

Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: December 20, 2021

S. Jacob, Ed.  
Independent  
K. Tiruveedhula  
Juniper Networks  
June 18, 2021

Benchmarking Methodology for EVPN and PBB-EVPN  
draft-ietf-bmwg-evpntest-09

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on December 20, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

## Table of Contents

|  |    |
|--|----|
| 1. Introduction . . . . .                                  | 3  |
| 1.1. Requirements Language . . . . .                       | 3  |
| 1.2. Terminologies . . . . .                               | 3  |
| 2. Test Topology . . . . .                                 | 4  |
| 3. Test Cases for EVPN Benchmarking . . . . .              | 7  |
| 3.1. Data Plane MAC Learning . . . . .                     | 7  |
| 3.2. Control Plane MAC Learning . . . . .                  | 8  |
| 3.3. MAC Flush-Local Link Failure and Relearning . . . . . | 9  |
| 3.4. MAC Flush-Remote Link Failure and Relearning. . . . . | 10 |
| 3.5. MAC Aging . . . . .                                   | 11 |
| 3.6. Remote MAC Aging . . . . .                            | 12 |
| 3.7. Control and Data plane MAC Learning . . . . .         | 12 |
| 3.8. High Availability. . . . .                            | 13 |
| 3.9. ARP/ND Scale . . . . .                                | 14 |
| 3.10. Scaling of Services . . . . .                        | 15 |
| 3.11. Scale Convergence . . . . .                          | 16 |
| 3.12. SOAK Test. . . . .                                   | 17 |
| 4. Test Cases for PBB-EVPN Benchmarking . . . . .          | 18 |
| 4.1. Data Plane Local MAC Learning . . . . .               | 18 |
| 4.2. Data Plane Remote MAC Learning . . . . .              | 18 |
| 4.3. MAC Flush-Local Link Failure . . . . .                | 19 |
| 4.4. MAC Flush-Remote Link Failure . . . . .               | 20 |
| 4.5. MAC Aging . . . . .                                   | 21 |
| 4.6. Remote MAC Aging. . . . .                             | 22 |
| 4.7. Local and Remote MAC Learning . . . . .               | 23 |
| 4.8. High Availability . . . . .                           | 23 |
| 4.9. Scale . . . . .                                       | 24 |
| 4.10. Scale Convergence . . . . .                          | 25 |
| 4.11. Soak Test . . . . .                                  | 26 |
| 5. Acknowledgments . . . . .                               | 27 |
| 6. IANA Considerations . . . . .                           | 27 |
| 7. Security Considerations . . . . .                       | 27 |
| 8. References . . . . .                                    | 27 |
| 8.1. Normative References . . . . .                        | 27 |
| 8.2. Informative References . . . . .                      | 28 |
| Appendix A. Appendix . . . . .                             | 28 |
| Authors' Addresses . . . . .                               | 28 |

## 1. Introduction

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

### 1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 8174 [RFC8174].

### 1.2. Terminologies

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

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

**AA:** All Active mode

**CE:** Customer Router/Devices/Switch.

**DF:** Designated Forwarder

**DUT:** Device under test.

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

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

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

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

**Interface:** Physical interface of a router/switch.

**IRB:** Integrated routing and bridging interface

**MAC:** Media Access Control addresses on a PE.

**MHPE2:** Multi homed Provider Edge router 2.

**MHPE1:** Multi homed Provider Edge router 1.

**SHPE3:** Single homed Provider Edge Router 3.

**PE:** Provider Edge device.

**P:** Provider Router.

**RR:** Route Reflector.

**RT:** Traffic Generator.

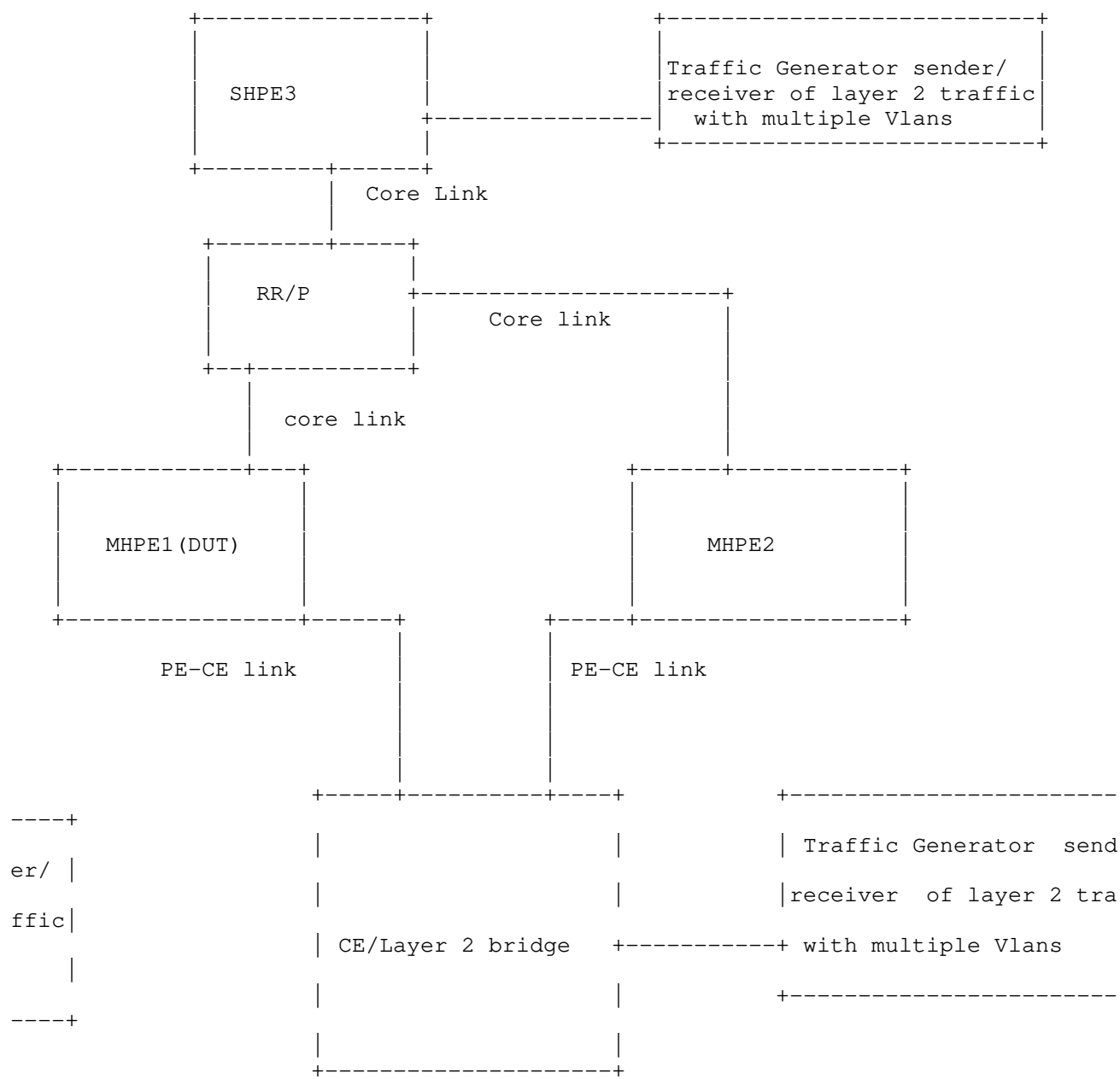
**Sub Interface:** Each physical Interfaces is subdivided in to a set of Logical units.

**SA:** Single Active

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

## 2. Test Topology

There are five routers in the Test setup. SHPE3, RR/P, MHPE1 and MHPE2 emulating a service provider network. CE is a customer device connected to MHPE1 and MHPE2. it is configured with bridge domains in multiple VLANs. The traffic generator is connected to the CE and SHPE3. The MHPE1 acts as DUT. The traffic generator will be used as sender and receiver of traffic. The test measurements are taken from the DUT. MHPE1 and MHPE2 are multi-homed routers connected to CE running single active mode. The traffic generator will be generating traffic at 10% of the line rate.



Topology 1

Test Setup

Figure 1

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

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

Figure 2

Test Setup Configurations:

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

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

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

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

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

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

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

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

The X below is used as variable to denote scale factor of the testing parameters. It must be in the multiples of 100.

### 3. Test Cases for EVPN Benchmarking

#### 3.1. Data Plane MAC Learning

Objective:

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

Topology : Topology 1

Procedure:

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

Confirm the DUT is up and running with EVPN.



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

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

The DUT will learn these X MAC in data plane.

Measurement :

Measure the time taken to learn X MAC locally in DUT EVPN MAC table. The data plane measurement is taken by considering DUT as black box. The range of MAC are known from traffic generator, the same must be learned in DUT, the time taken to learn X MAC is measured. The measurement is carried out using external server which polls the DUT using automated scripts which measures the MAC counter value with timestamp. Note at present there are no standard models to measure the counters.

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

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

### 3.2. Control Plane MAC Learning

Objective:

Measure the time taken to learn the control plane MAC.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

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

Ensure the frames must be destined to one EVPN instance.

The DUT will learn these X MAC in control plane.

#### Measurement :

Measure the time taken by the DUT to learn the X MAC in the data plane. The test is repeated for N times and the values are collected. The remote MAC learning rate is calculated by averaging the values obtained from N samples. N is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts which measures the MAC counter value with timestamp. Note at present there are no standard models to measure the counters.

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

### 3.3. MAC Flush-Local Link Failure and Relearning

#### Objective:

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

Topology : Topology 1

#### Procedure:

Confirm the DUT is up and running with EVPN.

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

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

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

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

The DUT and MHPE2 are running SA mode.

#### Measurement :

Measure the time taken for flushing these X MAC addresses. Measure the time taken to relearn these X MAC in DUT. The test is repeated for N times and the values are collected. The flush and the relearning time is calculated by averaging the values obtained by N samples. N is an arbitrary number to get a sufficient sample. The

time measured for each sample is denoted by  $T_1, T_2 \dots T_n$ . The measurement is carried out using external server which polls the DUT using automated scripts which measures the MAC counter value with timestamp.

Flush rate =  $(T_1 + T_2 + \dots T_n) / N$

Relearning rate =  $(T_1 + T_2 + \dots T_n) / N$

### 3.4. MAC Flush-Remote Link Failure and Relearning.

Objective:

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

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

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

Bring down the link between SHPE3 and traffic generator.

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

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

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

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

Measurement :

Measure the time taken to flush X remote MAC from EVPN MAC table of the DUT. Measure the time taken to relearn these X MAC in DUT. The test is repeated for N times and the values are collected. The flush rate is calculated by averaging the values obtained by N samples. N is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by  $T_1, T_2 \dots T_n$ . The measurement is carried

out using external server which polls the DUT using automated scripts which measures the MAC counter value with timestamp.

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

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

### 3.5. MAC Aging

Objective:

To measure the MAC Aging time.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

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

Ensure these X MAC addresses are learned in DUT.

Then stop the traffic.

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

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

The DUT and MHPE2 are running SA mode.

Measurement :

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

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

### 3.6. Remote MAC Aging

Objective:

Measure the control plane learned MAC Aging time.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

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

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

Then stop the traffic.

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

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

The DUT and MHPE2 are running SA mode.

Measurement :

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

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

### 3.7. Control and Data plane MAC Learning

Objective:

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

Topology : Topology 1

#### Procedure:

Confirm the DUT is up and running with EVPN.

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

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

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

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

DUT and MHPE2 are running in SA mode.

#### Measurement :

Measure the time taken to learn 2X MAC addresses in DUT EVPN MAC table. The test is repeated for N times and the values are collected. The MAC learning time is calculated by averaging the values obtained by N samples. N is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts which measures the MAC counter value with timestamp.

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

### 3.8. High Availability.

#### Objective:

Measure traffic loss during routing engine fail over.

Topology : Topology 1

#### Procedure:

Confirm the DUT is up and running with EVPN.

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

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

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

Ensure the DUT learn 2X MAC.

Then do a routing engine fail-over.

Measurement :

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

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

### 3.9. ARP/ND Scale

Measure the DUT scaling limit of ARP/ND.

Objective:

Measure the ARP/ND scale of the DUT.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

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

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

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

Measurement :

Measure the scale limit of type 2 MAC+ip/MAC+ipv6 route which DUT can learn. The test is repeated for N times and the values are collected. The scale limit is calculated by averaging the values obtained by N samples for both MAC+ip and MAC+ipv6. N is an arbitrary number to get a sufficient sample. The scale value obtained by each sample be v1,v2..vn for MAC+ipv4 and s1,s2..sn for MAC+ipv6. The measurement is carried out using external server which polls the DUT using automated scripts to find the scale limit of MAC+ipv4/MAC+ipv6.

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

Scale limit for MAC+ipv6 =  $(s1+s2+..sn)/N$

### 3.10. Scaling of Services

Objective:

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

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

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

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

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

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

Measurement :

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



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

### 3.11. Scale Convergence

#### Objective:

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

Topology : Topology 1

#### Procedure:

Confirm the DUT is up and running with EVPN.

Scale N EVIs in DUT, SHPE3 and MHPE2.

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

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

There will be 2X number of MAC addresses will be learned in DUT EVPN MAC table.

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

Then clear the BGP neighbours in the DUT.

Once the BGP session is in established state in DUT.

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

#### Measurement :

The DUT must learn 2X MAC addresses. Measure the time taken to learn 2X MAC in DUT. The test is repeated for N times and the values are collected. The convergence time is calculated by averaging the values obtained by N samples. N is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1, T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts which measures the MAC counter value with timestamp.

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

### 3.12. SOAK Test.

#### Objective:

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

Topology : Topology 1

#### Procedure:

Confirm the DUT is up and running with EVPN.

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

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

There will be 2X number of MAC addresses will be learned in DUT EVPN MAC table.

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

The DUT must run with traffic for 24 hours.

Every hour check for memory leak in EVPN process, CPU usage and crashes in DUT.

#### Measurement :

Take the hourly reading of CPU, process memory. There should not be any memory leak, crashes, CPU spikes. The CPU spike is determined as the sudden increase of CPU usage to 100 percent compared to the average usage. The average value vary from device to device. Memory leak is determined as the increase of memory usage with respect to time. The expectation is under steady state the memory usage for EVPN process should not increase with respect to time. The measurement is carried out using external server which polls the DUT using automated scripts which captures the CPU usage and process memory.

#### 4. Test Cases for PBB-EVPN Benchmarking

##### 4.1. Data Plane Local MAC Learning

Objective:

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

Topology : Topology 1

Procedure:

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

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

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

The DUT will learn these X MAC in data plane.

Measurement :

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

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

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

##### 4.2. Data Plane Remote MAC Learning

Objective:

To Record the time taken to learn the remote MAC.

Topology : Topology 1

#### Procedure:

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

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

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

The DUT will learn these X MAC in data plane.

#### Measurement :

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

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

### 4.3. MAC Flush-Local Link Failure

#### Objective:

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

Topology : Topology 1

#### Procedure:

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

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

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

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

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

The DUT and MHPE2 are running SA mode.

Measurement :

Measure the time taken for flushing these X MAC addresses. Measure the time taken to relearn these X MAC in DUT. The test is repeated for N times and the values are collected. The flush and the relearning time is calculated by averaging the values obtained by N samples. N is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts which measures the MAC counter value with timestamp.

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

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

#### 4.4. MAC Flush-Remote Link Failure

Objective:

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

Topology : Topology 1

Procedure:

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

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

Bring down the link between SHPE3 and traffic generator.

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

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

Measure time taken by the DUT to relearn these X MAC

**Measurement :**

Measure the time taken to flush X remote MAC from PBB-EVPN MAC table of the DUT. Measure the time taken to relearn these X MAC in DUT. The test is repeated for N times and the values are collected. The flush rate is calculated by averaging the values obtained by N samples. N is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts which measures the MAC counter value with timestamp.

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

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

**4.5. MAC Aging****Objective:**

Measure the MAC Aging time.

Topology : Topology 1

**Procedure:**

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

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

Ensure these X MAC addresses are learned in DUT.

Then stop the traffic.

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

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

The DUT and MHPE2 are running SA mode.

**Measurement :**

Measure the time taken to flush X MAC addresses due to aging. The test is repeated for N times and the values are collected. The aging

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

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

#### 4.6. Remote MAC Aging.

Objective:

Measure the remote MAC Aging time.

Topology : Topology 1

Procedure:

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

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

Ensure these X MAC addresses are learned in DUT.

Then stop the traffic.

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

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

The DUT and MHPE2 are running SA mode.

Measurement :

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

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

#### 4.7. Local and Remote MAC Learning

Objective:

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

Topology : Topology 1

Procedure:

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

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

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

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

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

DUT and MHPE2 are running in SA mode.

Measurement :

Measure the time taken to learn 2X MAC addresses in DUT PBB-EVPN MAC table. The test is repeated for N times and the values are collected. The MAC learning time is calculated by averaging the values obtained by N samples. N is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts which measures the MAC counter value with timestamp.

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

#### 4.8. High Availability

Objective:

Measure traffic loss during routing engine failover.

Topology : Topology 1

Procedure:



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

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

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

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

Ensure the DUT learn 2X MAC.

Then do a routing engine fail-over.

Measurement :

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

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

#### 4.9. Scale

Objective:

Measure the scale limit of DUT for PBB-EVPN.

Topology : Topology 1

Procedure:

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

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

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

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

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

Measurement :

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

#### 4.10. Scale Convergence

Objective:

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

Topology : Topology 1

Procedure:

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

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

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

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

There will be 2X number of MAC addresses will be learned in DUT PBB-EVPN MAC table.

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

Then clear the BGP neighbours in the DUT.

Once the BGP session is in established state in DUT.

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

Measurement :

The DUT must learn 2X MAC addresses. Measure the time taken to learn 2X MAC in DUT. The test is repeated for N times and the values are collected. The convergence time is calculated by averaging the values obtained by N samples. N is an arbitrary number to get a sufficient sample. The time measured for each sample is denoted by T1,T2...Tn. The measurement is carried out using external server which polls the DUT using automated scripts which measures the MAC counter value with timestamp.

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

#### 4.11. Soak Test

Objective:

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

Topology : Topology 1

Procedure:

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

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

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

There will be 2X number of MAC addresses will be learned in DUT PBB-EVPN MAC table.

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

The DUT must run with traffic for 24 hours.

Every hour check for memory leak in PBB-EVPN process, CPU usage and crashes in DUT.

Measurement :

Take the hourly reading of CPU, process memory. There should not be any memory leak, crashes, CPU spikes. The CPU spike is determined as the sudden increase of CPU usage to 100 percent compared to the average usage. The average value vary from device to device. Memory

leak is determined as the increase of memory usage with respect to time. The expectation is under steady state the memory usage for PBB-EVPN process should not increase with respect to time. The measurement is carried out using external server which polls the DUT using automated scripts which captures the CPU usage and process memory.

## 5. Acknowledgments

We would like to thank Fioccola Giuseppe of Telecom Italia reviewing our draft and commenting it. We would like to thank Sarah Banks for guiding and mentoring us. We take the opportunity to thank Al for reviewing our draft and gave us valuable comments.

## 6. IANA Considerations

This memo includes no request to IANA.

## 7. Security Considerations

The benchmarking tests described in this document are limited to the performance characterisation of controllers in a lab environment with isolated networks. The benchmarking network topology will be an independent test setup and MUST NOT be connected to devices that may forward the test traffic into a production network or misroute traffic to the test management network. Further, benchmarking is performed on a "black-box" basis, relying solely on measurements observable external to the controller. Security features mentioned in the RFC 7432 will affect the test results. Special capabilities SHOULD NOT exist in the controller specifically for benchmarking purposes. Any implications for network security arising from the controller SHOULD be identical in the lab and in production networks.

## 8. References

### 8.1. Normative References

- [RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <<https://www.rfc-editor.org/info/rfc2544>>.
- [RFC2899] Ginoza, S., "Request for Comments Summary RFC Numbers 2800-2899", RFC 2899, DOI 10.17487/RFC2899, May 2001, <<https://www.rfc-editor.org/info/rfc2899>>.

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

## 8.2. Informative References

[RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <<https://www.rfc-editor.org/info/rfc7432>>.

[RFC7623] Sajassi, A., Ed., Salam, S., Bitar, N., Isaac, A., and W. Henderickx, "Provider Backbone Bridging Combined with Ethernet VPN (PBB-EVPN)", RFC 7623, DOI 10.17487/RFC7623, September 2015, <<https://www.rfc-editor.org/info/rfc7623>>.

## Appendix A. Appendix

### Authors' Addresses

Sudhin Jacob (editor)  
Independent  
Bangalore  
India

Phone: +91 8061212543  
Email: [sudhinjacob@rediffmail.com](mailto:sudhinjacob@rediffmail.com)

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

Phone: +1 9785898861  
Email: [kishoret@juniper.net](mailto:kishoret@juniper.net)

Benchmarking Working Group  
Internet-Draft  
Intended status: Informational  
Expires: August 13, 2021

M. Konstantynowicz, Ed.  
V. Polak, Ed.  
Cisco Systems  
February 09, 2021

Multiple Loss Ratio Search for Packet Throughput (MLRsearch)  
draft-ietf-bmwg-mlrsearch-00

## Abstract

This document proposes changes to [RFC2544], specifically to packet throughput search methodology, by defining a new search algorithm referred to as Multiple Loss Ratio search (MLRsearch for short). Instead of relying on binary search with pre-set starting offered load, it proposes a novel approach discovering the starting point in the initial phase, and then searching for packet throughput based on defined packet loss ratio (PLR) input criteria and defined final trial duration time. One of the key design principles behind MLRsearch is minimizing the total test duration and searching for multiple packet throughput rates (each with a corresponding PLR) concurrently, instead of doing it sequentially.

The main motivation behind MLRsearch is the new set of challenges and requirements posed by NFV (Network Function Virtualization), specifically software based implementations of NFV data planes. Using [RFC2544] in the experience of the authors yields often not repetitive and not replicable end results due to a large number of factors that are out of scope for this draft. MLRsearch aims to address this challenge in a simple way of getting the same result sooner, so more repetitions can be done to describe the replicability.

## Status of This Memo

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

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

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

This Internet-Draft will expire on August 13, 2021.

## Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

|  |    |
|--|----|
| 1. Terminology . . . . .                     | 2  |
| 2. MLRsearch Background . . . . .            | 4  |
| 3. MLRsearch Overview . . . . .              | 5  |
| 4. Sample Implementation . . . . .           | 8  |
| 4.1. Input Parameters . . . . .              | 8  |
| 4.2. Initial Phase . . . . .                 | 9  |
| 4.3. Non-Initial Phases . . . . .            | 10 |
| 5. FD.io CSIT Implementation . . . . .       | 12 |
| 5.1. Additional details . . . . .            | 12 |
| 5.1.1. FD.io CSIT Input Parameters . . . . . | 14 |
| 5.2. Example MLRsearch Run . . . . .         | 14 |
| 6. IANA Considerations . . . . .             | 16 |
| 7. Security Considerations . . . . .         | 16 |
| 8. Acknowledgements . . . . .                | 17 |
| 9. References . . . . .                      | 17 |
| 9.1. Normative References . . . . .          | 17 |
| 9.2. Informative References . . . . .        | 17 |
| Authors' Addresses . . . . .                 | 17 |

## 1. Terminology

- o Frame size: size of an Ethernet Layer-2 frame on the wire, including any VLAN tags (dot1q, dot1ad) and Ethernet FCS, but excluding Ethernet preamble and inter-frame gap. Measured in bytes.
- o Packet size: same as frame size, both terms used interchangeably.

- o Device Under Test (DUT): In software networking, "device" denotes a specific piece of software tasked with packet processing. Such device is surrounded with other software components (such as operating system kernel). It is not possible to run devices without also running the other components, and hardware resources are shared between both. For purposes of testing, the whole set of hardware and software components is called "system under test" (SUT). As SUT is the part of the whole test setup performance of which can be measured by [RFC2544] methods, this document uses SUT instead of [RFC2544] DUT. Device under test (DUT) can be re-introduced when analysing test results using whitebox techniques, but this document sticks to blackbox testing.
- o System Under Test (SUT): System under test (SUT) is a part of the whole test setup whose performance is to be benchmarked. The complete test setup contains other parts, whose performance is either already established, or not affecting the benchmarking result.
- o Bi-directional throughput tests: involve packets/frames flowing in both transmit and receive directions over every tested interface of SUT/DUT. Packet flow metrics are measured per direction, and can be reported as aggregate for both directions and/or separately for each measured direction. In most cases bi-directional tests use the same (symmetric) load in both directions.
- o Uni-directional throughput tests: involve packets/frames flowing in only one direction, i.e. either transmit or receive direction, over every tested interface of SUT/DUT. Packet flow metrics are measured and are reported for measured direction.
- o Packet Loss Ratio (PLR): ratio of packets received relative to packets transmitted over the test trial duration, calculated using formula:  $PLR = (pkts\_transmitted - pkts\_received) / pkts\_transmitted$ . For bi-directional throughput tests aggregate PLR is calculated based on the aggregate number of packets transmitted and received.
- o Packet Throughput Rate: maximum packet offered load DUT/SUT forwards within the specified Packet Loss Ratio (PLR). In many cases the rate depends on the frame size processed by DUT/SUT. Hence packet throughput rate MUST be quoted with specific frame size as received by DUT/SUT during the measurement. For bi-directional tests, packet throughput rate should be reported as aggregate for both directions. Measured in packets-per-second (pps) or frames-per-second (fps), equivalent metrics.



- o Bandwidth Throughput Rate: a secondary metric calculated from packet throughput rate using formula:  $\text{bw\_rate} = \text{pkt\_rate} * (\text{frame\_size} + \text{L1\_overhead}) * 8$ , where L1\_overhead for Ethernet includes preamble (8 Bytes) and inter-frame gap (12 Bytes). For bi-directional tests, bandwidth throughput rate should be reported as aggregate for both directions. Expressed in bits-per-second (bps).
- o Non Drop Rate (NDR): maximum packet/bandwidth throughput rate sustained by DUT/SUT at PLR equal zero (zero packet loss) specific to tested frame size(s). MUST be quoted with specific packet size as received by DUT/SUT during the measurement. Packet NDR measured in packets-per-second (or fps), bandwidth NDR expressed in bits-per-second (bps).
- o Partial Drop Rate (PDR): maximum packet/bandwidth throughput rate sustained by DUT/SUT at PLR greater than zero (non-zero packet loss) specific to tested frame size(s). MUST be quoted with specific packet size as received by DUT/SUT during the measurement. Packet PDR measured in packets-per-second (or fps), bandwidth PDR expressed in bits-per-second (bps).
- o Maximum Receive Rate (MRR): packet/bandwidth rate regardless of PLR sustained by DUT/SUT under specified Maximum Transmit Rate (MTR) packet load offered by traffic generator. MUST be quoted with both specific packet size and MTR as received by DUT/SUT during the measurement. Packet MRR measured in packets-per-second (or fps), bandwidth MRR expressed in bits-per-second (bps).
- o Trial: a single measurement step. See [RFC2544] section 23.
- o Trial duration: amount of time over which packets are transmitted in a single measurement step.

## 2. MLRsearch Background

Multiple Loss Ratio search (MLRsearch) is a packet throughput search algorithm suitable for deterministic systems (as opposed to probabilistic systems). MLRsearch discovers multiple packet throughput rates in a single search, with each rate associated with a distinct Packet Loss Ratio (PLR) criteria.

For cases when multiple rates need to be found, this property makes MLRsearch more efficient in terms of time execution, compared to traditional throughput search algorithms that discover a single packet rate per defined search criteria (e.g. a binary search specified by [RFC2544]). MLRsearch reduces execution time even further by relying on shorter trial durations of intermediate steps,

with only the final measurements conducted at the specified final trial duration. This results in the shorter overall search execution time when compared to a traditional binary search, while guaranteeing the same results for deterministic systems.

In practice two rates with distinct PLRs are commonly used for packet throughput measurements of NFV systems: Non Drop Rate (NDR) with  $PLR=0$  and Partial Drop Rate (PDR) with  $PLR>0$ . The rest of this document describes MLRsearch for NDR and PDR. If needed, MLRsearch can be adapted to discover more throughput rates with different pre-defined PLRs.

Similarly to other throughput search approaches like binary search, MLRsearch is effective for SUTs/DUTs with PLR curve that is continuously flat or increasing with growing offered load. It may not be as effective for SUTs/DUTs with abnormal PLR curves.

MLRsearch relies on traffic generator to qualify the received packet stream as error-free, and invalidate the results if any disqualifying errors are present e.g. out-of-sequence frames.

MLRsearch can be applied to both uni-directional and bi-directional throughput tests.

For bi-directional tests, MLRsearch rates and ratios are aggregates of both directions, based on the following assumptions:

- o Traffic transmitted by traffic generator and received by SUT/DUT has the same packet rate in each direction, in other words the offered load is symmetric.
- o SUT/DUT packet processing capacity is the same in both directions, resulting in the same packet loss under load.

### 3. MLRsearch Overview

The main properties of MLRsearch:

- o MLRsearch is a duration aware multi-phase multi-rate search algorithm:
  - \* Initial Phase determines promising starting interval for the search.
  - \* Intermediate Phases progress towards defined final search criteria.

- \* Final Phase executes measurements according to the final search criteria.
- \* Final search criteria are defined by following inputs:
  - + PLRs associated with NDR and PDR.
  - + Final trial duration.
  - + Measurement resolution.
- o Initial Phase:
  - \* Measure MRR over initial trial duration.
  - \* Measured MRR is used as an input to the first intermediate phase.
- o Multiple Intermediate Phases:
  - \* Trial duration:
    - + Start with initial trial duration in the first intermediate phase.
    - + Converge geometrically towards the final trial duration.
  - \* Track two values for NDR and two for PDR:
    - + The values are called lower\_bound and upper\_bound.
    - + Each value comes from a specific trial measurement:
      - Most recent for that transmit rate.
      - As such the value is associated with that measurement's duration and loss.
    - + A bound can be valid or invalid:
      - Valid lower\_bound must conform with PLR search criteria.
      - Valid upper\_bound must not conform with PLR search criteria.
      - Example of invalid NDR lower\_bound is if it has been measured with non-zero loss.

- Invalid bounds are not real boundaries for the searched value:
    - o They are needed to track interval widths.
  - Valid bounds are real boundaries for the searched value.
  - Each non-initial phase ends with all bounds valid.
  - Bound can become invalid if it re-measured at a longer trial duration in a sub-sequent phase.
- \* Search:
- + Start with a large (lower\_bound, upper\_bound) interval width, that determines measurement resolution.
  - + Geometrically converge towards the width goal of the phase.
  - + Each phase halves the previous width goal.
    - First measurement of the next phase will be internal search which always gives a valid bound and brings the width to the new goal.
    - Only one bound then needs to be re-measured with new duration.
- \* Use of internal and external searches:
- + External search:
    - Measures at transmit rates outside the (lower\_bound, upper\_bound) interval.
    - Activated when a bound is invalid, to search for a new valid bound by multiplying (for example doubling) the interval width.
    - It is a variant of "exponential search".
  - + Internal search:
    - A "binary search" that measures at transmit rates within the (lower\_bound, upper\_bound) valid interval, halving the interval width.
- o Final Phase:

- \* Executed with the final test trial duration, and the final width goal that determines resolution of the overall search.
- o Intermediate Phases together with the Final Phase are called Non-Initial Phases.

The main benefits of MLRsearch vs. binary search include:

- o In general MLRsearch is likely to execute more trials overall, but likely less trials at a set final trial duration.
- o In well behaving cases, e.g. when results do not depend on trial duration, it greatly reduces (>50%) the overall duration compared to a single PDR (or NDR) binary search over duration, while finding multiple drop rates.
- o In all cases MLRsearch yields the same or similar results to binary search.
- o Note: both binary search and MLRsearch are susceptible to reporting non-repeatable results across multiple runs for very bad behaving cases.

Caveats:

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

#### 4. Sample Implementation

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

##### 4.1. Input Parameters

1. \*maximum\_transmit\_rate\* - Maximum Transmit Rate (MTR) of packets to be used by external traffic generator implementing MLRsearch, limited by the actual Ethernet link(s) rate, NIC model or traffic generator capabilities.
2. \*minimum\_transmit\_rate\* - minimum packet transmit rate to be used for measurements. MLRsearch fails if lower transmit rate needs to be used to meet search criteria.
3. \*final\_trial\_duration\* - required trial duration for final rate measurements.

4. *\*initial\_trial\_duration\** - trial duration for initial MLRsearch phase.
5. *\*final\_relative\_width\** - required measurement resolution expressed as (lower\_bound, upper\_bound) interval width relative to upper\_bound.
6. *\*packet\_loss\_ratio\** - maximum acceptable PLR search criterion for PDR measurements.
7. *\*number\_of\_intermediate\_phases\** - number of phases between the initial phase and the final phase. Impacts the overall MLRsearch duration. Less phases are required for well behaving cases, more phases may be needed to reduce the overall search duration for worse behaving cases.

#### 4.2. Initial Phase

1. First trial measures at configured maximum transmit rate (MTR) and discovers maximum receive rate (MRR).
  - \* IN: trial\_duration = initial\_trial\_duration.
  - \* IN: offered\_transmit\_rate = maximum\_transmit\_rate.
  - \* DO: single trial.
  - \* OUT: measured loss ratio.
  - \* OUT: MRR = measured receive rate. If loss ratio is zero, MRR is set below MTR so that interval width is equal to the width goal of the first intermediate phase.
2. Second trial measures at MRR and discovers MRR2.
  - \* IN: trial\_duration = initial\_trial\_duration.
  - \* IN: offered\_transmit\_rate = MRR.
  - \* DO: single trial.
  - \* OUT: measured loss ratio.
  - \* OUT: MRR2 = measured receive rate. If loss ratio is zero, MRR2 is set above MRR so that interval width is equal to the width goal of the first intermediate phase. MRR2 could end up being equal to MTR (for example if both measurements so far

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

3. Third trial measures at MRR2.

- \* IN: trial\_duration = initial\_trial\_duration.
- \* IN: offered\_transmit\_rate = MRR2.
- \* DO: single trial.
- \* OUT: measured loss ratio.

4.3. Non-Initial Phases

1. Main loop:

1. IN: trial\_duration for the current phase. Set to initial\_trial\_duration for the first intermediate phase; to final\_trial\_duration for the final phase; or to the element of interpolating geometric sequence for other intermediate phases. For example with two intermediate phases, trial\_duration of the second intermediate phase is the geometric average of initial\_trial\_duration and final\_trial\_duration.
2. IN: relative\_width\_goal for the current phase. Set to final\_relative\_width for the final phase; doubled for each preceding phase. For example with two intermediate phases, the first intermediate phase uses quadruple of final\_relative\_width and the second intermediate phase uses double of final\_relative\_width.
3. IN: ndr\_interval, pdr\_interval from the previous main loop iteration or the previous phase. If the previous phase is the initial phase, both intervals are formed by a (correctly ordered) pair of MRR2 and MRR. Note that the initial phase is likely to create intervals with invalid bounds.
4. DO: According to the procedure described in point 2., either exit the phase (by jumping to 1.7.), or calculate new transmit rate to measure with.
5. DO: Perform the trial measurement at the new transmit rate and trial\_duration, compute its loss ratio.
6. DO: Update the bounds of both intervals, based on the new measurement. The actual update rules are numerous, as NDR

external search can affect PDR interval and vice versa, but the result agrees with rules of both internal and external search. For example, any new measurement below an invalid lower\_bound becomes the new lower\_bound, while the old measurement (previously acting as the invalid lower\_bound) becomes a new and valid upper\_bound. Go to next iteration (1.3.), taking the updated intervals as new input.

7. OUT: current ndr\_interval and pdr\_interval. In the final phase this is also considered to be the result of the whole search. For other phases, the next phase loop is started with the current results as an input.

2. New transmit rate (or exit) calculation (for point 1.4.):

1. If there is an invalid bound then prepare for external search:
  - + IF the most recent measurement at NDR lower\_bound transmit rate had the loss higher than zero, then the new transmit rate is NDR lower\_bound decreased by two NDR interval widths.
  - + Else, IF the most recent measurement at PDR lower\_bound transmit rate had the loss higher than PLR, then the new transmit rate is PDR lower\_bound decreased by two PDR interval widths.
  - + Else, IF the most recent measurement at NDR upper\_bound transmit rate had no loss, then the new transmit rate is NDR upper\_bound increased by two NDR interval widths.
  - + Else, IF the most recent measurement at PDR upper\_bound transmit rate had the loss lower or equal to PLR, then the new transmit rate is PDR upper\_bound increased by two PDR interval widths.
2. Else, if interval width is higher than the current phase goal:
  - + IF NDR interval does not meet the current phase width goal, prepare for internal search. The new transmit rate is a in the middle of NDR lower\_bound and NDR upper\_bound.
  - + IF PDR interval does not meet the current phase width goal, prepare for internal search. The new transmit rate is a in the middle of PDR lower\_bound and PDR upper\_bound.



3. Else, if some bound has still only been measured at a lower duration, prepare to re-measure at the current duration (and the same transmit rate). The order of priorities is:
  - + NDR lower\_bound,
  - + PDR lower\_bound,
  - + NDR upper\_bound,
  - + PDR upper\_bound.
4. Else, do not prepare any new rate, to exit the phase. This ensures that at the end of each non-initial phase all intervals are valid, narrow enough, and measured at current phase trial duration.

## 5. FD.io CSIT Implementation

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

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

### 5.1. Additional details

This document so far has been describing a simplified version of MLRsearch algorithm. The full algorithm as implemented in CSIT contains additional logic, which makes some of the details (but not general ideas) above incorrect. Here is a short description of the additional logic as a list of principles, explaining their main differences from (or additions to) the simplified description, but without detailing their mutual interaction.

#### 1. Logarithmic transmit rate.

- \* In order to better fit the relative width goal, the interval doubling and halving is done differently.
- \* For example, the middle of 2 and 8 is 4, not 5.

#### 2. Optimistic maximum rate.

- \* The increased rate is never higher than the maximum rate.
- \* Upper bound at that rate is always considered valid.

3. Pessimistic minimum rate.

- \* The decreased rate is never lower than the minimum rate.
- \* If a lower bound at that rate is invalid, a phase stops refining the interval further (until it gets re-measured).

4. Conservative interval updates.

- \* Measurements above the current upper bound never update a valid upper bound, even if drop ratio is low.
- \* Measurements below the current lower bound always update any lower bound if drop ratio is high.

5. Ensure sufficient interval width.

- \* Narrow intervals make external search take more time to find a valid bound.
- \* If the new transmit increased or decreased rate would result in width less than the current goal, increase/decrease more.
- \* This can happen if the measurement for the other interval makes the current interval too narrow.
- \* Similarly, take care the measurements in the initial phase create wide enough interval.

6. Timeout for bad cases.

- \* The worst case for MLRsearch is when each phase converges to intervals way different than the results of the previous phase.
- \* Rather than suffer total search time several times larger than pure binary search, the implemented tests fail themselves when the search takes too long (given by argument `_timeout_`).

7. Pessimistic external search.

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

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

#### 5.1.1. FD.io CSIT Input Parameters

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

#### 5.2. Example MLRsearch Run

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

# Internet-DraftMultiple Loss Ratio Search for Packet ThroughFebruary 2021

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

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

# Internet-Draft Multiple Loss Ratio Search for Packet Through February 2021

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

## 6. IANA Considerations

No requests of IANA.

## 7. Security Considerations

Benchmarking activities as described in this memo are limited to technology characterization of a DUT/SUT using controlled stimuli in a laboratory environment, with dedicated address space and the constraints specified in the sections above.

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

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

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

## 8. Acknowledgements

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

## 9. References

### 9.1. Normative References

- [RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <<https://www.rfc-editor.org/info/rfc2544>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

### 9.2. Informative References

- [FDio-CSIT-MLRsearch] "FD.io CSIT Test Methodology - MLRsearch", February 2020, <[https://docs.fd.io/csit/rls2001/report/introduction/methodology\\_data\\_plane\\_throughput/methodology\\_mlrsearch\\_tests.html](https://docs.fd.io/csit/rls2001/report/introduction/methodology_data_plane_throughput/methodology_mlrsearch_tests.html)>.
- [PyPI-MLRsearch] "MLRsearch 0.3.0, Python Package Index", February 2020, <<https://pypi.org/project/MLRsearch/0.3.0/>>.

## Authors' Addresses

Maciek Konstantynowicz (editor)  
Cisco Systems

Email: [mkonstan@cisco.com](mailto:mkonstan@cisco.com)

Internet-Draft Multiple Loss Ratio Search for Packet Through February 2021

Vratko Polak (editor)  
Cisco Systems

Email: [vrpolak@cisco.com](mailto:vrpolak@cisco.com)

Benchmarking Working Group  
Internet-Draft  
Intended status: Informational  
Expires: January 13, 2022

M. Konstantynowicz, Ed.  
V. Polak, Ed.  
Cisco Systems  
July 12, 2021

Multiple Loss Ratio Search for Packet Throughput (MLRsearch)  
draft-ietf-bmwg-mlrsearch-01

## Abstract

This document proposes changes to [RFC2544], specifically to packet throughput search methodology, by defining a new search algorithm referred to as Multiple Loss Ratio search (MLRsearch for short). Instead of relying on binary search with pre-set starting offered load, it proposes a novel approach discovering the starting point in the initial phase, and then searching for packet throughput based on defined packet loss ratio (PLR) input criteria and defined final trial duration time. One of the key design principles behind MLRsearch is minimizing the total test duration and searching for multiple packet throughput rates (each with a corresponding PLR) concurrently, instead of doing it sequentially.

The main motivation behind MLRsearch is the new set of challenges and requirements posed by NFV (Network Function Virtualization), specifically software based implementations of NFV data planes. Using [RFC2544] in the experience of the authors yields often not repetitive and not replicable end results due to a large number of factors that are out of scope for this draft. MLRsearch aims to address this challenge in a simple way of getting the same result sooner, so more repetitions can be done to describe the replicability.

## Status of This Memo

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

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

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



This Internet-Draft will expire on January 13, 2022.

## Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

|  |    |
|--|----|
| 1. Terminology . . . . .                     | 2  |
| 2. MLRsearch Background . . . . .            | 5  |
| 3. MLRsearch Overview . . . . .              | 6  |
| 4. Sample Implementation . . . . .           | 9  |
| 4.1. Input Parameters . . . . .              | 9  |
| 4.2. Initial Phase . . . . .                 | 10 |
| 4.3. Non-Initial Phases . . . . .            | 11 |
| 5. FD.io CSIT Implementation . . . . .       | 15 |
| 5.1. Additional details . . . . .            | 15 |
| 5.1.1. FD.io CSIT Input Parameters . . . . . | 17 |
| 5.2. Example MLRsearch Run . . . . .         | 18 |
| 6. IANA Considerations . . . . .             | 20 |
| 7. Security Considerations . . . . .         | 20 |
| 8. Acknowledgements . . . . .                | 20 |
| 9. References . . . . .                      | 20 |
| 9.1. Normative References . . . . .          | 21 |
| 9.2. Informative References . . . . .        | 21 |
| Authors' Addresses . . . . .                 | 21 |

## 1. Terminology

- o Frame size: size of an Ethernet Layer-2 frame on the wire, including any VLAN tags (dot1q, dot1ad) and Ethernet FCS, but excluding Ethernet preamble and inter-frame gap. Measured in bytes (octets).
- o Packet size: same as frame size, both terms used interchangeably.

- o Device Under Test (DUT): In software networking, "device" denotes a specific piece of software tasked with packet processing. Such device is surrounded with other software components (such as operating system kernel). It is not possible to run devices without also running the other components, and hardware resources are shared between both. For purposes of testing, the whole set of hardware and software components is called "system under test" (SUT). As SUT is the part of the whole test setup performance of which can be measured by [RFC2544] methods, this document uses SUT instead of [RFC2544] DUT. Device under test (DUT) can be re-introduced when analysing test results using whitebox techniques, but this document sticks to blackbox testing.
- o System Under Test (SUT): System under test (SUT) is a part of the whole test setup whose performance is to be benchmarked. The complete test setup contains other parts, whose performance is either already established, or not affecting the benchmarking result.
- o Bi-directional throughput tests: involve packets/frames flowing in both transmit and receive directions over every tested interface of SUT/DUT. Packet flow metrics are measured per direction, and can be reported as aggregate for both directions and/or separately for each measured direction. In most cases bi-directional tests use the same (symmetric) load in both directions.
- o Uni-directional throughput tests: involve packets/frames flowing in only one direction, i.e. either transmit or receive direction, over every tested interface of SUT/DUT. Packet flow metrics are measured and are reported for measured direction.
- o Packet Loss Ratio (PLR): ratio of packets received relative to packets transmitted over the test trial duration, calculated using formula:  $PLR = (pkts\_transmitted - pkts\_received) / pkts\_transmitted$ . For bi-directional throughput tests aggregate PLR is calculated based on the aggregate number of packets transmitted and received.
- o Effective loss ratio: A corrected value of measured packet loss ratio chosen to avoid difficulties if SUT exhibits decreasing loss with increasing load. Maximum of packet loss ratios measured at the same duration on all loads smaller than (and including) the current one.
- o Target loss ratio: A packet loss ratio value acting as an input for search. The search is finding tight enough lower and upper bound in intended load, so that the lower bound has smaller or equal loss ratio, and upper bound has strictly larger loss ratio.

For the tightest upper bound, the effective loss ratio is the same as packet loss ratio. For the tightest lower bound, the effective loss ratio can be higher than the packet loss ratio, but still not larger than the target loss ratio.

- o Packet Throughput Rate: maximum packet offered load DUT/SUT forwards within the specified Packet Loss Ratio (PLR). In many cases the rate depends on the frame size processed by DUT/SUT. Hence packet throughput rate MUST be quoted with specific frame size as received by DUT/SUT during the measurement. For bi-directional tests, packet throughput rate should be reported as aggregate for both directions. Measured in packets-per-second (pps) or frames-per-second (fps), equivalent metrics.
- o Bandwidth Throughput Rate: a secondary metric calculated from packet throughput rate using formula:  $bw\_rate = pkt\_rate * (frame\_size + L1\_overhead) * 8$ , where L1\_overhead for Ethernet includes preamble (8 octets) and inter-frame gap (12 octets). For bi-directional tests, bandwidth throughput rate should be reported as aggregate for both directions. Expressed in bits-per-second (bps).
- o Non Drop Rate (NDR): maximum packet/bandwidth throughput rate sustained by DUT/SUT at PLR equal zero (zero packet loss) specific to tested frame size(s). MUST be quoted with specific packet size as received by DUT/SUT during the measurement. Packet NDR measured in packets-per-second (or fps), bandwidth NDR expressed in bits-per-second (bps).
- o Partial Drop Rate (PDR): maximum packet/bandwidth throughput rate sustained by DUT/SUT at PLR greater than zero (non-zero packet loss) specific to tested frame size(s). MUST be quoted with specific packet size as received by DUT/SUT during the measurement. Packet PDR measured in packets-per-second (or fps), bandwidth PDR expressed in bits-per-second (bps).
- o Maximum Receive Rate (MRR): packet/bandwidth rate regardless of PLR sustained by DUT/SUT under specified Maximum Transmit Rate (MTR) packet load offered by traffic generator. MUST be quoted with both specific packet size and MTR as received by DUT/SUT during the measurement. Packet MRR measured in packets-per-second (or fps), bandwidth MRR expressed in bits-per-second (bps).
- o Trial: a single measurement step. See [RFC2544] section 23.
- o Trial duration: amount of time over which packets are transmitted in a single measurement step.

## 2. MLRsearch Background

Multiple Loss Ratio search (MLRsearch) is a packet throughput search algorithm suitable for deterministic systems (as opposed to probabilistic systems). MLRsearch discovers multiple packet throughput rates in a single search, 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:

- o Traffic transmitted by traffic generator and received by SUT/DUT has the same packet rate in each direction, in other words the offered load is symmetric.
- o SUT/DUT packet processing capacity is the same in both directions, resulting in the same packet loss under load.

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.

### 3. MLRsearch Overview

The main properties of MLRsearch:

- o MLRsearch is a duration aware multi-phase multi-rate search algorithm:
  - \* Initial Phase determines promising starting interval for the search.
  - \* Intermediate Phases progress towards defined final search criteria.
  - \* Final Phase executes measurements according to the final search criteria.
  - \* Final search criteria are defined by following inputs:
    - + Target PLRs (e.g. 0.0 and 0.005 when searching for NDR and PDR).
    - + Final trial duration.
    - + Measurement resolution.
- o Initial Phase:
  - \* Measure MRR over initial trial duration.
  - \* Measured MRR is used as an input to the first intermediate phase.
- o Multiple Intermediate Phases:
  - \* Trial duration:
    - + Start with initial trial duration in the first intermediate phase.
    - + Converge geometrically towards the final trial duration.

- \* Track all previous trial measurement results:
  - + Duration, offered load and loss ratio are tracked.
  - + 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.
  - + The algorithm queries the results to find best lower and upper bounds.
    - Effective loss ratios are always used.
  - + The phase ends if all target loss ratios have tight enough bounds.
- \* Search:
  - + Iterate over target loss ratios in increasing order.
  - + 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.
  - + 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.
  - + 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.
  - + The logic guarantees that a measurement is never repeated with both duration and offered load being the same.
  - + 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:

- + 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).
- + 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.
    - o The idea is to avoid returning interval narrower than the current width goal. See sample implementation details, below.
- o Final Phase:
  - \* Executed with the final test trial duration, and the final width goal that determines resolution of the overall search.
- o Intermediate Phases together with the Final Phase are called Non-Initial Phases.
- o The returned bounds stay within prescribed min\_rate and max\_rate.
  - \* When returning min\_rate or max\_rate, the returned bounds may be invalid.
    - + 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:

- o In general MLRsearch is likely to execute more trials overall, but likely less trials at a set final trial duration.
- o In well behaving cases, e.g. when results do not depend on trial duration, it greatly reduces (>50%) the overall duration compared to a single PDR (or NDR) binary search over duration, while finding multiple drop rates.
- o In all cases MLRsearch yields the same or similar results to binary search.

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

Caveats:

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

#### 4. Sample Implementation

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

##### 4.1. Input Parameters

1. *\*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.

#### 4.2. Initial Phase

1. First trial measures at configured maximum transmit rate (MTR) and discovers maximum receive rate (MRR).
  - \* IN: trial\_duration = initial\_trial\_duration.
  - \* IN: offered\_transmit\_rate = maximum\_transmit\_rate.
  - \* DO: single trial.
  - \* OUT: measured loss ratio.
  - \* OUT: MRR = measured receive rate. 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.

4.3. 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 bisection (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.

## 5. FD.io CSIT Implementation

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

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

### 5.1. Additional details

This document so far has been describing a simplified version of MLRsearch algorithm. The full algorithm as implemented in CSIT contains additional logic, which makes some of the details (but not general ideas) above incorrect. Here is a short description of the additional logic as a list of principles, explaining their main differences from (or additions to) the simplified description, but without detailing their mutual interaction.

1. Logarithmic transmit rate.

- \* In order to better fit the relative width goal, the interval doubling and halving is done differently.
- \* For example, the middle of 2 and 8 is 4, not 5.

2. 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.
    - Example: If a successful transaction uses 10 packets, load is given in transactions per second, but loss ratio is calculated from packets, 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.

5.1.1. 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 quadrupled). 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.

## 5.2. Example MLRsearch Run

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

- o 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).

- o 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).

- o Both intervals are narrow enough.
- o 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).

- o 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).

- o 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).

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

- o 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).

- o 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).

- o PDR interval is also narrow enough.
- o Returning bounds:
- o NDR\_LOWER = 5112894.3238511775 pps; NDR\_UPPER = 5138587.208637197 pps;
- o PDR\_LOWER = 5190360.904111567 pps; PDR\_UPPER = 5216443.04126728 pps.

#### 6. IANA Considerations

No requests of IANA.

#### 7. Security Considerations

Benchmarking activities as described in this memo are limited to technology characterization of a DUT/SUT using controlled stimuli in a laboratory environment, with dedicated address space and the constraints specified in the sections above.

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

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

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

#### 8. Acknowledgements

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

#### 9. References

## 9.1. Normative References

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

## 9.2. Informative References

[FDio-CSIT-MLRsearch]  
"FD.io CSIT Test Methodology - MLRsearch", February 2021, <[https://docs.fd.io/csit/rls2101/report/introduction/methodology\\_data\\_plane\\_throughput/methodology\\_mlrsearch\\_tests.html](https://docs.fd.io/csit/rls2101/report/introduction/methodology_data_plane_throughput/methodology_mlrsearch_tests.html)>.

[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](mailto:mkonstan@cisco.com)

Vratko Polak (editor)  
Cisco Systems

Email: [vrpolak@cisco.com](mailto:vrpolak@cisco.com)

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 . . . . .                               | 3  |
| 2. Intentions of this document . . . . .               | 5  |
| 3. RFC2544 . . . . .                                   | 5  |
| 3.1. Throughput search . . . . .                       | 5  |
| 4. Problems . . . . .                                  | 6  |
| 4.1. Repeatability and Comparability . . . . .         | 6  |
| 4.2. Non-Zero Target Loss Ratios . . . . .             | 6  |
| 5. Solution ideas . . . . .                            | 7  |
| 5.1. Short duration trials . . . . .                   | 8  |
| 5.2. FRMOL as reasonable start . . . . .               | 8  |
| 5.3. Non-zero loss ratios . . . . .                    | 8  |
| 5.4. Concurrent ratio search . . . . .                 | 9  |
| 5.5. Load selection heuristics and shortcuts . . . . . | 9  |
| 6. Non-compliance with RFC2544 . . . . .               | 9  |
| 7. Additional Requirements . . . . .                   | 10 |
| 7.1. TODO: Search Stop Criteria . . . . .              | 10 |
| 7.2. Reliability of Test Equipment . . . . .           | 10 |
| 7.2.1. Very late frames . . . . .                      | 10 |
| 8. MLRsearch Background . . . . .                      | 11 |
| 9. MLRsearch Overview . . . . .                        | 13 |
| 10. Sample Implementation . . . . .                    | 16 |
| 10.1. Input Parameters . . . . .                       | 16 |
| 10.2. Initial Phase . . . . .                          | 17 |
| 10.3. Non-Initial Phases . . . . .                     | 18 |
| 11. FD.io CSIT Implementation . . . . .                | 22 |

|   |    |
|---|----|
| 11.1. Additional details . . . . .            | 22 |
| 11.1.1. FD.io CSIT Input Parameters . . . . . | 24 |
| 11.2. Example MLRsearch Run . . . . .         | 25 |
| 12. IANA Considerations . . . . .             | 27 |
| 13. Security Considerations . . . . .         | 27 |
| 14. Acknowledgements . . . . .                | 27 |
| 15. References . . . . .                      | 27 |
| 15.1. Normative References . . . . .          | 27 |
| 15.2. Informative References . . . . .        | 28 |
| Authors' Addresses . . . . .                  | 28 |

## 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 correctly 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 larger 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 overall search time".

- \* <https://datatracker.ietf.org/doc/html/rfc2544#section-26.1> with <https://www.rfc-editor.org/errata/eid422> finally 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 than the precision goal away from the first value).

Depending on SUT behavior, different benchmark groups can report significantly different Throughput 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, although 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 ratio, 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 overall 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 possibility 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 specifies what to do if the longer trial is no longer a valid bound (TODO define?), e.g. start an external search. Additionally one halving can be saved during the shorter duration search.

### 5.2. FRMOL as reasonable start

TODO expand: Overall 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 support at high loads. (Using low enough max load would work, but frequently that would be smaller than SUT's actual 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 bisection 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 quadrupled). 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](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](mailto:mkonstan@cisco.com)

Vratko Polak  
Cisco Systems  
Email: [vrpolak@cisco.com](mailto:vrpolak@cisco.com)



Benchmarking Methodology Working Group  
Internet-Draft  
Intended status: Informational  
Expires: November 22, 2021

B. Balarajah  
C. Rossenhoevel  
EANTC AG  
B. Monkman  
NetSecOPEN  
May 21, 2021

Benchmarking Methodology for Network Security Device Performance  
draft-ietf-bmwg-ngfw-performance-09

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on November 22, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

|  |    |
|--|----|
| 1. Introduction . . . . .  | 4  |
| 2. Requirements . . . . .  | 4  |
| 3. Scope . . . . .   | 4  |
| 4. Test Setup . . . . .  | 4  |
| 4.1. Test Bed Configuration . . . . .                              | 4  |
| 4.2. DUT/SUT Configuration . . . . .                               | 6  |
| 4.2.1. Security Effectiveness Configuration . . . . .              | 12 |
| 4.3. Test Equipment Configuration . . . . .                        | 12 |
| 4.3.1. Client Configuration . . . . .                              | 12 |
| 4.3.2. Backend Server Configuration . . . . .                      | 15 |
| 4.3.3. Traffic Flow Definition . . . . .                           | 17 |
| 4.3.4. Traffic Load Profile . . . . .                              | 17 |
| 5. Test Bed Considerations . . . . .                               | 18 |
| 6. Reporting . . . . .   | 19 |
| 6.1. Introduction . . . . .  | 19 |
| 6.2. Detailed Test Results . . . . .                               | 21 |
| 6.3. Benchmarks and Key Performance Indicators . . . . .           | 21 |
| 7. Benchmarking Tests . . . . .                                    | 22 |
| 7.1. Throughput Performance with Application Traffic Mix . . . . . | 23 |
| 7.1.1. Objective . . . . .   | 23 |
| 7.1.2. Test Setup . . . . .  | 23 |
| 7.1.3. Test Parameters . . . . .                                   | 23 |
| 7.1.4. Test Procedures and Expected Results . . . . .              | 25 |
| 7.2. TCP/HTTP Connections Per Second . . . . .                     | 26 |
| 7.2.1. Objective . . . . .   | 26 |
| 7.2.2. Test Setup . . . . .  | 26 |
| 7.2.3. Test Parameters . . . . .                                   | 26 |
| 7.2.4. Test Procedures and Expected Results . . . . .              | 27 |
| 7.3. HTTP Throughput . . . . .                                     | 29 |
| 7.3.1. Objective . . . . .   | 29 |
| 7.3.2. Test Setup . . . . .  | 29 |
| 7.3.3. Test Parameters . . . . .                                   | 29 |
| 7.3.4. Test Procedures and Expected Results . . . . .              | 31 |
| 7.4. HTTP Transaction Latency . . . . .                            | 32 |
| 7.4.1. Objective . . . . .   | 32 |
| 7.4.2. Test Setup . . . . .  | 32 |

|  |    |
|--|----|
| 7.4.3. Test Parameters . . . . .                                 | 32 |
| 7.4.4. Test Procedures and Expected Results . . . . .            | 34 |
| 7.5. Concurrent TCP/HTTP Connection Capacity . . . . .           | 35 |
| 7.5.1. Objective . . . . .                                       | 35 |
| 7.5.2. Test Setup . . . . .                                      | 35 |
| 7.5.3. Test Parameters . . . . .                                 | 35 |
| 7.5.4. Test Procedures and Expected Results . . . . .            | 37 |
| 7.6. TCP/HTTPS Connections per Second . . . . .                  | 38 |
| 7.6.1. Objective . . . . .                                       | 38 |
| 7.6.2. Test Setup . . . . .                                      | 38 |
| 7.6.3. Test Parameters . . . . .                                 | 38 |
| 7.6.4. Test Procedures and Expected Results . . . . .            | 40 |
| 7.7. HTTPS Throughput . . . . .                                  | 41 |
| 7.7.1. Objective . . . . .                                       | 41 |
| 7.7.2. Test Setup . . . . .                                      | 41 |
| 7.7.3. Test Parameters . . . . .                                 | 42 |
| 7.7.4. Test Procedures and Expected Results . . . . .            | 43 |
| 7.8. HTTPS Transaction Latency . . . . .                         | 44 |
| 7.8.1. Objective . . . . .                                       | 44 |
| 7.8.2. Test Setup . . . . .                                      | 44 |
| 7.8.3. Test Parameters . . . . .                                 | 44 |
| 7.8.4. Test Procedures and Expected Results . . . . .            | 46 |
| 7.9. Concurrent TCP/HTTPS Connection Capacity . . . . .          | 47 |
| 7.9.1. Objective . . . . .                                       | 47 |
| 7.9.2. Test Setup . . . . .                                      | 47 |
| 7.9.3. Test Parameters . . . . .                                 | 47 |
| 7.9.4. Test Procedures and Expected Results . . . . .            | 49 |
| 8. IANA Considerations . . . . .                                 | 50 |
| 9. Security Considerations . . . . .                             | 50 |
| 10. Contributors . . . . .                                       | 50 |
| 11. Acknowledgements . . . . .                                   | 51 |
| 12. References . . . . .   | 51 |
| 12.1. Normative References . . . . .                             | 51 |
| 12.2. Informative References . . . . .                           | 51 |
| Appendix A. Test Methodology – Security Effectiveness Evaluation | 52 |
| A.1. Test Objective . . . . .                                    | 52 |
| A.2. Test Bed Setup . . . . .                                    | 52 |
| A.3. Test Parameters . . . . .                                   | 53 |
| A.3.1. DUT/SUT Configuration Parameters . . . . .                | 53 |
| A.3.2. Test Equipment Configuration Parameters . . . . .         | 53 |
| A.4. Test Results Validation Criteria . . . . .                  | 53 |
| A.5. Measurement . . . . .                                       | 54 |
| A.6. Test Procedures and Expected Results . . . . .              | 55 |
| A.6.1. Step 1: Background Traffic . . . . .                      | 55 |
| A.6.2. Step 2: CVE Emulation . . . . .                           | 55 |
| Appendix B. DUT/SUT Classification . . . . .                     | 55 |
| Authors' Addresses . . . . .                                     | 56 |

## 1. Introduction

15 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. Security function implementations have evolved to more advanced areas and have diversified into intrusion detection and prevention, threat management, analysis of encrypted traffic, etc. In an industry of growing importance, well-defined, and reproducible key performance indicators (KPIs) are increasingly needed as they enable fair and reasonable comparison of network security functions. All these reasons have led to the creation of a new next-generation network security device benchmarking document and this document obsoletes [RFC3511].

## 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. 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 test bed environments, test tool requirements, and test result formats.

## 4. Test Setup

Test setup defined in this document is applicable 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. Test Bed Configuration

Test bed 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 test bed. For this reason, this document recommends avoiding external devices such as switches and routers in the test bed 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 test bed.

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

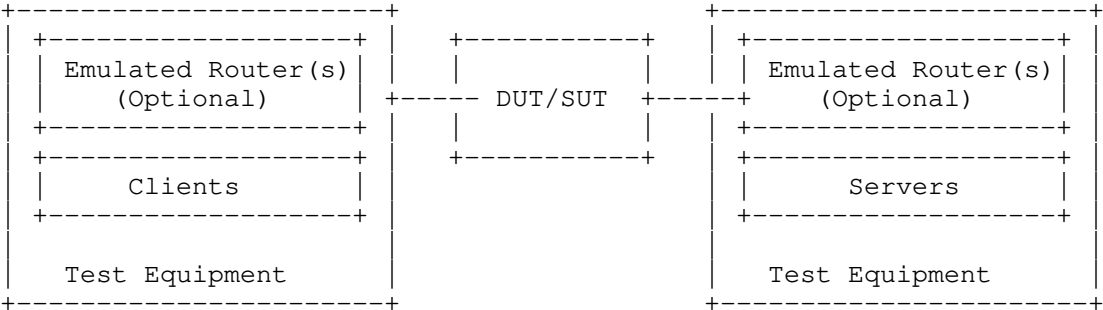


Figure 1: Test Bed Setup - Option 1

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

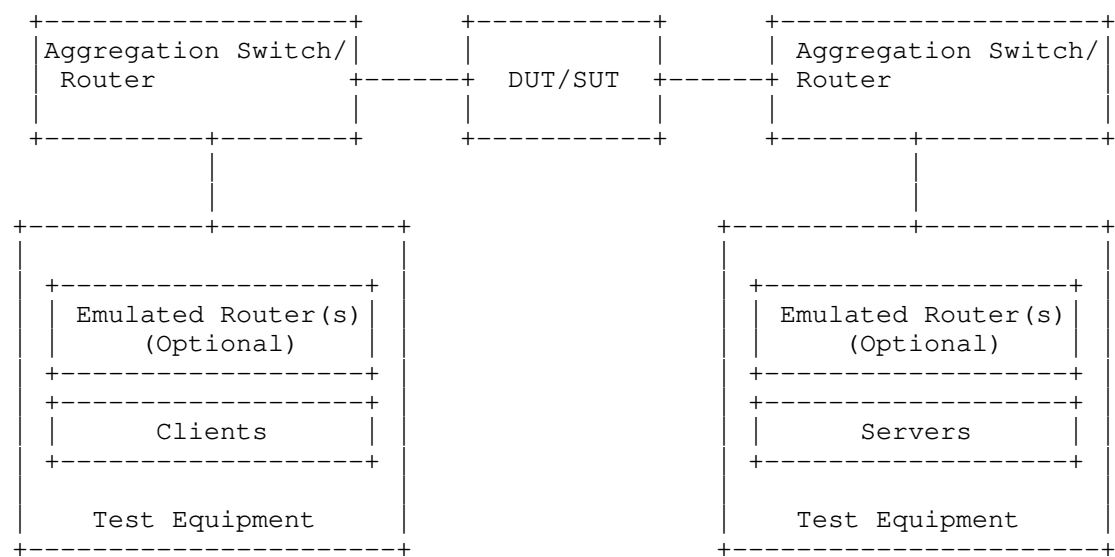


Figure 2: Test Bed Setup - Option 2

4.2. DUT/SUT Configuration

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

Table 1 and Table 2 below describe the RECOMMENDED and OPTIONAL sets of network security feature list for NGFW and NGIDS/NGIPS respectively. The selected security features SHOULD be consistently enabled on the DUT/SUT for all the 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 Features           | NGFW        |          |
|----------------------------|-------------|----------|
|                            | 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 Features           | NGIDS/NGIPS |          |
|----------------------------|-------------|----------|
|                            | 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: NGIDS/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

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

- o All RECOMMENDED security inspections enabled
- o Disposition of all flows of traffic are logged - Logging to an external device is permissible
- o Geographical location filtering and Application Identification and Control configured to be triggered based on a site or application from the defined traffic mix

In addition, a realistic number of access control rules (ACL) SHOULD be configured on the DUT/SUT where ACL's are configurable and also reasonable based on the deployment scenario. This document determines the number of access policy rules for four different classes of DUT/SUT; namely 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 Table 4 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 typical ACL based security policy. The ACL entries SHOULD be configured with routable IP subnets by the DUT/SUT. (Note: There will be differences between how security vendors implement ACL decision making.) The configured ACL MUST NOT block the security and measurement traffic used for the benchmarking tests.

|                   |                              |   |        | DUT/SUT<br>Classification<br># Rules |    |     |     |
|-------------------|------------------------------|---|--------|--------------------------------------|----|-----|-----|
| Rules Type        | Match<br>Criteria            | Description   | Action | XS                                   | S  | M   | L   |
| Application layer | Application                  | Any application not included in the measurement traffic   | block  | 5                                    | 10 | 20  | 50  |
| Transport layer   | Src IP and TCP/UDP Dst ports | Any src IP subnet used and any dst ports not used in the measurement traffic                    | block  | 25                                   | 50 | 100 | 250 |
| IP layer          | Src/Dst IP                   | Any src/dst IP subnet not used in the measurement traffic                                       | block  | 25                                   | 50 | 100 | 250 |
| Application layer | Application                  | Half of the applications included in the measurement traffic (see the note below)               | allow  | 10                                   | 10 | 10  | 10  |
| Transport layer   | Src IP and TCP/UDP Dst ports | Half of the src IP 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  |

Table 4: DUT/SUT Access List

Note: If the half of 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 2) of the DUT/SUT MUST be configured effectively in such a way to detect, prevent, and report the defined security Vulnerability sets. This Section defines the selection of the security Vulnerability sets from Common Vulnerabilities and Exposures (CVE) list for the testing. The vulnerability set 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.

#### 4.3.1.1. TCP Stack Attributes

The TCP stack SHOULD use a congestion control algorithm at client and server endpoints. The default IPv4 and IPv6 MSS segments size 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.

- o The IP blocks SHOULD consist of multiple unique, discontinuous static address blocks.
- o A default gateway is permitted.
- o 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 type. The Following options can be considered for a selection of traffic mix ratio.

(Option 1) 100 % IPv4, no IPv6

(Option 2) 80 % IPv4, 20% IPv6

(Option 3) 50 % IPv4, 50% IPv6

(Option 4) 20 % IPv4, 80% IPv6

(Option 5) no IPv4, 100% IPv6

Note: The IANA has assigned IP address range for the testing purpose as described in Section 8. If the test scenario requires additional number of IP addresses or subnets than the IANA assigned, this document recommends to use 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 emulated web client contains attributes that will materially affect how traffic is loaded. The objective is to emulate modern, typical browser attributes to improve realism of the result set.

For HTTP traffic emulation, the emulated browser MUST negotiate HTTP version 1.1 or higher. Depending on test scenarios and chosen HTTP version, the 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 browser MAY open Multiple concurrent streams per connection (multiplexing). If HTTP/3 is used the browser MUST open Quick UDP Internet Connections (QUIC) connection. 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 browser SHOULD advertise a User-Agent header. The 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. ECHDE-ECDSA-AES128-GCM-SHA256 with Prime256v1 (Signature Hash Algorithm: ecdsa\_secp256r1\_sha256 and Supported group: secp256r1)
2. ECDHE-RSA-AES128-GCM-SHA256 with RSA 2048 (Signature Hash Algorithm: rsa\_pkcs1\_sha256 and Supported group: secp256)
3. ECDHE-ECDSA-AES256-GCM-SHA384 with Secp521 (Signature Hash Algorithm: ecdsa\_secp384r1\_sha384 and Supported group: secp521r1)
4. ECDHE-RSA-AES256-GCM-SHA384 with RSA 4096 (Signature Hash Algorithm: rsa\_pkcs1\_sha384 and Supported group: secp256)

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

- o The server IP blocks SHOULD consist of unique, discontinuous static address blocks with one IP per Server Fully Qualified Domain Name (FQDN) endpoint per test port.
- o A default gateway is permitted. The 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.
- o 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 additional number of IP addresses or subnets than the IANA assigned, this document recommends to use 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 and settings chosen by the client (emulated web browser). The Server MUST advertise server type in the Server response header [RFC2616]. For HTTPS server, TLS 1.2 or higher MUST be used with a maximum record size of 16 KByte and MUST NOT use ticket resumption or Session ID reuse. The server SHOULD listen on port TCP 443. The server SHALL serve a certificate to the client. The HTTPS server MUST check Host SNI information with the FQDN if the SNI is in use. Cipher suite and key size on the server side MUST be configured similar to the client side configuration described in Section 4.3.1.3.



#### 4.3.3. Traffic Flow Definition

This section describes the traffic pattern between client and server endpoints. At the beginning of the test, the server endpoint initializes and will be ready to accept connection states including initialization of the TCP stack as well as bound HTTP and HTTPS servers. When a client endpoint is needed, it will initialize and be given attributes such as a MAC and IP address. The behavior of the client is to sweep through the given server IP space, 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 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's IP address or Fully Qualified Domain Names (FQDN) in Host Headers [RFC2616].

##### 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 to allow for better randomization of the start of client traffic. Each client will either open a new TCP connection or connect to a TCP persistence stack still open to that specific server. At any point that the 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 conical FQDN and URL path, as observed in the traffic mix used.

#### 4.3.4. Traffic Load Profile

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

1. During the Init phase, test bed devices including the client and server endpoints should negotiate layer 2-3 connectivity such as MAC learning and ARP. Only after successful MAC learning or ARP/ND resolution SHALL the test iteration move to the next phase. No measurements are made in this phase. The minimum RECOMMEND time for Init phase is 5 seconds. During this phase, the emulated clients SHOULD NOT initiate any sessions with the DUT/SUT, in contrast, the emulated servers should be ready to accept requests from DUT/SUT or from emulated clients.
  2. In the ramp up phase, the test equipment SHOULD start to generate the test traffic. It SHOULD use a set of approximate number of unique client IP addresses actively to generate traffic. The traffic SHOULD ramp up from zero to desired target objective. The target objective will be defined for each benchmarking test. The duration for the ramp up phase MUST be configured long enough, so that the test equipment does not overwhelm the DUT/SUT's 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 row results and calculating the statistics SHOULD be less than 2 seconds.
  4. In the ramp down/close phase, no new connections are established, and no measurements are made. The time duration for ramp up and ramp down phase SHOULD be the same.
  5. The last phase is administrative and will occur when the test equipment merges and collates the report data.
5. Test Bed Considerations

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

1. Ensure that any ancillary switching or routing functions between the system under test and the test equipment do not limit the

performance of the traffic generator. This is specifically important for virtualized components (vSwitches, vRouters).

2. Verify that the performance of the test equipment matches and reasonably exceeds the expected maximum performance of the DUT/SUT.
3. Assert that the test bed characteristics are stable during the entire test session. Several factors might influence stability specifically, for virtualized test beds. For example, additional workloads in a virtualized system, load balancing, and movement of virtual machines during the test, or simple issues such as additional heat created by high workloads leading to an emergency CPU performance reduction.

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

## 6. Reporting

This section describes how the final report should be formatted and presented. The final test report MAY have two major sections; Introduction and 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 test MUST be prominent.
2. Summary of test bed software and hardware details
  - A. DUT/SUT hardware/virtual configuration
    - + This section SHOULD clearly identify the make and model of the DUT/SUT
    - + The port interfaces, including speed and link information MUST be documented.
    - + If the DUT/SUT is a Virtual Network Function (VNF), host (server) hardware and software details, interface acceleration type such as DPDK and SR-IOV used CPU cores, used RAM, and the resource sharing (e.g. Pinning details and NUMA Node) configuration MUST be documented. The

virtual components such as Hypervisor, virtual switch version MUST be also documented.

- + Any additional hardware relevant to the DUT/SUT such as controllers MUST be documented

B. DUT/SUT software

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

C. DUT/SUT enabled features

- + Configured DUT/SUT features (see Table 1 and Table 2) MUST be documented
- + Attributes of those featured MUST be documented
- + Any additional relevant information about features MUST be documented

D. Test equipment hardware and software

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

E. Key test parameters

- + Used cipher suites and keys
- + IPv4 and IPv6 traffic distribution
- + Number of configured ACL

F. Details of application traffic mix used in the 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.

- o 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]).
- o 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 not by a RST.

- o 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) transaction, it must have a valid status code, and the appropriate FIN, FIN/ACK sequence must have been completed.
- o 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.
- o 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.
- o 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 millisecond.
- o 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 customer use case, users can choose the application traffic mix for this test. The details about the traffic mix MUST be documented in the report. At least the following traffic mix details MUST be documented and reported together with the test results:

Name of applications and layer 7 protocols

Percentage of emulated traffic for each application and layer 7 protocols

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

Used object sizes for each application and layer 7 protocols

### 7.1.2. Test Setup

Test bed setup MUST be configured as defined in Section 4. Any benchmarking test specific test bed 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 is configured without SSL inspection feature, the test report MUST explain the implications of this to the relevant application traffic mix encrypted traffic.

#### 7.1.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. Following parameters MUST be noted for this 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 inspected throughput: 10% of the "Target inspected throughput"

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

- a. Number of failed application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of total attempt transactions.
- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.

#### 7.1.3.5. Measurement

Following KPI metrics MUST be reported for this 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. 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 inspected throughput" rate as described in the parameters Section 7.1.3.2. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4. The DUT/SUT SHOULD reach the "Initial inspected 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 the parameter table. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4. The test equipment SHOULD start to measure and record all specified KPIs. 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 the different fixed HTTP response object sizes (the different load conditions) defined in Section 7.2.3.2.

### 7.2.2. Test Setup

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

### 7.2.3. Test Parameters

In this section, 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. 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" (an optional parameter for documentation)

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

- a. Number of failed Application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of total attempt transactions.
- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.
- c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (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 almost at 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. This test procedure MAY be repeated multiple times with different IP types; IPv4 only, IPv6 only and IPv4 and IPv6 mixed traffic distribution.

#### 7.2.4.1. Step 1: Test Initialization and Qualification

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

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

The DUT/SUT SHOULD reach the "Initial connections per second" before the sustain phase. The measured KPIs during the sustain phase MUST meet 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 be continued 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 the parameters table. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4.

During the ramp up and sustain phase of each test iteration, other KPIs such as inspected throughput, concurrent TCP connections and application transactions per second MUST NOT reach to the maximum value the DUT/SUT can support. The test results for specific test iterations SHOULD NOT be reported, if the above mentioned KPI (especially 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, 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

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

#### 7.3.3. Test Parameters

In this section, 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. 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 inspected throughput: 10% of "Target inspected throughput" (an optional parameter for documentation)

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 the table

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

Table 5: Mixed Objects

#### 7.3.3.3. Test Results Validation Criteria

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

- a. Number of failed Application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of attempt transactions.
- b. Traffic SHOULD be forwarded 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 almost at 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. 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 the parameters Section 7.3.3.2.

The traffic load profile SHOULD be defined as described in Section 4.3.4. The DUT/SUT SHOULD reach the "Initial 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 the parameters table. 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

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

#### 7.4.3. Test Parameters

In this section, 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. 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 connection 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" (an optional parameter for documentation)

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

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

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

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

#### 7.4.3.4. Measurement

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. 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 the parameters Section 7.4.3.2. The traffic load profile can be defined as described in Section 4.3.4.

The DUT/SUT SHOULD reach the "Initial objective" before the sustain phase. The measured KPIs during the sustain phase MUST meet 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 the parameters table. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4.

The test equipment SHOULD start to measure and record all specified KPIs. 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 TFB 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

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

#### 7.5.3. Test Parameters

In this section, 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. 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" (an optional parameter for documentation)

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

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

- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.
- c. During the sustain phase, traffic SHOULD be forwarded 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. 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 to the maximum value that the DUT/SUT can support.

The test equipment SHOULD start to measure and record KPIs defined in Section 7.5.3.4. 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 the test equipment configuration parameters Section 7.6.3.2 to measure connections per second performance under a variety of DUT Security inspection load conditions.

#### 7.6.2. Test Setup

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

#### 7.6.3. Test Parameters

In this section, 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. 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" (an optional parameter for documentation)

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 command 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 test Criteria is defined as test results validation criteria:

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

- c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (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 almost at 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 1KByte object size.

#### 7.6.4. Test Procedures and Expected Results

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

##### 7.6.4.1. Step 1: Test Initialization and Qualification

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

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



During the ramp up and sustain phase, other KPIs such as inspected throughput, concurrent TCP connections and application transactions per second MUST NOT reach the maximum value that the DUT/SUT can support. The test results for specific test iteration SHOULD NOT be reported, if the above mentioned KPI (especially 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 the parameter Section 7.7.3.2.

#### 7.7.2. Test Setup

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

### 7.7.3. Test Parameters

In this section, 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. 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 inspected throughput: 10% of "Target inspected throughput" (an optional parameter for documentation)

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

RECOMMENDED ciphers and keys defined in Section 4.3.1.3

RECOMMENDED HTTPS response object size: 1, 16, 64, 256 KByte, and mixed objects defined in the Table 5 under the Section 7.3.3.2.

#### 7.7.3.3. Test Results Validation Criteria

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

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

- b. Traffic SHOULD be forwarded 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 almost at 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. 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 inspected throughput" as defined in the parameters Section 7.7.3.2.

The traffic load profile SHOULD be defined as described in Section 4.3.4. The DUT/SUT SHOULD reach the "Initial inspected 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 the parameters table. 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 is running with sustainable HTTPS transactions per second supported by the DUT/SUT under different HTTPS response object size.

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

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

#### 7.8.2. Test Setup

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

#### 7.8.3. Test Parameters

In this section, 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. 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" (an optional parameter for documentation)

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

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

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

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

#### 7.8.3.4. Measurement

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. 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 parameters Section 7.8.3.2. The traffic load profile can be defined as described in Section 4.3.4.

The DUT/SUT SHOULD reach the "Initial objective" before the sustain phase. The measured KPIs during the sustain phase MUST meet 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 the parameters table. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4.

The test equipment SHOULD start to measure and record all specified KPIs. 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 TFB and TTLB.

### 7.9. Concurrent TCP/HTTPS Connection Capacity

#### 7.9.1. Objective

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

#### 7.9.2. Test Setup

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

#### 7.9.3. Test Parameters

In this section, 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. 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" (an optional parameter for documentation)

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

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



- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.
- c. During the sustain phase, traffic SHOULD be forwarded 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. 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

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

### 10. Contributors

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

Alex Samonte, Amritam Putatunda, Aria Eslambolchizadeh, David DeSanto, Jurrie Van Den Breekel, Ryan Liles, Samaresh Nair, Stephen Goudreault, and Tim Otto

## 11. Acknowledgements

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

Anand Vijayan, Baski Mohan, Chao Guo, Chris Brown, Chris Marshall, Jay Lindenauer, Michael Shannon, Mike Deichman, Ray Vinson, Ryan Riese, Tim Carlin, 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>>.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <<https://www.rfc-editor.org/info/rfc2616>>.
- [RFC2647] Newman, D., "Benchmarking Terminology for Firewall Performance", RFC 2647, DOI 10.17487/RFC2647, August 1999, <<https://www.rfc-editor.org/info/rfc2647>>.
- [RFC3511] Hickman, B., Newman, D., Tadjudin, S., and T. Martin, "Benchmarking Methodology for Firewall Performance", RFC 3511, DOI 10.17487/RFC3511, April 2003, <<https://www.rfc-editor.org/info/rfc3511>>.

- [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>>.
- [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>>.

## Appendix A. Test Methodology - Security Effectiveness Evaluation

### A.1. Test Objective

This test methodology verifies the DUT/SUT is able to detect, prevent and report the vulnerabilities.

In this test, background test traffic will be generated in order to utilize the DUT/SUT. In parallel, the CVEs will be sent to the DUT/SUT as encrypted and as well as clear text payload formats using a traffic generator. The selection of the CVEs is described in Section 4.2.1.

- o Number of blocked CVEs
- o Number of bypassed (nonblocked) CVEs
- o Background traffic performance (verify if the background traffic is impacted while sending CVE toward DUT/SUT)
- o Accuracy of DUT/SUT statistics in term of vulnerabilities reporting

### A.2. Test Bed Setup

The same Test bed 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:

- o Background Traffic: 45% of maximum HTTP throughput and 45% of Maximum HTTPS throughput supported by the DUT/SUT (measured with object size 64 KByte in the benchmarking tests "HTTP(S) Throughput" defined in Section 7.3 and Section 7.7.
- o RECOMMENDED CVE traffic transmission Rate: 10 CVEs per second
- o RECOMMEND to generate each CVE multiple times (sequentially) at 10 CVEs per second
- o Ciphers and Keys for the encrypted CVE traffic MUST use the same cipher configured for HTTPS traffic related benchmarking tests (Section 7.6 - Section 7.9)

### A.4. Test Results Validation Criteria

The following test Criteria is defined as test results validation criteria. Test results validation criteria MUST be monitored during the whole test duration.

- a. Number of failed Application transaction in the background traffic MUST be less than 0.01% of attempted transactions.
- b. Number of Terminated TCP connections of the background traffic (due to unexpected TCP RST sent by DUT/SUT) MUST be less than

0.01% of total initiated TCP connections in the background traffic.

- c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (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:

- o Blocked CVEs: It SHOULD be represented in the following ways:
  - \* Number of blocked CVEs out of total CVEs
  - \* Percentage of blocked CVEs
- o Unblocked CVEs: It SHOULD be represented in the following ways:
  - \* Number of unblocked CVEs out of total CVEs
  - \* Percentage of unblocked CVEs
- o Background traffic behavior: it SHOULD 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 that 5% and less than or equal to 10% (e.g. small spikes)
  - \* Heavily impacted: considered as "Heavily impacted" if any deviation of traffic forwarding rate is greater that 10% (e.g. large spikes) or reduced the background HTTP(S) throughput greater than 10%
- o DUT/SUT reporting accuracy: DUT/SUT MUST report all detected vulnerabilities.

Optional KPIs:

- o List of unblocked CVEs

#### A.6. Test Procedures and Expected Results

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

##### A.6.1. Step 1: Background Traffic

Generate the background traffic at the transmission rate defined in the parameter section.

The DUT/SUT MUST reach the target objective (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 the background traffic (in sustain phase), send the CVE traffic as defined in the parameter section.

The test equipment SHOULD start to measure and record all specified KPIs. 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 attempts 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, Small, Medium, and Large. The RECOMMENDED throughput values for the following categories are:

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

Small (S) - supported throughput greater than 1Gbit/s and less than or equal to 5Gbit/s

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

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

#### Authors' Addresses

Balamuhunthan Balarajah  
Berlin  
Germany

Email: [bm.balarajah@gmail.com](mailto:bm.balarajah@gmail.com)

Carsten Rossenhoevel  
EANTC AG  
Salzufer 14  
Berlin 10587  
Germany

Email: [cross@eantc.de](mailto:cross@eantc.de)

Brian Monkman  
NetSecOPEN  
417 Independence Court  
Mechanicsburg, PA 17050  
USA

Email: [bmonkman@netsecopen.org](mailto:bmonkman@netsecopen.org)



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

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

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 5 July 2022.

Copyright Notice

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

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

## Table of Contents

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

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

## 1. Introduction

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

## 2. Requirements

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

## 3. Scope

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

## 4. Test Setup

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

#### 4.1. Testbed Configuration

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

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

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

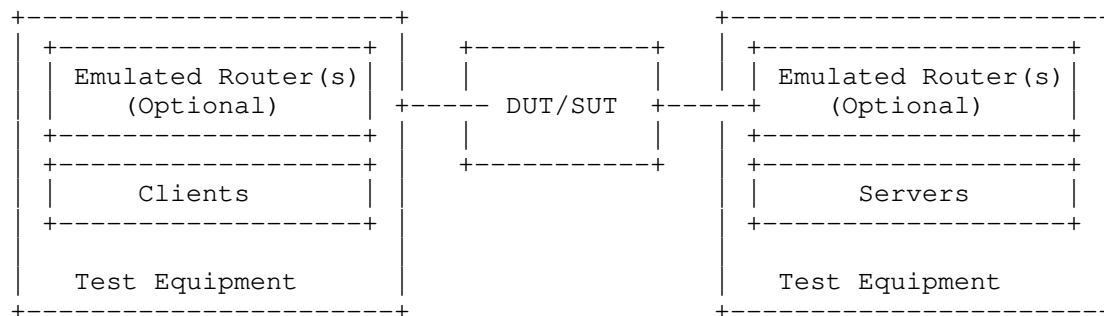


Figure 1: Testbed Setup - Option 1

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

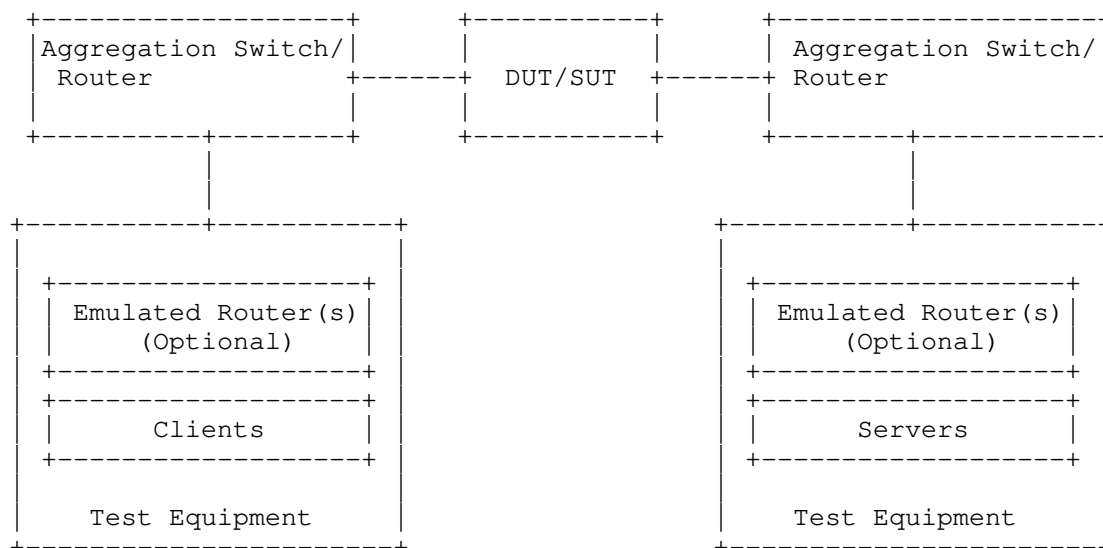


Figure 2: Testbed Setup - Option 2

#### 4.2. DUT/SUT Configuration

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

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

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

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

Table 1: NGFW Security Features

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

Table 2: NGIPS Security Features

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

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



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

Table 3: Security Feature Description

Below is a summary of the DUT/SUT configuration:

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

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

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

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

Figure 3: DUT/SUT Access List

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

#### 4.2.1. Security Effectiveness Configuration

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

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

#### 4.3. Test Equipment Configuration

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

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

##### 4.3.1. Client Configuration

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

#### 4.3.1.1. TCP Stack Attributes

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

#### 4.3.1.2. Client IP Address Space

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

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

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

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

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

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

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

(Option 1) 100 % IPv4, no IPv6

(Option 2) 80 % IPv4, 20% IPv6

(Option 3) 50 % IPv4, 50% IPv6

(Option 4) 20 % IPv4, 80% IPv6

(Option 5) no IPv4, 100% IPv6

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

#### 4.3.1.3. Emulated Web Browser Attributes

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

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

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

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

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

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

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

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

1. TLS\_AES\_128\_GCM\_SHA256
2. TLS\_AES\_256\_GCM\_SHA384
3. TLS\_CHACHA20\_POLY1305\_SHA256
4. TLS\_AES\_128\_CCM\_SHA256
5. TLS\_AES\_128\_CCM\_8\_SHA256

#### 4.3.2. Backend Server Configuration

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

#### 4.3.2.1. TCP Stack Attributes

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

#### 4.3.2.2. Server Endpoint IP Addressing

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

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

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

#### 4.3.2.3. HTTP / HTTPS Server Pool Endpoint Attributes

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



#### 4.3.3. Traffic Flow Definition

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

##### 4.3.3.1. Description of Intra-Client Behavior

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

##### 4.3.4. Traffic Load Profile

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

1. Init phase: Testbed devices including the client and server endpoints should negotiate layer 2-3 connectivity such as MAC learning and ARP. Only after successful MAC learning or ARP/ND resolution SHALL the test iteration move to the next phase. No measurements are made in this phase. The minimum RECOMMENDED time for Init phase is 5 seconds. During this phase, the emulated clients SHOULD NOT initiate any sessions with the DUT/SUT, in contrast, the emulated servers should be ready to accept requests from DUT/SUT or from emulated clients.
  2. Ramp up phase: The test equipment SHOULD start to generate the test traffic. It SHOULD use a set of the approximate number of unique client IP addresses to generate traffic. The traffic SHOULD ramp up from zero to desired target objective. The target objective is defined for each benchmarking test. The duration for the ramp up phase MUST be configured long enough that the test equipment does not overwhelm the DUT/SUTs stated performance metrics defined in Section 6.3 namely, TCP Connections Per Second, Inspected Throughput, Concurrent TCP Connections, and Application Transactions Per Second. No measurements are made in this phase.
  3. Sustain phase: Starts when all required clients are active and operating at their desired load condition. In the sustain phase, the test equipment SHOULD continue generating traffic to constant target value for a constant number of active clients. The minimum RECOMMENDED time duration for sustain phase is 300 seconds. This is the phase where measurements occur. The test equipment SHOULD measure and record statistics continuously. The sampling interval for collecting the raw results and calculating the statistics SHOULD be less than 2 seconds.
  4. Ramp down phase: No new connections are established, and no measurements are made. The time duration for ramp up and ramp down phase SHOULD be the same.
  5. Collection phase: The last phase is administrative and will occur when the test equipment merges and collates the report data.
5. Testbed Considerations

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

1. Perform the reference test either by configuring the DUT/SUT in the most trivial setup (fast forwarding) or without presence of the DUT/SUT.
2. Generate traffic from traffic generator. Choose a traffic profile used for HTTP or HTTPS throughput performance test with smallest object size.
3. Ensure that any ancillary switching or routing functions added in the test equipment does not limit the performance by introducing network metrics such as packet loss and latency. This is specifically important for virtualized components (e.g., vSwitches, vRouters).
4. Verify that the generated traffic (performance) of the test equipment matches and reasonably exceeds the expected maximum performance of the DUT/SUT.
5. Record the network performance metrics packet loss latency introduced by the test environment (without DUT/SUT).
6. Assert that the testbed characteristics are stable during the entire test session. Several factors might influence stability specifically, for virtualized testbeds. For example, additional workloads in a virtualized system, load balancing, and movement of virtual machines during the test, or simple issues such as additional heat created by high workloads leading to an emergency CPU performance reduction.

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

## 6. Reporting

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

### 6.1. Introduction

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

1. The time and date of the execution of the tests
2. Summary of testbed software and hardware details

## a. DUT/SUT hardware/virtual configuration

- \* This section SHOULD clearly identify the make and model of the DUT/SUT
- \* The port interfaces, including speed and link information
- \* If the DUT/SUT is a Virtual Network Function (VNF), host (server) hardware and software details, interface acceleration type such as DPDK and SR-IOV, used CPU cores, used RAM, resource sharing (e.g. Pinning details and NUMA Node) configuration details, hypervisor version, virtual switch version
- \* details of any additional hardware relevant to the DUT/SUT such as controllers

## b. DUT/SUT software

- \* Operating system name
- \* Version
- \* Specific configuration details (if any)

## c. DUT/SUT enabled features

- \* Configured DUT/SUT features (see Table 1 and Table 2)
- \* Attributes of the above-mentioned features
- \* Any additional relevant information about the features

## d. Test equipment hardware and software

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

## e. Key test parameters

- \* Used cipher suites and keys
- \* IPv4 and IPv6 traffic distribution

- \* Number of configured ACL
- f. Details of application traffic mix used in the benchmarking test "Throughput Performance with Application Traffic Mix" (Section 7.1)
  - \* Name of applications and layer 7 protocols
  - \* Percentage of emulated traffic for each application and layer 7 protocols
  - \* Percentage of encrypted traffic and used cipher suites and keys (The RECOMMENDED ciphers and keys are defined in Section 4.3.1.3)
  - \* Used object sizes for each application and layer 7 protocols

### 3. Results Summary / Executive Summary

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

## 6.2. Detailed Test Results

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

- a. KPIs MUST be documented separately for each benchmarking test. The format of the KPI metrics SHOULD be presented as described in Section 6.3.
- b. The next level of details SHOULD be graphs showing each of these metrics over the duration (sustain phase) of the test. This allows the user to see the measured performance stability changes over time.

## 6.3. Benchmarks and Key Performance Indicators

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

- \* Concurrent TCP Connections

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

\* TCP Connections Per Second

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

\* Application Transactions Per Second

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

\* TLS Handshake Rate

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

\* Inspected Throughput

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

\* Time to First Byte (TTFB)

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

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

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

## 7. Benchmarking Tests

### 7.1. Throughput Performance with Application Traffic Mix

#### 7.1.1. Objective

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

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

Name of applications and layer 7 protocols

Percentage of emulated traffic for each application and layer 7 protocol

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

Used object sizes for each application and layer 7 protocols

#### 7.1.2. Test Setup

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

#### 7.1.3. Test Parameters

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

#### 7.1.3.1. DUT/SUT Configuration Parameters

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

#### 7.1.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

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

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

#### 7.1.3.3. Traffic Profile

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

#### 7.1.3.4. Test Results Validation Criteria

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



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

#### 7.1.3.5. Measurement

Following KPI metrics MUST be reported for this benchmarking test:

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

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

Optional KPIs: TCP Connections Per Second and TLS Handshake Rate

#### 7.1.4. Test Procedures and Expected Results

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

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

##### 7.1.4.1. Step 1: Test Initialization and Qualification

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

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

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

#### 7.1.4.2. Step 2: Test Run with Target Objective

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

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

#### 7.1.4.3. Step 3: Test Iteration

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

### 7.2. TCP/HTTP Connections Per Second

#### 7.2.1. Objective

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

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

### 7.2.2. Test Setup

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

### 7.2.3. Test Parameters

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

#### 7.2.3.1. DUT/SUT Configuration Parameters

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

#### 7.2.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

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

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

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

#### 7.2.3.3. Test Results Validation Criteria

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

- a. Number of failed application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of total attempted transactions.
- b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.
- c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).
- d. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.

#### 7.2.3.4. Measurement

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

#### 7.2.4. Test Procedures and Expected Results

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

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

#### 7.2.4.1. Step 1: Test Initialization and Qualification

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

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

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

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

#### 7.2.4.2. Step 2: Test Run with Target Objective

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

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

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

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

#### 7.2.4.3. Step 3: Test Iteration

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

### 7.3. HTTP Throughput

#### 7.3.1. Objective

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

#### 7.3.2. Test Setup

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

#### 7.3.3. Test Parameters

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

##### 7.3.3.1. DUT/SUT Configuration Parameters

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

##### 7.3.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

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

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

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

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

Table 4: Mixed Objects

#### 7.3.3.3. Test Results Validation Criteria

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

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

- b. Traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).
- c. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.

#### 7.3.3.4. Measurement

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

#### 7.3.4. Test Procedures and Expected Results

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

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

##### 7.3.4.1. Step 1: Test Initialization and Qualification

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

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

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

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

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



#### 7.3.4.2. Step 2: Test Run with Target Objective

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

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

#### 7.3.4.3. Step 3: Test Iteration

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

### 7.4. HTTP Transaction Latency

#### 7.4.1. Objective

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

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

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

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

#### 7.4.2. Test Setup

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

#### 7.4.3. Test Parameters

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

##### 7.4.3.1. DUT/SUT Configuration Parameters

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

##### 7.4.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

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

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

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

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

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

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

#### 7.4.3.3. Test Results Validation Criteria

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

- a. Number of failed application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of attempt transactions.
- b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.
- c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).
- d. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.
- e. After ramp up the DUT MUST achieve the "Target objective" defined in Section 7.4.3.2 and remain in that state for the entire test duration (sustain phase).

#### 7.4.3.4. Measurement

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

#### 7.4.4. Test Procedures and Expected Results

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

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

#### 7.4.4.1. Step 1: Test Initialization and Qualification

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

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

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

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

#### 7.4.4.2. Step 2: Test Run with Target Objective

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

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

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

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

### 7.5. Concurrent TCP/HTTP Connection Capacity

#### 7.5.1. Objective

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

#### 7.5.2. Test Setup

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

### 7.5.3. Test Parameters

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

#### 7.5.3.1. DUT/SUT Configuration Parameters

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

#### 7.5.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

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

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

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

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

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

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

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

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

#### 7.5.3.3. Test Results Validation Criteria

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

- a. Number of failed application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transaction) of total attempted transactions.
- b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.
- c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

#### 7.5.3.4. Measurement

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

#### 7.5.4. Test Procedures and Expected Results

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

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

#### 7.5.4.1. Step 1: Test Initialization and Qualification

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

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

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

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

#### 7.5.4.2. Step 2: Test Run with Target Objective

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

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

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

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

#### 7.5.4.3. Step 3: Test Iteration

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

### 7.6. TCP/HTTPS Connections per Second

#### 7.6.1. Objective

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

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

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

#### 7.6.2. Test Setup

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

#### 7.6.3. Test Parameters

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

##### 7.6.3.1. DUT/SUT Configuration Parameters

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

##### 7.6.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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



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

RECOMMENDED ciphers and keys defined in Section 4.3.1.3

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

#### 7.6.3.3. Test Results Validation Criteria

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

- a. Number of failed application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of attempt transactions.
- b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.
- c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).
- d. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.

#### 7.6.3.4. Measurement

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

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

#### 7.6.4. Test Procedures and Expected Results

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

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

##### 7.6.4.1. Step 1: Test Initialization and Qualification

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

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

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

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

##### 7.6.4.2. Step 2: Test Run with Target Objective

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

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

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

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

#### 7.6.4.3. Step 3: Test Iteration

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

### 7.7. HTTPS Throughput

#### 7.7.1. Objective

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

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

#### 7.7.2. Test Setup

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

#### 7.7.3. Test Parameters

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

##### 7.7.3.1. DUT/SUT Configuration Parameters

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

##### 7.7.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

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

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

RECOMMENDED ciphers and keys defined in Section 4.3.1.3

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

#### 7.7.3.3. Test Results Validation Criteria

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

- a. Number of failed Application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of attempt transactions.
- b. Traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).
- c. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.

#### 7.7.3.4. Measurement

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

#### 7.7.4. Test Procedures and Expected Results

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

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

##### 7.7.4.1. Step 1: Test Initialization and Qualification

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

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

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

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

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

##### 7.7.4.2. Step 2: Test Run with Target Objective

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

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

#### 7.7.4.3. Step 3: Test Iteration

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

### 7.8. HTTPS Transaction Latency

#### 7.8.1. Objective

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

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

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

#### 7.8.2. Test Setup

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

#### 7.8.3. Test Parameters

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

##### 7.8.3.1. DUT/SUT Configuration Parameters

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

##### 7.8.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

RECOMMENDED cipher suites and key sizes defined in Section 4.3.1.3

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

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

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

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

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

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

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

#### 7.8.3.3. Test Results Validation Criteria

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

- a. Number of failed application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of attempt transactions.
- b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.
- c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

- d. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections at approximately the same rate.
- e. After ramp up the DUT/SUT MUST achieve the "Target objective" defined in the parameter Section 7.8.3.2 and remain in that state for the entire test duration (sustain phase).

#### 7.8.3.4. Measurement

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

#### 7.8.4. Test Procedures and Expected Results

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

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

##### 7.8.4.1. Step 1: Test Initialization and Qualification

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

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

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

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



#### 7.8.4.2. Step 2: Test Run with Target Objective

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

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

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

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

### 7.9. Concurrent TCP/HTTPS Connection Capacity

#### 7.9.1. Objective

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

#### 7.9.2. Test Setup

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

#### 7.9.3. Test Parameters

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

##### 7.9.3.1. DUT/SUT Configuration Parameters

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

##### 7.9.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

RECOMMENDED cipher suites and key sizes defined in Section 4.3.1.3

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

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

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

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

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

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

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

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

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

#### 7.9.3.3. Test Results Validation Criteria

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

- a. Number of failed application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of total attempted transactions.
- b. Number of terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.
- c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).

#### 7.9.3.4. Measurement

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

#### 7.9.4. Test Procedures and Expected Results

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

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

##### 7.9.4.1. Step 1: Test Initialization and Qualification

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

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

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

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

#### 7.9.4.2. Step 2: Test Run with Target Objective

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

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

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

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

#### 7.9.4.3. Step 3: Test Iteration

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

### 8. IANA Considerations

This document makes no specific request of IANA.

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

## 9. Security Considerations

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

## 10. Contributors

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

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

## 11. Acknowledgements

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

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

## 12. References

### 12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

### 12.2. Informative References

- [RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <<https://www.rfc-editor.org/info/rfc2544>>.
- [RFC2647] Newman, D., "Benchmarking Terminology for Firewall Performance", RFC 2647, DOI 10.17487/RFC2647, August 1999, <<https://www.rfc-editor.org/info/rfc2647>>.
- [RFC3511] Hickman, B., Newman, D., Tadjudin, S., and T. Martin, "Benchmarking Methodology for Firewall Performance", RFC 3511, DOI 10.17487/RFC3511, April 2003, <<https://www.rfc-editor.org/info/rfc3511>>.
- [RFC5180] Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, DOI 10.17487/RFC5180, May 2008, <<https://www.rfc-editor.org/info/rfc5180>>.
- [RFC6815] Bradner, S., Dubray, K., McQuaid, J., and A. Morton, "Applicability Statement for RFC 2544: Use on Production Networks Considered Harmful", RFC 6815, DOI 10.17487/RFC6815, November 2012, <<https://www.rfc-editor.org/info/rfc6815>>.
- [RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <<https://www.rfc-editor.org/info/rfc6890>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <<https://www.rfc-editor.org/info/rfc9000>>.

#### Appendix A. Test Methodology - Security Effectiveness Evaluation

### A.1. Test Objective

This test methodology verifies the DUT/SUT is able to detect, prevent, and report the vulnerabilities.

In this test, background test traffic will be generated to utilize the DUT/SUT. In parallel, the CVEs will be sent to the DUT/SUT as encrypted and as well as clear text payload formats using a traffic generator. The selection of the CVEs is described in Section 4.2.1.

The following KPIs are measured in this test:

- \* Number of blocked CVEs
- \* Number of bypassed (nonblocked) CVEs
- \* Background traffic performance (verify if the background traffic is impacted while sending CVE toward DUT/SUT)
- \* Accuracy of DUT/SUT statistics in term of vulnerabilities reporting

### A.2. Testbed Setup

The same testbed MUST be used for security effectiveness test and as well as for benchmarking test cases defined in Section 7.

### A.3. Test Parameters

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

#### A.3.1. DUT/SUT Configuration Parameters

DUT/SUT configuration parameters MUST conform to the requirements defined in Section 4.2. The same DUT configuration MUST be used for Security effectiveness test and as well as for benchmarking test cases defined in Section 7. The DUT/SUT MUST be configured in inline mode and all detected attack traffic MUST be dropped and the session SHOULD be reset

#### A.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. The same client and server IP ranges MUST be configured as used in the benchmarking test cases. In addition, the following parameters MUST be documented for this benchmarking test:

- \* Background Traffic: 45% of maximum HTTP throughput and 45% of Maximum HTTPS throughput supported by the DUT/SUT (measured with object size 64 KByte in the benchmarking tests "HTTP(S) Throughput" defined in Section 7.3 and Section 7.7).
- \* RECOMMENDED CVE traffic transmission Rate: 10 CVEs per second
- \* It is RECOMMENDED to generate each CVE multiple times (sequentially) at 10 CVEs per second
- \* Ciphers and keys for the encrypted CVE traffic MUST use the same cipher configured for HTTPS traffic related benchmarking tests (Section 7.6 – Section 7.9)

#### A.4. Test Results Validation Criteria

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

- a. Number of failed application transaction in the background traffic MUST be less than 0.01% of attempted transactions.
- b. Number of terminated TCP connections of the background traffic (due to unexpected TCP RST sent by DUT/SUT) MUST be less than 0.01% of total initiated TCP connections in the background traffic.
- c. During the sustain phase, traffic SHOULD be forwarded at a constant rate (considered as a constant rate if any deviation of traffic forwarding rate is less than 5%).
- d. False positive MUST NOT occur in the background traffic.

#### A.5. Measurement

Following KPI metrics MUST be reported for this test scenario:

Mandatory KPIs:

- \* Blocked CVEs: It SHOULD be represented in the following ways:
  - Number of blocked CVEs out of total CVEs
  - Percentage of blocked CVEs
- \* Unblocked CVEs: It SHOULD be represented in the following ways:



- Number of unblocked CVEs out of total CVEs
- Percentage of unblocked CVEs
- \* Background traffic behavior: It SHOULD be represented one of the followings ways:
  - No impact: Considered as "no impact'" if any deviation of traffic forwarding rate is less than or equal to 5 % (constant rate)
  - Minor impact: Considered as "minor impact" if any deviation of traffic forwarding rate is greater than 5% and less than or equal to 10% (i.e. small spikes)
  - Heavily impacted: Considered as "Heavily impacted" if any deviation of traffic forwarding rate is greater than 10% (i.e. large spikes) or reduced the background HTTP(S) throughput greater than 10%
- \* DUT/SUT reporting accuracy: DUT/SUT MUST report all detected vulnerabilities.

Optional KPIs:

- \* List of unblocked CVEs

#### A.6. Test Procedures and Expected Results

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

##### A.6.1. Step 1: Background Traffic

Generate background traffic at the transmission rate defined in Appendix A.3.2.

The DUT/SUT MUST reach the target objective (HTTP(S) throughput) in sustain phase. The measured KPIs during the sustain phase MUST meet all the test results validation criteria defined in Appendix A.4.

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

#### A.6.2. Step 2: CVE Emulation

While generating background traffic (in sustain phase), send the CVE traffic as defined in the parameter section.

The test equipment SHOULD start to measure and record all specified KPIs. Continue the test until all CVEs are sent.

The measured KPIs MUST meet all the test results validation criteria defined in Appendix A.4.

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

#### Appendix B. DUT/SUT Classification

This document aims to classify the DUT/SUT in four different categories based on its maximum supported firewall throughput performance number defined in the vendor datasheet. This classification MAY help user to determine specific configuration scale (e.g., number of ACL entries), traffic profiles, and attack traffic profiles, scaling those proportionally to DUT/SUT sizing category.

The four different categories are Extra Small (XS), Small (S), Medium (M), and Large (L). The RECOMMENDED throughput values for the following categories are:

Extra Small (XS) - Supported throughput less than or equal to 1Gbit/s

Small (S) - Supported throughput greater than 1Gbit/s and less than or equal to 5Gbit/s

Medium (M) - Supported throughput greater than 5Gbit/s and less than or equal to 10Gbit/s

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

#### Authors' Addresses

Balamuhunthan Balarajah  
Berlin  
Germany

Email: [bm.balarajah@gmail.com](mailto: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: November 18, 2021

G. Lencse  
Szechenyi Istvan University  
K. Shima  
IIJ Innovation Institute  
May 17, 2021

Benchmarking Methodology for Stateful NATxy Gateways using RFC 4814  
Pseudorandom Port Numbers  
draft-lencse-bmwg-benchmarking-stateful-00

## 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 NAT (NAT44, NAT64, NAT66) technologies. We discuss why using pseudorandom port numbers with stateful NAT gateways is a hard problem and recommend a solution.

## Status of This Memo

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

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

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

This Internet-Draft will expire on November 18, 2021.

## Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|   |    |
|---|----|
| 1. Introduction . . . . .   | 2  |
| 1.1. Requirements Language . . . . .                                    | 3  |
| 2. Pseudorandom Port Numbers and Stateful Translation . . . . .         | 3  |
| 3. Test Setup and Terminology . . . . .                                 | 4  |
| 4. Recommended Benchmarking Method . . . . .                            | 5  |
| 4.1. Restricted Port Number Ranges . . . . .                            | 5  |
| 4.2. Preliminary Test Phase . . . . .                                   | 6  |
| 4.3. Control of the Connection Tracking Table Entries . . . . .         | 7  |
| 4.4. Measurement of the Maximum Connection Establishment Rate . . . . . | 8  |
| 4.5. Real Test Phase . . . . .  | 8  |
| 4.6. Writing and Reading Order of the State Table . . . . .             | 9  |
| 4.7. Peculiarities of Stateful Testing . . . . .                        | 10 |
| 4.7.1. Timeout Budget . . . . .   | 10 |
| 4.7.2. Special Warning Against Non-zero Frame Loss Testing . . . . .    | 10 |
| 5. Implementation and Experience . . . . .                              | 10 |
| 6. Limitations of using UDP as Transport Layer Protocol . . . . .       | 11 |
| 7. Acknowledgements . . . . .   | 11 |
| 8. IANA Considerations . . . . .  | 11 |
| 9. Security Considerations . . . . .                                    | 11 |
| 10. References . . . . .  | 11 |
| 10.1. Normative References . . . . .                                    | 11 |
| 10.2. Informative References . . . . .                                  | 12 |
| Appendix A. Change Log . . . . .  | 13 |
| A.1. 00 . . . . .   | 13 |
| Authors' Addresses . . . . .  | 13 |

## 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.

- o 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. 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.

- o As for the public to private direction, the stateful DUT 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.

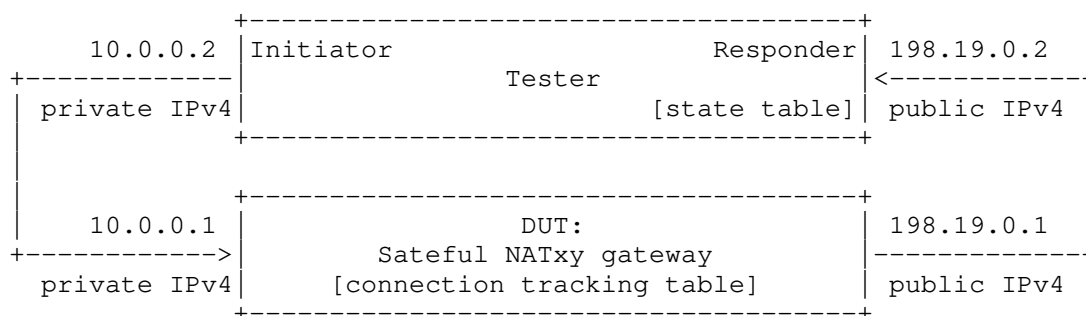


Figure 1: Test Setup for benchmarking stateful NATxy gateways

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.

- o 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 is unknown for the Tester.
- o Four tuple: The four numbers that identify a connection are source IP address, source port number, destination IP address, destination port number.
- o 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:

if they do not belong to an existing connection, the DUT will register a new connection into its connection tracking table.

- o 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.
- o 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 4.6.
- o 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.
- o 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 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 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 4.4).

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:

- o The applied frame rate be safely lower than the maximum connection establishment rate.
- o The initial transient of the filling of the connection tracking table of the DUT be finished.
- o 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.

- o The connections do not time out in the DUT even during the beginning of the real test phase.

#### 4.3. Control of the Connection Tracking Table Entries

The number of the entries in the connection tracking table of the DUT MAY be controlled by using all different source port number destination port number combinations.

Let NF and NC denote the number of test frames to send and the number of all possible source port number destination port number combinations, respectively.

1. If NF and NC are in the same order of magnitude, then the all different source port number destination port number combinations 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].
2. If NF is at least an order of magnitude less than NC, then a simpler solution may be used: the Initiator registers in a table and then it checks if a given source port number destination port number combination was already used (and if yes, then it MUST generate a new one).
3. If NF is at least two orders of magnitude less than NC, then mostly different source port number destination port number combinations can be generated without any specific provision.

Important warning: in normal router testing, the port number selection algorithm (whether it is pseudo-random or enumerated) 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].

[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.

The enumeration of the source port number destination port number combinations in increasing order MAY be used as an additional measurement to discover worst case performance.

#### 4.4. 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 (or mostly) 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. All different source port number destination port number combinations SHOULD be used, if the results of the measurement are published. Mostly different source port number destination port number combinations MAY be used, if the results of the measurement are not published, but they are used only to determine a good enough frame rate for the preliminary test phase preparing the test system for the real test phase.
2. In practice, we RECOMMEND the usage of binary search.
3. As for the successful translation, the Responder MAY (or SHOULD?) check that the source IP address is different than the original source IP address set by the Initiator.

#### 4.5. Real Test Phase

As for the traffic direction, there are three possible cases during the real test phase:

- o bidirectional traffic: The Initiator sends test frames to the Responder and the Responder sends test frames to the Initiator.

- o 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.
- o 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 4.6). 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].

#### 4.6. 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 thus there is no need to handle timeout.

The Responder can read its state table in various orders. We RECOMMEND one of the following ones:

- o round robin
- o pseudorandom (with restriction!)
- o random permutation (no position is repeated until all positions are used).

Pseudorandom reading order of the state table MAY NOT be used with unidirectional traffic from the Responder to the Initiator, because if a four tuple is not used until timeout time, then its connection is deleted from the connection tracking table of the DUT and a later use of the given four tuple will cause frame loss. There is no such problem, when bidirectional traffic is used, because then the state table of the Responder is periodically refreshed.

We do not see any problem in the round robin reading order, because the state table is filled using pseudorandom port numbers.

#### 4.7. Peculiarities of Stateful Testing

Stateful testing involves some issues not present in stateless testing.

##### 4.7.1. Timeout Budget

Even though we do black box testing, one MUST consider timeout and carefully manage timeout budget. For example, if the frame rate is high enough, then every single entry of the state table of the Responder is refreshed within timeout time and it prevents frame sending with a stale four tuple. If the entries of the state table are not refreshed (due to testing with single directional traffic from the Responder to the Initiator) then using all four tuples within timeout time can keep all connection tracking table entries of the DUT alive.

Special care should be taken for the lower frame rate in the preliminary phase.

If the binary search (or the decreasing of the applied frame rates during the frame loss rate test) results in a frame rate that is too low to keep alive the connection tracking table entries, then it results in the failure of the consecutive tests (the binary search of the throughput test counts down to zero).

##### 4.7.2. Special Warning Against Non-zero Frame Loss Testing

Several network performance tester vendors include a parameter called "Loss Tolerance" (or similar) for the throughput test and several benchmarking professionals actually use nonzero values [TOL2001]. If frames are lost during stateful testing (especially if it happens during a test with unidirectional traffic from the Responder to the Initiator) the refreshing of the corresponding connection tracking table element is not ensured and it may result in the loss of further frames (not due to the low performance of the DUT, but due to using a stale four tuple).

#### 5. Implementation and Experience

The "stateful" branch of siitperf [SIITPERF] is a partial implementation of this concept.

Our experience is documented in a paper currently under review [LEN2021].

## 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.

## 7. Acknowledgements

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

## 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.

## 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>>.
- [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, Dec 2020, <[https://www.iiij.ad.jp/en/dev/iir/pdf/iir\\_vol49\\_report\\_EN.pdf](https://www.iiij.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", under review in Computer Communications, may be revised or removed without notice, 2021, <<http://www.hit.bme.hu/~lencse/publications/SFNAT64-tester-for-review.pdf>>.

[SIITPERF]

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

[TOL2001]

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

## Appendix A. Change Log

### A.1. 00

Initial version.

#### Authors' Addresses

Gabor Lencse  
Szechenyi Istvan University  
Egyetem ter 1.  
Gyor H-9026  
Hungary

Email: [lencse@sze.hu](mailto:lencse@sze.hu)

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

Email: [keiichi@iijlab.net](mailto:keiichi@iijlab.net)



Benchmarking Methodology Working Group  
Internet-Draft  
Intended status: Informational  
Expires: 5 September 2022

G. Lencse  
Szechenyi Istvan University  
K. Shima  
IIJ Innovation Institute  
4 March 2022

Benchmarking Methodology for Stateful NATxy Gateways using RFC 4814  
Pseudorandom Port Numbers  
draft-lencse-bmwg-benchmarking-stateful-03

## 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 NAT (NAT44, NAT64, NAT66) technologies. We discuss why using pseudorandom port numbers with stateful NAT gateways is a hard problem and recommend a solution.

## 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 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 . . . . .  | 2  |
| 1.1. Requirements Language . . . . .                                       | 3  |
| 2. Pseudorandom Port Numbers and Stateful Translation . . . . .            | 3  |
| 3. Test Setup and Terminology . . . . .                                    | 4  |
| 4. Recommended Benchmarking Method . . . . .                               | 5  |
| 4.1. Restricted Port Number Ranges . . . . .                               | 5  |
| 4.2. Preliminary Test Phase . . . . .                                      | 6  |
| 4.3. Consideration of the Cases of Stateful Operation . . . . .            | 7  |
| 4.4. Control of the Connection Tracking Table Entries . . . . .            | 8  |
| 4.5. Measurement of the Maximum Connection Establishment<br>Rate . . . . . | 9  |
| 4.6. Real Test Phase . . . . .   | 10 |
| 4.7. Measurement of the Connection Tear Down Rate . . . . .                | 11 |
| 4.8. Writing and Reading Order of the State Table . . . . .                | 11 |
| 5. Implementation and Experience . . . . .                                 | 12 |
| 6. Limitations of using UDP as Transport Layer Protocol . . . . .          | 12 |
| 7. Acknowledgements . . . . .  | 12 |
| 8. IANA Considerations . . . . .   | 13 |
| 9. Security Considerations . . . . .                                       | 13 |
| 10. References . . . . .   | 13 |
| 10.1. Normative References . . . . .                                       | 13 |
| 10.2. Informative References . . . . .                                     | 13 |
| Appendix A. Change Log . . . . .   | 15 |
| A.1. 00 . . . . .  | 15 |
| A.2. 01 . . . . .  | 15 |
| A.3. 02 . . . . .  | 15 |
| A.4. 03 . . . . .  | 15 |
| Authors' Addresses . . . . .   | 15 |

## 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. 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.

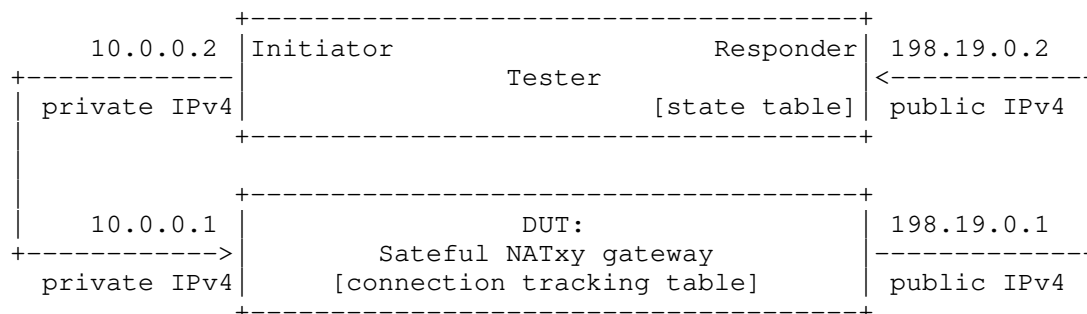


Figure 1: Test Setup for benchmarking stateful NATxy gateways

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 4.8.
- \* 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 direction). 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 entries 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.

#### 4.4. 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 all different 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.

The second extreme situation can be achieved by

- \* using all different source port number destination port number combinations in the preliminary phase and
- \* 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 pseudorandom all different source port number destination port number combinations 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]. This method also satisfies the criterion for the second case that all possible source port number destination port number combinations must be enumerated during the preliminary phase.

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 (or SHOULD?) check that the source IP address is different than the original source IP address set by the Initiator.

#### 4.6. 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 4.8). 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].

#### 4.7. 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:

$$\text{connection tear down rate} = N / (\text{TimestampB} - \text{TimestampA})$$

The connection tear down rate SHOULD be measured for various (important) 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.

#### 4.8. 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 a paper currently under second review [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 using 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-01, 21 February 2022, <<https://www.ietf.org/archive/id/draft-lencse-v6ops-transition-scalability-01.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.iiij.ad.jp/en/dev/iir/pdf/iir\\_vol49\\_report\\_EN.pdf](https://www.iiij.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", revised version, under second review in Computer Communications, may be revised or removed without notice, 2022, <<http://www.hit.bme.hu/~lencse/publications/SFNATxy-tester-revised.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 4.7 about the method for measuring connection tear down rate.

Updates for Section 5 about the implementation and experience.

## 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



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 12, 2022

V. Vassilev  
Lightside Instruments AS  
July 11, 2021

A YANG Data Model for Network Interconnect Tester Management  
draft-vassilev-bmwg-network-interconnect-tester-06

## 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 January 12, 2022.

## Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

|  |    |
|--|----|
| 1. Introduction . . . . .                                | 2  |
| 1.1. Terminology . . . . .                               | 2  |
| 1.1.1. Definitions and Acronyms . . . . .                | 2  |
| 1.1.2. Tree Diagram . . . . .                            | 3  |
| 1.2. Problem Statement . . . . .                         | 3  |
| 1.3. Objectives . . . . .                                | 3  |
| 1.4. Solution . . . . .                                  | 4  |
| 2. Using the network interconnect tester model . . . . . | 5  |
| 3. Traffic Generator Module Tree Diagram . . . . .       | 5  |
| 4. Traffic Analyzer Module Tree Diagram . . . . .        | 6  |
| 5. Traffic Generator Module YANG . . . . .               | 8  |
| 6. Traffic Analyzer Module YANG . . . . .                | 15 |
| 7. IANA Considerations . . . . .                         | 23 |
| 7.1. URI Registration . . . . .                          | 23 |
| 7.2. YANG Module Name Registration . . . . .             | 24 |
| 8. Security Considerations . . . . .                     | 24 |
| 8.1. ietf-traffic-generator.yang . . . . .               | 24 |
| 8.2. ietf-traffic-analyzer.yang . . . . .                | 25 |
| 9. References . . . . .                                  | 25 |
| 9.1. Normative References . . . . .                      | 25 |
| 9.2. Informative References . . . . .                    | 26 |
| Appendix A. Examples . . . . .                           | 26 |
| A.1. Basic Test Program . . . . .                        | 26 |
| A.2. Generating RFC2544 Testframes . . . . .             | 28 |
| Author's Address . . . . .                               | 28 |

## 1. Introduction

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

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

## 1.1. Terminology

## 1.1.1. Definitions and Acronyms

DUT: Device Under Test

TA: Traffic Analyzer

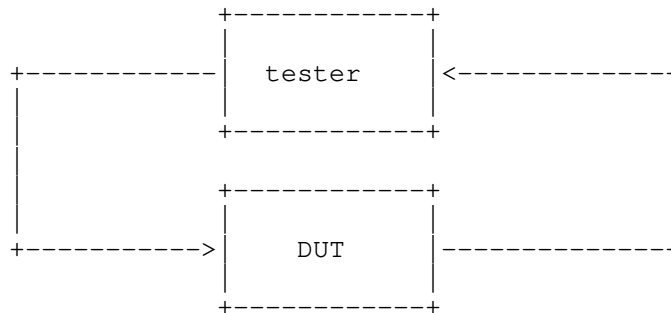
TG: Traffic Generator

### 1.1.2. Tree Diagram

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

### 1.2. Problem Statement

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



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

### 1.3. Objectives

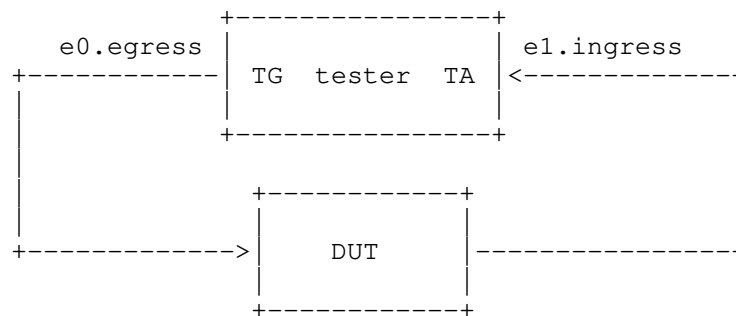
This section describes some of the design objectives for the model. It should:

- o 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.
- o have a mandatory single stream mode and optional multi stream mode.

- o 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.
- o provide means for configuration of traffic streams with dynamic frame data where frames contain fields with dynamic data like generation time and sequence number.
- o allow third parties to augment the base module with alternative dynamic fields of frame data extensions.
- o provide means for realtime synchronization and orchestration of the generated streams.
- o provide counters for received test traffic frames and octets.
- o provide latency statistic in the case of test traffic with dynamic frame data that includes timestamp.
- o provide sequence number errors in the case of test traffic with dynamic frame data that includes sequence number.

#### 1.4. Solution

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



## 2. Using the network interconnect tester model

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

## 3. Traffic Generator Module Tree Diagram

```

module: ietf-traffic-generator
  augment /if:interfaces/if:interface:
    +--rw traffic-generator {egress-direction}?
      +--rw (type)?
        +--:(single-stream)
          +--rw testframe-type?      identityref
          +--rw frame-size           uint32
          +--rw frame-data?          string
          +--rw interframe-gap       uint32
          +--rw interburst-gap?      uint32
          +--rw frames-per-burst?    uint32
          +--rw src-mac-address?     yang:mac-address {ethernet}?
          +--rw dst-mac-address?     yang:mac-address {ethernet}?
          +--rw ether-type?          uint16 {ethernet}?
        +--:(multi-stream)
          +--rw streams
            +--rw stream* [id]
              +--rw id               uint32
              +--rw testframe-type?  identityref
              +--rw frame-size       uint32
              +--rw frame-data?      string
              +--rw interframe-gap   uint32
              +--rw interburst-gap?  uint32
              +--rw frames-per-burst uint32
              +--rw frames-per-stream uint32
              +--rw interstream-gap  uint32
              +--rw src-mac-address?
                | yang:mac-address {ethernet}?
              +--rw dst-mac-address?
                | yang:mac-address {ethernet}?
              +--rw ether-type?      uint16 {ethernet}?
          +--rw realtime-epoch?
            | yang:date-and-time {realtime-epoch}?
          +--rw total-frames?      uint64
      +--rw traffic-generator-ingress {ingress-direction}?

```

```

+--rw (type)?
+--:(single-stream)
|   +--rw testframe-type?      identityref
|   +--rw frame-size          uint32
|   +--rw frame-data?         string
|   +--rw interframe-gap      uint32
|   +--rw interburst-gap?     uint32
|   +--rw frames-per-burst?   uint32
|   +--rw src-mac-address?     yang:mac-address {ethernet}?
|   +--rw dst-mac-address?     yang:mac-address {ethernet}?
|   +--rw ether-type?         uint16 {ethernet}?
+--:(multi-stream)
|   +--rw streams
|   |   +--rw stream* [id]
|   |   |   +--rw id          uint32
|   |   |   +--rw testframe-type? identityref
|   |   |   +--rw frame-size  uint32
|   |   |   +--rw frame-data? string
|   |   |   +--rw interframe-gap uint32
|   |   |   +--rw interburst-gap? uint32
|   |   |   +--rw frames-per-burst? uint32
|   |   |   +--rw frames-per-stream uint32
|   |   |   +--rw interstream-gap uint32
|   |   |   +--rw src-mac-address?
|   |   |   |   yang:mac-address {ethernet}?
|   |   |   +--rw dst-mac-address?
|   |   |   |   yang:mac-address {ethernet}?
|   |   |   +--rw ether-type?
|   |   |   |   uint16 {ethernet}?
|   +--rw realtime-epoch?
|   |   yang:date-and-time {realtime-epoch}?
+--rw total-frames?      uint64

```

#### 4. Traffic Analyzer Module Tree Diagram

```

module: ietf-traffic-analyzer
augment /if:interfaces/if:interface:
+--rw traffic-analyzer! {ingress-direction}?
|   +--rw filter! {filter}?
|   |   +--rw type      identityref
|   |   +--rw ether-type? uint16
|   +--rw capture {capture}?
|   |   +--rw start-trigger
|   |   |   +--rw (start-trigger)?
|   |   |   |   +--:(frame-index)
|   |   |   |   |   +--rw frame-index?      uint64
|   |   |   |   +--:(testframe-index)
|   |   |   |   |   +--rw testframe-index?   uint64

```

```

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

```

```

    +--ro payload-errors?      yang:counter64
    +--ro latency
      +--ro samples?          uint64
      +--ro min?              uint64
      +--ro max?              uint64
      +--ro average?          uint64
      +--ro latest?           uint64
    +--ro capture {capture}?
      +--ro frame* [sequence-number]
        +--ro sequence-number      uint64
        +--ro timestamp?           yang:date-and-time
        +--ro length?              uint32
        +--ro preceding-interframe-gap? uint32
        +--ro data?                string

```

## 5. Traffic Generator Module YANG

<CODE BEGINS> file "ietf-traffic-generator@2021-07-11.yang"

```

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

  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model For Interface Management";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import iana-if-type {
    prefix ianaift;
    reference
      "RFC 7224: IANA Interface Type YANG Module";
  }

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

    Editor:   Vladimir Vassilev
              <mailto:vladimir@lightside-instruments.com>";

```



## description

"This module contains a collection of YANG definitions for description and management of network interconnect testers.

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.";

## revision 2021-07-11 {

## description

"Initial revision.";

## reference

"RFC XXXX: A YANG Data Model for  
Network Interconnect Tester Management";

}

## feature egress-direction {

## description

"The device can generate traffic in the egress direction.";

}

## feature ingress-direction {

## description

"The device can generate traffic in the ingress direction.";

}

## feature multi-stream {

## description

"The device can generate multi-stream traffic.";

}

## feature ethernet {

## description

"The device can generate ethernet traffic.";

}

## feature realtime-epoch {

## description

"The device can generate traffic precisely  
at configured realtime epoch.";

```
}

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

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

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

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

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

```
grouping burst-data {
  description
    "Generated traffic burst parameters.";
  leaf testframe-type {
    type identityref {
      base tg:testframe-type;
    }
    default "tg:static";
    description
      "In case of dynamic testframes this leaf specifies
      the dynamic testframe identity.";
  }
  leaf frame-size {
    type uint32;
    mandatory true;
    description
      "Size of the frames generated. For example for
      ethernet interfaces the following definition
      applies:

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

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

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

```
type uint32;
mandatory true;
description
  "Length of the idle period between generated frames.
  For example for ethernet interfaces the following
  definition applies:

  Ethernet interframe-gap between transmission of frames
  known as the interframe gap (IFG). A brief recovery time
  between frames allows devices to prepare for
  reception of the next frame. The minimum
  interframe gap is 96 bit times (12 octet times) (the time it
  takes to transmit 96 bits (12 octets) of raw data on the
  medium). However the preamble (7 octets) and start of
  frame delimiter (1 octet) are considered a constant gap that
  should be included in the interframe-gap. Thus the minimum
  value for standard ethernet transmission should be considered
  20 octets."
}
leaf interburst-gap {
  type uint32;
  description
    "Similar to the interframe-gap but takes place between
    any two bursts of the stream."
}
leaf frames-per-burst {
  type uint32;
  description
    "Number of frames contained in a burst";
}
}

grouping multi-stream-data {
  description
    "Multi stream traffic generation parameters.";
  container streams {
    description
      "Non-presence container holding the configured stream list.";
    list stream {
      key "id";
      description
        "Each stream repeats a burst until frames-per-stream
        count is reached followed by interstream-gap delay.";
      leaf id {
        type uint32;
        description
          "Number specifying the order of the stream.";
      }
    }
  }
}
```

```
    uses burst-data;
    leaf frames-per-stream {
        type uint32;
        mandatory true;
        description
            "The count of frames to be generated before
             generation of the next stream is started.";
    }
    leaf interstream-gap {
        type uint32;
        mandatory true;
        description
            "Idle period after the last frame of the last burst.";
    }
}
}

grouping ethernet-data {
    description
        "Ethernet frame data specific parameters.";
    reference
        "IEEE 802-2014 Clause 9.2";
    leaf src-mac-address {
        type yang:mac-address;
        description
            "Source Address field of the generated Ethernet packet.";
    }
    leaf dst-mac-address {
        type yang:mac-address;
        description
            "Destination Address field of the generated Ethernet packet.";
    }
    leaf ether-type {
        type uint16;
        description
            "Length/Type field of the generated Ethernet packet.";
    }
}

augment "/if:interfaces/if:interface" {
    description
        "Traffic generator augmentations of ietf-interfaces.";
    container traffic-generator {
        if-feature "egress-direction";
        description
            "Traffic generator for egress direction.";
        choice type {
```

```
    description
      "Choice of the type of the data model of the generator.
       Single or multi stream.";
    case single-stream {
      uses burst-data;
    }
    case multi-stream {
      uses multi-stream-data;
    }
  }
  uses common-data;
}
container traffic-generator-ingress {
  if-feature "ingress-direction";
  description
    "Traffic generator for ingress direction.";
  choice type {
    description
      "Choice of the type of the data model of the generator.
       Single or multi stream.";
    case single-stream {
      uses burst-data;
    }
    case multi-stream {
      uses multi-stream-data;
    }
  }
  uses common-data;
}
}

augment "/if:interfaces/if:interface/tg:traffic-generator/tg:type/"
+ "tg:single-stream" {
  when "derived-from-or-self(..../if:type, 'ianaift:ethernetCsmacd')" {
    description
      "Ethernet interface type.";
  }
  if-feature "ethernet";
  description
    "Ethernet specific augmentation for egress
     single stream generator type.";
  uses ethernet-data;
}

augment "/if:interfaces/if:interface/tg:traffic-generator/"
+ "tg:type/tg:multi-stream/tg:streams/tg:stream" {
  when "derived-from-or-self(..../if:type,"
+ "'ianaift:ethernetCsmacd')" {
```

```
        description
            "Ethernet interface type.";
    }
    if-feature "ethernet";
    description
        "Ethernet specific augmentation for egress
        multi stream generator type.";
    uses ethernet-data;
}

augment "/if:interfaces/if:interface/tg:traffic-generator-ingress/"
    + "tg:type/tg:single-stream" {
    when "derived-from-or-self(..../if:type, 'ianaift:ethernetCsmacd')" {
        description
            "Ethernet interface type.";
    }
    if-feature "ethernet";
    description
        "Ethernet specific augmentation for ingress
        single stream generator type.";
    uses ethernet-data;
}

augment "/if:interfaces/if:interface/tg:traffic-generator-ingress/"
    + "tg:type/tg:multi-stream/tg:streams/tg:stream" {
    when "derived-from-or-self(..../..../if:type, "
        + "'ianaift:ethernetCsmacd')" {
        description
            "Ethernet interface type.";
    }
    if-feature "ethernet";
    description
        "Ethernet specific augmentation for ingress
        multi stream generator type.";
    uses ethernet-data;
}
}

<CODE ENDS>
```

## 6. Traffic Analyzer Module YANG

```
<CODE BEGINS> file "ietf-traffic-analyzer@2021-07-11.yang"

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

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

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

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

  Copyright (c) 2021 IETF Trust and the persons identified as
  authors of the code.  All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).

  This version of this YANG module is part of RFC XXXX; see
  the RFC itself for full legal notices.";

revision 2021-07-11 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for
    Network Interconnect Tester Management";
}

feature egress-direction {
  description
    "The device can analyze traffic from the egress direction.";
}
```



```
feature ingress-direction {
  description
    "The device can generate traffic from the ingress direction.";
}

feature filter {
  description
    "This feature indicates that the device implements
    filter that can specify a subset of packets to be
    analyzed when filtering is enabled.";
}

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

feature capture {
  description
    "This feature indicates that the device implements
    packet capture functionality.";
}

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

identity ethernet {
  base ta:filter;
  description
    "Ethernet packet fields filter.";
}

grouping statistics-data {
  description
    "Analyzer statistics.";
  leaf pkts {
    type yang:counter64;
    description
      "Total number of packets analyzed.";
  }
  leaf octets {
    type yang:counter64;
    description
      "This counter is identical with the in-octets/out-octets
```

```
        counters defined in RFC8343 except that it counts the
        octets since the analyzer was created.";
    }
    leaf idle-octets {
        if-feature "idle-octets-counter";
        type yang:counter64;
        description
            "Total accumulated period with no frame transmission
            taking place measured in octets at the current link
            speed. Octets not counted in ../octets but not idle are
            for example layer 1 framing octets - for Ethernet links
            7+1 preamble octets per packet.";
    }
    leaf errors {
        type yang:counter64;
        description
            "Count of packets with errors.
            Not counted in the pkts or captured.
            For example packets with CRC error.";
    }
    container testframe-stats {
        description
            "Statistics for testframe packets containing
            either sequence number, payload checksum,
            timestamp or any combination of these features.";
        leaf testframe-pkts {
            type yang:counter64;
            description
                "Total count of detected testframe packets.";
        }
        leaf sequence-errors {
            type yang:counter64;
            description
                "Total count of testframe packets with
                unexpected sequence number. After each sequence
                error the expected next sequence number is
                updated.";
        }
        leaf payload-errors {
            type yang:counter64;
            description
                "Total count of testframe packets with
                payload errors.";
        }
    }
    container latency {
        description
            "Latency statistics.";
        leaf samples {
```

```
    type uint64;
    description
      "Total count of packets used for estimating
       the latency statistics. Ideally
       samples=../testframe-stats.";
  }
  leaf min {
    type uint64;
    units "nanoseconds";
    description
      "Minimum measured latency.";
  }
  leaf max {
    type uint64;
    units "nanoseconds";
    description
      "Maximum measured latency.";
  }
  leaf average {
    type uint64;
    units "nanoseconds";
    description
      "The sum of all sampled latencies divided
       by the number of samples.";
  }
  leaf latest {
    type uint64;
    units "nanoseconds";
    description
      "Latency of the latest sample.";
  }
}
}
}

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

    description
      "Contains capture parameters.";

    container start-trigger {
      description
        "Configures when the capture start is triggered.";
      choice start-trigger {
```

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

```
grouping capture-data {
  description
    "Grouping with statistics and data
    of one or more captured frame.";
  container capture {
    if-feature "capture";
    description
      "Statistics and data of
      one or more captured frames.";
    list frame {
      key "sequence-number";
      description
        "Statistics and data of a captured frame.";
      leaf sequence-number {
        type uint64;
        description
          "Incremental counter of frames captured.";
      }
      leaf timestamp {
        type yang:date-and-time;
        description
          "Timestamp of the moment the frame was captured.";
      }
      leaf length {
        type uint32;
        description
          "Frame length. Ideally the data captured will be
          of the same length but can be shorter
          depending on implementation limitations.";
      }
      leaf preceding-interframe-gap {
        type uint32;
        units "nanoseconds";
        description
          "Measured delay between the reception of the previous
          frame was completed and the reception of the current
          frame was started.";
      }
      leaf data {
        type string {
          pattern '([0-9A-F]{2})*';
        }
        description
          "Raw data of the captured frame.";
      }
    }
  }
}
```

```
grouping filter-data {
  description
    "Grouping with a filter container specifying the filtering
    rules for processing only a specific subset of the
    frames.";
  container filter {
    if-feature "filter";
    presence "When present packets are
    filtered before analyzed according
    to the filter type";
    description
      "Contains the filtering rules for processing only
      a specific subset of the frames.";
    leaf type {
      type identityref {
        base ta:filter;
      }
      mandatory true;
      description
        "Type of the applied filter. External modules can
        define alternative filter type identities.";
    }
  }
}

augment "/if:interfaces/if:interface" {
  description
    "Traffic analyzer augmentations of ietf-interfaces.";
  container traffic-analyzer {
    if-feature "ingress-direction";
    presence "Enables the traffic analyzer for ingress traffic.";
    description
      "Traffic analyzer for ingress direction.";
    uses filter-data;
    uses capture-config-data;
    container state {
      config false;
      description
        "State data.";
      uses statistics-data;
      uses capture-data;
    }
  }
  container traffic-analyzer-egress {
    if-feature "egress-direction";
    presence "Enables the traffic analyzer for egress traffic.";
    description
      "Traffic analyzer for egress direction.";
```

```
    uses filter-data;
    uses capture-config-data;
    container state {
        config false;
        description
            "State data.";
        uses statistics-data;
        uses capture-data;
    }
}

augment "/if:interfaces/if:interface/ta:traffic-analyzer/ta:filter" {
    when "derived-from-or-self(ta:type, 'ta:ethernet')";
    description
        "Ethernet frame specific filter type.";
    leaf ether-type {
        type uint16;
        description
            "The Ethernet Type (or Length) value
            defined by IEEE 802.";
        reference
            "IEEE 802-2014 Clause 9.2";
    }
}
}
```

<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: tg
reference: RFC XXXX
```

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

## 8. Security Considerations

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:

- o /if:interfaces/if:interface/tg: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:

- o /if:interfaces/if:interface/ta:traffic-analyzer/ta:capture

## 9. References

### 9.1. Normative References

- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<https://www.rfc-editor.org/info/rfc6536>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", RFC 7224, DOI 10.17487/RFC7224, May 2014, <<https://www.rfc-editor.org/info/rfc7224>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.

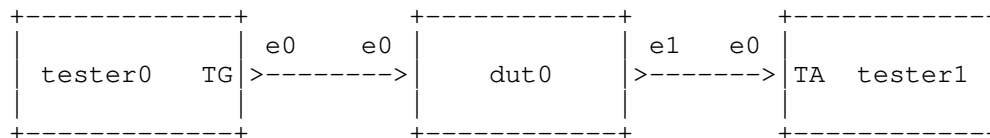
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.

## 9.2. Informative References

- [IEEE1588] IEEE, "IEEE 1588-2008", 2008.
- [IEEE802.3-2014] IEEE WG802.3 - Ethernet Working Group, "IEEE 802.3-2014", 2014.
- [RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <<https://www.rfc-editor.org/info/rfc2544>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.

## Appendix A. Examples

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



### A.1. Basic Test Program

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

```

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

# Configure DUTs and enable traffic-analyzers
net.node("dut0").edit( \
    "create /interfaces/interface[name='e0'] -- type=ethernetCsmacd")
  
```

```
net.node("dut0").edit(  
    "create /interfaces/interface[name='e1'] -- type=ethernetCsmacd")  
net.node("dut0").edit(  
    "create /flows/flow[id='t0'] -- match/in-port=e0 "  
    "actions/action[order='0']/output-action/out-port=e1")  
  
net.node("tester1").edit(  
    "create /interfaces/interface[name='e0']/traffic-analyzer")  
net.commit()  
  
#Get network state - before  
before=net.get()  
  
# Start traffic  
net.node("tester0").edit(  
    "create /interfaces/interface[name='e0']/traffic-generator -- "  
    "frame-size=64 interframe-gap=20")  
  
net.commit()  
  
time.sleep(60)  
  
# Stop traffic  
net.node("tester1").edit("delete /interfaces/interface[name='e0']/"  
    "traffic-generator")  
net.commit()  
  
#Get network state - after  
after=net.get()  
  
#Report  
sent_pkts=delta("tester0",before,after,  
    "/interfaces/interface[name='e0']/statistics/out-unicast-pkts")  
  
received_pkts=delta("tester1",before,after,  
    "/interfaces/interface[name='e0']/statistics/in-unicast-pkts")  
  
latency_max=absolute(after,  
    "/interfaces/interface[name='e0']/traffic-analyzer/state/"  
    "testframe-stats/latency/max")  
  
#Cleanup  
net.node("tester1").edit(  
    "delete /interfaces/interface/traffic-analyzer")  
net.node("dut0").edit("delete /flows")  
net.node("dut0").edit("delete /interfaces")  
net.commit()
```

## A.2. Generating RFC2544 Testframes

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

```
...
net.node("tester1").edit(
  "merge /interfaces/interface[name='e0']/"
  "traffic-generator -- frame-data="
  "6CA96F00000026CA96F000000108004500"
  "002ED4A5000000A115816C0000201C000"
  "0202C0200007001A00000010203040506"
  "0708090A0B0C0D0E0F101112")
...
```

## Author's Address

Vladimir Vassilev  
Lightside Instruments AS

Email: [vladimir@lightside-instruments.com](mailto:vladimir@lightside-instruments.com)

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 28 April 2022

V. Vassilev  
Lightside Instruments AS  
25 October 2021

A YANG Data Model for Network Interconnect Tester Management  
draft-vassilev-bmwg-network-interconnect-tester-07

## Abstract

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

## Status of This Memo

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

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

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

This Internet-Draft will expire on 28 April 2022.

## Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

|  |    |
|--|----|
| 1. Introduction . . . . .                                | 2  |
| 1.1. Terminology . . . . .                               | 2  |
| 1.1.1. Definitions and Acronyms . . . . .                | 2  |
| 1.1.2. Tree Diagram . . . . .                            | 3  |
| 1.2. Problem Statement . . . . .                         | 3  |
| 1.3. Objectives . . . . .                                | 3  |
| 1.4. Solution . . . . .                                  | 4  |
| 2. Using the network interconnect tester model . . . . . | 5  |
| 3. Traffic Generator Module Tree Diagram . . . . .       | 5  |
| 4. Traffic Analyzer Module Tree Diagram . . . . .        | 6  |
| 5. Traffic Generator Module YANG . . . . .               | 8  |
| 6. Traffic Analyzer Module YANG . . . . .                | 15 |
| 7. IANA Considerations . . . . .                         | 23 |
| 7.1. URI Registration . . . . .                          | 23 |
| 7.2. YANG Module Name Registration . . . . .             | 24 |
| 8. Security Considerations . . . . .                     | 24 |
| 8.1. ietf-traffic-generator.yang . . . . .               | 24 |
| 8.2. ietf-traffic-analyzer.yang . . . . .                | 25 |
| 9. References . . . . .                                  | 25 |
| 9.1. Normative References . . . . .                      | 25 |
| 9.2. Informative References . . . . .                    | 26 |
| Appendix A. Examples . . . . .                           | 26 |
| A.1. Basic Test Program . . . . .                        | 26 |
| A.2. Generating RFC2544 Testframes . . . . .             | 28 |
| Author's Address . . . . .                               | 28 |

## 1. Introduction

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

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

## 1.1. Terminology

## 1.1.1. Definitions and Acronyms

DUT: Device Under Test

TA: Traffic Analyzer

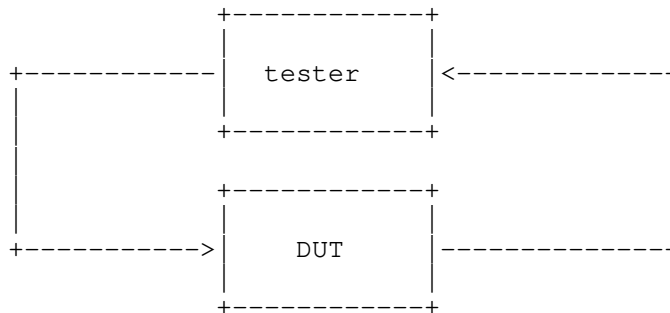
TG: Traffic Generator

### 1.1.2. Tree Diagram

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

### 1.2. Problem Statement

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



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

### 1.3. Objectives

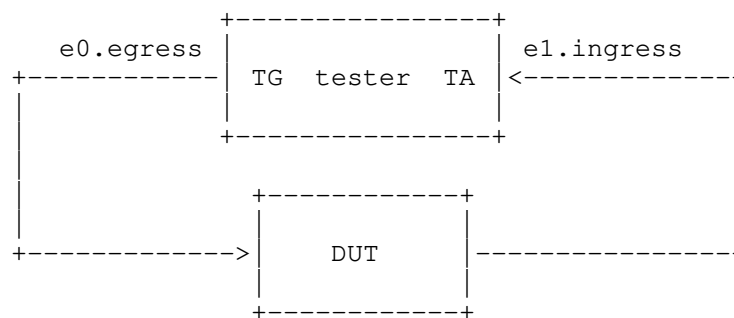
This section describes some of the design objectives for the model. It should:

- \* provide means to specify the generated traffic as streams of cyclic sequence of bursts with configurable frame size, frame data, interframe gap and interburst gap.
- \* have a mandatory single stream mode and optional multi stream mode.

- \* provide means for configuration of traffic streams with static frame data where frames with identical frame data are sent during the lifetime of the stream.
- \* provide means for configuration of traffic streams with dynamic frame data where frames contain fields with dynamic data like generation time and sequence number.
- \* allow third parties to augment the base module with alternative dynamic fields of frame data extensions.
- \* provide means for realtime synchronization and orchestration of the generated streams.
- \* provide counters for received test traffic frames and octets.
- \* provide latency statistic in the case of test traffic with dynamic frame data that includes timestamp.
- \* provide sequence number errors in the case of test traffic with dynamic frame data that includes sequence number.

#### 1.4. Solution

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





## 2. Using the network interconnect tester model

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

## 3. Traffic Generator Module Tree Diagram

```

module: ietf-traffic-generator
  augment /if:interfaces/if:interface:
    +--rw traffic-generator {egress-direction}?
      +--rw (type)?
        +--:(single-stream)
          +--rw testframe-type?      identityref
          +--rw frame-size           uint32
          +--rw frame-data?          string
          +--rw interframe-gap       uint32
          +--rw interburst-gap?      uint32
          +--rw frames-per-burst?    uint32
          +--rw src-mac-address?     yang:mac-address {ethernet}?
          +--rw dst-mac-address?     yang:mac-address {ethernet}?
          +--rw ether-type?          uint16 {ethernet}?
        +--:(multi-stream)
          +--rw streams
            +--rw stream* [id]
              +--rw id               uint32
              +--rw testframe-type?  identityref
              +--rw frame-size       uint32
              +--rw frame-data?      string
              +--rw interframe-gap   uint32
              +--rw interburst-gap?  uint32
              +--rw frames-per-burst? uint32
              +--rw frames-per-stream uint32
              +--rw interstream-gap  uint32
              +--rw src-mac-address?
                | yang:mac-address {ethernet}?
              +--rw dst-mac-address?
                | yang:mac-address {ethernet}?
              +--rw ether-type?      uint16 {ethernet}?
          +--rw realtime-epoch?
            | yang:date-and-time {realtime-epoch}?
          +--rw total-frames?      uint64
      +--rw traffic-generator-ingress {ingress-direction}?

```

```

+--rw (type)?
|   +--:(single-stream)
|   |   +--rw testframe-type?      identityref
|   |   +--rw frame-size           uint32
|   |   +--rw frame-data?          string
|   |   +--rw interframe-gap       uint32
|   |   +--rw interburst-gap?      uint32
|   |   +--rw frames-per-burst?    uint32
|   |   +--rw src-mac-address?     yang:mac-address {ethernet}?
|   |   +--rw dst-mac-address?     yang:mac-address {ethernet}?
|   |   +--rw ether-type?          uint16 {ethernet}?
|   +--:(multi-stream)
|   |   +--rw streams
|   |   |   +--rw stream* [id]
|   |   |   |   +--rw id           uint32
|   |   |   |   +--rw testframe-type? identityref
|   |   |   |   +--rw frame-size   uint32
|   |   |   |   +--rw frame-data?  string
|   |   |   |   +--rw interframe-gap uint32
|   |   |   |   +--rw interburst-gap? uint32
|   |   |   |   +--rw frames-per-burst? uint32
|   |   |   |   +--rw frames-per-stream uint32
|   |   |   |   +--rw interstream-gap uint32
|   |   |   |   +--rw src-mac-address?
|   |   |   |   |   yang:mac-address {ethernet}?
|   |   |   |   +--rw dst-mac-address?
|   |   |   |   |   yang:mac-address {ethernet}?
|   |   |   |   +--rw ether-type?
|   |   |   |   |   uint16 {ethernet}?
|   +--rw realtime-epoch?
|   |   yang:date-and-time {realtime-epoch}?
+--rw total-frames?      uint64

```

#### 4. Traffic Analyzer Module Tree Diagram

```

module: ietf-traffic-analyzer
augment /if:interfaces/if:interface:
+--rw traffic-analyzer! {ingress-direction}?
|   +--rw filter! {filter}?
|   |   +--rw type      identityref
|   |   +--rw ether-type? uint16
|   +--rw capture {capture}?
|   |   +--rw start-trigger
|   |   |   +--rw (start-trigger)?
|   |   |   |   +--:(frame-index)
|   |   |   |   |   +--rw frame-index?      uint64
|   |   |   |   +--:(testframe-index)
|   |   |   |   |   +--rw testframe-index?   uint64

```

```

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

```

```

    +--ro payload-errors?      yang:counter64
    +--ro latency
      +--ro samples?          uint64
      +--ro min?              uint64
      +--ro max?              uint64
      +--ro average?          uint64
      +--ro latest?           uint64
    +--ro capture {capture}?
      +--ro frame* [sequence-number]
        +--ro sequence-number      uint64
        +--ro timestamp?           yang:date-and-time
        +--ro length?              uint32
        +--ro preceding-interframe-gap? uint32
        +--ro data?                string

```

## 5. Traffic Generator Module YANG

<CODE BEGINS> file "ietf-traffic-generator@2021-10-25.yang"

```

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

  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model For Interface Management";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import iana-if-type {
    prefix ianaift;
    reference
      "RFC 7224: IANA Interface Type YANG Module";
  }

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

    Editor:   Vladimir Vassilev
              <mailto:vladimir@lightside-instruments.com>";

```

## description

"This module contains a collection of YANG definitions for description and management of network interconnect testers.

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>). This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.";

## revision 2021-10-25 {

## description

"Initial revision.";

## reference

"RFC XXXX: A YANG Data Model for Network Interconnect Tester Management";

}

## feature egress-direction {

## description

"The device can generate traffic in the egress direction.";

}

## feature ingress-direction {

## description

"The device can generate traffic in the ingress direction.";

}

## feature multi-stream {

## description

"The device can generate multi-stream traffic.";

}

## feature ethernet {

## description

"The device can generate ethernet traffic.";

}

## feature realtime-epoch {

## description

"The device can generate traffic precisely at configured realtime epoch.";

```
}

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

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

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

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

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

```
grouping burst-data {
  description
    "Generated traffic burst parameters.";
  leaf testframe-type {
    type identityref {
      base nttg:testframe-type;
    }
    default "nttg:static";
    description
      "In case of dynamic testframes this leaf specifies
       the dynamic testframe identity.";
  }
  leaf frame-size {
    type uint32;
    mandatory true;
    description
      "Size of the frames generated. For example for
       ethernet interfaces the following definition
       applies:

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

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

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

```
    type uint32;
    mandatory true;
    description
      "Length of the idle period between generated frames.
      For example for ethernet interfaces the following
      definition applies:

      Ethernet interframe-gap between transmission of frames
      known as the interframe gap (IFG). A brief recovery time
      between frames allows devices to prepare for
      reception of the next frame. The minimum
      interframe gap is 96 bit times (12 octet times) (the time it
      takes to transmit 96 bits (12 octets) of raw data on the
      medium). However the preamble (7 octets) and start of
      frame delimiter (1 octet) are considered a constant gap that
      should be included in the interframe-gap. Thus the minimum
      value for standard ethernet transmission should be considered
      20 octets."
  }
  leaf interburst-gap {
    type uint32;
    description
      "Similar to the interframe-gap but takes place between
      any two bursts of the stream."
  }
  leaf frames-per-burst {
    type uint32;
    description
      "Number of frames contained in a burst";
  }
}

grouping multi-stream-data {
  description
    "Multi stream traffic generation parameters.";
  container streams {
    description
      "Non-presence container holding the configured stream list.";
    list stream {
      key "id";
      description
        "Each stream repeats a burst until frames-per-stream
        count is reached followed by interstream-gap delay.";
      leaf id {
        type uint32;
        description
          "Number specifying the order of the stream."
      }
    }
  }
}
```



```
    uses burst-data;
    leaf frames-per-stream {
        type uint32;
        mandatory true;
        description
            "The count of frames to be generated before
             generation of the next stream is started.";
    }
    leaf interstream-gap {
        type uint32;
        mandatory true;
        description
            "Idle period after the last frame of the last burst.";
    }
}
}

grouping ethernet-data {
    description
        "Ethernet frame data specific parameters.";
    reference
        "IEEE 802-2014 Clause 9.2";
    leaf src-mac-address {
        type yang:mac-address;
        description
            "Source Address field of the generated Ethernet packet.";
    }
    leaf dst-mac-address {
        type yang:mac-address;
        description
            "Destination Address field of the generated Ethernet packet.";
    }
    leaf ether-type {
        type uint16;
        description
            "Length/Type field of the generated Ethernet packet.";
    }
}

augment "/if:interfaces/if:interface" {
    description
        "Traffic generator augmentations of ietf-interfaces.";
    container traffic-generator {
        if-feature "egress-direction";
        description
            "Traffic generator for egress direction.";
        choice type {
```

```
    description
      "Choice of the type of the data model of the generator.
       Single or multi stream.";
    case single-stream {
      uses burst-data;
    }
    case multi-stream {
      uses multi-stream-data;
    }
  }
  uses common-data;
}
container traffic-generator-ingress {
  if-feature "ingress-direction";
  description
    "Traffic generator for ingress direction.";
  choice type {
    description
      "Choice of the type of the data model of the generator.
       Single or multi stream.";
    case single-stream {
      uses burst-data;
    }
    case multi-stream {
      uses multi-stream-data;
    }
  }
  uses common-data;
}
}

augment "/if:interfaces/if:interface/nttg:traffic-generator/"
+ "nttg:type/nttg:single-stream" {
  when "derived-from-or-self(..../if:type, 'ianaift:ethernetCsmacd')" {
    description
      "Ethernet interface type.";
  }
  if-feature "ethernet";
  description
    "Ethernet specific augmentation for egress
     single stream generator type.";
  uses ethernet-data;
}

augment "/if:interfaces/if:interface/nttg:traffic-generator/"
+ "nttg:type/nttg:multi-stream/nttg:streams/nttg:stream" {
  when "derived-from-or-self(..../if:type, "
+ "'ianaift:ethernetCsmacd')" {
```

```
        description
          "Ethernet interface type.";
      }
      if-feature "ethernet";
      description
        "Ethernet specific augmentation for egress
        multi stream generator type.";
      uses ethernet-data;
    }

    augment "/if:interfaces/if:interface/nttg:traffic-generator-ingress/"
      + "nttg:type/nttg:single-stream" {
      when "derived-from-or-self(..../if:type, 'ianaift:ethernetCsmacd')" {
        description
          "Ethernet interface type.";
      }
      if-feature "ethernet";
      description
        "Ethernet specific augmentation for ingress
        single stream generator type.";
      uses ethernet-data;
    }

    augment "/if:interfaces/if:interface/nttg:traffic-generator-ingress/"
      + "nttg:type/nttg:multi-stream/nttg:streams/nttg:stream" {
      when "derived-from-or-self(..../..../if:type,"
        + "'ianaift:ethernetCsmacd')" {
        description
          "Ethernet interface type.";
      }
      if-feature "ethernet";
      description
        "Ethernet specific augmentation for ingress
        multi stream generator type.";
      uses ethernet-data;
    }
  }

<CODE ENDS>
```

## 6. Traffic Analyzer Module YANG

<CODE BEGINS> file "ietf-traffic-analyzer@2021-10-25.yang"

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

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

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

    Editor:    Vladimir Vassilev
               <mailto:vladimir@lightside-instruments.com>;

  description
    "This module contains a collection of YANG definitions for
    description and management of network interconnect testers.

    Copyright (c) 2021 IETF Trust and the persons identified as
    authors of the code.  All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD License
    set forth in Section 4.c of the IETF Trust's Legal Provisions
    Relating to IETF Documents
    (http://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC XXXX; see
    the RFC itself for full legal notices.";

  revision 2021-10-25 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for
      Network Interconnect Tester Management";
  }
}
```

```
feature egress-direction {
  description
    "The device can analyze traffic from the egress direction.";
}

feature ingress-direction {
  description
    "The device can generate traffic from the ingress direction.";
}

feature filter {
  description
    "This feature indicates that the device implements
    filter that can specify a subset of packets to be
    analyzed when filtering is enabled.";
}

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

feature capture {
  description
    "This feature indicates that the device implements
    packet capture functionality.";
}

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

identity ethernet {
  base nttta:filter;
  description
    "Ethernet packet fields filter.";
}

grouping statistics-data {
  description
    "Analyzer statistics.";
  leaf pkts {
    type yang:counter64;
    description
      "Total number of packets analyzed.";
  }
}
```

```
}
leaf octets {
  type yang:counter64;
  description
    "This counter is identical with the in-octets/out-octets
    counters defined in RFC8343 except that it counts the
    octets since the analyzer was created.";
}
leaf idle-octets {
  if-feature "idle-octets-counter";
  type yang:counter64;
  description
    "Total accumulated period with no frame transmission
    taking place measured in octets at the current link
    speed. Octets not counted in ../octets but not idle are
    for example layer 1 framing octets - for Ethernet links
    7+1 preamble octets per packet.";
}
leaf errors {
  type yang:counter64;
  description
    "Count of packets with errors.
    Not counted in the pkts or captured.
    For example packets with CRC error.";
}
container testframe-stats {
  description
    "Statistics for testframe packets containing
    either sequence number, payload checksum,
    timestamp or any combination of these features.";
  leaf testframe-pkts {
    type yang:counter64;
    description
      "Total count of detected testframe packets.";
  }
  leaf sequence-errors {
    type yang:counter64;
    description
      "Total count of testframe packets with
      unexpected sequence number. After each sequence
      error the expected next sequence number is
      updated.";
  }
  leaf payload-errors {
    type yang:counter64;
    description
      "Total count of testframe packets with
      payload errors.";
```

```
    }
    container latency {
      description
        "Latency statistics.";
      leaf samples {
        type uint64;
        description
          "Total count of packets used for estimating
           the latency statistics. Ideally
           samples=../testframe-stats.";
      }
      leaf min {
        type uint64;
        units "nanoseconds";
        description
          "Minimum measured latency.";
      }
      leaf max {
        type uint64;
        units "nanoseconds";
        description
          "Maximum measured latency.";
      }
      leaf average {
        type uint64;
        units "nanoseconds";
        description
          "The sum of all sampled latencies divided
           by the number of samples.";
      }
      leaf latest {
        type uint64;
        units "nanoseconds";
        description
          "Latency of the latest sample.";
      }
    }
  }
}

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

    description
      "Contains capture parameters.";
```

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

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



```
    }
  }
}

grouping capture-data {
  description
    "Grouping with statistics and data
    of one or more captured frame.";
  container capture {
    if-feature "capture";
    description
      "Statistics and data of
      one or more captured frames.";
    list frame {
      key "sequence-number";
      description
        "Statistics and data of a captured frame.";
      leaf sequence-number {
        type uint64;
        description
          "Incremental counter of frames captured.";
      }
      leaf timestamp {
        type yang:date-and-time;
        description
          "Timestamp of the moment the frame was captured.";
      }
      leaf length {
        type uint32;
        description
          "Frame length. Ideally the data captured will be
          of the same length but can be shorter
          depending on implementation limitations.";
      }
      leaf preceding-interframe-gap {
        type uint32;
        units "nanoseconds";
        description
          "Measured delay between the reception of the previous
          frame was completed and the reception of the current
          frame was started.";
      }
      leaf data {
        type string {
          pattern '([0-9A-F]{2})*';
        }
        description
          "Raw data of the captured frame.";
      }
    }
  }
}
```

```
    }
  }
}

grouping filter-data {
  description
    "Grouping with a filter container specifying the filtering
    rules for processing only a specific subset of the
    frames.";
  container filter {
    if-feature "filter";
    presence "When present packets are
    filtered before analyzed according
    to the filter type";
    description
      "Contains the filtering rules for processing only
      a specific subset of the frames.";
    leaf type {
      type identityref {
        base ntt:filter;
      }
      mandatory true;
      description
        "Type of the applied filter. External modules can
        define alternative filter type identities.";
    }
  }
}

augment "/if:interfaces/if:interface" {
  description
    "Traffic analyzer augmentations of ietf-interfaces.";
  container traffic-analyzer {
    if-feature "ingress-direction";
    presence "Enables the traffic analyzer for ingress traffic.";
    description
      "Traffic analyzer for ingress direction.";
    uses filter-data;
    uses capture-config-data;
    container state {
      config false;
      description
        "State data.";
      uses statistics-data;
      uses capture-data;
    }
  }
}
```

```
    container traffic-analyzer-egress {
      if-feature "egress-direction";
      presence "Enables the traffic analyzer for egress traffic.";
      description
        "Traffic analyzer for egress direction.";
      uses filter-data;
      uses capture-config-data;
      container state {
        config false;
        description
          "State data.";
        uses statistics-data;
        uses capture-data;
      }
    }
  }
}

augment "/if:interfaces/if:interface/ntta:traffic-analyzer/"
  + "ntta:filter" {
  when "derived-from-or-self(ntta:type, 'ntta:ethernet')";
  description
    "Ethernet frame specific filter type.";
  leaf ether-type {
    type uint16;
    description
      "The Ethernet Type (or Length) value
        defined by IEEE 802.";
    reference
      "IEEE 802-2014 Clause 9.2";
  }
}
}
```

<CODE ENDS>

## 7. IANA Considerations

This document registers two URIs and two YANG modules.

### 7.1. URI Registration

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

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

Registrant Contact: The IESG.

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

## 7.2. YANG Module Name Registration

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

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

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

## 8. Security Considerations

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

There are a number of data nodes defined in this YANG module which are writable/creatable/deletable (i.e. config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g. edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

### 8.1. ietf-traffic-generator.yang

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

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

## 8.2. ietf-traffic-analyzer.yang

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

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

## 9. References

### 9.1. Normative References

- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<https://www.rfc-editor.org/info/rfc6536>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", RFC 7224, DOI 10.17487/RFC7224, May 2014, <<https://www.rfc-editor.org/info/rfc7224>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.

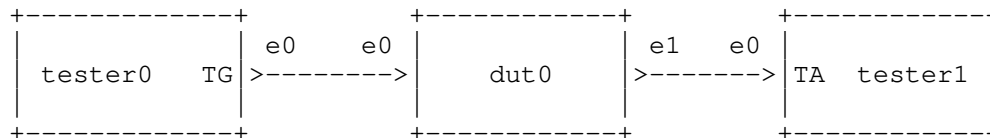
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.

## 9.2. Informative References

- [IEEE1588] IEEE, "IEEE 1588-2008", 2008.
- [IEEE802.3-2014] IEEE WG802.3 - Ethernet Working Group, "IEEE 802.3-2014", 2014.
- [RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <<https://www.rfc-editor.org/info/rfc2544>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.

## Appendix A. Examples

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



### A.1. Basic Test Program

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

```

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

# Configure DUTs and enable traffic-analyzers
net.node("dut0").edit( \
    "create /interfaces/interface[name='e0'] -- type=ethernetCsmacd")
net.node("dut0").edit(

```

```
"create /interfaces/interface[name='e1'] -- type=ethernetCsmacd")
net.node("dut0").edit(
  "create /flows/flow[id='t0'] -- match/in-port=e0 "
  "actions/action[order='0']/output-action/out-port=e1")

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

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

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

net.commit()

time.sleep(60)

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

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

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

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

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

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

## A.2. Generating RFC2544 Testframes

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

```
...
net.node("tester1").edit(
  "merge /interfaces/interface[name='e0']/"
  "traffic-generator -- frame-data="
  "6CA96F00000026CA96F000000108004500"
  "002ED4A5000000A115816C00000201C000"
  "0202C0200007001A00000010203040506"
  "0708090A0B0C0D0E0F101112")
...
```

## Author's Address

Vladimir Vassilev  
Lightside Instruments AS

Email: [vladimir@lightside-instruments.com](mailto:vladimir@lightside-instruments.com)