

Benchmarking Methodology Working Group
Internet-Draft
Intended status: Experimental
Expires: May 6, 2021

LM. Contreras
J. Rodriguez
L. Luque
Telefonica
November 2, 2020

5G transport network benchmarking
draft-contreras-bmwg-5g-02

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 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. Conventions used in this document	3
3. 5G services	3
4. Benchmarking aspects of transport networks in 5G	4
5. Key Performance Indicators	4
5.1. Control and management plane KPIs KPIs	4
5.2. Data plane KPIs	5
6. Guidance on 5G transport benchmarking	5
6.1. Benchmarking topology	5
6.2. IETF network slices	6
7. Security Considerations	7
8. IANA Considerations	7
9. References	7
9.1. Normative References	7
9.2. Informative References	7
Acknowledgments	8
Contributors	8
Authors' Addresses	9

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. Refer to [TMV] for a general overview of different aspects impacting 5G technology performance. 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, eMMB services are expected to provide peak bit rates of up to 1 Gbps, uRRLC services will require latencies as lower 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. With this respect the IETF has initiated the work in transport slicing (see [I-D.nsd-t-teas-ietf-network-slice-definition]).

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. Key Performance Indicators

In order to define benchmarking criteria it is convenient to formalize Key Performance Indicators (KPIs) to assist on the assessment of the performance of the technologies under analysis.

5.1. Control and management plane KPIs

[I-D.nsdtd-teas-ietf-network-slice-definition] introduces the concept of IETF Network Slice controller (NSC) as the element in charge of realizing, maintaining and monitor the IETF Network Slices as requested by higher level systems. The element itself can be assimilated to any other controller. From that perspective, it is possible to leverage on RFC8456 [RFC8456] to identify suitable KPIs. Thus, the following KPIs can be considered:

- o Performance KPIs, including asynchronous message processing time and rate, proactive and reactive IETF network slice provisioning time, etc.
- o Scalability KPIs, such as control sessions capacity, number of IETF network slices handled, etc.
- o Security KPIs, like exception handling, denial-of-service attacks, etc.

- o Reliability KPIs, as failover time for the NSC.

Apart from that, other KPIs related to the monitoring and maintenance of the IETF network slices can be considered, as the ones related to telemetry.

5.2. Data plane KPIs

Data Plane KPIs will help to predict data plane performance under different measurement conditions. Existing metrics to consider are:

- o Bandwidth, considered as the maximum achievable throughput between two points. Those points can represent the ingress and egress ports of a equipment (e.g., to determine maximum throughput of a single element) or to an end-to-end setup. The throughput could be differentiated in both directions of the link (i.e., uplink and downlink).
- o Latency, considered as the network delay when transmitting between source and destination endpoints. This can apply to a single box (e.g., delay induced by a router implementing certain technology) or to a network scenario defined by a certain topology. RFC2681 [RFC2681] and RFC7679 [RFC7679] discuss about two-way (i.e., round trip time) and one-way delay metrics, respectively.
- o Jitter, understood as jitter the observable packet delay variation (PDV) as defined by RFC3393 [RFC3393], which is measured by the difference in the one-way.
- o Other general data-plane related issues affected for the usage of specific data plane technologies and/or encapsulations, such as MTU size, etc.
- o Other data-plane related issues specific to 5G such as e.g. the capability of isolation, understood as the avoidance of interference (i.e., affection) of traffic from different users in case of one of those user misbehaves or consumes more resources than expected.

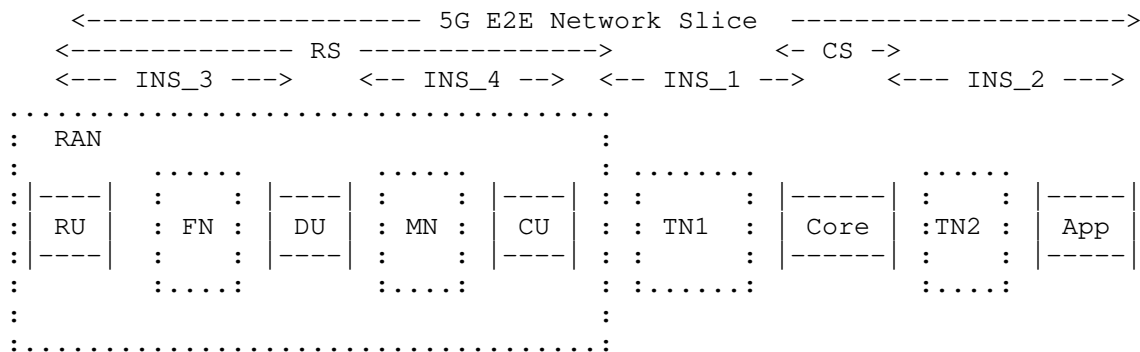
6. Guidance on 5G transport benchmarking

To be completed.

6.1. Benchmarking topology

5G networks can be as complex as the one in Figure 1, from [I-D.rokui-5g-ietf-network-slice]. It comprises of fronthaul, midhaul, backhaul and even backbone segments, spanning end-to-end.

Each of those segments will have particularities, in terms of technologies used or routing solutions in place. In addition to that, because of the specific needs of the traffic to be supported, there will be different requirements applying to each of those segments. A clear example is the fronthaul segment, where protocols like CPRI or eCPRI will impose strict latency and bandwidth requirements, for instance.



Legend

INS: 5G IETF Network Slice
 RS: RAN Slice
 CS: Core Slice
 FN: Fronthaul network
 MN: Midhaul network
 TN: Transport network
 DU: Distributed Unit
 CU: Central Unit
 RU: Radio Unit
 App: Mobile Application Servers

Figure 1: Transport segments in 5G networks

Since different restrictions apply, it will be necessary to consider specific topologies for each of these segments, able to represent typical but meaningful deployment scenarios

6.2. IETF network slices

On top of the network above, thanks to the network slicing approach, it will be possible to build logical networks tailored to specific needs and services (e.g., eMBB, uRLLC, etc). As consequence, for the different topologies defined for the distinct transport network segments, it can be necessary to benchmark distinct kind of IETF network slices. The distinction will come from the parametrization

used, expressed in base a number of parameters and attributes as described in [I-D.contreras-teas-slice-nbi].

An important aspect to test is the idea of isolation, or how a IETF network slice is not affected for a misbehavior on other IETF network slices supported by the same physical infrastructure. Different transport technologies can have distinct behaviors in this respect. For instance traffic policing or shaping mechanisms, hierarchical QoS, allocation of dedicated resources as FlexE calendar slots, etc. In this respect a common scenario can be solved following different strategies according to the capabilities of each transport technology in place.

7. Security Considerations

This draft does not include any security considerations.

8. IANA Considerations

This draft does not include any IANA considerations

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.contreras-teas-slice-nbi]
Contreras, L., Homma, S., and J. Ordonez-Lucena, "IETF Network Slice use cases and attributes for Northbound Interface of controller", draft-contreras-teas-slice-nbi-03 (work in progress), October 2020.

[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., Voyer, D., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-24 (work in progress), October 2020.

- [I-D.nsd-t-eas-ietf-network-slice-definition]
Rokui, R., Homma, S., Makhiyani, K., Contreras, L., and J. Tantsura, "Definition of IETF Network Slices", draft-nsd-t-eas-ietf-network-slice-definition-00 (work in progress), October 2020.
- [I-D.rokui-5g-ietf-network-slice]
Rokui, R., Homma, S., Foy, X., Contreras, L., Eardley, P., Makhiyani, K., Flinck, H., Schatzmayr, R., Tizghadam, A., Janz, C., and H. Yu, "IETF Network Slice for 5G and its characteristics", draft-rokui-5g-ietf-network-slice-00 (work in progress), November 2020.
- [RFC2681] Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, DOI 10.17487/RFC2681, September 1999, <<https://www.rfc-editor.org/info/rfc2681>>.
- [RFC3393] Demichelis, C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, DOI 10.17487/RFC3393, November 2002, <<https://www.rfc-editor.org/info/rfc3393>>.
- [RFC7679] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Delay Metric for IP Performance Metrics (IPPM)", STD 81, RFC 7679, DOI 10.17487/RFC7679, January 2016, <<https://www.rfc-editor.org/info/rfc7679>>.
- [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>>.
- [TMV] "Validating 5G Technology Performance", 5G PPP TMV WG white paper , June 2019.

Acknowledgments

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

Contributors

A. Florez and D. Artunedo (both from Telefonica) have also contributed to this document from their work in 5GENESIS project.

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 6, 2021

K. Sun
H. Yang
J. Lee
H. Nguyen
Y. Kim
Soongsil University
November 02, 2020

Considerations for Benchmarking Network Performance in Containerized
Infrastructures
draft-dcn-bmwg-containerized-infra-05

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 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. Terminology	3
3. Benchmarking Considerations	4
3.1. Comparison with the VM-based Infrastructure	4
3.2. Container Networking Classification	5
3.3. Resource Considerations	8
4. Benchmarking Scenarios for the Containerized Infrastructure .	10
5. Additional Considerations	12
6. Benchmarking Experience(Contiv-VPP)	13
6.1. Benchmarking Environment(Contiv-VPP)	13
6.2. Trouble shooting and Result	17
7. Benchmarking Experiment(SR-IoV-DPDK)	19
7.1. Benchmarking Environment(SR-IoV-DPDK)	19
7.2. Trouble shooting and Result(SR-IoV-DPDK)	23
8. Security Considerations	23
9. Acknowledgement	23
10. Informative References	23
Authors' Addresses	24

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.

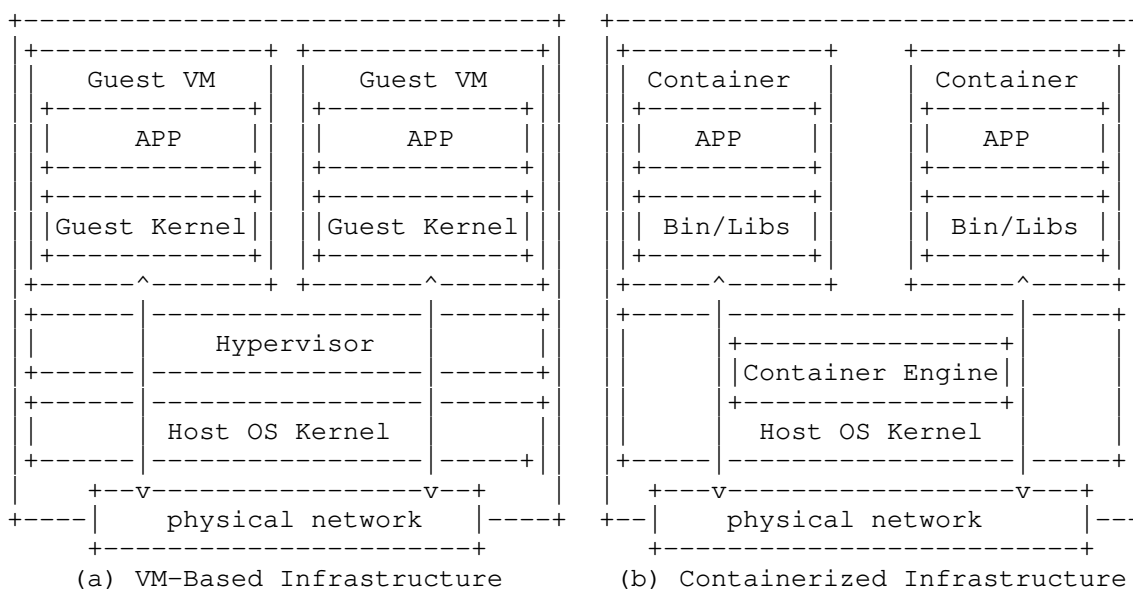


Figure 1: Comparison of NFV Infrastructures

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]

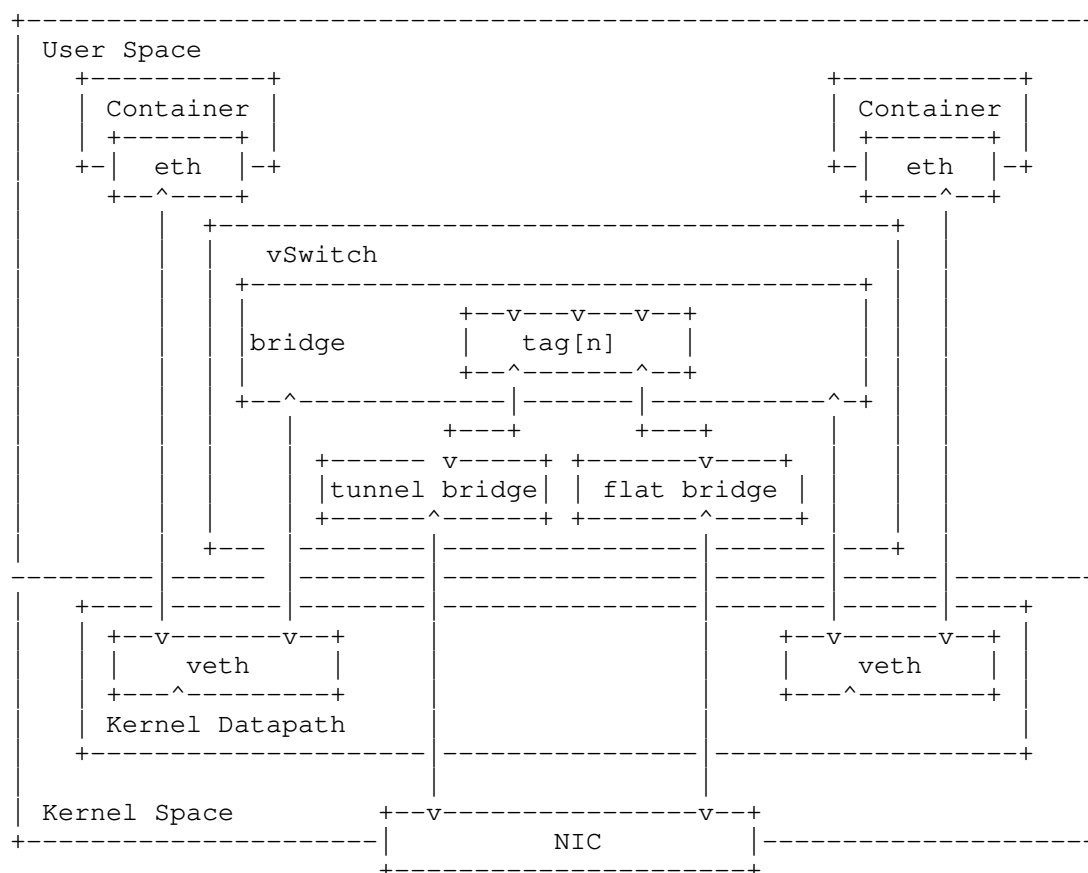


Figure 2: Examples of Kernel Space Network Model

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

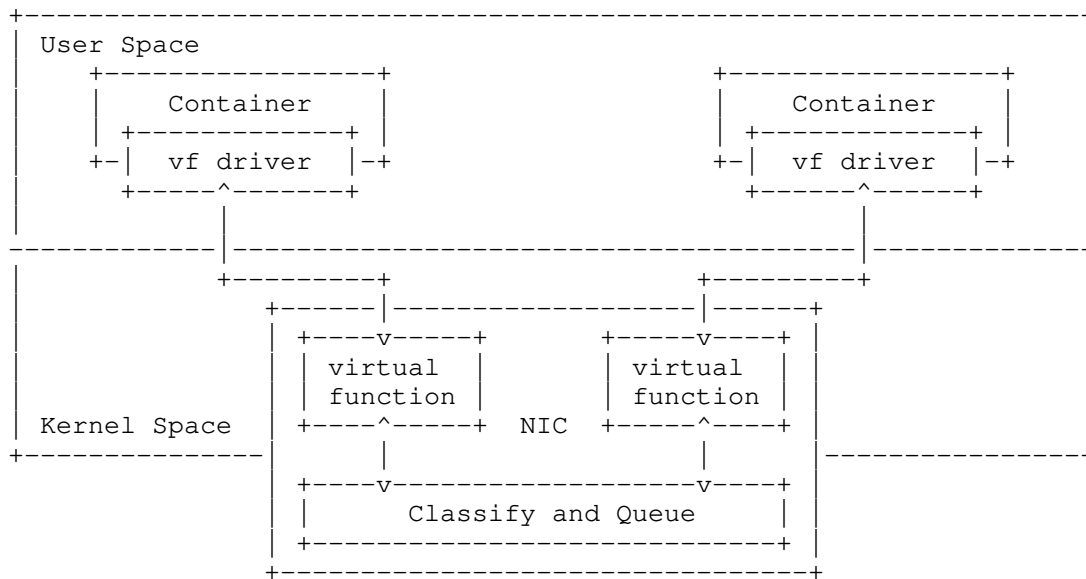


Figure 3: Examples of User Space Network Model - Device Pass-through

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

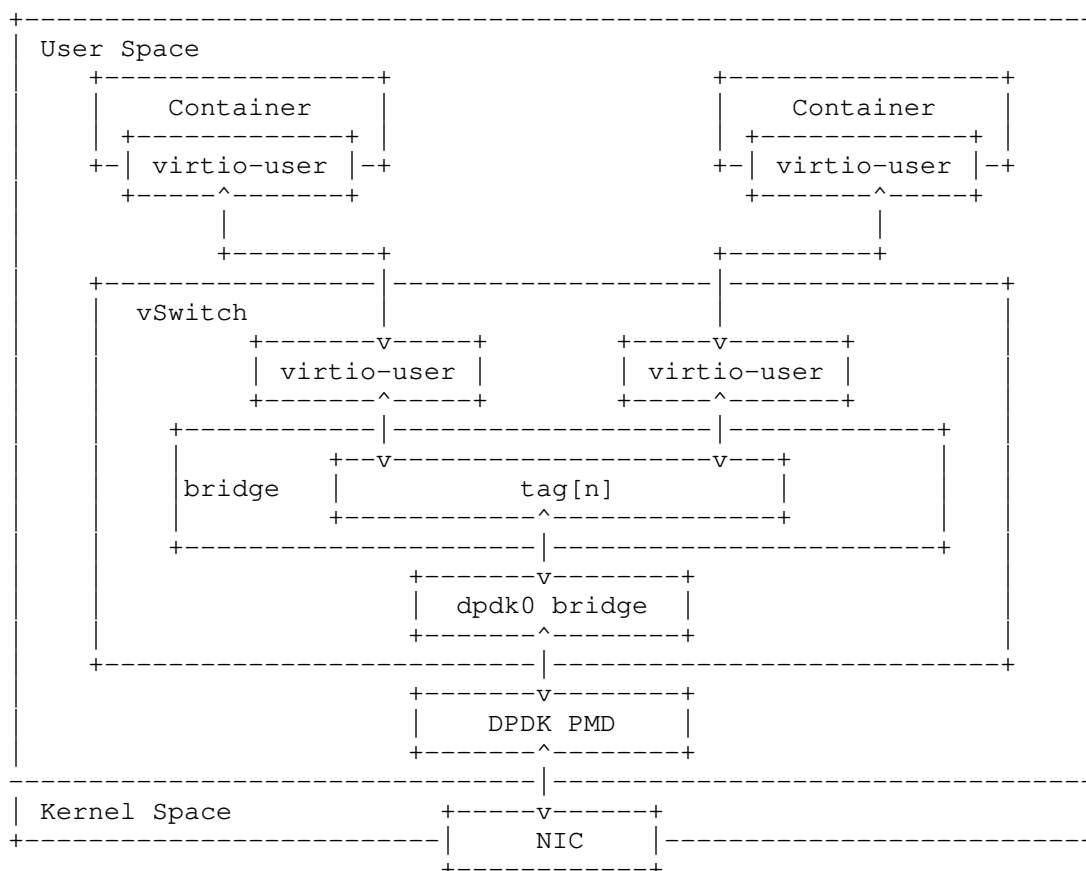


Figure 4: Examples of User Space Network Model - vSwitch Model using DPDK

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.

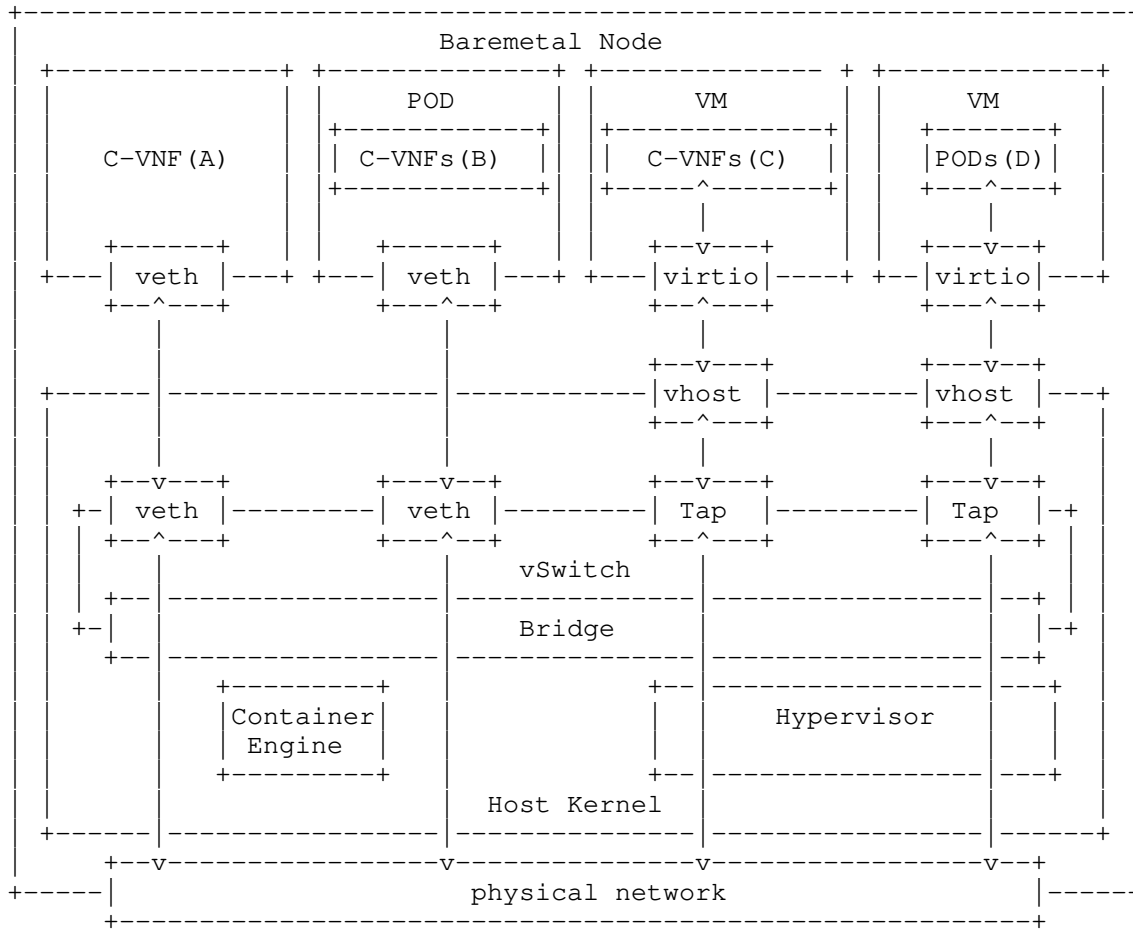


Figure 5: Examples of Networking Architecture based on Deployment Models - (A) C-VNF on Baremetal (B) Pod on Baremetal (BMP) (C) C-VNF on VM (D) Pod on VM (VMP)

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

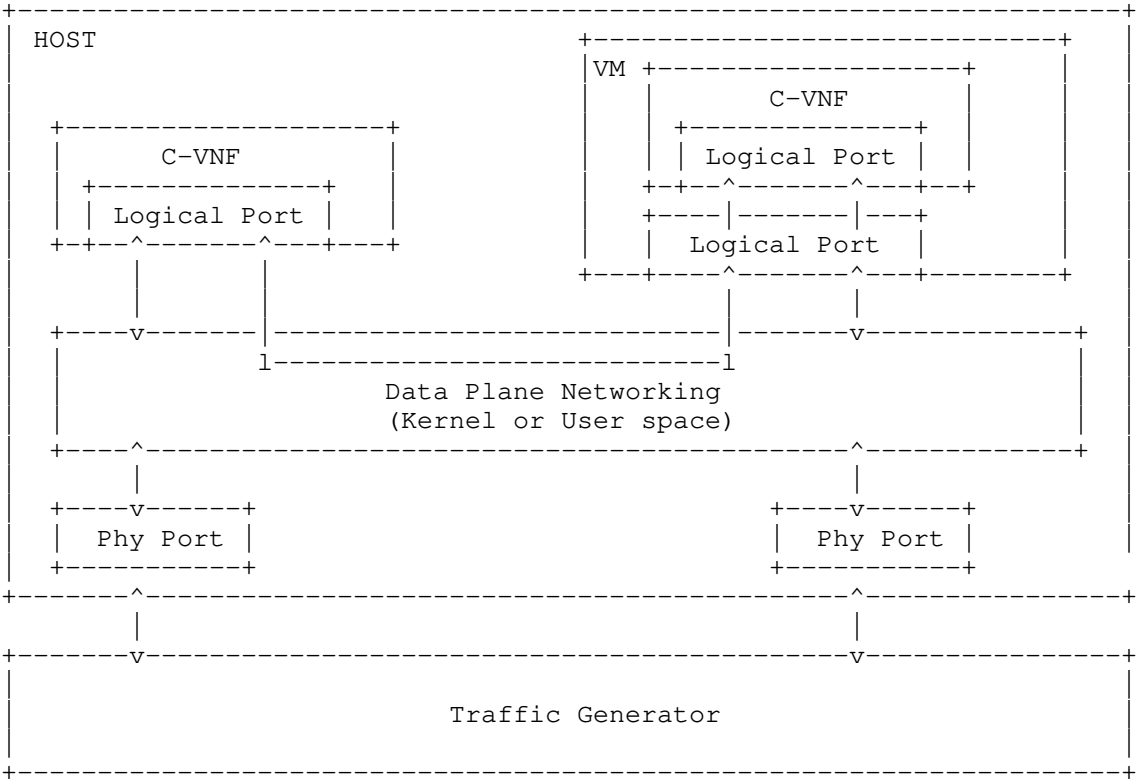


Figure 6: Single Host Test Scenario - BMP2VMP

- o VMP2VMP scenario

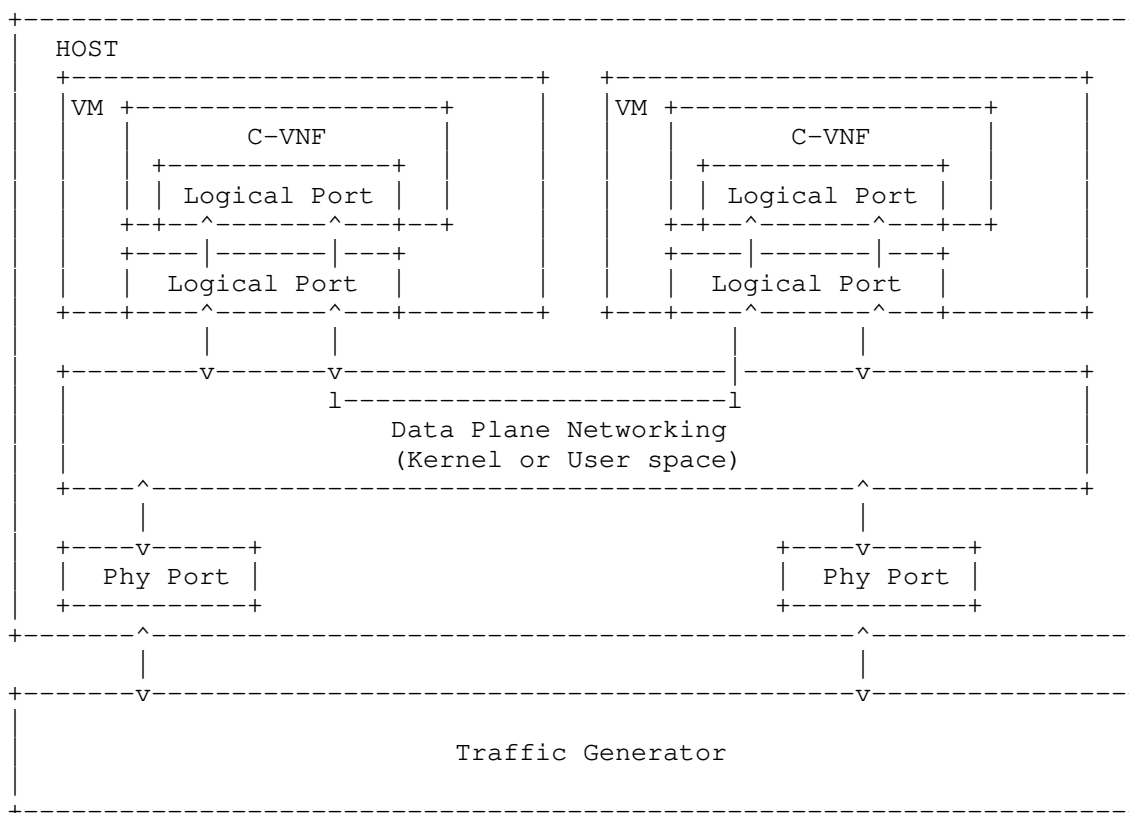


Figure 7: Single Host Test Scenario - VMP2VMP

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.

Regarding the above draft, it can be classified into two types of traffic for benchmark testing. One is North/South traffic and the other is East/West traffic. North/South has a architecture that receives data from other servers and routes them through VNF. 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. The one of the example is Service Function Chaining. Since network acceleration technology in a container environment has different accelerated areas depending on the method provided, performance differences may occur depending on traffic patterns.

6. Benchmarking Experience (Contiv-VPP)

6.1. Benchmarking Environment (Contiv-VPP)

In this test, our purpose is that we test performance of user space based model for container infrastructure and figure out relationship between resource allocation and network performance. With respect to this, we setup Contiv-VPP which is one of the user space based network solution 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 8: Test Environment-Server Specification

- o The architecture of benchmarking

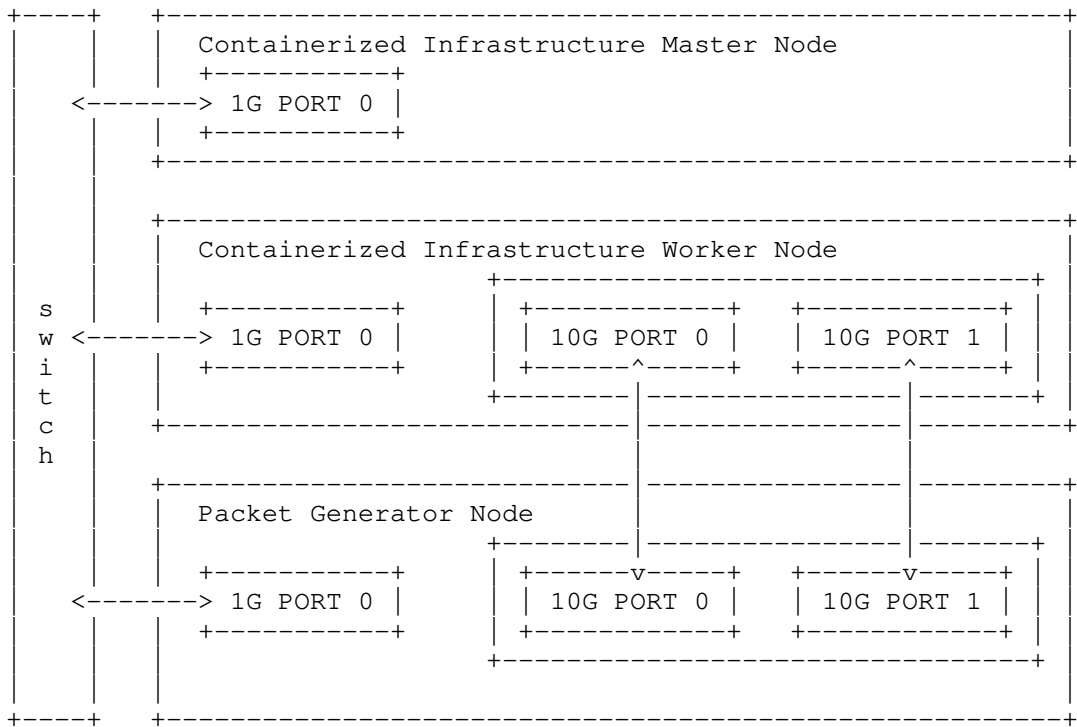


Figure 9: Test Environment-Architecture

o Network model of Containerized Infrastructure(User space Model)

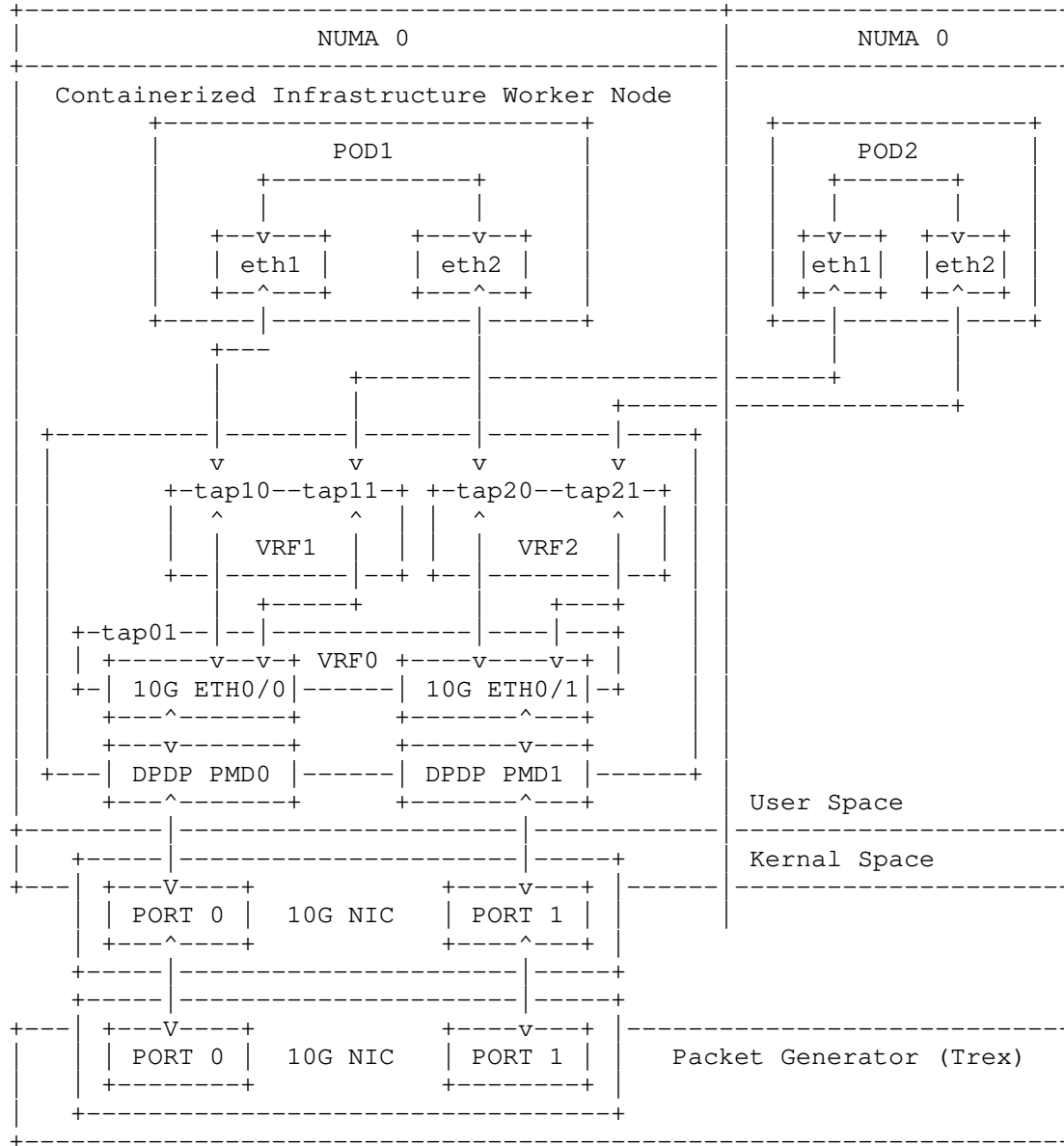


Figure 10: Test Environment-Network Architecture

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

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

6.2. Trouble shooting and Result

In this environment, we confirmed that the routing table doesn't work when we send packet 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. That is, 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.

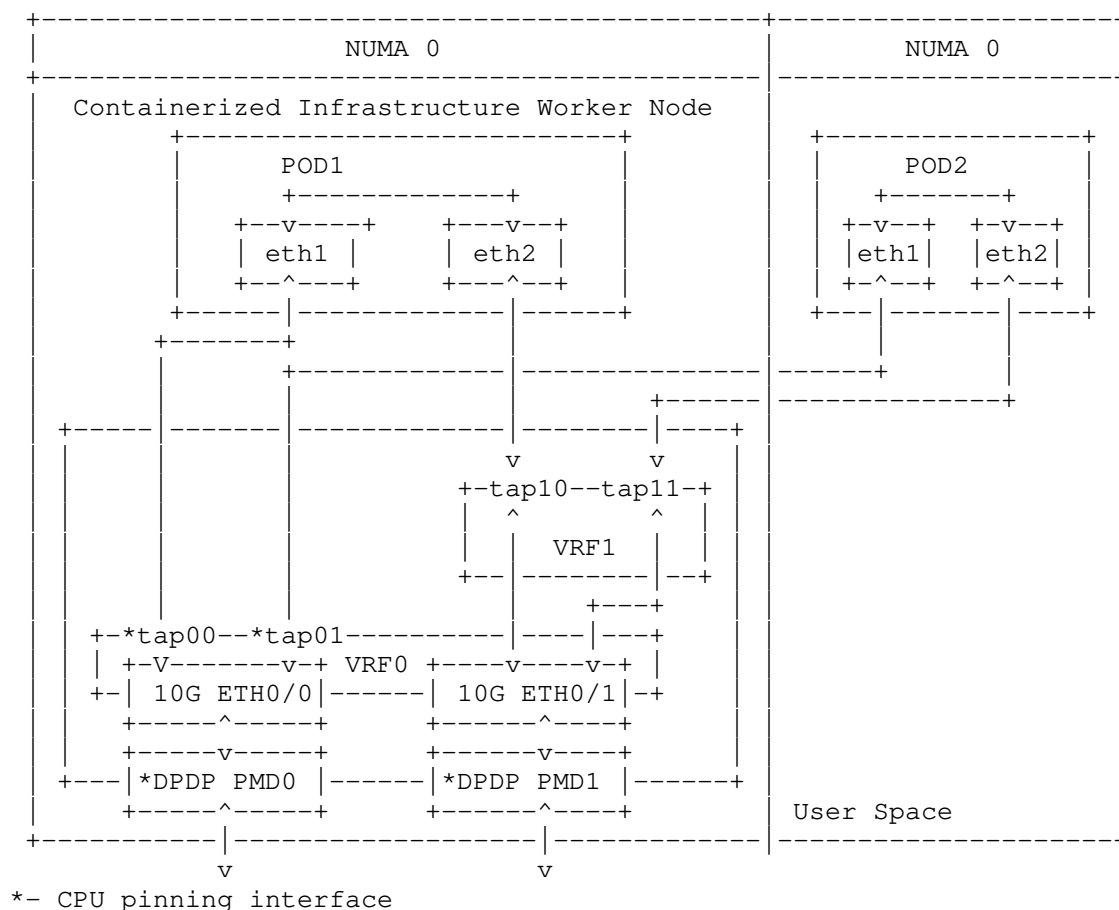


Figure 11: Test Environment-Network Architecture (CPU Pinning)

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 a specific PODs. Exclusive mode is that a specific POD dedicates a specific CPU to use. In shared mode, we assigned two CPU for several POD, in exclusive mode, we dedicated one CPU for one POD, independently. The result is like Figure 12. 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 network interface using Shared Mode and Exclusive Mode in the same node and different node respectively. In Exclusive and Shared mode tests, we confirmed that Exclusive mode showed better performance than Shared mode when same

NUMA cpu assigned, respectively. However, we confirmed that performance is reduced at the section between the vpp switch and the POD, so that it affect to total result.

Model	NUMA Mode (pinning)	Result (Gbps)
Switch only	N/A	3.1
	same NUMA	9.8
K8S Scheduler	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 12: Test Results

7. Benchmarking Experiment (SR-IoV-DPDK)

7.1. Benchmarking Environment (SR-IoV-DPDK)

In this test, our purpose is that we test performance of user space based 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 13: Test Environment-Server Specification

- o The architecture of benchmarking

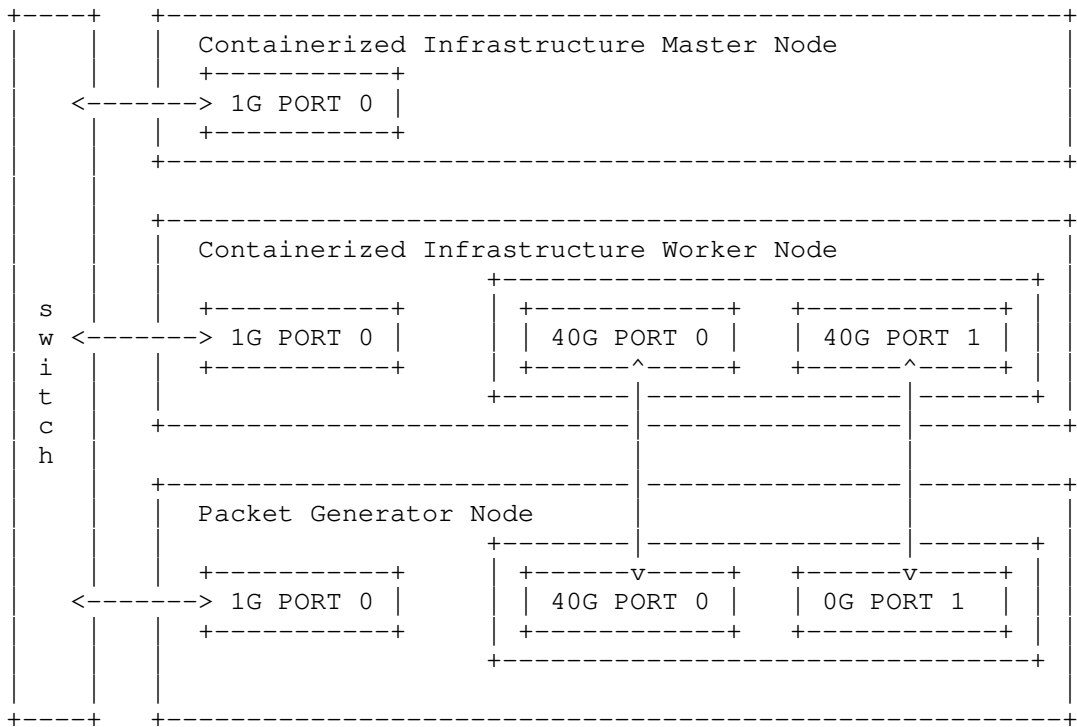


Figure 14: Test Environment-Architecture

o Network model of Containerized Infrastructure(User space Model)

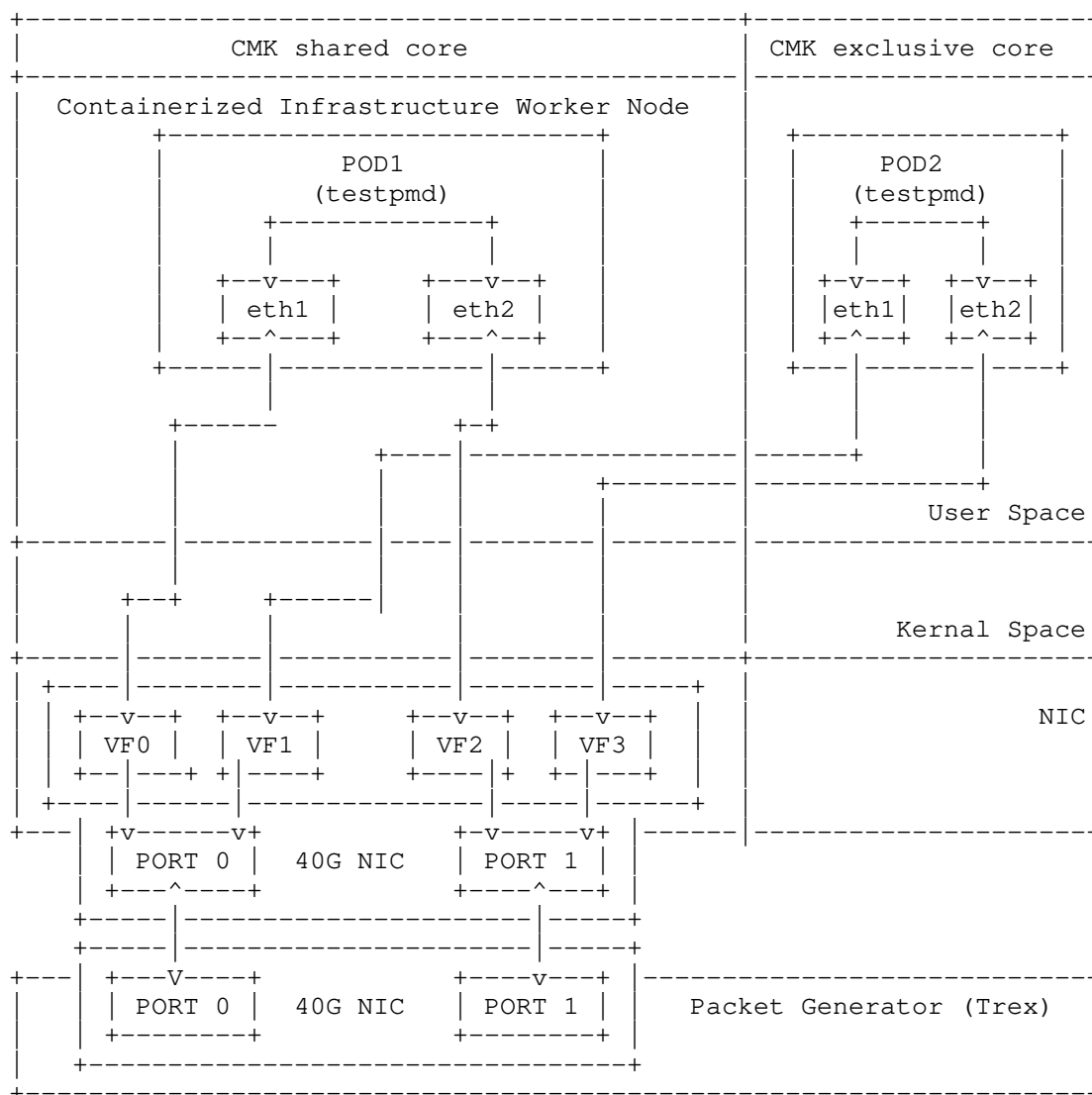


Figure 15: Test Environment-Network Architecture

We setup 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 to create 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 setup TREX to route packet from eth1 to eth2 in a POD.

7.2. Trouble shooting and Result (SR-IoV-DPDK)

TBD

8. Security Considerations

TBD

9. Acknowledgement

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

10. 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

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

Phone: +82 10 7448 4664
EMail: jangwon.lee@dcn.ssu.ac.kr

Quang Huy Nguyen
School of Electronic Engineering
Soongsil University
369, Sangdo-ro, Dongjak-gu
Seoul, Seoul 06978
Republic of Korea

Phone: +82 10 4281 0720
EMail: huynq@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

Network Working Group
Internet-Draft
Updates: 2544 (if approved)
Intended status: Informational
Expires: May 20, 2021

A. Morton
AT&T Labs
November 16, 2020

Updates for the Back-to-back Frame Benchmark in RFC 2544
draft-ietf-bmwg-b2b-frame-03

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 20, 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	10
7. Security Considerations	11
8. IANA Considerations	12
9. Acknowledgements	12
10. References	12
10.1. Normative References	12
10.2. Informative References	13
Author's Address	14

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 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 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) [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 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 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 **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 (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 Benchmark for Back-to-back Frames 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 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:

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:

Actual Buffer Time =

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

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. Warren Kumari's review improved readability in several key passages.

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-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>>.
- [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)>.

[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: Informational
Expires: February 8, 2021

S. Jacob, Ed.
K. Tiruveedhula
Juniper Networks
August 7, 2020

Benchmarking Methodology for EVPN and PBB-EVPN
draft-ietf-bmwg-evpntest-06

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 8, 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
1.1. Requirements Language	3
1.2. Terminologies	3
2. Test Topology	4
3. Test Cases for EVPN Benchmarking	7
3.1. Data Plane MAC Learning	7
3.2. Control Plane MAC Learning	8
3.3. MAC Flush-Local Link Failure and Relearning	9
3.4. MAC Flush-Remote Link Failure and Relearning.	10
3.5. MAC Aging	11
3.6. Remote MAC Aging	11
3.7. Control and Data plane MAC Learning	12
3.8. High Availability.	13
3.9. ARP/ND Scale	14
3.10. Scaling of Services	15
3.11. Scale Convergence	15
3.12. SOAK Test.	16
4. Test Cases for PBB-EVPN Benchmarking	17
4.1. Data Plane Local MAC Learning	17
4.2. Data Plane Remote MAC Learning	18
4.3. MAC Flush-Local Link Failure	19
4.4. MAC Flush-Remote Link Failure	20
4.5. MAC Aging	21
4.6. Remote MAC Aging.	21
4.7. Local and Remote MAC Learning	22
4.8. High Availability	23
4.9. Scale	24
4.10. Scale Convergence	25
4.11. Soak Test	26
5. Acknowledgments	26
6. IANA Considerations	27
7. Security Considerations	27
8. References	27
8.1. Normative References	27
8.2. Informative References	27
Appendix A. Appendix	28
Authors' Addresses	28

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 aging, 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 8174 [RFC8174].

1.2. Terminologies

Most of the terminology used in this documents comes from [RFC7432] and [RFC7632].

All-Active Redundancy Mode: When all PEs attached to an Ethernet segment are allowed to forward known unicast traffic to/from that Ethernet segment for a given VLAN, then the Ethernet segment is defined to be operating in All-Active redundancy mode.

AA: All Active mode

CE: Customer Router/Devices/Switch.

DF: Designated Forwarder

DUT: Device under test.

Ethernet Segment (ES): When a customer site (device or network) is connected to one or more PEs via a set of Ethernet links, then that set of links is referred to as an 'Ethernet segment'.

EVI: An EVPN instance spanning the Provider Edge (PE) devices participating in that EVPN.

Ethernet Segment Identifier (ESI): A unique non-zero identifier that identifies an Ethernet segment is called an 'Ethernet Segment Identifier'.

Ethernet Tag: An Ethernet tag identifies a particular broadcast domain, e.g., a VLAN. An EVPN instance consists of one or more broadcast domains.

Interface: Physical interface of a router/switch.

IRB: Integrated routing and bridging interface

MAC: Media Access Control addresses on a PE.

MHPE2: Multi homed Provider Edge router 2.

MHPE1: Multi homed Provider Edge router 1.

SHPE3: Single homed Provider Edge Router 3.

PE: Provider Edge device.

P: Provider Router.

RR: Route Reflector.

RT: Traffic Generator.

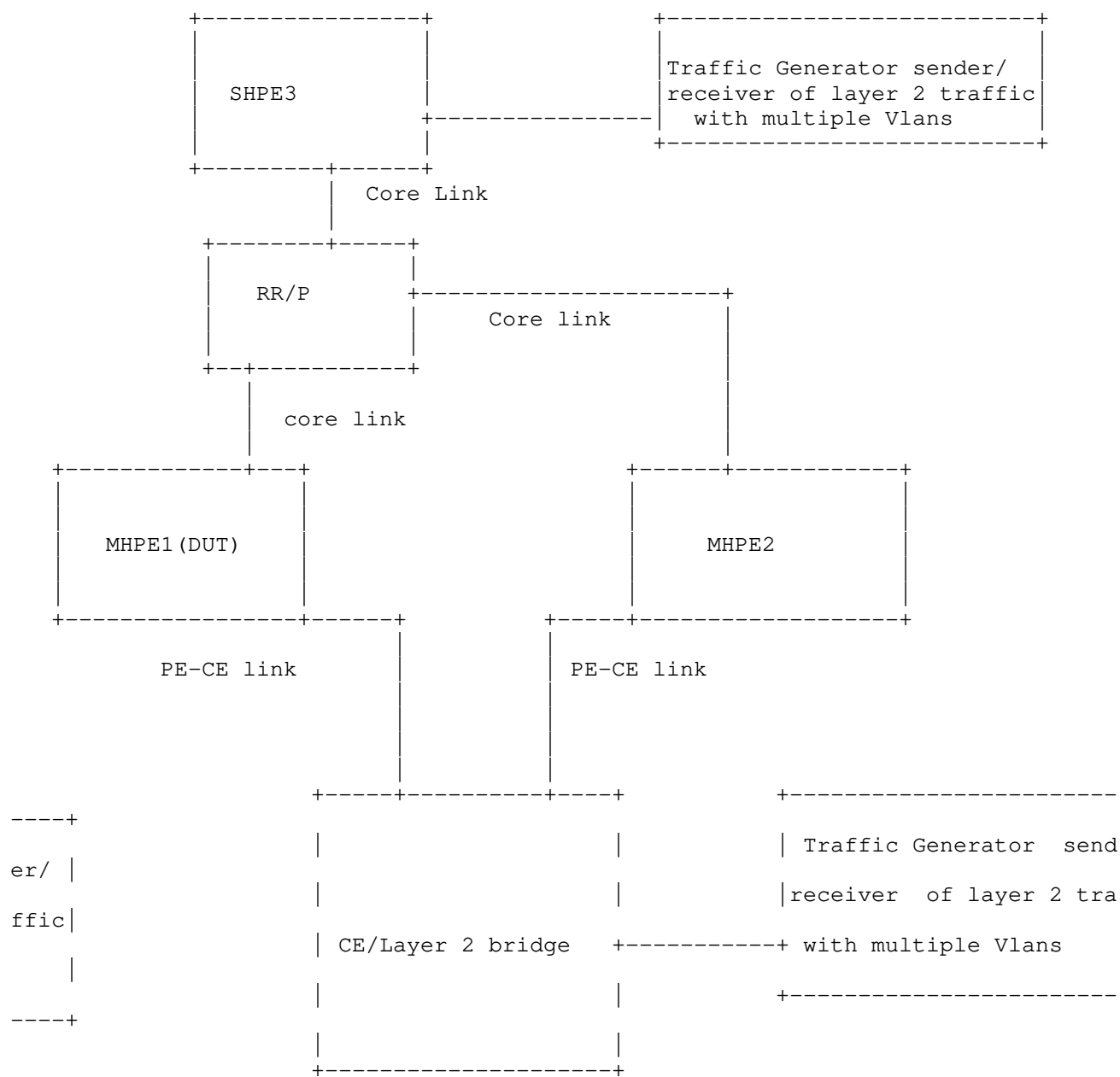
Sub Interface: Each physical Interfaces is subdivided in to a set of Logical units.

SA: Single Active

Single-Active Redundancy Mode: When a single PE (among all the PEs attached to an Ethernet segment) is the only PE allowed to forward traffic to/from a given Ethernet segment for a given VLAN, then that Ethernet segment is defined to be operating in Single-Active redundancy mode.

2. Test Topology

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 traffic generator is connected to the CE and SHPE3. The MHPE1 acts as DUT. The traffic generator will be used as sender and receiver of traffic. The test measurements are taken from the DUT. MHPE1 and MHPE2 are multi-homed routers connected to CE running single active mode. The traffic generator will be generating traffic at 10% of the line rate.



Topology 1

Test Setup

Figure 1

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

Table showing the traffic directions of various EVPN/PBB-EVPN benchmarking test cases. Depending on the test scenario, the traffic can be uni-directional or bi-directional (configured in the traffic generator).

Figure 2

Test Setup Configurations:

SHPE3 is configured with Interior Gateway protocols like OSPF or IS-

IS for underlay, LDP for MPLS support, Interior Border Gateway with EVPN address family for overlay support. This router must be configured with N EVPN/PBB-EVPN instances for testing. Traffic generator is connected to this router for sending and receiving traffic.

RR is configured with Interior Gateway protocols like OPSF or IS-IS for underlay, LDP for MPLS support, Interior Border Gateway with EVPN address family for overlay support. This router function as both provider router and a route reflector.

MHPE1 is configured with Interior Gateway protocols like OPSF or IS-IS for underlay, LDP for MPLS support, Interior Border Gateway with EVPN address family for overlay support. This router must be configured with N EVPN/PBB-EVPN instances for testing. This router is configured with ESI per vlan or ESI per interface. It is functioning as multi homing PE working on Single Active EVPN mode. This router serves as the DUT and it is connected to CE. MHPE1 is acting as DUT for all the test cases.

MHPE2 is configured with Interior Gateway protocols like OPSF or IS-IS for underlay, LDP for MPLS support, Interior Border Gateway with EVPN address family for overlay support. This router must be configured with N EVPN/PBB-EVPN instances for testing. This router is configured with ESI per vlan or ESI per interface. It is functioning as multi homing PE working on Single Active EVPN mode. It is connected to CE.

CE is acting as bridge configured with multiple vlans. The same vlans are configured on MHPE1, MHPE2, SHPE3. traffic generator is connected to CE. the traffic generator acts as sender or receiver of traffic.

Depending up on the test scenarios the traffic generators will be used to generate uni directional or bi directional flows.

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

3. Test Cases for EVPN Benchmarking

3.1. Data Plane MAC Learning

Objective:

Measure the time taken to learn the Data Plane MAC in DUT.

Topology : Topology 1

Procedure:

The data plane MAC learning can be measured using the parameters defined in RFC 2889 section 5.8.

Confirm the DUT is up and running with EVPN.

Traffic generator connected to CE must send frames with "X" different source and destination MAC address for one vlan, the same vlan must be present in all the devices except RR.

Send "X" unicast frames from CE to MHPE1(DUT) for one EVPN instance working in SA mode.

The DUT will learn these "X" MAC in data plane.

Measurement :

Measure the time taken to learn "X" MAC locally in DUT evpn MAC table. The data plane measurement is taken by considering DUT as black box. The range of MAC are known from traffic generator, the same must be learned in DUT, the time taken to learn "X" MAC is measured. The measurement is carried out using external server which polls the DUT using automated scripts.

The test is repeated for "N" times and the values are collected. The MAC learning rate is calculated by averaging the values obtained from "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn.

MAC learning rate = (T1+T2+..Tn)/N

3.2. Control Plane MAC Learning

Objective:

Measure the time taken to learn the control plane MAC.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

Traffic generator connected to SHPE3 must send frames with "X" different source and destination MAC address for one vlan, the same vlan must be present in all the devices except RR.

Ensure the frames must be destined to one EVPN instance.

The DUT will learn these "X" MAC in control plane.

Measurement :

Measure the time taken by the DUT to learn the "X" MAC in the data plane. The test is repeated for "N" times and the values are collected. The remote MAC learning rate is calculated by averaging the values obtained from "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts.

MAC learning rate = (T1+T2+..Tn)/N

3.3. MAC Flush-Local Link Failure and Relearning

Objective:

Measure the time taken to flush the Data Plane MAC and the time taken to relearn the same amount of MAC.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

Send X frames with X different source and destination MAC addresses to DUT from CE using traffic generator for one vlan.

Ensure the DUT learns all X MAC addresses in data plane.

Fail the DUT-CE link and measure the time taken to flush these X MAC from the EVPN MAC table.

Bring up the link which was made Down(the link between DUT and CE). Measure time taken by the DUT to relearn these "X" MAC.

The DUT and MHPE2 are running SA mode.

Measurement :

Measure the time taken for flushing these X MAC addresses. Measure the time taken to relearn these X MAC in DUT. 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. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts.

Flush rate = $(T1+T2+..Tn)/N$

Relearning rate = $(T1+T2+..Tn)/N$

3.4. MAC Flush-Remote Link Failure and Relearning.

Objective:

Measure the time taken to flush the Control plane MAC learned in DUT during remote link failure and the time taken to relearn.

Topology : Topology 1

Procedure:

confirm the DUT is up and running with EVPN.

Send X frames with X different source and destination MAC addresses to DUT from SHPE3 using traffic generator for one vlan.

Bring down the link between SHPE3 and traffic generator.

SHPE3 will withdraw the routes from DUT due to link failure.

Measure the time taken to flush the DUT EVPN MAC table. The DUT and MHPE2 are running SA mode.

Bring up the link which was made Down(the link between SHPE3 and traffic generator).

Measure time taken by the DUT to relearn these "X" MAC from control plane.

Measurement :

Measure the time taken to flush X remote MAC from EVPN MAC table of the DUT. Measure the time taken to relearn these X MAC in DUT. 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. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts.

Flush rate = $(T1+T2+..Tn)/N$

Relearning rate = $(T1+T2+..Tn)/N$

3.5. MAC Aging

Objective:

To measure the MAC aging time.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

Send X frames with X different source and destination MAC addresses to DUT from CE using traffic generator for one vlan.

Ensure these X MAC addresses are learned in DUT.

Then stop the traffic.

Ensure the DUT and other devices in the test are using the default timers for aging.

Measure the time taken to flush X MAC 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 addresses due to aging. The test is repeated for "N" times and the values are collected. The aging is calculated by averaging the values obtained by "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts.

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

3.6. Remote MAC Aging

Objective:

Measure the control plane learned MAC aging time.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

Send X frames with X different source and destination MAC addresses to DUT from SHPE3 using traffic generator for one vlan.

Ensure these X MAC addresses are learned in DUT via control plane.

Then stop the traffic.

Ensure the DUT and other devices in the test are using the default timers for aging.

Measure the time taken to flush X MAC from DUT EVPN MAC table due to aging.

The DUT and MHPE2 are running SA mode.

Measurement :

Measure the time taken to flush X remote MAC learned in DUT EVPN MAC table due to aging. The test is repeated for "N" times and the values are collected. The aging is calculated by averaging the values obtained by "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts.

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

3.7. Control and Data plane MAC Learning

Objective:

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

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

Send X frames with X different source and destination MAC addresses to DUT from SHPE3 using traffic generator for one vlan.

Send X frames with different source and destination MAC addresses from traffic generator connected to CE for one vlan.

The source and destination addresses of flows must be complimentary to have unicast flows.

Measure the time taken by the DUT to learn 2X in EVPN MAC table.

DUT and MHPE2 are running in SA mode.

Measurement :

Measure the time taken to learn 2X MAC addresses in DUT EVPN MAC table. 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. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts

MAC learning rate = $(T1+T2+..Tn)/N$

3.8. High Availability.

Objective:

Measure traffic loss during routing engine fail over.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

Send X frames from CE to DUT from traffic generator with X different source and destination MAC addresses.

Send X frames from traffic generator to SHPE3 with X different source and destination MAC addresses, so that 2X MAC address will be learned in the DUT.

There is a bi directional traffic flow with X pps in each direction.

Ensure the DUT learn 2X MAC.

Then do a routing engine fail-over.

Measurement :

The expectation of the test is 0 traffic loss with no change in the DF role. DUT should not withdraw any routes. But in cases where the DUT is not properly synchronized between master and standby, due to that packet loss are observed. In that scenario the packet loss is measured. The test is repeated for "N" times and the values are collected. The packet loss is calculated by averaging the values obtained by "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts to ensure the DUT learned 2X MAC. The packet drop is measured using traffic generator.

Packet loss in sec with 2X MAC addresses = $(T1+T2+..Tn)/N$

3.9. ARP/ND Scale

Measure the DUT scaling limit of ARP/ND.

Objective:

Measure the ARP/ND scale of the DUT.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

Send X arp/neighbor discovery(ND) from the traffic generator to DUT with different sender ip/ipv6,MAC addresses to the target IRB address configured in EVPN instance.

The EVPN instance learns the MAC+ip and MAC+ipv6 addresses from these request and advertise as type 2 MAC+ip/MAC+ipv6 route to remote provide edge routers which have same EVPN configurations.

The value of X must be increased at an incremental value of 5% of X, till the limit is reached. The limit is where the DUT can't learn any more type 2 MAC+ip/MAC+ipv6. The test must be separately conducted for arp and ND.

Measurement :

Measure the scale limit of type 2 MAC+ip/MAC+ipv6 route which DUT can learn. The test is repeated for "N" times and the values are

collected. The scale limit is calculated by averaging the values obtained by "N" samples for both MAC+ip and MAC+ipv6. "N" is an arbitrary number to get a sufficient sample. The scale value obtained by each sample be v1,v2..vn. The measurement is carried out using external server which polls the DUT using automated scripts to find the scale limit of MAC+ipv4/MAC+ipv6.

Scale limit for MAC+ip = $(v1+v2+..vn)/N$

Scale limit for MAC+ipv6 = $(v1+v2+..vn)/N$

3.10. Scaling of Services

Objective:

Measure the scale of EVPN instances that a DUT can hold.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

The DUT, MHPE2 and SHPE3 are scaled to "N" EVI.

Ensure routes received from MHPE2 and SHPE3 for "N" EVI in the DUT.

Then increment the scale of N by 5% of N till the limit is reached.

The limit is where the DUT cant learn any EVPN routes from its peers.

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. The scope of the test is find out the maximum evpn instance that a DUT can hold. The measurement is carried out using external server which polls the DUT using automated scripts to find the scale limit of EVPN instances.

3.11. Scale Convergence

Objective:

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

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

Scale N EVIs in DUT, SHPE3 and MHPE2.

Send F frames to DUT from CE using traffic generator with X different source and destination MAC addresses for N EVI's.

Send F frames from traffic generator to SHPE3 with X different source and destination MAC addresses.

There will be 2X number of MAC addresses 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 BGP session is in established state in DUT.

Measure the time taken to learn 2X MAC address in DUT MAC table.

Measurement :

The DUT must learn 2X MAC addresses. Measure the time taken to learn 2X MAC in DUT. 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. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1, T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts.

Time taken to learn 2X MAC in DUT = $(T1+T2+...Tn)/N$

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:

Confirm the DUT is up and running with EVPN.

Scale N EVI's in DUT, SHPE3 and MHPE2. Send F frames to DUT from CE using traffic generator with different X source and destination MAC addresses for N EVI's.

Send F frames from traffic generator to SHPE3 with X different source and destination MAC addresses.

There will be 2X number of MAC addresses 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 in EVPN process, CPU usage and crashes in DUT.

Measurement :

Take the hourly reading of CPU, process memory. There should not be any leak, crashes, CPU spikes. The CPU spike is determined as the CPU usage which shoots at 40 to 50 percent of the average usage. The average value vary from device to device. Memory leak is determined by increase usage of the memory for EVPN process. The expectation is under steady state the memory usage for EVPN process should not increase. The measurement is carried out using external server which polls the DUT using automated scripts which captures the CPU usage and process memory.

4. Test Cases for PBB-EVPN Benchmarking

4.1. Data Plane Local MAC Learning

Objective:

Measure the time taken to learn the Data Plane MAC in DUT.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with PBB-EVPN.

Traffic generator connected to CE must send frames with "X" different source and destination MAC address for one vlan, the same vlan must be present in all the devices except RR.

Send "X" unicast frames from CE to MHPE1(DUT) for one PBB-EVPN instance working in SA mode.

The DUT will learn these "X" MAC in data plane.

Measurement :

Measure the time taken to learn "X" MAC locally in DUT PBB-EVPN MAC table. The data plane measurement is taken by considering DUT as black box. The range of MAC are known from traffic generator, the same must be learned in DUT, the time taken to learn "X" MAC is measured. The measurement is carried out using external server which polls the DUT using automated scripts.

The test is repeated for "N" times and the values are collected. The MAC learning rate is calculated by averaging the values obtained from "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1, T2...Tn.

MAC learning rate = $(T1+T2+...Tn)/N$

4.2. Data Plane Remote MAC Learning

Objective:

To Record the time taken to learn the remote MAC.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with PBB-EVPN.

Traffic generator connected to SHPE3 must send frames with "X" different source and destination MAC address for one vlan, the same vlan must be present in all the devices except RR.

Ensure the frames must be destined to one PBB-EVPN instance.

The DUT will learn these "X" MAC in data plane.

Measurement :

Measure the time taken by the DUT to learn the "X" MAC in the data plane. The test is repeated for "N" times and the values are collected. The remote MAC learning rate is calculated by averaging the values obtained from "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts.

MAC learning rate = (T1+T2+..Tn)/N

4.3. MAC Flush-Local Link Failure

Objective:

Measure the time taken to flush the locally learned MAC and the time taken to relearn the same amount of MAC.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with PBB-EVPN.

Send X frames with X different source and destination MAC addresses to DUT from CE using traffic generator for one vlan.

Ensure the DUT learns all X MAC addresses in data plane.

Fail the DUT-CE link and measure the time taken to flush these X MAC from the PBB-EVPN MAC table.

Bring up the link which was made Down(the link between DUT and CE).Measure time taken by the DUT to relearn these "X" MAC.

The DUT and MHPE2 are running SA mode.

Measurement :

Measure the time taken for flushing these X MAC addresses. Measure the time taken to relearn these X MAC in DUT. 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. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts.

Flush rate = $(T1+T2+...Tn)/N$

Relearning rate = $(T1+T2+...Tn)/N$

4.4. MAC Flush-Remote Link Failure

Objective:

Measure the time taken to flush the remote MAC learned in DUT due to remote link failure and relearning it.

Topology : Topology 1

Procedure:

confirm the DUT is up and running with PBB-EVPN.

Send X frames with X different source and destination MAC addresses to DUT from SHPE3 using traffic generator for one vlan.

Bring down the link between SHPE3 and traffic generator.

Measure the time taken to flush the DUT PBB-EVPN MAC table. The DUT and MHPE2 are running SA mode.

Bring up the link which was made Down(the link between SHPE3 and traffic generator).

Measure time taken by the DUT to relearn these "X" MAC

Measurement :

Measure the time taken to flush X remote MAC from PBB-EVPN MAC table of the DUT. Measure the time taken to relearn these X MAC in DUT. 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. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts.

Flush rate = $(T1+T2+...Tn)/N$

Relearning rate = $(T1+T2+...Tn)/N$

4.5. MAC Aging

Objective:

Measure the MAC aging time.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with PBB-EVPN.

Send X frames with X different source and destination MAC addresses to DUT from CE using traffic generator for one vlan.

Ensure these X MAC addresses are learned in DUT.

Then stop the traffic.

Ensure the DUT and other devices in the test are using the default timers for aging.

Measure the time taken to flush X MAC from DUT PBB-EVPN MAC table due to aging.

The DUT and MHPE2 are running SA mode.

Measurement :

Measure the time taken to flush X MAC addresses due to aging. The test is repeated for "N" times and the values are collected. The aging is calculated averaging the values obtained by "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts.

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

4.6. Remote MAC Aging.

Objective:

Measure the remote MAC aging time.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with PBB-EVPN.

Send X frames with X different source and destination MAC addresses to DUT from SHPE3 using traffic generator for one vlan.

Ensure these X MAC addresses are learned in DUT.

Then stop the traffic.

Ensure the DUT and other devices in the test are using the default timers for aging.

Measure the time taken to flush X MAC from DUT PBB-EVPN MAC table due to aging.

The DUT and MHPE2 are running SA mode.

Measurement :

Measure the time taken to flush X remote MAC learned in DUT EVPN MAC table due to aging. The test is repeated for "N" times and the values are collected. The aging is calculated by averaging the values obtained by "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts.

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

4.7. Local and Remote MAC Learning

Objective:

Measure the time taken to learn both local and remote MAC.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with PBB-EVPN.

Send X frames with X different source and destination MAC addresses to DUT from SHPE3 using traffic generator for one vlan.

Send X frames with different source and destination MAC addresses from traffic generator connected to CE for one vlan.

The source and destination addresses of flows must be complimentary to have unicast flows.

Measure the time taken by the DUT to learn 2X in PBB-EVPN MAC table.

DUT and MHPE2 are running in SA mode.

Measurement :

Measure the time taken to learn 2X MAC addresses in DUT PBB-EVPN MAC table. 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. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts

MAC learning rate = $(T1+T2+..Tn)/N$

4.8. High Availability

Objective:

Measure traffic loss during routing engine failover.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with PBB-EVPN.

Send X frames from CE to DUT from traffic generator with X different source and destination MAC addresses.

Send X frames from traffic generator to SHPE3 with X different source and destination MAC addresses, so that 2X MAC address will be learned in the DUT.

There is a bi directional traffic flow with X pps in each direction.

Ensure the DUT learn 2X MAC.

Then do a routing engine fail-over.

Measurement :

The expectation of the test is 0 traffic loss with no change in the DF role. DUT should not withdraw any routes. But in cases where the DUT is not properly synchronized between master and standby, due to that packet loss are observed. In that scenario the packet loss is measured. The test is repeated for "N" times and the values are collected. The packet loss is calculated by averaging the values obtained by "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1, T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts to ensure the DUT learned 2X MAC. The packet drop is measured using traffic generator.

Packet loss in sec with 2X MAC addresses = $(T1+T2+..Tn)/N$

4.9. Scale

Objective:

Measure the scale limit of DUT for PBB-EVPN.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with PBB-EVPN.

The DUT, MHPE2 and SHPE3 are scaled to "N" PBB-EVI.

Ensure routes received from MHPE2 and SHPE3 for "N" PBB-EVI in the DUT.

Then increment the scale of N by 5% of N till the limit is reached.

The limit is where the DUT can't learn any EVPN routes from its peers.

Measurement :

There should not be any loss of route types 2, 3 and 4 in DUT. DUT must relearn all type 2, 3 and 4 from remote routers. The DUT must be subjected to various values of N to find the optimal scale limit. The scope of the test is find out the maximum evpn instance that a DUT can hold. The measurement is carried out using external server which polls the DUT using automated scripts to find the scale limit of PBB-EVPN instances.

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:

Confirm the DUT is up and running with PBB-EVPN.

Scale N PBB-EVIs in DUT, SHPE3 and MHPE2.

Send F frames to DUT from CE using traffic generator with X different source and destination MAC addresses for N PBB-EVI's.

Send F frames from traffic generator to SHPE3 with X different source and destination MAC addresses.

There will be 2X number of MAC addresses 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 BGP session is in established state in DUT.

Measure the time taken to learn 2X MAC address in DUT MAC table.

Measurement :

The DUT must learn 2X MAC addresses. Measure the time taken to learn 2X MAC in DUT. 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. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1, T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts.

Time taken to learn 2X MAC in DUT = $(T1+T2+...Tn)/N$

4.11. Soak Test

Objective:

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

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with PBB-EVPN.

Scale N PBB-EVI's in DUT, SHPE3 and MHPE2. Send F frames to DUT from CE using traffic generator with different X source and destination MAC addresses for N EVI's.

Send F frames from traffic generator to SHPE3 with X different source and destination MAC addresses.

There will be 2X number of MAC addresses 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 for memory leak in PBB-EVPN process, CPU usage and crashes in DUT.

Measurement :

Take the hourly reading of CPU, process memory. There should not be any leak, crashes, CPU spikes. The CPU spike is determined as the CPU usage which shoots at 40 to 50 percent of the average usage. The average value vary from device to device. Memory leak is determined by increase usage of the memory for PBB-EVPN process. The expectation is under steady state the memory usage for PBB-EVPN process should not increase. The measurement is carried out using external server which polls the DUT using automated scripts which captures the CPU usage and process memory.

5. Acknowledgments

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. We take the opportunity to thank Al for reviewing our draft and gave us valuable comments.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

The benchmarking tests described in this document are limited to the performance characterization of controllers in a lab environment with isolated networks. 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 controller. Special capabilities SHOULD NOT exist in the controller specifically for benchmarking purposes. Any implications for network security arising from the controller SHOULD be identical in the lab and in production networks.

8. References

8.1. 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>>.
- [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>>.
- [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>>.

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: May 3, 2021

B. Balarajah
C. Rossenhoevel
EANTC AG
B. Monkman
NetSecOPEN
October 30, 2020

Benchmarking Methodology for Network Security Device Performance
draft-ietf-bmwg-ngfw-performance-05

Abstract

This document provides benchmarking terminology and methodology for next-generation network security devices including next-generation firewalls (NGFW), next-generation intrusion detection and prevention systems (NGIDS/NGIPS) 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, test configuration parameters, and benchmarking methodology for NGFW and NGIDS/NGIPS 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 May 3, 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	4
2. Requirements	4
3. Scope	4
4. Test Setup	4
4.1. Testbed Configuration	4
4.2. DUT/SUT Configuration	6
4.2.1. Security Effectiveness Configuration	11
4.3. Test Equipment Configuration	12
4.3.1. Client Configuration	12
4.3.2. Backend Server Configuration	14
4.3.3. Traffic Flow Definition	15
4.3.4. Traffic Load Profile	16
5. Test Bed Considerations	17
6. Reporting	18
6.1. Key Performance Indicators	20
7. Benchmarking Tests	21
7.1. Throughput Performance With Application Traffic Mix	21
7.1.1. Objective	21
7.1.2. Test Setup	21
7.1.3. Test Parameters	21
7.1.4. Test Procedures and Expected Results	23
7.2. TCP/HTTP Connections Per Second	24
7.2.1. Objective	24
7.2.2. Test Setup	24
7.2.3. Test Parameters	24
7.2.4. Test Procedures and Expected Results	26
7.3. HTTP Throughput	27
7.3.1. Objective	27
7.3.2. Test Setup	27
7.3.3. Test Parameters	27
7.3.4. Test Procedures and Expected Results	29
7.4. TCP/HTTP Transaction Latency	30
7.4.1. Objective	30
7.4.2. Test Setup	30
7.4.3. Test Parameters	30
7.4.4. Test Procedures and Expected Results	32

7.5.	Concurrent TCP/HTTP Connection Capacity	33
7.5.1.	Objective	33
7.5.2.	Test Setup	34
7.5.3.	Test Parameters	34
7.5.4.	Test Procedures and Expected Results	35
7.6.	TCP/HTTPS Connections per Second	36
7.6.1.	Objective	36
7.6.2.	Test Setup	37
7.6.3.	Test Parameters	37
7.6.4.	Test Procedures and Expected Results	38
7.7.	HTTPS Throughput	40
7.7.1.	Objective	40
7.7.2.	Test Setup	40
7.7.3.	Test Parameters	40
7.7.4.	Test Procedures and Expected Results	42
7.8.	HTTPS Transaction Latency	43
7.8.1.	Objective	43
7.8.2.	Test Setup	43
7.8.3.	Test Parameters	43
7.8.4.	Test Procedures and Expected Results	45
7.9.	Concurrent TCP/HTTPS Connection Capacity	46
7.9.1.	Objective	46
7.9.2.	Test Setup	46
7.9.3.	Test Parameters	47
7.9.4.	Test Procedures and Expected Results	48
8.	IANA Considerations	49
9.	Security Considerations	50
10.	Contributors	50
11.	Acknowledgements	50
12.	References	50
12.1.	Normative References	50
12.2.	Informative References	51
Appendix A.	Test Methodology – Security Effectiveness Evaluation	51
A.1.	Test Objective	51
A.2.	Testbed setup	52
A.3.	Test Parameters	52
A.3.1.	DUT/SUT Configuration Parameters	52
A.3.2.	Test Equipment Configuration Parameters	52
A.4.	Test Results Validation Criteria	52
A.5.	Measurement	53
A.6.	Test Procedures and expected Results	54
A.6.1.	Step 1: Background traffic	54
A.6.2.	Step 2: CVE emulation	54
Authors' Addresses	54

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 as they enable fair and reasonable comparison of network security functions. All these reasons have led to the creation of a new next-generation security device benchmarking document.

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 next-generation security devices. It covers the validation of security effectiveness configurations of the 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 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 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.

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 (Neighbor Discovery) 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 a test equipment port aggregation, it is RECOMMENDED to configure the test setup as shown in Figure 2.

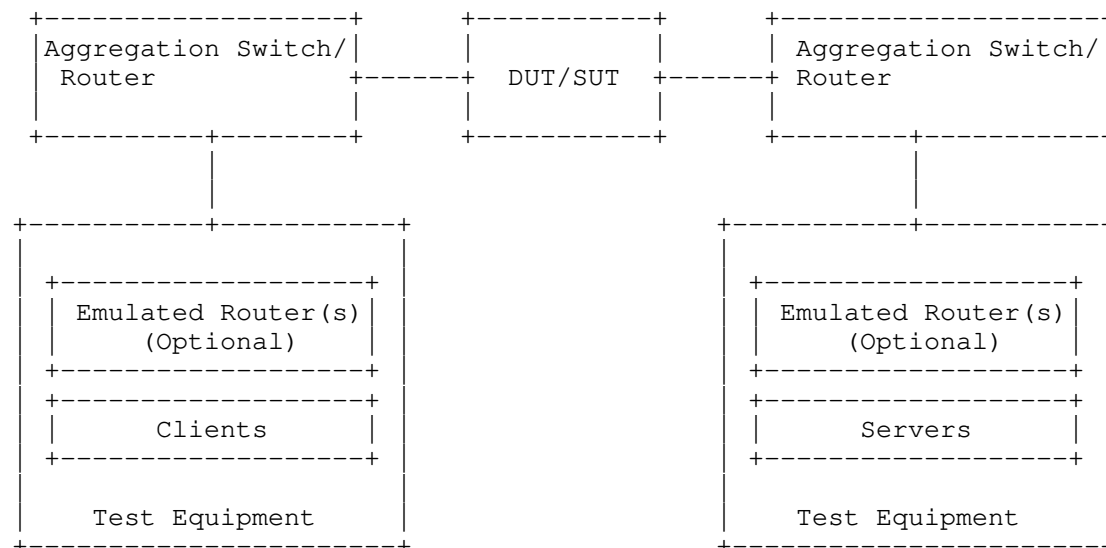


Figure 1: Testbed Setup - Option 1

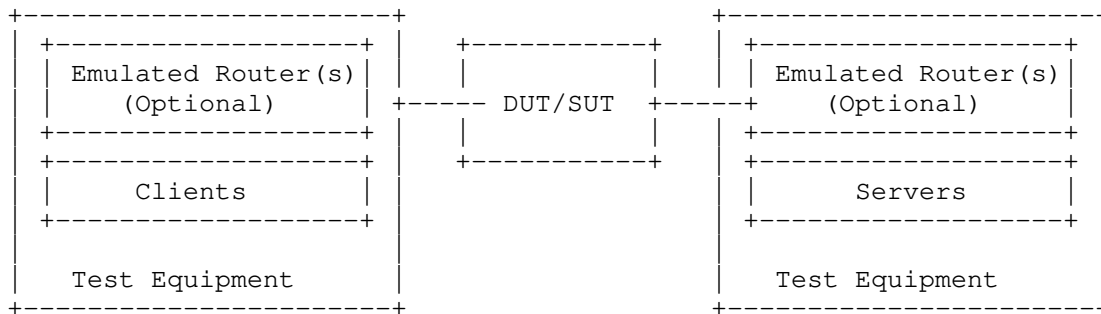


Figure 2: Testbed Setup - Option 2

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 and Table 2 below describe the sets of security feature list for NGFW and NGIDS/NGIPS that SHOULD be configured on the DUT/SUT respectively.

To improve repeatability, a summary of the DUT configuration including a 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	
Anti Spyware	x	
Antivirus	x	
Anti Botnet	x	
Web Filtering		x
DLP		x
DDoS		x
Certificate Validation		x
Logging and Reporting	x	
Application Identification	x	

Table 1: NGFW Security Features

DUT Features	NGIDS/NGIPS	
	RECOMMENDED	OPTIONAL
SSL Inspection	x	
Anti Spyware	x	
Antivirus	x	
Anti Botnet	x	
Logging and Reporting	x	
Application Identification	x	
Deep Packet Inspection	x	
Anti Evasion	x	

Table 2: NGIDS/NGIPS Security Features

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

DUT/SUT Features	Description
SSL Inspection	DUT/SUT intercept and decrypt inbound HTTPS traffic between servers and clients. Once the content inspection has been completed, DUT/SUT MUST encrypt the HTTPS traffic with ciphers and keys used by the clients and servers.
IDS/IPS	DUT/SUT MUST detect and block exploits targeting known and unknown vulnerabilities across the monitored network.
Anti Malware	DUT/SUT MUST detect and prevent 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 MUST detect and block initial infection or transmission of data.
Anti Botnet	DUT/SUT MUST detect traffic to or from botnets.
Anti Evasion	DUT/SUT MUST detect and mitigate attacks that have been obfuscated in some manner.
Web Filtering	DUT/SUT MUST detect and block malicious website including defined classifications of website across the monitored network.
DLP	DUT/SUT MUST detect and block the transmission of Personally Identifiable Information (PII) and specific files across the monitored network
Certificate Validation	DUT/SUT MUST validate certificates used in encrypted communications across the monitored network.
Logging and Reporting	DUT/SUT MUST be able to log and report all traffic at the flow level across the monitored.
Application Identification	DUT/SUT MUST detect known applications as defined within the traffic mix selected across the monitored network.

Table 3: Security Feature Description

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

- o All RECOMMENDED security inspection enabled
- o Disposition of all flows of traffic are logged - Logging to an external device is permissible
- 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, a realistic number of access control rules (ACL) MUST be configured on the DUT/SUT. However, this is applicable only for the security devices where ACL's are configurable and also the ACL configuration on NGIDS/NGIPS devices is OPTIONAL. This document determines the number of access policy rules for four 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 datasheet. 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 4 MUST be configured from top to bottom in the correct order as shown in the table. The ACL entries MUST be configured in Forward Information Base (FIB) table of 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 performance 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 4: DUT/SUT Access List

4.2.1. Security Effectiveness Configuration

The Security features (defined in table 1 and 2) of the DUT/SUT MUST be configured effectively in such a way 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 MUST 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 mainly focused on performance benchmarking. However, it is strongly RECOMMENDED to validate the security configuration of the DUT/SUT by evaluating the security effectiveness as a Prerequisite for performance benchmarking tests defined in the section 7. 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 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. The 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.

- o The IP blocks SHOULD consist of multiple unique, discontinuous static address blocks.
- o A default gateway is permitted.
- o The IPv4 Type of Service (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.

Desired total number of client IP = Target throughput [Mbit/s] /
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 client 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 persistence MAY be enabled depending on the 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 SHOULD 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 ciphers and keys are RECOMMENDED to use for HTTPS based benchmarking tests defined in Section 7.

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`)

Note: The above ciphers and keys were those commonly used enterprise grade encryption cipher suites. 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.

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.

- o 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.
- o A default gateway is permitted. The IPv4 ToS byte and IPv6 traffic class bytes should be set to '00' and '000000' respectively.
- o The server IP addresses SHOULD be distributed between IPv4 and IPv6 with a ratio identical to the clients distribution ratio.

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. The HTTPS server MUST check Host SNI information with the FQDN if the 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, 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 MUST use the servers IP address or Fully Qualified Domain Names (FQDN) in Host Headers. For TLS the client MAY use 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 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 service 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 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, 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 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, throughput, concurrent TCP connections, and application transactions per second. 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 minimum RECOMMENDED time duration for sustain phase is 300 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 the same.
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 testbed 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.

Testbed 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.

Testbed preparation may be performed either by configuring the DUT in the most trivial setup (fast forwarding) or without presence of the 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 time and date of the execution of the test MUST be prominent.

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 MUST be documented.
- + 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, and the resource sharing (e.g. Pinning details and NUMA Node) configuration MUST be documented. The virtual components such as Hypervisor, virtual switch version MUST be also documented.
- + Any additional hardware relevant to the DUT/SUT such as controllers MUST be documented

B. DUT/SUT Software

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

C. DUT/SUT Enabled Features

- + Configured DUT/SUT features (see Table 1 and Table 2) 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
- 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 test scenario
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

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 key performance indicators (KPIs) for overall benchmarking test 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 KPI measures the average concurrent open TCP connections in the sustaining period.
- o TCP Connections Per Second
This KPI measures the average established TCP connections per second in the sustaining period. Also this KPI measures average established and terminated TCP connections per second simultaneously for the test scenarios "TCP/HTTP(S) Connection Per Second" defined in Section 7.2 and Section 7.6.
- o Application Transactions Per Second
This KPI measures the average successfully completed application transactions per second in the sustaining period.
- o TLS Handshake Rate
This KPI measures the average TLS 1.2 or higher session formation rate within the sustaining period.
- o Throughput
This KPI 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 KPI 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 Time to First Byte (TTFB)
This KPI 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 Application Traffic Mix

7.1.1. Objective

Using a relevant application traffic mix, determine the maximum sustainable throughput performance supported by the DUT/SUT.

Based on customer use case, users can choose the 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 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

7.1.2. Test Setup

Testbed 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, the 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. In case the DUT is configured without SSL inspection feature, 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. 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 ciphers and keys defined in Section 4.3.1.3 are RECOMMENDED to use for this test scenario.

7.1.3.3. Traffic Profile

Traffic profile: Test scenario MUST be run with a relevant application traffic mix profile.

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

7.1.3.5. Measurement

Following KPI metrics MUST be reported for this test scenario.

Mandatory KPIs: average Throughput, TTFB (minimum, average, and maximum), TTLB (minimum, average, and maximum) and average Application Transactions Per Second

Note: TTLB MUST be reported along with min, max, and avg object size used in the traffic profile.

Optional KPIs: average TCP Connections Per Second and average TLS Handshake Rate

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 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 datasheet (if known)

Initial connections per second: 10% of "Target connections per second" (an optional parameter for documentation)

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 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 almost at the same rate

7.2.3.4. Measurement

Following KPI metric MUST be reported for each test iteration.

average TCP Connections Per Second

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 datasheet (if known)

Initial Throughput: 10% of "Target Throughput" (an optional parameter for documentation)

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 4: 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 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 almost at the same rate

7.3.3.4. Measurement

The KPI metrics MUST be reported for this test scenario:

average Throughput and average HTTP Transactions 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 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" (an optional parameter for documentation)

Initial objective for scenario 2: 10% of "Target objective for scenario 2" (an optional parameter for documentation)

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 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 almost 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:

TTFB (minimum, average and maximum) and TTLB (minimum, average and maximum)

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 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 datasheet (if known)

Initial concurrent connection: 10% of "Target concurrent connection" (an optional parameter for documentation)

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 metric MUST be reported for this test scenario:

average Concurrent TCP Connections

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 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 datasheet (if known)

Initial connections per second: 10% of "Target connections per second" (an optional parameter for documentation)

RECOMMENDED ciphers and keys defined in Section 4.3.1.3

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 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 almost 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 TLS Handshake Rate (TLS Handshake Rate can be measured in the test scenario using 1KB 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. 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 datasheet (if known)

Initial Throughput: 10% of "Target Throughput" (an optional parameter for documentation)

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 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 5: 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. 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 almost at the same rate

7.7.3.4. Measurement

The KPI metrics MUST be reported for this test scenario:

average Throughput and average HTTPS Transactions 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 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 sizes defined in Section 4.3.1.3

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" (an optional parameter for documentation)

Initial objective for scenario 2: 10% of "Target objective for scenario 2" (an optional parameter for documentation)

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 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 almost 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:

TTFB (minimum, average and maximum) and TTLB (minimum, average and maximum)

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 TTFB 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 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 sizes defined in Section 4.3.1.3

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

Initial concurrent connections: 10% of "Target concurrent connections" (an optional parameter for documentation)

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 metric MUST be reported for this test scenario:

average Concurrent TCP Connections

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. IANA Considerations

This document makes no request of IANA.

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

9. 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 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.

10. Contributors

The following individuals contributed significantly to the creation of this document:

Alex Samonte, Amritam Putatunda, Aria Eslambolchizadeh, David DeSanto, Jurrie Van Den Brekel, Ryan Liles, Samaresh Nair, Stephen Goudreault, 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, Baski Mohan, Chao Guo, Chris Brown, Chris Marshall, Jay Lindenauer, Michael Shannon, Mike Deichman, Ray Vinson, Ryan Riese, Tim Carlin, Tim Otto and Toulmay 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

- [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. 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 in order 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.

- o Number of blocked CVEs
- o Number of bypassed (nonblocked) CVEs
- o Background traffic performance (verify if the background traffic is impacted while sending CVE toward DUT/SUT)
- o 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 test scenario 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 test scenario:

- o 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 test scenarios "HTTP(S) Throughput" defined in Section 7.3 and Section 7.7.
- o RECOMMENDED CVE traffic transmission Rate: 10 CVEs per second
- o RECOMMEND to generate each CVE multiple times (sequentially) at 10 CVEs per second
- o Ciphers and Keys for the encrypted CVE traffic MUST use the same cipher configured for HTTPS traffic related benchmarking test scenarios (Section 7.6 - Section 7.9)

A.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 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
- 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:

- o Blocked CVEs: It should be represented in the following ways:
 - * Number of blocked CVEs out of total CVEs
 - * Percentage of blocked CVEs
- o Unblocked CVEs: It should be represented in the following ways:
 - * Number of unblocked CVEs out of total CVEs
 - * Percentage of unblocked CVEs
- o Background traffic behavior: it should represent one of the followings ways:
 - * No impact (traffic transmission at a constant rate)
 - * Minor impact (e.g. small spikes- +/- 100 Mbit/s)
 - * Heavily impacted (e.g. large spikes and reduced the background throughput > 100 Mbit/s)
- o DUT/SUT reporting accuracy: DUT/SUT MUST report all detected vulnerabilities.

Optional KPIs:

- o 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 the background traffic at the transmission rate defined in the parameter section.

The DUT/SUT MUST reach the target objective (throughput) in sustain phase. The measured KPIs during the sustain phase MUST meet the test validation criteria a, b, c and d 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 the 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. The frequency of measurement MUST be less than 2 seconds. Continue the test until all CVEs are sent.

The measured KPIs MUST meet all test validation criteria a, b, c, and d defined in Appendix A.4.

In addition, the DUT/SUT SHOULD report the vulnerabilities correctly.

Authors' Addresses

Balamuhunthan Balarajah
Berlin
Germany

Email: bm.balarajah@gmail.com

Carsten Rossenhoevel
EANTC AG
Salzufer 14
Berlin 10587
Germany

Email: cross@eantc.de

Brian Monkman
NetSecOPEN
417 Independence Court
Mechanicsburg, PA 17050
USA

Email: bmonkman@netsecopen.org

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 29, 2021

S. Jacob, Ed.
K. Tiruveedhula
Juniper Networks
October 26, 2020

Benchmarking Methodology for EVPN VPWS
draft-kishjac-bmwg-evpnpwstest-05

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, Fail over, 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 April 29, 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
1.1. Requirements Language	2
1.2. Terminologies	2
2. Test Topology	4
3. Test Cases	6
3.1. Local Failure Scenario 1	7
3.2. Local Failure Scenario 2	7
3.3. Core Failure	8
3.4. Link Flap	9
4. Scale Convergence	9
4.1. To measure the packet loss during the core link failure.	9
5. High Availability	10
5.1. To Record the whether there is traffic loss due to routing engine failover for redundancy test.	10
6. SOAK Test	11
6.1. To Measure the stability of the DUT with scale and traffic.	11
7. Acknowledgements	12
8. IANA Considerations	12
9. Security Considerations	12
10. References	12
10.1. Normative References	12
10.2. Informative References	12
Appendix A. Appendix	13
Authors' Addresses	13

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

All-Active Redundancy Mode: When all PEs attached to an Ethernet segment are allowed to forward known unicast traffic to/from that

Ethernet segment for a given VLAN, then the Ethernet segment is defined to be operating in All-Active redundancy mode.

AA: All Active mode

AC: Attachment Circuits

CE: Customer Router/Devices/Switch.

DF: Designated Forwarder

DUT: Device under test.

Ethernet Segment (ES): When a customer site (device or network) is connected to one or more PEs via a set of Ethernet links, then that set of links is referred to as an 'Ethernet segment'.

EVI: An EVPN instance spanning the Provider Edge (PE) devices participating in that EVPN.

Ethernet Segment Identifier (ESI): A unique non-zero identifier that identifies an Ethernet segment is called an 'Ethernet Segment Identifier'.

Ethernet Tag: An Ethernet tag identifies a particular broadcast domain, e.g., a VLAN. An EVPN instance consists of one or more broadcast domains.

Interface: Physical interface of a router/switch.

IRB: Integrated routing and bridging interface

MAC: Media Access Control addresses on a PE.

MHPE2: Multi homed Provider Edge router 2.

MHPE1: Multi homed Provider Edge router 1.

SHPE3: Single homed Provider Edge Router 3.

PE: Provider Edge device.

P: Provider Router.

RR: Route Reflector.

RT: Traffic Generator.

Sub Interface: Each physical Interfaces is subdivided into Logical units.

SA: Single Active

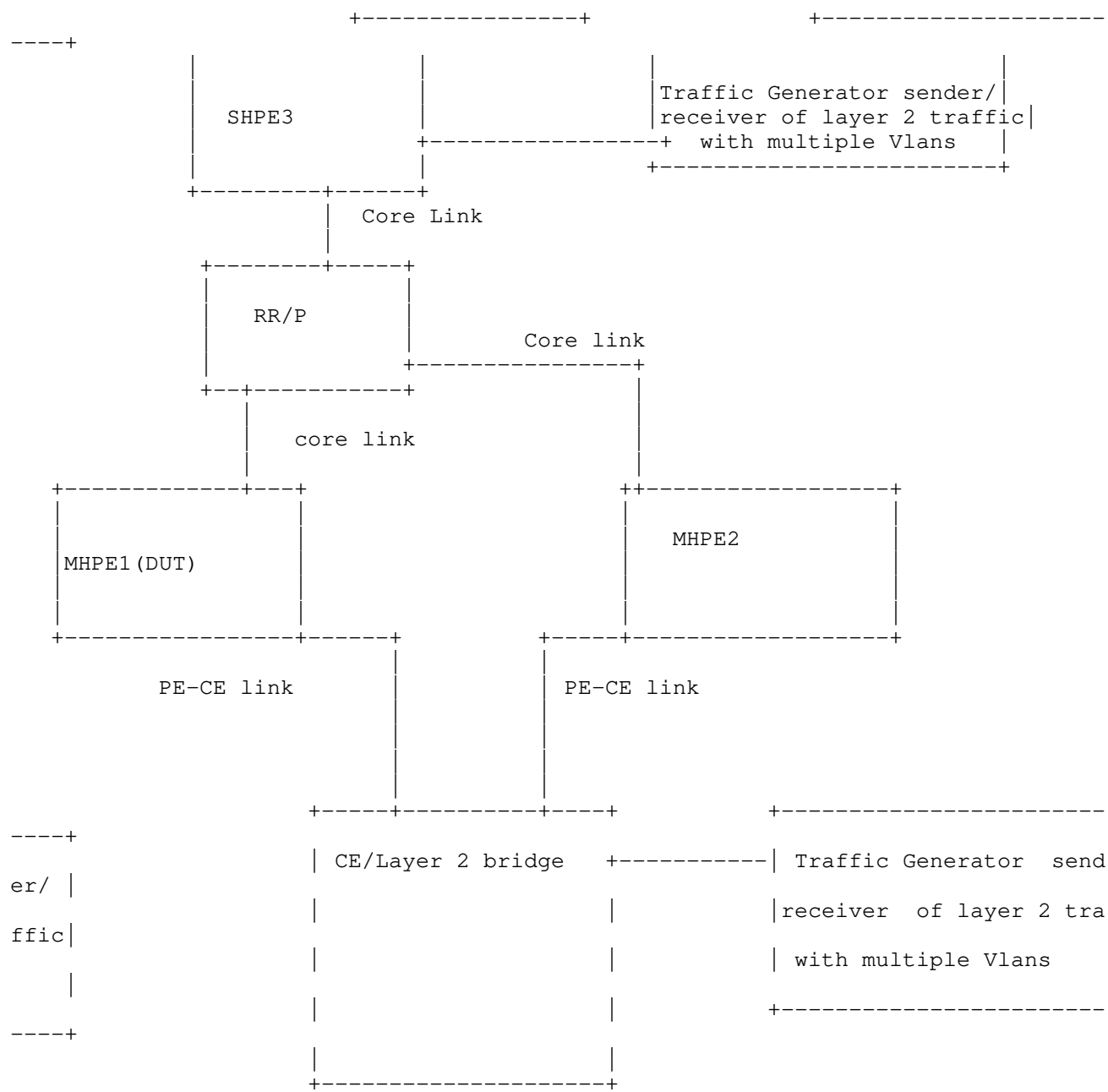
Single-Active Redundancy Mode: When only a single PE, among all the PEs attached to an Ethernet segment, is allowed to forward traffic to/from that Ethernet segment for a given VLAN, then the Ethernet segment is defined to be operating in Single-Active redundancy mode.

VPWS: Virtual private wire service.

2. Test Topology

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 traffic generator is connected to CE and SHPE3. The MHPE1 acts as DUT. The traffic generator will be used as sender and receiver of traffic. The DUT will be the reference point for all the test cases. MHPE1 and MHPE2 are multihome routers connected to CE running single active mode. The traffic generator will be generating traffic at 10% of the line rate.

Topology Diagram



Topology 1

Topology Diagram

Figure 1

Test Setup Configurations:

SHPE3 is configured with Interior Gateway protocols like OPSF or IS-IS for underlay, LDP for MPLS support, Interior Border Gateway with EVPN address family for overlay support. This router must be configured with N EVPN-VPWS instances for testing. Traffic generator is connected to this router for sending and receiving traffic.

RR is configured with Interior Gateway protocols like OPSF or IS-IS for underlay, LDP for MPLS support, Interior Border Gateway with EVPN address family for overlay support. This router acts as a provider router and as a route reflector.

MHPE1 is configured with Interior Gateway protocols like OPSF or IS-IS for underlay, LDP for MPLS support, Interior Border Gateway with EVPN address family for overlay support. This router must be configured with N EVPN-VPWS instances for testing. This router is configured with ESI per vlan or ESI per interface. It is functioning as multi homing PE working on Single Active EVPN mode. This router serves as the DUT and it is connected to CE. MHPE1 is acting as DUT for all the test cases.

MHPE2 is configured with Interior Gateway protocols like OPSF or IS-IS for underlay, LDP for MPLS support, Interior Border Gateway with EVPN address family for overlay support. This router must be configured with N EVPN-VPWS instances for testing. This router is configured with ESI per vlan or ESI per interface. It is functioning as multi homing PE working on Single Active EVPN mode. It is connected to CE.

CE is acting as bridge configured with multiple vlans, the same vlans are configured on MHPE1, MHPE2, SHPE3. traffic generator is connected to CE. The traffic generator acts as sender or receiver of traffic.

Depending up on the test scenarios the traffic generators will be used to generate uni directional or bi directional flows.

The above configuration will be serving as the base configuration for all 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. Local Failure Scenario 1

Objective:

Measure the time taken to switch from primary to backup during local link failure.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN-VPWS. The AC must be up and running. "N" AC's in MHPE1, MHPE2, working in SA mode. Ensure DUT is active and MHPE2 is backup PE. Send unicast packets to CE from traffic generator. The traffic is uni directional and it flows from CE to DUT working as Active router. Then shut the DUT-CE link, so that traffic from CE switches to MHPE2. Traffic must be tested with various line rate that from 10% to 98%.

Measurement :

Measure the time taken by the traffic to switch from Active router to the backup. The test is repeated for "N" times and the values are collected. The AC's local switch over time is calculated by averaging the values obtained by "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1, T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts. Fail over time must be measured for various line rate.

AC's switch over from primary to backup PE in sec = $(T1+T2+...Tn/N)$

3.2. Local Failure Scenario 2

Objective:

Measure time taken by remote PE to switch traffic from primary to backup during CE link failure.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN-VPWS. The AC must be up and running. "N" AC's in MHPE1, MHPE2, working in SA mode. Ensure DUT is active and MHPE2 is backup PE. Send unicast packets to SHPE3 from traffic generator. The traffic is uni directional and it flows from

SHPE3 to DUT working as Active router. Then shut the DUT-CE link, so the remote traffic flow switches from DUT to MHPE2. Traffic must be tested with various line rate that from 10% to 98%.

Measurement :

Measure the time taken by the traffic to switch from Active router to the backup. The test is repeated for "N" times and the values are collected. The AC's switch over time for the remote traffic is calculated by averaging the values obtained by "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn.The measurement is carried out using external server which polls the DUT using automated scripts. Fail over time must be measured for various line rate.

AC's switch over from primary to backup PE in sec = $(T1+T2+..Tn/N)$

3.3. Core Failure

Objective:

Measure the time taken by remote PE to switch traffic from primary to backup during core link failure.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN-VPWS. The AC must be up and running. "N" AC's in MHPE1,MHPE2, working in SA mode.Ensure DUT is active and MHPE2 is backup PE.Send unicast packets to SHPE3 from traffic generator. The traffic is uni directional and it flows from SHPE3 to DUT working as Active router. Then shut the DUT core link, so the remote traffic flow switches from DUT to MHPE2. Traffic must be tested with various line rate that from 10% to 98%.

Measurement :

Measure the time taken by the traffic to switch from Active router to the backup. The test is repeated for "N" times and the values are collected. The AC's switch over time for the remote traffic is calculated by averaging the values obtained by "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn.The measurement is carried out using external server which polls the DUT using automated scripts. Fail over time must be measured for various line rate.

AC's core Failure fail over time = $(T1+T2+...Tn/N)$

3.4. Link Flap

Objective:

Measure time taken by primary PE to regain control after the local PE-CE link flap.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN-VPWS. The AC must be up and running. "N" AC's in MHPE1, MHPE2, working in SA mode. Ensure DUT is active and MHPE2 is backup PE. Send unicast packets to CE from traffic generator. The traffic is uni directional and it flows from CE to DUT working as Active router. Then shut the DUT core link, so the local traffic flow switches from DUT to MHPE2. Once the fail over is performed. Bring the link up. Now the DUT becomes the Active router. Measure time taken by the DUT to regain the traffic. Traffic must be tested with various line rate that from 10% to 98%.

Measurement :

Measure the time taken by the traffic to switch back to the DUT. The test is repeated for "N" times and the values are collected. The AC's switch over time for the remote traffic is calculated by averaging the values obtained by "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1, T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts. Fail over time must be measured for various line rate.

Time taken to switch back to primary (DUT) once the link is restored = $(T1+T2+...Tn/N)$

4. Scale Convergence

4.1. To measure the packet loss during the core link failure.

Objective:

Measure the convergence at a higher number of AC's

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN-VPWS. The AC must be up and running. "N*100" AC's in MHPE1, MHPE2, working in SA mode. Ensure DUT is active and MHPE2 is backup PE. Send unicast packets to CE from traffic generator and send traffic from traffic generator to SHPE3. The traffic is directional and it flows from CE to DUT and from DUT to CE, working as Active router. Then shut the DUT core link, so the traffic flow switches from DUT to MHPE2. Measure traffic switching time. Traffic must be tested with various line rate that from 10% to 98%.

Measurement :

Measure the time taken by the traffic to switch from DUT to MHPE2. The test is repeated for "N" times and the values are collected. The AC's switch over time for the traffic is calculated by averaging the values obtained by "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1, T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts. Fail over time must be measured for various line rate.

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

5. High Availability

5.1. To Record the whether there is traffic loss due to routing engine failover for redundancy test.

Objective:

Measure the traffic loss during routing engine fail over.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN-VPWS. The AC must be up and running. "N*100" AC's in MHPE1, MHPE2, working in SA mode. Ensure DUT is active and MHPE2 is backup PE. Send unicast packets to CE and SHPE3 from traffic generator. The traffic is directional and it flows from CE to DUT and from DUT to CE, working as Active router. Do a routing engine fail over once the traffic is stabilized in DUT. Traffic must be tested with various line rate that from 10% to 98%. The expectation is 0 packet loss, no role change in AC's.

Measurement :

The expectation of the test is 0 traffic loss with no change in the DF role. DUT should not withdraw any routes. But in cases where the DUT is not properly synchronized between master and standby, due to that packet loss are observed. In that scenario the packet loss is measured. The test is repeated for "N" times and the values are collected. The packet loss is calculated by averaging the values obtained by "N" samples.

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

6. SOAK Test

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.

6.1. 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 1

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. The CPU spike is determined as the CPU usage which shoots at 40 to 50 percent of the average usage. The average value vary from device to device. Memory leak is determined by increase usage of the memory for EVPN-VPWS process. The expectation is under steady state the memory usage for EVPN-VPWS process should not increase.

7. Acknowledgements

We would like to thank Al and Sarah for the support.

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

The benchmarking tests described in this document are limited to the performance characterization of controllers in a lab environment with isolated networks. 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 controller. Special capabilities SHOULD NOT exist in the controller specifically for benchmarking purposes. Any implications for network security arising from the controller SHOULD be identical in the lab and in production networks.

10. References

10.1. 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>>.

10.2. 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 Methodology Working Group
Internet-Draft
Intended status: Informational
Expires: November 21, 2020

G. Lencse
BUTE
K. Shima
IIJ-II
May 20, 2020

An Upgrade to Benchmarking Methodology for Network Interconnect Devices
draft-lencse-bmwg-rfc2544-bis-00

Abstract

RFC 2544 has defined a benchmarking methodology for network interconnect devices. We recommend a few upgrades to it for producing more reasonable results. The recommended upgrades can be classified into two categories: the application of the novelties of RFC 8219 for the legacy RFC 2544 use cases and the following new ones. Checking a reasonably small timeout individually for every single frame in the throughput and frame loss rate benchmarking procedures. Performing a statistically relevant number of tests for all benchmarking procedures. Addition of an optional non-zero frame loss acceptance criterion for the throughput measurement procedure and defining its reporting format.

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 November 21, 2020.

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
1.1. Requirements Language	3
2. Recommendation to Backport the Novelities of RFC8219	3
3. Improved Throughput and Frame Loss Rate Measurement Procedures using Individual Frame Timeout	3
4. Requirement of Statistically Relevant Number of Tests	4
5. An Optional Non-zero Frame Loss Acceptance Criterion for the Throughput Measurement Procedure	5
6. Acknowledgements	6
7. IANA Considerations	6
8. Security Considerations	7
9. References	7
9.1. Normative References	7
9.2. Informative References	7
Appendix A. Change Log	8
A.1. 00	8
Authors' Addresses	8

1. Introduction

[RFC2544] has defined a benchmarking methodology for network interconnect devices. [RFC5180] addressed IPv6 specificities and also added technology updates, but declared IPv6 transition technologies out of its scope. [RFC8219] addressed the IPv6 transition technologies, and it added further measurement procedures (e.g. for packet delay variation (PDV) and inter packet delay variation (IPDV)). It has also recommended to perform multiple tests (at least 20), and it proposed median as summarizing function and 1st and 99th percentiles as the measure of variation of the results of the multiple tests. This is a significant change compared to [RFC2544], which always used only average as summarizing function. [RFC8219] also redefined the latency measurement procedure with the requirement of marking at least 500 frames with identifying tags for latency measurements, instead of using only a single one. However, all these improvements apply only for the IPv6 transition technologies, and no update was made to [RFC2544] / [RFC5180], which we believe to be desirable.

Moreover, [RFC8219] has reused the throughput and frame loss rate benchmarking procedures from [RFC2544] with no changes. When we tested their feasibility with a few SIIT [RFC7915] implementations, we have pointed out three possible improvements in [LEN2020A]:

- o Checking a reasonably small timeout individually for every single frame with the throughput and frame loss rate benchmarking procedures.
- o Performing a statistically relevant number of tests for these two benchmarking procedures.
- o Addition of an optional non-zero frame loss acceptance criterion for the throughput benchmarking procedure and defining its reporting format.

1.1. 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 BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Recommendation to Backport the Novelties of RFC8219

Besides addressing IPv6 transition technologies, [RFC8219] has also made several technological upgrades reflecting the current state of the art of networking technologies and benchmarking. But all the novelties mentioned in Section 1 of this document currently apply only for the benchmarking of IPv6 transition technologies. We contend that they could be simply backported to the benchmarking of network interconnect devices. For example, siitperf [SIITPERF], our [RFC8219] compliant DPDK-based software Tester was designed for benchmarking different SIIT [RFC7915] (also called stateless NAT64) implementations, but if it is configured to have the same IP version on both sides, it can be used to test IPv4 or IPv6 (or dual stack) routers [LEN2020B]. We highly recommend the backporting of the latency, PDV and IPDV benchmarking measurement procedures of [RFC8219].

3. Improved Throughput and Frame Loss Rate Measurement Procedures using Individual Frame Timeout

The throughput measurement procedure defined in [RFC2544] only counts the number of the sent and received test frames, but it does not identify the test frames individually. On the one hand, this approach allows the Tester to send always the very same test frame to

the DUT, which was very likely an important advantage in 1999. However, on the other hand, thus the Tester cannot check if the order of the frames is kept, or if the frames arrive back to the Tester within a given timeout time. (Perhaps none of them was an issue of hardware based network interconnect devices in 1999. But today network packet forwarding and manipulation is often implemented in software having larger buffers and producing potentially higher latencies.)

Whereas real-time applications are obviously time sensitive, other applications like HTTP or FTP are often considered throughput hungry and time insensitive. However, we have demonstrated that when we applied 100ms delay to 1% of the test frames, the throughput of HTTP download dropped by more than 50% [LEN2020C]. Therefore, an advanced throughput measurement procedure that checks the timeout time for every single test frame may produce more reasonable results. We have shown that this measurement is now feasible [LEN2020B]. In this case, we used 64-bit integers to identify the test frames and measured the latency of the frames as required by the PDV measurement procedure in Section 7.3.1. of [RFC8219]. In our particular test, we used 10ms as frame timeout, which could be a suitable value, but we recommend further studies to determine the recommended timeout value.

We recommend that the reported results of the improved throughput and frame loss rate measurements SHOULD include the applied timeout value.

4. Requirement of Statistically Relevant Number of Tests

Section 4 of [RFC2544] says that: "Furthermore, selection of the tests to be run and evaluation of the test data must be done with an understanding of generally accepted testing practices regarding repeatability, variance and statistical significance of small numbers of trials." It is made a stronger requirement (by using a "MUST") in Section 3 of [RFC5180] stating that: "Test execution and results analysis MUST be performed while observing generally accepted testing practices regarding repeatability, variance, and statistical significance of small numbers of trials." But no practical guidelines are provided concerning the minimally necessary number of tests.

[RFC8219] mentions at four different places that the tests must be repeated at least 20 times. These places are the benchmarking procedures for:

- o latency (Section 7.2)
- o packet delay variation (Section 7.3.1)

- o inter packet delay variation (Section 7.3.2)
- o DNS64 performance (Section 9.2).

We believe that a similar guideline for the minimal number of tests would be helpful for the throughput and frame loss rate benchmarking procedures. We consider 20 as an affordable number of minimum repetitions of the frame loss rate measurements. However, as for throughput measurements, we contend that the binary search may require rather high number of steps in certain situations (e.g. tens of millions of frames per second rate and high resolution) that the requirement of at least 20 repetitions of the binary search would result in unreasonably high measurement execution times. Therefore, we recommend to use an algorithm that checks the statistical properties of the results of the tests and it may stop before 20 repetitions, if the results are consistent, but it may require more than 20 repetitions, if the results are scattered. (The algorithm is yet to be developed.)

5. An Optional Non-zero Frame Loss Acceptance Criterion for the Throughput Measurement Procedure

When we defined the measurement procedure for DNS64 performance in Section 9.2 of [RFC8219], we followed both spirit and wording of the [RFC2544] throughput measurement procedure including the requirement for absolutely zero packet loss. We have elaborated our underlying considerations in our research paper [LEN2017] as follows:

1. Our goal is a well-defined performance metric, which can be measured simply and efficiently. Allowing any packet loss would result in a need for scanning/trying a large range of rates to discover the highest rate of successfully processed DNS queries.
2. Even if users may tolerate a low loss rate (please note the DNS uses UDP with no guarantee for delivery), it cannot be arbitrarily high, thus, we could not avoid defining a limit. However, any other limits than zero percent would be hardly defensible.
3. Other benchmarking procedures use the same criteria of zero packet loss and this is the standard in IETF Benchmarking Methodology Working Group.

On the one hand, we still consider our arguments valid, however, on the other hand, we are aware of different arguments for the justification of an optional non-zero frame loss acceptance criterion, too:

- o Frame loss is present in our networks from the very beginning and our applications are well prepared to handle frame loss. They can definitely tolerate some low frame loss rates like 0.01% (1 frame from 10,000 frames).
- o It is a wide-spread practice among benchmarking professionals to allow a certain low rate of frame loss for a long time [TOL2001] and commercially available network performance testers allow to specify a parameter usually called as "Loss Tolerance" to express a zero or non-zero acceptance criterion for throughput measurements.
- o Today network packet forwarding and manipulation is often implemented in software. They do not work the same as the hardware-based forwarding devices, and may be affected by other processes running in the same host hardware. So it is not feasible to require 0% of frame loss in such forwarding devices.
- o Forwarding devices (especially but not necessarily only the software-based ones) may today also have larger buffers and thus they may produce potentially higher latencies. As we have shown in Section 3, late packets are not really useful for the applications, and thus they are to be considered as lost ones. For being strict with the latency during throughput measurements (e.g. 10ms timeout), we should make up with the loss tolerance to provide meaningful benchmarking results.
- o Likely due to the high frame loss rates can be experienced in WiFi networks, the latest development direction of TCP congestion control algorithms considers loss no more a sign of congestion (e.g. TCP BBR).

So we felt the necessity of having options to allow frame loss. Therefore, we recommend that throughput measurement with some low tolerated frame loss rates like 0.001% or 0.01% be a recognized optional test for network interconnect devices. To avoid the possibility of gaming, our recommendation is that the results of such tests MUST clearly state the applied loss tolerance rate.

6. Acknowledgements

The authors would like to thank ... (TBD)

7. IANA Considerations

This document does not make any request to IANA.

8. Security Considerations

We have no further security considerations beyond that of [RFC8219]. Perhaps they should be cited here so that they be applied not only for the benchmarking of IPv6 transition technologies, but also for the benchmarking of all network interconnect devices.

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>>.
- [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>>.
- [RFC7915] Bao, C., Li, X., Baker, F., Anderson, T., and F. Gont, "IP/ICMP Translation Algorithm", RFC 7915, DOI 10.17487/RFC7915, June 2016, <<https://www.rfc-editor.org/info/rfc7915>>.
- [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>>.
- [RFC8219] Georgescu, M., Pislaru, L., and G. Lencse, "Benchmarking Methodology for IPv6 Transition Technologies", RFC 8219, DOI 10.17487/RFC8219, August 2017, <<https://www.rfc-editor.org/info/rfc8219>>.

9.2. Informative References

- [LEN2017] Lencse, G., Georgescu, M., and Y. Kadobayashi, "Benchmarking Methodology for DNS64 Servers", Computer Communications, vol. 109, no. 1, pp. 162-175, DOI: 10.1016/j.comcom.2017.06.004, Sep 2017, <<http://www.hit.bme.hu/~lencse/publications/ECC-2017-B-M-DNS64-revised.pdf>>.

[LEN2020A]

Lencse, G. and K. Shima, "Performance analysis of SIIT implementations: Testing and improving the methodology", Computer Communications, vol. 156, no. 1, pp. 54-67, DOI: 10.1016/j.comcom.2020.03.034, Apr 2020, <<http://www.hit.bme.hu/~lencse/publications/ECC-2020-SIIT-Performance-published.pdf>>.

[LEN2020B]

Lencse, G., "Design and Implementation of a Software Tester for Benchmarking Stateless NAT64 Gateways", under second review in IEICE Transactions on Communications, <<http://www.hit.bme.hu/~lencse/publications/IEICE-2020-siitperf-revised.pdf>>.

[LEN2020C]

Lencse, G., Shima, K., and A. Kovacs, "Gaming with the Throughput and the Latency Benchmarking Measurement Procedures of RFC 2544", under review in International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems, <<http://www.hit.bme.hu/~lencse/publications/IJATES2-2020-Gaming-RFC2544-for-review.pdf>>.

[SIITPERF]

Lencse, G. and Y. Kadobayashi, "Siitperf: An RFC 8219 compliant SIIT (stateless NAT64) tester written in C++ using DPDK", source code, available from GitHub, 2019, <<https://github.com/lencsegabor/siitperf>>.

[TOL2001]

Tolly, K., "The real meaning of zero-loss testing", IT World Canada, 2001, <<https://www.itworldcanada.com/article/kevin-tolly-the-real-meaning-of-zero-loss-testing/33066>>.

Appendix A. Change Log

A.1. 00

Initial version.

Authors' Addresses

Gabor Lencse
Budapest University of Technology and Economics
Magyar Tudosok korutja 2.
Budapest H-1117
Hungary

Email: lencse@hit.bme.hu

Keiichi Shima
IIJ Innovation Institute
Iidabashi Grand Bloom, 2-10-2 Fujimi
Chiyoda-ku, Tokyo 102-0071
Japan

Email: keiichi@iijlab.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:

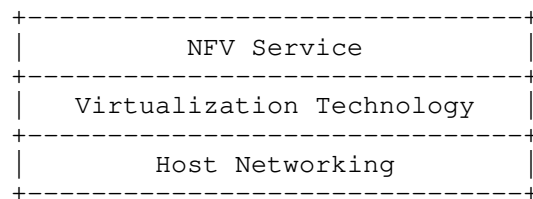


Figure 1. NFV software technology stack.

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:

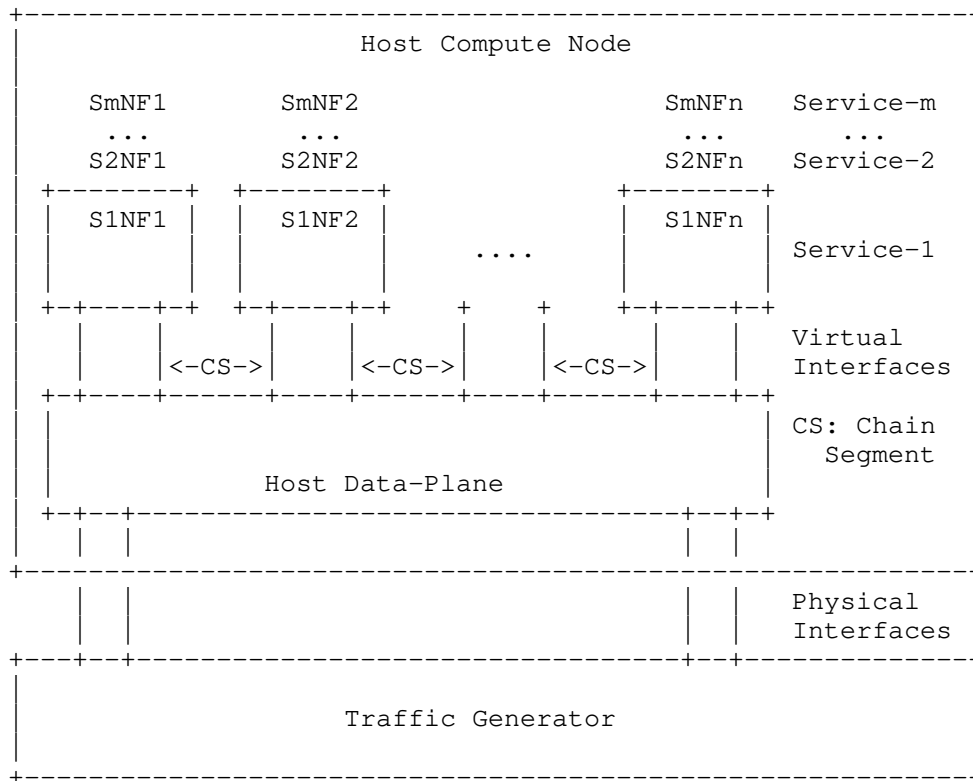


Figure 2. NF chain topology forming a service instance.

NF pipeline topology:

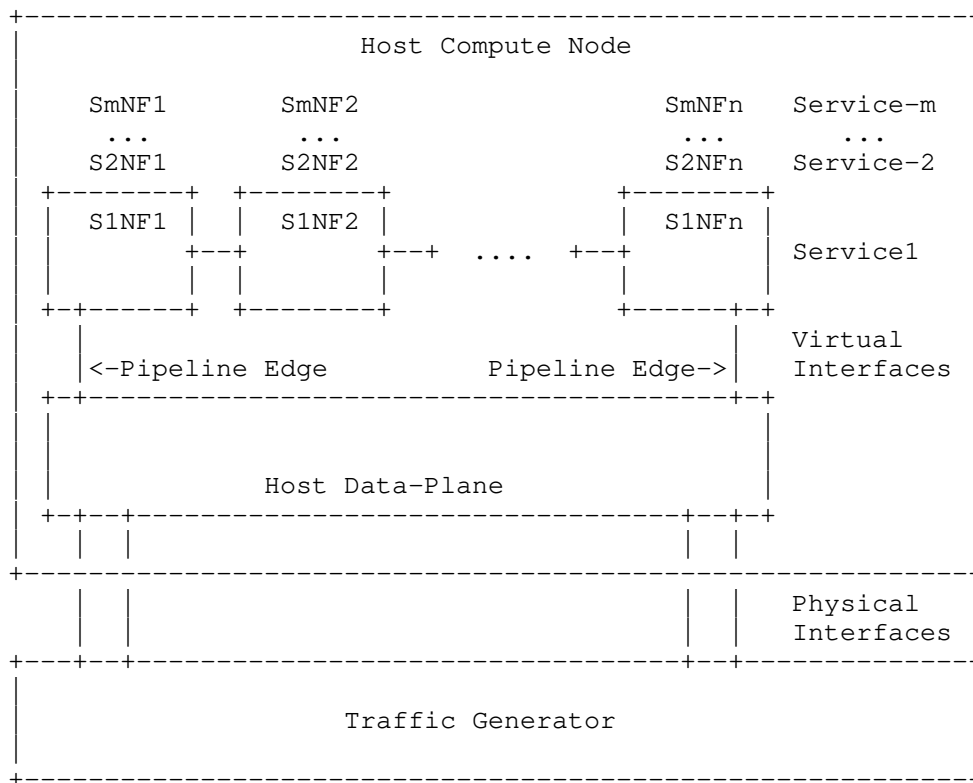


Figure 3. NF pipeline topology forming a service instance.

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:

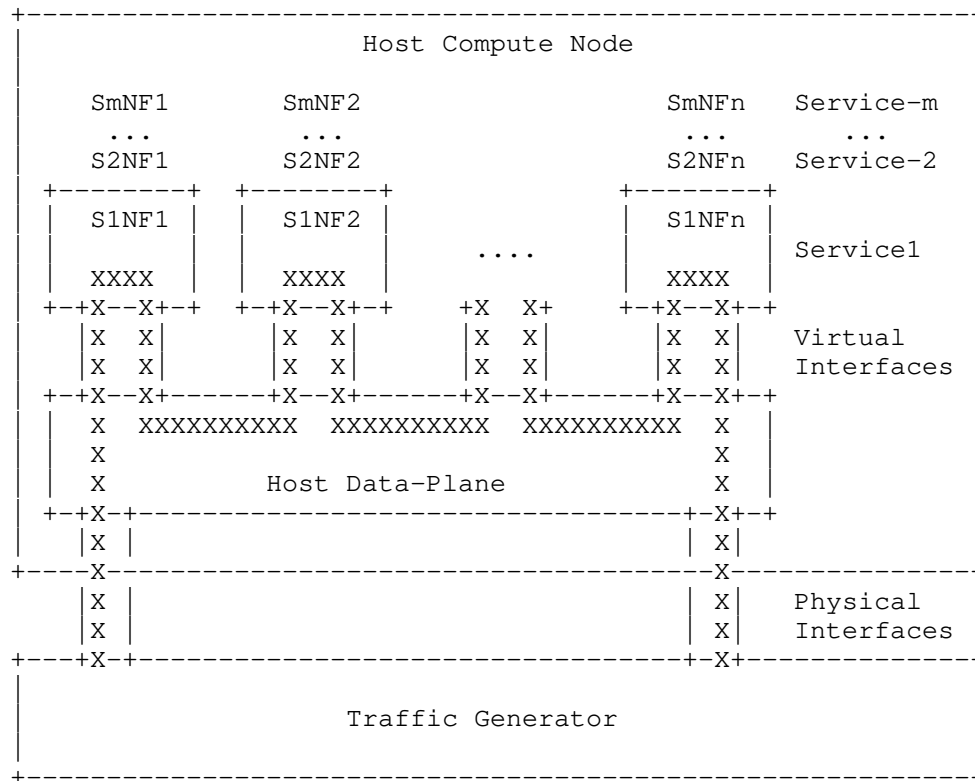


Figure 4. Snake packet path thru NF chain topology.

Pipeline packet path:

Figure 5. Pipeline packet path thru NF pipeline topology.

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.

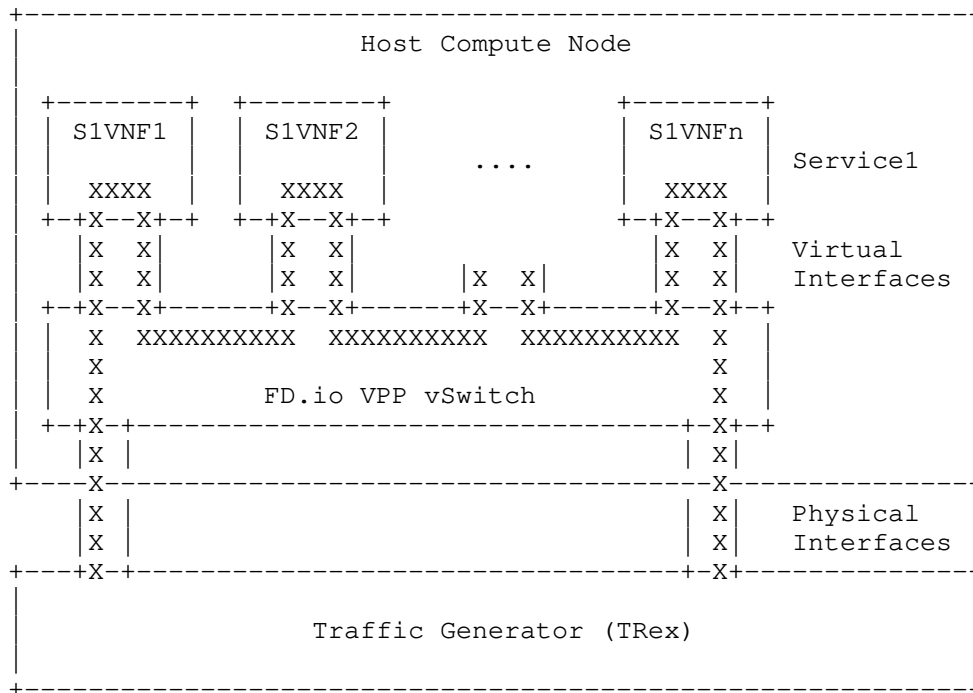


Figure 6. VNF service chain test setup.

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.

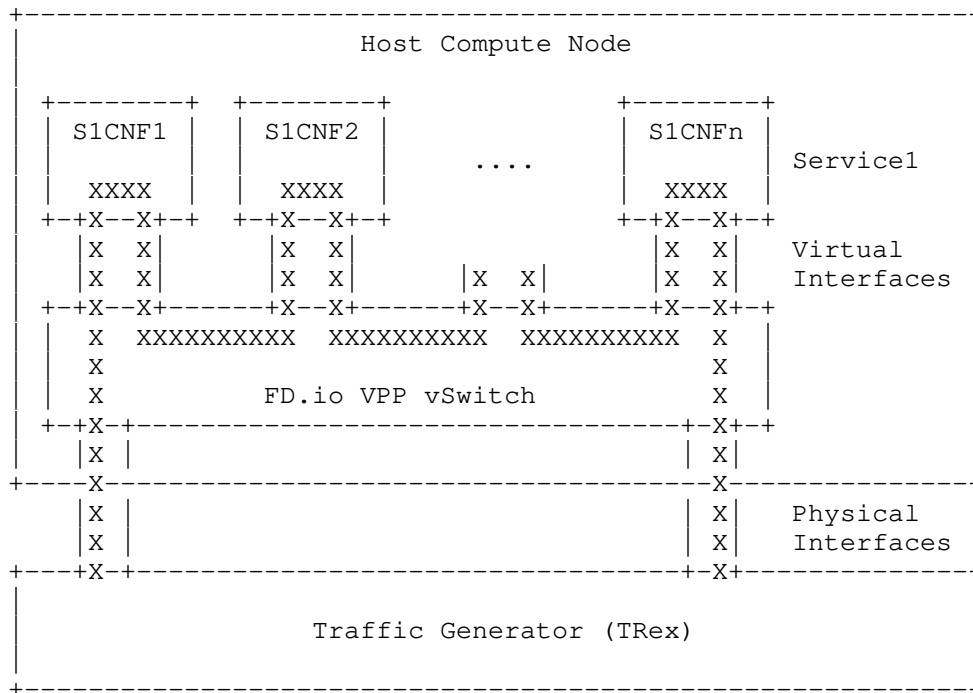


Figure 7. CNF service chain test setup.

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 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 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.

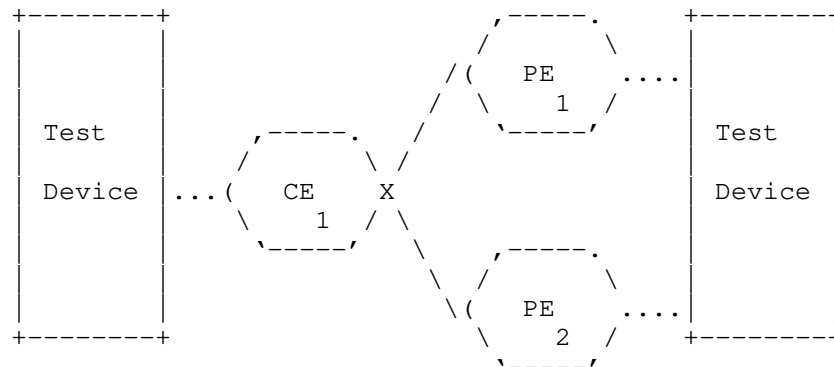


Figure 1 SUT for Throughput and other Ethernet Segment Tests

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.

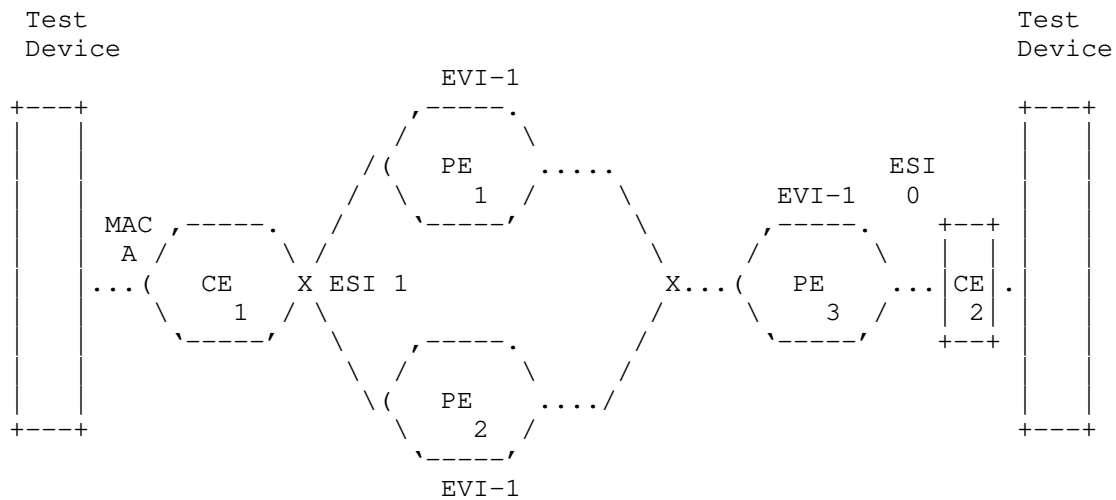


Figure 2 SUT with BGP & MPLS interconnecting multiple PE-ESI-CE locations

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

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., [ETSI14c], [ETSI19f] 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 [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 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.

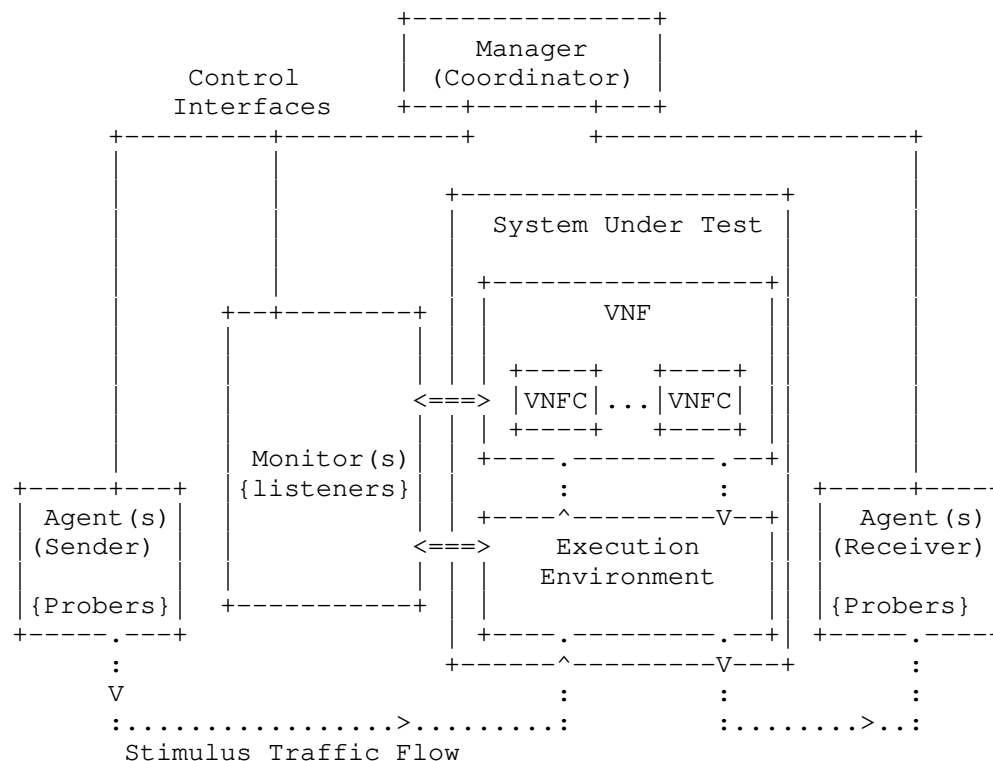


Figure 1: A VNF Benchmarking Architectural Framework

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

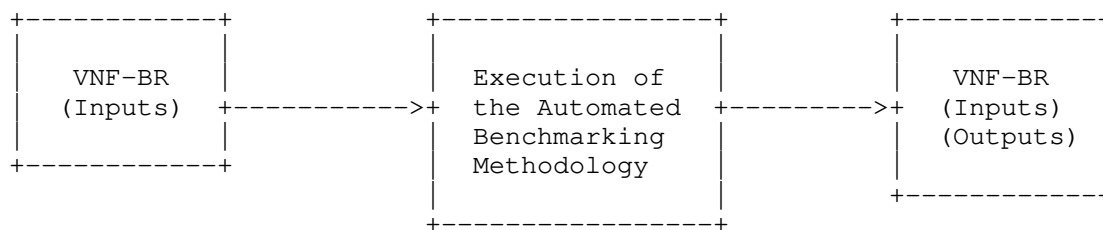


Figure 2: VNF benchmarking process inputs and outputs

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 vnfbfd {
            uses vnfbfd: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/>

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 9, 2021

V. Vassilev
Lightside Instruments
September 5, 2020

A YANG Data Model for Network Interconnect Tester Management
draft-vassilev-bmwg-network-interconnect-tester-04

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 March 9, 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
1.1. Terminology	2
1.1.1. Definitions and Acronyms	2
1.1.2. Tree Diagram	2
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	8
6. Traffic Analyzer Module YANG	16
7. IANA Considerations	23
7.1. URI Registration	23
7.2. YANG Module Name Registration	23
8. Security Considerations	24
9. References	24
9.1. Normative References	24
9.2. Informative References	24
Appendix A. Examples	24
A.1. Basic Test Program	25
A.2. Generating RFC2544 Testframes	26
Author's Address	26

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

1.1.1. Definitions and Acronyms

DUT: Device Under Test

TA: Traffic Analyzer

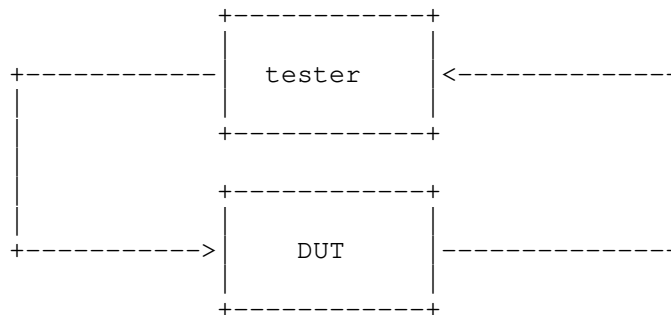
TG: Traffic Generator

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

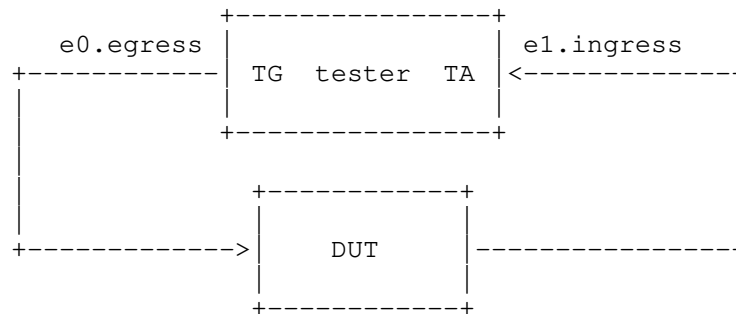
Network interconnect tests require active network elements part of the tested network that generate test traffic and network elements that analyze the test traffic at one or more points of its path. A network interconnect tester is a device that can either generate test traffic, analyze test traffic 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. 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. 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 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 realtime-epoch?                       yang:date-and-time
                                          {realtime-epoch}?
+--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 realtime-epoch?                    yang:date-and-time
|                                         {realtime-epoch}?
+---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@2020-09-05.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;
    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) 2020 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 2020-09-05 {
  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 ethernet-vlan {
  if-feature "ethernet";
  description
    "The device can generate vlan tagged ethernet traffic.";
}

feature realtime-epoch {
  description
    "The device can generate traffic precisely
    at configured realtime epoch.";
}

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 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 than 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 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.";
  }
}
```

```
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 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-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@2020-09-05.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;
        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) 2020 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 2020-09-05 {
  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 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

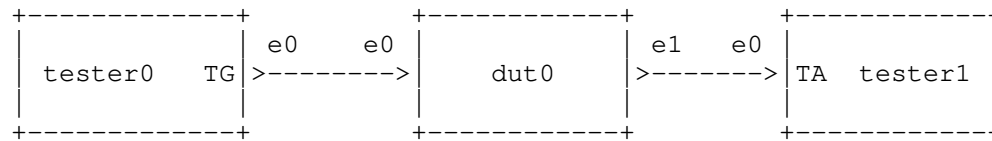
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, 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=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"
  "002ED4A5000000A115816C0000201C000"
  "0202C0200007001A00000010203040506"
  "0708090A0B0C0D0E0F101112")
...
```

Author's Address

Vladimir Vassilev
Lightside Instruments

Email: vladimir@lightside-instruments.com

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 30, 2021

S. Jacob, Ed.
V. Nagarajan
Juniper Networks
October 27, 2020

Benchmarking Methodology for EVPN Multicasting
draft-vikjac-bmwg-evpnmultest-05

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 April 30, 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
1.1. Requirements Language	2
1.2. Terminologies	3
2. Test Topology	4
3. Test Cases	6
3.1. Learning Rate	6
3.2. Flush Rate	7
3.3. Leave Latency	7
3.4. Join Latency	8
3.5. Leave Latency of N Vlan in DUT	9
3.6. Join Latency of N vlans in DUT working EVPN AA mode	9
3.7. Leave Latency of DUT operating in EVPN AA	10
3.8. Join Latency with reception of Type 6 route	11
4. Link Flap	11
4.1. Packet Loss measurement in DUT due to CE link Failure	12
4.2. Core Link Failure in EVPN AA	12
4.3. Routing Failure in DUT operating in EVPN-VXLAN AA	13
5. High Availability	14
5.1. Routing Engine Fail over.	14
6. SOAK Test	14
6.1. Stability of the DUT with traffic.	15
7. Acknowledgments	15
8. IANA Considerations	15
9. Security Considerations	15
10. References	16
10.1. Normative References	16
10.2. Informative References	16
Appendix A. Appendix	16
Authors' Addresses	16

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

All-Active Redundancy Mode: When all PEs attached to an Ethernet segment are allowed to forward known unicast traffic to/from that Ethernet segment for a given VLAN, then the Ethernet segment is defined to be operating in All-Active redundancy mode.

AA: All Active mode

CE: Customer Router/Devices/Switch.

DF: Designated Forwarder

DUT: Device under test.

EBGP: Exterior Border Gateway Protocol.

Ethernet Segment (ES): When a customer site (device or network) is connected to one or more PEs via a set of Ethernet links, then that set of links is referred to as an 'Ethernet segment'.

EVI: An EVPN instance spanning the leaf, spine devices participating in that EVPN.

EVPN: Ethernet Virtual Private Network

Ethernet Segment Identifier (ESI): A unique non-zero identifier that identifies an Ethernet segment is called an 'Ethernet Segment Identifier'.

Ethernet Tag: An Ethernet tag identifies a particular broadcast domain, e.g., a VLAN. An EVPN instance consists of one or more broadcast domains.

Interface: Physical interface of a router/switch.

IGMP: Internet Group Management Protocol

IBGP: Interior Border Gateway Protocol

IRB: Integrated routing and bridging interface

MAC: Media Access Control addresses on a PE.

MLD: Multicast Listener Discovery

NVO: Network Visualization Overlay

RT Traffic Generator.

Sub Interface Each physical Interfaces is subdivided into Logical units.

SA Single Active

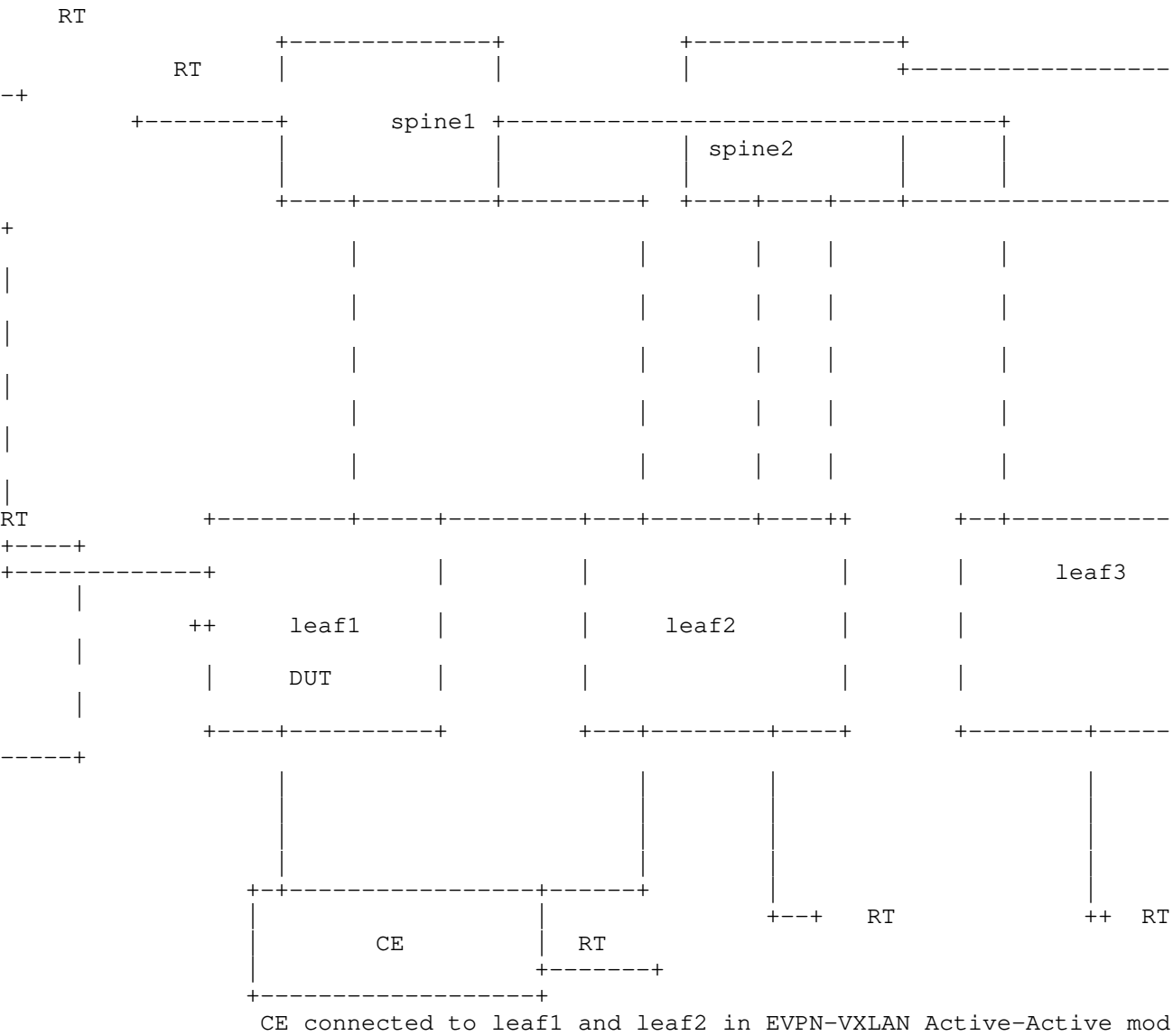
Single-Active Redundancy Mode: When only a single PE, among all the PEs attached to an Ethernet segment, is allowed to forward traffic to/from that Ethernet segment for a given VLAN, then the Ethernet segment is defined to be operating in Single-Active redundancy mode.

VXLAN: Virtual Extensible LAN

2. Test Topology

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 traffic generator is connected to CE,leaf1,leaf2,leaf3,spine1 and spine 2 to emulate multicast source and host generating IGMP join/leave.

Topology Diagram



e.

Topology 1

Topology Diagram

Figure 1

Test Setup Configurations:

Leaf1, Leaf2,Leaf3 are configured with Exterior Border Gateway protocol as the underlay protocol. The routes are advertised over it. The EVPN signaling is enabled on it in order to have the overlay reachability. Leaves are configured with "N" EVPN-VXLAN EVI's. CE

is multi homed to leaf1 and leaf2. The Interface connecting to the CE is configured with ESI per interface or ESI per vlan. Leaf1 and leaf2 are running EVPN-VXLAN AA mode to CE.

Spine1,spine2 are configured with Exterior Border Gateway protocol as the underlay protocol. The routes are advertised over it. The EVPN signaling is enabled over it to have the overlay reachability. Spines are configured with "N" EVPN-VXLAN EVI's. Traffic generators are connected spine1,spine2. Spine1 and Spine2 work as single home EVPN-VXLAN EVI's.

CE is acting as bridge configured with multiple vlans,the same vlans are configured on leaf1 and leaf2. Traffic generator is connected to CE. The traffic generator acts as sender or receiver of traffic.

Depending up on the test scenarios the traffic generators will be used to generate igmp membership report or multicast traffic.

The above configuration will be serving as the base configuration for all 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. Learning Rate

Objective:

Measure the time taken to learn X1...Xn IGMP join generated by host/ hosts.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN-VXLAN.Traffic generator connected to leaf1 must send IGMP membership report for groups X1... Xn to a vlan present in leaf1,leaf2 which is a part of EVPN-VLXAN EVI.Measure the time taken to learn X1..Xn (*,G) entries in the DUT.

Measurement :

Measure the time taken by the DUT to learn the "X" IGMP membership report. The test is repeated for "N" times and the values are collected. The IGMP membership report learning rate is calculated by

averaging the values obtained from "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn.The measurement is carried out using external server which polls the DUT using automated scripts.

Learning Rate = (T1+T2+..Tn)/N

3.2. Flush Rate

Objective:

Measure the time taken to Flush the X1... Xn (*,G) entries in DUT.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN-VXLAN.Traffic generator connected to the leaf1 must send IGMP membership report for groups X1... Xn to a vlan present in leaf1 which is a part of EVPN-VLXAN EVI. Stop the membership report from traffic generator. Measure the time taken to Flush X1..Xn (*,G) entries in the DUT.

Measurement :

Measure the time taken by the DUT to flush the "X" (*,G) entries The test is repeated for "N" times and the values are collected. The flush rate is calculated by averaging the values obtained from "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn.The measurement is carried out using external server which polls the DUT using automated scripts.

Flush Rate = (T1+T2+..Tn)/N

3.3. Leave Latency

Objective:

Measure the time taken by the DUT to stop forwarding the multicast traffic during the receipt of IGMP leave from RT.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN-VXLAN.Traffic generator connected to the leaf1 must send IGMP membership report for groups

X1... Xn to a vlan present in leaf1,leaf2 which is a part of EVPN-VLXAN EVI. Send multicast traffic from the RT port connected to spine1 to these groups requested by the leaf1. The leaf1 must receives multicast traffic.Send the IGMP leave message from the traffic generator to the leaf1. Measure the time taken by leaf1 to Flush X1..Xn (*,G) entries and stop forwarding the multicast traffic to RT.

Measurement :

Measure the time taken by the DUT to stop forwarding the multicast traffic. The test is repeated for "N" times and the values are collected. The leave latency is calculated by averaging the values obtained from "N" samples. "N" is an arbitrary number to get a sufficient sample.The time measured for each sample is denoted by T1,T2...Tn.The measurement is carried out using external server which polls the DUT using automated scripts.

Leave Latency = $(T1+T2+...Tn)/N$

3.4. Join Latency

Objective:

Measure the time taken by the DUT to create IGMP entries for N vlans.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN-VXLAN. Ensure the route reachability. The RT port connected to spine1 acts the source of the multicast traffic. The RT port connected to leaf1 acts as receiver of multicast traffic. Send IGMP membership report for groups X1...Xn from RT port connected to leaf1. The leaf1 has N vlans subscribed to these groups. Send multicast traffic from source.Measure the time taken to forward the multicast traffic to the receiver.

Measurement :

Measure the time taken by the DUT to forward the multicast traffic to these "N" vlans. The test is repeated for "N" times and the values are collected. The join latency is calculated by averaging the values obtained from "N" samples. "N" is an arbitrary number to get a sufficient sample.The time measured for each sample is denoted by T1,T2...Tn.The measurement is carried out using external server which polls the DUT using automated scripts.

$$\text{Join Latency} = (T1+T2+..Tn)/N$$

3.5. Leave Latency of N Vlans in DUT

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:

Confirm the DUT is up and running with EVPN-VXLAN. Ensure the route reachability. The RT port connected to spine1 acts the source of the multicast traffic. The RT port connected to leaf1 acts as receiver of multicast traffic. Send IGMP membership report for groups X1...Xn from RT port connected to leaf1. The leaf1 has N vlans subscribed to these groups. Send multicast traffic from source. Once the traffic is in steady state, send IGMP leave message to these groups. Once the leaf1 receiver the leave messages. it will flush the entries and stop forwarding the traffic to the receiver.

Measurement :

Measure the time taken by the DUT to stop forwarding the multicast traffic to these "N" vlans. The test is repeated for "N" times and the values are collected. The leave latency is calculated by averaging the values obtained from "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts.

$$\text{Leave Latency} = (T1+T2+..Tn)/N$$

3.6. Join Latency of N vlans in DUT working EVPN AA mode

Objective:

Measure 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:

Confirm the DUT is up and running with EVPN-VXLAN. Ensure the route reachability. The RT port connected to spine1 acts the source of the

multicast traffic. The RT port connected to CE acts as receiver of multicast traffic. leaf1 and leaf2 are multi homed EVPN-VXLAN EVI's running AA mode. The leaf1 and leaf2 have "N" vlans configured in EVPN-VXLAN EVI's, these vlans subscribe to multicast groups ranging from X1...Xn. Send IGMP membership report to these groups from RT connected to CE for these "N" vlans. Send multicast traffic from source to these groups. Measure time taken by the EVPN DF to forward the multicast traffic to the CE.

Measurement :

Measure the time taken by the EVPN DF to forward the multicast traffic for "N" vlans. The test is repeated for "N" times and the values are collected. The join latency is calculated by averaging the values obtained from "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts.

Join Latency = (T1+T2+...Tn)/N

3.7. Leave Latency of DUT operating in EVPN AA

Objective:

Measure 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:

Confirm the DUT is up and running with EVPN-VXLAN. Ensure the route reachability. The RT port connected to spine1 acts the source of the multicast traffic. The RT port connected to CE acts as receiver of multicast traffic. leaf1 and leaf2 are multi homed EVPN-VXLAN EVI's running AA mode. The leaf1 and leaf2 have "N" vlans configured in EVPN-VXLAN EVI's, these vlans subscribe to multicast groups ranging from X1...Xn. Send IGMP membership report to these groups from RT connected to CE for these "N" vlans. Send multicast traffic from source to these groups. Once traffic reaches steady state, send IGMP leave from RT connected to CE. Measure the time taken by the EVPN DF to stop forward the multicast traffic to the CE.

Measurement :

Measure the time taken by the EVPN DF to stop forward the multicast traffic for "N" vlans. The test is repeated for "N" times and the

values are collected. The leave latency is calculated by averaging the values obtained from "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts.

Leave Latency = $(T1+T2+...Tn/N)$

3.8. Join Latency with reception of Type 6 route

Objective:

Measure the time takes to forward 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. Packet Loss measurement in DUT due to CE link Failure

Objective:

Measure the packet loss during the CE to DF(DUT) link failure.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN-VXLAN. Ensure the route reachability. The RT port connected to spine1 acts the source of the multicast traffic. The RT port connected to CE acts as receiver of multicast traffic. leaf1 and leaf2 are multi homed EVPN-VXLAN EVI's running AA mode. The leaf1 and leaf2 have "N" vlans configured in EVPN-VXLAN EVI's, these vlans subscribe to multicast groups ranging from X1...Xn. Send IGMP membership report to these groups from RT connected to CE for these "N" vlans. Send multicast traffic from source to these groups. The DF is the leaf1(DUT). Disable the link between DF and CE. Traffic switch to the new DF. Measure the loss of the traffic.

Measurement :

Measure the packet loss duration during the link disable. The test is repeated for "N" times and the values are collected. The packet loss duration is calculated by averaging the values obtained from "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn.

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

4.2. Core Link Failure in EVPN AA

Objective:

Measure the packet loss during the DF core failure

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN-VXLAN. Ensure the route reachability. The RT port connected to spine1 acts the source of the multicast traffic. The RT port connected to CE acts as receiver of multicast traffic. leaf1 and leaf2 are multi homed EVPN-VXLAN EVI's

running AA mode. The leaf1 and leaf2 have "N" vlans configured in EVPN-VXLAN EVI's, these vlans subscribe to multicast groups ranging from X1...Xn. Send IGMP membership report to these groups from RT connected to CE for these "N" vlans. Send multicast traffic from source to these groups. The DF is the leaf1(DUT). Disable all the core links of DUT. Traffic switch to the new DF. Measure the loss of the traffic.

Measurement :

Measure the packet loss duration during the core link disable. The test is repeated for "N" times and the values are collected. The packet loss duration is calculated by averaging the values obtained from "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1, T2...Tn.

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

4.3. Routing Failure in DUT operating in EVPN-VXLAN AA

Objective:

Measure the packet loss during the DF routing failure

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN-VXLAN. Ensure the route reachability. The RT port connected to spine1 acts the source of the multicast traffic. The RT port connected to CE acts as receiver of multicast traffic. leaf1 and leaf2 are multi homed EVPN-VXLAN EVI's running AA mode. The leaf1 and leaf2 have "N" vlans configured in EVPN-VXLAN EVI's, these vlans subscribe to multicast groups ranging from X1...Xn. Send IGMP membership report to these groups from RT connected to CE for these "N" vlans. Send multicast traffic from source to these groups. The DF is the leaf1(DUT). Perform restart routing DUT. Traffic switch to the new DF. Measure the loss of the traffic.

Measurement :

Measure the packet loss duration during the routing failure in DUT. The test is repeated for "N" times and the values are collected. The packet loss duration is calculated by averaging the values obtained from "N" samples. "N" is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1, T2...Tn.

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

5. High Availability

5.1. Routing Engine Fail over.

Objective:

Measure traffic loss during routing engine failover.

Topology : Topology 3

Procedure:

Confirm the DUT is up and running with EVPN-VXLAN. Ensure the route reachability. The RT port connected to spine1 acts the source of the multicast traffic. The RT port connected to CE acts as receiver of multicast traffic. leaf1 and leaf2 are multi homed EVPN-VXLAN EVI's running AA mode. The leaf1 and leaf2 have "N" vlans configured in EVPN-VXLAN EVI's, these vlans subscribe to multicast groups ranging from X1...Xn. Send IGMP membership report to these groups from RT connected to CE for these "N" vlans. Send multicast traffic from source to these groups. The DF is the leaf1(DUT). Perform routing engine failover in DUT. Traffic switch to the new DF. Measure the loss of the traffic.

Measurement :

The expectation of the test is 0 traffic loss with no change in the DF role. DUT should not withdraw any routes. But in cases where the DUT is not properly synchronized between master and standby, due to that packet loss are observed. In that scenario the packet loss is measured. The test is repeated for "N" times and the values are collected. The packet loss is calculated by averaging the values obtained by "N" samples.

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

6. 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.

6.1. Stability of the DUT with traffic.

Objective:

Measure the stability of the DUT in a scaled environment with traffic.

Topology : Topology 3

Procedure:

Confirm the DUT is up and running with EVPN-VXLAN. Ensure the route reachability. The RT port connected to spine1 acts the source of the multicast traffic. The RT port connected to CE acts as receiver of multicast traffic. leaf1 and leaf2 are multi homed EVPN-VXLAN EVI's running AA mode. The leaf1 and leaf2 have "N" vlans configured in EVPN-VXLAN EVI's, these vlans subscribe to multicast groups ranging from X1...Xn. Send IGMP membership report to these groups from RT connected to CE for these "N" vlans. Send multicast traffic from source to these groups. The DF is the leaf1(DUT). Traffic will be forwarded to the CE by the DF. Run the traffic for "T" time interval.

Measurement :

Take the hourly reading of CPU, process memory. There should not be any leak, crashes, CPU spikes. The CPU spike is determined as the CPU usage which shoots at 40 to 50 percent of the average usage. The average value vary from device to device. Memory leak is determined by increase usage of the memory for EVPN-VPWS process. The expectation is under steady state the memory usage for EVPN-VXLAN, IGMP processes should not increase.

7. Acknowledgments

We would like to thank Al and Sarah for the support.

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

The benchmarking tests described in this document are limited to the performance characterization of controllers in a lab environment with isolated networks. 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 controller. Special capabilities SHOULD NOT exist in the controller specifically for benchmarking purposes. Any implications for network security arising from the controller SHOULD be identical in the lab and in production networks.

10. References

10.1. 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>>.

10.2. 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: September 7, 2020

M. Konstantynowicz, Ed.
V. Polak, Ed.
Cisco Systems
March 06, 2020

Probabilistic Loss Ratio Search for Packet Throughput (PLRsearch)
draft-vpolak-bmwg-plrsearch-03

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 September 7, 2020.

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. 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.	Known Implementations	16
5.1.	FD.io CSIT Implementation Specifics	16
5.1.1.	Measurement Delay	17
5.1.2.	Rounding Errors and Underflows	17
5.1.3.	Fitting Functions	17
5.1.4.	Prior Distributions	19
5.1.5.	Integrator	19
5.1.6.	Offered Load Selection	20
6.	IANA Considerations	20
7.	Security Considerations	21
8.	Acknowledgements	21
9.	References	21
9.1.	Normative References	21
9.2.	Informative References	21
	Authors' Addresses	22

1. Motivation

Network providers are interested in throughput a networking 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

Due to the indeterministic nature of certain NFV systems that are the targetted by PLRsearch algorithm, existing network benchmarking terms are explicated and a number of new terms and assumptions are introduced.

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 predicitions. 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 accomodate 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 chose 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. Known Implementations

The only known working implementation of PLRsearch is in Linux Foundation FD.io CSIT open-source project [FDio-CSIT-PLRsearch].

5.1. FD.io CSIT Implementation Specifics

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.

5.1.1. Measurement Delay

In a sample implemenation in FD.io CSIT project, there is roughly 0.5 second delay between trials due to restrictons 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).

5.1.2. 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.

5.1.3. 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.

5.1.3.1. 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:

$$r = (a * (1 + E^{(m/a)}) * \text{Log}[(E^{(b/a)} + E^{(m/a)}) / (1 + E^{(m/a)})]) / E^{(m/a)}$$

5.1.3.2. 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:

$$r = ((a*(E^{(-(b-m)^2/a^2)} - E^{-(m^2/a^2)}))/\text{Sqrt}[\text{Pi}] + m*\text{Erfc}[m/a] + (b-m)*\text{Erfc}[(-b+m)/a])/(1 + \text{Erf}[m/a])$$

5.1.4. 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.

5.1.5. 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 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.

5.1.6. 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

To be added.

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)", February 2020, <<https://tools.ietf.org/html/draft-vpolak-mkonstan-bmwg-mlrsearch>>.

[FDio-CSIT-PLRsearch]

"FD.io CSIT Test Methodology - PLRsearch", February 2020,
<[https://docs.fd.io/csit/rls2001/report/introduction/
methodology_data_plane_throughput/
methodology_plrsearch.html](https://docs.fd.io/csit/rls2001/report/introduction/methodology_data_plane_throughput/methodology_plrsearch.html)>.

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: September 7, 2020

M. Konstantynowicz, Ed.
V. Polak, Ed.
Cisco Systems
March 06, 2020

Multiple Loss Ratio Search for Packet Throughput (MLRsearch)
draft-vpolak-mkonstan-bmwg-mlrsearch-03

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 in a simple way of getting the same result sooner, so more repetitions can be done to describe the replicability.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 7, 2020.

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. 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
5. FD.io CSIT Implementation	12
5.1. Additional details	12
5.1.1. FD.io CSIT Input Parameters	14
5.2. Example MLRsearch Run	14
6. IANA Considerations	16
7. Security Considerations	16
8. Acknowledgements	17
9. References	17
9.1. Normative References	17
9.2. Informative References	17
Authors' Addresses	17

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 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 test setup 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 and/or separately for each measured direction. 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: $\text{bw_rate} = \text{pkt_rate} * (\text{frame_size} + \text{L1_overhead}) * 8$, 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. See [RFC2544] section 23.
- o Trial duration: amount of time over which packets are transmitted in a single 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 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 Traffic transmitted by traffic generator and received by SUT/DUT has the same packet rate in each direction, in other words the offered 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 are 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 a longer trial duration in a 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.
 - First measurement of the next phase will be internal search which always gives a valid bound and brings the width to the new goal.
 - Only one bound then needs to be re-measured with new duration.
- * 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 multiplying (for example 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, which is a simplified version of the existing implementation.

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.
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.
3. **final_trial_duration** - required trial duration for final rate measurements.

4. **initial_trial_duration** - trial duration for initial MLRsearch phase.
5. **final_relative_width** - required measurement resolution expressed as (lower_bound, upper_bound) interval width relative to upper_bound.
6. **packet_loss_ratio** - maximum acceptable PLR search criterion for PDR measurements.
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.

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. If loss ratio is zero, MRR is set below MTR so that interval width is equal to the width goal of the first intermediate phase.
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. If loss ratio is zero, MRR2 is set above MRR so that interval width is equal to the width goal of the first intermediate phase. MRR2 could end up being equal to MTR (for example if both measurements so far

had zero loss), which was already measured, step 3 is skipped in that case.

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 are formed by a (correctly ordered) pair of MRR2 and 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.
 - + 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. Else, if interval width is higher than the current phase goal:
 - + IF NDR interval does not meet the current phase width goal, prepare for internal search. The new transmit rate is a in the middle of NDR lower_bound and NDR upper_bound.
 - + IF PDR interval does not meet the current phase width goal, prepare for internal search. The new transmit rate is a in the middle 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.

5. FD.io CSIT Implementation

The only known working implementation of MLRsearch is in the open-source code running in Linux Foundation FD.io CSIT project [FDio-CSIT-MLRsearch] as part of a Continuous Integration / Continuous Development (CI/CD) framework.

MLRsearch is also available as a Python package in [PyPI-MLRsearch].

5.1. Additional details

This document so far has been describing a simplified version of MLRsearch algorithm. The full algorithm as implemented in CSIT 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 the current upper bound never update a valid upper bound, even if drop ratio is low.
- * Measurements below the 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_`).

7. Pessimistic external search.

- * Valid bound becoming invalid on re-measurement with higher duration is frequently a sign of SUT behaving in non-deterministic way (from blackbox point of view). If the final width interval goal is too narrow compared to width of rate region where SUT is non-deterministic, it is quite likely that there will be multiple invalid bounds before the external search finds a valid one.

- * In this case, external search can be sped up by increasing interval width more rapidly. As only powers of two ensure the subsequent internal search will not result in needlessly narrow interval, a parameter `_doublings_` is introduced to control the pessimism of external search. For example three doublings result in interval width being multiplied by eight in each external search iteration.

5.1.1. FD.io CSIT Input Parameters

1. `*maximum_transmit_rate*` - Typical values: 2 * 14.88 Mpps for 64B 10GE link rate, 2 * 18.75 Mpps for 64B 40GE NIC (specific model).
2. `*minimum_transmit_rate*` - Value: 2 * 10 kpps (traffic generator limitation).
3. `*final_trial_duration*` - Value: 30 seconds.
4. `*initial_trial_duration*` - Value: 1 second.
5. `*final_relative_width*` - Value: 0.005 (0.5%).
6. `*packet_loss_ratio*` - Value: 0.005 (0.5%).
7. `*number_of_intermediate_phases*` - Value: 2. The value has been chosen based on limited experimentation to date. More experimentation needed to arrive to clearer guidelines.
8. `*timeout*` - Limit for the overall search duration (for one search). If MLRsearch oversteps this limit, it immediately declares the test failed, to avoid wasting even more time on a misbehaving SUT. Value: 600 (seconds).
9. `*doublings*` - Number of doublings when computing new interval width in external search. Value: 2 (interval width is quadrupled). Value of 1 is best for well-behaved SUTs, but value of 2 has been found to decrease overall search time for worse-behaved SUT configurations, contributing more to the overall set of different SUT configurations tested.

5.2. Example MLRsearch Run

The following table shows data from a real test run in CSIT (using the default input values as above). The first column is the phase, the second is the trial measurement performed (aggregate bidirectional offered load in megapackets per second, and trial duration in seconds). Each of last four columns show one bound as updated after the measurement (duration truncated to save space).

Internet-DraMultiple Loss Ratio Search for Packet Throughput March 2020

Loss ratio is not shown, but invalid bounds are marked with a plus sign.

Phase	Trial	NDR lower	NDR upper	PDR lower	PDR upper
init.	37.50 1.00	N/A	37.50 1.	N/A	37.50 1.
init.	10.55 1.00	+10.55 1.	37.50 1.	+10.55 1.	37.50 1.
init.	9.437 1.00	+9.437 1.	10.55 1.	+9.437 1.	10.55 1.
int 1	6.053 1.00	6.053 1.	9.437 1.	6.053 1.	9.437 1.
int 1	7.558 1.00	7.558 1.	9.437 1.	7.558 1.	9.437 1.
int 1	8.446 1.00	8.446 1.	9.437 1.	8.446 1.	9.437 1.
int 1	8.928 1.00	8.928 1.	9.437 1.	8.928 1.	9.437 1.
int 1	9.179 1.00	8.928 1.	9.179 1.	9.179 1.	9.437 1.
int 1	9.052 1.00	9.052 1.	9.179 1.	9.179 1.	9.437 1.
int 1	9.307 1.00	9.052 1.	9.179 1.	9.179 1.	9.307 1.
int 2	9.115 5.48	9.115 5.	9.179 1.	9.179 1.	9.307 1.
int 2	9.243 5.48	9.115 5.	9.179 1.	9.243 5.	9.307 1.
int 2	9.179 5.48	9.115 5.	9.179 5.	9.243 5.	9.307 1.
int 2	9.307 5.48	9.115 5.	9.179 5.	9.243 5.	+9.307 5.

int 2	9.687 5.48	9.115 5.	9.179 5.	9.307 5.	9.687 5.
int 2	9.495 5.48	9.115 5.	9.179 5.	9.307 5.	9.495 5.
int 2	9.401 5.48	9.115 5.	9.179 5.	9.307 5.	9.401 5.
final	9.147 30.0	9.115 5.	9.147 30	9.307 5.	9.401 5.
final	9.354 30.0	9.115 5.	9.147 30	9.307 5.	9.354 30
final	9.115 30.0	+9.115 30	9.147 30	9.307 5.	9.354 30
final	8.935 30.0	8.935 30	9.115 30	9.307 5.	9.354 30
final	9.025 30.0	9.025 30	9.115 30	9.307 5.	9.354 30
final	9.070 30.0	9.070 30	9.115 30	9.307 5.	9.354 30
final	9.307 30.0	9.070 30	9.115 30	9.307 30	9.354 30

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", February 2020, <https://docs.fd.io/csit/rls2001/report/introduction/methodology_data_plane_throughput/methodology_mlrsearch_tests.html>.
- [PyPI-MLRsearch] "MLRsearch 0.3.0, Python Package Index", February 2020, <<https://pypi.org/project/MLRsearch/0.3.0/>>.

Authors' Addresses

Maciek Konstantynowicz (editor)
Cisco Systems

Email: mkonstan@cisco.com

Internet-Draft Multiple Loss Ratio Search for Packet Throughput March 2020

Vratko Polak (editor)
Cisco Systems

Email: vrpolak@cisco.com