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

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

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 2 September 2022.

Copyright Notice

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

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

Table of Contents

1. Introduction

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

Recently NFV infrastructure has evolved to include a lightweight virtualized platform called the containerized infrastructure, where VNFs share the same host Operating System(OS) and are logically isolated by using a different namespace. While previous NFV infrastructure uses a hypervisor to allocate resources for Virtual Machine(VMs) and instantiate VNFs, the containerized infrastructure virtualizes resources without a hypervisor, making containers very lightweight and more efficient in infrastructure resource utilization compared to the VM-based NFV infrastructure. When we consider benchmarking for VNFs in the containerized infrastructure, it may have a different System Under Test(SUT) and Device Under Test(DUT) configuration compared with both black-box benchmarking and VM-based NFV infrastructure as described in [RFC8172]. Accordingly, additional configuration parameters and testing strategies may be required.

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

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

isolation, hugepages, service function chaining. The benchmark experiences of various combinations of these mentioned configurations and and networking models are also presented in this draft as the references to set up and benchmark containerized infrastructure. Note that, although the detailed configurations of both infrastructures differ, the new benchmarks and metrics defined in [RFC8172] can be equally applied in containerized infrastructure from a generic-NFV point of view, and therefore defining additional metrics or methodologies are out of scope.

2. Terminology

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

3. Containerized Infrastructure Overview

For benchmarking of the containerized infrastructure, as mentioned in [RFC8172], the basic approach is to reuse existing benchmarking methods developed within the BMWG. Various network function specifications defined in BMWG should still be applied to containerized VNF(C-VNF)s for the performance comparison with physical network functions and VM-based VNFs. A major distinction of the containerized infrastructure from the VM-based infrastructure is the absence of a hypervisor. Without hypervisor, all C- VNFs share the same host resources, including but not limited to computing, storage, and networking resources, as well as the host Operating System(OS), kernel, and libraries. These architectural differences bring additional considerations of resource management impacts for benchmarking.

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

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

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

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

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

```
 +-------------------------------------------------------------------+
 |                          Baremetal Node                           |
 | +--------------+ +--------------+ +-------------- + +-------------+ |
 | |              | |     POD      | |      VM       | |     VM      | |
 | |              | |+------------+| |+-------------+| |  +-------+  | |
 | |   C-VNF(A)   | || C-VNFs(B)  || || C-VNFs(C)  || |  |PODs(D)|  | |
 | |              | |+------------+| |+-----^-------+| |  +---^---+  | |
 | |              | |              | |      |        | |      |      | |
 | |   +------+   | |   +------+   | |   +--v---+   | |  +---v--+   | |
 | +---| veth |---+ +---| veth |---+ +---|virtio|----+ +--|virtio|---+ |
 |     +--^---+         +---^--+         +--^---+         +---^--+     |
 |        |                 |               |                |        |
 |        |                 |            +--v---+         +---v--+     |
 | +------|-----------------|------------|vhost |---------|vhost |---+ |
 | |      |                 |            +--^---+         +---^--+   | |
 | |      |                 |               |                |      | |
 | |   +--v---+         +---v--+         +--v---+         +---v--+   | |
 | | +-| veth |---------| veth |---------| Tap  |---------| Tap  |-+ | |
 | | | +--^---+         +---^--+         +--^---+         +---^--+ | | |
 | | |    |                 |    vSwitch    |                |    | | |
 | | | +--|-----------------|---------------|----------------|--+ | | |
 | | +-|  |                 |    Bridge     |                |  |-+ | |
 | |   +--|-----------------|---------------|----------------|--+   | |
 | |      |   +---------+   |            +--|----------------|---+  | |
 | |      |   |Container|   |            |  |    Hypervisor  |   |  | |
 | |      |   | Engine  |   |            |  |                |   |  | |
 | |      |   +---------+   |            +--|----------------|---+  | |
 | |      |                 |  Host Kernel  |                |      | |
 | +------|-----------------|---------------|----------------|------+ |
 |     +--v-----------------v---------------v----------------v--+     |
 +-----|                     physical network                   |-----+
       +--------------------------------------------------------+
```

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

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

o BMP2VMP scenario

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

Figure 2: Single Host Test Scenario - BMP2VMP

o VMP2VMP scenario

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

Figure 3: Single Host Test Scenario - VMP2VMP

4. Networking Models in Containerized Infrastructure

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

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

4.1. Kernel-space vSwitch Model

```
    +-------------------------------------------------------------------+
    | User Space                                                        |
    |   +-----------+                                  +-----------+    |
    |   |   C-VNF   |                                  |   C-VNF   |    |
    |   | +-------+ |                                  | +-------+ |    |
    |   +-|  eth  |-+                                  +-|  eth  |-+    |
    |     +---^---+                                      +---^---+      |
    |         |                                              |          |
    |         |     +----------------------------------+     |          |
    |         |     |                                  |     |          |
    |         |     |  Networking Controller / Agent   |     |          |
    |         |     |                                  |     |          |
    |         |     +-----------------^^---------------+     |          |
    ----------|-----------------------||---------------------|----------
    |     +---v---+                   ||                 +---v---+      |
    |  +--|  veth |-------------------vv-----------------|  veth |--+   |
    |  |  +-------+          vSwitch Component           +-------+  |   |
    |  |            (OVS Kernel Datapath, Linux Bridge, ..)           |   |
    |  |                                                              |   |
    |  +------------------------------^---------------------------+   |
    |                                 |                                 |
    | Kernel Space         +-----------v----------+                   |
    +----------------------|          NIC         |--------------------+
                           +----------------------+
```

Figure 4: Examples of Kernel-Space vSwitch Model

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

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

4.2. User-space vSwitch Model

```
    +-------------------------------------------------------------------+
    | User Space                                                        |
    |   +---------------+                          +---------------+    |
    |   |     C-VNF     |                          |     C-VNF     |    |
    |   | +-----------+ |     +-----------------+  | +-----------+ |    |
    |   | |virtio-user| |     |    Networking   |  | |virtio-user|-|    |
    |   +-|   / eth   |-+     | Controller/Agent|  +-|   / eth   |-+    |
    |     +-----^-----+       +-------^^--------+    +-----^-----+      |
    |           |                     ||                    |            |
    |           |                     ||                    |            |
    |     +-----v-----+               ||              +-----v-----+      |
    |     | vhost-user|               ||              | vhost-user|      |
    |  +--|  / memif  |--------------vv--------------|  / memif  |--+   |
    |  |  +-----------+                               +-----------+  |   |
    |  |                          vSwtich                            |   |
    |  |                       +--------------+                      |   |
    |  +-----------------------|  PMD Driver  |----------------------+   |
    |                          |              |                          |
    |                          +-------^------+                          |
    ----------------------------------|-----------------------------------
    |                                 |                                  |
    |                                 |                                  |
    |                                 |                                  |
    | Kernel Space          +----------V-----------+                   |
    +----------------------|          NIC          |-------------------+
                           +-----------------------+
```

Figure 5: Examples of User-Space vSwitch Model

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

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

4.3. eBPF Acceleration Model

```
    +-------------------------------------------------------------------+
    | User Space                                                        |
    |    +----------------+                    +----------------+      |
    |    |      C-VNF     |                    |      C-VNF     |      |
    |    | +------------+ |                    | +------------+ |      |
    |    +-|    veth    |-+                    +-|    veth    |-+      |
    |      +-----^------+                      +------^-----+        |
    |            |                                     |              |
    -------------|-------------------------------------|--------------
    |      +-----v------+                      +------v-----+        |
    |      |    veth    |                      |    veth    |        |
    |      +-----^------+                      +------^-----+        |
    |            |                                     |              |
    |      +-----v------+                      +------v-----+        |
    |      |  tc eBPF   |                      |  tc eBPF   |        |
    |      |   Egress   |                      |   Egress   |        |
    |      +-----^------+                      +------^-----+        |
    |            |                                     |              |
    |            +--------------+          +-------------+            |
    |                           |          |                          |
    |                         +-v----------v-+                        |
    |                         |   tc eBPF    |                        |
    |                         |   Ingress    |                        |
    |                         +------^-------+                        |
    |                                |                                |
    |                         +------v-------+                        |
    |                         |     XDP      |                        |
    |                         +------^-------+                        |
    |                                |                                |
    | Kernel Space            +--------v--------+                    |
    +-----------------------|       NIC       |-----------------------+
                            +-----------------+
```

Figure 6: Examples of eBPF Acceleration Model

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

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

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

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

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

4.4. Smart-NIC Acceleration Model

```
+-------------------------------------------------------------------+
| User Space                                                        |
|    +-----------------+                   +-----------------+     |
|    |      C-VNF      |                   |      C-VNF      |     |
|    | +-------------+ |                   | +-------------+ |     |
|    +-|  vf driver  |-+                   +-|  vf driver  |-+     |
|      +-----^-------+                       +------^------+       |
|            |                                      |              |
-------------|--------------------------------------|--------------
|            +---------+                   +---------+             |
|               +------|-------------------|------+                |
|               | +----v-----+       +-----v----+ |                |
|               | | virtual  |       | virtual  | |                |
|               | | function |       | function | |                |
| Kernel Space  | +----^-----+  NIC  +-----^----+ |                |
+---------------|      |                   |      |----------------+
                | +----v-------------------v----+ |
                | |      Classify and Queue     | |
                | +-----------------------------+ |
                +---------------------------------+
```

Figure 7: Examples of Smart-NIC Acceleration Model

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

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

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

4.5. Model Combination

```
+-------------------------------------------------------+
| User Space                                            |
| +--------------------+         +--------------------+ |
| |        C-VNF       |         |        C-VNF       | |
| | +------+  +------+ |         | +------+  +------+ | |
| +-| veth |--| veth |-+         +-| veth |--| veth |-+ |
|   +---^--+  +---^--+             +--^---+  +---^--+   |
|       |         |                   |          |      |
|       |         |                   |          |      |
|       |     +---v--------+  +-------v----+     |      |
|       |     | vhost-user |  | vhost-user |     |      |
|       |  +--|  / memif   |--|  / memif   |--+  |      |
|       |  |  +------------+  +------------+  |  |      |
|       |  |             vSwitch              |  |      |
|       |  +----------------------------------+  |      |
|       |                                        |      |
--------|----------------------------------------|-------
|       +-----------+              +-------------+      |
|             +----|--------------|---+                 |
|             |+---v--+       +---v--+|                 |
|             ||  vf  |       |  vf  ||                 |
|             |+------+       +------+|                 |
| Kernel Space |                      |                 |
+--------------|           NIC        |----------------+
               +----------------------+
```

Figure 8: Examples of Model Combination deployment

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

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

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

5. Performance Impacts

5.1. CPU Isolation / NUMA Affinity

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

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

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

o Worse performance with VNF sharing CPUs across NUMA

5.2. Hugepages

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

5.3. Service Function Chaining

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

5.4. Additional Considerations

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

6. Security Considerations

TBD

7. References

7.1. Informative References

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

[Cilium] "Cilium Documentation", March 2022, <https://docs.cilium.io/en/stable//>.

[cilium-benchmark] Cilium, "CNI Benchmark: Understanding Cilium Network Performance", May 2021, <https://cilium.io/blog/2021/05/11/cni-benchmark>.

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

[DPDK_eBPF] "DPDK-Berkeley Packet Filter Library", August 2021, <https://doc.dpdk.org/guides/prog_guide/bpf_lib.html>.

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

[ETSI-NFV-IFA-038]
"Network Functions Virtualisation (NFV) Release 4; Architectural Framework; Report on network connectivity for container-based VNF", November 2021.

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

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

[Intel-EPA]
Intel, "Enhanced Platform Awareness in Kubernetes", 2018, <https://builders.intel.com/docs/networkbuilders/enhanced-platform-awareness-feature-brief.pdf>.

[Intel-SRIOV-NFV]
Patrick, K. and J. Brian, "SR-IOV for NFV Solutions Practical Considerations and Thoughts", February 2017.

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

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

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

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

[RFC8172] Morton, A., "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure", RFC 8172, July 2017, <https://www.rfc-editor.org/rfc/rfc8172>.

[RFC8204] Tahhan, M., O'Mahony, B., and A. Morton, "Benchmarking Virtual Switches in the Open Platform for NFV (OPNFV)", RFC 8204, September 2017, <https://www.rfc-editor.org/rfc/rfc8204>.

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

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

[userspace-cni]
"Userspace CNI Plugin", August 2021, <https://github.com/intel/userspace-cni-network-plugin>.

[ViNePERF] Anuket Project, "Cross-NUMA performance measurements with VSPERF", March 2019, <https://wiki.anuket.io/display/HOME/Cross-NUMA+performance+measurements+with+VSPERF>.

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

Appendix A. Benchmarking Experience(Contiv-VPP)

A.1. Benchmarking Environment

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

o Three physical server for benchmarking

Node Name	Specification	Description
Conatiner Control for Master	- Intel(R) Xeon(R) CPU E5-2690 (2Socket X 12Core) - MEM 128G - DISK 2T - Control plane : 1G	Container Deployment and Network Allocation - ubuntu 18.04 - Kubernetes Master - CNI Conterller .. Contive VPP Controller .. Contive VPP Agent
Conatiner Service for Worker	- Intel(R) Xeon(R) Gold 6148 (2socket X 20Core) - MEM 128G - DISK 2T - Control plane : 1G - Data plane : MLX 10G (1NIC 2PORT)	Container Service - ubuntu 18.04 - Kubernetes Worker - CNI Agent .. Contive VPP Agent
Packet Generator	- 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 - CentOS 7 - installed Trex 2.4

Figure 9: Test Environment-Server Specification

o The architecture of benchmarking

```
+----+   +-------------------------------------------------------+
|    |   |  Containerized Infrastructure Master Node             |
|    |   |  +-----------+                                        |
|   <-------> 1G PORT 0 |                                        |
|    |   |  +-----------+                                        |
|    |   +-------------------------------------------------------+
|    |
|    |   +-------------------------------------------------------+
|    |   |  Containerized Infrastructure Worker Node             |
|    |   |                    +----------------------------------+
| s  |   |  +-----------+     | +------------+   +------------+ |
| w <-------> 1G PORT 0 |     | | 10G PORT 0 |   | 10G PORT 1 | |
| i  |   |  +-----------+     | +------^-----+   +------^-----+ |
| t  |   |                    +--------|----------------|-------+
| c  |   +----------------------------|----------------|---------+
| h  |                                |                |
|    |   +----------------------------|----------------|---------+
|    |   |  Packet Generator Node     |                |         |
|    |   |                    +--------|----------------|-------+
|    |   |  +-----------+     | +------v-----+   +------v-----+ |
|   <-------> 1G PORT 0 |     | | 10G PORT 0 |   | 10G PORT 1 | |
|    |   |  +-----------+     | +------------+   +------------+ |
|    |   |                    +----------------------------------+
|    |   |                                                       |
+----+   +-------------------------------------------------------+
```

Figure 10: Test Environment-Architecture

o Network model of Containerized Infrastructure(User space Model)

```
   +---------------------------------------------+---------------------+
   |                   NUMA 0                    |        NUMA 0       |
   +---------------------------------------------|---------------------+
   |  Containerized Infrastructure Worker Node   |                     |
   |        +---------------------------+        |  +----------------+ |
   |        |           POD1            |        |  |     POD2       | |
   |        |      +-------------+      |        |  |   +-------+    | |
   |        |      |             |      |        |  |   |       |    | |
   |        |   +--v---+     +---v--+   |        |  | +-v--+  +-v--+ | |
   |        |   | eth1 |     | eth2 |   |        |  | |eth1|  |eth2| | |
   |        |   +--^---+     +---^--+   |        |  | +-^--+  +-^--+ | |
   |        +------|-------------|------+        |  +---|-------|----+ |
   |            +---             |               |      |       |      |
   |            |        +-------|---------------|------+       |      |
   |            |        |       |        +------|--------------+      |
   | +----------|--------|-------|--------|----+                      |
   | |          v        v       v        v    |                      |
   | |       +-tap10--tap11-+ +-tap20--tap21-+ |                      |
   | |       |  ^        ^  | |  ^        ^  | |                      |
   | |       |  |  VRF1  |  | |  |  VRF2  |  | |                      |
   | |       +--|--------|--+ +--|--------|--+ |                      |
   | |          |  +-----+       |    +---+    |                      |
   | | +-tap01--|--|-------------|----|---+     |                      |
   | | | +------v--v-+ VRF0 +----v----v-+ |     |                      |
   | | +-| 10G ETH0/0|------| 10G ETH0/1|-+     |                      |
   | |   +---^-------+      +-------^---+       |                      |
   | |   +---v-------+      +-------v---+       |                      |
   | +---| DPDK PMD0 |------| DPDK PMD1 |------+                       |
   |     +---^-------+      +-------^---+          User Space           |
   +---------|----------------------|------------|---------------------+
   |   +-----|----------------------|-----+       Kernal Space          |
   +---| +---V----+            +----v---+ |------|---------------------+
       | | PORT 0 |  10G NIC   | PORT 1 | |      |
       | +---^----+            +----^---+ |
       +-----|----------------------|-----+
       +-----|----------------------|-----+
   +---| +---V----+            +----v---+ |----------------------------+
   |   | | PORT 0 |  10G NIC   | PORT 1 | |   Packet Generator (Trex)  |
   |   | +--------+            +--------+ |                            |
   |   +----------------------------------+                            |
   +-------------------------------------------------------------------+
```

Figure 11: Test Environment-Network Architecture

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

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

A.2. Trouble shooting and Result

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

```
+---------------------------------------------+---------------------+
|                   NUMA 0                    |        NUMA 0       |
+---------------------------------------------|---------------------+
|  Containerized Infrastructure Worker Node   |                     |
|        +---------------------------+        |  +----------------+ |
|        |      POD1                 |        |  |     POD2       | |
|        |      +-------------+      |        |  |   +-------+    | |
|        |   +--v----+    +---v--+   |        |  | +-v--+  +-v--+ | |
|        |   | eth1 |     | eth2 |   |        |  | |eth1|  |eth2| | |
|        |   +--^---+     +---^--+   |        |  | +-^--+  +-^--+ | |
|        +------|-------------|------+        |  +---|-------|----+ |
|       +-------+             |               |      |       |      |
|        |       +-------------|---------------|------+       |      |
|        |       |             |        +------|--------------+      |
| +-----|-------|-------------|--------|----+ |                     |
| |      |       |             v        v    | |                     |
| |      |       |          +-tap10--tap11-+ | |                     |
| |      |       |          |  ^        ^  | | |                     |
| |      |       |          |  |  VRF1  |  | | |                     |
| |      |       |          +--|--------|--+ | |                     |
| |      |       |             +---+         | |                     |
| | +-*tap00--*tap01----------|----|---+      | |                     |
| | | +-V-------v-+ VRF0 +----v----v-+ |      | |                     |
| | +-| 10G ETH0/0|------| 10G ETH0/1|-+      | |                     |
| |   +-----^-----+      +------^----+        | |                     |
| |   +-----v-----+      +------v----+        | |                     |
| +---|*DPDK PMD0 |------|*DPDK PMD1 |------+ |                     |
|     +-----^-----+      +------^----+         | User Space          |
+-----------|--------------------|-------------|---------------------+
            v                    v
  *- CPU pinning interface
```

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

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

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

Figure 13: Test Results

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

B.1. Benchmarking Environment

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

o Three physical server for benchmarking

Node Name	Specification	Description
Conatiner Control for Master	- Intel(R) Core(TM) i5-6200U CPU (1socket x 4Core) - MEM 8G - DISK 500GB - Control plane : 1G	Container Deployment and Network Allocation - ubuntu 18.04 - Kubernetes Master - CNI Conterller MULTUS CNI SRIOV plugin with DPDK
Conatiner Service for Worker	- 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 - Centos 7.7 - Kubernetes Worker - CNI Agent MULTUS CNI SRIOV plugin with DPDK
Packet Generator	- 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 - CentOS 7.7 - installed Trex 2.4

Figure 14: Test Environment-Server Specification

o The architecture of benchmarking

```
+----+   +-------------------------------------------------------+
|    |   |  Containerized Infrastructure Master Node             |
|    |   |  +-----------+                                        |
|   <-------> 1G PORT 0 |                                        |
|    |   |  +-----------+                                        |
|    |   +-------------------------------------------------------+
|    |
|    |   +-------------------------------------------------------+
|    |   |  Containerized Infrastructure Worker Node             |
|    |   |                    +----------------------------------+
| s  |   |  +-----------+     | +------------+   +------------+ |
| w <-------> 1G PORT 0 |     | | 40G PORT 0 |   | 40G PORT 1 | |
| i  |   |  +-----------+     | +------^-----+   +------^-----+ |
| t  |   |                    +--------|----------------|-------+
| c  |   +----------------------------|----------------|---------+
| h  |                                |                |
|    |   +----------------------------|----------------|---------+
|    |   |  Packet Generator Node     |                |         |
|    |   |                    +--------|----------------|-------+
|    |   |  +-----------+     | +------v-----+   +------v-----+ |
|   <-------> 1G PORT 0 |     | | 40G PORT 0 |   | 40G PORT 1 | |
|    |   |  +-----------+     | +------------+   +------------+ |
|    |   |                    +----------------------------------+
|    |   |                                                       |
+----+   +-------------------------------------------------------+
```

Figure 15: Test Environment-Architecture

o Network model of Containerized Infrastructure(User space Model)

```
+---------------------------------------------+---------------------+
|             CMK shared core                 | CMK exclusive core  |
+---------------------------------------------|---------------------+
|  Containerized Infrastructure Worker Node   |                     |
|        +---------------------------+        |  +----------------+ |
|        |           POD1            |        |  |     POD2       | |
|        |         (testpmd)         |        |  |   (testpmd)    | |
|        |      +-------------+      |        |  |   +-------+    | |
|        |      |             |      |        |  |   |       |    | |
|        |   +--v---+     +---v--+   |        |  | +-v--+  +-v--+ | |
|        |   | eth1 |     | eth2 |   |        |  | |eth1|  |eth2| | |
|        |   +--^---+     +---^--+   |        |  | +-^--+  +-^--+ | |
|        +------|-------------|------+        |  +---|-------|----+ |
|               |             |               |      |       |      |
|         +------           +-+               |      |       |      |
|         |            +----|-----------------|------+       |      |
|         |            |    |        +--------|--------------+      |
|         |            |    |        |        |           User Space|
+---------|------------|----|--------|--------|---------------------+
|         |            |    |        |        |                     |
|      +--+     +------|    |        |        |         Kernal Space|
+------|--------|-----------|--------|--------+---------------------+
| +----|--------|-----------|--------|-----+  |                     |
| | +--v--+  +--v--+     +--v--+  +--v--+  |  |                 NIC|
| | | VF0 |  | VF1 |     | VF2 |  | VF3 |  |  |                     |
| | +--|---+ +|----+     +----|+  +-|---+  |  |                     |
| +----|------|---------------|-----|------+  |                     |
+---| +v------v+            +-v-----v+ |------|---------------------+
    | | PORT 0 |  40G NIC   | PORT 1 | |
    | +---^----+            +----^---+ |
    +-----|----------------------|-----+
    +-----|----------------------|-----+
+---| +---V----+            +----v---+ |----------------------------+
|   | | PORT 0 |  40G NIC   | PORT 1 | |   Packet Generator (Trex)  |
|   | +--------+            +--------+ |                            |
|   +----------------------------------+                            |
+-------------------------------------------------------------------+
```

Figure 16: Test Environment-Network Architecture

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

B.2. Trouble shooting and Results

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

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

Figure 17: SR-IOV CPU Pinning Test Results

Appendix C. Benchmarking Experience(Multi-pod Test)

C.1. Benchmarking Overview

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

```
     +-----------------------------------------------------------------+
     |Worker Node                                                      |
     |   +-------------------------------------------------------+     |
     |   |Kubernetes                                             |     |
     |   |   +--------------+                +--------------+   |     |
     |   |   |     pod1     |                |     pod2     |   |     |
     |   |   |  +--------+  |                |  +--------+  |   |     |
     |   |   |  | L2FWD  |  |                |  | L2FWD  |  |   |     |
     |   |   |  +---^--v-+  |                |  +--^--v--+  |   |     |
     |   |   |  | DPDK   |  |                |  | DPDK   |  |   |     |
     |   |   |  +---^--v-+  |                |  +--^--v--+  |   |     |
     |   |   +------^--v----+                +-----^--v-----+   |     |
     |   |          ^  v                           ^  v          |     |
     |   |   +------^--v>>>>>>>>>>>>>>>>>>>>>>>>>>>>^--v-----+   |     |
     |   |   |      ^  OVS-DPDK / VPP-memif vSwitch   v     |   |     |
     |   |   +------^--------------------------------v-----+   |     |
     |   |   |      ^           PMD Driver            v     |   |     |
     |   |   +------^--------------------------------v-----+   |     |
     |   |          ^                                 v         |     |
     |   +----------^--------------------------------v-----------+     |
     |              ^                                 v               |
     |   +----------^--------------------------------v---------+       |
     |   |          ^             40G NIC             v         |       |
     |   |   +------^-------+                +--------v-----+   |       |
     +---|---|    Port 0    |----------------|    Port 1    |---|-------+
         |   +------^-------+                +--------v-----+   |
         +----------^--------------------------------v---------+
             +------^-------+                +--------v-----+
     +-------|    Port 0    |----------------|    Port 1    |---------+
     |       +------^-------+                +--------v-----+         |
     |                  Traffic Generator (TRex)                      |
     |                                                                |
     +----------------------------------------------------------------+
```

Figure 18: Multi-pod Benchmarking Scenario

C.2. Hardware Configurations

Node Name	Specification	Description
Conatiner Control for Master	- Intel(R) Core(TM) E5-2620v3 @ 2.40GHz (1socket x 12Cores) - MEM 32GB - DISK 1TB - NIC: Control plane: 1G - OS: CentOS Linux7.9	Container Deployment and Network Allocation - ubuntu 18.04 - Kubernetes Master - CNI Controller - MULTUS CNI - DPDK-OVS/VPP-memif
Conatiner Service for Worker	- Intel(R) Xeon(R) Gold 6148 @ 2.40GHz (2socket X 40Cores) - MEM 256GB - DISK 2TB - NIC - Control plane: 1G - Data plane: XL710-qda2 (1NIC 2PORT- 40Gb) - OS: CentOS Linux 7.9	- Container dpdk-L2fwd - Kubernetes Worker - CNI Agent - Multus CNI - DPDK-OVS/VPP-memif
Packet Generator	- Intel(R) Xeon(R) Gold 6148 @ 2.4Ghz (2Socket X 40Core) - MEM 256GB - DISK 2TB - NIC - Data plane: XL710-qda2 (1NIC 2PORT - 40Gb) - OS: CentOS Lunix 7.9	Packet Generator - Installed Trex v2.92

Figure 19: Hardware Configurations for Multi-pod Benchmarking

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

C.3. NUMA Allocation Scenario

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

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

Table 1: NUMA Allocation Scenarios

C.4. Traffic Generator Configurations

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

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

C.5. Benchmark Results and Trouble-shootings

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

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

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

Authors' Addresses

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

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

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

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

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

Benchmarking Working Group
Internet-Draft
Intended status: Informational
Expires: 8 September 2022

M. Konstantynowicz, Ed.
V. Polak
Cisco Systems
7 March 2022

Multiple Loss Ratio Search for Packet Throughput (MLRsearch)
draft-ietf-bmwg-mlrsearch-02

Abstract

TODO: Update after all sections are ready.

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 8 September 2022.

Copyright Notice

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

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

Table of Contents

1. Terminology

TODO: Update after most other sections are updated.

* TODO: The current text uses Throughput for the zero loss ratio load. Is the capital T needed/useful?

* DUT and SUT: see the definitions in https://gerrit.fd.io/r/c/csit/+/35545

* Traffic Generator (TG) and Traffic Analyzer (TA): see https://datatracker.ietf.org/doc/html/rfc6894#section-4 TODO: Maybe there is an earlier RFC?

* Overall search time: the time it takes to find all required loads within their precision goals, starting from zero trials measured at given DUT configuration and traffic profile.

* TODO: traffic profile?

* Intended load: https://datatracker.ietf.org/doc/html/rfc2285#section-3.5.1

* Offered load: https://datatracker.ietf.org/doc/html/rfc2285#section-3.5.2

* Maximum offered load (MOL): see https://datatracker.ietf.org/doc/html/rfc2285#section-3.5.3

* Forwarding rate at maximum offered load (FRMOL) https://datatracker.ietf.org/doc/html/rfc2285#section-3.6.2

* Trial Loss Count: the number of frames transmitted minus the number of frames received. Negative count is possible, e.g. when SUT duplicates some frames.

* Trial Loss Ratio: ratio of frames received relative to frames transmitted over the trial duration. For bi-directional throughput tests, the aggregate ratio is calculated, based on the aggregate number of frames transmitted and received. If the trial loss count is negative, its absolute value MUST be used to keep compliance with RFC2544.

* Safe load: any value, such that trial measurement at this (or lower) intended load is correcrly handled by both TG and TA, regardless of SUT behavior. Frequently, it is not known what the safe load is.

* Max load (TODO rename?): Maximal intended load to be used during search. Benchmarking team decides which value is low enough to guarantee values reported by TG and TA are reliable. It has to be a safe load, but it can be lower than a safe load estimate for added safety. See the subsection on unreliable test equipment below. This value MUST NOT be higher than MOL, which itself MUST NOT be higher than Maximum Frame Rate https://datatracker.ietf.org/doc/html/rfc2544#section-20

* Min load: Minimal intended load to be used during search. Benchmarking team decides which value is high enough to guarantee the trial measurement results are valid. E.g. considerable overall search time can be saved by declaring SUT faulty if min load trial shows too high loss rate. Zero frames per second is a valid min load value

* Effective loss ratio: a corrected value of trial loss ratio chosen to avoid difficulties if SUT exhibits decreasing loss ratio with increasing load. It is the maximum of trial loss ratios measured at the same duration on all loads smaller than (and including) the current one.

* Target loss ratio: a loss ratio value acting as an input for the search. The search is finding tight enough lower and upper bounds in intended load, so that the measurement at the lower bound has smaller or equal trial loss ratio, and upper bound has strictly larger trial loss ratio. For the tightest upper bound, the effective loss ratio is the same as trial loss ratio at that upper bound load. For the tightest lower bound, the effective loss ratio can be higher than the trial loss ratio at that lower bound, but still not larger than the target loss ratio.

* TODO: Search algorithm.

* TODO: Precision goal.

* TODO: Define a "benchmarking group".

* TODO: Upper and lower bound.

* TODO: Valid and invalid bound?

* TODO: Interval and interval width?

TODO: Mention NIC/PCI bandwidth/pps limits can be lower than bandwidth of medium.

2. Intentions of this document

The intention of this document is to provide recommendations for: * optimizing search for multiple target loss ratios at once, * speeding up the overall search time, * improve search results repeatability and comparability.

No part of RFC2544 is intended to be obsoleted by this document.

3. RFC2544

3.1. Throughput search

It is useful to restate the key requirements of RFC2544 using the new terminology (see section Terminology).

The following sections of RFC2544 are of interest for this document.

* https://datatracker.ietf.org/doc/html/rfc2544#section-20 Mentions the max load SHOULD not be larget than the theoretical maximum rate for the frame size on the media.

* https://datatracker.ietf.org/doc/html/rfc2544#section-23 Lists the actions to be done for each trial measurement, it also mentions loss rate as an example of trial measurement results. This document uses loss count instead, as that is the quantity that is easier for the current test equipment to measure, e.g. it is not affected by the real traffic duration. TODO: Time uncertainty again.

* https://datatracker.ietf.org/doc/html/rfc2544#section-24 Mentions "full length trials" leading to the Throughput found, as opposed to shorter trial durations, allowed in an attempt to "minimize the length of search procedure". This document talks about "final trial duration" and aims to "optimize overal search time".

* https://datatracker.ietf.org/doc/html/rfc2544#section-26.1 with https://www.rfc-editor.org/errata/eid422 finaly states requirements for the search procedure. It boils down to "increase intended load upon zero trial loss and decrease intended load upon non-zero trial loss".

No additional constraints are placed on the load selection, and there is no mention of an exit condition, e.g. when there is enough trial measurements to proclaim the largest load with zero trial loss (and final trial duration) to be the Throughput found.

4. Problems

4.1. Repeatability and Comparability

RFC2544 does not suggest to repeat Throughput search, and from just one Throughput value, it cannot be determined how repeatable that value is (how likely it is for a repeated Throughput search to end up with a value less then the precision goal away from the first value).

Depending on SUT behavior, different benchmark groups can report significantly different Througput values, even when using identical SUT and test equipment, just because of minor differences in their search algorithm (e.g. different max load value).

While repeatability can be addressed by repeating the search several times, the differences in the comparability scenario may be systematic, e.g. seeming like a bias in one or both benchmark groups.

MLRsearch algorithm does not really help with the repeatability problem. This document RECOMMENDS to repeat a selection of "important" tests ten times, so users can ascertain the repeatability of the results.

TODO: How to report? Average and standard deviation?

Following MLRsearch algorithm leaves less freedom for the benchmark groups to encounter the comparability problem, alghough more research is needed to determine the effect of MLRsearch’s tweakable parameters.

4.2. Non-Zero Target Loss Ratios

https://datatracker.ietf.org/doc/html/rfc1242#section-3.17 defines Throughput as: The maximum rate at which none of the offered frames are dropped by the device.

and then it says: Since even the loss of one frame in a data stream can cause significant delays while waiting for the higher level protocols to time out, it is useful to know the actual maximum data rate that the device can support.

New "software DUTs" (traffic forwarding programs running on commercial-off-the-shelf compute server hardware) frequently exhibit quite low repeatability of Throughput results per above definition.

This is due to, in general, throughput rates of software DUTs (programs) being sensitive to server resource allocation by OS during runtime, as well as any interrupts or blocking of software threads involved in packet processing.

To deal with this, this document recommends discovery of multiple throughput rates of interest for software DUTs that run on general purpose COTS servers (with x86, AArch64 Instruction Set Architectures): * throughput rate with target of zero packet loss ratio. * at least one throughput rate with target of non-zero packet loss ratio.

In our experience, the higher the target loss ratio is, the better is the repeatability of the corresponding load found.

TODO: Define a good name for a load corresponding to a specific non-zero target loss ration, while keeping Throughput for the load corresponding to zero target loss ratio.

This document RECOMMENDS the benchmark groups to search for corresponding loads to at least one non-zero target loss ratio. This document does not suggest any particular non-zero target loss ratio value to search the corresponding load for.

5. Solution ideas

This document gives several independent ideas on how to lower the (average) overall search time, while remaining unconditionally compliant with RFC2544 (and adding some of extensions).

This document also specifies one particular way to combine all the ideas into a single search algorithm class (single logic with few tweakable parameters).

Little to no research has been done into the question of which combination of ideas achieves the best compromise with respect to overal search time, high repeatability and high comparability.

TODO: How important it is to discuss particular implementation choices, especially when motivated by non-deterministic SUT behavior?

5.1. Short duration trials

https://datatracker.ietf.org/doc/html/rfc2544#section-24 already mentions the possibity of using shorter duration for trials that are not part of "final determination".

Obviously, the upper and lower bound from a smaller duration trial can be used as the initial upper and lower bound for the final determination.

MLRsearch makes it clear a re-measurement is always needed (new trial measurement with the same load but longer duration). It also specifes what to do if the longer trial is no longer a valid bound (TODO define?), e.g. start an external search. Additionaly one halving can be saved during the shorter duration search.

5.2. FRMOL as reasonable start

TODO expand: Overal search ends with "final determination" search, preceded by "shorter duration search" preceded by "bound initialization", where the bounds can be considerably different from min and max load.

For SUTs with high repeatability, the FRMOL is usually a good approximation of Throughput. But for less repeatable SUTs, forwarding rate (TODO define) is frequently a bad approximation to Throughput, therefore halving and other robust-to-worst-case approaches have to be used. Still, forwarding rate at FRMOL load can be a good initial bound.

5.3. Non-zero loss ratios

See the "Popularity of non-zero target loss ratios" section above.

TODO: Define "trial measurement result classification criteria", or keep reusing long phrases without definitions?

A search for a load corresponding to a non-zero target loss rate is very similar to a search for Throughput, just the criterion when to increase or decrease the intended load for the next trial measurement uses the comparison of trial loss ratio to the target loss ratio (instead of comparing loss count to zero) Any search algorithm that works for Throughput can be easily used also for non-zero target loss rates, perhaps with small modifications in places where the measured forwarding rate is used.

Note that it is possible to search for multiple loss ratio goals if needed.

5.4. Concurrent ratio search

A single trial measurement result can act as an upper bound for a lower target loss ratio, and as a lower bound for a higher target loss ratio at the same time. This is an example of how it can be advantageous to search for all loss ratio goals "at once", or at least "reuse" trial measurement result done so far.

Even when a search algorithm is fully deterministic in load selection while focusing on a single loss ratio and trial duration, the choice of iteration order between target loss ratios and trial durations can affect the obtained results in subtle ways. MLRsearch offers one particular ordering.

5.5. Load selection heuristics and shortcuts

Aside of the two heuristics already mentioned (FRMOL based initial bounds and saving one halving when increasing trial duration), there are other tricks that can save some overall search time at the cost of keeping the difference between final lower and upper bound intentionally large (but still within the precision goal).

TODO: Refer implementation subsections on: * Uneven splits. * Rounding the interval width up. * Using old invalid bounds for interval width guessing.

The impact on overall duration is probably small, and the effect on result distribution maybe even smaller. TODO: Is the two-liner above useful at all?

6. Non-compliance with RFC2544

It is possible to achieve even faster search times by abandoning some requirements and suggestions of RFC2544, mainly by reducing the wait times at start and end of trial.

Such results are therefore no longer compliant with RFC2544 (or at least not unconditionally), but they may still be useful for internal usage, or for comparing results of different DUTs achieved with an identical non-compliant algorithm.

TODO: Refer to the subsection with CSIT customizations.

7. Additional Requirements

RFC2544 can be understood as having a number of implicit requirements. They are made explicit in this section (as requirements for this document, not for RFC2544).

Recommendations on how to properly address the implicit requirements are out of scope of this document.

7.1. TODO: Search Stop Criteria

TODO: Mention the timeout parameter?

7.2. Reliability of Test Equipment

Both TG and TA MUST be able to handle correctly every intended load used during the search.

On TG side, the difference between Intended Load and Offered Load MUST be small.

TODO: How small? Difference of one packet may not be measurable due to time uncertainties.

TODO expand: time uncertainty.

To ensure that, max load (see Terminology) has to be set to low enough value. Benchmark groups MAY list the max load value used, especially if the Throughput value is equal (or close) to the max load.

Solutions (even problem formulations) for the following open problems are outside of the scope of this document: * Detecting when the test equipment operates above its safe load. * Finding a large but safe load value. * Correcting any result affected by max load value not being a safe load.

7.2.1. Very late frames

RFC2544 requires quite conservative time delays see https://datatracker.ietf.org/doc/html/rfc2544#section-23 to prevent frames buffered in one trial measurement to be counted as received in a subsequent trial measurement.

However, for some SUTs it may still be possible to buffer enough frames, so they are still sending them (perhaps in bursts) when the next trial measurement starts. Sometimes, this can be detected as a negative trial loss count, e.g. TA receiving more frames than TG has sent during this trial measurement. Frame duplication is another way of causing the negative trial loss count.

https://datatracker.ietf.org/doc/html/rfc2544#section-10 recommends to use sequence numbers in frame payloads, but generating and verifying them requires test equipment resources, which may be not plenty enough to suport at high loads. (Using low enough max load would work, but frequently that would be smaller than SUT's sctual Throughput.)

RFC2544 does not offer any solution to the negative loss problem, except implicitly treating negative trial loss counts the same way as positive trial loss counts.

This document also does not offer any practical solution.

Instead, this document SUGGESTS the search algorithm to take any precaution necessary to avoid very late frames.

This document also REQUIRES any detected duplicate frames to be counted as additional lost frames. This document also REQUIRES, any negative trial loss ratio to be treated as positive trial loss ratio of the same absolute value.

!!! Nothing below is up-to-date with draft v02. !!!

8. MLRsearch Background

TODO: Old section, probably obsoleted by preceding section(s).

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, each rate is associated with a distinct Packet Loss Ratio (PLR) criterion.

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 with NDR and PDR pair as an example.

Similarly to other throughput search approaches like binary search, MLRsearch is effective for SUTs/DUTs with PLR curve that is non-decreasing with growing offered load. It may not be as effective for SUTs/DUTs with abnormal PLR curves, although it will always converge to some value.

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:

* 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.

* SUT/DUT packet processing capacity is the same in both directions, resulting in the same packet loss under load.

MLRsearch can be applied even without those assumptions, but in that case the aggregate loss ratio is less useful as a metric.

MLRsearch can be used for network transactions consisting of more than just one packet, or anything else that has intended load as input and loss ratio as output (duration as input is optional). This text uses mostly packet-centric language.

9. MLRsearch Overview

The main properties of MLRsearch:

* 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:
    - o Target PLRs (e.g. 0.0 and 0.005 when searching for NDR and PDR).
    - o Final trial duration.
    - o Measurement resolution.
* Initial Phase:
  - Measure MRR over initial trial duration.
  - Measured MRR is used as an input to the first intermediate phase.
* Multiple Intermediate Phases:
  - Trial duration:
    - o Start with initial trial duration in the first intermediate phase.
    - o Converge geometrically towards the final trial duration.
  - Track all previous trial measurement results:
    - o Duration, offered load and loss ratio are tracked.
    - o Effective loss ratios are tracked.

+ While in practice, real loss ratios can decrease with increasing load, effective loss ratios never decrease. This is achieved by sorting results by load, and using the effective loss ratio of the previous load if the current loss ratio is smaller than that.

o The algorithm queries the results to find best lower and upper bounds.

+ Effective loss ratios are always used.

o The phase ends if all target loss ratios have tight enough bounds.

- Search:

o Iterate over target loss ratios in increasing order.

o If both upper and lower bound are in measurement results for this duration, apply bisect until the bounds are tight enough, and continue with next loss ratio.

o If a bound is missing for this duration, but there exists a bound from the previous duration (compatible with the other bound at this duration), re-measure at the current duration.

o If a bound in one direction (upper or lower) is missing for this duration, and the previous duration does not have a compatible bound, compute the current "interval size" from the second tightest bound in the other direction (lower or upper respectively) for the current duration, and choose next offered load for external search.

o The logic guarantees that a measurement is never repeated with both duration and offered load being the same.

o The logic guarantees that measurements for higher target loss ratio iterations (still within the same phase duration) do not affect validity and tightness of bounds for previous target loss ratio iterations (at the same duration).

- Use of internal and external searches:

o External search:

+ It is a variant of "exponential search".

+ The "interval size" is multiplied by a configurable constant (powers of two work well with the subsequent internal search).

o Internal search:

+ A variant of binary search that measures at offered load between the previously found bounds.

+ The interval does not need to be split into exact halves, if other split can get to the target width goal faster.

* The idea is to avoid returning interval narrower than the current width goal. See sample implementation details, below.

* Final Phase:

- Executed with the final test trial duration, and the final width goal that determines resolution of the overall search.

* Intermediate Phases together with the Final Phase are called Non-Initial Phases.

* The returned bounds stay within prescribed min_rate and max_rate.

- When returning min_rate or max_rate, the returned bounds may be invalid.

o E.g. upper bound at max_rate may come from a measurement with loss ratio still not higher than the target loss ratio.

The main benefits of MLRsearch vs. binary search include:

* In general, MLRsearch is likely to execute more trials overall, but likely less trials at a set final trial duration.

* 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.

* In all cases MLRsearch yields the same or similar results to binary search.

* Note: both binary search and MLRsearch are susceptible to reporting non-repeatable results across multiple runs for very bad behaving cases.

Caveats:

* Worst case MLRsearch can take longer than a binary search, e.g. in case of drastic changes in behaviour for trials at varying durations.

  - Re-measurement at higher duration can trigger a long external search. That never happens in binary search, which uses the final duration from the start.

10. Sample Implementation

Following is a brief description of a sample MLRsearch implementation, which is a simplified version of the existing implementation.

10.1. Input Parameters

1. *max_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. *min_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_ratios* - list of maximum acceptable PLR search criteria.

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.

10.2. 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. Received rate is computed as intended load multiplied by pass ratio (which is one minus loss ratio). This is useful when loss ratio is computed from a different metric than intended load. For example, intended load can be in transactions (multiple packets each), but loss ratio is computed on level of packets, not transactions.

   * Example: If MTR is 10 transactions per second, and each transaction has 10 packets, and receive rate is 90 packets per second, then loss rate is 10%, and MRR is computed to be 9 transactions per second.

   If MRR is too close to MTR, MRR is set below MTR so that interval width is equal to the width goal of the first intermediate phase. If MRR is less than min_rate, min_rate is used.

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 MRR2 is less than min_rate, min_rate is used. If loss ratio is less or equal to the smallest target loss ratio, MRR2 is set to a value 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.

* OUT: MRR3 = measured receive rate. If MRR3 is less than min_rate, min_rate is used. If step 3 is not skipped, the first trial measurement is forgotten. This is done because in practice (if MRR2 is above MRR), external search from MRR and MRR2 is likely to lead to a faster intermediate phase than a bisect between MRR2 and MTR.

10.3. Non-Initial Phases

1. Main phase 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: Measurement results from the previous phase (previous duration).

    4. Internal target ratio loop:

        1. IN: Target loss ratio for this iteration of ratio loop.

        2. IN: Measurement results from all previous ratio loop iterations of current phase (current duration).

        3. DO: According to the procedure described in point 2:

            1. either exit the phase (by jumping to 1.5),

2. or exit loop iteration (by continuing with next target loss ratio, jumping to 1.4.1),

3. or calculate new transmit rate to measure with.

4. DO: Perform the trial measurement at the new transmit rate and current trial duration, compute its loss ratio.

5. DO: Add the result and go to next iteration (1.4.1), including the added trial result in 1.4.2.

5. OUT: Measurement results from this phase.

6. OUT: In the final phase, bounds for each target loss ratio are extracted and returned.

1. If a valid bound does not exist, use min_rate or max_rate.

2. New transmit rate (or exit) calculation (for point 1.4.3):

1. If the previous duration has the best upper and lower bound, select the middle point as the new transmit rate.

1. See 2.5.3. below for the exact splitting logic.

2. This can be a no-op if interval is narrow enough already, in that case continue with 2.2.

3. Discussion, assuming the middle point is selected and measured:

1. Regardless of loss rate measured, the result becomes either best upper or best lower bound at current duration.

2. So this condition is satisfied at most once per iteration.

3. This also explains why previous phase has double width goal:

1. We avoid one more bisection at previous phase.

2. At most one bound (per iteration) is re-measured with current duration.

3. Each re-measurement can trigger an external search.

4. Such surprising external searches are the main hurdle in achieving low overall search durations.

5. Even without 1.1, there is at most one external search per phase and target loss ratio.

6. But without 1.1 there can be two re-measurements, each coming with a risk of triggering external search.

2. If the previous duration has one bound best, select its transmit rate. In deterministic case this is the last measurement needed this iteration.

3. If only upper bound exists in current duration results:

    1. This can only happen for the smallest target loss ratio.

    2. If the upper bound was measured at min_rate, exit the whole phase early (not investigating other target loss ratios).

    3. Select new transmit rate using external search:

        1. For computing previous interval size, use:

            1. second tightest bound at current duration,

            2. or tightest bound of previous duration, if compatible and giving a more narrow interval,

            3. or target interval width if none of the above is available.

            4. In any case increase to target interval width if smaller.

        2. Quadruple the interval width.

        3. Use min_rate if the new transmit rate is lower.

4. If only lower bound exists in current duration results:

    1. If the lower bound was measured at max_rate, exit this iteration (continue with next lowest target loss ratio).

2. Select new transmit rate using external search:

    1. For computing previous interval size, use:

        1. second tightest bound at current duration,

        2. or tightest bound of previous duration, if compatible and giving a more narrow interval,

        3. or target interval width if none of the above is available.

        4. In any case increase to target interval width if smaller.

    2. Quadruple the interval width.

    3. Use max_rate if the new transmit rate is higher.

5. The only remaining option is both bounds in current duration results.

    1. This can happen in two ways, depending on how the lower bound was chosen.

        1. It could have been selected for the current loss ratio, e.g. in re-measurement (2.2) or in initial bisect (2.1).

        2. It could have been found as an upper bound for the previous smaller target loss ratio, in which case it might be too low.

        3. The algorithm does not track which one is the case, as the decision logic works well regardless.

    2. Compute "extending down" candidate transmit rate exactly as in 2.3.

    3. Compute "bisecting" candidate transmit rate:

        1. Compute the current interval width from the two bounds.

        2. Express the width as a (float) multiple of the target width goal for this phase.

3. If the multiple is not higher than one, it means the width goal is met. Exit this iteration and continue with next higher target loss ratio.

4. If the multiple is two or less, use half of that for new width if the lower subinterval.

5. Round the multiple up to nearest even integer.

6. Use half of that for new width if the lower subinterval.

7. Example: If lower bound is 2.0 and upper bound is 5.0, and width goal is 1.0, the new candidate transmit rate will be 4.0. This can save a measurement when 4.0 has small loss. Selecting the average (3.5) would never save a measurement, giving more narrow bounds instead.

4. If either candidate computation want to exit the iteration, do as bisecting candidate computation says.

5. The remaining case is both candidates wanting to measure at some rate. Use the higher rate. This prefers external search down narrow enough interval, competing with perfectly sized lower bisect subinterval.

11. 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].

11.1. 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. 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_).

3. Intended count.

   * The number of packets to send during the trial should be equal to the intended load multiplied by the duration.

     - Also multiplied by a coefficient, if loss ratio is calculated from a different metric.

       o Example: If a successful transaction uses 10 packets, load is given in transactions per second, but loss ratio is calculated from packets, so the coefficient to get intended count of packets is 10.

   * But in practice that does not work.

     - It could result in a fractional number of packets,

     - so it has to be rounded in a way traffic generator chooses,

     - which may depend on the number of traffic flows and traffic generator worker threads.

4. Attempted count. As the real number of intended packets is not known exactly, the computation uses the number of packets traffic generator reports as sent. Unless overridden by the next point.

5. Duration stretching.

   * In some cases, traffic generator may get overloaded, causing it to take significantly longer (than duration) to send all packets.

   * The implementation uses an explicit stop,

    - causing lower attempted count in those cases.

  * The implementation tolerates some small difference between attempted count and intended count.

    - 10 microseconds worth of traffic is sufficient for our tests.

  * If the difference is higher, the unsent packets are counted as lost.

    - This forces the search to avoid the regions of high duration stretching.

    - The final bounds describe the performance of not just SUT, but of the whole system, including the traffic generator.

6. Excess packets.

  * In some test (e.g. using TCP flows) Traffic generator reacts to packet loss by retransmission. Usually, such packet loss is already affecting loss ratio. If a test also wants to treat retransmissions due to heavily delayed packets also as a failure, this is once again visible as a mismatch between the intended count and the attempted count.

  * The CSIT implementation simply looks at absolute value of the difference, so it offers the same small tolerance before it starts marking a "loss".

7. For result processing, we use lower bounds and ignore upper bounds.

11.1.1. FD.io CSIT Input Parameters

1. *max_rate* - Typical values: 2 * 14.88 Mpps for 64B 10GE link rate, 2 * 18.75 Mpps for 64B 40GE NIC (specific model).

2. *min_rate* - Value: 2 * 9001 pps (we reserve 9000 pps for latency measurements).

3. *final_trial_duration* - Value: 30.0 seconds.

4. *initial_trial_duration* - Value: 1.0 second.

5. *final_relative_width* - Value: 0.005 (0.5%).

6. *packet_loss_ratios* - Value: 0.0, 0.005 (0.0% for NDR, 0.5% for PDR).

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: 600.0 (seconds).

9. *expansion_coefficient* - Width multiplier for external search. Value: 4.0 (interval width is quadroupled). Value of 2.0 is best for well-behaved SUTs, but value of 4.0 has been found to decrease overall search time for worse-behaved SUT configurations, contributing more to the overall set of different SUT configurations tested.

11.2. Example MLRsearch Run

The following list describes a search from a real test run in CSIT (using the default input values as above).

* Initial phase, trial duration 1.0 second.

Measurement 1, intended load 18750000.0 pps (MTR), measured loss ratio 0.7089514628479618 (valid upper bound for both NDR and PDR).

Measurement 2, intended load 5457160.071600716 pps (MRR), measured loss ratio 0.018650817320118702 (new tightest upper bounds).

Measurement 3, intended load 5348832.933500009 pps (slightly less than MRR2 in preparation for first intermediate phase target interval width), measured loss ratio 0.00964383362905351 (new tightest upper bounds).

* First intermediate phase starts, trial duration still 1.0 seconds.

Measurement 4, intended load 4936605.579021453 pps (no lower bound, performing external search downwards, for NDR), measured loss ratio 0.0 (valid lower bound for both NDR and PDR).

Measurement 5, intended load 5138587.208637197 pps (bisecting for NDR), measured loss ratio 0.0 (new tightest lower bounds).

Measurement 6, intended load 5242656.244044665 pps (bisecting), measured loss ratio 0.013523745379347257 (new tightest upper bounds).

* Both intervals are narrow enough.

* Second intermediate phase starts, trial duration 5.477225575051661 seconds.

Measurement 7, intended load 5190360.904111567 pps (initial bisect for NDR), measured loss ratio 0.0023533920869969953 (NDR upper bound, PDR lower bound).

Measurement 8, intended load 5138587.208637197 pps (re-measuring NDR lower bound), measured loss ratio 1.2080222912800403e-06 (new tightest NDR upper bound).

* The two intervals have separate bounds from now on.

Measurement 9, intended load 4936605.381062318 pps (external NDR search down), measured loss ratio 0.0 (new valid NDR lower bound).

Measurement 10, intended load 5036583.888432355 pps (NDR bisect), measured loss ratio 0.0 (new tightest NDR lower bound).

Measurement 11, intended load 5087329.903232804 pps (NDR bisect), measured loss ratio 0.0 (new tightest NDR lower bound).

* NDR interval is narrow enough, PDR interval not ready yet.

Measurement 12, intended load 5242656.244044665 pps (re-measuring PDR upper bound), measured loss ratio 0.0101174866190136 (still valid PDR upper bound).

* Also PDR interval is narrow enough, with valid bounds for this duration.

* Final phase starts, trial duration 30.0 seconds.

Measurement 13, intended load 5112894.3238511775 pps (initial bisect for NDR), measured loss ratio 0.0 (new tightest NDR lower bound).

Measurement 14, intended load 5138587.208637197 (re-measuring NDR upper bound), measured loss ratio 2.030389804256833e-06 (still valid PDR upper bound).

* NDR interval is narrow enough, PDR interval not yet.

Measurement 15, intended load 5216443.04126728 pps (initial bisect for PDR), measured loss ratio 0.005620871287975237 (new tightest PDR upper bound).

Measurement 16, intended load 5190360.904111567 (re-measuring PDR lower bound), measured loss ratio 0.0027629971184465604 (still valid PDR lower bound).

* PDR interval is also narrow enough.

* Returning bounds:

* NDR_LOWER = 5112894.3238511775 pps; NDR_UPPER = 5138587.208637197 pps;

* PDR_LOWER = 5190360.904111567 pps; PDR_UPPER = 5216443.04126728 pps.

12. IANA Considerations

No requests of IANA.

13. 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.

14. Acknowledgements

Many thanks to Alec Hothan of OPNFV NFVbench project for thorough review and numerous useful comments and suggestions.

15. References

15.1. 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>.

15.2. Informative References

[FDio-CSIT-MLRsearch] "FD.io CSIT Test Methodology - MLRsearch", November 2021, <https://s3-docs.fd.io/csit/rls2110/report/introduction/methodology_data_plane_throughput/methodology_data_plane_throughput.html#mlrsearch-tests>.

[PyPI-MLRsearch] "MLRsearch 0.4.0, Python Package Index", April 2021, <https://pypi.org/project/MLRsearch/0.4.0/>.

Authors' Addresses

Maciek Konstantynowicz (editor)
Cisco Systems
Email: mkonstan@cisco.com

Vratko Polak
Cisco Systems
Email: vrpolak@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 5 September 2024

V. Vassilev
Lightside Instruments AS

4 March 2024

A YANG Data Model for Network Tester Management
draft-ietf-bmwg-network-tester-cfg-04

Abstract

This document introduces new YANG model for use in network interconnect testing containing modules of traffic generator and traffic analyzer.

Status of This Memo

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

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

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

This Internet-Draft will expire on 5 September 2024.

Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction

There is a need for standard mechanism to allow the specification and implementation of the transactions part of network tests. The mechanism should allow the control and monitoring of the data plane traffic in a transactional way. This document defines two YANG modules for test traffic generator and analyzer.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) defined in RFC 8342.

1.1. Terminology

1.1.1. Definitions and Acronyms

DUT: Device Under Test

TA: Traffic Analyzer

TG: Traffic Generator

1.1.2. Tree Diagram

For a reference to the annotations used in tree diagrams included in this document, please see YANG Tree Diagrams [RFC8340].

1.2. Problem Statement

Network interconnect tests require active network elements part of the tested network that generate test traffic and network elements that analyze the test traffic at one or more points of its path. A network interconnect tester is a device that can either generate test traffic, analyze test traffic or both. Here is a figure borrowed from [RFC2544] representing the horseshoe test setup topology consisting of a single tester and a single DUT connected in a network interconnect loop.

```
                  +------------+
                  |            |
    +-------------|  tester    |<-------------+
    |             |            |              |
    |             +------------+              |
    |                                         |
    |             +------------+              |
    |             |            |              |
    +------------>|    DUT     |--------------+
                  |            |
                  +------------+
```

This document attempts to address the problem of defining YANG model of a network interconnect tester that can be used for development of vendor independent network interconnect tests and utilize the advantages of transactional management using standard protocols like NETCONF.

1.3. Objectives

This section specifies the design objectives for the model. It should:

* provide means to specify the generated traffic as streams of cyclic sequence of bursts with configurable frame size, frame data, interframe gap and interburst gap.

* have a mandatory single stream mode and optional multi stream mode.

* provide means for configuration of traffic streams with static frame data where frames with identical frame data are sent during the lifetime of the stream.

* provide means for configuration of traffic streams with dynamic frame data where frames contain fields with dynamic data like generation time and sequence number.

* allow third parties to augment the base module with alternative dynamic fields of frame data extensions.

* provide means for realtime synchronization and orchestration of the generated streams.

* provide counters for received test traffic frames and octets.

* provide latency statistic in the case of test traffic with dynamic frame data that includes timestamp.

* provide sequence number errors in the case of test traffic with dynamic frame data that includes sequence number.

* provide means for capturing traffic frames data.

1.4. Solution

The proposed model splits the design into 2 modules - 1) Traffic Generator module (TG), 2) Traffic Analyzer module (TA). The modules are implemented as augmentations of the ietf-interfaces [RFC8343] module adding configuration and state data that models the functionality of a network interconnect tester. The TA and TG modules concept is illustrated with the following diagram of a tester with two interfaces (named e0 and e1) connected in a loop with single DUT:

```
                  +----------------+
     e0.egress    |                | e1.ingress
  +------------   | TG  tester  TA |<-------------+
  |               |                |              |
  |               +----------------+              |
  |                                               |
  |                +------------+                 |
  |                |            |                 |
  +------------->  |    DUT     |-----------------+
                   |            |
                   +------------+
```

2. Using the network interconnect tester model

Basic example of how the model can be used in transactional network test program to control the testers part of a network and report counter statistics and timing measurement data is presented in Appendix A. All example cases present the configuration and state data from a single test trial. The search algorithm logic that operates to control the trial configuration is outside the scope of this document. One of the examples demonstrates the use of the [RFC2544] defined testframe packet.

3. Traffic Generator Module Tree Diagram

```
module: ietf-traffic-generator
  augment /if:interfaces/if:interface:
    +--rw traffic-generator
       +--rw (type)?
       |  +--:(single-stream)
       |  |  +--rw testframe-type?     identityref
       |  |  +--rw frame-size          uint32
       |  |  +--rw frame-data?         string
       |  |  +--rw interframe-gap      uint32
       |  |  +--rw interburst-gap?     uint32
       |  |  +--rw frames-per-burst?   uint32
       |  |  +--rw modifiers
       |  |     +--rw modifier* [id]
       |  |        +--rw id             uint32
       |  |        +--rw action         identityref
       |  |        +--rw offset         uint32
       |  |        +--rw mask           string
       |  |        +--rw repetitions    uint32
       |  +--:(multi-stream)
       |     +--rw streams
       |        +--rw stream* [id]
       |           +--rw id                   uint32
       |           +--rw testframe-type?      identityref
       |           +--rw frame-size           uint32
       |           +--rw frame-data?          string
       |           +--rw interframe-gap       uint32
       |           +--rw interburst-gap?      uint32
       |           +--rw frames-per-burst?    uint32
       |           +--rw frames-per-stream    uint32
       |           +--rw interstream-gap      uint32
       |           +--rw modifiers
       |              +--rw modifier* [id]
       |                 +--rw id             uint32
       |                 +--rw action         identityref
       |                 +--rw offset         uint32
       |                 +--rw mask           string
       |                 +--rw repetitions    uint32
       +--rw realtime-epoch?
       |        yang:date-and-time {realtime-epoch}?
       +--rw total-frames?
```

4. Traffic Analyzer Module Tree Diagram

```
module: ietf-traffic-analyzer
  augment /if:interfaces/if:interface:
    +--rw traffic-analyzer!
       +--rw testframe-filter! {testframe-filter}?
       |  +--rw type    identityref
       |  +--rw mask?   string
       |  +--rw data?   string
       +--rw capture {capture}?
       |  +--rw start-trigger
       |  |  +--rw (start-trigger)?
       |  |     +--:(frame-index)
       |  |     |  +--rw frame-index?       uint64
       |  |     +--:(testframe-index)
       |  |        +--rw testframe-index?   uint64
       |  +--rw stop-trigger
       |     +--rw (stop-trigger)?
       |        +--:(when-full)
       |           +--rw when-full?   empty
       +--ro state
          +--ro pkts?              yang:counter64
          +--ro octets?            yang:counter64
          +--ro idle-octets?       yang:counter64 {idle-octets-counter}?
          +--ro errors?            yang:counter64
          +--ro testframe-stats
          |  +--ro testframe-pkts?    yang:counter64
          |  +--ro sequence-errors?   yang:counter64
          |  +--ro payload-errors?    yang:counter64
          |  +--ro latency
          |     +--ro samples?   uint64
          |     +--ro min?       uint64
          |     +--ro max?       uint64
          |     +--ro average?   uint64
          |     +--ro latest?    uint64
          +--ro capture {capture}?
             +--ro frame* [sequence-number]
                +--ro sequence-number    uint64
                +--ro timestamp?         yang:date-and-time
                +--ro length?            uint32
                +--ro data?              string
```

5. Traffic Generator Module YANG

<CODE BEGINS> file "ietf-traffic-generator@2024-03-04.yang"

```
module ietf-traffic-generator {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-traffic-generator";
  prefix nttg;

  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model For Interface Management";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  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) 2024 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 Revised BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://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 2024-03-04 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for
       Network Tester Management";
  }
```

```
    feature multi-stream {
      description
        "The device can generate multi-stream traffic.";
    }

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

    identity testframe-type {
      description
        "Base identity for all testframe types.";
    }

    identity static {
      base testframe-type;
      description
        "Identity for static testframe.
         The frame data and size are constant.";
    }

    identity dynamic {
      base testframe-type;
      description
        "Identity to be used as base for dynamic
         testframe type identities defined
         in external modules.

         When used itself it identifies dynamic testframe
         where the last 18 octets of the payload contain
         incrementing sequence number field (8 octets)
         followed by timestamp field in the
         IEEE 1588-2008 format (10 octets). If frame data is defined
         for the last 18 octets of the payload it will be ignored
         and overwritten with dynamic data according to this
         specification.";
    }

    identity modifier-action-type {
      description
        "Base identity for all modifier action types.";
    }

    identity increment {
      base modifier-action-type;
      description
```

```
        "Identity for increment modifier action.";
    }

    identity decrement {
      base modifier-action-type;
      description
        "Identity for decrement modifier action.";
    }

    identity random {
      base modifier-action-type;
      description
        "Identity for random modifier action.";
    }

    grouping common-data {
      description
        "Common configuration data.";
      leaf realtime-epoch {
        if-feature "realtime-epoch";
        type yang:date-and-time;
        description
          "If this leaf is present the stream generation will start
           at the specified realtime epoch.";
      }
      leaf total-frames {
        type uint64;
        description
          "If this leaf is present the traffic generation will stop
           after the specified number of frames are generated.";
      }
    }

    grouping burst-data {
      description
        "Generated traffic burst parameters.";
      leaf testframe-type {
        type identityref {
          base nttg:testframe-type;
        }
        default "nttg:static";
        description
          "In case of dynamic testframes this leaf specifies
           the dynamic testframe identity.";
      }
      leaf frame-size {
        type uint32;
        mandatory true;
```

```
        description
          "Size of the frames generated. For example for
           Ethernet interfaces the following definition
           applies:

           Ethernet frame-size in octets includes:
           * Destination Address (6 octets),
           * Source Address (6 octets),
           * Frame Type (2 octets),
           * Data (min 46 octets or 42 octets + 4 octets 802.1Q tag),
           * CRC Checksum (4 octets).

           Ethernet frame-size does not include:
           * Preamble (dependent on MAC configuration
                      by default 7 octets),
           * Start of frame delimiter (1 octet)

           Minimum standard Ethernet frame-size is 64 bytes but
           generators might support smaller sizes for validation.";
      }
      leaf frame-data {
        type string {
          pattern '([0-9A-F]{2})*';
        }
        description
          "The raw  frame data specified as hexadecimal string.
           The string length of the specified data can be
           shorter then the 2*../frame-size value
           specifying only the header or the header and the
           payload with or without the 4 byte CRC Checksum
           in the case of a Ethernet frame. If the frame data string
           specified is longer then ../frame-size*2 the excess
           of the specified frame data is trunkated.

           In multi-stream mode if frame-data leaf for one of the
           streams is not specified, it is the value specified for
           the closest stream that precedes it that is used.";
      }
      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 modifier-data {
      description
        "Modifier parameters.";
      container modifiers {
        description
          "Container holding the configured modifiers list.";
        list modifier {
          key "id";
          description
            "Each modifier specifies action to be performed
             on data at certain offset.";
          leaf id {
            type uint32;
            description
              "Number specifying the identifier of the modifier.";
          }
          leaf action {
            type identityref {
              base nttg:modifier-action-type;
            }
            mandatory true;
            description
              "In case of dynamic testframes this leaf specifies
               the dynamic testframe identity.";
          }
```

```
          leaf offset {
            type uint32;
            mandatory true;
            description
              "Offset in octets of the modified data of the frame.";
          }
          leaf mask {
            type string {
              pattern '([0-9A-F]{2})*';
            }
            mandatory true;
            description
              "Bit mask of the actual bits affected by the modifier.";
          }
          leaf repetitions {
            type uint32;
            mandatory true;
            description
              "Count of the packets that will repeat the data before
               the modifier makes the next update.";
          }
        }
      }
    }

    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.";
          }
          uses modifier-data;
        }
      }
    }

    augment "/if:interfaces/if:interface" {
      description
        "Traffic generator augmentations of ietf-interfaces.";
      container traffic-generator {
        description
          "Traffic generator configuration data.";
        choice type {
          description
            "Choice of the type of the data model of the generator.
             Single or multi stream.";
          case single-stream {
            uses burst-data;
            uses modifier-data;
          }
          case multi-stream {
            uses multi-stream-data;
          }
        }
        uses common-data;
      }
    }
  }

   <CODE ENDS>
```

6. Traffic Analyzer Module YANG

```
   <CODE BEGINS> file "ietf-traffic-analyzer@2024-03-04.yang"

 module ietf-traffic-analyzer {
   yang-version 1.1;
   namespace "urn:ietf:params:xml:ns:yang:ietf-traffic-analyzer";
   prefix ntta;

   import ietf-interfaces {
     prefix if;
```

```
    reference
      "RFC 8343: A YANG Data Model For Interface Management";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  organization
    "IETF Benchmarking Methodology Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/bmwg/>
     WG List:  <mailto:bmwg@ietf.org>

     Editor:   Vladimir Vassilev
               <mailto:vladimir@lightside-instruments.com>";
  description
    "This module contains a collection of YANG definitions for
     description and management of network interconnect testers.

     Copyright (c) 2024 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 Revised BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://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 2024-03-04 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for
       Network Tester Management";
  }

  feature testframe-filter {
    description
      "This feature indicates that the device implements
       filter that can specify a subset of packets to be
       analyzed as testframes.";
  }
```

```
   feature idle-octets-counter {
     description
       "This feature indicates that the device implements
        idle-octets counter that accumulates the time
        the link is not utilized. The minimum required
        idle gaps are not counted as idle octets.";
   }
   feature capture {
     description
       "This feature indicates that the device implements
        packet capture functionality.";
   }

   identity testframe-filter {
     description
       "Base testframe-filter identity.";
   }

   identity bit-field-match {
     base ntta:testframe-filter;
     description
       "Bit field matching filter according to template data and mask.";
   }

   grouping statistics-data {
     description
       "Analyzer statistics.";
     leaf pkts {
       type yang:counter64;
       description
         "Total number of packets analyzed.";
     }
     leaf octets {
       type yang:counter64;
       description
         "This counter is identical with the in-octets/out-octets
          counters defined in RFC8343 except that it counts the
          octets since the analyzer was created.";
     }
     leaf idle-octets {
       if-feature "idle-octets-counter";
       type yang:counter64;
       description
         "Total accumulated period with no frame transmission
          taking place measured in octets at the current link
          speed. Octets not counted in ../octets but not idle are
          for example layer 1 framing octets - for Ethernet interfaces
          7+1 preamble octets per packet.";
```

```
     }
     leaf errors {
       type yang:counter64;
       description
         "Count of packets with errors.
          Not counted in the pkts or captured.
          For example packets with CRC error.";
     }
     container testframe-stats {
       description
         "Statistics for received testframes containing
          either sequence number, payload checksum,
          timestamp or any combination of these features.";
       leaf pkts {
         type yang:counter64;
         description
           "Total count of detected testframes.";
       }
       leaf sequence-errors {
         type yang:counter64;
         description
           "Total count of testframes 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 testframes with
            payload errors.";
       }
       container latency {
         description
           "Latency statistics.";
         leaf samples {
           type uint64;
           description
             "Total count of packets used for estimating
              the latency statistics. Ideally
              samples=../testframe-stats.";
         }
         leaf min {
           type uint64;
           units "nanoseconds";
           description
             "Minimum measured latency.";
         }
```

```
        leaf max {
          type uint64;
          units "nanoseconds";
          description
            "Maximum measured latency.";
        }
        leaf average {
          type uint64;
          units "nanoseconds";
          description
            "The sum of all sampled latencies divided
             by the number of samples.";
        }
        leaf latest {
          type uint64;
          units "nanoseconds";
          description
            "Latency of the latest sample.";
        }
      }
    }
  }

  grouping capture-config-data {
    description
      "Grouping with a capture configuration container.";
    container capture {
      if-feature "capture";

      description
        "Contains capture parameters.";

      container start-trigger {
          description
            "Configures when the capture start is triggered.";
        choice start-trigger {
          description
            "If none of the cases in this choice are configured the
             capture process starts from the first frame received.";
          case frame-index {
            description
              "Start capturing frames at the specified frame index.";
            leaf frame-index {
              type uint64;
              description
                "First captured frame index.";
            }
          }
```

```
          case testframe-index {
            description
              "Start capturing frames at the specified
               testframe index.";
            leaf testframe-index {
              type uint64;
              description
                "Starts capture at specified testframe index.";
            }
          }
        }
      }
      container stop-trigger {
          description
            "Configures when the capture is stopped.";
        choice stop-trigger {
          description
            "If none of the cases in this choice are configured the
             captured frames are always the last frames received for
             as many frames the implementation can buffer.";
          case when-full {
            description
              "Stops capturing when the implementation can not store
               more frames.";
            leaf when-full {
                type empty;
                description
                  "When present in configuration capture stops when
                   the capture buffer is full.";
            }
          }
        }
      }
    }
  }

  grouping capture-data {
    description
      "Grouping with statistics and data
       of one or more captured frame.";
    container capture {
      if-feature "capture";
      description
        "Statistics and data of
         one or more captured frames.";
      list frame {
        key "sequence-number";
        description
```

```
          "Statistics and data of a captured frame.";
        leaf sequence-number {
          type uint64;
          description
            "Incremental counter of frames captured.";
        }
        leaf timestamp {
          type yang:date-and-time;
          description
            "Timestamp of the moment the frame was captured.";
        }
        leaf length {
          type uint32;
          description
            "Frame length. Ideally the data captured will be
             of the same length but can be shorter
             depending on implementation limitations.";
        }
        leaf 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 testframe-filter {
      if-feature "testframe-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 ntta:testframe-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 {
      presence "Enables the traffic analyzer.";
      description
        "Traffic analyzer for ingress direction.";
      uses filter-data;
      uses capture-config-data;
      container state {
        config false;
        description
          "State data.";
        uses statistics-data;
        uses capture-data;
      }
    }
  }

  augment "/if:interfaces/if:interface/ntta:traffic-analyzer/"
        + "ntta:testframe-filter" {
    when "derived-from-or-self(ntta:type, 'ntta:bit-field-match')";
    description
      "Logical AND of masked bit fields.";
    leaf mask {
      type string {
        pattern '([0-9A-F]{2})*';
      }
      description
        "Specifies bit field mask for comparison.
         Non-masked bit fields are ignored.";
    }
    leaf data {
      type string {
        pattern '([0-9A-F]{2})*';
      }
      description
        "Specify data to be matched according to the specified mask.";
    }
  }
}
```

<CODE ENDS>

7. IANA Considerations

This document registers two URIs and two YANG modules.

7.1. URI Registration

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-traffic-generator
URI: urn:ietf:params:xml:ns:yang:ietf-traffic-analyzer

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

7.2. YANG Module Name Registration

This document registers two YANG module in the YANG Module Names registry YANG [RFC6020].

name: ietf-traffic-generator
namespace: urn:ietf:params:xml:ns:yang:ietf-traffic-generator
prefix: nttg
reference: RFC XXXX

name: ietf-traffic-analyzer
namespace: urn:ietf:params:xml:ns:yang:ietf-traffic-analyzer
prefix: ntta
reference: RFC XXXX

8. Security Considerations

The YANG modules defined in this document are designed to be accessed via the NETCONF protocol RFC 6241 [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory to implement secure transport is SSH RFC 6242 [RFC6242]. The NETCONF access control model RFC 6536 [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of data nodes defined in this YANG module which are writable/creatable/deletable (i.e. config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g. edit-config) to

these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

8.1. ietf-traffic-generator.yang

The ietf-traffic-generator YANG module controls a stateless traffic generator which is intended to be used for testing and verification purposes but can be used for malicious purposes like generating network traffic part of a Denial-of-Service (DoS) attack. This should be taken into consideration when granting write access to the following container and descendant data nodes:

* /if:interfaces/if:interface/nttg:traffic-generator

8.2. ietf-traffic-analyzer.yang

The ietf-traffic-analyzer YANG module controls a traffic analyzer which is designed for use in testing and verification but can be used for reading information contained in packets sent and received on any of the interfaces on systems that implement the capture feature. This should be taken into consideration when granting read access to the following container and descendant data nodes:

* /if:interfaces/if:interface/ntta:traffic-analyzer/ntta:capture

9. References

9.1. Normative References

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", RFC 7224, DOI 10.17487/RFC7224, May 2014, <https://www.rfc-editor.org/info/rfc7224>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

9.2. Informative References

[IEEE1588] IEEE, "IEEE 1588-2008", 2008.

[IEEE802.3-2014] IEEE WG802.3 - Ethernet Working Group, "IEEE 802.3-2014", 2014.

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

Appendix A. Examples

The following topology will be used for the examples in this section:

```
+-------------+          +------------+         +------------+
|             | e0    e0 |            | e1   e0 |            |
| tester0   TG|>-------->|    dut0    |>------->|TA  tester1 |
|             |          |            |         |            |
+-------------+          +------------+         +------------+
```

A.1. Basic Test Program

This pseudo code program orchestrates a network test and shows how the model can be used:

```
#Connect to network
net=connect("topology.xml")

# Configure DUTs and enable traffic-analyzers
net.node("dut0").edit( \
  "create /interfaces/interface[name='e0'] -- type=ethernetCsmacd")
net.node("dut0").edit(
  "create /interfaces/interface[name='e1'] -- type=ethernetCsmacd")
net.node("dut0").edit(
  "create /flows/flow[id='t0'] -- match/in-port=e0 "
  "actions/action[order='0']/output-action/out-port=e1")

net.node("tester1").edit(
  "create /interfaces/interface[name='e0']/traffic-analyzer")
net.commit()

#Get network state - before
before=net.get()

# Start traffic
net.node("tester0).edit(
  "create /interfaces/interface[name='e0']/traffic-generator -- "
  "frame-size=64 interframe-gap=20")

net.commit()

time.sleep(60)

# Stop traffic
net.node("tester1").edit("delete /interfaces/interface[name='e0']/"
                         "traffic-generator")
net.commit()

#Get network state - after
after=net.get()

#Report
sent_pkts=delta("tester0",before,after,
  "/interfaces/interface[name='e0']/statistics/out-unicast-pkts")

received_pkts=delta("tester1",before,after,
  "/interfaces/interface[name='e0']/statistics/in-unicast-pkts")
```

```
latency_max=absolute(after,
  "/interfaces/interface[name='e0']/traffic-analyzer/state/"
  "testframe-stats/latency/max")

#Cleanup
net.node("tester1").edit(
   "delete /interfaces/interface/traffic-analyzer")
net.node("dut0").edit("delete /flows")
net.node("dut0").edit("delete /interfaces")
net.commit()
```

A.2. Generating RFC2544 Testframes

In sec. C.2.6.4 Test Frames a detailed format is specified. The frame-data leaf allows full control over the generated frames payload.

```
...
net.node("tester1").edit(
  "merge /interfaces/interface[name='e0']/"
  "traffic-generator -- frame-data="
  "6CA96F0000026CA96F00000108004500"
  "002ED4A500000A115816C0000201C000"
  "0202C0200007001A0000010203040506"
  "0708090A0B0C0D0E0F101112")
...
```

A.3. Generating frames with dynamic data fields

Adding modifier to single stream traffic generator configuration. In this example the the highest 16 bits of the source MAC address of the generated Ethernet frame will be incremented.

```
...
net.node("tester1").edit(
  "merge /interfaces/interface[name='e0']/"
  "traffic-generator/modifiers/modifier[id='0'] -- "
  "offset=6 mask=FFFF action=increment repetitions=1")
...
```

Author's Address

Vladimir Vassilev
Lightside Instruments AS
Email: vladimir@lightside-instruments.com

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

B. Balarajah

C. Rossenhoevel
EANTC AG

B. Monkman
NetSecOPEN

January 2022

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 5 July 2022.

Copyright Notice

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

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

Table of Contents

1. Introduction

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

2. Requirements

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

3. Scope

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

4. Test Setup

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

4.1. Testbed Configuration

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

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

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

```
+-----------------------+                   +-----------------------+
| +-------------------+ |   +-----------+   | +-------------------+ |
| | Emulated Router(s)| |   |           |   | | Emulated Router(s)| |
| |    (Optional)     | +----- DUT/SUT  +-----+    (Optional)     | |
| +-------------------+ |   |           |   | +-------------------+ |
| +-------------------+ |   +-----------+   | +-------------------+ |
| |     Clients       | |                   | |      Servers      | |
| +-------------------+ |                   | +-------------------+ |
|                       |                   |                       |
|   Test Equipment      |                   |   Test Equipment      |
+-----------------------+                   +-----------------------+
```

Figure 1: Testbed Setup - Option 1

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

```
    +-------------------+      +-----------+      +--------------------+
    |Aggregation Switch/|      |           |      | Aggregation Switch/|
    | Router            +------+  DUT/SUT  +------+ Router             |
    |                   |      |           |      |                    |
    +----------+--------+      +-----------+      +--------+-----------+
               |                                          |
               |                                          |
   +-----------+-----------+                   +-----------+-----------+
   |                       |                   |                       |
   | +-------------------+ |                   | +-------------------+ |
   | | Emulated Router(s)| |                   | | Emulated Router(s)| |
   | |     (Optional)    | |                   | |     (Optional)    | |
   | +-------------------+ |                   | +-------------------+ |
   | +-------------------+ |                   | +-------------------+ |
   | |      Clients      | |                   | |      Servers      | |
   | +-------------------+ |                   | +-------------------+ |
   |                       |                   |                       |
   |    Test Equipment     |                   |    Test Equipment     |
   +-----------------------+                   +-----------------------+
```

Figure 2: Testbed Setup - Option 2

4.2. DUT/SUT Configuration

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

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

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

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

Table 1: NGFW Security Features

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

Table 2: NGIPS Security Features

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

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

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

Table 3: Security Feature Description

Below is a summary of the DUT/SUT configuration:

* DUT/SUT MUST be configured in "inline" mode.

* "Fail-Open" behavior MUST be disabled.

* All RECOMMENDED security features are enabled.

* Logging SHOULD be enabled. DUT/SUT SHOULD log all traffic at the flow level - Logging to an external device is permissible.

* Geographical location filtering, and Application Identification and Control SHOULD be configured to trigger based on a site or application from the defined traffic mix.

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

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

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

Figure 3: DUT/SUT Access List

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

4.2.1. Security Effectiveness Configuration

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

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

4.3. Test Equipment Configuration

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

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

4.3.1. Client Configuration

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

4.3.1.1. TCP Stack Attributes

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

4.3.1.2. Client IP Address Space

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

* The IP blocks SHOULD consist of multiple unique, discontinuous static address blocks.

* A default gateway is permitted.

* The DSCP (differentiated services code point) marking is set to DF (Default Forwarding) ’000000’ on IPv4 Type of Service (ToS) field and IPv6 traffic class field.

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

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

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

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

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

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

(Option 1) 100 % IPv4, no IPv6

(Option 2) 80 % IPv4, 20% IPv6

(Option 3) 50 % IPv4, 50% IPv6

(Option 4) 20 % IPv4, 80% IPv6

(Option 5) no IPv4, 100% IPv6

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

4.3.1.3. Emulated Web Browser Attributes

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

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

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

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

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

1. ECDHE-ECDSA-AES128-GCM-SHA256 with Prime256v1 (Signature Hash Algorithm: ecdsa_secp256r1_sha256 and Supported group: secp256r1)

2. ECDHE-RSA-AES128-GCM-SHA256 with RSA 2048 (Signature Hash Algorithm: rsa_pkcs1_sha256 and Supported group: secp256r1)

3. ECDHE-ECDSA-AES256-GCM-SHA384 with Secp521 (Signature Hash Algorithm: ecdsa_secp384r1_sha384 and Supported group: secp521r1)

4. ECDHE-RSA-AES256-GCM-SHA384 with RSA 4096 (Signature Hash Algorithm: rsa_pkcs1_sha384 and Supported group: secp256r1)

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

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

1. TLS_AES_128_GCM_SHA256

2. TLS_AES_256_GCM_SHA384

3. TLS_CHACHA20_POLY1305_SHA256

4. TLS_AES_128_CCM_SHA256

5. TLS_AES_128_CCM_8_SHA256

4.3.2. Backend Server Configuration

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

4.3.2.1. TCP Stack Attributes

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

4.3.2.2. Server Endpoint IP Addressing

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

* The server IP blocks SHOULD consist of unique, discontinuous static address blocks with one IP per server Fully Qualified Domain Name (FQDN) endpoint per test port.

* A default gateway is permitted. The DSCP (differentiated services code point) marking is set to DF (Default Forwarding) '000000' on IPv4 Type of Service (ToS) field and IPv6 traffic class field.

* The server IP addresses SHOULD be distributed between IPv4 and IPv6 with a ratio identical to the clients distribution ratio.

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

4.3.2.3. HTTP / HTTPS Server Pool Endpoint Attributes

The server pool for HTTP SHOULD listen on TCP port 80 and emulate the same HTTP version (HTTP 1.1 or HTTP/2 or HTTP/3) and settings chosen by the client (emulated web browser). The Server MUST advertise server type in the Server response header [RFC7230]. For HTTPS server, TLS 1.2 or higher MUST be used with a maximum record size of 16 KByte and MUST NOT use ticket resumption or session ID reuse. The server SHOULD listen on TCP port 443 for HTTP version 1.1 and 2. For HTTP/3 (HTTP over QUIC) the server SHOULD listen on UDP 443. The server SHALL serve a certificate to the client. The HTTPS server MUST check host SNI information with the FQDN if SNI is in use. Cipher suite and key size on the server side MUST be configured similar to the client side configuration described in Section 4.3.1.3.

4.3.3. Traffic Flow Definition

This section describes the traffic pattern between client and server endpoints. At the beginning of the test, the server endpoint initializes and will be ready to accept connection states including initialization of the TCP stack as well as bound HTTP and HTTPS servers. When a client endpoint is needed, it will initialize and be given attributes such as a MAC and IP address. The behavior of the client is to sweep through the given server IP space, generating a recognizable service by the DUT. Sequential and pseudorandom sweep methods are acceptable. The method used MUST be stated in the final report. Thus, a balanced mesh between client endpoints and server endpoints will be generated in a client IP and port to server IP and port combination. Each client endpoint performs the same actions as other endpoints, with the difference being the source IP of the client endpoint and the target server IP pool. The client MUST use the server IP address or FQDN in the host header [RFC7230].

4.3.3.1. Description of Intra-Client Behavior

Client endpoints are independent of other clients that are concurrently executing. When a client endpoint initiates traffic, this section describes how the client steps through different services. Once the test is initialized, the client endpoints randomly hold (perform no operation) for a few milliseconds for better randomization of the start of client traffic. Each client will either open a new TCP connection or connect to a TCP persistence stack still open to that specific server. At any point that the traffic profile may require encryption, a TLS encryption tunnel will form presenting the URL or IP address request to the server. If using SNI, the server MUST then perform an SNI name check with the proposed FQDN compared to the domain embedded in the certificate. Only when correct, will the server process the HTTPS response object. The initial response object to the server is based on benchmarking tests described in Section 7. Multiple additional sub-URLs (response objects on the service page) MAY be requested simultaneously. This MAY be to the same server IP as the initial URL. Each sub-object will also use a canonical FQDN and URL path, as observed in the traffic mix used.

4.3.4. Traffic Load Profile

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

1. Init phase: Testbed devices including the client and server endpoints should negotiate layer 2-3 connectivity such as MAC learning and ARP. Only after successful MAC learning or ARP/ND resolution SHALL the test iteration move to the next phase. No measurements are made in this phase. The minimum RECOMMENDED time for Init phase is 5 seconds. During this phase, the emulated clients SHOULD NOT initiate any sessions with the DUT/SUT, in contrast, the emulated servers should be ready to accept requests from DUT/SUT or from emulated clients.

2. Ramp up phase: The test equipment SHOULD start to generate the test traffic. It SHOULD use a set of the approximate number of unique client IP addresses to generate traffic. The traffic SHOULD ramp up from zero to desired target objective. The target objective is defined for each benchmarking test. The duration for the ramp up phase MUST be configured long enough that the test equipment does not overwhelm the DUT/SUTs stated performance metrics defined in Section 6.3 namely, TCP Connections Per Second, Inspected Throughput, Concurrent TCP Connections, and Application Transactions Per Second. No measurements are made in this phase.

3. Sustain phase: Starts when all required clients are active and operating at their desired load condition. In the sustain phase, the test equipment SHOULD continue generating traffic to constant target value for a constant number of active clients. The minimum RECOMMENDED time duration for sustain phase is 300 seconds. This is the phase where measurements occur. The test equipment SHOULD measure and record statistics continuously. The sampling interval for collecting the raw results and calculating the statistics SHOULD be less than 2 seconds.

4. Ramp down phase: No new connections are established, and no measurements are made. The time duration for ramp up and ramp down phase SHOULD be the same.

5. Collection phase: The last phase is administrative and will occur when the test equipment merges and collates the report data.

5. Testbed Considerations

This section describes steps for a reference test (pre-test) that control the test environment including test equipment, focusing on physical and virtualized environments and as well as test equipments. Below are the RECOMMENDED steps for the reference test.

1. Perform the reference test either by configuring the DUT/SUT in the most trivial setup (fast forwarding) or without presence of the DUT/SUT.

2. Generate traffic from traffic generator. Choose a traffic profile used for HTTP or HTTPS throughput performance test with smallest object size.

3. Ensure that any ancillary switching or routing functions added in the test equipment does not limit the performance by introducing network metrics such as packet loss and latency. This is specifically important for virtualized components (e.g., vSwitches, vRouters).

4. Verify that the generated traffic (performance) of the test equipment matches and reasonably exceeds the expected maximum performance of the DUT/SUT.

5. Record the network performance metrics packet loss latency introduced by the test environment (without DUT/SUT).

6. Assert that the testbed characteristics are stable during the entire test session. Several factors might influence stability specifically, for virtualized testbeds. For example, additional workloads in a virtualized system, load balancing, and movement of virtual machines during the test, or simple issues such as additional heat created by high workloads leading to an emergency CPU performance reduction.

The reference test SHOULD be performed before the benchmarking tests (described in section 7) start.

6. Reporting

This section describes how the benchmarking test report should be formatted and presented. It is RECOMMENDED to include two main sections in the report, namely the introduction and the detailed test results sections.

6.1. Introduction

The following attributes SHOULD be present in the introduction section of the test report.

1. The time and date of the execution of the tests

2. Summary of testbed software and hardware details

a. DUT/SUT hardware/virtual configuration

* This section SHOULD clearly identify the make and model of the DUT/SUT

* The port interfaces, including speed and link information

* If the DUT/SUT is a Virtual Network Function (VNF), host (server) hardware and software details, interface acceleration type such as DPDK and SR-IOV, used CPU cores, used RAM, resource sharing (e.g. Pinning details and NUMA Node) configuration details, hypervisor version, virtual switch version

* details of any additional hardware relevant to the DUT/SUT such as controllers

b. DUT/SUT software

* Operating system name

* Version

* Specific configuration details (if any)

c. DUT/SUT enabled features

* Configured DUT/SUT features (see Table 1 and Table 2)

* Attributes of the above-mentioned features

* Any additional relevant information about the features

d. Test equipment hardware and software

* Test equipment vendor name

* Hardware details including model number, interface type

* Test equipment firmware and test application software version

e. Key test parameters

* Used cipher suites and keys

* IPv4 and IPv6 traffic distribution

* Number of configured ACL

f. Details of application traffic mix used in the benchmarking test "Throughput Performance with Application Traffic Mix" (Section 7.1)

* Name of applications and layer 7 protocols

* Percentage of emulated traffic for each application and layer 7 protocols

* Percentage of encrypted traffic and used cipher suites and keys (The RECOMMENDED ciphers and keys are defined in Section 4.3.1.3)

* Used object sizes for each application and layer 7 protocols

3. Results Summary / Executive Summary

a. Results SHOULD resemble a pyramid in how it is reported, with the introduction section documenting the summary of results in a prominent, easy to read block.

6.2. Detailed Test Results

In the result section of the test report, the following attributes SHOULD be present for each benchmarking test.

a. KPIs MUST be documented separately for each benchmarking test. The format of the KPI metrics SHOULD be presented as described in Section 6.3.

b. The next level of details SHOULD be graphs showing each of these metrics over the duration (sustain phase) of the test. This allows the user to see the measured performance stability changes over time.

6.3. Benchmarks and Key Performance Indicators

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

* Concurrent TCP Connections

The aggregate number of simultaneous connections between hosts across the DUT/SUT, or between hosts and the DUT/SUT (defined in [RFC2647]).

* TCP Connections Per Second

  The average number of successfully established TCP connections per second between hosts across the DUT/SUT, or between hosts and the DUT/SUT. The TCP connection MUST be initiated via a TCP three-way handshake (SYN, SYN/ACK, ACK). Then the TCP session data is sent. The TCP session MUST be closed via either a TCP three-way close (FIN, FIN/ACK, ACK), or a TCP four-way close (FIN, ACK, FIN, ACK), and MUST NOT by RST.

* Application Transactions Per Second

  The average number of successfully completed transactions per second. For a particular transaction to be considered successful, all data MUST have been transferred in its entirety. In case of HTTP(S) transactions, it MUST have a valid status code (200 OK), and the appropriate FIN, FIN/ACK sequence MUST have been completed.

* TLS Handshake Rate

  The average number of successfully established TLS connections per second between hosts across the DUT/SUT, or between hosts and the DUT/SUT.

* Inspected Throughput

  The number of bits per second of examined and allowed traffic a network security device is able to transmit to the correct destination interface(s) in response to a specified offered load. The throughput benchmarking tests defined in Section 7 SHOULD measure the average Layer 2 throughput value when the DUT/SUT is "inspecting" traffic. This document recommends presenting the inspected throughput value in Gbit/s rounded to two places of precision with a more specific Kbit/s in parenthesis.

* Time to First Byte (TTFB)

  TTFB is the elapsed time between the start of sending the TCP SYN packet from the client and the client receiving the first packet of application data from the server or DUT/SUT. The benchmarking tests HTTP Transaction Latency (Section 7.4) and HTTPS Transaction Latency (Section 7.8) measure the minimum, average and maximum TTFB. The value SHOULD be expressed in milliseconds.

* URL Response time / Time to Last Byte (TTLB)

  URL Response time / TTLB is the elapsed time between the start of sending the TCP SYN packet from the client and the client receiving the last packet of application data from the server or DUT/SUT. The benchmarking tests HTTP Transaction Latency (Section 7.4) and HTTPS Transaction Latency (Section 7.8) measure the minimum, average and maximum TTLB. The value SHOULD be expressed in millisecond.

7. Benchmarking Tests

7.1. Throughput Performance with Application Traffic Mix

7.1.1. Objective

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

Based on the test customer’s specific use case, testers can choose the relevant application traffic mix for this test. The details about the traffic mix MUST be documented in the report. At least the following traffic mix details MUST be documented and reported together with the test results:

Name of applications and layer 7 protocols

Percentage of emulated traffic for each application and layer 7 protocol

Percentage of encrypted traffic and used cipher suites and keys (The RECOMMENDED ciphers and keys are defined in Section 4.3.1.3.)

Used object sizes for each application and layer 7 protocols

7.1.2. Test Setup

Testbed setup MUST be configured as defined in Section 4. Any benchmarking test specific testbed configuration changes MUST be documented.

7.1.3. Test Parameters

In this section, the benchmarking test specific parameters SHOULD be defined.

7.1.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific benchmarking test MUST be documented. In case the DUT/SUT is configured without SSL inspection, the test report MUST explain the implications of this to the relevant application traffic mix encrypted traffic.

7.1.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target inspected throughput: Aggregated line rate of interface(s) used in the DUT/SUT or the value defined based on requirement for a specific deployment scenario

Initial throughput: 10% of the "Target inspected throughput" Note: Initial throughput is not a KPI to report. This value is configured on the traffic generator and used to perform Step 1: "Test Initialization and Qualification" described under the Section 7.1.4.

One of the ciphers and keys defined in Section 4.3.1.3 are RECOMMENDED to use for this benchmarking test.

7.1.3.3. Traffic Profile

Traffic profile: This test MUST be run with a relevant application traffic mix profile.

7.1.3.4. Test Results Validation Criteria

The following criteria are the test results validation criteria. The test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

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

b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.

7.1.3.5. Measurement

Following KPI metrics MUST be reported for this benchmarking test:

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

Note: TTLB MUST be reported along with the object size used in the traffic profile.

Optional KPIs: TCP Connections Per Second and TLS Handshake Rate

7.1.4. Test Procedures and Expected Results

The test procedures are designed to measure the inspected throughput performance of the DUT/SUT at the sustaining period of traffic load profile. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (initial throughput) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IP types: IPv4 only, IPv6 only, and IPv4 and IPv6 mixed traffic distribution.

7.1.4.1. Step 1: Test Initialization and Qualification

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

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

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

7.1.4.2. Step 2: Test Run with Target Objective

Configure test equipment to generate traffic at the "Target inspected throughput" rate defined in Section 7.1.3.2. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4. The test equipment SHOULD start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective ("Target inspected throughput") in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.1.4.3. Step 3: Test Iteration

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

7.2. TCP/HTTP Connections Per Second

7.2.1. Objective

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

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

7.2.2. Test Setup

Testbed setup SHOULD be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.2.3. Test Parameters

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

7.2.3.1. DUT/SUT Configuration Parameters

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

7.2.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target connections per second: Initial value from product datasheet or the value defined based on requirement for a specific deployment scenario

Initial connections per second: 10% of "Target connections per second" (Note: Initial connections per second is not a KPI to report. This value is configured on the traffic generator and used to perform the Step1: "Test Initialization and Qualification" described under the Section 7.2.4.

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

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

7.2.3.3. Test Results Validation Criteria

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

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

b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.

c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

d. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.

7.2.3.4. Measurement

TCP Connections Per Second MUST be reported for each test iteration (for each object size).

7.2.4. Test Procedures and Expected Results

The test procedure is designed to measure the TCP connections per second rate of the DUT/SUT at the sustaining period of the traffic load profile. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial connections per second) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IP types: IPv4 only, IPv6 only, and IPv4 and IPv6 mixed traffic distribution.

7.2.4.1. Step 1: Test Initialization and Qualification

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

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

The DUT/SUT SHOULD reach the "Initial connections per second" before the sustain phase. The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Section 7.2.3.3.

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

7.2.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish the target objective ("Target connections per second") defined in Section 7.2.3.2. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4.

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

The test equipment SHOULD start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective ("Target connections per second") in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.2.4.3. Step 3: Test Iteration

Determine the achievable TCP connections per second within the test results validation criteria.

7.3. HTTP Throughput

7.3.1. Objective

Determine the sustainable inspected throughput of the DUT/SUT for HTTP transactions varying the HTTP response object size.

7.3.2. Test Setup

Testbed setup SHOULD be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.3.3. Test Parameters

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

7.3.3.1. DUT/SUT Configuration Parameters

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

7.3.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target inspected throughput: Aggregated line rate of interface(s) used in the DUT/SUT or the value defined based on requirement for a specific deployment scenario

Initial throughput: 10% of "Target inspected throughput" Note: Initial throughput is not a KPI to report. This value is configured on the traffic generator and used to perform Step 1: "Test Initialization and Qualification" described under Section 7.3.4.

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

RECOMMENDED HTTP response object size: 1, 16, 64, 256 KByte, and mixed objects defined in Table 4.

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

Table 4: Mixed Objects

7.3.3.3. Test Results Validation Criteria

The following criteria are the test results validation criteria. The test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

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

b. Traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

c. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.

7.3.3.4. Measurement

Inspected Throughput and HTTP Transactions per Second MUST be reported for each object size.

7.3.4. Test Procedures and Expected Results

The test procedure is designed to measure HTTP throughput of the DUT/SUT. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial throughput) and meets the test results validation criteria when it was very minimal utilized. Step 2 determines the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution and HTTP response object sizes.

7.3.4.1. Step 1: Test Initialization and Qualification

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

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

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

The measured KPIs during the sustain phase MUST meet the test results validation criteria "a" defined in Section 7.3.3.3. The test results validation criteria "b" and "c" are OPTIONAL for step 1.

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

7.3.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish the target objective ("Target inspected throughput") defined in Section 7.3.3.2. The test equipment SHOULD start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.3.4.3. Step 3: Test Iteration

Determine the achievable inspected throughput within the test results validation criteria and measure the KPI metric Transactions per Second. Final test iteration MUST be performed for the test duration defined in Section 4.3.4.

7.4. HTTP Transaction Latency

7.4.1. Objective

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

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

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

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

7.4.2. Test Setup

Testbed setup SHOULD be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.4.3. Test Parameters

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

7.4.3.1. DUT/SUT Configuration Parameters

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

7.4.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

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

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

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

Note: The Initial objectives are not a KPI to report. These values are configured on the traffic generator and used to perform the Step1: "Test Initialization and Qualification" described under the Section 7.4.4.

HTTP transaction per TCP connection: Test scenario 1 with single transaction and test scenario 2 with 10 transactions.

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

7.4.3.3. Test Results Validation Criteria

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

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

b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.

c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

d. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.

e. After ramp up the DUT MUST achieve the "Target objective" defined in Section 7.4.3.2 and remain in that state for the entire test duration (sustain phase).

7.4.3.4. Measurement

TTFB (minimum, average, and maximum) and TTLB (minimum, average and maximum) MUST be reported for each object size.

7.4.4. Test Procedures and Expected Results

The test procedure is designed to measure TTFB or TTLB when the DUT/SUT is operating close to 50% of its maximum achievable connections per second or inspected throughput. The test procedure consists of two major steps: Step 1 ensures the DUT/SUT is able to reach the initial performance values and meets the test results validation criteria when it was very minimally utilized. Step 2 measures the latency values within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IP types (IPv4 only, IPv6 only and IPv4 and IPv6 mixed traffic distribution), HTTP response object sizes and single and multiple transactions per connection scenarios.

7.4.4.1. Step 1: Test Initialization and Qualification

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

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

The DUT/SUT SHOULD reach the "Initial objective" before the sustain phase. The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Section 7.4.3.3.

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

7.4.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish "Target objective" defined in Section 7.4.3.2. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4.

The test equipment SHOULD start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT MUST reach the desired value of the target objective in the sustain phase.

Measure the minimum, average, and maximum values of TTFB and TTLB.

7.5. Concurrent TCP/HTTP Connection Capacity

7.5.1. Objective

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

7.5.2. Test Setup

Testbed setup SHOULD be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.5.3. Test Parameters

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

7.5.3.1. DUT/SUT Configuration Parameters

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

7.5.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target concurrent connection: Initial value from product datasheet or the value defined based on requirement for a specific deployment scenario.

Initial concurrent connection: 10% of "Target concurrent connection" Note: Initial concurrent connection is not a KPI to report. This value is configured on the traffic generator and used to perform the Step1: "Test Initialization and Qualification" described under the Section 7.5.4.

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

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

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

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

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

X = ("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 criteria are the test results validation criteria. The Test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

a. Number of failed application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transaction) of total attempted transactions.

b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.

c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

7.5.3.4. Measurement

Average Concurrent TCP Connections MUST be reported for this benchmarking test.

7.5.4. Test Procedures and Expected Results

The test procedure is designed to measure the concurrent TCP connection capacity of the DUT/SUT at the sustaining period of traffic load profile. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial concurrent connection) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution.

7.5.4.1. Step 1: Test Initialization and Qualification

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

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

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

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

7.5.4.2. Step 2: Test Run with Target Objective

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

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

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

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.5.4.3. Step 3: Test Iteration

Determine the achievable concurrent TCP connections capacity within the test results validation criteria.

7.6. TCP/HTTPS Connections per Second

7.6.1. Objective

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

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

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

7.6.2. Test Setup

Testbed setup SHOULD be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.6.3. Test Parameters

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

7.6.3.1. DUT/SUT Configuration Parameters

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

7.6.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target connections per second: Initial value from product datasheet or the value defined based on requirement for a specific deployment scenario.

Initial connections per second: 10% of "Target connections per second" Note: Initial connections per second is not a KPI to report. This value is configured on the traffic generator and used to perform the Step1: "Test Initialization and Qualification" described under the Section 7.6.4.

RECOMMENDED ciphers and keys defined in Section 4.3.1.3

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

7.6.3.3. Test Results Validation Criteria

The following criteria are the test results validation criteria. The test results validation criteria MUST be monitored during the whole test duration.

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

b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.

c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

d. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.

7.6.3.4. Measurement

TCP connections per second MUST be reported for each test iteration (for each object size).

The KPI metric TLS Handshake Rate can be measured in the test using 1 KByte object size.

7.6.4. Test Procedures and Expected Results

The test procedure is designed to measure the TCP connections per second rate of the DUT/SUT at the sustaining period of traffic load profile. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial connections per second) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution.

7.6.4.1. Step 1: Test Initialization and Qualification

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

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

The DUT/SUT SHOULD reach the "Initial connections per second" before the sustain phase. The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Section 7.6.3.3.

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

7.6.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish "Target connections per second" defined in Section 7.6.3.2. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4.

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

The test equipment SHOULD start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective ("Target connections per second") in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.6.4.3. Step 3: Test Iteration

Determine the achievable connections per second within the test results validation criteria.

7.7. HTTPS Throughput

7.7.1. Objective

Determine the sustainable inspected throughput of the DUT/SUT for HTTPS transactions varying the HTTPS response object size.

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

7.7.2. Test Setup

Testbed setup SHOULD be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.7.3. Test Parameters

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

7.7.3.1. DUT/SUT Configuration Parameters

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

7.7.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Target inspected throughput: Aggregated line rate of interface(s) used in the DUT/SUT or the value defined based on requirement for a specific deployment scenario.

Initial throughput: 10% of "Target inspected throughput" Note: Initial throughput is not a KPI to report. This value is configured on the traffic generator and used to perform the Step1: "Test Initialization and Qualification" described under the Section 7.7.4.

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

RECOMMENDED ciphers and keys defined in Section 4.3.1.3

RECOMMENDED HTTPS response object size: 1, 16, 64, 256 KByte, and mixed objects defined in Table 4 under Section 7.3.3.2.

7.7.3.3. Test Results Validation Criteria

The following criteria are the test results validation criteria. The test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

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

b. Traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

c. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.

7.7.3.4. Measurement

Inspected Throughput and HTTP Transactions per Second MUST be reported for each object size.

7.7.4. Test Procedures and Expected Results

The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial throughput) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution and HTTPS response object sizes.

7.7.4.1. Step 1: Test Initialization and Qualification

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

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

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

The measured KPIs during the sustain phase MUST meet the test results validation criteria "a" defined in Section 7.7.3.3. The test results validation criteria "b" and "c" are OPTIONAL for step 1.

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

7.7.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish the target objective ("Target inspected throughput") defined in Section 7.7.3.2. The test equipment SHOULD start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.7.4.3. Step 3: Test Iteration

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

7.8. HTTPS Transaction Latency

7.8.1. Objective

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

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

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

7.8.2. Test Setup

Testbed setup SHOULD be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.8.3. Test Parameters

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

7.8.3.1. DUT/SUT Configuration Parameters

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

7.8.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

RECOMMENDED cipher suites and key sizes defined in Section 4.3.1.3

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

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

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

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

Note: The Initial objectives are not a KPI to report. These values are configured on the traffic generator and used to perform the Step1: "Test Initialization and Qualification" described under the Section 7.8.4.

HTTPS transaction per TCP connection: Test scenario 1 with single transaction and scenario 2 with 10 transactions

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

7.8.3.3. Test Results Validation Criteria

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

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

b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.

c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

d. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.

e. After ramp up the DUT/SUT MUST achieve the "Target objective" defined in the parameter Section 7.8.3.2 and remain in that state for the entire test duration (sustain phase).

7.8.3.4. Measurement

TTFB (minimum, average, and maximum) and TTLB (minimum, average and maximum) MUST be reported for each object size.

7.8.4. Test Procedures and Expected Results

The test procedure is designed to measure TTFB or TTLB when the DUT/SUT is operating close to 50% of its maximum achievable connections per second or inspected throughput. The test procedure consists of two major steps: Step 1 ensures the DUT/SUT is able to reach the initial performance values and meets the test results validation criteria when it was very minimally utilized. Step 2 measures the latency values within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IP types (IPv4 only, IPv6 only and IPv4 and IPv6 mixed traffic distribution), HTTPS response object sizes and single, and multiple transactions per connection scenarios.

7.8.4.1. Step 1: Test Initialization and Qualification

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

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

The DUT/SUT SHOULD reach the "Initial objective" before the sustain phase. The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Section 7.8.3.3.

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

7.8.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish "Target objective" defined in Section 7.8.3.2. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4.

The test equipment SHOULD start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT MUST reach the desired value of the target objective in the sustain phase.

Measure the minimum, average, and maximum values of TTFB and TTLB.

7.9. Concurrent TCP/HTTPS Connection Capacity

7.9.1. Objective

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

7.9.2. Test Setup

Testbed setup SHOULD be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.9.3. Test Parameters

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

7.9.3.1. DUT/SUT Configuration Parameters

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

7.9.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

RECOMMENDED cipher suites and key sizes defined in Section 4.3.1.3

Target concurrent connections: Initial value from product datasheet or the value defined based on requirement for a specific deployment scenario.

Initial concurrent connections: 10% of "Target concurrent connections" Note: Initial concurrent connection is not a KPI to report. This value is configured on the traffic generator and used to perform the Step1: "Test Initialization and Qualification" described under the Section 7.9.4.

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

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

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

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

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

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

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

7.9.3.3. Test Results Validation Criteria

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

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

b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.

c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

7.9.3.4. Measurement

Average Concurrent TCP Connections MUST be reported for this benchmarking test.

7.9.4. Test Procedures and Expected Results

The test procedure is designed to measure the concurrent TCP connection capacity of the DUT/SUT at the sustaining period of traffic load profile. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial concurrent connection) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution.

7.9.4.1. Step 1: Test Initialization and Qualification

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

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

During the sustain phase, the DUT/SUT SHOULD reach the "Initial concurrent TCP connections". The measured KPIs during the sustain phase MUST meet the test results validation criteria "a" and "b" defined in Section 7.9.3.3.

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

7.9.4.2. Step 2: Test Run with Target Objective

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

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

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

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.9.4.3. Step 3: Test Iteration

Determine the achievable concurrent TCP connections within the test results validation criteria.

8. IANA Considerations

This document makes no specific request of IANA.

The IANA has assigned IPv4 and IPv6 address blocks in [RFC6890] that have been registered for special purposes. The IPv6 address block 2001:2::/48 has been allocated for the purpose of IPv6 Benchmarking [RFC5180] and the IPv4 address block 198.18.0.0/15 has been allocated for the purpose of IPv4 Benchmarking [RFC2544]. This assignment was made to minimize the chance of conflict in case a testing device were to be accidentally connected to part of the Internet.

9. Security Considerations

The primary goal of this document is to provide benchmarking terminology and methodology for next-generation network security devices for use in a laboratory isolated test environment. However, readers should be aware that there is some overlap between performance and security issues. Specifically, the optimal configuration for network security device performance may not be the most secure, and vice-versa. The cipher suites recommended in this document are for test purpose only. The cipher suite recommendation for a real deployment is outside the scope of this document.

10. Contributors

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

Alex Samonte, Amritam Putatunda, Aria Eslambolchizadeh, Chao Guo, Chris Brown, Cory Ford, David DeSanto, Jurrie Van Den Breekel, Michelle Rhines, Mike Jack, Ryan Liles, Samaresh Nair, Stephen Goudreault, Tim Carlin, and Tim Otto.

11. Acknowledgements

The authors wish to acknowledge the members of NetSecOPEN for their participation in the creation of this document. Additionally, the following members need to be acknowledged:

Anand Vijayan, Chris Marshall, Jay Lindenauer, Michael Shannon, Mike Deichman, Ryan Riese, and Toulnay Orkun.

12. References

12.1. Normative References

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

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

12.2. Informative References

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

[RFC2647] Newman, D., "Benchmarking Terminology for Firewall Performance", RFC 2647, DOI 10.17487/RFC2647, August 1999, <https://www.rfc-editor.org/info/rfc2647>.

[RFC3511] Hickman, B., Newman, D., Tadjudin, S., and T. Martin, "Benchmarking Methodology for Firewall Performance", RFC 3511, DOI 10.17487/RFC3511, April 2003, <https://www.rfc-editor.org/info/rfc3511>.

[RFC5180] Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, DOI 10.17487/RFC5180, May 2008, <https://www.rfc-editor.org/info/rfc5180>.

[RFC6815] Bradner, S., Dubray, K., McQuaid, J., and A. Morton, "Applicability Statement for RFC 2544: Use on Production Networks Considered Harmful", RFC 6815, DOI 10.17487/RFC6815, November 2012, <https://www.rfc-editor.org/info/rfc6815>.

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <https://www.rfc-editor.org/info/rfc6890>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

Appendix A. Test Methodology - Security Effectiveness Evaluation

A.1. Test Objective

This test methodology verifies the DUT/SUT is able to detect, prevent, and report the vulnerabilities.

In this test, background test traffic will be generated to utilize the DUT/SUT. In parallel, the CVEs will be sent to the DUT/SUT as encrypted and as well as clear text payload formats using a traffic generator. The selection of the CVEs is described in Section 4.2.1.

The following KPIs are measured in this test:

* Number of blocked CVEs

* Number of bypassed (nonblocked) CVEs

* Background traffic performance (verify if the background traffic is impacted while sending CVE toward DUT/SUT)

* Accuracy of DUT/SUT statistics in term of vulnerabilities reporting

A.2. Testbed Setup

The same testbed MUST be used for security effectiveness test and as well as for benchmarking test cases defined in Section 7.

A.3. Test Parameters

In this section, the benchmarking test specific parameters SHOULD be defined.

A.3.1. DUT/SUT Configuration Parameters

DUT/SUT configuration parameters MUST conform to the requirements defined in Section 4.2. The same DUT configuration MUST be used for Security effectiveness test and as well as for benchmarking test cases defined in Section 7. The DUT/SUT MUST be configured in inline mode and all detected attack traffic MUST be dropped and the session SHOULD be reset

A.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. The same client and server IP ranges MUST be configured as used in the benchmarking test cases. In addition, the following parameters MUST be documented for this benchmarking test:

* Background Traffic: 45% of maximum HTTP throughput and 45% of Maximum HTTPS throughput supported by the DUT/SUT (measured with object size 64 KByte in the benchmarking tests "HTTP(S) Throughput" defined in Section 7.3 and Section 7.7).

* RECOMMENDED CVE traffic transmission Rate: 10 CVEs per second

* It is RECOMMENDED to generate each CVE multiple times (sequentially) at 10 CVEs per second

* Ciphers and keys for the encrypted CVE traffic MUST use the same cipher configured for HTTPS traffic related benchmarking tests (Section 7.6 - Section 7.9)

A.4. Test Results Validation Criteria

The following criteria are the test results validation criteria. The test results validation criteria MUST be monitored during the whole test duration.

a. Number of failed application transaction in the background traffic MUST be less than 0.01% of attempted transactions.

b. Number of terminated TCP connections of the background traffic (due to unexpected TCP RST sent by DUT/SUT) MUST be less than 0.01% of total initiated TCP connections in the background traffic.

c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

d. False positive MUST NOT occur in the background traffic.

A.5. Measurement

Following KPI metrics MUST be reported for this test scenario:

Mandatory KPIs:

* Blocked CVEs: It SHOULD be represented in the following ways:

  - Number of blocked CVEs out of total CVEs

  - Percentage of blocked CVEs

* Unblocked CVEs: It SHOULD be represented in the following ways:

  - Number of unblocked CVEs out of total CVEs

  - Percentage of unblocked CVEs

* Background traffic behavior: It SHOULD be represented one of the followings ways:

  - No impact: Considered as "no impact’" if any deviation of traffic forwarding rate is less than or equal to 5 % (constant rate)

  - Minor impact: Considered as "minor impact" if any deviation of traffic forwarding rate is greater than 5% and less than or equal to10% (i.e. small spikes)

  - Heavily impacted: Considered as "Heavily impacted" if any deviation of traffic forwarding rate is greater than 10% (i.e. large spikes) or reduced the background HTTP(S) throughput greater than 10%

* DUT/SUT reporting accuracy: DUT/SUT MUST report all detected vulnerabilities.

Optional KPIs:

* List of unblocked CVEs

A.6. Test Procedures and Expected Results

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

A.6.1. Step 1: Background Traffic

Generate background traffic at the transmission rate defined in Appendix A.3.2.

The DUT/SUT MUST reach the target objective (HTTP(S) throughput) in sustain phase. The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Appendix A.4.

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

A.6.2. Step 2: CVE Emulation

While generating background traffic (in sustain phase), send the CVE traffic as defined in the parameter section.

The test equipment SHOULD start to measure and record all specified KPIs. Continue the test until all CVEs are sent.

The measured KPIs MUST meet all the test results validation criteria defined in Appendix A.4.

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

Appendix B. DUT/SUT Classification

This document aims to classify the DUT/SUT in four different categories based on its maximum supported firewall throughput performance number defined in the vendor datasheet. This classification MAY help user to determine specific configuration scale (e.g., number of ACL entries), traffic profiles, and attack traffic profiles, scaling those proportionally to DUT/SUT sizing category.

The four different categories are Extra Small (XS), Small (S), Medium (M), and Large (L). The RECOMMENDED throughput values for the following categories are:

Extra Small (XS) - Supported throughput less than or equal to1Gbit/s

Small (S) - Supported throughput greater than 1Gbit/s and less than or equal to 5Gbit/s

Medium (M) - Supported throughput greater than 5Gbit/s and less than or equal to10Gbit/s

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

Authors' Addresses

Balamuhunthan Balarajah
Berlin
Germany

Email: bm.balarajah@gmail.com

Carsten Rossenhoevel
EANTC AG
Salzufer 14
10587 Berlin
Germany

Email: cross@eantc.de

Brian Monkman
NetSecOPEN
417 Independence Court
Mechanicsburg, PA 17050
United States of America

Email: bmonkman@netsecopen.org

Benchmarking Methodology Working Group
Internet-Draft
Obsoletes: 3511 (if approved)
Intended status: Informational
Expires: 25 April 2023

B. Balarajah

C. Rossenhoevel
EANTC AG

B. Monkman
NetSecOPEN

22 October 2022

Benchmarking Methodology for Network Security Device Performance
draft-ietf-bmwg-ngfw-performance-15

Abstract

This document provides benchmarking terminology and methodology for next-generation network security devices including next-generation firewalls (NGFW) and next-generation intrusion prevention systems (NGIPS). The main areas covered in this document are test terminology, test configuration parameters, and benchmarking methodology for NGFW and NGIPS. (It is assumed that readers have a working knowledge of these devices and the security functionality they contain.) This document aims to improve the applicability, reproducibility, and transparency of benchmarks and to align the test methodology with today's increasingly complex layer 7 security-centric network application use cases. As a result, this document makes RFC3511 obsolete.

Status of This Memo

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

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

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

This Internet-Draft will expire on 25 April 2023.

Copyright Notice

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

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

Table of Contents

1. Introduction

18 years have passed since IETF initially recommended test methodology and terminology for firewalls ([RFC3511]). Firewalls have evolved significantly from the days of simple ACL filters. As the underlying technology progresses and improves, recommending test methodology and terminology for firewalls, requirements, and expectations for network security elements has increased tremendously. Security function implementations have evolved and diversified into intrusion detection and prevention, threat management, analysis of encrypted traffic, and more. In an industry of growing importance, well-defined and reproducible key performance indicators (KPIs) are increasingly needed to enable fair and reasonable comparison of network security functions. These reasons led to the creation of a new next-generation network security device benchmarking document, which makes [RFC3511] obsolete. Measurement of performance for processing of IP fragmented traffic (see Section 5.9 of [RFC3511]) was not included in this document since IP fragmentation does today not commonly occur in traffic anymore, unlike it might have been at the time when [RFC3511] was written. It should also be noted that [RFC2647] retains significant value and has been consulted frequently while creating this document.

For a more detailed explanation of what an NGFW is see the Wikipedia article [Wiki-NGFW].

2. Requirements

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

3. Scope

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

The performance testing methodology described in this document is not intended for security devices/systems that rely on machine learning or behavioral analysis. If such features are present in a Device Under Test/System Under Test (DUT/SUT), they should be disabled.

4. Test Setup

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

4.1. Testbed Configuration

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

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

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

```
+-----------------------+                   +-----------------------+
| +-------------------+ |   +-----------+   | +-------------------+ |
| | Emulated Router(s)| |   |           |   | | Emulated Router(s)| |
| |    (Optional)     | +----- DUT/SUT  +-----+    (Optional)     | |
| +-------------------+ |   |           |   | +-------------------+ |
| +-------------------+ |   +-----------+   | +-------------------+ |
| |     Clients       | |                   | |      Servers      | |
| +-------------------+ |                   | +-------------------+ |
|                       |                   |                       |
|   Test Equipment      |                   |   Test Equipment      |
+-----------------------+                   +-----------------------+
```

Figure 1: Testbed Setup - Option 1

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

```
 +-------------------+      +-----------+      +--------------------+
 |Aggregation Switch/|      |           |      | Aggregation Switch/|
 | Router           +------+  DUT/SUT  +------+ Router             |
 |                   |      |           |      |                    |
 +----------+--------+      +-----------+      +--------+-----------+
            |                                           |
            |                                           |
+-----------+-----------+                   +-----------+-----------+
|                       |                   |                       |
| +-------------------+ |                   | +-------------------+ |
| | Emulated Router(s)| |                   | | Emulated Router(s)| |
| |     (Optional)    | |                   | |     (Optional)    | |
| +-------------------+ |                   | +-------------------+ |
| +-------------------+ |                   | +-------------------+ |
| |      Clients      | |                   | |      Servers      | |
| +-------------------+ |                   | +-------------------+ |
|                       |                   |                       |
|    Test Equipment     |                   |    Test Equipment     |
+-----------------------+                   +-----------------------+
```

Figure 2: Testbed Setup - Option 2

4.2. DUT/SUT Configuration

The same DUT/SUT configuration MUST be used for all benchmarking tests described in Section 7. Since each DUT/SUT will have its own unique configuration, users MUST configure their devices with the same parameters and security features that would be used in the actual deployment of the device or a typical deployment. The DUT/SUT MUST be configured in "Inline" mode so that the traffic is actively inspected by the DUT/SUT.

Table 2 and Table 3 below describe the RECOMMENDED and OPTIONAL sets of network security features for NGFW and NGIPS, respectively. If the recommended security features are not enabled in the DUT/SUT for any reason, the reason MUST be reported with the benchmarking test results. For example, one reason for not enabling the anti-virus feature in NGFW may be that this security feature was not required for a particular customer deployment scenario. It MUST be also noted in the benchmarking test report that not enabling the specific recommended security features may impact the performance of the DUT/SUT. The selected security features MUST be consistently enabled on the DUT/SUT for all benchmarking tests described in Section 7.

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

The following table provides a brief description of the security features and these are approximate taxonomies of features commonly found in currently deployed NGFW and NGIDS. The features provided by specific implementations may be named differently and not necessarily have configuration settings that align with the taxonomy.

DUT/SUT Features	Description
TLS Inspection	DUT/SUT intercepts and decrypts inbound HTTPS traffic between servers and clients. Once the content inspection has been completed, DUT/SUT encrypts the HTTPS traffic with ciphers and keys used by the clients and servers. For TLS1.3, the DUT works as a middlebox (proxy) and it holds the certificates and Pre-Shared Keys (PSK) that are trusted by the client and represent the identity of the real server.
IDS/IPS	DUT/SUT detects and blocks exploits targeting known and unknown vulnerabilities across the monitored network.
Anti-Malware	DUT/SUT detects and prevents the transmission of malicious executable code and any associated communications across the monitored network. This includes data exfiltration as well as command and control channels.
Anti-Spyware	Anti-Spyware is a subcategory of Anti Malware. Spyware transmits information without the

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

Table 1: Security Feature Description

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

Table 2: NGFW Security Features

DUT/SUT (NGIPS) Features	RECOMMENDED	OPTIONAL
TLS Inspection	x	
Anti-Malware	x	
Anti-Spyware	x	
Anti-Botnet	x	
Application Identification	x	
Deep Packet Inspection (DPI)	x	
Anti-Evasion	x	

Table 3: NGIPS Security Features

Note: With respect to TLS Inspection, there are scenarios where it will be optional.

Below is a summary of the DUT/SUT configuration:

* DUT/SUT MUST be configured in "inline" mode.

* "Fail-Open" behavior MUST be disabled.

* All RECOMMENDED security features are enabled.

* Logging and reporting MUST be enabled. DUT/SUT SHOULD log all traffic at the flow level (5-tuple). If the DUT/SUT is designed to log all traffic at different levels (e.g. IP packet levels), it is acceptable to conduct tests. However, this MUST be noted in the test report. Logging to an external device is permissible.

* Geographical location filtering SHOULD be configured. If the DUT/SUT is not designed to perform geographical location filtering, it is acceptable to conduct tests without this feature. However, this MUST be noted in the test report.

* Application Identification and Control MUST be configured to trigger applications from the defined traffic mix.

In addition, a realistic number of access control rules (ACL) SHOULD be configured on the DUT/SUT where ACLs are configurable and reasonable based on the deployment scenario. For example, it is acceptable not to configure ACLs in an NGIPS since NGIPS devices do not require the use of ACLs in most deployment scenarios. This document determines the number of access policy rules for four different classes of DUT/SUT: Extra Small (XS), Small (S), Medium (M), and Large (L). A sample DUT/SUT classification is described in Appendix B.

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

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

Figure 3: DUT/SUT Access List

Note 1: Based on the test customer's specific use case, the testers can increase the number of rules.

Note 2: If half of the applications included in the test traffic is less than 10, the missing number of ACL entries (dummy rules) can be configured for any application traffic not included in the test traffic.

Note 3: In the event, the DUT/SUT is designed to not use ACLs it is acceptable to conduct tests without them. However, this MUST be noted in the test report.

4.2.1. Security Effectiveness Configuration

The selected security features (defined in Table 2 and Table 3) of the DUT/SUT MUST be configured effectively to detect, prevent, and report the defined security vulnerability sets. This section defines the selection of the security vulnerability sets from the Common Vulnerabilities and Exposures (CVE) list for testing. The vulnerability set should reflect a minimum of 500 CVEs from no older than 10 calendar years to the current year. These CVEs should be selected with a focus on in-use software commonly found in business applications, with a Common Vulnerability Scoring System (CVSS) Severity of High (7-10).

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

4.3. Test Equipment Configuration

In general, test equipment allows configuring parameters in different protocol layers. Extensive proof of concept tests conducted to support preparation of this document showed that benchmarking results are strongly affected by the choice of protocol stack parameters; especially OSI layer 4 transport protocol parameters. For more information on how TCP and QUIC parameters will impact performance review [fastly]. To achieve reproducible results that will be representative for real deployment scenarios, careful specification and documentation of the parameters are required.

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

4.3.1. Client Configuration

This section specifies which parameters should be considered while configuring emulated client endpoints in the test equipment. Also, this section specifies the RECOMMENDED values for certain parameters. The values are the defaults typically used in most of the client operating system types.

Pre-standard evaluations have shown that it is possible to set a wide range of arbitrary parameters for OSI layer 4 transport protocols on test equipment leading to client-specific results optimization; however, only well-defined common parameter sets help to establish meaningful and comparable benchmarking results. For these reasons, this document recommends specific sets of transport protocol parameters to be configured on test equipment used for benchmarking.

4.3.1.1. TCP Stack Attributes

The TCP stack of the emulated client endpoints MUST fulfill the TCP requirements defined in [RFC9293] (See Appendix B.). In addition, this section specifies the RECOMMENDED values for TCP parameters configured using the following parameters:

The IPv4 and IPv6 Maximum Segment Size (MSS) are set to 1460 bytes and 1440 bytes respectively. TX and RX initial receive window sizes are set to 65535 bytes. The client’s initial congestion window should not exceed 10 times the MSS. Delayed ACKs are permitted and the maximum client delayed ACK should not exceed 10 times of the MSS before a forced ACK also, the maximum delayed ACK timer is allowed to be set to 200 ms. Up to three retries are allowed before a timeout event is declared. TCP PSH flag is set to high in all traffic. The source port range is in the range of 1024 - 65535. The clients initiate TCP connections via a three-way handshake (SYN, SYN/ACK, ACK) and close TCP connections via either a TCP three-way close (FIN, FIN/ACK, ACK) or a TCP four-way close (FIN, ACK, FIN, ACK).

4.3.1.2. QUIC Specification

QUIC stack emulation on the test equipment MUST conform to [RFC9000] and [RFC9001]. This section specifies the RECOMMENDED values for certain QUIC parameters to be configured on test equipment used for benchmarking purposes only. QUIC Stream type (defined in section 2.1 of [RFC9000]) is set to "Client-Initiated, Bidirectional". 0-RTT and early data are Disabled. QUIC Connection termination method is Immediate close (section 10.2 of [RFC9000]. Flow control is enabled. UDP payloads are set to datagram size of 1232 bytes for IPv6 and 1252 bytes for IPv4. In addition, transport parameters and default values defined in section 18.2 of [RFC9000] are RECOMMENDED to configure on test equipment. Also, this document references Appendixes B.1 and B.2 of [RFC9002] for congestion control related constants and variables. Any configured QUIC and UDP parameter(s) MUST be documented in the test report.

4.3.1.3. Client IP Address Space

The client IP space contains the following attributes.

* If multiple IP blocks are used, they MUST be consist of multiple unique, discontinuous static address blocks.

* A default gateway MAY be used.

* The DSCP (differentiated services code point) marking should be set to DF (Default Forwarding) ’000000’ on IPv4 Type of Service (ToS) field and IPv6 traffic class field.

* Extension header(s) MAY be used for IPv6 clients. If multiple extension headers are needed for traffic emulation, this document references [RFC8200] to choose the correct order of the extension headers within an IPv6 packet. Testing with extension header(s) may impact the performance of the DUT. The extension headers MUST be documented and reported.

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

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

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

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

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

Client IP addresses MUST be distributed between IPv4 and IPv6 based on deployment and use case scenario. The following options MAY be considered for a selection of ratios for both IP addresses and traffic load distribution.

(Option 1) 100 % IPv4, no IPv6

(Option 2) 80 % IPv4, 20% IPv6

(Option 3) 50 % IPv4, 50% IPv6

(Option 4) 20 % IPv4, 80% IPv6

(Option 5) no IPv4, 100% IPv6

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

4.3.1.4. Emulated Web Browser Attributes

The client (emulated web browser) contains attributes that will materially affect the traffic load. The objective is to emulate modern, typical browser attributes to improve the relevance of the result set for typical deployment scenarios.

The emulated browser MUST negotiate HTTP version 1.1 or higher. The emulated browser SHOULD advertise a User-Agent header. The emulated browser MUST enforce content length validation. HTTP header compression MAY be set to enable. If HTTP header compression is configurable in the test equipment, it MUST be documented if it was enabled or disabled. Depending on test scenarios and chosen HTTP version, the emulated browser MAY open multiple TCP or QUIC connections per Server endpoint IP at any time depending on how many sequential transactions need to be processed.

For HTTP/2 traffic emulation, the emulated browser opens multiple concurrent streams per connection (multiplexing). For HTTPS requests, the emulated browser MUST send "h2" protocol identifier using the TLS extension Application Layer Protocol Negotiation

(ALPN). The following default values (see [Undertow]) are the RECOMMENDED setting for certain HTTP/2 parameters to be configured on test equipment used for benchmarking purposes only:

* Maximum Frame size: 16384 bytes

* Initial Window size: 65535 bytes

* HPACK Header table size: 4096 bytes

* Server PUSH enable: false (Note: in [Undertow] the default setting is true. However, for testing purposes, this document recommends setting the value false for server push.)

This document refers to [RFC9113] for further details of HTTP/2. If any additional parameters are used to configure the test equipment, they MUST be documented.

For HTTP/3 traffic emulation, the emulated browsers initiate secure QUIC connections using TLS 1.3 ([RFC9001] describes how TLS is used to secure QUIC). This document refers to [RFC9114] for HTTP/3 specifications. The specification for transport protocol parameters is defined in Section 4.3.1.2. QPACK configuration settings such as MAX_TABLE_CAPACITY and QPACK_BLOCKED_STREAMS are set to zero (default) as defined in [RFC9204]. Any HTTP/3 parameters used for test equipment configuration MUST be documented.

For encrypted traffic, the following attributes are defined as the negotiated encryption parameters. The test clients MUST use TLS version 1.2 or higher. The TLS record size MAY be optimized for the HTTPS response object size up to a record size of 16 KBytes. If Server Name Indication (SNI) is required (especially if the server is identified by a domain name), the client endpoint MUST send TLS extension Server Name Indication (SNI) information when opening a security tunnel. Each client connection MUST perform a full TLS handshake and session reuse or resumption MUST be disabled. (Note: Real web browsers use session reuse or resumption. However, for testing purposes, this feature must not be used to measure the DUT/SUT performance in the worst-case scenario.)

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

1. ECDHE-ECDSA-AES128-GCM-SHA256 with Prime256v1 (Signature Hash Algorithm: ecdsa_secp256r1_sha256 and Supported group: secp256r1)

2. ECDHE-RSA-AES128-GCM-SHA256 with RSA 2048 (Signature Hash Algorithm: rsa_pkcs1_sha256 and Supported group: secp256r1)

3. ECDHE-ECDSA-AES256-GCM-SHA384 with Secp384r1 (Signature Hash Algorithm: ecdsa_secp384r1_sha384 and Supported group: secp384r1)

4. ECDHE-RSA-AES256-GCM-SHA384 with RSA 4096 (Signature Hash Algorithm: rsa_pkcs1_sha384 and Supported group: secp384r1)

Note: The above ciphers and keys were those commonly used for enterprise-grade encryption cipher suites for TLS 1.2 as of the time of publication (2022). 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.

IANA recommends the following cipher suites for use with TLS 1.3 defined in [RFC8446].

1. TLS_AES_128_GCM_SHA256

2. TLS_AES_256_GCM_SHA384

3. TLS_CHACHA20_POLY1305_SHA256

4. TLS_AES_128_CCM_SHA256

4.3.2. Backend Server Configuration

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

4.3.2.1. TCP Stack Attributes

The TCP stack on the server-side MUST be configured similar to the client-side configuration described in Section 4.3.1.1

4.3.2.2. QUIC Specification

The QUIC parameters on the server-side MUST be configured similar to the client-side configuration. Any configured QUIC Parameter(s) MUST be documented in the report.

4.3.2.3. Server Endpoint IP Addressing

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

* The server IP blocks MUST consist of unique, discontinuous static address blocks with one IP per server Fully Qualified Domain Name (FQDN) endpoint per test port.

* A default gateway is permitted. The DSCP (differentiated services code point) marking is set to DF (Default Forwarding) '000000' on IPv4 Type of Service (ToS) field and IPv6 traffic class field. Extension header(s) for the IPv6 server is permitted. If multiple extension headers are required, this document referenced [RFC8200] to choose the correct order of the extension headers within an IPv6 packet.

* The server IP address distribution between IPv4 and IPv6 MUST be identical to the client IP address distribution ratio.

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

4.3.2.4. HTTP / HTTPS Server Pool Endpoint Attributes

The HTTP 1.1 and HTTP/2 server pools listen on TCP ports 80 and 443 for HTTP and HTTPS. HTTP/3 server pool listens on UDP port 443 or any port. The server MUST emulate the same HTTP version (HTTP 1.1 or HTTP/2 or HTTP/3) and settings chosen by the client (emulated web browser). For the HTTPS server, TLS 1.2 or higher MUST be used with a maximum record size of 16 KByte. Ticket resumption or session ID reuse MUST NOT be used for TLS 1.2 and also session Ticket or session cache MUST NOT be used for TLS 1.3. The server MUST serve a certificate to the client. Cipher suite and key size on the server-side MUST be configured similar to the client-side configuration described in Section 4.3.1.4.

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 or QUIC stack as well as bound HTTP and HTTPS servers. When a client endpoint is needed, it will initialize and be given attributes such as a MAC and IP address. The behavior of the client is to sweep through the given server IP space, generating a recognizable service by the DUT. Sequential and pseudorandom sweep methods are acceptable. The method used MUST be stated in the final report. Thus, a balanced mesh between client endpoints and server endpoints will be generated in a client IP and port to server IP and port combination. Each client endpoint performs the same actions as other endpoints, with the difference being the source IP of the client endpoint and the target server IP pool. The client MUST use the server IP address or FQDN in the host

header.

4.3.3.1. Description of Intra-Client Behavior

Client endpoints are independent of other clients that are concurrently executing. When a client endpoint initiates traffic, this section describes how the client steps through different services. Once the test is initialized, the client endpoints randomly hold (perform no operation) for a few milliseconds for better randomization of the start of client traffic. Each client (HTTP 1.1 or HTTP/2) will either open a new TCP connection or connect to an HTTP persistent connection still open to that specific server. HTTP/3 clients will open UDP streams within QUIC connections. At any point that the traffic profile may require encryption, a TLS encryption tunnel will form presenting the URL or IP address request to the server. If using SNI, the server MUST then perform an SNI name check with the proposed FQDN compared to the domain embedded in the certificate. Only when correct, will the server process the HTTPS response object. The initial response object to the server is based on benchmarking tests described in Section 7. Multiple additional sub-URLs (response objects on the service page) MAY be requested simultaneously. This MAY be to the same server IP as the initial URL. Each sub-object will also use a canonical FQDN and URL path.

4.3.4. Traffic Load Profile

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

1. Init phase: Testbed devices including the client and server endpoints should negotiate layer 2-3 connectivity such as MAC learning and ARP/ND. Only after successful MAC learning or ARP/ND SHALL the test iteration move to the next phase. No measurements are made in this phase. The minimum recommended time for the Init phase is 5 seconds. During this phase, the emulated clients MUST NOT initiate any sessions with the DUT/SUT, in contrast, the emulated servers should be ready to accept requests from DUT/SUT or emulated clients.

2. Ramp up phase: The test equipment MUST start to generate the test traffic. It MUST use a set of the approximate number of unique client IP addresses to generate traffic. The traffic MUST ramp up from zero to desired target objective. The target objective is defined for each benchmarking test. The duration for the ramp up phase MUST be configured long enough that the test equipment does not overwhelm the DUT/SUTs stated performance metrics

defined in Section 6.3 namely, TCP or QUIC Connections Per Second, Inspected Throughput, Concurrent TCP or QUIC Connections, and Application Transactions Per Second. No measurements are made in this phase.

3. Sustain phase: Starts when all required clients are active and operating at their desired load condition. In the sustain phase, the test equipment MUST continue generating traffic to a constant target value for a constant number of active clients. The minimum RECOMMENDED time duration for sustain phase is 300 seconds. This is the phase where measurements occur. The test equipment MUST measure and record statistics continuously. The sampling interval for collecting the raw results and calculating the statistics MUST be less than 2 seconds.

4. Ramp down phase: The test traffic slows down from the target number to 0, and no measurements are made.

5. Collection phase: The last phase is administrative and will occur when the test equipment merges and collates the report data.

5. Testbed Considerations

This section describes steps for a reference test (pre-test) that control the test environment including test equipment, focusing on physical and virtualized environments and as well as test equipment. Below are the RECOMMENDED steps for the reference test.

1. Perform the reference test either by configuring the DUT/SUT in the most trivial setup (fast forwarding) or without the presence of the DUT/SUT.

2. Generate traffic from traffic generator. Choose a traffic profile used for the HTTP or HTTPS throughput performance test with the smallest object size.

3. Ensure that any ancillary switching or routing functions added in the test equipment do not limit the performance by introducing network metrics such as packet loss and latency. This is specifically important for virtualized components (e.g., vSwitches, vRouters).

4. Verify that the generated traffic (performance) of the test equipment matches and reasonably exceeds the expected maximum performance of the DUT/SUT.

5. Record the network performance metrics packet loss and latency introduced by the test environment (without DUT/SUT).

6. Assert that the testbed characteristics are stable during the entire test session. Several factors might influence stability specifically, for virtualized testbeds. For example, additional workloads in a virtualized system, load balancing, and movement of virtual machines during the test, or simple issues such as additional heat created by high workloads leading to an emergency CPU performance reduction.

The reference test MUST be performed before the benchmarking tests (described in section 7) start.

6. Reporting

This section describes how the benchmarking test report should be formatted and presented. It is RECOMMENDED to include two main sections in the report: the introduction and the detailed test results sections.

6.1. Introduction

The following attributes should be present in the introduction section of the test report.

1. The time and date of the execution of the tests

2. Summary of testbed software and hardware details

   a. DUT/SUT hardware/virtual configuration

      * This section should clearly identify the make and model of the DUT/SUT

      * The port interfaces, including speed and link information

      * If the DUT/SUT is a Virtual Network Function (VNF), host (server) hardware and software details, interface acceleration type such as DPDK and SR-IOV, used CPU cores, used RAM, resource sharing (e.g. Pinning details and NUMA Node) configuration details, hypervisor version, virtual switch version

      * details of any additional hardware relevant to the DUT/SUT such as controllers

   b. DUT/SUT software

      * Operating system name

* Version

* Specific configuration details (if any)

c. DUT/SUT enabled features

* Configured DUT/SUT features (see Table 2 and Table 3)

* Attributes of the above-mentioned features

* Any additional relevant information about the features

d. Test equipment hardware and software

* Test equipment vendor name

* Hardware details including model number, interface type

* Test equipment firmware and test application software version

* If the test equipment is a virtual solution, the host (server) hardware and software details, interface acceleration type such as DPDK and SR-IOV, used CPU cores, used RAM, resource sharing (e.g. Pinning details and NUMA Node) configuration details, hypervisor version, virtual switch version

e. Key test parameters

* Used cipher suites and keys

* IPv4 and IPv6 traffic distribution

* Number of configured ACL

* TCP, UDP stack parameter if tested

* QUIC, HTTP/2, and HTTP/3 parameters if tested

f. Details of application traffic mix used in the benchmarking test "Throughput Performance with Application Traffic Mix" (Section 7.1)

* Name of applications and layer 7 protocols

* Percentage of emulated traffic for each application and layer 7 protocols

* Percentage of encrypted traffic and used cipher suites and keys (The RECOMMENDED ciphers and keys are defined in Section 4.3.1.4)

* Used object sizes for each application and layer 7 protocols

3. Results Summary / Executive Summary

   a. Results should be presented with an introduction section documenting the summary of results in a prominent, easy to read block.

6.2. Detailed Test Results

In the result section of the test report, the following attributes should be present for each benchmarking test.

a. KPIs MUST be documented separately for each benchmarking test. The format of the KPI metrics MUST be presented as described in Section 6.3.

b. The next level of detail should be graphs showing each of these metrics over the duration (sustain phase) of the test. This allows the user to see the measured performance stability changes over time.

6.3. Benchmarks and Key Performance Indicators

This section lists key performance indicators (KPIs) for overall benchmarking tests. All KPIs MUST be measured during the sustain phase of the traffic load profile described in Section 4.3.4. Also, the KPIs MUST be measured from the result output of test equipment.

* Concurrent TCP Connections

  The aggregate number of simultaneous connections between hosts across the DUT/SUT, or between hosts and the DUT/SUT (defined in [RFC2647]).

* Concurrent QUIC Connections

  The aggregate number of simultaneous connections between hosts across the DUT/SUT.

* TCP Connections Per Second

The average number of successfully established TCP connections per second between hosts across the DUT/SUT, or between hosts and the DUT/SUT. As described in Section 4.3.1.1, the TCP connections are initiated by clients via a TCP three-way handshake (SYN, SYN/ACK, ACK). Then the TCP session data is sent and then the TCP sessions are closed via either a TCP three-way close (FIN, FIN/ACK, ACK) or a TCP four-way close (FIN, ACK, FIN, ACK). The TCP sessions MUST NOT be closed by RST.

* QUIC Connections Per Second

  The average number of successfully established QUIC connections per second between hosts across the DUT/SUT. As described in Section 4.3.1.2, the QUIC connections are initiated by clients. Then the data is sent and then the QUIC sessions are closed by "immediate close" method.

  Since QUIC specification defined in Section 4.3.1.2 recommends disabling 0-RTT and early data, this KPI focused on 1-RTT handshake. If required, 0-RTT can be also measured in separate test runs while enabling 0-RTT and early data in the test equipment.

* Application Transactions Per Second

  The average number of successfully completed transactions per second. For a particular transaction to be considered successful, all data MUST have been transferred in its entirety. In case of HTTP(S) transactions, it MUST have a valid status code (200 OK).

* TLS Handshake Rate

  The average number of successfully established TLS connections per second between hosts across the DUT/SUT, or between hosts and the DUT/SUT.

  For TLS1.3 the handshake rate can be measured with 0-RTT or 1-RTT handshake. The transport protocol can be either TCP or QUIC.

* Inspected Throughput

  The number of bits per second of examined and allowed traffic a network security device is able to transmit to the correct destination interface(s) in response to a specified offered load. The throughput benchmarking tests defined in Section 7 SHOULD measure the average Layer 2 throughput value when the DUT/SUT is "inspecting" traffic. It is also acceptable to measure other OSI Layer throughput. However, the measured layer (e.g. Layer 3

throughput) MUST be noted in the report and the user MUST be aware of the implication while comparing the throughput performance of multiple DUT/SUTs measured in different OSI Layers. This document recommends presenting the inspected throughput value in Gbit/s rounded to two places of precision with a more specific Kbit/s in parenthesis.

* Time to First Byte (TTFB)

  TTFB is the elapsed time between the start of sending the TCP SYN packet or QUIC initial Client Hello from the client and the client receiving the first packet of application data from the server via DUT/SUT. The benchmarking tests HTTP Transaction Latency (Section 7.4) and HTTPS Transaction Latency (Section 7.8) measure the minimum, average and maximum TTFB. The value should be expressed in milliseconds.

* URL Response time / Time to Last Byte (TTLB)

  URL Response time / TTLB is the elapsed time between the start of sending the TCP SYN packet or QUIC initial Client Hello from the client and the client receiving the last packet of application data from the server via DUT/SUT. The benchmarking tests HTTP Transaction Latency (Section 7.4) and HTTPS Transaction Latency (Section 7.8) measure the minimum, average and maximum TTLB. The value should be expressed in milliseconds.

7. Benchmarking Tests

This section mainly focuses on the benchmarking tests with HTTP/1.1 or HTTP/2 traffic which uses TCP as the transport protocol. In particular, this section does not define specific benchmarking tests for QUIC or HTTP/3 related KPIs. However, the test methodology defined in the benchmarking tests TCP/QUIC Connections Per Second with HTTPS Traffic (Section 7.6), HTTPS Transaction Latency (Section 7.8), HTTPS Throughput (Section 7.7), and Concurrent TCP/QUIC Connection Capacity with HTTPS Traffic (Section 7.9) can be used to test QUIC or HTTP/3 related KPIs. The throughput performance test with the application traffic mix defined in Section 7.1 can be performed with any other application traffic including HTTP/3.

7.1. Throughput Performance with Application Traffic Mix

7.1.1. Objective

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

Based on the test customer's specific use case, testers can choose the relevant application traffic mix for this test. The details about the traffic mix MUST be documented in the report. At least the following traffic mix details MUST be documented and reported together with the test results:

Name of applications and layer 7 protocols

Percentage of emulated traffic for each application and layer 7 protocol

Percentage of encrypted traffic and used cipher suites and keys (The RECOMMENDED ciphers and keys are defined in Section 4.3.1.4.)

Used object sizes for each application and layer 7 protocols

7.1.2. Test Setup

Testbed setup MUST be configured as defined in Section 4. Any benchmarking test specific testbed configuration changes MUST be documented.

7.1.3. Test Parameters

In this section, the benchmarking test specific parameters are defined.

7.1.3.1. DUT/SUT Configuration Parameters

DUT/SUT parameters MUST conform to the requirements defined in Section 4.2. Any configuration changes for this specific benchmarking test MUST be documented. In case the DUT/SUT is configured without TLS inspection, the test report MUST explain the implications of this to the relevant application traffic mix encrypted traffic.

7.1.3.2. Test Equipment Configuration Parameters

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

Client IP address ranges defined in Section 4.3.1.3

Server IP address ranges defined in Section 4.3.2.3

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.3

Target inspected throughput: Aggregated line rate of the interface(s) used in the DUT/SUT or the value defined based on the requirement for a specific deployment scenario

Initial throughput: 10% of the "Target inspected throughput" Note: Initial throughput is not a KPI to report. This value is configured on the traffic generator and used to perform Step 1: "Test Initialization and Qualification" described under Section 7.1.4.

One of the ciphers and keys defined in Section 4.3.1.4 are RECOMMENDED to use for this benchmarking test.

7.1.3.3. Traffic Profile

Traffic profile: This test MUST be run with a relevant application traffic mix profile.

7.1.3.4. Test Results Validation Criteria

The following criteria are the test results validation criteria. The test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

a. Number of failed application transactions 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. If HTTP/3 is used, the number of failed QUIC connections due to unexpected HTTP/3 error codes MUST be less than 0.001% (1 out of 100,000 connections) of total initiated QUIC connections.

7.1.3.5. Measurement

The following KPI metrics MUST be reported for this benchmarking test:

Mandatory KPIs (benchmarks): Inspected Throughput and Application Transactions Per Second

Note: TTLB MUST be reported along with the object size used in the traffic profile.

Optional TCP stack related KPIs: TCP Connections Per Second, TLS Handshake Rate, TTFB (minimum, average, and maximum), TTLB (minimum, average, and maximum)

Optional QUIC stack related KPIs: QUIC connection per second and concurrent QUIC connections

7.1.4. Test Procedures and Expected Results

The test procedures are designed to measure the inspected throughput performance of the DUT/SUT at the sustaining period of the traffic load profile. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (initial throughput) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines whether the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IP types: IPv4 only, IPv6 only, and IPv4 and IPv6 mixed traffic distribution.

7.1.4.1. Step 1: Test Initialization and Qualification

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

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

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

7.1.4.2. Step 2: Test Run with Target Objective

Configure test equipment to generate traffic at the "Target inspected throughput" rate defined in Section 7.1.3.2. The test equipment MUST follow the traffic load profile definition as described in Section 4.3.4. The test equipment MUST start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective ("Target inspected throughput") in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.1.4.3. Step 3: Test Iteration

Determine the achievable average inspected throughput within the test results validation criteria. The final test iteration MUST be performed for the test duration defined in Section 4.3.4.

7.2. TCP/HTTP Connections Per Second

7.2.1. Objective

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

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

7.2.2. Test Setup

Testbed setup MUST be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.2.3. Test Parameters

In this section, benchmarking test specific parameters are defined.

7.2.3.1. DUT/SUT Configuration Parameters

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

7.2.3.2. Test Equipment Configuration Parameters

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

Client IP address ranges defined in Section 4.3.1.3

Server IP address ranges defined in Section 4.3.2.3

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.3

Target connections per second: Initial value from product datasheet or the value defined based on the requirement for a specific deployment scenario

Initial connections per second: 10% of "Target connections per second" (Note: Initial connections per second is not a KPI to report. This value is configured on the traffic generator and used to perform Step1: "Test Initialization and Qualification" described under Section 7.2.4.)

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

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

7.2.3.3. Test Results Validation Criteria

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

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

b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.

c. During the sustain phase, traffic MUST be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

d. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections MUST be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.

7.2.3.4. Measurement

TCP Connections Per Second MUST be reported for each test iteration (for each object size).

7.2.4. Test Procedures and Expected Results

The test procedure is designed to measure the TCP connections per second rate of the DUT/SUT at the sustaining period of the traffic load profile. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial connections per second) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines whether the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IP types: IPv4 only, IPv6 only, and IPv4 and IPv6 mixed traffic distribution.

7.2.4.1. Step 1: Test Initialization and Qualification

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

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

The DUT/SUT MUST reach the "Initial connections per second" before the sustain phase. The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Section 7.2.3.3.

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

7.2.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish the target objective ("Target connections per second") defined in Section 7.2.3.2. The test equipment MUST follow the traffic load profile definition as described in Section 4.3.4.

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

The test equipment MUST start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective ("Target connections per second") in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.2.4.3. Step 3: Test Iteration

Determine the achievable TCP connections per second within the test results validation criteria.

7.3. HTTP Throughput

7.3.1. Objective

Determine the sustainable inspected throughput of the DUT/SUT for HTTP transactions varying the HTTP response object size.

7.3.2. Test Setup

Testbed setup MUST be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.3.3. Test Parameters

In this section, benchmarking test specific parameters are defined.

7.3.3.1. DUT/SUT Configuration Parameters

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

7.3.3.2. Test Equipment Configuration Parameters

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

Client IP address ranges defined in Section 4.3.1.3

Server IP address ranges defined in Section 4.3.2.3

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.3

Target inspected throughput: Aggregated line rate of the interface(s) used in the DUT/SUT or the value defined based on the requirement for a specific deployment scenario

Initial throughput: 10% of "Target inspected throughput" Note: Initial throughput is not a KPI to report. This value is configured on the traffic generator and used to perform Step 1: "Test Initialization and Qualification" described under Section 7.3.4.

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

RECOMMENDED HTTP response object size: 1, 16, 64, 256 KByte, and mixed objects defined in Table 4.

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

Table 4: Mixed Objects

7.3.3.3. Test Results Validation Criteria

The following criteria are the test results validation criteria. The test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

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

b. Traffic MUST be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

c. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections MUST be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.

7.3.3.4. Measurement

Inspected Throughput and HTTP Transactions per Second MUST be reported for each object size.

7.3.4. Test Procedures and Expected Results

The test procedure is designed to measure HTTP throughput of the DUT/SUT. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial throughput) and meets the test results validation criteria when it was very minimal utilized. Step 2 determines whether the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution and HTTP response object sizes.

7.3.4.1. Step 1: Test Initialization and Qualification

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

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

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

The measured KPIs during the sustain phase MUST meet the test results validation criteria "a" defined in Section 7.3.3.3. The test results validation criteria "b" and "c" are OPTIONAL for step 1.

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

7.3.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish the target objective ("Target inspected throughput") defined in Section 7.3.3.2. The test equipment MUST start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.3.4.3. Step 3: Test Iteration

Determine the achievable inspected throughput within the test results validation criteria and measure the KPI metric Transactions per Second. The final test iteration MUST be performed for the test duration defined in Section 4.3.4.

7.4. HTTP Transaction Latency

7.4.1. Objective

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

Test iterations MUST be performed with different HTTP response object sizes in two different scenarios. One with a single transaction and the other with multiple transactions within a single TCP connection. For consistency, both the single and multiple transaction tests MUST be configured with the same HTTP version

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

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

7.4.2. Test Setup

Testbed setup MUST be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.4.3. Test Parameters

In this section, benchmarking test specific parameters are defined.

7.4.3.1. DUT/SUT Configuration Parameters

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

7.4.3.2. Test Equipment Configuration Parameters

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

Client IP address ranges defined in Section 4.3.1.3

Server IP address ranges defined in Section 4.3.2.3

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.3

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

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

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

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

Note: The Initial objectives are not a KPI to report. These values are configured on the traffic generator and used to perform Step1: "Test Initialization and Qualification" described under Section 7.4.4.

HTTP transaction per TCP connection: Test scenario 1 with a single transaction and test scenario 2 with 10 transactions.

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

7.4.3.3. Test Results Validation Criteria

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

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

b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.

c. During the sustain phase, traffic MUST be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

d. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections MUST be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.

e. After ramp up the DUT MUST achieve the "Target objective" defined in Section 7.4.3.2 and remain in that state for the entire test duration (sustain phase).

7.4.3.4. Measurement

TTFB (minimum, average, and maximum) and TTLB (minimum, average, and maximum) MUST be reported for each object size.

7.4.4. Test Procedures and Expected Results

The test procedure is designed to measure TTFB or TTLB when the DUT/SUT is operating close to 50% of its maximum achievable connections per second or inspected throughput. The test procedure consists of two major steps: Step 1 ensures the DUT/SUT is able to reach the initial performance values and meets the test results validation criteria when it was very minimally utilized. Step 2 measures the latency values within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IP types (IPv4 only, IPv6 only, and IPv4 and IPv6 mixed traffic distribution), HTTP response object sizes, and single and multiple transactions per connection scenarios.

7.4.4.1. Step 1: Test Initialization and Qualification

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

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

The DUT/SUT MUST reach the "Initial objective" before the sustain phase. The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Section 7.4.3.3.

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

7.4.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish the "Target objective" defined in Section 7.4.3.2. The test equipment MUST follow the traffic load profile definition as described in Section 4.3.4.

The test equipment MUST start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT MUST reach the desired value of the target objective in the sustain phase.

Measure the minimum, average, and maximum values of TTFB and TTLB.

7.5. Concurrent TCP/HTTP Connection Capacity

7.5.1. Objective

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

7.5.2. Test Setup

Testbed setup MUST be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.5.3. Test Parameters

In this section, benchmarking test specific parameters are defined.

7.5.3.1. DUT/SUT Configuration Parameters

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

7.5.3.2. Test Equipment Configuration Parameters

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

Client IP address ranges defined in Section 4.3.1.3

Server IP address ranges defined in Section 4.3.2.3

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.3

Target concurrent connection: Initial value from product datasheet or the value defined based on the requirement for a specific deployment scenario.

Initial concurrent connection: 10% of "Target concurrent connection" Note: Initial concurrent connection is not a KPI to report. This value is configured on the traffic generator and used to perform Step1: "Test Initialization and Qualification" described under Section 7.5.4.

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

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

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

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

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

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

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

7.5.3.3. Test Results Validation Criteria

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

a. Number of failed application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 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 MUST be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

7.5.3.4. Measurement

Average Concurrent TCP Connections MUST be reported for this benchmarking test.

7.5.4. Test Procedures and Expected Results

The test procedure is designed to measure the concurrent TCP connection capacity of the DUT/SUT at the sustaining period of the traffic load profile. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial concurrent connection) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines whether the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distributions.

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 MUST be defined as described in Section 4.3.4.

During the sustain phase, the DUT/SUT MUST reach the "Initial concurrent TCP connections". The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Section 7.5.3.3.

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

7.5.4.2. Step 2: Test Run with Target Objective

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

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

The test equipment MUST start to measure and record KPIs defined in Section 7.5.3.4. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.5.4.3. Step 3: Test Iteration

Determine the achievable concurrent TCP connections capacity within the test results validation criteria.

7.6. TCP/QUIC Connections per Second with HTTPS Traffic

7.6.1. Objective

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

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

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

7.6.2. Test Setup

Testbed setup MUST be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.6.3. Test Parameters

In this section, benchmarking test specific parameters are defined.

7.6.3.1. DUT/SUT Configuration Parameters

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

7.6.3.2. Test Equipment Configuration Parameters

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

Client IP address ranges defined in Section 4.3.1.3

Server IP address ranges defined in Section 4.3.2.3

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.3

Target connections per second: Initial value from product datasheet or the value defined based on the requirement for a specific deployment scenario.

Initial connections per second: 10% of "Target connections per second" (Note: Initial connections per second is not a KPI to report. This value is configured on the traffic generator and used to perform Step1: "Test Initialization and Qualification" described under Section 7.6.4.)

RECOMMENDED ciphers and keys defined in Section 4.3.1.4

The client MUST negotiate HTTPS and close the connection without error immediately after the completion of one transaction. In each test iteration, the client MUST send a GET request requesting a fixed HTTPS response object size. The RECOMMENDED object sizes are 1, 2, 4, 16, and 64 KByte.

7.6.3.3. Test Results Validation Criteria

The following criteria are the test results validation criteria. The test results validation criteria MUST be monitored during the whole test duration.

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

b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections. If HTTP/3 is used, the number of terminated QUIC connections due to unexpected errors MUST be less than 0.001% (1 out of 100,000 connections) of total initiated QUIC connections.

c. During the sustain phase, traffic MUST be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

d. Concurrent TCP connections generation rate MUST be constant during steady state and any deviation of concurrent TCP connections MUST be less than 10%. If HTTP/3 is used, the concurrent QUIC connections generation rate MUST be constant during steady state and any deviation of concurrent QUIC connections MUST be less than 10%. This confirms the DUT opens and closes connections at approximately the same rate.

7.6.3.4. Measurement

If HTTP 1.1 or HTTP/2 is used, TCP connections per second MUST be reported for each test iteration (for each object size).

If HTTP/3 is used, QUIC connections per second MUST be measured and reported for each test iteration (for each object size).

The KPI metric TLS Handshake Rate can be measured in the test using 1 KByte object size.

7.6.4. Test Procedures and Expected Results

The test procedure is designed to measure the TCP or QUIC connections per second rate of the DUT/SUT at the sustaining period of the traffic load profile. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial connections per second) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines whether the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distributions.

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 the 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 MUST be defined as described in Section 4.3.4.

The DUT/SUT MUST reach the "Initial connections per second" before the sustain phase. The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Section 7.6.3.3.

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

7.6.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish "Target connections per second" as defined in Section 7.6.3.2. The test equipment MUST follow the traffic load profile definition as described in Section 4.3.4.

During the ramp up and sustain phase, other KPIs such as inspected throughput, concurrent TCP/QUIC connections, and application transactions per second MUST NOT reach the maximum value the DUT/SUT

can support. The test results for the specific test iteration MUST NOT be reported as valid results, if the above mentioned KPI (especially inspected throughput) reaches the maximum value. (Example: If the test iteration with 64 KByte of HTTPS response object size reached the maximum inspected throughput limitation of the DUT, the test iteration MAY be interrupted, and the result for 64 KByte should not be reported).

The test equipment MUST start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective ("Target connections per second") in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.6.4.3. Step 3: Test Iteration

Determine the achievable connections per second within the test results validation criteria.

7.7. HTTPS Throughput

7.7.1. Objective

Determine the sustainable inspected throughput of the DUT/SUT for HTTPS transactions varying the HTTPS response object size.

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

7.7.2. Test Setup

Testbed setup MUST be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.7.3. Test Parameters

In this section, benchmarking test specific parameters are defined.

7.7.3.1. DUT/SUT Configuration Parameters

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

7.7.3.2. Test Equipment Configuration Parameters

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

Client IP address ranges defined in Section 4.3.1.3

Server IP address ranges defined in Section 4.3.2.3

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.3

Target inspected throughput: Aggregated line rate of the interface(s) used in the DUT/SUT or the value defined based on the requirement for a specific deployment scenario.

Initial throughput: 10% of "Target inspected throughput" Note: Initial throughput is not a KPI to report. This value is configured on the traffic generator and used to perform Step1: "Test Initialization and Qualification" described under Section 7.7.4.

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

RECOMMENDED ciphers and keys defined in Section 4.3.1.4

RECOMMENDED HTTPS response object size: 1, 16, 64, 256 KByte, and mixed objects defined in Table 4 under Section 7.3.3.2.

7.7.3.3. Test Results Validation Criteria

The following criteria are the test results validation criteria. The test results validation criteria MUST be monitored during the whole sustain phase of the traffic load profile.

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

b. Traffic MUST be generated at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

c. Concurrent generated TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections MUST be less than 10%. If HTTP/3 is used, the concurrent generated QUIC connections MUST be constant during steady state and any deviation of concurrent QUIC connections MUST be less than 10%. This confirms the DUT opens and closes connections at approximately the same rate.

7.7.3.4. Measurement

Inspected Throughput and HTTPS Transactions per Second MUST be reported for each object size.

7.7.4. Test Procedures and Expected Results

The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial throughput) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines whether the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution and HTTPS response object sizes.

7.7.4.1. Step 1: Test Initialization and Qualification

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

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

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

The measured KPIs during the sustain phase MUST meet the test results validation criteria "a" defined in Section 7.7.3.3. The test results validation criteria "b", and "c" are OPTIONAL for step 1.

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

7.7.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish the target objective ("Target inspected throughput") defined in Section 7.7.3.2. The test equipment MUST start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.7.4.3. Step 3: Test Iteration

Determine the achievable average inspected throughput within the test results validation criteria. The final test iteration MUST be performed for the test duration defined in Section 4.3.4.

7.8. HTTPS Transaction Latency

7.8.1. Objective

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

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

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

7.8.2. Test Setup

Testbed setup MUST be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.8.3. Test Parameters

In this section, benchmarking test specific parameters are defined.

7.8.3.1. DUT/SUT Configuration Parameters

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

7.8.3.2. Test Equipment Configuration Parameters

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

Client IP address ranges defined in Section 4.3.1.3

Server IP address ranges defined in Section 4.3.2.3

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.3

RECOMMENDED cipher suites and key sizes defined in Section 4.3.1.4

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

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

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

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

Note: The Initial objectives are not a KPI to report. These values are configured on the traffic generator and used to perform Step1: "Test Initialization and Qualification" described under Section 7.8.4.

HTTPS transaction per TCP or QUIC connection: Test scenario 1 with a single transaction and scenario 2 with 10 transactions

HTTPS with GET request requesting a single object. The RECOMMENDED object sizes are 1, 16, and 64 KByte. For each test iteration, the client MUST request a single HTTPS response object size.

7.8.3.3. Test Results Validation Criteria

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

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

b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections. If HTTP/3 is used, the number of terminated QUIC connections due to unexpected errors MUST be less than 0.001% (1 out of 100,000 connections) of total initiated QUIC connections.

c. During the sustain phase, traffic MUST be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

d. Concurrent TCP or QUIC connections MUST be constant during steady state and any deviation of concurrent TCP connections MUST be less than 10%. If HTTP/3 is used, the concurrent generated QUIC connections MUST be constant during steady state and any deviation of concurrent QUIC connections MUST be less than 10%. This confirms the DUT opens and closes connections at approximately the same rate.

e. After ramp up the DUT/SUT MUST achieve the "Target objective" defined in the parameter Section 7.8.3.2 and remain in that state for the entire test duration (sustain phase).

7.8.3.4. Measurement

TTFB (minimum, average, and maximum) and TTLB (minimum, average, and maximum) MUST be reported for each object size.

7.8.4. Test Procedures and Expected Results

The test procedure is designed to measure TTFB or TTLB when the DUT/SUT is operating close to 50% of its maximum achievable connections per second or inspected throughput. The test procedure consists of two major steps: Step 1 ensures the DUT/SUT is able to reach the initial performance values and meets the test results validation criteria when it was very minimally utilized. Step 2 measures the latency values within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IP types (IPv4 only, IPv6 only, and IPv4 and IPv6 mixed traffic distribution), HTTPS response object sizes, and single, and multiple transactions per connection scenarios.

7.8.4.1. Step 1: Test Initialization and Qualification

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

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

The DUT/SUT MUST reach the "Initial objective" before the sustain phase. The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Section 7.8.3.3.

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

7.8.4.2. Step 2: Test Run with Target Objective

Configure test equipment to establish the "Target objective" defined in Section 7.8.3.2. The test equipment MUST follow the traffic load profile definition as described in Section 4.3.4.

The test equipment MUST start to measure and record all specified KPIs. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT MUST reach the desired value of the target objective in the sustain phase.

Measure the minimum, average, and maximum values of TTFB and TTLB.

7.9. Concurrent TCP/QUIC Connection Capacity with HTTPS Traffic

7.9.1. Objective

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

7.9.2. Test Setup

Testbed setup MUST be configured as defined in Section 4. Any specific testbed configuration changes (number of interfaces and interface type, etc.) MUST be documented.

7.9.3. Test Parameters

In this section, benchmarking test specific parameters are defined.

7.9.3.1. DUT/SUT Configuration Parameters

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

7.9.3.2. Test Equipment Configuration Parameters

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

Client IP address ranges defined in Section 4.3.1.3

Server IP address ranges defined in Section 4.3.2.3

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.3

RECOMMENDED cipher suites and key sizes defined in Section 4.3.1.4

Target concurrent connections: Initial value from product datasheet or the value defined based on the requirement for a specific deployment scenario.

Initial concurrent connections: 10% of "Target concurrent connections" Note: Initial concurrent connection is not a KPI to report. This value is configured on the traffic generator and used to perform Step1: "Test Initialization and Qualification" described under Section 7.9.4.

Connections per second during ramp up phase: 50% of maximum connections per second measured in benchmarking test TCP/QUIC Connections per second with HTTPS Traffic (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 transactions with persistence and each client can open multiple concurrent connections per server endpoint IP.

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

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

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

7.9.3.3. Test Results Validation Criteria

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

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

b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections. If HTTP/3 is used, the number of terminated QUIC connections due to unexpected errors MUST be less than 0.001% (1 out of 100,000 connections) of total initiated QUIC connections

c. During the sustain phase, traffic MUST be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

7.9.3.4. Measurement

Average Concurrent TCP or QUIC Connections MUST be reported for this benchmarking test.

7.9.4. Test Procedures and Expected Results

The test procedure is designed to measure the concurrent TCP connection capacity of the DUT/SUT at the sustaining period of the traffic load profile. The test procedure consists of three major steps: Step 1 ensures the DUT/SUT is able to reach the performance value (Initial concurrent connection) and meets the test results validation criteria when it was very minimally utilized. Step 2 determines whether the DUT/SUT is able to reach the target performance value within the test results validation criteria. Step 3 determines the maximum achievable performance value within the test results validation criteria.

This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distributions.

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 MUST be defined as described in Section 4.3.4.

During the sustain phase, the DUT/SUT MUST reach the "Initial concurrent connections". The measured KPIs during the sustain phase MUST meet the test results validation criteria "a", and "b" defined in Section 7.9.3.3.

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

7.9.4.2. Step 2: Test Run with Target Objective

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

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

The test equipment MUST start to measure and record KPIs defined in Section 7.9.3.4. Continue the test until all traffic profile phases are completed.

Within the test results validation criteria, the DUT/SUT is expected to reach the desired value of the target objective in the sustain phase. Follow step 3, if the measured value does not meet the target value or does not fulfill the test results validation criteria.

7.9.4.3. Step 3: Test Iteration

Determine the achievable concurrent TCP/QUIC connections within the test results validation criteria.

8. IANA Considerations

This document makes no specific request of IANA.

The IANA has assigned IPv4 and IPv6 address blocks in [RFC6890] that have been registered for special purposes. The IPv6 address block 2001:2::/48 has been allocated for the purpose of IPv6 Benchmarking [RFC5180] and the IPv4 address block 198.18.0.0/15 has been allocated for the purpose of IPv4 Benchmarking [RFC2544]. This assignment was made to minimize the chance of conflict in case a testing device were to be accidentally connected to the part of the Internet.

9. Security Considerations

The primary goal of this document is to provide benchmarking terminology and methodology for next-generation network security devices for use in a laboratory isolated test environment. However, readers should be aware that there is some overlap between performance and security issues. Specifically, the optimal configuration for network security device performance may not be the most secure, and vice-versa. Testing security platforms with working exploits and malware carries risks. Ensure proper access controls are implemented to prevent unintended exposure to vulnerable networks or systems. The cipher suites recommended in this document are for test purposes only. The cipher suite recommendation for a real deployment is outside the scope of this document.

Security assessment of an NGFW/NGIPS product could also include an analysis whether any type of uncommon traffic characteristics would have a significant impact on performance. Such performance impacts would allow an attacker to use such specifically crafted traffic as a DoS attack to reduce the remaining performance available to other traffic through the NGFW/NGIPS. Such uncommon traffic characteristics might include for example IP fragmented traffic, specific type of application traffic, or uncommonly high HTTP transaction rate traffic.

10. Contributors

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

Alex Samonte, Amritam Putatunda, Aria Eslambolchizadeh, Chao Guo, Chris Brown, Cory Ford, David DeSanto, Jurrie Van Den Breekel, Michelle Rhines, Mike Jack, Ryan Liles, Samaresh Nair, Stephen Goudreault, Tim Carlin, and Tim Otto.

11. Acknowledgements

The authors wish to acknowledge the members of NetSecOPEN for their participation in the creation of this document. Additionally, the following members need to be acknowledged:

Anand Vijayan, Chris Marshall, Jay Lindenauer, Michael Shannon, Mike Deichman, Ryan Riese, and Toulnay 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

[fastly] Oku, K. and J. Iyengar, "Can QUIC match TCP's computational efficiency?", 30 April 2020, <https://www.fastly.com/blog/measuring-quic-vs-tcp-computational-efficiency>.

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

[RFC2647] Newman, D., "Benchmarking Terminology for Firewall Performance", RFC 2647, DOI 10.17487/RFC2647, August 1999, <https://www.rfc-editor.org/info/rfc2647>.

[RFC3511] Hickman, B., Newman, D., Tadjudin, S., and T. Martin, "Benchmarking Methodology for Firewall Performance", RFC 3511, DOI 10.17487/RFC3511, April 2003, <https://www.rfc-editor.org/info/rfc3511>.

[RFC5180] Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, DOI 10.17487/RFC5180, May 2008, <https://www.rfc-editor.org/info/rfc5180>.

[RFC6815] Bradner, S., Dubray, K., McQuaid, J., and A. Morton, "Applicability Statement for RFC 2544: Use on Production Networks Considered Harmful", RFC 6815, DOI 10.17487/RFC6815, November 2012, <https://www.rfc-editor.org/info/rfc6815>.

[RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <https://www.rfc-editor.org/info/rfc6890>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/info/rfc9001>.

[RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <https://www.rfc-editor.org/info/rfc9002>.

[RFC9113] Thomson, M., Ed. and C. Benfield, Ed., "HTTP/2", RFC 9113, DOI 10.17487/RFC9113, June 2022, <https://www.rfc-editor.org/info/rfc9113>.

[RFC9114] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/info/rfc9114>.

[RFC9204] Krasic, C., Bishop, M., and A. Frindell, Ed., "QPACK: Field Compression for HTTP/3", RFC 9204, DOI 10.17487/RFC9204, June 2022, <https://www.rfc-editor.org/info/rfc9204>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

[Undertow] "An in depth overview of HTTP/2", <https://undertow.io/blog/2015/04/27/An-in-depth-overview-of-HTTP2.html>.

[Wiki-NGFW] "", <https://en.wikipedia.org/wiki/Next-generation_firewall>.

Appendix A. Test Methodology - Security Effectiveness Evaluation

A.1. Test Objective

This test methodology verifies the DUT/SUT is able to detect, prevent, and report the vulnerabilities.

In this test, background test traffic will be generated to utilize the DUT/SUT. In parallel, a number of malicious traffic will be sent to the DUT/SUT as encrypted and as well as clear text payload formats using a traffic generator. Section 4.2.1 defines the selection of the malicious traffic from the Common Vulnerabilities and Exposures (CVE) list for testing.

The following KPIs are measured in this test:

* Number of blocked CVEs

* Number of bypassed (nonblocked) CVEs

* Background traffic performance (verify if the background traffic is impacted while sending CVE toward DUT/SUT)

* Accuracy of DUT/SUT statistics in terms of vulnerabilities reporting

A.2. Testbed Setup

The same testbed MUST be used for security effectiveness tests and as well as for benchmarking test cases defined in Section 7.

A.3. Test Parameters

In this section, the benchmarking test specific parameters are 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 the 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 MUST be reset

A.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. The same client and server IP ranges MUST be configured as used in the benchmarking test cases. In addition, the following parameters MUST be documented for this benchmarking test:

* Background Traffic: 45% of maximum HTTP throughput and 45% of Maximum HTTPS throughput supported by the DUT/SUT (measured with object size 64 KByte in the benchmarking tests "HTTP(S) Throughput" defined in Section 7.3 and Section 7.7).

* RECOMMENDED CVE traffic transmission Rate: 10 CVEs per second

* It is RECOMMENDED to generate each CVE multiple times (sequentially) at 10 CVEs per second

* Ciphers and keys for the encrypted CVE traffic MUST use the same cipher configured for HTTPS traffic related benchmarking tests (Section 7.6 - Section 7.9)

A.4. Test Results Validation Criteria

The following criteria are the test results validation criteria. The test results validation criteria MUST be monitored during the whole test duration.

a. Number of failed application transactions in the background traffic MUST be less than 0.01% of attempted transactions.

b. Number of terminated TCP or QUIC connections of the background traffic (due to unexpected errors) MUST be less than 0.01% of total initiated TCP connections in the background traffic.

c. During the sustain phase, traffic MUST be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

d. False positive MUST NOT occur in the background traffic.

A.5. Measurement

The following KPI metrics MUST be reported for this test scenario:

Mandatory KPIs:

* Blocked CVEs: They MUST be represented in the following ways:

  - Number of blocked CVEs out of total CVEs

  - Percentage of blocked CVEs

* Unblocked CVEs: They MUST be represented in the following ways:

  - Number of unblocked CVEs out of total CVEs

  - Percentage of unblocked CVEs

* Background traffic behavior: It MUST be represented in one of the followings ways:

  - No impact: Considered as "no impact’" if any deviation of traffic forwarding rate is less than or equal to 5 % (constant rate)

  - Minor impact: Considered as "minor impact" if any deviation of traffic forwarding rate is greater than 5% and less than or equal to10% (i.e. small spikes)

  - Heavily impacted: Considered as "Heavily impacted" if any deviation of traffic forwarding rate is greater than 10% (i.e. large spikes) or reduced the background HTTP(S) throughput greater than 10%

* DUT/SUT reporting accuracy: DUT/SUT MUST report all detected vulnerabilities.

Optional KPIs:

* List of unblocked CVEs

A.6. Test Procedures and Expected Results

The test procedure is designed to measure the security effectiveness of the DUT/SUT at the sustaining period of the traffic load profile. The test procedure consists of two major steps. This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distributions.

A.6.1. Step 1: Background Traffic

Generate background traffic at the transmission rate defined in Appendix A.3.2.

The DUT/SUT MUST reach the target objective (HTTP(S) throughput) in sustain phase. The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Appendix A.4.

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

A.6.2. Step 2: CVE Emulation

While generating background traffic (in sustain phase), send the CVE traffic as defined in the parameter section.

The test equipment MUST start to measure and record all specified KPIs. Continue the test until all CVEs are sent.

The measured KPIs MUST meet all the test results validation criteria defined in Appendix A.4.

In addition, the DUT/SUT should either report the detected vulnerabilities in the log correctly or if, for example, a different naming convention is used, there MUST be reference material available that will allow for verification that the correct vulnerability was detected. This reference material MUST be cited in the report.

Appendix B. DUT/SUT Classification

This document aims to classify the DUT/SUT into four different categories based on its maximum supported firewall throughput performance number defined in the vendor datasheet. This classification MAY help users to determine specific configuration scales (e.g., number of ACL entries), traffic profiles, and attack traffic profiles, scaling those proportionally to DUT/SUT sizing category.

The four different categories are Extra Small (XS), Small (S), Medium (M), and Large (L). The RECOMMENDED throughput values for the following categories are:

Extra Small (XS) - Supported throughput less than or equal to1Gbit/s

Small (S) - Supported throughput greater than 1Gbit/s and less than or equal to 5Gbit/s

Medium (M) - Supported throughput greater than 5Gbit/s and less than or equal to 10Gbit/s

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

Authors' Addresses

Balamuhunthan Balarajah
Berlin
Germany
Email: bm.balarajah@gmail.com

Carsten Rossenhoevel
EANTC AG
Salzufer 14
10587 Berlin
Germany
Email: cross@eantc.de

Brian Monkman
NetSecOPEN
417 Independence Court
Mechanicsburg, PA 17050
United States of America
Email: bmonkman@netsecopen.org

Benchmarking Methodology Working Group
Internet-Draft
Intended status: Informational
Expires: 26 October 2023

Z. Lai
H. Li
Y. Deng
Q. Wu
J. Liu
Tsinghua University

24 April 2023

Problems and Requirements of Evaluation Methodology for Integrated Space and Terrestrial Networks

draft-lai-bmwg-istn-methodology-03

Abstract

With the rapid evolution of the aerospace industry, many "NewSpace" upstarts are actively deploying their mega-constellations in low earth orbits (LEO) and building integrated space and terrestrial networks (ISTN), promising to provide pervasive, low-latency, and high-throughput Internet service globally. Due to the high manufacturing, launching, and updating cost of LEO mega-constellations, it is expected that ISTNs can be well designed and evaluated before the launch of satellites. However, the progress of designing, assessing, and understanding new network functionalities and protocols for futuristic ISTNs faces a substantial obstacle: lack of standardized evaluation methodology with acceptable realism (e.g. can involve the unique dynamic behaviors of ISTNs), flexibility, and cost. This memo first reviews the unique characteristics of LEO mega-constellations. Further, it analyzes the limitation of existing evaluation and analysis methodologies under ISTN environments. Finally, it outlines the key requirements of future evaluation methodology tailored for ISTNs.

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 26 October 2023.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction

Integrated Space and Terrestrial Networks (ISTN), combining diverse spacecrafts and ground infrastructures, are extending the frontier of todays terrestrial network, promising to provide low-latency, high-bandwidth Internet access with broader coverage globally.

Low earth orbit (LEO) satellites are the key building block for constructing ISTNs. Recently, we have witnessed a renaissance in the space industry, stimulating an exponential increase in constructing mega-constellations. As compared with their predecessor, cutting-edge satellites can be equipped with high-resolution sensors, space-grade multi-core processors, high-data-rate communication links, and multifunctional space software.

While ISTNs hold great promise, to completely unleash the network potential of emerging ISTN, it still needs to address a series of new technical issues. The unique characteristics of LEO satellites (e.g., high-dynamics), not only impose new challenges at various layers of the ISTN networking stack but also open the door to many new technical problems. With many unexplored problems facing the "NewSpace" industry, it is thus foreseen that in the near future, there will be a surge of new efforts (e.g. topology, addressing, routing, transport, etc.) to rethink and reshape the networking stack in ISTNs. In addition, the cost/timeline of manufacturing, launching, operating, and updating satellite constellations is typically much higher/longer than that in traditional terrestrial networks. Therefore, it is expected that new network functionalities and protocols can be well evaluated before they are launched and deployed in realistic satellite constellations.

However, the network community lacks the proper analysis tools and evaluation methodologies that can mimic the unique dynamic behavior to analyze many of the ISTN challenges that have been highlighted by prior works. At high level, existing evaluation methodologies in the network community can typically be grouped into three major categories: live networks or platforms, simulation, and emulation. However, the feasibility and flexibility of live satellite networks are technically and economically limited. The abstraction level of network simulation could be too high to capture low-level system effects. Existing network emulators fail to characterize the high dynamicity of LEO satellites and thus cannot accomplish an environment with acceptable fidelity. The community hence needs a reasonable and standardized evaluation methodology to build proper experimental environments which can mimic the behavior of ISTNs, supporting the community to deeply understand the problems, and to evaluate new functionalities and protocols (e.g. for topology, addressing, routing, transport, etc.) for ISTNs, before the mega-

constellation is completely deployed. In this memo, we first review the unique characteristics of emerging LEO mega-constellations and the key challenges of integrating satellites and terrestrial Internet. Further, we analyze the limitation of existing network analysis tools and evaluation methodologies in ISTNs. Finally, we outline key requirements of evaluation methodologies tailored for ISTNs.

2. Notation and Terminology

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].

This document uses the following acronyms and terminologies:

Mega-constellation: A group of satellites working as a system.

LEO: Low Earth Orbit with an altitude no more than 2000 km.

MEO: Medium Earth Orbit with an altitude from 2000 km to 35786 km.

GEO: Geostationary Earth Orbit with an altitude of 35786 km.

NGSO: Non-Geostationary Orbit

LSN: LEO Satellite Networks

ISTN: Integrated Satellite and Terrestrial Network

ISL: Inter-satellite Links

EO: Earth Observation

GS: Ground Station

AS: Autonomous System

EOS: Earth Observation Satellite

BGP: Border Gateway Protocol [RFC4271]

OSPF: Open Shortest Path First [RFC2328]

VM: Virtual Machine

3. Quick Primer for Integrated Space and Terrestrial Networks

Emerging mega-constellations with inter-satellite links (ISLs) can build a satellite network in outer space, and further be integrated with terrestrial ground infrastructures to construct an integrated space and terrestrial network (ISTN).

3.1. Mega-constellation

A constellation is a group of satellites working as a system to give a coverage of the earth surface, among which satellites are positioned in fixed orbital planes with regular trajectories. LEO and MEO satellites often belong to a constellation, because a single satellite only covers a small area with high angular velocity. Thus, continuous coverage over an area could be maintained by the relay within a constellation, as compared with GEO satellites that only provides a permanent coverage over a target area. Walker Delta constellation is the most common formation for constellations. It is defined as a bunch of circular orbits with a fixed inclination, satellite number, number of equally spaced planes and the relative spacing between satellites in adjacent planes. The famous Ballard rosette constellation is another name of Walker Delta constellation, where it uses a different notation. Near-polar Walker Star is one of this kind, initially used by Iridium [Iridium]. Constellations with a higher inclination give the polar regions more chances to get accessed. The well-known emerging commercial constellations are Starlink [Starlink-Fcc], Kuiper [Kuiper-Fcc] and Telesat [Telesat-Fcc], as shown in Table 1 below. And all of them contain more than one shell.

Name and Shell	Altitude (km)	Inclination (degree)	# of orbits	# of satellites per orbit
Starlink S1	550	53	72	22
Starlink S2	540	53.2	72	22
Starlink S3	570	70	36	20
Starlink S4	560	97.6	6	58
Starlink S5	560	97.6	4	43
Kuiper K1	630	51.9	34	34
Kuiper K2	610	42	36	36
Kuiper K3	590	33	28	28
Telesat T1	1015	98.98	27	13
Telesat T2	1325	50.88	40	33

Table 1: Mega-constellation information.

3.2. Topological Dynamics

Unlike geostationary satellite networks or terrestrial core infrastructure that keep a stable topology, LEO satellite networks suffer from high topological dynamics, since LEO satellites move fast, causing short-lived coverage for fixed terrestrial users. For example, considering the first shell of Starlink Phase-I, a fixed user sees each satellite for only up to 3 minutes in one pass, after which the satellite moves away from the users perspective. Table 2 shows the medium space-ground link churn intervals [link-churn-interval] between existing GS and constellations. If

each GS only uses one antenna to connect the satellite with the shortest distance, the medium interval is no more than one minute. This kind of high dynamic motion incurs frequent link changes between LEO satellites and GS or users, thus causing frequent topology changes. Moreover, inter-satellites visibility may also change if LEO satellites move in different directions or in different shells, resulting in connectivity change of ISLs.

Such high LEO dynamics can impose significant challenges in the networking stack of ISTNs. The high dynamics make the logical network and mega-constellations and physical ISTN inconsistent. One big challenge is how to overcome the routing oscillation properly in the high dynamic ISTN environment. Frequent satellite-GS link changes make the inter-connectivity of space and ground segments in ISTNs unstable. Thus, the routing have to be re-calculated every time the link changes. In addition, the topological dynamics also result in RTT fluctuations in end-to-end paths, involving new challenges for congestion control in ISTNs, as a RTT variation observed by end-host might not indicate congestions.

Name	Interval (s)
Starlink	3.0901
Kuiper	5.0562
OneWeb	10.6824
Telesat	45.5696

Table 2: Space-ground link churn interval.

3.3. Limited Resources

Space resources (e.g. CPU, energy) on satellites are limited, as compared with terrestrial network. Since resource-constrained satellites such as nanosatellites are only able to carry certain sennsing or transferring missions, energy-consuming or complex tasks may not be achievable in these satellites. Such complicated tasks include on-board target identification and instant and continuous disaster monitoring.

For example, the CPU frequency of current spaceborne processors (e.g. RAD5545 [RAD5545], RAD750 [RAD750]) is only up to 466MHz per core. More recently, some low energy-consuming commodity processors are

used in space to complete certain remote sensing missions under a limited CPU capacity. [raspberry-pi] With a constrained computation ability and limited storage and energy, satellite functions and lifetime are greatly repressed.

3.4. Long Manufacturing and Deployment Duration

Different from terrestrial network infrastructures, the timeline of manufacturing and deploying satellite networks could be much longer due to the high cost and complex process during the development and launch period. Satellites, as well as the orbit and spectrum they used, have to be regulated, and launches have to be carefully scheduled (e.g. to avoid the impact of poor weather conditions). In addition, the maintenance and update cost of a satellite network is also typically much higher than that in a terrestrial network.

For example, a review of 24 Air Force and Navy space vehicle (SV) development programs found that on average it took about 7.5 years from contract start to launch a government satellite. [Development-Timeline] Commercial satellite programs typically take 2 to 3 years from contract start to launch. [Production-Cycles] SpaceXs Starlink constellation plan to launch about 42,000 satellites to construct a mega-constellation in outer space. On 15 October 2019, the United States Federal Communications Commission (FCC) submitted filings to the International Telecommunication Union (ITU) on SpaceX's behalf to arrange spectrum for 30,000 additional Starlink satellites to supplement the 12,000 Starlink satellites already approved by the FCC. As of the end of 2022, SpaceX has launched about 3,000 Starlink satellites, which is about 7% of the ultimate constellation plan consisting of 42,000 satellites. In the meanwhile, Starlink has attracted more than one million subscribers. Foreseeably, it may take many years to complete the entire constellation deployment. Even the first phase of Starlink which consists of about 4400 satellites is not expected to be completed until 2024.

4. Problem Statement: We Need the Right Evaluation Methodology

The unique characteristics of LEO mega-constellations involve new challenges on various layers of the networking stack of ISTNs. On one hand, it is foreseen that in the near future, there will be a surge of new network functionalities and protocols designed or optimized for ISTNs. On the other hand, because the cost/timeline of manufacturing, launching, operating, and updating satellite constellations is typically much higher/longer than that in traditional terrestrial networks, it is expected that those new network functionalities and protocols tailored for ISTNs should be well evaluated before they are launched and deployed in realistic

satellite constellations.

Existing methodologies for testing, assessing, and understanding a network functionality or protocol can typically be classified into three categories: (1) live networks; (2) network simulators; and (3) network emulators. The subsections discuss these three categories of network evaluation methodologies, along with their using deficiencies and possible remedies respectively.

4.1. Live networks and platforms

Representative platforms such as Emulab [Emulab] and Sparta [Sparta] are successful pioneers that build a large-scale experimental network environment. These test environments are originally designed to provide special and exclusive test services for affiliated universities, scientific research institutions or Internet business companies. And for the resource competition, each independent experiment needs to completely monopolize a part of the test bed, so the researcher cannot deploy the experiment until being allocated with enough nodes. PlanetLab [PlanetLab] is truly global ground testbed prototype. Started from 2003, it consists of 1353 nodes at 717 sites spanning 48 countries. Together the nodes form a global network system to support new design of network services.

The live platforms described above were initially proposed for terrestrial networks and they are developed and repaired at the same time. The key limitation of them in an ISTN environment is that they are designed for terrestrial network experiments, and do not incorporate the realistic characteristic of LEO mega-constellations to support experiments and evaluations in ISTNs.

We may search for help from live satellites, but still there is only limited help. It seems that with the help of live ISTN, researchers are capable to assess, verify and evaluate their ideas and thoughts. Live ISTN can give a real constellation-consistency and stack-consistency testing environment. However, current satellites only provide users a bent-pipe service, which is purely relaying the transmission messages, such as the current deployment of Starlink [Starlink]. The construction is far from a comprehensive ISTN, so the research scope is limited. Even if there is a live ISTN, it lacks flexibility, owning to the inconvenient control over satellites. Besides, the access to satellites is also limited.

Therefore, live networks or platforms for terrestrial networks can give us a large-scale experimental environment but they lack the support for ISTN characteristics. On the other hand, live ISTN is able to guarantee a real space environment, but it is not that affordable and flexible.

4.2. Network Simulators

Simulators are tools that enable researchers to reproduce their testing experiments by simulating a real-world process or system over time. Simulators work by using discrete event simulation to calculate the interactive states among all the network entities, ranging from switches, routers, nodes, access points, links and so on. While working fast and efficiently, the fidelity is only brought by the state variable changes at discrete points.

Such tools like Systems Tool Kit (STK) [Systems-Tool-Kit] and General Mission Analysis Tool (GMAT) [General-Mission-Analysis-Tool] are good for orbit analysis. STK is a powerful tool to help researchers to model the behavior of mission entities in aerospace, telecommunications and so forth. It also provides visualization and analysis functions. GMAT is a similar tool for space trajectory optimization and mission modeling. Nevertheless, these tools do not support networking simulations such as topology and protocol simulations. ns-3 [ns-3] goes a step further with support for Internet simulation, but on the contrary, it was not designed for ISTN and lacks the support for high-dynamics of ISTN. StarPerf [StarPerf] is a simulator that helps researchers to study network performance under a range of constellation conditions. But still, it lacks the ability to support interactive network traffic simulation and system codes in the systems.

Overall, while flexible and low-cost, the realism of simulators is not content enough, because they are difficult to describe the low-level characteristics. In other words, simulators are being too object-oriented to involve additional overhead in the actual execution of programs. Besides, when accessing the network performance, a number of recent emerging algorithms for congestion control, reliable transmission or even protocols are not supported, for example ns-3 [ns-3] only supports basic congestion control like Reno [RFC6582] and so forth, so the need to work with some new algorithms cannot be satisfied and the research to discover new mechanisms, such as new routing algorithms and re-transmission schemes, is extensively prohibited. Another problem of simulators, such as ns-3 [ns-3], is that it difficult to trace or understand the previous codes, without appropriate documentations. Simulators usually face the additional compatibility problem, which means they are not portable with other systems, or they do not support kernel codes. Since there are multiple simulators developed by different group of users, sometimes users are required to be familiar with the writing language, scripting style and modelling technique.

4.3. Network Emulators

Emulators are another kind of paradigm for network evaluation over a virtual network. The difference between a simulator and an emulator is that emulators leverage VM or containers to keep the realism which is close to actual performances. Therefore, in emulators, virtual nodes. virtual network links, virtual models of traffic, and protocols are all applied. Emulators are capable to run real kernel and application code. Thus, emulators not only support diverse topology design, but also protocol emulation in a synthetic network environment. They emulate the network behavior in a more real way. Mininet [Mininet] is commonly regarded as the most illustrious emulator for networking with its strong ability to support experiments with Software-Defined Networking (SDN) [Software-defined-networking] systems. EstiNet [EstiNet] is another emulator that supports evaluating and testing the performances of software-defined networks. Based on containers, they can emulate real TCP/IP protocol stack in the Linux kernel. However, existing emulation tools lack the ability to construct the dynamic links and orbits in ISTN like simulators. Thus, more problems could happen in higher-level protocols such as routing protocols (e.g. OSPF and BGP). Besides, since emulators run containers or virtual machines which occupy more software overhead, as compared with simulators, it will be hard to emulate the large-scale mega-constellations.

To conclude, emulators are relatively good methodologies for network experiments, but emulators still have limitations when using them for ISTN research. While keeping a moderate realism by using VM or containers for entity emulation and flexibility, emulators still lack the supports for ISTN characteristics, such as frequent link changes, satellite network topology uncertainty, and so on. More specifically, current emulators only support fixed network topology emulation. It is not flexible to emulate the time-varying link packet loss, bandwidth, and other traits. A possible way is to frequently replace the link with a new one from time to time sequentially for the entire ISTN. However, it is far from the real situation. Besides, VM or containers are able to deploy a range of network nodes in a physical server, but the actual CPU, memory and other resources should not be shared in reality for each satellite. In addition, it is still difficult to emulate thousands or ten thousand of satellites for ISTN even with VM or containers, subject to hardware limitations. For flexibility, some emulators do not support a good network animator tool. Especially in ISTN emulation, GUI is important for users to observe and analyze orbit trajectories and real time satellite positions.

4.4. Summary

In this section, we explain the necessity of an evaluation methodology specifically for ISTNs. Then we demonstrate the problems with existing methodologies related to ISTNs. The performance comparison result is shown in Table 3. Above all, ISTNs should be designed first and then launched. Live satellites enable good realism but they lack flexibility and require very high cost as well as a very long deployment period. Other testing tools such as simulators and emulators are either functional for merely aerospace analysis or simply terrestrial networks. None of the existing methodologies guarantees a practical and user-friendly methodology while keeping the evaluation environment realism with low costs.

Platform/Tool	Realism	Flexibility	Cost	Cross-domain Dataset Support
Live satellite network	Y	N	High	Y
ns-3 [ns-3]	N	Y	Low	N
Hypatia [Hypatia]	N	Y	Low	N
StarPerf [StarPerf]	N	Y	Low	N
Mininet [Mininet]	N	Y	Low	N

Table 3: Existing platforms/tools for network analysis and evaluation. (Y for yes/N for no)

5. Requirements: New Evaluation Methodology Tailored for ISTNs

A proper evaluation methodology tailored for ISTNs is expected to help developers, researchers, engineers to explore various design-space of the networking stack of ISTNs in a technically and economically feasible manner. Based on the comparative analysis results in the prior section, we sum up the following requirements for the new evaluation methodology in ISTNs.

5.1. Realism

The first requirement is realism. Realism represents the testing authenticity and fidelity. The evaluation methodology is expected to keep the actual characteristics of mega-constellations. In other words, the orbit-level information including the latitude, longitude, and height of each satellite in any given time and the same information for GS and elevation angles of antennas of each GS. Note that the constellation information also determines the visibility, links and even topology of ISTN.s Since the mega-constellations are unstable, how the temporal satellite locations, visibility, link propagation delays and so on should also be considered carefully. In addition, it requires the network nodes to communicate and negotiate their messages following the actual protocol process. For example, when doing a test for OSPF in an ISTN, we would like the nodes to send Hello packets, Link-State-Request (LSR) packets, Link-State-Update (LSU) packets and so on. A real network stack is preferred to provide researchers an opportunity to see the performance of different protocols in ISTNs.

5.2. Flexibility

Another requirement is flexibility and feasibility. The testing methodology should be technically easy to use and easy to learn. Without extra modifications or process, the methodology should help researchers learn and use it without much effort and can evaluate their ideas as they wish, which means it should support flexible, controllable environments for researchers.

5.3. Low-cost and Easy-to-use

Meanwhile, the evaluation methodology is expected to be low-cost. A well-acceptable methodology should be economically feasible for users to create an evaluation environment. Researchers do not want to conduct their tests all in live ISTN, which is over-cumbersome and unaffordable, let alone launching their own spacecraft. Even if there are a number of orbiting satellites, whether users can easily gain access to satellites is also a problem.

5.4. Cross-domain Dataset Support

The evaluation methodology is expected to be driven by realistic datasets from multi-dimensions to support its realism. Multi-dimension refers to multi-disciplinary research on ISTNs. Since a standard ISTN evaluation methodology not only contains high-level benchmarks from topology, routing to transmission, but also considers the low-level traits such as wireless link conditions, weather conditions and Earth rotations. To be more concrete, the former one

requires knowledge in networks while the latter one relies more on aerospace. Hence, to build a high-fidelity methodology, we need community efforts both from networks and aerospace. On the other hand, an authentic dataset is an indispensable element for data driven testing methodology. Actual data is the first step to obtain a realistic emulation. with characteristics of a real ISTN. Thus, the dataset is a collection of messages for testing, in which geographical mega-constellation information (orbit number, satellite number, height), orbital information (orbit inclination angle and link strategies), weather information as well as ground station information (positions, antenna angle and so forth) are involved.

6. Conclusion

To conclude, the emergence of mega-constellations brings us new opportunities for the development of ISTN that extends the Internet to the space era. Combined with terrestrial networks, ISTN is expected to supply pervasive, low-latency and high-speed services to users globally, which greatly enhances the current Internet. At the same time, the unique characteristics (e.g. high-dynamics) of ISTN impose challenges in topology, routing, transportation, application, and security. However, we simply believe addressing the challenges also gives us open opportunities for future research by our community-driven effort. To accelerate the research speed and to help make testing more feasible, new methodologies that satisfy user requirements should be proposed. To this extent, this draft reviews the limitation of existing network analysis tools in ISTNs, considering the unique characteristics of emerging LSNs and the key challenges. This draft further analyzes the limitation of existing evaluation methodologies in ISTN environments. Finally, this draft outlines key requirements of evaluation methodologies tailored for future ISTNs.

7. Acknowledgements

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

This entire draft discusses security considerations from different perspectives of ISTN in Section 3.

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>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC6582] Henderson, T., Floyd, S., Gurtov, A., and Y. Nishida, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 6582, DOI 10.17487/RFC6582, April 2012, <https://www.rfc-editor.org/info/rfc6582>.

10.2. Informative References

[Development-Timeline] "Development-Timeline", <http://www.iceaaonline.com/ready/wp-content/uploads/2014/03/Davis-Satellite-ICEAASoCal-090915.pdf>.

[Emulab] "Emulab", <https://www.emulab.net/portal/frontpage.php>.

[EstiNet] "EstiNet", <http://www.estinet.com/fckimages/14117014621397232509.pdf>.

[General-Mission-Analysis-Tool] "General-Mission-Analysis-Tool", <https://en.wikipedia.org/wiki/General_Mission_Analysis_Tool>.

[Hypatia] "Hypatia", <http://people.inf.ethz.ch/asingla/papers/imc2020-hypatia.pdf>.

[Iridium] "Iridium", <https://en.wikipedia.org/wiki/Iridium>.

[Kuiper-Fcc] "Kuiper-Fcc", <https://www.itu.int/ITU-R/space/asreceived/Publication/DisplayPublication/8716>.

[link-churn-interval] "link-churn-interval", <https://conferences.sigcomm.org/hotnets/2021/accepted.html>.

[Mininet] "Mininet", <http://mininet.org/>.

[ns-3] "ns-3", <https://www.nsnam.org/>.

[PlanetLab]
"PlanetLab", <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.99.7006&rep=rep1&type=pdf>.

[Production-Cycles]
"Production-Cycles", <http://www.futron.com/upload/wysiwyg/Resources/Whitepapers/Satellite_Manufacturing_Productio n_Cycles_0504.pdf>.

[RAD5545] "RAD5545", <https://www.baesystems.com/en-media/uploadFile/20210407074148/1434594567983.pdf>.

[RAD750] "RAD750", <https://www.baesystems.com/en-media/uploadFile/20210407041505/1434555689265.pdf>.

[raspberry-pi]
"raspberry-pi", <https:// www.raspberrypi.com/news/raspberry-pi-in-space/>.

[Software-defined-networking]
"Software-defined-networking", <https://en.wikipedia.org/wiki/Software-defined_networking>.

[Sparta] "Sparta", <https://s3-us-west-2.amazonaws.com/ieeeshutpages/xplore/xplore-shut-page.html>.

[Starlink] "Starlink", <https://en.wikipedia.org/wiki/Starlink>.

[Starlink-Fcc]
"Starlink-Fcc", <https://fcc.report/IBFS/SAT-LOA-20161115-00118/1158350>.

[StarPerf] "StarPerf", <https://www.semanticscholar.org/paper/StarPerf%3A-Characterizing-Network-Performance-for-Lai-Li/18aa32309eb857afdace1ad02d1091ae64dcd330>.

[Systems-Tool-Kit]
"Systems-Tool-Kit", <https://www.agi.com/products/gmat>.

[Telesat-Fcc]
"Telesat-Fcc", <https://fcc.report/IBFS/SAT-MPL-20200526-00053/2378318.pdf>.

Authors' Addresses

Zeqi Lai
Tsinghua University
30 ShuangQing Ave
Beijing
100089
China
Email: zeqilai@tsinghua.edu.cn

Hewu Li
Tsinghua University
30 ShuangQing Ave
Beijing
100084
China
Email: lihewu@cernet.edu.cn

Yangtao Deng
Tsinghua University
30 ShuangQing Ave
Beijing
100084
China
Email: dengyt21@mails.tsinghua.edu.cn

Qian Wu
Tsinghua University
30 ShuangQing Ave
Beijing
100084
China
Email: wuqian@cernet.edu.cn

Jun Liu
Tsinghua University
30 ShuangQing Ave
Beijing
100084
China

Email: juneliu@mail.tsinghua.edu.cn

Benchmarking Methodology Working Group
Internet-Draft
Intended status: Informational
Expires: 1 January 2023

G. Lencse
Szechenyi Istvan University
K. Shima
IIJ Innovation Institute
30 June 2022

Benchmarking Methodology for Stateful NATxy Gateways using RFC 4814 Pseudorandom Port Numbers
draft-lencse-bmwg-benchmarking-stateful-04

Abstract

RFC 2544 has defined a benchmarking methodology for network interconnect devices. RFC 5180 addressed IPv6 specificities and it also provided a technology update, but excluded IPv6 transition technologies. RFC 8219 addressed IPv6 transition technologies, including stateful NAT64. However, none of them discussed how to apply RFC 4814 pseudorandom port numbers to any stateful NATxy (NAT44, NAT64, NAT66) technologies. We discuss why using pseudorandom port numbers with stateful NATxy gateways is a difficult problem. We recommend a solution limiting the port number ranges and using two phases: the preliminary phase and the real test phase. We show how the classic performance measurement procedures (e.g. throughput, frame loss rate, latency, etc.) can be carried out. We also define new performance metrics and measurement procedures for maximum connection establishment rate, connection tear down rate and connection tracking table capacity measurements.

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 1 January 2023.

Copyright Notice

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

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

Table of Contents

1. Introduction

[RFC2544] has defined a comprehensive benchmarking methodology for network interconnect devices, which is still in use. It was mainly IP version independent, but it used IPv4 in its examples. [RFC5180] addressed IPv6 specificities and also added technology updates, but declared IPv6 transition technologies out of its scope. [RFC8219] addressed the IPv6 transition technologies, including stateful NAT64. It has reused several benchmarking procedures from [RFC2544] (e.g. throughput, frame loss rate), it has redefined the latency measurement, and added further ones, e.g. the PDV (packet delay variation) measurement.

However, none of them discussed, how to apply [RFC4814] pseudorandom port numbers, when benchmarking stateful NATxy (NAT44, NAT64, NAT66) gateways. We are not aware of any other RFCs that address this question.

First, we discuss why using pseudorandom port numbers with stateful NATxy gateways is a hard problem.

Then we recommend a solution.

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. Pseudorandom Port Numbers and Stateful Translation

In its appendix, [RFC2544] has defined a frame format for test frames including specific source and destination port numbers. [RFC4814] recommends to use pseudorandom and uniformly distributed values for both source and destination port numbers. However, stateful NATxy (NAT44, NAT64, NAT66) solutions use the port numbers to identify connections. The usage of pseudorandom port numbers causes different problems depending on the direction.

* As for the private to public direction, pseudorandom source and destination port numbers could be used, however, this approach would be a denial of service attack against the stateful NATxy gateway, because it would exhaust its connection tracking table capacity. To that end, let us see some calculations using the recommendations of RFC 4814:

- The recommended source port range is: 1024-65535, thus its size is: 64512.

- The recommended destination port range is: 1-49151, thus its size is: 49151.

- The number of source and destination port number combinations is: 3,170,829,312.

We note that section 12 of [RFC2544] also requires testing with 256 destination networks, which further increases the number of connection tracking table entries.

* As for the public to private direction, the stateful DUT (Device Under Test) would drop any packets that do not belong to an existing connection, therefore, the direct usage of pseudorandom port numbers from the above-mentioned ranges is not feasible.

3. Test Setup and Terminology

Our methodology works with any IP version. We use IPv4 in the Test Setup shown in Figure 1 to facilitate its easy understanding based on the well-known stateful NAT44 (also called NAPT: Network Address and Port Translation) solution.

```
                 +-------------------------------------+
       10.0.0.2  |Initiator                   Responder| 198.19.0.2
  +-------------|                Tester               |<------------+
  | private IPv4|                         [state table]| public IPv4 |
  |             +-------------------------------------+             |
  |                                                                 |
  |             +-------------------------------------+             |
  |    10.0.0.1 |                 DUT:                | 198.19.0.1  |
  +------------>|        Sateful NATxy gateway        |-------------+
    private IPv4|     [connection tracking table]     | public IPv4
                +-------------------------------------+
```

Figure 1: Test Setup for benchmarking stateful NATxy gateways

As for transport layer protocol, [RFC2544] recommended testing with UDP, and it was kept also in [RFC8219]. For the general recommendation, we also keep UDP, thus the port numbers in the following text are to be understood as UDP port numbers. We discuss the limitation of this approach in Section 6.

We define the most important elements of our proposed benchmarking system as follows.

* Connection tracking table: The stateful NATxy gateway uses a connection tracking table to be able to perform the stateful translation in the public to private direction. Its size, policy and content are unknown for the Tester.

* Four tuple: The four numbers that identify a connection are source IP address, source port number, destination IP address, destination port number.

* State table: The Responder of the Tester extracts the four tuple from each received test frame and stores it in its state table. Recommendation is given for writing and reading order of the state table in Section 4.10.

* Initiator: The port of the Tester that may initiate a connection through the stateful DUT in the private to public direction. Theoretically, it can use any source and destination port numbers from the ranges recommended by [RFC4814]: if the used four tuple does not belong to an existing connection, the DUT will register a new connection into its connection tracking table.

* Responder: The port of the Tester that may not initiate a connection through the stateful DUT in the public to private direction. It may send only frames that belong to an existing connection. To that end, it uses four tuples that have been previously extracted from the received test frames and stored in its state table.

* Preliminary test phase: Test frames are sent only by the Initiator to the Responder through the DUT to fill both the connection tracking table of the DUT and the state table of the Responder. This is a newly introduced operation phase for stateful NATxy benchmarking. The necessity of this phase is explained in Section 4.2.

* Real test phase: The actual test (e.g. throughput, latency, etc.) is performed in this phase after the completion of the preliminary test phase. Test frames are sent as required (e.g. bidirectional test or unidirectional test in any of the two directions).

4. Recommended Benchmarking Method

4.1. Restricted Port Number Ranges

The Initiator SHOULD use restricted ranges for source and destination port numbers to avoid the denial of service attack like event against the connection tracking table of the DUT described in Section 2. The size of the source port number range SHOULD be larger (e.g. in the order of a few times ten thousand), whereas the size of the destination port number range SHOULD be smaller (may vary from a few to several hundreds or thousands as needed). The rationale is that source and destination port numbers that can be observed in the Internet traffic are not symmetrical. Whereas source port numbers may be random, there are a few very popular destination port numbers (e.g. 443, 80, etc., see [IIR2020]) and others hardly occur. And we have found that their role is also asymmetric in the Linux kernel routing hash function [LEN2020].

The product of the sizes of the two ranges can be used as a parameter. The performance of the stateful NATxy gateway MAY be examined as a function of this parameter.

4.2. Preliminary Test Phase

The preliminary phase serves two purposes:

1. The connection tracking table of the DUT is filled. It is important, because its maximum connection establishment rate may be lower than its maximum frame forwarding rate (that is throughput).

2. The state table of the Responder is filled with valid four tuples. It is a precondition for the Responder to be able to transmit frames that belong to connections exist in the connection tracking table of the DUT.

Whereas the above two things are always necessary before the real test phase, the preliminary phase can be used without the real test phase. It is done so, when the maximum connection establishment rate is measured (as described in Section 4.5).

A preliminary test phase MUST be performed before all tests performed in the real test phase. In this phase, the following things happen:

1. The Initiator sends test frames to the Responder through the DUT at a specific frame rate.

2. The DUT performs the stateful translation of the test frames and it also stores the new combinations in its connection tracking table.

3. The Responder receives the translated test frames and updates its state table with the received four tuples. The responder transmits no test frames during the preliminary phase.

When the preliminary test phase is performed in preparation to the real test phase, the applied frame rate and the duration of the preliminary phase SHOULD be carefully selected so that:

* The applied frame rate be safely lower than the maximum connection establishment rate.

* Enough four tuples be stored in the state table of the Responder so that it can generate frames with the proper distribution of the four tuples.

Please refer to Section 4.4 for further conditions regarding timeout and port number combinations.

4.3. Consideration of the Cases of Stateful Operation

We consider the most important Events that may happen during the operation of a stateful NATxy gateway, and the Actions of the gateway as follows.

1. EVENT: A packet not belonging to an existing connection arrives in the private to public direction. ACTION: A new connection is registered into the connection tracking table and the packet is translated and forwarded.

2. EVENT: A packet not belonging to an existing connection arrives in the public to private direction. ACTION: The packet is discarded.

3. EVENT: A packet belonging to an existing connection arrives (in any dicection). ACTION: The packet is translated and forwarded and the timeout counter of the corresponding connection tracking table entry is reset.

4. EVENT: A connection tracking table entry times out. ACTION: The entry is deleted from the connection tracking table.

Due to "black box" testing, the Tester is not able to directly examine (or delete) the entries of the connection tracking table. But the entires can be and MUST be controlled by setting an appropriate timeout value and carefully selecting the port numbers of the packets (as described in Section 4.4) to be able to produce meaningful and repeatable measurement results.

We aim to support the measurement of the following performance characteristics of a stateful NATxy gateway:

1. maximum connection establishment rate

2. all "classic" performance metrics like throughput, frame loss rate, latency, etc.

3. connection tear down rate

4. connection tracking table capacity

4.4. Control of the Connection Tracking Table Entries

It is necessary to control the connection tracking table entries of the DUT in order to achieve clear conditions for the measurements. We can simply achieve the following two extreme situations:

1. All frames create a new entry in the connection tracking table of the DUT and no old entries are deleted during the test. This is required for measuring the maximum connection establishment rate.

2. No new entries are created in the connection tracking table of the DUT and no old ones are deleted during the test. This is ideal for the real test phase measurements, like throughput, latency, etc.

From this point we use the following three assumptions:

1. A single source address destination address pair is used for all tests. We make this assumption for simplicity. Of course, we are aware that [RFC2544] requires testing also with 256 different destination networks.

2. The connection tracking table of the stateful NATxy is large enough to store all connections defined by the different source port number destination port number combinations.

3. Each experiment is started with an empty connection tracking table. (It can be ensured by deleting its content before the experiment.)

The first extreme situation can be achieved by

* using different source port number destination port number combinations for every single test frame in the preliminary phase and

* setting the UDP timeout of the NATxy gateway to a value higher than the length of the preliminary phase.

The second extreme situation can be achieved by

* enumerating all the possible source port number destination port number combinations in the preliminary phase and

* setting the UDP timeout of the NATxy gateway to a value higher than the length of the preliminary phase plus the gap between the two phases plus the length of the real test phase.

[RFC4814] REQUIRES pseudorandom port numbers, which we believe is a good approximation of the distribution of the source port numbers a NATxy gateway on the Internet may face with.

We note that although the enumeration of all possible source port number destination port number combinations is not a requirement for the first extreme situation and the usage of different source port number destination port number combinations is not a requirement for the second extreme situation, pseudorandom enumeration of source port number destination port number combinations is a good solution in both cases. It may be computing efficiently generated by preparing a random permutation of the previously enumerated all possible source port number destination port number combinations using Dustenfeld's random shuffle algorithm [DUST1964].

Important warning: in normal (non-NAT) router testing, the port number selection algorithm, whether it is pseudo-random or enumerated in increasing (or decreasing) order does not affect final results. However, our experience with iptables shows that if the connection tracking table is filled using port number enumeration in increasing order, then the maximum connection establishment rate of iptables degrades significantly compared to its performance using pseudorandom port numbers [LEN2021].

The enumeration of the source port number destination port number combinations in increasing or decreasing order (or in any other specific order) MAY be used as an additional measurement.

4.5. Measurement of the Maximum Connection Establishment Rate

The maximum connection establishment rate is an important characteristic of the stateful NATxy gateway and its determination is necessary for the safe execution of the preliminary test phase (without frame loss) before the real test phase.

The measurement procedure of the maximum connection establishment rate is very similar to the throughput measurement procedure defined in [RFC2544].

Procedure: The Initiator sends a specific number of test frames using all different source port number destination port number combinations at a specific rate through the DUT. The Responder counts the frames that are successfully translated by the DUT. If the count of offered frames is equal to the count of received frames, the rate of the offered stream is raised and the test is rerun. If fewer frames are received than were transmitted, the rate of the offered stream is reduced and the test is rerun.

The maximum connection establishment rate is the fastest rate at which the count of test frames successfully translated by the DUT is equal to the number of test frames sent to it by the Initiator.

Notes:

1. In practice, we RECOMMEND the usage of binary search.

2. As for the successful translation, the Responder MAY check that the source IP address is different than the original source IP address set by the Initiator. However, it is still not a guarantee for the establishment of the connection in the DUT. Therefore we RECOMMEND the usage of the validation of the connection establishment defined in Section 4.6.

4.6. Validation of Connection Establishment

Due to "black box" testing, the entries of the connection tracking table of the DUT may not be directly examined, but the presence of the connections can be checked easily by sending frames from the Responder to the Initiator in the Real Test Phase using all four tuples stored in the state table of the Tester (at a low enough frame rate). The arrival of all test frames indicates that the connections are really present.

Procedure: When all the desired N number of test frames were sent by the Initiator to the Receiver at frame rate R in the Preliminary Phase for the maximum connection establishment rate measurement, and the Receiver has successfully received all the N frames, the establishment of the connections is checked in the Real Test Phase as follows:

* The Responder sends test frames to the Initiator at frame rate: r=R*alpha, for the duration of N/r using a different four tuple from its state table for each test frame.

* The Initiator counts the received frames, and if all N frames are arrived then the frame rate of the maximum connection establishment rate is raised, otherwise lowered (as well as in the case if test frames were missing in the preliminary phase).

Notes:

* The alpha is a kind of "safety factor", its aim is to make sure that the frame rate used for the validation is not too high, and test may fail only in the case if at least one connection is not present in the connection tracking table of the DUT. (So alpha should be typically less than 1, e.g. 0.8 or 0.5.)

* The duration of N/r and the frame rate of r means that N frames are sent for validation.

* The order of four tuple selection is arbitrary provided that all four tuples MUST be used.

* Please refer to Section 4.9 for a short analysis of the operation of the measurement and what problems may occur.

4.7. Real Test Phase

As for the traffic direction, there are three possible cases during the real test phase:

* bidirectional traffic: The Initiator sends test frames to the Responder and the Responder sends test frames to the Initiator.

* unidirectional traffic from the Initiator to the Responder: The Initiator sends test frames to the Responder but the Responder does not send test frames to the Initiator.

* unidirectional traffic from the Responder to the Initiator: The Responder sends test frames to the Initiator but the Initiator does not send test frames to the Responder.

If the Initiator sends test frames, then it uses pseudorandom source port numbers and destination port numbers from the restricted port number ranges. The responder receives the test frames, updates its state table and processes the test frames as required by the given measurement procedure (e.g. only counts them for throughput test, handles timestamps for latency or PDV tests, etc.).

If the Responder sends test frames, then it uses the four tuples from its state table. The reading order of the state table may follow different policies (discussed in Section 4.10). The Initiator receives the test frames, and processes them as required by the given measurement procedure.

As for the actual measurement procedures, we RECOMMEND to use the updated ones from Section 7 of [RFC8219].

4.8. Measurement of the Connection Tear Down Rate

Connection tear down can cause significant load for the NATxy gateway. The connection tear down performance can be measured as follows:

1. Load a certain number of connections (N) into the connection tracking table of the DUT (in the same way as done to measure the maximum connection establishment rate).

2. Record TimestampA.

3. Delete the content of the connection tracking table of the DUT.

4. Record TimestampB.

The connection tear down rate can be computed as:

connection tear down rate = N / (TimestampB - TimestampA)

The connection tear down rate SHOULD be measured for various values of N.

We assume that the content of the connection tracking table may be deleted by an out-of-band control mechanism specific to the given NATxy gateway implementation. (E.g. by removing the appropriate kernel module under Linux.)

We are aware that the performance of removing the entire content of the connection tracking table at one time may be different from removing all the entries one by one.

4.9. Measurement of the Connection Tracking Table Capacity

The connection tracking table capacity is an important metric of stateful NATxy gateways. Its measurement is not easy, because an elementary step of a validated maximum connection establishment rate measurement (defined in Section 4.6) may have only a few distinct observable outcomes, but some of them they may have different root causes:

1. During the preliminary phase, the number of test frames received by the Responder is less than the number of test frames sent by the Initiator. It may have different root causes, including:

    1. The R frame sending rate was higher than the maximum connection establishment rate. (Note that now the maximum connection establishment rate is considered unknown, because we can not measure the maximum connection establishment without our assumption 2 in Section 4.4!) This root cause may be eliminated by lowering the R rate and re-executing the test. (This step may be performed multiple times, while R>0.)

    2. The capacity of the connection tracking table of the DUT has been exhausted. (And either the DUT does not want to delete connections or the deletion of the connections makes it slower. This case is not investigated further in the preliminary phase.)

2. During the preliminary phase, the number of test frames received by the Responder equals the number of test frames sent by the Initiator. In this case the connections are validated in the Real Test Phase. The validation may have two kinds of observable results:

    1. The number of validation frames received by the Initiator equals the number of validation frames sent by the Responder. (It proves that the capacity of the connection tracking table of the DUT is enough and both R and r were chosen properly.)

    2. The number of validation frames received by the Initiator is less than the number of validation frames sent by the Responder. This phenomenon may have various root causes:

1. The capacity of the connection tracking table of the DUT has been exhausted. (It does not matter, whether some existing connections are discarded and new ones are stored, or the new connections are discarded. Some connections are lost anyway, and it makes validation fail.)

2. The R frame sending rate used by the Initiator was too high in the Preliminary Phase and thus some connections were not established, even though all test frames arrived to the Responder. This root cause may be eliminated by lowering the R rate and re-executing the test. (This step may be performed multiple times, while R>0.)

3. The r frame sending rate used by the Responder was too high in the Real Test Phase and thus some test frames did not arrive to the Initiator, even though all connections were present in the connection tracking table of the DUT. This root cause may be eliminated by lowering the r rate and re-executing the test. (This step may be performed multiple times, while r>0.)

And here is the problem: as the above three root causes are indistinguishable, it is not easy to decide, whether R or r should be decreased.

We have some experience with benchmarking stateful NATxy gateways. When we tested iptables with very high number of connections, the 256GB RAM of the DUT was exhausted and it stopped responding. Such a situation may make the connection tracking table capacity measurements rather inconvenient. We include this possibility in our recommended measurement procedure, but we do not address the detection and elimination of such a situation. (E.g. how the algorithm can reset the DUT.)

For the connection tracking table size measurement, fist we need a safe number: C0. It is a precondition, that C0 number of connections can surely be stored in the connection tracking table of the DUT. Using C0, one can determine the maximum connection establishment rate using C0 number of connections. It is done with a binary search using validation. The result is: R0. The values C0 and R0 will serve as "safe" starting values for the following two searches.

First, we perform an exponential search to find the order of magnitude of the connection tracking table capacity. The search stops if the DUT collapses OR the maximum connection establishment rate severely drops (e.g. to its one tenth) due to doubling the number of connections.

Then, the result of the exponential search gives the order of magnitude of the size of the connection tracking table. Before disclosing the possible algorithms to determine the size of the connection tracking table, we consider a three possible replacement policies of the NATxy gateway:

1. The gateway does not delete any live connections until their timeout expires.

2. The gateway replaces the live connections according to LRU (least recently used) policy.

3. The gateway does a garbage collection, when its connection tracking table is full and a frame with a new four tuple arrives. During the garbage collection, it deletes the K least recently used connections, where K greater than 1.

Now, we examine, what happens and how many validation frames arrive in the there cases. Let the size of the connection tracking table be S, and the number of preliminary frames be N, where S is less than N.

1. The connections defined by the first S test frames are registered into the connection tracking table of the DUT, and the last N-S connections are lost. (It is a another question if the last N-S test frames are translated and forwarded in the preliminary or simply dropped.) During validation, the validation frames with four tuples corresponding to the first S test frames will arrive to the Initiator, and the other N-S validation frames will be lost.

2. All connections are registered into the connection tracking table of the DUT, but the first N-S connections are replaced (and thus lost). During validation, the validation frames with four tuples corresponding to the last S test frames will arrive to the Initiator, and the other N-S validation frames will be lost.

3. Depending on the values of K, S and N, maybe less than S connections will survive. In the worst case, only S-K+1 validation frames arrive, even though, the size of the connection tracking table is S.

If we know that the stateful NATxy gateway uses the first or second replacement policy, and we also know that both R and r rates are low enough, then the final step of determining the size of the connection tracking table is simple. If Responder sent N validation frames and the Initator received N’ of them, then the size of the connection tracking table is N’.

In the general case, we perform a binary search to find the exact value of the connection tracking table capacity within E error. The search chooses the lower half of the interval if the DUT collapses OR the maximum connection establishment rate severely drops (e.g. to its half) otherwise it chooses the higher half. The search stops if the size of the interval is less than the E error.

The algorithms for the general case are defined using C like pseudocode in Figure 2. In practice, this algorithm may be made more efficient in a way that the binary search for the maximum connection establishment rate stops, if an elementary test fails at a rate under RS*beta or RS*gamma during the external search or during the final binary search for the capacity of the connection tracking table, respectively. (This saves a lot a execution time by eliminating the long lasting tests at low rates.)

```
// The binary_search_for_maximum_connection_establishment_rate(c,r)
// function performs a binary search for the maximum connection
// establishment rate in the [0, r] interval using c number of
// connections.

// This is an exponential search for finding the order of magnitude
// of the connection tracking table capacity
// Variables:
//   C0 and R0 are beginning safe values for connection tracking table
       size and connection establishment rate, respectively
//   CS and RS are their currently used safe values
//   CT and RT are their values for current examination
//   beta is a factor expressing unacceptable drop of R (e.g. beta=0.1)
R0=binary_search_for_maximum_connection_establishment_rate(C0,maxrate);
for ( CS=C0, RS=R0;  1; CS=CT, RS=RT )
{
  CT=2*CS;
  RT=binary_search_for_maximum_connection_establishment_rate(CT,RS);
  if ( DUT_collapsed || RT < RS*beta )
    break;
}
// here the size of the connection tracking table is between CS and CT

// This the final binary search for finding the connection tracking
// table capacity within E error
// Variables:
//   CS and RS are the safe values for connection tracking table size
//     and connection establishment rate, respectively
//   C and R are the values for current examination
//   gamma is a factor expressing unacceptable drop of R
//     (e.g. gamma=0.5)
for ( D=CT-CS;  D>E; D=CT-CS )
{
  C=(CS+CT)/2;
  R=binary_search_for_maximum_connection_establishment_rate(C,RS);
  if ( DUT_collapsed || R < RS*gamma)
    CT=C; // take the lower half of the interval
  else
    CS=C,RS=R; // take the upper half of the interval
}
// here the size of the connection tracking table is CS within E error
```

Figure 2: Measurement of the Connection Tracking Table Capacity

4.10. Writing and Reading Order of the State Table

As for writing policy of the state table of the Responder, we RECOMMEND round robin, because it ensures that its entries are automatically kept fresh and consistent with that of the connection tracking table of the DUT.

The Responder can read its state table in various orders, for example:

* pseudorandom

* round robin

We RECOMMEND pseudorandom to follow the spirit of [RFC4814]. Round robin may be used as a computationally cheaper alternative.

5. Implementation and Experience

The "stateful" branch of siitperf [SIITPERF] is an implementation of this concept. It is documented in this (open access) paper [LEN2022].

Our experience with this methodology using siitperf for measuring the scalability of the iptables stateful NAT44 and Jool stateful NAT64 implementations is described in [I-D.lencse-v6ops-transition-scalability].

6. Limitations of using UDP as Transport Layer Protocol

Stateful NATxy solutions handle TCP and UDP differently, e.g. iptables uses 30s timeout for UDP and 60s timeout for TCP. Thus benchmarking results produced using UDP do not necessarily characterize the performance of a NATxy gateway well enough, when they are used for forwarding Internet traffic. As for the given example, timeout values of the DUT may be adjusted, but it requires extra consideration.

Other differences in handling UDP or TCP are also possible. Thus we recommend that further investigations are to be performed in this field.

As a mitigation of this problem, we recommend that testing with protocols usig TCP (like HTTP and HTTPS) can be performed as described in [I-D.ietf-bmwg-ngfw-performance]. This approach also solves the potential problem of protocol helpers may be present in the stateful DUT.

7. Acknowledgements

The authors would like to thank Al Morton, Sarah Banks, Edwin Cordeiro, Lukasz Bromirski and Sandor Repas for their comments.

8. IANA Considerations

This document does not make any request to IANA.

9. 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 stateful NATxy gateways.

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>.

[RFC4814] Newman, D. and T. Player, "Hash and Stuffing: Overlooked Factors in Network Device Benchmarking", RFC 4814, DOI 10.17487/RFC4814, March 2007, <https://www.rfc-editor.org/info/rfc4814>.

[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>.

[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>.

10.2. Informative References

[DUST1964] Durstenfeld, R., "Algorithm 235: Random permutation", Communications of the ACM, vol. 7, no. 7, p.420., DOI 10.1145/364520.364540, July 1964, <https://dl.acm.org/doi/10.1145/364520.364540>.

[I-D.ietf-bmwg-ngfw-performance] Balarajah, B., Rossenhoevel, C., and B. Monkman, "Benchmarking Methodology for Network Security Device Performance", Work in Progress, Internet-Draft, draft-ietf-bmwg-ngfw-performance-13, 12 January 2022, <https://www.ietf.org/archive/id/draft-ietf-bmwg-ngfw-performance-13.txt>.

[I-D.lencse-v6ops-transition-scalability] Lencse, G., "Scalability of IPv6 Transition Technologies for IPv4aaS", Work in Progress, Internet-Draft, draft-lencse-v6ops-transition-scalability-02, 7 March 2022, <https://www.ietf.org/archive/id/draft-lencse-v6ops-transition-scalability-02.txt>.

[IIR2020] Kurahashi, T., Matsuzaki, Y., Sasaki, T., Saito, T., and F. Tsutsuji, "Periodic observation report: Internet trends as seen from IIJ infrastructure - 2020", Internet Infrastructure Review, vol. 49, December 2020, <https://www.iij.ad.jp/en/dev/iir/pdf/iir_vol49_report_EN.pdf>.

[LEN2020] Lencse, G., "Adding RFC 4814 Random Port Feature to Siitperf: Design, Implementation and Performance Estimation", International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems, vol 9, no 3, pp. 18-26., DOI 10.11601/ijates.v9i3.291, 2020, <http://www.hit.bme.hu/~lencse/publications/291-1113-1-PB.pdf>.

[LEN2021] Lencse, G., "Design and Implementation of a Software Tester for Benchmarking Stateful NAT64 Gateways: Theory and Practice of Extending Siitperf for Stateful Tests", it was under review in Computer Communications, then it was significantly rewritten, 2021, <http://www.hit.bme.hu/~lencse/publications/SFNAT64-tester-for-review.pdf>.

[LEN2022] Lencse, G., "Design and Implementation of a Software Tester for Benchmarking Stateful NAT64xy Gateways: Theory and Practice of Extending Siitperf for Stateful

Tests", Computer Communications, vol. 172, no. 1, pp. 75-88, August 1, 2022, DOI 10.1016/j.comcom.2022.05.028, 2022, <http://www.hit.bme.hu/~lencse/publications/ECC-2022-SFNAT64xy-Tester-published.pdf>.

[SIITPERF] Lencse, G., "Siitperf: An RFC 8219 compliant SIIT (stateless NAT64) tester written in C++ using DPDK", source code, available from GitHub, 2019-2022, <https://github.com/lencsegabor/siitperf>.

Appendix A. Change Log

A.1. 00

Initial version.

A.2. 01

Updates based on the comments received on the BMWG mailing list and minor corrections.

A.3. 02

Section 4.4 was completely re-written. As a consequence, the occurrences of the now undefined "mostly different" source port number destination port number combinations were deleted from Section 4.5, too.

A.4. 03

Added Section 4.3 about the consideration of the cases of stateful operation.

Consistency checking. Removal of some parts obsoleted by the previous re-writing of Section 4.4.

Added Section 4.8 about the method for measuring connection tear down rate.

Updates for Section 5 about the implementation and experience.

A.5. 04

Update of the abstract.

Added Section 4.6 about validation of connection establishment.

Added Section 4.9 about the method for measuring connection tracking table capacity.

Consistency checking and corrections.

Authors' Addresses

Gabor Lencse
Szechenyi Istvan University
Gyor
Egyetem ter 1.
H-9026
Hungary
Email: lencse@sze.hu

Keiichi Shima
IIJ Innovation Institute
Iidabashi Grand Bloom, 2-10-2 Fujimi, Tokyo
102-0071
Japan
Email: keiichi@iijlab.net

v6ops
Internet-Draft
Intended status: Informational
Expires: 16 April 2024

G. Lencse
Széchenyi István University
14 October 2023

Scalability of IPv6 Transition Technologies for IPv4aaS
draft-lencse-v6ops-transition-scalability-05

Abstract

Several IPv6 transition technologies have been developed to provide customers with IPv4-as-a-Service (IPv4aaS) for ISPs with an IPv6-only access and/or core network. All these technologies have their advantages and disadvantages, and depending on existing topology, skills, strategy and other preferences, one of these technologies may be the most appropriate solution for a network operator.

This document examines the scalability of the five most prominent IPv4aaS technologies (464XLAT, Dual Stack Lite, Lightweight 4over6, MAP-E, MAP-T) considering two aspects: (1) how their performance scales up with the number of CPU cores, (2) how their performance degrades, when the number of concurrent sessions is increased until hardware limit is reached.

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 16 April 2024.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction

IETF has standardized several IPv6 transition technologies [LEN2019] and occupied a neutral position trusting the selection of the most appropriate ones to the market. [RFC9313] provides a comprehensive comparative analysis of the five most prominent IPv4aaS technologies to assist operators with this problem. This document adds one more detail: measurement data regarding the scalability of the examined IPv4aaS technologies.

This draft is a collection of various measurement results. Some measurements with the iptables stateful NAT44 implementation and the Jool stateful NAT64 implementation were performed directly for this draft. Some other results published in open access research papers are added gradually.

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. Scalability of iptables

2.1. Introduction to iptables

Netfilter [NETFLTR] is a widely used firewall, NAT and packet mangling framework for Linux. It is often called as "iptables" after the name of its user space command line tool. From our point of view, iptables is used as a stateful NAT44 solution. (Also called as NAPT: Network Address and Port Translation.) It is a free and open source software under the GPLv2 license.

This document deals with iptables for multiple considerations:

* To provide a reference for the scalability of various stateful NAT64 implementations. (We use it to prove that a stateful NATxy solution does not need to exhibit a poor scalability.)

* To provide IPv6 operators with a basis for comparison if is it worth using an IPv4aaS solution over Carrier-grade NAT.

* To prove the scalability of iptables, when iptables is used as a part of the CE of MAP-T (see later).

2.2. Measurement Method

[RFC8219] has defined a benchmarking methodology for IPv6 transition technologies. [I-D.ietf-bmwg-benchmarking-stateful] has amended it by addressing how to benchmark stateful NATxy gateways using pseudorandom port numbers recommended by [RFC4814]. It has defined measurement procedures for maximum connection establishment rate, connection tear down rate and connection table capacity measurement, plus it reused the classic measurement procedures like throughput, latency, frame loss rate, etc. from [RFC8219]. Besides the new metrics, we used throughput to characterize the performance of the examined system.

The scalability of iptables is examined in two aspects:

* How its performance scales up with the number of CPU cores?

* How its performance degrades, when the number of concurrent sessions is increased?

```
                 +-------------------------------------+
       10.0.0.2 |Initiator                    Responder| 198.19.0.2
  +-------------|                Tester                |<------------+
  | private IPv4|                         [state table]| public IPv4 |
                 +-------------------------------------+              |
  |                                                                   |
                 +-------------------------------------+              |
  |    10.0.0.1 |                 DUT:                | 198.19.0.1   |
  +------------>|        Sateful NAT44 gateway        |-------------+
    private IPv4|     [connection tracking table]     | public IPv4
                 +-------------------------------------+
```

Figure 1: Test setup for benchmarking stateful NAT44 gateways

The test setup in Figure 1 was followed. The two devices, the Tester and the DUT (Device Under Test), were both Dell PowerEdge R430 servers having two 2.1GHz Intel Xeon E5-2683 v4 CPUs, 384GB 2400MHz DDR4 RAM and Intel 10G dual port X540 network adapters. The NICs of the servers were interconnected by direct cables, and the CPU clock frequecy was set to fixed 2.1 GHz on both servers. They had Debian 9.13 Linux operating system with 4.9.0-16-amd64 kernel. The measurements were performed by siitperf [LEN2021] using the "stateful" branch (latest commit Aug. 16, 2021). The DPDK version was 16.11.11-1+deb9u2. The version of iptables was 1.6.0.

The ratio of number of connections in the connection tracking table and the value of the hashsize parameter of iptables significantly influences its performance. Although the default setting is hashsize=nf_conntrack_max/8, we have usually set hashsize=nf_conntrack_max to increase the performance of iptables, which was crucial, when high number of connections were used, because then the execution time of the tests was dominated by the preliminary phase, when several hundereds of millions connections had to be established. (In some cases, we had to use different settings due to memory limitations. The tables presenting the results always contain these parameters.)

The size of the port number pool is an important parameter of the bechmarking method for stateful NATxy gateways, thus it is also given for all tests.

2.3. Performance scale up against the number of CPU cores

To examine how the performance of iptables scales up with the number of CPU cores, the number of active CPU cores was set to 1, 2, 4, 8, 16 using the "maxcpus=" kernel parameter.

The number of connections was always 4,000,000 using 4,000 different source port numbers and 1,000 different destination port numbers. Both the connection tracking table size and the hash table size was set to 2^23.

The error of the binary search was chosen to be lower than 0.1% of the expected results. The experiments were executed 10 times.

Besides the connection establishment rate and the throughput of iptables, also the throughput of the IPv4 packet forwarding of the Linux kernel was measured to provide a basis for comparison.

The results are presented in Figure 2. The unit for the maximum connection establishment rate is 1,000 connections per second. The unit for throughput is 1,000 packets per second (measured with bidirectional traffic, and the number of all packets per second is displayed).

num. CPU cores	1	2	4	8	16
src ports	4,000	4,000	4,000	4,000	4,000
dst ports	1,000	1,000	1,000	1,000	1,000
num. conn.	4,000,000	4,000,000	4,000,000	4,000,000	4,000,000
conntrack t. s.	2^23	2^23	2^23	2^23	2^23
hash table size	2^23	2^23	2^23	2^23	2^23
c.t.s/num.conn.	2.097	2.097	2.097	2.097	2.097
num. experiments	10	10	10	10	10
error	100	100	100	1,000	1,000
cps median	223.5	371.1	708.7	1,341	2,383
cps min	221.6	367.7	701.7	1,325	2,304
cps max	226.7	375.9	723.6	1,376	2,417
cps rel. scale up	1	0.830	0.793	0.750	0.666
throughput median	414.9	742.3	1,379	2,336	4,557
throughput min	413.9	740.6	1,373	2,311	4,436
throughput max	416.1	746.9	1,395	2,361	4,627
tp. rel. scale up	1	0.895	0.831	0.704	0.686
IPv4 packet forwarding (using the same port number ranges)					
error	200	500	1,000	1,000	1,000
throughput median	910.9	1,523	3,016	5,920	11,561
throughput min	874.8	1,485	2,951	5,811	10,998
throughput max	914.3	1,534	3,037	5,940	11,627
tp. rel. scale up	1	0.836	0.828	0.812	0.793
throughput ratio (%)	45.5	48.8	45.7	39.5	39.4

Figure 2: Scale up of iptables against the number of CPU cores (Please refer to the next figure for the explanation of the abbreviations.)

abbreviation	explanation
num. CPU cores	number of CPU cores
src ports	size of the source port number range
dst ports	size of the destination port number range
num. conn.	number of connections = src ports * dst ports
conntrack t. s.	size of the connection tracking table of the DUT
hash table size	size of the hash table of the DUT
c.t.s/num.conn.	conntrack table size / number of connections
num. experiments	number of experiments
error	the difference between the upper and the lower bound of the binary search when it stops
cps (median/min/max)	maximum connection establishment rate (median, minimum, maximum)
cps rel. scale up	the relative scale up of the maximum connection establishment rate against the number of CPU cores
tp. rel. scale up	the relative scale up of the throughput
throughput ratio (%)	the ratio of the throughput of iptables and the throughput of IPv4 packet forwarding

Figure 3: Explanation of the abbreviations for the scale up of iptables against the number of CPU cores

Whereas the throughput of IPv4 packet forwarding scaled up from 0.91Mpps to 11.56Mpps showing a relative scale up of 0.793, the throughput of iptables scaled up from 414.9kpps to 4,557kpps showing a relative scale up of 0.686 (and the relative scale up of the maximum connection establishment rate is only 0.666). On the one hand, this is the price of the stateful operation. On the other hand, this result is quite good compared to the scale-up results of NSD (a high performance authoritative DNS server) presented in Table 9 of [LEN2020], which is only 0.52. (1,454,661/177,432=8.2-fold performance using 16 cores.) And DNS is not a stateful technology.

2.4. Performance degradation caused by the number of sessions

To examine how the performance of iptables degrades with the number connections in the connection tracking table, the number of connections was increased fourfold by doubling the size of both the source port number range and the destination port number range. Both the connection tracking table size and the hash table size was also increased four fold. However, we reached the limits of the hardware at 400,000,000 connections: we could not set the size of the hash table to 2^29 but only to 2^28. The same value was used at 800,000,000 connections too, when the number of connections was only

doubled, because 1.6 billion connections would not fit into the memory.

The error of the binary search was chosen to be lower than 0.1% of the expected results. The experiments were executed 10 times (except for the very long lasting measurements with 800,000,000 connections).

The results are presented in Figure 4. The unit for the maximum connection establishment rate is 1,000,000 connections per second. The unit for throughput is 1,000,000 packets per second (measured with bidirectional traffic, and the number of all packets per second is displayed).

num. conn.	1.56M	6.25M	25M	100M	400M	800M
src ports	2,500	5,000	10,000	20,000	40,000	40,000
dst ports	625	1,250	2,500	5,000	10,000	20,000
conntrack t. s.	2^21	2^23	2^25	2^27	2^29	2^30
hash table size	2^21	2^23	2^25	2^27	2^28	2^28
num. exp.	10	10	10	10	10	5
error	1,000	1,000	1,000	1,000	1,000	1,000
n.c./h.t.s.	0.745	0.745	0.745	0.745	1.490	2.980
cps median	2.406	2.279	2.278	2.237	2.013	1.405
cps min	2.358	2.226	2.226	2.124	1.983	1.390
cps max	2.505	2.315	2.317	2.290	2.050	1.440
throughput med.	5.326	4.369	4.510	4.516	4.244	3.689
throughput min	5.217	4.240	3.994	4.373	4.217	3.670
throughput max	5.533	4.408	4.572	4.537	4.342	3.709

Figure 4: Performance of iptables against the number of sessions

The performance of iptables shows degradation at 6.25M connections compared to 1.56M connections very likely due to the exhaustion of the L3 cache of the CPU of the DUT. Then the performance of iptables is fearly constant up to 100M connections. A small performance decrease can be observed at 400M connections due to the lower hash table size. A more significant performance decrease can be observed at 800M connections. It is caused by two factors:

* on average, about 3 connections were hashed to the same place

* non NUMA local memory was also used.

We note that the CPU has 2 NUMA nodes, cores 0, 2, ... 14 belong to NUMA node 0, and cores 1, 3, ... 15 belong to NUMA node 1. The maximum memory consumption with 400,000,000 connections was below 150GB, thus it could be stored in NUMA local memory.

Therefore, we have pointed out important limitations of the stateful NAT44 technology:

* there is a performance decrease, when approaching hardware limits

* there is a hardware limit, beyond which the system cannot handle the connections at all (e.g. 1600M connections would not fit into the memory).

Therefore, we can conclude that, on the one hand, a well tailored hashing may guarantee an excellent scale-up of stateful NAT44 regarding the number of connections in a wide range, however, on the other hand, stateful operation has its limits resulting both in performance decrease, when approaching hardware limits and also in inability to handle more sessions, when reaching the memory limits.

2.5. Connection tear down rate

[I-D.ietf-bmwg-benchmarking-stateful] has defined connection tear down rate measurement as an aggregate measurement, that is, N number of connections are loaded into the connection tracking table of the DUT and then the entire content of the connection tracking table is deleted, and its deletion time is measured (T). Finally, the connection tear down rate is computed as: N/T.)

We have observed that the deletion of an empty connection tracking table of iptables my take a significant amount of time depending on its size. Therefore, we made our measurements more accurate by subtracting the deletion time of the empty connection tracking table from that of the filled one, thus we got the time spent with the deleting of the connections.

The same setup and parameters were used as in Section 2.4 and the experiments were executed 10 times (except for the long lasting measurements with 800,000,000 connections).

The results are presented in Figure 5.

num. conn.	1.56M	6.35M	25M	100M	400M	800M
src ports	2,500	5,000	10,000	20,000	40,000	40,000
dst ports	625	1,250	2,500	5,000	10,000	20,000
conntrack t. s.	2^21	2^23	2^25	2^27	2^29	2^30
hash table size	2^21	2^23	2^25	2^27	2^28	2^28
num. exp.	10	10	10	10	10	5
n.c./h.t.s.	0.745	0.745	0.745	0.745	1.490	2.980
full contr. del med	4.33	18.05	74.47	305.33	1,178.3	2,263.1
full contr. del min	4.25	17.93	72.04	299.06	1,164.0	2,259.6
full contr. del max	4.38	18.20	75.13	310.05	1,188.3	2,275.2
empty contr. del med	0.55	1.28	4.17	15.74	31.2	31.2
empty contr. del min	0.55	1.26	4.16	15.73	31.1	31.1
empty contr. del max	0.57	1.29	4.22	15.79	31.2	31.2
conn. deletion time	3.78	16.77	70.30	289.59	1,147.2	2,232.0
conn. tear d. rate	413,360	372,689	355,619	345,316	348,690	358,429

Figure 5: Connection tear down rate of iptables against the number of connections

The connection tear down performance of iptables shows significant degradation at 6.25M connections compared to 1.56M connections very likely due to the exhaustion of the L3 cache of the CPU of the DUT. Then it shows only a minor degradation up to 100M connections. A small performance increase can be observed at 400M connections due to the relatively lower hash table size. A more visible performance decrease can be observed at 800M connections. It is likely caused by keeping the hash table size constant and doubling the number of connections. The same thing that caused performance degradation of the maximum connection establishment rate and throughput, made now the deletion of the connections faster and thus caused an increase of the connection tear down rate.

We note that according to the recommended settings of iptables, 8 connections are hashed to each place of the hash table on average, but we wilfully used much smaller number (0.745 whenever it was possible) to increase the maximum connection estabilishment rate and thus to speed up experimenting. However, finally this choice significantly slowed down our experiments due to the very low connection tear down rate.

2.6. Connection tracking table capacity

[I-D.ietf-bmwg-benchmarking-stateful] has defined connection tracking table capacity measurement using the following quantities:

* C0: initial safe value for the size of the connection tracking table (the connection tracking table can surely store C0 entries)

* R0: safe connection establisment rate for C0 connections (measured initially)

* CS: safe value for the size of the connection tracking table during the current measurement (taken from the previous iteration step)

* RS: safe connection establisment rate for CS connections during the current measurement (measured during the previous iteration)

* CT: the currently tested size of the connection tracking table during the exponential search; also used in the final binary search.

* RT: the currently used connection establisment rate for testing with CT number of connections during the exponential search

* alpha: safety factor to prevent that connection validation fails due to sending the validation frames at a too high rate

* beta: factor to express a too high drop of the connection establishment rate during the exponential search

* gamma: factor to express a too high drop of the connection establishment rate during the final binary search

First, the order of magnitude of the size of the connection tracking table is determined by an exponential search. When it stops, then the C capacity of the connection tracking table is between CS and CT=2*CS.

Then the C size of the connection tracking table is determined by a binary search within E error.

Measurements were performed with the following parameters: hashsize=nf_conntrack_max=2**22=4,194,304; R0=1,000,000; E=1, alpha=1.0; beta=0.2; gamma=0.4. The measurements were performed 10 times to see the stability of the results.

	C0	R0	CS	RS	CT	C
median	1,000,000	2,562,500	4,000,000	2,250,945	8,000,000	4,194,301
min	1,000,000	2,437,500	4,000,000	2,139,953	8,000,000	4,194,300
max	1,000,000	2,687,500	4,000,000	2,327,269	8,000,000	4,194,302

Figure 6: Connection tracking table capacity measurement resultss for iptables (actual size: 4,194,304)

The results are presented in Figure 6. The exponential search finished at its third step (CS=4,000,000 and CT=8,000,000). And the result of the final binary search was always very close to 4,194,304.

3. Scalability of Jool

3.1. Introduction to Jool

Jool [JOOLMX] is an open source SIIT and stateful NAT64 implementation for Linux. Since its version 4.2 it also supports MAP-T. It has been developed by NIC Mexico in cooperation with ITESM (Monterrey Institute of Technology and Higher Education). Its source code is released under GPLv2 license.

3.2. Measurement Method

The same methodology was used as in Section 2, but now the test setup in Figure 7 was followed. The same Tester and DUT devices were used as before, but the operating system of the DUT was updated to Debian 10.11 with 4.19.0-18-amd64 kernel to meet the requirement of the jool-tools package. The version of Jool was 4.1.6. (The most mature version of Jool at the date of starting the measurements, Relase Date: 2021-12-10.)

```
                 +-------------------------------------+
     2001:2::2 |Initiator                    Responder| 198.19.0.2
   +-------------|                Tester                |<------------+
   | IPv6 address|                         [state table]| IPv4 address|
   |             +-------------------------------------+             |
   |                                                                 |
   |             +-------------------------------------+             |
   |   2001:2::1 |                 DUT:                | 198.19.0.1  |
   +------------>|        Sateful NAT64 gateway        |-------------+
     IPv6 address|     [connection tracking table]     | IPv4 address
                 +-------------------------------------+
```

Figure 7: Test setup for benchmarking stateful NAT64 gateways

Unlike with iptables, we did not find any way to tune the hashsize or any other parameters of Jool.

3.3. Performance scale up against the number of CPU cores

The number of connections was always 1,000,000 using 2,000 different source port numbers and 500 different destination port numbers.

The error of the binary search was chosen to be lower than 0.1% of the expected results. The experiments were executed 10 times.

The results are presented in Figure 8. The unit for the maximum connection establishment rate is 1,000 connections per second. The unit for throughput is 1,000 packets per second (measured with bidirectional traffic, and the number of all packets per second is displayed).

num. CPU cores	1	2	4	8	16
src ports	2,000	2,000	2,000	2,000	2,000
dst ports	500	500	500	500	500
num. conn.	1,000,000	1,000,000	1,000,000	1,000,000	1,000,000
num. experiments	10	10	10	10	10
error	100	100	100	100	100
cps median	228.6	358.5	537.4	569.9	602.6
cps min	226.5	352.5	530.7	562.0	593.7
cps max	230.5	362.4	543	578.3	609.7
cps rel. scale up	1	0.784	0.588	0.312	0.165
throughput median	251.8	405.7	582.4	604.1	612.3
throughput min	249.8	402.9	573.2	587.3	599.8
throughput max	253.3	409.6	585.7	607.2	616.6
tp. rel. scale up	1	0.806	0.578	0.300	0.152

Figure 8: Scale up of Jool against the number of CPU cores

Both the maximum connection establishment rate and the throughput scaled up poorly with the number of active CPU cores. The increase of the performance was very low above 4 CPU cores.

3.4. Performance degradation caused by the number of sessions

To examine how the performance of Jool degrades with the number connections, the number of connections was increased fourfold by doubling the size of both the source port number range and the destination port number range. We did not reach the limits of the hardware regarding the number of connections, because unlike iptables, Jool worked also with 1.6 billion connections.

The error of the binary search was chosen to be lower than 0.1% of the expected results and the experiments were executed 10 times (except for the very long lasting measurements with 800 million and 1.6 billion connections to save execution time).

The results are presented in Figure 9. The unit for the maximum connection establishment rate is 1,000 connections per second. The unit for throughput is 1,000 packets per second (measured with bidirectional traffic, and the number of all packets per second is displayed).

num. conn.	1.56M	6.35M	25M	100M	400M	1600M
src ports	2,500	5,000	10,000	20,000	40,000	40,000
dst ports	625	1,250	2,500	5,000	10,000	40,000
num. exp.	10	10	10	10	5	5
error	100	100	100	100	1,000	1,000
cps median	480.2	394.8	328.6	273.0	243.0	232.0
cps min	468.6	392.7	324.9	269.4	243.0	230.5
cps max	484.9	397.4	331.3	280.6	244.5	233.6
throughput med.	511.5	423.9	350.0	286.5	257.8	198.4
throughput min	509.2	420.3	348.2	284.2	257.8	195.3
throughput max	513.1	428.3	352.5	290.8	260.9	201.6

Figure 9: Performance of Jool against the number of sessions

The performance of Jool shows degradation at the entire range of the number of connections. We did not analyze the root cause of the degradation yet. And we are not aware of the implementation of its connection tracking table. We also plan to check the memory consumption of Jool, what is definitely lower that that of iptables.

3.5. Connection tear down rate

Basically, the same measurement method was used as in Section 2.5, however having no parameter of Jool to tune, only a single measurement series was performed to determine the deletion time of the empty connection tracking table. The median, minimum and maximum values of the 10 measurements were 0.46s, 0.42s and 0.50s respectively.

The same setup and parameters were used as in Section 3.4 and the experiments were executed 10 times (except for the long lasting measurements with 800,000,000 connections).

The results are presented in Figure 10. The unit for the connection tear down rate is 1,000,000 connections per second.

num. conn.	1.56M	6.35M	25M	100M	400M	1600M
src ports	2,500	5,000	10,000	20,000	40,000	40,000
dst ports	625	1,250	2,500	5,000	10,000	40,000
num. exp.	10	10	10	10	10	5
full contr. del med	0.87	2.05	7.84	36.38	126.09	474.68
full contr. del min	0.80	2.02	7.80	36.27	125.84	473.20
full contr. del max	0.91	2.09	7.94	36.80	127.54	481.38
empty contr. del med	0.46	0.46	0.46	0.46	0.46	0.46
conn. deletion time	0.41	1.59	7.38	35.92	125.63	474.22
conn. t. d. r. (M)	3.811	3.931	3.388	2.784	3.184	3.374

Figure 10: Connection tear down rate of Jool against the number of connections

The connection tear down performance of Jool is excellent at any number of connections. It is about and order of magnitude higher that its connection establishment rate and than the connection tear down rate of iptables. (A slight degradation can be observed at 100M connections.)

3.6. Validation of connection establishment

The measurement of connection establishment rate with validation was performed using different values for the "alpha" parameter.

The results are presented in Figure 11. It is well visible that alpha values 0.8 and 0.6 cause significant decrease of the validated rate, therefore, they are unsuitable. Values 0.5 and 0.25 make no difference compared to the unvalidated connection establishment rate. (The less than 1,000 cps increase of the median is deliberately a measurement error.)

alpha	0.8	0.6	0.5	0.25	no validation
num. conn.	4,000,000	4,000,000	4,000,000	4,000,000	4,000,000
src ports	4,000	4,000	4,000	4,000	4,000
dst ports	1,000	1,000	1,000	1,000	1,000
num. exp.	10	10	10	10	10
error	100	100	100	100	100
cps median	323,534	429,491	479,296	479,199	478,417
cps min	322,948	426,464	473,339	474,120	474,902
cps max	325,097	431,542	483,690	483,299	484,667

Figure 11: Connection establishment rate rate of Jool against the alpha parameter

4. Scalability of OpenBSD PF

4.1. Introduction to OpenBSD PF

PF [PFBOOK] is the packet filtering solution of OpenBSD. PF supports statetul NAT64 since May 1, 2012, the release of OpenBSD 5.1. Its connection tracking system uses red-black tree, which is a kind of balanced tree that can ensure O(log(n)) search time, where n is the number of elements in the tree.

It is noted that the main focus of the OpenBSD project is security and not performance.

4.2. Measurement Method

The same methodology was used as in Section 3, and the same test setup shown in Figure 7 was followed. The same Tester and DUT devices were used as before, but the operating system of the DUT was OpenBSD 7.1 with GENERIC.MP#465 amd64 kernel. All the details of the measurements can be found in [LEN2023]. It is also explained in this open access paper, why there was no point in measuring the scalability of OpenBSD PF against the number of CPU cores.

Similarly to Jool, no parameters of OpenBSD PF were tuned.

4.3. Performance degradation caused by the number of sessions

To examine how the performance of OpenBSD PF degrades with the number connections, the number of connections was increased tenfold by keeping the size of source port number range a fixed value (40,000) and increasing the size of the destination port number range tenfold. This time we only aimed to make a rough assessment of the scalability of the performance of OpenBSD PF againts the number of connections, and we did not aim to reach the limits of the hardware regarding the number of connections.

The error of the binary search was chosen to be lower than 0.1% of the expected results and the experiments were executed 10 times.

The results are presented in Figure 12. The unit for the maximum connection establishment rate is 1 connection per second. The unit for throughput is 1 frame per second (measured with bidirectional traffic, and the number of all packets per second is displayed).

The results show a moderate performance degradation, to a similar extent as Jool. However the results themselves are significantly lower than that of Jool.

Number of connections	400,000	4,000,000	40,000,000
Source port numbers	40,000	40,000	40,000
Destination port numbers	10	100	1,000
Error (cps)	50	40	50
Median (cps)	120,214	85,039	74,022
Minimum (cps)	118,701	84,882	73,680
Maximum (cps)	122,411	85,351	74,266
Median / previous median	-	0.71	0.87
Error (fps)	200	80	100
Median (fps)	237,304	198,828	173,338
Minimum (fps)	236,912	198,046	172,946
Maximum (fps)	250,584	199,452	174,120
Median / previous median	-	0.84	0.87

Figure 12: Performance of OpenBSD PF against the number of sessions

4.4. Connection tear down rate

Basically, the same measurement method was used as in Section 3.5.

The same setup and parameters were used as in Section 4.3 and the experiments were executed 10 times.

The results are presented in Figure 13. The unit for the connection tear down rate is 1 connection per second.

Number of connections	400,000	4,000,000	40,000,000
Source port numbers	40,000	40,000	40,000
Destination port numbers	10	100	1,000
Filled table deletion time med. (s)	1.45	11.56	94.20
Filled table deletion time min. (s)	1.36	11.03	91.73
Filled table deletion time max. (s)	1.78	13.81	118.52
Empty table deletion time med. (s)	0.37	0.37	0.37
Empty table deletion time min. (s)	0.36	0.36	0.36
Empty table deletion time max. (s)	0.37	0.37	0.37
Connections deletion time (s)	1.08	11.19	93.83
Connection tear down rate (cps)	370,370	357,622	426,303

Figure 13: Connection tear down rate of OpenBSD PF against the number of connections

The connection tear down performance of OpenBSD PF is moderate at any number of connections. And the results are about an order of magnitude lower than that of Jool.

5. Scalability Comparison of the Jool Implementation of the 464XLAT and of the MAP-T IPv4aaS Technologies using DNS Traffic

This section summarizes the essence of our meeasurements for the scalability comparison of the Jool implementation of the 464XLAT and of the MAP-T IPv4aaS technologies presented in [LEN2022]. The measurements did not comply with the requirements of [RFC8219], but the results give an insight into the scalability of the Jool implementation of the two technologies. Because of the limitations of the measurement method, only their scalability with the number of CPU cores was examined.

5.1. 464XLAT Scalability Measurements and Results

The measurement setup for the scalability analysis of 464XLAT is shown in Figure 14.

```
+--------+                        --                        +--------+
|  CLAT  |-----------------------(VLAN)----------------------|  PLAT  |
| (p098) |2001:db8:100::1/64    (3248)   2001:db8:100::2/64| (p099) |
+---+----+                        --                        +---+----+
    |192.168.0.1/24                                 203.0.113.2/24|
    |                                                             |
   -+-                                                           -+-
  (VLAN)                                                        (VLAN)
  (3247)                                                        (4249)
   -+-                                                           -+-
    |                                                             |
    |192.168.0.4/24                                203.0.113.56/24|
+---+----+                                                      +---+----+
|Tester A| Running dns64perf++            Running Knot DNS |Tester B|
| (p097) | measurement program                 to generate | (p100) |
+--------+                                     reply packets +--------+
```

Figure 14: Test setup for benchmarking 464XLAT using DNS traffic

The p097 - p100 devices were the same type of Dell PowerEdge R430 servers residing at NICT StarBED as before, and Debian Linux 10.11 operating system with kernel version 4.19 was used. Both CLAT and PLAT was implemented by Jool [JOOLMX]. To faciliate a fair comparison with MAP-T, Jool version 4.2.0-rc2 was used as Jool supports MAP-T from its 4.2 version.

The measurement traffic was generated by the dns64perf++ program, which sent DNS qeries for "AAAA" records and counted the valid replies. The reverse traffic was generated by the "Knot DNS"

authoritative DNS server. As not direct cable connections, but a switch with VLANS was used, we allowed 0.01% packet loss during the binary search to find the highest supported rate. To measure how the performance of the 464XLAT test system scaled up with the number of CPU cores, the number of CPU cores of the CLAT and PLAT devices were set to: 1, 2, 4, 8, and 16, whereas the number of CPU cores of the A and B part of the Tester was always 32.

The number of connections was always 1600. (Dns64perf++ used 16 thread pairs, and the number of source port numbers per sending thread was set to 100. The destination port number was always 53, the well-known port number for DNS.) The reason behind this low number of connections was to use the same number of connections as with MAP-T, which had the limit of 2048 source port numbers per subscriber.

The results are presented in Figure 15. It is well visible that the scalability of the system is moderate, the addition of the last 8 cores results in only 4% performance increase.

num. CPU cores	1	2	4	8	16
Median (qps)	165,403	236,869	425,021	510,155	530,064
Minimum (qps)	163,280	236,185	420,311	499,999	529,881
Maximum (qps)	167,041	237,553	425,782	510,321	530,222
current/previous	-	1.43	1.79	1.20	1.04

Figure 15: The number of resolved queries per second as a function of the number of active CPU cores of the CLAT and PLAT devices of the 464XLAT test system

5.2. MAP-T Scalability Measurements and Results

The measurement setup for the scalability analysis of MAP-T is shown in Figure 16.

The configuration of the test system and the measurement method was the same as with 464XLAT.

The results are presented in Figure 17. It is well visible that the scalability of the system is much better now.

All further details can be found in our open access paper [LEN2022].

```
   +--------+                        --                        +--------+
   |   CE   |----------------------(VLAN)----------------------|   BR   |
   | (p098) |2001:db8:6::11b/64    (3248)     2001:db8:6::1/64| (p099) |
   +---+----+                        --                        +---+----+
       |192.168.0.1/24                                203.0.113.2/24|
       |                                                            |
      -+-                                                          -+-
     (VLAN)                                                       (VLAN)
     (3247)                                                       (4249)
      -+-                                                          -+-
       |                                                            |
       |192.168.0.4/24                                203.0.113.56/24|
   +---+----+                                                   +---+----+
   |Tester A| Running dns64perf++            Running Knot DNS |Tester B|
   | (p097) | measurement program                 to generate | (p100) |
   +--------+                                    reply packets +--------+
```

Figure 16: Test setup for benchmarking MAP-T using DNS traffic

num. CPU cores	1	2	4	8	16
Median (qps)	143,498	250,858	475,051	835,180	1,183,615
Minimum (qps)	140,624	245,311	470,311	812,499	1,179,686
Maximum (qps)	145,314	251,953	475,818	837,295	1,184,387
current/previous	-	1.75	1.89	1.76	1.42

Figure 17: The number of resolved queries per second as a function of the number of active CPU cores of the CE and BR devices of the MAP-T test system

6. Acknowledgements

The measurements were carried out by remotely using the resources of NICT StarBED, 2-12 Asahidai, Nomi-City, Ishikawa 923-1211, Japan. The author would like to thank Shuuhei Takimoto for the possibility to use StarBED, as well as to Satoru Gonno and Makoto Yoshida for their help and advice in StarBED usage related issues.

The author would like to thank Ole Troan for his comments on the v6ops mailing list, while the scalalability measurements of iptables were intended to be a part of the draft later published as [RFC9313].

7. IANA Considerations

This document does not make any request to IANA.

8. Security Considerations

TBD.

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>.

[RFC4814] Newman, D. and T. Player, "Hash and Stuffing: Overlooked Factors in Network Device Benchmarking", RFC 4814, DOI 10.17487/RFC4814, March 2007, <https://www.rfc-editor.org/info/rfc4814>.

[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>.

[RFC9313] Lencse, G., Palet Martinez, J., Howard, L., Patterson, R., and I. Farrer, "Pros and Cons of IPv6 Transition Technologies for IPv4-as-a-Service (IPv4aaS)", RFC 9313, DOI 10.17487/RFC9313, October 2022, <https://www.rfc-editor.org/info/rfc9313>.

9.2. Informative References

[I-D.ietf-bmwg-benchmarking-stateful] Lencse, G. and K. Shima, "Benchmarking Methodology for Stateful NATxy Gateways using RFC 4814 Pseudorandom Port Numbers", Work in Progress, Internet-Draft, draft-ietf-bmwg-benchmarking-stateful-04, 12 September 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-bmwg-benchmarking-stateful-04>.

[JOOLMX] NIC Mexico, "Jool: SIIT and NAT64", The home page of Jool, 2022, <https://jool.mx/>.

[LEN2019] Lencse, G. and Y. Kadobayashi, "Comprehensive Survey of IPv6 Transition Technologies: A Subjective Classification for Security Analysis", IEICE Transactions on Communications, vol. E102-B, no.10, pp. 2021-2035., DOI: 10.1587/transcom.2018EBR0002, 1 October 2019, <http://www.hit.bme.hu/~lencse/publications/e102-b_10_2021.pdf>.

[LEN2020] Lencse, G., "Benchmarking Authoritative DNS Servers", IEEE Access, vol. 8. pp. 130224-130238, DOI: 10.1109/ACCESS.2020.3009141, July 2020, <https://ieeexplore.ieee.org/document/9139929>.

[LEN2021] Lencse, G., "Design and Implementation of a Software Tester for Benchmarking Stateless NAT64 Gateways", IEICE Transactions on Communications, DOI: 10.1587/transcom.2019EBN0010, 1 February 2021, <http://www.hit.bme.hu/~lencse/publications/IEICE-2020-siitperf-revised.pdf>.

[LEN2022] Lencse, G. and N. Nagy, "Towards the scalability comparison of the Jool implementation of the 464XLAT and of the MAP-T IPv4aaS technologies", International Journal of Communication Systems, DOI: 10.1002/dac.5354, 22 September 2022, <https://onlinelibrary.wiley.com/doi/10.1002/dac.5354>.

[LEN2023] Lencse, G., Shima, K., and K. Cho, "Benchmarking methodology for stateful NAT64 gateways", Computer Communications, DOI: 10.1016/j.comcom.2023.08.009, 1 October 2023, <https://www.sciencedirect.com/science/article/pii/S0140366423002931>.

[NETFLTR] The Netfilter’s webmasters, "Netfilter: Firewalling, NAT, and packet mangling for Linux", The netfilter.org project home page, 2021, <https://www.netfilter.org/>.

[PFBOOK] Hansteen, P. N. M., "The Book of PF: A No-Nonsense Guide to the OpenBSD Firewall", 3rd ed., San Francisco, No Starch Press, ISBN: 978-1-59327-589-1, 2014, <https://nostarch.com/pf3>.

Appendix A. Change Log

A.1. 00

Initial version: scale up of iptables.

A.2. 01

Added the scale up of Jool.

A.3. 02

Connection tear down rate measurements of iptables and Jool.

A.4. 03

Added: introductions to iptables and Jool, connection tracking table capacity measurement of iptables and connection validation measurement of Jool.

A.5. 04

Added: scalability comparison of the Jool implementation of the 464XLAT and of the MAP-T IPv4aaS technologies using DNS traffic.

A.6. 05

Added: scalability of OpenBSD PF.

Author's Address

Gábor Lencse
Széchenyi István University
Gyr
Egyetem tér 1.
H-9026
Hungary
Email: lencse@sze.hu

BMWG
Internet-Draft
Intended status: Informational
Expires: January 7, 2023

G. Fioccola
E. Vasilenko
P. Volpato
Huawei Technologies
L. Contreras
Telefonica

July 6, 2022

Benchmarking Methodology for MPLS Segment Routing
draft-vfv-bmwg-srmpls-bench-meth-02

Abstract

This document defines a methodology for benchmarking Segment Routing (SR) performance for Segment Routing over MPLS (SR-MPLS). It builds upon [RFC2544], [RFC5695] and [RFC8402].

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], RFC 8174 [RFC8174].

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 7, 2023.

Copyright Notice

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

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

Segment Routing (SR), defined in [RFC8402], leverages the source routing paradigm. The headend node steers a packet through an SR Policy [I-D.ietf-spring-segment-routing-policy], instantiated as an ordered list of segments. A segment, referred to by its Segment Identifier (SID), can have a semantic local to an SR node or global within an SR domain. SR supports per-flow explicit routing while

maintaining per-flow state only at the ingress nodes to the SR domain.

However, there is no standard method defined to compare and contrast the foundational SR packet forwarding capabilities of network devices. This document aims to extend the efforts of [RFC1242] and [RFC2544] to SR network.

The SR architecture can be instantiated on two data-plane: SR over MPLS (SR-MPLS) and SR over IPv6 (SRv6). This document is limited to SR-MPLS.

It is expected that future documents may cover the benchmarking of SR-MPLS applications such as Layer 3 VPN (L3VPN) [RFC4364], EVPN [RFC7432], Fast ReRoute [I-D.ietf-rtgwg-segment-routing-ti-lfa], etc.

SR can be directly applied to the Multiprotocol Label Switching (MPLS) architecture with no change to the forwarding plane [RFC8660]. A segment is encoded as an MPLS label. An SR Policy is instantiated as a stack of labels.

SR-MPLS involves 3 types of forwarding plane operations:

- o PUSH consists of the insertion of one or more segments on top of the incoming packet It is the outer label of the SR-MPLS label stack.

- o NEXT consists of the inspection of the next segment. The active segment is completed and the next segment is activated. It is a POP of the top label in SR-MPLS.

- o CONTINUE happens when the active segment is not completed; hence, it remains active. It is a SWAP of the top label in SR-MPLS.

SR list for PUSH operation is typically constructed by SR Policy in ingress node, see [I-D.ietf-spring-segment-routing-policy].

[RFC5695] describes a methodology specific to the benchmarking of MPLS forwarding devices, by considering the most common MPLS packet forwarding scenarios and corresponding performance measurements.

The purpose of this document is to describe a methodology specific to the benchmarking of Segment Routing. The methodology described is a complement for [RFC5695].

2. SR-MPLS Forwarding

In MPLS, a Prefix-SID is allocated in the form of an MPLS label. For SR-MPLS, Segment Routing does not require any change to the MPLS forwarding plane. An SR Policy is instantiated through the MPLS Label Stack: the Segment IDs (SIDs) of a Segment List are inserted as MPLS Labels. The classical forwarding functions available for MPLS networks allow implementing the SR operations.

The operations applied by the SR-MPLS forwarding plane are PUSH, NEXT, and CONTINUE.

The PUSH operation corresponds to the Label Push function, according to the MPLS label pushing rules specified in [RFC3032]. It consists of pushing one or more MPLS labels on top of an incoming packet then sending it out of a particular physical interface or virtual interface towards a particular next hop.

The NEXT operation corresponds to the Label Pop function, which consists of removing the topmost label. The action before and/or after the popping depends on the instruction associated with the active SID on the received packet prior to the popping. It is equivalent to Penultimate Hop Popping (PHP).

The CONTINUE operation corresponds to the Label Swap function, according to the MPLS label-swapping rules in [RFC3031]. It consists of associating an incoming label with an outgoing interface and outgoing label and forwarding the packet to the outgoing interface. It is equivalent to Ultimate Hop Popping (UHP).

The encapsulation of an IP packet into an SR-MPLS packet is performed at the edge of an SR-MPLS domain, reusing the MPLS Forwarding Equivalent Class (FEC) concept. A Forwarding Equivalent Class (FEC) can be associated with an SR Policy ([I-D.ietf-spring-segment-routing-policy]). When pushing labels onto a packet’s label stack, the Time-to-Live (TTL) field and the Traffic Class (TC) field of each label stack entry must also be set.

All SR nodes in the SR domain use an IGP signaling extension to advertise their own prefix SIDs. After receiving the advertised prefix SIDs, each SR node calculates the prefix SIDs to the advertisers. The prefix SID advertisement can be an absolute value advertisement or an index value advertisement. In this regard, the mapping of Segments to MPLS Labels (SIDs) is an important process in the SR-MPLS data plane. Each router can advertise its own available label space to be used for Global Segments called Segment Routing Global Block (SRGB) and an identical range of labels (SRGB) should be used in all routers in order to simplify services and operations. In

the SR domain Global Segments can be identified by an index, which has to be re-mapped into a label, or by an absolute value. This is relevant for the nodes that perform the NEXT operation to the segments, because the label for the next segments needs to be crafted accordingly.

[I-D.ietf-spring-segment-routing-policy] specifies the concepts of SR Policy and steering into an SR Policy. The header of a packet steered in an SR Policy is augmented with the ordered list of segments associated with that SR Policy. SR Policy state is instantiated only on the headend node, that steers a flow into an SR Policy. Indeed intermediate and endpoint nodes do not require any state to be maintained. SR Policies can be instantiated on the headend dynamically and on demand basis. SR policy may be installed by PCEP [RFC8664], BGP [I-D.ietf-idr-segment-routing-te-policy], or via manual configuration on the router. PCEP signaling can be the case of a controller based deployment. For all these reasons, SR Policies scale better than traditional TE mechanisms.

3. Test Methodology

3.1. Test Setup

The test setup in general is compliant with section 6 of [RFC2544] but augmented by the methodology specified in section 4 of [RFC5695] using many ports. In fact, it is needed to test the packet forwarding engine that may have different performance based on the number of ports served. The Device Under Test (DUT) may have oversubscribed ports, then traffic for such ports should be proportionally decreased according to the specific DUT oversubscription ratio. All ports served by a particular packet forwarding engine should be loaded in reverse proportion to the claimed oversubscription ratio. Tests SHOULD be done with bidirectional traffic that better reflects the real environment for SR-MPLS nodes. It is OPTIONAL to choose non-equal proportion for upstream and downstream traffic for some specific aggregation nodes.

The recommended topology for SR-MPLS Forwarding Benchmarking should be the same as MPLS and it is described in section 4 of [RFC5695]. Port numbers involved in the tests and their oversubscription ratio MUST be reported. In general, MPLS labels at the bottom of the stack may be used to encode services (L2/L3 VPNs) but it is out of the scope of this document. This document is benchmarking only "source routing". Hence, SIDs represent only prefix and adjacency segments.

Segment Routing may also be implemented as a software network function in an NFV Infrastructure and, in this case, additional considerations should be done. [RFC9004] updates the procedures of

the test to measure the Back-to-Back Frames since their characterization is relevant in software-packet processing. Also, [ETSI-GR-NFV-TST-007] describes test guidelines for NFV capabilities that require interactions between the components implementing NFV functionality.

3.2. Label Distribution Support

As specified in [RFC8402], in the context of an IGP-based distributed control plane, two topological segments are defined: the IGP-Adjacency segment and the IGP-Prefix segment; while, in the context of a BGP-based distributed control plane, two topological segments are defined: the BGP peer segment and the BGP Prefix segment.

It is RECOMMENDED that the DUT and test tool support at least one option for SID stack construction:

- IS-IS Extensions for Segment Routing [RFC8667]
- OSPF Extensions for Segment Routing [RFC8665]
- Segment Routing Prefix Segment Identifier Extensions for BGP [RFC8669]
- Segment Routing Policy Architecture [I-D.ietf-spring-segment-routing-policy].

It is RECOMMENDED that at least one routing protocol (OSPF or ISIS or BGP) should be used for the construction of the simplest stack of 1 SID. It is RECOMMENDED that SR policy should be used for the construction of a stack with 2 SIDs. It is possible to test longer SID lists if there is an interest.

It is RECOMMENDED that the top SID on the list (outer label) should be an adjacency type to emulate the traffic engineering scenario. In all cases, SID stack configuration SHOULD happen before packet forwarding would be started. Control plane convergence speed is not the subject of the present tests.

The label distribution method and SR policy construction method used MUST be reported according to Section 4.

3.3. Frame Formats and Sizes

The tests for SR-MPLS will use the Frame characteristics similarly to section 4.1.5 of [RFC5695], except the need for a bigger MTU to accommodate many MPLS labels.

It is to be noted that [RFC5695] requires exactly a single entry in the MPLS label stack in an MPLS packet that is not enough to simulate typical SR SID list. MPLS label values used in any test case MUST be outside the reserved label value (0-15) unless stated otherwise. The number of entries in the label stack MUST be reported.

According to section 4.1.4.2 of [RFC5695], the payload is RECOMMENDED to have an IP packet (IPv6 or IPv4 with UDP or TCP) to better represent the real environment.

It is assumed that the test would be for Ethernet media only. Other media is possible (see section 4.1.5.2 of [RFC5695] for the POS example). Recommended frame sizes are presented below. Any other frame sized may be added if suspected of abnormal behavior. For example, some architectures may allocate buffer memory in big fixed chunks that may drop performance if frame sizes are chosen just a few octet more than the fixed chunk size (the second chunk would have a very low memory utilization).

Recommended frame sizes are the following:

- Ethernet Minimal: 64+n*4

- DUT Minimal Wire Speed: 128-256 (it depends on the range recommended in the DUT specification)

- Ethernet Typical: 1518+n*4

- DUT Maximum Wire Speed: 9000

Note that n*4 octets are added in the previous calculations to accommodate MPLS labels needed for respective tests. The typical frame size values are listed above for the DUT minimal and maximum wire speed, but they can be modified according to the DUT characteristics. Indeed, the minimum wire speed frame size can be considered based on the DUT specification but, in some cases, many tests may be needed in the search for the real minimum wire speed frame size. VLAN tag may additionally increase the frame size. VLAN tag tests are OPTIONAL.

3.4. Protocol Addresses

IANA reserved an IPv6 address block 2001:0200::/48 ([RFC4773]) for use with IPv6 benchmark testing and block 198.18.0.0/15 ([RFC3330]) for IPv4 benchmark testing. The type of infrastructure protocol (IPv6 vs IPv4) that should be used for IGP and BGP in the tests should be chosen according to the test purpose and requirements.

3.5. Trial Duration

The test portion of each trial SHOULD be at least 10 seconds longer than the hold time for the respective protocol configuration to verify that the DUT can maintain a stable control plane when the data-forwarding plane is under stress. IGP protocols typically have a shorter hold time, some BGP default configuration may be up to 180 seconds. It is needed to check the default hold time of the DUT for the respective protocol used.

3.6. Traffic Verification

Traffic verification is following section 4.1.8 of [RFC5695].

4. Reporting Format

There are a few parameters that need to be changed in section 5 of [RFC5695] for SR MPLS tests. New parameters that MUST be reported are:

- o Port numbers involved in the tests and their respective oversubscription ratio.
- o Upstream/downstream traffic proportion (equal bidirectional or some other split).
- o SR-MPLS Forwarding Operations (PUSH/ NEXT/ CONTINUE).
- o Number of Segments considered in the MPLS Label Stack and the type of SIDs used (Global/Local).
- o SR Policy construction method (PCEP, BGP, manual configuration).
- o Type of the payload (IPv6/IPv4, UDP/TCP).

Some parameters MAY be changed:

- o Label Distribution protocol and IGP are the same in the context of SR MPLS. Hence, it is called "label distribution".
- o Port media type may be reported only one time for all tests if only Ethernet media would be tested

5. SR-MPLS Forwarding Benchmarking Tests

In general, tests are compliant with [RFC2544] but the important correction discussed in section 6 of [RFC5695] is applied: ports chosen for every test MUST stress all ports served by one forwarding

engine. It is better to check the DUT specification for the relationship between ports and the forwarding engine to minimize the number of ports involved. But it is possible to understand the worst case by looking at the throughput and latency from the trial tests. If any doubt exists about how full is the offered load for the forwarding engine then it is better to stress all ports of the line card or all ports for the whole router with a centralized forwarding engine. Partial load on forwarding engine would show optimistic results. Controllable traffic distribution between many ports (as specified in section 4 of [RFC5695]) would need separate SID announcements for separate ports. The search for No-Drop Rate (NDR) should be done for every test as explained in section 6 of [RFC5695].

5.1. Throughput

This section contains the description of the tests that are related to the characterization of a DUT's SR-MPLS traffic forwarding throughput.

The list of segments for SR-MPLS is represented as a stack of MPLS labels. There are three distinct operations to be tested: PUSH, NEXT and CONTINUE. These correspond to the three forwarding operations of an MPLS packet: PUSH (or LSP Ingress), POP (or LSP Egress), or SWAP. It is separately discussed only for throughput tests as an example.

5.1.1. Throughput for SR-MPLS PUSH

Objective: To obtain the DUT's Throughput during the PUSH forwarding operation. It is similar to label Push or LSP Ingress forwarding operation, as per section 6.1.1 of [RFC5695].

Procedure: Similar to [RFC5695] with potential extension to test SID list longer than 1 SID (2 are recommended, many are possible).

Reporting Format: Similar to [RFC5695] with the additional parameters specified in Section 4.

5.1.2. Throughput for SR-MPLS NEXT

Objective: To obtain the DUT's Throughput during the NEXT forwarding operation. It is equivalent to MPLS Label Pop or Penultimate Hop Popping (PHP), as per section 6.1.3 of [RFC5695].

Procedure: Similar to [RFC5695] with potential extension to test SID list longer than 1 SID (2 are recommended, many are possible).

Reporting Format: Similar to [RFC5695] with the additional parameters specified in Section 4.

5.1.3. Throughput for SR-MPLS CONTINUE

Objective: To obtain the DUT's Throughput during the CONTINUE forwarding operation. It is equivalent to MPLS Label Swap or Ultimate Hop Popping (UHP), as per section 6.1.2 of [RFC5695]. Non-reserved MPLS label values MUST be used.

Procedure: Similar to [RFC5695] with potential extension to test SID list longer than 1 SID (2 are recommended, many are possible).

Reporting Format: Similar to [RFC5695] with the additional parameters specified in Section 4.

5.2. Latency

Objective: To determine the latency as defined in section 6.2 of [RFC5695] for each of the SR-MPLS forwarding operations (PUSH, NEXT, CONTINUE).

Procedure: Similar to [RFC5695] with potential extension to test SID list longer than 1 SID (2 are recommended, many are possible).

Reporting Format: Similar to [RFC5695] with the additional parameters specified in Section 4.

5.3. Frame Loss

Objective: To determine the frame-loss rate (as defined in section 6.3 of [RFC5695]) for each of the SR-MPLS forwarding operations of a DUT throughout the entire range of input data rates and frame sizes.

Procedure: Similar to [RFC5695] with potential extension to test SID list longer than 1 SID (2 are recommended, many are possible).

Reporting Format: Similar to [RFC5695] with the additional parameters specified in Section 4.

5.4. System Recovery

Objective: To characterize the speed at which a DUT recovers from an overload condition for each of the SR-MPLS forwarding operations.

Procedure: Similar to section 6.4 of [RFC5695].

Reporting Format: Similar to [RFC5695] with the additional parameters specified in Section 4.

5.5. Reset

Objective: To characterize the speed at which a DUT recovers from a device or software reset for each of the SR-MPLS forwarding operations.

Procedure: Similar to section 6.5 of [RFC5695].

Reporting Format: Similar to [RFC5695] with the additional parameters specified in Section 4.

It is OPTIONAL to extend the Reset tests according to [RFC6201] in order to reset only part of the DUT: only line card reset, only process reset (for example ISIS), only one routing engine reset in the configuration with routing engine redundancy, full power interruption, partial power interruption, etc.

6. Security Considerations

Benchmarking methodologies are limited to technology characterization in a laboratory environment, with dedicated address space and constraints. 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 production networks. The benchmarking network topology is 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.

There are no specific security considerations within the scope of this document.

7. IANA Considerations

This document has no IANA actions.

8. Acknowledgements

The authors would like to thank Al Morton for the precious comments and suggestions.

9. References

9.1. Normative References

[RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <https://www.rfc-editor.org/info/rfc1242>.

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

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

[RFC3330] IANA, "Special-Use IPv4 Addresses", RFC 3330, DOI 10.17487/RFC3330, September 2002, <https://www.rfc-editor.org/info/rfc3330>.

[RFC4773] Huston, G., "Administration of the IANA Special Purpose IPv6 Address Block", RFC 4773, DOI 10.17487/RFC4773, December 2006, <https://www.rfc-editor.org/info/rfc4773>.

[RFC5695] Akhter, A., Asati, R., and C. Pignataro, "MPLS Forwarding Benchmarking Methodology for IP Flows", RFC 5695, DOI 10.17487/RFC5695, November 2009, <https://www.rfc-editor.org/info/rfc5695>.

[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>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

[RFC8660] Bashandy, A., Ed., Filsfils, C., Ed., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with the MPLS Data Plane", RFC 8660, DOI 10.17487/RFC8660, December 2019, <https://www.rfc-editor.org/info/rfc8660>.

9.2. Informative References

[ETSI-GR-NFV-TST-007]
ETSI, "ETSI GR NFV-TST 007: Network Functions Virtualisation (NFV) Release 3; Testing; Guidelines on Interoperability Testing for MANO", 2020, <https://www.etsi.org/deliver/etsi_gr/NFV-TST/001_099/007/02.06.01_60/gr_nfv-tst007v020601p.pdf>.

[I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Talaulikar, K., Mattes, P., Jain, D., and S. Lin, "Advertising Segment Routing Policies in BGP", draft-ietf-idr-segment-routing-te-policy-18 (work in progress), June 2022.

[I-D.ietf-rtgwg-segment-routing-ti-lfa]
Litkowski, S., Bashandy, A., Filsfils, C., Francois, P., Decraene, B., and D. Voyer, "Topology Independent Fast Reroute using Segment Routing", draft-ietf-rtgwg-segment-routing-ti-lfa-08 (work in progress), January 2022.

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-22 (work in progress), March 2022.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <https://www.rfc-editor.org/info/rfc3031>.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <https://www.rfc-editor.org/info/rfc3032>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <https://www.rfc-editor.org/info/rfc4364>.

[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>.

[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>.

[RFC8664] Sivabalan, S., Filsfils, C., Tantsura, J., Henderickx, W., and J. Hardwick, "Path Computation Element Communication Protocol (PCEP) Extensions for Segment Routing", RFC 8664, DOI 10.17487/RFC8664, December 2019, <https://www.rfc-editor.org/info/rfc8664>.

[RFC8665] Psenak, P., Ed., Previdi, S., Ed., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", RFC 8665, DOI 10.17487/RFC8665, December 2019, <https://www.rfc-editor.org/info/rfc8665>.

[RFC8667] Previdi, S., Ed., Ginsberg, L., Ed., Filsfils, C., Bashandy, A., Gredler, H., and B. Decraene, "IS-IS Extensions for Segment Routing", RFC 8667, DOI 10.17487/RFC8667, December 2019, <https://www.rfc-editor.org/info/rfc8667>.

[RFC8669] Previdi, S., Filsfils, C., Lindem, A., Ed., Sreekantiah, A., and H. Gredler, "Segment Routing Prefix Segment Identifier Extensions for BGP", RFC 8669, DOI 10.17487/RFC8669, December 2019, <https://www.rfc-editor.org/info/rfc8669>.

[RFC9004] Morton, A., "Updates for the Back-to-Back Frame Benchmark in RFC 2544", RFC 9004, DOI 10.17487/RFC9004, May 2021, <https://www.rfc-editor.org/info/rfc9004>.

Authors' Addresses

Giuseppe Fioccola
Huawei Technologies
Riesstrasse, 25
Munich 80992
Germany

Email: giuseppe.fioccola@huawei.com

Eduard Vasilenko
Huawei Technologies
17/4 Krylatskaya str.
Moscow 121614
Russia

Email: vasilenko.eduard@huawei.com

Paolo Volpato
Huawei Technologies
Via Lorenteggio, 240
Milan 20147
Italy

Email: paolo.volpato@huawei.com

Luis Miguel Contreras Murillo
Telefonica
Spain

Email: luismiguel.contrerasmurillo@telefonica.com

BMWG
Internet-Draft
Intended status: Informational
Expires: 25 April 2024

G. Fioccola
E. Vasilenko
P. Volpato
Huawei Technologies
L. Contreras
Telefonica
B. Decraene
Orange

23 October 2023

Benchmarking Methodology for MPLS Segment Routing
draft-vfv-bmwg-srmpls-bench-meth-08

Abstract

This document defines a methodology for benchmarking Segment Routing (SR) performance for Segment Routing over MPLS (SR-MPLS). It builds upon RFC 2544, RFC 5695 and RFC 8402.

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 25 April 2024.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction

Segment Routing (SR), defined in [RFC8402], leverages the source routing paradigm. The headend node steers a packet through an SR Policy [RFC9256], instantiated as an ordered list of segments. A segment, referred to by its Segment Identifier (SID), can have a semantic local to an SR node or global within an SR domain. SR supports per-class explicit routing while maintaining per-class state only at the ingress nodes to the SR domain.

However, there is no standard method defined to compare and contrast the foundational SR packet forwarding capabilities of network devices. This document aims to extend the efforts of [RFC1242], [RFC2544] and [RFC5695] to SR network.

The SR architecture can be instantiated on two data-plane: SR over MPLS (SR-MPLS) and SR over IPv6 (SRv6). This document is limited to SR-MPLS.

SR can be directly applied to the Multiprotocol Label Switching (MPLS) architecture [RFC8660]. A segment is encoded as an MPLS label. An SR Policy is instantiated as a stack of labels.

The MPLS label stack in scope of this document has a minimum of two entries, e.g. two SIDs. But it is RECOMMENDED that the tests described in the next sections can be applied to label stacks with more than two SIDs. The reason for having a minimum of two SIDs, hence two labels, is to simulate a SID list, e.g. to simulate the explicit steering of a packet flow through different paths/nodes.

It is expected that future documents may cover the benchmarking of SR-MPLS applications such as Layer 3 VPN (L3VPN) [RFC4364], EVPN [RFC7432], Fast ReRoute [I-D.ietf-rtgwg-segment-routing-ti-lfa], etc.

SR-MPLS involves 3 types of forwarding plane operations (PUSH/ NEXT/ CONTINUE) as further described in Section 2. SR list for PUSH operation is typically constructed by the source node with a SR Policy, see [RFC9256].

[RFC5695] describes a methodology specific to the benchmarking of MPLS forwarding devices, by considering the most common MPLS packet forwarding scenarios and corresponding performance measurements.

The purpose of this document is to describe a methodology specific to the benchmarking of Segment Routing. The methodology described is a complement for [RFC5695].

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], RFC 8174 [RFC8174].

2. SR-MPLS Forwarding

SR leverages the source routing paradigm. In MPLS, the ordered list of segments is encoded as a stack of MPLS labels. An SR Policy is instantiated through the MPLS Label Stack: the Segment IDs (SIDs) of a Segment List are inserted as MPLS Labels. Hence SR-MPLS Segment List typically contains more labels. The classical forwarding functions available for MPLS networks allow implementing the SR operations.

The operations applied by the SR-MPLS forwarding plane are PUSH, NEXT, and CONTINUE.

The PUSH operation corresponds to the Label Push function, according to the MPLS label pushing rules specified in [RFC3032]. It consists of pushing one or more MPLS labels on top of an incoming packet then sending it out of a particular physical or virtual interface towards a particular next hop.

The NEXT operation corresponds to the Label Pop function, which consists of removing the topmost label. The action before and/or after the popping depends on the instruction associated with the active SID on the received packet prior to the popping. It is equivalent to Penultimate Hop Popping (PHP).

The CONTINUE operation corresponds to the Label Swap function, according to the MPLS label-swapping rules in [RFC3031]. It consists of associating an incoming label with an outgoing interface and outgoing label and forwarding the packet to the outgoing interface. It is equivalent to Ultimate Hop Popping (UHP).

The encapsulation of an IP packet into an SR-MPLS packet is performed at the edge of an SR-MPLS domain, reusing the MPLS Forwarding Equivalent Class (FEC) concept. A Forwarding Equivalent Class (FEC) can be associated with an SR Policy ([RFC9256]). When pushing labels onto a packet’s label stack, the Time-to-Live (TTL) field and the Traffic Class (TC) field of each label stack entry must also be set.

All SR nodes in the SR domain use an IGP signaling extension to advertise their own prefix SIDs. After receiving the advertised prefix SIDs, each SR node calculates the prefix SIDs to the

advertisers. The prefix SID advertisement can be an absolute value advertisement or an index value advertisement. In this regard, the mapping of Segments to MPLS Labels (SIDs) is an important process in the SR-MPLS data plane. Each router can advertise its own available label space to be used for Global Segments called Segment Routing Global Block (SRGB) and an identical range of labels (SRGB) should be used in all routers in order to simplify services and operations. In the SR domain Global Segments can be identified by an index, which has to be re-mapped into a label, or by an absolute value. This is relevant for the nodes that perform the NEXT operation to the segments, because the label for the next segments needs to be crafted accordingly.

[RFC9256] specifies the concepts of SR Policy and steering into an SR Policy. The header of a packet steered in an SR Policy is augmented with the ordered list of segments associated with that SR Policy. SR Policy state is instantiated only on the headend node, which steers a flow into an SR Policy. Intermediate and endpoint nodes do not require any state to be maintained. SR Policies can be instantiated on the headend dynamically and on demand basis. SR policy may be installed by PCEP [RFC8664], BGP [I-D.ietf-idr-segment-routing-te-policy], or via manual configuration on the router. PCEP and BGP signaling of SR Policies can be the case of a controller-based deployment.

3. Test Methodology

3.1. Test Setup

The test setup in general is compliant with section 6 of [RFC2544] but augmented by the methodology specified in section 4 of [RFC5695] using many interfaces. It is needed to test the packet forwarding engine that may have different performance based on the number of interfaces served. The Device Under Test (DUT) may have oversubscribed interfaces, then traffic for such interfaces should be proportionally decreased according to the specific DUT oversubscription ratio. All interfaces served by a particular packet forwarding engine should be loaded in reverse proportion to the claimed oversubscription ratio. Tests SHOULD be done with bidirectional traffic that better reflects the real environment for SR-MPLS nodes. It is OPTIONAL to choose non-equal proportion for upstream and downstream traffic for some specific aggregation nodes.

The RECOMMENDED topology for SR-MPLS Forwarding Benchmarking should be the same used for MPLS benchmarking, as described in section 4 of [RFC5695]. A simplified view is reported below for reference.

```
                    +----------+
           +--------|          |<---------+
           | +------|  Tester  |<-------+ |
           | | +----|          |<-----+ | |
           | | |    +----------+      | | |
           | | |                      | | |
           | | |     +--------+       | | |
           | | +----->|        |------+ | |
           | +------>|  DUT   |--------+ |
           +--------->|        |----------+
                      +--------+
```

Figure 1: Test environment for SR-MPLS Forwarding Benchmarking

Differently from [RFC5695], this document prefers the use of the term "interface" instead of "port" as an interface may be either virtual or physical.

Interface numbers involved in the tests and their oversubscription ratio MUST be reported. SIDs represent only prefix and adjacency segments. In general, MPLS labels at the bottom of the stack may be used to encode services (L2/L3 VPNs) but it is out of the scope of this document.

Segment Routing may also be implemented as a software network function in an NFV Infrastructure and, in this case, additional considerations should be done. [ETSI-GR-NFV-TST-007] describes test guidelines for NFV capabilities that require interactions between the components implementing NFV functionality.

Special capabilities SHOULD NOT exist in the DUT/SUT specifically for benchmarking purposes.

3.2. Label Distribution Support

As specified in [RFC8402], in the context of an IGP-based distributed control plane, two topological segments are defined: the IGP-Adjacency segment and the IGP-Prefix segment; while, in the context of a BGP-based distributed control plane, two topological segments are defined: the BGP peer segment and the BGP Prefix segment.

It is RECOMMENDED that the DUT and test tool support at least one option for SID stack construction:

* IS-IS Extensions for Segment Routing [RFC8667]

* OSPF Extensions for Segment Routing [RFC8665]

* Segment Routing Prefix Segment Identifier Extensions for BGP [RFC8669]

* Segment Routing Policy Architecture [RFC9256].

A routing protocol (OSPF or ISIS or BGP) SHOULD be used for the construction of the simplest stack of 1 SID. It is RECOMMENDED that SR policy should be used for the construction of a stack with 2 SIDs. It is possible to test longer SID lists if there is an interest.

It is RECOMMENDED that the top SID on the list (outer label) should be an adjacency type to emulate the traffic engineering scenario. In all cases, SID stack configuration SHOULD happen before packet forwarding would be started. Control plane convergence speed is not the subject of the present tests.

The label distribution method and SR policy construction method used MUST be reported according to Section 4.

3.3. Frame Formats and Sizes

The tests for SR-MPLS will use Frame characteristics similarly to section 4.1.5 of [RFC5695], except the need for a bigger MTU to accommodate many MPLS labels.

It is to be noted that [RFC5695] requires exactly a single entry in the MPLS label stack. For the scope of this document, this is not enough to simulate a typical SR-MPLS SID list. MPLS label values used in any test case MUST be outside the reserved label value (0-15). The number of entries in the label stack MUST be reported.

According to section 4.1.4.2 of [RFC5695], the payload is RECOMMENDED to have an IP packet (IPv6 or IPv4 with UDP or TCP) to better represent the real environment.

It is assumed that the test would be for Ethernet media only. Other media is possible (see section 4.1.5.2 of [RFC5695] for the POS example). Some layer 2 technologies (like POS/PPP) have bit- or byte- stuffing then [RFC4814] may help to calculate real performance more accurately or else 1-2% error is expected. The most popular layer 2 technology for SR is Ethernet, it does not have stuffing.

Section 4.1.5 of [RFC5695] observes that the presence of an MPLS label has the effect of increasing the maximum frame payload size [RFC3032] so that "the resulting Layer 2 frame is Z octets more than the conventional maximum frame payload size, where Z = 4 x number of entries in the label stack".

As already stated, the SID list in scope of this document is composed of two SIDs. Accordingly, it is RECOMMENDED to set the media MTU value to the effective maximum frame payload size [RFC3032], which equals 2 * Z octets + conventional maximum frame payload size. It is expected that such a change in the media MTU value only impacts the effective Maximum Frame Payload Size for MPLS packets, but not for IP packets. The depth of the label stack is set to Z = 4 x 2 = 8 octets.

The resulting Ethernet frame structure is depicted in the next figure.

```
<-------------------------72-1526B-------------------------->
<---18B---><--4B--><--4B--><-----------46-1500-------------->
+---------+-------+-------+---------+----------------------+
|         | MPLS  | MPLS  |         |         |            |
| Layer 2 | Label | Label | Layer 3 | Layer 4 | High layers |
+---------+-------+-------+---------+----------------------+
```

Figure 2: Ethernet Frame Structure

RECOMMENDED frame sizes are presented below. Any other frame sizes may be added if suspected of abnormal behavior. For example, some architectures may allocate buffer memory in big fixed chunks that may drop performance if frame sizes are chosen just a few octets more than the fixed chunk size (the second chunk would have a very low memory utilization).

RECOMMENDED frame sizes are the following:

* Ethernet Minimal: 64+n*4 (n=2)

* DUT Minimal Wire Speed: typically 128-256 (it depends on the DUT specification)

* Ethernet Typical: 1518+n*4 (n=2)

* DUT Maximum: 9000 (or any claimed maximum)

where n is the number of labels (SID Depth).

Note that n*4 octets are added in the previous calculations to accommodate MPLS labels needed for respective tests. The typical frame size values are listed above for the DUT minimal wire speed and maximum, but they can be modified according to the DUT characteristics. The minimum wire speed frame size can be considered based on the DUT specification but, in some cases, many tests may be needed in the search for the real minimum wire speed frame size. VLAN tag may additionally increase the frame size. VLAN tag tests are OPTIONAL.

3.4. Protocol Addresses

IANA reserved an IPv6 address block 2001:0002::/48 ([RFC4773]) for use with IPv6 benchmark testing and block 198.18.0.0/15 ([RFC3330]) for IPv4 benchmark testing. The type of infrastructure protocol (IPv6 vs IPv4) that should be used for IGP and BGP in the tests should be chosen according to the test purpose and requirements.

As it is discussed in section 3.1, there is a need to load the whole forwarding engine (on all interfaces). [RFC4814] discusses the importance to have many flows with address randomization for acceptable hash-based load balancing that is implemented in all forwarding engines. In the context of this document, it may also be relevant for SIDs, because SIDs may be used for hash to choose the next link (depending on DUT default or desired configuration). It is important to check what exactly is used for the hash load balancing algorithm on the DUT to keep these numbers sufficiently random and at volume. It is very often that IP addresses and transport protocol ports are used instead of SIDs.

3.5. Trial Duration

The test portion of each trial must take into account the respective protocol configuration. IGP protocols typically have a shorter hold time, while some BGP default configurations may be up to 180 seconds. It is needed to check the default hold time of the DUT for the respective protocol used.

In general, the test portion of each trial SHOULD be no less than 250 seconds, which is a reasonable value based on common hold time values. But a test can also adapt to the real setup and select a different value if default configuration has been changed. The test portion of each trial can be chosen at least 10 seconds longer than the hold time to verify that the DUT can maintain a stable control plane when the data-forwarding plane is under stress.

3.6. Traffic Verification

Traffic verification is following section 10 of [RFC2544] and section 4.1.8 of [RFC5695].

As stated in section 10 of [RFC2544], "the test equipment SHOULD discard any frames received during a test run that are not actual forwarded test frames. For example, keep-alive and routing update frames SHOULD NOT be included in the count of received frames. In all cases, sent traffic MUST be accounted for, whether it was received on the wrong interface, the correct interface, or not received at all. In all cases, the test equipment SHOULD verify the length of the received frames and check that they match the expected length.

Preferably, the test equipment SHOULD include sequence numbers (or signature) in the transmitted frames and check for these numbers on the received frames. If this is done, the reported results SHOULD include in addition to the number of frames dropped, the number of frames that were received out of order, the number of duplicate frames received and the number of gaps in the received frame numbering sequence".

Many test tools may, by default, only verify that they have received the embedded signature on the receive side. However, some SR-MPLS tests assumes headers modifications. All packets SHOULD be checked of the correct headers values on the receiving side.

In addition, section 4.1.8 of [RFC5695] requires that "the presence or absence of the MPLS label stack, every field value inside the label stack, if present, ethertype (0x8847 or 0x8848 versus 0x0800 or 0x86DD), frame sequencing, and frame check sequence (FCS) MUST be verified in the received frame". This "to verify that the packets received by the test tool carry the expected MPLS label".

3.7. Buffer tests

Back-to-back frame test was initially discussed in section 26.4 [RFC2544] and later improved in [RFC9004] which is considered the comprehensive reference for back-to-back frame test. Modern forwarding engines are typically flexible in the buffer distribution between different interfaces. Hence, like for all other benchmarking tests, it is important to stress the forwarding engine on all interfaces. It should be necessary to perform throughput tests first because only frame sizes that stress DUT below wire-speed can be used for back-to-back tests. Buffers would be filled with the rate equal to the difference between the theoretical maximum frame rate (wire-speed) and DUT measured throughput for the respective frame size.

The test time could be much shorter than recommended in [RFC9004] because typical SR DUT is hardware-based with claimed buffers between 30ms to 100ms. It is better to consult with the vendor to find a good starting search point. If DUT is software-based then [RFC9004] recommendation for 2-30 seconds is applied.

Queuing SHOULD NOT have weighted random early detection (WRED) or any other mechanism that may start dropping packets before the buffer is filled. Queuing SHOULD be configured for the tail drop which is, typically, a non-default configuration. Back-to-back frame test is rather complex and expensive (50 runs for every frame size). Hence, it is OPTIONAL for SR-MPLS.

4. Reporting Format

There are a few parameters that need to be changed in section 5 of [RFC5695] for SR MPLS tests.

Reporting parameter preserved from [RFC5695]:

* Throughput in bytes per second and frames per second

* Frame sizes in Octets (see Section 3.3)

* Interface speed (10/50/100/400/800/etc GE)

* Interface encapsulation (Ethernet or Ethernet VLAN)

* Interface media type (probably Ethernet)

Parameters changed from [RFC5695]:

* SR-MPLS Forwarding Operations (PUSH/ NEXT/ CONTINUE).

* Label Distribution protocol and IGP are the same in the context of SR-MPLS. Hence, it is called "label distribution".

New parameters that MUST be reported are:

* Interface numbers involved for ingress and egress in the tests and their respective oversubscription ratio.

* Upstream/downstream traffic proportion (equal bidirectional or some other split).

* Number of Segments considered in the MPLS Label Stack and the type of SIDs used (Global/Local).

* SR Policy construction method (PCEP, BGP, manual configuration).

* Type of the payload (IPv6/IPv4, UDP/TCP).

* Time to recover from the overload state

* Time to recover from the reset state and reset type (particular module in reset)

* Tested buffers size in frames with respective frame size (for the optional back-to-back test); it is possible to record calculated buffer time for wire-speed throughput in milliseconds.

Some parameters may be the same for all tests (like Media type or Ethernet encapsulation) then it may be reported one time.

5. SR-MPLS Forwarding Benchmarking Tests

In general, tests are compliant with [RFC2544] but the important correction discussed in section 6 of [RFC5695] is applied: interfaces chosen for every test MUST stress all interfaces served by one forwarding engine. It is better to check the DUT specification for the relationship between interfaces and the forwarding engine to minimize the number of interfaces involved. But it is possible to understand the worst case by looking at the throughput and latency from the trial tests. If any doubt exists about how full is the offered load for the forwarding engine then it is better to stress all interfaces of the line card or all interfaces for the whole router with a centralized forwarding engine. A partial load on the forwarding engine would show optimistic results. Controllable traffic distribution between many interfaces (as specified in section 4 of [RFC5695]) would need separate SID announcements for separate interfaces.

The performance of modern packet forwarding engines may be huge that may need to involve many testers to sufficiently load the DUT as presented on figure 3. Then results correlation and recalculation of the real performance would be an additional burden.

```
                 +----------+
        +-------|  Tester1 |<-------+
        | +-----|          |<-----+ |
        | |     +----------+      | |
        | |                       | |
        | |      +--------+       | |
        | +----->|        |-------+ |
        +------->|  DUT   |---------+
        +------->|        |---------+
        | +----->|        |-------+ |
        | |      +--------+       | |
        | |                       | |
        | |     +----------+      | |
        | +-----|  Tester2 |<-----+ |
        +-------|          |<-------+
                +----------+
```

Figure 3: Many testers

As specified in section 6 of [RFC5695], the traffic is sent from test tool Tx interface(s) to the DUT at a constant load for a fixed-time interval, and is received from the DUT on test tool Rx interface(s). If any frame loss is detected, then a new iteration is needed where the offered load is decreased and the sender will transmit again. An iterative search algorithm MUST be used to determine the maximum offered frame rate with a zero frame loss (No-Drop Rate - NDR). Each iteration should involve varying the offered load of the traffic, while keeping the other parameters (test duration, number of interfaces, number of addresses, frame size, etc.) constant, until the maximum rate at which none of the offered frames are dropped is determined.

The test can be repeated with a varying number of Segments pushed on ingress in order to measure the resulting maximum number. It can also be tested the maximum number of Segments that are correctly load-balanced in transit by only changing the Nth label in the stack and detect when load-balancing fails.

Therefore, the two main parameters that can be evaluated are:

Maximum offered frame rate,

Maximum number of Segments that can be pushed and hashed by the SR node for load-balancing.

5.1. Throughput

This section contains a description of the tests that are related to the characterization of a DUT's SR-MPLS traffic forwarding throughput.

The list of segments for SR-MPLS is represented as a stack of MPLS labels. There are three distinct operations to be tested: PUSH, NEXT and CONTINUE. These correspond to the three forwarding operations of an MPLS packet: PUSH (or LSP Ingress), POP (or LSP Egress), or SWAP. It is separately discussed only for throughput tests as an example.

5.1.1. Throughput for SR-MPLS PUSH

Objective: To obtain the DUT's Throughput during the PUSH forwarding operation. It is similar to label Push or LSP Ingress forwarding operation, as per section 6.1.1 of [RFC5695] and section 26.1 of [RFC2544].

Procedure: Similar to [RFC5695] extended to test SID list longer than 1 SID (more than 2 are RECOMMENDED). SID list can be from 1 to N SIDs. N could be specified a priori or measured as part of the test. The test tool must advertise and learn the IP prefix(es) and SIDs on respective sides, as per Section 3.4, and must use one option for SID stack construction, as per Section 3.2, on its receive and transmit interfaces towards the DUT.

Reporting Format: A table with all parameters specified in Section 4.

5.1.2. Throughput for SR-MPLS NEXT

Objective: To obtain the DUT's Throughput during the NEXT forwarding operation. It is equivalent to MPLS Label Pop or Penultimate Hop Popping (PHP), as per section 6.1.3 of [RFC5695] and section 26.1 of [RFC2544].

Procedure: Similar to [RFC5695] extended to test SID list longer than 1 SID (more than 2 are RECOMMENDED). SID list can be from 1 to N SIDs. N could be specified a priori or measured as part of the test. The test tool must advertise and learn the IP prefix(es) and SIDs on respective sides, as per Section 3.4, and must use one option for SID stack construction, as per Section 3.2, on its receive and transmit interfaces towards the DUT.

Reporting Format: A table with all parameters specified in Section 4.

5.1.3. Throughput for SR-MPLS CONTINUE

Objective: To obtain the DUT's Throughput during the CONTINUE forwarding operation. It is equivalent to MPLS Label Swap or Ultimate Hop Popping (UHP), as per section 6.1.2 of [RFC5695] and section 26.1 of [RFC2544]. Non-reserved MPLS label values MUST be used.

Procedure: Similar to [RFC5695] extended to test SID list longer than 1 SID (more than 2 are RECOMMENDED). SID list can be from 1 to N SIDs. N could be specified a priori or measured as part of the test. The test tool must advertise and learn the IP prefix(es) and SIDs on respective sides, as per Section 3.4, and must use one option for SID stack construction, as per Section 3.2, on its receive and transmit interfaces towards the DUT.

Reporting Format: A table with all parameters specified in Section 4.

5.2. Buffers size

Back-to-back frame test is OPTIONAL and SHOULD be performed only after throughput tests because it SHOULD use only frame sizes that DUT is not capable to forward wire-speed, as explained in Section 3.7.

Objective: To determine the buffer size as defined in section 6 of [RFC9004] for each of the SR-MPLS forwarding operations.

Procedure: Should be inherited from [RFC9004] with 2 SIDs RECOMMENDED (many SIDs are possible). Despite the simple general idea for filling the buffer until tail drop, [RFC9004] has many details for procedure, precautions, and calculations that would be too lengthy to copy here.

Reporting Format: A table with all parameters specified in Section 4.

5.3. Latency

Objective: To determine the latency as defined in section 6.2 of [RFC5695] and section 26.2 of [RFC2544] for each of the SR-MPLS forwarding operations (PUSH, NEXT, CONTINUE). It is RECOMMENDED to test all three test types discussed in Section 5.1.

Procedure: Similar to Section 5.1. It is OPTIONAL to improve the procedure according to section 7.2 of [RFC8219] with calculations for typical and worst-case latency.

Reporting Format: A table with all parameters specified in Section 4.

5.4. Frame Loss

Objective: To determine the frame-loss rate (as defined in section 6.3 of [RFC5695] and section 26.3 of [RFC2544]) for each of the SR-MPLS forwarding operations of a DUT throughout the entire range of input data rates and frame sizes. The primary idea is to see what would be the frame loss under the overload conditions. It may be that overloaded forwarding engine would forward less traffic than in the situation close to the overload. Throughput may drop below the possible maximum. As per section 26.3 of [RFC2544], it is RECOMMENDED to have the data for all tested frame sizes with 10% load step above the wire-speed throughput measured in Section 5.1. It is RECOMMENDED to test all three test types discussed in Section 5.1.

Procedure: Similar to Section 5.1.

Reporting Format: A table with all parameters specified in Section 4.

5.5. System Recovery

Objective: To characterize the speed at which a DUT recovers from an overload condition for each of the SR-MPLS forwarding operations. It is RECOMMENDED to test all three test types discussed in Section 5.1.

Procedure: Similar to section 6.4 of [RFC5695] or section 26.5 of [RFC2544]. Send a stream of frames at a rate 110% of the recorded throughput rate or the maximum rate for the media, whichever is lower, for at least 60 seconds. At Timestamp A reduce the frame rate to 50% of the above rate and record the time of the last frame lost (Timestamp B). The system recovery time is determined by subtracting Timestamp B from Timestamp A. The test SHOULD be repeated a number of times and the average of the recorded values being reported.

Reporting Format: A table with all parameters specified in Section 4.

5.6. Reset

All type of reset tests are OPTIONAL.

Objective: To characterize the speed at which a DUT recovers from a device or software reset for each of the SR-MPLS forwarding operations. According to section 1.3 of [RFC6201] it is possible to measure frame loss or time stamps (depending on the test tool capability). According to section 4 of [RFC6201] reset could be: 1) hardware, 2) software, or 3) power interruption. All resets may be partial, i.e. only for a particular part of hardware (line card) or software (module). Especial interest may be to test redundant power supplies or routing engines to make sure that reset does not affect

the traffic. Hardware reset may be soft (command for reset) or hard (physical removal and insertion of the module). These types of reset SHOULD be treated as different. It is OPTIONAL to test all three test types discussed in Section 5.1, typically they would give the same result.

Procedure: It is inherited from [RFC6201] (see it for more details). It is simple in essence: create the traffic, initiate a reset, measure the time for the traffic lost.

Reporting Format: A table with all parameters specified in Section 4.

6. Security Considerations

Benchmarking methodologies are limited to technology characterization in a laboratory environment, with dedicated address space and constraints. 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 production networks. The benchmarking network topology is 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.

There are no specific security considerations within the scope of this document.

7. IANA Considerations

This document has no IANA actions.

8. Acknowledgements

The authors would like to thank Al Morton, Gabor Lencse, Boris Khasanov for the precious comments and suggestions.

9. References

9.1. Normative References

[RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <https://www.rfc-editor.org/info/rfc1242>.

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

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

[RFC3330] IANA, "Special-Use IPv4 Addresses", RFC 3330, DOI 10.17487/RFC3330, September 2002, <https://www.rfc-editor.org/info/rfc3330>.

[RFC4773] Huston, G., "Administration of the IANA Special Purpose IPv6 Address Block", RFC 4773, DOI 10.17487/RFC4773, December 2006, <https://www.rfc-editor.org/info/rfc4773>.

[RFC4814] Newman, D. and T. Player, "Hash and Stuffing: Overlooked Factors in Network Device Benchmarking", RFC 4814, DOI 10.17487/RFC4814, March 2007, <https://www.rfc-editor.org/info/rfc4814>.

[RFC5695] Akhter, A., Asati, R., and C. Pignataro, "MPLS Forwarding Benchmarking Methodology for IP Flows", RFC 5695, DOI 10.17487/RFC5695, November 2009, <https://www.rfc-editor.org/info/rfc5695>.

[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>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

[RFC8660] Bashandy, A., Ed., Filsfils, C., Ed., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with the MPLS Data Plane", RFC 8660, DOI 10.17487/RFC8660, December 2019, <https://www.rfc-editor.org/info/rfc8660>.

9.2. Informative References

[ETSI-GR-NFV-TST-007]
ETSI, "ETSI GR NFV-TST 007: Network Functions Virtualisation (NFV) Release 3; Testing; Guidelines on

Interoperability Testing for MANO", 2020, <https://www.etsi.org/deliver/etsi_gr/NFV-TST/001_099/007/03.01.01_60/gr_NFV-TST007v030101p.pdf>.

[I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Talaulikar, K., Mattes, P., and D. Jain, "Advertising Segment Routing Policies in BGP", Work in Progress, Internet-Draft, draft-ietf-idr-segment-routing-te-policy-25, 26 September 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-segment-routing-te-policy-25>.

[I-D.ietf-rtgwg-segment-routing-ti-lfa]
Litkowski, S., Bashandy, A., Filsfils, C., Francois, P., Decraene, B., and D. Voyer, "Topology Independent Fast Reroute using Segment Routing", Work in Progress, Internet-Draft, draft-ietf-rtgwg-segment-routing-ti-lfa-11, 30 June 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-rtgwg-segment-routing-ti-lfa-11>.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <https://www.rfc-editor.org/info/rfc3031>.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <https://www.rfc-editor.org/info/rfc3032>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <https://www.rfc-editor.org/info/rfc4364>.

[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>.

[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>.

[RFC8664] Sivabalan, S., Filsfils, C., Tantsura, J., Henderickx, W., and J. Hardwick, "Path Computation Element Communication Protocol (PCEP) Extensions for Segment Routing", RFC 8664, DOI 10.17487/RFC8664, December 2019, <https://www.rfc-editor.org/info/rfc8664>.

[RFC8665] Psenak, P., Ed., Previdi, S., Ed., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", RFC 8665, DOI 10.17487/RFC8665, December 2019, <https://www.rfc-editor.org/info/rfc8665>.

[RFC8667] Previdi, S., Ed., Ginsberg, L., Ed., Filsfils, C., Bashandy, A., Gredler, H., and B. Decraene, "IS-IS Extensions for Segment Routing", RFC 8667, DOI 10.17487/RFC8667, December 2019, <https://www.rfc-editor.org/info/rfc8667>.

[RFC8669] Previdi, S., Filsfils, C., Lindem, A., Ed., Sreekantiah, A., and H. Gredler, "Segment Routing Prefix Segment Identifier Extensions for BGP", RFC 8669, DOI 10.17487/RFC8669, December 2019, <https://www.rfc-editor.org/info/rfc8669>.

[RFC9004] Morton, A., "Updates for the Back-to-Back Frame Benchmark in RFC 2544", RFC 9004, DOI 10.17487/RFC9004, May 2021, <https://www.rfc-editor.org/info/rfc9004>.

[RFC9256] Filsfils, C., Talaulikar, K., Ed., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", RFC 9256, DOI 10.17487/RFC9256, July 2022, <https://www.rfc-editor.org/info/rfc9256>.

Authors' Addresses

Giuseppe Fioccola
Huawei Technologies
Palazzo Verrocchio, Centro Direzionale Milano 2
20054 Segrate (Milan)
Italy
Email: giuseppe.fioccola@huawei.com

Eduard Vasilenko
Huawei Technologies
17/4 Krylatskaya str.
Moscow
Email: vasilenko.eduard@huawei.com

Paolo Volpato
Huawei Technologies
Palazzo Verrocchio, Centro Direzionale Milano 2
20054 Segrate (Milan)
Italy
Email: paolo.volpato@huawei.com

Luis Miguel Contreras Murillo
Telefonica
Spain
Email: luismiguel.contrerasmurillo@telefonica.com

Bruno Decraene
Orange
France
Email: bruno.decraene@orange.com

BMWG
Internet-Draft
Intended status: Informational
Expires: January 7, 2023

G. Fioccola
E. Vasilenko
P. Volpato
Huawei Technologies
L. Contreras
Telefonica

July 6, 2022

Benchmarking Methodology for IPv6 Segment Routing
draft-vfv-bmwg-srv6-bench-meth-02

Abstract

This document defines a methodology for benchmarking Segment Routing (SR) performance for Segment Routing over IPv6 (SRv6). It builds upon [RFC2544], [RFC5180], [RFC5695] and [RFC8402].

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], RFC 8174 [RFC8174].

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 7, 2023.

Copyright Notice

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

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

Segment Routing (SR), defined in [RFC8402], leverages the source routing paradigm. The headend node steers a packet through an SR Policy [I-D.ietf-spring-segment-routing-policy], instantiated as an ordered list of segments. A segment, referred to by its Segment Identifier (SID), can have a semantic local to an SR node or global within an SR domain. SR supports per-flow explicit routing while

maintaining per-flow state only at the ingress nodes to the SR domain.

However, there is no standard method defined to compare and contrast the foundational SR packet forwarding capabilities of network devices. This document aims to extend the efforts of [RFC1242] and [RFC2544] to SR network.

The SR architecture can be instantiated on two data-plane: SR over MPLS (SR-MPLS) and SR over IPv6 (SRv6). This document is limited to SRv6.

This document is limited to Headend encapsulations (H.Encaps.xxx) and segment Endpoints (End, End.X). It is expected that future documents may cover the benchmarking of SRv6 applications with decapsulation (End.Dxxx), Binding (End.Bxxx), Fast ReRoute [I-D.ietf-rtgwg-segment-routing-ti-lfa], etc.

SR can be applied to the IPv6 architecture with a new type of routing header called the SR Header (SRH) [RFC8754]. An instruction is associated with a segment and encoded as an IPv6 address. An SRv6 segment is also called an SRv6 SID. An SR Policy is instantiated as an ordered list of SRv6 SIDs in the routing header. The active segment is indicated by the Destination Address (DA) of the packet.

SRv6 involves 3 types of forwarding plane operations:

- o PUSH consists of the insertion of one or more segments on top of the incoming packet. It is the SRH attachment in the case of SRv6.

- o NEXT consists of the inspection of the next segment. The active segment is completed and the next segment is activated. It is the copy of the next segment from the SRH to the destination address of the IPv6 header in SRv6.

- o CONTINUE happens when the active segment is not completed; hence, it remains active. It is the plain IPv6 forwarding action of a regular IPv6 packet according to its destination address in SRv6.

SR list for PUSH operation is typically constructed by SR Policy in ingress node, see [I-D.ietf-spring-segment-routing-policy].

[RFC5695] describes a methodology specific to the benchmarking of MPLS forwarding devices, by considering the most common MPLS packet forwarding scenarios and corresponding performance measurements.

[RFC5180] provides benchmarking methodology recommendations that address IPv6-specific aspects, such as evaluating the forwarding performance of traffic containing extension headers.

The purpose of this document is to describe a methodology specific to the benchmarking of Segment Routing over IPv6. The methodology described is a complement for [RFC5180] and [RFC5695].

2. SRv6 Forwarding

An SRv6 SID is allocated in the form of an IPv6 address. For the IPv6 data plane, a new type of IPv6 Routing Extension Header, called Segment Routing Header (SRH) has been defined [RFC8754]. The SRH contains the Segment List as an ordered list of IPv6 addresses: each address in the list is a SID. A dedicated field, referred to as Segments Left, is used to maintain the pointer to the active SID of the Segment List.

There are three different categories of nodes that may be involved in segment routing networks.

The SR source node is the headend node and steers a packet into an SR Policy. It can be a host originating an IPv6 packet or an SR domain ingress router encapsulating a received packet into an outer IPv6 packet and inserts the SRH in the outer IPv6 header. It sets the first SID of the SR Policy as the IPv6 Destination Address of the packet.

The SR transit node forwards packets destined to a remote segment as a normal IPv6 packet based on the IPv6 destination address, because the IPv6 destination address does not locally match with a segment. Indeed, according to [RFC8200] the only node allowed to inspect the Routing Extension Header (and therefore the SRH) is the node corresponding to the destination address of the packet.

The SR segment endpoint node receives packets whose IPv6 destination address is locally configured as a segment. It creates Forwarding Information Base (FIB) entries for its local SIDs. For each SR packet, it inspects the SRH, may prepare some actions (like forwarding through a particular port), then replaces the IPv6 destination address with the new active segment.

The operations applied by the SRv6 packet processing are different at the SR source, transit and SR segment endpoint nodes.

The processing of the SR source node corresponds to the sequence of the insertion of the SRH, composed of SIDs stored in reverse order, and setting of the IPv6 Destination Address as the first SID of the

SR Policy. It can be performed by encapsulating a packet into an outer IPv6 packet with an SRH.

The processing of the SR segment endpoint node corresponds to the detection of the new active segment, which is the next segment in the Segment List and the related modification of the IPv6 destination address of the outer IPv6 header. Then packets are forwarded on the basis of the IPv6 forwarding table.

The processing of the SR transit node corresponds to normal forwarding of the packets containing the SR header. In SRv6 the transit nodes do not need to be SRv6 aware, as every IPv6 router can act as an SRv6 transit node since any IPv6 node will maintain a plain IPv6 FIB entry for any prefix, no matter if the prefix represents a segment or not.

[I-D.ietf-spring-segment-routing-policy] specifies the concepts of SR Policy and steering into an SR Policy. The header of a packet steered in an SR Policy is augmented with the ordered list of segments associated with that SR Policy. SR Policy state is instantiated only on the headend node, that steers a flow into an SR Policy. Indeed intermediate and endpoint nodes do not require any state to be maintained. SR Policies can be instantiated on the headend dynamically and on demand basis. SR policy may be installed by PCEP [RFC8664], BGP [I-D.ietf-idr-segment-routing-te-policy], or via manual configuration on the router. PCEP signaling can be the case of a controller based deployment. For all these reasons, SR Policies scale better than traditional TE mechanisms.

In addition to the basic SRv6 packet processing, the SRv6 Network Programming model [RFC8986] describes a set of functions that can be associated to segments and executed in a given SRv6 node.

Examples of such functions are described in [RFC8986], but, in practice, any behavior and function can be associated with a local SID in a node, to apply any special processing on the packet. The definition of a standardized set of segment routing functions facilitates the deployment of SR domains with interoperable equipment from multiple vendors.

According to [RFC8986], 128 bit SID can be logically split into three fields and interpreted as LOCATOR:FUNCTION:ARGS (in short LOC:FUNCT:ARG) where LOC includes the L most significant bits, FUNCT the following F bits and ARG the remaining A bits, where L+F+A=128. The LOC corresponds to an IPv6 prefix (for example with a length of 48, 56 or 64 bits) that can be distributed by the routing protocols and provides the reachability of a node that hosts some functions. All the different functions residing in a node have a different FUNCT

code, so that their SIDs will be different. The ARG bits are used to provide information (arguments) to a function. From the routing point of view, the solution is scalable, as a single prefix is distributed for a node, which implements a potentially large number of functions and related arguments.

3. Test Methodology

3.1. Test Setup

The test setup in general is compliant with section 6 of [RFC2544] but augmented by the methodology specified in section 4 of [RFC5695] using many ports. It is needed to test the packet forwarding engine that may have different performance based on the number of ports served. The Device Under Test (DUT) may have oversubscribed ports, then traffic for such ports should be proportionally decreased according to the specific DUT oversubscription ratio. All ports served by a particular packet forwarding engine should be loaded in reverse proportion to the claimed oversubscription ratio. Tests SHOULD be done with bidirectional traffic that better reflects the real environment for SRv6 nodes. It is OPTIONAL to choose a non-equal proportion for upstream and downstream traffic for some specific aggregation nodes.

The recommended topology for SRv6 Forwarding Benchmarking should be the same as MPLS and it is described in section 4 of [RFC5695]. Port numbers involved in the tests and their oversubscription ratio MUST be reported. In general, Functions of the last SID (called "behavior" in [RFC8986]) may be used to encode services (similar to L2/L3 VPNs and much more) but it is out of the scope of this document. This document is benchmarking only "source routing". Hence, SIDs represent only Headend encapsulation (H.Encaps.xxx) or segment Endpoint (End, End.X) that may be carried in IGP extensions.

It is OPTIONAL to test SRH in the combination with any other extension headers (fragmentation, hop-by-hop, destination options, etc.) but in all tests, SRH header should be present for the test to be relevant for SRv6. It is RECOMMENDED to follow section 5.3 of [RFC5180] to introduce other extension headers in proportion 1%, 10%, 50% that may better reflect real use cases.

Segment Routing may also be implemented as a software network function in an NFV Infrastructure and, in this case, additional considerations should be done. [RFC9004] updates the procedures of the test to measure the Back-to-Back Frames since their characterization is relevant in software-packet processing. Also, [ETSI-GR-NFV-TST-007] describes test guidelines for NFV capabilities

that require interactions between the components implementing NFV functionality.

3.2. Locator and Endpoint behaviors methods

As specified in [RFC8986], topological segments have the structure that consists of Locator and Endpoint behavior (End, End.X, etc), the latter may have a few different flavors (PSP, USP, USD).

It is RECOMMENDED that the DUT and test tool support at least one option for SID stack construction:

- o IS-IS Extensions to Support Segment Routing over IPv6 Dataplane [I-D.ietf-lsr-isis-srv6-extensions]

- o OSPFv3 Extensions for SRv6 [I-D.ietf-lsr-ospfv3-srv6-extensions]

- o Segment Routing Policy Architecture [I-D.ietf-spring-segment-routing-policy].

It is RECOMMENDED that at least one routing protocol (OSPF or ISIS) should be used for the construction of the simplest SRH with 1 SID. It is RECOMMENDED that SR policy should be used for the construction of SRH with 2 SIDs. It is possible to test longer SRH if there is an interest.

It is RECOMMENDED that the top SID on the list should have an End.X flavor type to emulate traffic engineering scenario. In all cases, SID stack configuration SHOULD happen before packet forwarding would be started. Control plane convergence speed is not the subject of the present tests.

The Locator and Endpoint construction method and SR policy construction method used MUST be reported according to Section 4.

3.3. Frame Formats and Sizes

SRv6 tests will use the Frame characteristics similarly to section 4.1.5 of [RFC5695], except the need for a bigger MTU to accommodate SRH.

It is to be noted that [RFC5695] requires exactly a single entry in the MPLS label stack in an MPLS packet that is not enough to simulate a typical SR SID list. The number of entries in SRH MUST be reported.

According to section 4.1.4.2 of [RFC5695], the payload is RECOMMENDED to have an IP packet (IPv6 or IPv4 with UDP or TCP) to better represent the real environment.

It is assumed that the test would be for Ethernet media only. Other media is possible (see section 4.1.5.2 of [RFC5695] for the POS example). Recommended frame sizes are presented below. Any other frame sized may be added if suspected of abnormal behavior. For example, some architectures may allocate buffer memory in big fixed chunks that may drop performance if frame sizes are chosen just a few octet more than the fixed chunk size (the second chunk would have a very low memory utilization).

Recommended frame sizes are the following:

- o Ethernet Minimal: 64+8+n*16

- o DUT Minimal Wire Speed: 128-256 (it depends on the range recommended in the DUT specification)

- o Ethernet Typical: 1518+8+n*16

- o DUT Maximum Wire Speed: 9000

Note that 8 octets are added in the previous calculations for the SRH header itself. While n*16 octets are added to accommodate SID entries. The typical frame size values are listed above for the DUT minimal and maximum wire speed, but they can be modified according to the DUT characteristics. Indeed, the minimum wire speed frame size can be considered based on the DUT specification but, in some cases, many tests may be needed in the search for the real minimum wire speed frame size. VLAN tag may additionally increase the frame size. VLAN tag tests are OPTIONAL.

3.4. Protocol Addresses

IANA reserved an IPv6 address block 2001:0200::/48 for use with IPv6 benchmark testing (see section 8 of [RFC5180]). IPv6 source and destination addresses for the test streams SHOULD belong to the IPv6 range assigned by IANA. It is not principal what Locator blocks would be chosen for tests. It may be /52, /56, /64, or even bigger. It is possible to test a few different Locator blocks if there is a need.

3.5. Trial Duration

The test portion of each trial SHOULD be at least 10 seconds longer than the hold time for the respective protocol configuration to verify that the DUT can maintain a stable control plane when the data-forwarding plane is under stress. IGP protocols typically have a shorter hold time, some BGP default configuration may be up to 180 seconds. It is needed to check the default hold time of the DUT for the respective protocol used.

3.6. Traffic Verification

Traffic verification is following section 4.1.8 of [RFC5695].

4. Reporting Format

There are a few parameters that must be changed in section 5 of [RFC5695] for SRv6 tests. New parameters that MUST be reported are:

- o Port numbers involved in the tests and their respective oversubscription ratio.
- o Upstream/downstream traffic proportion (equal bidirectional or some other split).
- o SRv6 Forwarding Operations (PUSH/ NEXT/ CONTINUE).
- o Number of Segments considered in the SRH and the type of behavior used (according to [RFC8986]).
- o SR Policy construction method (PCEP, BGP, manual configuration).
- o Type of the payload (IPv6/IPv4, UDP/TCP).

Some parameters MAY be changed:

- o Label Distribution protocol and IGP are the same in the context of SRv6. Hence, it is called "Locator and Endpoint behaviors methods".
- o Port media type may be reported only one time for all tests if only Ethernet media would be tested

5. SRv6 Forwarding Benchmarking Tests

In general, tests are compliant with [RFC2544] but the important correction discussed in section 6 of [RFC2544] is applied: ports chosen for every test MUST stress all ports served by one forwarding

engine. It is better to check the DUT specification for the relationship between ports and the forwarding engine to minimize the number of ports involved. But it is possible to understand the worst case by looking at the throughput and latency from the trial tests. If any doubt exists about how full is the offered load for the forwarding engine then it is better to stress all ports of the line card or all ports for the whole router with a centralized forwarding engine. Partial load on forwarding engine would show optimistic results. Controllable traffic distribution between many ports (as specified in section 4 of [RFC5695]) would need separate SID announcements for separate ports. The search for No-Drop Rate (NDR) should be done for every test as explained in section 6 of [RFC5695].

5.1. Throughput

This section contains the description of the tests that are related to the characterization of a DUT's SRv6 traffic forwarding throughput.

The list of segments for SRv6 is represented as a list of IPv6 addresses, included in the SRH. There are three distinct types of nodes that are involved in segment routing networks.

5.1.1. Throughput of a Source Node

Objective: To obtain the DUT's Throughput during the packet processing of a Source Node. It is when the Source SR node, which corresponds to the headend node, encapsulates a received packet into an outer IPv6 packet and inserts the SR Header (SRH) as a Routing Extension Header in the outer IPv6 header. The Segment List in the SRH is composed of SIDs and the Source SR node sets the first SID of the SR Policy as the IPv6 Destination Address of the packet.

Procedure: Similar to [RFC5695] with potential extension to test SID list longer than 1 SID (2 are recommended, many are possible).

Reporting Format: Similar to [RFC5180] with the additional parameters specified in Section 4.

5.1.2. Throughput of a Segment Endpoint Node

Objective: To obtain the DUT's Throughput during the packet processing of a Segment Endpoint Node. It is when the SR Segment Endpoint node receives packets whose IPv6 destination address is locally configured as a segment. The SR Segment Endpoint node inspects the SR header: it detects the new active segment, i.e. the next segment in the Segment List, modifies the IPv6 destination

address of the outer IPv6 header and forwards the packet on the basis of the IPv6 forwarding table.

Procedure: Similar to [RFC5695] with potential extension to test SID list longer than 1 SID (2 are recommended, many are possible).

Reporting Format: Similar to [RFC5180] with the additional parameters specified in Section 4.

5.1.3. Throughput of a Transit Node

Objective: To obtain the DUT's Throughput during the packet processing of a Transit Node. It is when a Transit node forwards the packet containing the SR header as a normal IPv6 packet because the IPv6 destination address does not locally match with a segment.

Procedure: Similar to [RFC5695] with potential extension to test SID list longer than 1 SID (2 are recommended, many are possible).

Reporting Format: Similar to [RFC5180] with the additional parameters specified in Section 4.

5.2. Latency

Objective: To determine the latency as defined in section 6.2 of [RFC5695] for each of the SRv6 forwarding operations.

Procedure: Similar to [RFC5695] with potential extension to test SID list longer than 1 SID (2 are recommended, many are possible).

Reporting Format: Similar to [RFC5180] with the additional parameters specified in Section 4.

5.3. Frame Loss

Objective: To determine the frame-loss rate (as defined in section 6.3 of [RFC5695]) for each of the SRv6 forwarding operations of a DUT throughout the entire range of input data rates and frame sizes.

Procedure: Similar to [RFC5695] with potential extension to test SID list longer than 1 SID (2 are recommended, many are possible).

Reporting Format: Similar to [RFC5180] with the additional parameters specified in Section 4.

5.4. System Recovery

Objective: To characterize the speed at which a DUT recovers from an overload condition for each of the SRv6 forwarding operations.

Procedure: Similar to section 6.4 of [RFC5695].

Reporting Format: Similar to [RFC5180] with the additional parameters specified in Section 4.

5.5. Reset

Objective: To characterize the speed at which a DUT recovers from a device or software reset for each of the SRv6 forwarding operations.

Procedure: Similar to section 6.5 of [RFC5695].

Reporting Format: Similar to [RFC5180] with the additional parameters specified in Section 4.

It is OPTIONAL to extend the Reset tests according to [RFC6201] in order to reset only part of the DUT: only line card reset, only process reset (for example ISIS), only one routing engine reset in the configuration with routing engine redundancy, full power interruption, partial power interruption, etc.

6. Security Considerations

Benchmarking methodologies are limited to technology characterization in a laboratory environment, with dedicated address space and constraints. 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 production networks. The benchmarking network topology is 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.

There are no specific security considerations within the scope of this document.

7. IANA Considerations

This document has no IANA actions.

8. Acknowledgements

The authors would like to thank Al Morton for the precious comments and suggestions.

9. References

9.1. Normative References

[RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <https://www.rfc-editor.org/info/rfc1242>.

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

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

[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>.

[RFC5695] Akhter, A., Asati, R., and C. Pignataro, "MPLS Forwarding Benchmarking Methodology for IP Flows", RFC 5695, DOI 10.17487/RFC5695, November 2009, <https://www.rfc-editor.org/info/rfc5695>.

[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>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <https://www.rfc-editor.org/info/rfc8986>.

9.2. Informative References

[ETSI-GR-NFV-TST-007]
ETSI, "ETSI GR NFV-TST 007: Network Functions Virtualisation (NFV) Release 3; Testing; Guidelines on Interoperability Testing for MANO", 2020, <https://www.etsi.org/deliver/etsi_gr/NFV-TST/001_099/007/02.06.01_60/gr_nfv-tst007v020601p.pdf>.

[I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Talaulikar, K., Mattes, P., Jain, D., and S. Lin, "Advertising Segment Routing Policies in BGP", draft-ietf-idr-segment-routing-te-policy-18 (work in progress), June 2022.

[I-D.ietf-lsr-isis-srv6-extensions]
Psenak, P., Filsfils, C., Bashandy, A., Decraene, B., and Z. Hu, "IS-IS Extensions to Support Segment Routing over IPv6 Dataplane", draft-ietf-lsr-isis-srv6-extensions-18 (work in progress), October 2021.

[I-D.ietf-lsr-ospfv3-srv6-extensions]
Li, Z., Hu, Z., Talaulikar, K., and P. Psenak, "OSPFv3 Extensions for SRv6", draft-ietf-lsr-ospfv3-srv6-extensions-05 (work in progress), July 2022.

[I-D.ietf-rtgwg-segment-routing-ti-lfa]
Litkowski, S., Bashandy, A., Filsfils, C., Francois, P., Decraene, B., and D. Voyer, "Topology Independent Fast Reroute using Segment Routing", draft-ietf-rtgwg-segment-routing-ti-lfa-08 (work in progress), January 2022.

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-22 (work in progress), March 2022.

[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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8664] Sivabalan, S., Filsfils, C., Tantsura, J., Henderickx, W., and J. Hardwick, "Path Computation Element Communication Protocol (PCEP) Extensions for Segment Routing", RFC 8664, DOI 10.17487/RFC8664, December 2019, <https://www.rfc-editor.org/info/rfc8664>.

[RFC9004] Morton, A., "Updates for the Back-to-Back Frame Benchmark in RFC 2544", RFC 9004, DOI 10.17487/RFC9004, May 2021, <https://www.rfc-editor.org/info/rfc9004>.

Authors' Addresses

Giuseppe Fioccola
Huawei Technologies
Riesstrasse, 25
Munich 80992
Germany

Email: giuseppe.fioccola@huawei.com

Eduard Vasilenko
Huawei Technologies
17/4 Krylatskaya str.
Moscow 121614
Russia

Email: vasilenko.eduard@huawei.com

Paolo Volpato
Huawei Technologies
Via Lorenteggio, 240
Milan 20147
Italy

Email: paolo.volpato@huawei.com

Luis Miguel Contreras Murillo
Telefonica
Spain

Email: luismiguel.contrerasmurillo@telefonica.com

BMWG
Internet-Draft
Intended status: Informational
Expires: 25 April 2024

G. Fioccola
E. Vasilenko
P. Volpato
Huawei Technologies
L. Contreras
Telefonica
B. Decraene
Orange

23 October 2023

Benchmarking Methodology for IPv6 Segment Routing
draft-vfv-bmwg-srv6-bench-meth-08

Abstract

This document defines a methodology for benchmarking Segment Routing (SR) performance for Segment Routing over IPv6 (SRv6). It builds upon RFC 2544, RFC 5180, RFC 5695 and RFC 8402.

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 25 April 2024.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction

Segment Routing (SR), defined in [RFC8402], leverages the source routing paradigm. The headend node steers a packet through an SR Policy [RFC9256], instantiated as an ordered list of segments. A segment, referred to by its Segment Identifier (SID), can have a semantic local to an SR node or global within an SR domain. SR supports per-class explicit routing while maintaining per-class state only at the ingress nodes to the SR domain.

However, there is no standard method defined to compare and contrast the foundational SR packet forwarding capabilities of network devices. This document aims to extend the efforts of [RFC1242], [RFC2544], [RFC5180] and [RFC5695] to SR network.

The SR architecture can be instantiated on two data-plane: SR over MPLS (SR-MPLS) and SR over IPv6 (SRv6). This document is limited to SRv6.

This document is limited to Headend encapsulations (H.Encaps.xxx) and segment Endpoints (End, End.X). It is expected that future documents may cover the benchmarking of SRv6 applications with decapsulation (End.Dxxx), Binding (End.Bxxx), Fast ReRoute [I-D.ietf-rtgwg-segment-routing-ti-lfa], Compressed SID [I-D.ietf-spring-srv6-srh-compression], etc.

SR can be applied to the IPv6 architecture with a new type of routing header called the SR Header (SRH) [RFC8754]. An instruction is associated with a segment and encoded as an IPv6 address. An SRv6 segment is also called an SRv6 SID. An SR Policy is instantiated as an ordered list of SRv6 SIDs in the routing header. The active segment is indicated by the Destination Address (DA) of the packet.

SRv6 involves 3 types of forwarding plane operations (PUSH/ NEXT/ CONTINUE) as further described in Section 2. SR list for PUSH operation is typically constructed by the source node with a SR Policy, see [RFC9256]. NEXT operation is done by a SR segment endpoint node. CONTINUE operation happens for a SR transit node.

[RFC5695] describes a methodology specific to the benchmarking of MPLS forwarding devices, by considering the most common MPLS packet forwarding scenarios and corresponding performance measurements.

[RFC5180] provides benchmarking methodology recommendations that address IPv6-specific aspects, such as evaluating the forwarding performance of traffic containing extension headers.

The purpose of this document is to describe a methodology specific to the benchmarking of Segment Routing over IPv6. The methodology described is a complement for [RFC5180] and [RFC5695].

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], RFC 8174 [RFC8174].

2. SRv6 Forwarding

SR leverages the source routing paradigm. In SRv6, a SID is allocated as an IPv6 address. For the IPv6 data plane, a new type of IPv6 Routing Extension Header, called Segment Routing Header (SRH) has been defined [RFC8754]. The SRH contains the Segment List as an ordered list of IPv6 addresses: each address in the list is a SID. Hence SRv6 Segment list typically contains more SIDs. A dedicated field, referred to as Segments Left, is used to maintain the pointer to the active SID of the Segment List.

There are three different categories of nodes that may be involved in segment routing networks.

The SR source node is the headend node and steers a packet into an SR Policy. It can be a host originating an IPv6 packet or an SR domain ingress router encapsulating a received packet into an outer IPv6 packet and inserts the SRH in the outer IPv6 header. It sets the first SID of the SR Policy as the IPv6 Destination Address of the packet.

The SR transit node forwards packets destined for a remote segment as a normal IPv6 packet based on the IPv6 destination address, because the IPv6 destination address does not locally match with a segment. According to [RFC8200] the only node allowed to inspect the Routing Extension Header (and therefore the SRH) is the node corresponding to the destination address of the packet.

The SR segment endpoint node receives packets whose IPv6 destination address is locally configured as a segment. It creates Forwarding Information Base (FIB) entries for its local SIDs. For each SR packet, it inspects the SRH, may prepare some actions (like forwarding through a particular interface), then replaces the IPv6 destination address with the new active segment.

The operations applied by the SRv6 packet processing are different at the SR source, transit, and SR segment endpoint nodes.

The processing of the SR source node corresponds to the sequence of insertion of the SRH, composed of SIDs stored in reverse order, and setting of the IPv6 Destination Address as the first SID of the SR Policy. It can be performed by encapsulating a packet into an outer IPv6 packet with an SRH.

The processing of the SR segment endpoint node corresponds to the detection of the new active segment, which is the next segment in the Segment List and the related modification of the IPv6 destination address of the outer IPv6 header. Then packets are forwarded on the basis of the IPv6 forwarding table.

The processing of the SR transit node corresponds to normal forwarding of the packets containing the SR header. In SRv6, the transit nodes do not need to be SRv6 aware, as every IPv6 router can act as an SRv6 transit node since any IPv6 node will maintain a plain IPv6 FIB entry for any prefix, no matter if the prefix represents a segment or not.

[RFC9256] specifies the concepts of SR Policy and steering into an SR Policy. The header of a packet steered in an SR Policy is augmented with the ordered list of segments associated with that SR Policy. SR Policy state is instantiated only on the headend node, which steers a flow into an SR Policy. Intermediate and endpoint nodes do not require any state to be maintained. SR Policies can be instantiated on the headend dynamically and on demand basis. SR policy may be installed by PCEP [RFC8664], BGP [I-D.ietf-idr-segment-routing-te-policy], or via manual configuration on the router. PCEP and BGP signaling of SR Policies can be the case of a controller-based deployment.

In addition to the basic SRv6 packet processing, the SRv6 Network Programming model [RFC8986] describes a set of functions that can be associated to segments and executed in a given SRv6 node.

Examples of such functions are described in [RFC8986], but, in practice, any behavior and function can be associated with a local SID in a node, to apply any special processing on the packet. The definition of a standardized set of segment routing functions facilitates the deployment of SR domains with interoperable equipment from multiple vendors.

According to [RFC8986], 128 bit SID can be logically split into three fields and interpreted as LOCATOR:FUNCTION:ARGS (in short LOC:FUNCT:ARG) where LOC includes the L most significant bits, FUNCT the following F bits and ARG the remaining A bits, where L+F+A=128. The LOC corresponds to an IPv6 prefix (for example with a length of 48, 56, or 64 bits) that can be distributed by the routing protocols

and provides the reachability of a node that hosts some functions. All the different functions residing in a node have a different FUNCT code, so that their SIDs will be different. The ARG bits are used to provide information (arguments) to a function. From the routing point of view, the solution is scalable, as a single prefix is distributed for a node, which implements a potentially large number of functions and related arguments.

3. Test Methodology

3.1. Test Setup

The test setup in general is compliant with section 6 of [RFC2544] but augmented by the methodology specified in section 4 of [RFC5695] using many interfaces. It is needed to test the packet forwarding engine that may have different performance based on the number of interfaces served. The Device Under Test (DUT) may have oversubscribed interfaces, then traffic for such interfaces should be proportionally decreased according to the specific DUT oversubscription ratio. All interfaces served by a particular packet forwarding engine should be loaded in reverse proportion to the claimed oversubscription ratio. Tests SHOULD be done with bidirectional traffic that better reflects the real environment for SRv6 nodes. It is OPTIONAL to choose a non-equal proportion for upstream and downstream traffic for some specific aggregation nodes.

The RECOMMENDED topology for SRv6 Forwarding Benchmarking should be the same used for MPLS benchmarking, as described in section 4 of [RFC5695]. A simplified view is reported below for reference.

```
                    +----------+
          +---------|          |<---------+
          | +-------|  Tester  |<-------+ |
          | | +-----|          |<-----+ | |
          | | |     +----------+      | | |
          | | |                       | | |
          | | |      +--------+       | | |
          | | +----->|        |-------+ | |
          | +------->|  DUT   |---------+ |
          +--------->|        |-----------+
                     +--------+
```

Figure 1: Test environment for SRv6 Forwarding Benchmarking

Differently from [RFC5695], this document prefers the use of the term "interface" instead of "port" as an interface may be either virtual or physical.

The RECOMMENDED topology for SRv6 Forwarding Benchmarking should be the same as MPLS and it is described in section 4 of [RFC5695]. Interface numbers involved in the tests and their oversubscription ratio MUST be reported. This document is benchmarking only "source routing". Hence, SIDs represent only Headend encapsulation (H.Encaps.xxx) or segment Endpoint (End, End.X) that may be carried in IGP extensions. In general, Functions of the last SID (called "behavior" in [RFC8986]) may be used to encode services (similar to L2/L3 VPNs and much more) but it is out of the scope of this document.

It is OPTIONAL to test SRH in the combination with any other extension headers (fragmentation, hop-by-hop, destination options, etc.) but in all tests, the SRH header should be present for the test to be relevant for SRv6. It is RECOMMENDED to follow section 5.3 of [RFC5180] to introduce other extension headers in proportion 1%, 10%, 50% that may better reflect real use cases.

Segment Routing may also be implemented as a software network function in an NFV Infrastructure and, in this case, additional considerations should be done. [ETSI-GR-NFV-TST-007] describes test guidelines for NFV capabilities that require interactions between the components implementing NFV functionality.

Special capabilities SHOULD NOT exist in the DUT/SUT specifically for benchmarking purposes.

3.2. Locator and Endpoint behaviors methods

As specified in [RFC8986], topological segments have the structure that consists of Locator and Endpoint behavior (H.Encaps, End, End.X, etc), the latter may have a few different flavors (PSP, USP, USD). Different combinations of behavior and flavor are recommended for every test.

It is RECOMMENDED that the DUT and test tool support at least one option for SID stack construction:

* IS-IS Extensions to Support Segment Routing over IPv6 Dataplane [RFC9352]

* OSPFv3 Extensions for SRv6 [I-D.ietf-lsr-ospfv3-srv6-extensions]

* Segment Routing Policy Architecture [RFC9256].

A routing protocol (OSPF or IS-IS) SHOULD be used for the construction of the simplest SRH with 1 SID. It is RECOMMENDED that SR policy should be used for the construction of SRH with 2 SIDs. It

is OPTIONAL to test longer SRH if there is an interest. For longer SRH lists, it is suggested to consider DUT maximum, or "network diameter" plus one SID for the service, where "network diameter" is the maximum number of SR hops that the target SRv6 network have, it is typically between 3 to 7 hops.

It is RECOMMENDED that the top SID on the list should have an End.X flavor type to emulate traffic engineering scenario. In all cases, SID stack configuration SHOULD happen before packet forwarding would be started. Control plane convergence speed is not the subject of the present tests.

The Locator and Endpoint construction method and SR policy construction method used MUST be reported according to Section 4.

3.3. Frame Formats and Sizes

SRv6 tests will use Frame characteristics similarly to section 4.1.5 of [RFC5695], except the need for a bigger MTU to accommodate SRH.

It is assumed that MTU is big enough to accommodate all frame sizes proposed below. Fragmentation is not an option for SRv6.

It is to be noted that [RFC5695] requires exactly a single entry in the MPLS label stack in an MPLS packet that is not enough to simulate a typical SR SID list. The number of entries in SRH MUST be reported.

According to section 4.1.4.2 of [RFC5695], the payload is RECOMMENDED to have an IP packet (IPv6 or IPv4 with UDP or TCP) to better represent the real environment. The minimal Ethernet payload (46B) could not accommodate the whole IPv6 stack (not enough room for TCP or UDP), hence only IPv4 is possible to use if the test for minimal Ethernet payload is needed. It is possible to choose the bigger payload size for the IPv6 only environment.

It is assumed that the test would be for Ethernet media only. Other media is possible (see section 4.1.5.2 of [RFC5695] for the POS example). Some layer 2 technologies (like POS/PPP) have bit- or byte- stuffing then [RFC4814] may help to calculate real performance more accurately or else 1-2% error is expected. The most popular layer 2 technology for SR is Ethernet, it does not have stuffing.

RECOMMENDED frame sizes are presented below. Any other frame sizes may be added if suspected of abnormal behavior. For example, some architectures may allocate buffer memory in big fixed chunks that may drop performance if frame sizes are chosen just a few octet more than the fixed chunk size (the second chunk would have a very low memory utilization).

The resulting Ethernet frame structure is depicted in the next figure.

```
<---18B---><-40B-><8+n*16B><--------46-1500-9000B----------->
+---------+-------+-------+---------+----------------------+
|         | Outer |       |  Inner  |         |            |
| Layer 2 | IPv6  |  SRH  | Layer 3 | Layer 4 | High layers |
+---------+-------+-------+---------+----------------------+
```

Figure 2: Ethernet Frame Structure

RECOMMENDED payload sizes (encapsulated packet with L3 headers and above) are the following:

* Ethernet Minimal: 46

* DUT Minimal Wire Speed: typically 128-256 (it depends on the DUT specification)

* Ethernet Typical: 1500

* DUT Maximum: 9000 (or any claimed maximum)

It is evident from the figure 2 that in all cases we need to add 40+8+n*16 bytes for the needed frame size, where

40 octets are added for the outer (tunnel) IPv6 header

8 octets are added for the SRH header itself

n is the number of segments multiplied on 16 octet SID size.

The typical frame size values are listed above for the DUT minimal wire speed and maximum, but they can be modified according to the DUT characteristics. The minimum wire speed frame size can be considered based on the DUT specification but, in some cases, many tests may be needed in the search for the real minimum wire speed frame size. VLAN tag may additionally increase the frame size. VLAN tag tests are OPTIONAL.

3.4. Protocol Addresses

IANA reserved an IPv6 address block 2001:0002::/48 for use with IPv6 benchmark testing (see section 8 of [RFC5180]). IPv6 source and destination addresses for the test streams SHOULD belong to the IPv6 range assigned by IANA. It is not principal what Locator blocks would be chosen for tests. It may be /52, /56, /64, or even bigger. It is possible to test a few different Locator blocks if there is a need.

As it is discussed in section 3.1, there is a need to load the whole forwarding engine (on all interfaces). [RFC4814] discusses the importance to have many flows with address randomization for acceptable hash-based load balancing that is implemented in all forwarding engines. Note that IPv6 flow label randomization can be used, according to [RFC6438]. In the context of this document, it may also be relevant for SIDs, because SIDs may be used for hash to choose the next link (depending on DUT default or desired configuration). It is important to check what exactly is used for the hash load balancing algorithm on the DUT to keep these numbers sufficiently random and at volume. It is very often that IP addresses and transport protocol ports are used instead of SIDs.

3.5. Trial Duration

The test portion of each trial must take into account the respective protocol configuration. IGP protocols typically have a shorter hold time, while some BGP default configurations may be up to 180 seconds. It is needed to check the default hold time of the DUT for the respective protocol used.

In general, the test portion of each trial SHOULD be no less than 250 seconds, which is a reasonable value based on common hold time values. But a test can also adapt to the real setup and select a different value if default configuration has been changed. The test portion of each trial can be chosen at least 10 seconds longer than the hold time to verify that the DUT can maintain a stable control plane when the data-forwarding plane is under stress.

3.6. Traffic Verification

Traffic verification is following section 10 of [RFC2544] and section 4.1.8 of [RFC5695]. The text is copied here for your convenience.

"The test equipment SHOULD discard any frames received during a test run that are not actual forwarded test frames. For example, keep-alive and routing update frames SHOULD NOT be included in the count of received frames. In all cases, sent traffic MUST be accounted

for, whether it was received on the wrong port, the correct port, or not received at all. In all cases, the test equipment SHOULD verify the length of the received frames and check that they match the expected length.

Preferably, the test equipment SHOULD include sequence numbers (or signature) in the transmitted frames and check for these numbers on the received frames. If this is done, the reported results SHOULD include in addition to the number of frames dropped, the number of frames that were received out of order, the number of duplicate frames received and the number of gaps in the received frame numbering sequence".

Many test tools may, by default, only verify that they have received the embedded signature on the receive side. However, some SRv6 tests assumes headers modifications (add or delete SRH, replace destination address, adjust "segments left"). All packets SHOULD be checked of the correct headers values on the receiving side.

3.7. Buffer tests

Back-to-back frame test was initially discussed in section 26.4 [RFC2544] and later improved in [RFC9004] which is considered the comprehensive reference for Back-to-back frame test. Modern forwarding engines are typically flexible in the buffer distribution between different interfaces. Hence, like for all other benchmarking tests, it is important to stress the forwarding engine on all interfaces. It should be necessary to perform throughput tests first because only frame sizes that stress DUT below wire-speed can be used for back-to-back tests. Buffers would be filled with the rate equal to the difference between the theoretical maximum frame rate (wire-speed) and DUT measured throughput for the respective frame size.

The test time could be much shorter than recommended in [RFC9004] because typical SR DUT is hardware-based with claimed buffers between 30ms to 100ms. It is better to consult with the vendor to find a good starting search point. If DUT is software-based then [RFC9004] recommendation for 2-30 seconds is applied.

Queuing SHOULD NOT have weighted random early detection (WRED) or any other mechanism that may start dropping packets before the buffer is filled. Queuing SHOULD be configured for the tail drop which is typically a non-default configuration. Back-to-back frame test is rather complex and expensive (50 runs for every frame size). Hence, it is OPTIONAL for SRv6.

4. Reporting Format

There are a few parameters that must be changed in section 5 of [RFC5695] for SRv6 tests.

Reporting parameter preserved from [RFC5695]:

* Throughput in bytes per second and frames per second

* Frame sizes in Octets (see Section 3.3)

* Interface speed (10/50/100/400/800/etc GE)

* Interface encapsulation (Ethernet or Ethernet VLAN)

* Interface media type (probably Ethernet)

Parameters changed from [RFC5695]:

* SRv6 Forwarding Operations (PUSH/ NEXT/ CONTINUE).

* Label Distribution protocol and IGP are the same in the context of SRv6. Hence, it can be called "Locator and Endpoint behaviors methods".

New parameters that MUST be reported are:

* Interface numbers involved for ingress and egress in the tests and their respective oversubscription ratio.

* Upstream/downstream traffic proportion (equal bidirectional or some other split).

* Number of Segments considered in the SRH.

* Behavior (H.Encaps, etc.) and Flavor (PSP, USP, USD) used for tests (according to [RFC8986]).

* SR Policy construction method (PCEP, BGP, manual configuration).

* Type of the payload (IPv6/IPv4, UDP/TCP).

* Time to recover from the overload state

* Time to recover from the reset state and reset type (particular module in reset)

* Tested buffers size in frames with respective frame size (for the optional back-to-back test); it is possible to record calculated buffer time for wire-speed throughput in milliseconds.

Some parameters may be the same for all tests (like Media type or Ethernet encapsulation) then it may be reported one time.

5. SRv6 Forwarding Benchmarking Tests

In general, tests are compliant with [RFC2544] but the important correction discussed in section 6 of [RFC2544] is applied: interfaces chosen for every test MUST stress all interfaces served by one forwarding engine. It is better to check the DUT specification for the relationship between interfaces and the forwarding engine to minimize the number of interfaces involved. But it is possible to understand the worst case by looking at the throughput and latency from the trial tests. If any doubt exists about how full is the offered load for the forwarding engine then it is better to stress all interfaces of the line card or all interfaces for the whole router with a centralized forwarding engine. A partial load on the forwarding engine would show optimistic results. Controllable traffic distribution between many interfaces (as specified in section 4 of [RFC5695]) would need separate SID announcements for separate interfaces.

The performance of modern packet forwarding engines may be huge that may need to involve many testers to sufficiently load the DUT as presented on figure 3. Then results correlation and recalculation of the real performance would be an additional burden.

```
                +----------+
        +-------|  Tester1 |<-------+
        | +-----|          |<-----+ |
        | |     +----------+      | |
        | |                       | |
        | |      +--------+       | |
        | +----->|        |-------+ |
        +------->|  DUT   |---------+
        +------->|        |---------+
        | +----->|        |-------+ |
        | |      +--------+       | |
        | |                       | |
        | |     +----------+      | |
        | +-----|  Tester2 |<-----+ |
        +-------|          |<-------+
                +----------+
```

Figure 3: Many testers

As specified in section 6 of [RFC5695], the traffic is sent from test tool Tx interface(s) to the DUT at a constant load for a fixed-time interval, and is received from the DUT on test tool Rx interface(s). If any frame loss is detected, then a new iteration is needed where the offered load is decreased and the sender will transmit again. An iterative search algorithm MUST be used to determine the maximum offered frame rate with a zero frame loss (No-Drop Rate - NDR). Each iteration should involve varying the offered load of the traffic, while keeping the other parameters (test duration, number of interfaces, number of addresses, frame size, etc.) constant, until the maximum rate at which none of the offered frames are dropped is determined.

The test can be repeated with a varying number of Segments pushed on ingress in order to measure the resulting maximum number. It can also be tested the maximum number of Segments that are correctly load-balanced in transit by only changing the Nth label in the stack and detect when load-balancing fails.

Therefore, the two main parameters that can be evaluated are:

Maximum offered frame rate,

Maximum number of Segments that can be pushed and hashed by the SR node for load-balancing.

5.1. Throughput

This section contains a description of the tests that are related to the characterization of a DUT's SRv6 traffic forwarding throughput.

The list of segments for SRv6 is represented as a list of IPv6 addresses, included in the SRH. There are three distinct types of nodes that are involved in segment routing networks that may represent four different cases.

5.1.1. Throughput of a Source Node

Objective: To obtain the DUT's Throughput during the packet processing of a Source Node. It is when the Source SR node, which corresponds to the headend node, encapsulates a received packet into an outer IPv6 packet and inserts the SR Header (SRH) as a Routing Extension Header in the outer IPv6 header. The Segment List in the SRH is composed of SIDs and the Source SR node sets the first SID of the SR Policy as the IPv6 Destination Address of the packet. The RECOMMENDED headend behavior is H.Encaps, in case of interest for another behavior (H.Encaps.Red or H.Encaps.L2 or H.Encaps.L2.Red) it is OPTIONAL to test it with proper reporting.

Procedure: Similar to section 6.1 of [RFC5695] or section 26.1 of [RFC2544] with extension to test SID list longer than 1 SID (more than 2 are RECOMMENDED). SID list can be from 1 to N SIDs. N could be specified a priori or measured as part of the test. The test tool must advertise and learn the IP prefix(es) and SID(s) on respective sides, as per Section 3.4, and must use one option for SID stack construction, as per Section 3.2, on its receive and transmit interfaces towards the DUT.

Reporting Format: A table, similar to [RFC5180], with all parameters specified in Section 4.

5.1.2. Throughput of a transit Segment Endpoint Node

Objective: To obtain the DUT's Throughput during the packet processing of a Segment Endpoint Node. It is when the SR Segment Endpoint node receives packets whose IPv6 destination address is locally configured as a segment. The SR Segment Endpoint node inspects the SR header: it detects the new active segment, i.e. the next segment in the Segment List, modifies the IPv6 destination address of the outer IPv6 header and forwards the packet on the basis of the IPv6 forwarding table. The RECOMMENDED endpoint behavior is End.X, in case of interest for another behavior (End, End.T, End.BM, End.B6.Encaps, End.B6.Encaps.Red) it is OPTIONAL to test it with proper reporting. SRH SL (Segment Left) is assumed to be bigger than zero for this test. Moreover, it is assumed that DUT would not need to delete headers (no PSP, USD, or USP).

Procedure: Similar to section 6.1 of [RFC5695] or section 26.1 of [RFC2544] with extension to test SID list longer than 1 SID (more than 2 are RECOMMENDED). SID list can be from 1 to N SIDs. N could be specified a priori or measured as part of the test. The test tool must advertise and learn the IP prefix(es) and SID(s) on respective sides, as per Section 3.4, and must use one option for SID stack construction, as per Section 3.2, on its receive and transmit interfaces towards the DUT.

Reporting Format: A table, similar to [RFC5180], with all parameters specified in Section 4.

5.1.3. Throughput of a Segment Endpoint Node with decapsulation

Objective: To obtain the DUT's Throughput during the packet processing of a Segment Endpoint Node that needs decapsulation. It is when the SR Segment Endpoint node receives packets whose IPv6 destination address is locally configured as a segment and SL (segment left) in the SRH header is decremented to zero. The SR Segment Endpoint node inspects the SR header: it detects the new

active segment, i.e. the next segment in the Segment List, modifies the IPv6 destination address of the outer IPv6 header, decapsulate the packet, and forwards the packet on the basis of the IPv6 forwarding table. The RECOMMENDED endpoint decapsulation behavior is End with USD flavor, in case of interest for another flavor (PSP, USP) it is OPTIONAL to test it with proper reporting.

Procedure: Similar to section 6.1 of [RFC5695] or section 26.1 of [RFC2544] with extension to test SID list longer than 1 SID (more than 2 are RECOMMENDED). SID list can be from 1 to N SIDs. N could be specified a priori or measured as part of the test. The test tool must advertise and learn the IP prefix(es) and SID(s) on respective sides, as per Section 3.4, and must use one option for SID stack construction, as per Section 3.2, on its receive and transmit interfaces towards the DUT.

Reporting Format: A table, similar to [RFC5180], with all parameters specified in Section 4.

5.1.4. Throughput of a Transit Node

Objective: To obtain the DUT's Throughput during the packet processing of a Transit Node. It is when a Transit node forwards the packet containing the SR header as a normal IPv6 packet because the IPv6 destination address does not locally match with a segment.

Procedure: Similar to section 6.1 of [RFC5695] or section 26.1 of [RFC2544] with extension to test SID list longer than 1 SID (more than 2 are RECOMMENDED). SID list can be from 1 to N SIDs. N could be specified a priori or measured as part of the test. The test tool must advertise and learn the IP prefix(es) and SID(s) on respective sides, as per Section 3.4, and must use one option for SID stack construction, as per Section 3.2, on its receive and transmit interfaces towards the DUT.

Reporting Format: A table, similar to [RFC5180], with all parameters specified in Section 4.

5.2. Buffers size

Back-to-back frame test is OPTIONAL and SHOULD be performed only after throughput tests because it SHOULD use only frame sizes that DUT is not capable to forward wire-speed, as explained in Section 3.7.

Objective: To determine the buffer size as defined in section 6 of [RFC9004] for each of the SRv6 forwarding operations.

Procedure: Should be inherited from [RFC9004] with 2 SIDs RECOMMENDED (many SIDs are possible). Despite the simple general idea for filling the buffer until tail drop, [RFC9004] has many details for procedure, precautions, and calculations that would be too lengthy to copy here.

Reporting Format: A table, similar to [RFC5180], with all parameters specified in Section 4.

5.3. Latency

Objective: To determine the latency as defined in section 6.2 of [RFC5695] and section 26.2 of [RFC2544] for each of the SRv6 forwarding operations. It is RECOMMENDED to test all four test types discussed in Section 5.1.

Procedure: Similar to Section 5.1. It is OPTIONAL to improve the procedure according to section 7.2 of [RFC8219] with calculations for typical and worst-case latency.

Reporting Format: A table, similar to [RFC5180], with all parameters specified in Section 4.

5.4. Frame Loss

Objective: To determine the frame-loss rate (as defined in section 6.3 of [RFC5695] or and section 26.3 of [RFC2544]) for each of the SRv6 forwarding operations of a DUT throughout the entire range of input data rates and frame sizes. The primary idea is to see what would be the frame loss under the overload conditions. It may be that overloaded forwarding engine would forward less traffic than in the situation close to the overload. Throughput may drop below the possible maximum. As for the section 26.3 of [RFC2544], it is RECOMMENDED to have the data for all tested frame sizes with 10% load step above the wire-speed throughput measured in Section 5.1. It is RECOMMENDED to test all four test types discussed in Section 5.1.

Procedure: Similar to Section 5.1.

Reporting Format: A table, similar to [RFC5180], with all parameters specified in Section 4.

5.5. System Recovery

Objective: To characterize the speed at which a DUT recovers from an overload condition for each of the SRv6 forwarding operations. It is RECOMMENDED to test all four test types discussed in Section 5.1.

Procedure: Similar to section 6.4 of [RFC5695] or section 26.5 of [RFC2544]. Send a stream of frames at a rate 110% of the recorded throughput rate or the maximum rate for the media, whichever is lower, for at least 60 seconds. At Timestamp A reduce the frame rate to 50% of the above rate and record the time of the last frame lost (Timestamp B). The system recovery time is determined by subtracting Timestamp B from Timestamp A. The test SHOULD be repeated a number of times and the average of the recorded values being reported.

Reporting Format: A table, similar to [RFC5180], with all parameters specified in Section 4.

5.6. Reset

All type of reset tests are OPTIONAL.

Objective: To characterize the speed at which a DUT recovers from a device or software reset for each of the SRv6 forwarding operations. According to section 1.3 of [RFC6201] it is possible to measure frame loss or time stamps (depending on the test tool capability). According to section 4 of [RFC6201] reset could be: 1) hardware, 2) software, or 3) power interruption. All resets may be partial, i.e. only for a particular part of hardware (line card) or software (module). Especial interest may be to test redundant power supplies or routing engines to make sure that reset does not affect the traffic. Hardware reset may be soft (command for reset) or hard (physical removal and insertion of the module). These types of reset SHOULD be treated as different. It is OPTIONAL to test all four test types discussed in Section 5.1, typically they would give the same result.

Procedure: It is inherited from [RFC6201] (see it for more details). It is simple in essence: create the traffic, initiate a reset, measure the time for the traffic lost.

Reporting Format: A table, similar to [RFC5180], with all parameters specified in Section 4.

6. Security Considerations

Benchmarking methodologies are limited to technology characterization in a laboratory environment, with dedicated address space and constraints. 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 production networks. The benchmarking network topology is 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.

There are no specific security considerations within the scope of this document.

7. IANA Considerations

This document has no IANA actions.

8. Acknowledgements

The authors would like to thank Al Morton, Gabor Lencse, Boris Khasanov for the precious comments and suggestions.

9. References

9.1. Normative References

[RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <https://www.rfc-editor.org/info/rfc1242>.

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

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

[RFC4814] Newman, D. and T. Player, "Hash and Stuffing: Overlooked Factors in Network Device Benchmarking", RFC 4814, DOI 10.17487/RFC4814, March 2007, <https://www.rfc-editor.org/info/rfc4814>.

[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>.

[RFC5695] Akhter, A., Asati, R., and C. Pignataro, "MPLS Forwarding Benchmarking Methodology for IP Flows", RFC 5695, DOI 10.17487/RFC5695, November 2009, <https://www.rfc-editor.org/info/rfc5695>.

[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>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <https://www.rfc-editor.org/info/rfc8986>.

9.2. Informative References

[ETSI-GR-NFV-TST-007]
ETSI, "ETSI GR NFV-TST 007: Network Functions Virtualisation (NFV) Release 3; Testing; Guidelines on Interoperability Testing for MANO", 2020, <https://www.etsi.org/deliver/etsi_gr/NFV-TST/001_099/007/03.01.01_60/gr_NFV-TST007v030101p.pdf>.

[I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Talaulikar, K., Mattes, P., and D. Jain, "Advertising Segment Routing Policies in BGP", Work in Progress, Internet-Draft, draft-ietf-idr-segment-routing-te-policy-25, 26 September 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-segment-routing-te-policy-25>.

[I-D.ietf-lsr-ospfv3-srv6-extensions]
Li, Z., Hu, Z., Talaulikar, K., and P. Psenak, "OSPFv3 Extensions for SRv6", Work in Progress, Internet-Draft, draft-ietf-lsr-ospfv3-srv6-extensions-15, 21 June 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-lsr-ospfv3-srv6-extensions-15>.

[I-D.ietf-rtgwg-segment-routing-ti-lfa]
Litkowski, S., Bashandy, A., Filsfils, C., Francois, P., Decraene, B., and D. Voyer, "Topology Independent Fast Reroute using Segment Routing", Work in Progress, Internet-Draft, draft-ietf-rtgwg-segment-routing-ti-lfa-11, 30 June 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-rtgwg-segment-routing-ti-lfa-11>.

[I-D.ietf-spring-srv6-srh-compression]
Cheng, W., Filsfils, C., Li, Z., Decraene, B., and F. Clad, "Compressed SRv6 Segment List Encoding in SRH", Work in Progress, Internet-Draft, draft-ietf-spring-srv6-srh-compression-08, 12 September 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-spring-srv6-srh-compression-08>.

[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>.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <https://www.rfc-editor.org/info/rfc6438>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8664] Sivabalan, S., Filsfils, C., Tantsura, J., Henderickx, W., and J. Hardwick, "Path Computation Element Communication Protocol (PCEP) Extensions for Segment Routing", RFC 8664, DOI 10.17487/RFC8664, December 2019, <https://www.rfc-editor.org/info/rfc8664>.

[RFC9004] Morton, A., "Updates for the Back-to-Back Frame Benchmark in RFC 2544", RFC 9004, DOI 10.17487/RFC9004, May 2021, <https://www.rfc-editor.org/info/rfc9004>.

[RFC9256] Filsfils, C., Talaulikar, K., Ed., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", RFC 9256, DOI 10.17487/RFC9256, July 2022, <https://www.rfc-editor.org/info/rfc9256>.

[RFC9352] Psenak, P., Ed., Filsfils, C., Bashandy, A., Decraene, B., and Z. Hu, "IS-IS Extensions to Support Segment Routing over the IPv6 Data Plane", RFC 9352, DOI 10.17487/RFC9352, February 2023, <https://www.rfc-editor.org/info/rfc9352>.

Authors' Addresses

Giuseppe Fioccola
Huawei Technologies
Palazzo Verrocchio, Centro Direzionale Milano 2
20054 Segrate (Milan)
Italy
Email: giuseppe.fioccola@huawei.com

Eduard Vasilenko
Huawei Technologies
17/4 Krylatskaya str.
Moscow
Email: vasilenko.eduard@huawei.com

Paolo Volpato
Huawei Technologies
Palazzo Verrocchio, Centro Direzionale Milano 2
20054 Segrate (Milan)
Italy
Email: paolo.volpato@huawei.com

Luis Miguel Contreras Murillo
Telefonica
Spain
Email: luismiguel.contrerasmurillo@telefonica.com

Bruno Decraene
Orange
France
Email: bruno.decraene@orange.com