble a pyramid in how it is reported, with the introduction section documenting the summary of results in a prominent, easy to read block.
2. In the result section of the test report, the following attributes should be present for each test scenario.
 - a. KPIs MUST be documented separately for each test scenario. The format of the KPI metrics should be presented as described in Section 6.1.
 - b. The next level of details SHOULD be graphs showing each of these metrics over the duration (sustain phase) of the test. This allows the user to see the measured performance stability changes over time.

6.1. Key Performance Indicators

This section lists KPIs for overall benchmarking tests scenarios. All KPIs MUST be measured during the sustain phase of the traffic load profile described in Section 4.3.4. All KPIs MUST be measured from the result output of test equipment.

- o Concurrent TCP Connections
This key performance indicator measures the average concurrent open TCP connections in the sustaining period.
- o TCP Connections Per Second
This key performance indicator measures the average established TCP connections per second in the sustaining period. For "TCP/HTTP(S) Connection Per Second" benchmarking test scenario, the KPI

is measured average established and terminated TCP connections per second simultaneously.

- o Application Transactions Per Second
This key performance indicator measures the average successfully completed application transactions per second in the sustaining period.
- o TLS Handshake Rate
This key performance indicator measures the average TLS 1.2 or higher session formation rate within the sustaining period.
- o Throughput
This key performance indicator measures the average Layer 2 throughput within the sustaining period as well as average packets per seconds within the same period. The value of throughput SHOULD be presented in Gbit/s rounded to two places of precision with a more specific Kbit/s in parenthesis. Optionally, goodput MAY also be logged as an average goodput rate measured over the same period. Goodput result SHALL also be presented in the same format as throughput.
- o URL Response time / Time to Last Byte (TTLB)
This key performance indicator measures the minimum, average and maximum per URL response time in the sustaining period. The latency is measured at Client and in this case would be the time duration between sending a GET request from Client and the receipt of the complete response from the server.
- o Application Transaction Latency
This key performance indicator measures the minimum, average and maximum the amount of time to receive all objects from the server. The value of application transaction latency SHOULD be presented in millisecond rounded to zero decimal.
- o Time to First Byte (TTFB)
This key performance indicator will measure minimum, average and maximum the time to first byte. TTFB is the elapsed time between sending the SYN packet from the client and receiving the first byte of application data from the DUT/SUT. TTFB SHOULD be expressed in millisecond.

7. Benchmarking Tests

7.1. Throughput Performance With NetSecOPEN Traffic Mix

7.1.1. Objective

Using NetSecOPEN traffic mix, determine the maximum sustainable throughput performance supported by the DUT/SUT. (see Appendix A for details about traffic mix)

This test scenario is RECOMMENDED to perform twice; one with SSL inspection feature enabled and the second scenario with SSL inspection feature disabled on the DUT/SUT.

7.1.2. Test Setup

Test bed setup MUST be configured as defined in Section 4. Any test scenario specific test bed configuration changes MUST be documented.

7.1.3. Test Parameters

In this section, test scenario specific parameters SHOULD be defined.

7.1.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific test scenario MUST be documented.

7.1.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. Following parameters MUST be noted for this test scenario:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target throughput: It can be defined based on requirements. Otherwise it represents aggregated line rate of interface(s) used in the DUT/SUT

Initial throughput: 10% of the "Target throughput"

One of the following ciphers and keys are RECOMMENDED to use for this test scenarios.

1. ECHDE-ECDSA-AES128-GCM-SHA256 with Prime256v1 (Signature Hash Algorithm: ecdsa_secp256r1_sha256 and Supported group: secp256r1)
2. ECDHE-RSA-AES128-GCM-SHA256 with RSA 2048 (Signature Hash Algorithm: rsa_pkcs1_sha256 and Supported group: secp256)
3. ECDHE-ECDSA-AES256-GCM-SHA384 with Secp521 (Signature Hash Algorithm: ecdsa_secp384r1_sha384 and Supported group: secp521r1)
4. ECDHE-RSA-AES256-GCM-SHA384 with RSA 4096 (Signature Hash Algorithm: rsa_pkcs1_sha384 and Supported group: secp256)

7.1.3.3. Traffic Profile

Traffic profile: Test scenario MUST be run with a single application traffic mix profile (see Appendix A for details about traffic mix). The name of the NetSecOPEN traffic mix MUST be documented.

7.1.3.4. Test Results Validation Criteria

The following test Criteria is defined as test results validation criteria. Test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

- a. Number of failed Application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of total attempt transactions
- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections
- c. Maximum deviation (max. dev) of application transaction time or TTLB (Time To Last Byte) MUST be less than X (The value for "X" will be finalized and updated after completion of PoC test)
The following equation MUST be used to calculate the deviation of application transaction latency or TTLB
$$\text{max. dev} = \max((\text{avg_latency} - \text{min_latency}), (\text{max_latency} - \text{avg_latency})) / (\text{Initial latency})$$

Where, the initial latency is calculated using the following equation. For this calculation, the latency values (min', avg' and max') MUST be measured during test procedure step 1 as defined in Section 7.1.4.1.
The variable latency represents application transaction latency or TTLB.

Initial latency:= min((avg' latency - min' latency) | (max' latency - avg' latency))

- d. Maximum value of Time to First Byte (TTFB) MUST be less than X

7.1.3.5. Measurement

Following KPI metrics MUST be reported for this test scenario.

Mandatory KPIs: average Throughput, average Concurrent TCP connections, TTLB/application transaction latency (minimum, average and maximum) and average application transactions per second

Optional KPIs: average TCP connections per second, average TLS handshake rate and TTFB

7.1.4. Test Procedures and expected Results

The test procedures are designed to measure the throughput performance of the DUT/SUT at the sustaining period of traffic load profile. The test procedure consists of three major steps.

7.1.4.1. Step 1: Test Initialization and Qualification

Verify the link status of the all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure traffic load profile of the test equipment to generate test traffic at the "Initial throughput" rate as described in the parameters Section 7.1.3.2. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4. The DUT/SUT SHOULD reach the "Initial throughput" during the sustain phase. Measure all KPI as defined in Section 7.1.3.5. The measured KPIs during the sustain phase MUST meet validation criteria "a" and "b" defined in Section 7.1.3.4.

If the KPI metrics do not meet the validation criteria, the test procedure MUST NOT be continued to step 2.

7.1.4.2. Step 2: Test Run with Target Objective

Configure test equipment to generate traffic at the "Target throughput" rate defined in the parameter table. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4. The test equipment SHOULD start to measure and record all specified KPIs. The frequency of KPI metric measurements SHOULD be 2 seconds. Continue the test until all traffic profile phases are completed.

The DUT/SUT is expected to reach the desired target throughput during the sustain phase. In addition, the measured KPIs MUST meet all validation criteria. Follow step 3, if the KPI metrics do not meet the validation criteria.

7.1.4.3. Step 3: Test Iteration

Determine the maximum and average achievable throughput within the validation criteria. Final test iteration MUST be performed for the test duration defined in Section 4.3.4.

7.2. TCP/HTTP Connections Per Second

7.2.1. Objective

Using HTTP traffic, determine the maximum sustainable TCP connection establishment rate supported by the DUT/SUT under different throughput load conditions.

To measure connections per second, test iterations MUST use different fixed HTTP response object sizes defined in Section 7.2.3.2.

7.2.2. Test Setup

Test bed setup SHOULD be configured as defined in Section 4. Any specific test bed configuration changes such as number of interfaces and interface type, etc. MUST be documented.

7.2.3. Test Parameters

In this section, test scenario specific parameters SHOULD be defined.

7.2.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific test scenario MUST be documented.

7.2.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. Following parameters MUST be documented for this test scenario:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target connections per second: Initial value from product data sheet (if known)

Initial connections per second: 10% of "Target connections per second"

The client SHOULD negotiate HTTP 1.1 and close the connection with FIN immediately after completion of one transaction. In each test iteration, client MUST send GET command requesting a fixed HTTP response object size.

The RECOMMENDED response object sizes are 1, 2, 4, 16, 64 KByte

7.2.3.3. Test Results Validation Criteria

The following test Criteria is defined as test results validation criteria. Test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

- a. Number of failed Application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of total attempt transactions
- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections
- c. During the sustain phase, traffic should be forwarded at a constant rate
- d. Concurrent TCP connections SHOULD be constant during steady state. Any deviation of concurrent TCP connections MUST be less than 10%. This confirms the DUT opens and closes TCP connections almost at the same rate

7.2.3.4. Measurement

Following KPI metrics MUST be reported for each test iteration.

Mandatory KPIs: average TCP connections per second, average Throughput and Average Time to First Byte (TTFB).

7.2.4. Test Procedures and Expected Results

The test procedure is designed to measure the TCP connections per second rate of the DUT/SUT at the sustaining period of the traffic load profile. The test procedure consists of three major steps. This test procedure MAY be repeated multiple times with different IP types; IPv4 only, IPv6 only and IPv4 and IPv6 mixed traffic distribution.

7.2.4.1. Step 1: Test Initialization and Qualification

Verify the link status of all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure the traffic load profile of the test equipment to establish "initial connections per second" as defined in the parameters Section 7.2.3.2. The traffic load profile SHOULD be defined as described in Section 4.3.4.

The DUT/SUT SHOULD reach the "Initial connections per second" before the sustain phase. The measured KPIs during the sustain phase MUST meet validation criteria a, b, c, and d defined in Section 7.2.3.3.

If the KPI metrics do not meet the validation criteria, the test procedure MUST NOT be continued to "Step 2".

7.2.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish "Target connections per second" defined in the parameters table. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4.

During the ramp up and sustain phase of each test iteration, other KPIs such as throughput, concurrent TCP connections and application transactions per second MUST NOT reach to the maximum value the DUT/SUT can support. The test results for specific test iterations SHOULD NOT be reported, if the above mentioned KPI (especially throughput) reaches the maximum value. (Example: If the test iteration with 64 KByte of HTTP response object size reached the maximum throughput limitation of the DUT, the test iteration MAY be interrupted and the result for 64 KByte SHOULD NOT be reported).

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement SHOULD be 2 seconds. Continue the test until all traffic profile phases are completed.

The DUT/SUT is expected to reach the desired target connections per second rate at the sustain phase. In addition, the measured KPIs MUST meet all validation criteria.

Follow step 3, if the KPI metrics do not meet the validation criteria.

7.2.4.3. Step 3: Test Iteration

Determine the maximum and average achievable connections per second within the validation criteria.

7.3. HTTP Throughput

7.3.1. Objective

Determine the throughput for HTTP transactions varying the HTTP response object size.

7.3.2. Test Setup

Test bed setup SHOULD be configured as defined in Section 4. Any specific test bed configuration changes such as number of interfaces and interface type, etc. must be documented.

7.3.3. Test Parameters

In this section, test scenario specific parameters SHOULD be defined.

7.3.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific test scenario MUST be documented.

7.3.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. Following parameters MUST be documented for this test scenario:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target Throughput: Initial value from product data sheet (if known)

Initial Throughput: 10% of "Target Throughput"

Number of HTTP response object requests (transactions) per connection: 10

RECOMMENDED HTTP response object size: 1 KByte, 16 KByte, 64 KByte, 256 KByte and mixed objects defined in the table

Object size (KByte)	Number of requests/ Weight
0.2	1
6	1
8	1
9	1
10	1
25	1
26	1
35	1
59	1
347	1

Table 3: Mixed Objects

7.3.3.3. Test Results Validation Criteria

The following test Criteria is defined as test results validation criteria. Test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile

- a. Number of failed Application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of attempt transactions.
- b. Traffic should be forwarded constantly.

- c. Concurrent connetions MUST be constant. The deviation of concurrent TCP connection MUST NOT increase more than 10%

7.3.3.4. Measurement

The KPI metrics MUST be reported for this test scenario:

Average Throughput, average HTTP transactions per second, concurrent connections, and average TCP connections per second.

7.3.4. Test Procedures and Expected Results

The test procedure is designed to measure HTTP throughput of the DUT/SUT. The test procedure consists of three major steps. This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution and HTTP response object sizes.

7.3.4.1. Step 1: Test Initialization and Qualification

Verify the link status of the all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure traffic load profile of the test equipment to establish "Initial Throughput" as defined in the parameters Section 7.3.3.2.

The traffic load profile SHOULD be defined as described in Section 4.3.4. The DUT/SUT SHOULD reach the "Initial Throughput" during the sustain phase. Measure all KPI as defined in Section 7.3.3.4.

The measured KPIs during the sustain phase MUST meet the validation criteria "a" defined in Section 7.3.3.3.

If the KPI metrics do not meet the validation criteria, the test procedure MUST NOT be continued to "Step 2".

7.3.4.2. Step 2: Test Run with Target Objective

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement SHOULD be 2 seconds. Continue the test until all traffic profile phases are completed.

The DUT/SUT is expected to reach the desired "Target Throughput" at the sustain phase. In addition, the measured KPIs must meet all validation criteria.

Perform the test separately for each HTTP response object size.

Follow step 3, if the KPI metrics do not meet the validation criteria.

7.3.4.3. Step 3: Test Iteration

Determine the maximum and average achievable throughput within the validation criteria. Final test iteration MUST be performed for the test duration defined in Section 4.3.4.

7.4. TCP/HTTP Transaction Latency

7.4.1. Objective

Using HTTP traffic, determine the average HTTP transaction latency when DUT is running with sustainable HTTP transactions per second supported by the DUT/SUT under different HTTP response object sizes.

Test iterations MUST be performed with different HTTP response object sizes in two different scenarios. one with a single transaction and the other with multiple transactions within a single TCP connection. For consistency both the single and multiple transaction test MUST be configured with HTTP 1.1.

Scenario 1: The client MUST negotiate HTTP 1.1 and close the connection with FIN immediately after completion of a single transaction (GET and RESPONSE).

Scenario 2: The client MUST negotiate HTTP 1.1 and close the connection FIN immediately after completion of 10 transactions (GET and RESPONSE) within a single TCP connection.

7.4.2. Test Setup

Test bed setup SHOULD be configured as defined in Section 4. Any specific test bed configuration changes such as number of interfaces and interface type, etc. MUST be documented.

7.4.3. Test Parameters

In this section, test scenario specific parameters SHOULD be defined.

7.4.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific test scenario MUST be documented.

7.4.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3 . Following parameters MUST be documented for this test scenario:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target objective for scenario 1: 50% of the maximum connection per second measured in test scenario TCP/HTTP Connections Per Second (Section 7.2)

Target objective for scenario 2: 50% of the maximum throughput measured in test scenario HTTP Throughput (Section 7.3)

Initial objective for scenario 1: 10% of Target objective for scenario 1"

Initial objective for scenario 2: 10% of "Target objective for scenario 2"

HTTP transaction per TCP connection: test scenario 1 with single transaction and the second scenario with 10 transactions

HTTP 1.1 with GET command requesting a single object. The RECOMMENDED object sizes are 1, 16 or 64 KByte. For each test iteration, client MUST request a single HTTP response object size.

7.4.3.3. Test Results Validation Criteria

The following test Criteria is defined as test results validation criteria. Test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile. Ramp up and ramp down phase SHOULD NOT be considered.

Generic criteria:

- a. Number of failed Application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of attempt transactions.

- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections
- c. During the sustain phase, traffic should be forwarded at a constant rate.
- d. Concurrent TCP connections should be constant during steady state. This confirms the DUT opens and closes TCP connections at the same rate.
- e. After ramp up the DUT MUST achieve the "Target objective" defined in the parameter Section 7.4.3.2 and remain in that state for the entire test duration (sustain phase).

7.4.3.4. Measurement

Following KPI metrics MUST be reported for each test scenario and HTTP response object sizes separately:

average TCP connections per second and average application transaction latency

All KPI's are measured once the target throughput achieves the steady state.

7.4.4. Test Procedures and Expected Results

The test procedure is designed to measure the average application transaction latencies or TTLB when the DUT is operating close to 50% of its maximum achievable throughput or connections per second. This test procedure CAN be repeated multiple times with different IP types (IPv4 only, IPv6 only and IPv4 and IPv6 mixed traffic distribution), HTTP response object sizes and single and multiple transactions per connection scenarios.

7.4.4.1. Step 1: Test Initialization and Qualification

Verify the link status of the all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure traffic load profile of the test equipment to establish "Initial objective" as defined in the parameters Section 7.4.3.2. The traffic load profile can be defined as described in Section 4.3.4.

The DUT/SUT SHOULD reach the "Initial objective" before the sustain phase. The measured KPIs during the sustain phase MUST meet the validation criteria a, b, c, d, e and f defined in Section 7.4.3.3.

If the KPI metrics do not meet the validation criteria, the test procedure MUST NOT be continued to "Step 2".

7.4.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish "Target objective" defined in the parameters table. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4.

During the ramp up and sustain phase, other KPIs such as throughput, concurrent TCP connections and application transactions per second MUST NOT reach to the maximum value that the DUT/SUT can support. The test results for specific test iterations SHOULD NOT be reported, if the above mentioned KPI (especially throughput) reaches to the maximum value. (Example: If the test iteration with 64 KByte of HTTP response object size reached the maximum throughput limitation of the DUT, the test iteration MAY be interrupted and the result for 64 KByte SHOULD NOT be reported).

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement SHOULD be 2 seconds. Continue the test until all traffic profile phases are completed. DUT/SUT is expected to reach the desired "Target objective" at the sustain phase. In addition, the measured KPIs MUST meet all validation criteria.

Follow step 3, if the KPI metrics do not meet the validation criteria.

7.4.4.3. Step 3: Test Iteration

Determine the maximum achievable connections per second within the validation criteria and measure the latency values.

7.5. Concurrent TCP/HTTP Connection Capacity

7.5.1. Objective

Determine the maximum number of concurrent TCP connections that the DUT/ SUT sustains when using HTTP traffic.

7.5.2. Test Setup

Test bed setup SHOULD be configured as defined in Section 4. Any specific test bed configuration changes such as number of interfaces and interface type, etc. must be documented.

7.5.3. Test Parameters

In this section, test scenario specific parameters SHOULD be defined.

7.5.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific test scenario MUST be documented.

7.5.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. Following parameters MUST be noted for this test scenario:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target concurrent connection: Initial value from product data sheet (if known)

Initial concurrent connection: 10% of "Target concurrent connection"

Maximum connections per second during ramp up phase: 50% of maximum connections per second measured in test scenario TCP/HTTP Connections per second (Section 7.2)

Ramp up time (in traffic load profile for "Target concurrent connection"): "Target concurrent connection" / "Maximum connections per second during ramp up phase"

Ramp up time (in traffic load profile for "Initial concurrent connection"): "Initial concurrent connection" / "Maximum connections per second during ramp up phase"

The client MUST negotiate HTTP 1.1 with persistence and each client MAY open multiple concurrent TCP connections per server endpoint IP.

Each client sends 10 GET commands requesting 1 KByte HTTP response object in the same TCP connection (10 transactions/TCP connection) and the delay (think time) between the transaction MUST be X seconds.

$$X = ("Ramp\ up\ time" + "steady\ state\ time") / 10$$

The established connections SHOULD remain open until the ramp down phase of the test. During the ramp down phase, all connections SHOULD be successfully closed with FIN.

7.5.3.3. Test Results Validation Criteria

The following test Criteria is defined as test results validation criteria. Test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

- a. Number of failed Application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transaction) of total attempted transactions
- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections
- c. During the sustain phase, traffic should be forwarded constantly

7.5.3.4. Measurement

Following KPI metrics MUST be reported for this test scenario:

average Throughput, Concurrent TCP connections (minimum, average and maximum), TTLB/ application transaction latency (minimum, average and maximum) and average application transactions per second.

7.5.4. Test Procedures and expected Results

The test procedure is designed to measure the concurrent TCP connection capacity of the DUT/SUT at the sustaining period of traffic load profile. The test procedure consists of three major steps. This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution.

7.5.4.1. Step 1: Test Initialization and Qualification

Verify the link status of the all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure test equipment to establish "Initial concurrent TCP connections" defined in Section 7.5.3.2. Except ramp up time, the traffic load profile SHOULD be defined as described in Section 4.3.4.

During the sustain phase, the DUT/SUT SHOULD reach the "Initial concurrent TCP connections". The measured KPIs during the sustain phase MUST meet the validation criteria "a" and "b" defined in Section 7.5.3.3.

If the KPI metrics do not meet the validation criteria, the test procedure MUST NOT be continued to "Step 2".

7.5.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish "Target concurrent TCP connections". The test equipment SHOULD follow the traffic load profile definition (except ramp up time) as described in Section 4.3.4.

During the ramp up and sustain phase, the other KPIs such as throughput, TCP connections per second and application transactions per second MUST NOT reach to the maximum value that the DUT/SUT can support.

The test equipment SHOULD start to measure and record KPIs defined in Section 7.5.3.4. The frequency of measurement SHOULD be 2 seconds. Continue the test until all traffic profile phases are completed.

The DUT/SUT is expected to reach the desired target concurrent connection at the sustain phase. In addition, the measured KPIs must meet all validation criteria.

Follow step 3, if the KPI metrics do not meet the validation criteria.

7.5.4.3. Step 3: Test Iteration

Determine the maximum and average achievable concurrent TCP connections capacity within the validation criteria.

7.6. TCP/HTTPS Connections per second

7.6.1. Objective

Using HTTPS traffic, determine the maximum sustainable SSL/TLS session establishment rate supported by the DUT/SUT under different throughput load conditions.

Test iterations MUST include common cipher suites and key strengths as well as forward looking stronger keys. Specific test iterations MUST include ciphers and keys defined in Section 7.6.3.2.

For each cipher suite and key strengths, test iterations MUST use a single HTTPS response object size defined in the test equipment configuration parameters Section 7.6.3.2 to measure connections per second performance under a variety of DUT Security inspection load conditions.

7.6.2. Test Setup

Test bed setup SHOULD be configured as defined in Section 4. Any specific test bed configuration changes such as number of interfaces and interface type, etc. MUST be documented.

7.6.3. Test Parameters

In this section, test scenario specific parameters SHOULD be defined.

7.6.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific test scenario MUST be documented.

7.6.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. Following parameters MUST be documented for this test scenario:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target connections per second: Initial value from product data sheet (if known)

Initial connections per second: 10% of "Target connections per second"

RECOMMENDED ciphers and keys:

1. ECHDE-ECDSA-AES128-GCM-SHA256 with Prime256v1 (Signature Hash Algorithm: ecdsa_secp256r1_sha256 and Supported group: secp256r1)
2. ECDHE-RSA-AES128-GCM-SHA256 with RSA 2048 (Signature Hash Algorithm: rsa_pkcs1_sha256 and Supported group: secp256)
3. ECDHE-ECDSA-AES256-GCM-SHA384 with Secp521 (Signature Hash Algorithm: ecdsa_secp384r1_sha384 and Supported group: secp521r1)
4. ECDHE-RSA-AES256-GCM-SHA384 with RSA 4096 (Signature Hash Algorithm: rsa_pkcs1_sha384 and Supported group: secp256)

The client MUST negotiate HTTPS 1.1 and close the connection with FIN immediately after completion of one transaction. In each test iteration, client MUST send GET command requesting a fixed HTTPS response object size. The RECOMMENDED object sizes are 1, 2, 4, 16, 64 KByte.

7.6.3.3. Test Results Validation Criteria

The following test Criteria is defined as test results validation criteria:

- a. Number of failed Application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of attempt transactions
- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections
- c. During the sustain phase, traffic should be forwarded at a constant rate
- d. Concurrent TCP connections SHOULD be constant during steady state. This confirms that the DUT open and close the TCP connections at the same rate

7.6.3.4. Measurement

Following KPI metrics MUST be reported for this test scenario:

average TCP connections per second, average Throughput and Average Time to TCP First Byte.

7.6.4. Test Procedures and expected Results

The test procedure is designed to measure the TCP connections per second rate of the DUT/SUT at the sustaining period of traffic load profile. The test procedure consists of three major steps. This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution.

7.6.4.1. Step 1: Test Initialization and Qualification

Verify the link status of all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure traffic load profile of the test equipment to establish "Initial connections per second" as defined in Section 7.6.3.2. The traffic load profile CAN be defined as described in Section 4.3.4.

The DUT/SUT SHOULD reach the "Initial connections per second" before the sustain phase. The measured KPIs during the sustain phase MUST meet the validation criteria a, b, c, and d defined in Section 7.6.3.3.

If the KPI metrics do not meet the validation criteria, the test procedure MUST NOT be continued to "Step 2".

7.6.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish "Target connections per second" defined in the parameters table. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4.

During the ramp up and sustain phase, other KPIs such as throughput, concurrent TCP connections and application transactions per second MUST NOT reach the maximum value that the DUT/SUT can support. The test results for specific test iteration SHOULD NOT be reported, if the above mentioned KPI (especially throughput) reaches the maximum value. (Example: If the test iteration with 64 KByte of HTTPS response object size reached the maximum throughput limitation of the DUT, the test iteration can be interrupted and the result for 64 KByte SHOULD NOT be reported).

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement SHOULD be 2 seconds. Continue the test until all traffic profile phases are completed.

The DUT/SUT is expected to reach the desired target connections per second rate at the sustain phase. In addition, the measured KPIs must meet all validation criteria.

Follow the step 3, if the KPI metrics do not meet the validation criteria.

7.6.4.3. Step 3: Test Iteration

Determine the maximum and average achievable connections per second within the validation criteria.

7.7. HTTPS Throughput

7.7.1. Objective

Determine the throughput for HTTPS transactions varying the HTTPS response object size.

Test iterations MUST include common cipher suites and key strengths as well as forward looking stronger keys. Specific test iterations MUST include the ciphers and keys defined in the parameter Section 7.7.3.2.

7.7.2. Test Setup

Test bed setup SHOULD be configured as defined in Section 4. Any specific test bed configuration changes such as number of interfaces and interface type, etc. must be documented.

7.7.3. Test Parameters

In this section, test scenario specific parameters SHOULD be defined.

7.7.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific test scenario MUST be documented.

7.7.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. Following parameters MUST be documented for this test scenario:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target Throughput: Initial value from product data sheet (if known)

Initial Throughput: 10% of "Target Throughput"

Number of HTTPS response object requests (transactions) per connection: 10

RECOMMENDED ciphers and keys:

1. ECHDE-ECDSA-AES128-GCM-SHA256 with Prime256v1 (Signature Hash Algorithm: ecdsa_secp256r1_sha256 and Supported group: secp256r1)
2. ECDHE-RSA-AES128-GCM-SHA256 with RSA 2048 (Signature Hash Algorithm: rsa_pkcs1_sha256 and Supported group: secp256)
3. ECDHE-ECDSA-AES256-GCM-SHA384 with Secp521 (Signature Hash Algorithm: ecdsa_secp384r1_sha384 and Supported group: secp521r1)
4. ECDHE-RSA-AES256-GCM-SHA384 with RSA 4096 (Signature Hash Algorithm: rsa_pkcs1_sha384 and Supported group: secp256)

RECOMMENDED HTTPS response object size: 1 KByte, 2 KByte, 4 KByte, 16 KByte, 64 KByte, 256 KByte and mixed object defined in the table below.

Object size (KByte)	Number of requests/ Weight
0.2	1
6	1
8	1
9	1
10	1
25	1
26	1
35	1
59	1
347	1

Table 4: Mixed Objects

7.7.3.3. Test Results Validation Criteria

The following test Criteria is defined as test results validation criteria. Test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

- a. Number of failed Application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of attempt transactions.
- b. Traffic should be forwarded constantly.
- c. The deviation of concurrent TCP connections MUST be less than 10%

7.7.3.4. Measurement

The KPI metrics MUST be reported for this test scenario:

Average Throughput, Average transactions per second, concurrent connections, and average TCP connections per second.

7.7.4. Test Procedures and Expected Results

The test procedure consists of three major steps. This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution and HTTPS response object sizes.

7.7.4.1. Step 1: Test Initialization and Qualification

Verify the link status of the all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure traffic load profile of the test equipment to establish "initial throughput" as defined in the parameters Section 7.7.3.2.

The traffic load profile should be defined as described in Section 4.3.4. The DUT/SUT SHOULD reach the "Initial Throughput" during the sustain phase. Measure all KPI as defined in Section 7.7.3.4.

The measured KPIs during the sustain phase MUST meet the validation criteria "a" defined in Section 7.7.3.3.

If the KPI metrics do not meet the validation criteria, the test procedure MUST NOT be continued to "Step 2".

7.7.4.2. Step 2: Test Run with Target Objective

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement SHOULD be 2 seconds. Continue the test until all traffic profile phases are completed.

The DUT/SUT is expected to reach the desired "Target Throughput" at the sustain phase. In addition, the measured KPIs MUST meet all validation criteria.

Perform the test separately for each HTTPS response object size.

Follow step 3, if the KPI metrics do not meet the validation criteria.

7.7.4.3. Step 3: Test Iteration

Determine the maximum and average achievable throughput within the validation criteria. Final test iteration MUST be performed for the test duration defined in Section 4.3.4.

7.8. HTTPS Transaction Latency

7.8.1. Objective

Using HTTPS traffic, determine the average HTTPS transaction latency when DUT is running with sustainable HTTPS transactions per second supported by the DUT/SUT under different HTTPS response object size.

Scenario 1: The client **MUST** negotiate HTTPS and close the connection with FIN immediately after completion of a single transaction (GET and RESPONSE).

Scenario 2: The client **MUST** negotiate HTTPS and close the connection with FIN immediately after completion of 10 transactions (GET and RESPONSE) within a single TCP connection.

7.8.2. Test Setup

Test bed setup **SHOULD** be configured as defined in Section 4. Any specific test bed configuration changes such as number of interfaces and interface type, etc. **MUST** be documented.

7.8.3. Test Parameters

In this section, test scenario specific parameters **SHOULD** be defined.

7.8.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters **MUST** conform to the requirements defined in Section 4.2. Any configuration changes for this specific test scenario **MUST** be documented.

7.8.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters **MUST** conform to the requirements defined in Section 4.3. Following parameters **MUST** be documented for this test scenario:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

RECOMMENDED cipher suites and key size: ECDHE-ECDSA-AES256-GCM-SHA384 with Secp521 bits key size (Signature Hash Algorithm: ecdsa_secp384r1_sha384 and Supported group: secp521r1)

Target objective for scenario 1: 50% of the maximum connections per second measured in test scenario TCP/HTTPS Connections per second (Section 7.6)

Target objective for scenario 2: 50% of the maximum throughput measured in test scenario HTTPS Throughput (Section 7.7)

Initial objective for scenario 1: 10% of Target objective for scenario 1"

Initial objective for scenario 2: 10% of "Target objective for scenario 2"

HTTPS transaction per TCP connection: test scenario 1 with single transaction and the second scenario with 10 transactions

HTTPS 1.1 with GET command requesting a single 1, 16 or 64 KByte object. For each test iteration, client MUST request a single HTTPS response object size.

7.8.3.3. Test Results Validation Criteria

The following test Criteria is defined as test results validation criteria. Test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile. Ramp up and ramp down phase SHOULD NOT be considered.

Generic criteria:

- a. Number of failed Application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of attempt transactions.
- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections
- c. During the sustain phase, traffic should be forwarded at a constant rate.
- d. Concurrent TCP connections should be constant during steady state. This confirms the DUT opens and closes TCP connections at the same rate.
- e. After ramp up the DUT MUST achieve the "Target objective" defined in the parameter Section 7.8.3.2 and remain in that state for the entire test duration (sustain phase).

7.8.3.4. Measurement

Following KPI metrics MUST be reported for each test scenario and HTTPS response object sizes separately:

average TCP connections per second and average application transaction latency or TTLB

All KPI's are measured once the target connections per second achieves the steady state.

7.8.4. Test Procedures and Expected Results

The test procedure is designed to measure average application transaction latency or TTLB when the DUT is operating close to 50% of its maximum achievable connections per second. This test procedure can be repeated multiple times with different IP types (IPv4 only, IPv6 only and IPv4 and IPv6 mixed traffic distribution), HTTPS response object sizes and single and multiple transactions per connection scenarios.

7.8.4.1. Step 1: Test Initialization and Qualification

Verify the link status of the all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure traffic load profile of the test equipment to establish "Initial objective" as defined in the parameters Section 7.8.3.2. The traffic load profile can be defined as described in Section 4.3.4.

The DUT/SUT SHOULD reach the "Initial objective" before the sustain phase. The measured KPIs during the sustain phase MUST meet the validation criteria a, b, c, d, e and f defined in Section 7.8.3.3.

If the KPI metrics do not meet the validation criteria, the test procedure MUST NOT be continued to "Step 2".

7.8.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish "Target objective" defined in the parameters table. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4.

During the ramp up and sustain phase, other KPIs such as throughput, concurrent TCP connections and application transactions per second MUST NOT reach to the maximum value that the DUT/SUT can support. The test results for specific test iterations SHOULD NOT be reported,

if the above mentioned KPI (especially throughput) reaches to the maximum value. (Example: If the test iteration with 64 KByte of HTTP response object size reached the maximum throughput limitation of the DUT, the test iteration MAY be interrupted and the result for 64 KByte SHOULD NOT be reported).

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement SHOULD be 2 seconds. Continue the test until all traffic profile phases are completed. DUT/SUT is expected to reach the desired "Target objective" at the sustain phase. In addition, the measured KPIs MUST meet all validation criteria.

Follow step 3, if the KPI metrics do not meet the validation criteria.

7.8.4.3. Step 3: Test Iteration

Determine the maximum achievable connections per second within the validation criteria and measure the latency values.

7.9. Concurrent TCP/HTTPS Connection Capacity

7.9.1. Objective

Determine the maximum number of concurrent TCP connections that the DUT/SUT sustains when using HTTPS traffic.

7.9.2. Test Setup

Test bed setup SHOULD be configured as defined in Section 4. Any specific test bed configuration changes such as number of interfaces and interface type, etc. MUST be documented.

7.9.3. Test Parameters

In this section, test scenario specific parameters SHOULD be defined.

7.9.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific test scenario MUST be documented.

7.9.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. Following parameters MUST be documented for this test scenario:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

RECOMMENDED cipher suites and key size: ECDHE-ECDSA-AES256-GCM-SHA384 with Secp521 bits key size (Signature Hash Algorithm: ecdsa_secp384r1_sha384 and Supported group: secp521r1)

Target concurrent connections: Initial value from product data sheet (if known)

Initial concurrent connections: 10% of "Target concurrent connections"

Connections per second during ramp up phase: 50% of maximum connections per second measured in test scenario TCP/HTTPS Connections per second (Section 7.6)

Ramp up time (in traffic load profile for "Target concurrent connections"): "Target concurrent connections" / "Maximum connections per second during ramp up phase"

Ramp up time (in traffic load profile for "Initial concurrent connections"): "Initial concurrent connections" / "Maximum connections per second during ramp up phase"

The client MUST perform HTTPS transaction with persistence and each client can open multiple concurrent TCP connections per server endpoint IP.

Each client sends 10 GET commands requesting 1 KByte HTTPS response objects in the same TCP connections (10 transactions/TCP connection) and the delay (think time) between each transactions MUST be X seconds.

$$X = ("Ramp\ up\ time" + "steady\ state\ time") / 10$$

The established connections SHOULD remain open until the ramp down phase of the test. During the ramp down phase, all connections SHOULD be successfully closed with FIN.

7.9.3.3. Test Results Validation Criteria

The following test Criteria is defined as test results validation criteria. Test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

- a. Number of failed Application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of total attempted transactions
- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections
- c. During the sustain phase, traffic SHOULD be forwarded constantly

7.9.3.4. Measurement

Following KPI metrics MUST be reported for this test scenario:

Average Throughput, max. Min. Avg. Concurrent TCP connections, TTLB/application transaction latency and average application transactions per second

7.9.4. Test Procedures and expected Results

The test procedure is designed to measure the concurrent TCP connection capacity of the DUT/SUT at the sustaining period of traffic load profile. The test procedure consists of three major steps. This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution.

7.9.4.1. Step 1: Test Initialization and Qualification

Verify the link status of all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure test equipment to establish "initial concurrent TCP connections" defined in Section 7.9.3.2. Except ramp up time, the traffic load profile SHOULD be defined as described in Section 4.3.4.

During the sustain phase, the DUT/SUT SHOULD reach the "Initial concurrent TCP connections". The measured KPIs during the sustain

phase MUST meet the validation criteria "a" and "b" defined in Section 7.9.3.3.

If the KPI metrics do not meet the validation criteria, the test procedure MUST NOT be continued to "Step 2".

7.9.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish "Target concurrent TCP connections". The test equipment SHOULD follow the traffic load profile definition (except ramp up time) as described in Section 4.3.4.

During the ramp up and sustain phase, the other KPIs such as throughput, TCP connections per second and application transactions per second MUST NOT reach to the maximum value that the DUT/SUT can support.

The test equipment SHOULD start to measure and record KPIs defined in Section 7.9.3.4. The frequency of measurement SHOULD be 2 seconds. Continue the test until all traffic profile phases are completed.

The DUT/SUT is expected to reach the desired target concurrent connections at the sustain phase. In addition, the measured KPIs MUST meet all validation criteria.

Follow step 3, if the KPI metrics do not meet the validation criteria.

7.9.4.3. Step 3: Test Iteration

Determine the maximum and average achievable concurrent TCP connections within the validation criteria.

8. Formal Syntax

9. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

10. Security Considerations

The primary goal of this document is to provide benchmarking terminology and methodology for next-generation network security devices. However, readers should be aware that there is some overlap

between performance and security issues. Specifically, the optimal configuration for network security device performance may not be the most secure, and vice-versa. The Cipher suites are recommended in this document are just for test purpose only. The Cipher suite recommendation for a real deployment is outside the scope of this document.

11. Acknowledgements

Acknowledgements will be added in the future release.

12. Contributors

The authors would like to thank the many people that contributed their time and knowledge to this effort.

Specifically, to the co-chairs of the NetSecOPEN Test Methodology working group and the NetSecOPEN Security Effectiveness working group – Alex Samonte, Aria Eslambolchizadeh, Carsten Rossenhoevel and David DeSanto.

Additionally, the following people provided input, comments and spent time reviewing the myriad of drafts. If we have missed anyone the fault is entirely our own. Thanks to – Amritam Putatunda, Chao Guo, Chris Chapman, Chris Pearson, Chuck McAuley, David White, Jurrie Van Den Brekel, Michelle Rhines, Rob Andrews, Samaresh Nair, and Tim Winters.

13. References

13.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [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>>.

13.2. Informative References

- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <<https://www.rfc-editor.org/info/rfc2616>>.

- [RFC2647] Newman, D., "Benchmarking Terminology for Firewall Performance", RFC 2647, DOI 10.17487/RFC2647, August 1999, <<https://www.rfc-editor.org/info/rfc2647>>.
- [RFC3511] Hickman, B., Newman, D., Tadjudin, S., and T. Martin, "Benchmarking Methodology for Firewall Performance", RFC 3511, DOI 10.17487/RFC3511, April 2003, <<https://www.rfc-editor.org/info/rfc3511>>.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <<https://www.rfc-editor.org/info/rfc5681>>.

Appendix A. NetSecOPEN Basic Traffic Mix

A traffic mix for testing performance of next generation firewalls MUST scale to stress the DUT based on real-world conditions. In order to achieve this the following MUST be included:

- o Clients connecting to multiple different server FQDNs per application
- o Clients loading apps and pages with connections and objects in specific orders
- o Multiple unique certificates for HTTPS/TLS
- o A wide variety of different object sizes
- o Different URL paths
- o Mix of HTTP and HTTPS

A traffic mix for testing performance of next generation firewalls MUST also facility application identification using different detection methods with and without decryption of the traffic. Such as:

- o HTTP HOST based application detection
- o HTTPS/TLS Server Name Indication (SNI)
- o Certificate Subject Common Name (CN)

The mix MUST be of sufficient complexity and volume to render differences in individual apps as statistically insignificant. For example, changes in like to like apps - such as one type of video service vs. another both consist of larger objects whereas one news

site vs. another both typically have more connections than other apps because of trackers and embedded advertising content. To achieve sufficient complexity, a mix MUST have:

- o Thousands of URLs each client walks thru
- o Hundreds of FQDNs each client connects to
- o Hundreds of unique certificates for HTTPS/TLS
- o Thousands of different object sizes per client in orders matching applications

The following is a description of what a popular application in an enterprise traffic mix contains.

Table 5 lists the FQDNs, number of transactions and bytes transferred as an example client interacts with Office 365 Outlook, Word, Excel, PowerPoint, SharePoint and Skype.

Office365 FQDN	Bytes	Transaction
r1.res.office365.com	14,056,960	192
s1-word-edit-15.cdn.office.net	6,731,019	22
company1-my.sharepoint.com	6,269,492	42
swx.cdn.skype.com	6,100,027	12
static.sharepointonline.com	6,036,947	41
spoprod-a.akamaihd.net	3,904,250	25
s1-excel-15.cdn.office.net	2,767,941	16
outlook.office365.com	2,047,301	86
shellprod.msocdn.com	1,008,370	11
word-edit.officeapps.live.com	932,080	25
res.delve.office.com	760,146	2
s1-powerpoint-15.cdn.office.net	557,604	3
appsforoffice.microsoft.com	511,171	5

powerpoint.officeapps.live.com	471,625	14
excel.officeapps.live.com	342,040	14
s1-officeapps-15.cdn.office.net	331,343	5
webdir0a.online.lync.com	66,930	15
portal.office.com	13,956	1
config.edge.skype.com	6,911	2
clientlog.portal.office.com	6,608	8
webdir.online.lync.com	4,343	5
graph.microsoft.com	2,289	2
nam.loki.delve.office.com	1,812	5
login.microsoftonline.com	464	2
login.windows.net	232	1

Table 5: Office365

Clients MUST connect to multiple server FQDNs in the same order as real applications. Connections MUST be made when the client is interacting with the application and MUST NOT first setup up all connections. Connections SHOULD stay open per client for subsequent transactions to the same FQDN similar to how a web browser behaves. Clients MUST use different URL Paths and Object sizes in orders as they are observed in real Applications. Clients MAY also setup multiple connections per FQDN to process multiple transactions in a sequence at the same time. Table 6 has a partial example sequence of the Office 365 Word application transactions.

FQDN	URL Path	Object size
company1-my.sharepoint.com	/personal...	23,132
word-edit.officeapps.live.com	/we/WsaUpload.ashx	2
static.sharepointonline.com	/bld/.../blank.js	454

static.sharepointonline.com	/bld/.../initstrings.js	23,254
static.sharepointonline.com	/bld/.../init.js	292,740
company1-my.sharepoint.com	/ScriptResource...	102,774
company1-my.sharepoint.com	/ScriptResource...	40,329
company1-my.sharepoint.com	/WebResource...	23,063
word-edit.officeapps.live.com	/we/wordeditorframe.aspx...	60,657
static.sharepointonline.com	/bld/_layouts/.../blank.js	454
s1-word-edit-15.cdn.office.net	/we/s/.../EditSurface.css	19,201
s1-word-edit-15.cdn.office.net	/we/s/.../WordEditor.css	221,397
s1-officeapps-15.cdn.office.net	/we/s/.../MicrosoftAjax.js	107,571
s1-word-edit-15.cdn.office.net	/we/s/.../wacbootwe.js	39,981
s1-officeapps-15.cdn.office.net	/we/s/.../CommonIntl.js	51,749
s1-word-edit-15.cdn.office.net	/we/s/.../Compat.js	6,050
s1-word-edit-15.cdn.office.net	/we/s/.../Box4Intl.js	54,158
s1-word-edit-15.cdn.office.net	/we/s/.../WoncaIntl.js	24,946
s1-word-edit-15.cdn.office.net	/we/s/.../WordEditorIntl.js	53,515
s1-word-edit-15.cdn.office.net	/we/s/.../WordEditorExp.js	1,978,712

s1-word-edit-15.cdn.office.net	/we/s/.../jSanity.js	10,912	
word-edit.officeapps.live.com	/we/OneNote.ashx	145,708	

Table 6: Office365 Word Transactions

For application identification the HTTPS/TLS traffic MUST include realistic Certificate Subject Common Name (CN) data as well as Server Name Indications (SNI). For example, a DUT MAY detect Facebook Chat traffic by inspecting the certificate and detecting *.facebook.com in the certificate subject CN and subsequently detect the word chat in the FQDN 5-edge-chat.facebook.com and identify traffic on the connection to be Facebook Chat.

Table 7 includes further examples in SNI and CN pairs for several FQDNs of Office 365.

Server Name Indication (SNI)	Certificate Subject Common Name (CN)
r1.res.office365.com	*.res.outlook.com
login.windows.net	graph.windows.net
webdir0a.online.lync.com	*.online.lync.com
login.microsoftonline.com	stamp2.login.microsoftonline.com
webdir.online.lync.com	*.online.lync.com
graph.microsoft.com	graph.microsoft.com
outlook.office365.com	outlook.com
appsforoffice.microsoft.com	appsforoffice.microsoft.com

Table 7: Office365 SNI and CN Pairs Examples

NetSecOPEN has provided a reference enterprise perimeter traffic mix with dozens of applications, hundreds of connections, and thousands of transactions.

The enterprise perimeter traffic mix consists of 70% HTTPS and 30% HTTP by Bytes, 58% HTTPS and 42% HTTP by Transactions. By

connections with a single connection per FQDN the mix consists of 43% HTTPS and 57% HTTP. With multiple connections per FQDN the HTTPS percentage is higher.

Table 8 is a summary of the NetSecOPEN enterprise perimeter traffic mix sorted by bytes with unique FQDNs and transactions per applications.

Application	FQDNs	Transactions	Bytes
Office365	26	558	52,931,947
Box	4	90	23,276,089
Salesforce	6	365	23,137,548
Gmail	13	139	16,399,289
Linkedin	10	206	15,040,918
DailyMotion	8	77	14,751,514
GoogleDocs	2	71	14,205,476
Wikia	15	159	13,909,777
Foxnews	82	499	13,758,899
Yahoo Finance	33	254	13,134,011
Youtube	8	97	13,056,216
Facebook	4	207	12,726,231
CNBC	77	275	11,939,566
Lightreading	27	304	11,200,864
BusinessInsider	16	142	11,001,575
Alexa	5	153	10,475,151
CNN	41	206	10,423,740
Twitter Video	2	72	10,112,820
Cisco Webex	1	213	9,988,417

Slack	3	40	9,938,686
Google Maps	5	191	8,771,873
SpectrumIEEE	7	145	8,682,629
Yelp	9	146	8,607,645
Vimeo	12	74	8,555,960
Wikihow	11	140	8,042,314
Netflix	3	31	7,839,256
Instagram	3	114	7,230,883
Morningstar	30	150	7,220,121
Docusign	5	68	6,972,738
Twitter	1	100	6,939,150
Tumblr	11	70	6,877,200
Whatsapp	3	46	6,829,848
Imdb	16	251	6,505,227
NOAAgov	1	44	6,316,283
IndustryWeek	23	192	6,242,403
Spotify	18	119	6,231,013
AutoNews	16	165	6,115,354
Evernote	3	47	6,063,168
NatGeo	34	104	6,026,344
BBC News	18	156	5,898,572
Investopedia	38	241	5,792,038
Pinterest	8	102	5,658,994
Succesfactors	2	112	5,049,001

AbaJournal	6	93	4,985,626
Pbworks	4	78	4,670,980
NetworkWorld	42	153	4,651,354
WebMD	24	280	4,416,736
OilGasJournal	14	105	4,095,255
Trello	5	39	4,080,182
BusinessWire	5	109	4,055,331
Dropbox	5	17	4,023,469
Nejm	20	190	4,003,657
OilGasDaily	7	199	3,970,498
Chase	6	52	3,719,232
MedicalNews	6	117	3,634,187
Marketwatch	25	142	3,291,226
Imgur	5	48	3,189,919
NPR	9	83	3,184,303
Onelogin	2	31	3,132,707
Concur	2	50	3,066,326
Service-now	1	37	2,985,329
Apple itunes	14	80	2,843,744
BerkeleyEdu	3	69	2,622,009
MSN	39	203	2,532,972
Indeed	3	47	2,325,197
MayoClinic	6	56	2,269,085
Ebay	9	164	2,219,223

UCLAedu	3	42	1,991,311
ConstructionDive	5	125	1,828,428
EducationNews	4	78	1,605,427
BofA	12	68	1,584,851
ScienceDirect	7	26	1,463,951
Reddit	8	55	1,441,909
FoodBusinessNews	5	49	1,378,298
Amex	8	42	1,270,696
Weather	4	50	1,243,826
Wikipedia	3	27	958,935
Bing	1	52	697,514
ADP	1	30	508,654
Grand Total	983	10021	569,819,095

Table 8: Summary of NetSecOPEN Enterprise Perimeter Traffic Mix

Authors' Addresses

Balamuhunthan Balarajah

Email: bm.balarajah@gmail.com

Carsten Rossenhoevel

EANTC AG

Salzufer 14

Berlin 10587

Germany

Email: cross@eantc.de

Brian Monkman
NetSecOPEN

Email: bmonkman@netsecopen.org

Benchmarking Methodology Working Group
Internet-Draft
Obsoletes: 3511 (if approved)
Intended status: Informational
Expires: 16 July 2022

B. Balarajah
C. Rossenhoevel
EANTC AG
B. Monkman
NetSecOPEN
January 2022

Benchmarking Methodology for Network Security Device Performance
draft-ietf-bmwg-ngfw-performance-13

Abstract

This document provides benchmarking terminology and methodology for next-generation network security devices including next-generation firewalls (NGFW), next-generation intrusion prevention systems (NGIPS), and unified threat management (UTM) implementations. The main areas covered in this document are test terminology, test configuration parameters, and benchmarking methodology for NGFW and NGIPS. This document aims to improve the applicability, reproducibility, and transparency of benchmarks and to align the test methodology with today's increasingly complex layer 7 security centric network application use cases. As a result, this document makes [RFC3511] obsolete.

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 5 July 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	4
2. Requirements	4
3. Scope	4
4. Test Setup	4
4.1. Testbed Configuration	5
4.2. DUT/SUT Configuration	6
4.2.1. Security Effectiveness Configuration	12
4.3. Test Equipment Configuration	12
4.3.1. Client Configuration	12
4.3.2. Backend Server Configuration	15
4.3.3. Traffic Flow Definition	17
4.3.4. Traffic Load Profile	17
5. Testbed Considerations	18
6. Reporting	19
6.1. Introduction	19
6.2. Detailed Test Results	21
6.3. Benchmarks and Key Performance Indicators	21
7. Benchmarking Tests	23
7.1. Throughput Performance with Application Traffic Mix	23
7.1.1. Objective	23
7.1.2. Test Setup	23
7.1.3. Test Parameters	23
7.1.4. Test Procedures and Expected Results	25
7.2. TCP/HTTP Connections Per Second	26
7.2.1. Objective	26
7.2.2. Test Setup	27
7.2.3. Test Parameters	27
7.2.4. Test Procedures and Expected Results	28
7.3. HTTP Throughput	30
7.3.1. Objective	30
7.3.2. Test Setup	30
7.3.3. Test Parameters	30
7.3.4. Test Procedures and Expected Results	32
7.4. HTTP Transaction Latency	33
7.4.1. Objective	33
7.4.2. Test Setup	33
7.4.3. Test Parameters	34

7.4.4. Test Procedures and Expected Results	35
7.5. Concurrent TCP/HTTP Connection Capacity	36
7.5.1. Objective	36
7.5.2. Test Setup	36
7.5.3. Test Parameters	37
7.5.4. Test Procedures and Expected Results	38
7.6. TCP/HTTPS Connections per Second	39
7.6.1. Objective	40
7.6.2. Test Setup	40
7.6.3. Test Parameters	40
7.6.4. Test Procedures and Expected Results	42
7.7. HTTPS Throughput	43
7.7.1. Objective	43
7.7.2. Test Setup	43
7.7.3. Test Parameters	43
7.7.4. Test Procedures and Expected Results	45
7.8. HTTPS Transaction Latency	46
7.8.1. Objective	46
7.8.2. Test Setup	46
7.8.3. Test Parameters	46
7.8.4. Test Procedures and Expected Results	48
7.9. Concurrent TCP/HTTPS Connection Capacity	49
7.9.1. Objective	49
7.9.2. Test Setup	49
7.9.3. Test Parameters	49
7.9.4. Test Procedures and Expected Results	51
8. IANA Considerations	52
9. Security Considerations	53
10. Contributors	53
11. Acknowledgements	53
12. References	53
12.1. Normative References	53
12.2. Informative References	53
Appendix A. Test Methodology - Security Effectiveness	
Evaluation	54
A.1. Test Objective	55
A.2. Testbed Setup	55
A.3. Test Parameters	55
A.3.1. DUT/SUT Configuration Parameters	55
A.3.2. Test Equipment Configuration Parameters	55
A.4. Test Results Validation Criteria	56
A.5. Measurement	56
A.6. Test Procedures and Expected Results	57
A.6.1. Step 1: Background Traffic	57
A.6.2. Step 2: CVE Emulation	58
Appendix B. DUT/SUT Classification	58
Authors' Addresses	58

1. Introduction

18 years have passed since IETF recommended test methodology and terminology for firewalls initially ([RFC3511]). The requirements for network security element performance and effectiveness have increased tremendously since then. In the eighteen years since [RFC3511] was published, recommending test methodology and terminology for firewalls, requirements and expectations for network security elements has increased tremendously. Security function implementations have evolved to more advanced areas and have diversified into intrusion detection and prevention, threat management, analysis of encrypted traffic, etc. In an industry of growing importance, well-defined, and reproducible key performance indicators (KPIs) are increasingly needed to enable fair and reasonable comparison of network security functions. All these reasons have led to the creation of a new next-generation network security device benchmarking document, which makes [RFC3511] obsolete.

2. Requirements

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119], [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Scope

This document provides testing terminology and testing methodology for modern and next-generation network security devices that are configured in Active ("Inline", see Figure 1 and Figure 2) mode. It covers the validation of security effectiveness configurations of network security devices, followed by performance benchmark testing. This document focuses on advanced, realistic, and reproducible testing methods. Additionally, it describes testbed environments, test tool requirements, and test result formats.

4. Test Setup

Test setup defined in this document applies to all benchmarking tests described in Section 7. The test setup MUST be contained within an Isolated Test Environment (see Section 3 of [RFC6815]).

4.1. Testbed Configuration

Testbed configuration MUST ensure that any performance implications that are discovered during the benchmark testing aren't due to the inherent physical network limitations such as the number of physical links and forwarding performance capabilities (throughput and latency) of the network devices in the testbed. For this reason, this document recommends avoiding external devices such as switches and routers in the testbed wherever possible.

In some deployment scenarios, the network security devices (Device Under Test/System Under Test) are connected to routers and switches, which will reduce the number of entries in MAC or ARP tables of the Device Under Test/System Under Test (DUT/SUT). If MAC or ARP tables have many entries, this may impact the actual DUT/SUT performance due to MAC and ARP/ND (Neighbor Discovery) table lookup processes. This document also recommends using test equipment with the capability of emulating layer 3 routing functionality instead of adding external routers in the testbed.

The testbed setup Option 1 (Figure 1) is the RECOMMENDED testbed setup for the benchmarking test.

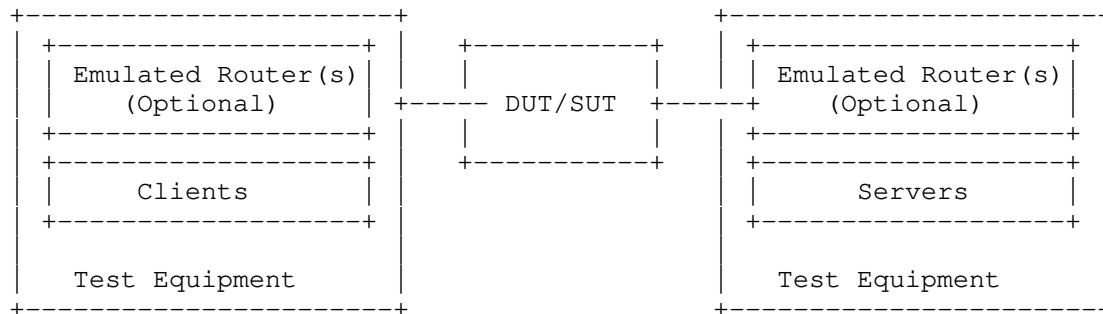


Figure 1: Testbed Setup - Option 1

If the test equipment used is not capable of emulating layer 3 routing functionality or if the number of used ports is mismatched between test equipment and the DUT/SUT (need for test equipment port aggregation), the test setup can be configured as shown in Figure 2.

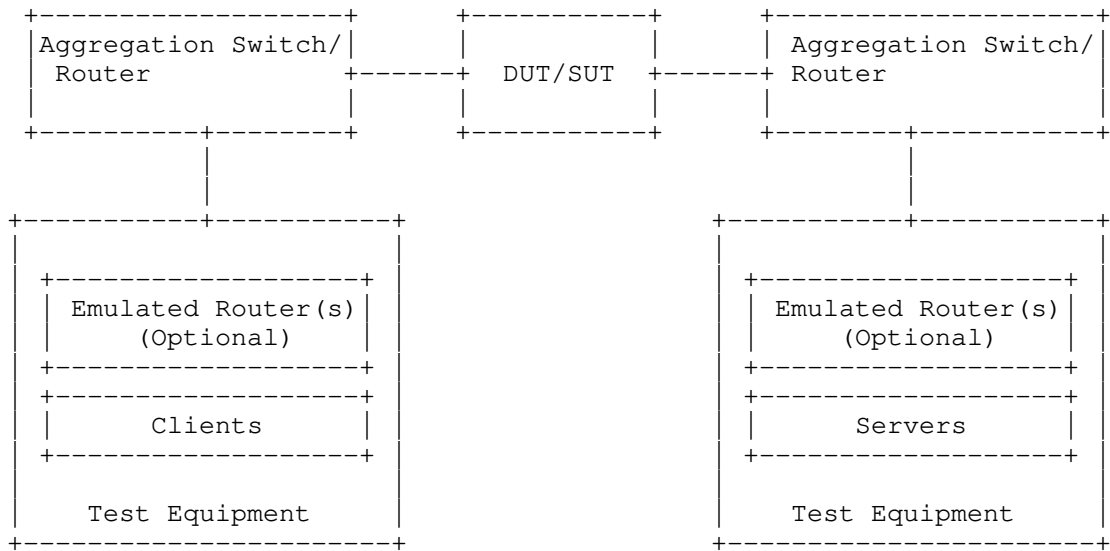


Figure 2: Testbed Setup - Option 2

4.2. DUT/SUT Configuration

A unique DUT/SUT configuration MUST be used for all benchmarking tests described in Section 7. Since each DUT/SUT will have its own unique configuration, users SHOULD configure their device with the same parameters and security features that would be used in the actual deployment of the device or a typical deployment in order to achieve maximum network security coverage. The DUT/SUT MUST be configured in "Inline" mode so that the traffic is actively inspected by the DUT/SUT. Also "Fail-Open" behavior MUST be disabled on the DUT/SUT.

Table 1 and Table 2 below describe the RECOMMENDED and OPTIONAL sets of network security feature list for NGFW and NGIPS respectively. The selected security features SHOULD be consistently enabled on the DUT/SUT for all benchmarking tests described in Section 7.

To improve repeatability, a summary of the DUT/SUT configuration including a description of all enabled DUT/SUT features MUST be published with the benchmarking results.

DUT/SUT (NGFW) Features	RECOMMENDED	OPTIONAL
SSL Inspection	x	
IDS/IPS	x	
Anti-Spyware	x	
Anti-Virus	x	
Anti-Botnet	x	
Web Filtering		x
Data Loss Protection (DLP)		x
DDoS		x
Certificate Validation		x
Logging and Reporting	x	
Application Identification	x	

Table 1: NGFW Security Features

DUT/SUT (NGIPS) Features	RECOMMENDED	OPTIONAL
SSL Inspection	x	
Anti-Malware	x	
Anti-Spyware	x	
Anti-Botnet	x	
Logging and Reporting	x	
Application Identification	x	
Deep Packet Inspection	x	
Anti-Evasion	x	

Table 2: NGIPS Security Features

The following table provides a brief description of the security features.

DUT/SUT Features	Description
SSL Inspection	DUT/SUT intercepts and decrypts inbound HTTPS traffic between servers and clients. Once the content inspection has been completed, DUT/SUT encrypts the HTTPS traffic with ciphers and keys used by the clients and servers.
IDS/IPS	DUT/SUT detects and blocks exploits targeting known and unknown vulnerabilities across the monitored network.
Anti-Malware	DUT/SUT detects and prevents the transmission of malicious executable code and any associated communications across the monitored network. This includes data exfiltration as well as command and control channels.
Anti-Spyware	Anti-Spyware is a subcategory of Anti Malware. Spyware transmits information without the user's knowledge or permission. DUT/SUT

	detects and block initial infection or transmission of data.
Anti-Botnet	DUT/SUT detects traffic to or from botnets.
Anti-Evasion	DUT/SUT detects and mitigates attacks that have been obfuscated in some manner.
Web Filtering	DUT/SUT detects and blocks malicious website including defined classifications of website across the monitored network.
DLP	DUT/SUT detects and prevents data breaches and data exfiltration, or it detects and blocks the transmission of sensitive data across the monitored network.
Certificate Validation	DUT/SUT validates certificates used in encrypted communications across the monitored network.
Logging and Reporting	DUT/SUT logs and reports all traffic at the flow level across the monitored network.
Application Identification	DUT/SUT detects known applications as defined within the traffic mix selected across the monitored network.

Table 3: Security Feature Description

Below is a summary of the DUT/SUT configuration:

- * DUT/SUT MUST be configured in "inline" mode.
- * "Fail-Open" behavior MUST be disabled.
- * All RECOMMENDED security features are enabled.
- * Logging SHOULD be enabled. DUT/SUT SHOULD log all traffic at the flow level - Logging to an external device is permissible.
- * Geographical location filtering, and Application Identification and Control SHOULD be configured to trigger based on a site or application from the defined traffic mix.

In addition, a realistic number of access control rules (ACL) SHOULD be configured on the DUT/SUT where ACLs are configurable and reasonable based on the deployment scenario. This document determines the number of access policy rules for four different classes of DUT/SUT: Extra Small (XS), Small (S), Medium (M), and Large (L). A sample DUT/SUT classification is described in Appendix B.

The Access Control Rules (ACL) defined in Figure 3 MUST be configured from top to bottom in the correct order as shown in the table. This is due to ACL types listed in specificity decreasing order, with "block" first, followed by "allow", representing a typical ACL based security policy. The ACL entries SHOULD be configured with routable IP subnets by the DUT/SUT. (Note: There will be differences between how security vendors implement ACL decision making.) The configured ACL MUST NOT block the security and measurement traffic used for the benchmarking tests.

				DUT/SUT Classification # Rules			
Rules Type	Match Criteria	Description	Action	XS	S	M	L
Application layer	Application	Any application not included in the measurement traffic	block	5	10	20	50
Transport layer	SRC IP and TCP/UDP DST ports	Any SRC IP subnet used and any DST ports not used in the measurement traffic	block	25	50	100	250
IP layer	SRC/DST IP	Any SRC/DST IP subnet not used in the measurement traffic	block	25	50	100	250
Application layer	Application	Half of the applications included in the measurement traffic (see the note below)	allow	10	10	10	10
Transport layer	SRC IP and TCP/UDP DST ports	Half of the SRC IPs used and any DST ports used in the measurement traffic (one rule per subnet)	allow	>1	>1	>1	>1
IP layer	SRC IP	The rest of the SRC IP subnet range used in the measurement traffic (one rule per subnet)	allow	>1	>1	>1	>1

Figure 3: DUT/SUT Access List

Note: If half of the applications included in the measurement traffic is less than 10, the missing number of ACL entries (dummy rules) can be configured for any application traffic not included in the measurement traffic.

4.2.1. Security Effectiveness Configuration

The Security features (defined in Table 1 and Table 2) of the DUT/SUT MUST be configured effectively to detect, prevent, and report the defined security vulnerability sets. This section defines the selection of the security vulnerability sets from Common vulnerabilities and Exposures (CVE) list for the testing. The vulnerability set SHOULD reflect a minimum of 500 CVEs from no older than 10 calendar years to the current year. These CVEs SHOULD be selected with a focus on in-use software commonly found in business applications, with a Common vulnerability Scoring System (CVSS) Severity of High (7-10).

This document is primarily focused on performance benchmarking. However, it is RECOMMENDED to validate the security features configuration of the DUT/SUT by evaluating the security effectiveness as a prerequisite for performance benchmarking tests defined in the section 7. In case the benchmarking tests are performed without evaluating security effectiveness, the test report MUST explain the implications of this. The methodology for evaluating security effectiveness is defined in Appendix A.

4.3. Test Equipment Configuration

In general, test equipment allows configuring parameters in different protocol layers. These parameters thereby influence the traffic flows which will be offered and impact performance measurements.

This section specifies common test equipment configuration parameters applicable for all benchmarking tests defined in Section 7. Any benchmarking test specific parameters are described under the test setup section of each benchmarking test individually.

4.3.1. Client Configuration

This section specifies which parameters SHOULD be considered while configuring clients using test equipment. Also, this section specifies the RECOMMENDED values for certain parameters. The values are the defaults used in most of the client operating systems currently.

4.3.1.1. TCP Stack Attributes

The TCP stack SHOULD use a congestion control algorithm at client and server endpoints. The IPv4 and IPv6 Maximum Segment Size (MSS) SHOULD be set to 1460 bytes and 1440 bytes respectively and a TX and RX initial receive windows of 64 KByte. Client initial congestion window SHOULD NOT exceed 10 times the MSS. Delayed ACKs are permitted and the maximum client delayed ACK SHOULD NOT exceed 10 times the MSS before a forced ACK. Up to three retries SHOULD be allowed before a timeout event is declared. All traffic MUST set the TCP PSH flag to high. The source port range SHOULD be in the range of 1024 - 65535. Internal timeout SHOULD be dynamically scalable per RFC 793. The client SHOULD initiate and close TCP connections. The TCP connection MUST be initiated via a TCP three-way handshake (SYN, SYN/ACK, ACK), and it MUST be closed via either a TCP three-way close (FIN, FIN/ACK, ACK), or a TCP four-way close (FIN, ACK, FIN, ACK).

4.3.1.2. Client IP Address Space

The sum of the client IP space SHOULD contain the following attributes.

- * The IP blocks SHOULD consist of multiple unique, discontinuous static address blocks.
- * A default gateway is permitted.
- * The DSCP (differentiated services code point) marking is set to DF (Default Forwarding) '000000' on IPv4 Type of Service (ToS) field and IPv6 traffic class field.

The following equation can be used to define the total number of client IP addresses that will be configured on the test equipment.

Desired total number of client IP = Target throughput [Mbit/s] /
Average throughput per IP address [Mbit/s]

As shown in the example list below, the value for "Average throughput per IP address" can be varied depending on the deployment and use case scenario.

- (Option 1) DUT/SUT deployment scenario 1 : 6-7 Mbit/s per IP (e.g. 1,400-1,700 IPs per 10Gbit/s throughput)
- (Option 2) DUT/SUT deployment scenario 2 : 0.1-0.2 Mbit/s per IP (e.g. 50,000-100,000 IPs per 10Gbit/s throughput)

Based on deployment and use case scenario, client IP addresses SHOULD be distributed between IPv4 and IPv6. The following options MAY be considered for a selection of traffic mix ratio.

(Option 1) 100 % IPv4, no IPv6

(Option 2) 80 % IPv4, 20% IPv6

(Option 3) 50 % IPv4, 50% IPv6

(Option 4) 20 % IPv4, 80% IPv6

(Option 5) no IPv4, 100% IPv6

Note: The IANA has assigned IP address range for the testing purpose as described in Section 8. If the test scenario requires more IP addresses or subnets than the IANA assigned, this document recommends using non routable Private IPv4 address ranges or Unique Local Address (ULA) IPv6 address ranges for the testing.

4.3.1.3. Emulated Web Browser Attributes

The client emulated web browser (emulated browser) contains attributes that will materially affect how traffic is loaded. The objective is to emulate modern, typical browser attributes to improve realism of the result set.

For HTTP traffic emulation, the emulated browser MUST negotiate HTTP version 1.1 or higher. Depending on test scenarios and chosen HTTP version, the emulated browser MAY open multiple TCP connections per Server endpoint IP at any time depending on how many sequential transactions need to be processed. For HTTP/2 or HTTP/3, the emulated browser MAY open multiple concurrent streams per connection (multiplexing). HTTP/3 emulated browser uses QUIC ([RFC9000]) as transport protocol. HTTP settings such as number of connection per server IP, number of requests per connection, and number of streams per connection MUST be documented. This document refers to [RFC8446] for HTTP/2. The emulated browser SHOULD advertise a User-Agent header. The emulated browser SHOULD enforce content length validation. Depending on test scenarios and selected HTTP version, HTTP header compression MAY be set to enable or disable. This setting (compression enabled or disabled) MUST be documented in the report.

For encrypted traffic, the following attributes SHALL define the negotiated encryption parameters. The test clients MUST use TLS version 1.2 or higher. TLS record size MAY be optimized for the HTTPS response object size up to a record size of 16 KByte. If

Server Name Indication (SNI) is required in the traffic mix profile, the client endpoint MUST send TLS extension Server Name Indication (SNI) information when opening a security tunnel. Each client connection MUST perform a full handshake with server certificate and MUST NOT use session reuse or resumption.

The following TLS 1.2 supported ciphers and keys are RECOMMENDED to use for HTTPS based benchmarking tests defined in Section 7.

1. ECDHE-ECDSA-AES128-GCM-SHA256 with Prime256v1 (Signature Hash Algorithm: `ecdsa_secp256r1_sha256` and Supported group: `secp256r1`)
2. ECDHE-RSA-AES128-GCM-SHA256 with RSA 2048 (Signature Hash Algorithm: `rsa_pkcs1_sha256` and Supported group: `secp256r1`)
3. ECDHE-ECDSA-AES256-GCM-SHA384 with Secp521 (Signature Hash Algorithm: `ecdsa_secp384r1_sha384` and Supported group: `secp521r1`)
4. ECDHE-RSA-AES256-GCM-SHA384 with RSA 4096 (Signature Hash Algorithm: `rsa_pkcs1_sha384` and Supported group: `secp256r1`)

Note: The above ciphers and keys were those commonly used enterprise grade encryption cipher suites for TLS 1.2. It is recognized that these will evolve over time. Individual certification bodies SHOULD use ciphers and keys that reflect evolving use cases. These choices MUST be documented in the resulting test reports with detailed information on the ciphers and keys used along with reasons for the choices.

[RFC8446] defines the following cipher suites for use with TLS 1.3.

1. TLS_AES_128_GCM_SHA256
2. TLS_AES_256_GCM_SHA384
3. TLS_CHACHA20_POLY1305_SHA256
4. TLS_AES_128_CCM_SHA256
5. TLS_AES_128_CCM_8_SHA256

4.3.2. Backend Server Configuration

This section specifies which parameters should be considered while configuring emulated backend servers using test equipment.

4.3.2.1. TCP Stack Attributes

The TCP stack on the server side SHOULD be configured similar to the client side configuration described in Section 4.3.1.1. In addition, server initial congestion window MUST NOT exceed 10 times the MSS. Delayed ACKs are permitted and the maximum server delayed ACK MUST NOT exceed 10 times the MSS before a forced ACK.

4.3.2.2. Server Endpoint IP Addressing

The sum of the server IP space SHOULD contain the following attributes.

- * The server IP blocks SHOULD consist of unique, discontinuous static address blocks with one IP per server Fully Qualified Domain Name (FQDN) endpoint per test port.
- * A default gateway is permitted. The DSCP (differentiated services code point) marking is set to DF (Default Forwarding) '000000' on IPv4 Type of Service (ToS) field and IPv6 traffic class field.
- * The server IP addresses SHOULD be distributed between IPv4 and IPv6 with a ratio identical to the clients distribution ratio.

Note: The IANA has assigned IP address range for the testing purpose as described in Section 8. If the test scenario requires more IP addresses or subnets than the IANA assigned, this document recommends using non routable Private IPv4 address ranges or Unique Local Address (ULA) IPv6 address ranges for the testing.

4.3.2.3. HTTP / HTTPS Server Pool Endpoint Attributes

The server pool for HTTP SHOULD listen on TCP port 80 and emulate the same HTTP version (HTTP 1.1 or HTTP/2 or HTTP/3) and settings chosen by the client (emulated web browser). The Server MUST advertise server type in the Server response header [RFC7230]. For HTTPS server, TLS 1.2 or higher MUST be used with a maximum record size of 16 KByte and MUST NOT use ticket resumption or session ID reuse. The server SHOULD listen on TCP port 443 for HTTP version 1.1 and 2. For HTTP/3 (HTTP over QUIC) the server SHOULD listen on UDP 443. The server SHALL serve a certificate to the client. The HTTPS server MUST check host SNI information with the FQDN if SNI is in use. Cipher suite and key size on the server side MUST be configured similar to the client side configuration described in Section 4.3.1.3.

4.3.3. Traffic Flow Definition

This section describes the traffic pattern between client and server endpoints. At the beginning of the test, the server endpoint initializes and will be ready to accept connection states including initialization of the TCP stack as well as bound HTTP and HTTPS servers. When a client endpoint is needed, it will initialize and be given attributes such as a MAC and IP address. The behavior of the client is to sweep through the given server IP space, generating a recognizable service by the DUT. Sequential and pseudorandom sweep methods are acceptable. The method used MUST be stated in the final report. Thus, a balanced mesh between client endpoints and server endpoints will be generated in a client IP and port to server IP and port combination. Each client endpoint performs the same actions as other endpoints, with the difference being the source IP of the client endpoint and the target server IP pool. The client MUST use the server IP address or FQDN in the host header [RFC7230].

4.3.3.1. Description of Intra-Client Behavior

Client endpoints are independent of other clients that are concurrently executing. When a client endpoint initiates traffic, this section describes how the client steps through different services. Once the test is initialized, the client endpoints randomly hold (perform no operation) for a few milliseconds for better randomization of the start of client traffic. Each client will either open a new TCP connection or connect to a TCP persistence stack still open to that specific server. At any point that the traffic profile may require encryption, a TLS encryption tunnel will form presenting the URL or IP address request to the server. If using SNI, the server MUST then perform an SNI name check with the proposed FQDN compared to the domain embedded in the certificate. Only when correct, will the server process the HTTPS response object. The initial response object to the server is based on benchmarking tests described in Section 7. Multiple additional sub-URLs (response objects on the service page) MAY be requested simultaneously. This MAY be to the same server IP as the initial URL. Each sub-object will also use a canonical FQDN and URL path, as observed in the traffic mix used.

4.3.4. Traffic Load Profile

The loading of traffic is described in this section. The loading of a traffic load profile has five phases: Init, ramp up, sustain, ramp down, and collection.

1. Init phase: Testbed devices including the client and server endpoints should negotiate layer 2-3 connectivity such as MAC learning and ARP. Only after successful MAC learning or ARP/ND resolution SHALL the test iteration move to the next phase. No measurements are made in this phase. The minimum RECOMMENDED time for Init phase is 5 seconds. During this phase, the emulated clients SHOULD NOT initiate any sessions with the DUT/SUT, in contrast, the emulated servers should be ready to accept requests from DUT/SUT or from emulated clients.
 2. Ramp up phase: The test equipment SHOULD start to generate the test traffic. It SHOULD use a set of the approximate number of unique client IP addresses to generate traffic. The traffic SHOULD ramp up from zero to desired target objective. The target objective is defined for each benchmarking test. The duration for the ramp up phase MUST be configured long enough that the test equipment does not overwhelm the DUT/SUTs stated performance metrics defined in Section 6.3 namely, TCP Connections Per Second, Inspected Throughput, Concurrent TCP Connections, and Application Transactions Per Second. No measurements are made in this phase.
 3. Sustain phase: Starts when all required clients are active and operating at their desired load condition. In the sustain phase, the test equipment SHOULD continue generating traffic to constant target value for a constant number of active clients. The minimum RECOMMENDED time duration for sustain phase is 300 seconds. This is the phase where measurements occur. The test equipment SHOULD measure and record statistics continuously. The sampling interval for collecting the raw results and calculating the statistics SHOULD be less than 2 seconds.
 4. Ramp down phase: No new connections are established, and no measurements are made. The time duration for ramp up and ramp down phase SHOULD be the same.
 5. Collection phase: The last phase is administrative and will occur when the test equipment merges and collates the report data.
5. Testbed Considerations

This section describes steps for a reference test (pre-test) that control the test environment including test equipment, focusing on physical and virtualized environments and as well as test equipments. Below are the RECOMMENDED steps for the reference test.

1. Perform the reference test either by configuring the DUT/SUT in the most trivial setup (fast forwarding) or without presence of the DUT/SUT.
2. Generate traffic from traffic generator. Choose a traffic profile used for HTTP or HTTPS throughput performance test with smallest object size.
3. Ensure that any ancillary switching or routing functions added in the test equipment does not limit the performance by introducing network metrics such as packet loss and latency. This is specifically important for virtualized components (e.g., vSwitches, vRouters).
4. Verify that the generated traffic (performance) of the test equipment matches and reasonably exceeds the expected maximum performance of the DUT/SUT.
5. Record the network performance metrics packet loss latency introduced by the test environment (without DUT/SUT).
6. Assert that the testbed characteristics are stable during the entire test session. Several factors might influence stability specifically, for virtualized testbeds. For example, additional workloads in a virtualized system, load balancing, and movement of virtual machines during the test, or simple issues such as additional heat created by high workloads leading to an emergency CPU performance reduction.

The reference test SHOULD be performed before the benchmarking tests (described in section 7) start.

6. Reporting

This section describes how the benchmarking test report should be formatted and presented. It is RECOMMENDED to include two main sections in the report, namely the introduction and the detailed test results sections.

6.1. Introduction

The following attributes SHOULD be present in the introduction section of the test report.

1. The time and date of the execution of the tests
2. Summary of testbed software and hardware details

a. DUT/SUT hardware/virtual configuration

- * This section SHOULD clearly identify the make and model of the DUT/SUT
- * The port interfaces, including speed and link information
- * If the DUT/SUT is a Virtual Network Function (VNF), host (server) hardware and software details, interface acceleration type such as DPDK and SR-IOV, used CPU cores, used RAM, resource sharing (e.g. Pinning details and NUMA Node) configuration details, hypervisor version, virtual switch version
- * details of any additional hardware relevant to the DUT/SUT such as controllers

b. DUT/SUT software

- * Operating system name
- * Version
- * Specific configuration details (if any)

c. DUT/SUT enabled features

- * Configured DUT/SUT features (see Table 1 and Table 2)
- * Attributes of the above-mentioned features
- * Any additional relevant information about the features

d. Test equipment hardware and software

- * Test equipment vendor name
- * Hardware details including model number, interface type
- * Test equipment firmware and test application software version

e. Key test parameters

- * Used cipher suites and keys
- * IPv4 and IPv6 traffic distribution

- * Number of configured ACL
- f. Details of application traffic mix used in the benchmarking test "Throughput Performance with Application Traffic Mix" (Section 7.1)
 - * Name of applications and layer 7 protocols
 - * Percentage of emulated traffic for each application and layer 7 protocols
 - * Percentage of encrypted traffic and used cipher suites and keys (The RECOMMENDED ciphers and keys are defined in Section 4.3.1.3)
 - * Used object sizes for each application and layer 7 protocols

3. Results Summary / Executive Summary

- a. Results SHOULD resemble a pyramid in how it is reported, with the introduction section documenting the summary of results in a prominent, easy to read block.

6.2. Detailed Test Results

In the result section of the test report, the following attributes SHOULD be present for each benchmarking test.

- a. KPIs MUST be documented separately for each benchmarking test. The format of the KPI metrics SHOULD be presented as described in Section 6.3.
- b. The next level of details SHOULD be graphs showing each of these metrics over the duration (sustain phase) of the test. This allows the user to see the measured performance stability changes over time.

6.3. Benchmarks and Key Performance Indicators

This section lists key performance indicators (KPIs) for overall benchmarking tests. All KPIs MUST be measured during the sustain phase of the traffic load profile described in Section 4.3.4. All KPIs MUST be measured from the result output of test equipment.

- * Concurrent TCP Connections

The aggregate number of simultaneous connections between hosts across the DUT/SUT, or between hosts and the DUT/SUT (defined in [RFC2647]).

* TCP Connections Per Second

The average number of successfully established TCP connections per second between hosts across the DUT/SUT, or between hosts and the DUT/SUT. The TCP connection MUST be initiated via a TCP three-way handshake (SYN, SYN/ACK, ACK). Then the TCP session data is sent. The TCP session MUST be closed via either a TCP three-way close (FIN, FIN/ACK, ACK), or a TCP four-way close (FIN, ACK, FIN, ACK), and MUST NOT by RST.

* Application Transactions Per Second

The average number of successfully completed transactions per second. For a particular transaction to be considered successful, all data MUST have been transferred in its entirety. In case of HTTP(S) transactions, it MUST have a valid status code (200 OK), and the appropriate FIN, FIN/ACK sequence MUST have been completed.

* TLS Handshake Rate

The average number of successfully established TLS connections per second between hosts across the DUT/SUT, or between hosts and the DUT/SUT.

* Inspected Throughput

The number of bits per second of examined and allowed traffic a network security device is able to transmit to the correct destination interface(s) in response to a specified offered load. The throughput benchmarking tests defined in Section 7 SHOULD measure the average Layer 2 throughput value when the DUT/SUT is "inspecting" traffic. This document recommends presenting the inspected throughput value in Gbit/s rounded to two places of precision with a more specific Kbit/s in parenthesis.

* Time to First Byte (TTFB)

TTFB is the elapsed time between the start of sending the TCP SYN packet from the client and the client receiving the first packet of application data from the server or DUT/SUT. The benchmarking tests HTTP Transaction Latency (Section 7.4) and HTTPS Transaction Latency (Section 7.8) measure the minimum, average and maximum TTFB. The value SHOULD be expressed in milliseconds.

* URL Response time / Time to Last Byte (TTLB)

URL Response time / TTLB is the elapsed time between the start of sending the TCP SYN packet from the client and the client receiving the last packet of application data from the server or DUT/SUT. The benchmarking tests HTTP Transaction Latency (Section 7.4) and HTTPS Transaction Latency (Section 7.8) measure the minimum, average and maximum TTLB. The value SHOULD be expressed in millisecond.

7. Benchmarking Tests

7.1. Throughput Performance with Application Traffic Mix

7.1.1. Objective

Using a relevant application traffic mix, determine the sustainable inspected throughput supported by the DUT/SUT.

Based on the test customer's specific use case, testers can choose the relevant application traffic mix for this test. The details about the traffic mix MUST be documented in the report. At least the following traffic mix details MUST be documented and reported together with the test results:

Name of applications and layer 7 protocols

Percentage of emulated traffic for each application and layer 7 protocol

Percentage of encrypted traffic and used cipher suites and keys (The RECOMMENDED ciphers and keys are defined in Section 4.3.1.3.)

Used object sizes for each application and layer 7 protocols

7.1.2. Test Setup

Testbed setup MUST be configured as defined in Section 4. Any benchmarking test specific testbed configuration changes MUST be documented.

7.1.3. Test Parameters

In this section, the benchmarking test specific parameters SHOULD be defined.

7.1.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific benchmarking test MUST be documented. In case the DUT/SUT is configured without SSL inspection, the test report MUST explain the implications of this to the relevant application traffic mix encrypted traffic.

7.1.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. The following parameters MUST be documented for this benchmarking test:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target inspected throughput: Aggregated line rate of interface(s) used in the DUT/SUT or the value defined based on requirement for a specific deployment scenario

Initial throughput: 10% of the "Target inspected throughput" Note: Initial throughput is not a KPI to report. This value is configured on the traffic generator and used to perform Step 1: "Test Initialization and Qualification" described under the Section 7.1.4.

One of the ciphers and keys defined in Section 4.3.1.3 are RECOMMENDED to use for this benchmarking test.

7.1.3.3. Traffic Profile

Traffic profile: This test MUST be run with a relevant application traffic mix profile.

7.1.3.4. Test Results Validation Criteria

The following criteria are the test results validation criteria. The test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

- a. Number of failed application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of total attempted transactions.
- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.

7.1.3.5. Measurement

Following KPI metrics MUST be reported for this benchmarking test:

Mandatory KPIs (benchmarks): Inspected Throughput, TTFB (minimum, average, and maximum), TTLB (minimum, average, and maximum) and Application Transactions Per Second

Note: TTLB MUST be reported along with the object size used in the traffic profile.

Optional KPIs: TCP Connections Per Second and TLS Handshake Rate

7.1.4. Test Procedures and Expected Results

The test procedures are designed to measure the inspected throughput performance of the DUT/SUT at the sustaining period of traffic load profile. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (initial throughput) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IP types: IPv4 only, IPv6 only, and IPv4 and IPv6 mixed traffic distribution.

7.1.4.1. Step 1: Test Initialization and Qualification

Verify the link status of all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure traffic load profile of the test equipment to generate test traffic at the "Initial throughput" rate as described in Section 7.1.3.2. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4. The DUT/SUT SHOULD reach the "Initial throughput" during the sustain phase. Measure all KPI as defined in Section 7.1.3.5. The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Section 7.1.3.4.

If the KPI metrics do not meet the test results validation criteria, the test procedure MUST NOT be continued to step 2.

7.1.4.2. Step 2: Test Run with Target Objective

Configure test equipment to generate traffic at the "Target inspected throughput" rate defined in Section 7.1.3.2. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4. The test equipment SHOULD start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective ("Target inspected throughput") in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.1.4.3. Step 3: Test Iteration

Determine the achievable average inspected throughput within the test results validation criteria. Final test iteration MUST be performed for the test duration defined in Section 4.3.4.

7.2. TCP/HTTP Connections Per Second

7.2.1. Objective

Using HTTP traffic, determine the sustainable TCP connection establishment rate supported by the DUT/SUT under different throughput load conditions.

To measure connections per second, test iterations MUST use different fixed HTTP response object sizes (the different load conditions) defined in Section 7.2.3.2.

7.2.2. Test Setup

Testbed setup SHOULD be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.2.3. Test Parameters

In this section, benchmarking test specific parameters SHOULD be defined.

7.2.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific benchmarking test MUST be documented.

7.2.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. The following parameters MUST be documented for this benchmarking test:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target connections per second: Initial value from product datasheet or the value defined based on requirement for a specific deployment scenario

Initial connections per second: 10% of "Target connections per second" (Note: Initial connections per second is not a KPI to report. This value is configured on the traffic generator and used to perform the Step1: "Test Initialization and Qualification" described under the Section 7.2.4.

The client SHOULD negotiate HTTP and close the connection with FIN immediately after completion of one transaction. In each test iteration, client MUST send GET request requesting a fixed HTTP response object size.

The RECOMMENDED response object sizes are 1, 2, 4, 16, and 64 KByte.

7.2.3.3. Test Results Validation Criteria

The following criteria are the test results validation criteria. The Test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

- a. Number of failed application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of total attempted transactions.
- b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.
- c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).
- d. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.

7.2.3.4. Measurement

TCP Connections Per Second MUST be reported for each test iteration (for each object size).

7.2.4. Test Procedures and Expected Results

The test procedure is designed to measure the TCP connections per second rate of the DUT/SUT at the sustaining period of the traffic load profile. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial connections per second) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IP types: IPv4 only, IPv6 only, and IPv4 and IPv6 mixed traffic distribution.

7.2.4.1. Step 1: Test Initialization and Qualification

Verify the link status of all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure the traffic load profile of the test equipment to establish "Initial connections per second" as defined in Section 7.2.3.2. The traffic load profile SHOULD be defined as described in Section 4.3.4.

The DUT/SUT SHOULD reach the "Initial connections per second" before the sustain phase. The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Section 7.2.3.3.

If the KPI metrics do not meet the test results validation criteria, the test procedure MUST NOT continue to "Step 2".

7.2.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish the target objective ("Target connections per second") defined in Section 7.2.3.2. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4.

During the ramp up and sustain phase of each test iteration, other KPIs such as inspected throughput, concurrent TCP connections and application transactions per second MUST NOT reach the maximum value the DUT/SUT can support. The test results for specific test iterations SHOULD NOT be reported, if the above-mentioned KPI (especially inspected throughput) reaches the maximum value. (Example: If the test iteration with 64 KByte of HTTP response object size reached the maximum inspected throughput limitation of the DUT/SUT, the test iteration MAY be interrupted and the result for 64 KByte SHOULD NOT be reported.)

The test equipment SHOULD start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective ("Target connections per second") in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.2.4.3. Step 3: Test Iteration

Determine the achievable TCP connections per second within the test results validation criteria.

7.3. HTTP Throughput

7.3.1. Objective

Determine the sustainable inspected throughput of the DUT/SUT for HTTP transactions varying the HTTP response object size.

7.3.2. Test Setup

Testbed setup SHOULD be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.3.3. Test Parameters

In this section, benchmarking test specific parameters SHOULD be defined.

7.3.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific benchmarking test MUST be documented.

7.3.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. The following parameters MUST be documented for this benchmarking test:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target inspected throughput: Aggregated line rate of interface(s) used in the DUT/SUT or the value defined based on requirement for a specific deployment scenario

Initial throughput: 10% of "Target inspected throughput" Note: Initial throughput is not a KPI to report. This value is configured on the traffic generator and used to perform Step 1: "Test Initialization and Qualification" described under Section 7.3.4.

Number of HTTP response object requests (transactions) per connection: 10

RECOMMENDED HTTP response object size: 1, 16, 64, 256 KByte, and mixed objects defined in Table 4.

Object size (KByte)	Number of requests/ Weight
0.2	1
6	1
8	1
9	1
10	1
25	1
26	1
35	1
59	1
347	1

Table 4: Mixed Objects

7.3.3.3. Test Results Validation Criteria

The following criteria are the test results validation criteria. The test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

- a. Number of failed application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of attempt transactions.

- b. Traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).
- c. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.

7.3.3.4. Measurement

Inspected Throughput and HTTP Transactions per Second MUST be reported for each object size.

7.3.4. Test Procedures and Expected Results

The test procedure is designed to measure HTTP throughput of the DUT/SUT. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial throughput) and meets the test results validation criteria when it was very minimal utilized. Step 2 determines the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution and HTTP response object sizes.

7.3.4.1. Step 1: Test Initialization and Qualification

Verify the link status of all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure traffic load profile of the test equipment to establish "Initial inspected throughput" as defined in Section 7.3.3.2.

The traffic load profile SHOULD be defined as described in Section 4.3.4. The DUT/SUT SHOULD reach the "Initial inspected throughput" during the sustain phase. Measure all KPI as defined in Section 7.3.3.4.

The measured KPIs during the sustain phase MUST meet the test results validation criteria "a" defined in Section 7.3.3.3. The test results validation criteria "b" and "c" are OPTIONAL for step 1.

If the KPI metrics do not meet the test results validation criteria, the test procedure MUST NOT be continued to "Step 2".

7.3.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish the target objective ("Target inspected throughput") defined in Section 7.3.3.2. The test equipment SHOULD start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.3.4.3. Step 3: Test Iteration

Determine the achievable inspected throughput within the test results validation criteria and measure the KPI metric Transactions per Second. Final test iteration MUST be performed for the test duration defined in Section 4.3.4.

7.4. HTTP Transaction Latency

7.4.1. Objective

Using HTTP traffic, determine the HTTP transaction latency when DUT is running with sustainable HTTP transactions per second supported by the DUT/SUT under different HTTP response object sizes.

Test iterations MUST be performed with different HTTP response object sizes in two different scenarios. One with a single transaction and the other with multiple transactions within a single TCP connection. For consistency both the single and multiple transaction test MUST be configured with the same HTTP version

Scenario 1: The client MUST negotiate HTTP and close the connection with FIN immediately after completion of a single transaction (GET and RESPONSE).

Scenario 2: The client MUST negotiate HTTP and close the connection FIN immediately after completion of 10 transactions (GET and RESPONSE) within a single TCP connection.

7.4.2. Test Setup

Testbed setup SHOULD be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.4.3. Test Parameters

In this section, benchmarking test specific parameters SHOULD be defined.

7.4.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific benchmarking test MUST be documented.

7.4.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. The following parameters MUST be documented for this benchmarking test:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target objective for scenario 1: 50% of the connections per second measured in benchmarking test TCP/HTTP Connections Per Second (Section 7.2)

Target objective for scenario 2: 50% of the inspected throughput measured in benchmarking test HTTP Throughput (Section 7.3)

Initial objective for scenario 1: 10% of "Target objective for scenario 1"

Initial objective for scenario 2: 10% of "Target objective for scenario 2"

Note: The Initial objectives are not a KPI to report. These values are configured on the traffic generator and used to perform the Step1: "Test Initialization and Qualification" described under the Section 7.4.4.

HTTP transaction per TCP connection: Test scenario 1 with single transaction and test scenario 2 with 10 transactions.

HTTP with GET request requesting a single object. The RECOMMENDED object sizes are 1, 16, and 64 KByte. For each test iteration, client MUST request a single HTTP response object size.

7.4.3.3. Test Results Validation Criteria

The following criteria are the test results validation criteria. The Test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

- a. Number of failed application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of attempt transactions.
- b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.
- c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).
- d. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.
- e. After ramp up the DUT MUST achieve the "Target objective" defined in Section 7.4.3.2 and remain in that state for the entire test duration (sustain phase).

7.4.3.4. Measurement

TTFB (minimum, average, and maximum) and TTLB (minimum, average and maximum) MUST be reported for each object size.

7.4.4. Test Procedures and Expected Results

The test procedure is designed to measure TTFB or TTLB when the DUT/SUT is operating close to 50% of its maximum achievable connections per second or inspected throughput. The test procedure consists of two major steps: Step 1 ensures the DUT/SUT is able to reach the initial performance values and meets the test results validation criteria when it was very minimally utilized. Step 2 measures the latency values within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IP types (IPv4 only, IPv6 only and IPv4 and IPv6 mixed traffic distribution), HTTP response object sizes and single and multiple transactions per connection scenarios.

7.4.4.1. Step 1: Test Initialization and Qualification

Verify the link status of all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure traffic load profile of the test equipment to establish "Initial objective" as defined in Section 7.4.3.2. The traffic load profile SHOULD be defined as described in Section 4.3.4.

The DUT/SUT SHOULD reach the "Initial objective" before the sustain phase. The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Section 7.4.3.3.

If the KPI metrics do not meet the test results validation criteria, the test procedure MUST NOT be continued to "Step 2".

7.4.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish "Target objective" defined in Section 7.4.3.2. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4.

The test equipment SHOULD start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT MUST reach the desired value of the target objective in the sustain phase.

Measure the minimum, average, and maximum values of TTFB and TTLB.

7.5. Concurrent TCP/HTTP Connection Capacity

7.5.1. Objective

Determine the number of concurrent TCP connections that the DUT/ SUT sustains when using HTTP traffic.

7.5.2. Test Setup

Testbed setup SHOULD be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.5.3. Test Parameters

In this section, benchmarking test specific parameters SHOULD be defined.

7.5.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific benchmarking test MUST be documented.

7.5.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. The following parameters MUST be noted for this benchmarking test:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target concurrent connection: Initial value from product datasheet or the value defined based on requirement for a specific deployment scenario.

Initial concurrent connection: 10% of "Target concurrent connection" Note: Initial concurrent connection is not a KPI to report. This value is configured on the traffic generator and used to perform the Step1: "Test Initialization and Qualification" described under the Section 7.5.4.

Maximum connections per second during ramp up phase: 50% of maximum connections per second measured in benchmarking test TCP/HTTP Connections per second (Section 7.2)

Ramp up time (in traffic load profile for "Target concurrent connection"): "Target concurrent connection" / "Maximum connections per second during ramp up phase"

Ramp up time (in traffic load profile for "Initial concurrent connection"): "Initial concurrent connection" / "Maximum connections per second during ramp up phase"

The client MUST negotiate HTTP and each client MAY open multiple concurrent TCP connections per server endpoint IP.

Each client sends 10 GET requests requesting 1 KByte HTTP response object in the same TCP connection (10 transactions/TCP connection) and the delay (think time) between each transaction MUST be X seconds.

$$X = (\text{"Ramp up time"} + \text{"steady state time"}) / 10$$

The established connections SHOULD remain open until the ramp down phase of the test. During the ramp down phase, all connections SHOULD be successfully closed with FIN.

7.5.3.3. Test Results Validation Criteria

The following criteria are the test results validation criteria. The Test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

- a. Number of failed application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transaction) of total attempted transactions.
- b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.
- c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

7.5.3.4. Measurement

Average Concurrent TCP Connections MUST be reported for this benchmarking test.

7.5.4. Test Procedures and Expected Results

The test procedure is designed to measure the concurrent TCP connection capacity of the DUT/SUT at the sustaining period of traffic load profile. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial concurrent connection) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution.

7.5.4.1. Step 1: Test Initialization and Qualification

Verify the link status of all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure test equipment to establish "Initial concurrent TCP connections" defined in Section 7.5.3.2. Except ramp up time, the traffic load profile SHOULD be defined as described in Section 4.3.4.

During the sustain phase, the DUT/SUT SHOULD reach the "Initial concurrent TCP connections". The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Section 7.5.3.3.

If the KPI metrics do not meet the test results validation criteria, the test procedure MUST NOT be continued to "Step 2".

7.5.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish the target objective ("Target concurrent TCP connections"). The test equipment SHOULD follow the traffic load profile definition (except ramp up time) as described in Section 4.3.4.

During the ramp up and sustain phase, the other KPIs such as inspected throughput, TCP connections per second, and application transactions per second MUST NOT reach the maximum value the DUT/SUT can support.

The test equipment SHOULD start to measure and record KPIs defined in Section 7.5.3.4. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.5.4.3. Step 3: Test Iteration

Determine the achievable concurrent TCP connections capacity within the test results validation criteria.

7.6. TCP/HTTPS Connections per Second

7.6.1. Objective

Using HTTPS traffic, determine the sustainable SSL/TLS session establishment rate supported by the DUT/SUT under different throughput load conditions.

Test iterations MUST include common cipher suites and key strengths as well as forward looking stronger keys. Specific test iterations MUST include ciphers and keys defined in Section 7.6.3.2.

For each cipher suite and key strengths, test iterations MUST use a single HTTPS response object size defined in Section 7.6.3.2 to measure connections per second performance under a variety of DUT/SUT security inspection load conditions.

7.6.2. Test Setup

Testbed setup SHOULD be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.6.3. Test Parameters

In this section, benchmarking test specific parameters SHOULD be defined.

7.6.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific benchmarking test MUST be documented.

7.6.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. The following parameters MUST be documented for this benchmarking test:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target connections per second: Initial value from product datasheet or the value defined based on requirement for a specific deployment scenario.

Initial connections per second: 10% of "Target connections per second" Note: Initial connections per second is not a KPI to report. This value is configured on the traffic generator and used to perform the Step1: "Test Initialization and Qualification" described under the Section 7.6.4.

RECOMMENDED ciphers and keys defined in Section 4.3.1.3

The client MUST negotiate HTTPS and close the connection with FIN immediately after completion of one transaction. In each test iteration, client MUST send GET request requesting a fixed HTTPS response object size. The RECOMMENDED object sizes are 1, 2, 4, 16, and 64 KByte.

7.6.3.3. Test Results Validation Criteria

The following criteria are the test results validation criteria. The test results validation criteria MUST be monitored during the whole test duration.

- a. Number of failed application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of attempt transactions.
- b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.
- c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).
- d. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.

7.6.3.4. Measurement

TCP connections per second MUST be reported for each test iteration (for each object size).

The KPI metric TLS Handshake Rate can be measured in the test using 1 KByte object size.

7.6.4. Test Procedures and Expected Results

The test procedure is designed to measure the TCP connections per second rate of the DUT/SUT at the sustaining period of traffic load profile. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial connections per second) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution.

7.6.4.1. Step 1: Test Initialization and Qualification

Verify the link status of all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure traffic load profile of the test equipment to establish "Initial connections per second" as defined in Section 7.6.3.2. The traffic load profile SHOULD be defined as described in Section 4.3.4.

The DUT/SUT SHOULD reach the "Initial connections per second" before the sustain phase. The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Section 7.6.3.3.

If the KPI metrics do not meet the test results validation criteria, the test procedure MUST NOT be continued to "Step 2".

7.6.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish "Target connections per second" defined in Section 7.6.3.2. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4.

During the ramp up and sustain phase, other KPIs such as inspected throughput, concurrent TCP connections, and application transactions per second MUST NOT reach the maximum value the DUT/SUT can support. The test results for specific test iteration SHOULD NOT be reported, if the above mentioned KPI (especially inspected throughput) reaches the maximum value. (Example: If the test iteration with 64 KByte of HTTPS response object size reached the maximum inspected throughput limitation of the DUT, the test iteration MAY be interrupted and the result for 64 KByte SHOULD NOT be reported).

The test equipment SHOULD start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective ("Target connections per second") in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.6.4.3. Step 3: Test Iteration

Determine the achievable connections per second within the test results validation criteria.

7.7. HTTPS Throughput

7.7.1. Objective

Determine the sustainable inspected throughput of the DUT/SUT for HTTPS transactions varying the HTTPS response object size.

Test iterations MUST include common cipher suites and key strengths as well as forward looking stronger keys. Specific test iterations MUST include the ciphers and keys defined in Section 7.7.3.2.

7.7.2. Test Setup

Testbed setup SHOULD be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.7.3. Test Parameters

In this section, benchmarking test specific parameters SHOULD be defined.

7.7.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific benchmarking test MUST be documented.

7.7.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. The following parameters MUST be documented for this benchmarking test:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target inspected throughput: Aggregated line rate of interface(s) used in the DUT/SUT or the value defined based on requirement for a specific deployment scenario.

Initial throughput: 10% of "Target inspected throughput" Note: Initial throughput is not a KPI to report. This value is configured on the traffic generator and used to perform the Step1: "Test Initialization and Qualification" described under the Section 7.7.4.

Number of HTTPS response object requests (transactions) per connection: 10

RECOMMENDED ciphers and keys defined in Section 4.3.1.3

RECOMMENDED HTTPS response object size: 1, 16, 64, 256 KByte, and mixed objects defined in Table 4 under Section 7.3.3.2.

7.7.3.3. Test Results Validation Criteria

The following criteria are the test results validation criteria. The test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

- a. Number of failed Application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of attempt transactions.
- b. Traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).
- c. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.

7.7.3.4. Measurement

Inspected Throughput and HTTP Transactions per Second MUST be reported for each object size.

7.7.4. Test Procedures and Expected Results

The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial throughput) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution and HTTPS response object sizes.

7.7.4.1. Step 1: Test Initialization and Qualification

Verify the link status of all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure traffic load profile of the test equipment to establish "Initial throughput" as defined in Section 7.7.3.2.

The traffic load profile SHOULD be defined as described in Section 4.3.4. The DUT/SUT SHOULD reach the "Initial throughput" during the sustain phase. Measure all KPI as defined in Section 7.7.3.4.

The measured KPIs during the sustain phase MUST meet the test results validation criteria "a" defined in Section 7.7.3.3. The test results validation criteria "b" and "c" are OPTIONAL for step 1.

If the KPI metrics do not meet the test results validation criteria, the test procedure MUST NOT be continued to "Step 2".

7.7.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish the target objective ("Target inspected throughput") defined in Section 7.7.3.2. The test equipment SHOULD start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.7.4.3. Step 3: Test Iteration

Determine the achievable average inspected throughput within the test results validation criteria. Final test iteration MUST be performed for the test duration defined in Section 4.3.4.

7.8. HTTPS Transaction Latency

7.8.1. Objective

Using HTTPS traffic, determine the HTTPS transaction latency when DUT/SUT is running with sustainable HTTPS transactions per second supported by the DUT/SUT under different HTTPS response object size.

Scenario 1: The client MUST negotiate HTTPS and close the connection with FIN immediately after completion of a single transaction (GET and RESPONSE).

Scenario 2: The client MUST negotiate HTTPS and close the connection with FIN immediately after completion of 10 transactions (GET and RESPONSE) within a single TCP connection.

7.8.2. Test Setup

Testbed setup SHOULD be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.8.3. Test Parameters

In this section, benchmarking test specific parameters SHOULD be defined.

7.8.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific benchmarking test MUST be documented.

7.8.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. The following parameters MUST be documented for this benchmarking test:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

RECOMMENDED cipher suites and key sizes defined in Section 4.3.1.3

Target objective for scenario 1: 50% of the connections per second measured in benchmarking test TCP/HTTPS Connections per second (Section 7.6)

Target objective for scenario 2: 50% of the inspected throughput measured in benchmarking test HTTPS Throughput (Section 7.7)

Initial objective for scenario 1: 10% of "Target objective for scenario 1"

Initial objective for scenario 2: 10% of "Target objective for scenario 2"

Note: The Initial objectives are not a KPI to report. These values are configured on the traffic generator and used to perform the Step1: "Test Initialization and Qualification" described under the Section 7.8.4.

HTTPS transaction per TCP connection: Test scenario 1 with single transaction and scenario 2 with 10 transactions

HTTPS with GET request requesting a single object. The RECOMMENDED object sizes are 1, 16, and 64 KByte. For each test iteration, client MUST request a single HTTPS response object size.

7.8.3.3. Test Results Validation Criteria

The following criteria are the test results validation criteria. The Test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

- a. Number of failed application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of attempt transactions.
- b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.
- c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

- d. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.
- e. After ramp up the DUT/SUT MUST achieve the "Target objective" defined in the parameter Section 7.8.3.2 and remain in that state for the entire test duration (sustain phase).

7.8.3.4. Measurement

TTFB (minimum, average, and maximum) and TTLB (minimum, average and maximum) MUST be reported for each object size.

7.8.4. Test Procedures and Expected Results

The test procedure is designed to measure TTFB or TTLB when the DUT/SUT is operating close to 50% of its maximum achievable connections per second or inspected throughput. The test procedure consists of two major steps: Step 1 ensures the DUT/SUT is able to reach the initial performance values and meets the test results validation criteria when it was very minimally utilized. Step 2 measures the latency values within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IP types (IPv4 only, IPv6 only and IPv4 and IPv6 mixed traffic distribution), HTTPS response object sizes and single, and multiple transactions per connection scenarios.

7.8.4.1. Step 1: Test Initialization and Qualification

Verify the link status of all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure traffic load profile of the test equipment to establish "Initial objective" as defined in the Section 7.8.3.2. The traffic load profile SHOULD be defined as described in Section 4.3.4.

The DUT/SUT SHOULD reach the "Initial objective" before the sustain phase. The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Section 7.8.3.3.

If the KPI metrics do not meet the test results validation criteria, the test procedure MUST NOT be continued to "Step 2".

7.8.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish "Target objective" defined in Section 7.8.3.2. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4.

The test equipment SHOULD start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT MUST reach the desired value of the target objective in the sustain phase.

Measure the minimum, average, and maximum values of TTFB and TTLB.

7.9. Concurrent TCP/HTTPS Connection Capacity

7.9.1. Objective

Determine the number of concurrent TCP connections the DUT/SUT sustains when using HTTPS traffic.

7.9.2. Test Setup

Testbed setup SHOULD be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.9.3. Test Parameters

In this section, benchmarking test specific parameters SHOULD be defined.

7.9.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific benchmarking test MUST be documented.

7.9.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. The following parameters MUST be documented for this benchmarking test:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

RECOMMENDED cipher suites and key sizes defined in Section 4.3.1.3

Target concurrent connections: Initial value from product datasheet or the value defined based on requirement for a specific deployment scenario.

Initial concurrent connections: 10% of "Target concurrent connections" Note: Initial concurrent connection is not a KPI to report. This value is configured on the traffic generator and used to perform the Step1: "Test Initialization and Qualification" described under the Section 7.9.4.

Connections per second during ramp up phase: 50% of maximum connections per second measured in benchmarking test TCP/HTTPS Connections per second (Section 7.6)

Ramp up time (in traffic load profile for "Target concurrent connections"): "Target concurrent connections" / "Maximum connections per second during ramp up phase"

Ramp up time (in traffic load profile for "Initial concurrent connections"): "Initial concurrent connections" / "Maximum connections per second during ramp up phase"

The client MUST perform HTTPS transaction with persistence and each client can open multiple concurrent TCP connections per server endpoint IP.

Each client sends 10 GET requests requesting 1 KByte HTTPS response objects in the same TCP connections (10 transactions/TCP connection) and the delay (think time) between each transaction MUST be X seconds.

$$X = ("Ramp\ up\ time" + "steady\ state\ time") / 10$$

The established connections SHOULD remain open until the ramp down phase of the test. During the ramp down phase, all connections SHOULD be successfully closed with FIN.

7.9.3.3. Test Results Validation Criteria

The following criteria are the test results validation criteria. The Test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

- a. Number of failed application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of total attempted transactions.
- b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.
- c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

7.9.3.4. Measurement

Average Concurrent TCP Connections MUST be reported for this benchmarking test.

7.9.4. Test Procedures and Expected Results

The test procedure is designed to measure the concurrent TCP connection capacity of the DUT/SUT at the sustaining period of traffic load profile. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial concurrent connection) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution.

7.9.4.1. Step 1: Test Initialization and Qualification

Verify the link status of all connected physical interfaces. All interfaces are expected to be in "UP" status.

Configure test equipment to establish "Initial concurrent TCP connections" defined in Section 7.9.3.2. Except ramp up time, the traffic load profile SHOULD be defined as described in Section 4.3.4.

During the sustain phase, the DUT/SUT SHOULD reach the "Initial concurrent TCP connections". The measured KPIs during the sustain phase MUST meet the test results validation criteria "a" and "b" defined in Section 7.9.3.3.

If the KPI metrics do not meet the test results validation criteria, the test procedure MUST NOT be continued to "Step 2".

7.9.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish the target objective ("Target concurrent TCP connections"). The test equipment SHOULD follow the traffic load profile definition (except ramp up time) as described in Section 4.3.4.

During the ramp up and sustain phase, the other KPIs such as inspected throughput, TCP connections per second, and application transactions per second MUST NOT reach to the maximum value that the DUT/SUT can support.

The test equipment SHOULD start to measure and record KPIs defined in Section 7.9.3.4. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.9.4.3. Step 3: Test Iteration

Determine the achievable concurrent TCP connections within the test results validation criteria.

8. IANA Considerations

This document makes no specific request of IANA.

The IANA has assigned IPv4 and IPv6 address blocks in [RFC6890] that have been registered for special purposes. The IPv6 address block 2001:2::/48 has been allocated for the purpose of IPv6 Benchmarking [RFC5180] and the IPv4 address block 198.18.0.0/15 has been allocated for the purpose of IPv4 Benchmarking [RFC2544]. This assignment was made to minimize the chance of conflict in case a testing device were to be accidentally connected to part of the Internet.

9. Security Considerations

The primary goal of this document is to provide benchmarking terminology and methodology for next-generation network security devices for use in a laboratory isolated test environment. However, readers should be aware that there is some overlap between performance and security issues. Specifically, the optimal configuration for network security device performance may not be the most secure, and vice-versa. The cipher suites recommended in this document are for test purpose only. The cipher suite recommendation for a real deployment is outside the scope of this document.

10. Contributors

The following individuals contributed significantly to the creation of this document:

Alex Samonte, Amritam Putatunda, Aria Eslambolchizadeh, Chao Guo, Chris Brown, Cory Ford, David DeSanto, Jurrie Van Den Breekel, Michelle Rhines, Mike Jack, Ryan Liles, Samaresh Nair, Stephen Goudreaault, Tim Carlin, and Tim Otto.

11. Acknowledgements

The authors wish to acknowledge the members of NetSecOPEN for their participation in the creation of this document. Additionally, the following members need to be acknowledged:

Anand Vijayan, Chris Marshall, Jay Lindenauer, Michael Shannon, Mike Deichman, Ryan Riese, and Toulney Orkun.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

[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

- [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>>.
- [RFC2647] Newman, D., "Benchmarking Terminology for Firewall Performance", RFC 2647, DOI 10.17487/RFC2647, August 1999, <<https://www.rfc-editor.org/info/rfc2647>>.
- [RFC3511] Hickman, B., Newman, D., Tadjudin, S., and T. Martin, "Benchmarking Methodology for Firewall Performance", RFC 3511, DOI 10.17487/RFC3511, April 2003, <<https://www.rfc-editor.org/info/rfc3511>>.
- [RFC5180] Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, DOI 10.17487/RFC5180, May 2008, <<https://www.rfc-editor.org/info/rfc5180>>.
- [RFC6815] Bradner, S., Dubray, K., McQuaid, J., and A. Morton, "Applicability Statement for RFC 2544: Use on Production Networks Considered Harmful", RFC 6815, DOI 10.17487/RFC6815, November 2012, <<https://www.rfc-editor.org/info/rfc6815>>.
- [RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <<https://www.rfc-editor.org/info/rfc6890>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <<https://www.rfc-editor.org/info/rfc9000>>.

Appendix A. Test Methodology - Security Effectiveness Evaluation

A.1. Test Objective

This test methodology verifies the DUT/SUT is able to detect, prevent, and report the vulnerabilities.

In this test, background test traffic will be generated to utilize the DUT/SUT. In parallel, the CVEs will be sent to the DUT/SUT as encrypted and as well as clear text payload formats using a traffic generator. The selection of the CVEs is described in Section 4.2.1.

The following KPIs are measured in this test:

- * Number of blocked CVEs
- * Number of bypassed (nonblocked) CVEs
- * Background traffic performance (verify if the background traffic is impacted while sending CVE toward DUT/SUT)
- * Accuracy of DUT/SUT statistics in term of vulnerabilities reporting

A.2. Testbed Setup

The same testbed MUST be used for security effectiveness test and as well as for benchmarking test cases defined in Section 7.

A.3. Test Parameters

In this section, the benchmarking test specific parameters SHOULD be defined.

A.3.1. DUT/SUT Configuration Parameters

DUT/SUT configuration parameters MUST conform to the requirements defined in Section 4.2. The same DUT configuration MUST be used for Security effectiveness test and as well as for benchmarking test cases defined in Section 7. The DUT/SUT MUST be configured in inline mode and all detected attack traffic MUST be dropped and the session SHOULD be reset

A.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. The same client and server IP ranges MUST be configured as used in the benchmarking test cases. In addition, the following parameters MUST be documented for this benchmarking test:

- * Background Traffic: 45% of maximum HTTP throughput and 45% of Maximum HTTPS throughput supported by the DUT/SUT (measured with object size 64 KByte in the benchmarking tests "HTTP(S) Throughput" defined in Section 7.3 and Section 7.7).
- * RECOMMENDED CVE traffic transmission Rate: 10 CVEs per second
- * It is RECOMMENDED to generate each CVE multiple times (sequentially) at 10 CVEs per second
- * Ciphers and keys for the encrypted CVE traffic MUST use the same cipher configured for HTTPS traffic related benchmarking tests (Section 7.6 – Section 7.9)

A.4. Test Results Validation Criteria

The following criteria are the test results validation criteria. The test results validation criteria MUST be monitored during the whole test duration.

- a. Number of failed application transaction in the background traffic MUST be less than 0.01% of attempted transactions.
- b. Number of terminated TCP connections of the background traffic (due to unexpected TCP RST sent by DUT/SUT) MUST be less than 0.01% of total initiated TCP connections in the background traffic.
- c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).
- d. False positive MUST NOT occur in the background traffic.

A.5. Measurement

Following KPI metrics MUST be reported for this test scenario:

Mandatory KPIs:

- * Blocked CVEs: It SHOULD be represented in the following ways:
 - Number of blocked CVEs out of total CVEs
 - Percentage of blocked CVEs
- * Unblocked CVEs: It SHOULD be represented in the following ways:

- Number of unblocked CVEs out of total CVEs
- Percentage of unblocked CVEs
- * Background traffic behavior: It SHOULD be represented one of the followings ways:
 - No impact: Considered as "no impact'" if any deviation of traffic forwarding rate is less than or equal to 5 % (constant rate)
 - Minor impact: Considered as "minor impact" if any deviation of traffic forwarding rate is greater than 5% and less than or equal to 10% (i.e. small spikes)
 - Heavily impacted: Considered as "Heavily impacted" if any deviation of traffic forwarding rate is greater than 10% (i.e. large spikes) or reduced the background HTTP(S) throughput greater than 10%
- * DUT/SUT reporting accuracy: DUT/SUT MUST report all detected vulnerabilities.

Optional KPIs:

- * List of unblocked CVEs

A.6. Test Procedures and Expected Results

The test procedure is designed to measure the security effectiveness of the DUT/SUT at the sustaining period of the traffic load profile. The test procedure consists of two major steps. This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution.

A.6.1. Step 1: Background Traffic

Generate background traffic at the transmission rate defined in Appendix A.3.2.

The DUT/SUT MUST reach the target objective (HTTP(S) throughput) in sustain phase. The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Appendix A.4.

If the KPI metrics do not meet the acceptance criteria, the test procedure MUST NOT be continued to "Step 2".

A.6.2. Step 2: CVE Emulation

While generating background traffic (in sustain phase), send the CVE traffic as defined in the parameter section.

The test equipment SHOULD start to measure and record all specified KPIs. Continue the test until all CVEs are sent.

The measured KPIs MUST meet all the test results validation criteria defined in Appendix A.4.

In addition, the DUT/SUT SHOULD report the vulnerabilities correctly.

Appendix B. DUT/SUT Classification

This document aims to classify the DUT/SUT in four different categories based on its maximum supported firewall throughput performance number defined in the vendor datasheet. This classification MAY help user to determine specific configuration scale (e.g., number of ACL entries), traffic profiles, and attack traffic profiles, scaling those proportionally to DUT/SUT sizing category.

The four different categories are Extra Small (XS), Small (S), Medium (M), and Large (L). The RECOMMENDED throughput values for the following categories are:

Extra Small (XS) - Supported throughput less than or equal to 1Gbit/s

Small (S) - Supported throughput greater than 1Gbit/s and less than or equal to 5Gbit/s

Medium (M) - Supported throughput greater than 5Gbit/s and less than or equal to 10Gbit/s

Large (L) - Supported throughput greater than 10Gbit/s

Authors' Addresses

Balamuhunthan Balarajah
Berlin
Germany

Email: bm.balarajah@gmail.com

Carsten Rossenhoevel
EANTC AG
Salzufer 14
10587 Berlin
Germany

Email: cross@eantc.de

Brian Monkman
NetSecOPEN
417 Independence Court
Mechanicsburg, PA 17050
United States of America

Email: bmonkman@netsecopen.org

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: December 19, 2019

S. Jacob, Ed.
K. Tiruveedhula
Juniper Networks
June 17, 2019

Benchmarking Methodology for EVPN VPWS
draft-kishjac-bmwg-evpnvpwstest-02

Abstract

This document defines methodologies for benchmarking EVPN-VPWS performance. EVPN-VPWS is defined in RFC 8214, and is being deployed in Service Provider networks. Specifically this document defines the methodologies for benchmarking EVPN-VPWS Scale convergence, Scale, Core isolation, high availability and longevity.

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 December 19, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Language	3
1.2. Terminologies	3
2. Test Topology	3
3. Test Cases	7
3.1. How long it takes to switch from primary to backup during local link failure	7
3.2. How long it takes to remote PE to switch traffic from primary to back up path during link failure in CE	8
3.3. How long it takes to remote PE to switch traffic from primary to back up path during core failure	8
3.4. How long it takes to primary PE to regain control after the local link flap	9
4. Activate/deactivate AC's	10
4.1. To Add M number of attachment circuits.	10
4.2. Deactivate/Activate M number of attachment circuits.	10
5. Scale Convergence	11
5.1. To measure the packet loss during the core link failure.	11
6. High Availability	11
6.1. To Record the whether there is traffic loss due to routing engine failover for redundancy test.	12
7. SOAK Test	12
7.1. To Measure the stability of the DUT with scale and traffic.	12
8. Acknowledgements	13
9. IANA Considerations	13
10. Security Considerations	13
11. References	13
11.1. Normative References	13
11.2. Informative References	13
Appendix A. Appendix	14
Authors' Addresses	14

1. Introduction

EVPN-VPWS is defined in RFC 8214, discusses how VPWS can be combined with EVPNs to provide a new/combined solution. This draft defines methodologies that can be used to benchmark RFC 8214 solutions. Further, this draft provides methodologies for benchmarking the performance of EVPN VPWS Scale, Scale Convergence, Core isolation, longevity, high availability.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2. Terminologies

MHPE Multi homed Provide Edge router.

RR Route Reflector.

P Provider Router.

CE Customer Router/Devices/Switch.

MHPE2 Multi homed Provider Edge router 2.

MHPE1 Multi homed Provider Edge router 1.

SHPE3 Single homed Provider Edge Router 3.

AA EVPN Terminologies AA All-Active.

AC Attachment Circuit(customer EVPN-VPWS Service over the Provider network

SA EVPN Terminologies SA Single-Active.

RT Router Tester.

Sub Interface Each physical Interfaces is subdivided in to Logical units.

EVI EVPN Instances which will be running on sub interface or physical port of the provider Edge routers.

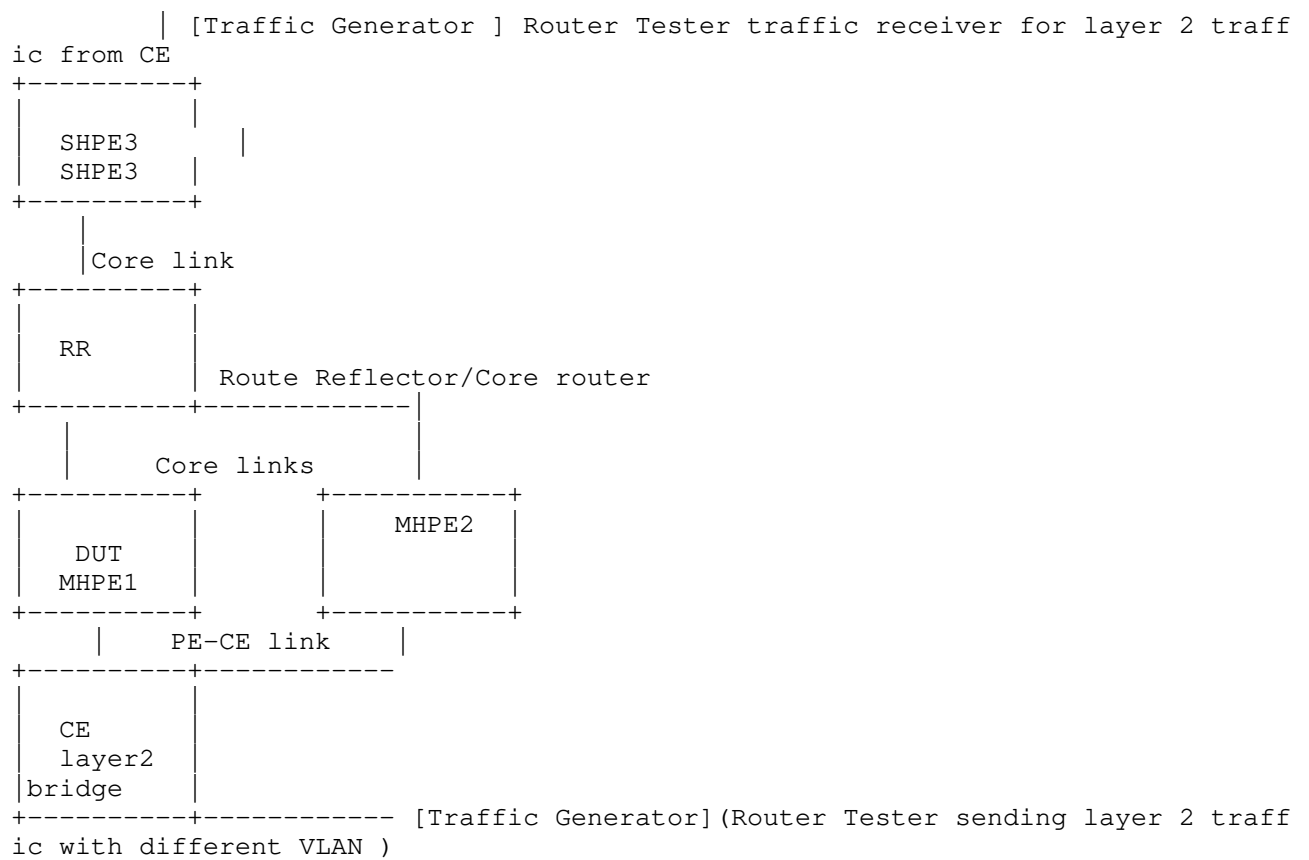
DF Designated Forwarder.

ESI Ethernet Segment Identifier.

2. Test Topology

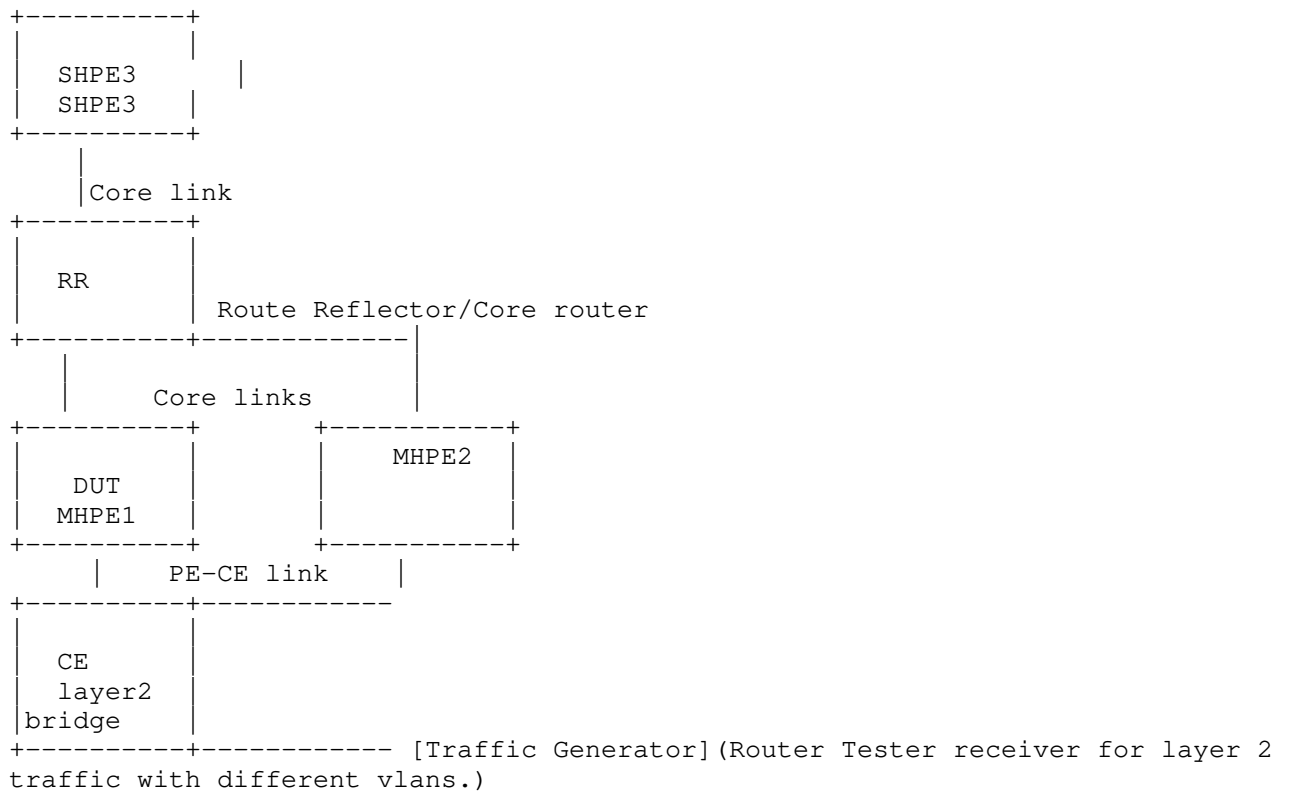
EVPN-VPWS Services running on SHPE3, MHPE1 and MHPE2 in Single Active Mode:

Topology Diagram

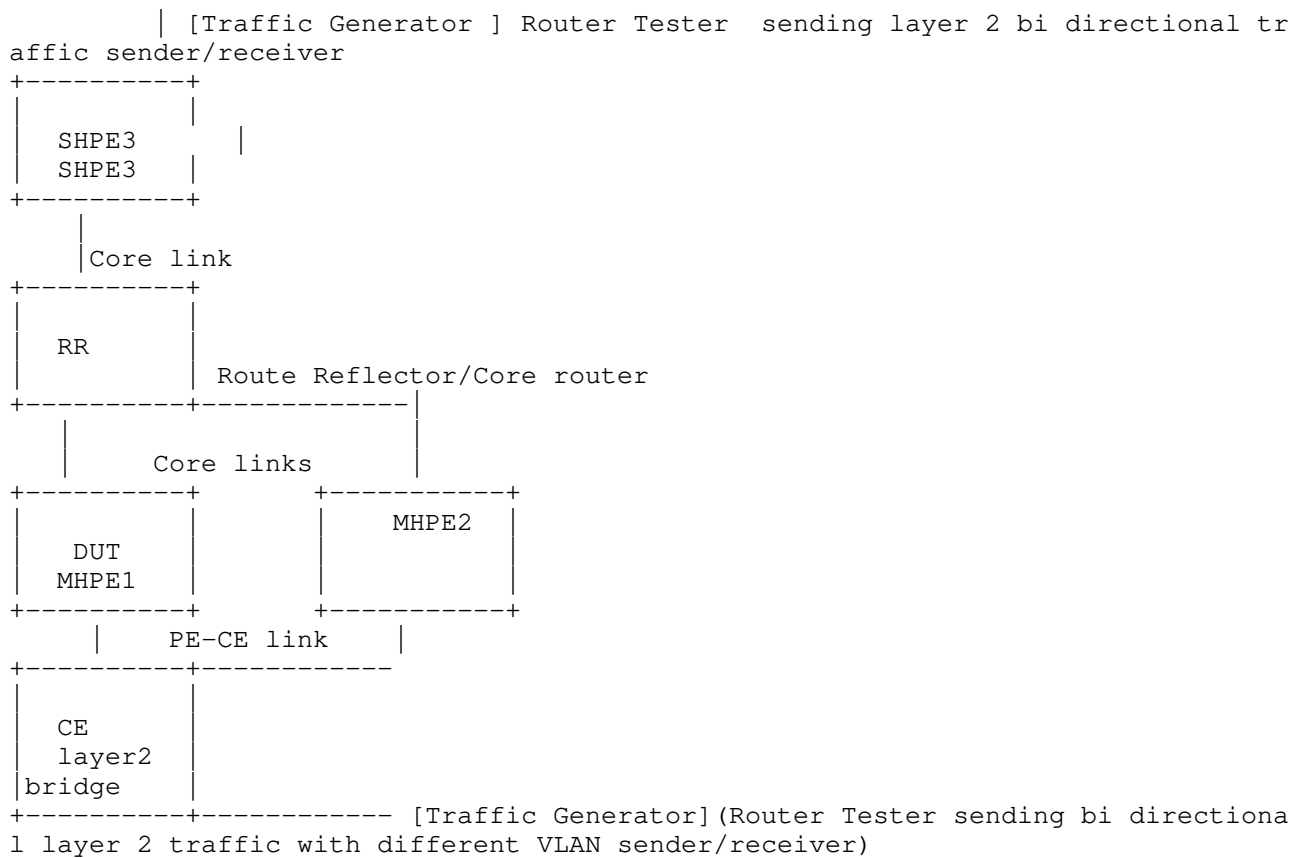


Topology 1

[Traffic Generator] Router Tester sending layer 2 traffic.



Topology 2



Topology 3

Topology Diagram

Figure 1

There are five routers in the topology. SHPE3, RR/P, MHPE1 and MHPE2 emulating a service provider network. CE is a customer device connected to MHPE1 and MHPE2, it is configured with bridge domains in different vlans. The router tester is connected to CE and SHPE3. The MHPE1 acts as DUT. The RT will act as sender and receiver. The measurement will be taken in DUT.

All routers except CE is configured with OSPF/IS-IS, LDP, MPLS, BGP with EVPN address family.

All routers except CE must have IBGP configured with RR acting as route reflector.

MHPE1,MHPE2,SHPE3 must be configured with "N" EVPN-VPWS instances depends up on the cases.

MHPE1 and MHEPE2 must be configured with ESI per vlan or ESI on IFD.

MHPE1 and MHEPE2 are running Single Active mode of EVPN-VPWS.

CE is acting as bridge configured with vlans that is configured on MHPE1,MHPE2,SHPE3.

Depends up on the test traffic will be flowing uni directional or bi directional depends on the topology mentioned above.

The above configuration will serve as base configuration for all the test cases.

3. Test Cases

The following tests are conducted to measure the packet loss during the local link and core failure in DUT with Scaled AC's.

3.1. How long it takes to switch from primary to backup during local link failure

Objective:

To Record the time taken to switch from primary to backup during local link failure.

Topology : Topology 1

Procedure:

Configure "N" AC's in SHPE3 and MHPE1,MHPE2, working in SA mode.Ensure MHPE2 is active and DUT is backup PE.Send "X" unicast packets from CE to MHPE2 AC's working in SA. Then shut the MHPE2-CE link, so that traffic from CE switches to DUT.

Measurement :

Measure the time taken to switch the traffic from active to backup, the traffic will flow from MHPE1 to SHPE3. Measure the time taken to switch the traffic.

Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The switching time is calculated by averaging the values obtained from "N" samples.

AC's switch over from primary to backup PE in sec = $(T1+T2+..Tn/N)$

3.2. How long it takes to remote PE to switch traffic from primary to back up path during link failure in CE

Objective:

To Record the time taken by remote PE to switch traffic from primary to backup during CE link failure.

Topology : Topology 2

Procedure:

Configure "N" AC's in SHPE3 and MHPE1,MHPE2, working in SA mode.Ensure MHPE2 is active and DUT is backup PE.Send "X" unicast packets from RT to SHPE3 Ac's.Then shut the MHPE2-CE link, this failure will be notified to remote PE and traffic switch to backup path.

Measurement :

Measure the time taken to switch the traffic from active to backup, the traffic will flow from SHPE3 to MHPE1. Measure the time taken to switch the traffic.

Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The switching time is calculated by averaging the values obtained from "N" samples.

AC's switch over from primary to backup PE in sec = $(T1+T2+..Tn/N)$

3.3. How long it takes to remote PE to switch traffic from primary to back up path during core failure

Objective:

To Record the time taken by remote PE to switch traffic from primary to backup during core link failure.

Topology : Topology 2

Procedure:

Configure "N" AC's in SHPE3 and MHPE1,MHPE2, working in SA mode.Ensure MHPE2 is active and DUT is backup PE.Send "X" unicast packets from RT to SHPE3 Ac's.Then shut the core link of MHPE2,this failure will be notified to remote PE and traffic switch to backup path.

Measurement :

Measure the time taken to switch the traffic from active to backup, the traffic will flow from SHPE3 to MHPE1. Measure the time taken to switch the traffic.

Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The switching time is calculated by averaging the values obtained from "N" samples.

AC's in remote PE switches from primary to backup PE in sec due to core failure = $(T1+T2+..Tn/N)$

3.4. How long it takes to primary PE to regain control after the local link flap

Objective:

To Record the time taken by primary PE to regain control after the local PE-CE link flap.

Topology : Topology 1

Procedure:

Configure "N" AC's in SHPE3 and MHPE1,MHPE2, working in SA mode.Ensure MHPE2 is standby and DUT is primary PE.Send "X" unicast packets from CE to all Ac's in MHPE1(DUT).Then shut the link of MHPE1-CE,this failure will be notified to remote PE and traffic switch to backup path. Then bring up the link of MHPE1-CE.Now the traffic switches to DUT.

Measurement :

Measure the time taken to switch the traffic from MHPE2 to DUT, the traffic will flow from MHPE1 to SHPE3. Measure the time taken to switch the traffic.

Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The switching time is calculated by averaging the values obtained from "N" samples.

Time taken to switch back to primary(DUT) once the link is restored =
(T1+T2+...Tn/N)

4. Activate/deactivate AC's

4.1. To Add M number of attachment circuits.

Objective:

To measure the performance of the DUT while adding M AC's on the fly.

Topology : Topology 3

Procedure:

Configure "N" AC's in SHPE3 and MHPE1,MHPE2, working in SA mode.Ensure MHPE2 is active and DUT is backup PE.Send "X" unicast packets from RT to SHPE3 to all Ac's and send "X" unicast packets from CE to MHPE1(DUT),let the DUT is the active and the MHPE2 must be standby. DUT will be forwarding the traffic to CE from SHPE3 and the traffic from CE to SHPE3.Then add "M" AC's on SHPE1,DUT and MHPE2 on the fly. these AC' must be in SA mode.

Measurement :

There should be 0 traffic loss in existing services while addition of these ACs.

4.2. Deactivate/Activate M number of attachment circuits.

Objective:

To measure the performance of the DUT while deactivating/activating AC's.

Topology : Topology 3

Procedure:

Configure "N" AC's in SHPE3 and MHPE1,MHPE2, working in SA mode.Ensure MHPE2 is active and DUT is backup PE.Send "X" unicast packets from RT to SHPE3 to all Ac's and send "X" unicast packets from CE to MHPE1(DUT),let the DUT is the active and the MHPE2 must be standby.DUT will be forwarding the traffic to CE and from CE to SHPE3.Then deactivate "M" AC's on SHPE1,DUT and MHPE2 on the fly. these AC' must be removed from forwarding plane. Stop the traffic

for these AC's. Activate the AC's in all PE's. then start the traffic, measure the time taken by "M" AC's to forward the traffic.

Measurement :

Measure the packet loss in sec during this deactivating/activating AC's. Repeat the test "N" times and plot the data. The packet loss is calculated by averaging the values obtained from "N" samples.

packet loss in sec = $(T1+T2+..Tn/N)$

5. Scale Convergence

5.1. To measure the packet loss during the core link failure.

Objective:

To Measure the convergence at a higher number of AC's

Topology : Topology 3

Procedure:

Configure "N" AC's in SHPE3 and MHPE1, MHPE2, working in SA mode. The scale factor must be in the multiples of thousands. DF election must be priority based not on the default RFC 7432, it should not be MOD based DF election. Send "X" unicast packets from RT to SHPE3 to all AC's and send "X" unicast packets from CE to MHPE1 (DUT), let the DUT be the active and the MHPE2 must be standby. DUT will be forwarding the traffic to CE from SHPE3 and from CE to SHPE3. Then flap the core link of the DUT.

Measurement :

Measure the packet loss in seconds once the core link is restored. Repeat the test "N" times and plot the data. The packet loss is calculated by averaging the values obtained from "N" samples.

Packet loss in sec = $(T1+T2+..Tn/N)$

6. High Availability

- 6.1. To Record the whether there is traffic loss due to routing engine failover for redundancy test.

Objective:

To record traffic loss during routing engine failover.

Topology : Topology 3

Procedure:

Configure "N" AC's in SHPE3 and MHPE1,MHPE2, working in SA mode.Ensure MHPE2 is active and DUT is backup PE.Send "X" unicast packets from RT to SHPE3 to all Ac's and send "X" unicast packets from CE to MHPE1(DUT),let the DUT is the active and the MHPE2 must be standby. DUT will be forwarding the traffic to CE and from CE to SHPE3.Then do a routing engine fail-over.

Measurement :

There should be 0 traffic loss which is the ideal case, No change in the DF role. DUT should not withdraw any routes.Repeat the test "N" times and plot the data.The packet loss is calculated by averaging the values obtained from "N" samples.

Packet loss in sec = $(T1+T2+..Tn/N)$

7. SOAK Test

This is measuring the performance of DUT running with scaled configuration with traffic over a peroid of time "T' ". In each interval "t1" the parameters measured are CPU usage, memory usage, crashes.

- 7.1. To Measure the stability of the DUT with scale and traffic.

Objective:

To measure the stability of the DUT in a scaled environment with traffic.

Topology : Topology 3

Procedure:

Scale N AC's in DUT, SHPE3 and MHPE2. Send F frames to DUT from CE using traffic generator with different X SA and DA for N EVI's. Send F frames from traffic generator to SHPE3 with X different SA and DA. There is a bi directional traffic flow with F pps in each direction. The DUT must run with traffic for 24 hours, every hour check for memory leak, crash.

Measurement :

Take the hourly reading of CPU, process memory. There should not be any leak, crashes, CPU spikes.

8. Acknowledgements

We would like to thank Al and Sarah for the support.

9. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

There is no additional consideration from RFC 6192.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [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>>.
- [RFC2899] Ginoza, S., "Request for Comments Summary RFC Numbers 2800-2899", RFC 2899, DOI 10.17487/RFC2899, May 2001, <<https://www.rfc-editor.org/info/rfc2899>>.

11.2. Informative References

- [RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <<https://www.rfc-editor.org/info/rfc7432>>.

[RFC8214] Boutros, S., Sajassi, A., Salam, S., Drake, J., and J. Rabadan, "Virtual Private Wire Service Support in Ethernet VPN", RFC 8214, DOI 10.17487/RFC8214, August 2017, <<https://www.rfc-editor.org/info/rfc8214>>.

Appendix A. Appendix

Authors' Addresses

Sudhin Jacob (editor)
Juniper Networks
Bangalore
India

Phone: +91 8061212543
Email: sjacob@juniper.net

Kishore Tiruveedhula
Juniper Networks
10 Technology Park Dr
Westford, MA 01886
USA

Phone: +1 9785898861
Email: kishoret@juniper.net

Benchmarking Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2020

M. Konstantynowicz, Ed.
P. Mikus, Ed.
Cisco Systems
July 08, 2019

NFV Service Density Benchmarking
draft-mkonstan-nf-service-density-01

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 January 9, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Terminology	3
2. Motivation	4
2.1. Problem Description	4
2.2. Proposed Solution	4
3. NFV Service	5
3.1. Topology	6
3.2. Configuration	8
3.3. Packet Path(s)	9
4. Virtualization Technology	12
5. Host Networking	13
6. NFV Service Density Matrix	14
7. Compute Resource Allocation	15
8. NFV Service Data-Plane Benchmarking	19
9. Sample NFV Service Density Benchmarks	19
9.1. Interpreting the Sample Results	20
9.2. Benchmarking MRR Throughput	20
9.3. VNF Service Chain	20
9.4. CNF Service Chain	21
9.5. CNF Service Pipeline	22
9.6. Sample Results: FD.io CSIT	23
9.7. Sample Results: CNCF/CNFs	24
9.8. Sample Results: OPNFV NFVbench	26
10. IANA Considerations	26
11. Security Considerations	26
12. Acknowledgements	26
13. References	27
13.1. Normative References	27
13.2. Informative References	27
Authors' Addresses	28

1. Terminology

- o **NFV:** Network Function Virtualization, a general industry term describing network functionality implemented in software.
- o **NFV service:** a software based network service realized by a topology of interconnected constituent software network function applications.
- o **NFV service instance:** a single instantiation of NFV service.
- o **Data-plane optimized software:** any software with dedicated threads handling data-plane packet processing e.g. FD.io VPP (Vector Packet Processor), OVS-DPDK.
- o **Packet Loss Ratio (PLR):** ratio of packets received relative to packets transmitted over the test trial duration, calculated using formula: $PLR = (pkts_transmitted - pkts_received) / pkts_transmitted$. For bi-directional throughput tests aggregate PLR is calculated based on the aggregate number of packets transmitted and received.
- o **Packet Throughput Rate:** maximum packet offered load DUT/SUT forwards within the specified Packet Loss Ratio (PLR). In many cases the rate depends on the frame size processed by DUT/SUT. Hence packet throughput rate MUST be quoted with specific frame size as received by DUT/SUT during the measurement. For bi-directional tests, packet throughput rate should be reported as aggregate for both directions. Measured in packets-per-second (pps) or frames-per-second (fps), equivalent metrics.
- o **Non Drop Rate (NDR):** maximum packet/bandwidth throughput rate sustained by DUT/SUT at PLR equal zero (zero packet loss) specific to tested frame size(s). MUST be quoted with specific packet size as received by DUT/SUT during the measurement. Packet NDR measured in packets-per-second (or fps), bandwidth NDR expressed in bits-per-second (bps).
- o **Partial Drop Rate (PDR):** maximum packet/bandwidth throughput rate sustained by DUT/SUT at PLR greater than zero (non-zero packet loss) specific to tested frame size(s). MUST be quoted with specific packet size as received by DUT/SUT during the measurement. Packet PDR measured in packets-per-second (or fps), bandwidth PDR expressed in bits-per-second (bps).
- o **Maximum Receive Rate (MRR):** packet/bandwidth rate regardless of PLR sustained by DUT/SUT under specified Maximum Transmit Rate (MTR) packet load offered by traffic generator. MUST be quoted

with both specific packet size and MTR as received by DUT/SUT during the measurement. Packet MRR measured in packets-per-second (or fps), bandwidth MRR expressed in bits-per-second (bps).

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 data based decisions about the NFV technology and service design best suited to meet requirements of their use cases. Thereby obtained benchmarking data would aid in selection of the most appropriate NFV infrastructure design and platform and enable more accurate capacity planning, an important element for commercial viability of the NFV service.

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:

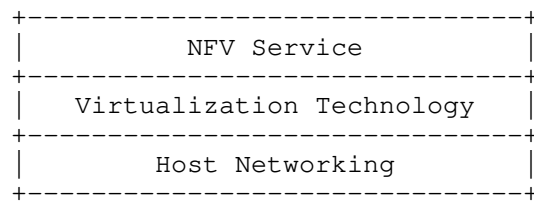


Figure 1. NFV software technology stack.

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 the system under test configuration. Each of the compute node configurations identified in this document 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.

In both cases multiple NFV service topologies are running in parallel. Both topologies are shown in figures 2. and 3. below.

NF chain topology:

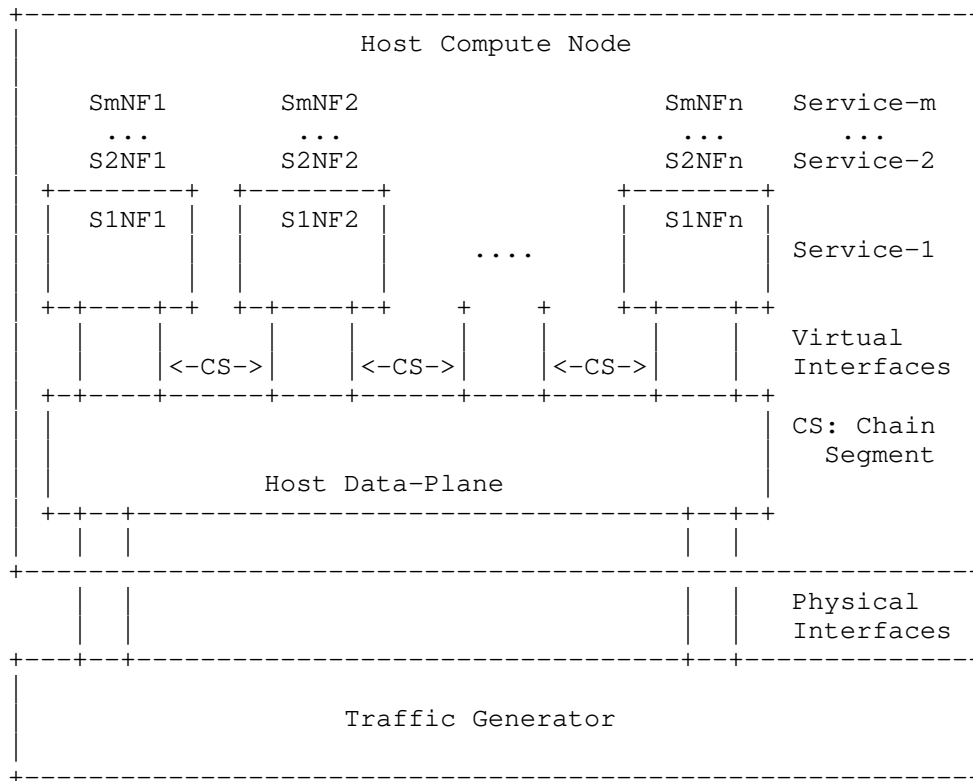


Figure 2. NF chain topology forming a service instance.

NF pipeline topology:

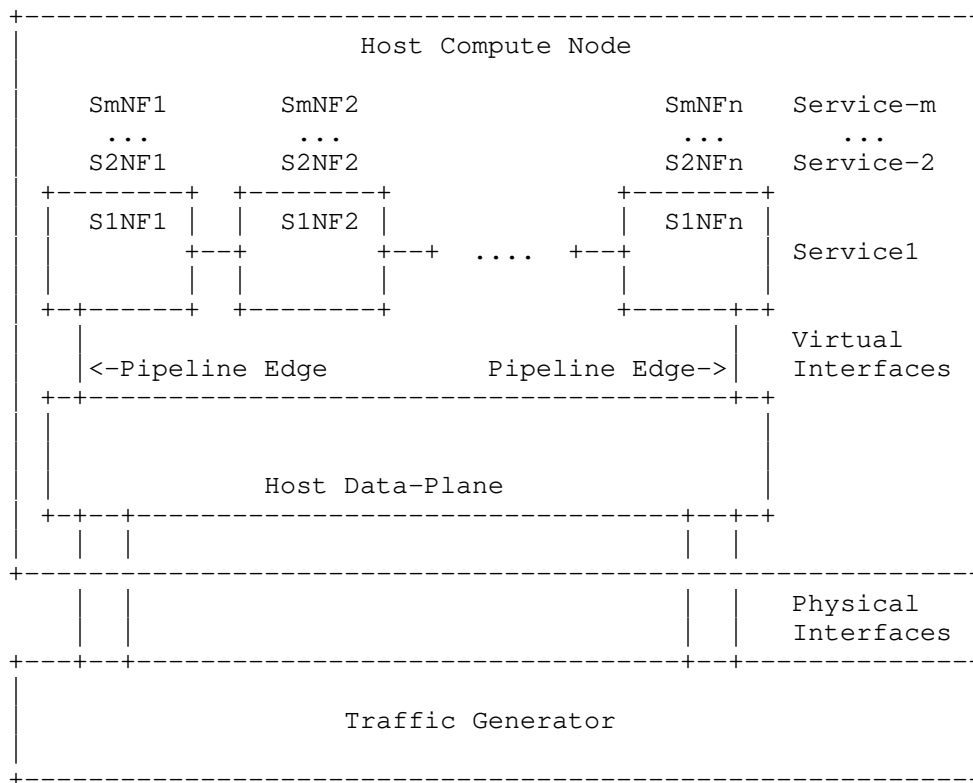


Figure 3. NF pipeline topology forming a service instance.

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 logical network connectivity between the NFs, and no NFV service specific configurations. NF specific functionality is emulated using IPv4/IPv6 routing.

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:

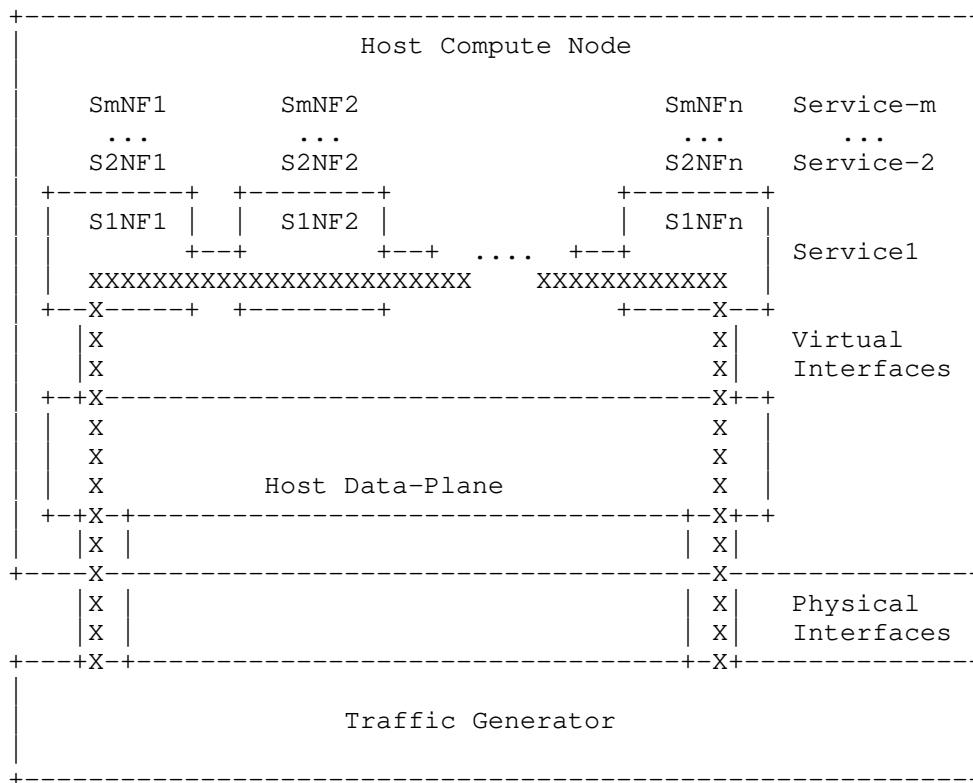


Figure 5. Pipeline packet path thru NF pipeline topology.

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 the 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:

- o Linux Kernel-Mode Networking.
- o Linux User-Mode vSwitch.
- o Virtual Machine vSwitch.
- o Linux Container vSwitch.
- o 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 benchmarks are done with the offered packet load distributed equally across all configured NFV service instances. This approach should provide representative benchmarking data for each tested topology and configuration, and a good guesstimate of maximum performance required for capacity planning.

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:

- o 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.
- o 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

o 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 Data-Plane Benchmarking

NF service density scenarios should have their data-plane performance benchmarked using existing and/or emerging network benchmarking standards as noted earlier.

Following metrics should be measured (or calculated) and reported:

- o Packet throughput rate (packets-per-second)
 - * Specific to tested packet size or packet sequence (e.g. some type of packet size mix sent in recurrent sequence).
 - * Applicable types of throughput rate: NDR, PDR, MRR.
- o (Calculated) Bandwidth throughput rate (bits-per-second) corresponding to the measured packet throughput rate.
- o Packet one-way latency (seconds)
 - * Measured at different packet throughput rates load e.g. light, medium, heavy.

Listed metrics should be itemized per service instance and per direction (e.g. forward/reverse) for latency.

9. Sample 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, ii) as part of CNCF CNF Testbed initiative [CNCF-CNF-Testbed] and iii) in OPNFV NFVbench project.

In the first two 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.

OPNFV NFVbench project is focusing on benchmarking the actual production deployments that are aligned with OPNFV specifications.

9.1. Interpreting the Sample Results

TODO How to interpret and avoid misreading included results? And how to avoid falling into the trap of using these results to draw generalized conclusions about performance of different virtualization technologies, e.g. VM and Containers, irrespective of deployment scenarios and what VNFs and CNFs are in the actual use.

9.2. Benchmarking 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 specified Maximum Transmit Rate (MTR) packet load offered by traffic generator over a set trial duration, regardless of packet loss ratio (PLR). MTR for specified Ethernet frame size was 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.

9.3. 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.

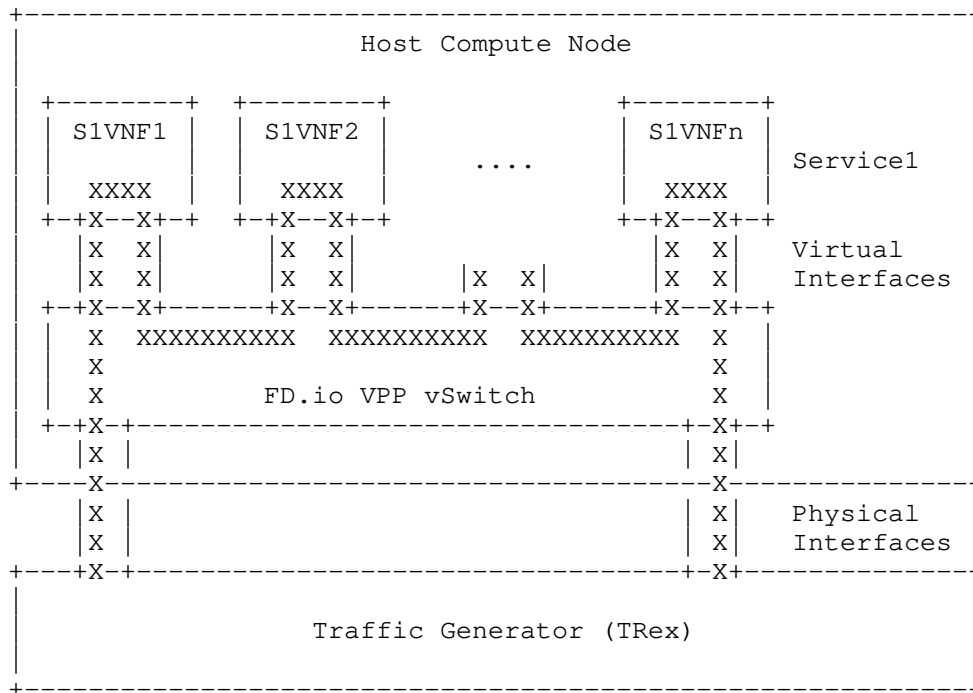


Figure 6. VNF service chain test setup.

9.4. 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.

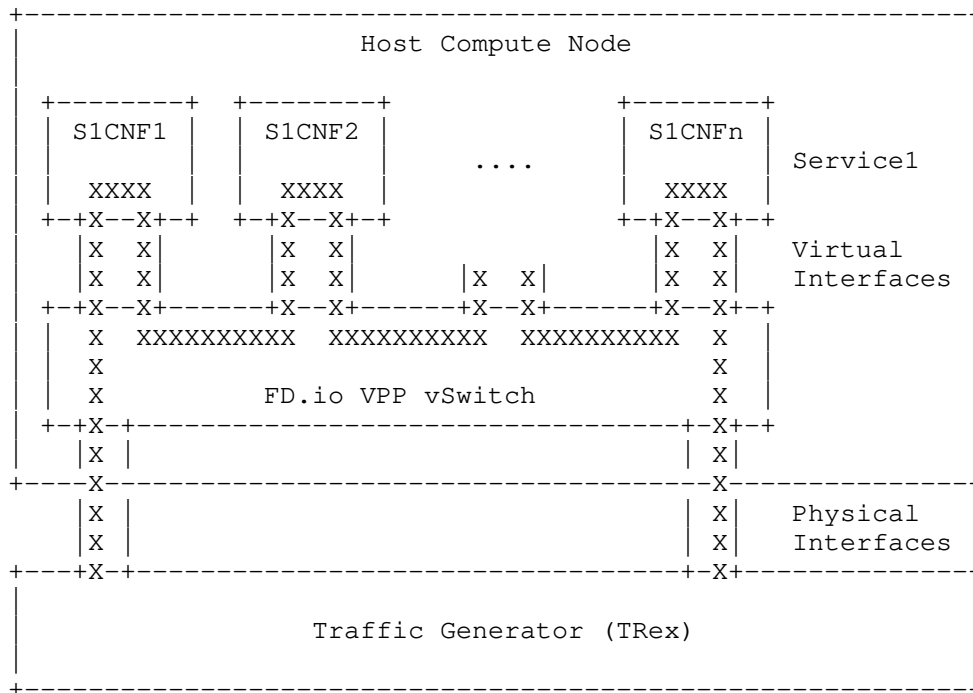


Figure 7. CNF service chain test setup.

9.5. 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.

2. CNF Service Chains

- * CNF: VPP v19.04-release
 - + IPv4 routing
 - + NF-1c
- * vSwitch: VPP v19.04-release
 - + L2 MAC switching
 - + vSwitch-1c, vSwitch-2c
- * frame sizes: 64B, IMIX

3. CNF Service Pipelines

- * CNF: VPP v19.04-release
 - + IPv4 routing
 - + NF-1c
- * vSwitch: VPP v19.04-release
 - + L2 MAC switching
 - + vSwitch-1c, vSwitch-2c
- * frame sizes: 64B, IMIX

More information is available in FD.io CSIT-1904 report, with specific references listed below:

- o Testbed: [CSIT-1904-testbed-2n-skx]
- o Test environment: [CSIT-1904-test-environment]
- o Methodology: [CSIT-1904-nfv-density-methodology]
- o Results: [CSIT-1904-nfv-density-results]

9.7. Sample Results: CNCF/CNFs

CNCF CI team introduced a CNF testbed initiative focusing on benchmarking 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]

9.8. Sample Results: OPNFV NFVbench

TODO Add short NFVbench based test description, and NFVbench sweep chart with single VM per service instance: Y-axis packet throughput rate or bandwidth throughput rate, X-axis number of concurrent service instances.

10. IANA Considerations

No requests of IANA.

11. Security Considerations

Benchmarking activities as described in this memo are limited to technology characterization of a DUT/SUT using controlled stimuli in a laboratory environment, with dedicated address space and the constraints specified in the sections above.

The benchmarking network topology will be an independent test setup and MUST NOT be connected to devices that may forward the test traffic into a production network or misroute traffic to the test management network.

Further, benchmarking is performed on a "black-box" basis, relying solely on measurements observable external to the DUT/SUT.

Special capabilities SHOULD NOT exist in the DUT/SUT specifically for benchmarking purposes. Any implications for network security arising from the DUT/SUT SHOULD be identical in the lab and in production networks.

12. Acknowledgements

Thanks to Vratko Polak of FD.io CSIT project and Michael Pedersen of the CNCF Testbed initiative for their contributions and useful suggestions. Extended thanks to Alec Hothan of OPNFV NFVbench project for numerous comments, suggestions and references to his/team work in the OPNFV/NFVbench project.

13. References

13.1. 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>>.

13.2. 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", July 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-1904-nfv-density-methodology] "FD.io CSIT Test Methodology: NFV Service Density", June 2019, <https://docs.fd.io/csit/rls1904/report/introduction/methodology_nfv_service_density.html>.
- [CSIT-1904-nfv-density-results] "FD.io CSIT Test Results: NFV Service Density", June 2019, <https://docs.fd.io/csit/rls1904/report/vpp_performance_tests/nf_service_density/index.html>.
- [CSIT-1904-test-environment] "FD.io CSIT Test Environment", June 2019, <https://docs.fd.io/csit/rls1904/report/vpp_performance_tests/test_environment.html>.

- [CSIT-1904-testbed-2n-skx]
"FD.io CSIT Test Bed", June 2019,
<https://docs.fd.io/csit/rls1904/report/introduction/physical_testbeds.html#node-xeon-skylake-2n-skx>.
- [draft-vpolak-bmwg-plrsearch]
"Probabilistic Loss Ratio Search for Packet Throughput (PLRsearch)", July 2019,
<<https://tools.ietf.org/html/draft-vpolak-bmwg-plrsearch>>.
- [draft-vpolak-mkonstan-bmwg-mlrsearch]
"Multiple Loss Ratio Search for Packet Throughput (MLRsearch)", July 2019, <<https://tools.ietf.org/html/draft-vpolak-mkonstan-bmwg-mlrsearch>>.
- [LFN-FDio-CSIT]
"Fast Data io, Continuous System Integration and Testing Project", July 2019, <<https://wiki.fd.io/view/CSIT>>.
- [NFVbench]
"NFVbench Data Plane Performance Measurement Features", July 2019, <<https://opnfv-nfvbench.readthedocs.io/en/latest/testing/user/userguide/readme.html>>.
- [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", July 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

Network Working Group
Internet-Draft
Updates: ???? (if approved)
Intended status: Informational
Expires: May 7, 2020

A. Morton
J. Uttaro
AT&T Labs
November 4, 2019

Benchmarks and Methods for Multihomed EVPN
draft-morton-bmwg-multihome-evpn-03

Abstract

Fundamental Benchmarking Methodologies for Network Interconnect Devices of interest to the IETF are defined in RFC 2544. Key benchmarks applicable to restoration and multi-homed sites are in RFC 6894. This memo applies these methods to Multihomed nodes implemented on Ethernet Virtual Private Networks (EVPN).

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14[RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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 May 7, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Scope and Goals	3
3. Motivation	3
4. Test Setups	3
4.1. Basic Configuration	5
5. Procedure for Full Mesh Throughput Characterization	6
5.1. Address Learning Phase	6
5.2. Test for a Single Frame Size and Number of Unicast Flows	6
5.3. Detailed Procedure	6
5.4. Test Repetition	7
5.5. Benchmark Calculations	7
5.6. Reporting	7
6. Procedure for Mass Withdrawal Characterization	7
6.1. Address Learning Phase	8
6.2. Test for a Single Frame Size and Number of Flows	8
6.3. Test Repetition	8
6.4. Benchmark Calculations	8
7. Reporting	9
8. Security Considerations	9
9. IANA Considerations	10
10. Acknowledgements	10
11. References	10
11.1. Normative References	10
11.2. Informative References	11
Authors' Addresses	12

1. Introduction

The IETF's fundamental Benchmarking Methodologies are defined in [RFC2544], supported by the terms and definitions in [RFC1242], and [RFC2544] actually obsoletes an earlier specification, [RFC1944].

This memo recognizes the importance of Ethernet Virtual Private Network (EVPN) Multihoming connectivity scenarios, where a CE device is connected to 2 or more PEs using an instance of an Ethernet Segment.

In an all-active or Active-Active scenario, CE-PE traffic is load-balanced across two or more PEs.

Mass-withdrawal of routes may take place when an autodiscovery route is used on a per Ethernet Segment basis, and there is a link failure on one of the Ethernet Segment links (or when configuration changes take place).

Although EVPN depends on address-learning in the control-plane, the Ethernet Segment Instance is permitted to use "the method best suited to the CE: data-plane learning, IEEE 802.1x, the Link Layer Discovery Protocol (LLDP), IEEE 802.1aq, Address Resolution Protocol (ARP), management plane, or other protocols" [RFC7432].

This memo seeks to benchmark these important cases (and others).

2. Scope and Goals

The scope of this memo is to define a method to unambiguously perform tests, measure the benchmark(s), and report the results for Capacity of EVPN Multihoming connectivity scenarios, and other key restoration activities (such as address withdrawal) covering link failure in the Active-Active scenario.

The goal is to provide more efficient test procedures where possible, and to expand reporting with additional interpretation of the results. The tests described in this memo address some key multihoming scenarios implemented on a Device Under Test (DUT) or System Under Test (SUT).

3. Motivation

The Multihoming scenarios described in this memo emphasize features with practical value to the industry that have seen deployment. Therefore, these scenarios deserve further attention that follows from benchmarking activities and further study.

4. Test Setups

For simple Capacity/Throughput Benchmarks, the Test Setup MUST be consistent with Figure 1 of [RFC2544], or Figure 2 when the tester's sender and receiver are different devices.

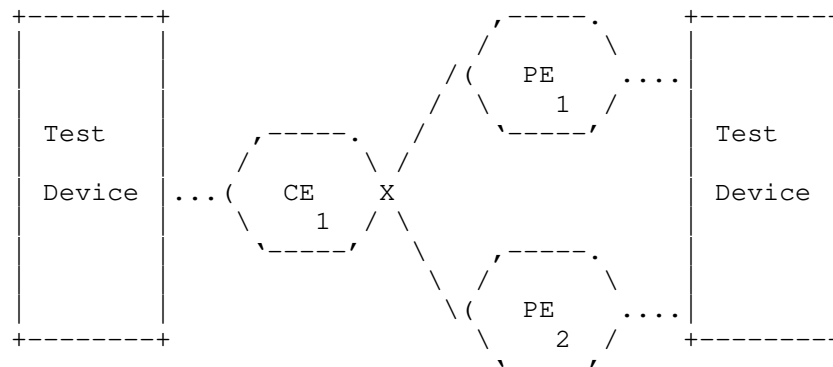


Figure 1 SUT for Throughput and other Ethernet Segment Tests

In Figure 1, the System Under Test (SUT) is comprised of a single CE device and two or more PE devices.

The tester SHALL be connected to all CE and every PE, and be capable of simultaneously sending and receiving frames on all ports with connectivity. The tester SHALL be capable of generating multiple flows (according to a 5-tuple definition, or any sub-set of the 5-tuple). The tester SHALL be able to control the IP capacity of sets of individual flows, and the presence of sets of flows on specific interface ports.

The tester SHALL be capable of generating and receiving a full mesh of Unicast flows, as described in section 3.0 of [RFC2889]:

"In fully meshed traffic, each interface of a DUT/SUT is set up to both receive and transmit frames to all the other interfaces under test."

Other mandatory testing aspects described in [RFC2544] and [RFC2889] MUST be included, unless explicitly modified in the next section.

The ingress and egress link speeds and link layer protocols MUST be specified and used to compute the maximum theoretical frame rate when respecting the minimum inter-frame gap.

A second test case is where a BGP backbone implements MPLS-LDP to provide connectivity between multiple PE - ESI - CE locations.

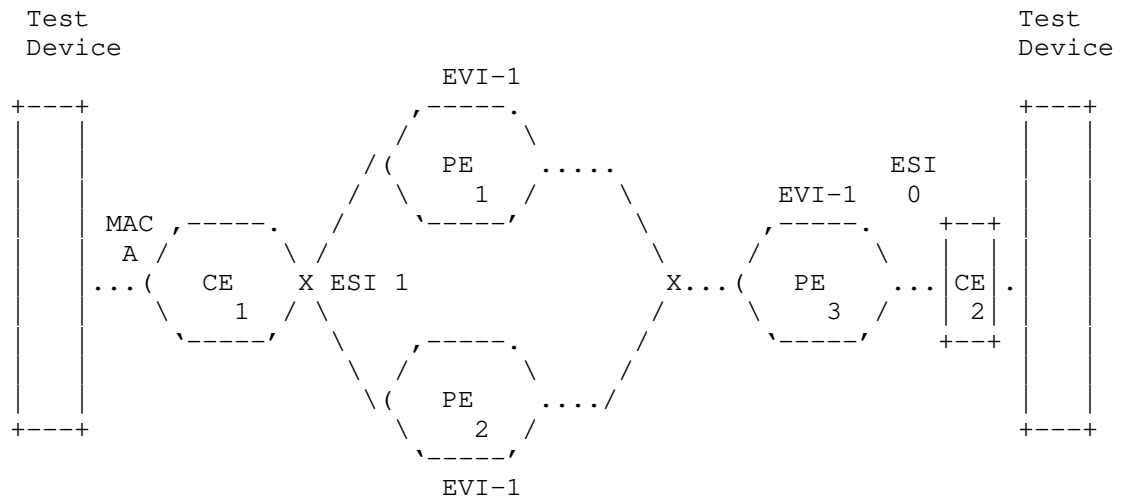


Figure 2 SUT with BGP & MPLS interconnecting multiple PE-ESI-CE locations

PE1 learns MAC A via data plane learning, PE1 and PE2 share ESI 1 (Ethernet Segment Identifier) and advertise an Ether A-D route with ESI 1 to PE3, PE1 also advertises MAC A to PE3. PE3 instantiates either Active/Backup or Active/Active towards PE1 and PE2 (Assume PE1 is Active in Active/Backup scenario) for MAC A.

All Link speeds MUST be reported, along with complete device configurations in the SUT and Test Device(s).

Additional Test Setups and configurations will be provided in this section, after review.

One capacity benchmark pertains to the number of ESIs that a network with multiple PE - ESI - CE locations can support.

4.1. Basic Configuration

This configuration serves as the base configuration for all test cases.

All routers except CE are configured with OSPF/IS-IS, LDP, MPLS, BGP with EVPN address family.

All routers except CE must have IBGP configured.

PE1, PE2, PE3 must be configured with an EVI context (EVI 1).

PE1 and PE2 must be configured with a non-zero ESI indicating that the two VLANs coming from CE1 belong to the same ethernet segment (ESI 1).

PE1 and PE2 are running Single Active mode of EVPN.

CE1 and CE2 are acting as bridges configured with VLANs that are configured on PE1, PE2, PE3.

In [RFC2889] procedures that follow, the test traffic will be bidirectional.

5. Procedure for Full Mesh Throughput Characterization

Objective: To characterize the ability of a DUT/SUT to process frames between CE and one or more PEs in a multihomed connectivity scenario. Figure 1 gives the least-complex test setup. Figure 2 gives a possible alternative with full BGP and MPLS interconnection.

The Procedure follows.

5.1. Address Learning Phase

"For every address, learning frames MUST be sent to the DUT/SUT to allow the DUT/SUT to update its address tables properly." [RFC2889]

5.2. Test for a Single Frame Size and Number of Unicast Flows

Each trial in the test requires configuring a number of flows (from 100 to 100k) and a fixed frame size (64 octets to 128, 256, 512, 1024, 1280 and 1518 bytes, as per [RFC2544]). Frame formats MUST be specified, they are as described in section 4 of [RFC2889].

Only one of frame size and number of flows SHALL change for each test.

5.3. Detailed Procedure

The Procedure SHALL follow section 5.1 of [RFC2889].

Specifically, the Throughput measurement parameters found in section 5.1.2 of [RFC2889] SHALL be configured and reported with the results.

The procedure for transmitting Frames on each port is described in section 5.1.3 of [RFC2889] and SHALL be followed (adapting to the number of ports in the test setup).

Once the traffic is started, the procedure for Measurements described in section 5.1.4 of [RFC2889] SHALL be followed (adapting to the number of ports in the test setup). The section on Throughput measurement (5.1.4 of [RFC2889]) SHALL be followed.

In the case that one or more of the CE and PE are virtual implementations, then the search algorithm of [TST009] that provides consistent results when faced with host transient activity SHOULD be used (Binary Search with Loss Verification).

5.4. Test Repetition

The test MUST be repeated N times for each frame size in the subset list, and each Throughput value made available for further processing (below).

5.5. Benchmark Calculations

For each Frame size and number of flows, calculate the following summary statistics for Throughput values over the N tests:

- o Average (Benchmark)
- o Minimum
- o Maximum
- o Standard Deviation

Comparison will determine how the load was balanced among PEs.

5.6. Reporting

The recommendation for graphical reporting provided in Section 5.1.4 of [RFC2889]) SHOULD be followed, along with the specifications in Section 7 below.

6. Procedure for Mass Withdrawal Characterization

Objective: To characterize the ability of a DUT/SUT to process frames between CE and one or more PE in a multihomed connectivity scenario when a mass withdrawal takes place. Figure 2 gives the test setup.

The Procedure follows.

6.1. Address Learning Phase

"For every address, learning frames MUST be sent to the DUT/SUT to allow the DUT/SUT update its address tables properly." [RFC2889]

6.2. Test for a Single Frame Size and Number of Flows

Each trial in the test requires Configuring a number of flows (from 100 to 100k) and a fixed frame size (64 octets to 128, 256, 512, 1024, 1280 and 1518 bytes, as per [RFC2544]).

Only one of frame size and number of flows SHALL change for each test.

The Offered Load SHALL be transmitted at the Throughput level corresponding to the level previously determined for the selected Frame size and number of Flows in use (see section 5).

The Procedure SHALL follow section 5.1 of [RFC2889] (except there is no need to search for the Throughput level). See section 5 above for additional requirements, especially section 5.3.

When traffic has been sent for 5 seconds one of the CE-PE links on the ESI SHALL be disabled, and the time of this action SHALL be recorded for further calculations. For example, if the CE1 link to PE1 is disabled, this should trigger a Mass withdrawal of EVI-1 addresses, and the subsequent re-routing of traffic to PE2.

Frame losses are expected to be recorded during the restoration time. Time for restoration may be estimated as described in section 3.5 of [RFC6412].

6.3. Test Repetition

The test MUST be repeated N times for each frame size in the subset list, and each restoration time value made available for further processing (below).

6.4. Benchmark Calculations

For each Frame size and number of flows, calculate the following summary statistics for Loss (or Time to return to Throughput level after restoration) values over the N tests:

- o Average (Benchmark)
- o Minimum

- o Maximum
- o Standard Deviation

7. Reporting

The results SHOULD be reported in the format of a table with a row for each of the tested frame sizes and Number of Flows. There SHOULD be columns for the frame size with number of flows, and for the resultant average frame count (or time) for each type of data stream tested.

The number of tests Averaged for the Benchmark, N, MUST be reported.

The Minimum, Maximum, and Standard Deviation across all complete tests SHOULD also be reported.

The Corrected DUT Restoration Time SHOULD also be reported, as applicable.

Frame Size, octets + # Flows	Ave Benchmark, fps, frames or time	Min,Max,StdDev	Calculated Time, Sec
64,100	26000	25500,27000,20	0.00004

Throughput or Loss/Restoration Time Results

Static and configuration parameters:

Number of test repetitions, N

Minimum Step Size (during searches), in frames.

8. Security Considerations

Benchmarking activities as described in this memo are limited to technology characterization using controlled stimuli in a laboratory environment, with dedicated address space and the other constraints [RFC2544].

The benchmarking network topology will be an independent test setup and MUST NOT be connected to devices that may forward the test traffic into a production network, or misroute traffic to the test management network. See [RFC6815].

Further, benchmarking is performed on a "black-box" basis, relying solely on measurements observable external to the DUT/SUT.

Special capabilities SHOULD NOT exist in the DUT/SUT specifically for benchmarking purposes. Any implications for network security arising from the DUT/SUT SHOULD be identical in the lab and in production networks.

9. IANA Considerations

This memo makes no requests of IANA.

10. Acknowledgements

Thanks to Sudhin Jacob for his review and comments on the bmwg-list.

Thanks to Aman Shaikh for sharing his comments on the draft directly with the authors.

11. References

11.1. Normative References

- [RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <<https://www.rfc-editor.org/info/rfc1242>>.
- [RFC1944] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 1944, DOI 10.17487/RFC1944, May 1996, <<https://www.rfc-editor.org/info/rfc1944>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [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>>.
- [RFC2889] Mandeville, R. and J. Perser, "Benchmarking Methodology for LAN Switching Devices", RFC 2889, DOI 10.17487/RFC2889, August 2000, <<https://www.rfc-editor.org/info/rfc2889>>.

- [RFC5180] Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, DOI 10.17487/RFC5180, May 2008, <<https://www.rfc-editor.org/info/rfc5180>>.
- [RFC6201] Asati, R., Pignataro, C., Calabria, F., and C. Olvera, "Device Reset Characterization", RFC 6201, DOI 10.17487/RFC6201, March 2011, <<https://www.rfc-editor.org/info/rfc6201>>.
- [RFC6412] Poretsky, S., Imhoff, B., and K. Michielsen, "Terminology for Benchmarking Link-State IGP Data-Plane Route Convergence", RFC 6412, DOI 10.17487/RFC6412, November 2011, <<https://www.rfc-editor.org/info/rfc6412>>.
- [RFC6815] Bradner, S., Dubray, K., McQuaid, J., and A. Morton, "Applicability Statement for RFC 2544: Use on Production Networks Considered Harmful", RFC 6815, DOI 10.17487/RFC6815, November 2012, <<https://www.rfc-editor.org/info/rfc6815>>.
- [RFC6985] Morton, A., "IMIX Genome: Specification of Variable Packet Sizes for Additional Testing", RFC 6985, DOI 10.17487/RFC6985, July 2013, <<https://www.rfc-editor.org/info/rfc6985>>.
- [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>>.

11.2. Informative References

- [OPNFV-2017] Cooper, T., Morton, A., and S. Rao, "Dataplane Performance, Capacity, and Benchmarking in OPNFV", June 2017, <<https://wiki.opnfv.org/download/attachments/10293193/VSPERF-Dataplane-Perf-Cap-Bench.pptx?api=v2>>.
- [RFC8239] Avramov, L. and J. Rapp, "Data Center Benchmarking Methodology", RFC 8239, DOI 10.17487/RFC8239, August 2017, <<https://www.rfc-editor.org/info/rfc8239>>.

[TST009] Morton, R. A., "ETSI GS NFV-TST 009 V3.2.1 (2019-06),
"Network Functions Virtualisation (NFV) Release 3;
Testing; Specification of Networking Benchmarks and
Measurement Methods for NFVI", June 2019,
<https://www.etsi.org/deliver/etsi_gs/NFV-TST/001_099/009/03.01.01_60/gs_NFV-TST009v030101p.pdf>.

[VSPERF-b2b] Morton, A., "Back2Back Testing Time Series (from CI)",
June 2017, <[https://wiki.opnfv.org/display/vsperf/Traffic+Generator+Testing#TrafficGeneratorTesting-AppendixB:Back2BackTestingTimeSeries\(fromCI\)](https://wiki.opnfv.org/display/vsperf/Traffic+Generator+Testing#TrafficGeneratorTesting-AppendixB:Back2BackTestingTimeSeries(fromCI))>.

[VSPERF-BSLV] Morton, A. and S. Rao, "Evolution of Repeatability in
Benchmarking: Fraser Plugfest (Summary for IETF BMWG)",
July 2018,
<<https://datatracker.ietf.org/meeting/102/materials/slides-102-bmwg-evolution-of-repeatability-in-benchmarking-fraser-plugfest-summary-for-ietf-bmwg-00>>.

Authors' Addresses

Al Morton
AT&T Labs
200 Laurel Avenue South
Middletown,, NJ 07748
USA

Phone: +1 732 420 1571
Fax: +1 732 368 1192
Email: acm@research.att.com

Jim Uttaro
AT&T Labs
200 Laurel Avenue South
Middletown,, NJ 07748
USA

Email: uttaro@att.com

Network Working Group
Internet-Draft
Updates: ???? (if approved)
Intended status: Informational
Expires: May 6, 2021

A. Morton
J. Uttaro
AT&T Labs
November 2, 2020

Benchmarks and Methods for Multihomed EVPN
draft-morton-bmwg-multihome-evpn-04

Abstract

Fundamental Benchmarking Methodologies for Network Interconnect Devices of interest to the IETF are defined in RFC 2544. Key benchmarks applicable to restoration and multi-homed sites are in RFC 6894. This memo applies these methods to Multihomed nodes implemented on Ethernet Virtual Private Networks (EVPN).

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14[RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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 May 6, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Scope and Goals	3
3. Motivation	3
4. Test Setups	3
4.1. Basic Configuration	5
5. Procedure for Full Mesh Throughput Characterization	6
5.1. Address Learning Phase	6
5.2. Test for a Single Frame Size and Number of Unicast Flows	6
5.3. Detailed Procedure	6
5.4. Test Repetition	7
5.5. Benchmark Calculations	7
5.6. Reporting	7
6. Procedure for Mass Withdrawal Characterization	7
6.1. Address Learning Phase	8
6.2. Test for a Single Frame Size and Number of Flows	8
6.3. Test Repetition	8
6.4. Benchmark Calculations	8
7. Reporting	9
8. Security Considerations	9
9. IANA Considerations	10
10. Acknowledgements	10
11. References	10
11.1. Normative References	10
11.2. Informative References	11
Authors' Addresses	12

1. Introduction

The IETF's fundamental Benchmarking Methodologies are defined in[RFC2544], supported by the terms and definitions in [RFC1242], and [RFC2544] actually obsoletes an earlier specification, [RFC1944].

This memo recognizes the importance of Ethernet Virtual Private Network (EVPN) Multihoming connectivity scenarios, where a CE device is connected to 2 or more PEs using an instance of an Ethernet Segment.

In an all-active or Active-Active scenario, CE-PE traffic is load-balanced across two or more PEs.

Mass-withdrawal of routes may take place when an autodiscovery route is used on a per Ethernet Segment basis, and there is a link failure on one of the Ethernet Segment links (or when configuration changes take place).

Although EVPN depends on address-learning in the control-plane, the Ethernet Segment Instance is permitted to use "the method best suited to the CE: data-plane learning, IEEE 802.1x, the Link Layer Discovery Protocol (LLDP), IEEE 802.1aq, Address Resolution Protocol (ARP), management plane, or other protocols" [RFC7432].

This memo seeks to benchmark these important cases (and others).

2. Scope and Goals

The scope of this memo is to define a method to unambiguously perform tests, measure the benchmark(s), and report the results for Capacity of EVPN Multihoming connectivity scenarios, and other key restoration activities (such as address withdrawl) covering link failure in the Active-Active scenario.

The goal is to provide more efficient test procedures where possible, and to expand reporting with additional interpretation of the results. The tests described in this memo address some key multihoming scenarios implemented on a Device Under Test (DUT) or System Under Test (SUT).

3. Motivation

The Multihoming scenarios described in this memo emphasize features with practical value to the industry that have seen deployment. Therefore, these scenarios deserve further attention that follows from benchmarking activities and further study.

4. Test Setups

For simple Capacity/Throughput Benchmarks, the Test Setup MUST be consistent with Figure 1 of [RFC2544], or Figure 2 when the tester's sender and receiver are different devices.

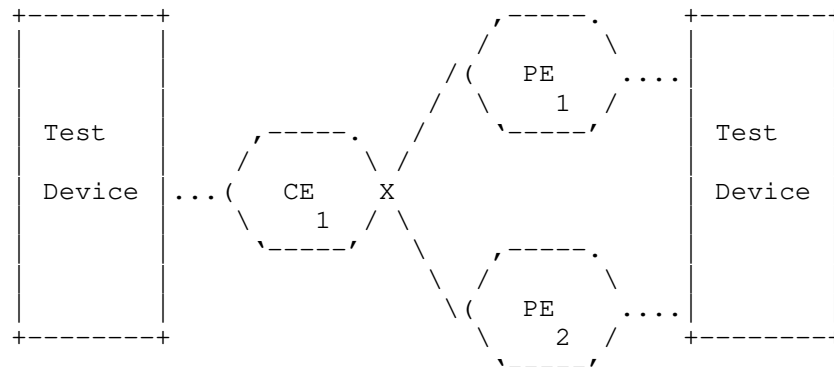


Figure 1 SUT for Throughput and other Ethernet Segment Tests

In Figure 1, the System Under Test (SUT) is comprised of a single CE device and two or more PE devices.

The tester SHALL be connected to all CE and every PE, and be capable of simultaneously sending and receiving frames on all ports with connectivity. The tester SHALL be capable of generating multiple flows (according to a 5-tuple definition, or any sub-set of the 5-tuple). The tester SHALL be able to control the IP capacity of sets of individual flows, and the presence of sets of flows on specific interface ports.

The tester SHALL be capable of generating and receiving a full mesh of Unicast flows, as described in section 3.0 of [RFC2889]:

"In fully meshed traffic, each interface of a DUT/SUT is set up to both receive and transmit frames to all the other interfaces under test."

Other mandatory testing aspects described in [RFC2544] and [RFC2889] MUST be included, unless explicitly modified in the next section.

The ingress and egress link speeds and link layer protocols MUST be specified and used to compute the maximum theoretical frame rate when respecting the minimum inter-frame gap.

A second test case is where a BGP backbone implements MPLS-LDP to provide connectivity between multiple PE - ESI - CE locations.

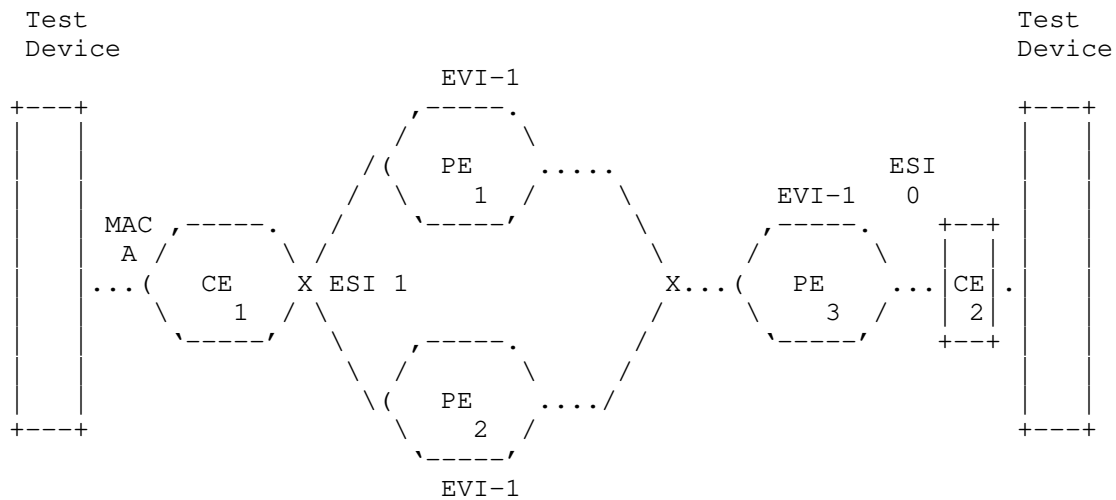


Figure 2 SUT with BGP & MPLS interconnecting multiple PE-ESI-CE locations

PE1 learns MAC A via data plane learning, PE1 and PE2 share ESI 1 (Ethernet Segment Identifier) and advertise an Ether A-D route with ESI 1 to PE3, PE1 also advertises MAC A to PE3. PE3 instantiates either Active/Backup or Active/Active towards PE1 and PE2 (Assume PE1 is Active in Active/Backup scenario) for MAC A.

All Link speeds MUST be reported, along with complete device configurations in the SUT and Test Device(s).

Additional Test Setups and configurations will be provided in this section, after review.

One capacity benchmark pertains to the number of ESIs that a network with multiple PE - ESI - CE locations can support.

4.1. Basic Configuration

This configuration serves as the base configuration for all test cases.

All routers except CE are configured with OSPF/IS-IS, LDP, MPLS, BGP with EVPN address family.

All routers except CE must have IBGP configured.

PE1, PE2, PE3 must be configured with an EVI context (EVI 1).

PE1 and PE2 must be configured with a non-zero ESI indicating that the two VLANs coming from CE1 belong to the same ethernet segment (ESI 1).

PE1 and PE2 are running Single Active mode of EVPN.

CE1 and CE2 are acting as bridges configured with VLANs that are configured on PE1, PE2, PE3.

In [RFC2889] procedures that follow, the test traffic will be bidirectional.

5. Procedure for Full Mesh Throughput Characterization

Objective: To characterize the ability of a DUT/SUT to process frames between CE and one or more PEs in a multihomed connectivity scenario. Figure 1 gives the least-complex test setup. Figure 2 gives a possible alternative with full BGP and MPLS interconnection.

The Procedure follows.

5.1. Address Learning Phase

"For every address, learning frames MUST be sent to the DUT/SUT to allow the DUT/SUT to update its address tables properly." [RFC2889]

5.2. Test for a Single Frame Size and Number of Unicast Flows

Each trial in the test requires configuring a number of flows (from 100 to 100k) and a fixed frame size (64 octets to 128, 256, 512, 1024, 1280 and 1518 bytes, as per [RFC2544]). Frame formats MUST be specified, they are as described in section 4 of [RFC2889].

Only one of frame size and number of flows SHALL change for each test.

5.3. Detailed Procedure

The Procedure SHALL follow section 5.1 of [RFC2889].

Specifically, the Throughput measurement parameters found in section 5.1.2 of [RFC2889] SHALL be configured and reported with the results.

The procedure for transmitting Frames on each port is described in section 5.1.3 of [RFC2889] and SHALL be followed (adapting to the number of ports in the test setup).

Once the traffic is started, the procedure for Measurements described in section 5.1.4 of [RFC2889] SHALL be followed (adapting to the number of ports in the test setup). The section on Throughput measurement (5.1.4 of [RFC2889]) SHALL be followed.

In the case that one or more of the CE and PE are virtual implementations, then the search algorithm of [TST009] that provides consistent results when faced with host transient activity SHOULD be used (Binary Search with Loss Verification).

5.4. Test Repetition

The test MUST be repeated N times for each frame size in the subset list, and each Throughput value made available for further processing (below).

5.5. Benchmark Calculations

For each Frame size and number of flows, calculate the following summary statistics for Throughput values over the N tests:

- o Average (Benchmark)
- o Minimum
- o Maximum
- o Standard Deviation

Comparison will determine how the load was balanced among PEs.

5.6. Reporting

The recommendation for graphical reporting provided in Section 5.1.4 of [RFC2889]) SHOULD be followed, along with the specifications in Section 7 below.

6. Procedure for Mass Withdrawal Characterization

Objective: To characterize the ability of a DUT/SUT to process frames between CE and one or more PE in a multihomed connectivity scenario when a mass withdrawal takes place. Figure 2 gives the test setup.

The Procedure follows.

6.1. Address Learning Phase

"For every address, learning frames MUST be sent to the DUT/SUT to allow the DUT/SUT update its address tables properly." [RFC2889]

6.2. Test for a Single Frame Size and Number of Flows

Each trial in the test requires configuring a number of flows (from 100 to 100k) and a fixed frame size (64 octets to 128, 256, 512, 1024, 1280 and 1518 bytes, as per [RFC2544]).

Only one of frame size and number of flows SHALL change for each test.

The Offered Load SHALL be transmitted at the Throughput level corresponding to the level previously determined for the selected Frame size and number of Flows in use (see section 5).

The Procedure SHALL follow section 5.1 of [RFC2889] (except there is no need to search for the Throughput level). See section 5 above for additional requirements, especially section 5.3.

When traffic has been sent for 5 seconds one of the CE-PE links on the ESI SHALL be disabled, and the time of this action SHALL be recorded for further calculations. For example, if the CE1 link to PE1 is disabled, this should trigger a Mass withdrawal of EVI-1 addresses, and the subsequent re-routing of traffic to PE2.

Frame losses are expected to be recorded during the restoration time. Time for restoration may be estimated as described in section 3.5 of [RFC6412].

6.3. Test Repetition

The test MUST be repeated N times for each frame size in the subset list, and each restoration time value made available for further processing (below).

6.4. Benchmark Calculations

For each Frame size and number of flows, calculate the following summary statistics for Loss (or Time to return to Throughput level after restoration) values over the N tests:

- o Average (Benchmark)
- o Minimum

- o Maximum
- o Standard Deviation

7. Reporting

The results SHOULD be reported in the format of a table with a row for each of the tested frame sizes and Number of Flows. There SHOULD be columns for the frame size with number of flows, and for the resultant average frame count (or time) for each type of data stream tested.

The number of tests Averaged for the Benchmark, N, MUST be reported.

The Minimum, Maximum, and Standard Deviation across all complete tests SHOULD also be reported.

The Corrected DUT Restoration Time SHOULD also be reported, as applicable.

Frame Size, octets + # Flows	Ave Benchmark, fps, frames or time	Min,Max,StdDev	Calculated Time, Sec
64,100	26000	25500,27000,20	0.00004

Throughput or Loss/Restoration Time Results

Static and configuration parameters:

Number of test repetitions, N

Minimum Step Size (during searches), in frames.

8. Security Considerations

Benchmarking activities as described in this memo are limited to technology characterization using controlled stimuli in a laboratory environment, with dedicated address space and the other constraints [RFC2544].

The benchmarking network topology will be an independent test setup and MUST NOT be connected to devices that may forward the test traffic into a production network, or misroute traffic to the test management network. See [RFC6815].

Further, benchmarking is performed on a "black-box" basis, relying solely on measurements observable external to the DUT/SUT.

Special capabilities SHOULD NOT exist in the DUT/SUT specifically for benchmarking purposes. Any implications for network security arising from the DUT/SUT SHOULD be identical in the lab and in production networks.

9. IANA Considerations

This memo makes no requests of IANA.

10. Acknowledgements

Thanks to Sudhin Jacob for his review and comments on the bmwg-list.

Thanks to Aman Shaikh for sharing his comments on the draft directly with the authors.

11. References

11.1. Normative References

- [RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <<https://www.rfc-editor.org/info/rfc1242>>.
- [RFC1944] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 1944, DOI 10.17487/RFC1944, May 1996, <<https://www.rfc-editor.org/info/rfc1944>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [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>>.
- [RFC2889] Mandeville, R. and J. Perser, "Benchmarking Methodology for LAN Switching Devices", RFC 2889, DOI 10.17487/RFC2889, August 2000, <<https://www.rfc-editor.org/info/rfc2889>>.

- [RFC5180] Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, DOI 10.17487/RFC5180, May 2008, <<https://www.rfc-editor.org/info/rfc5180>>.
- [RFC6201] Asati, R., Pignataro, C., Calabria, F., and C. Olvera, "Device Reset Characterization", RFC 6201, DOI 10.17487/RFC6201, March 2011, <<https://www.rfc-editor.org/info/rfc6201>>.
- [RFC6412] Poretsky, S., Imhoff, B., and K. Michielsen, "Terminology for Benchmarking Link-State IGP Data-Plane Route Convergence", RFC 6412, DOI 10.17487/RFC6412, November 2011, <<https://www.rfc-editor.org/info/rfc6412>>.
- [RFC6815] Bradner, S., Dubray, K., McQuaid, J., and A. Morton, "Applicability Statement for RFC 2544: Use on Production Networks Considered Harmful", RFC 6815, DOI 10.17487/RFC6815, November 2012, <<https://www.rfc-editor.org/info/rfc6815>>.
- [RFC6985] Morton, A., "IMIX Genome: Specification of Variable Packet Sizes for Additional Testing", RFC 6985, DOI 10.17487/RFC6985, July 2013, <<https://www.rfc-editor.org/info/rfc6985>>.
- [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>>.

11.2. Informative References

- [OPNFV-2017] Cooper, T., Morton, A., and S. Rao, "Dataplane Performance, Capacity, and Benchmarking in OPNFV", June 2017, <<https://wiki.opnfv.org/download/attachments/10293193/VSPERF-Dataplane-Perf-Cap-Bench.pptx?api=v2>>.
- [RFC8239] Avramov, L. and J. Rapp, "Data Center Benchmarking Methodology", RFC 8239, DOI 10.17487/RFC8239, August 2017, <<https://www.rfc-editor.org/info/rfc8239>>.

[TST009] Morton, R. A., "ETSI GS NFV-TST 009 V3.2.1 (2019-06),
"Network Functions Virtualisation (NFV) Release 3;
Testing; Specification of Networking Benchmarks and
Measurement Methods for NFVI"", June 2019,
<https://www.etsi.org/deliver/etsi_gs/NFV-TST/001_099/009/03.01.01_60/gs_NFV-TST009v030101p.pdf>.

[VSPERF-b2b] Morton, A., "Back2Back Testing Time Series (from CI)",
June 2017, <[https://wiki.opnfv.org/display/vsperf/Traffic+Generator+Testing#TrafficGeneratorTesting-AppendixB:Back2BackTestingTimeSeries\(fromCI\)](https://wiki.opnfv.org/display/vsperf/Traffic+Generator+Testing#TrafficGeneratorTesting-AppendixB:Back2BackTestingTimeSeries(fromCI))>.

[VSPERF-BSLV] Morton, A. and S. Rao, "Evolution of Repeatability in
Benchmarking: Fraser Plugfest (Summary for IETF BMWG)",
July 2018,
<<https://datatracker.ietf.org/meeting/102/materials/slides-102-bmwg-evolution-of-repeatability-in-benchmarking-fraser-plugfest-summary-for-ietf-bmwg-00>>.

Authors' Addresses

Al Morton
AT&T Labs
200 Laurel Avenue South
Middletown,, NJ 07748
USA

Phone: +1 732 420 1571
Fax: +1 732 368 1192
Email: acm@research.att.com

Jim Uttaro
AT&T Labs
200 Laurel Avenue South
Middletown,, NJ 07748
USA

Email: uttaro@att.com

BMWG
Internet-Draft
Intended status: Informational
Expires: May 7, 2020

R. Rosa, Ed.
C. Rothenberg
UNICAMP
M. Peuster
H. Karl
UPB
November 4, 2019

Methodology for VNF Benchmarking Automation
draft-rosa-bmwg-vnfbench-05

Abstract

This document describes a common methodology for the automated benchmarking of Virtualized Network Functions (VNFs) executed on general-purpose hardware. Specific cases of automated benchmarking methodologies for particular VNFs can be derived from this document. Two open source reference implementations are reported as running code embodiments of the proposed, automated benchmarking methodology.

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 May 7, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	4
3. Scope	4
4. Considerations	4
4.1. VNF Testing Methods	5
4.2. Benchmarking Procedures	5
4.2.1. Phase I: Deployment	6
4.2.2. Phase II: Configuration	6
4.2.3. Phase III: Execution	6
4.2.4. Phase IV: Report	7
5. Generic VNF Benchmarking Architectural Framework	7
5.1. Deployment Scenarios	10
6. Methodology	10
6.1. VNF Benchmarking Descriptor (VNF-BD)	12
6.1.1. Descriptor Headers	12
6.1.2. Target Information	12
6.1.3. Experiments	13
6.1.4. Environment	13
6.1.5. Scenario	13
6.1.6. Proceedings	14
6.2. VNF Performance Profile (VNF-PP)	15
6.2.1. Headers	15
6.2.2. Reports	15
6.3. Procedures	17
6.3.1. Pre-Execution	17
6.3.2. Automated Execution	18
6.3.3. Post-Execution	19
6.4. Particular Cases	19
6.4.1. Capacity	19
6.4.2. Redundancy	20
6.4.3. Isolation	20
6.4.4. Failure Handling	20
6.4.5. Elasticity and Flexibility	20
6.4.6. Handling Configurations	21
6.4.7. White Box VNF	21
7. Open Source Reference Implementations	21
7.1. Gym	22
7.2. tng-bench	22
8. Security Considerations	23
9. IANA Considerations	24
10. Acknowledgement	24

11. References	24
11.1. Normative References	24
11.2. Informative References	25
Authors' Addresses	26

1. Introduction

The Benchmarking Methodology Working Group (BMWG) already presented considerations for benchmarking of VNFs and their infrastructure in [RFC8172]. Similar to the motivation given in [RFC8172], the following aspects justify the need for VNF benchmarking: (i) pre-deployment infrastructure dimensioning to realize associated VNF performance profiles; (ii) comparison factor with physical network functions; (iii) and output results for analytical VNF development.

Even if many methodologies already described by the BMWG, e.g., self-contained black-box benchmarking, can be applied to VNF benchmarking scenarios, further considerations have to be made. This is, on the one hand, because VNFs, which are software components, do not have strict and clear execution boundaries and depend on underlying virtualization environment parameters as well as management and orchestration decisions [ETS14a]. On the other hand, can and should the flexible, software-based nature of VNFs be exploited to fully automate the entire benchmarking procedure end-to-end. This is an inherent need to align VNF benchmarking with the agile methods enabled by the concept of Network Functions Virtualization (NFV) [ETS14e]. More specifically it allows: (i) the development of agile performance-focused DevOps methodologies for Continuous Integration and Delivery (CI/CD) of VNFs; (ii) the creation of on-demand VNF test descriptors for upcoming execution environments; (iii) the path for precise-analytics of automated catalogues of VNF performance profiles; (iv) and run-time mechanisms to assist VNF lifecycle orchestration/management workflows, e.g., automated resource dimensioning based on benchmarking insights.

This document describes basic methodologies and guidelines to fully automate VNF benchmarking procedures, without limiting the automated process to a specific benchmark or infrastructure. After presenting initial considerations, the document first describes a generic architectural framework to setup automated benchmarking experiments. Second, the automation methodology is discussed, with a particular focus on experiment and procedure description approaches to support reproducibility of the automated benchmarks, a key challenge in VNF benchmarking. Finally, two independent, open-source reference implementations are presented. The document addresses state-of-the-art work on VNF benchmarking from scientific publications and current developments in other standardization bodies (e.g., [ETS14c], [ETS19f] and [RFC8204]) wherever possible.

2. Terminology

Common benchmarking terminology contained in this document is derived from [RFC1242]. The reader is assumed to be familiar with the terminology as defined in the European Telecommunications Standards Institute (ETSI) NFV document [ETSI14b]. Some of these terms, and others commonly used in this document, are defined below.

NFV: Network Function Virtualization - the principle of separating network functions from the hardware they run on by using virtual hardware abstraction.

VNF: Virtualized Network Function - a software-based network function. A VNF can be either represented by a single entity or be composed by a set of smaller, interconnected software components, called VNF components (VNFCs) [ETSI14d]. Those VNFs are also called composed VNFs.

VNFC: Virtualized Network Function Component - a software component that implements (parts of) the VNF functionality. A VNF can consist of a single VNFC or multiple, interconnected VNFCs [ETSI14d]

VNFD: Virtualised Network Function Descriptor - configuration template that describes a VNF in terms of its deployment and operational behaviour, and is used in the process of VNF onboarding and managing the life cycle of a VNF instance.

NS: Network Service - a collection of interconnected VNFs forming a end-to-end service. The interconnection is often done using chaining of functions.

3. Scope

This document assumes VNFs as black boxes when defining their benchmarking methodologies. White box approaches are assumed and analysed as a particular case under the proper considerations of internal VNF instrumentation, later discussed in this document.

This document outlines a methodology for VNF benchmarking, specifically addressing its automation.

4. Considerations

VNF benchmarking considerations are defined in [RFC8172]. Additionally, VNF pre-deployment testing considerations are well explored in [ETSI14c]. Further, ETSI provides test specifications for networking benchmarks and measurement methods for NFV infrastructure

in [ETSI19f] which complements the presented work on VNF benchmarking methodologies.

4.1. VNF Testing Methods

Following ETSI's model in [ETSI14c], we distinguish three methods for VNF evaluation:

Benchmarking: Where parameters (e.g., CPU, memory, storage) are provided and the corresponding performance metrics (e.g., latency, throughput) are obtained. Note, such evaluations might create multiple reports, for example, with minimal latency or maximum throughput results.

Verification: Both parameters and performance metrics are provided and a stimulus verifies if the given association is correct or not.

Dimensioning: Performance metrics are provided and the corresponding parameters obtained. Note, multiple deployments may be required, or if possible, underlying allocated resources need to be dynamically altered.

Note: Verification and Dimensioning can be reduced to Benchmarking. Therefore, we focus on Benchmarking in the rest of the document.

4.2. Benchmarking Procedures

A (automated) benchmarking procedure can be divided into three sub-procedures:

Trial: Is a single process or iteration to obtain VNF performance metrics from benchmarking measurements. A Test should always run multiple Trials to get statistical confidence about the obtained measurements.

Test: Defines unique structural and functional parameters (e.g., configurations, resource assignment) for benchmarked components to perform one or multiple Trials. Each Test must be executed following a particular benchmarking scenario composed by a Method. Proper measures must be taken to ensure statistical validity (e.g., independence across Trials of generated load patterns).

Method: Consists of one or more Tests to benchmark a VNF. A Method can explicitly list ranges of parameter values for the configuration of a benchmarking scenario and its components. Each value of such a range is to be realized in a Test. I.e., Methods can define parameter studies.

In general, automated VNF benchmarking Tests must capture relevant causes of performance variability. To dissect a VNF benchmarking Test, in the sections that follow different benchmarking phases are categorized defining generic operations that may be automated. When automating a VNF benchmarking methodology, all the influencing aspects on the performance of a VNF must be carefully analyzed and comprehensively reported, in each phase of the overall benchmarking process.

4.2.1. Phase I: Deployment

The placement (i.e., assignment and allocation of resources) and the interconnection, physical and/or virtual, of network function(s) and benchmarking components can be realized by orchestration platforms (e.g., OpenStack, Kubernetes, Open Source MANO). In automated manners, the realization of a benchmarking testbed/scenario through those means usually rely on network service templates (e.g., TOSCA, Heat, YANG). Such descriptors have to capture all relevant details of the execution environment to allow the benchmarking framework to correctly instantiate the SUT as well as helper functions required for a Test.

4.2.2. Phase II: Configuration

The configuration of benchmarking components and VNFs (e.g., populate routing table, load PCAP source files in source of traffic stimulus) to execute the Test settings can be realized by programming interfaces in an automated way. In the scope of NFV, there might exist management interfaces to control a VNF during a benchmarking Test. Likewise, infrastructure or orchestration components can establish the proper configuration of an execution environment to realize all the capabilities enabling the description of the benchmarking Test. Each configuration registry, its deployment timestamp and target, must all be contained in the VNF benchmarking report.

4.2.3. Phase III: Execution

In the execution of a benchmarking Test, the VNF configuration can be programmed to be changed by itself or by a VNF management platform. It means that during a Trial execution, particular behaviors of a VNF can be automatically triggered, e.g., auto-scaling of its internal components. Those must be captured in the detailed procedures of the VNF execution and its performance report. I.e., the execution of a Trial can determine arrangements of internal states inside a VNF, which can interfere in observed benchmarking metrics. For instance, in a particular benchmarking case where the monitoring measurements of the VNF and/or execution environment are available for extraction,

Tests should be run to verify if the monitoring of the VNF and/or execution environment can impact the VNF performance metrics.

4.2.4. Phase IV: Report

The report of a VNF benchmarking Method might contain generic metrics (e.g., CPU and memory consumption) and VNF-specific traffic processing metrics (e.g., transactions or throughput), which can be stored and processed in generic or specific ways (e.g., by statistics or machine learning algorithms). More details about possible metrics and the corresponding capturing methods can be found in [ETS19g]. If automated procedures are applied over the generation of a benchmarking report, those must be detailed in the report itself, jointly with their input raw measurements and output processed data. I.e., any algorithm used in the generation of processed metrics must be disclosed in the report.

5. Generic VNF Benchmarking Architectural Framework

A generic VNF benchmarking architectural framework, shown in Figure 1, establishes the disposal of essential components and control interfaces, explained below, that enable the automation of VNF benchmarking methodologies.

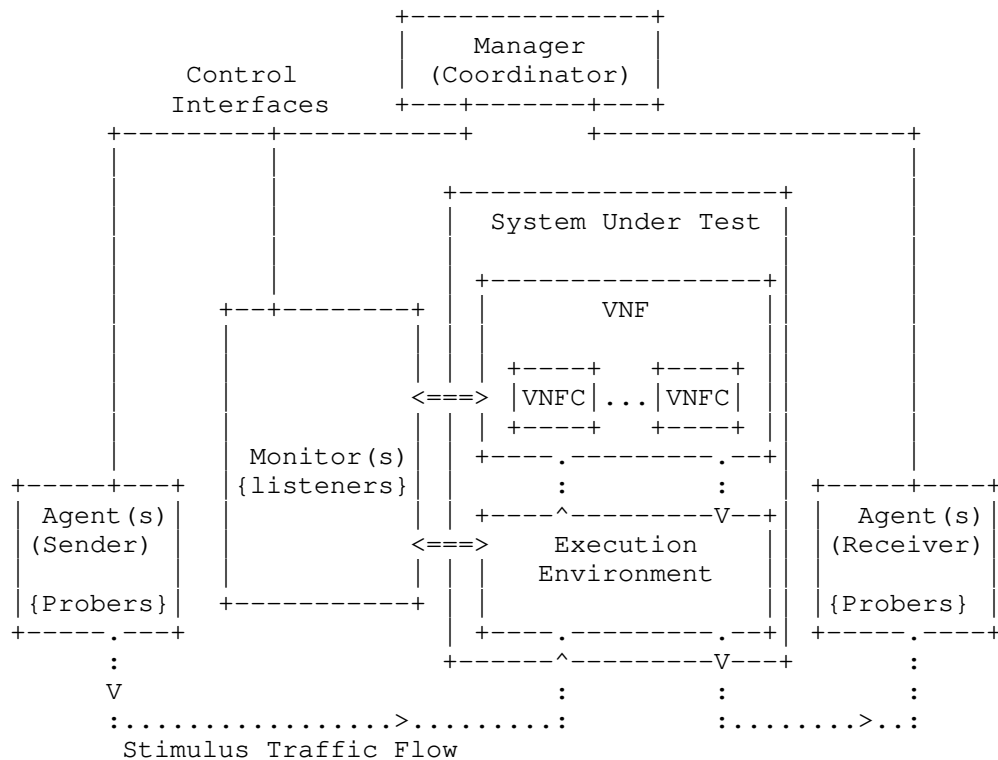


Figure 1: Generic VNF Benchmarking Setup

Virtualized Network Function (VNF) -- consists of one or more software components, so called VNF components (VNFC), adequate for performing a network function according to allocated virtual resources and satisfied requirements in an execution environment. A VNF can demand particular configurations for benchmarking specifications, demonstrating variable performance based on available virtual resources/parameters and configured enhancements targeting specific technologies (e.g., NUMA, SR-IOV, CPU-Pinning).

Execution Environment -- defines a virtualized and controlled composition of capabilities necessary for the execution of a VNF. An execution environment stands as a general purpose level of virtualization with abstracted resources available for one or more VNFs. It can also define specific technology habilitation, incurring in viable settings for enhancing the performance of VNFs.

Agent (Active Probing) -- executes active stimulus using probers, i.e., benchmarking tools, to benchmark and collect network and system performance metrics. A single Agent can perform localized benchmarks in execution environments (e.g., stress tests on CPU, memory, disk I/O) or can generate stimulus traffic and the other end be the VNF itself where, for example, one-way latency is evaluated. The interaction among distributed Agents enable the generation and collection of end-to-end metrics (e.g., frame loss rate, latency) measured from stimulus traffic flowing through a VNF. An Agent can be defined by a physical or virtual network function.

Prober -- defines an abstraction layer for a software or hardware tool able to generate stimulus traffic to a VNF or perform stress tests on execution environments. Probers might be specific or generic to an execution environment or a VNF. For an Agent, a Prober must provide programmable interfaces for its life cycle management, e.g., configuration of operational parameters, execution of stilumus, parsing of extracted metrics, and debugging options. Specific Probers might be developed to abstract and to realize the description of particular VNF benchmarking methodologies.

Monitor (Passive Probing) -- when possible is instantiated inside the System Under Test, VNF and/or infrastructure (e.g., as a plug-in process in a virtualized execution environment), to perform the passive monitoring, using Listeners, for the extraction of metrics while Agents' stimuli takes place. Monitors observe particular properties according to the execution environment and VNFs capabilities, i.e., exposed passive monitoring interfaces. Multiple Listeners can be executed at once in synchrony with a Prober' stimulus on a SUT. A Monitor can be defined as a virtualized network function.

Listener -- defines one or more software interfaces for the extraction of metrics monitored in a target VNF and/or execution environment. A Listener must provide programmable interfaces for its life cycle management workflows, e.g., configuration of operational parameters, execution of passive monitoring captures, parsing of extracted metrics, and debugging options (also see [ETS19g]). Varied methods of passive performance monitoring might be implemented as a Listener, depending on the interfaces exposed by the VNF and/or execution environment.

Manager -- performs (i) the discovery of available Agents/Monitors and their respective features (i.e., available Probers/Listeners and execution environment capabilities), (ii) the coordination and synchronization of activities of Agents and Monitors to perform a benchmarking Test, (iii) the collection, processing and aggregation of all VNF benchmarking measurements that correlates the VNF stimuli and the, possible, SUT monitored metrics. A Manager executes the main configuration, operation, and management actions to deliver the VNF benchmarking report. Hence, it detains interfaces open for users interact with the whole benchmarking framework, realizing, for instance, the retrieval of the framework characteristics (e.g., available benchmarking components and their probers/listeners), the coordination of benchmarking tasks, the processing and the retrieval of benchmarking metrics, among other operational and management functionalities. A Manager can be defined as a physical or virtualized network function.

5.1. Deployment Scenarios

A deployment scenario realizes the actual instantiation of physical and/or virtual components of a Generic VNF Benchmarking Architectural Framework needed to habilitate the execution of an automated VNF benchmarking methodology. The following considerations hold for a deployment scenario:

- o Not all components are mandatory for a Test, possible to be disposed in varied settings.
- o Components can be composed in a single entity and be defined as black or white boxes. For instance, Manager and Agents could jointly define one hardware/software entity to perform a VNF benchmarking Test and present measurement results.
- o Monitor can be defined by multiple instances of software components, each addressing a VNF or execution environment.
- o Agents can be disposed in varied topology setups, included the possibility of multiple input and output ports of a VNF being directly connected each in one Agent.
- o All benchmarking components defined in a deployment scenario must perform the synchronization of clocks.

6. Methodology

Portability is an intrinsic characteristic of VNFs and allows them to be deployed in multiple environments. This enables various benchmarking setups in varied deployment scenarios. A VNF

benchmarking methodology must be described in a clear and objective manner following four basic principles:

- o **Comparability:** Output of Tests shall be simple to understand and process, in a human-readable format, coherent, and easily reusable (e.g., inputs for analytic applications).
- o **Repeatability:** A Test setup shall be comprehensively defined through a flexible design model that can be interpreted and executed by the testing platform repeatedly but supporting customization.
- o **Configurability:** Open interfaces and extensible messaging models shall be available between benchmarking components for flexible composition of Test descriptors and platform configurations.
- o **Interoperability:** Tests shall be ported to different environments using lightweight components.

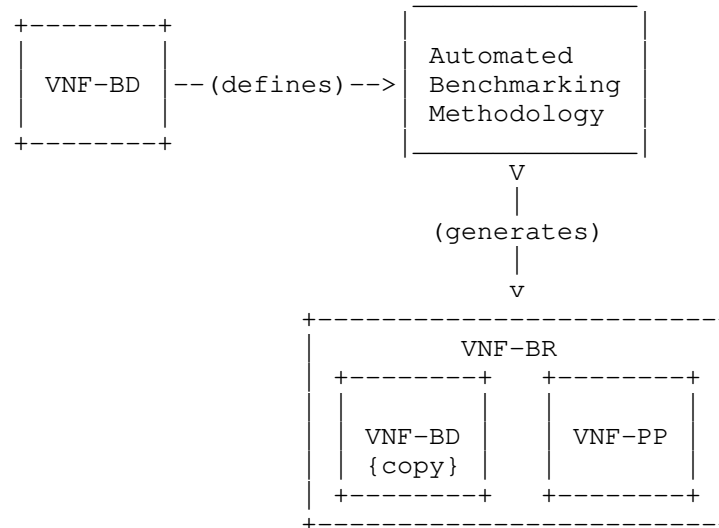


Figure 2: VNF benchmarking process inputs and outputs

As shown in Figure 2, the outcome of an automated VNF benchmarking methodology, must be captured in a VNF Benchmarking Report (VNF-BR), consisting of two parts:

VNF Benchmarking Descriptor (VNF-BD) -- contains all required definitions and requirements to deploy, configure, execute, and reproduce VNF benchmarking tests. VNF-BDs are defined by the developer of a benchmarking methodology and serve as input to the benchmarking process, before being included in the generated VNF-BR.

VNF Performance Profile (VNF-PP) -- contains all measured metrics resulting from the execution of a benchmarking. Additionally, it might also contain additional recordings of configuration parameters used during the execution of the benchmarking scenario to facilitate comparability of VNF-BRs.

A VNF-BR correlates structural and functional parameters of VNF-BD with extracted VNF benchmarking metrics of the obtained VNF-PP. The content of each part of a VNF-BR is described in the following sections.

6.1. VNF Benchmarking Descriptor (VNF-BD)

VNF Benchmarking Descriptor (VNF-BD) -- an artifact that specifies a Method of how to measure a VNF Performance Profile. The specification includes structural and functional instructions and variable parameters at different abstraction levels (e.g., topology of the deployment scenario, benchmarking target metrics, parameters of benchmarking components). A VNF-BD may be specific to a VNF or applicable to several VNF types. It can be used to elaborate a VNF benchmark deployment scenario aiming at the extraction of particular VNF performance metrics. An initial YANG model for VNF-BDs is available at [yang-vnf-bd].

The following items define the VNF-BD contents.

6.1.1. Descriptor Headers

The definition of parameters concerning the descriptor file, e.g., its version, identifier, name, author and description.

6.1.2. Target Information

General information addressing the target VNF(s) the VNF-BD is applicable, with references to any specific characteristics, i.e., the VNF type, model, version/release, author, vendor, architectural components, among any other particular features.

6.1.3. Experiments

The specification of the number of executions for Trials, Tests and Method. The execution of a VNF-BD corresponds to the execution of the specified Method.

6.1.4. Environment

The details referring to the name, description, and information associated with the interfaces needed for the management and orchestration (MANO), if necessary, of the specified VNF-BD scenario. I.e., it refers to a specific interface that receives the VNF-BD scenario information and converts it to the template needed for an orchestration platform. In this case, the means to the manager component interface such orchestration platform must be provided, as well as its outcome orchestration status information (e.g., management interfaces of deployed components).

Further details about the concrete execution environment is, however, provided by the VNF-PP to keep the VNF-BD technology agnostic (see Section 6.2.2.1).

6.1.5. Scenario

This section contains all information needed to describe the deployment of all involved functional components mandatory for the execution of the benchmarking Tests addressed by the VNF-BD.

6.1.5.1. Nodes

Information about each component in a benchmarking setup (see Section 5). It contains the identification, name, image, role (i.e., agent, monitor, SUT), connection-points and resource requirements (i.e., allocation of CPU, memory, disk).

A SUT is always considered to be the full set of VNFCs comprised by the VNF. Thus all VNFCs of a VNF are always benchmarked together because all of them are needed to implement the full set of the VNF's functionalities. This is aligned with the results presented in [Peu-b] that show that VNF compositions should be benchmarked as a whole. Benchmarking single VNFCs in isolation could be considered as a particular case, i.e., whitebox scenario which is out of scope of this draft.

The lifecycle specification of a node lists all the workflows that must be realized on it during a Test. For instance, main workflows include: create, start, stop, delete. Particular workflows can be specified containing the required parameters and implementation.

Those details must reflect the actions taken on or by a node that might affect the VNF performance profile.

6.1.5.2. Links

Links contain information about the data plane links interconnecting the components of a benchmarking setup. Links refer to two or more connection-points of a node. A link might refer to be part of a network. Depending on the link type, the network might be implemented as a layer 2 mesh, or as directional-oriented traffic forwarding flow entries. Links also detain resource requirements, specifying the minimum bandwidth, latency, and frame loss rate for the execution of benchmarking Tests.

6.1.5.3. Policies

Involves the definition of execution environment policies to run the Tests. Policies might specify the (anti-)affinity placement rules for each component in the topology, min/max allocation of resources, and specific enabling technologies (e.g., DPDK, SR-IOV, PCIE) needed for each component.

6.1.6. Proceedings

This information is utilized by the Manager component to execute the benchmarking Tests. It consists of agent(s) and monitor(s) settings, detailing their prober(s)/listener(s) specification and running parameters.

Agents: Defines a list containing the Agent(s) needed for the VNF-BD tests. The information of each Agent contains its host environment, making reference to a node specified in the VNF-BD scenario (Section 6.1.5). In addition, each Agent also is defined with the the configured toolset of the Prober(s) and their running parameters fulfilled (e.g., stimulus workload, traffic format/trace, configurations to enable hardware capabilities, if existent). In each Prober, it is also detailed the output metrics to be extracted from it when running the benchmarking Tests.

Monitors: Defines a list containing the Monitor(s) needed for the VNF-BD tests. The information of each Monitor contains its host environment, making reference to a node specified in the VNF-BD scenario (Section 6.1.5) and detailing the placement settings of it (e.g., internal or external with the target VNF and/or execution environment). In addition, each Monitor also is defined with the the configured toolset of the Listener(s) and their running parameters fulfilled (e.g., tap interfaces, period of monitoring, interval among the measurements). In each Listener,

it is also detailed the output metrics to be extracted from it when running the benchmarking Tests.

6.2. VNF Performance Profile (VNF-PP)

VNF Performance Profile (VNF-PP) -- an output artifact of a VNF-BD execution performed by a Manager component. It contains all the metrics from monitor(s) and/or agent(s) components after realizing and executing the prober(s) and/or the listener(s) proceedings specified in its corresponding VNF-BD. Metrics are grouped according to the execution of the trial(s) and test(s) defined by a VNF-BD. A VNF-PP is specific to a unique VNF specification (e.g., image, version, format). It can be used to derive and extract statistics of particular VNF performance metrics recorded in the VNF-PP.

More specifically, each VNF-PP is defined by a structure that allows benchmarking results to be presented in a logical and unified format. A VNF-PP report is the result of an unique Test, while its content, the so called snapshot(s), correspond to a single Trial each. A snapshot is defined by a single Agent or Monitor. Each snapshot contains evaluation(s), each one being the output of a single prober/listener. And each evaluation contains one or more metrics. In summary, a VNF-PP aggregates results from reports (i.e., Test(s)); a report aggregates Agent(s) and Monitor(s) results (i.e., Trial(s)); a snapshot aggregates prober(s) or listener(s) results; and an evaluation aggregates metrics. An initial YANG model for VNF-PPs is available at [yang-vnf-pp].

The following items define the VNF-PP contents.

6.2.1. Headers

The definition of header parameters concerning the VNF-PP, e.g., its version, identifier, name, author, description, and the timestamp of when it was created.

6.2.2. Reports

The list of reports stored in a single VNF-PP. Each report is the result of a Test execution, containing the snapshot(s) provided by all the Agent(s) and Monitor(s) specified to execute that Test. Besides this, each report contains the details of the execution environment in which a Test was conducted.

6.2.2.1. Execution Environment

Specific information is required to describe the environment on which a benchmarking Test was executed.

If not already defined by the VNF-BD scenario (Section 6.1.5), for each component in the deployment scenario of the VNF benchmarking setup, the following topics must be detailed:

Hardware: Contains any information associated with the underlying hardware capabilities offered and used by the component during the benchmarking Test. Examples of such specification include allocated CPU architecture, connected NIC specification, allocated memory DIMM, etc. In addition, any information concerning details of resource isolation must also be described in this part of the report.

Software: Contains any information associated with the software apparatus offered and used during the benchmarking Test. Examples include details of operating system, kernel, hypervisor, container technologies, orchestration system, network element managers (EM), etc. and their specific versions. This section should also provide information about the configurations used by the involved software components.

Optionally, each report might contain references to an orchestration description document (e.g., TOSCA, YANG, HEAT) to clarify the technological aspects of the execution environment and any specific parameters that it might define.

6.2.2.2. Snapshots

The list of snapshots extracted for a single report. Each snapshot is the result of a Trial execution, containing the evaluations provided by a single Agent or Monitor. It must contain the origin of the evaluations, i.e., the name of the source component, its role (i.e., Agent or Monitor), and the name of the host where it was placed.

6.2.2.2.1. Evaluations

The list of evaluations extracted for a single snapshot. Each evaluation must contain the source of the metrics in it. Such source must be specified by a unique identifier, name, type (i.e., prober or listener), version of the tool it interfaces, and the raw full command/call used by such tool to extract the evaluation metrics. In particular, an evaluation must contain the exact timestamp values of

the moments the prober/listener proceeded with the start and stop operations that incurred in the successful extraction of its metrics.

6.2.2.2.1.1. Metrics

The list of metrics extracted from a single evaluation. Metrics are output of a prober or listener execution. Each one of them must be specified with a clear definition of name, value, unit of value(s), and value(s). The value(s) of a metric can be specified as a single scalar, a vector of scalars, and a vector of tuples (e.g., timestamp and value).

Regarding the definition of the VNF-PP metrics, their specificity depends on the VNF benchmarking methodology. I.e., in general each methodology defines its own metrics, which might depend on the measurement interfaces available for Agents and Monitors (i.e., probers/listeners), and how they are implemented.

6.3. Procedures

The methodology for VNF Benchmarking Automation encompasses the process defined in Figure 2, i.e., the procedures that translate a VNF-BD into a VNF-PP composing a VNF-BR by the means of the components specified in Figure 1. This section details the sequence of events that realize such process.

6.3.1. Pre-Execution

Before the execution of benchmarking Tests, some procedures must be performed:

1. A VNF-BD must be defined to be later instantiated into a deployment scenario and have executed its Tests. Such a description must contain all the structural and functional settings defined in Section 6.1. At the end of this step, the complete Method of benchmarking the target VNF is defined.
2. The VNF target image must be prepared to be benchmarked, having all its capabilities fully described. In addition all the probers and listeners defined in the VNF-BD must be implemented to realize the benchmark Tests. At the end of this step, the complete set of components of the benchmarking VNF-BD deployment scenario is defined.
3. The environment needed for a VNF-BD must be defined to realize its deployment scenario, in an automated or manual method. This step might count on the instantiation of orchestration platforms and the composition of specific topology descriptors needed by

those platforms to realize the VNF-BD deployment scenario. At the end of this step, the whole environment needed to instantiate the components of a VNF-BD deployment scenario is defined.

6.3.2. Automated Execution

Satisfied all the pre-execution procedures, the automated execution of the Tests specified by the VNF-BD follow. The execution time of this automated execution can either be bound by a fixed time limit specified in the VNF-BD or coupled to an exit condition, e.g., convergence of measured values.

1. Upon the parsing of a VNF-BD, the Manager must detect the VNF-BD variable input field (e.g., list of resources values) and compose the all the permutations of parameters. For each permutation, the Manager must elaborate a VNF-BD instance. Each VNF-BD instance defines a Test, and it will have its deployment scenario instantiated accordingly. I.e., the Manager must interface an orchestration platform to realize the automated instantiation of each deployment scenario defined by a VNF-BD instance (i.e., a Test). The Manager must iterate through all the VNF-BD instances to finish the whole set of Tests defined by all the permutations of the VNF-BD input fields.
2. Given a VNF-BD instance, the Manager, using the VNF-BD environment settings, must interface an orchestrator platform requesting the deployment of a scenario to realize a Test. To perform such step, The Manager might interface a management function responsible to properly parse the deployment scenario specifications into the orchestration platform interface format.
3. An orchestration platform must deploy the scenario requested by the Manager, assuring the requirements and policies specified on it. In addition, the orchestration platform must acknowledge the deployed scenario to the Manager specifying the management interfaces of the VNF and the other components in the running instances for the benchmarking Test.
4. Agent(s) and Monitor(s) (if existing) and the target VNF must be configured by the Manager according to the components settings defined in the VNF-BD instance. After this step, the whole VNF-BD Test will be ready to be executed.
5. Manager must interface Agent(s) and Monitor(s) (if existing) via management interfaces to require the execution of the benchmarking probers (and listeners, if existing), and retrieve expected metrics captured during or at the end of each Trial. I.e., for a single Test, according to the VNF-BD execution settings, the

Manager must guarantee that one or more Trials realize the required measurements to characterize the performance behavior of a VNF.

6. Output measurements from each obtained benchmarking Test, and its possible Trials, must be collected by the Manager, until all the Tests are finished. In the execution settings of the parsed VNF-BD, the Manager must check the Method repetition, and perform the whole set of VNF-BD Tests (i.e., since step 1), until all methods specified are finished.
7. Collected all measurements from the VNF-BD (Trials, Tests and Methods) execution, the intended metrics, as described in the VNF-BD, must be parsed, extracted and combined to create the corresponding VNF-PP. The combination of used VNF-BD and generated VNF-PP compose the resulting VNF benchmark report (VNF-BR).

6.3.3. Post-Execution

After the process of a VNF-BD execution, some automated procedures, not necessarily mandatory, can be performed to improve the quality and utility of a VNF-BR:

1. Archive the raw output contained in the VNF-PP, perform statistical analysis on it, or train machine learning models with the collected data.
2. Evaluate the analysis output to the detection of any possible cause-effect factors and/or intrinsic correlations in the VNF-BR (e.g., outliers).
3. Review the input VNF-BD and modify it to realize the proper extraction of the target VNF metrics based on the performed research. Iterate in the previous steps until composing a stable and representative VNF-BR.

6.4. Particular Cases

As described in [RFC8172], VNF benchmarking might require to change and adapt existing benchmarking methodologies. More specifically, the following cases need to be considered.

6.4.1. Capacity

VNFs are usually deployed inside containers or VMs to build an abstraction layer between physical resources and the resources available to the VNF. According to [RFC8172], it may be more

representative to design experiments in a way that the VMs hosting the VNFs are operating at maximum of 50% utilization and split the workload among several VMs, to mitigate side effects of overloaded VMs. Those cases are supported by the presented automation methodologies through VNF-BDs that enable direct control over the resource assignments and topology layouts used for a benchmarking experiment.

6.4.2. Redundancy

As a VNF might be composed of multiple components (VNFCs), there exist different schemas of redundancy where particular VNFCs would be in active or standby mode. For such cases, particular monitoring endpoints should be specified in VNF-BD so listeners can capture the relevant aspects of benchmarking when VNFCs would be in active/standby modes. In this particular case, capturing the relevant aspects of internal functionalities of a VNF and its internal components provides important measurements to characterize the dynamics of a VNF, those must be reflected in its VNF-PP.

6.4.3. Isolation

One of the main challenges of NFV is to create isolation between VNFs. Benchmarking the quality of this isolation behavior can be achieved by Agents that take the role of a noisy neighbor, generating a particular workload in synchrony with a benchmarking procedure over a VNF. Adjustments of the Agent's noisy workload, frequency, virtualization level, among others, must be detailed in the VNF-BD.

6.4.4. Failure Handling

Hardware and software components will fail or have errors and thus trigger healing actions of the benchmarked VNFs (self-healing). Benchmarking procedures must also capture the dynamics of this VNF behavior, e.g., if a container or VM restarts because the VNF software crashed. This results in offline periods that must be captured in the benchmarking reports, introducing additional metrics, e.g., max. time-to-heal. The presented concept, with a flexible VNF-PP structure to record arbitrary metrics, enables automation of this case.

6.4.5. Elasticity and Flexibility

Having software based network functions and the possibility of a VNF to be composed by multiple components (VNFCs), internal events of the VNF might trigger changes in VNF behavior, e.g., activating functionalities associated with elasticity such as automated scaling. These state changes and triggers (e.g. the VNF's scaling state) must

be captured in the benchmarking results (VNF-PP) to provide a detailed characterization of the VNF's performance behavior in different states.

6.4.6. Handling Configurations

As described in [RFC8172], does the sheer number of test conditions and configuration combinations create a challenge for VNF benchmarking. As suggested, machine readable output formats, as they are presented in this document, will allow automated benchmarking procedures to optimize the tested configurations. Approaches for this are, e.g., machine learning-based configuration space sub-sampling methods, such as [Peu-c].

6.4.7. White Box VNF

A benchmarking setup must be able to define scenarios with and without monitoring components inside the VNFs and/or the hosting container or VM. If no monitoring solution is available from within the VNFs, the benchmark is following the black-box concept. If, in contrast, those additional sources of information from within the VNF are available, VNF-PPs must be able to handle these additional VNF performance metrics.

7. Open Source Reference Implementations

Currently, technical motivating factors in favor of the automation of VNF benchmarking methodologies comprise: (i) the facility to run high-fidelity and commodity traffic generators by software; (ii) the existent means to construct synthetic traffic workloads purely by software (e.g., handcrafted pcap files); (iii) the increasing availability of datasets containing actual sources of production traffic able to be reproduced in benchmarking tests; (iv) the existence of a myriad of automating tools and open interfaces to programmatically manage VNFs; (v) the varied set of orchestration platforms enabling the allocation of resources and instantiation of VNFs through automated machineries based on well-defined templates; (vi) the ability to utilize a large tool set of software components to compose pipelines that mathematically analyze benchmarking metrics in automated ways.

In simple terms, network softwarization enables automation. There are two open source reference implementations that are build to automate benchmarking of Virtualized Network Functions (VNFs).

7.1. Gym

The software, named Gym, is a framework for automated benchmarking of Virtualized Network Functions (VNFs). It was coded following the initial ideas presented in a 2015 scientific paper entitled "VBaaS: VNF Benchmark-as-a-Service" [Rosa-a]. Later, the evolved design and prototyping ideas were presented at IETF/IRTF meetings seeking impact into NFVRG and BMWG.

Gym was built to receive high-level test descriptors and execute them to extract VNFs profiles, containing measurements of performance metrics - especially to associate resources allocation (e.g., vCPU) with packet processing metrics (e.g., throughput) of VNFs. From the original research ideas [Rosa-a], such output profiles might be used by orchestrator functions to perform VNF lifecycle tasks (e.g., deployment, maintenance, tear-down).

In [Rosa-b] Gym was utilized to benchmark a decomposed IP Multimedia Subsystem VNF. And in [Rosa-c], a virtual switch (Open vSwitch - OVS) was the target VNF of Gym for the analysis of VNF benchmarking automation. Such articles validated Gym as a prominent open source reference implementation for VNF benchmarking tests. Such articles set important contributions as discussion of the lessons learned and the overall NFV performance testing landscape, included automation.

Gym stands as one open source reference implementation that realizes the VNF benchmarking methodologies presented in this document. Gym is released as open source tool under Apache 2.0 license [gym].

7.2. tng-bench

Another software that focuses on implementing a framework to benchmark VNFs is the "5GTANGO VNF/NS Benchmarking Framework" also called "tng-bench" (previously "son-profile") and was developed as part of the two European Union H2020 projects SONATA NFV and 5GTANGO [tango]. Its initial ideas were presented in [Peu-a] and the system design of the end-to-end prototype was presented in [Peu-b].

Tng-bench aims to be a framework for the end-to-end automation of VNF benchmarking processes. Its goal is to automate the benchmarking process in such a way that VNF-PPs can be generated without further human interaction. This enables the integration of VNF benchmarking into continuous integration and continuous delivery (CI/CD) pipelines so that new VNF-PPs are generated on-the-fly for every new software version of a VNF. Those automatically generated VNF-PPs can then be bundled with the VNFs and serve as inputs for orchestration systems, fitting to the original research ideas presented in [Rosa-a] and [Peu-a].

Following the same high-level VNF testing purposes as Gym, namely: Comparability, repeatability, configurability, and interoperability, tng-bench specifically aims to explore description approaches for VNF benchmarking experiments. In [Peu-b] a prototype specification for VNF-BDs is presented which not only allows to specify generic, abstract VNF benchmarking experiments, it also allows to describe sets of parameter configurations to be tested during the benchmarking process, allowing the system to automatically execute complex parameter studies on the SUT, e.g., testing a VNF's performance under different CPU, memory, or software configurations.

Tng-bench was used to perform a set of initial benchmarking experiments using different VNFs, like a Squid proxy, an Nginx load balancer, and a Socat TCP relay in [Peu-b]. Those VNFs have not only been benchmarked in isolation, but also in combined setups in which up to three VNFs were chained one after each other. These experiments were used to test tng-bench for scenarios in which composed VNFs, consisting of multiple VNF components (VNFCs), have to be benchmarked. The presented results highlight the need to benchmark composed VNFs in end-to-end scenarios rather than only benchmark each individual component in isolation, to produce meaningful VNF-PPs for the complete VNF.

Tng-bench is actively developed and released as open source tool under Apache 2.0 license [tng-bench]. A larger set of example benchmarking results of various VNFs is available in [Peu-d].

8. Security Considerations

Benchmarking tests described in this document are limited to the performance characterization of VNFs in a lab environment with isolated network.

The benchmarking network topology will be an independent test setup and MUST NOT be connected to devices that may forward the test traffic into a production network, or misroute traffic to the test management network.

Special capabilities SHOULD NOT exist in the VNF benchmarking deployment scenario specifically for benchmarking purposes. Any implications for network security arising from the VNF benchmarking deployment scenario SHOULD be identical in the lab and in production networks.

9. IANA Considerations

This document does not require any IANA actions.

10. Acknowledgement

The authors would like to thank the support of Ericsson Research, Brazil. Parts of this work have received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. H2020-ICT-2016-2 761493 (5GTANGO: <https://5gtango.eu>).

11. References

11.1. Normative References

- [ETSI14a] ETSI, "Architectural Framework - ETSI GS NFV 002 V1.2.1", Dec 2014, <http://www.etsi.org/deliver/etsi_gs/NFV/001/_099/002/01.02.01-_60/gs_NFV002v010201p.pdf>.
- [ETSI14b] ETSI, "Terminology for Main Concepts in NFV - ETSI GS NFV 003 V1.2.1", Dec 2014, <http://www.etsi.org/deliver/etsi_gs/NFV/001_099-/003/01.02.01_60/gs_NFV003v010201p.pdf>.
- [ETSI14c] ETSI, "NFV Pre-deployment Testing - ETSI GS NFV TST001 V1.1.1", April 2016, <http://docbox.etsi.org/ISG/NFV/Open/DRAFTS/TST001_-_Pre-deployment_Validation/NFV-TST001v0015.zip>.
- [ETSI14d] ETSI, "Network Functions Virtualisation (NFV); Virtual Network Functions Architecture - ETSI GS NFV SWA001 V1.1.1", December 2014, <https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-SWA%20001v1.1.1%20-%20GS%20-%20Virtual%20Network%20Function%20Architecture.pdf>.
- [ETSI14e] ETSI, "Report on CI/CD and Devops - ETSI GS NFV TST006 V0.0.9", April 2018, <https://docbox.etsi.org/isg/nfv/open/drafts/TST006_CICD_and_Devops_report>.
- [ETSI19f] ETSI, "Specification of Networking Benchmarks and Measurement Methods for NFVI - ETSI GS NFV-TST 009 V3.2.1", June 2019, <https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-TST%20009v3.2.1%20-%20GS%20-%20NFVI_Benchmarks.pdf>.

- [ETSI19g] ETSI, "NFVI Compute and Network Metrics Specification - ETSI GS NFV-TST 008 V3.2.1", March 2019, <https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-TST%20008v3.2.1%20-%20GS%20-%20NFVI%20Compute%20and%20Nwk%20Metrics%20-%20Spec.pdf>.
- [RFC1242] S. Bradner, "Benchmarking Terminology for Network Interconnection Devices", July 1991, <<https://www.rfc-editor.org/info/rfc1242>>.
- [RFC8172] A. Morton, "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure", July 2017, <<https://www.rfc-editor.org/info/rfc8172>>.
- [RFC8204] M. Tahhan, B. O'Mahony, A. Morton, "Benchmarking Virtual Switches in the Open Platform for NFV (OPNFV)", September 2017, <<https://www.rfc-editor.org/info/rfc8204>>.

11.2. Informative References

- [gym] "Gym Framework Source Code", <<https://github.com/intrig-unicamp/gym>>.
- [Peu-a] M. Peuster, H. Karl, "Understand Your Chains: Towards Performance Profile-based Network Service Management", Fifth European Workshop on Software Defined Networks (EWSN) , 2016, <<http://ieeexplore.ieee.org/document/7956044/>>.
- [Peu-b] M. Peuster, H. Karl, "Profile Your Chains, Not Functions: Automated Network Service Profiling in DevOps Environments", IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) , 2017, <<http://ieeexplore.ieee.org/document/8169826/>>.
- [Peu-c] M. Peuster, H. Karl, "Understand your chains and keep your deadlines: Introducing time-constrained profiling for NFV", IEEE/IFIP 14th International Conference on Network and Service Management (CNSM) , 2018, <<https://ris.uni-paderborn.de/record/6016>>.
- [Peu-d] M. Peuster and S. Schneider and H. Karl, "The Softwarised Network Data Zoo", IEEE/IFIP 15th International Conference on Network and Service Management (CNSM) , 2019, <<https://sndzoo.github.io/>>.

- [Rosa-a] R. V. Rosa, C. E. Rothenberg, R. Szabo, "VBaaS: VNF Benchmark-as-a-Service", Fourth European Workshop on Software Defined Networks , Sept 2015, <<http://ieeexplore.ieee.org/document/7313620>>.
- [Rosa-b] R. Rosa, C. Bertoldo, C. Rothenberg, "Take your VNF to the Gym: A Testing Framework for Automated NFV Performance Benchmarking", IEEE Communications Magazine Testing Series , Sept 2017, <<http://ieeexplore.ieee.org/document/8030496>>.
- [Rosa-c] R. V. Rosa, C. E. Rothenberg, "Taking Open vSwitch to the Gym: An Automated Benchmarking Approach", IV Workshop pre-IETF/IRTF, CSBC Brazil, July 2017, <<https://intrig.dca.fee.unicamp.br/wp-content/plugins/papercite/pdf/rosa2017taking.pdf>>.
- [tango] "5GTANGO: Development and validation platform for global industry-specific network services and apps", <<https://5gtango.eu>>.
- [tng-bench] "5GTANGO VNF/NS Benchmarking Framework", <<https://github.com/sonata-nfv/tng-sdk-benchmark>>.
- [yang-vnf-bd] "VNF-BD YANG model", <<https://github.com/raphaelvrosa/vnf-bench-model/tree/master/vnf-br/yang/vnfbd>>.
- [yang-vnf-pp] "VNF-PP YANG model", <<https://github.com/raphaelvrosa/vnf-bench-model/tree/master/vnf-br/yang/vnfpp>>.

Authors' Addresses

Raphael Vicente Rosa (editor)
University of Campinas
Av. Albert Einstein, 400
Campinas, Sao Paulo 13083-852
Brazil

Email: rvrosa@dca.fee.unicamp.br
URI: <https://intrig.dca.fee.unicamp.br/raphaelvrosa/>

Christian Esteve Rothenberg
University of Campinas
Av. Albert Einstein, 400
Campinas, Sao Paulo 13083-852
Brazil

Email: chesteve@dca.fee.unicamp.br
URI: <http://www.dca.fee.unicamp.br/~chesteve/>

Manuel Peuster
Paderborn University
Warburgerstr. 100
Paderborn 33098
Germany

Email: manuel.peuster@upb.de
URI: <https://peuster.de>

Holger Karl
Paderborn University
Warburgerstr. 100
Paderborn 33098
Germany

Email: holger.karl@upb.de
URI: <https://cs.uni-paderborn.de/cn/>

BMWG
Internet-Draft
Intended status: Informational
Expires: April 23, 2021

R. Rosa, Ed.
C. Rothenberg
UNICAMP
M. Peuster
H. Karl
UPB
October 20, 2020

Methodology for VNF Benchmarking Automation
draft-rosa-bmwg-vnfbench-06

Abstract

This document describes a common methodology for the automated benchmarking of Virtualized Network Functions (VNFs) executed on general-purpose hardware. Specific cases of automated benchmarking methodologies for particular VNFs can be derived from this document. An open source reference implementation is reported as running code embodiment of the proposed, automated benchmarking methodology.

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 April 23, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	4
3. Scope	6
4. Considerations	6
4.1. VNF Assessment Methods	7
4.2. Benchmarking Stages	7
4.3. Architectural Framework	8
4.4. Scenarios	10
4.5. Phases of a Benchmarking Test	11
4.5.1. Phase I: Deployment	11
4.5.2. Phase II: Configuration	11
4.5.3. Phase III: Execution	12
4.5.4. Phase IV: Result	12
5. Methodology	12
5.1. VNF Benchmarking Descriptor (VNF-BD)	13
5.2. VNF Performance Profile (VNF-PP)	13
5.3. VNF Benchmarking Report (VNF-BR)	14
5.4. Procedures	14
5.4.1. Plan	15
5.4.2. Realization	16
5.4.3. Summary	17
6. Particular Cases	18
6.1. Capacity	18
6.2. Redundancy	18
6.3. Isolation	18
6.4. Failure Handling	18
6.5. Elasticity and Flexibility	19
6.6. Handling Configurations	19
6.7. White Box VNF	19
7. Open Source Reference Implementation	19
7.1. Gym	20
7.2. Related work: tng-bench	20
8. Security Considerations	21
9. IANA Considerations	22
10. YANG Modules	23
10.1. VNF-Benchmarking Descriptor	23
10.2. VNF Performance Profile	34
10.3. VNF Benchmarking Report	41
11. Acknowledgement	46
12. References	46
12.1. Normative References	46

12.2. Informative References	47
Authors' Addresses	49

1. Introduction

In [RFC8172] the Benchmarking Methodology Working Group (BMWG) presented considerations for benchmarking of VNFs and their infrastructure, similar to the motivation given, the following aspects reinforce and justify the need for VNF benchmarking: (i) pre-deployment infrastructure dimensioning to realize associated VNF performance profiles; (ii) comparison factor with physical network functions; (iii) and output results for analytical VNF development.

Even if many methodologies the BMWG already describes, e.g., self-contained black-box benchmarking, can be applied to VNF benchmarking scenarios, further considerations have to be made. This is because VNFs, which are software components, might not have strict and clear execution boundaries and depend on underlying virtualization environment parameters as well as management and orchestration decisions [ETS14a].

Different enabling technologies advent of Software Defined Networking (SDN) and Network Functions Virtualization (NFV) have propitiated the disaggregation of VNFs and benchmarking tools, turning their Application Programming Interfaces (APIs) open and programmable. This process have occurred mostly by: (i) the decoupling of network function's control and data planes; (ii) the development of VNFs as multi-layer and distributed software components; (iii) and the existence of multiple underlying hardware abstractions to be utilized by VNFs.

Utilizing SDN and NFV enabling technologies, a diversity of benchmarking tools have been created to facilitate the active stimulus and the passive monitoring of a VNF via diverse software abstraction layers, propitiating a wide variety of abstractions for benchmarking mechanisms in the formulation of a VNF benchmarking methodology. In this manner of establishing the disaggregation of a VNF benchmarking setup, the abstracted VNF benchmarking mechanisms can be programmable, enabling the execution of their underlying technologies by the means of well defined parameters and producing a report with standardized metrics.

Turning programmable the execution of a VNF benchmarking methodology enables a richer apparatus for the benchmarking of a VNF and consequently facilitates the high-fidelity assessment of a VNF behaviour. Estimating the behaviour of a VNF depends on three correlated factors:

Internal configuration: Each use case of the VNF might define specific settings for it to work properly, and even each VNF might dispose of specific settings to be configured.

Hardware and software execution environment: A myriad of capabilities offered by execution environments might match in a large diversity of manners the possible internal software arrangements that each VNF might be programmable.

Network workload specificities: Depending on the use case, a VNF might be placed in different settings, operating under varied traffic profiles and in demand of a specific performance behavior.

The role of a VNF benchmarking methodology consists in defining how to tackle the diversity of settings imposed by the above enlisted factors in order to extract performance metrics associated with particular VNF packet processing behaviors. The sample space of testing such diversity of settings can be extensively large, turning manual benchmarking experiments prohibitively expensive. Indeed, portability as an intrinsic characteristic of VNFs allows them to be deployed in multiple execution environments, enabling benchmarking setups in a myriad of settings. Thus, the establishment of a methodology for VNF benchmarking automation detains utter importance.

Accordingly, can and should the flexible, software-based nature of VNFs be exploited to fully automate the entire benchmarking methodology end-to-end. This is an inherent need to align VNF benchmarking with the agile methods enabled by the concept of Network Functions Virtualization (NFV) [ETSI14e]. More specifically it allows: (i) the development of agile performance-focused DevOps methodologies for Continuous Integration and Delivery (CI/CD) of VNFs; (ii) the creation of on-demand VNF test descriptors for upcoming execution environments; (iii) the path for precise-analytics of automated catalogues of VNF performance profiles; (iv) and run-time mechanisms to assist VNF lifecycle orchestration/management workflows, e.g., automated resource dimensioning based on benchmarking insights.

2. Terminology

Common benchmarking terminology contained in this document is derived from [RFC1242]. The reader is assumed to be familiar with the terminology as defined in the European Telecommunications Standards Institute (ETSI) NFV document [ETSI14b]. Some of these terms, and others commonly used in this document, are defined below.

NFV: Network Function Virtualization - the principle of separating network functions from the hardware they run on by using virtual hardware abstraction.

VNF: Virtualized Network Function - a software-based network function. A VNF can be either represented by a single entity or be composed by a set of smaller, interconnected software components, called VNF components (VNFCs) [ETSI14d]. Those VNFs are also called composed VNFs.

VNFC: Virtualized Network Function Component - a software component that implements (parts of) the VNF functionality. A VNF can consist of a single VNFC or multiple, interconnected VNFCs [ETSI14d]

VNFD: Virtualised Network Function Descriptor - configuration template that describes a VNF in terms of its deployment and operational behaviour, and is used in the process of VNF onboarding and managing the life cycle of a VNF instance.

NS: Network Service - a collection of interconnected VNFs forming a end-to-end service. The interconnection is often done using chaining of functions.

VNF Benchmarking Descriptor (VNF-BD) -- contains all the definitions and requirements to deploy, configure, execute, and reproduce VNF benchmarking tests. A VNF-BD is defined by the developer of a VNF benchmarking methodology and serve as input to the execution of an automated benchmarking methodology.

VNF Performance Profile (VNF-PP) -- in a well defined structure contains all the measured metrics resulting from the execution of automated VNF benchmarking tests defined by a specific VNF-BD. Additionally, it might also contain additional recordings of configuration parameters used during the execution of the benchmarking setup.

VNF Benchmarking Report (VNF-BR) -- contains all the definition of the inputs and outputs of an automated VNF benchmarking methodology. The inputs define the necessary VNF-BD and a respective list of variables referencing the VNF-BD fields that must be utilized to define the sample space of the VNF benchmarking settings. The outputs consist of a list of entries, each one contains one of the combinations of the sampled variables from the inputs, the input VNF-BD parsed with such combination of variables, and the obtained VNF-PP resulting from the automated realization of the parsed VNF-BD. A VNF-BR might contain the settings definitions of the orchestrator platform that realizes

the instantiation of the benchmarking setup to enable the VNF-BD fullfilment.

3. Scope

This document assumes VNFs as black boxes when defining their benchmarking methodologies. White box approaches are assumed and analysed as a particular case under the proper considerations of internal VNF instrumentation, later discussed in this document.

This document outlines a methodology for VNF benchmarking, specifically addressing its automation, without limiting the automated process to a specific benchmarking case or infrastructure. The document addresses state-of-the-art work on VNF benchmarking from scientific publications and current developments in other standardization bodies (e.g., [ETS14c], [ETS19f] and [RFC8204]) wherever possible.

Whenever utilizing the specifications of this document, a particular automated VNF benchmarking methodology must be described in a clear and objective manner following four basic principles:

- o Comparability: The output of a benchmarking test shall be simple to understand and process, in a human-readable format, coherent, and easily reusable (e.g., inputs for analytic applications).
- o Repeatability: A benchmarking setup shall be comprehensively defined through a flexible design model that can be interpreted and executed by the testing platform repeatedly but supporting customization.
- o Configurability: Open interfaces and extensible messaging models shall be available between benchmarking components for flexible composition of a benchmarking test descriptor and environment configurations.
- o Interoperability: A benchmarking test shall be ported to different environments, using lightweight components whenever possible.

4. Considerations

VNF benchmarking considerations are defined in [RFC8172]. Additionally, VNF pre-deployment testing considerations are well explored in [ETS14c]. Further, ETSI provides test specifications for networking benchmarks and measurement methods for NFV infrastructure in [ETS19f], which complements the presented work on VNF benchmarking methodologies.

4.1. VNF Assessment Methods

Following ETSI's model in [ETSI14c], we distinguish three methods for a VNF evaluation:

Benchmarking: Where parameters (e.g., CPU, memory, storage) are provided and the corresponding performance metrics (e.g., latency, throughput) are obtained. Note, such evaluations might create multiple reports, for example, with minimal latency or maximum throughput results.

Verification: Both parameters and performance metrics are provided and a stimulus verifies if the given association is correct or not.

Dimensioning: Performance metrics are provided and the corresponding parameters obtained. Note, multiple deployments may be required, or if possible, underlying allocated resources need to be dynamically altered.

Note: Verification and Dimensioning can be reduced to Benchmarking.

4.2. Benchmarking Stages

The realization of an automated benchmarking methodology can be divided into three stages:

Trial: Is a single process or iteration to obtain VNF performance metrics from benchmarking measurements. A Test MUST always run multiple Trials to get statistical confidence about the obtained measurements.

Test: Defines unique structural and functional parameters (e.g., configurations, resource assignment) for benchmarked components to perform one or multiple Trials. Each Test must be executed following a particular benchmarking scenario composed by a Method. Proper measures must be taken to ensure statistical validity (e.g., independence across Trials of generated load patterns).

Method: Consists of one or more Tests to benchmark a VNF. A Method can explicitly list ranges of parameter values for the configuration of a benchmarking scenario and its components. Each value of such a range is to be realized in a Test. I.e., Methods can define parameter studies.

4.3. Architectural Framework

A VNF benchmarking architectural framework, shown in Figure 1, establishes the disposal of essential components and control interfaces, explained below, that realize the automation of a VNF benchmarking methodology.

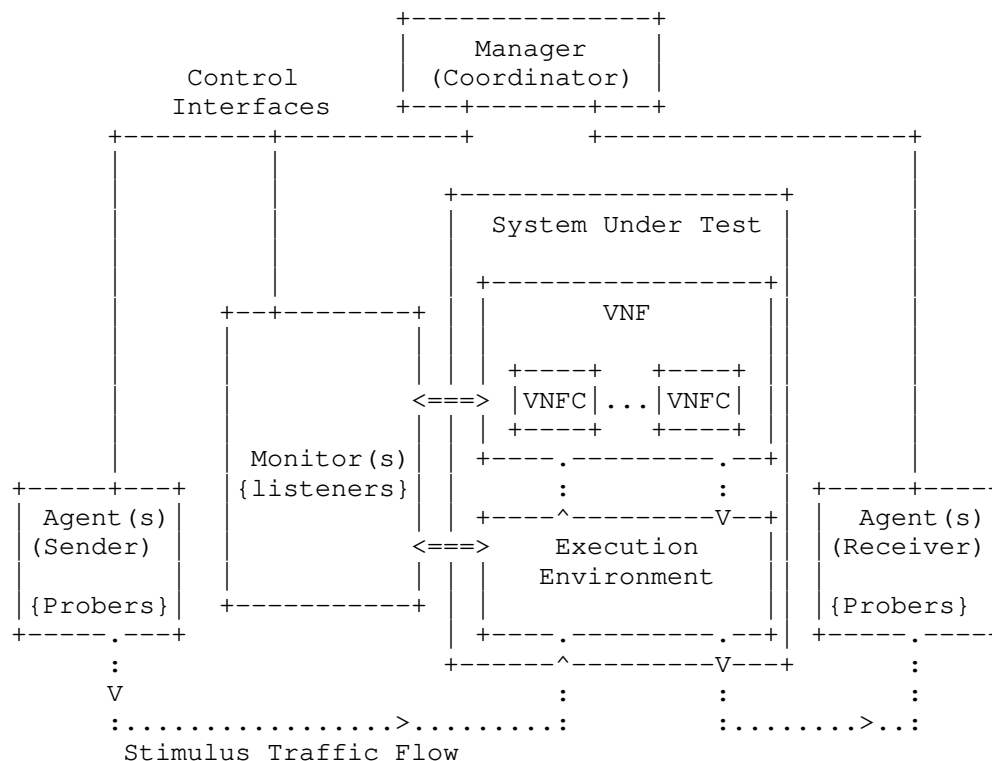


Figure 1: A VNF Benchmarking Architectural Framework

Virtualized Network Function (VNF) -- consists of one or more software components, so called VNF components (VNFC), adequate for performing a network function according to allocated virtual resources and satisfied requirements in an execution environment. A VNF can demand particular settings for benchmarking specifications, demonstrating variable performance based on available virtual resource parameters and configured enhancements targeting specific technologies (e.g., NUMA, SR-IOV, CPU-Pinning).

Execution Environment -- defines a virtualized and controlled composition of capabilities necessary for the execution of a VNF. An execution environment stands as a general purpose level of virtualization with abstracted resources available for one or more VNFs. It can also define specific technology qualifications, incurring in viable settings for enhancing the performance of VNFs, satisfying their particular enhancement requirements. An execution environment must be defined with the proper virtualization technologies feasible for the allocation of a VNF. The means to programmatically control the execution environment capabilities must be well defined for its life cycle management.

Agent (Active Prospection) -- executes active stimulus using probers, to benchmark and collect network and system performance metrics. A single Agent can perform localized benchmarks in execution environments (e.g., stress tests on CPU, memory, storage Input/Output) or can generate stimulus traffic and the other end be the VNF itself where, for example, one-way latency is evaluated. The interaction among two or more Agents enable the generation and collection of end-to-end metrics (e.g., frame loss rate, latency) measured from stimulus traffic flowing through a VNF. An Agent can be defined by a physical or virtual network function, and it must provide programmable interfaces for its life cycle management.

Prober -- defines an abstraction layer for a software or hardware tool able to generate stimulus traffic to a VNF or perform stress tests on execution environments. Probers might be specific or generic to an execution environment or a VNF. For an Agent, a Prober must provide programmable interfaces for its life cycle management, e.g., configuration of operational parameters, execution of stimulus, parsing of extracted metrics, and debugging options. Specific Probers might be developed to abstract and to realize the description of particular VNF benchmarking methodologies.

Monitor (Passive Prospection) -- when possible is instantiated inside the System Under Test, VNF and/or execution environment, to perform the passive monitoring, using Listeners, for the extraction of metrics while Agents' stimuli takes place. Monitors observe particular properties according to the execution environment and VNF capabilities, i.e., exposed passive monitoring interfaces. Multiple Listeners can be executed at once in synchrony with a Prober' stimulus on a SUT. A Monitor can be defined as a virtualized network function, and it must provide programmable interfaces for its life cycle management.

Listener -- defines one or more software interfaces for the extraction of metrics monitored in a target VNF and/or execution environment. A Listener must provide programmable interfaces for its life cycle management workflows, e.g., configuration of operational parameters, execution of passive monitoring captures, parsing of extracted metrics, and debugging options (also see [ETSI19g]). Varied methods of passive performance monitoring might be implemented as a Listener, depending on the interfaces exposed by the VNF and/or the execution environment.

Manager -- performs (i) the discovery of available Agents and Monitors and their respective features (i.e., available Probers/Listeners and their execution environment capabilities), (ii) the coordination and synchronization of activities of Agents and Monitors to perform a benchmarking Test, (iii) the collection, processing and aggregation of all VNF benchmarking (active and passive) metrics, which correlates the characteristics of the VNF traffic stimuli and the, possible, SUT monitoring. A Manager executes the main configuration, operation, and management actions to deliver the VNF benchmarking metrics. Hence, it detains interfaces open for users interact with the whole benchmarking framework, realizing, for instance, the retrieval of the framework characteristics (e.g., available benchmarking components and their probers/listeners), the coordination of benchmarking tests, the processing and the retrieval of benchmarking metrics, among other operational and management functionalities. A Manager can be defined as a physical or virtualized network function, and it must provide programmable interfaces for its life cycle management.

4.4. Scenarios

A scenario, as well referred as a benchmarking setup, consists of the actual instantiation of physical and/or virtual components of a "VNF Benchmarking Architectural Framework" needed to habilitate the execution of an automated VNF benchmarking methodology. The following considerations hold for a scenario:

- o Not all components are mandatory for a Test, possible to be disposed in varied setups.
- o Components can be aggregated in a single entity and be defined as black or white boxes. For instance, Manager and Agents could jointly define one hardware or software entity to perform a VNF benchmarking Test.

- o Monitor can be defined by multiple instances of distributed software components, each one addressing one or more VNF or execution environment monitoring interfaces.
- o Agents can be disposed in varied topology setups, included the possibility of multiple input and output ports of a VNF being directly connected each in one Agent.
- o All benchmarking components defined in a scenario must perform the synchronization of clocks.

4.5. Phases of a Benchmarking Test

In general, an automated benchmarking methodology must execute Tests repeatedly so it must capture the relevant causes of the performance variability of a VNF. To dissect a VNF benchmarking Test, in the sections that follow a set of benchmarking phases are categorized defining generic operations that may be automated. When executing an automated VNF benchmarking methodology, all the influencing aspects on the performance of a VNF must be carefully analyzed and comprehensively reported in each automated phase of a benchmarking Test.

4.5.1. Phase I: Deployment

The placement (i.e., assignment and allocation of resources) and the interconnection, physical and/or virtual, of network function(s) and benchmarking components can be realized by orchestration platforms (e.g., OpenStack, Kubernetes, Open Source MANO). In automated manners, the realization of a benchmarking scenario through those means usually rely on network service templates (e.g., TOSCA, YANG, Heat, and Helm Charts). Such descriptors have to capture all relevant details of the execution environment to allow the benchmarking framework to correctly instantiate the SUT as well as helper functions required for a Test.

4.5.2. Phase II: Configuration

The configuration of benchmarking components and VNFs (e.g., populate routing table, load PCAP source files in source of traffic stimulus) to execute the Test settings can be realized by programming interfaces in an automated way. In the scope of NFV, there might exist management interfaces to control a VNF during a benchmarking Test. Likewise, infrastructure or orchestration components can establish the proper configuration of an execution environment to realize all the capabilities enabling the description of the benchmarking Test. Each configuration registry, its deployment

timestamp and target, must all be contained in the report of a VNF benchmarking Test.

4.5.3. Phase III: Execution

In the execution of a benchmarking Test, the VNF configuration can be programmed to be changed by itself or by a VNF management platform. It means that during a Trial execution, particular behaviors of a VNF can be automatically triggered, e.g., auto-scaling of its internal components. Those must be captured in the detailed procedures of the VNF execution and its performance report. I.e., the execution of a Trial can determine arrangements of internal states inside a VNF, which can interfere in observed benchmarking metrics. For instance, in a particular benchmarking case where the monitoring measurements of the VNF and/or execution environment are available for extraction, comparison Tests must be run to verify if the monitoring of the VNF and/or execution environment can impact the VNF performance metrics.

4.5.4. Phase IV: Result

The result of a VNF benchmarking Test might contain generic metrics (e.g., CPU and memory consumption) and VNF-specific traffic processing metrics (e.g., transactions or throughput), which can be stored and processed in generic or specific ways (e.g., by statistics or machine learning algorithms). More details about possible metrics and the corresponding capturing methods can be found in [ETS19g]. If automated procedures are applied over the generation of a benchmarking Test result, those must be explained in the result itself, jointly with their input raw measurements and output processed data. For instance, any algorithm used in the generation of processed metrics must be disclosed in the Test result.

5. Methodology

The execution of an automated benchmarking methodology consists in elaborating a VNF Benchmarking Report, its inputs and outputs. The inputs part of a VNF-BR must be written by a VNF benchmarking tester. When the VNF-BR, with its inputs fulfilled, is requested from the Manager component of a implementation of the "VNF Benchmarking Architectural Framework", the Manager must utilize the inputs part to obtain the outputs part of the VNF-BR, addressing the execution of the automated benchmarking methodology as defined in Section 5.4.

The flow of information in the execution of an automated benchmarking methodology can be represented by the YANG modules defined by this document. The sections that follow present an overview of such modules.

5.1. VNF Benchmarking Descriptor (VNF-BD)

VNF Benchmarking Descriptor (VNF-BD) -- an artifact that specifies how to realize the Test(s) and Trial(s) of an automated VNF benchmarking methodology in order to obtain a VNF Performance Profile. The specification includes structural and functional instructions and variable parameters at different abstraction levels, such as the topology of the benchmarking scenario, and the execution parameters of prober(s)/listener(s) in the required Agent(s)/Monitor(s). A VNF-BD may be specific to a VNF or applicable to several VNF types.

More specifically, a VNF-BD is defined by a scenario and its proceedings. The scenario defines nodes (i.e., benchmarking components) and links interconnecting them, a topology that must be instantiated in order to execute the VNF-BD proceedings. The proceedings contain the specification of the required Agent(s) and Monitor(s) needed in the scenario nodes. Detailed in each Agent/Monitor follows the specification of the Prober(s)/Listener(s) required for the execution of the Tests, and in the details of each Prober/Listener follows the specification of its execution parameters. In the header of a VNF-BD is specified the number of Tests and Trials that a Manager must run them. Each Test realizes a unique instantiation of the scenario, while each Trial realizes a unique execution of the proceedings in the instantiated scenario of a Test. The VNF-BD YANG module is presented in Section 10.1.

5.2. VNF Performance Profile (VNF-PP)

VNF Performance Profile (VNF-PP) -- an output artifact of a VNF-BD execution performed by a Manager component. It contains all the metrics from Monitor(s) and/or Agent(s) components after realizing the execution of the Prober(s) and/or the Listener(s) proceedings, specified in its corresponding VNF-BD. Metrics are logically grouped according to the execution of the Trial(s) and Test(s) defined by a VNF-BD. A VNF-PP is specifically associated with a unique VNF-BD.

More specifically, a VNF-PP is defined by a structure that allows benchmarking results to be presented in a logical and unified format. A VNF-PP report is the result of a unique Test, while its content, the so called snapshot(s), each containing the results of the execution of a single Trial. Each snapshot is built by a single Agent or Monitor. A snapshot contains evaluation(s), each one being the output of the execution of a single Prober or Listener. An evaluation contains one or more metrics. In summary, a VNF-PP aggregates the results from reports (i.e., the Test(s)); a report aggregates Agent(s) and Monitor(s) results (i.e., the Trial(s)); a snapshot aggregates Prober(s) or Listener(s) results; and an

evaluation aggregates metrics. The VNF-PP YANG module is presented in Section 10.2.

5.3. VNF Benchmarking Report (VNF-BR)

VNF Benchmarking Report (VNF-BR) -- the core artifact of an automated VNF benchmarking methodology consisted of three parts: a header, inputs and output. The header refers to the VNF-BR description items (e.g., author, version, name), the description of the target SUT (e.g., the VNF version, release, name), and the environment settings specifying the parameters needed to instantiate the benchmarking scenario via an orchestration platform. The inputs contain the definitions needed to execute the automated benchmarking methodology of the target SUT, a VNF-BD and its variables settings. The outputs contain the results of the execution of the inputs, a list of entries, each one containing a VNF-BD filled with one of the combinations of the input variables settings, and the obtained VNF-PP reported after the execution of the Test(s) and Trial(s) of the parsed VNF-BD. The process of utilizing the VNF-BR inputs to generate its outputs concerns the realization of an automated VNF benchmarking methodology, explained in details in Section 5.4.2. The VNF-BR YANG module is presented in Section 10.3.

In details, each one of the variables in the inputs part of a VNF-BR is defined by: a name (the actual name of the variable); a path (the YANG path of the variable in the input VNF-BD); a type (the type of the values, such as string, int, float, etc); class (one of: stimulus, resource, configuration); and values (a list of the variable actual values). The values of all the variables must be combined all-by-all, generating a list containing the whole sample space of variables settings that must be used to create the VNF-BD instances. A VNF-BD instance is defined as the result of the parsing of one of those combinations of input variables into the VNF-BD of the VNF-BR inputs. The parsing takes place when the variable path is utilized to set its value in the VNF-BD. Iteratively, all the VNF-BD instances must have its Test(s) and Trial(s) executed to generate its corresponding VNF-PP. After all the VNF-BD instances had their VNF-PP accomplished, the realization of the whole automated VNF benchmarking methodology is complete, fulfilling the outputs part of the VNF-BR as shown in Figure 2.

5.4. Procedures

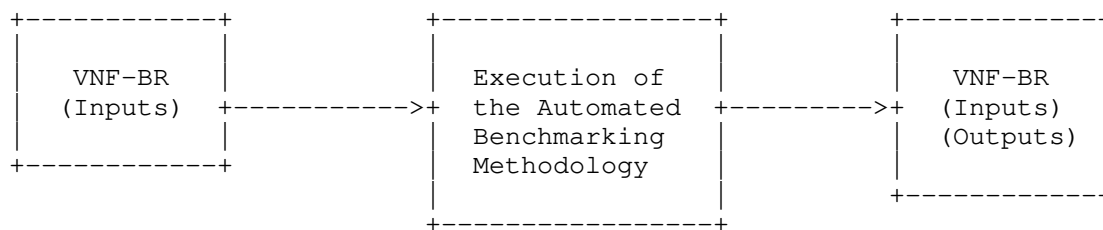


Figure 2: VNF benchmarking process inputs and outputs

The methodology for VNF benchmarking automation encompasses the process defined in Figure 2, i.e., the procedures that utilize the inputs part to obtain the outputs part of a VNF-BR. This section details the procedures that realize such process.

5.4.1. Plan

The plan of an automated VNF benchmarking methodology consists in the definition of all the header and the inputs part of a VNF-BR, the artifacts to be utilized by the realization of the methodology, and the establishment of the execution environment where the methodology takes place. The topics below contain the details of such planning.

1. The writing of a VNF-BD must be done utilizing the VNF-BD YANG module Section 10.1. A VNF-BD composition must determine the scenario and the proceedings. The VNF-BD must be added to the inputs part of an instance of the VNF-BR YANG model.
2. All the variables in the inputs part of a VNF-BR must be defined. Each variable must contain all its fields fullfiled according to the VNF-BR YANG module Section 10.3.
3. All the software artifacts needed for the instantiation of the VNF-BD scenario must be made and turn available for the execution of the Test(s) and Trial(s). The artifacts include the definition of software components that realize the role of the functional components of the Benchmarking Architectural Framework, i.e., the Manager, the Agent and the Monitor and their respective Probers and Listeners.
4. The header of the VNF-BR instance must be written, stating the VNF-BR description items, the specification of the SUT settings, and the definition of the environment parameters, feasible for the instantiation of the VNF-BD scenario when executing the automated VNF benchmarking methodology.

5. The execution environment needed for a VNF-BD scenario must be prepared to be utilized by an orchestration platform to automate instantiation of the scenario nodes and links needed for the execution of a Test. The orchestration platform interface parameters must be referenced in the VNF-BR header. The orchestration platform must have access to the software artifacts that are referenced in the VNF-BD scenario to be able to manage their life cycle.
6. The Manager component must be instantiated, the execution environment must be turned available, and the orchestration platform must have access to the execution environment and the software artifacts that are referenced in the scenario of the VNF-BD in the inputs part of the VNF-BR.

5.4.2. Realization

Accomplished all the planning procedures, the process of the realization of the automated benchmarking methodology must be realized as the following topics describe.

1. The realization of the benchmarking procedures starts when the VNF-BR composed in the planning procedures is submitted to the Manager component. It triggers the automated execution of the benchmarking methodology defined by the inputs part of the VNF-BR.
2. Manager computes all the combinations of values from the lists of inputs in the VNF-BD, part of the submitted VNF-BR. Each combination of variables are used to define a Test. The VNF-BD submitted serves as a template for each combination of variables. Each parsing of each combination of variables by the VNF-BD template creates a so called VNF-BD instance. The Manager must iterate through all the VNF-BD instances to finish the whole set of Tests defined by all the combinations of variables and their respective parsed VNF-BD. The Manager iterates through the following steps until all the Tests are accomplished.
3. The Manager must interface an orchestration platform to realize the automated instantiation of the deployment scenario defined by a VNF-BD instance (i.e., a Test). To perform such step, The Manager might interface a management function responsible to properly parse the deployment scenario specifications into the orchestration platform interface format. The environment specifications of the VNF-BR header provide the guidelines to interface the orchestration platform. The orchestration platform must deploy the scenario requested by the Manager, assuring the requirements and policies specified on it. In addition, the orchestration platform must acknowledge the deployed scenario to

the Manager specifying the management interfaces of the VNF SUT and the other components in the running instances for the benchmarking scenario. Only when the scenario is correctly deployed the execution of the VNF-BD instance Test(s) and Trial(s) must occur, otherwise the whole execution of the VNF-BR must be aborted and an error message must be added to the VNF-BR outputs describing the problems that occurred in the instantiation of the VNF-BD scenario. If the scenario is successfully deployed, the VNF-BD Test proceedings can be executed.

4. Manager must interface Agent(s) and Monitor(s) via their management interfaces to require the execution of the VNF-BD proceedings, which consist in running the specified Probers and Listeners using the defined parameters, and retrieve their output metrics captured at the end of each Trial. Thus, a Trial conceives the execution of the proceedings of the VNF-BD instance. The number of Trials is defined in each VNF-BD instance. After the execution of all defined Trials the execution of a Test ends.
5. Output measurements from each obtained benchmarking Trials that compose a Test result must be collected by the Manager, until all the Tests are finished. Each set of collected measurements from each VNF-BD instance Trials and Tests must be used to elaborate a VNF-PP by the Manager component. The respective VNF-PP, its associated VNF-BD instance and its input variables compose one of the entries of the list of outputs of the VNF-BR. After all the list of combinations of input variables is explored to obtain the whole list of instances of VNF-BDs and elaborated VNF-PPs, the Manager component returns the original VNF-BR submitted to it, including the outputs part properly filled.

5.4.3. Summary

After the realization of an automated benchmarking methodology, some automated procedures can be performed to improve the quality and the utility of the obtained VNF-BR, as described in the following topics.

1. Archive the raw outputs contained in the VNF-BR, perform statistical analysis on it, or train machine learning models with the collected data.
2. Evaluate the analysis output to the detection of any possible cause-effect factors and/or intrinsic correlations in the VNF-BR outputs (e.g., outliers).
3. Review the inputs of a VNF-BR, VNF-BD and variables, and modify them to realize the proper extraction of the target VNF metrics based on the intended goal of the VNF benchmarking methodology

(e.g., throughput). Iterate in the previous steps until composing a stable and representative VNF-BR.

6. Particular Cases

As described in [RFC8172], VNF benchmarking might require to change and adapt existing benchmarking methodologies. More specifically, the following cases need to be considered.

6.1. Capacity

VNFs are usually deployed inside containers or VMs to build an abstraction layer between physical resources and the resources available to the VNF. According to [RFC8172], it may be more representative to design experiments in a way that the VMs hosting the VNFs are operating at maximum of 50% utilization and split the workload among several VMs, to mitigate side effects of overloaded VMs. Those cases are supported by the presented automation methodologies through VNF-BDs that enable direct control over the resource assignments and topology layouts used for a benchmarking experiment.

6.2. Redundancy

As a VNF might be composed of multiple components (VNFCs), there exist different schemas of redundancy where particular VNFCs would be in active or standby mode. For such cases, particular monitoring endpoints should be specified in VNF-BD so listeners can capture the relevant aspects of benchmarking when VNFCs would be in active/standby modes. In this particular case, capturing the relevant aspects of internal functionalities of a VNF and its internal components provides important measurements to characterize the dynamics of a VNF, those must be reflected in its VNF-PP.

6.3. Isolation

One of the main challenges of NFV is to create isolation between VNFs. Benchmarking the quality of this isolation behavior can be achieved by Agents that take the role of a noisy neighbor, generating a particular workload in synchrony with a benchmarking procedure over a VNF. Adjustments of the Agent's noisy workload, frequency, virtualization level, among others, must be detailed in the VNF-BD.

6.4. Failure Handling

Hardware and software components will fail or have errors and thus trigger healing actions of the benchmarked VNFs (self-healing). Benchmarking procedures must also capture the dynamics of this VNF

behavior, e.g., if a container or VM restarts because the VNF software crashed. This results in offline periods that must be captured in the benchmarking reports, introducing additional metrics, e.g., max. time-to-heal. The presented concept, with a flexible VNF-PP structure to record arbitrary metrics, enables automation of this case.

6.5. Elasticity and Flexibility

Having software based network functions and the possibility of a VNF to be composed by multiple components (VNFCs), internal events of the VNF might trigger changes in VNF behavior, e.g., activating functionalities associated with elasticity such as automated scaling. These state changes and triggers (e.g. the VNF's scaling state) must be captured in the benchmarking results (VNF-PP) to provide a detailed characterization of the VNF's performance behavior in different states.

6.6. Handling Configurations

As described in [RFC8172], does the sheer number of test conditions and configuration combinations create a challenge for VNF benchmarking. As suggested, machine readable output formats, as they are presented in this document, will allow automated benchmarking procedures to optimize the tested configurations. Approaches for this are, e.g., machine learning-based configuration space sub-sampling methods, such as [Peu-c].

6.7. White Box VNF

A benchmarking setup must be able to define scenarios with and without monitoring components inside the VNFs and/or the hosting container or VM. If no monitoring solution is available from within the VNFs, the benchmark is following the black-box concept. If, in contrast, those additional sources of information from within the VNF are available, VNF-PPs must be able to handle these additional VNF performance metrics.

7. Open Source Reference Implementation

Currently, technical motivating factors in favor of the automation of VNF benchmarking methodologies comprise: (i) the facility to run high-fidelity and commodity traffic generators by software; (ii) the existent means to construct synthetic traffic workloads purely by software (e.g., handcrafted pcap files); (iii) the increasing availability of datasets containing actual sources of production traffic able to be reproduced in benchmarking tests; (iv) the existence of a myriad of automating tools and open interfaces to

programmatically manage VNFs; (v) the varied set of orchestration platforms enabling the allocation of resources and instantiation of VNFs through automated machineries based on well-defined templates; (vi) the ability to utilize a large tool set of software components to compose pipelines that mathematically analyze benchmarking metrics in automated ways.

In simple terms, the enlisted factors above justify that network softwarization enables the automation of VNF benchmarking methodologies. There exists an open source reference implementation that is built to demonstrate the concepts and methodology of this document in order to automate the benchmarking of Virtualized Network Functions.

7.1. Gym

The software, named Gym, is a framework for automated benchmarking of Virtualized Network Functions (VNFs). It was coded following the initial ideas presented in a 2015 scientific paper entitled "VBaaS: VNF Benchmark-as-a-Service" [Rosa-a]. Later, the evolved design and prototyping ideas were presented at IETF/IRTF meetings seeking impact into NFVRG and BMWG.

Gym was built to receive high-level test descriptors and execute them to extract VNFs profiles, containing measurements of performance metrics - especially to associate resources allocation (e.g., vCPU) with packet processing metrics (e.g., throughput) of VNFs. From the original research ideas [Rosa-a], such output profiles might be used by orchestrator functions to perform VNF lifecycle tasks (e.g., deployment, maintenance, tear-down).

In [Rosa-b] Gym was utilized to benchmark a decomposed IP Multimedia Subsystem VNF. And in [Rosa-c], a virtual switch (Open vSwitch - OVS) was the target VNF of Gym for the analysis of VNF benchmarking automation. Such articles validated Gym as a prominent open source reference implementation for VNF benchmarking tests. Such articles set important contributions as discussion of the lessons learned and the overall NFV performance testing landscape, included automation.

Gym stands as one open source reference implementation that realizes the VNF benchmarking methodologies presented in this document. Gym is released as open source tool under Apache 2.0 license [gym].

7.2. Related work: tng-bench

Another software that focuses on implementing a framework to benchmark VNFs is the "5GTANGO VNF/NS Benchmarking Framework" also called "tng-bench" (previously "son-profile") and was developed as

part of the two European Union H2020 projects SONATA NFV and 5GTANGO [tango]. Its initial ideas were presented in [Peu-a] and the system design of the end-to-end prototype was presented in [Peu-b].

Tng-bench aims to be a framework for the end-to-end automation of VNF benchmarking processes. Its goal is to automate the benchmarking process in such a way that VNF-PPs can be generated without further human interaction. This enables the integration of VNF benchmarking into continuous integration and continuous delivery (CI/CD) pipelines so that new VNF-PPs are generated on-the-fly for every new software version of a VNF. Those automatically generated VNF-PPs can then be bundled with the VNFs and serve as inputs for orchestration systems, fitting to the original research ideas presented in [Rosa-a] and [Peu-a].

Following the same high-level VNF testing purposes as Gym, namely: Comparability, repeatability, configurability, and interoperability, tng-bench specifically aims to explore description approaches for VNF benchmarking experiments. In [Peu-b] a prototype specification for VNF-BDs is presented which not only allows to specify generic, abstract VNF benchmarking experiments, it also allows to describe sets of parameter configurations to be tested during the benchmarking process, allowing the system to automatically execute complex parameter studies on the SUT, e.g., testing a VNF's performance under different CPU, memory, or software configurations.

Tng-bench was used to perform a set of initial benchmarking experiments using different VNFs, like a Squid proxy, an Nginx load balancer, and a Socat TCP relay in [Peu-b]. Those VNFs have not only been benchmarked in isolation, but also in combined setups in which up to three VNFs were chained one after each other. These experiments were used to test tng-bench for scenarios in which composed VNFs, consisting of multiple VNF components (VNFCs), have to be benchmarked. The presented results highlight the need to benchmark composed VNFs in end-to-end scenarios rather than only benchmark each individual component in isolation, to produce meaningful VNF-PPs for the complete VNF.

Tng-bench is actively developed and released as open source tool under Apache 2.0 license [tng-bench]. A larger set of example benchmarking results of various VNFs is available in [Peu-d].

8. Security Considerations

Benchmarking tests described in this document are limited to the performance characterization of VNFs in a lab environment with isolated network.

The benchmarking network topology will be an independent test setup and MUST NOT be connected to devices that may forward the test traffic into a production network, or misroute traffic to the test management network.

Special capabilities SHOULD NOT exist in the VNF benchmarking deployment scenario specifically for benchmarking purposes. Any implications for network security arising from the VNF benchmarking deployment scenario SHOULD be identical in the lab and in production networks.

9. IANA Considerations

This document registers one URI in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

URI: urn:ietf:params:xml:ns:yang:ietf-vnf-bd
Registrant Contact: The BMWG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-vnf-pp
Registrant Contact: The BMWG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-vnf-br
Registrant Contact: The BMWG of the IETF.
XML: N/A, the requested URI is an XML namespace.

Figure 3

This document registers three YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registration is requested:

```
name:      ietf-vnf-bd
namespace: urn:ietf:params:xml:ns:yang:ietf-vnf-bd
prefix:    vnf-bd
reference:  RFC CCCC

name:      ietf-vnf-pp
namespace: urn:ietf:params:xml:ns:yang:ietf-vnf-pp
prefix:    vnf-pp
reference:  RFC CCCC

name:      ietf-vnf-br
namespace: urn:ietf:params:xml:ns:yang:ietf-vnf-br
prefix:    vnf-br
reference:  RFC CCCC
```

Figure 4

10. YANG Modules

The following sections contain the YANG modules defined by this document.

10.1. VNF-Benchmarking Descriptor

```
module vnf-bd {
  namespace "urn:ietf:params:xml:ns:yang:vnf-bd";
  prefix "vnf-bd";

  organization "IETF/BMWG";
  contact "Raphael Vicente Rosa <raphaelvrosa@gmail.com>,
  Manuel Peuster <peuster@mail.uni-paderborn.de>";

  description "Yang module for a VNF Benchmarking
  Descriptor (VNF-BD).";

  revision "2019-08-13" {
    description "V0.3: Reviewed proceedings,
    tool - not VNF specific";
    reference "";
  }

  revision "2019-03-13" {
    description "V0.2: Reviewed role, policies, connection-points,
    lifecycle workflows, resources";
    reference "";
  }

  revision "2019-02-28" {
```

```
    description "V0.1: First release";
    reference "";
}

typedef workflows {
    type enumeration {
        enum create {
            description "When calling the create workflow.";
        }
        enum configure {
            description "When calling the configure workflow.";
        }
        enum start {
            description "When calling the start workflow.";
        }
        enum stop {
            description "When calling the stop workflow.";
        }
        enum delete {
            description "When calling the delete workflow.";
        }
        enum custom {
            description "When calling a custom workflow.";
        }
    }
    description "Defines basic life cycle workflows for a
node in a scenario.";
}

grouping node_requirements {
    container resources {
        container cpu {
            leaf vcpus {
                type uint32;
                description "The number of cores to be allocated
for a node.";
            }
            leaf cpu_bw {
                type string;
                description "The CPU bandwidth (CFS limit in 0.01-1.0)";
            }
            leaf pinning {
                type string;
                description "The list of CPU cores, separated by comma,
that a node must be pinned to.";
            }
        }
        description "The node CPU resources that must
be allocated for a benchmarking Test.";
    }
}
```



```
}
container memory {
  leaf size {
    type uint32;
    description "The memory allocation size.";
  }
  leaf unit {
    type string;
    description "The memory unit.";
  }
  description "The node memory resources
that must be allocated for a benchmarking
Test.";
}
container storage {
  leaf size {
    type uint32;
    description "The storage allocation size.";
  }
  leaf unit {
    type string;
    description "The storage unit.";
  }
  leaf volumes {
    type string;
    description "Volumes to be allocated by
a node storage.
A volume defines a mapping of an outside storage
partition inside the node storage system.
Volumes must be separated by comma and be defined
using a colon to separate the node internal and external
references of storage system paths.";
  }
  description "The node storage resources
that must be allocated for a benchmarking Test.";
}

description "The set of resources that must be allocated
for a node in a benchmarking Test.";
}

description "'The grouping determining the
resource requirements for a node in a scenario.'"
}

grouping connection_points {
  leaf id {
```

```
    type string;
    description "The connection-point
    unique identifier";
  }
  leaf interface {
    type string;
    description "The name of the node interface
    associated with the connection-point.";
  }
  leaf type {
    type string;
    description "The type of the network the
    connection-point interface is attached to.";
  }
  leaf address {
    type string;
    description "The Network address of the
    connection-point. It can be specified as a
    Ethernet MAC address, a IPv4 address or an IPv6 address.";
  }
  description "A connections-point of a node.";
}

grouping nodes {
  leaf id {
    type string;
    description "The unique identifier of a node
    in a scenario.";
  }
  leaf type {
    type string;
    description "The type of a node.";
  }
  leaf image {
    type string;
    description "The name of the image to be used to instantiate
    a node.";
  }
  leaf format {
    type string;
    description "The node format (e.g., container, process, VM).";
  }
  leaf role {
    type string;
    description "The role of the node in the Test scenario.
    The role must be one of: manager, agent, monitor, sut.";
  }
}
```

```
uses node_requirements;

list connection_points {
  key "id";
  uses connection_points;
  description "The list of connection points of a node.";
}

list relationships {
  key "name";
  leaf name {
    type string;
    description "Name of the relationship.";
  }
  leaf type {
    type string;
    description "Type of the relationship.";
  }
  leaf target {
    type string;
    description "Target of the relationship.";
  }

  description "Relationship of a node with the other
scenario components.";
}

list lifecycle {
  key "workflow";
  leaf workflow {
    type workflows;
    description "The type of the Workflow.";
  }
  leaf name {
    type string;
    description "The workflow name.";
  }
}

list parameters {
  key "input";
  leaf input {
    type string;
    description "The name of the parameter.";
  }
  leaf value {
    type string;
    description "The value of the parameter";
  }
}
```

```
    }

    description "The list of parameters to be
    applied to the node workflow.";
  }

  leaf-list implementation {
    type string;
    description "The workflow implementation.";
  }

  description "The life cycle workflows to be
  applied to this node.";
}

description "The specification of a node to be used
in a scenario for a benchmarking Test.";
}

grouping link {
  leaf id {
    type string;
    description "The link unique identifier.";
  }
  leaf name {
    type string;
    description "The name of the link.";
  }
  leaf type {
    type string;
    description "The type of the link.";
  }
  leaf network {
    type string;
    description "The network the link belongs to.";
  }
  leaf-list connection_points {
    type leafref {
      path "../../nodes/connection_points/id";
    }
    description "Reference to the connection points of nodes
    the link is adjacent.";
  }
  description "A link between nodes in a scenario.";
}

grouping scenario {
  list nodes {
```

```
    key "id";
    uses nodes;
    description "The list of nodes that must be
    instantiated in a scenario in order to enable
    a benchmarking Test.";
}

list links {
    key "id";
    uses link;
    description "The list of links among nodes that must be
    instantiated in a scenario in order to enable
    a benchmarking Test.";
}

list policies {
    key "name";
    leaf name {
        type string;
        description "The name of the policy.";
    }
    leaf type {
        type string;
        description "The type of the policy";
    }
    leaf targets {
        type string;
        description "The targets of the policy.
        Uuid of nodes and/or links separated by comma.";
    }
    leaf action {
        type string;
        description "The action of the policy";
    }
}

description "Definition of policies to be
utilized on the instantiation of the scenario.
A policy is defined by a name, it type,
the targets (nodes and/or links) to which it must
be applied to, and the proper action that
realizes the policy.";
}

description "Describes the deployment of all
involved functional components mandatory for
the execution of a benchmarking Test.";
}
```

```
grouping tool {
  leaf id {
    type uint32;
    description "The unique identifier of a tool.
    This information specifies how a tool can be
    identified in a list of probers/listeners of an
    Agent/Monitor.";
  }
  leaf instances {
    type uint32;
    description "The number of the tool instances that
    must be executed in parallel.";
  }
  leaf name {
    type string;
    description "The name of a tool.";
  }
  list parameters {
    key "input";
    leaf input {
      type string;
      description "The input key of a parameter";
    }
    leaf value {
      type string;
      description "The value of a parameter";
    }
    description "List of parameters for the execution
    of the tool. Each tool detains the proper set of running
    parameters that must be utilized to realize a benchmarking
    test.";
  }
}

container sched {
  leaf from {
    type uint32;
    default 0;
    description "The initial time (in seconds)
    of the execution of the tool.";
  }

  leaf until {
    type uint32;
    description "The final/maximum time (in seconds)
    of the execution of the tool summed all its instances
    repeat, duration and interval parameters.";
  }
}
```

```
    leaf duration {
      type uint32;
      description "The total duration (in seconds) of the execution
        of each instance of the tool.";
    }

    leaf interval {
      type uint32;
      description "The interval (in seconds) to be awaited
        among each one of the instances of the
        execution of the tool.";
    }

    leaf repeat {
      type uint32;
      description "The number of times the tool must be executed.";
    }

    description "The scheduling parameters of a tool.
      Each Agent/Monitor must utilize the scheduling parameters
      to perform the execution of its tools (probers/listeners)
      accordingly.";
  }

  description "A tool to be used in a benchmarking test.
    A tool can be inferred as a prober or a listener.";
}

grouping component {
  leaf uuid {
    type string;
    description "A unique identifier";
  }
  leaf name {
    type string;
    description "The name of component";
  }
}

description "A generic component.";
}

grouping agent {
  uses component;

  list probers {
    key "id";
    uses tool;
    description "Defines a list of the Prober(s)";
  }
}
```

```
        that must be used in a benchmarking test.";
    }
    description "An Agent defined by its uuid,
name and the mandatory list of probers to be used
by a benchmarking test.";
}

grouping monitor {
    uses component;

    list listeners {
        key "id";
        uses tool;
        description "Defines a list of the Listeners(s)
that must used in a benchmarking test.";
    }
    description "A Monitor defined by its uuid,
name and the mandatory list of probers to be used
by a benchmarking test.";
}

grouping proceedings {
    list agents {
        key "uuid";
        uses agent;
        description "Defines a list containing the
Agent(s) needed for a VNF-BD test.";
    }

    list monitors {
        key "uuid";
        uses monitor;
        description "Defines a list containing the
Monitor(s) needed for a VNF-BD test.";
    }
    description "Information utilized by a Manager
component to execute a benchmarking test.";
}

grouping vnf-bd {

    container experiments {
        leaf trials {
            type uint32;
            default 1;
            description "Number of trials.
A trial is a single process or iteration
to obtain VNF performance metrics from
```



```
        benchmarking the VNF-BD proceedings.";
    }
    leaf tests {
        type uint32;
        default 1;
        description "Number of tests.
        Each test defines unique structural
        and functional parameters (e.g., configurations,
        resource assignment) for benchmarked components
        to perform one or multiple Trials.
        Each Test must be executed following a
        particular scenario.";
    }
    description "Defines the number of trials and tests
    the VNF-BD must execute.";
}

container scenario {
    uses scenario;
    description "Scenarios defined by this VNF-BD.
    A scenario contains all information needed to describe
    the deployment of all involved functional components
    mandatory for the execution of a benchmarking Test.";
}

container proceedings {
    uses proceedings;
    description "Proceedings of VNF-BD.
    The proceedings are utilized by the Manager component
    to execute a benchmarking Test. It consists of
    agent(s)/monitor(s) settings, detailing their
    prober(s)/listener(s) specification and
    running parameters.";
}

description "A single VNF-BD.
A VNF-BD contains all required definitions and
requirements to deploy, configure, execute, and
reproduce VNF benchmarking tests.";
}

uses vnf-bd;
}
```

Figure 5

10.2. VNF Performance Profile

```
module vnf-pp {
  namespace "urn:ietf:params:xml:ns:yang:vnf-pp";
  prefix "vnf-pp";

  organization "IETF/BMWG";
  contact "Raphael Vicente Rosa <raphaelvrosa@gmail.com>,
  Manuel Peuster <peuster@mail.uni-paderborn.de>";

  description "Yang module for a VNF Performance Profile (VNF-PP).";

  revision "2019-10-15" {
    description "Reviewed VNF-PP structure -
    defines reports, snapshots, evaluations";
    reference "";
  }

  revision "2019-08-13" {
    description "V0.1: First release";
    reference "";
  }

  grouping tuple {
    description "A tuple used as key-value.";
    leaf key {
      type string;
      description "Tuple key.";
    }

    leaf value {
      type string;
      description "Tuple value.";
    }
  }

  grouping metric {
    leaf name {
      type string;
      description "The metric name";
    }

    leaf unit {
      type string;
      description "The unit of the metric value(s).";
    }
  }
}
```

```
leaf type {
  type string;
  mandatory true;
  description "The data type encoded in the value.
  It must refer to a known variable type, i.e.,
  string, float, uint, etc.";
}

choice value {
  case scalar {
    leaf scalar {
      type string;
      mandatory true;
      description "A single scalar value.";
    }
  }
  case vector {
    leaf-list vector {
      type string;
      min-elements 1;
      description "A list of scalar values";
    }
  }
  case series {
    list series {
      key "key";
      uses tuple;
      description "A list of key/values,
      e.g., a timeseries.";
    }
  }
}

mandatory true;
description "Value choice: scalar, vector, series.
A metric can only contain a value with one of them.";

description "A metric that holds the recorded benchmarking
results, can be a single value (scalar), a list of values
(vector), or a list of key/value
data (series), e.g., for timeseries.";

}

grouping evaluation {
  leaf id {
    type string;
    description "The evaluation
```

```
    unique identifier.";
  }

  leaf instance {
    type uint32;
    description "The unique identifier of the
parallel instance of the prober/listener that
was executed and created the evaluation.";
  }

  leaf repeat {
    type uint32;
    description "The unique identifier of the
prober/listener repeatition instance
was executed and created the evaluation.";
  }

  container source {

    leaf id {
      type string;
      description "The unique identifier of the source
of the evaluation,
i.e., the prober/listener unique identifier.";
    }

    leaf name {
      type string;
      description "The name of the source of the evaluation,
i.e., the prober/listener name.";
    }

    leaf type {
      type string;
      description "The type of the source of the evaluation,
i.e., one of prober or listener, that was used to obtain
it.";
    }

    leaf version {
      type string;
      description "The version of the tool interfacing
the prober/listener that was used to obtain
the evaluation.";
    }

    leaf call {
      type string;
    }
  }
}
```

```
        description "The full call of the tool realized by
        the source of the evaluation that performed
        the acquisition of the metrics.";
    }

    description "The details regarding the
    source of the evaluation.";
}

container timestamp {

    leaf start {
        type string;
        description "Time (date, hour, minute, second)
        when the evaluation started";
    }

    leaf stop {
        type string;
        description "Time (date, hour, minute, second)
        when the evaluation stopped";
    }

    description "Timestamps of the procedures
    that realized the extraction of the evaluation.";
}

list metrics {
    key "name";
    uses metric;
    description "List of metrics obtained
    from a single evaluation.";
}

leaf error {
    type string;
    description "Error, if existent,
    when obtaining evaluation.";
}

description "The set of metrics and their source
associated with a single Trial.";
}

grouping snapshot {
    leaf id {
        type string;
```

```
    description "The snapshot
    unique identifier.";
}

leaf trial {
    type uint32;
    description "The identifier of the trial
    when the snapshot was obtained.";
}

container origin {

    leaf id {
        type string;
        description "The unique identifier of the
        component of the origin of the snapshot,
        i.e., the agent or monitor unique identifier.";
    }

    leaf role {
        type string;
        description "The role of the component,
        origin of the snapshot, i.e.,
        one of agent or monitor.";
    }

    leaf host {
        type string;
        description "The hostname where the
        source of the snapshot was placed.";
    }

    description "The detailed origin of
    the snapshot.";
}

list evaluations {
    key "id";
    uses evaluation;
    description "The list of evaluations
    contained in a single snapshot Test.";
}

leaf timestamp {
    type string;
    description "Time (date, hour, minute, second)
    when the snapshot was created.";
```

```
    }

    leaf error {
      type string;
      description "Error, if existent,
        when obtaining the snapshot.";
    }

    description "The set of evaluations and their origin
      output of the execution of a single trial.";
  }

  grouping report {
    leaf id {
      type string;
      description "The report unique identifier.";
    }

    leaf test {
      type uint32;
      description "The identifier of the Test
        when the snapshots were obtained.";
    }

    list snapshots {
      key "id";
      uses snapshot;
      description "List of snapshots contained
        in a single report.";
    }

    leaf timestamp {
      type string;
      description "Time (date, hour, minute, second)
        when the report was created.";
    }

    leaf error {
      type string;
      description "Error, if existent,
        when obtaining the report.";
    }

    description "The set of snapshots output
      of a single Test.";
  }
```

```
grouping header {
  leaf id {
    type string;
    description "Unique identifier of the VNF-PP.";
  }
  leaf name {
    type string;
    description "Name of the VNF-PP.";
  }
  leaf version {
    type string;
    description "Version of the VNF-PP.";
  }
  leaf description {
    type string;
    description "Description of the VNF-PP.";
  }
  leaf timestamp {
    type string;
    description "Time (date, hour, minute, second)
when the VNF-PP was created.";
  }

  description "The header content of a VNF-PP.";
}

grouping vnf-pp {

  uses header;

  list reports {
    key "id";
    uses report;
    description "List of the reports of a VNF-PP.";
  }

  description "A single VNF-PP.";
}

uses vnf-pp;
}
```

Figure 6

10.3. VNF Benchmarking Report

```
module vnf-br {
  namespace "urn:ietf:params:xml:ns:yang:vnf-br";
  prefix "vnf-br";

  import vnf-bd {
    prefix "vnfbd";
    revision-date 2020-10-08;
  }

  import vnf-pp {
    prefix "vnfpp";
    revision-date 2020-10-08;
  }

  organization "IETF/BMWG";
  contact "Raphael Vicente Rosa <raphaelvrosa@gmail.com>,"
    "Manuel Peuster <peuster@mail.uni-paderborn.de>";
  description "Yang model for a VNF Benchmark Report (VNF-BR).";

  revision "2020-09-09" {
    description "V0.2: Review the structure
      and the grouping/leaf descriptions.";
    reference "";
  }

  revision "2020-09-09" {
    description "V0.1: First release";
    reference "";
  }

  grouping variable {
    leaf name {
      type string;
      description "The name of the variable.";
    }
    leaf path {
      type string;
      description "The VNF-BD YANG path of the
        variable.";
    }
    leaf type {
      type string;
      description "The type of the
        variable values.";
    }
    leaf class {
```

```
        type string;
        description "The class of the
        variable (one of resource, stimulus,
        configuration).";
    }
    leaf-list values {
        type string;
        description "The list of values
        of the variable.";
    }
}

grouping output {
    leaf id {
        type string;
        description "The output unique identifier.";
    }
    list variables {
        key "name";
        leaf name { type string; }
        leaf value { type string; }
        description "The list of instance of variables
        from VNF-BR:inputs utilized by a VNF-BD to
        generate a VNF-PP.";
    }
}

container vnfbd {
    uses vnfbd:vnf-bd;
    description "The VNF-BD that was executed
    to generate a output.";
}

container vnfpp {
    uses vnfpp:vnf-pp;
    description "The output VNF-PP of the
    execution of a VNF-BD.";
}
}

grouping vnf {
    leaf id {
        type string;
        description "The VNF unique identifier.";
    }
    leaf name {
        type string;
        description "The VNF name.";
    }
}
```

```
    leaf version {
      type string;
      description "The VNF version.";
    }
    leaf author {
      type string;
      description "The author of the VNF.";
    }
    leaf description {
      type string;
      description "The description of the VNF.";
    }
    description "The details of the VNF SUT.";
  }

  grouping header {
    leaf id {
      type string;
      description "The unique identifier of the VNF-BR ";
    }
    leaf name {
      type string;
      description "The name of the VNF-BR.";
    }
    leaf version {
      type string;
      description "The VNF-BR version.";
    }
    leaf author {
      type string;
      description "The VNF-BR author.";
    }
    leaf description {
      type string;
      description "The description of the VNF-BR.";
    }
  }

  container vnf {
    uses vnf;
    description "The VNF-BR target SUT VNF.";
  }

  container environment {
    leaf name {
      type string;
      description "The environment name";
    }
    leaf description {
```

```
    type string;
    description "A description
of the environment";
}
leaf deploy {
    type boolean;
    description "Defines if (True) the environment enables
the automated deployment by an orchestrator platform.";
}
container orchestrator {
    leaf name {
        type string;
        description "Name of the orchestrator
platform.";
    }

    leaf type {
        type string;
        description "The type of the orchestrator
platform.";
    }

    leaf description {
        type string;
        description "The description of the
orchestrator platform.";
    }

    list parameters {
        key "input";
        leaf input {
            type string;
            description "The name of the parameter";
        }
        leaf value {
            type string;
            description "The value of the parameter";
        }
    }

    description "List of orchestrator
input parameters.";
}

description "The specification of the orchestration platform
settings of a VNF-BR.";
}

description "The environment settings of a VNF-BR.";
```

```
    }

    description "Defines the content of a VNF-BR header.";
}

grouping vnf-br {
    description "Grouping for a single vnf-br.";

    uses header;

    container inputs {
        list variables {
            key "name";
            uses variable;
            description "The list of
            input variables.";
        }

        container vnfbd {
            uses vnfbd:vnf-bd;
            description "The input VNF-BD.";
        }

        description "The inputs needed to
        realize a VNF-BR.";
    }

    list outputs {
        key "id";
        uses output;
        description "The list of outputs
        of a VNF-BR.";
    }

    container timestamp {
        leaf start {
            type string;
            description "Time (date, hour, minute, second)
            of when the VNF-BR realization started";
        }

        leaf stop {
            type string;
            description "Time (date, hour, minute, second)
            of when the VNF-BR realization stopped";
        }

        description "Timestamps of the procedures that
```

```
    realized the realization of a VNF-BR.";
  }

  leaf error {
    type string;
    description "The VNF-BR error,
if occurred during its realization.";
  }
}

uses vnf-br;
}
```

Figure 7

11. Acknowledgement

The authors would like to thank the support of Ericsson Research, Brazil. Parts of this work have received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. H2020-ICT-2016-2 761493 (5GTANGO: <https://5gtango.eu>).

12. References

12.1. Normative References

- [ETSI14a] ETSI, "Architectural Framework - ETSI GS NFV 002 V1.2.1", Dec 2014, <http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01-_60/gs_NFV002v010201p.pdf>.
- [ETSI14b] ETSI, "Terminology for Main Concepts in NFV - ETSI GS NFV 003 V1.2.1", Dec 2014, <http://www.etsi.org/deliver/etsi_gs/NFV/001_099-/003/01.02.01_60/gs_NFV003v010201p.pdf>.
- [ETSI14c] ETSI, "NFV Pre-deployment Testing - ETSI GS NFV TST001 V1.1.1", April 2016, <http://docbox.etsi.org/ISG/NFV/Open/DRAFTS/TST001_-_Pre-deployment_Validation/NFV-TST001v0015.zip>.
- [ETSI14d] ETSI, "Network Functions Virtualisation (NFV); Virtual Network Functions Architecture - ETSI GS NFV SWA001 V1.1.1", December 2014, <https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-SWA%20001v1.1.1%20-%20GS%20-%20Virtual%20Network%20Function%20Architecture.pdf>.

- [ETSI14e] ETSI, "Report on CI/CD and Devops - ETSI GS NFV TST006 V0.0.9", April 2018,
<https://docbox.etsi.org/isg/nfv/open/drafts/TST006_CICD_and_Devops_report>.
- [ETSI19f] ETSI, "Specification of Networking Benchmarks and Measurement Methods for NFVI - ETSI GS NFV-TST 009 V3.2.1", June 2019,
<https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-TST%20009v3.2.1%20-%20GS%20-%20NFVI_Benchmarks.pdf>.
- [ETSI19g] ETSI, "NFVI Compute and Network Metrics Specification - ETSI GS NFV-TST 008 V3.2.1", March 2019,
<https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-TST%20008v3.2.1%20-%20GS%20-%20NFVI%20Compute%20and%20Nwk%20Metrics%20-%20Spec.pdf>.
- [RFC1242] S. Bradner, "Benchmarking Terminology for Network Interconnection Devices", July 1991,
<<https://www.rfc-editor.org/info/rfc1242>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC8172] A. Morton, "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure", July 2017,
<<https://www.rfc-editor.org/info/rfc8172>>.
- [RFC8204] M. Tahhan, B. O'Mahony, A. Morton, "Benchmarking Virtual Switches in the Open Platform for NFV (OPNFV)", September 2017, <<https://www.rfc-editor.org/info/rfc8204>>.

12.2. Informative References

- [gym] "Gym Framework Source Code",
<<https://github.com/intrig-unicamp/gym>>.
- [Peu-a] M. Peuster, H. Karl, "Understand Your Chains: Towards Performance Profile-based Network Service Management", Fifth European Workshop on Software Defined Networks (EWSDN) , 2016,
<<http://ieeexplore.ieee.org/document/7956044>>.

- [Peu-b] M. Peuster, H. Karl, "Profile Your Chains, Not Functions: Automated Network Service Profiling in DevOps Environments", IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) , 2017, <<http://ieeexplore.ieee.org/document/8169826/>>.
- [Peu-c] M. Peuster, H. Karl, "Understand your chains and keep your deadlines: Introducing time-constrained profiling for NFV", IEEE/IFIP 14th International Conference on Network and Service Management (CNSM) , 2018, <<https://ris.uni-paderborn.de/record/6016>>.
- [Peu-d] M. Peuster and S. Schneider and H. Karl, "The Softwarised Network Data Zoo", IEEE/IFIP 15th International Conference on Network and Service Management (CNSM) , 2019, <<https://sndzoo.github.io/>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [Rosa-a] R. V. Rosa, C. E. Rothenberg, R. Szabo, "VBaaS: VNF Benchmark-as-a-Service", Fourth European Workshop on Software Defined Networks , Sept 2015, <<http://ieeexplore.ieee.org/document/7313620>>.
- [Rosa-b] R. Rosa, C. Bertoldo, C. Rothenberg, "Take your VNF to the Gym: A Testing Framework for Automated NFV Performance Benchmarking", IEEE Communications Magazine Testing Series , Sept 2017, <<http://ieeexplore.ieee.org/document/8030496>>.
- [Rosa-c] R. V. Rosa, C. E. Rothenberg, "Taking Open vSwitch to the Gym: An Automated Benchmarking Approach", IV Workshop pre-IETF/IRTF, CSBC Brazil, July 2017, <<https://intrig.dca.fee.unicamp.br/wp-content/plugins/papercite/pdf/rosa2017taking.pdf>>.
- [tango] "5GTANGO: Development and validation platform for global industry-specific network services and apps", <<https://5gtango.eu>>.
- [tng-bench] "5GTANGO VNF/NS Benchmarking Framework", <<https://github.com/sonata-nfv/tng-sdk-benchmark>>.

Authors' Addresses

Raphael Vicente Rosa (editor)
University of Campinas
Av. Albert Einstein, 400
Campinas, Sao Paulo 13083-852
Brazil

Email: rvrosa@dca.fee.unicamp.br
URI: <https://intrig.dca.fee.unicamp.br/raphaelvrosa/>

Christian Esteve Rothenberg
University of Campinas
Av. Albert Einstein, 400
Campinas, Sao Paulo 13083-852
Brazil

Email: chesteve@dca.fee.unicamp.br
URI: <http://www.dca.fee.unicamp.br/~chesteve/>

Manuel Peuster
Paderborn University
Warburgerstr. 100
Paderborn 33098
Germany

Email: manuel.peuster@upb.de
URI: <https://peuster.de>

Holger Karl
Paderborn University
Warburgerstr. 100
Paderborn 33098
Germany

Email: holger.karl@upb.de
URI: <https://cs.uni-paderborn.de/cn/>

i»¿INTERNET-DRAFT

Kommu	BMWG	S.
Mware	Internet-Draft	V
Rapp	Intended status: Informational	J.
Mware	Expires: Sep 2019	V
2019		Mar 11,

Considerations for Benchmarking Network Virtualization Platforms
draft-skommu-bmwg-nvp-03.txt

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

This Internet-Draft will expire on September 11, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

Current network benchmarking methodologies are focused on physical networking components and do not consider the actual application layer traffic patterns and hence do not reflect the traffic that virtual networking components work with when using network virtualization overlays (NVO3). The purpose of this document is to distinguish and highlight benchmarking considerations when testing and evaluating virtual networking components in the data center.

Table of Contents

3	1. Introduction	
4	2. Conventions used in this document	
4	3. Definitions	
4	3.1. System Under Test (SUT)	
5	3.2. Network Virtualization Platform	
6	3.3. Microservices	
7	4. Scope	
7	4.1.1. Scenario 1	
7	4.1.2. Scenario 2	
7	4.1.3. Learning	
7	4.1.4. Flow Optimization	
7	4.1.5. Out of scope	
8	4.2. Virtual Networking for Datacenter Applications	
8	4.3. Interaction with Physical Devices	
9	5. NVP Benchmarking Considerations	
2	5.1. Learning	1
2	5.2. Traffic Flow Optimizations	1
2	5.2.1. Fast Path	1
2	5.2.2. Dedicated cores / Co-processors	1
3	5.2.3. Prioritizing and de-prioritizing active flows	1
3	5.3. Server Architecture Considerations	1
3	5.3.1. NVE Component considerations	1
7	5.3.2. Frame format/sizes within the Hypervisor	1
7	5.3.3. Baseline testing with Logical Switch	1
7	5.3.4. Repeatability	1

7	5.3.5. Tunnel encap/decap outside the Hypervisor	1
8	5.3.6. SUT Hypervisor Profile	1
0	5.4. Benchmarking Tools Considerations	2
0	5.4.1. Considerations for NVE	2
0	5.4.2. Considerations for Split-NVE	2
0	6. Control Plane Scale Considerations	2
1	6.1.1. VM Events	2
2	6.1.2. Scale	2
2	6.1.3. Control Plane Performance at Scale	2

7. Security Considerations	2
8. IANA Considerations	2
9. Conclusions	2
10. References	2
10.1. Normative References	2
10.2. Informative References	2
11. Acknowledgments	2
Appendix A. Partial List of Parameters to Document	2
A.1. CPU	2
A.2. Memory	2
A.3. NIC	2
A.4. Hypervisor	2
A.5. Guest VM	2
A.6. Overlay Network Physical Fabric	2
A.7. Gateway Network Physical Fabric	2
A.8. Metrics	2

1. Introduction

Datacenter virtualization that includes both compute and network virtualization is growing rapidly as the industry continues to look for ways to improve productivity, flexibility and at the same time cut costs. Network virtualization is comparatively new and expected

to grow tremendously similar to compute virtualization. There are multiple vendors and solutions out in the market. Each vendor often

has their own recommendations on how to benchmark their solutions thus making it difficult to perform an apples-to-apples comparison between different solutions. Hence, the need for a vendor, product and cloud agnostic way to benchmark network virtualization solutions

to help with comparison and make informed decisions when it comes to selecting the right network virtualization solution.

Applications traditionally have been segmented using VLANs and ACLs between the VLANs. This model does not scale because of the 4K scalability

limitations of VLANs. Overlays such as VXLAN were designed to address the limitations of VLANs.

With VXLAN, applications are segmented based on VXLAN encapsulation (specifically the VNI field in the VXLAN header), which is similar

to VLAN ID in the 802.1Q VLAN tag, however without the 4K scale limitations of VLANs. For a more detailed discussion on this subject

ct

please refer RFC 7364 'Problem Statement: Overlays for Network Virtualization'.

VXLAN is just one of several Network Virtualization Overlays (NVO). Some of the others include STT, Geneve and NVGRE. STT and Geneve have expanded on the capabilities of VXLAN. Please refer IETF's nv

o3

Kommu & Rapp

Expires Sep 11, 2019

[Page 3]

r working group < <https://datatracker.ietf.org/wg/nvo3/documents/> > fo
d more information.

f Modern application architectures, such as Micro-services, because o
f IP based connectivity within the app, place high demands on the
networking and security when compared to the traditional three tier
r app models such as web, app and db. Benchmarks MUST consider whethe
the proposed solution is able to scale up to the demands of such
applications and not just a three-tier architecture.

The benchmarks will be utilizing the various terminology and
definitions from the NVO3 working group including RFC 8014 and RFC
8394.

2. Conventions used in this document

d The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
CP "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", an
"OPTIONAL" in this document are to be interpreted as described in B
14 [RFC2119] [RFC8174] when, and only when, they appear in all
capitals, as shown here.

3. Definitions

3.1. System Under Test (SUT)

m Traditional hardware based networking devices generally use the
device under test (DUT) model of testing. In this model, apart fro
any allowed configuration, the DUT is a black box from a testing
es perspective. This method works for hardware based networking devic
since the device itself is not influenced by any other components
outside the DUT.

as Virtual networking components cannot leverage DUT model of testing
the DUT is not just the virtual device but includes the hardware
components that were used to host the virtual device

Hence System Under Test (SUT) model MUST be used instead of the
traditional device under test

With SUT model, the virtual networking component along with all
software and hardware components that host the virtual networking
component MUST be considered as part of the SUT.

Virtual networking components, because of their dependency on the underlying hardware and other software components, may end up leveraging NIC offload benefits, such as TCP Segmentation Offload (TSO), Large Receive Offload (LRO) and Rx / Tx Filters. Such underlying hardware and software level features, even though they may

not be part of virtual networking stack itself, MUST be considered and documented. Note: Physical switches and routers, including those ones that act as initiators for NVOs, work with L2/L3 packets and may not be able to leverage TCP enhancements such as TSO.

Please refer to section 5 Figure 1 for a visual representation of System Under Test in the case of Intra-Host testing and section 5 Figure 2 for System Under Test in the case of Inter-Host testing.

3.2. Network Virtualization Platform

This document focuses on the Network Virtualization Overlay platform as outlined in RFC 8014 and use cases from RFC 8394.

Network Virtualization platforms, function closer to the application layer and are able to work with not only L2/L3 packets but also segments that leverage TCP optimizations such as Large Segment Offload (LSO).

NVPs leverage TCP stack optimizations such as TCP Segmentation Offload (TSO) and Large Receive Offload (LRO) that enables NVPs to work with much larger payloads of up to 64K unlike their counterparts such as NFVs.

Because of the difference in the payload, which translates into one operation per 64K of payload in NVP versus ~40 operations for the same amount of payload in NFV because of having to divide it to MTU sized packets, results in considerable difference in performance between NFV and NVP.

Please refer to figure 1 for a pictorial representation of this primary difference between NPV and NFV for a 64K payload segment/packet running on network set to 1500 bytes MTU.

Note: Payload sizes in figure 1 are approximates.

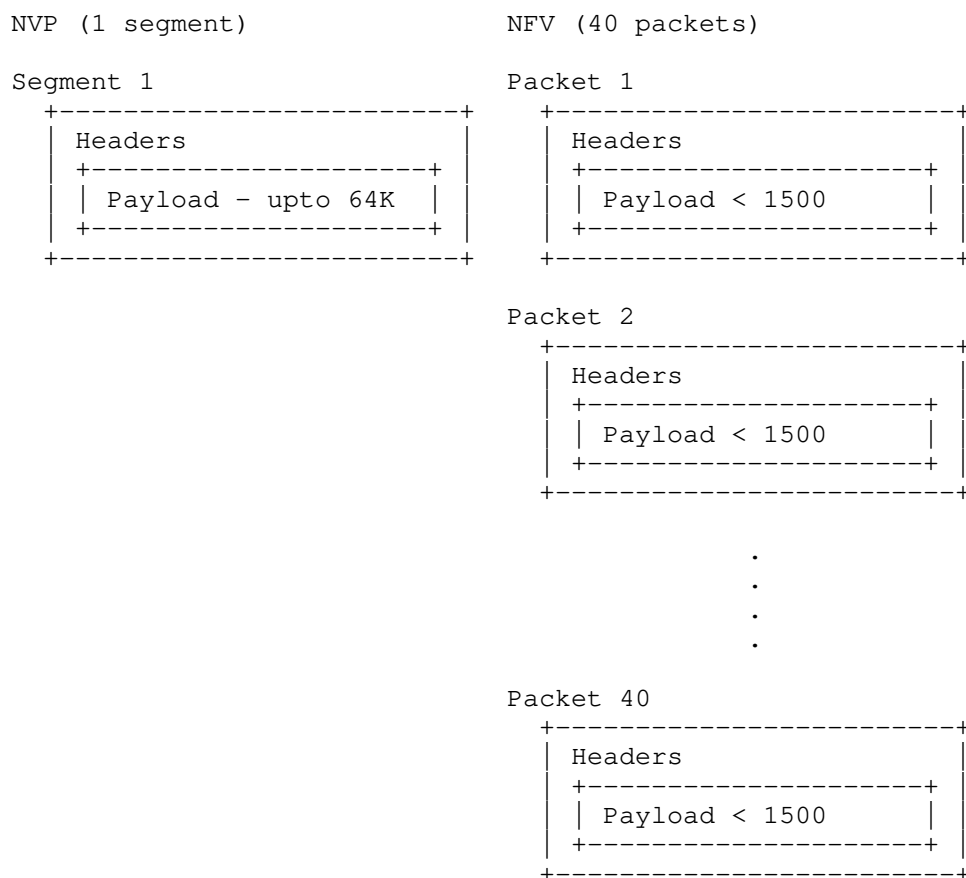


Figure 1! Payload NPV vs NFV

Hence, normal benchmarking methods are not relevant to the NVPs.

Instead, newer methods that leverage TCP optimizations MUST be used for testing Network Virtualization Platforms.

3.3. Microservices

Moving from traditional monolithic application architectures such as the three tier web, app and db architectures to microservices model open up networking and security stacks to new scale and performance

related challenges. At a high level, in a microservices model, a traditional monolithic app that may use few IPs is broken down into 100s of individual one-responsibility-only applications where each application has connectivity and security related requirements. These 100s of small one-responsibility-only micro-services need the

own IP and also secured into their own segments, hence pushing the scale boundaries of the overlay from both simple segmentation perspective and also from a security perspective.

For more details regarding microservices, please refer to wiki on microservices: <https://en.wikipedia.org/wiki/Microservices>

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.

4.1.5. Out of scope

This document does not address Network Function Virtualization which has been covered already by previous IETF documents

(https://datatracker.ietf.org/doc/draft-ietf-bmwg-virtual-net/?include_text=1).

of Network Function Virtualization (NFV) focuses on being independent networking hardware while providing the same functionality. In the case of NFV, traditional benchmarking methodologies recommended by IETF may be used. Considerations for Benchmarking Virtual Network Functions and Their Infrastructure IETF document addresses benchmarking NFVs.

s Typical NFV implementations emulate in software, the characteristic
l and features of physical switches. They are similar to any physical L2/L3 switch from the perspective of the packet size, which is typically enforced based on the maximum transmission unit used.

4.2. Virtual Networking for Datacenter Applications

ic This document focuses on the virtual networking for east-west traffic
er within on-prem datacenter and/or cloud. For example, in a three tier app such web, app and db, this document focuses on the east-west traffic between web and app.

ed This document addresses scale requirements for modern application architectures such as Micro-services to consider whether the proposed solution is able to scale up to the demands of micro-services application models that basically have 100s of small services communicating on some standard ports such as http/https using protocols such as REST.

4.3. Interaction with Physical Devices

a Virtual network components MUST NOT be tested independent of other components within the system. Example, unlike a physical router or firewall, where the tests can be focused solely on the device, when testing a virtual router or firewall, multiple other devices may become part of the SUT. Hence the characteristics of these other traditional networking switches and routers, LB, FW etc. MUST be considered.

- o Hashing method used
- o Over-subscription rate
- o Throughput available
- o Latency characteristics

5. NVP Benchmarking Considerations

In virtual environments, the SUT may often share resources and reside on the same physical hardware with other components involved in the tests. Hence SUT MUST be clearly documented. In these tests, a single hypervisor may host multiple servers, switches, routers, firewalls etc.

Intra host testing: Intra host testing helps in reducing the number of components involved in a test. For example, intra host testing would help focus on the System Under Test, logical switch and the hardware that is running the hypervisor that hosts the logical switch, and eliminate other components. Because of the nature of virtual infrastructures and multiple elements being hosted on the same physical infrastructure, influence from other components cannot be completely ruled out. For example, unlike in physical infrastructures, logical routing or distributed firewall MUST NOT be benchmarked independent of logical switching. System Under Test definition MUST include all components involved with that particular test.

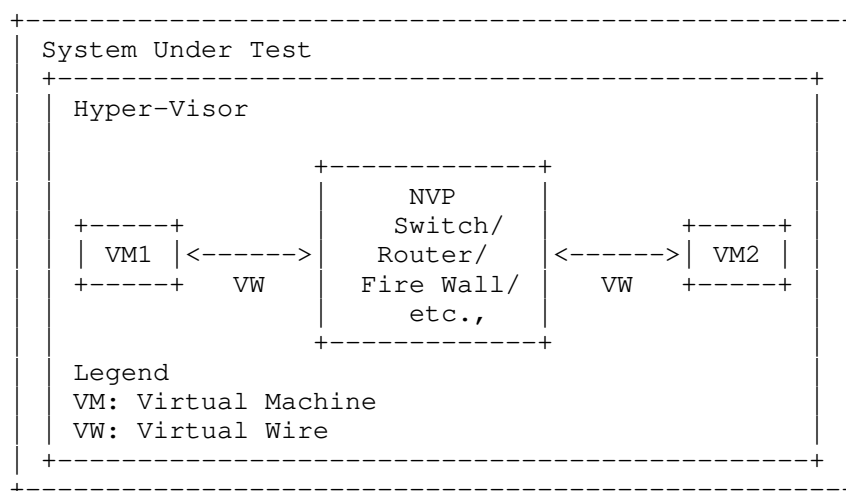


Figure 2! Intra-Host System Under Test

In the above figure, we only address the NVE co-located with the hypervisor.

Inter-host testing: Inter-host testing helps in profiling the underlying network interconnect performance. For example, when testing Logical Switching, inter host testing would not only test t

f logical switch component but also any other devices that are part o
f the physical data center fabric that connects the two hypervisors.
f System Under Test MUST be well defined to help with repeatability o
tests. System Under Test definition in the case of inter host
testing, MUST include all components, including the underlying
network fabric.

Figure 2 is a visual representation of system under test for inter-
host testing.

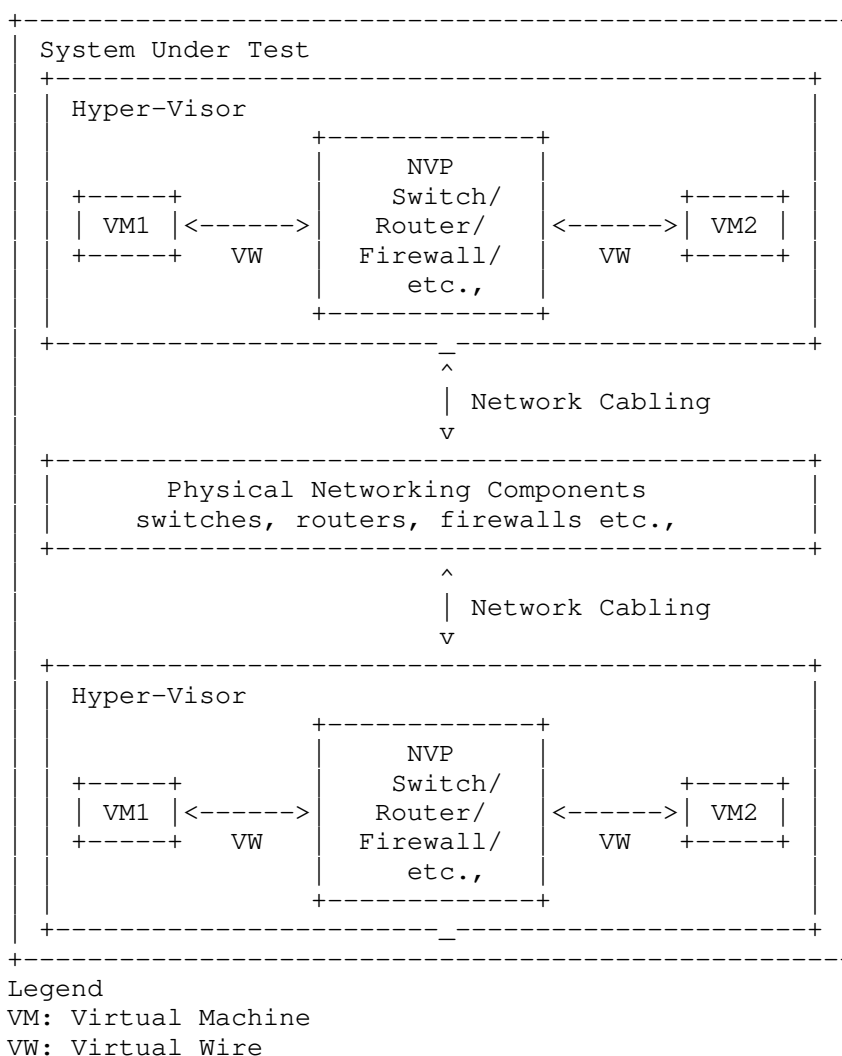


Figure 3! Inter-Host System Under Test

Virtual components have a direct dependency on the physical infrastructure that is hosting these resources. Hardware characteristics of the physical host impact the performance of the virtual components. The components that are being tested and the impact of the other hardware components within the hypervisor on th

performance of the SUT MUST be documented. Virtual component performance is influenced by the physical hardware components withi

n the hypervisor. Access to various offloads such as TCP segmentatio
offload, may have significant impact on performance. Firmware and
driver differences may also significantly impact results based on
whether the specific driver leverages any hardware level offloads
d offered. Packet processing could be executed on shared or dedicate
cores on the main processor or via a dedicated co-processor or
embedded processor on NIC.

ng Hence, all physical components of the physical server running the
hypervisor that hosts the virtual components MUST be documented alo
with the firmware and driver versions of all the components used to
help ensure repeatability of test results. For example, BIOS
configuration of the server MUST be documented as some of those
changes are designed to improve performance. Please refer to
Appendix A for a partial list of parameters to document.

5.1. Learning

ts SUT needs to learn all the addresses before running any tests.
Address learning rate MUST be considered in the overall performance
metrics because address learning rate has a high impact in
microservices based use cases where there is huge churn of end poin
as they are created and destroyed on demand. In these cases, both
the throughput at stable state, and the time taken to get to stable
state MUST be tested and documented.

5.2. Traffic Flow Optimizations

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

5.2.1. Fast Path

he 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 t
final state for that packet, and leverage it to skip the process fo
r 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.

5.3. Server Architecture Considerations

When testing physical networking components, the approach taken is

to consider the device as a black-box. With virtual infrastructure, this approach would no longer help as the virtual networking components are an intrinsic part of the hypervisor they are running on and are directly impacted by the server architecture used. Server

hardware components define the capabilities of the virtual networking components. Hence, server architecture MUST be documented in detail

to help with repeatability of tests. And the entire hardware and software components become the SUT.

5.3.1. NVE Component considerations

5.3.1.1. NVE co-located

Components of NVE co-located may be hypervisor based or offloaded entirely to the NIC card or a hybrid model. In the case of hypervisor-based model, they may be running in user space or kernel space. Further, they may use dedicated cores, shared cores or in some cases dedicated co-processors. All the components and the process used MUST be documented.

5.3.1.2. NVE split

NVE split scenario generally has three primary components as documented per RFC 8394.

"tNVE: Terminal-side NVE. The portion of Split-NVE functionalitie

s located on the end device supporting virtualization. The tNVE interacts with a Tenant System through an internal interface in the end device." tNVE may be made of either hypervisor controlled components such as hypervisor provided switches or NVE controlled

components where the network functionality is not provided by the hypervisor. In either case, the components used MUST be documented

"nNVE: Network-side NVE. The portion of Split-NVE functionalities located on the network device that is directly or indirectly connected to the end device that contains the corresponding NVE. Th

nNVE normally performs encapsulation to and decapsulation from the overlay network." All the functionality provided by the nNVE MUST documented.

"External NVE: The physical network device that contains the nNVE. Networking device hardware specs MUST be documented. Please use Appendix A for an example of the specs that MUST be documented.

In either case, NVE co-located or NVE split all the components MUST be documented. Where possible, individual components MUST be tested independent of the entire system. For example, where possible, hypervisor provided switching functionality MUST be tested independent of the NVE.

Per RFC 8014, "For the split-NVE case, protocols will be needed that allow the hypervisor and NVE to negotiate and set up the necessary state so that traffic sent across the access link between a server and the NVE can be associated with the correct virtual network instance." Supported VM lifecycle events, from RFC 8394 section 2, MUST be documented as part of the benchmark process. This process MUST also include how the hypervisor and the external NVE have signaled each other to reach an agreement. Example, see section 2.

of RFC 8394 "VM creation event". The process used to update agreement status MUST also be documented.

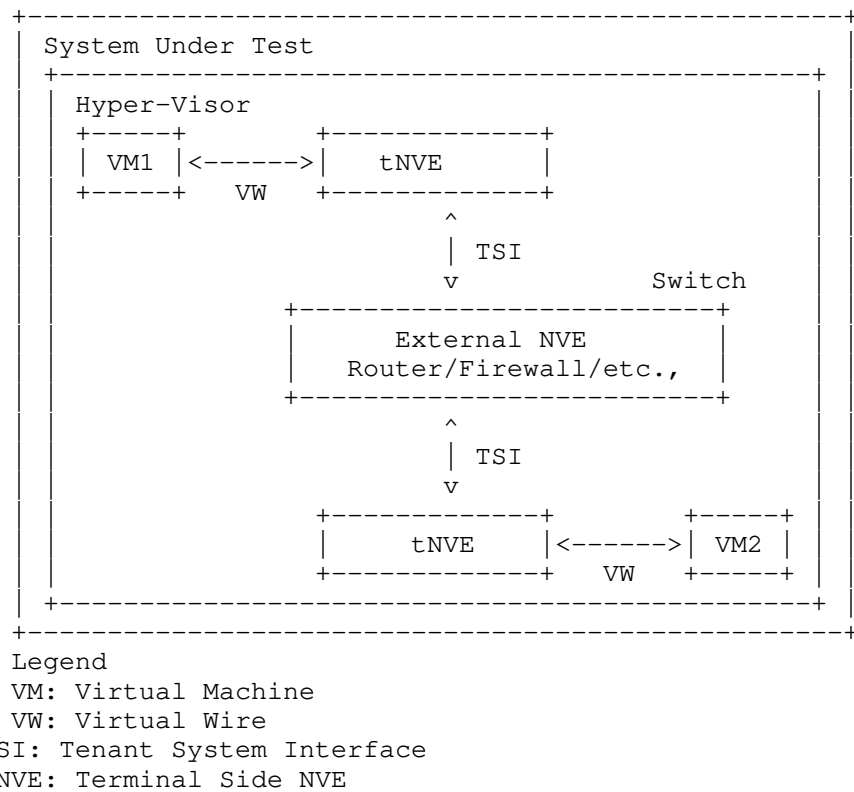


Figure 4 NVE Split collocated - System Under Test

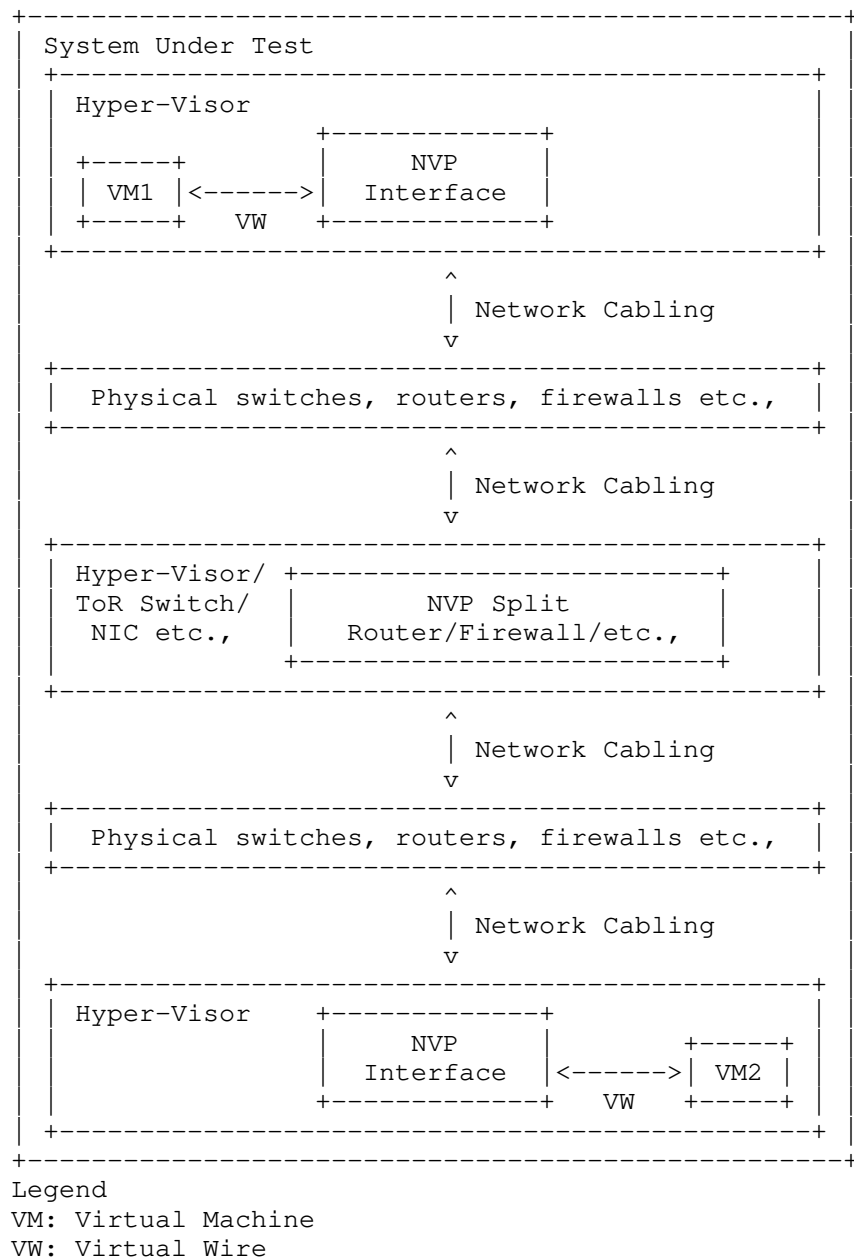


Figure 5 NVE Split not collocated - System Under Test

5.3.2. Frame format/sizes within the Hypervisor

Maximum Transmission Unit (MTU) limits physical network component's frame sizes. The most common max supported MTU for physical device is 9000. However, 1500 MTU is the standard. Physical network testing and NFV uses these MTU sizes for testing. However, the virtual networking components that live inside a hypervisor, may work with much larger segments because of the availability of hardware and software based offloads. Hence, the normal smaller packets based testing is not relevant for performance testing of virtual networking components. All the TCP related configuration such as TSO size, number of RSS queues MUST be documented along with any other physical NIC related configuration.

NVE co-located may have a different performance profile when compared with NVE split because, the NVE co-located may have access to offloads that may not be available when the packet has to traverse the physical link. Such differences MUST be documented.

5.3.3. Baseline testing with Logical Switch

Logical switch is often an intrinsic component of the test system along with any other hardware and software components used for testing. Also, other logical components cannot be tested independently of the Logical Switch.

5.3.4. Repeatability

To ensure repeatability of the results, in the physical network component testing, much care is taken to ensure the tests are conducted with exactly the same parameters. Example parameters such as MAC addresses used.

When testing NVP components with an application layer test tool, there may be a number of components within the system that may not be available to tune or to ensure they maintain a desired state. Example: housekeeping functions of the underlying Operating System.

Hence, tests MUST be repeated a number of times and each test case MUST be run for at least 2 minutes if test tool provides such an option. Results SHOULD be derived from multiple test runs. Variance between the tests SHOULD be documented.

5.3.5. Tunnel encap/decap outside the Hypervisor

Logical network components may also have performance impact based on the functionality available within the physical fabric. Physical

a fabric that supports NVO encap/decap is one such case that may have
on different performance profile. Any such functionality that exists
o the physical fabric MUST be part of the test result documentation t
ensure repeatability of tests. In this case SUT MUST include the
physical fabric if its being used for encap/decap operations.

5.3.6. SUT Hypervisor Profile

of Physical networking equipment has well defined physical resource
characteristics such as type and number of ASICs/SoCs used, amount
memory, type and number of processors etc., Virtual networking
components performance is dependent on the physical hardware that
hosts the hypervisor. Hence the physical hardware usage, which is
part of SUT, for a given test MUST be documented, for example, CPU
usage when running logical router.

he CPU usage, changes based on the type of hardware available within t
physical server. For example, TCP Segmentation Offload greatly
reduces CPU usage by offloading the segmentation process to the NIC
it card on the sender side. Receive side scaling offers similar benef
re on the receive side. Hence, availability and status of such hardwa
MUST be documented along with actual CPU/Memory usage when the
virtual networking components have access to such offload capable
hardware.

Following is a partial list of components that MUST be documented
both in terms of what is available and also what is used by the SUT

- o CPU - type, speed, available instruction sets (e.g. AES-NI)
- o Memory - type, amount
- o Storage - type, amount
- o NIC Cards -
 - * Type
 - * number of ports
 - * offloads available/used - following is a partial list o
f possible features
 - o TCP Segmentation Offload
 - o Large Receive Offload

- o Checksum Offloads
 - o Receive Side Scaling
 - o Other Queuing Mechanisms
 - * drivers, firmware (if applicable)
 - * HW revision
 - o Libraries such as DPDK if available and used
 - o Number and type of VMs used for testing and
 - * vCPUs
 - * RAM
 - * Storage
 - * Network Driver
 - * Any prioritization of VM resources
 - * Operating System type, version and kernel if applicable
 - * TCP Configuration Changes - if any
 - * MTU
 - o Test tool
 - * Workload type
 - * Protocol being tested
 - * Number of threads
 - * Version of tool
 - o For inter-hypervisor tests,
 - * Physical network devices that are part of the test
- o Note: For inter-hypervisor tests, system under test is no longer only the virtual component that is being

virtual components become part of the system under test.

5.4. Benchmarking Tools Considerations

5.4.1. Considerations for NVE

Virtual network components in NVE work closer to the application layer than the physical networking components, which enables the virtual network components to take advantage of TCP optimizations such as TCP Segmentation Offload (TSO) and Large Receive Offload (LRO). Because of this optimizations, virtual network components work with type and size of segments that are often not the same type and size that the physical network works with. Hence, testing virtual network components MUST be done with application layer segments instead of the physical network layer packets. Testing MUST be done with application layer testing tools such as iperf, netperf etc.,

5.4.2. Considerations for Split-NVE

In the case of Split-NVE, since they may not leverage any TCP related optimizations, typical network test tools focused on packet processing MUST be used. However, the tools used MUST be able to leverage Receive Side Scaling (RSS).

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 VM

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

VM Creation

For the purposes of NVP control plane testing, VM Creation event is when a VM starts participating for the first time on a NVP provided network. This involves various actions on the tNVE and NVP. Please refer to 2.1 "VM Creation Event" of RFC 8394 for more details.

In order to rule out any Hypervisor imposed limitations, System Under Test must first be profiled and baselined with-out the use of NVP components. For the purposes of baselining control plane, the VM used may have very small footprint such as DSL Linux which runs in 16MB RAM.

Once a baseline has been established for a single HV, a similar exercise MUST be done on multiple HVs to establish a baseline for the entire hypervisor domain. However, it may not be practical to have physical hosts and hence nested hosts may be used for this purpose

6.1.1. VM Events

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

- o VM Creation: Time taken to join the VMs to the SUT provided network
- o Policy Realization: Time taken for policy realization on the VM
- o 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 used:

- 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.

6.1.2. Scale

SUT must also be tested to determine the maximum scale supported. Scale can be multi-faceted such as the following:

- o Total # of VMs per Host
- o Total # of VMs per one SUT Domain
- o Total # of Hosts per one SUT Domain
- o Total # of Logical Switches per one SUT Domain
 - * Total # of VMs per one SUT provided Logical Switch
 - o Per Host
 - o Per SUT Domain
- o Total # of Logical Routers per one SUT Domain
 - * Total # of Logical Switches per one Logical Router
 - * Total # of VMs on a single Logical Router
- o Total # of Firewall Sections
- o Total # of Firewall Rules per Section
- o Total # of Firewall Rules applied per VM
- o Total # of Firewall Rules applied per Host
- o Total # of Firewall Rules per SUT

6.1.3. Control Plane Performance at Scale

Benchmarking MUST also test and document the control performance at scale. That is,

- o Total # VMs that can be created in parallel
 - * How long does the action take
- o Total # of VMs that can be migrated in parallel
 - * How long does the action take

- o Total amount of time taken to apply 1 firewall across the entire VMs under a SUT
- o Time taken to apply 1000s rules on a SUT

7. Security Considerations

Benchmarking activities as described in this memo are limited to technology characterization of a Device Under Test/System Under Test (DUT/SUT) using controlled stimuli in a laboratory environment, with dedicated address space and the constraints specified in the sections above.

The benchmarking network topology will be an independent test setup and MUST NOT be connected to devices that may forward the test traffic into a production network, or misroute traffic to the test management network.

Further, benchmarking is performed on a 'black-box' basis, relying solely on measurements observable external to the DUT/SUT.

Special capabilities SHOULD NOT exist in the DUT/SUT specifically for benchmarking purposes. Any implications for network security arising from the DUT/SUT SHOULD be identical in the lab and in production networks.

8. IANA Considerations

No IANA Action is requested at this time.

9. Conclusions

Network Virtualization Platforms, because of their proximity to the application layer and since they can take advantage of TCP stack optimizations, do not function on packets/sec basis. Hence, traditional benchmarking methods, while still relevant for Network Function Virtualization, are not designed to test Network Virtualization Platforms. Also, advances in application architectures such as micro-services, bring new challenges and need benchmarking not just around throughput and latency but also around scale. New benchmarking methods that are designed to take advantage

of the TCP optimizations or needed to accurately benchmark performance of the Network Virtualization Platforms

10. References

10.1. Normative References

[RFC7364] T. Narten, E. Gray, D. Black, L. Fang, L. Kreeger, M. Napierala, 'Problem Statement: Overlays for Network Virtualization', RFC 7364, October 2014, <https://datatracker.ietf.org/doc/rfc7364/>

An [RFC8014] D. Black, J. Hudson, L. Kreeger, M. Lasserre, T. Narten ' Architecture for Data-Center Network Virtualization o ver Layer 3 (NVO3)', RFC 8014, December 2016, <https://tools.ietf.org/html/rfc8014>

1- [RFC8394] Y. Li, D. Eastlake 3rd, L. Kreeger, T. Narten, D. Black 'Split Network Virtualization Edge (Split-NVE) Contro Plane Requirements', RFC 8394, May 2018, <https://tools.ietf.org/html/rfc8394>

[nv03] IETF, WG, Network Virtualization Overlays, <<https://datatracker.ietf.org/wg/nv03/documents/>>

10.2. Informative References

k [RFC8172] A. Morton 'Considerations for Benchmarking Virtual Networ Functions and Their Infrastructure', RFC 8172, July 2 017, <https://tools.ietf.org/html/rfc8172>

11. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

Appendix A. Partial List of Parameters to Document

A.1. CPU

- CPU Vendor
- CPU Number
- CPU Architecture
- # of Sockets (CPUs)
- # of Cores
- Clock Speed (GHz)
- Max Turbo Freq. (GHz)
- Cache per CPU (MB)
- # of Memory Channels
- Chipset
- Hyperthreading (BIOS Setting)
- Power Management (BIOS Setting)
- VT-d
- Shared vs Dedicated packet processing
- User space vs Kernel space packet processing

A.2. Memory

- Memory Speed (MHz)
- DIMM Capacity (GB)
- # of DIMMs
- DIMM configuration
- Total DRAM (GB)

A.3. NIC

Vendor
Model
Port Speed (Gbps)
Ports
PCIe Version
PCIe Lanes
Bonded
Bonding Driver
Kernel Module Name
Driver Version
VXLAN TSO Capable
VXLAN RSS Capable
Ring Buffer Size RX
Ring Buffer Size TX

A.4. Hypervisor

Hypervisor Name
Version/Build
Based on
Hotfixes/Patches
OVS Version/Build
IRQ balancing
vCPUs per VM
Modifications to HV

Modifications to HV TCP stack

Number of VMs

IP MTU

Flow control TX (send pause)

Flow control RX (honor pause)

Encapsulation Type

A.5. Guest VM

Guest OS & Version

Modifications to VM

IP MTU Guest VM (Bytes)

Test tool used

Number of NetPerf Instances

Total Number of Streams

Guest RAM (GB)

A.6. Overlay Network Physical Fabric

Vendor

Model

and Type of Ports

Software Release

Interface Configuration

Interface/Ethernet MTU (Bytes)

Flow control TX (send pause)

Flow control RX (honor pause)

A.7. Gateway Network Physical Fabric

Vendor

Model

and Type of Ports

Software Release

Interface Configuration

Interface/Ethernet MTU (Bytes)

Flow control TX (send pause)

Flow control RX (honor pause)

A.8. Metrics

Drops on the virtual infrastructure

Drops on the physical underlay infrastructure

Authors' Addresses

Samuel Kommu
VMware
3401 Hillview Ave
Palo Alto, CA, 94304

Email: skommu@vmware.com

Jacob Rapp
VMware
3401 Hillview Ave
Palo Alto, CA, 94304

Email: jrapp@vmware.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 10, 2020

V. Vassilev
Lightside Instruments
October 8, 2019

A YANG Data Model for Network Interconnect Tester Management
draft-vassilev-bmwg-network-interconnect-tester-02

Abstract

This document introduces new YANG model for use in network interconnect testing containing modules for traffic generator and traffic analyzer.

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 April 10, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	2
1.1.1. Definitions and Acronyms	2
1.1.2. Tree Diagram	3
1.2. Problem Statement	3
1.3. Solution	3
2. Using the network interconnect tester model	4
3. Traffic Generator Module Tree Diagram	4
4. Traffic Analyzer Module Tree Diagram	6
5. Traffic Generator Module YANG	7
6. Traffic Analyzer Module YANG	15
7. IANA Considerations	22
7.1. URI Registration	22
7.2. YANG Module Name Registration	22
8. Security Considerations	23
9. References	23
9.1. Normative References	23
9.2. Informative References	23
Appendix A. Examples	24
A.1. Basic Test Program	24
A.2. Generating RFC2544 Testframes	25
Author's Address	25

1. Introduction

There is a need for standard mechanism to allow the specification and implementation of the transactions part of network tests. The mechanism should allow the control and monitoring of the data plane traffic in a transactional way. This document defines YANG modules for test traffic generator, analyzer and internal interface loopback.

1.1. Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119].

1.1.1. Definitions and Acronyms

DUT: Device Under Test

TA: Traffic Analyzer

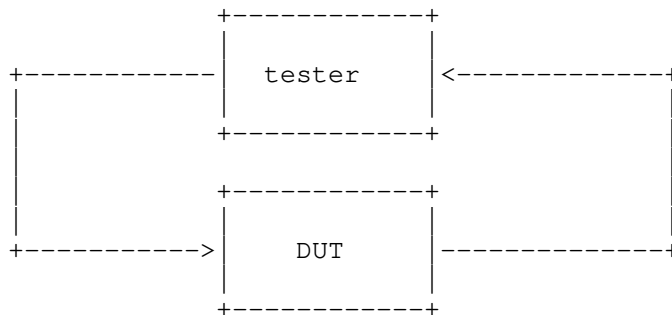
TG: Traffic Generator

1.1.2. Tree Diagram

For a reference to the annotations used in tree diagrams included in this draft, please see YANG Tree Diagrams [RFC8340].

1.2. Problem Statement

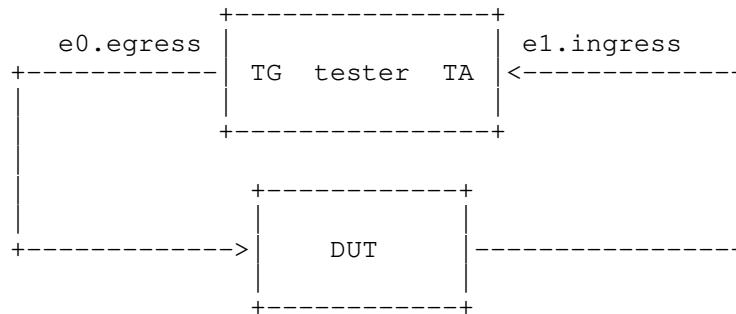
Network interconnect tests require active network elements part of the tested network that generate test traffic and network elements that analyze the test traffic at one or more points of its path. A network interconnect tester is a device that can either generate test traffic, analyze test traffic or both. Here is a figure borrowed from [RFC2544] representing the horseshoe test setup topology consisting of a single tester and a single DUT connected in a network interconnect loop.



This document attempts to address the problem of defining YANG model of a network interconnect tester that can be used for development of vendor independent network interconnect tests and utilize the advantages of transactional management using standard protocols like NETCONF.

1.3. Solution

The proposed model splits the design into 3 modules - 1) Traffic Generator module (TG), 2) Traffic Analyzer module (TA). The modules are implemented as augmentations of the ietf-interfaces module adding configuration and state data that models the functionality of a tester. The TA and TG modules concept is illustrated with the following diagram of a tester with two interfaces (named e0 and e1) connected in a loop with single DUT:



2. Using the network interconnect tester model

Basic example of how the model can be used in transactional network test API to control the testers part of a network and report counter statistics and timing measurement data is presented in Appendix A. One of the examples demonstrates the use of the [RFC2544] defined testframe packet.

3. Traffic Generator Module Tree Diagram

```

module: ietf-traffic-generator
augment /if:interfaces/if:interface:
  +--rw traffic-generator {egress-direction}?
    |
    |   +--rw (type)?
    |   |   +--:(single-stream)
    |   |   |   +--rw frame-size          uint32
    |   |   |   +--rw (frame-data-type)?
    |   |   |   |   +--:(raw-frame-data)
    |   |   |   |   |   +--rw frame-data?  string
    |   |   |   +--rw interframe-gap      uint32
    |   |   |   +--rw interburst-gap?     uint32
    |   |   |   +--rw frames-per-burst?   uint32
    |   |   |   +--rw src-mac-address?    yang:mac-address {ethernet}?
    |   |   |   +--rw dst-mac-address?    yang:mac-address {ethernet}?
    |   |   |   +--rw ether-type?         uint16 {ethernet}?
    |   |   |   +--rw vlan {ethernet-vlan,ethernet}?
    |   |   |   |   +--rw id              uint16
    |   |   |   |   +--rw tpid?           uint16
    |   |   |   |   +--rw pcp?            uint8
    |   |   |   |   +--rw cfi?            uint8
    |   |   +--:(multi-stream)
    |   |   |   +--rw streams
    |   |   |   |   +--rw stream* [id]
    |   |   |   |   |   +--rw id          uint32
    |   |   |   |   |   +--rw frame-size  uint32
  
```

```

    +--rw (frame-data-type)?
    |   +--:(raw-frame-data)
    |   |   +--rw frame-data?    string
    +--rw interframe-gap        uint32
    +--rw interburst-gap?       uint32
    +--rw frames-per-burst?     uint32
    +--rw frames-per-stream     uint32
    +--rw interstream-gap       uint32
    +--rw src-mac-address?      yang:mac-address {ethernet}?
    +--rw dst-mac-address?      yang:mac-address {ethernet}?
    +--rw ether-type?           uint16 {ethernet}?
    +--rw vlan {ethernet-vlan,ethernet}?
    |   +--rw id                uint16
    |   +--rw tpid?            uint16
    |   +--rw pcp?             uint8
    |   +--rw cfi?             uint8
    +--rw total-frames?         uint64
+--rw traffic-generator-ingress {ingress-direction}?
+--rw (type)?
|   +--:(single-stream)
|   |   +--rw frame-size        uint32
|   |   +--rw (frame-data-type)?
|   |   |   +--:(raw-frame-data)
|   |   |   |   +--rw frame-data?    string
|   |   +--rw interframe-gap    uint32
|   |   +--rw interburst-gap?   uint32
|   |   +--rw frames-per-burst? uint32
|   |   +--rw src-mac-address?   yang:mac-address {ethernet}?
|   |   +--rw dst-mac-address?   yang:mac-address {ethernet}?
|   |   +--rw ether-type?       uint16 {ethernet}?
|   |   +--rw vlan {ethernet-vlan,ethernet}?
|   |   |   +--rw id            uint16
|   |   |   +--rw tpid?        uint16
|   |   |   +--rw pcp?         uint8
|   |   |   +--rw cfi?         uint8
|   +--:(multi-stream)
|   |   +--rw streams
|   |   |   +--rw stream* [id]
|   |   |   |   +--rw id                uint32
|   |   |   |   +--rw frame-size        uint32
|   |   |   |   +--rw (frame-data-type)?
|   |   |   |   |   +--:(raw-frame-data)
|   |   |   |   |   |   +--rw frame-data?    string
|   |   |   +--rw interframe-gap    uint32
|   |   |   +--rw interburst-gap?   uint32
|   |   |   +--rw frames-per-burst? uint32
|   |   |   +--rw frames-per-stream uint32
|   |   |   +--rw interstream-gap   uint32

```



```

|         +---rw src-mac-address?      yang:mac-address {ethernet}?
|         +---rw dst-mac-address?      yang:mac-address {ethernet}?
|         +---rw ether-type?           uint16 {ethernet}?
|         +---rw vlan {ethernet-vlan, ethernet}?
|             +---rw id                uint16
|             +---rw tpid?             uint16
|             +---rw pcp?              uint8
|             +---rw cfi?              uint8
+---rw total-frames?                  uint64
augment /if:interfaces-state/if:interface/if:statistics:
+---ro generated-pkts?                yang:counter64
+---ro generated-octets?              yang:counter64
+---ro generated-ingress-pkts?        yang:counter64 {ingress-direction}?
+---ro generated-ingress-octets?      yang:counter64 {ingress-direction}?

```

4. Traffic Analyzer Module Tree Diagram

```

module: ietf-traffic-analyzer
augment /if:interfaces/if:interface:
+---rw traffic-analyzer! {ingress-direction}?
|   +---rw filter! {filter}?
|   |   +---rw type                identityref
|   |   +---rw ether-type?         uint16
|   +---ro state
|   |   +---ro pkts?                yang:counter64
|   |   +---ro errors?              yang:counter64
|   |   +---ro testframe-stats
|   |   |   +---ro testframe-pkts?   yang:counter64
|   |   |   +---ro sequence-errors?  yang:counter64
|   |   |   +---ro payload-errors?   yang:counter64
|   |   |   +---ro latency
|   |   |   |   +---ro samples?      uint64
|   |   |   |   +---ro min?          uint64
|   |   |   |   +---ro max?          uint64
|   |   |   |   +---ro average?      uint64
|   |   |   |   +---ro latest?       uint64
|   |   +---ro capture {capture}?
|   |   |   +---ro frame* [sequence-number]
|   |   |   |   +---ro sequence-number      uint64
|   |   |   |   +---ro timestamp?           yang:date-and-time
|   |   |   |   +---ro length?              uint32
|   |   |   |   +---ro preceding-interframe-gap?  uint32
|   |   |   |   +---ro data?                string
|   +---rw traffic-analyzer-egress! {egress-direction}?
|   |   +---rw filter! {filter}?
|   |   |   +---rw type                identityref
|   |   +---ro state
|   |   |   +---ro pkts?                yang:counter64

```

```

    +--ro errors?                yang:counter64
    +--ro testframe-stats
      |
      +--ro testframe-pkts?      yang:counter64
      +--ro sequence-errors?    yang:counter64
      +--ro payload-errors?     yang:counter64
      +--ro latency
        |
        +--ro samples?          uint64
        +--ro min?              uint64
        +--ro max?              uint64
        +--ro average?          uint64
        +--ro latest?           uint64
    +--ro capture {capture}?
      +--ro frame* [sequence-number]
        |
        +--ro sequence-number    uint64
        +--ro timestamp?         yang:date-and-time
        +--ro length?            uint32
        +--ro preceding-interframe-gap? uint32
        +--ro data?              string
augment /if:interfaces-state/if:interface/if:statistics:
    +--ro testframe-pkts?        yang:counter64 {ingress-direction}?
    +--ro testframe-sequence-errors? yang:counter64 {ingress-direction}?
    +--ro testframe-payload-errors? yang:counter64 {ingress-direction}?
augment /if:interfaces-state/if:interface/if:statistics:
    +--ro testframe-egress-pkts?   yang:counter64 {egress-direction}?
    +--ro testframe-egress-sequence-errors? yang:counter64 {egress-direction}?
    +--ro testframe-egress-payload-errors? yang:counter64 {egress-direction}?

```

5. Traffic Generator Module YANG

<CODE BEGINS> file "ietf-traffic-generator@2019-10-08.yang"

```

module ietf-traffic-generator {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-traffic-generator";
  prefix tg;

  import ietf-interfaces {
    prefix if;
  }
  import ietf-yang-types {
    prefix yang;
  }
  import iana-if-type {
    prefix ianaift;
  }

  organization
    "IETF Benchmarking Methodology Working Group";

```

contact

"WG Web: <<http://tools.ietf.org/wg/bmwg/>>
WG List: <<mailto:bmwg@ietf.org>>

Editor: Vladimir Vassilev
<<mailto:vladimir@lightside-instruments.com>>;

description

"This module contains a collection of YANG definitions for
description and management of network interconnect testers.

Copyright (c) 2019 IETF Trust and the persons identified as
authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or
without modification, is permitted pursuant to, and subject
to the license terms contained in, the Simplified BSD License
set forth in Section 4.c of the IETF Trust's Legal Provisions
Relating to IETF Documents
(<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see
the RFC itself for full legal notices.";

revision 2019-10-08 {

description

"Initial revision.";

reference

"RFC XXXX: Network Interconnect Tester";

}

feature egress-direction {

description

"The device can generate traffic in the egress direction.";

}

feature ingress-direction {

description

"The device can generate traffic in the ingress direction.";

}

feature multi-stream {

description

"The device can generate multi-stream traffic.";

}

feature ethernet {

description

"The device can generate ethernet traffic.";

```
}

feature ethernet-vlan {
  if-feature "ethernet";
  description
    "The device can generate vlan tagged ethernet traffic.";
}

grouping traffic-generator-burst-data {
  description
    "Generated traffic burst parameters.";
  leaf frame-size {
    type uint32;
    mandatory true;
    description
      "Size of the frames generated. For example for
      ethernet interfaces the following definition
      applies:

      Ethernet frame-size in octets includes:
      * Destination Address (6 octets),
      * Source Address (6 octets),
      * Frame Type (2 octets),
      * Data (min 46 octets or 42 octets + 4 octets 802.1Q tag),
      * CRC Checksum (4 octets).

      Ethernet frame-size does not include:
      * Preamble (dependent on MAC configuration
        by default 7 octets),
      * Start of frame delimiter (1 octet)

      Minimum standard ethernet frame-size is 64 bytes but
      generators might support smaller sizes for validation.";
  }
  choice frame-data-type {
    description
      "Choice of frame data type generated.";
    case raw-frame-data {
      leaf frame-data {
        type string {
          pattern '([0-9A-F]{2})*';
        }
        must 'string-length(.) <= (../frame-size*2)';
        description
          "The raw frame data specified as hexadecimal string.
          The specified data can be shorter then the ../frame-size
          value specifying only the header or the header and the
          payload without for example the 4 byte CRC Checksum
```

```
        in the case of a Ethernet frame.";
    }
}
leaf interframe-gap {
    type uint32;
    mandatory true;
    description
        "Length of the idle period between generated frames.
        For example for ethernet interfaces the following
        definition applies:

        Ethernet interframe-gap between transmission of frames
        known as the interframe gap (IFG). A brief recovery time
        between frames allows devices to prepare for
        reception of the next frame. The minimum
        interframe gap is 96 bit times (12 octet times) (the time it
        takes to transmit 96 bits (12 octets) of raw data on the
        medium). However the preamble (7 octets) and start of
        frame delimiter (1 octet) are considered a constant gap that
        should be included in the interframe-gap. Thus the minimum
        value for standard ethernet transmission should be considered
        20 octets.";
}
leaf interburst-gap {
    type uint32;
    description
        "Similar to the interframe-gap but takes place between
        any two bursts of the stream.";
}
leaf frames-per-burst {
    type uint32;
    description
        "Number of frames contained in a burst";
}
}

grouping traffic-generator-multi-stream-data {
    description
        "Multi stream traffic generation parameters.";
    container streams {
        description
            "Non-presence container holding the configured stream list.";
        list stream {
            key "id";
            description
                "Each stream repeats a burst until frames-per-stream
                count is reached followed by interstream-gap delay.";
        }
    }
}
```

```
    leaf id {
      type uint32;
      description
        "Number specifying the order of the stream.";
    }
    uses traffic-generator-burst-data;
    leaf frames-per-stream {
      type uint32;
      mandatory true;
      description
        "The count of frames to be generated before
        generation of the next stream is started.";
    }
    leaf interstream-gap {
      type uint32;
      mandatory true;
      description
        "Idle period after the last frame of the last burst.";
    }
  }
}

grouping ethernet-data {
  description
    "Ethernet frame data specific parameters.";
  reference
    "IEEE 802-2014 Clause 9.2";
  leaf src-mac-address {
    type yang:mac-address;
    description
      "Source Address field of the generated Ethernet packet.";
  }
  leaf dst-mac-address {
    type yang:mac-address;
    description
      "Destination Address field of the generated Ethernet packet.";
  }
  leaf ether-type {
    type uint16;
    description
      "Length/Type field of the generated Ethernet packet.";
  }
  container vlan {
    if-feature "ethernet-vlan";
    description
      "VLAN tag fields..";
    leaf id {
```

```
    type uint16 {
      range "0..4095";
    }
    mandatory true;
    description
      "VLAN id.";
  }
  leaf tpid {
    type uint16;
    default "33024";
    description
      "Configures the Tag Protocol Identifier (TPID)
      of the 802.1q VLAN tag sent. This value is used
      together with the vlan id for filtering incoming
      vlan tagged packets.";
  }
  leaf pcpi {
    type uint8 {
      range "0..7";
    }
    default "0";
    description
      "Configures the IEEE 802.1p Priority Code Point (PCP) value
      of the transmitted 802.1q VLAN tag.";
  }
  leaf cfi {
    type uint8 {
      range "0..1";
    }
    default "0";
    description
      "Configures the Canonical Format Identifier (CFI) field
      (shall be 0 for Ethernet switches) of the transmitted
      802.1q VLAN tag.";
  }
}
}

augment "/if:interfaces/if:interface" {
  description
    "Traffic generator augmentations of ietf-interfaces.";
  container traffic-generator {
    if-feature "egress-direction";
    description
      "Traffic generator for egress direction.";
    choice type {
      description
        "Choice of the type of the data model of the generator.
```

```
        Single or multi stream.";
    case single-stream {
        uses traffic-generator-burst-data;
    }
    case multi-stream {
        uses traffic-generator-multi-stream-data;
    }
}
leaf total-frames {
    type uint64;
    description
        "If this leaf is present the stream generation will stop
        after the specified number of frames are generated.";
}
}
container traffic-generator-ingress {
    if-feature "ingress-direction";
    description
        "Traffic generator for ingress direction.";
    choice type {
        description
            "Choice of the type of the data model of the generator.
            Single or multi stream.";
        case single-stream {
            uses traffic-generator-burst-data;
        }
        case multi-stream {
            uses traffic-generator-multi-stream-data;
        }
    }
    leaf total-frames {
        type uint64;
        description
            "If this leaf is present the stream generation will stop
            after the specified number of frames are generated.";
    }
}
}

augment "/if:interfaces-state/if:interface/if:statistics" {
    description
        "Counters of generated traffic octets and packets.";
    leaf generated-pkts {
        type yang:counter64;
        description
            "Traffic generator packets sent.";
    }
    leaf generated-octets {
```



```
    type yang:counter64;
    description
      "Traffic generator octets sent.";
  }
  leaf generated-ingress-pkts {
    if-feature "ingress-direction";
    type yang:counter64;
    description
      "Traffic generator packets generated in ingress mode.";
  }
  leaf generated-ingress-octets {
    if-feature "ingress-direction";
    type yang:counter64;
    description
      "Traffic generator octets generated in ingress mode.";
  }
}

augment "/if:interfaces/if:interface/tg:traffic-generator/tg:type/"
+ "tg:single-stream" {
  when "derived-from-or-self(..../if:type, 'ianaift:ethernetCsmacd')" {
    description
      "Ethernet interface type.";
  }
  if-feature "ethernet";
  description
    "Ethernet specific augmentation for egress
      single stream generator type.";
  uses ethernet-data;
}

augment "/if:interfaces/if:interface/tg:traffic-generator/tg:type/"
+ "tg:multi-stream/tg:streams/tg:stream" {
  when "derived-from-or-self(..../if:type, 'ianaift:ethernetCsmacd')" {
    description
      "Ethernet interface type.";
  }
  if-feature "ethernet";
  description
    "Ethernet specific augmentation for egress
      multi stream generator type.";
  uses ethernet-data;
}

augment "/if:interfaces/if:interface/tg:traffic-generator-ingress/tg:type/"
+ "tg:single-stream" {
  when "derived-from-or-self(..../if:type, 'ianaift:ethernetCsmacd')" {
    description
```

```
        "Ethernet interface type.";
    }
    if-feature "ethernet";
    description
        "Ethernet specific augmentation for ingress
        single stream generator type.";
    uses ethernet-data;
}

augment "/if:interfaces/if:interface/tg:traffic-generator-ingress/tg:type/"
    + "tg:multi-stream/tg:streams/tg:stream" {
    when "derived-from-or-self(..../if:type, 'ianaift:ethernetCsmacd')" {
        description
            "Ethernet interface type.";
    }
    if-feature "ethernet";
    description
        "Ethernet specific augmentation for ingress
        multi stream generator type.";
    uses ethernet-data;
}
}
```

<CODE ENDS>

6. Traffic Analyzer Module YANG

<CODE BEGINS> file "ietf-traffic-analyzer@2019-10-08.yang"

```
module ietf-traffic-analyzer {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-traffic-analyzer";
    prefix ta;

    import ietf-interfaces {
        prefix if;
    }
    import ietf-yang-types {
        prefix yang;
    }

    organization
        "IETF Benchmarking Methodology Working Group";
    contact
        "WG Web:  <http://tools.ietf.org/wg/bmwg/>
        WG List:  <mailto:bmwg@ietf.org>

        Editor:   Vladimir Vassilev
```

```
        <mailto:vladimir@lightside-instruments.com>";
description
  "This module contains a collection of YANG definitions for
  description and management of network interconnect testers.

  Copyright (c) 2019 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).

  This version of this YANG module is part of RFC XXXX; see
  the RFC itself for full legal notices.";

revision 2019-10-08 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: Network Interconnect Tester";
}

feature egress-direction {
  description
    "The device can analyze traffic from the egress direction.";
}

feature ingress-direction {
  description
    "The device can generate traffic from the ingress direction.";
}

feature filter {
  description
    "This feature indicates that the device implements
    filter that can specify a subset of packets to be
    analyzed when filtering is enabled.";
}

feature capture {
  description
    "This feature indicates that the device implements
    packet capture functionality.";
}
```

```
identity filter {
  description
    "Base filter identity.";
}

identity ethernet {
  base ta:filter;
  description
    "Ethernet packet fields filter.";
}

grouping statistics-data {
  description
    "Analyzer statistics.";
  leaf pkts {
    type yang:counter64;
    description
      "Total number of packets analyzed.";
  }
  leaf errors {
    type yang:counter64;
    description
      "Count of packets with errors.
      Not counted in the pkts or captured.
      For example packets with CRC error.";
  }
}

container testframe-stats {
  description
    "Statistics for testframe packets containing
    either sequence number, payload checksum,
    timestamp or any combination of these features.";
  leaf testframe-pkts {
    type yang:counter64;
    description
      "Total count of detected testframe packets.";
  }
  leaf sequence-errors {
    type yang:counter64;
    description
      "Total count of testframe packets with
      unexpected sequence number. After each sequence
      error the expected next sequence number is
      updated.";
  }
  leaf payload-errors {
    type yang:counter64;
    description
      "Total count of testframe packets with
```

```
        payload errors.";
    }
    container latency {
        description
            "Latency statistics.";
        leaf samples {
            type uint64;
            description
                "Total count of packets used for estimating
                 the latency statistics. Ideally
                 samples=../testframe-stats.";
        }
        leaf min {
            type uint64;
            units "nanoseconds";
            description
                "Minimum measured latency.";
        }
        leaf max {
            type uint64;
            units "nanoseconds";
            description
                "Maximum measured latency.";
        }
        leaf average {
            type uint64;
            units "nanoseconds";
            description
                "The sum of all sampled latencies divided
                 by the number of samples.";
        }
        leaf latest {
            type uint64;
            units "nanoseconds";
            description
                "Latency of the latest sample.";
        }
    }
}

grouping capture-data {
    description
        "Grouping with statistics and data
         of one or more captured frame.";
    container capture {
        if-feature "capture";
        description
```

```
    "Statistics and data of
      one or more captured frames.";
list frame {
  key "sequence-number";
  description
    "Statistics and data of a captured frame.";
  leaf sequence-number {
    type uint64;
    description
      "Incremental counter of frames captured.";
  }
  leaf timestamp {
    type yang:date-and-time;
    description
      "Timestamp of the moment the frame was captured.";
  }
  leaf length {
    type uint32;
    description
      "Frame length. Ideally the data captured will be
        of the same length but can be shorter
        depending on implementation limitations.";
  }
  leaf preceding-interframe-gap {
    type uint32;
    units "nanoseconds";
    description
      "Measured delay between the reception of the previous
        frame was completed and the reception of the current
        frame was started.";
  }
  leaf data {
    type string {
      pattern '([0-9A-F]{2})*';
    }
    description
      "Raw data of the captured frame.";
  }
}
}

grouping filter-data {
  description
    "Grouping with a filter container specifying the filtering
      rules for processing only a specific subset of the
      frames.";
  container filter {
```

```
    if-feature "filter";
    presence "When present packets are
              filtered before analyzed according
              to the filter type";
    description
      "Contains the filtering rules for processing only
       a specific subset of the frames.";
    leaf type {
      type identityref {
        base ta:filter;
      }
      mandatory true;
      description
        "Type of the applied filter. External modules can
         define alternative filter type identities.";
    }
  }
}

augment "/if:interfaces/if:interface" {
  description
    "Traffic analyzer augmentations of ietf-interfaces.";
  container traffic-analyzer {
    if-feature "ingress-direction";
    presence "Enables the traffic analyzer for ingress traffic.";
    description
      "Traffic analyzer for ingress direction.";
    uses filter-data;
    container state {
      config false;
      description
        "State data.";
      uses statistics-data;
      uses capture-data;
    }
  }
  container traffic-analyzer-egress {
    if-feature "egress-direction";
    presence "Enables the traffic analyzer for egress traffic.";
    description
      "Traffic analyzer for egress direction.";
    uses filter-data;
    container state {
      config false;
      description
        "State data.";
      uses statistics-data;
      uses capture-data;
    }
  }
}
```

```
    }
  }
}

augment "/if:interfaces/if:interface/ta:traffic-analyzer/ta:filter" {
  when "ta:type = 'ta:ethernet'";
  description
    "Ethernet frame specific filter type.";
  leaf ether-type {
    type uint16;
    description
      "The Ethernet Type (or Length) value
       defined by IEEE 802.";
    reference
      "IEEE 802-2014 Clause 9.2";
  }
}

augment "/if:interfaces-state/if:interface/if:statistics" {
  if-feature "ingress-direction";
  description
    "Counters implemented by ports with analyzers.";
  leaf testframe-pkts {
    type yang:counter64;
    description
      "Testframe packets recognized by the traffic analyzer.";
  }
  leaf testframe-sequence-errors {
    type yang:counter64;
    description
      "Testframe packets part of the recognized total
       but with unexpected sequence number.";
  }
  leaf testframe-payload-errors {
    type yang:counter64;
    description
      "Testframe packets part of the recognized total
       but with payload errors.";
  }
}

augment "/if:interfaces-state/if:interface/if:statistics" {
  if-feature "egress-direction";
  description
    "Counters implemented by ports with egress analyzers.";
  leaf testframe-egress-pkts {
    type yang:counter64;
    description
```



```
        "Testframe egress packets recognized by the traffic analyzer.";
    }
    leaf testframe-egress-sequence-errors {
        type yang:counter64;
        description
            "Testframe egress packets part of the recognized total
             but with unexpected sequence number.";
    }
    leaf testframe-egress-payload-errors {
        type yang:counter64;
        description
            "Testframe egress packets part of the recognized total
             but with payload errors.";
    }
}
}
```

<CODE ENDS>

7. IANA Considerations

This document registers three URIs and three YANG modules.

7.1. URI Registration

This document registers three URIs in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-traffic-generator
URI: urn:ietf:params:xml:ns:yang:ietf-traffic-analyzer
URI: urn:ietf:params:xml:ns:yang:ietf-loopback

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

7.2. YANG Module Name Registration

This document registers three YANG module in the YANG Module Names registry YANG [RFC6020].

```
name: ietf-traffic-generator
namespace: urn:ietf:params:xml:ns:yang:ietf-traffic-generator
prefix: tg
reference: RFC XXXX

name: ietf-traffic-analyzer
namespace: urn:ietf:params:xml:ns:yang:ietf-traffic-analyzer
prefix: ta
reference: RFC XXXX
```

8. Security Considerations

This document does not introduce any new security concerns in addition to those specified in [RFC7950], section 15.

9. References

9.1. Normative References

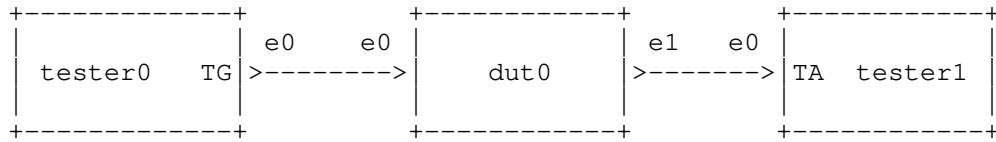
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.

9.2. Informative 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>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.

Appendix A. Examples

The following topology will be used for the examples in this section:



A.1. Basic Test Program

This program based on transactional network test API shows how the modules can be used:

```

#Connect to network
net=tntapi.connect("topology.xml")

# Configure DUTs and enable traffic-analyzers
net.node("dut0").edit( \
    "create /interfaces/interface[name='e0'] -- type=ethernetCsmacd")
net.node("dut0").edit(
    "create /interfaces/interface[name='e1'] -- type=ethernetCsmacd")
net.node("dut0").edit(
    "create /flows/flow[id='t0'] -- match/in-port=e0 "
    "actions/action[order='0']/output-action/out-port=e0")

net.node("tester1").edit(
    "create /interfaces/interface[name='e0']/traffic-analyzer")
net.commit()

#Get network state - before
before=net.get()

# Start traffic
net.node("tester0").edit(
    "create /interfaces/interface[name='e0']/traffic-generator -- "
    "frame-size=64 interframe-gap=20")

net.commit()

time.sleep(60)

# Stop traffic
net.node("tester1").edit("delete /interfaces/interface[name='e0']/\"
    "traffic-generator")
net.commit()

```

```
#Get network state - after
after=net.get()

#Report
sent_pkts=delta("tester0",before,after,
    "/interfaces/interface[name='e0']/statistics/out-unicast-pkts")

received_pkts=delta("tester1",before,after,
    "/interfaces/interface[name='e0']/statistics/in-unicast-pkts")

latency_max=absolute(after,
    "/interfaces/interface[name='e0']/traffic-analyzer/state/"
    "testframe-stats/latency/max")

#Cleanup
net.node("tester1").edit(
    "delete /interfaces/interface/traffic-analyzer")
net.node("dut0").edit("delete /flows")
net.node("dut0").edit("delete /interfaces")
net.commit()
```

A.2. Generating RFC2544 Testframes

In sec. C.2.6.4 Test Frames a detailed format is specified. The frame-data leaf allows full control over the generated frames payload.

```
...
net.node("tester1").edit(
    "merge /interfaces/interface[name='e0']/"
    "traffic-generator -- frame-data="
    "6CA96F0000026CA96F00000108004500"
    "002ED4A500000A115816C0000201C000"
    "0202C0200007001A0000010203040506"
    "0708090A0B0C0D0E0F101112")
...
```

Author's Address

Vladimir Vassilev
Lightside Instruments

Email: vladimir@lightside-instruments.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 28 April 2022

V. Vassilev
Lightside Instruments AS
25 October 2021

A YANG Data Model for Network Interconnect Tester Management
draft-vassilev-bmwg-network-interconnect-tester-07

Abstract

This document introduces new YANG model for use in network interconnect testing containing modules of traffic generator and traffic analyzer.

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 28 April 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	2
1.1.1. Definitions and Acronyms	2
1.1.2. Tree Diagram	3
1.2. Problem Statement	3
1.3. Objectives	3
1.4. Solution	4
2. Using the network interconnect tester model	5
3. Traffic Generator Module Tree Diagram	5
4. Traffic Analyzer Module Tree Diagram	6
5. Traffic Generator Module YANG	8
6. Traffic Analyzer Module YANG	15
7. IANA Considerations	23
7.1. URI Registration	23
7.2. YANG Module Name Registration	24
8. Security Considerations	24
8.1. ietf-traffic-generator.yang	24
8.2. ietf-traffic-analyzer.yang	25
9. References	25
9.1. Normative References	25
9.2. Informative References	26
Appendix A. Examples	26
A.1. Basic Test Program	26
A.2. Generating RFC2544 Testframes	28
Author's Address	28

1. Introduction

There is a need for standard mechanism to allow the specification and implementation of the transactions part of network tests. The mechanism should allow the control and monitoring of the data plane traffic in a transactional way. This document defines two YANG modules for test traffic generator and analyzer.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) defined in RFC 8342.

1.1. Terminology

1.1.1. Definitions and Acronyms

DUT: Device Under Test

TA: Traffic Analyzer

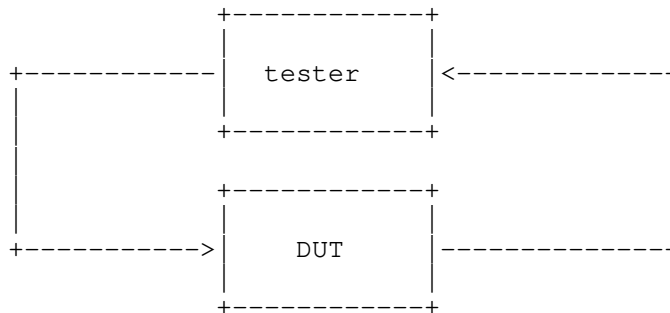
TG: Traffic Generator

1.1.2. Tree Diagram

For a reference to the annotations used in tree diagrams included in this document, please see YANG Tree Diagrams [RFC8340].

1.2. Problem Statement

Network interconnect tests require active network elements part of the tested network that generate test traffic and network elements that analyze the test traffic at one or more points of its path. A network interconnect tester is a device that can either generate test traffic, analyze test traffic or both. Here is a figure borrowed from [RFC2544] representing the horseshoe test setup topology consisting of a single tester and a single DUT connected in a network interconnect loop.



This document attempts to address the problem of defining YANG model of a network interconnect tester that can be used for development of vendor independent network interconnect tests and utilize the advantages of transactional management using standard protocols like NETCONF.

1.3. Objectives

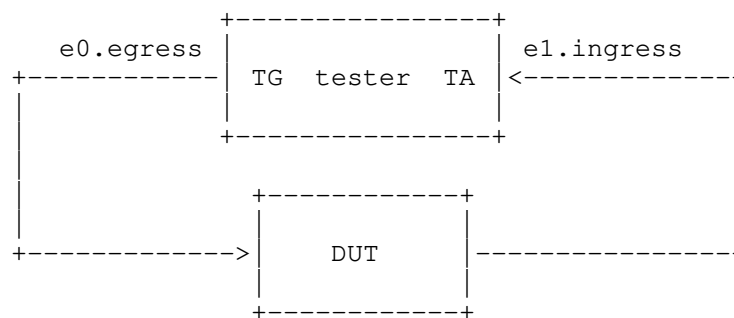
This section describes some of the design objectives for the model. It should:

- * provide means to specify the generated traffic as streams of cyclic sequence of bursts with configurable frame size, frame data, interframe gap and interburst gap.
- * have a mandatory single stream mode and optional multi stream mode.

- * provide means for configuration of traffic streams with static frame data where frames with identical frame data are sent during the lifetime of the stream.
- * provide means for configuration of traffic streams with dynamic frame data where frames contain fields with dynamic data like generation time and sequence number.
- * allow third parties to augment the base module with alternative dynamic fields of frame data extensions.
- * provide means for realtime synchronization and orchestration of the generated streams.
- * provide counters for received test traffic frames and octets.
- * provide latency statistic in the case of test traffic with dynamic frame data that includes timestamp.
- * provide sequence number errors in the case of test traffic with dynamic frame data that includes sequence number.

1.4. Solution

The proposed model splits the design into 2 modules - 1) Traffic Generator module (TG), 2) Traffic Analyzer module (TA). The modules are implemented as augmentations of the ietf-interfaces [RFC8343] module adding configuration and state data that models the functionality of a network interconnect tester. The TA and TG modules concept is illustrated with the following diagram of a tester with two interfaces (named e0 and e1) connected in a loop with single DUT:



2. Using the network interconnect tester model

Basic example of how the model can be used in transactional network test program to control the testers part of a network and report counter statistics and timing measurement data is presented in Appendix A. All example cases present the configuration and state data from a single test trial. The search algorithm logic that operates to control the trial configuration is outside the scope of this document. One of the examples demonstrates the use of the [RFC2544] defined testframe packet.

3. Traffic Generator Module Tree Diagram

```

module: ietf-traffic-generator
  augment /if:interfaces/if:interface:
    +--rw traffic-generator {egress-direction}?
      +--rw (type)?
        +--:(single-stream)
          +--rw testframe-type?      identityref
          +--rw frame-size           uint32
          +--rw frame-data?          string
          +--rw interframe-gap       uint32
          +--rw interburst-gap?      uint32
          +--rw frames-per-burst?    uint32
          +--rw src-mac-address?     yang:mac-address {ethernet}?
          +--rw dst-mac-address?     yang:mac-address {ethernet}?
          +--rw ether-type?          uint16 {ethernet}?
        +--:(multi-stream)
          +--rw streams
            +--rw stream* [id]
              +--rw id               uint32
              +--rw testframe-type?  identityref
              +--rw frame-size       uint32
              +--rw frame-data?      string
              +--rw interframe-gap   uint32
              +--rw interburst-gap?  uint32
              +--rw frames-per-burst uint32
              +--rw frames-per-stream uint32
              +--rw interstream-gap  uint32
              +--rw src-mac-address?
                | yang:mac-address {ethernet}?
              +--rw dst-mac-address?
                | yang:mac-address {ethernet}?
              +--rw ether-type?      uint16 {ethernet}?
          +--rw realtime-epoch?
            | yang:date-and-time {realtime-epoch}?
          +--rw total-frames?      uint64
      +--rw traffic-generator-ingress {ingress-direction}?

```

```

+--rw (type)?
|   +--:(single-stream)
|   |   +--rw testframe-type?      identityref
|   |   +--rw frame-size           uint32
|   |   +--rw frame-data?          string
|   |   +--rw interframe-gap       uint32
|   |   +--rw interburst-gap?      uint32
|   |   +--rw frames-per-burst?    uint32
|   |   +--rw src-mac-address?      yang:mac-address {ethernet}?
|   |   +--rw dst-mac-address?      yang:mac-address {ethernet}?
|   |   +--rw ether-type?          uint16 {ethernet}?
|   +--:(multi-stream)
|   |   +--rw streams
|   |   |   +--rw stream* [id]
|   |   |   |   +--rw id           uint32
|   |   |   |   +--rw testframe-type? identityref
|   |   |   |   +--rw frame-size   uint32
|   |   |   |   +--rw frame-data?  string
|   |   |   |   +--rw interframe-gap uint32
|   |   |   |   +--rw interburst-gap? uint32
|   |   |   |   +--rw frames-per-burst? uint32
|   |   |   |   +--rw frames-per-stream uint32
|   |   |   |   +--rw interstream-gap uint32
|   |   |   |   +--rw src-mac-address?
|   |   |   |   |   yang:mac-address {ethernet}?
|   |   |   |   +--rw dst-mac-address?
|   |   |   |   |   yang:mac-address {ethernet}?
|   |   |   |   +--rw ether-type?
|   |   |   |   |   uint16 {ethernet}?
|   +--rw realtime-epoch?
|   |   yang:date-and-time {realtime-epoch}?
+--rw total-frames?      uint64

```

4. Traffic Analyzer Module Tree Diagram

```

module: ietf-traffic-analyzer
augment /if:interfaces/if:interface:
+--rw traffic-analyzer! {ingress-direction}?
|   +--rw filter! {filter}?
|   |   +--rw type      identityref
|   |   +--rw ether-type? uint16
|   +--rw capture {capture}?
|   |   +--rw start-trigger
|   |   |   +--rw (start-trigger)?
|   |   |   |   +--:(frame-index)
|   |   |   |   |   +--rw frame-index?      uint64
|   |   |   |   +--:(testframe-index)
|   |   |   |   |   +--rw testframe-index?    uint64

```

```

    +---rw stop-trigger
        +---rw (stop-trigger)?
            +---:(when-full)
                +---rw when-full?    empty
+---ro state
    +---ro pkts?                    yang:counter64
    +---ro octets?                  yang:counter64
    +---ro idle-octets?             yang:counter64 {idle-octets-counter}?
    +---ro errors?                  yang:counter64
    +---ro testframe-stats
        +---ro testframe-pkts?     yang:counter64
        +---ro sequence-errors?    yang:counter64
        +---ro payload-errors?     yang:counter64
        +---ro latency
            +---ro samples?         uint64
            +---ro min?             uint64
            +---ro max?             uint64
            +---ro average?         uint64
            +---ro latest?          uint64
    +---ro capture {capture}?
        +---ro frame* [sequence-number]
            +---ro sequence-number    uint64
            +---ro timestamp?         yang:date-and-time
            +---ro length?            uint32
            +---ro preceding-interframe-gap?  uint32
            +---ro data?              string
+---rw traffic-analyzer-egress! {egress-direction}?
    +---rw filter! {filter}?
        | +---rw type    identityref
    +---rw capture {capture}?
        +---rw start-trigger
            +---rw (start-trigger)?
                +---:(frame-index)
                    | +---rw frame-index?    uint64
                +---:(testframe-index)
                    | +---rw testframe-index?  uint64
        +---rw stop-trigger
            +---rw (stop-trigger)?
                +---:(when-full)
                    +---rw when-full?    empty
+---ro state
    +---ro pkts?                    yang:counter64
    +---ro octets?                  yang:counter64
    +---ro idle-octets?             yang:counter64 {idle-octets-counter}?
    +---ro errors?                  yang:counter64
    +---ro testframe-stats
        +---ro testframe-pkts?     yang:counter64
        +---ro sequence-errors?    yang:counter64

```

```

    +--ro payload-errors?      yang:counter64
    +--ro latency
      +--ro samples?          uint64
      +--ro min?              uint64
      +--ro max?              uint64
      +--ro average?          uint64
      +--ro latest?           uint64
    +--ro capture {capture}?
      +--ro frame* [sequence-number]
        +--ro sequence-number      uint64
        +--ro timestamp?           yang:date-and-time
        +--ro length?              uint32
        +--ro preceding-interframe-gap? uint32
        +--ro data?                string

```

5. Traffic Generator Module YANG

<CODE BEGINS> file "ietf-traffic-generator@2021-10-25.yang"

```

module ietf-traffic-generator {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-traffic-generator";
  prefix nttg;

  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model For Interface Management";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import iana-if-type {
    prefix ianaift;
    reference
      "RFC 7224: IANA Interface Type YANG Module";
  }

  organization
    "IETF Benchmarking Methodology Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/bmwg/>
    WG List:  <mailto:bmwg@ietf.org>

    Editor:   Vladimir Vassilev
              <mailto:vladimir@lightside-instruments.com>";

```

description

"This module contains a collection of YANG definitions for description and management of network interconnect testers.

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>). This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.";

revision 2021-10-25 {

description

"Initial revision.";

reference

"RFC XXXX: A YANG Data Model for Network Interconnect Tester Management";

}

feature egress-direction {

description

"The device can generate traffic in the egress direction.";

}

feature ingress-direction {

description

"The device can generate traffic in the ingress direction.";

}

feature multi-stream {

description

"The device can generate multi-stream traffic.";

}

feature ethernet {

description

"The device can generate ethernet traffic.";

}

feature realtime-epoch {

description

"The device can generate traffic precisely at configured realtime epoch.";

```
}

identity testframe-type {
  description
    "Base identity for all testframe types.";
}

identity static {
  base testframe-type;
  description
    "Identity for static testframe.
    The frame data and size are constant.";
}

identity dynamic {
  base testframe-type;
  description
    "Identity to be used as base for dynamic
    testframe type identities defined
    in external modules.

    When used itself it identifies dynamic testframe
    where the last 18 octets of the payload contain
    incrementing sequence number field (8 octets)
    followed by timestamp field in the
    IEEE 1588-2008 format (10 octets). If frame data is defined
    for the last 18 octets of the payload it will be ignored
    and overwritten with dynamic data according to this
    specification.";
}

grouping common-data {
  description
    "Common configuration data.";
  leaf realtime-epoch {
    if-feature "realtime-epoch";
    type yang:date-and-time;
    description
      "If this leaf is present the stream generation will start
      at the specified realtime epoch.";
  }
  leaf total-frames {
    type uint64;
    description
      "If this leaf is present the traffic generation will stop
      after the specified number of frames are generated.";
  }
}
```

```
grouping burst-data {
  description
    "Generated traffic burst parameters.";
  leaf testframe-type {
    type identityref {
      base nttg:testframe-type;
    }
    default "nttg:static";
    description
      "In case of dynamic testframes this leaf specifies
       the dynamic testframe identity.";
  }
  leaf frame-size {
    type uint32;
    mandatory true;
    description
      "Size of the frames generated. For example for
       ethernet interfaces the following definition
       applies:

       Ethernet frame-size in octets includes:
       * Destination Address (6 octets),
       * Source Address (6 octets),
       * Frame Type (2 octets),
       * Data (min 46 octets or 42 octets + 4 octets 802.1Q tag),
       * CRC Checksum (4 octets).

       Ethernet frame-size does not include:
       * Preamble (dependent on MAC configuration
         by default 7 octets),
       * Start of frame delimiter (1 octet)

       Minimum standard ethernet frame-size is 64 bytes but
       generators might support smaller sizes for validation.";
  }
  leaf frame-data {
    type string {
      pattern '([0-9A-F]{2})*';
    }
    must 'string-length(.)<=../frame-size*2';
    description
      "The raw frame data specified as hexadecimal string.
       The specified data can be shorter then the ../frame-size
       value specifying only the header or the header and the
       payload with or without the 4 byte CRC Checksum
       in the case of a Ethernet frame.";
  }
  leaf interframe-gap {
```

```
    type uint32;
    mandatory true;
    description
      "Length of the idle period between generated frames.
      For example for ethernet interfaces the following
      definition applies:

      Ethernet interframe-gap between transmission of frames
      known as the interframe gap (IFG). A brief recovery time
      between frames allows devices to prepare for
      reception of the next frame. The minimum
      interframe gap is 96 bit times (12 octet times) (the time it
      takes to transmit 96 bits (12 octets) of raw data on the
      medium). However the preamble (7 octets) and start of
      frame delimiter (1 octet) are considered a constant gap that
      should be included in the interframe-gap. Thus the minimum
      value for standard ethernet transmission should be considered
      20 octets.";
  }
  leaf interburst-gap {
    type uint32;
    description
      "Similar to the interframe-gap but takes place between
      any two bursts of the stream.";
  }
  leaf frames-per-burst {
    type uint32;
    description
      "Number of frames contained in a burst";
  }
}

grouping multi-stream-data {
  description
    "Multi stream traffic generation parameters.";
  container streams {
    description
      "Non-presence container holding the configured stream list.";
    list stream {
      key "id";
      description
        "Each stream repeats a burst until frames-per-stream
        count is reached followed by interstream-gap delay.";
      leaf id {
        type uint32;
        description
          "Number specifying the order of the stream.";
      }
    }
  }
}
```



```
    uses burst-data;
    leaf frames-per-stream {
        type uint32;
        mandatory true;
        description
            "The count of frames to be generated before
             generation of the next stream is started.";
    }
    leaf interstream-gap {
        type uint32;
        mandatory true;
        description
            "Idle period after the last frame of the last burst.";
    }
}
}

grouping ethernet-data {
    description
        "Ethernet frame data specific parameters.";
    reference
        "IEEE 802-2014 Clause 9.2";
    leaf src-mac-address {
        type yang:mac-address;
        description
            "Source Address field of the generated Ethernet packet.";
    }
    leaf dst-mac-address {
        type yang:mac-address;
        description
            "Destination Address field of the generated Ethernet packet.";
    }
    leaf ether-type {
        type uint16;
        description
            "Length/Type field of the generated Ethernet packet.";
    }
}

augment "/if:interfaces/if:interface" {
    description
        "Traffic generator augmentations of ietf-interfaces.";
    container traffic-generator {
        if-feature "egress-direction";
        description
            "Traffic generator for egress direction.";
        choice type {
```

```
    description
      "Choice of the type of the data model of the generator.
       Single or multi stream.";
    case single-stream {
      uses burst-data;
    }
    case multi-stream {
      uses multi-stream-data;
    }
  }
  uses common-data;
}
container traffic-generator-ingress {
  if-feature "ingress-direction";
  description
    "Traffic generator for ingress direction.";
  choice type {
    description
      "Choice of the type of the data model of the generator.
       Single or multi stream.";
    case single-stream {
      uses burst-data;
    }
    case multi-stream {
      uses multi-stream-data;
    }
  }
  uses common-data;
}
}

augment "/if:interfaces/if:interface/nttg:traffic-generator/"
+ "nttg:type/nttg:single-stream" {
  when "derived-from-or-self(..../if:type, 'ianaift:ethernetCsmacd')" {
    description
      "Ethernet interface type.";
  }
  if-feature "ethernet";
  description
    "Ethernet specific augmentation for egress
     single stream generator type.";
  uses ethernet-data;
}

augment "/if:interfaces/if:interface/nttg:traffic-generator/"
+ "nttg:type/nttg:multi-stream/nttg:streams/nttg:stream" {
  when "derived-from-or-self(..../if:type,"
  + "'ianaift:ethernetCsmacd')" {
```

```
        description
          "Ethernet interface type.";
      }
      if-feature "ethernet";
      description
        "Ethernet specific augmentation for egress
        multi stream generator type.";
      uses ethernet-data;
    }

    augment "/if:interfaces/if:interface/nttg:traffic-generator-ingress/"
      + "nttg:type/nttg:single-stream" {
      when "derived-from-or-self(..../if:type, 'ianaift:ethernetCsmacd')" {
        description
          "Ethernet interface type.";
      }
      if-feature "ethernet";
      description
        "Ethernet specific augmentation for ingress
        single stream generator type.";
      uses ethernet-data;
    }

    augment "/if:interfaces/if:interface/nttg:traffic-generator-ingress/"
      + "nttg:type/nttg:multi-stream/nttg:streams/nttg:stream" {
      when "derived-from-or-self(..../..../if:type,"
        + "'ianaift:ethernetCsmacd')" {
        description
          "Ethernet interface type.";
      }
      if-feature "ethernet";
      description
        "Ethernet specific augmentation for ingress
        multi stream generator type.";
      uses ethernet-data;
    }
  }

<CODE ENDS>
```

6. Traffic Analyzer Module YANG

<CODE BEGINS> file "ietf-traffic-analyzer@2021-10-25.yang"

```
module ietf-traffic-analyzer {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-traffic-analyzer";
  prefix ntta;

  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model For Interface Management";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  organization
    "IETF Benchmarking Methodology Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/bmwg/>
    WG List:  <mailto:bmwg@ietf.org>

    Editor:    Vladimir Vassilev
               <mailto:vladimir@lightside-instruments.com>;

  description
    "This module contains a collection of YANG definitions for
    description and management of network interconnect testers.

    Copyright (c) 2021 IETF Trust and the persons identified as
    authors of the code.  All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD License
    set forth in Section 4.c of the IETF Trust's Legal Provisions
    Relating to IETF Documents
    (http://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC XXXX; see
    the RFC itself for full legal notices.";

  revision 2021-10-25 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for
      Network Interconnect Tester Management";
  }
}
```

```
feature egress-direction {
  description
    "The device can analyze traffic from the egress direction.";
}

feature ingress-direction {
  description
    "The device can generate traffic from the ingress direction.";
}

feature filter {
  description
    "This feature indicates that the device implements
    filter that can specify a subset of packets to be
    analyzed when filtering is enabled.";
}

feature idle-octets-counter {
  description
    "This feature indicates that the device implements
    idle-octets counter that accumulates the time
    the link is not utilized. The minimum required
    idle gaps are not counted as idle octets.";
}

feature capture {
  description
    "This feature indicates that the device implements
    packet capture functionality.";
}

identity filter {
  description
    "Base filter identity.";
}

identity ethernet {
  base nttta:filter;
  description
    "Ethernet packet fields filter.";
}

grouping statistics-data {
  description
    "Analyzer statistics.";
  leaf pkts {
    type yang:counter64;
    description
      "Total number of packets analyzed.";
  }
}
```

```
}
leaf octets {
  type yang:counter64;
  description
    "This counter is identical with the in-octets/out-octets
    counters defined in RFC8343 except that it counts the
    octets since the analyzer was created.";
}
leaf idle-octets {
  if-feature "idle-octets-counter";
  type yang:counter64;
  description
    "Total accumulated period with no frame transmission
    taking place measured in octets at the current link
    speed. Octets not counted in ../octets but not idle are
    for example layer 1 framing octets - for Ethernet links
    7+1 preamble octets per packet.";
}
leaf errors {
  type yang:counter64;
  description
    "Count of packets with errors.
    Not counted in the pkts or captured.
    For example packets with CRC error.";
}
container testframe-stats {
  description
    "Statistics for testframe packets containing
    either sequence number, payload checksum,
    timestamp or any combination of these features.";
  leaf testframe-pkts {
    type yang:counter64;
    description
      "Total count of detected testframe packets.";
  }
  leaf sequence-errors {
    type yang:counter64;
    description
      "Total count of testframe packets with
      unexpected sequence number. After each sequence
      error the expected next sequence number is
      updated.";
  }
  leaf payload-errors {
    type yang:counter64;
    description
      "Total count of testframe packets with
      payload errors.";
```

```
    }
    container latency {
      description
        "Latency statistics.";
      leaf samples {
        type uint64;
        description
          "Total count of packets used for estimating
           the latency statistics. Ideally
           samples=../testframe-stats.";
      }
      leaf min {
        type uint64;
        units "nanoseconds";
        description
          "Minimum measured latency.";
      }
      leaf max {
        type uint64;
        units "nanoseconds";
        description
          "Maximum measured latency.";
      }
      leaf average {
        type uint64;
        units "nanoseconds";
        description
          "The sum of all sampled latencies divided
           by the number of samples.";
      }
      leaf latest {
        type uint64;
        units "nanoseconds";
        description
          "Latency of the latest sample.";
      }
    }
  }
}

grouping capture-config-data {
  description
    "Grouping with a capture configuration container.";
  container capture {
    if-feature "capture";

    description
      "Contains capture parameters.";
```

```
container start-trigger {
  description
    "Configures when the capture start is triggered.";
  choice start-trigger {
    description
      "If none of the cases in this choice are configured the
      capture process starts from the first frame received.";
    case frame-index {
      description
        "Start capturing frames at the specified frame index.";
      leaf frame-index {
        type uint64;
        description
          "First captured frame index.";
      }
    }
    case testframe-index {
      description
        "Start capturing frames at the specified
        testframe index.";
      leaf testframe-index {
        type uint64;
        description
          "Starts capture as specified testframe index.";
      }
    }
  }
}
container stop-trigger {
  description
    "Configures when the capture is stopped.";
  choice stop-trigger {
    description
      "If none of the cases in this choice are configured the
      captured frames are always the last frames received for
      as many frames the implementation can buffer.";
    case when-full {
      description
        "Stops capturing when the implementation can not store
        more frames.";
      leaf when-full {
        type empty;
        description
          "When present in configuration capture stops when
          the capture buffer is full.";
      }
    }
  }
}
```



```
    }  
  }  
}  
  
grouping capture-data {  
  description  
    "Grouping with statistics and data  
    of one or more captured frame.";  
  container capture {  
    if-feature "capture";  
    description  
      "Statistics and data of  
      one or more captured frames.";  
    list frame {  
      key "sequence-number";  
      description  
        "Statistics and data of a captured frame.";  
      leaf sequence-number {  
        type uint64;  
        description  
          "Incremental counter of frames captured.";  
      }  
      leaf timestamp {  
        type yang:date-and-time;  
        description  
          "Timestamp of the moment the frame was captured.";  
      }  
      leaf length {  
        type uint32;  
        description  
          "Frame length. Ideally the data captured will be  
          of the same length but can be shorter  
          depending on implementation limitations.";  
      }  
      leaf preceding-interframe-gap {  
        type uint32;  
        units "nanoseconds";  
        description  
          "Measured delay between the reception of the previous  
          frame was completed and the reception of the current  
          frame was started.";  
      }  
      leaf data {  
        type string {  
          pattern '([0-9A-F]{2})*';  
        }  
        description  
          "Raw data of the captured frame.";  
      }  
    }  
  }  
}
```

```
    }
  }
}

grouping filter-data {
  description
    "Grouping with a filter container specifying the filtering
    rules for processing only a specific subset of the
    frames.";
  container filter {
    if-feature "filter";
    presence "When present packets are
    filtered before analyzed according
    to the filter type";
    description
      "Contains the filtering rules for processing only
      a specific subset of the frames.";
    leaf type {
      type identityref {
        base ntt:filter;
      }
      mandatory true;
      description
        "Type of the applied filter. External modules can
        define alternative filter type identities.";
    }
  }
}

augment "/if:interfaces/if:interface" {
  description
    "Traffic analyzer augmentations of ietf-interfaces.";
  container traffic-analyzer {
    if-feature "ingress-direction";
    presence "Enables the traffic analyzer for ingress traffic.";
    description
      "Traffic analyzer for ingress direction.";
    uses filter-data;
    uses capture-config-data;
    container state {
      config false;
      description
        "State data.";
      uses statistics-data;
      uses capture-data;
    }
  }
}
```

```
    container traffic-analyzer-egress {
      if-feature "egress-direction";
      presence "Enables the traffic analyzer for egress traffic.";
      description
        "Traffic analyzer for egress direction.";
      uses filter-data;
      uses capture-config-data;
      container state {
        config false;
        description
          "State data.";
        uses statistics-data;
        uses capture-data;
      }
    }
  }
}

augment "/if:interfaces/if:interface/ntta:traffic-analyzer/"
  + "ntta:filter" {
  when "derived-from-or-self(ntta:type, 'ntta:ethernet')";
  description
    "Ethernet frame specific filter type.";
  leaf ether-type {
    type uint16;
    description
      "The Ethernet Type (or Length) value
        defined by IEEE 802.";
    reference
      "IEEE 802-2014 Clause 9.2";
  }
}
}
```

<CODE ENDS>

7. IANA Considerations

This document registers two URIs and two YANG modules.

7.1. URI Registration

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-traffic-generator
URI: urn:ietf:params:xml:ns:yang:ietf-traffic-analyzer

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

7.2. YANG Module Name Registration

This document registers two YANG module in the YANG Module Names registry YANG [RFC6020].

```
name: ietf-traffic-generator
namespace: urn:ietf:params:xml:ns:yang:ietf-traffic-generator
prefix: nttg
reference: RFC XXXX
```

```
name: ietf-traffic-analyzer
namespace: urn:ietf:params:xml:ns:yang:ietf-traffic-analyzer
prefix: ntta
reference: RFC XXXX
```

8. Security Considerations

The YANG modules defined in this document are designed to be accessed via the NETCONF protocol RFC 6241 [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory to implement secure transport is SSH RFC 6242 [RFC6242]. The NETCONF access control model RFC 6536 [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of data nodes defined in this YANG module which are writable/creatable/deletable (i.e. config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g. edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

8.1. ietf-traffic-generator.yang

The ietf-traffic-generator YANG module controls a stateless traffic generator which is intended to be used for testing and verification purposes but can be used for malicious purposes like generating network traffic part of a Denial-of-Service (DoS) attack. This should be taken into consideration when granting write access to the following container and descendant data nodes:

```
* /if:interfaces/if:interface/nttg:traffic-generator
```

8.2. ietf-traffic-analyzer.yang

The ietf-traffic-analyzer YANG module controls a traffic analyzer which is designed for use in testing and verification but can be used for reading information contained in packets sent and received on any of the interfaces on systems that implement the capture feature. This should be taken into consideration when granting read access to the following container and descendant data nodes:

```
* /if:interfaces/if:interface/ntta:traffic-analyzer/ntta:capture
```

9. References

9.1. Normative References

- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<https://www.rfc-editor.org/info/rfc6536>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", RFC 7224, DOI 10.17487/RFC7224, May 2014, <<https://www.rfc-editor.org/info/rfc7224>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.

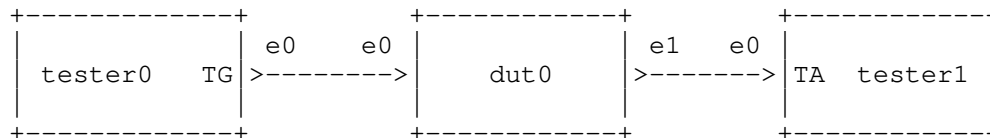
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.

9.2. Informative References

- [IEEE1588] IEEE, "IEEE 1588-2008", 2008.
- [IEEE802.3-2014] IEEE WG802.3 - Ethernet Working Group, "IEEE 802.3-2014", 2014.
- [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>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.

Appendix A. Examples

The following topology will be used for the examples in this section:



A.1. Basic Test Program

This pseudo code program orchestrates a network test and shows how the model can be used:

```

#Connect to network
net=connect("topology.xml")

# Configure DUTs and enable traffic-analyzers
net.node("dut0").edit( \
    "create /interfaces/interface[name='e0'] -- type=ethernetCsmacd")
net.node("dut0").edit(

```

```
"create /interfaces/interface[name='e1'] -- type=ethernetCsmacd")
net.node("dut0").edit(
  "create /flows/flow[id='t0'] -- match/in-port=e0 "
  "actions/action[order='0']/output-action/out-port=e1")

net.node("tester1").edit(
  "create /interfaces/interface[name='e0']/traffic-analyzer")
net.commit()

#Get network state - before
before=net.get()

# Start traffic
net.node("tester0").edit(
  "create /interfaces/interface[name='e0']/traffic-generator -- "
  "frame-size=64 interframe-gap=20")

net.commit()

time.sleep(60)

# Stop traffic
net.node("tester1").edit("delete /interfaces/interface[name='e0']/\"
                          \"traffic-generator")
net.commit()

#Get network state - after
after=net.get()

#Report
sent_pkts=delta("tester0",before,after,
  "/interfaces/interface[name='e0']/statistics/out-unicast-pkts")

received_pkts=delta("tester1",before,after,
  "/interfaces/interface[name='e0']/statistics/in-unicast-pkts")

latency_max=absolute(after,
  "/interfaces/interface[name='e0']/traffic-analyzer/state/"
  "testframe-stats/latency/max")

#Cleanup
net.node("tester1").edit(
  "delete /interfaces/interface/traffic-analyzer")
net.node("dut0").edit("delete /flows")
net.node("dut0").edit("delete /interfaces")
net.commit()
```

A.2. Generating RFC2544 Testframes

In sec. C.2.6.4 Test Frames a detailed format is specified. The frame-data leaf allows full control over the generated frames payload.

```
...
net.node("tester1").edit(
  "merge /interfaces/interface[name='e0']/"
  "traffic-generator -- frame-data="
  "6CA96F00000026CA96F000000108004500"
  "002ED4A5000000A115816C00000201C000"
  "0202C0200007001A00000010203040506"
  "0708090A0B0C0D0E0F101112")
...
```

Author's Address

Vladimir Vassilev
Lightside Instruments AS

Email: vladimir@lightside-instruments.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: December 28, 2019

S. Jacob, Ed.
V. Nagarajan
Juniper Networks
June 26, 2019

Benchmarking Methodology for EVPN Multicasting
draft-vikjac-bmwg-evpnmultest-02

Abstract

This document defines methodologies for benchmarking IGMP proxy performance over EVPN-VXLAN. IGMP proxy over EVPN is defined in draft-ietf-bess-evpn-igmp-mld-proxy-02, and is being deployed in data center networks. Specifically this document defines the methodologies for benchmarking IGMP proxy convergence, leave latency Scale, Core isolation, high availability and longevity.

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 December 28, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
1.2. Terminologies	3
2. Test Topology	4
3. Test Cases	6
3.1. How long it takes to learn (X1...Xn) IGMP join messages in DUT	6
3.2. How long it takes to clear the (*,G) entries in the DUT .	7
3.3. How long it takes the DUT to stop forwarding the traffic(Measuring the leave latency)	7
3.4. How long it takes to learn (X1...Xn) IGMP join messages for N vlans in DUT	8
3.5. How long it takes to clear the (*,G) entries in the DUT for N vlans	9
3.6. How long it takes the DUT to stop forwarding the traffic for N vlans(Measuring the leave latency)	9
3.7. How long it takes to learn (X1...Xn) IGMP join messages for N vlans in DUT working EVPN AA mode	10
3.8. How long it takes to clear the (*,G) entries for N vlans in DUT working EVPN AA	11
3.9. How long it takes the DUT operating in EVPN AA to stop forwarding the traffic for N vlans(Measuring the leave latency)	11
3.10. How long does it take the DUT in EVPN AA to handle Join Timeout and stop forwarding	12
3.11. How long does it take an Ingress to learn a remote Type-6 join, create state and forwarding	13
4. Link Flap	13
4.1. To Measure the multicast packet loss in EVPN AA scenario on a CE link failure	14
4.2. To Measure the multicast packet loss in EVPN AA scenario on a core failure	14
4.3. To Measure the multicast packet loss in EVPN AA scenario on a routing failure	15
5. Scale Convergence	15
5.1. To measure the packet loss during the core link failure.	16
6. High Availability	16
6.1. To Record the whether there is traffic loss due to routing engine failover for redundancy test.	16
7. SOAK Test	17
7.1. To Measure the stability of the DUT with scale and traffic.	17
8. Acknowledgements	18

9. IANA Considerations	18
10. Security Considerations	18
11. References	18
11.1. Normative References	18
11.2. Informative References	18
Appendix A. Appendix	18
Authors' Addresses	18

1. Introduction

IGMP proxy over EVPN-VXLAN is defined in draft-ietf-bess-evpn-igmp-mld-proxy-02, and is being deployed in data center networks. Specifically this document defines the methodologies for benchmarking IGMP proxy convergence, leave latency Scale, Core isolation, high availability and longevity.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2. Terminologies

Leaf A layer 2 or layer 3 capable device

Spine layer 3 capable device which is used to inter connect leaves

CE Customer Router/Devices/Switch.

AA EVPN Terminologies AA All-Active.

AC Attachment Circuit

RT Router Tester.

DUT Device under Test.

Sub Interface Each physical Interfaces is subdivided in to Logical units.

EVI EVPN Instances which will be running on sub interface or physical port of the provider Edge routers.

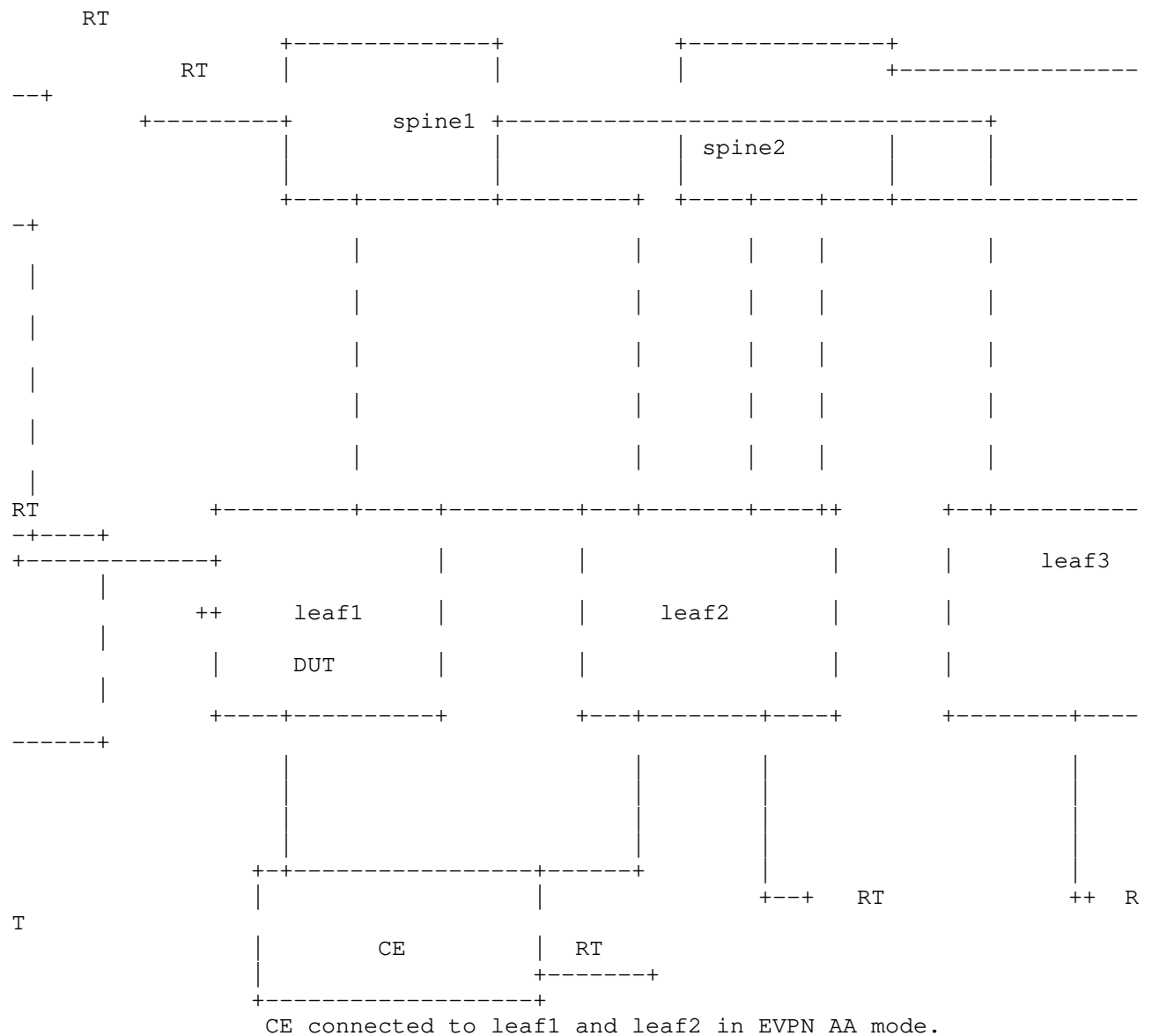
DF Designated Forwarder.

ESI Ethernet Segment Identifier.

2. Test Topology

EVPN Overlay Network running on leaf1, leaf2 leaf3, spine1 and spine 2
:

Topology Diagram



Topology 1

Topology Diagram

Figure 1

There are six routers in the topology. Leaf1,leaf2, leaf3,spine1,spine2 emulating a data center network. CE is a customer device connected to leaf1 and leaf2, it is configured with bridge domains in different vlans. The router tester is connected to CE,leaf1,leaf2,leaf3,spine1 and spine 2 to emulate multicast source and host generating igmp join/leave.

All routers except CE are configured with EBGp for the underlay

All router are configured with EVPN-VXLAN overlay

All leaves and spine must be configured "N" EVPN-VXLAN instances depends up on the cases.

Leaf1 and Leaf2 must be configured with ESI per vlan or ESI on IFD.

Leaf1 and leaf2 are running Active Active mode of EVPN-VXLAN.

CE is acting as bridge configured with vlans

Depends up on the test multicast traffic/host will be emulated by RT

The above configuration will serve as base configuration for all the test cases.

3. Test Cases

The following tests are conducted to measure the learning rate,leave rate,leave latency of IGMP messages which propagates in leaf and spine.

3.1. How long it takes to learn (X1...Xn) IGMP join messages in DUT

Objective:

To Record the time taken to learn X1...Xn igmp join generated by host/hosts.

Topology : Topology 1

Procedure:

Configure "N" evpn-vxlan in leaf1,leaf2,leaf3,spine1 and spine2.Leaf1 and leaf2 are connected to CE which are working in EVPN AA mode. Configure a vlan in RT which is present in leaf1 then send igmp join messages for groups X1... Xn from RT to this vlan present in leaf1. Measure the time taken to learn X1..Xn (*,G) entries in the DUT.

Measurement :

Measure the time taken to learn the X1....Xn groups creating (*,G) entries in the DUT.

Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The time is calculated by averaging the values obtained from "N" samples.

Time taken by DUT to learn and create $X1...Xn$ (*,G) entries in DUT which is measured in sec = $(T1+T2+...Tn/N)$

3.2. How long it takes to clear the (*,G) entries in the DUT

Objective:

To Record the time taken to clear the $X1... Xn$ (*,G) entries in DUT.

Topology : Topology 1

Procedure:

Configure "N" evpn-vxlan in leaf1,leaf2,leaf3,spine1 and spine2. Leaf1 and leaf2 are connected to CE which are working in EVPN AA mode. Configure a vlan in RT which is present in leaf1, then send igmp join messages for groups ranging from $X1...Xn$ from RT to this vlan present in leaf1 Then stop these igmp join messages from RT.

Measurement :

Measure the time taken to flush these $X1...Xn$ (*,G) entries in DUT.

Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The time is calculated by averaging the values obtained from "N" samples.

Time taken by DUT to flush these $X1...Xn$ (*,G) entries in sec = $(T1+T2+...Tn/N)$

3.3. How long it takes the DUT to stop forwarding the traffic(Measuring the leave latency)

Objective:

To Record the time taken by the DUT to stop forwarding the multicast traffic during the receipt of IGMP leave from RT.

Topology : Topology 1

Procedure:

Configure "N" evpn-vxlan in leaf1,leaf2,leaf3,spine1 and spine2. Leaf1 and leaf2 are connected to CE which are working in EVPN AA

mode. Configure a vlan in RT which is present in leaf1, then send igmp join from RT for this vlan to leaf1 for groups ranging from "X1...Xn". Then send traffic to these groups from spine1. Traffic flows from spine1 to leaf1. Send IGMP leave messages for these groups from RT to leaf1. Measure the time taken by the DUT to stop these multicast traffic to RT.

Measurement :

Measure the time taken by DUT to stop the multicast traffic flowing towards RT.

Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The time is calculated by averaging the values obtained from "N" samples.

Time taken by DUT to stop the traffic towards RT connected in leaf1 in sec = $(T1+T2+..Tn/N)$

3.4. How long it takes to learn (X1...Xn) IGMP join messages for N vlans in DUT

Objective:

To Record the time taken to learn X1...Xn IGMP join generated by host/hosts located in N vlans.

Topology : Topology 1

Procedure:

Configure "N" evpn-vxlan in leaf1,leaf2,leaf3,spine1 and spine2. Leaf1 and leaf2 are connected to CE which are working in EVPN AA mode. Configure N vlans in RT, these vlans must be present in leaf1, then send igmp join messages for the groups ranging from X1...Xn for these N vlans from RT. Measure the time taken to learn these X1..Xn (*,G) entries in the DUT for N vlans.

Measurement :

Measure the time taken to learn the X1....Xn groups creating (*,G) entries in the DUT for N vlans.

Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The time is calculated by averaging the values obtained from "N" samples.

Time taken by DUT to learn and create $X1...Xn$ (*,G) entries for N vlans in DUT which is measured in $sec = (T1+T2+...Tn/N)$

3.5. How long it takes to clear the (*,G) entries in the DUT for N vlans

Objective:

To Record the time taken to clear the $X1... Xn$ (*,G) entries in DUT for N vlans.

Topology : Topology 1

Procedure:

Configure "N" evpn-vxlan in leaf1,leaf2,leaf3,spine1 and spine2. Leaf1 and leaf2 are connected to CE which are working in AA mode. Configure N vlans in RT, these vlans must be present in leaf1, then send igmp join messages for groups ranging from $X1...Xn$ for these N vlans from RT. Then stop these IGMP messages.

Measurement :

Measure the time taken to flush these $X1...Xn$ (*,G) entries in DUT.

Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The time is calculated by averaging the values obtained from "N" samples.

Time taken by DUT to flush these $X1...Xn$ (*,G) entries in $sec = (T1+T2+...Tn/N)$

3.6. How long it takes the DUT to stop forwarding the traffic for N vlans (Measuring the leave latency)

Objective:

To Record the time taken by the DUT to stop forwarding the multicast traffic to N vlans during the receipt of IGMP leave messages from RT.

Topology : Topology 1

Procedure:

Configure "N" evpn-vxlan in leaf1,leaf2,leaf3,spine1 and spine2. Leaf1 and leaf2 are connected to CE which are working in EVPN AA mode. Configure N vlans in RT which are present in leaf1, then send igmp join messages from RT for groups ranging from $X1...Xn$ to these vlans

present in leaf1. Then send traffic to these groups from spine1. Traffic flows from spine1 to leaf1. Send the IGMP leave messages for these groups in all vlans. Measure the time taken by the DUT to stop the traffic for these group flowing towards RT.

Measurement :

Measure the time taken by DUT to stop the multicast traffic flowing towards RT.

Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The time is calculated by averaging the values obtained from "N" samples.

Time taken by DUT to stop the traffic towards RT in sec = $(T1+T2+..Tn/N)$

3.7. How long it takes to learn (X1...Xn) IGMP join messages for N vlans in DUT working EVPN AA mode

Objective:

To Record the time taken to learn X1...Xn IGMP join generated by host/hosts located in N vlans in DUT operating in EVPN AA mode.

Topology : Topology 1

Procedure:

Configure "N" evpn-vxlan in leaf1,leaf2,leaf3,spine1 and spine2. Leaf1 and leaf2 are connected to CE which are working in EVPN AA mode. Configure N vlans in RT, these vlans must be present in leaf1,leaf2, then send igmp join messages for the groups ranging from X1...Xn for these N vlans from RT to CE connected to leaf1 and leaf2 working EVPN AA mode. Measure the time taken to learn these X1..Xn (*,G) entries in the DUT for N vlans.

Measurement :

Measure the time taken to learn the X1....Xn groups by creating (*,G) entries in the DUT for N vlans.

Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The time is calculated by averaging the values obtained from "N" samples.

Time taken by DUT to learn and create X1...Xn (*,G) entries for N vlans which is measured in sec = $(T1+T2+..Tn/N)$

3.8. How long it takes to clear the (*,G) entries for N vlans in DUT working EVPN AA

Objective:

To Record the time taken to clear the X1... Xn (*,G) entries in DUT for N vlans.

Topology : Topology 1

Procedure:

Configure "N" evpn-vxlan in leaf1,leaf2,leaf3,spine1 and spine2. Leaf1 and leaf2 are connected to CE which are working in AA mode. Configure N vlans in RT, these vlans must be present in leaf1, then send igmp join messages for groups ranging from X1...Xn for these N vlans from RT to CE which is connected to leaf1 and leaf2 working in EVPN AA mode. Then stop these IGMP messages.

Measurement :

Measure the time taken to flush these X1...Xn (*,G) entries in DUT.

Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The time is calculated by averaging the values obtained from "N" samples.

Time taken by DUT to flush these X1...Xn (*,G) entries in sec = $(T1+T2+..Tn/N)$

3.9. How long it takes the DUT operating in EVPN AA to stop forwarding the traffic for N vlans (Measuring the leave latency)

Objective:

To Record the time taken by the DUT to stop forwarding the multicast traffic to N vlans during the receipt of IGMP leave messages from RT.

Topology : Topology 1

Procedure:

Configure "N" evpn-vxlan in leaf1,leaf2,leaf3,spine1 and spine2. Leaf1 and leaf2 are connected to CE which are working in EVPN AA mode. Configure N vlans in RT which are present in leaf1, then send igmp join messages from RT connected to CE for groups ranging from X1...Xn to these vlans. The CE in turn forwards the igmp messages to leaf1 and leaf2 operating in EVPN AA mode. Then send traffic to these

groups from spine1. Traffic flows from spine1 to CE. Send the IGMP leave messages for these groups in all vlans from RT connected to CE. Measure the time taken by the DUT to stop the traffic for these group flowing towards RT.

Measurement :

Measure the time taken by DUT to stop the multicast traffic flowing towards RT.

Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The time is calculated by averaging the values obtained from "N" samples.

Time taken by DUT to stop the traffic towards RT in sec = $(T1+T2+..Tn/N)$

3.10. How long does it take the DUT in EVPN AA to handle Join Timeout and stop forwarding

Objective:

To record the time takes for handling of Type-7 withdrawal and clearing the state and stop forwarding the traffic.

Topology : Topology 1

Procedure:

Configure "N" evpn-vxlan in leaf1,leaf2,leaf3,spine1 and spine2. Leaf1 and leaf2 are connected to CE which are working in EVPN AA mode. Configure N vlans in RT which are present in leaf1, then send igmp join messages from RT connected to CE for groups ranging from X1...Xn to these vlans. The CE in turn forwards the igmp messages to leaf1 and leaf2 operating in EVPN AA mode. Then send traffic to these groups from spine1. Traffic flows from spine1 to CE. Send the IGMP leave messages for these groups in all vlans from RT connected to CE. The iGMP leave must reach the leaf1. It will send type 7 withdrawal to DUT working in EVPN AA. Measure the time taken by the DUT to stop the traffic flowing to CE. This time will give the leave latency due to type 7 withdrawal.

Measurement :

Measure the time taken by DUT to stop the multicast traffic flowing towards RT.

Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The time is calculated by averaging the values obtained from "N" samples.

Time taken by DUT to stop the traffic towards RT in sec =
 $(T1+T2+..Tn/N)$

3.11. How long does it take an Ingress to learn a remote Type-6 join, create state and forwarding

Objective:

To record the time takes for forwarding the traffic by DUT after the receipt of type 6 join from peer MHPE in same ESI.

Topology : Topology 1

Procedure:

Configure "N" evpn-vxlan in leaf1,leaf2,leaf3,spine1 and spine2. Leaf1 and leaf2 are connected to CE which are working in EVPN AA mode. Configure N vlans in RT which are present in leaf1, then send igmp join messages from RT connected to CE for groups ranging from X1...Xn to these vlans. The CE in turn forwards the igmp messages to leaf2 operating in EVPN AA mode. leaf2 and leaf1 are working EVPN AA mode. Leaf 2 will send the type 6 join to the DUT(leaf 1). Then send traffic to these groups from spine1. Traffic flows from spine1 to CE. Measure the time taken by DUT to forward the traffic after the receipt of type 6 join from leaf1.

Measurement :

Measure the time taken by DUT to forward the multicast traffic flowing towards RT.

Repeat these test and plot the data. The test is repeated for "N" times and the values are collected. The time is calculated by averaging the values obtained from "N" samples.

Time taken by DUT to forward the traffic towards RT in sec =
 $(T1+T2+..Tn/N)$

4. Link Flap

4.1. To Measure the multicast packet loss in EVPN AA scenario on a CE link failure

Objective:

To measure the packet loss during the CE to DF link failure.

Topology : Topology 1

Procedure:

Configure "N" evpn-vxlan in leaf1,leaf2,leaf3,spine1 and spine2. Leaf1 and leaf2 are connected to CE which are working in EVPN AA mode. Configure N vlans in RT which are present in leaf1, then send igmp join messages from RT connected to CE for groups ranging from X1...Xn to these vlans. The CE in turn forwards the igmp messages to leaf1 and leaf2 operating in EVPN AA mode. Then send traffic to these groups from spine1. Traffic flows from spine1 to CE. Fail the DF-CE link. The NON DF now will act as DF and start forwarding the multicast traffic.

Measurement :

Measure the multicast packet loss during the link failure. Repeat the test "N" times and plot the data. The packet loss is calculated by averaging the values obtained from "N" samples.

Packet loss in sec = $(T1+T2+...Tn/N)$

4.2. To Measure the multicast packet loss in EVPN AA scenario on a core failure

Objective:

To measure the packet loss during the DF core failure

Topology : Topology 1

Procedure:

Configure "N" evpn-vxlan in leaf1,leaf2,leaf3,spine1 and spine2. Leaf1 and leaf2 are connected to CE which are working in EVPN AA mode. Configure N vlans in RT which are present in leaf1, then send igmp join messages from RT connected to CE for groups ranging from X1...Xn to these vlans. The CE in turn forwards the igmp messages to leaf1 and leaf2 operating in EVPN AA mode. Then send traffic to these

groups from spine1. Traffic flows from spine1 to CE. Fail the DF core link. The NON DF now will act as the DF and starts forwarding the multicast traffic.

Measurement :

Measure the multicast packet loss during the link failure. Repeat the test "N" times and plot the data. The packet loss is calculated by averaging the values obtained from "N" samples.

Packet loss in sec = $(T1+T2+..Tn/N)$

4.3. To Measure the multicast packet loss in EVPN AA scenario on a routing failure

Objective:

To measure the packet loss during the DF routing failure

Topology : Topology 1

Procedure:

Configure "N" evpn-vxlan in leaf1, leaf2, leaf3, spine1 and spine2. Leaf1 and leaf2 are connected to CE which are working in EVPN AA mode. Configure N vlans in RT which are present in leaf1, then send igmp join messages from RT connected to CE for groups ranging from X1...Xn to these vlans. The CE in turn forwards the igmp messages to leaf1 and leaf2 operating in EVPN AA mode. Then send traffic to these groups from spine1. Traffic flows from spine1 to CE. Fail the DF by restart routing. The NON DF now will act as the DF and starts forwarding the multicast traffic.

Measurement :

Measure the multicast packet loss during the link failure. Repeat the test "N" times and plot the data. The packet loss is calculated by averaging the values obtained from "N" samples.

Packet loss in sec = $(T1+T2+..Tn/N)$

5. Scale Convergence

5.1. To measure the packet loss during the core link failure.

Objective:

To Measure the convergence at a higher number of vlans and igmp joins.

Topology : Topology 1

Procedure:

Configure "N" evpn-vxlan in leaf1,leaf2,leaf3,spine1 and spine2. Leaf1 and leaf2 are connected to CE which are working in EVPN AA mode. Configure N vlans in RT which are present in leaf1, then send igmp join messages from RT connected to CE for groups ranging from X1...Xn to these vlans. The CE in turn forwards the igmp messages to leaf1 and leaf2 operating in EVPN AA mode. Then send traffic to these groups from spine1. Traffic flows from spine1 to CE. Fail the core link of DF. The NON DF now will act as DF and start forwarding the multicast traffic. The vlans and the multicast groups must be a higher value of N taken at random.

Measurement :

Measure the packet loss in seconds once the core link is restored. Repeat the test "N" times and plot the data. The packet loss is calculated by averaging the values obtained from "N" samples.

Packet loss in sec = $(T1+T2+..Tn/N)$

6. High Availability

6.1. To Record the whether there is traffic loss due to routing engine failover for redundancy test.

Objective:

To record traffic loss during routing engine failover.

Topology : Topology 3

Procedure:

Configure "N" evpn-vxlan in leaf1,leaf2,leaf3,spine1 and spine2. Leaf1 and leaf2 are connected to CE which are working in EVPN AA mode. Configure N vlans in RT which are present in leaf1, then send igmp

join messages from RT connected to CE for groups ranging from X1...Xn to these vlans. The CE in turn forwards the igmp messages to leaf1 and leaf2 operating in EVPN AA mode. Then send traffic to these groups from spine1. Traffic flows from spine1 to CE. Then perform a routing engine failure.

Measurement :

There should be 0 traffic loss which is the ideal case, No change in the DF role. DUT should not withdraw any routes. Repeat the test "N" times and plot the data. The packet loss is calculated by averaging the values obtained from "N" samples.

Packet loss in sec = $(T1+T2+...Tn/N)$

7. SOAK Test

This is measuring the performance of DUT running with scaled configuration with traffic over a period of time "T' ". In each interval "t1" the parameters measured are CPU usage, memory usage, crashes.

7.1. To Measure the stability of the DUT with scale and traffic.

Objective:

To measure the stability of the DUT in a scaled environment with traffic.

Topology : Topology 3

Procedure:

Configure "N" evpn-vxlan in leaf1, leaf2, leaf3, spine1 and spine2. Leaf1 and leaf2 are connected to CE which are working in EVPN AA mode. Configure N vlans in RT which are present in leaf1, then send igmp join messages from RT connected to CE for groups ranging from X1...Xn to these vlans. The CE in turn forwards the igmp messages to leaf1 and leaf2 operating in EVPN AA mode. Then send traffic to these groups from spine1. Traffic flows from spine1 to CE.

Measurement :

Take the hourly reading of CPU, process memory. There should not be any leak, crashes, CPU spikes.

8. Acknowledgements

We would like to thank Al and Sarah for the support.

9. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

There is no additional consideration from RFC 6192.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [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>>.
- [RFC2899] Ginoza, S., "Request for Comments Summary RFC Numbers 2800-2899", RFC 2899, DOI 10.17487/RFC2899, May 2001, <<https://www.rfc-editor.org/info/rfc2899>>.

11.2. Informative References

- [RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <<https://www.rfc-editor.org/info/rfc7432>>.

Appendix A. Appendix

Authors' Addresses

Sudhin Jacob (editor)
Juniper Networks
Bangalore, Karnataka 560103
India

Phone: +91 8061212543
Email: sjacob@juniper.net

Vikram Nagarajan
Juniper Networks
Bangalore, Karnataka 560103
India

Phone: +91 8061212543
Email: vikramna@juniper.net

Benchmarking Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2020

M. Konstantynowicz, Ed.
V. Polak, Ed.
Cisco Systems
July 08, 2019

Probabilistic Loss Ratio Search for Packet Throughput (PLRsearch)
draft-vpolak-bmwg-plrsearch-02

Abstract

This document addresses challenges while applying methodologies described in [RFC2544] to benchmarking software based NFV (Network Function Virtualization) data planes over an extended period of time, sometimes referred to as "soak testing". Packet throughput search approach proposed by this document assumes that system under test is probabilistic in nature, and not deterministic.

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 January 9, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Motivation	3
2. Relation To RFC2544	4
3. Terms And Assumptions	4
3.1. Device Under Test	4
3.2. System Under Test	4
3.3. SUT Configuration	4
3.4. SUT Setup	4
3.5. Network Traffic	5
3.6. Packet	5
3.6.1. Packet Offered	5
3.6.2. Packet Received	5
3.6.3. Packet Lost	5
3.6.4. Other Packets	5
3.7. Traffic Profile	6
3.8. Traffic Generator	6
3.9. Offered Load	6
3.10. Trial Measurement	6
3.11. Trial Duration	7
3.12. Packet Loss	7
3.12.1. Loss Count	7
3.12.2. Loss Rate	7
3.12.3. Loss Ratio	7
3.13. Trial Order Independent System	7
3.14. Trial Measurement Result Distribution	8
3.15. Average Loss Ratio	8
3.16. Duration Independent System	8
3.17. Load Regions	9
3.17.1. Zero Loss Region	9
3.17.2. Guaranteed Loss Region	9
3.17.3. Non-Deterministic Region	9
3.17.4. Normal Region Ordering	9
3.18. Deterministic System	10
3.19. Throughput	10
3.20. Deterministic Search	10
3.21. Probabilistic Search	10
3.22. Loss Ratio Function	11
3.23. Target Loss Ratio	11
3.24. Critical Load	11
3.25. Critical Load Estimate	11
3.26. Fitting Function	11
3.27. Shape of Fitting Function	11
3.28. Parameter Space	12
4. Abstract Algorithm	12

4.1.	High level description	12
4.2.	Main Ideas	12
4.2.1.	Trial Durations	13
4.2.2.	Target Loss Ratio	13
4.3.	PLRsearch Building Blocks	13
4.3.1.	Bayesian Inference	13
4.3.2.	Iterative Search	14
4.3.3.	Fitting Functions	14
4.3.4.	Measurement Impact	14
4.3.5.	Fitting Function Coefficients Distribution	15
4.3.6.	Exit Condition	15
4.3.7.	Integration	15
4.3.8.	Optimizations	15
4.3.9.	Offered Load Selection	16
4.3.10.	Trend Analysis	16
5.	Sample Implementation Specifics: FD.io CSIT	16
5.1.	Measurement Delay	16
5.2.	Rounding Errors and Underflows	17
5.3.	Fitting Functions	17
5.3.1.	Stretch Function	18
5.3.2.	Erf Function	18
5.4.	Prior Distributions	19
5.5.	Integrator	19
5.6.	Offered Load Selection	20
6.	IANA Considerations	20
7.	Security Considerations	20
8.	Acknowledgements	21
9.	References	21
9.1.	Normative References	21
9.2.	Informative References	21
	Authors' Addresses	21

1. Motivation

Network providers are interested in throughput a system can sustain.

[RFC2544] assumes loss ratio is given by a deterministic function of offered load. But NFV software systems are not deterministic enough. This makes deterministic algorithms (such as Binary Search per [RFC2544] and [draft-vpolak-mkonstan-bmwg-mlrsearch] with single trial) to return results, which when repeated show relatively high standard deviation, thus making it harder to tell what "the throughput" actually is.

We need another algorithm, which takes this indeterminism into account.

2. Relation To RFC2544

The aim of this document is to become an extension of [RFC2544] suitable for benchmarking networking setups such as software based NFV systems.

3. Terms And Assumptions

3.1. Device Under Test

In software networking, "device" denotes a specific piece of software tasked with packet processing. Such device is surrounded with other software components (such as operating system kernel). It is not possible to run devices without also running the other components, and hardware resources are shared between both.

For purposes of testing, the whole set of hardware and software components is called "system under test" (SUT). As SUT is the part of the whole test setup performance of which can be measured by [RFC2544] methods, this document uses SUT instead of [RFC2544] DUT.

Device under test (DUT) can be re-introduced when analysing test results using whitebox techniques, but that is outside the scope of this document.

3.2. System Under Test

System under test (SUT) is a part of the whole test setup whose performance is to be benchmarked. The complete methodology contains other parts, whose performance is either already established, or not affecting the benchmarking result.

3.3. SUT Configuration

Usually, system under test allows different configurations, affecting its performance. The rest of this document assumes a single configuration has been chosen.

3.4. SUT Setup

Similarly to [RFC2544], it is assumed that the system under test has been updated with all the packet forwarding information it needs, before the trial measurements (see below) start.

3.5. Network Traffic

Network traffic is a type of interaction between system under test and the rest of the system (traffic generator), used to gather information about the system under test performance. PLRsearch is applicable only to areas where network traffic consists of packets.

3.6. Packet

Unit of interaction between traffic generator and the system under test. Term "packet" is used also as an abstraction of Ethernet frames.

3.6.1. Packet Offered

Packet can be offered, which means it is sent from traffic generator to the system under test.

Each offered packet is assumed to become received or lost in a short time.

3.6.2. Packet Received

Packet can be received, which means the traffic generator verifies it has been processed. Typically, when it is successfully sent from the system under test to traffic generator.

It is assumed that each received packet has been caused by an offered packet, so the number of packets received cannot be larger than the number of packets offered.

3.6.3. Packet Lost

Packet can be lost, which means sent but not received in a timely manner.

It is assumed that each lost packet has been caused by an offered packet, so the number of packets lost cannot be larger than the number of packets offered.

Usually, the number of packets lost is computed as the number of packets offered, minus the number of packets received.

3.6.4. Other Packets

PLRsearch is not considering other packet behaviors known from networking (duplicated, reordered, greatly delayed), assuming the

test specification reclassifies those behaviors to fit into the first three categories.

3.7. Traffic Profile

Usually, the performance of the system under test depends on a "type" of a particular packet (for example size), and "composition" if the network traffic consists of a mixture of different packet types.

Also, some systems under test contain multiple "ports" packets can be offered to and received from.

All such qualities together (but not including properties of trial measurements) are called traffic profile.

Similarly to system under test configuration, this document assumes only one traffic profile has been chosen for a particular test.

3.8. Traffic Generator

Traffic generator is the part of the whole test setup, distinct from the system under test, responsible both for offering packets in a highly predictable manner (so the number of packets offered is known), and for counting received packets in a precise enough way (to distinguish lost packets from tolerably delayed packets).

Traffic generator must offer only packets compatible with the traffic profile, and only count similarly compatible packets as received.

Criteria defining which received packets are compatible are left for test specification to decide.

3.9. Offered Load

Offered load is an aggregate rate (measured in packets per second) of network traffic offered to the system under test, the rate is kept constant for the duration of trial measurement.

3.10. Trial Measurement

Trial measurement is a process of stressing (previously setup) system under test by offering traffic of a particular offered load, for a particular duration.

After that, the system has a short time to become idle, while the traffic generator decides how many packets were lost.

After that, another trial measurement (possibly with different offered load and duration) can be immediately performed. Traffic generator should ignore received packets caused by packets offered in previous trial measurements.

3.11. Trial Duration

Duration for which the traffic generator was offering packets at constant offered load.

In theory, care has to be taken to ensure the offered load and trial duration predict integer number of packets to offer, and that the traffic generator really sends appropriate number of packets within precisely enough timed duration. In practice, such consideration do not change PLRsearch result in any significant way.

3.12. Packet Loss

Packet loss is any quantity describing a result of trial measurement.

It can be loss count, loss rate or loss ratio. Packet loss is zero (or non-zero) if either of the three quantities are zero (or non-zero, respectively).

3.12.1. Loss Count

Number of packets lost (or delayed too much) at a trial measurement by the system under test as determined by packet generator. Measured in packets.

3.12.2. Loss Rate

Loss rate is computed as loss count divided by trial duration. Measured in packets per second.

3.12.3. Loss Ratio

Loss ratio is computed as loss count divided by number of packets offered. Measured as a real (in practice rational) number between zero or one (including).

3.13. Trial Order Independent System

Trial order independent system is a system under test, proven (or just assumed) to produce trial measurement results that display trial order independence.

That means when a pair of consequent trial measurements are performed, the probability to observe a pair of specific results is the same, as the probability to observe the reversed pair of results when performing the reversed pair of consequent measurements.

PLRsearch assumes the system under test is trial order independent.

In practice, most system under test are not entirely trial order independent, but it is not easy to devise an algorithm taking that into account.

3.14. Trial Measurement Result Distribution

When a trial order independent system is subjected to repeated trial measurements of constant duration and offered load, Law of Large Numbers implies the observed loss count frequencies will converge to a specific probability distribution over possible loss counts.

This probability distribution is called trial measurement result distribution, and it depends on all properties fixed when defining it. That includes the system under test, its chosen configuration, the chosen traffic profile, the offered load and the trial duration.

As the system is trial order independent, trial measurement result distribution does not depend on results of few initial trial measurements, of any offered load or (finite) duration.

3.15. Average Loss Ratio

Probability distribution over some (finite) set of states enables computation of probability-weighted average of any quantity evaluated on the states (called the expected value of the quantity).

Average loss ratio is simply the expected value of loss ratio for a given trial measurement result distribution.

3.16. Duration Independent System

Duration independent system is a trial order independent system, whose trial measurement result distribution is proven (or just assumed) to display practical independence from trial duration. See definition of trial duration for discussion on practical versus theoretical.

The only requirement is for average loss ratio to be independent of trial duration.

In theory, that would necessitate each trial measurement result distribution to be a binomial distribution. In practice, more distributions are allowed.

PLRsearch assumes the system under test is duration independent, at least for trial durations typically chosen for trial measurements initiated by PLRsearch.

3.17. Load Regions

For a duration independent system, trial measurement result distribution depends only on offered load.

It is convenient to name some areas of offered load space by possible trial results.

3.17.1. Zero Loss Region

A particular offered load value is said to belong to zero loss region, if the probability of seeing non-zero loss trial measurement result is exactly zero, or at least practically indistinguishable from zero.

3.17.2. Guaranteed Loss Region

A particular offered load value is said to belong to guaranteed loss region, if the probability of seeing zero loss trial measurement result (for non-negligible count of packets offered) is exactly zero, or at least practically indistinguishable from zero.

3.17.3. Non-Deterministic Region

A particular offered load value is said to belong to non-deterministic region, if the probability of seeing zero loss trial measurement result (for non-negligible count of packets offered) is practically distinguishable from both zero and one.

3.17.4. Normal Region Ordering

Although theoretically the three regions can be arbitrary sets, this document assumes they are intervals, where zero loss region contains values smaller than non-deterministic region, which in turn contains values smaller than guaranteed loss region.

3.18. Deterministic System

A hypothetical duration independent system with normal region ordering, whose non-deterministic region is extremely narrow (only present due to "practical distinguishability" and cases when the expected number of packets offered is not an integer).

A duration independent system which is not deterministic is called non-deterministic system.

3.19. Throughput

Throughput is the highest offered load provably causing zero packet loss for trial measurements of duration at least 60 seconds.

For duration independent systems with normal region ordering, the throughput is the highest value within the zero loss region.

3.20. Deterministic Search

Any algorithm that assumes each measurement is a proof of the offered load belonging to zero loss region (or not) is called deterministic search.

This definition includes algorithms based on "composite measurements" which perform multiple trial measurements, somehow re-classifying results pointing at non-deterministic region.

Binary Search is an example of deterministic search.

Single run of a deterministic search launched against a deterministic system is guaranteed to find the throughput with any prescribed precision (not better than non-deterministic region width).

Multiple runs of a deterministic search launched against a non-deterministic system can return varied results within non-deterministic region. The exact distribution of deterministic search results depends on the algorithm used.

3.21. Probabilistic Search

Any algorithm which performs probabilistic computations based on observed results of trial measurements, and which does not assume that non-deterministic region is practically absent, is called probabilistic search.

A probabilistic search algorithm, which would assume that non-deterministic region is practically absent, does not really need to

perform probabilistic computations, so it would become a deterministic search.

While probabilistic search for estimating throughput is possible, it would need a careful model for boundary between zero loss region and non-deterministic region, and it would need a lot of measurements of almost surely zero loss to reach good precision.

3.22. Loss Ratio Function

For any duration independent system, the average loss ratio depends only on offered load (for a particular test setup).

Loss ratio function is the name used for the function mapping offered load to average loss ratio.

This function is initially unknown.

3.23. Target Loss Ratio

Input parameter of PLRsearch. The average loss ratio the output of PLRsearch aims to achieve.

3.24. Critical Load

Aggregate rate of network traffic, which would lead to average loss ratio exactly matching target loss ratio, if used as the offered load for infinite many trial measurement.

3.25. Critical Load Estimate

Any quantitative description of the possible critical load PLRsearch is able to give after observing finite amount of trial measurements.

3.26. Fitting Function

Any function PLRsearch uses internally instead of the unknown loss ratio function. Typically chosen from small set of formulas (shapes) with few parameters to tweak.

3.27. Shape of Fitting Function

Any formula with few undetermined parameters.

3.28. Parameter Space

A subset of Real Coordinate Space. A point of parameter space is a vector of real numbers. Fitting function is defined by shape (a formula with parameters) and point of parameter space (specifying values for the parameters).

4. Abstract Algorithm

4.1. High level description

PLRsearch accepts some input arguments, then iteratively performs trial measurements at varying offered loads (and durations), and returns some estimates of critical load.

PLRsearch input arguments form three groups.

First group has a single argument: `measurer`. This is a callback (function) accepting offered load and duration, and returning the measured loss count.

Second group consists of load related arguments required for `measurer` to work correctly, typically minimal and maximal load to offer. Also, target loss ratio (if not hardcoded) is a required argument.

Third group consists of time related arguments. Typically the duration for the first trial measurement, duration increment per subsequent trial measurement, and total time for search. Some PLRsearch implementation may use estimation accuracy parameters as an exit condition instead of total search time.

The returned quantities should describe the final (or best) estimate of critical load. Implementers can choose any description that suits their users, typically it is average and standard deviation, or lower and upper boundary.

4.2. Main Ideas

The search tries to perform measurements at offered load close to the critical load, because measurement results at offered loads far from the critical load give less information on precise location of the critical load. As virtually every trial measurement result alters the estimate of the critical load, offered loads vary as they approach the critical load.

The only quantity of trial measurement result affecting the computation is loss count. No latency (or other information) is taken into account.

PLRsearch uses Bayesian Inference, computed using numerical integration, which takes long time to get reliable enough results. Therefore it takes some time before the most recent measurement result starts affecting subsequent offered loads and critical rate estimates.

During the search, PLRsearch spawns few processes that perform numerical computations, the main process is calling the measurer to perform trial measurements, without any significant delays between them. The durations of the trial measurements are increasing linearly, as higher number of trial measurement results take longer to process.

4.2.1. Trial Durations

[RFC2544] motivates the usage of at least 60 second duration by the idea of the system under test slowly running out of resources (such as memory buffers).

Practical results when measuring NFV software systems show that relative change of trial duration has negligible effects on average loss ratio, compared to relative change in offered load.

While the standard deviation of loss ratio usually shows some effects of trial duration, they are hard to model. So PLRsearch assumes SUT is duration independent, and chooses trial durations only based on numeric integration requirements.

4.2.2. Target Loss Ratio

(TODO: Link to why we think $1e-7$ is acceptable loss ratio.)

4.3. PLRsearch Building Blocks

Here we define notions used by PLRsearch which are not applicable to other search methods, nor probabilistic systems under test in general.

4.3.1. Bayesian Inference

PLRsearch uses a fixed set of fitting function shapes, and uses Bayesian inference to track posterior distribution on each fitting function parameter space.

Specifically, the few parameters describing a fitting function become the model space. Given a prior over the model space, and trial duration results, a posterior distribution is computed, together with quantities describing the critical load estimate.

Likelihood of a particular loss count is computed using Poisson distribution of average loss rate given by the fitting function (at specific point of parameter space).

Side note: Binomial Distribution is a better fit compared to Poisson distribution (acknowledging that the number of packets lost cannot be higher than the number of packets offered), but the difference tends to be relevant only in high loss region. Using Poisson distribution lowers the impact of measurements in high loss region, thus helping the algorithm to converge towards critical load faster.

4.3.2. Iterative Search

The idea PLRsearch is to iterate trial measurements, using Bayesian inference to compute both the current estimate of the critical load and the next offered load to measure at.

The required numerical computations are done in parallel with the trial measurements.

This means the result of measurement "n" comes as an (additional) input to the computation running in parallel with measurement "n+1", and the outputs of the computation are used for determining the offered load for measurement "n+2".

Other schemes are possible, aimed to increase the number of measurements (by decreasing their duration), which would have even higher number of measurements run before a result of a measurement affects offered load.

4.3.3. Fitting Functions

To make the space of possible loss ratio functions more tractable the algorithm uses only few fitting function shapes for its predictions. As the search algorithm needs to evaluate the function also far away from the critical load, the fitting function have to be reasonably behaved for every positive offered load, specifically cannot cannot predict non-positive packet loss ratio.

4.3.4. Measurement Impact

Results from trials far from the critical load are likely to affect the critical load estimate negatively, as the fitting functions do not need to be good approximations there. This is true mainly for guaranteed loss region, as in zero loss region even badly behaved fitting function predicts loss count to be "almost zero", so seeing a measurement confirming the loss has been zero indeed has small impact.

Discarding some results, or "suppressing" their impact with ad-hoc methods (other than using Poisson distribution instead of binomial) is not used, as such methods tend to make the overall search unstable. We rely on most of measurements being done (eventually) near the critical load, and overweighting far-off measurements (eventually) for well-behaved fitting functions.

4.3.5. Fitting Function Coefficients Distribution

To accommodate systems with different behaviours, a fitting function is expected to have few numeric parameters affecting its shape (mainly affecting the linear approximation in the critical region).

The general search algorithm can use whatever increasing fitting function, some specific functions are described later.

It is up to implementer to choose a fitting function and prior distribution of its parameters. The rest of this document assumes each parameter is independently and uniformly distributed over a common interval. Implementers are to add non-linear transformations into their fitting functions if their prior is different.

4.3.6. Exit Condition

Exit condition for the search is either the standard deviation of the critical load estimate becoming small enough (or similar), or overall search time becoming long enough.

The algorithm should report both average and standard deviation for its critical load posterior.

4.3.7. Integration

The posterior distributions for fitting function parameters are not be integrable in general.

The search algorithm utilises the fact that trial measurement takes some time, so this time can be used for numeric integration (using suitable method, such as Monte Carlo) to achieve sufficient precision.

4.3.8. Optimizations

After enough trials, the posterior distribution will be concentrated in a narrow area of the parameter space. The integration method should take advantage of that.

Even in the concentrated area, the likelihood can be quite small, so the integration algorithm should avoid underflow errors by some means, for example by tracking the logarithm of the likelihood.

4.3.9. Offered Load Selection

The simplest rule is to set offered load for next trial measurement equal to the current average (both over posterior and over fitting function shapes) of the critical load estimate.

Contrary to critical load estimate computation, heuristic algorithms affecting offered load selection do not introduce instability, and can help with convergence speed.

4.3.10. Trend Analysis

If the reported averages follow a trend (maybe without reaching equilibrium), average and standard deviation COULD refer to the equilibrium estimates based on the trend, not to immediate posterior values.

But such post-processing is discouraged, unless a clear reason for the trend is known. Frequently, presence of such a trend is a sign of some of PLRsearch assumption being violated (usually trial order independence or duration independence).

It is RECOMMENDED to report any trend quantification together with direct critical load estimate, so users can draw their own conclusion. Alternatively, trend analysis may be a part of exit conditions, requiring longer searches for systems displaying trends.

5. Sample Implementation Specifics: FD.io CSIT

The search receives `min_rate` and `max_rate` values, to avoid measurements at offered loads not supported by the traffic generator.

The implemented tests cases use bidirectional traffic. The algorithm stores each rate as bidirectional rate (internally, the algorithm is agnostic to flows and directions, it only cares about overall counts of packets sent and packets lost), but debug output from traffic generator lists unidirectional values.

5.1. Measurement Delay

In a sample implementation in FD.io CSIT project, there is roughly 0.5 second delay between trials due to restrictions imposed by packet traffic generator in use (T-Rex).

As measurements results come in, posterior distribution computation takes more time (per sample), although there is a considerable constant part (mostly for inverting the fitting functions).

Also, the integrator needs a fair amount of samples to reach the region the posterior distribution is concentrated at.

And of course, speed of the integrator depends on computing power of the CPUs the algorithm is able to use.

All those timing related effects are addressed by arithmetically increasing trial durations with configurable coefficients (currently 5.1 seconds for the first trial, each subsequent trial being 0.1 second longer).

5.2. Rounding Errors and Underflows

In order to avoid them, the current implementation tracks natural logarithm (instead of the original quantity) for any quantity which is never negative. Logarithm of zero is minus infinity (not supported by Python), so special value "None" is used instead. Specific functions for frequent operations (such as "logarithm of sum of exponentials") are defined to handle None correctly.

5.3. Fitting Functions

Current implementation uses two fitting functions. In general, their estimates for critical rate differ, which adds a simple source of systematic error, on top of posterior dispersion reported by integrator. Otherwise the reported stdev of critical rate estimate is unrealistically low.

Both functions are not only increasing, but also convex (meaning the rate of increase is also increasing).

As Primitive Function to any positive function is an increasing function, and Primitive Function to any increasing function is convex function; both fitting functions were constructed as double Primitive Function to a positive function (even though the intermediate increasing function is easier to describe).

As not any function is integrable, some more realistic functions (especially with respect to behavior at very small offered loads) are not easily available.

Both fitting functions have a "central point" and a "spread", varied by simply shifting and scaling (in x-axis, the offered load direction) the function to be doubly integrated. Scaling in y-axis

(the loss rate direction) is fixed by the requirement of transfer rate staying nearly constant in very high offered loads.

In both fitting functions (as they are a double Primitive Function to a symmetric function), the "central point" turns out to be equal to the aforementioned limiting transfer rate, so the fitting function parameter is named "mrr", the same quantity CSIT Maximum Receive Rate tests are designed to measure.

Both fitting functions return logarithm of loss rate, to avoid rounding errors and underflows. Parameters and offered load are not given as logarithms, as they are not expected to be extreme, and the formulas are simpler that way.

Both fitting functions have several mathematically equivalent formulas, each can lead to an overflow or underflow in different places. Overflows can be eliminated by using different exact formulas for different argument ranges. Underflows can be avoided by using approximate formulas in affected argument ranges, such ranges have their own formulas to compute. At the end, both fitting function implementations contain multiple "if" branches, discontinuities are a possibility at range boundaries.

5.3.1. Stretch Function

The original function (before applying logarithm) is Primitive Function to Logistic Function. The name "stretch" is used for related a function in context of neural networks with sigmoid activation function.

Formula for stretch fitting function: average loss rate (r) computed from offered load (b), mrr parameter (m) and spread parameter (a), given as InputForm of Wolfram language:

$$r = (a * (1 + E^{(m/a)}) * \text{Log}[(E^{(b/a)} + E^{(m/a)}) / (1 + E^{(m/a)})]) / E^{(m/a)}$$

5.3.2. Erf Function

The original function is double Primitive Function to Gaussian Function. The name "erf" comes from error function, the first primitive to Gaussian.

Formula for erf fitting function: average loss rate (r) computed from offered load (b), mrr parameter (m) and spread parameter (a), given as InputForm of Wolfram language:

$$r = ((a * (E^{(-(b - m)^2/a^2)} - E^{-(m^2/a^2)})) / \text{Sqrt}[Pi] + m * \text{Erfc}[m/a] + (b - m) * \text{Erfc}[(-b + m)/a]) / (1 + \text{Erf}[m/a])$$

5.4. Prior Distributions

The numeric integrator expects all the parameters to be distributed (independently and) uniformly on an interval $(-1, 1)$.

As both "mrr" and "spread" parameters are positive and not dimensionless, a transformation is needed. Dimensionality is inherited from max_rate value.

The "mrr" parameter follows a Lomax Distribution with alpha equal to one, but shifted so that mrr is always greater than 1 packet per second.

The "stretch" parameter is generated simply as the "mrr" value raised to a random power between zero and one; thus it follows a Reciprocal Distribution.

5.5. Integrator

After few measurements, the posterior distribution of fitting function arguments gets quite concentrated into a small area. The integrator is using Monte Carlo with Importance Sampling where the biased distribution is Bivariate Gaussian distribution, with deliberately larger variance. If the generated sample falls outside $(-1, 1)$ interval, another sample is generated.

The center and the covariance matrix for the biased distribution is based on the first and second moments of samples seen so far (within the computation), with the following additional features designed to avoid hyper-focused distributions.

Each computation starts with the biased distribution inherited from the previous computation (zero point and unit covariance matrix is used in the first computation), but the overall weight of the data is set to the weight of the first sample of the computation. Also, the center is set to the first sample point. When additional samples come, their weight (including the importance correction) is compared to the weight of data seen so far (within the computation). If the new sample is more than one e-fold more impactful, both weight values (for data so far and for the new sample) are set to (geometric) average of the two weights. Finally, the actual sample generator uses covariance matrix scaled up by a configurable factor (8.0 by default).

This combination showed the best behavior, as the integrator usually follows two phases. First phase (where inherited biased distribution or single big samples are dominating) is mainly important for locating the new area the posterior distribution is concentrated at.

The second phase (dominated by whole sample population) is actually relevant for the critical rate estimation.

5.6. Offered Load Selection

First two measurements are hardcoded to happen at the middle of rate interval and at max_rate. Next two measurements follow MRR-like logic, offered load is decreased so that it would reach target loss ratio if offered load decrease lead to equal decrease of loss rate.

Basis for offered load for next trial measurements is the integrated average of current critical rate estimate, averaged over fitting function.

There is one workaround implemented, aimed at reducing the number of consequent zero loss measurements. The workaround first stores every measurement result which loss ratio was the targeted loss ratio or higher. Sorted list (called lossy loads) of such results is maintained.

When a sequence of one or more zero loss measurement results is encountered, a smallest of lossy loads is drained from the list. If the estimate average is smaller than the drained value, a weighted average of this estimate and the drained value is used as the next offered load. The weight of the drained value doubles with each additional consecutive zero loss results.

This behavior helps the algorithm with convergence speed, as it does not need so many zero loss result to get near critical load. Using the smallest (not drained yet) of lossy loads makes it sure the new offered load is unlikely to result in big loss region. Draining even if the estimate is large enough helps to discard early measurements when loss hapened at too low offered load. Current implementation adds 4 copies of lossy loads and drains 3 of them, which leads to fairly stable behavior even for somewhat inconsistent SUTs.

6. IANA Considerations

No requests of IANA.

7. Security Considerations

Benchmarking activities as described in this memo are limited to technology characterization of a DUT/SUT using controlled stimuli in a laboratory environment, with dedicated address space and the constraints specified in the sections above.

The benchmarking network topology will be an independent test setup and MUST NOT be connected to devices that may forward the test traffic into a production network or misroute traffic to the test management network.

Further, benchmarking is performed on a "black-box" basis, relying solely on measurements observable external to the DUT/SUT.

Special capabilities SHOULD NOT exist in the DUT/SUT specifically for benchmarking purposes. Any implications for network security arising from the DUT/SUT SHOULD be identical in the lab and in production networks.

8. Acknowledgements

..

9. References

9.1. 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>>.

9.2. Informative References

[draft-vpolak-mkonstan-bmwg-mlrsearch]
"Multiple Loss Ratio Search for Packet Throughput (MLRsearch)", July 2019, <<https://tools.ietf.org/html/draft-vpolak-mkonstan-bmwg-mlrsearch>>.

Authors' Addresses

Maciek Konstantynowicz (editor)
Cisco Systems

Email: mkonstan@cisco.com

Vratko Polak (editor)
Cisco Systems

Email: vrpolak@cisco.com

Benchmarking Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2020

M. Konstantynowicz, Ed.
V. Polak, Ed.
Cisco Systems
July 08, 2019

Multiple Loss Ratio Search for Packet Throughput (MLRsearch)
draft-vpolak-mkonstan-bmwg-mlrsearch-02

Abstract

This document proposes changes to [RFC2544], specifically to packet throughput search methodology, by defining a new search algorithm referred to as Multiple Loss Ratio search (MLRsearch for short). Instead of relying on binary search with pre-set starting offered load, it proposes a novel approach discovering the starting point in the initial phase, and then searching for packet throughput based on defined packet loss ratio (PLR) input criteria and defined final trial duration time. One of the key design principles behind MLRsearch is minimizing the total test duration and searching for multiple packet throughput rates (each with a corresponding PLR) concurrently, instead of doing it sequentially.

The main motivation behind MLRsearch is the new set of challenges and requirements posed by NFV (Network Function Virtualization), specifically software based implementations of NFV data planes. Using [RFC2544] in the experience of the authors yields often not repetitive and not replicable end results due to a large number of factors that are out of scope for this draft. MLRsearch aims to address this challenge and define a common (standard?) way to evaluate NFV packet throughput performance that takes into account varying characteristics of NFV systems under test.

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 January 9, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Terminology	2
2. MLRsearch Background	4
3. MLRsearch Overview	5
4. Sample Implementation	8
4.1. Input Parameters	8
4.2. Initial Phase	9
4.3. Non-Initial Phases	10
4.4. Sample MLRsearch Run	12
5. Known Implementations	12
5.1. FD.io CSIT Implementation Deviations	12
6. IANA Considerations	13
7. Security Considerations	14
8. Acknowledgements	14
9. References	14
9.1. Normative References	14
9.2. Informative References	14
Authors' Addresses	15

1. Terminology

- o Frame size: size of an Ethernet Layer-2 frame on the wire, including any VLAN tags (dot1q, dot1ad) and Ethernet FCS, but excluding Ethernet preamble and inter-frame gap. Measured in bytes.
- o Packet size: same as frame size, both terms used interchangeably.

- o Inner L2 size: for tunneled L2 frames only, size of an encapsulated Ethernet Layer-2 frame, preceded with tunnel header, and followed by tunnel trailer. Measured in Bytes.
- o Inner IP size: for tunneled IP packets only, size of an encapsulated IPv4 or IPv6 packet, preceded with tunnel header, and followed by tunnel trailer. Measured in Bytes.
- o Device Under Test (DUT): In software networking, "device" denotes a specific piece of software tasked with packet processing. Such device is surrounded with other software components (such as operating system kernel). It is not possible to run devices without also running the other components, and hardware resources are shared between both. For purposes of testing, the whole set of hardware and software components is called "system under test" (SUT). As SUT is the part of the whole test setup performance of which can be measured by [RFC2544] methods, this document uses SUT instead of [RFC2544] DUT. Device under test (DUT) can be re-introduced when analysing test results using whitebox techniques, but this document sticks to blackbox testing.
- o System Under Test (SUT): System under test (SUT) is a part of the whole test setup whose performance is to be benchmarked. The complete methodology contains other parts, whose performance is either already established, or not affecting the benchmarking result.
- o Bi-directional throughput tests: involve packets/frames flowing in both transmit and receive directions over every tested interface of SUT/DUT. Packet flow metrics are measured per direction, and can be reported as aggregate for both directions (i.e. throughput) and/or separately for each measured direction (i.e. latency). In most cases bi-directional tests use the same (symmetric) load in both directions.
- o Uni-directional throughput tests: involve packets/frames flowing in only one direction, i.e. either transmit or receive direction, over every tested interface of SUT/DUT. Packet flow metrics are measured and are reported for measured direction.
- o Packet Loss Ratio (PLR): ratio of packets received relative to packets transmitted over the test trial duration, calculated using formula: $PLR = (pkts_transmitted - pkts_received) / pkts_transmitted$. For bi-directional throughput tests aggregate PLR is calculated based on the aggregate number of packets transmitted and received.

- o Packet Throughput Rate: maximum packet offered load DUT/SUT forwards within the specified Packet Loss Ratio (PLR). In many cases the rate depends on the frame size processed by DUT/SUT. Hence packet throughput rate MUST be quoted with specific frame size as received by DUT/SUT during the measurement. For bi-directional tests, packet throughput rate should be reported as aggregate for both directions. Measured in packets-per-second (pps) or frames-per-second (fps), equivalent metrics.
- o Bandwidth Throughput Rate: a secondary metric calculated from packet throughput rate using formula: $bw_rate = pkt_rate * (frame_size + L1_overhead) * 8$, where L1_overhead for Ethernet includes preamble (8 Bytes) and inter-frame gap (12 Bytes). For bi-directional tests, bandwidth throughput rate should be reported as aggregate for both directions. Expressed in bits-per-second (bps).
- o Non Drop Rate (NDR): maximum packet/bandwidth throughput rate sustained by DUT/SUT at PLR equal zero (zero packet loss) specific to tested frame size(s). MUST be quoted with specific packet size as received by DUT/SUT during the measurement. Packet NDR measured in packets-per-second (or fps), bandwidth NDR expressed in bits-per-second (bps).
- o Partial Drop Rate (PDR): maximum packet/bandwidth throughput rate sustained by DUT/SUT at PLR greater than zero (non-zero packet loss) specific to tested frame size(s). MUST be quoted with specific packet size as received by DUT/SUT during the measurement. Packet PDR measured in packets-per-second (or fps), bandwidth PDR expressed in bits-per-second (bps).
- o Maximum Receive Rate (MRR): packet/bandwidth rate regardless of PLR sustained by DUT/SUT under specified Maximum Transmit Rate (MTR) packet load offered by traffic generator. MUST be quoted with both specific packet size and MTR as received by DUT/SUT during the measurement. Packet MRR measured in packets-per-second (or fps), bandwidth MRR expressed in bits-per-second (bps).
- o Trial: a single measurement step.
- o Trial duration: amount of time over which packets are transmitted and received in a single throughput measurement step.

2. MLRsearch Background

Multiple Loss Ratio search (MLRsearch) is a packet throughput search algorithm suitable for deterministic systems (as opposed to probabilistic systems). MLRsearch discovers multiple packet

throughput rates in a single search, with each rate associated with a distinct Packet Loss Ratio (PLR) criteria.

For cases when multiple rates need to be found, this property makes MLRsearch more efficient in terms of time execution, compared to traditional throughput search algorithms that discover a single packet rate per defined search criteria (e.g. a binary search specified by [RFC2544]). MLRsearch reduces execution time even further by relying on shorter trial durations of intermediate steps, with only the final measurements conducted at the specified final trial duration. This results in the shorter overall search execution time when compared to a traditional binary search, while guaranteeing the same results for deterministic systems.

In practice two rates with distinct PLRs are commonly used for packet throughput measurements of NFV systems: Non Drop Rate (NDR) with $PLR=0$ and Partial Drop Rate (PDR) with $PLR>0$. The rest of this document describes MLRsearch for NDR and PDR. If needed, MLRsearch can be easily adapted to discover more throughput rates with different pre-defined PLRs.

Similarly to other throughput search approaches like binary search, MLRsearch is effective for SUTs/DUTs with PLR curve that is continuously flat or increasing with growing offered load. It may not be as effective for SUTs/DUTs with abnormal PLR curves.

MLRsearch relies on traffic generator to qualify the received packet stream as error-free, and invalidate the results if any disqualifying errors are present e.g. out-of-sequence frames.

MLRsearch can be applied to both uni-directional and bi-directional throughput tests.

For bi-directional tests, MLRsearch rates and ratios are aggregates of both directions, based on the following assumptions:

- o Packet rates transmitted by traffic generator and received by SUT/DUT are the same in each direction, in other words the load is symmetric.
- o SUT/DUT packet processing capacity is the same in both directions, resulting in the same packet loss under load.

3. MLRsearch Overview

The main properties of MLRsearch:

- o MLRsearch is a duration aware multi-phase multi-rate search algorithm:
 - * Initial Phase determines promising starting interval for the search.
 - * Intermediate Phases progress towards defined final search criteria.
 - * Final Phase executes measurements according to the final search criteria.
 - * Final search criteria is defined by following inputs:
 - + PLRs associated with NDR and PDR.
 - + Final trial duration.
 - + Measurement resolution.
- o Initial Phase:
 - * Measure MRR over initial trial duration.
 - * Measured MRR is used as an input to the first intermediate phase.
- o Multiple Intermediate Phases:
 - * Trial duration:
 - + Start with initial trial duration in the first intermediate phase.
 - + Converge geometrically towards the final trial duration.
 - * Track two values for NDR and two for PDR:
 - + The values are called lower_bound and upper_bound.
 - + Each value comes from a specific trial measurement:
 - Most recent for that transmit rate.
 - As such the value is associated with that measurement's duration and loss.
 - + A bound can be valid or invalid:

- Valid lower_bound must conform with PLR search criteria.
 - Valid upper_bound must not conform with PLR search criteria.
 - Example of invalid NDR lower_bound is if it has been measured with non-zero loss.
 - Invalid bounds are not real boundaries for the searched value:
 - o They are needed to track interval widths.
 - Valid bounds are real boundaries for the searched value.
 - Each non-initial phase ends with all bounds valid.
 - Bound can become invalid if it re-measured at longer trial duration in sub-sequent phase.
- * Search:
- + Start with a large (lower_bound, upper_bound) interval width, that determines measurement resolution.
 - + Geometrically converge towards the width goal of the phase.
 - + Each phase halves the previous width goal.
- * Use of internal and external searches:
- + External search:
 - Measures at transmit rates outside the (lower_bound, upper_bound) interval.
 - Activated when a bound is invalid, to search for a new valid bound by doubling the interval width.
 - It is a variant of "exponential search".
 - + Internal search:
 - A "binary search" that measures at transmit rates within the (lower_bound, upper_bound) valid interval, halving the interval width.
- o Final Phase:

- * Executed with the final test trial duration, and the final width goal that determines resolution of the overall search.
- o Intermediate Phases together with the Final Phase are called Non-Initial Phases.

The main benefits of MLRsearch vs. binary search include:

- o In general MLRsearch is likely to execute more trials overall, but likely less trials at a set final trial duration.
- o In well behaving cases, e.g. when results do not depend on trial duration, it greatly reduces (>50%) the overall duration compared to a single PDR (or NDR) binary search over duration, while finding multiple drop rates.
- o In all cases MLRsearch yields the same or similar results to binary search.
- o Note: both binary search and MLRsearch are susceptible to reporting non-repeatable results across multiple runs for very bad behaving cases.

Caveats:

- o Worst case MLRsearch can take longer than a binary search e.g. in case of drastic changes in behaviour for trials at varying durations.

4. Sample Implementation

Following is a brief description of a sample MLRsearch implementation based on the open-source code running in FD.io CSIT project as part of a Continuous Integration / Continuous Development (CI/CD) framework.

4.1. Input Parameters

1. **maximum_transmit_rate** - Maximum Transmit Rate (MTR) of packets to be used by external traffic generator implementing MLRsearch, limited by the actual Ethernet link(s) rate, NIC model or traffic generator capabilities. Sample defaults: 2 * 14.88 Mpps for 64B 10GE link rate, 2 * 18.75 Mpps for 64B 40GE NIC (specific model) maximum rate (lower than 2 * 59.52 Mpps 40GE link rate).
2. **minimum_transmit_rate** - minimum packet transmit rate to be used for measurements. MLRsearch fails if lower transmit rate needs

to be used to meet search criteria. Default: 2 * 10 kpps (could be higher).

3. *final_trial_duration* - required trial duration for final rate measurements. Default: 30 sec.
4. *initial_trial_duration* - trial duration for initial MLRsearch phase. Default: 1 sec.
5. *final_relative_width* - required measurement resolution expressed as (lower_bound, upper_bound) interval width relative to upper_bound. Default: 0.5%.
6. *packet_loss_ratio* - maximum acceptable PLR search criteria for PDR measurements. Default: 0.5%.
7. *number_of_intermediate_phases* - number of phases between the initial phase and the final phase. Impacts the overall MLRsearch duration. Less phases are required for well behaving cases, more phases may be needed to reduce the overall search duration for worse behaving cases. Default (2). (Value chosen based on limited experimentation to date. More experimentation needed to arrive to clearer guidelines.)

4.2. Initial Phase

1. First trial measures at configured maximum transmit rate (MTR) and discovers maximum receive rate (MRR).
 - * IN: trial_duration = initial_trial_duration.
 - * IN: offered_transmit_rate = maximum_transmit_rate.
 - * DO: single trial.
 - * OUT: measured loss ratio.
 - * OUT: MRR = measured receive rate.
2. Second trial measures at MRR and discovers MRR2.
 - * IN: trial_duration = initial_trial_duration.
 - * IN: offered_transmit_rate = MRR.
 - * DO: single trial.
 - * OUT: measured loss ratio.

* OUT: MRR2 = measured receive rate.

3. Third trial measures at MRR2.

* IN: trial_duration = initial_trial_duration.

* IN: offered_transmit_rate = MRR2.

* DO: single trial.

* OUT: measured loss ratio.

4.3. Non-Initial Phases

1. Main loop:

1. IN: trial_duration for the current phase. Set to initial_trial_duration for the first intermediate phase; to final_trial_duration for the final phase; or to the element of interpolating geometric sequence for other intermediate phases. For example with two intermediate phases, trial_duration of the second intermediate phase is the geometric average of initial_trial_duration and final_trial_duration.
2. IN: relative_width_goal for the current phase. Set to final_relative_width for the final phase; doubled for each preceding phase. For example with two intermediate phases, the first intermediate phase uses quadruple of final_relative_width and the second intermediate phase uses double of final_relative_width.
3. IN: ndr_interval, pdr_interval from the previous main loop iteration or the previous phase. If the previous phase is the initial phase, both intervals have lower_bound = MRR2, upper_bound = MRR. Note that the initial phase is likely to create intervals with invalid bounds.
4. DO: According to the procedure described in point 2., either exit the phase (by jumping to 1.7.), or calculate new transmit rate to measure with.
5. DO: Perform the trial measurement at the new transmit rate and trial_duration, compute its loss ratio.
6. DO: Update the bounds of both intervals, based on the new measurement. The actual update rules are numerous, as NDR external search can affect PDR interval and vice versa, but

the result agrees with rules of both internal and external search. For example, any new measurement below an invalid `lower_bound` becomes the new `lower_bound`, while the old measurement (previously acting as the invalid `lower_bound`) becomes a new and valid `upper_bound`. Go to next iteration (1.3.), taking the updated intervals as new input.

7. OUT: current `ndr_interval` and `pdr_interval`. In the final phase this is also considered to be the result of the whole search. For other phases, the next phase loop is started with the current results as an input.
2. New transmit rate (or exit) calculation (for point 1.4.):
 1. If there is an invalid bound then prepare for external search:
 - + IF the most recent measurement at NDR `lower_bound` transmit rate had the loss higher than zero, then the new transmit rate is NDR `lower_bound` decreased by two NDR interval widths or the amount needed to hit the current width goal, whichever is larger.
 - + Else, IF the most recent measurement at PDR `lower_bound` transmit rate had the loss higher than PLR, then the new transmit rate is PDR `lower_bound` decreased by two PDR interval widths.
 - + Else, IF the most recent measurement at NDR `upper_bound` transmit rate had no loss, then the new transmit rate is NDR `upper_bound` increased by two NDR interval widths.
 - + Else, IF the most recent measurement at PDR `upper_bound` transmit rate had the loss lower or equal to PLR, then the new transmit rate is PDR `upper_bound` increased by two PDR interval widths.
 2. If interval width is higher than the current phase goal:
 - + Else, IF NDR interval does not meet the current phase width goal, prepare for internal search. The new transmit rate is a geometric average of NDR `lower_bound` and NDR `upper_bound`.
 - + Else, IF PDR interval does not meet the current phase width goal, prepare for internal search. The new transmit rate is a geometric average of PDR `lower_bound` and PDR `upper_bound`.

3. Else, IF some bound has still only been measured at a lower duration, prepare to re-measure at the current duration (and the same transmit rate). The order of priorities is:

- + NDR lower_bound,
- + PDR lower_bound,
- + NDR upper_bound,
- + PDR upper_bound.

4. Else, do not prepare any new rate, to exit the phase. This ensures that at the end of each non-initial phase all intervals are valid, narrow enough, and measured at current phase trial duration.

4.4. Sample MLRsearch Run

TODO add a sample MLRsearch run with values.

5. Known Implementations

The only known working implementation of MLRsearch is in Linux Foundation FD.io CSIT project [FDio-CSIT-MLRsearch]. MLRsearch is also available as a Python package in [PyPI-MLRsearch].

5.1. FD.io CSIT Implementation Deviations

This document so far has been describing a simplified version of MLRsearch algorithm. The full algorithm as implemented contains additional logic, which makes some of the details (but not general ideas) above incorrect. Here is a short description of the additional logic as a list of principles, explaining their main differences from (or additions to) the simplified description, but without detailing their mutual interaction.

1. Logarithmic transmit rate.

- * In order to better fit the relative width goal, the interval doubling and halving is done differently.
- * For example, the middle of 2 and 8 is 4, not 5.

2. Optimistic maximum rate.

- * The increased rate is never higher than the maximum rate.

- * Upper bound at that rate is always considered valid.
3. Pessimistic minimum rate.
 - * The decreased rate is never lower than the minimum rate.
 - * If a lower bound at that rate is invalid, a phase stops refining the interval further (until it gets re-measured).
 4. Conservative interval updates.
 - * Measurements above current upper bound never update a valid upper bound, even if drop ratio is low.
 - * Measurements below current lower bound always update any lower bound if drop ratio is high.
 5. Ensure sufficient interval width.
 - * Narrow intervals make external search take more time to find a valid bound.
 - * If the new transmit increased or decreased rate would result in width less than the current goal, increase/decrease more.
 - * This can happen if the measurement for the other interval makes the current interval too narrow.
 - * Similarly, take care the measurements in the initial phase create wide enough interval.
 6. Timeout for bad cases.
 - * The worst case for MLRsearch is when each phase converges to intervals way different than the results of the previous phase.
 - * Rather than suffer total search time several times larger than pure binary search, the implemented tests fail themselves when the search takes too long (given by argument `_timeout_`).
6. IANA Considerations
- No requests of IANA.

7. Security Considerations

Benchmarking activities as described in this memo are limited to technology characterization of a DUT/SUT using controlled stimuli in a laboratory environment, with dedicated address space and the constraints specified in the sections above.

The benchmarking network topology will be an independent test setup and MUST NOT be connected to devices that may forward the test traffic into a production network or misroute traffic to the test management network.

Further, benchmarking is performed on a "black-box" basis, relying solely on measurements observable external to the DUT/SUT.

Special capabilities SHOULD NOT exist in the DUT/SUT specifically for benchmarking purposes. Any implications for network security arising from the DUT/SUT SHOULD be identical in the lab and in production networks.

8. Acknowledgements

Many thanks to Alec Hothan of OPNFV NFVbench project for thorough review and numerous useful comments and suggestions.

9. References

9.1. 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>>.

9.2. Informative References

- [FDio-CSIT-MLRsearch] "FD.io CSIT Test Methodology - MLRsearch", June 2019, <https://docs.fd.io/csit/rls1904/report/introduction/methodology_data_plane_throughput/methodology_mlrsearch_tests.html>.

[PyPI-MLRsearch]

"MLRsearch 0.2.0, Python Package Index", August 2018,
<<https://pypi.org/project/MLRsearch/>>.

Authors' Addresses

Maciek Konstantynowicz (editor)
Cisco Systems

Email: mkonstan@cisco.com

Vratko Polak (editor)
Cisco Systems

Email: vrpolak@cisco.com