

Benchmarking Methodology Working Group  
Internet-Draft  
Intended status: Informational  
Expires: September 23, 2018

B. Balarajah  
C. Rossenhoevel  
EANTC AG  
March 22, 2018

Benchmarking Methodology for Network Security Device Performance  
draft-balarajah-bmwg-ngfw-performance-03

## Abstract

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

## Status of This Memo

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

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

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

This Internet-Draft will expire on September 23, 2018.

## Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Requirements . . . . .	3
3. Scope . . . . .	3
4. Test Setup . . . . .	4
4.1. Testbed Configuration . . . . .	4
4.2. DUT/SUT Configuration . . . . .	5
4.3. Test Equipment Configuration . . . . .	8
4.3.1. Client Configuration . . . . .	9
4.3.2. Backend Server Configuration . . . . .	10
4.3.3. Traffic Flow Definition . . . . .	11
4.3.4. Traffic Load Profile . . . . .	12
5. Test Bed Considerations . . . . .	13
6. Reporting . . . . .	14
6.1. Key Performance Indicators . . . . .	15
7. Benchmarking Tests . . . . .	16
7.1. Throughput Performance With NetSecOPEN Traffic Mix . . . . .	16
7.1.1. Objective . . . . .	16
7.1.2. Test Setup . . . . .	17
7.1.3. Test Parameters . . . . .	17
7.1.4. Test Procedures and expected Results . . . . .	19
7.2. Concurrent TCP Connection Capacity With HTTP Traffic . . . . .	20
7.2.1. Objective . . . . .	20
7.2.2. Test Setup . . . . .	20
7.2.3. Test Parameters . . . . .	20
7.2.4. Test Procedures and expected Results . . . . .	22
7.3. TCP/HTTP Connections Per Second . . . . .	23
7.3.1. Objective . . . . .	23
7.3.2. Test Setup . . . . .	24
7.3.3. Test Parameters . . . . .	24
7.3.4. Test Procedures and Expected Results . . . . .	25
7.4. HTTP Transactions Per Second . . . . .	26
7.4.1. Objective . . . . .	26
7.5. HTTP Throughput . . . . .	27
7.5.1. Objective . . . . .	27
7.6. HTTP Transaction Latency . . . . .	27
7.6.1. Objective . . . . .	27
7.7. Concurrent SSL/TLS Connection Capacity . . . . .	27
7.7.1. Objective . . . . .	27
7.8. SSL/TLS Handshake Rate . . . . .	27
7.8.1. Objective . . . . .	27
7.8.2. Test Setup . . . . .	28

7.8.3. Test Parameters . . . . .	28
7.8.4. Test Procedures and expected Results . . . . .	30
7.9. HTTPS Transaction Per Second . . . . .	31
7.9.1. Objective . . . . .	31
7.10. HTTPS Throughput . . . . .	31
7.10.1. Objective . . . . .	31
7.11. HTTPS Transaction Latency . . . . .	31
7.11.1. Objective . . . . .	31
8. Formal Syntax . . . . .	31
9. IANA Considerations . . . . .	31
10. Security Considerations . . . . .	32
11. Acknowledgements . . . . .	32
12. Normative References . . . . .	32
Appendix A. NetSecOPEN Basic Traffic Mix . . . . .	32
Authors' Addresses . . . . .	40

## 1. Introduction

15 years have passed since IETF recommended test methodology and terminology for firewalls initially (RFC 2647, RFC 3511). The requirements for network security element performance and effectiveness have increased tremendously since then. Security function implementations have evolved to more advanced areas and have diversified into intrusion detection and prevention, threat management, analysis of encrypted traffic, etc. In an industry of growing importance, well-defined and reproducible key performance indicators (KPIs) are increasingly needed: They enable fair and reasonable comparison of network security functions. All these reasons have led to the creation of a new next-generation firewall benchmarking document.

## 2. Requirements

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

## 3. Scope

This document provides testing terminology and testing methodology next-generation firewalls and related security functions. It covers two main areas: Performance benchmarks and security effectiveness testing. The 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 will be applicable to all of the benchmarking test cases described in Section 7.

##### 4.1. Testbed Configuration

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

In the typical deployment, the security devices (DUT/SUT) will not have a large number of entries in MAC or ARP tables, which impact the actual DUT/SUT performance due to MAC and ARP table lookup processes. Therefore, depend on number of used IP address in client and server side, it is recommended to connect Layer 3 device(s) between test equipment and DUT/SUT as shown in Figure 1.

If the test equipment is capable to emulate layer 3 routing functionality and there is no need for test equipment ports aggregation, it is recommended to configure the test setup as shown in Figure 2.

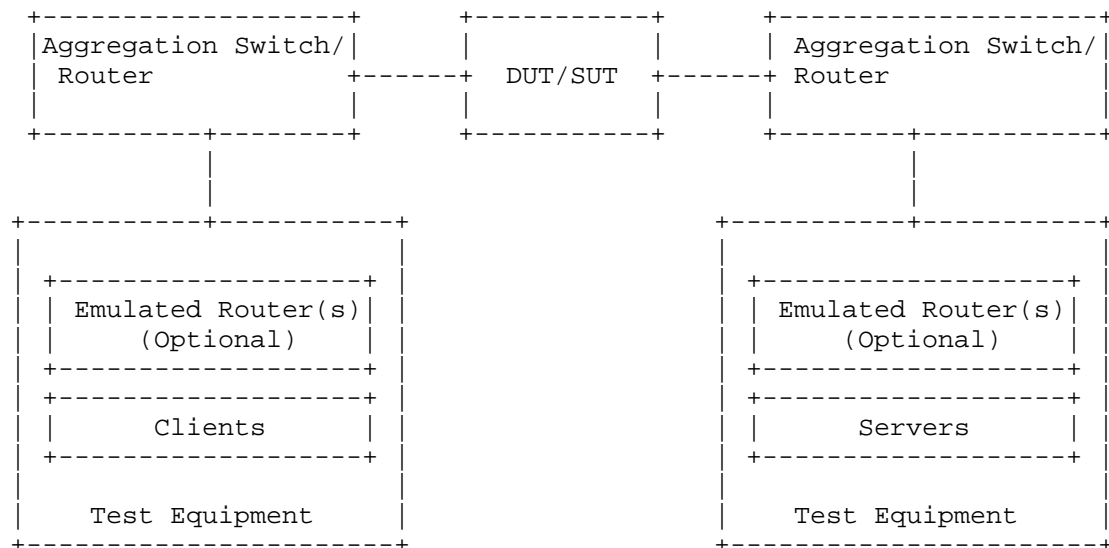


Figure 1: Testbed Setup - Option 1

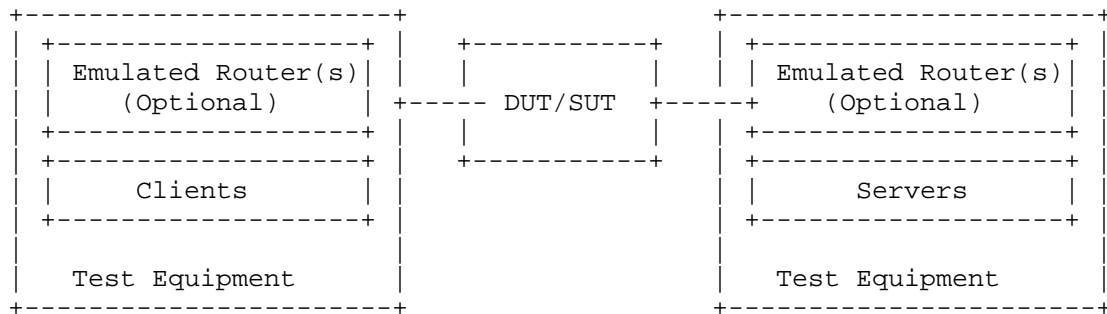


Figure 2: Testbed Setup - Option 2

#### 4.2. DUT/SUT Configuration

An unique DUT/SUT configuration MUST be used for all of the 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 that would be used in the actual deployment of the device or a typical deployment. Also it is mandatory to enable all the security features on the DUT/SUT in order to achieve maximum security coverage for a specific deployment scenario.

This document attempts to define the recommended security features which SHOULD be consistently enabled for all test cases. The table below describes the recommended sets of feature list which SHOULD be configured on the DUT/SUT. In order to improve repeatability, a summary of the DUT configuration including description of all enabled DUT/SUT features MUST be published with the benchmarking results.

Device									
NGFW					NGIPS	ADC	WAF	BPS	SSL Broker
DUT Features	Feature	Included in initial Scope	Added to future Scope	Future test standards to be developed					
SSL Inspection	x		x						
IDS/IPS	x	x							
Web Filtering	x		x						
Antivirus	x	x							
Anti Spyware	x	x							
Anti Botnet	x	x							
DLP	x		x						
DDoS	x		x						
Certificate Validation	x		x						
Logging and Reporting	x	x							
Application Identification	x	x							

Table 1: DUT/SUT Feature List

In addition, it is also recommended to configure a realistic number of access policy rules on the DUT/SUT. This document determines the number of access policy rules for three different class of DUT/SUT. The classification of the DUT/SUT MAY be based on its maximum supported throughput performance number. This document classifies the DUT/SUT in three different categories; namely small, medium and maximum.

The recommended throughput values for the following classes are;

Small - supported throughput less than 5Gbit/s

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

Large - supported throughput greater than 10Gbit/s

The access rule defined in the table 2 MUST be configured from top to bottom in correct order. The configured access policy rule MUST NOT block the test traffic used for the benchmarking test scenario.

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

Table 2: DUT/SUT Access List

#### 4.3. Test Equipment Configuration

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



This document attempts to explicitly specify which test equipment parameters SHOULD be configurable, any such parameter(s) MUST be noted in the test report.

#### 4.3.1. Client Configuration

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

##### 4.3.1.1. TCP Stack Attributes

The TCP stack SHOULD use a TCP Reno variant, which include congestion avoidance, back off and windowing, retransmission and recovery on every TCP connection between client and server endpoints. The default IPv4 and IPv6 MSS segments size MUST be set to 1460 bytes and 1440 bytes and a TX and RX receive windows of 32768 bytes. Delayed ACKs are permitted, but it SHOULD be limited to either a 200 msec delay timeout or 3000 in bytes before a forced ACK. Up to 3 retries SHOULD be allowed before a timeout event is declared. All traffic MUST set the TCP PSH flag to high. The source port range SHOULD be in the range of 1024 - 65535. Internal timeout SHOULD be dynamically scalable per RFC 793.

##### 4.3.1.2. Client IP Address Space

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

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

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

(Idea 1) 6-7 Mbps per IP= 1,400-1,700 IPs per 10Gbit/s throughput

(Idea 2) 0.1-0.2 Mbps per IP = 50,000-100,000 IPs per 10Gbit/s  
throughput

Based on deployment and usecase scenario, client IP addresses SHOULD be distributed between IPv4 and IPv6 type. This document recommends using the following ratio(s) between IPv4 and IPv6:

(Idea 1) 100 % IPv4, no IPv6

(Idea 2) 80 % IPv4, 20 % IPv6

(Idea 3) 50 % IPv4, 50 % IPv6

(Idea 4) 0 % IPv4, 100 % IPv6

#### 4.3.1.3. Emulated Web Browser Attributes

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

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

For encrypted traffic, the following attributes shall define the negotiated encryption parameters. The tests must use TLSv1.2 or higher with a record size of 16383, commonly used cipher suite and key strength. Session reuse or ticket resumption may be used for subsequent connections to the same Server endpoint IP. The client endpoint must send TLS Extension SNI information when opening up a security tunnel. Server certificate validation should be disabled. Server certificate validation should be disabled. Cipher suite and certificate size should be defined in the parameter session of benchmarking tests.

#### 4.3.2. Backend Server Configuration

This document attempts to specify which parameters should be considerable while configuring emulated backend servers using test equipment.

##### 4.3.2.1. TCP Stack Attributes

The TCP stack SHOULD use a TCP Reno variant, which include congestion avoidance, back off and windowing, retransmission and recovery on every TCP connection between client and server endpoints. The default IPv4 MSS segment size MUST be set to 1460 bytes and a TX and RX receive windows of at least 32768 bytes. Delayed ACKs are permitted but SHOULD be limited to either a 200 msec delay timeout or 3000 in bytes before a forced ACK. Up to 3 retries SHOULD be allowed before a timeout event is declared. All traffic MUST set the TCP PSH

flag to high. The source port range SHOULD be in the range of 1024 - 65535. Internal timeout should be dynamically scalable per RFC 793.

#### 4.3.2.2. Server Endpoint IP Addressing

The server IP blocks should consist of unique, continuous static address blocks with one IP per Server FQDN endpoint per test port. The IPv4 ToS byte should be set to '00'. The source mac address of the server endpoints shall be the same emulating routed behavior. Each Server FQDN should have it's own unique IP address. The Server IP addressing should be fixed to the same number of FQDN entries.

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

The emulated server pool for HTTP should listen on TCP port 80 and emulated HTTP version 1.1 with persistence. For HTTPS server, the pool must have the same basic attributes of an HTTP server pool plus attributes for SSL/TLS. The server must advertise a server type. For HTTPS server, TLS 1.2 or higher must be used with a record size of 16383 bytes and ticket resumption or Session ID reuse enabled. The server must listen on port TCP 443. The server shall serve a certificate to the client. It is required that the HTTPS server also check Host SNI information with the Fully Qualified Domain Name (FQDN). Client certificate validation should be disabled. Cipher suite and certificate size should be defined in the parameter session of benchmarking tests.

#### 4.3.3. Traffic Flow Definition

The section describes the traffic pattern between the client and server endpoints. At the beginning of the test, the server endpoint initializes and will be in a ready to accept connection state 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 the MAC and IP address. The behavior of the client is to sweep though the given server IP space, sequentially generating a recognizable service by the DUT. Thus, a balanced, mesh between client endpoints and server endpoints will be generated in a client port server port combination. Each client endpoint performs the same actions as other endpoints, with the difference being the source IP of the client endpoint and the target server IP pool. The client shall use Fully Qualified Domain Names in Host Headers and for TLS 1.2 Server Name Indication (SNI).

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

Client endpoints are independent of other clients that are concurrently executing. When a client endpoint initiate traffic, this section will describe how the steps through different services. Once initialized, the user should randomly hold (perform no operation) for a few milliseconds to allow for better randomization of start of client traffic. The client will then either open up a new TCP connection or connect to a TCP persistence stack still open to that specific server. At any point that the service profile may require encryption, a TLS 1.2 encryption tunnel will form presenting the URL request to the server. The server will then perform an SNI name check with the proposed FQDN compared to the domain embedded in the certificate. Only when correct, will the server process the object. The initial object to the server may not have a fixed size; its size is based on benchmarking tests described in Section 7. Multiple additional sub-URLs (Objects on the service page) may be requested simultaneously. This may or may not 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 will be described in this section. The loading of an traffic load profile has five distinct phases: Init, ramp up, sustain, ramp down/close, and collection.

Within 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 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.

In the ramp up phase, the test equipment should start to generate the test traffic. It should use a set approximate number of unique client IP addresses actively to generate traffic. The traffic should ramp from zero to desired target objective. The target objective will be defined for each benchmarking test. The duration for the ramp up phase must be configured long enough, so that the test equipment do not overwhelm DUT/SUT's supported performance metrics namely; connection setup rate, concurrent connection and application transaction. The recommended time duration for the ramp up phase is 180- 300 seconds. No measurements are made in this phase.

In the sustain phase, the test equipment should keep to generate traffic to constant target value for a constant number of active client IPs. The recommended time duration for sustain phase is 600 seconds. This is the phase where measurements occur.

In the ramp down/close phase, no new connection is established and no measurements are made. The recommended duration of this phase is between 180 to 300 seconds.

The last phase is administrative and will be when the tester merges and collates the report data.

## 5. Test Bed Considerations

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

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

Test bed reference pre-tests help to ensure that the desired traffic generator aspects such as maximum throughput and the network performance metrics such as maximum latency and maximum packet loss are met.

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

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

## 6. Reporting

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

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

### A. DUT Hardware/Virtual Configuration

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

### B. DUT Software

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

### C. DUT Enabled Features

- + Specific features, such as logging, NGFW, DPI must be documented
- + Attributes of those featured must be documented
- + Any additional relevant information about features must be documented

#### D. Test equipment hardware and software

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

#### 4. Results Summary / Executive Summary

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

#### 6.1. Key Performance Indicators

This section lists KPIs for overall benchmarking tests scenarios. All KPIs MUST be measured in whole period of sustain phase as described in Section 4.3.4. All KPIs MUST be measured from test equipment's result output.

- o TCP Concurrent Connection  
This key performance indicator will measure the average concurrent open TCP connections in the sustaining period.
- o TCP Connection Setup Rate  
This key performance indicator will measure the average established TCP connections per second in the sustaining period. For Session setup rate benchmarking test scenario, the KPI will measure average established and terminated TCP connections per second simultaneously.
- o Application Transaction Rate  
This key performance indicator will measure the average successful transactions per seconds in the sustaining period.

- o TLS Handshake Rate  
This key performance indicator will measure the average TLS 1.2 or higher session formation rate within the sustaining period.
- o Throughput  
This key performance indicator will measure the average Layer 1 throughput within the sustaining period as well as average packets per seconds within the same period. The value of throughput should be presented in Gbps rounded to two places of precision with a more specific kbps in parenthesis. Optionally, goodput may also be logged as an average goodput rate measured over the same period. Goodput result shall also be presented in the same format as throughput.
- o URL Response time / Time to Last Byte (TTLB)  
This key performance indicator will measure the minimum, average and maximum per URL response time in the sustaining period as well as the average variance in the same period.
- o Application Transaction Time  
This key performance indicator will measure the minimum, average and maximum the amount of time to receive all objects from the server.
- o Time to First Byte (TTFB)  
This key performance indicator will measure minimum, average and maximum the time to first byte. TTFB is the elapsed time between sending the SYN packet from the client and receiving the first byte of application data from the DUT/SUT. TTFB SHOULD be expressed in millisecond.
- o TCP Connect Time  
This key performance indicator will measure minimum, average and maximum TCP connect time. It is elapsed between the time the client sends a SYN packet and the time it receives the SYN/ACK. TCP connect time SHOULD be expressed in millisecond.

## 7. Benchmarking Tests

### 7.1. Throughput Performance With NetSecOPEN Traffic Mix

#### 7.1.1. Objective

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



#### 7.1.2. Test Setup

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

#### 7.1.3. Test Parameters

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

##### 7.1.3.1. DUT/SUT Configuration Parameters

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

##### 7.1.3.2. Test Equipment Configuration Parameters

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

Client IP address range

Server IP address range

Traffic distribution ratio between IPv4 and IPv6

Traffic load objective or specification type (e.g. Throughput, SimUsers and etc.)

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

Initial throughput: Initial throughput MAY be up to 10% of the "Target throughput"

##### 7.1.3.3. Traffic Profile

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

##### 7.1.3.4. Test Results Acceptance Criteria

The following test Criteria is defined as test results acceptance criteria

- a. Number of failed Application transaction MUST be 0.01%.
- b. Number of Terminated TCP connection due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.01%
- c. Maximum deviation (max. dev) of application transaction time / TTLB (Time To Last Byte) MUST be less than X (The value for "X" will be finalized and updated in future draft release)  
The following equation MUST be used to calculate the deviation of application transaction time or TTLB.

$$\text{max. dev} = \max((\text{avg\_latency} - \text{min\_latency}), (\text{max\_latency} - \text{avg\_latency})) / (\text{Initial latency})$$

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

The variable latency represents application transaction time or TTLB.

$$\text{Initial latency} := \min((\text{avg}' \text{ latency} - \text{min}' \text{ latency}) \mid (\text{max}' \text{ latency} - \text{avg}' \text{ latency}))$$

- d. Maximum value of TCP connect time must be less than Xms (The value for "X" will be finalized and updated in future draft release). The definition for TCP connect time is found in Section 6.1.
- e. Maximum value of Time to First Byte must be less than 2\* TCP connect time.

Test Acceptance criteria for this test scenario MUST be monitored during the sustain phase of the traffic load profile only.

#### 7.1.3.5. Measurement

Following KPI metrics MUST be reported for this test scenario.

Mandatory KPIs: average Throughput, maximum Concurrent TCP connection, TTLB/application transaction time (minimum, average and maximum) and average application transaction rate

Optional KPIs: average TCP connection setup rate, average TLS handshake rate, TCP connect time and TTFB

#### 7.1.4. Test Procedures and expected Results

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

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

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

Configure traffic load profile of the test equipment to generate test traffic at "initial throughput" rate as described in the parameters section. 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 acceptance criteria "a" and "b" defined in Section 7.1.3.4.

If the KPI metrics do not meet the acceptance 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 "Target throughput" rate defined in the parameter table. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4. The test equipment SHOULD start to measure and record all specified KPIs. The frequency of KPI metrics measurement MUST be less than 5 seconds. Continue the test until all traffic profile phases are completed.

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

##### 7.1.4.3. Step 3: Test Iteration with Binary Search

Use binary search algorithm to configure the desired traffic load profile for each test iteration. Binary search algorithm can be implemented using the parameter; Resolution =  $0.01 \times \text{Target throughput}$  and Backoff = 50%.

Determine the maximum and average achievable throughput within the acceptance criteria.

## 7.2. Concurrent TCP Connection Capacity With HTTP Traffic

### 7.2.1. Objective

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

### 7.2.2. Test Setup

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

### 7.2.3. Test Parameters

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

#### 7.2.3.1. DUT/SUT Configuration Parameters

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

#### 7.2.3.2. Test Equipment Configuration Parameters

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

Client IP address range

Server IP address range

Traffic distribution ratio between IPv4 and IPv6

Traffic load objective or specification type (e.g Throughput, SimUsers and etc.)

Target concurrent connection: It can be defined based on requirements

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

#### 7.2.3.2.1. Client Configuration Parameters

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

Test scenario SHOULD be run with a single traffic profile with following attributes:

HTTP 1.1 with GET command requesting 10 Kbyte objects with random MIME type.

The test equipment SHOULD perform HTTP transactions within each TCP connection subsequently. The frequency of transactions MUST be defined to achieve X% of total throughput that DUT can support. The suggested value of X is 25. It will be finalized and updated in the next draft version.

During the sustain state of concurrent connection and traffic load, a minimal % of TCP connection SHOULD be closed and re-opened.

#### 7.2.3.3. Test Results Acceptance Criteria

The following test Criteria is defined as test results acceptance criteria

- a. Number of failed Application transaction MUST be less than 0.01% of attempt transaction.
- b. Number of Terminated TCP connection due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.01% of total initiated TCP sessions.
- c. During the sustain phase, traffic should be forwarded constantly at the rate defined in the parameter Section 7.2.3.
- d. Maximum deviation (max. dev) of application transaction time / TTLB (Time To Last Byte) MUST be less than Xms (The value for "X" will be finalized and updated in future draft release). The following equation MUST be used to calculate the deviation of application transaction time or TTLB.

$$\text{max. dev} = \max((\text{avg\_latency} - \text{min\_latency}), (\text{max\_latency} - \text{avg\_latency})) / (\text{Initial latency})$$

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

The variable latency represents application transaction time or TTLB.

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

- e. Maximum value of TCP connect time must be less than Xms (The value for "X" will be finalized and updated in future draft release). The definition for TCP connect time is found in Section 6.1.
- f. Maximum value of Time to First Byte must be less than 2\* TCP connect time.

Test Acceptance criteria for this test scenario MUST be monitored during the sustain phase of the traffic load profile only.

#### 7.2.3.4. Measurement

Following KPI metrics MUST be reported for this test scenario;

average Throughput, max. Min. Avg. Concurrent TCP connection, TTLB/ application transaction time (minimum, average and maximum) and average application transaction rate.

#### 7.2.4. 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.

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

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

Configure traffic load profile of the test equipment to establish "initial concurrent connection" as defined in the parameters section. The traffic load profile should be defined as described in Section 4.3.4.

The DUT/SUT SHOULD reach the "initial concurrent connection" during the sustain phase. The measured KPIs during the sustain phase MUST meet the acceptance criteria "a" and "b" defined in Section 7.2.3.3

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

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

Configure test equipment to establish "Target concurrent connection" 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, the other KPIs such as throughput, TCP connection rate and application transaction MUST NOT reach to the maximum value that the DUT/SUT can support. Throughput, TCP connection rate and application transaction should not be reached more than X% of maximum value that DUT can support. The suggested value of X is 25. It will be finalized and updated in the next draft version.

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement MUST be less than 5 seconds. Continue the test until all traffic profile phases are completed.

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

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

#### 7.2.4.3. Step 3: Test Iteration with Binary Search

Use binary search algorithm to configure the desired traffic load profile for each test iteration. Binary search algorithm can be implemented using the parameter; Resolution =  $0.01 * \text{"Target concurrent connection"}$  and Backoff = 50%.

Determine the maximum and average achievable throughput within the acceptance criteria.

### 7.3. TCP/HTTP Connections Per Second

#### 7.3.1. Objective

Using HTTP traffic, determine the maximum and average value of TCP session establishment rate supported by the DUT/SUT.

Test parameters and test procedures will be added in the future release.

### 7.3.2. Test Setup

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

### 7.3.3. Test Parameters

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

#### 7.3.3.1. DUT/SUT Configuration Parameters

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

#### 7.3.3.2. Test Equipment Configuration Parameters

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

- Client IP address range
- Server IP address range
- Traffic distribution ratio between IPv4 and IPv6
- Target connections per second
- Initial connections per second: 10% of "Target connections per second"

##### 7.3.3.2.1. Client Configuration Parameters'

Test scenario SHOULD be run with a single traffic profile with following attributes:

The client MUST negotiate HTTP 1.1 and close the connection immediately after completion of the transaction.

HTTP 1.1 with GET command requesting a single 1, 16 or 64 Kbyte object.



#### 7.3.3.3. Test Results Acceptance Criteria

The following test Criteria is defined as test results acceptance criteria.

- a. Number of failed Application transaction MUST be less than 0.01% of attempt transaction.
- b. Number of Terminated TCP connection due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.01% of total initiated TCP sessions
- c. During the sustain phase, traffic should be forwarded at a constant rate
- d. During the sustain phase, Average Time to TCP First Byte MUST be constant and not increase more than 10%.
- e. Concurrent TCP connection should be constant during steady state. This confirms the DUT opens and closes the session at the same rate.

#### 7.3.3.4. Measurement

Following KPI metrics MUST be reported for this test scenario.

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

#### 7.3.4. Test Procedures and Expected Results

The test procedure is designed to measure the TCP connection 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.

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

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

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

The DUT/SUT SHOULD reach the "initial connections per second" before the sustain phase. The measured KPIs during the sustain phase MUST meet the acceptance criteria a, b, c and d defined in section 7.3.3.3.

If the KPI metrics do not meet the acceptance 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 "Target connections per second" defined in the parameters table. The test equipment SHOULD follow the traffic load profile definition as described in the section 4.3.4.

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

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement MUST be less than 5 seconds. Continue the test until all traffic profile phases are completed.

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

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

#### 7.3.4.3. Step 3: Test Iteration with Binary Search

Use binary search algorithm to configure the desired traffic load profile for each test iteration. Binary search algorithm can be implemented using the parameter; Resolution =  $0.01 \times$  "Target connections per second" and Backoff = 50%.

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

### 7.4. HTTP Transactions Per Second

#### 7.4.1. Objective

Determine maximum and average HTTP transaction rate supported by the DUT/SUT.

Test parameters and test test procedures will be added in the future release.

## 7.5. HTTP Throughput

### 7.5.1. Objective

Determine the average throughput performance of the DUT/SUT when using HTTP traffic.

Test parameters and test test procedures will be added in the future release.

## 7.6. HTTP Transaction Latency

### 7.6.1. Objective

Determine the minimum, average and maximum values of HTTP transaction latency at 80% throughput rate measured in "HTTP Throughput" test scenario.

Test parameters and test test procedures will be added in the future release.

## 7.7. Concurrent SSL/TLS Connection Capacity

### 7.7.1. Objective

Usin encrypted traffic (HTTPS), determine the maximum number of concurrent TCP connection that DUT/SUT sustains.

Test parameters and test test procedures will be added in the future release.

## 7.8. SSL/TLS Handshake Rate

### 7.8.1. Objective

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

Test iterations MUST include common cipher suites and key strengths as well as forward looking stronger keys. Specific test iterations MUST include the following ciphers and keys:

1. ECHDE-ECDSA-AES128-GCM-SHA256 with Prime256v1

2. ECDHE-RSA-AES128-GCM-SHA256 with RSA 2048
3. ECDHE-ECDSA-AES256-GCM-SHA384 with Secp384
4. ECDHE-RSA-AES256-GCM-SHA384 with RSA 3072

For each cipher suite and key strengths, test iterations MUST use a single HTTP transaction object size of 1KB, 16KB and 64KB to measure connections per second performance under a variety of DUT Security inspection load conditions.

#### 7.8.2. Test Setup

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

#### 7.8.3. Test Parameters

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

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

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

##### 7.8.3.2. Test Equipment Configuration Parameters

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

- Client IP address range
- Server IP address range
- Traffic distribution ratio between IPv4 and IPv6
- Target connections per second
- Initial connections per second: 10% of "Target connections per second"

#### 7.8.3.2.1. Client Configuration Parameters'

Test scenario SHOULD be run with a single traffic profile with following attributes:

The client MUST negotiate HTTPS 1.1 and close the connection immediately after completion of the transaction.

HTTPS 1.1 with GET command requesting a single 1, 16 and 64 Kbyte object.

Each client connection MUST perform a full handshake with server certificate and MUST NOT use session reuse or resumption.

TLS record size MAY be optimized for the object size up to a record size of 16K.

#### 7.8.3.3. Test Results Acceptance Criteria

The following test Criteria is defined as test results acceptance criteria.

- a. Number of failed Application transaction MUST be less than 0.01% of attempt transaction.
- b. Number of Terminated TCP connection due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.01% of total initiated TCP sessions.
- c. During the sustain phase, traffic should be forwarded at a constant rate.
- d. During the sustain phase, Average Time to TCP First Byte MUST be constant and not increase more than 10%.
- e. Concurrent TCP connection should be constant during steady state. This confirms the DUT opens and closes the session at the same rate.

#### 7.8.3.4. Measurement

Following KPI metrics MUST be reported for this test scenario.

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

#### 7.8.4. Test Procedures and expected Results

The test procedure is designed to measure the TCP connection 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.

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

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

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

The DUT/SUT SHOULD reach the "initial connections per second" before the sustain phase. The measured KPIs during the sustain phase MUST meet the acceptance criteria a, b, c and d defined in section 7.8.3.3.

If the KPI metrics do not meet the acceptance 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 connections per second" defined in the parameters table. The test equipment SHOULD follow the traffic load profile definition as described in the section 4.3.4.

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

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement MUST be less than 5 seconds. Continue the test until all traffic profile phases are completed.

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

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

#### 7.8.4.3. Step 3: Test Iteration with Binary Search

Use binary search algorithm to configure the desired traffic load profile for each test iteration. Binary search algorithm can be implemented using the parameter; Resolution = 0.01 \* " Target connections per second " and Backoff = 50%.

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

### 7.9. HTTPS Transaction Per Second

#### 7.9.1. Objective

Determine maximum and average HTTPS transaction rate supported by the DUT/SUT.

Test parameters and test procedures will be added in the future release.

### 7.10. HTTPS Throughput

#### 7.10.1. Objective

Determine the average throughput performance of the DUT/SUT when using HTTPS traffic.

Test parameters and test procedures will be added in the future release.

### 7.11. HTTPS Transaction Latency

#### 7.11.1. Objective

Determine the minimum, average and maximum values of HTTPS transaction latency at 80% throughput rate measured in "HTTPS Throughput" test scenario.

Test parameters and test procedures will be added in the future release.

## 8. Formal Syntax

## 9. IANA Considerations

This document makes no request of IANA.

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

#### 10. Security Considerations

Security consideration will be added in the future release.

#### 11. Acknowledgements

Acknowledgements will be added in the future release.

#### 12. 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>>.

#### Appendix A. NetSecOPEN Basic Traffic Mix

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

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

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

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



The mix MUST be of sufficient complexity and volume to render differences in individual apps as statistically insignificant. For example, changes in like to like apps - such as one type of video service vs. another both consist of larger objects whereas one news site vs. another both typically have more connections than other apps because of trackers and embedded advertising content. To achieve sufficient complexity, a mix MUST have:

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

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

Table 3 lists the FQDNs, number of transactions and bytes transferred as an example client interacts with Office 365 Outlook, Word, Excel, Powerpoint, Sharepoint and Skype.

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

sl-powerpoint-15.cdn.office.net	557,604	3	
appsforoffice.microsoft.com	511,171	5	
powerpoint.officeapps.live.com	471,625	14	
excel.officeapps.live.com	342,040	14	
sl-officeapps-15.cdn.office.net	331,343	5	
webdir0a.online.lync.com	66,930	15	
portal.office.com	13,956	1	
config.edge.skype.com	6,911	2	
clientlog.portal.office.com	6,608	8	
webdir.online.lync.com	4,343	5	
graph.microsoft.com	2,289	2	
nam.loki.delve.office.com	1,812	5	
login.microsoftonline.com	464	2	
login.windows.net	232	1	

Table 3: Office365

Clients MUST connect to multiple server FQDNs in the same order as real applications. Connections MUST be made when the client is interacting with the application and NOT first setup up all connections. Connections SHOULD stay open per client for subsequent transactions to the same FQDN similar to how a web browser behaves. Clients MUST use different URL Paths and Object sizes in orders as they are observed in real Applications. Clients MAY also setup multiple connections per FQDN to process multiple transactions in a sequence at the same time. Table 4 has a partial example sequence of the Office 365 Word application transactions.

FQDN	URL Path	Object size
company1-my.sharepoint.com	/personal...	23,132

word-edit.officeapps.live.com	/we/WsaUpload.ashx	2
static.sharepointonline.com	/bld/.../blank.js	454
static.sharepointonline.com	/bld/.../ initstrings.js	23,254
static.sharepointonline.com	/bld/.../init.js	292,740
company1-my.sharepoint.com	/ScriptResource...	102,774
company1-my.sharepoint.com	/ScriptResource...	40,329
company1-my.sharepoint.com	/WebResource...	23,063
word-edit.officeapps.live.com	/we/wordeditorframe. aspx...	60,657
static.sharepointonline.com	/bld/_layouts/.../ blank.js	454
s1-word-edit-15.cdn.office.net	/we/s/.../ EditSurface.css	19,201
s1-word-edit-15.cdn.office.net	/we/s/.../ WordEditor.css	221,397
s1-officeapps-15.cdn.office.net	/we/s/.../ Microsoft Ajax.js	107,571
s1-word-edit-15.cdn.office.net	/we/s/.../ wacbootwe.js	39,981
s1-officeapps-15.cdn.office.net	/we/s/.../ CommonIntl.js	51,749
s1-word-edit-15.cdn.office.net	/we/s/.../ Compat.js	6,050
s1-word-edit-15.cdn.office.net	/we/s/.../ Box4Intl.js	54,158
s1-word-edit-15.cdn.office.net	/we/s/.../ WoncaIntl.js	24,946
s1-word-edit-15.cdn.office.net	/we/s/.../	53,515

	WordEditorIntl.js	
sl-word-edit-15.cdn.office.net	/we/s/.../ WordEditorExp.js	1,978,712
sl-word-edit-15.cdn.office.net	/we/s/.../jSanity.js	10,912
word-edit.officeapps.live.com	/we/OneNote.ashx	145,708

Table 4: Office365 Word Transactions

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

Table 5 includes further examples in SNI and CN pairs for several FQDNs of Office 365.

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

Table 5: Office365 SNI and CN Pairs Examples

NetSecOPEN has provided a reference enterprise perimeter traffic mix with dozens of applications, hundreds of connections, and thousands of transactions. (link to spreadsheet with details)

NetSecOPEN has provided a reference enterprise perimeter traffic mix with dozens of applications, hundreds of connections, and thousands of transactions. (link to spreadsheet with details)

The enterprise perimeter traffic mix consists of 70% HTTPS and 30% HTTP by Bytes, 58% HTTPS and 42% HTTP by Transactions. By connections with a single connection per FQDN the mix consists of 43% HTTPS and 57% HTTP. With multiple connections per FQDN the HTTPS percentage is higher.

Table 6 is a summary of the NetSecOPEN enterprise perimeter traffic mix sorted by bytes with unique FQDNs and transactions per applications.

Application	FQDNs	Transactions	Bytes
Office365	26	558	52,931,947
Box	4	90	23,276,089
Salesforce	6	365	23,137,548
Gmail	13	139	16,399,289
Linkedin	10	206	15,040,918
DailyMotion	8	77	14,751,514
GoogleDocs	2	71	14,205,476
Wikia	15	159	13,909,777
Foxnews	82	499	13,758,899
Yahoo Finance	33	254	13,134,011
Youtube	8	97	13,056,216
Facebook	4	207	12,726,231
CNBC	77	275	11,939,566
Lightreading	27	304	11,200,864

BusinessInsider	16	142	11,001,575
Alexa	5	153	10,475,151
CNN	41	206	10,423,740
Twitter Video	2	72	10,112,820
Cisco Webex	1	213	9,988,417
Slack	3	40	9,938,686
Google Maps	5	191	8,771,873
SpectrumIEEE	7	145	8,682,629
Yelp	9	146	8,607,645
Vimeo	12	74	8,555,960
Wikihow	11	140	8,042,314
Netflix	3	31	7,839,256
Instagram	3	114	7,230,883
Morningstar	30	150	7,220,121
Docusign	5	68	6,972,738
Twitter	1	100	6,939,150
Tumblr	11	70	6,877,200
Whatsapp	3	46	6,829,848
Imdb	16	251	6,505,227
NOAAgov	1	44	6,316,283
IndustryWeek	23	192	6,242,403
Spotify	18	119	6,231,013
AutoNews	16	165	6,115,354
Evernote	3	47	6,063,168

NatGeo	34	104	6,026,344
BBC News	18	156	5,898,572
Investopedia	38	241	5,792,038
Pinterest	8	102	5,658,994
Succesfactors	2	112	5,049,001
AbaJournal	6	93	4,985,626
Pbworks	4	78	4,670,980
NetworkWorld	42	153	4,651,354
WebMD	24	280	4,416,736
OilGasJournal	14	105	4,095,255
Trello	5	39	4,080,182
BusinessWire	5	109	4,055,331
Dropbox	5	17	4,023,469
Nejm	20	190	4,003,657
OilGasDaily	7	199	3,970,498
Chase	6	52	3,719,232
MedicalNews	6	117	3,634,187
Marketwatch	25	142	3,291,226
Imgur	5	48	3,189,919
NPR	9	83	3,184,303
Onelogin	2	31	3,132,707
Concur	2	50	3,066,326
Service-now	1	37	2,985,329
Apple itunes	14	80	2,843,744

BerkeleyEdu	3	69	2,622,009
MSN	39	203	2,532,972
Indeed	3	47	2,325,197
MayoClinic	6	56	2,269,085
Ebay	9	164	2,219,223
UCLAedu	3	42	1,991,311
ConstructionDive	5	125	1,828,428
EducationNews	4	78	1,605,427
BofA	12	68	1,584,851
ScienceDirect	7	26	1,463,951
Reddit	8	55	1,441,909
FoodBusinessNews	5	49	1,378,298
Amex	8	42	1,270,696
Weather	4	50	1,243,826
Wikipedia	3	27	958,935
Bing	1	52	697,514
ADP	1	30	508,654
Grand Total	983	10021	569,819,095

Table 6: Summary of NetSecOPEN Enterprise Perimeter Traffic Mix

Authors' Addresses



Balamuhunthan Balarajah  
EANTC AG  
Salzufer 14  
Berlin 10587  
Germany

Email: balarajah@eantc.de

Carsten Rossenhoevel  
EANTC AG  
Salzufer 14  
Berlin 10587  
Germany

Email: cross@eantc.de

Benchmarking Methodology Working Group  
Internet-Draft  
Intended status: Informational  
Expires: April 17, 2019

B. Balarajah  
C. Rossenhoevel  
EANTC AG  
October 14, 2018

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

## Abstract

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

## Status of This Memo

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

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

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

This Internet-Draft will expire on April 17, 2019.

## Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Requirements . . . . .	4
3. Scope . . . . .	4
4. Test Setup . . . . .	4
4.1. Testbed Configuration . . . . .	4
4.2. DUT/SUT Configuration . . . . .	5
4.3. Test Equipment Configuration . . . . .	8
4.3.1. Client Configuration . . . . .	9
4.3.2. Backend Server Configuration . . . . .	10
4.3.3. Traffic Flow Definition . . . . .	11
4.3.4. Traffic Load Profile . . . . .	12
5. Test Bed Considerations . . . . .	13
6. Reporting . . . . .	14
6.1. Key Performance Indicators . . . . .	15
7. Benchmarking Tests . . . . .	16
7.1. Throughput Performance With NetSecOPEN Traffic Mix . . . . .	17
7.1.1. Objective . . . . .	17
7.1.2. Test Setup . . . . .	17
7.1.3. Test Parameters . . . . .	17
7.1.4. Test Procedures and expected Results . . . . .	19
7.2. TCP/HTTP Connections Per Second . . . . .	20
7.2.1. Objective . . . . .	20
7.2.2. Test Setup . . . . .	20
7.2.3. Test Parameters . . . . .	20
7.2.4. Test Procedures and Expected Results . . . . .	21
7.3. HTTP Transaction per Second . . . . .	23
7.3.1. Objective . . . . .	23
7.3.2. Test Setup . . . . .	23
7.3.3. Test Parameters . . . . .	23
7.3.4. Test Procedures and Expected Results . . . . .	24
7.4. TCP/HTTP Transaction Latency . . . . .	26
7.4.1. Objective . . . . .	26
7.4.2. Test Setup . . . . .	26
7.4.3. Test Parameters . . . . .	26
7.4.4. Test Procedures and Expected Results . . . . .	28
7.5. HTTP Throughput . . . . .	29
7.5.1. Objective . . . . .	29
7.5.2. Test Setup . . . . .	29
7.5.3. Test Parameters . . . . .	30
7.5.4. Test Procedures and Expected Results . . . . .	32
7.6. Concurrent TCP/HTTP Connection Capacity . . . . .	33

7.6.1.	Objective . . . . .	33
7.6.2.	Test Setup . . . . .	33
7.6.3.	Test Parameters . . . . .	33
7.6.4.	Test Procedures and expected Results . . . . .	34
7.7.	TCP/HTTPS Connections per second . . . . .	36
7.7.1.	Objective . . . . .	36
7.7.2.	Test Setup . . . . .	36
7.7.3.	Test Parameters . . . . .	36
7.7.4.	Test Procedures and expected Results . . . . .	38
7.8.	HTTPS Transaction per Second . . . . .	39
7.8.1.	Objective . . . . .	39
7.8.2.	Test Setup . . . . .	40
7.8.3.	Test Parameters . . . . .	40
7.8.4.	Test Procedures and Expected Results . . . . .	42
7.9.	HTTPS Transaction Latency . . . . .	43
7.9.1.	Objective . . . . .	43
7.9.2.	Test Setup . . . . .	43
7.9.3.	Test Parameters . . . . .	43
7.9.4.	Test Procedures and Expected Results . . . . .	45
7.10.	HTTPS Throughput . . . . .	46
7.10.1.	Objective . . . . .	46
7.10.2.	Test Setup . . . . .	47
7.10.3.	Test Parameters . . . . .	47
7.10.4.	Test Procedures and Expected Results . . . . .	49
7.11.	Concurrent TCP/HTTPS Connection Capacity . . . . .	50
7.11.1.	Objective . . . . .	50
7.11.2.	Test Setup . . . . .	50
7.11.3.	Test Parameters . . . . .	50
7.11.4.	Test Procedures and expected Results . . . . .	52
8.	Formal Syntax . . . . .	53
9.	IANA Considerations . . . . .	53
10.	Acknowledgements . . . . .	54
11.	Contributors . . . . .	54
12.	References . . . . .	54
12.1.	Normative References . . . . .	54
12.2.	Informative References . . . . .	54
Appendix A.	NetSecOPEN Basic Traffic Mix . . . . .	55
Authors' Addresses	. . . . .	63

## 1. Introduction

15 years have passed since IETF recommended test methodology and terminology for firewalls initially ([RFC2647], [RFC3511]). The requirements for network security element performance and effectiveness have increased tremendously since then. Security function implementations have evolved to more advanced areas and have diversified into intrusion detection and prevention, threat management, analysis of encrypted traffic, etc. In an industry of

growing importance, well-defined and reproducible key performance indicators (KPIs) are increasingly needed: They enable fair and reasonable comparison of network security functions. All these reasons have led to the creation of a new next-generation firewall benchmarking document.

## 2. Requirements

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

## 3. Scope

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

## 4. Test Setup

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

### 4.1. Testbed Configuration

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

However, in the typical deployment, the security devices (DUT/SUT) are connected to routers and switches which will reduce the number of entries in MAC or ARP tables of the DUT/SUT. If MAC or ARP tables have many entries, this may impact the actual DUT/SUT performance due to MAC and ARP/ND table lookup processes. Therefore, it is RECOMMENDED to connect Layer 3 device(s) between test equipment and DUT/SUT as shown in Figure 1.

If the test equipment is capable to emulate layer 3 routing functionality and there is no need for test equipment ports aggregation, it is RECOMMENDED to configure the test setup as shown in Figure 2.

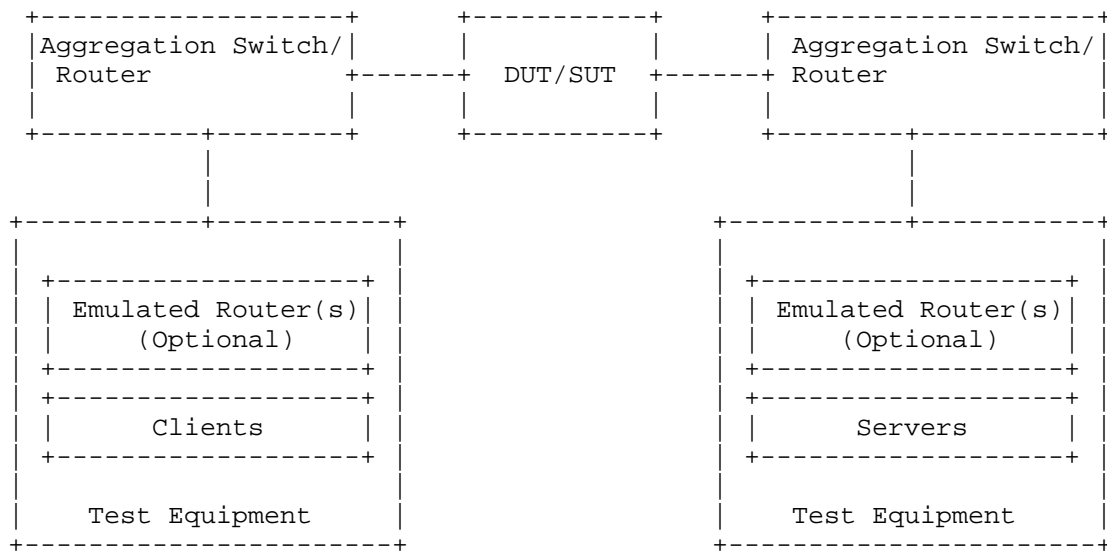


Figure 1: Testbed Setup - Option 1

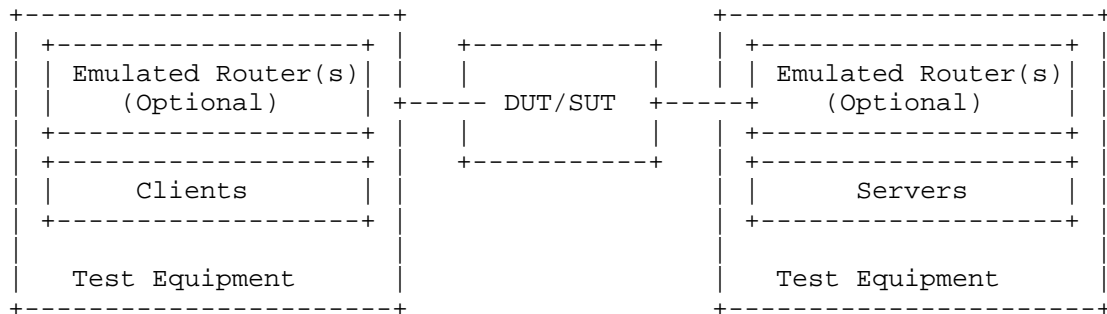


Figure 2: Testbed Setup - Option 2

#### 4.2. DUT/SUT Configuration

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

This document attempts to define the recommended security features which SHOULD be consistently enabled for all the benchmarking tests

described in Section 7. The table 1 below describes the RECOMMENDED sets of feature list which SHOULD be configured on the DUT/SUT.

Based on customer use case, user can take a decision to enable or disable SSL inspection feature for "Throughput Performance with NetSecOPEN Traffic Mix" test scenario described in Section 7.1

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

+-----+   NGFW   +-----+			
DUT Features	Feature	Included in initial Scope	Added to future Scope
SSL Inspection	x	x	
IDS/IPS	x	x	
Web Filtering	x		x
Antivirus	x	x	
Anti Spyware	x	x	
Anti Botnet	x	x	
DLP	x		x
DDoS	x		x
Certificate Validation	x		x
Logging and Reporting	x	x	
Application Identification	x	x	

Table 1: DUT/SUT Feature List

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

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

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

The RECOMMENDED throughput values for the following classes are:

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

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

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

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

The access rule defined in the table 2 MUST be configured from top to bottom in correct order shown in the table. The configured access policy rule MUST NOT block the test traffic used for the benchmarking test scenarios.



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

Table 2: DUT/SUT Access List

#### 4.3. Test Equipment Configuration

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

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

#### 4.3.1. Client Configuration

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

##### 4.3.1.1. TCP Stack Attributes

The TCP stack SHOULD use a TCP Reno variant, which include congestion avoidance, back off and windowing, retransmission, and recovery on every TCP connection between client and server endpoints. The default IPv4 and IPv6 MSS segments size MUST be set to 1460 bytes and 1440 bytes respectively and a TX and RX receive windows of 32768 bytes. Client initial congestion window MUST NOT exceed 10 times the MSS. Delayed ACKs are permitted and the maximum client delayed Ack MUST NOT exceed 10 times the MSS before a forced ACK. Up to 3 retries SHOULD be allowed before a timeout event is declared. All traffic MUST set the TCP PSH flag to high. The source port range SHOULD be in the range of 1024 - 65535. Internal timeout SHOULD be dynamically scalable per RFC 793. Client SHOULD initiate and close TCP connections. TCP connections MUST be closed via FIN.

##### 4.3.1.2. Client IP Address Space

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

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

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

(Idea 1) 6-7 Mbps per IP (e.g. 1,400-1,700 IPs per 10Gbit/s throughput)

(Idea 2) 0.1-0.2 Mbps 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. This document recommends using the following ratio(s) between IPv4 and IPv6:

(Idea 1) 100 % IPv4, no IPv6

(Idea 2) 80 % IPv4, 20 % IPv6

(Idea 3) 50 % IPv4, 50 % IPv6

(Idea 4) 0 % IPv4, 100 % IPv6

#### 4.3.1.3. Emulated Web Browser Attributes

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

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

For encrypted traffic, the following attributes shall define the negotiated encryption parameters. The tests MUST use TLSv1.2 or higher with a record size of 16383, commonly used cipher suite and key strength. Depending on test scenario, Session reuse or ticket resumption MAY be used for subsequent connections to the same Server endpoint IP. The client endpoint MUST send TLS Extension Server Name Indication (SNI) information when opening a security tunnel. Cipher suite and certificate size should be defined in the parameter session of each test scenario.

#### 4.3.2. Backend Server Configuration

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

##### 4.3.2.1. TCP Stack Attributes

The TCP stack SHOULD use a TCP Reno variant, which include congestion avoidance, back off and windowing, retransmission, and recovery on every TCP connection between client and server endpoints. The default IPv4 and IPv6 MSS segment size MUST be set to 1460 bytes and

1440 bytes respectively and a TX and RX receive windows of at least 32768 bytes. 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. Up to 3 retries SHOULD be allowed before a timeout event is declared. All traffic MUST set the TCP PSH flag to high. The source port range SHOULD be in the range of 1024 - 65535. Internal timeout should be dynamically scalable per RFC 793.

#### 4.3.2.2. Server Endpoint IP Addressing

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

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

The server pool for HTTP SHOULD listen on TCP port 80 and emulate HTTP version 1.1 with persistence. The server MUST advertise a server type. For HTTPS server, TLS 1.2 or higher MUST be used with a record size of 16383 bytes and ticket resumption or Session ID reuse SHOULD be enabled based on test scenario. The server MUST listen on port TCP 443. The server shall serve a certificate to the client. It is REQUIRED that the HTTPS server also check Host SNI information with the FQDN. Cipher suite and certificate size should be defined in the parameter section of each test scenario.

#### 4.3.3. Traffic Flow Definition

The section describes the traffic pattern between the client and server endpoints. At the beginning of the test, the server endpoint initializes and will be in a ready to accept connection state 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 the MAC and IP address. The behavior of the client is to sweep through the given server IP space, sequentially generating a recognizable service by the DUT. Thus, a balanced, mesh between client endpoints and server endpoints will be generated in a client port server port combination. Each client endpoint performs the same actions as other endpoints, with the difference being the source IP of the client endpoint and the target server IP pool. The client shall use Fully Qualified Domain Names (FQDN) in Host Headers and for TLS Server Name Indication (SNI).

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

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

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.

In the ramp up phase, the test equipment SHOULD start to generate the test traffic. It SHOULD use a set approximate number of unique client IP addresses actively to generate traffic. The traffic should ramp from zero to desired target objective. The target objective will be defined for each benchmarking test. The duration for the ramp up phase MUST be configured long enough, so that the test equipment does not overwhelm DUT/SUT's supported performance metrics namely; connections per second, concurrent TCP connections, and application transactions per second. The RECOMMENDED time duration

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

In the sustain phase, the test equipment SHOULD continue generating traffic to constant target value for a constant number of active client IPs. The RECOMMENDED time duration for sustain phase is 600 seconds. This is the phase where measurements occur.

In the ramp down/close phase, no new connections are established, and no measurements are made. The time duration for ramp up and ramp down phase SHOULD be same. The RECOMMENDED duration of this phase is between 180 to 300 seconds.

The last phase is administrative and will be when the tester merges and collates the report data.

## 5. Test Bed Considerations

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

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

Test bed reference pre-tests help to ensure that the desired traffic generator aspects such as maximum throughput and the network performance metrics such as maximum latency and maximum packet loss are met.

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

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

## 6. Reporting

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

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

### A. DUT Hardware/Virtual Configuration

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

### B. DUT Software

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

### C. DUT Enabled Features

- + Specific features, such as logging, NGFW, DPI MUST be documented

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

D. Test equipment hardware and software

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

4. Results Summary / Executive Summary

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

6.1. Key Performance Indicators

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

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



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

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

## 7. Benchmarking Tests

## 7.1. Throughput Performance With NetSecOPEN Traffic Mix

### 7.1.1. Objective

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

### 7.1.2. Test Setup

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

### 7.1.3. Test Parameters

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

#### 7.1.3.1. DUT/SUT Configuration Parameters

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

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

#### 7.1.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Traffic load objective or specification type (e.g. Throughput, SimUsers and etc.)

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

Initial throughput: 10% of the "Target throughput"

#### 7.1.3.3. Traffic Profile

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

#### 7.1.3.4. Test Results Acceptance Criteria

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

- a. Number of failed Application transaction MUST be less than 0.01% of total attempt transactions
- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.01% of total initiated TCP connections
- c. Maximum deviation (max. dev) of application transaction time or TTLB (Time To Last Byte) MUST be less than X (The value for "X" will be finalized and updated after completion of PoC test)  
The following equation MUST be used to calculate the deviation of application transaction latency or TTLB  
$$\text{max. dev} = \max((\text{avg\_latency} - \text{min\_latency}), (\text{max\_latency} - \text{avg\_latency})) / (\text{Initial latency})$$
  
Where, the initial latency is calculated using the following equation. For this calculation, the latency values (min', avg' and max') MUST be measured during test procedure step 1 as defined in Section 7.1.4.1.  
The variable latency represents application transaction latency or TTLB.  
$$\text{Initial latency} := \min((\text{avg}' \text{ latency} - \text{min}' \text{ latency}) \mid (\text{max}' \text{ latency} - \text{avg}' \text{ latency}))$$
- d. Maximum value of Time to First Byte MUST be less than X

#### 7.1.3.5. Measurement

Following KPI metrics MUST be reported for this test scenario.

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

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

#### 7.1.4. Test Procedures and expected Results

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

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

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

Configure traffic load profile of the test equipment to generate test traffic at "initial throughput" rate as described in the parameters section. 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 acceptance criteria "a" and "b" defined in Section 7.1.3.4.

If the KPI metrics do not meet the acceptance 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 "Target throughput" rate defined in the parameter table. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4. The test equipment SHOULD start to measure and record all specified KPIs. The frequency of KPI metric measurements MUST be less than 5 seconds. Continue the test until all traffic profile phases are completed.

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

##### 7.1.4.3. Step 3: Test Iteration

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

## 7.2. TCP/HTTP Connections Per Second

### 7.2.1. Objective

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

To measure connections per second, test iterations MUST use different fixed HTTP response object sizes defined in the test equipment configuration parameters section 7.2.3.2.

### 7.2.2. Test Setup

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

### 7.2.3. Test Parameters

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

#### 7.2.3.1. DUT/SUT Configuration Parameters

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

#### 7.2.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

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

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

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

#### 7.2.3.3. Test Results Acceptance Criteria

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

- a. Number of failed Application transaction MUST be less than 0.01% of total attempt transactions
- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.01% of total initiated TCP connections
- c. During the sustain phase, traffic should be forwarded at a constant rate
- d. Concurrent TCP connections SHOULD be constant during steady state. The deviation of concurrent TCP connections MUST be less than 10%. This confirms that DUT open and close the TCP connections almost at the same rate

#### 7.2.3.4. Measurement

Following KPI metrics MUST be reported for each test iteration.

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

#### 7.2.4. Test Procedures and Expected Results

The test procedure is designed to measure the TCP connections per second rate of the DUT/SUT at the sustaining period of 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 the all connected physical interfaces. All interfaces are expected to be "UP" status.

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

The DUT/SUT SHOULD reach the "initial connections per second" before the sustain phase. The measured KPIs during the sustain phase MUST meet the acceptance criteria a, b, c, and d defined in section 7.3.3.3.

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

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

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

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

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement MUST be less than 5 seconds. Continue the test until all traffic profile phases are completed.

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

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

#### 7.2.4.3. Step 3: Test Iteration

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

### 7.3. HTTP Transaction per Second

#### 7.3.1. Objective

Using HTTP 1.1 traffic, determine the maximum sustainable HTTP transactions per second supported by the DUT/SUT under different throughput load conditions.

To measure transactions per second performance under a variety of DUT Security inspection load conditions, each test iteration MUST use different fixed HTTP response object sizes defined in the test equipment configuration parameters section 7.3.3.2.

#### 7.3.2. Test Setup

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

#### 7.3.3. Test Parameters

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

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

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

##### 7.3.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2



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

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

Test scenario SHOULD be run with a single traffic profile with following attributes:

The client MUST negotiate HTTP 1.1 and close the connections with FIN immediately after completion of 10 transactions. In each test iteration, client MUST send GET command requesting a fixed HTTP response object size. The RECOMMENDED object sizes are 1, 16 and 64 KByte

#### 7.3.3.3. Test Results Acceptance Criteria

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

- a. Number of failed Application transactions MUST be zero
- b. Number of Terminated HTTP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.01% of total initiated HTTP sessions
- c. Traffic should be forwarded at a constant rate
- d. Average Time to TCP First Byte MUST be constant and not increase more than 10%
- e. The deviation of concurrent TCP connection Must be less than 10%

#### 7.3.3.4. Measurement

Following KPI metrics MUST be reported for this test scenario.

average TCP Connections per second, average Throughput, Average Time to TCP First Byte and average application transaction latency.

#### 7.3.4. Test Procedures and Expected Results

The test procedure is designed to measure the HTTP transactions per second 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.

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

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

Configure traffic load profile of the test equipment to establish "initial HTTP transactions per second" as defined in the parameters section. The traffic load profile CAN be defined as described in the section 4.3.4.

The DUT/SUT SHOULD reach the "initial HTTP transactions per second" before the sustain phase. The measured KPIs during the sustain phase MUST meet the acceptance criteria a, b, c, and d defined in section 7.3.3.3.

If the KPI metrics do not meet the acceptance 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 "Target HTTP transactions per second" defined in the parameters table. The test equipment SHOULD follow the traffic load profile definition as described in the section 4.3.4.

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

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement MUST be less than 5 seconds. Continue the test until all traffic profile phases are completed.

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

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

#### 7.3.4.3. Step 3: Test Iteration

Determine the maximum and average achievable HTTP transactions per second within the acceptance criteria. Final test iteration MUST be performed for the test duration defined in Section 4.3.4.

### 7.4. TCP/HTTP Transaction Latency

#### 7.4.1. Objective

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

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

#### 7.4.2. Test Setup

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

#### 7.4.3. Test Parameters

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

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

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

##### 7.4.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

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

HTTP transaction per TCP connection: one test scenario with single transaction and another scenario with 10 transactions

Test scenario SHOULD be run with a single traffic profile with following attributes:

To measure application transaction latency with a single connection per transaction and a single connection with multiple transactions the tests should run twice:

1st test run: The client MUST negotiate HTTP 1.1 and close the connection with FIN immediately after completion of the transaction.

2nd test run: The client MUST negotiate HTTP 1.1 and close the connection after 10 transactions (GET and RESPONSE) within a single TCP connection.

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

#### 7.4.3.3. Test Results Acceptance Criteria

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

Generic criteria:

- a. Number of failed Application transaction MUST be zero.
- b. Number of Terminated TCP connection due to unexpected TCP RST sent by DUT/SUT MUST be zero.
- c. During the sustain phase, traffic should be forwarded at a constant rate.
- d. During the sustain phase, Average connect time and average transaction time MUST be constant and latency deviation SHOULD not increase more than 10%.

- e. Concurrent TCP connections should be constant during steady state. This confirms the DUT opens and closes TCP connections at the same rate.
- f. After ramp up the DUT MUST achieve the target connections per second objective defined in the parameter section 7.4.3.2 and it remains in that state for the entire test duration (sustain phase).

#### 7.4.3.4. Measurement

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

average TCP connections per second and average application transaction latency needs to be recorded.

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

#### 7.4.4. Test Procedures and Expected Results

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

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

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

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

The DUT/SUT SHOULD reach the "initial connections per second" before the sustain phase. The measured KPIs during the sustain phase MUST meet the acceptance criteria a, b, c, d ,e and f defined in section 7.4.3.3.

If the KPI metrics do not meet the acceptance 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 connections per second" defined in the parameters table. The test equipment SHOULD follow the traffic load profile definition as described in the section 4.3.4.

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

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement MUST be less than 5 seconds. Continue the test until all traffic profile phases are completed. DUT/SUT is expected to reach the desired target connections per second rate at the sustain phase. In addition, the measured KPIs must meet all acceptance criteria.

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

#### 7.4.4.3. Step 3: Test Iteration

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

### 7.5. HTTP Throughput

#### 7.5.1. Objective

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

#### 7.5.2. Test Setup

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

### 7.5.3. Test Parameters

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

#### 7.5.3.1. DUT/SUT Configuration Parameters

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

#### 7.5.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

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

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

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

Table 3: Mixed Objects

#### 7.5.3.3. Test Results Acceptance Criteria

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

- a. Number of failed Application transaction MUST be less than 0.01% of attempt transaction.
- b. Traffic should be forwarded constantly.
- c. The deviation of concurrent TCP connection Must be less than 10%
- d. The deviation of average HTTP transaction latency MUST be less than 10%

#### 7.5.3.4. Measurement

The KPI metrics MUST be reported for this test scenario:



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

#### 7.5.4. 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.

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

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

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

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 7.5.3.4.

The measured KPIs during the sustain phase MUST meet the acceptance criteria "a" defined in Section 7.5.3.3.

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

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

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement MUST be less than 5 seconds. Continue the test until all traffic profile phases are completed.

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

Perform the test separately for each HTTP response object size (16k, 64k, 256k and mixed HTTP response objects).

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

#### 7.5.4.3. Step 3: Test Iteration

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

### 7.6. Concurrent TCP/HTTP Connection Capacity

#### 7.6.1. Objective

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

#### 7.6.2. Test Setup

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

#### 7.6.3. Test Parameters

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

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

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

##### 7.6.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

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

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

Each client sends 10 GET commands requesting 1Kbyte HTTP response object in the same TCP connection (10 transactions/TCP connection) and the delay (think time) between the transaction MUST be X seconds. The value for think time (X) MUST be defined to achieve 15% of maximum throughput measured in test scenario 7.5.

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.

#### 7.6.3.3. Test Results Acceptance Criteria

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

- a. Number of failed Application transaction MUST be zero
- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.01% of total initiated TCP connections
- c. During the sustain phase, traffic should be forwarded constantly at the rate defined in the parameter section 7.6.3.2
- d. During the sustain phase, the maximum deviation (max. dev) of application transaction latency or TTLB (Time To Last Byte) MUST be less than 10%

#### 7.6.3.4. Measurement

Following KPI metrics MUST be reported for this test scenario:

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

#### 7.6.4. 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.

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

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

Configure test equipment to generate background traffic as defined in section 7.6.3.2. Measure throughput, concurrent TCP connections, and TCP connections per second.

While generating the background traffic, configure another traffic profile on the test equipment to establish "initial concurrent TCP connections" defined in the section 7.6.3.2. The traffic load profile CAN be defined as described in the section Error: Reference source not found.

During the sustain phase, the DUT/SUT SHOULD reach the "initial concurrent TCP connections" plus concurrent TCP connections measured in background traffic. The measured KPIs during the sustain phase MUST meet the acceptance criteria "a" and "b" defined in the section Error: Reference source not found

If the KPI metrics do not meet the acceptance 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 concurrent TCP connections" defined in the parameters table. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4.

Configure test equipment to establish "Target concurrent TCP connections" minus concurrent TCP connections measured in background traffic. The test equipment SHOULD follow the traffic load profile definition as described in the section Error: Reference source not found.

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

The test equipment SHOULD start to measure and record KPIs defined in section 7.6.3.4. The frequency of measurement MUST be less than 5 seconds. Continue the test until all traffic profile phases are completed.

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

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

#### 7.6.4.3. Step 3: Test Iteration

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

### 7.7. TCP/HTTPS Connections per second

#### 7.7.1. Objective

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

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

##### 7.7.3.2

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 7.7.3.2 to measure connections per second performance under a variety of DUT Security inspection load conditions.

#### 7.7.2. Test Setup

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

#### 7.7.3. Test Parameters

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

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

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

#### 7.7.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

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

Ciphers and keys:

1. ECHDE-ECDSA-AES128-GCM-SHA256 with Prime256v1 (Signature Hash Algorithmn: ecdsa\_secp256r1\_sha256 and Supported group: secp256r1)
2. ECDHE-RSA-AES128-GCM-SHA256 with RSA 2048 (Signature Hash Algorithmn: rsa\_pkcs1\_sha256 and Supported group: secp256)
3. ECDHE-ECDSA-AES256-GCM-SHA384 with Secp521 (Signature Hash Algorithmn: ecdsa\_secp256r1\_sha384 and Supported group: secp521r1)
4. ECDHE-RSA-AES256-GCM-SHA384 with RSA 4096 (Signature Hash Algorithmn: rsa\_pkcs1\_sha384 and Supported group: secp256)

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

Each client connection MUST perform a full handshake with server certificate (no Certificate on client side) and MUST NOT use session reuse or resumption. TLS record size MAY be optimized for the HTTPS response object size up to a record size of 16K.

#### 7.7.3.3. Test Results Acceptance Criteria

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

- a. Number of failed Application transaction MUST be less than 0.01% of attempt transactions
- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.01% of total initiated TCP connections
- c. During the sustain phase, traffic should be forwarded at a constant rate
- d. Concurrent TCP connections SHOULD be constant during steady state. This confirms that DUT open and close the TCP connections at the same rate

#### 7.7.3.4. Measurement

Following KPI metrics MUST be reported for this test scenario:

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

#### 7.7.4. 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.

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

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

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

The DUT/SUT SHOULD reach the "initial connections per second" before the sustain phase. The measured KPIs during the sustain phase MUST meet the acceptance criteria a, b, c, and d defined in section 7.7.3.3.

If the KPI metrics do not meet the acceptance 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 "Target connections per second" defined in the parameters table. The test equipment SHOULD follow the traffic load profile definition as described in the section 4.3.4.

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

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement MUST be less than 5 seconds. Continue the test until all traffic profile phases are completed.

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

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

#### 7.7.4.3. Step 3: Test Iteration

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

### 7.8. HTTPS Transaction per Second

#### 7.8.1. Objective

Using HTTPS traffic, determine the maximum sustainable HTTPS transactions per second supported by the DUT/SUT under different throughput load conditions.

To measure transactions per second performance under a variety of DUT Security inspection load conditions, each test iteration MUST use different fixed HTTPS transaction object sizes defined in the test equipment configuration parameters section 7.8.3.2.



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 7.8.3.2.

#### 7.8.2. Test Setup

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

#### 7.8.3. Test Parameters

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

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

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

##### 7.8.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

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

Ciphers and keys:

1. ECHDE-ECDSA-AES128-GCM-SHA256 with Prime256v1 (Signature Hash Algorithmn: ecdsa\_secp256r1\_sha256 and Supported group: secp256r1)
2. ECDHE-RSA-AES128-GCM-SHA256 with RSA 2048 (Signature Hash Algorithmn: rsa\_pkcs1\_sha256 and Supported group: secp256)

3. ECDHE-ECDSA-AES256-GCM-SHA384 with Secp521 (Signature Hash Algorithmn: ecdsa\_secp256r1\_sha384 and Supported group: secp521r1)
4. ECDHE-RSA-AES256-GCM-SHA384 with RSA 4096 (Signature Hash Algorithmn: rsa\_pkcs1\_sha384 and Supported group: secp256)

The client MUST negotiate HTTPS 1.1 and close the connection with FIN immediately after completion of 10 transactions.

HTTPS 1.1 with GET command requesting a single 1, 16 and 64 KByte objects.

Each client connection MUST perform a full handshake with server certificate and SHOULD NOT use session reuse or resumption.

TLS record size MAY be optimized for the object size up to a record size of 16K.

#### 7.8.3.3. Test Results Acceptance Criteria

The following test Criteria is defined as test results acceptance criteria. Test results acceptance 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 MUST be zero
- b. Number of Terminated HTTP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.01% of total initiated HTTP sessions
- c. Average Time to TCP First Byte MUST be constant and not increase more than 10%
- d. The deviation of concurrent TCP connection Must be less than 10%

#### 7.8.3.4. Measurement

Following KPI metrics MUST be reported for this test scenario.

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

#### 7.8.4. Test Procedures and Expected Results

The test procedure is designed to measure the HTTPS transactions 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, HTTPS response object sizes and ciphers and keys.

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

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

Configure traffic load profile of the test equipment to establish "initial HTTPS transactions per second" as defined in the parameters section. The traffic load profile CAN be defined as described in the section 4.3.4.

The DUT/SUT SHOULD reach the "initial HTTPS transactions per second" before the sustain phase. The measured KPIs during the sustain phase MUST meet the acceptance criteria a, b, c, and d defined in section 7.8.3.3.

If the KPI metrics do not meet the acceptance 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 HTTPS transactions per second" defined in the parameters table. The test equipment SHOULD follow the traffic load profile definition as described in the section 4.3.4.

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

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement MUST be less than 5 seconds. Continue the test until all traffic profile phases are completed.

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

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

#### 7.8.4.3. Step 3: Test Iteration

Determine the maximum and average achievable HTTPS transactions per second within the acceptance criteria. Final test iteration MUST be performed for the test duration defined in Section 4.3.4.

### 7.9. HTTPS Transaction Latency

#### 7.9.1. Objective

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

Test iterations MUST be performed with different HTTPS response object sizes twice, one with a single transaction and the other with multiple transactions within a single TCP connection.

#### 7.9.2. Test Setup

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

#### 7.9.3. Test Parameters

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

##### 7.9.3.1. DUT/SUT Configuration Parameters

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

##### 7.9.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Cipher suites and key size: ECDHE-ECDSA-AES256-GCM-SHA384 with Secp521 bits key size (Signature Hash Algorithmn: ecdsa\_secp256r1\_sha384 and Supported group: secp521r1)

Target connections per second: 50% of the value measured in test scenario TCP/HTTPS Connections per second (Section 7.7)

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

HTTPS transaction per connection: one test scenario with a single transaction and another scenario with 10 transactions

Test scenario SHOULD be run with a single traffic profile with following attributes:

To measure application transaction latency with a single connection per transaction and single connection with multiple transactions the tests should run twice:

1st test run: The client MUST negotiate HTTPS 1.1 and close the connection with FIN immediately after completion of the transaction.

2nd test run: The client MUST negotiate HTTPS 1.1 and close the connection after 10 transactions (GET and RESPONSE) within a single TCP connection.

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

#### 7.9.3.3. Test Results Acceptance Criteria

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

Generic creteria:

- a. Number of failed Application transactions MUST be zero

- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be zero.
- c. During the sustain phase, traffic should be forwarded at a constant rate.
- d. During the sustain phase and average application transaction latency MUST be constant and latency deviation SHOULD NOT increase more than 10%.
- e. Concurrent TCP connections SHOULD be constant during steady state. This confirms the DUT opens and closes the TCP connections at the same rate.
- f. After ramp up the DUT MUST achieve the target connections per second objective defined in the parameter section and remain in that state for the entire duration of the sustain phase.

#### 7.9.3.4. Measurement

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

average TCP connections per second and average application transaction latency or TTLB needs to be recorded.

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

#### 7.9.4. Test Procedures and Expected Results

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

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

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

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

The DUT/SUT SHOULD reach the "initial connections per second" before the sustain phase. The measured KPIs during the sustain phase MUST meet the acceptance criteria a, b, c, d ,e and f defined in section 7.4.3.3.

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

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

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

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

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement MUST be less than 5 seconds. Continue the test until all traffic profile phases are completed. DUT/SUT is expected to reach the desired target connections per second rate at the sustain phase. In addition, the measured KPIs must meet all acceptance criteria.

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

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

#### 7.9.4.3. Step 3: Test Iteration

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

### 7.10. HTTPS Throughput

#### 7.10.1. Objective

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

Test iterations MUST include common cipher suites and key strengths as well as forward looking stronger keys. Specific test iterations

MUST include the ciphers and keys defined in the parameter section 7.10.3.2.

#### 7.10.2. 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.

#### 7.10.3. Test Parameters

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

##### 7.10.3.1. DUT/SUT Configuration Parameters

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

##### 7.10.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

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

Ciphers and keys:

1. ECHDE-ECDSA-AES128-GCM-SHA256 with Prime256v1 (Signature Hash Algorithmn: ecdsa\_secp256r1\_sha256 and Supported group: secp256r1)
2. ECDHE-RSA-AES128-GCM-SHA256 with RSA 2048 (Signature Hash Algorithmn: rsa\_pkcs1\_sha256 and Supported group: secp256)



3. ECDHE-ECDSA-AES256-GCM-SHA384 with Secp521 (Signature Hash Algorithmn: ecdsa\_secp256r1\_sha384 and Supported group: secp521r1)
4. ECDHE-RSA-AES256-GCM-SHA384 with RSA 4096 (Signature Hash Algorithmn: rsa\_pkcs1\_sha384 and Supported group: secp256)

HTTPS response object size: 16KB, 64KB, 256KB and mixed object defined in the table below.

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

Table 4: Mixed Objects

Each client connection MUST perform a full handshake with server certificate (no Certificate on client side) and 50% of connection SHOULD use session reuse or resumption.

TLS record size MAY be optimized for the HTTPS response object size up to a record size of 16K.

#### 7.10.3.3. Test Results Acceptance Criteria

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

- a. Number of failed Application transaction MUST be less than 0.01% of attempt transaction.
- b. Traffic should be forwarded constantly.
- c. The deviation of concurrent TCP connection Must be less than 10%
- d. The deviation of average application transaction latency MUST be less than 10%

#### 7.10.3.4. Measurement

The KPI metrics MUST be reported for this test scenario:

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

#### 7.10.4. 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.

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

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

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

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 7.10.3.4.

The measured KPIs during the sustain phase MUST meet the acceptance criteria "a" defined in Section 7.10.3.3.

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

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

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement MUST be less than 5 seconds. Continue the test until all traffic profile phases are completed.

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

Perform the test separately for each HTTPS response object size (16k, 64k, 256k and mixed HTTPS response objects).

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

#### 7.10.4.3. Step 3: Test Iteration

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

### 7.11. Concurrent TCP/HTTPS Connection Capacity

#### 7.11.1. Objective

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

#### 7.11.2. 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.

#### 7.11.3. Test Parameters

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

##### 7.11.3.1. DUT/SUT Configuration Parameters

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

#### 7.11.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in the section Error: Reference source not found. Following parameters MUST be documented for this test scenario:

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

Cipher suites and key size: ECDHE-ECDSA-AES256-GCM-SHA384 with Secp521 bits key size (Signature Hash Algorithmn: ecdsa\_secp256r1\_sha384 and Supported group: secp521r1)

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

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

Maximum connections per second during ramp up phase: 50% of maximum connections per second measured in test scenario TCP/HTTPS Connections per second (Section 7.7)

Throughput for background traffic: 10% of maximum throughput measured in test scenario HTTPS Throughput (Section 7.10)7.10 using an HTTPS response object size of 16Kbyte with a matching cipher and key size to what is being tested in this test

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 times of GET commands requesting 1Kbyte HTTPS response object in the same TCP connections (10 transactions/TCP connection) and the delay (think time) between the transaction MUST be X seconds. The value for think time (X) MUST be defined to achieve 15% of maximum throughput measured in test scenario 7.10.

The established connections (except background traffic connection) SHOULD remain open until the end phase of the test. During the ramp down phase, all connections should be successfully closed with FIN.

#### 7.11.3.3. Test Results Acceptance Criteria

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

- a. Number of failed Application transactions MUST be zero.
- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.01% of total initiated TCP connections
- c. During the sustain phase, traffic should be forwarded constantly at the rate defined in the parameter section 7.11.3.2
- d. During the sustain phase, then maximum deviation (max. dev) of application transaction latency or TTLB (Time To Last Byte) MUST be less than 10%

#### 7.11.3.4. Measurement

Following KPI metrics MUST be reported for this test scenario:

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

#### 7.11.4. 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.

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

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

Configure test equipment to generate background traffic as defined in section 7.3.11.2. Measure throughput, concurrent TCP connections, and connections per second.

While generating the background traffic, configure another traffic profile on the test equipment to establish "initial concurrent TCP connections" defined in the section 7.11.3.2. The traffic load

profile CAN be defined as described in the section Error: Reference source not found

During the sustain phase, the DUT/SUT SHOULD reach the "initial concurrent TCP connections" plus concurrent TCP connections measured in background traffic. The measured KPIs during the sustain phase MUST meet the acceptance criteria "a" and "b" defined in the section Error: Reference source not found

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

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

Configure test equipment to establish "Target concurrent TCP connections" minus concurrent TCP connections measured in background traffic. The test equipment SHOULD follow the traffic load profile definition as described in the section 4.3.4

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

The test equipment SHOULD start to measure and record KPIs defined in section 7.11.3.4. The frequency of measurement MUST be less than 5 seconds. Continue the test until all traffic profile phases are completed.

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

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

#### 7.11.4.3. Step 3: Test Iteration

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

### 8. Formal Syntax

### 9. IANA Considerations

This document makes no request of IANA.

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

## 10. Acknowledgements

Acknowledgements will be added in the future release.

## 11. Contributors

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

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

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

## 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>>.

### 12.2. Informative References

- [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>>.

## Appendix A. NetSecOPEN Basic Traffic Mix

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

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

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

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

The mix MUST be of sufficient complexity and volume to render differences in individual apps as statistically insignificant. For example, changes in like to like apps - such as one type of video service vs. another both consist of larger objects whereas one news site vs. another both typically have more connections then other apps because of trackers and embedded advertising content. To achieve sufficient complexity, a mix MUST have:

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



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

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

Office365 FQDN	Bytes	Transaction
rl.res.office365.com	14,056,960	192
s1-word-edit-15.cdn.office.net	6,731,019	22
company1-my.sharepoint.com	6,269,492	42
swx.cdn.skype.com	6,100,027	12
static.sharepointonline.com	6,036,947	41
spoprod-a.akamaihd.net	3,904,250	25
s1-excel-15.cdn.office.net	2,767,941	16
outlook.office365.com	2,047,301	86
shellprod.msocdn.com	1,008,370	11
word-edit.officeapps.live.com	932,080	25
res.delve.office.com	760,146	2
s1-powerpoint-15.cdn.office.net	557,604	3
appsforoffice.microsoft.com	511,171	5
powerpoint.officeapps.live.com	471,625	14
excel.officeapps.live.com	342,040	14
s1-officeapps-15.cdn.office.net	331,343	5
webdir0a.online.lync.com	66,930	15
portal.office.com	13,956	1
config.edge.skype.com	6,911	2

clientlog.portal.office.com	6,608	8	
+-----+-----+-----+			
webdir.online.lync.com	4,343	5	
+-----+-----+-----+			
graph.microsoft.com	2,289	2	
+-----+-----+-----+			
nam.loki.delve.office.com	1,812	5	
+-----+-----+-----+			
login.microsoftonline.com	464	2	
+-----+-----+-----+			
login.windows.net	232	1	
+-----+-----+-----+			

Table 5: Office365

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

FQDN	URL Path	Object size	
+=====+			
company1-my.sharepoint.com	/personal...	23,132	
+-----+			
word-edit.officeapps.live.com	/we/WsaUpload.ashx	2	
+-----+			
static.sharepointonline.com	/bld/.../blank.js	454	
+-----+			
static.sharepointonline.com	/bld/.../ initstrings.js	23,254	
+-----+			
static.sharepointonline.com	/bld/.../init.js	292,740	
+-----+			
company1-my.sharepoint.com	/ScriptResource...	102,774	
+-----+			
company1-my.sharepoint.com	/ScriptResource...	40,329	
+-----+			
company1-my.sharepoint.com	/WebResource...	23,063	
+-----+			
word-edit.officeapps.live.com	/we/wordeditorframe.	60,657	

	aspx...	
static.sharepointonline.com	/bld/_layouts/.../ blank.js	454
s1-word-edit-15.cdn.office.net	/we/s/.../ EditSurface.css	19,201
s1-word-edit-15.cdn.office.net	/we/s/.../ WordEditor.css	221,397
s1-officeapps-15.cdn.office.net	/we/s/.../ Microsoft Ajax.js	107,571
s1-word-edit-15.cdn.office.net	/we/s/.../ wacbootwe.js	39,981
s1-officeapps-15.cdn.office.net	/we/s/.../ CommonIntl.js	51,749
s1-word-edit-15.cdn.office.net	/we/s/.../ Compat.js	6,050
s1-word-edit-15.cdn.office.net	/we/s/.../ Box4Intl.js	54,158
s1-word-edit-15.cdn.office.net	/we/s/.../ WoncaIntl.js	24,946
s1-word-edit-15.cdn.office.net	/we/s/.../ WordEditorIntl.js	53,515
s1-word-edit-15.cdn.office.net	/we/s/.../ WordEditorExp.js	1,978,712
s1-word-edit-15.cdn.office.net	/we/s/.../jSanity.js	10,912
word-edit.officeapps.live.com	/we/OneNote.ashx	145,708

Table 6: Office365 Word Transactions

For application identification the HTTPS/TLS traffic MUST include realistic Certificate Subject Common Name (CN) data as well as Server Name Indications. For example, a DUT may detect Facebook Chat traffic by inspecting the certificate and detecting \*.facebook.com in the certificate subject CN and subsequently detect the word chat in

the FQDN 5-edge-chat.facebook.com and identify traffic on the connection to be Facebook Chat.

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

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

Table 7: Office365 SNI and CN Pairs Examples

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

The enterprise perimeter traffic mix consists of 70% HTTPS and 30% HTTP by Bytes, 58% HTTPS and 42% HTTP by Transactions. By connections with a single connection per FQDN the mix consists of 43% HTTPS and 57% HTTP. With multiple connections per FQDN the HTTPS percentage is higher.

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

Application	FQDNs	Transactions	Bytes
Office365	26	558	52,931,947

Box	4	90	23,276,089	
+-----+	+-----+	+-----+	+-----+	+-----+
Salesforce	6	365	23,137,548	
+-----+	+-----+	+-----+	+-----+	+-----+
Gmail	13	139	16,399,289	
+-----+	+-----+	+-----+	+-----+	+-----+
LinkedIn	10	206	15,040,918	
+-----+	+-----+	+-----+	+-----+	+-----+
DailyMotion	8	77	14,751,514	
+-----+	+-----+	+-----+	+-----+	+-----+
GoogleDocs	2	71	14,205,476	
+-----+	+-----+	+-----+	+-----+	+-----+
Wikia	15	159	13,909,777	
+-----+	+-----+	+-----+	+-----+	+-----+
Foxnews	82	499	13,758,899	
+-----+	+-----+	+-----+	+-----+	+-----+
Yahoo Finance	33	254	13,134,011	
+-----+	+-----+	+-----+	+-----+	+-----+
Youtube	8	97	13,056,216	
+-----+	+-----+	+-----+	+-----+	+-----+
Facebook	4	207	12,726,231	
+-----+	+-----+	+-----+	+-----+	+-----+
CNBC	77	275	11,939,566	
+-----+	+-----+	+-----+	+-----+	+-----+
Lightreading	27	304	11,200,864	
+-----+	+-----+	+-----+	+-----+	+-----+
BusinessInsider	16	142	11,001,575	
+-----+	+-----+	+-----+	+-----+	+-----+
Alexa	5	153	10,475,151	
+-----+	+-----+	+-----+	+-----+	+-----+
CNN	41	206	10,423,740	
+-----+	+-----+	+-----+	+-----+	+-----+
Twitter Video	2	72	10,112,820	
+-----+	+-----+	+-----+	+-----+	+-----+
Cisco Webex	1	213	9,988,417	
+-----+	+-----+	+-----+	+-----+	+-----+
Slack	3	40	9,938,686	
+-----+	+-----+	+-----+	+-----+	+-----+
Google Maps	5	191	8,771,873	
+-----+	+-----+	+-----+	+-----+	+-----+
SpectrumIEEE	7	145	8,682,629	
+-----+	+-----+	+-----+	+-----+	+-----+
Yelp	9	146	8,607,645	
+-----+	+-----+	+-----+	+-----+	+-----+
Vimeo	12	74	8,555,960	
+-----+	+-----+	+-----+	+-----+	+-----+
Wikihow	11	140	8,042,314	
+-----+	+-----+	+-----+	+-----+	+-----+

Netflix	3	31	7,839,256	
+-----+	+-----+	+-----+	+-----+	+-----+
Instagram	3	114	7,230,883	
+-----+	+-----+	+-----+	+-----+	+-----+
Morningstar	30	150	7,220,121	
+-----+	+-----+	+-----+	+-----+	+-----+
Docusign	5	68	6,972,738	
+-----+	+-----+	+-----+	+-----+	+-----+
Twitter	1	100	6,939,150	
+-----+	+-----+	+-----+	+-----+	+-----+
Tumblr	11	70	6,877,200	
+-----+	+-----+	+-----+	+-----+	+-----+
Whatsapp	3	46	6,829,848	
+-----+	+-----+	+-----+	+-----+	+-----+
Imdb	16	251	6,505,227	
+-----+	+-----+	+-----+	+-----+	+-----+
NOAAgov	1	44	6,316,283	
+-----+	+-----+	+-----+	+-----+	+-----+
IndustryWeek	23	192	6,242,403	
+-----+	+-----+	+-----+	+-----+	+-----+
Spotify	18	119	6,231,013	
+-----+	+-----+	+-----+	+-----+	+-----+
AutoNews	16	165	6,115,354	
+-----+	+-----+	+-----+	+-----+	+-----+
Evernote	3	47	6,063,168	
+-----+	+-----+	+-----+	+-----+	+-----+
NatGeo	34	104	6,026,344	
+-----+	+-----+	+-----+	+-----+	+-----+
BBC News	18	156	5,898,572	
+-----+	+-----+	+-----+	+-----+	+-----+
Investopedia	38	241	5,792,038	
+-----+	+-----+	+-----+	+-----+	+-----+
Pinterest	8	102	5,658,994	
+-----+	+-----+	+-----+	+-----+	+-----+
Succesfactors	2	112	5,049,001	
+-----+	+-----+	+-----+	+-----+	+-----+
AbaJournal	6	93	4,985,626	
+-----+	+-----+	+-----+	+-----+	+-----+
Pbworks	4	78	4,670,980	
+-----+	+-----+	+-----+	+-----+	+-----+
NetworkWorld	42	153	4,651,354	
+-----+	+-----+	+-----+	+-----+	+-----+
WebMD	24	280	4,416,736	
+-----+	+-----+	+-----+	+-----+	+-----+
OilGasJournal	14	105	4,095,255	
+-----+	+-----+	+-----+	+-----+	+-----+
Trello	5	39	4,080,182	
+-----+	+-----+	+-----+	+-----+	+-----+

BusinessWire	5	109	4,055,331	
+-----+	+-----+	+-----+	+-----+	+-----+
Dropbox	5	17	4,023,469	
+-----+	+-----+	+-----+	+-----+	+-----+
Nejm	20	190	4,003,657	
+-----+	+-----+	+-----+	+-----+	+-----+
OilGasDaily	7	199	3,970,498	
+-----+	+-----+	+-----+	+-----+	+-----+
Chase	6	52	3,719,232	
+-----+	+-----+	+-----+	+-----+	+-----+
MedicalNews	6	117	3,634,187	
+-----+	+-----+	+-----+	+-----+	+-----+
Marketwatch	25	142	3,291,226	
+-----+	+-----+	+-----+	+-----+	+-----+
Imgur	5	48	3,189,919	
+-----+	+-----+	+-----+	+-----+	+-----+
NPR	9	83	3,184,303	
+-----+	+-----+	+-----+	+-----+	+-----+
Onelogin	2	31	3,132,707	
+-----+	+-----+	+-----+	+-----+	+-----+
Concur	2	50	3,066,326	
+-----+	+-----+	+-----+	+-----+	+-----+
Service-now	1	37	2,985,329	
+-----+	+-----+	+-----+	+-----+	+-----+
Apple itunes	14	80	2,843,744	
+-----+	+-----+	+-----+	+-----+	+-----+
BerkeleyEdu	3	69	2,622,009	
+-----+	+-----+	+-----+	+-----+	+-----+
MSN	39	203	2,532,972	
+-----+	+-----+	+-----+	+-----+	+-----+
Indeed	3	47	2,325,197	
+-----+	+-----+	+-----+	+-----+	+-----+
MayoClinic	6	56	2,269,085	
+-----+	+-----+	+-----+	+-----+	+-----+
Ebay	9	164	2,219,223	
+-----+	+-----+	+-----+	+-----+	+-----+
UCLAedu	3	42	1,991,311	
+-----+	+-----+	+-----+	+-----+	+-----+
ConstructionDive	5	125	1,828,428	
+-----+	+-----+	+-----+	+-----+	+-----+
EducationNews	4	78	1,605,427	
+-----+	+-----+	+-----+	+-----+	+-----+
BofA	12	68	1,584,851	
+-----+	+-----+	+-----+	+-----+	+-----+
ScienceDirect	7	26	1,463,951	
+-----+	+-----+	+-----+	+-----+	+-----+
Reddit	8	55	1,441,909	
+-----+	+-----+	+-----+	+-----+	+-----+

FoodBusinessNews	5	49	1,378,298	
+-----+	+-----+	+-----+	+-----+	+-----+
Amex	8	42	1,270,696	
+-----+	+-----+	+-----+	+-----+	+-----+
Weather	4	50	1,243,826	
+-----+	+-----+	+-----+	+-----+	+-----+
Wikipedia	3	27	958,935	
+-----+	+-----+	+-----+	+-----+	+-----+
Bing	1	52	697,514	
+-----+	+-----+	+-----+	+-----+	+-----+
ADP	1	30	508,654	
+-----+	+-----+	+-----+	+-----+	+-----+
+-----+	+-----+	+-----+	+-----+	+-----+
Grand Total	983	10021	569,819,095	
+-----+	+-----+	+-----+	+-----+	+-----+

Table 8: Summary of NetSecOPEN Enterprise Perimeter Traffic Mix

## Authors' Addresses

Balamuhunthan Balarajah  
 EANTC AG  
 Salzufer 14  
 Berlin 10587  
 Germany

Email: balarajah@eantc.de

Carsten Rossenhoevel  
 EANTC AG  
 Salzufer 14  
 Berlin 10587  
 Germany

Email: cross@eantc.de



BMWG  
Internet-Draft  
Intended status: Informational  
Expires: September 11, 2017

L. Huang, Ed.  
R. Gu, Ed.  
China Mobile  
Bob. Mandeville  
Iometrix  
Brooks. Hickman  
Spirent Communications  
March 10, 2017

Benchmarking Methodology for Virtualization Network Performance  
draft-huang-bmwg-virtual-network-performance-02

Abstract

As the virtual network has been widely established in IDC, the performance of virtual network has become a valuable consideration to the IDC managers. This draft introduces a benchmarking methodology for virtualization network performance based on virtual switch.

Status of This Memo

This Internet-Draft is submitted to IETF 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 <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on September 11, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. Test Considerations . . . . .	3
4. Key Performance Indicators . . . . .	5
5. Test Setup . . . . .	6
6. Benchmarking Tests . . . . .	7
6.1. Throughput . . . . .	7
6.1.1. Objectives . . . . .	7
6.1.2. Configuration parameters . . . . .	7
6.1.3. Test parameters . . . . .	8
6.1.4. Test process . . . . .	8
6.1.5. Test result format . . . . .	8
6.2. Frame loss rate . . . . .	8
6.2.1. Objectives . . . . .	9
6.2.2. Configuration parameters . . . . .	9
6.2.3. Test parameters . . . . .	9
6.2.4. Test process . . . . .	9
6.2.5. Test result format . . . . .	10
6.3. CPU consumption . . . . .	10
6.3.1. Objectives . . . . .	10
6.3.2. Configuration parameters . . . . .	10
6.3.3. Test parameters . . . . .	11
6.3.4. Test process . . . . .	11
6.3.5. Test result format . . . . .	11
6.4. MEM consumption . . . . .	12
6.4.1. Objectives . . . . .	12
6.4.2. Configuration parameters . . . . .	12
6.4.3. Test parameters . . . . .	13
6.4.4. Test process . . . . .	13
6.4.5. Test result format . . . . .	13
6.5. Latency . . . . .	14
6.5.1. Objectives . . . . .	15
6.5.2. Configuration parameters . . . . .	15
6.5.3. Test parameters . . . . .	15
6.5.4. Test process . . . . .	15
6.5.5. Test result format . . . . .	16
7. Security Considerations . . . . .	16
8. IANA Considerations . . . . .	16
9. Normative References . . . . .	16
Authors' Addresses . . . . .	17

## 1. Introduction

As the virtual network has been widely established in IDC, the performance of virtual network has become a valuable consideration to the IDC managers. This draft introduces a benchmarking methodology

for virtualization network performance based on virtual switch as the DUT.

## 2. Terminology

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

## 3. Test Considerations

In a conventional test setup with Non-Virtual test ports, it is quite legitimate to assume that test ports provide the golden standard in measuring the performance metrics. If test results are sub optimal, it is automatically assumed that the Device-Under-Test (DUT) is at fault. For example, when testing throughput at a given frame size, if the test result shows less than 100% throughput, we can safely conclude that it's the DUT that can not deliver line rate forwarding at that frame size(s). We never doubt that the tester can be an issue.

While in a virtual test environment where both the DUT as well as the test tool itself are software based, it's quite a different story. Just like the DUT, tester running as software will have its own performance peak under various conditions.

There are two types of vSwitch according to different installation location. One is VM based vSwitch which is installed on a virtual machine, another is vSwitch directly installed on the host OS (similar to hypervisor).The latter is much more popular currently.

Tester's calibration is essential in benchmarking testing in a virtual environment. Furthermore, to reduce the enormous combination of various conditions, tester must be calibrated with the exact same combination and parameter settings the user wants to measure against the DUT. A slight variation of conditions and parameter values will cause inaccurate measurements of the DUT.

While it is difficult to list the exact combination and parameter settings, the following table attempts to give the most common example how to calibrate a tester before testing a DUT (VSWITCH).

Sample calibration permutation:

Hypervisor Type	VM VNIC Speed	VM Memory CPU Allocation	Frame Size	Throughput
ESXi	1G/10G	512M/1Core	64	
			128	
			256	
			512	
			1024	
			1518	

Figure 1: Sample Calibration Permutation

Key points are as following:

a) The hypervisor type is of ultimate importance to the test results. VM tester(s) MUST be installed on the same hypervisor type as the DUT (VSWITCH). Different hypervisor type has an influence on the test result.

b) The VNIC speed will have an impact on testing results. Testers MUST calibrate against all VNIC speeds.

c) VM allocations of CPU resources and memory have an influence on test results.

d) Frame sizes will affect the test results dramatically due to the nature of virtual machines.

e) Other possible extensions of above table: The number of VMs to be created, latency reading, one VNIC per VM vs. multiple VM sharing one VNIC, and uni-directional traffic vs. bi-directional traffic.

Besides, the compute environment including the hardware should be also recorded.

Compute environment componenets	Model
CPU	
Memory	
Hard Disk	
10G Adaptors	
Blade/Motherboard	

Figure 2: Compute Environment

It's important to confirm test environment for tester's calibration as close to the environment a virtual DUT (VSWITCH) involved in for the benchmark test. Key points which SHOULD be noticed in test setup are listed as follows.

1. One or more VM tester(s) need to be created for both traffic generation and analysis.
2. vSwitch has an influence on performance penalty due to extra resource occupation.
3. VNIC and its type is needed in the test setup to once again accommodate performance penalty when DUT (VSWITCH) is created.

In summary, calibration should be done in such an environment that all possible factors which may negatively impact test results should be taken into consideration.

#### 4. Key Performance Indicators

We listed numbers of key performance indicators for virtual network below:

- a) Throughput under various frame sizes: forwarding performance under various frame sizes is a key performance indicator of interest.
- b) DUT consumption of CPU: when adding one or more VM(s), DUT (VSWITCH) will consume more CPU. Vendors can allocate appropriate CPU to reach the line rate performance.

c) DUT consumption of MEM: when adding one or more VM(s), DUT (VSWITCH) will consume more memory. Vendors can allocate appropriate MEM to reach the line rate performance.

d) Latency readings: Some applications are highly sensitive on latency. It's important to get the latency reading with respective to various conditions.

Other indicators such as VxLAN maximum supported by the virtual switch and so on can be added in the scene when VxLAN is needed.

## 5. Test Setup

The test setup is classified into two traffic models: Model A and Model B.

In traffic model A: A physical tester connects to the server which bears the DUT (VSWITCH) and Virtual tester to verify the benchmark of server.

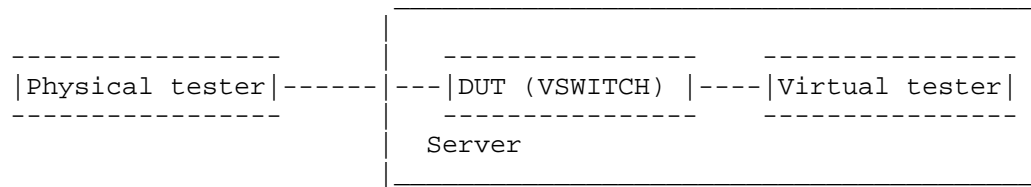


Figure 3: test model A

In traffic model B: Two virtual testers are used to verify the benchmark. In this model, two testers are installed in one server.

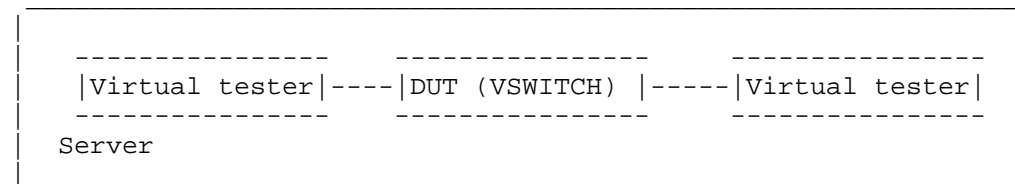


Figure 4: test model B

In our test, the test bed is constituted by physical servers of the Dell with a pair of 10GE NIC and physical tester. Virtual tester which occupies 2 vCPU and 8G MEM and DUT (VSWITCH) are installed in the server. 10GE switch and 1GE switch are used for test traffic and management respectively.

This test setup is also available in the VxLAN measurement.

## 6. Benchmarking Tests

### 6.1. Throughput

Unlike traditional test cases where the DUT and the tester are separated, virtual network test has been brought in unparalleled challenges. In virtual network test, the virtual tester and the DUT (VSWITCH) are in one server which means they are physically converged, so the test and DUT (VSWITCH) are sharing the same CPU and MEM resources of one server. Theoretically, the virtual tester's operation may have influence on the DUT (VSWITCH)'s performance. However, for the specialty of virtualization, this method is the only way to test the performance of a virtual DUT.

Under the background of existing technology, when we test the virtual switch's throughput, the concept of traditional physical switch CANNOT be applicable. The traditional throughput indicates the switches' largest forwarding capability, for certain bytes selected and under zero-packet-lose conditions. But in virtual environments, virtual variations on virtual network will be much greater than that of dedicated physical devices. As the DUT and the tester cannot be separated, it proves that the DUT (VSWITCH) realize such network performances under certain circumstances.

Therefore, we change the bytes in virtual environment to test the maximum value which we think of the indicator of throughput. It's conceivable that the throughput should be tested on both the test model A and B. The tested throughput has certain referential meanings to value the performance of the virtual DUT.

#### 6.1.1. Objectives

The objective of the test is to determine the throughput of the DUT (VSWITCH), which the DUT can support.

#### 6.1.2. Configuration parameters

Network parameters should be defined as follows:

- a) the number of virtual tester (VMs)
- b) the number of vNIC of virtual tester
- c) the CPU type of the server
- d) vCPU allocated for virtual tester (VMs)

e) memory allocated for virtual tester (VMs)

f) the number and rate of server NIC

#### 6.1.3. Test parameters

a) test repeated times

b) test frame length

#### 6.1.4. Test process

1. Configure the VM tester to offer traffic to the V-Switch.
2. Increase the number of vCPU in the tester until the traffic has no packet loss.
3. Record the max throughput on VSwitch.
4. Change the frame length and repeat from step1 to step4.

#### 6.1.5. Test result format

Byte	Throughput (Gbps)
64	
128	
256	
512	
1024	
1518	

Figure 5: test result format

#### 6.2. Frame loss rate

Frame loss rate is also an important indicator in evaluating the performance of virtual switch. As is defined in RFC 1242, percentage of frames that should have been forwarded which actually fails to be forwarded due to lack of resources needs to be tested. Both model A



and model B are tested. Frame loss rate is an important indicator in evaluating the performance of virtual switches.

#### 6.2.1. Objectives

The objective of the test is to determine the frame loss rate under different data rates and frame sizes..

#### 6.2.2. Configuration parameters

Network parameters should be defined as follows:

- a) the number of virtual tester (VMs)
- b) the number of vNIC of virtual tester
- c) the CPU type of the server
- d) vCPU allocated for virtual tester (VMs)
- e) memory allocated for virtual tester (VMs)
- f) the number and rate of server NIC

#### 6.2.3. Test parameters

- a) test repeated times
- b) test frame length
- c) test frame rate

#### 6.2.4. Test process

1. Configure the VM tester to offer traffic to the V-Switch with the input frame changing from the maximum rate to the rate with no frame loss at reducing 10% intervals according to RFC 2544.
2. Record the input frame count and output count on VSwitch.
3. Calculate the frame loss percentage under different frame rate.
4. Change the frame length and repeat from step1 to step4.

## 6.2.5. Test result format

Byte	Maximum rate (Gbps)	90% Maximum rate (Gbps)	80% Maximum rate (Gbps)	...	rate with no loss (Gbps)
64					
128					
256					
512					
1024					
1518					

Figure 6: test result format

## 6.3. CPU consumption

The objective of the test is to determine the CPU load of DUT(VSWITCH). The operation of DUT (VSWITCH) can increase the CPU load of host server. Different V-Switches have different CPU occupation. This can be an important indicator in benchmarking the virtual network performance.

## 6.3.1. Objectives

The objective of this test is to verify the CPU consumption caused by the DUT (VSWITCH).

## 6.3.2. Configuration parameters

Network parameters should be defined as follows:

- a) the number of virtual tester (VMs)
- b) the number of vNIC of virtual tester
- c) the CPU type of the server
- d) vCPU allocated for virtual tester (VMs)
- e) memory allocated for virtual tester (VMs)

f) the number and rate of server NIC

#### 6.3.3. Test parameters

- a) test repeated times
- b) test frame length
- c) traffic rate

#### 6.3.4. Test process

1. Configure the VM tester to offer traffic to the V-Switch with certain traffic rate. The traffic rate could be different ratio of NIC's speed.
2. Record vSwitch's CPU usage on the host OS if no packets loss happens.
3. Change the traffic rate and repeat from step1 to step2.
4. Change the frame length and repeat from step1 to step3.

#### 6.3.5. Test result format

Byte	Traffic Rate(Gbps)	CPU usage of vSwitch
64	50% of NIC speed	
	75%	
	90%	
128	50% of NIC speed	
	75%	
	90%	
~ ~ ~ ~		
1500	50% of NIC speed	
	75%	
	90%	

Figure 7: test result format

#### 6.4. MEM consumption

The objective of the test is to determine the Memory load of DUT(VSWITCH). The operation of DUT (VSWITCH) can increase the Memory load of host server. Different V-Switches have different memory occupation. This can be an important indicator in benchmarking the virtual network performance.

##### 6.4.1. Objectives

The objective of this test is to verify the memory consumption by the DUT (VSWITCH) on the Host server.

##### 6.4.2. Configuration parameters

Network parameters should be defined as follows:

- a) the number of virtual tester (VMs)
- b) the number of vNIC of virtual tester
- c) the CPU type of the server

- d) vCPU allocated for virtual tester (VMs)
- e) memory allocated for virtual tester (VMs)
- f) the number and rate of server NIC

#### 6.4.3. Test parameters

- a) test repeated times
- b) test frame length

#### 6.4.4. Test process

1. Configure the VM tester to offer traffic to the V-Switch with certain traffic rate. The traffic rate could be different ratio of NIC's speed.
2. Record vSwitch's MEM usage on the host OS if no packets loss happens.
3. Change the traffic rate and repeat from step1 to step2.
4. Change the frame length and repeat from step1 to step3.

#### 6.4.5. Test result format

Byte	Traffic Rate(Gbps)	MEM usage of vSwitch
64	50% of NIC speed	
	75%	
	90%	
128	50% of NIC speed	
	75%	
	90%	
~	~	~
1500	50% of NIC speed	
	75%	
	90%	

Figure 8: test result format

#### 6.5. Latency

Physical tester's time refers from its own clock or other time source, such as GPS, which can achieve the accuracy of 10ns. While in virtual network circumstances, the virtual tester gets its reference time from the clock of Linux systems. However, due to current methods, the clock of different servers or VMs can't synchronize accuracy. Although VMs of some higher versions of CentOS or Fedora can achieve the accuracy of 1ms, we can get better results if the network can provide better NTP connections.

Instead of finding a better synchronization of clock to improve the accuracy of the test, we consider to use an echo server in order to forward the traffic back to the virtual switch.

We use the traffic model A as the latency test model by substituting the virtual tester with the echo server, which is used to echo the traffic. Thus the one-way delay equals to half of the round-trip time.

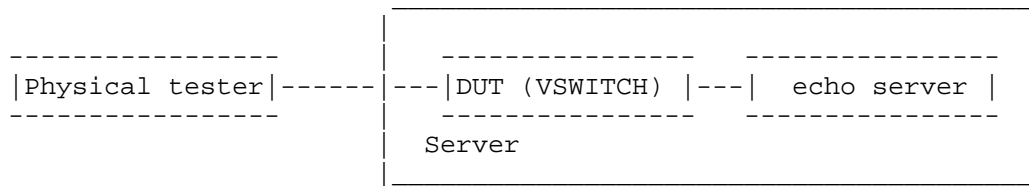


Figure 9: time delay test model

#### 6.5.1. Objectives

The objective of this test is to verify the DUT (VSWITCH) for latency of the flow. This can be an important indicator in benchmarking the virtual network performance.

#### 6.5.2. Configuration parameters

Network parameters should be defined as follows:

- a) the number of virtual tester (VMs)
- b) the number of vNIC of virtual tester
- c) the CPU type of the server
- d) vCPU allocated for virtual tester (VMs)
- e) memory allocated for virtual tester (VMs)
- f) the number and rate of server NIC

#### 6.5.3. Test parameters

- a) test repeated times
- b) test frame length

#### 6.5.4. Test process

1. Configure the physical tester to offer traffic to the V-Switch with the traffic value of throughput tested in 6.1.
2. Under the same throughput, record the time of transmitting the traffic and receiving the traffic by the physical tester with and without the DUT.

3. Calculate the time difference value between receiving and transmitting the traffic..
4. Calculate the time delay with time difference value with and without the DUT.
5. Change the frame length and repeat from step1 to step4.

#### 6.5.5. Test result format

Byte	Latency
64	
128	
256	
512	
1024	
1518	

Figure 10: test result format

#### 7. Security Considerations

None.

#### 8. IANA Considerations

None.

#### 9. Normative References

- [RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <<http://www.rfc-editor.org/info/rfc1242>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.



- [RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, DOI 10.17487/RFC2234, November 1997, <<http://www.rfc-editor.org/info/rfc2234>>.
- [RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <<http://www.rfc-editor.org/info/rfc2544>>.

Authors' Addresses

Lu Huang (editor)  
China Mobile  
32 Xuanwumen West Ave, Xicheng District  
Beijing 100053  
China

Email: [hlisname@yahoo.com](mailto:hlisname@yahoo.com)

Rong Gu (editor)  
China Mobile  
32 Xuanwumen West Ave, Xicheng District  
Beijing 100053  
China

Email: [gurong@chinamobile.com](mailto:gurong@chinamobile.com)

Bob Mandeville  
Iometrix  
3600 Fillmore Street Suite 409  
San Francisco, CA 94123  
USA

Email: [bob@iometrix.com](mailto:bob@iometrix.com)

Brooks Hickman  
Spirent Communications  
1325 Borregas Ave  
Sunnyvale, CA 94089  
USA

Email: [Brooks.Hickman@spirent.com](mailto:Brooks.Hickman@spirent.com)

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: January 4, 2018

T. Kim  
B. Koo  
J. Park  
E. Paik  
KT  
July 03, 2017

Considerations for Benchmarking Service Function Chain  
draft-kim-bmwg-sfc-benchmark-00

Abstract

Service Function Chain(SFC) is a ordered set of service functions. Packets flow restrictively at the service functions according to the order. To enable a network service, operator composes the service function chain logically. Though SFC is efficient where network/ service requirements are dynamically changing, the reliability of SFC should be guaranteed. This memo describes the considerations for benchmarking SFC reliability.

Requirements Language

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

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 <http://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 4, 2018.

## Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	2
2. Scope . . . . .	3
3. Considerations for Benchmarking SFC Reliability . . . . .	3
3.1. Configuration Parameters for Benchmarking Test . . . . .	3
3.2. Testing Parameter Benchmarking Test . . . . .	4
4. Security Considerations . . . . .	4
5. IANA Considerations . . . . .	5
6. Normative References . . . . .	5
Authors' Addresses . . . . .	5

## 1. Introduction

As Service Function Chain(SFC) is the ordered set of service functions. It is logically defined on demand of a service. To enable the service, SDN controller set flow rules at each physical/virtual switch which belongs to the SFC. SFC is efficient where the network/service requirements are keep changing dynamically. The number of physical/virtual switches which will accept the flow rules is differ from the size of the domain or service.

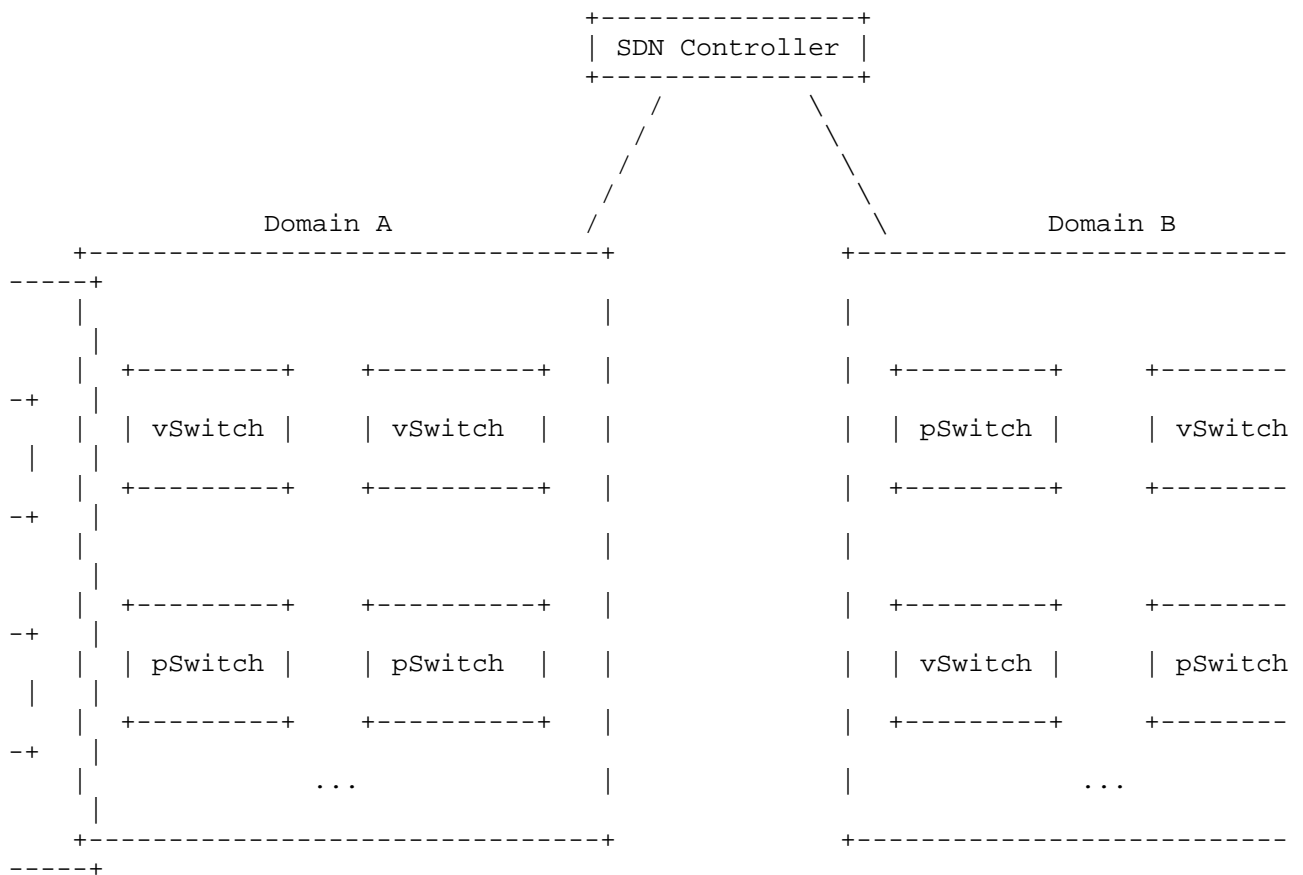
As an operator perspective, at the stage of SFC creation, modification, and deletion, the reliability of SFC should always be guaranteed. To apply the change of the SFC, SDN controller will set flow rules at some switches and delete flow rules at other switches. For certain reasons such as the heavy traffic on the target switches which should accept new rules or the link failure between the target switches and the SDN controller, the new SFC may not be applied properly.

This draft memo describes considerations for benchmarking Service Function Chain reliability.

## 2. Scope

At the time of writing this memo, SFC standardization is now in progress. But operators and vendors are implementing SFC their own way. This memo does not target NSH enabled architecture and target general operation circumstances. The scope of SFC reliability benchmark is when the initial SFC is already provisioned and the traffic also flows over the certain SFCs, and SFC needs to be updated. Also, SFC is made over multi-domain network, which covers the whole country.

This figure is an example of the network.



## 3. Considerations for Benchmarking SFC Reliability

This section defines and lists considerations which must be addressed to benchmark the reliability of SFC

### 3.1. Configuration Parameters for Benchmarking Test

This section lists the parameters affecting the SFC reliability. To apply new SFC, SDN controller set rules to the target switches. Depending on the status of the swithes and the network, the new SFC can be applied right as intended, or not. The right operation of SFC as intended includes the right time of the operation activates.



- o Types of Switches : Virtual switch or Physical switch
- o The number of switches in target SFC domain
  - \* Depending on the composition of the target SFC, the number of switches which need to update their flow tables is different.
- o The Usage of Flow table of the target switch
  - \* When the new SFC rule needs to setup, if the flow table entries are not enough and have to stored elsewhere, not TCAM, the usage of flow table can affect the reliability of SFC.
    - + TCAM Usage
    - + Flow table Entries
- o The physical distances between the Controller and Switch
  - \* As the network grows broad, the delay is same as propagation delay. And this make SFC Activation time different.
- o The traffic loads on the target switch
  - \* The limitaion of the CPU, when the target switch needs to process large amount of the traffic, the new SFC rules setup cannot be done in intended time.

### 3.2. Testing Parameter Benchmarking Test

This section describes the testing parameter for Benchmark SFC Reliability. In terms of operation, the reliability of SFC is "operate the SFC in right time and at right path."

#### Rule Activation Time

- o The time interval from the new flow rule setup requests to the time when packets start to flow following the new matched rule.

TBD

## 4. Security Considerations

TBD.

## 5. IANA Considerations

No IANA Action is requested at this time.

## 6. 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, <<http://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, <<http://www.rfc-editor.org/info/rfc2544>>.
- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <<http://www.rfc-editor.org/info/rfc7665>>.

## Authors' Addresses

Taekhee Kim  
KT  
Infra R&D Lab. KT  
17 Woomyeon-dong, Seocho-gu  
Seoul 137-792  
Korea

Phone: +82-2-526-6688  
Fax: +82-2-526-5200  
Email: [taekhee.kim@kt.com](mailto:taekhee.kim@kt.com)

Bummo Koo  
KT  
Infra R&D Lab. KT  
17 Woomyeon-dong, Seocho-gu  
Seoul 137-792  
Korea

Phone: +82-2-526-6688  
Fax: +82-2-526-5200  
Email: [bm.koo@kt.com](mailto:bm.koo@kt.com)

Jisu Park  
KT  
Infra R&D Lab. KT  
17 Woomyeon-dong, Seocho-gu  
Seoul 137-792  
Korea

Phone: +82-2-526-6688  
Fax: +82-2-526-5200  
Email: jisupark@kt.com

EunKyoung Paik  
KT  
Infra R&D Lab. KT  
17 Woomyeon-dong, Seocho-gu  
Seoul 137-792  
Korea

Phone: +82-2-526-5233  
Fax: +82-2-526-5200  
Email: eun.paik@kt.com  
URI: <http://mmlab.snu.ac.kr/~eun/>



Internet Engineering Task Force  
Internet-Draft  
Intended status: Standards Track  
Expires: December 26, 2018

S. Jacob, Ed.  
K. Tiruveedhula  
Juniper Networks  
June 24, 2018

Benchmarking Methodology for EVPN and PBB-EVPN  
draft-kishjac-bmwg-evpntest-10

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 26, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	3
1.1. Requirements Language . . . . .	3
1.2. Terminologies . . . . .	3
2. Test Topology . . . . .	4
3. Test Cases . . . . .	6
3.1. How long it takes to learn local mac address in EVPN . . . . .	6
3.2. How long it takes to learn local mac address in PBB EVPN . . . . .	6
3.3. How long it takes to learn the remote macs . . . . .	7
3.4. PBB-EVPN How long it takes to learn the mac from remote peer . . . . .	8
3.5. How long it takes to flush the local macs due to CE link flap and measure the relearning rate of MACs . . . . .	8
3.6. PBB-EVPN how long it takes to flush the local macs and measure the relearning rate of macs during PE-CE link flap . . . . .	9
3.7. How long it takes to flush the remote macs, due to remote link failure. . . . .	10
3.8. PBB-EVPN How long it takes to flush the remote macs due to remote link failure . . . . .	10
3.9. To measure the MAC aging time. . . . .	11
3.10. PBB-EVPN To measure the MAC aging time. . . . .	12
3.11. How long it takes to age out the remote macs . . . . .	12
3.12. PBB-EVPN How long it takes to age out the remote macs. . . . .	13
3.13. How long it takes to learn both local and remote macs. . . . .	14
3.14. PBB-EVPN How long it takes to learn both local and remote macs . . . . .	14
4. High Availability . . . . .	15
4.1. To Record the whether there is traffic loss due to routing engine failover for redundancy test. . . . .	15
4.2. PBB-EVPN To Record the whether there is traffic loss due to routing engine failover for redundancy test . . . . .	16
5. ARP/ND Scale . . . . .	16
5.1. To find ARP/ND scale . . . . .	16
6. Scale . . . . .	17
6.1. To Measure the scale limit of DUT with trigger (Scale without traffic) . . . . .	17
6.2. PBB-EVPN To measure the scale limit with trigger. . . . .	17
6.3. To measure the convergence time of DUT with scale and traffic. . . . .	18
6.4. .PBB-EVPN To measure the convergence time of DUT with scale and traffic. . . . .	19
7. SOAK Test . . . . .	19
7.1. To Measure the stability of the DUT with scale and traffic. . . . .	19
7.2. PBB-EVPN to measure the stability of DUT with scale and traffic. . . . .	20

8. Acknowledgements . . . . .	21
9. IANA Considerations . . . . .	21
10. Security Considerations . . . . .	21
11. References . . . . .	21
11.1. Normative References . . . . .	21
11.2. Informative References . . . . .	21
Appendix A. Appendix . . . . .	22
Authors' Addresses . . . . .	22

## 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 EVPN to provide a new/combined solution. This draft defines methodologies that can be used to benchmark both RFC 7432 and RFC 7623 solutions. Further, this draft provides methodologies for benchmarking the performance of EVPN data and control planes, MAC learning, MAC flushing, MAC ageing, convergence, high availability, and scale.

### 1.1. Requirements Language

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

### 1.2. Terminologies

MHPE Multi homed Provide Edge router.

RR Route Reflector.

P Provider Router.

CE Customer Router/Devices/Switch.

MHPE2 Multi homed Provider Edge router 2.

MHPE1 Multi homed Provider Edge router 1.

SHPE3 Single homed Provider Edge Router 3.

AA EVPN Terminologies AA All-Active.

SA EVPN Terminologies SA Single-Active.

RT Router Tester.

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

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

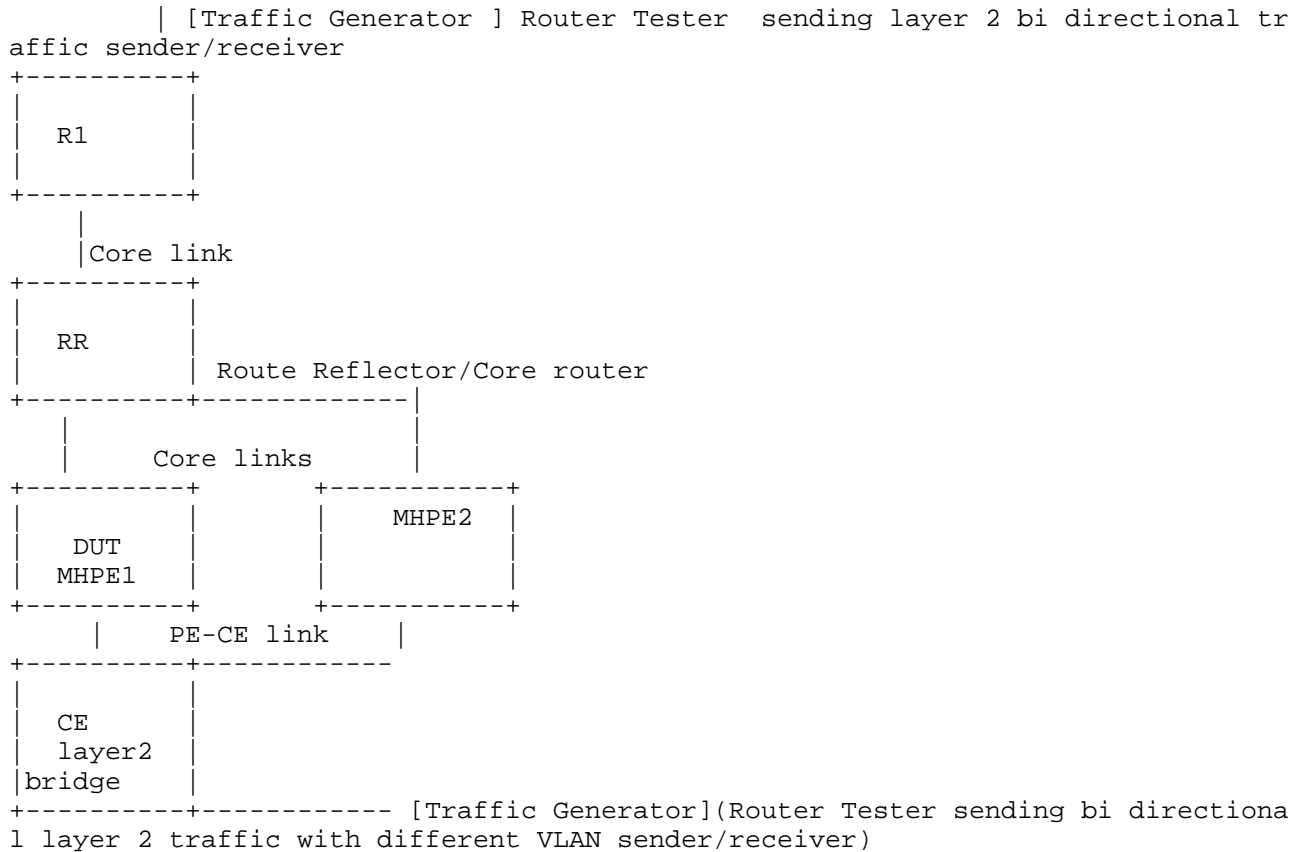
DF Designated Forwarder.

ESI Ethernet Segment Identifier.

## 2. Test Topology

EVPN/PBB-EVPN Services running on R1, MHPE1 and MHPE2 in Single Active Mode:

## Topology Diagram



## Topology Diagram

Figure 1

There are five routers in the topology. R1, RR/P, MHPE1 and MHPE2 emulating a service provider network. CE is a customer device connected to MHPE1 and MHPE2, it is configured with bridge domains in different vlans. The Router tester is connected to R1 and CE which send layer 2 traffic for configured vlans. MHPE1, MHPE2, RR/P, R1 run MPLS. The EVPN/PBB-EVPN services are running on MHPE1, MHPE2 and R1. The MHPE1 acts DUT. The RT will act as sender and receiver. The measurement will be taken in DUT.

### 3. Test Cases

The following tests are conducted to measure the time taken to learn the "X" number of MAC's locally in EVI . The data plane learning of MAC will happen locally from connected interface. The control plane learning of MAC is through BGP advertisements from the remote PE(SHPE3). The control plane learning of "X" MAC. The data plane MAC learning can be measured using the parameters defined in RFC 2889 section 5.8.

#### 3.1. How long it takes to learn local mac address in EVPN

Objective:

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

Procedure:

Configure EVPN EVI in R1, MHPE2, DUT. All 4 routers except CE are running MPLS, BGP, RR is acting as route reflector to R1,MHPE2 and DUT. For MH PE ESI must be configured per IFD/Interface. Using RT (traffic generator)to send the traffic to the CE. The traffic is unidirectional. Since CE is working in bridge mode, frames will be send to ingress sub interface of DUT. The BGP must be established in R1, MHPE1(DUT), RR, MHPE2. Send "X" unicast frames from CE to MHPE1(DUT) working in SA mode with "X" different source and destination address from RT. The DUT must learn these "X" macs in data plane.

Measurement :

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

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

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

#### 3.2. How long it takes to learn local mac address in PBB EVPN

Objective:

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

#### Procedure:

Configure PBB-EVPN EVI in R1,MHPE2,DUT. All 4 routers except CE are running MPLS, BGP, RR is acting as route reflector to R1,MHPE2 and DUT. Once the BGP comes up. Record the DUT PBB-EVPN table. For MH PE ESI must be configured per IFD/Interface. From RT (traffic generator) send the traffic to the DUT. The BGP must be established in R1,MHPE1 (DUT), RR, MHPE2. The traffic is unidirectional. Since CE is working in bridge mode, frames will be send to ingress sub interface of DUT.

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

#### Measurement :

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

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

### 3.3. How long it takes to learn the remote macs

#### Objective:

To Record the time taken to learn the remote macs.

#### Procedure:

Configure EVPN EVI in R1,MHPE2,DUT. All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT. Record the DUT EVPN table. For MH PE ESI must be configured per IFD/Interface. Using RT(traffic generator) send the traffic to R1.The traffic is uni directional. There wont be any traffic flow from CE to DUT during this test. The BGP must be in established state. The MACS learned in R1 will be advertised to DUT by BGP.

Send X frames with X different SA and DA to R1 from RT. R1 will advertise these locally learned macs to MHPE1 and MHPE2 via control plane.Measure the time taken to learn these X MACs from remote peer in DUT EVPN MAC address table.The DUT and MHPE2 are running SA mode.

**Measurement :**

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

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

**3.4. PBB-EVPN How long it takes to learn the mac from remote peer****Objective:**

To Record the time taken to learn the remote macs.

**Procedure:**

Configure PBB-EVPN EVI in R1, MHPE2, DUT. All 4 routers except CE are running MPLS, BGP, RR is acting as route reflector to R1, MHPE2 and DUT. Record the DUT PBB-EVPN table. For MHPE ESI must be configured per IFD/Interface. Using RT (traffic generator) send the traffic to R1. The traffic is uni directional. There won't be any traffic flow from CE to DUT during this test. The BGP must be in established state.

Send X frames with X different SA and DA to R1 from RT. These macs will be flooded to MHPE1 and MHPE2 by R1. The DUT and MHPE2 are running SA mode.

**Measurement :**

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

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

**3.5. How long it takes to flush the local macs due to CE link flap and measure the relearning rate of MACs****Objective:**

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

**Procedure:**



Configure EVPN EVI in R1,MHPE2,DUT. All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT. Once the BGP is established. Record the DUT EVPN table. For MH PE ESI must be configured per IFD/Interface. Using RT(traffic generator) send the traffic. In this scenario traffic will be only send from CE side.

Send X frames with X different SA and DA to DUT from CE using traffic generator. Wait till the MHPE1 learns all X MAC address. Then fail the MHPE1 CE link and measure the time taken to flush these X MACs from the EVPN MAC table. Bring up the link which was made Down(the link between MHPE1 and CE).Measure time taken to relearn it. The DUT and MHPE2 are running SA mode.

Measurement :

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

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

### 3.6. PBB-EVPN how long it takes to flush the local macs and measure the relearning rate of macs during PE-CE link flap

Objective:

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

Procedure:

Configure PBB-EVPN EVI in R1,MHPE2,DUT. All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT. Once the BGP is established. Record the DUT PBB-EVPN table. For MH PE ESI must be configured per IFD/Interface. Using RT(traffic generator) send the traffic to the CE. The traffic is uni directional.

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

Measurement :

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

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

### 3.7. How long it takes to flush the remote macs, due to remote link failure.

Objective:

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

Procedure:

Configure EVPN EVI in R1,MHPE2,DUT.All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT. Once the BGP is established Record the DUT EVPN table. For MHPE ESI must be configured per IFD/Interface. Using RT(traffic generator) send the traffic to R1.There wont be any traffic flowing to CE from RT.

Send X frames with X different SA and DA to DUT from R1 using traffic generator. Bring down the link between R1 and traffic generator. Then measure the time taken to flush the DUT EVPN MAC table. The DUT and MHPE2 are running SA mode.

Measurement :

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

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

### 3.8. PBB-EVPN How long it takes to flush the remote macs due to remote link failure

Objective:

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

Procedure:

Configure PBB-EVPN EVI in R1,MHPE2,DUT.All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT. Once the BGP is established Record the DUT PBB-EVPN MAC table. For MHPE ESI must be configured per IFD/Interface. Using RT(traffic generator) send the traffic to R1.In this scenario traffic will be flowing only from R1.

Send X frames with X different SA and DA to DUT from R1 using traffic generator. Bring down the link between R1 and traffic generator. Then measure the time taken to flush the DUT PBB-EVPN MAC address table. The remote MACs will be learned by Data plane, but the B-MAC will be learned by control plane. The DUT and MHPE2 are running SA mode.

Measurement :

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

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

3.9. To measure the MAC aging time.

Objective:

To measure the mac aging time.

Procedure:

Configure EVPN EVI in R1,MHPE2,DUT. All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT .Once the BGP is established. Record the DUT EVPN table. For MHPE ESI must be configured per IFD/Interface. Using RT(traffic generator), send the traffic to the DUT. The traffic will be flowing from CE to DUT. There wont be any traffic from R1.

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

Measurement :

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

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

### 3.10. PBB-EVPN To measure the MAC aging time.

Objective:

To measure the mac aging time.

Procedure:

Configure PBB-EVPN EVI in R1,MHPE2,DUT.All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT.Once the BGP is established. Record the DUT PBB-EVPN MAC table. For MH PE ESI must be configured per IFD/Interface. Using RT(traffic generator) send the traffic to the DUT. The traffic is unidirectional flowing from CE to DUT.

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

Measurement :

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

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

### 3.11. How long it takes to age out the remote macs

Objective:

To measure the remote mac aging time.

Procedure:

Configure EVPN EVI in R1,MHPE2,DUT.All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT.

Once the BGP is established. Record the DUT EVPN table. For MH PE ESI must be configured per IFD/Interface. Using RT(traffic generator) send the traffic to R1. There won't be any traffic from CE.

Send X frames with X different SA and DA to DUT from R1 using traffic generator. Stop the traffic at remote PE R1. Due to MAC aging R1 will withdraw its routes from DUT and MHPE2. Measure the time taken to remove these MACs from DUT EVPN MAC table. DUT and MHPE2 are running in SA mode

Measurement :

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

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

### 3.12. PBB-EVPN How long it takes to age out the remote macs.

Objective:

To measure the remote mac aging time.

Procedure:

Configure PBB-EVPN EVI in R1, MHPE2, DUT. All 4 routers except CE are running MPLS, BGP, RR is acting as route reflector to R1, MHPE2 and DUT. Once the BGP is established. Record the DUT MAC table. For MH PE ESI must be configured per IFD/Interface. Using RT(traffic Generator) send the traffic to R1. There is no traffic from CE side.

Send X frames with X different SA and DA to DUT from R1 using traffic generator. Stop the traffic at remote PE(R1). Measure the time taken to remove these remote MACs from DUT PBB-EVPN MAC table. The DUT and MHPE2 are running in SA mode.

Measurement :

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

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

### 3.13. How long it takes to learn both local and remote macs.

#### Objective:

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

#### Procedure:

Configure EVPN EVI in R1,MHPE2,DUT.All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT. Once the BGP is established. Record the DUT EVPN table. For MH PE ESI must be configured per IFD/Interface. Using RT(traffic generator) send the traffic to the routers. The traffic is bi directional.

Send X frames with X different SA and DA to DUT from R1 using traffic generator. Send X frames with different SA and DA from traffic generator connected to CE. The SA and DA of flows must be complimentary to have unicast flows. Measure the time taken by the DUT to learn 2X in EVPN MAC. DUT and MHPE2 are running in SA mode.

#### Measurement :

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

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

### 3.14. PBB-EVPN How long it takes to learn both local and remote macs

#### Objective:

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

#### Procedure:

Configure PBB-EVPN EVI in R1,MHPE2,DUT.All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT. Once the BGP is established. Record the DUT EVPN table.For MH PE ESI must be configured per IFD/Interface. Using RT(traffic generator) send the traffic to the routers.

Send X frames with X different SA and DA to DUT from R1 using traffic generator. Send X frames with different SA and DA from traffic generator connected to CE. The SA and DA of flows must be

complimentary to have unicast flows. Measure the time taken by the DUT to learn 2X in MAC table. DUT and MHPE2 are running in SA mode.

Measurement :

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

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

#### 4. High Availability

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

Objective:

To record traffic loss during routing engine failover.

Procedure:

Configure EVPN EVI in R1,MHPE2,DUT.All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT. Once the BGP is established. Record the DUT EVPN table. For MH PE ESI must be configured per IFD/Interface. Using RT(traffic generator) Send bi directional to the routers.

Send X frames from CE to DUT from traffic generator with X different SA and DA. Send X frames from traffic generator to R1 with X different SA and DA so that 2X MAC address will be learned in DUT. There is a bi directional traffic flow with X pps in each direction. Then do a routing engine fail-over.

Measurement :

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

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

#### 4.2. PBB-EVPN To Record the whether there is traffic loss due to routing engine failover for redundancy test

##### Objective:

To record traffic loss during routing engine failover.

##### Procedure:

Configure PBB-EVPN EVI in R1,MHPE2,DUT. All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT. Once the BGP is established. Record the DUT PBB-EVPN table. For MH PE ESI must be configured per IFD/Interface. Using RT(traffic generator) send the traffic to the routers.

Send X frames to DUT with X different SA and DA from CE using the traffic generator. Send X frames from traffic generator to R1 with X different SA and DA so that 2X MAC address will be Learned in DUT. There is a bi directional traffic flow with X pps in each direction. Then do a routing engine fail-over.

##### Measurement :

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

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

#### 5. ARP/ND Scale

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

##### 5.1. To find ARP/ND scale

##### Objective:

To Record the ARP/ND scale of the DUT.

##### Procedure:

Configure EPN EVI in R1,MHPE2,DUT.All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT. Once the BGP is established. Record the DUT EVPN table. For MH PE ESI must be configured per IFD/Interface. Using RT(traffic generator) send arp/ICMPv6 request to the DUT which has gateway configured.



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

Measurement :

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

## 6. Scale

This is to measure the performance of DUT in scaling to "X" EVPN instances. The measured parameters are CPU usage, memory leak, crashes.

### 6.1. To Measure the scale limit of DUT with trigger (Scale without traffic)

Objective:

To measure the scale limit of DUT for EVPN.

Procedure:

Configure EVPN EVI in R1,MHPE2,DUT.All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT. Once the BGP is established. Record the DUT EVPN table. For MHPE,DUT ESI must be configured per IFD/Interface.

The DUT,MHPE2 and R1 are scaled to "N" EVI.Clear BGP neighbors of the DUT. Once adjacency is established in the DUT. Measure the routes received from MHPE2 and R1 for "N" EVI in the DUT.

Measurement :

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

### 6.2. PBB-EVPN To measure the scale limit with trigger.

Objective:

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

Procedure:

Configure "N" PBB-EVPN instances in R1,MHPE2,DUT.All 4 routers except CE are running MPLS,BGP,RR is acting route reflector to R1,MHPE2 and DUT. Once BGP is established. Record the DUT PBB-EVPN table.For MHPE ESI must be configured on IFD/Interface.

The DUT,MHPE2 and R1 are scaled to "N" PBB-EVPN instances. Clear BGP neighbors in the DUT Once adjacency is established in DUT, check routes received from R1 and MHPE2.

Measurement :

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

### 6.3. To measure the convergence time of DUT with scale and traffic.

Objective:

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

Procedure:

Configure EVPN EVI in R1,MHPE2,DUT.All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT. Once the BGP is established. Record the DUT EVPN table. For MH PE ESI must be configured per IFD/Interface. Using RT(traffic generator)send the traffic to the routers.

Scale N EVIs in DUT,R1 and MHPE2.Send F frames to DUT from CE using traffic generator with X different SA and DA for N EVI's. Send F frames from traffic generator to R1 with X different SA and DA. There will be 2X number of MAC address will be learned in DUT EVPN MAC table. There is a bi directional traffic flow with F pps in each direction. Then clear the BGP neighbors in the DUT. Once the adjacency is restored in DUT. Measure the time taken to learn 2X MAC address in DUT MAC table.

Measurement :

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

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

#### 6.4. PBB-EVPN To measure the convergence time of DUT with scale and traffic.

##### Objective:

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

##### Procedure:

Configure PBB-EVPN instances in R1,MHPE2,DUT.All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT. Once BGP is established. Record the DUT EVPN table. For MH PE ESI must be configured per IFD/Interface. Using RT(traffic generator) send the traffic to the routers.

Scale N PBB-EVI's in DUT,R1 and MHPE2.Send F frames to DUT from CE using traffic generator with X different SA and DA for N EVI's. Send F frames from traffic generator to R1 with X different SA and DA. There will be 2X number of MAC address will be learned in DUT PBB-EVPN MAC table. There is a bi directional traffic flow with F pps in each direction. Then clear the BGP neighbors in the DUT. Once the adjacency is restored in DUT. Measure the time taken to learn 2X MAC address in DUT PBB-MAC table.

##### Measurement :

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

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

#### 7. SOAK Test

This is measuring the performance of DUT running with scaled configuration with traffic over a peroid of time "T'". In each interval "t1" the parameters measured are CPU usage, memory usage, crashes.

##### 7.1. To Measure the stability of the DUT with scale and traffic.

##### Objective:

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

#### Procedure:

Configure EVPN EVI in R1,MHPE2,DUT.All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT. Once the BGP is established. Record the DUT EVPN table. For MH PE ESI must be configured per IFD/Interface. Using RT(traffic generator) send the traffic to the routers.

Scale N EVI's in DUT,R1 and MHPE2.Send F frames to DUT from CE using traffic generator with different X SA and DA for N EVI's. Send F frames from traffic generator to R1 with X different SA and DA. There will be 2X number of MAC address will be learned in DUT EVPN MAC table. There is a bi directional traffic flow with F pps in each direction. The DUT must run with traffic for 24 hours, every hour check for memory leak, crash.

#### Measurement :

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

### 7.2. PBB-EVPN to measure the stability of DUT with scale and traffic.

#### Objective:

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

#### Procedure:

Configure N PBB-EVPN instances in R1, MHPE2, DUT. All 4 routers except CE are running MPLS,BGP,RR is acting as route reflector to R1,MHPE2 and DUT. Once the BGP comes up Record the DUT EVPN table. for MH PE ESI must be configured per IFD/Interface. Using RT(traffic generator)send the traffic to the routers.

Scale N PBB-EVI's in DUT,R1 and MHPE2.Send F frames to DUT from CE using traffic generator with X different SA and DA for N EVI's. Send F frames from traffic generator to R1 with X different SA and DA. There will be 2X number of MAC address will be learned in DUT PBB-EVPN MAC table. There is a bi directional traffic flow with F pps in Each direction. The DUT must run with traffic for 24 hours, every hour check the memory leak, crashes.

#### Measurement :

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

## 8. Acknowledgements

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

## 9. IANA Considerations

This memo includes no request to IANA.

## 10. Security Considerations

There is no additional consideration from RFC 6192.

## 11. References

### 11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <<https://www.rfc-editor.org/info/rfc2544>>.
- [RFC2899] Ginoza, S., "Request for Comments Summary RFC Numbers 2800-2899", RFC 2899, DOI 10.17487/RFC2899, May 2001, <<https://www.rfc-editor.org/info/rfc2899>>.

### 11.2. Informative References

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

Appendix A. Appendix

Authors' Addresses

Sudhin Jacob (editor)  
Juniper Networks  
Bangalore  
India

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

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

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

Network Working Group  
Internet-Draft  
Updates: 2544 (if approved)  
Intended status: Informational  
Expires: January 2, 2019

A. Morton  
AT&T Labs  
July 1, 2018

Updates for the Back-to-back Frame Benchmark in RFC 2544  
draft-morton-bmwg-b2b-frame-02

Abstract

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

This memo updates Section 26.4 of RFC 2544.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 2, 2019.

## Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

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

## 1. Introduction

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



Recent testing experience with the Back-to-back Frame test and Benchmark in Section 26.4 of [RFC2544] indicates that an update is warranted [OPNFV-2017] [VSPERF-b2b]. This memo describes the rationale and provides the updated method.

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

## 2. Scope and Goals

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

The goal is to provide more efficient test procedures where possible, and to expand reporting with additional interpretation of the results.

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

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

## 3. Motivation

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

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

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

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

1. Back-to-back Frame Benchmark was very consistent for some fixed frame sizes, and somewhat variable for others.
2. The Back-to-back Frame length reported for large frame sizes was unexpectedly long, and no explanation or measurement limit condition was indicated.
3. Calculation of the extent of buffer time in the DUT helped to explain the results observed with all frame sizes (for example, some frame sizes cannot exceed the frame header processing rate of the DUT and therefore no buffering occurs, therefore the results depended on the test equipment and not the DUT).
4. It was found that the actual buffer time in the DUT could be estimated using results from the Throughput tests conducted according to Section 26.1 of [RFC2544], because it appears that the DUT's frame processing rate may tend to increase the estimate.

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

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

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

#### 4. Prerequisites

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

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

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

Note that:

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

The Back-to-back Benchmark described in Section 3.1 of [RFC1242] MUST be measured directly by the tester. Additional measurement requirements are described below in Section 5.

#### 5. Back-to-back Frames

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

The Procedure follows.

##### 5.1. Preparing the list of Frame sizes

From the list of RECOMMENDED Frame sizes (Section 9 of [RFC2544]), select the subset of Frame sizes whose measured Throughput was less than the maximum theoretical Frame Rate. These are the only Frame

sizes where it is possible to produce a burst of frames that cause the DUT buffers to fill and eventually overflow, producing one or more discarded frames.

## 5.2. Test for a Single Frame Size

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

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

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

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

@@@ OPNFV VSPERF Test Results presented at IETF-102 will have relevance for answering these questions.

@@@ Should a particular search algorithm be included? @@@ Yes, probably more than one. There are multiple roles for benchmark tests (Highly repetitive CI and Highly Accurate & Repeatable Experiments).

@@@ Should the search include trial repetition whenever frame loss is observed, to avoid the effects of background loss (un-related to buffer overflow)? @@@ Yes, OPNFV VSPERF results illustrate the value of this, and ETSI NFV TST009 Annex B describes the theory behind this new algorithm.

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

## 5.3. Test Repetition

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

#### 5.4. Benchmark Calculations

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

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

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

Implied DUT Buffer Time =

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

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

Corrected DUT Buffer Time =

Measured Throughput  
= Implied DUT Buffer Time \* -----  
Max Theoretical Frame Rate

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

#### 6. Reporting

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

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

The Minimum, Maximum, and Standard Deviation across all complete tests SHOULD also be reported.

The Corrected DUT Buffer Time SHOULD also be reported.

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

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

#### Back-to-Back Frame Results

Static and configuration parameters:

Number of test repetitions, N

Minimum Step Size (during searches), in frames.

#### 7. Security Considerations

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

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

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

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

#### 8. IANA Considerations

This memo makes no requests of IANA.

#### 9. Acknowledgements

Thanks to Trevor Cooper, Sridhar Rao, and Martin Kozik of the VSPERF project for many contributions to the testing [VSPERF-b2b]. Yoshiaki Itou has also investigated the topic, and made useful suggestions.

## 10. References

### 10.1. Normative References

- [RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <<https://www.rfc-editor.org/info/rfc1242>>.
- [RFC1944] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 1944, DOI 10.17487/RFC1944, May 1996, <<https://www.rfc-editor.org/info/rfc1944>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <<https://www.rfc-editor.org/info/rfc2544>>.
- [RFC5180] Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, DOI 10.17487/RFC5180, May 2008, <<https://www.rfc-editor.org/info/rfc5180>>.
- [RFC6201] Asati, R., Pignataro, C., Calabria, F., and C. Olvera, "Device Reset Characterization", RFC 6201, DOI 10.17487/RFC6201, March 2011, <<https://www.rfc-editor.org/info/rfc6201>>.
- [RFC6815] Bradner, S., Dubray, K., McQuaid, J., and A. Morton, "Applicability Statement for RFC 2544: Use on Production Networks Considered Harmful", RFC 6815, DOI 10.17487/RFC6815, November 2012, <<https://www.rfc-editor.org/info/rfc6815>>.
- [RFC6985] Morton, A., "IMIX Genome: Specification of Variable Packet Sizes for Additional Testing", RFC 6985, DOI 10.17487/RFC6985, July 2013, <<https://www.rfc-editor.org/info/rfc6985>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

## 10.2. Informative References

[OPNFV-2017]

Cooper, T., Morton, A., and S. Rao, "Dataplane Performance, Capacity, and Benchmarking in OPNFV", June 2017, <<https://wiki.opnfv.org/download/attachments/10293193/VSPERF-Dataplane-Perf-Cap-Bench.pptx?api=v2>>.

[RFC8239]

Avramov, L. and J. Rapp, "Data Center Benchmarking Methodology", RFC 8239, DOI 10.17487/RFC8239, August 2017, <<https://www.rfc-editor.org/info/rfc8239>>.

[VSPERF-b2b]

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

## Author's Address

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

Phone: +1 732 420 1571  
Fax: +1 732 368 1192  
Email: [acmorton@att.com](mailto:acmorton@att.com)



Network Working Group  
Internet-Draft  
Updates: 2544 (if approved)  
Intended status: Informational  
Expires: September 3, 2019

A. Morton  
AT&T Labs  
March 2, 2019

Updates for the Back-to-back Frame Benchmark in RFC 2544  
draft-morton-bmwg-b2b-frame-05

Abstract

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

This memo updates Section 26.4 of RFC 2544.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 3, 2019.

## Copyright Notice

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

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

## Table of Contents

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

## 1. Introduction

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

Recent testing experience with the Back-to-back Frame test and Benchmark in Section 26.4 of [RFC2544] indicates that an update is warranted [OPNFV-2017] [VSPERF-b2b]. This memo describes the rationale and provides the updated method.

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

## 2. Scope and Goals

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

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

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

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

## 3. Motivation

Section 3.1 of [RFC1242] describes the rationale for the Back-to-back Frames Benchmark. To summarize, there are several reasons that devices on a network produce bursts of frames at the minimum allowed spacing, and it is therefore worthwhile to understand the Device

Under Test (DUT) limit on the length of such bursts in practice. Also, [RFC1242] states:

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

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

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

1. Back-to-back Frame Benchmark was very consistent for some fixed frame sizes, and somewhat variable for others.
2. The Back-to-back Frame length reported for large frame sizes was unexpectedly long, and no explanation or measurement limit condition was indicated.
3. Calculation of the extent of buffer time in the DUT helped to explain the results observed with all frame sizes (for example, some frame sizes cannot exceed the frame header processing rate of the DUT and therefore no buffering occurs, therefore the results depended on the test equipment and not the DUT).
4. It was found that the actual buffer time in the DUT could be estimated using results from the Throughput tests conducted according to Section 26.1 of [RFC2544], because it appears that the DUT's frame processing rate may tend to increase the estimate.

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

[VSPERF-b2b] provides the details of the calculation to estimate the actual buffer storage available in the DUT, using results from the

Throughput tests for each frame size, and the maximum theoretical frame rate for the DUT links (which constrain the minimum frame spacing). We present some of these details here.

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

```

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

```

So, in the back2back frame testing:

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

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

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

#### 4. Prerequisites

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

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

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

Note that:

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

The Back-to-back Benchmark described in Section 3.1 of [RFC1242] MUST be measured directly by the tester. Additional measurement requirements are described below in Section 5.

## 5. Back-to-back Frames

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

The Procedure follows.

### 5.1. Preparing the list of Frame sizes

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

### 5.2. Test for a Single Frame Size

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

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

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

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

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

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

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

### 5.3. Test Repetition

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

### 5.4. Benchmark Calculations

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

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

- o Standard Deviation

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

Implied DUT Buffer Time =

$$\text{Average num of Back-to-back Frames} / \text{Max Theoretical Frame Rate}$$

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

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

Corrected DUT Buffer Time =

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

where:

1. The "Measured Throughput" is the RFC2544 Throughput Benchmark for the frame size tested, and MUST be expressed in Frames per second in this equation.
2. The "Max Theoretical Frame Rate" is a calculated value for the interface speed and link layer technology used, and MUST be expressed in Frames per second in this equation.

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

## 6. Reporting

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

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



The Minimum, Maximum, and Standard Deviation across all complete tests SHOULD also be reported.

The Corrected DUT Buffer Time SHOULD also be reported.

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

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

#### Back-to-Back Frame Results

Static and configuration parameters:

Number of test repetitions, N

Minimum Step Size (during searches), in frames.

#### 7. Security Considerations

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

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

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

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

#### 8. IANA Considerations

This memo makes no requests of IANA.

## 9. Acknowledgements

Thanks to Trevor Cooper, Sridhar Rao, and Martin Klokik of the VSPERF project for many contributions to the testing [VSPERF-b2b]. Yoshiaki Itou has also investigated the topic, and made useful suggestions.

## 10. References

### 10.1. Normative References

- [RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <<https://www.rfc-editor.org/info/rfc1242>>.
- [RFC1944] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 1944, DOI 10.17487/RFC1944, May 1996, <<https://www.rfc-editor.org/info/rfc1944>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <<https://www.rfc-editor.org/info/rfc2544>>.
- [RFC5180] Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, DOI 10.17487/RFC5180, May 2008, <<https://www.rfc-editor.org/info/rfc5180>>.
- [RFC6201] Asati, R., Pignataro, C., Calabria, F., and C. Olvera, "Device Reset Characterization", RFC 6201, DOI 10.17487/RFC6201, March 2011, <<https://www.rfc-editor.org/info/rfc6201>>.
- [RFC6815] Bradner, S., Dubray, K., McQuaid, J., and A. Morton, "Applicability Statement for RFC 2544: Use on Production Networks Considered Harmful", RFC 6815, DOI 10.17487/RFC6815, November 2012, <<https://www.rfc-editor.org/info/rfc6815>>.

- [RFC6985] Morton, A., "IMIX Genome: Specification of Variable Packet Sizes for Additional Testing", RFC 6985, DOI 10.17487/RFC6985, July 2013, <<https://www.rfc-editor.org/info/rfc6985>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

## 10.2. Informative References

- [OPNFV-2017] Cooper, T., Morton, A., and S. Rao, "Dataplane Performance, Capacity, and Benchmarking in OPNFV", June 2017, <<https://wiki.opnfv.org/download/attachments/10293193/VSPERF-Dataplane-Perf-Cap-Bench.pptx?api=v2>>.
- [RFC8239] Avramov, L. and J. Rapp, "Data Center Benchmarking Methodology", RFC 8239, DOI 10.17487/RFC8239, August 2017, <<https://www.rfc-editor.org/info/rfc8239>>.
- [TST009] ETSI Network Function Virtualization ISG, "ETSI GS NFV-TST 009 V3.1.1 (2018-10), "Network Functions Virtualisation (NFV) Release 3; Testing; Specification of Networking Benchmarks and Measurement Methods for NFVI"", October 2018, <[https://www.etsi.org/deliver/etsi\\_gs/NFV-TST/001\\_099/009/03.01.01\\_60/gs\\_NFV-TST009v030101p.pdf](https://www.etsi.org/deliver/etsi_gs/NFV-TST/001_099/009/03.01.01_60/gs_NFV-TST009v030101p.pdf)>.
- [VSPERF-b2b] Morton, A., "Back2Back Testing Time Series (from CI)", June 2017, <[https://wiki.opnfv.org/display/vsperf/Traffic+Generator+Testing#TrafficGeneratorTesting-AppendixB:Back2BackTestingTimeSeries\(fromCI\)](https://wiki.opnfv.org/display/vsperf/Traffic+Generator+Testing#TrafficGeneratorTesting-AppendixB:Back2BackTestingTimeSeries(fromCI))>.
- [VSPERF-BSLV] Morton, A. and S. Rao, "Evolution of Repeatability in Benchmarking: Fraser Plugfest (Summary for IETF BMWG)", July 2018, <<https://datatracker.ietf.org/meeting/102/materials/slides-102-bmwg-evolution-of-repeatability-in-benchmarking-fraser-plugfest-summary-for-ietf-bmwg-00>>.

Author's Address

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

Phone: +1 732 420 1571  
Fax: +1 732 368 1192  
Email: [acmorton@att.com](mailto:acmorton@att.com)

BMWG  
Internet-Draft  
Intended status: Informational  
Expires: January 3, 2019

R. Rosa, Ed.  
C. Rothenberg  
UNICAMP  
M. Peuster  
H. Karl  
UPB  
July 2, 2018

Methodology for VNF Benchmarking Automation  
draft-rosa-bmwg-vnfbench-02

Abstract

This document describes a common methodology for automated benchmarking of Virtualized Network Functions (VNFs) executed on general-purpose hardware. Specific cases of benchmarking methodologies for particular VNFs can be derived from this document. Two open source reference implementations are reported as running code embodiments of the proposed, automated benchmarking methodology.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 3, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. Scope . . . . .	4
4. Considerations . . . . .	4
4.1. VNF Testing Methods . . . . .	4
4.2. Generic VNF Benchmarking Setup . . . . .	5
4.3. Deployment Scenarios . . . . .	7
4.4. Influencing Aspects . . . . .	8
5. Methodology . . . . .	9
5.1. VNF Benchmarking Descriptor (VNF-BD) . . . . .	10
5.1.1. Procedures Configuration . . . . .	10
5.1.2. Target Information . . . . .	10
5.1.3. Deployment Scenario . . . . .	10
5.2. VNF Performance Profile (VNF-PP) . . . . .	11
5.2.1. Execution Environment . . . . .	12
5.2.2. Measurement Results . . . . .	12
5.3. Automated Benchmarking Procedures . . . . .	13
5.4. Particular Cases . . . . .	15
6. Open Source Reference Implementations . . . . .	16
6.1. Gym . . . . .	16
6.2. tng-bench . . . . .	17
7. Security Considerations . . . . .	18
8. IANA Considerations . . . . .	18
9. Acknowledgement . . . . .	18
10. References . . . . .	18
10.1. Normative References . . . . .	18
10.2. Informative References . . . . .	19
Authors' Addresses . . . . .	20

## 1. Introduction

Benchmarking Methodology Working Group (BMWG) initiated efforts, approaching considerations in [RFC8172], to develop methodologies for benchmarking VNFs. Similarly described in [RFC8172], VNF benchmark motivating aspects define: (i) pre-deployment infrastructure dimensioning to realize associated VNF performance profiles; (ii) comparison factor with physical network functions; (iii) and output results for analytical VNF development.

Having no strict and clear execution boundaries, different from earlier self-contained black-box benchmarking methodologies described

in BMWG, a VNF depends on underlying virtualized environment parameters [ETSI14a], intrinsic factors to be analyzed when one investigates the performance of a VNF. This document stands as a ground methodology guide for VNF benchmarking automation. It addresses the state-of-the-art publications and the current developments in similar standardization efforts (e.g., [ETSI14c] and [RFC8204]) towards benchmarking VNFs.

Automating the extraction of VNF performance metrics propitiates: (i) the development of agile performance-focused DevOps methodologies for Continuous Integration and Delivery (CI/CD) of VNFs; (ii) the creation of on-demand VNF test descriptors for upcoming execution environments; (iii) the path for precise-analytics of extensively automated catalogues of VNF performance profiles; (iv) and run-time profiling mechanisms to assist VNF lifecycle orchestration/management workflows.

## 2. Terminology

Common benchmarking terminology contained in this document is derived from [RFC1242]. Also, the reader is assumed to be familiar with the terminology as defined in the European Telecommunications Standards Institute (ETSI) NFV document [ETSI14b]. Some of these terms, and others commonly used in this document, are defined below.

NFV: Network Function Virtualization - The principle of separating network functions from the hardware they run on by using virtual hardware abstraction.

NFVI PoP: NFV Infrastructure Point of Presence - Any combination of virtualized compute, storage and network resources.

NFVI: NFV Infrastructure - Collection of NFVI PoPs under one orchestrator.

VIM: Virtualized Infrastructure Manager - functional block that is responsible for controlling and managing the NFVI compute, storage and network resources, usually within one operator's Infrastructure Domain (e.g. NFVI-PoP).

VNFM: Virtualized Network Function Manager - functional block that is responsible for controlling and managing the VNF life-cycle.

NFVO: NFV Orchestrator - functional block that manages the Network Service (NS) life-cycle and coordinates the management of NS life-cycle, VNF life-cycle (supported by the VNFM) and NFVI resources (supported by the VIM) to ensure an optimized allocation of the necessary resources and connectivity.

**VNF:** Virtualized Network Function - a software-based network function. A VNF can be either represented by a single entity or be composed by a set of smaller, interconnected software components, called VNF components (VNFCs) [ETSI14d]. Those VNFs are also called composed VNFs.

**VNFD:** Virtualised Network Function Descriptor - configuration template that describes a VNF in terms of its deployment and operational behaviour, and is used in the process of VNF onboarding and managing the life cycle of a VNF instance.

**VNFC:** Virtualized Network Function Component - a software component that implements (parts of) the VNF functionality. A VNF can consist of a single VNFC or multiple, interconnected VNFCs [ETSI14d]

**VNF-FG:** Virtualized Network Function Forwarding Graph - an ordered list of VNFs or VNFCs creating a service chain.

### 3. Scope

This document assumes VNFs as black boxes when defining their benchmarking methodologies. White box approaches are assumed and analysed as a particular case under the proper considerations of internal VNF instrumentation, later discussed in this document.

In what follows, this document outlines a basis methodology for VNF benchmarking, specifically addressing its automation.

### 4. Considerations

VNF benchmarking considerations are defined in [RFC8172]. Additionally, VNF pre-deployment testing considerations are well explored in [ETSI14c].

#### 4.1. VNF Testing Methods

Following the ETSI's model in [ETSI14c], we distinguish three methods for VNF evaluation:

**Benchmarking:** Where parameters (e.g., cpu, memory, storage) are provided and the corresponding performance metrics (e.g., latency, throughput) are obtained. Note, such request might create multiple reports, for example, with minimal latency or maximum throughput results.

**Verification:** Both parameters and performance metrics are provided and a stimulus verify if the given association is correct or not.



**Dimensioning:** Where performance metrics are provided and the corresponding parameters obtained. Note, multiple deployment interactions may be required, or if possible, underlying allocated resources need to be dynamically altered.

**Note:** Verification and Dimensioning can be reduced to Benchmarking. Therefore, we detail Benchmarking in what follows.

#### 4.2. Generic VNF Benchmarking Setup

A generic VNF benchmarking setup is shown in Figure 1, and its components are explained below. Note here, not all components are mandatory, and VNF benchmarking scenarios, further explained, can dispose its components in varied settings.

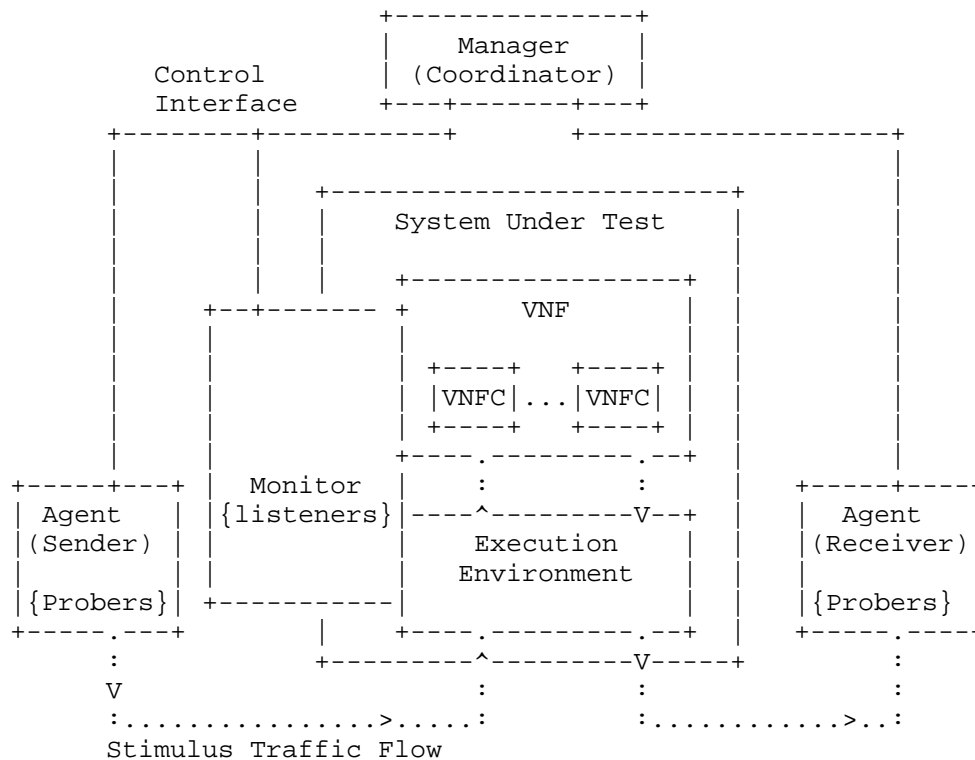


Figure 1: Generic VNF Benchmarking Setup

Agent -- executes active stimulus using probers, i.e., benchmarking tools, to benchmark and collect network and system performance

metrics. While a single Agent is capable of performing localized benchmarks in execution environments (e.g., stress tests on CPU, memory, disk I/O), the interaction among distributed Agents enable the generation and collection of VNF end-to-end metrics (e.g., frame loss rate, latency). In a benchmarking setup, one Agent can create the stimuli and the other end be the VNF itself where, for example, one-way latency is evaluated. An Agent can be defined by a physical or virtual network function.

Prober -- defines a software/hardware-based tool able to generate stimulus traffic specific to a VNF (e.g., sipp) or generic to multiple VNFs (e.g., pktgen). A Prober must provide programmable interfaces for its life cycle management workflows, e.g., configuration of operational parameters, execution of stimuli, parsing of extracted metrics, and debugging options. Specific Probers might be developed to abstract and to realize the description of particular VNF benchmarking methodologies.

Monitor -- when possible, it is instantiated inside the target VNF or NFVI PoP (e.g., as a plug-in process in a virtualized environment) to perform passive monitoring, using listeners, for metrics collection based on benchmark tests evaluated according to Agents' stimuli. Different from the active approach of Agents that can be seen as generic benchmarking VNFs, Monitors observe particular properties according to NFVI PoPs and VNFs capabilities. A Monitor can be defined as a virtual network function.

Listener -- defines one or more software interfaces for the extraction of particular metrics monitored in a target VNF and/or execution environment. A Listener must provide programmable interfaces for its life cycle management workflows, e.g., configuration of operational parameters, execution of monitoring captures, parsing of extracted metrics, and debugging options. White-box benchmarking approaches must be carefully analyzed, as varied methods of performance monitoring might be coded as a Listener, possibly impacting the VNF and/or execution environment performance results.

Manager -- in a VNF benchmarking setup, a Manager is responsible for (i) the coordination and synchronization of activities of Agents and Monitors, (ii) collecting and parsing all VNF benchmarking results, and (iii) aggregating the inputs and parsed benchmark

outputs to construct a VNF performance profile, which defines a report that correlates the VNF stimuli and the monitored metrics. A Manager executes the main configuration, operation, and management actions to deliver the VNF benchmarking results. A Manager can be defined as a physical or virtual network function, and be split into multiple sub-components, each responsible for separated functional aspects of the overall Manager component.

Virtualized Network Function (VNF) -- consists of one or more software components, so called VNF components (VNFC), adequate for performing a network function according to allocated virtual resources and satisfied requirements in an execution environment. A VNF can demand particular configurations for benchmarking specifications, demonstrating variable performance profiles based on available virtual resources/parameters and configured enhancements targeting specific technologies (e.g., NUMA, SR-IOV, CPU-Pinning).

Execution Environment -- defines a virtualized and controlled composition of capabilities necessary for the execution of a VNF. An execution environment stands as a general purpose level of virtualization with abstracted resources available for one or more VNFs. It can also define specific technology habilitation, incurring in viable settings for enhancing the performance of VNFs.

#### 4.3. Deployment Scenarios

A VNF benchmark deployment scenario establishes the physical and/or virtual instantiation of components defined in a VNF benchmarking setup.

The following considerations hold for deployment scenarios:

- o Components can be composed in a single entity and be defined as black or white boxes. For instance, Manager and Agents could jointly define one hardware/software entity to perform a VNF benchmark and present results.
- o Monitor is not a mandatory component and must be considered only when performed white box benchmarking approaches for a VNF and/or its execution environment.
- o Monitor can be defined by multiple instances of software components, each addressing a VNF or execution environment and their respective open interfaces for the extraction of metrics.

- o Agents can be disposed in varied topology setups, included the possibility of multiple input and output ports of a VNF being directly connected each in one Agent.
- o All benchmarking components defined in a deployment scenario must perform the synchronization of clocks.

#### 4.4. Influencing Aspects

In general, VNF benchmarks must capture relevant causes of performance variability. Concerning a deployment scenario, influencing aspects on the performance of a VNF can be observed in:

**Deployment Scenario Topology:** The disposition of components can define particular interconnections among them composing a specific case/method of VNF benchmarking.

**Execution Environment:** The availability of generic and specific capabilities satisfying VNF requirements define a skeleton of opportunities for the allocation of VNF resources. In addition, particular cases can define multiple VNFs interacting in the same execution environment of a benchmarking setup.

**VNF:** A detailed description of functionalities performed by a VNF sets possible traffic forwarding and processing operations it can perform on packets, added to its running requirements and specific configurations, which might affect and compose a benchmarking setup.

**Agent:** The toolset available for the benchmarking stimulus of a VNF and its characteristics of packets format and workload can interfere in a benchmarking setup. VNFs can support specific traffic format as stimulus.

**Monitor:** In a particular benchmarking setup where measurements of VNF and/or execution environment metrics are available for extraction, an important analysis consist in verifying if the Monitor components can impact performance metrics of the VNF and the underlying execution environment.

**Manager:** The overall composition of VNF benchmarking procedures can determine arrangements of internal states inside a VNF, which can interfere in observed benchmarking metrics.

The listed influencing aspects must be carefully analyzed while automating a VNF benchmarking methodology.

## 5. Methodology

Portability is an intrinsic characteristic of VNFs and allows them to be deployed in multiple environments. This enables various benchmarking procedures in varied deployment scenarios. A VNF benchmarking methodology must be described in a clear and objective manner in order to allow effective repeatability and comparability of the test results. Those results, the outcome of a VNF benchmarking process, are captured in a VNF Benchmarking Report (VNF-BR) as shown in Figure 2.

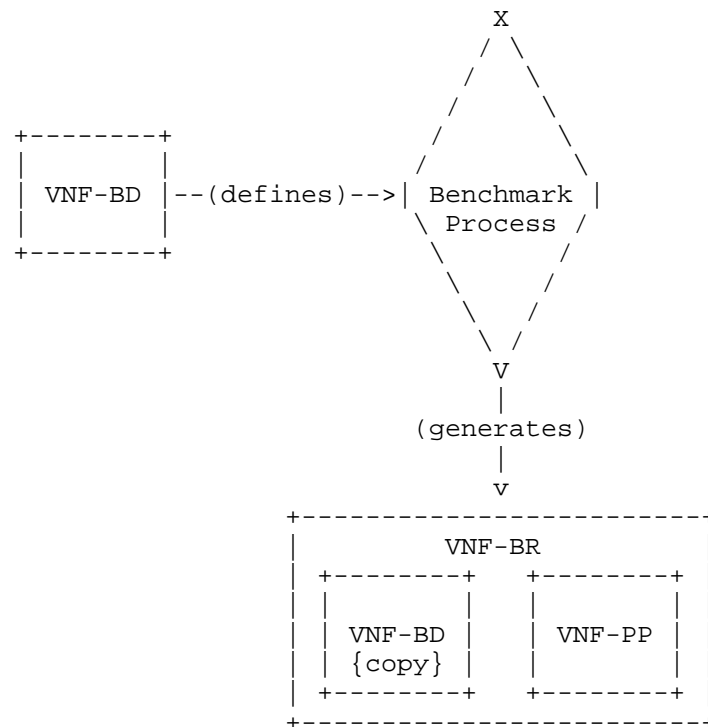


Figure 2: VNF benchmarking process inputs and outputs

VNF Benchmarking reports comprise two parts:

VNF Benchmarking Descriptor (VNF-BD) -- contains all required definitions and requirements to configure, execute and reproduce VNF benchmarking experiments. VNF-BDs are defined by the developer of a benchmarking experiment and serve as input to the benchmarking process, before being included in the generated VNF-BR.

VNF Performance Profile (VNF-PP) -- contains all measured metrics resulting from the execution of a benchmark. Additionally, it might also contain additional recordings of configuration parameters used during the execution of the benchmarking scenario to facilitate comparability of VNF-BRs.

A VNF-BR correlates structural and functional parameters of VNF-BD with extracted VNF benchmarking metrics of the obtained VNF-PP. The content of each part of a VNF-BR is described in the following sections.

#### 5.1. VNF Benchmarking Descriptor (VNF-BD)

VNF Benchmarking Descriptor (VNF-BD) -- an artifact that specifies a method of how to measure a VNF Performance Profile. The specification includes structural and functional instructions and variable parameters at different abstraction levels (e.g., topology of the deployment scenario, benchmarking target metrics, parameters of benchmarking components). VNF-BD may be specific to a VNF or applicable to several VNF types. A VNF-BD can be used to elaborate a VNF benchmark deployment scenario aiming at the extraction of particular VNF performance metrics.

The following items define the VNF-BD contents.

##### 5.1.1. Procedures Configuration

The definition of parameters concerning the execution of the benchmarking procedures (see Section 5.3), for instance, containing the number of repetitions and duration of each test.

##### 5.1.2. Target Information

General information addressing the target VNF, with references to any of its specific characteristics (e.g., type, model, version/release, architectural components, etc). In addition, it defines the metrics to be extracted when running the benchmarking tests.

##### 5.1.3. Deployment Scenario

This section of a VNF-BD contains all information needed to describe the deployment of all involved components used during the benchmarking test.

#### 5.1.3.1. Topology

Information about the experiment topology, concerning the disposition of the components in a benchmarking setup (see Section 4.2). It must define the role of each component and how they are interconnected (i.e., interface, link and network characteristics).

#### 5.1.3.2. Requirements

Involves the definition of execution environment requirements to execute the tests. Therefore, they concern all required capabilities needed for the execution of the target VNF and the other components composing the benchmarking setup. Examples of specifications involve: min/max allocation of resources, specific enabling technologies (e.g., DPDK, SR-IOV, PCIE).

#### 5.1.3.3. Parameters

Involves any specific configuration of benchmarking components in a setup described the the deployment scenario topology.

**VNF Configurations:** Defines any specific configuration that must be loaded into the VNF to execute the benchmarking experiments (e.g., routing table, firewall rules, vIMS subscribers profile).

**VNF Resources:** Contains particular VNF resource configurations that should be tested during the benchmarking process, e.g., test the VNF for configurations with 2, 4, and 8 vCPUs associated.

**Agents:** Defines the configured toolset of available probers and related benchmarking/active metrics, available workloads, traffic formats/traces, and configurations to enable hardware capabilities (if existent).

**Monitors:** defines the configured toolset of available listeners and related monitoring/passive metrics, configuration of the interfaces with the monitoring target (VNF and/or execution environment), and configurations to enable specific hardware capabilities (if existent).

### 5.2. VNF Performance Profile (VNF-PP)

VNF Performance Profile (VNF-PP) -- defines a mapping between resources allocated to a VNF (e.g., CPU, memory) as well as assigned configurations (e.g., routing table used by the VNF) and the VNF performance metrics (e.g., throughput, latency between in/out ports) obtained in a benchmarking test conducted using a VNF-BD. Logically, packet processing metrics are presented in a specific format

addressing statistical significance (e.g., median, standard deviation, percentiles) where a correspondence among VNF parameters and the delivery of a measured VNF performance exists.

The following items define the VNF-PP contents.

#### 5.2.1. Execution Environment

Execution environment information is has to be included in every VNF-PP and is required to describe the environment on which a benchmark was actually executed.

Ideally, any person who has a VNF-BD and its complementing VNF-PP with its execution environment information available, must be able to reproduce the same deployment scenario and VNF benchmarking tests to obtain identical VNF-PP measurement results.

If not already defined by the VNF-BD deployment scenario requirements (Section 5.1.3), for each component in the VNF benchmarking setup, the following topics must be detailed:

**Hardware Specs:** Contains any information associated with the underlying hardware capabilities offered and used by the component during the benchmarking tests. Examples of such specification include allocated CPU architecture, connected NIC specs, allocated memory DIMM, etc. In addition, any information concerning details of resource isolation must also be described in this part of the VNF-PP.

**Software Specs:** Contains any information associated with the software apparatus offered and used during the benchmarking tests. Examples include versions of operating systems, kernels, hypervisors, container image versions, etc.

Optionally, a VNF-PP execution environment might contain references to an orchestration description document (e.g., HEAT template) to clarify technological aspects of the execution environment and any specific parameters that it might contain for the VNF-PP.

#### 5.2.2. Measurement Results

Measurement results concern the extracted metrics, output of benchmarking procedures, classified into:

**VNF Processing/Active Metrics:** Concerns metrics explicitly defined by or extracted from direct interactions of Agents with a VNF. Those can be defined as generic metric related to network packet



processing (e.g., throughput, latency) or metrics specific to a particular VNF (e.g., vIMS confirmed transactions, DNS replies).

**VNF Monitored/Passive Metrics:** Concerns the Monitors' metrics captured from a VNF execution, classified according to the virtualization level (e.g., baremetal, VM, container) and technology domain (e.g., related to CPU, memory, disk) from where they were obtained.

Depending on the configuration of the benchmarking setup and the planned use cases for the resulting VNF-PPs, measurement results can be stored as raw data, e.g., time series data about CPU utilization of the VNF during a throughput benchmark. In the case of VNFs composed of multiple VNFCs, those resulting data should be represented as vectors, capturing the behavior of each VNFC, if available from the used monitoring systems. Alternatively, more compact representation formats can be used, e.g., statistical information about a series of latency measurements, including averages and standard deviations. The exact output format to be used is defined in the complementing VNF-BD (Section 5.1).

A VNF performance profile must address the combined set of classified items in the 3x3 Matrix Coverage defined in [RFC8172].

### 5.3. Automated Benchmarking Procedures

VNF benchmarking offers the possibility of defining distinct aspects/steps that may or may not be automated:

**Orchestration:** placement (assignment/allocation of resources) and interconnection (physical/virtual) of network function(s) and benchmark components (e.g., OpenStack/Kubernetes templates, NFV description solutions, like OSM, SONATA, ONAP) -> Defines deployment scenario.

**Management/Configuration:** benchmark components and VNF are configured to execute the experiment/test (e.g., populate routing table, load pcap source files in agent) -> Realizes VNF-BD settings.

**Execution:** Tests/experiments are executed according to configuration and orchestrated components -> Runs the VNF benchmarking cases.

**Output:** There might be generic VNF footprint metrics (e.g., CPU, memory) and specific VNF traffic processing metrics (e.g., transactions or throughput). Output processing must be taken into

account (e.g., if sampling is applied or not) in a generic (statistics) or specific (clustering) ways -> Generates VNF-PP.

For the purposes of dissecting the execution procedures, consider the following definitions:

**Trial:** is a single process or iteration to obtain VNF benchmarking metrics as a singular measurement.

**Test:** Defines strict parameters for benchmarking components to perform one or more trials. Multiple Trials must be performed for statistical significance of the obtained benchmarking results of a Test. Each Trial must be executed following a particular deployment scenario composed by a VNF-BD. Proper measures must be taken to ensure statistic validity (e.g., independence across trials of generated load patterns).

**Method:** Consists of a VNF-BD, including one or more Tests to benchmark a VNF. A Method can explicitly list ranges of parameter values for the configuration of benchmarking components. Each value of such a range is to be realized in a Test. I.e., Methods can define parameter studies.

The following sequence of events composes basic, general procedures to execute a Test (as defined above).

1. A VNF-BD must be defined to be later instantiated into and executed as a deployment scenario. Such a description must contain all the structural and functional settings defined in Section 5.1. At the end of this step, the complete method of benchmarking the target VNF is defined.
2. Via an automated orchestrator or in a manual process, all the components of the VNF benchmark setup must be allocated and interconnected. VNF and the execution environment must be configured to properly address the VNF benchmark stimuli.
3. Manager, Agent(s) and Monitor(s) (if existing), must be started and configured to execute the benchmark stimuli and retrieve expected metrics captured during or at the end of each Trial. One or more trials realize the required measurements to characterize the performance behavior of a VNF according to the benchmark setup defined in the VNF-BD.
4. Output results from each obtained benchmarking test must be collected by the Manager. In an automated or manual process, intended metrics, as described in the VNF-BD, are extracted and combined to the final VNF-PP. The combination of used VNF-BD and

generated VNF-PP make up the resulting VNF benchmark report (VNF-BR).

#### 5.4. Particular Cases

Configurations and procedures concerning particular cases of VNF benchmarks address testing methodologies proposed in [RFC8172]. In addition to the general description previously defined, some details must be taken into consideration in the following VNF benchmarking cases.

**Noisy Neighbor:** An Agent can assume the role of a noisy neighbor, generating a particular workload in synchrony with a benchmarking procedure over a VNF. Adjustments of the noisy workload stimulus type, frequency, virtualization level, among others, must be detailed in the VNF-BD.

**Representative Capacity:** An average value of workload must be specified as an Agent stimulus. Considering a long-term analysis, the VNF must be configured to properly address a desired average behavior of performance in comparison with the value of the workload stimulus.

**Flexibility and Elasticity:** Having the possibility of a VNF be composed by multiple components (VNFCs), internal events of the VNF might trigger changes in VNF behavior, e.g., activating functionalities associated with elasticity such as load balancing. In this sense, a detailed characterization of a VNF must be specified (e.g. the VNFCs scaling state) and be contained in the VNF-PP and thus the VNF benchmarking report.

**On Failures:** Similar to the case before, VNF benchmarking setups must also capture the dynamics involved in VNF behavior. In case of failures, a VNF might restart itself and possibly result in an offline period (e.g., self healing). A VNF-PP and benchmarking report must capture such variation of VNF states.

**White Box VNF:** A benchmarking setup must define deployment scenarios to be compared with and without monitor components into the VNF and/or the execution environment, in order to analyze if the VNF performance is affected. The VNF-PP and benchmarking report must contain such analysis of performance variability, together with all the extracted VNF performance metrics.

## 6. Open Source Reference Implementations

There are two open source reference implementations that are build to automate benchmarking of Virtualized Network Functions (VNFs).

### 6.1. Gym

The software, named Gym, is a framework for automated benchmarking of Virtualized Network Functions (VNFs). It was coded following the initial ideas presented in a 2015 scientific paper entitled "VBaaS: VNF Benchmark-as-a-Service" [Rosa-a]. Later, the evolved design and prototyping ideas were presented at IETF/IRTF meetings seeking impact into NFVRG and BMWG.

Gym was built to receive high-level test descriptors and execute them to extract VNFs profiles, containing measurements of performance metrics - especially to associate resources allocation (e.g., vCPU) with packet processing metrics (e.g., throughput) of VNFs. From the original research ideas [Rosa-a], such output profiles might be used by orchestrator functions to perform VNF lifecycle tasks (e.g., deployment, maintenance, tear-down).

The proposed guiding principles, elaborated in [Rosa-b], to design and build Gym can be composed in multiple practical ways for different VNF testing purposes:

- o Comparability: Output of tests shall be simple to understand and process, in a human-read able format, coherent, and easily reusable (e.g., inputs for analytic applications).
- o Repeatability: Test setup shall be comprehensively defined through a flexible design model that can be interpreted and executed by the testing platform repeatedly but supporting customization.
- o Configurability: Open interfaces and extensible messaging models shall be available between components for flexible composition of test descriptors and platform configurations.
- o Interoperability: Tests shall be ported to different environments using lightweight components.

In [Rosa-b] Gym was utilized to benchmark a decomposed IP Multimedia Subsystem VNF. And in [Rosa-c], a virtual switch (Open vSwitch - OVS) was the target VNF of Gym for the analysis of VNF benchmarking automation. Such articles validated Gym as a prominent open source reference implementation for VNF benchmarking tests. Such articles set important contributions as discussion of the lessons learned and the overall NFV performance testing landscape, included automation.

Gym stands as one open source reference implementation that realizes the VNF benchmarking methodologies presented in this document. Gym is being released open source at [Gym]. The code repository includes also VNF Benchmarking Descriptor (VNF-BD) examples on the vIMS and OVS targets as described in [Rosa-b] and [Rosa-c].

## 6.2. tng-bench

Another software that focuses on implementing a framework to benchmark VNFs is the "5GTANGO VNF/NS Benchmarking Framework" also called "tng-bench" (previously "son-profile") and was is as part of the two European Union H2020 projects SONATA NFV and 5GTANGO [tango]. Its initial ideas were presented in [Peu-a] and the system design of the end-to-end prototype was presented in [Peu-b].

Tng-bench's aims to act as a framework for the end-to-end automation of VNF benchmarking processes. Its goal is to automate the benchmarking process in such a way that VNF-PPs can be generated without further human interaction. This enables the integration of VNF benchmarking into continuous integration and continuous delivery (CI/CD) pipelines so that new VNF-PPs are generated on-the-fly for every new software version of a VNF. Those automatically generated VNF-PPs can then be bundled with the VNFs and serve as inputs for orchestration systems, fitting to the original research ideas presented in [Rosa-a] and [Peu-a].

Following the same high-level VNF testing purposes as Gym, namely: Comparability, repeatability, configurability, and interoperability, tng-bench specifically aims to explore description approaches for VNF benchmarking experiments. In [Peu-b] a prototype specification VNF-BDs is presented which not only allows to specify generic, abstract VNF benchmarking experiments, it also allows to describe sets of parameter configurations to be tested during the benchmarking process, allowing the system to automatically execute complex parameter studies on the SUT, e.g., testing a VNF's performance under different CPU, memory, or software configurations.

Tng-bench was used to perform a set of initial benchmarking experiments using different VNFs, like a Squid proxy, an Nginx load balancer, and a Socat TCP relay in [Peu-b]. Those VNFs have not only been benchmarked in isolation, but also in combined setups in which up to three VNFs were chained one after each other. These experiments were used to test tng-bench for scenarios in which composed VNFs, consisting of multiple VNF components (VNFCs), have to be benchmarked. The presented results highlight the need to benchmark composed VNFs in end-to-end scenarios rather than only benchmark each individual component in isolation, to produce meaningful VNF-PPs for the complete VNF.

Tng-bench is actively developed and released as open source tool under Apache 2.0 license [tng-bench].

## 7. Security Considerations

Benchmarking tests described in this document are limited to the performance characterization of VNFs in a lab environment with isolated network.

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

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

## 8. IANA Considerations

This document does not require any IANA actions.

## 9. Acknowledgement

The authors would like to thank the support of Ericsson Research, Brazil. Parts of this work have received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. H2020-ICT-2016-2 761493 (5GTANGO: <https://5gtango.eu>).

## 10. References

### 10.1. Normative References

- [ETSI14a] ETSI, "Architectural Framework - ETSI GS NFV 002 V1.2.1", Dec 2014, <[http://www.etsi.org/deliver/etsi\\_gs/NFV/001\\_099/002/01.02.01-\\_60/gs\\_NFV002v010201p.pdf](http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01-_60/gs_NFV002v010201p.pdf)>.
- [ETSI14b] ETSI, "Terminology for Main Concepts in NFV - ETSI GS NFV 003 V1.2.1", Dec 2014, <[http://www.etsi.org/deliver/etsi\\_gs/NFV/001\\_099-/003/01.02.01\\_60/gs\\_NFV003v010201p.pdf](http://www.etsi.org/deliver/etsi_gs/NFV/001_099-/003/01.02.01_60/gs_NFV003v010201p.pdf)>.
- [ETSI14c] ETSI, "NFV Pre-deployment Testing - ETSI GS NFV TST001 V1.1.1", April 2016, <[http://docbox.etsi.org/ISG/NFV/Open/DRAFTS/TST001\\_-\\_Pre-deployment\\_Validation/NFV-TST001v0015.zip](http://docbox.etsi.org/ISG/NFV/Open/DRAFTS/TST001_-_Pre-deployment_Validation/NFV-TST001v0015.zip)>.

- [ETSI14d] ETSI, "Network Functions Virtualisation (NFV); Virtual Network Functions Architecture - ETSI GS NFV SWA001 V1.1.1", December 2014, <[https://docbox.etsi.org/ISG/NFV/Open/Publications\\_pdf/Specs-Reports/NFV-SWA%20001v1.1.1%20-%20GS%20-%20Virtual%20Network%20Function%20Architecture.pdf](https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-SWA%20001v1.1.1%20-%20GS%20-%20Virtual%20Network%20Function%20Architecture.pdf)>.
- [RFC1242] S. Bradner, "Benchmarking Terminology for Network Interconnection Devices", July 1991, <<https://www.rfc-editor.org/info/rfc1242>>.
- [RFC8172] A. Morton, "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure", July 2017, <<https://www.rfc-editor.org/info/rfc8172>>.
- [RFC8204] M. Tahhan, B. O'Mahony, A. Morton, "Benchmarking Virtual Switches in the Open Platform for NFV (OPNFV)", September 2017, <<https://www.rfc-editor.org/info/rfc8204>>.

## 10.2. Informative References

- [Gym] "Gym Home Page", <<https://github.com/intrig-unicamp/gym>>.
- [Peu-a] M. Peuster, H. Karl, "Understand Your Chains: Towards Performance Profile-based Network Service Management", Fifth European Workshop on Software Defined Networks (EWSDN) , 2016, <<http://ieeexplore.ieee.org/document/7956044/>>.
- [Peu-b] M. Peuster, H. Karl, "Profile Your Chains, Not Functions: Automated Network Service Profiling in DevOps Environments", IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) , 2017, <<http://ieeexplore.ieee.org/document/8169826/>>.
- [Rosa-a] R. V. Rosa, C. E. Rothenberg, R. Szabo, "VBaaS: VNF Benchmark-as-a-Service", Fourth European Workshop on Software Defined Networks , Sept 2015, <<http://ieeexplore.ieee.org/document/7313620>>.
- [Rosa-b] R. Rosa, C. Bertoldo, C. Rothenberg, "Take your VNF to the Gym: A Testing Framework for Automated NFV Performance Benchmarking", IEEE Communications Magazine Testing Series , Sept 2017, <<http://ieeexplore.ieee.org/document/8030496>>.

- [Rosa-c] R. V. Rosa, C. E. Rothenberg, "Taking Open vSwitch to the Gym: An Automated Benchmarking Approach", IV Workshop pre-IETF/IRTF, CSBC Brazil, July 2017, <<https://intrig.dca.fee.unicamp.br/wp-content/plugins/papercite/pdf/rosa2017taking.pdf>>.
- [tango] "5GTANGO: Development and validation platform for global industry-specific network services and apps", <<https://5gtango.eu>>.
- [tng-bench] "5GTANGO VNF/NS Benchmarking Framework", <<https://github.com/sonata-nfv/tng-sdk-benchmark>>.

## Authors' Addresses

Raphael Vicente Rosa (editor)  
University of Campinas  
Av. Albert Einstein, 400  
Campinas, Sao Paulo 13083-852  
Brazil

Email: [rvrosa@dca.fee.unicamp.br](mailto:rvrosa@dca.fee.unicamp.br)  
URI: <https://intrig.dca.fee.unicamp.br/raphaelvrosa/>

Christian Esteve Rothenberg  
University of Campinas  
Av. Albert Einstein, 400  
Campinas, Sao Paulo 13083-852  
Brazil

Email: [chesteve@dca.fee.unicamp.br](mailto:chesteve@dca.fee.unicamp.br)  
URI: <http://www.dca.fee.unicamp.br/~chesteve/>

Manuel Peuster  
Paderborn University  
Warburgerstr. 100  
Paderborn 33098  
Germany

Email: [manuel.peuster@upb.de](mailto:manuel.peuster@upb.de)  
URI: <http://go.upb.de/peuster>



Holger Karl  
Paderborn University  
Warburgerstr. 100  
Paderborn 33098  
Germany

Email: [holger.karl@upb.de](mailto:holger.karl@upb.de)  
URI: <https://cs.uni-paderborn.de/cn/>

BMWG  
Internet-Draft  
Intended status: Informational  
Expires: April 23, 2021

R. Rosa, Ed.  
C. Rothenberg  
UNICAMP  
M. Peuster  
H. Karl  
UPB  
October 20, 2020

Methodology for VNF Benchmarking Automation  
draft-rosa-bmwg-vnfbench-06

Abstract

This document describes a common methodology for the automated benchmarking of Virtualized Network Functions (VNFs) executed on general-purpose hardware. Specific cases of automated benchmarking methodologies for particular VNFs can be derived from this document. An open source reference implementation is reported as running code embodiment of the proposed, automated benchmarking methodology.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 23, 2021.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	4
3. Scope . . . . .	6
4. Considerations . . . . .	6
4.1. VNF Assessment Methods . . . . .	7
4.2. Benchmarking Stages . . . . .	7
4.3. Architectural Framework . . . . .	8
4.4. Scenarios . . . . .	10
4.5. Phases of a Benchmarking Test . . . . .	11
4.5.1. Phase I: Deployment . . . . .	11
4.5.2. Phase II: Configuration . . . . .	11
4.5.3. Phase III: Execution . . . . .	12
4.5.4. Phase IV: Result . . . . .	12
5. Methodology . . . . .	12
5.1. VNF Benchmarking Descriptor (VNF-BD) . . . . .	13
5.2. VNF Performance Profile (VNF-PP) . . . . .	13
5.3. VNF Benchmarking Report (VNF-BR) . . . . .	14
5.4. Procedures . . . . .	14
5.4.1. Plan . . . . .	15
5.4.2. Realization . . . . .	16
5.4.3. Summary . . . . .	17
6. Particular Cases . . . . .	18
6.1. Capacity . . . . .	18
6.2. Redundancy . . . . .	18
6.3. Isolation . . . . .	18
6.4. Failure Handling . . . . .	18
6.5. Elasticity and Flexibility . . . . .	19
6.6. Handling Configurations . . . . .	19
6.7. White Box VNF . . . . .	19
7. Open Source Reference Implementation . . . . .	19
7.1. Gym . . . . .	20
7.2. Related work: tng-bench . . . . .	20
8. Security Considerations . . . . .	21
9. IANA Considerations . . . . .	22
10. YANG Modules . . . . .	23
10.1. VNF-Benchmarking Descriptor . . . . .	23
10.2. VNF Performance Profile . . . . .	34
10.3. VNF Benchmarking Report . . . . .	41
11. Acknowledgement . . . . .	46
12. References . . . . .	46
12.1. Normative References . . . . .	46

12.2. Informative References . . . . .	47
Authors' Addresses . . . . .	49

## 1. Introduction

In [RFC8172] the Benchmarking Methodology Working Group (BMWG) presented considerations for benchmarking of VNFs and their infrastructure, similar to the motivation given, the following aspects reinforce and justify the need for VNF benchmarking: (i) pre-deployment infrastructure dimensioning to realize associated VNF performance profiles; (ii) comparison factor with physical network functions; (iii) and output results for analytical VNF development.

Even if many methodologies the BMWG already describes, e.g., self-contained black-box benchmarking, can be applied to VNF benchmarking scenarios, further considerations have to be made. This is because VNFs, which are software components, might not have strict and clear execution boundaries and depend on underlying virtualization environment parameters as well as management and orchestration decisions [ETS14a].

Different enabling technologies advent of Software Defined Networking (SDN) and Network Functions Virtualization (NFV) have propitiated the disaggregation of VNFs and benchmarking tools, turning their Application Programming Interfaces (APIs) open and programmable. This process have occurred mostly by: (i) the decoupling of network function's control and data planes; (ii) the development of VNFs as multi-layer and distributed software components; (iii) and the existence of multiple underlying hardware abstractions to be utilized by VNFs.

Utilizing SDN and NFV enabling technologies, a diversity of benchmarking tools have been created to facilitate the active stimulus and the passive monitoring of a VNF via diverse software abstraction layers, propitiating a wide variety of abstractions for benchmarking mechanisms in the formulation of a VNF benchmarking methodology. In this manner of establishing the disaggregation of a VNF benchmarking setup, the abstracted VNF benchmarking mechanisms can be programmable, enabling the execution of their underlying technologies by the means of well defined parameters and producing a report with standardized metrics.

Turning programmable the execution of a VNF benchmarking methodology enables a richer apparatus for the benchmarking of a VNF and consequently facilitates the high-fidelity assessment of a VNF behaviour. Estimating the behaviour of a VNF depends on three correlated factors:

Internal configuration: Each use case of the VNF might define specific settings for it to work properly, and even each VNF might dispose of specific settings to be configured.

Hardware and software execution environment: A myriad of capabilities offered by execution environments might match in a large diversity of manners the possible internal software arrangements that each VNF might be programmable.

Network workload specificities: Depending on the use case, a VNF might be placed in different settings, operating under varied traffic profiles and in demand of a specific performance behavior.

The role of a VNF benchmarking methodology consists in defining how to tackle the diversity of settings imposed by the above enlisted factors in order to extract performance metrics associated with particular VNF packet processing behaviors. The sample space of testing such diversity of settings can be extensively large, turning manual benchmarking experiments prohibitively expensive. Indeed, portability as an intrinsic characteristic of VNFs allows them to be deployed in multiple execution environments, enabling benchmarking setups in a myriad of settings. Thus, the establishment of a methodology for VNF benchmarking automation detains utter importance.

Accordingly, can and should the flexible, software-based nature of VNFs be exploited to fully automate the entire benchmarking methodology end-to-end. This is an inherent need to align VNF benchmarking with the agile methods enabled by the concept of Network Functions Virtualization (NFV) [ETSI14e]. More specifically it allows: (i) the development of agile performance-focused DevOps methodologies for Continuous Integration and Delivery (CI/CD) of VNFs; (ii) the creation of on-demand VNF test descriptors for upcoming execution environments; (iii) the path for precise-analytics of automated catalogues of VNF performance profiles; (iv) and run-time mechanisms to assist VNF lifecycle orchestration/management workflows, e.g., automated resource dimensioning based on benchmarking insights.

## 2. Terminology

Common benchmarking terminology contained in this document is derived from [RFC1242]. The reader is assumed to be familiar with the terminology as defined in the European Telecommunications Standards Institute (ETSI) NFV document [ETSI14b]. Some of these terms, and others commonly used in this document, are defined below.

NFV: Network Function Virtualization - the principle of separating network functions from the hardware they run on by using virtual hardware abstraction.

VNF: Virtualized Network Function - a software-based network function. A VNF can be either represented by a single entity or be composed by a set of smaller, interconnected software components, called VNF components (VNFCs) [ETSI14d]. Those VNFs are also called composed VNFs.

VNFC: Virtualized Network Function Component - a software component that implements (parts of) the VNF functionality. A VNF can consist of a single VNFC or multiple, interconnected VNFCs [ETSI14d]

VNFD: Virtualised Network Function Descriptor - configuration template that describes a VNF in terms of its deployment and operational behaviour, and is used in the process of VNF onboarding and managing the life cycle of a VNF instance.

NS: Network Service - a collection of interconnected VNFs forming a end-to-end service. The interconnection is often done using chaining of functions.

VNF Benchmarking Descriptor (VNF-BD) -- contains all the definitions and requirements to deploy, configure, execute, and reproduce VNF benchmarking tests. A VNF-BD is defined by the developer of a VNF benchmarking methodology and serve as input to the execution of an automated benchmarking methodology.

VNF Performance Profile (VNF-PP) -- in a well defined structure contains all the measured metrics resulting from the execution of automated VNF benchmarking tests defined by a specific VNF-BD. Additionally, it might also contain additional recordings of configuration parameters used during the execution of the benchmarking setup.

VNF Benchmarking Report (VNF-BR) -- contains all the definition of the inputs and outputs of an automated VNF benchmarking methodology. The inputs define the necessary VNF-BD and a respective list of variables referencing the VNF-BD fields that must be utilized to define the sample space of the VNF benchmarking settings. The outputs consist of a list of entries, each one contains one of the combinations of the sampled variables from the inputs, the input VNF-BD parsed with such combination of variables, and the obtained VNF-PP resulting from the automated realization of the parsed VNF-BD. A VNF-BR might contain the settings definitions of the orchestrator platform that realizes

the instantiation of the benchmarking setup to enable the VNF-BD fullfilment.

### 3. Scope

This document assumes VNFs as black boxes when defining their benchmarking methodologies. White box approaches are assumed and analysed as a particular case under the proper considerations of internal VNF instrumentation, later discussed in this document.

This document outlines a methodology for VNF benchmarking, specifically addressing its automation, without limiting the automated process to a specific benchmarking case or infrastructure. The document addresses state-of-the-art work on VNF benchmarking from scientific publications and current developments in other standardization bodies (e.g., [ETS14c], [ETS19f] and [RFC8204]) wherever possible.

Whenever utilizing the specifications of this document, a particular automated VNF benchmarking methodology must be described in a clear and objective manner following four basic principles:

- o Comparability: The output of a benchmarking test shall be simple to understand and process, in a human-readable format, coherent, and easily reusable (e.g., inputs for analytic applications).
- o Repeatability: A benchmarking setup shall be comprehensively defined through a flexible design model that can be interpreted and executed by the testing platform repeatedly but supporting customization.
- o Configurability: Open interfaces and extensible messaging models shall be available between benchmarking components for flexible composition of a benchmarking test descriptor and environment configurations.
- o Interoperability: A benchmarking test shall be ported to different environments, using lightweight components whenever possible.

### 4. Considerations

VNF benchmarking considerations are defined in [RFC8172]. Additionally, VNF pre-deployment testing considerations are well explored in [ETS14c]. Further, ETSI provides test specifications for networking benchmarks and measurement methods for NFV infrastructure in [ETS19f], which complements the presented work on VNF benchmarking methodologies.

#### 4.1. VNF Assessment Methods

Following ETSI's model in [ETSI14c], we distinguish three methods for a VNF evaluation:

**Benchmarking:** Where parameters (e.g., CPU, memory, storage) are provided and the corresponding performance metrics (e.g., latency, throughput) are obtained. Note, such evaluations might create multiple reports, for example, with minimal latency or maximum throughput results.

**Verification:** Both parameters and performance metrics are provided and a stimulus verifies if the given association is correct or not.

**Dimensioning:** Performance metrics are provided and the corresponding parameters obtained. Note, multiple deployments may be required, or if possible, underlying allocated resources need to be dynamically altered.

**Note:** Verification and Dimensioning can be reduced to Benchmarking.

#### 4.2. Benchmarking Stages

The realization of an automated benchmarking methodology can be divided into three stages:

**Trial:** Is a single process or iteration to obtain VNF performance metrics from benchmarking measurements. A Test MUST always run multiple Trials to get statistical confidence about the obtained measurements.

**Test:** Defines unique structural and functional parameters (e.g., configurations, resource assignment) for benchmarked components to perform one or multiple Trials. Each Test must be executed following a particular benchmarking scenario composed by a Method. Proper measures must be taken to ensure statistical validity (e.g., independence across Trials of generated load patterns).

**Method:** Consists of one or more Tests to benchmark a VNF. A Method can explicitly list ranges of parameter values for the configuration of a benchmarking scenario and its components. Each value of such a range is to be realized in a Test. I.e., Methods can define parameter studies.



### 4.3. Architectural Framework

A VNF benchmarking architectural framework, shown in Figure 1, establishes the disposal of essential components and control interfaces, explained below, that realize the automation of a VNF benchmarking methodology.

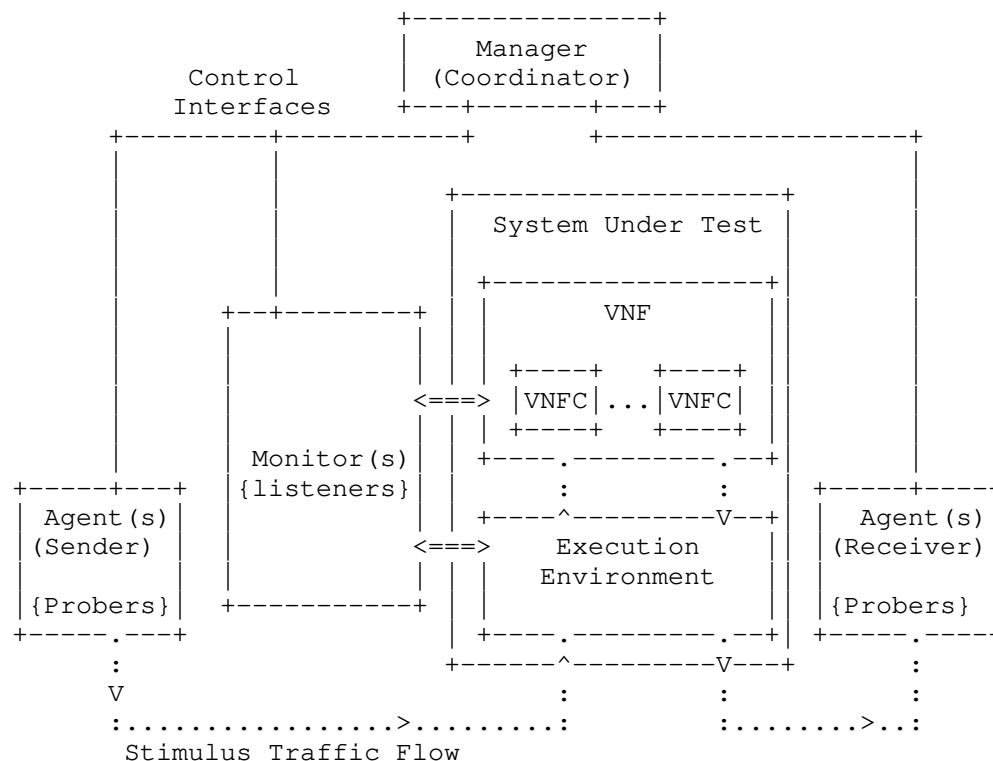


Figure 1: A VNF Benchmarking Architectural Framework

Virtualized Network Function (VNF) -- consists of one or more software components, so called VNF components (VNFC), adequate for performing a network function according to allocated virtual resources and satisfied requirements in an execution environment. A VNF can demand particular settings for benchmarking specifications, demonstrating variable performance based on available virtual resource parameters and configured enhancements targeting specific technologies (e.g., NUMA, SR-IOV, CPU-Pinning).

Execution Environment -- defines a virtualized and controlled composition of capabilities necessary for the execution of a VNF. An execution environment stands as a general purpose level of virtualization with abstracted resources available for one or more VNFs. It can also define specific technology qualifications, incurring in viable settings for enhancing the performance of VNFs, satisfying their particular enhancement requirements. An execution environment must be defined with the proper virtualization technologies feasible for the allocation of a VNF. The means to programmatically control the execution environment capabilities must be well defined for its life cycle management.

Agent (Active Prospection) -- executes active stimulus using probers, to benchmark and collect network and system performance metrics. A single Agent can perform localized benchmarks in execution environments (e.g., stress tests on CPU, memory, storage Input/Output) or can generate stimulus traffic and the other end be the VNF itself where, for example, one-way latency is evaluated. The interaction among two or more Agents enable the generation and collection of end-to-end metrics (e.g., frame loss rate, latency) measured from stimulus traffic flowing through a VNF. An Agent can be defined by a physical or virtual network function, and it must provide programmable interfaces for its life cycle management.

Prober -- defines an abstraction layer for a software or hardware tool able to generate stimulus traffic to a VNF or perform stress tests on execution environments. Probers might be specific or generic to an execution environment or a VNF. For an Agent, a Prober must provide programmable interfaces for its life cycle management, e.g., configuration of operational parameters, execution of stimulus, parsing of extracted metrics, and debugging options. Specific Probers might be developed to abstract and to realize the description of particular VNF benchmarking methodologies.

Monitor (Passive Prospection) -- when possible is instantiated inside the System Under Test, VNF and/or execution environment, to perform the passive monitoring, using Listeners, for the extraction of metrics while Agents' stimuli takes place. Monitors observe particular properties according to the execution environment and VNF capabilities, i.e., exposed passive monitoring interfaces. Multiple Listeners can be executed at once in synchrony with a Prober' stimulus on a SUT. A Monitor can be defined as a virtualized network function, and it must provide programmable interfaces for its life cycle management.

Listener -- defines one or more software interfaces for the extraction of metrics monitored in a target VNF and/or execution environment. A Listener must provide programmable interfaces for its life cycle management workflows, e.g., configuration of operational parameters, execution of passive monitoring captures, parsing of extracted metrics, and debugging options (also see [ETSI19g]). Varied methods of passive performance monitoring might be implemented as a Listener, depending on the interfaces exposed by the VNF and/or the execution environment.

Manager -- performs (i) the discovery of available Agents and Monitors and their respective features (i.e., available Probers/Listeners and their execution environment capabilities), (ii) the coordination and synchronization of activities of Agents and Monitors to perform a benchmarking Test, (iii) the collection, processing and aggregation of all VNF benchmarking (active and passive) metrics, which correlates the characteristics of the VNF traffic stimuli and the, possible, SUT monitoring. A Manager executes the main configuration, operation, and management actions to deliver the VNF benchmarking metrics. Hence, it detains interfaces open for users interact with the whole benchmarking framework, realizing, for instance, the retrieval of the framework characteristics (e.g., available benchmarking components and their probers/listeners), the coordination of benchmarking tests, the processing and the retrieval of benchmarking metrics, among other operational and management functionalities. A Manager can be defined as a physical or virtualized network function, and it must provide programmable interfaces for its life cycle management.

#### 4.4. Scenarios

A scenario, as well referred as a benchmarking setup, consists of the actual instantiation of physical and/or virtual components of a "VNF Benchmarking Architectural Framework" needed to habilitate the execution of an automated VNF benchmarking methodology. The following considerations hold for a scenario:

- o Not all components are mandatory for a Test, possible to be disposed in varied setups.
- o Components can be aggregated in a single entity and be defined as black or white boxes. For instance, Manager and Agents could jointly define one hardware or software entity to perform a VNF benchmarking Test.

- o Monitor can be defined by multiple instances of distributed software components, each one addressing one or more VNF or execution environment monitoring interfaces.
- o Agents can be disposed in varied topology setups, included the possibility of multiple input and output ports of a VNF being directly connected each in one Agent.
- o All benchmarking components defined in a scenario must perform the synchronization of clocks.

#### 4.5. Phases of a Benchmarking Test

In general, an automated benchmarking methodology must execute Tests repeatedly so it must capture the relevant causes of the performance variability of a VNF. To dissect a VNF benchmarking Test, in the sections that follow a set of benchmarking phases are categorized defining generic operations that may be automated. When executing an automated VNF benchmarking methodology, all the influencing aspects on the performance of a VNF must be carefully analyzed and comprehensively reported in each automated phase of a benchmarking Test.

##### 4.5.1. Phase I: Deployment

The placement (i.e., assignment and allocation of resources) and the interconnection, physical and/or virtual, of network function(s) and benchmarking components can be realized by orchestration platforms (e.g., OpenStack, Kubernetes, Open Source MANO). In automated manners, the realization of a benchmarking scenario through those means usually rely on network service templates (e.g., TOSCA, YANG, Heat, and Helm Charts). Such descriptors have to capture all relevant details of the execution environment to allow the benchmarking framework to correctly instantiate the SUT as well as helper functions required for a Test.

##### 4.5.2. Phase II: Configuration

The configuration of benchmarking components and VNFs (e.g., populate routing table, load PCAP source files in source of traffic stimulus) to execute the Test settings can be realized by programming interfaces in an automated way. In the scope of NFV, there might exist management interfaces to control a VNF during a benchmarking Test. Likewise, infrastructure or orchestration components can establish the proper configuration of an execution environment to realize all the capabilities enabling the description of the benchmarking Test. Each configuration registry, its deployment

timestamp and target, must all be contained in the report of a VNF benchmarking Test.

#### 4.5.3. Phase III: Execution

In the execution of a benchmarking Test, the VNF configuration can be programmed to be changed by itself or by a VNF management platform. It means that during a Trial execution, particular behaviors of a VNF can be automatically triggered, e.g., auto-scaling of its internal components. Those must be captured in the detailed procedures of the VNF execution and its performance report. I.e., the execution of a Trial can determine arrangements of internal states inside a VNF, which can interfere in observed benchmarking metrics. For instance, in a particular benchmarking case where the monitoring measurements of the VNF and/or execution environment are available for extraction, comparison Tests must be run to verify if the monitoring of the VNF and/or execution environment can impact the VNF performance metrics.

#### 4.5.4. Phase IV: Result

The result of a VNF benchmarking Test might contain generic metrics (e.g., CPU and memory consumption) and VNF-specific traffic processing metrics (e.g., transactions or throughput), which can be stored and processed in generic or specific ways (e.g., by statistics or machine learning algorithms). More details about possible metrics and the corresponding capturing methods can be found in [ETS19g]. If automated procedures are applied over the generation of a benchmarking Test result, those must be explained in the result itself, jointly with their input raw measurements and output processed data. For instance, any algorithm used in the generation of processed metrics must be disclosed in the Test result.

### 5. Methodology

The execution of an automated benchmarking methodology consists in elaborating a VNF Benchmarking Report, its inputs and outputs. The inputs part of a VNF-BR must be written by a VNF benchmarking tester. When the VNF-BR, with its inputs fulfilled, is requested from the Manager component of a implementation of the "VNF Benchmarking Architectural Framework", the Manager must utilize the inputs part to obtain the outputs part of the VNF-BR, addressing the execution of the automated benchmarking methodology as defined in Section 5.4.

The flow of information in the execution of an automated benchmarking methodology can be represented by the YANG modules defined by this document. The sections that follow present an overview of such modules.

### 5.1. VNF Benchmarking Descriptor (VNF-BD)

VNF Benchmarking Descriptor (VNF-BD) -- an artifact that specifies how to realize the Test(s) and Trial(s) of an automated VNF benchmarking methodology in order to obtain a VNF Performance Profile. The specification includes structural and functional instructions and variable parameters at different abstraction levels, such as the topology of the benchmarking scenario, and the execution parameters of prober(s)/listener(s) in the required Agent(s)/Monitor(s). A VNF-BD may be specific to a VNF or applicable to several VNF types.

More specifically, a VNF-BD is defined by a scenario and its proceedings. The scenario defines nodes (i.e., benchmarking components) and links interconnecting them, a topology that must be instantiated in order to execute the VNF-BD proceedings. The proceedings contain the specification of the required Agent(s) and Monitor(s) needed in the scenario nodes. Detailed in each Agent/Monitor follows the specification of the Prober(s)/Listener(s) required for the execution of the Tests, and in the details of each Prober/Listener follows the specification of its execution parameters. In the header of a VNF-BD is specified the number of Tests and Trials that a Manager must run them. Each Test realizes a unique instantiation of the scenario, while each Trial realizes a unique execution of the proceedings in the instantiated scenario of a Test. The VNF-BD YANG module is presented in Section 10.1.

### 5.2. VNF Performance Profile (VNF-PP)

VNF Performance Profile (VNF-PP) -- an output artifact of a VNF-BD execution performed by a Manager component. It contains all the metrics from Monitor(s) and/or Agent(s) components after realizing the execution of the Prober(s) and/or the Listener(s) proceedings, specified in its corresponding VNF-BD. Metrics are logically grouped according to the execution of the Trial(s) and Test(s) defined by a VNF-BD. A VNF-PP is specifically associated with a unique VNF-BD.

More specifically, a VNF-PP is defined by a structure that allows benchmarking results to be presented in a logical and unified format. A VNF-PP report is the result of a unique Test, while its content, the so called snapshot(s), each containing the results of the execution of a single Trial. Each snapshot is built by a single Agent or Monitor. A snapshot contains evaluation(s), each one being the output of the execution of a single Prober or Listener. An evaluation contains one or more metrics. In summary, a VNF-PP aggregates the results from reports (i.e., the Test(s)); a report aggregates Agent(s) and Monitor(s) results (i.e., the Trial(s)); a snapshot aggregates Prober(s) or Listener(s) results; and an

evaluation aggregates metrics. The VNF-PP YANG module is presented in Section 10.2.

### 5.3. VNF Benchmarking Report (VNF-BR)

VNF Benchmarking Report (VNF-BR) -- the core artifact of an automated VNF benchmarking methodology consisted of three parts: a header, inputs and output. The header refers to the VNF-BR description items (e.g., author, version, name), the description of the target SUT (e.g., the VNF version, release, name), and the environment settings specifying the parameters needed to instantiate the benchmarking scenario via an orchestration platform. The inputs contain the definitions needed to execute the automated benchmarking methodology of the target SUT, a VNF-BD and its variables settings. The outputs contain the results of the execution of the inputs, a list of entries, each one containing a VNF-BD filled with one of the combinations of the input variables settings, and the obtained VNF-PP reported after the execution of the Test(s) and Trial(s) of the parsed VNF-BD. The process of utilizing the VNF-BR inputs to generate its outputs concerns the realization of an automated VNF benchmarking methodology, explained in details in Section 5.4.2. The VNF-BR YANG module is presented in Section 10.3.

In details, each one of the variables in the inputs part of a VNF-BR is defined by: a name (the actual name of the variable); a path (the YANG path of the variable in the input VNF-BD); a type (the type of the values, such as string, int, float, etc); class (one of: stimulus, resource, configuration); and values (a list of the variable actual values). The values of all the variables must be combined all-by-all, generating a list containing the whole sample space of variables settings that must be used to create the VNF-BD instances. A VNF-BD instance is defined as the result of the parsing of one of those combinations of input variables into the VNF-BD of the VNF-BR inputs. The parsing takes place when the variable path is utilized to set its value in the VNF-BD. Iteratively, all the VNF-BD instances must have its Test(s) and Trial(s) executed to generate its corresponding VNF-PP. After all the VNF-BD instances had their VNF-PP accomplished, the realization of the whole automated VNF benchmarking methodology is complete, fulfilling the outputs part of the VNF-BR as shown in Figure 2.

### 5.4. Procedures

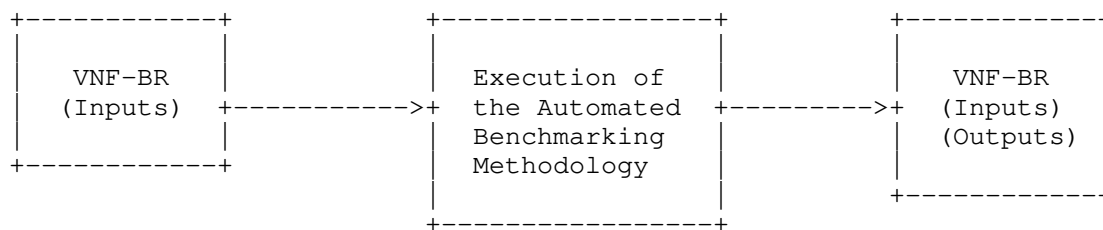


Figure 2: VNF benchmarking process inputs and outputs

The methodology for VNF benchmarking automation encompasses the process defined in Figure 2, i.e., the procedures that utilize the inputs part to obtain the outputs part of a VNF-BR. This section details the procedures that realize such process.

#### 5.4.1. Plan

The plan of an automated VNF benchmarking methodology consists in the definition of all the header and the inputs part of a VNF-BR, the artifacts to be utilized by the realization of the methodology, and the establishment of the execution environment where the methodology takes place. The topics below contain the details of such planning.

1. The writing of a VNF-BD must be done utilizing the VNF-BD YANG module Section 10.1. A VNF-BD composition must determine the scenario and the proceedings. The VNF-BD must be added to the inputs part of an instance of the VNF-BR YANG model.
2. All the variables in the inputs part of a VNF-BR must be defined. Each variable must contain all its fields fullfiled according to the VNF-BR YANG module Section 10.3.
3. All the software artifacts needed for the instantiation of the VNF-BD scenario must be made and turn available for the execution of the Test(s) and Trial(s). The artifacts include the definition of software components that realize the role of the functional components of the Benchmarking Architectural Framework, i.e., the Manager, the Agent and the Monitor and their respective Probers and Listeners.
4. The header of the VNF-BR instance must be written, stating the VNF-BR description items, the specification of the SUT settings, and the definition of the environment parameters, feasible for the instantiation of the VNF-BD scenario when executing the automated VNF benchmarking methodology.



5. The execution environment needed for a VNF-BD scenario must be prepared to be utilized by an orchestration platform to automate instantiation of the scenario nodes and links needed for the execution of a Test. The orchestration platform interface parameters must be referenced in the VNF-BR header. The orchestration platform must have access to the software artifacts that are referenced in the VNF-BD scenario to be able to manage their life cycle.
6. The Manager component must be instantiated, the execution environment must be turned available, and the orchestration platform must have access to the execution environment and the software artifacts that are referenced in the scenario of the VNF-BD in the inputs part of the VNF-BR.

#### 5.4.2. Realization

Accomplished all the planning procedures, the process of the realization of the automated benchmarking methodology must be realized as the following topics describe.

1. The realization of the benchmarking procedures starts when the VNF-BR composed in the planning procedures is submitted to the Manager component. It triggers the automated execution of the benchmarking methodology defined by the inputs part of the VNF-BR.
2. Manager computes all the combinations of values from the lists of inputs in the VNF-BD, part of the submitted VNF-BR. Each combination of variables are used to define a Test. The VNF-BD submitted serves as a template for each combination of variables. Each parsing of each combination of variables by the VNF-BD template creates a so called VNF-BD instance. The Manager must iterate through all the VNF-BD instances to finish the whole set of Tests defined by all the combinations of variables and their respective parsed VNF-BD. The Manager iterates through the following steps until all the Tests are accomplished.
3. The Manager must interface an orchestration platform to realize the automated instantiation of the deployment scenario defined by a VNF-BD instance (i.e., a Test). To perform such step, The Manager might interface a management function responsible to properly parse the deployment scenario specifications into the orchestration platform interface format. The environment specifications of the VNF-BR header provide the guidelines to interface the orchestration platform. The orchestration platform must deploy the scenario requested by the Manager, assuring the requirements and policies specified on it. In addition, the orchestration platform must acknowledge the deployed scenario to

the Manager specifying the management interfaces of the VNF SUT and the other components in the running instances for the benchmarking scenario. Only when the scenario is correctly deployed the execution of the VNF-BD instance Test(s) and Trial(s) must occur, otherwise the whole execution of the VNF-BR must be aborted and an error message must be added to the VNF-BR outputs describing the problems that occurred in the instantiation of the VNF-BD scenario. If the scenario is successfully deployed, the VNF-BD Test proceedings can be executed.

4. Manager must interface Agent(s) and Monitor(s) via their management interfaces to require the execution of the VNF-BD proceedings, which consist in running the specified Probers and Listeners using the defined parameters, and retrieve their output metrics captured at the end of each Trial. Thus, a Trial conceives the execution of the proceedings of the VNF-BD instance. The number of Trials is defined in each VNF-BD instance. After the execution of all defined Trials the execution of a Test ends.
5. Output measurements from each obtained benchmarking Trials that compose a Test result must be collected by the Manager, until all the Tests are finished. Each set of collected measurements from each VNF-BD instance Trials and Tests must be used to elaborate a VNF-PP by the Manager component. The respective VNF-PP, its associated VNF-BD instance and its input variables compose one of the entries of the list of outputs of the VNF-BR. After all the list of combinations of input variables is explored to obtain the whole list of instances of VNF-BDs and elaborated VNF-PPs, the Manager component returns the original VNF-BR submitted to it, including the outputs part properly filled.

#### 5.4.3. Summary

After the realization of an automated benchmarking methodology, some automated procedures can be performed to improve the quality and the utility of the obtained VNF-BR, as described in the following topics.

1. Archive the raw outputs contained in the VNF-BR, perform statistical analysis on it, or train machine learning models with the collected data.
2. Evaluate the analysis output to the detection of any possible cause-effect factors and/or intrinsic correlations in the VNF-BR outputs (e.g., outliers).
3. Review the inputs of a VNF-BR, VNF-BD and variables, and modify them to realize the proper extraction of the target VNF metrics based on the intended goal of the VNF benchmarking methodology

(e.g., throughput). Iterate in the previous steps until composing a stable and representative VNF-BR.

## 6. Particular Cases

As described in [RFC8172], VNF benchmarking might require to change and adapt existing benchmarking methodologies. More specifically, the following cases need to be considered.

### 6.1. Capacity

VNFs are usually deployed inside containers or VMs to build an abstraction layer between physical resources and the resources available to the VNF. According to [RFC8172], it may be more representative to design experiments in a way that the VMs hosting the VNFs are operating at maximum of 50% utilization and split the workload among several VMs, to mitigate side effects of overloaded VMs. Those cases are supported by the presented automation methodologies through VNF-BDs that enable direct control over the resource assignments and topology layouts used for a benchmarking experiment.

### 6.2. Redundancy

As a VNF might be composed of multiple components (VNFCs), there exist different schemas of redundancy where particular VNFCs would be in active or standby mode. For such cases, particular monitoring endpoints should be specified in VNF-BD so listeners can capture the relevant aspects of benchmarking when VNFCs would be in active/standby modes. In this particular case, capturing the relevant aspects of internal functionalities of a VNF and its internal components provides important measurements to characterize the dynamics of a VNF, those must be reflected in its VNF-PP.

### 6.3. Isolation

One of the main challenges of NFV is to create isolation between VNFs. Benchmarking the quality of this isolation behavior can be achieved by Agents that take the role of a noisy neighbor, generating a particular workload in synchrony with a benchmarking procedure over a VNF. Adjustments of the Agent's noisy workload, frequency, virtualization level, among others, must be detailed in the VNF-BD.

### 6.4. Failure Handling

Hardware and software components will fail or have errors and thus trigger healing actions of the benchmarked VNFs (self-healing). Benchmarking procedures must also capture the dynamics of this VNF

behavior, e.g., if a container or VM restarts because the VNF software crashed. This results in offline periods that must be captured in the benchmarking reports, introducing additional metrics, e.g., max. time-to-heal. The presented concept, with a flexible VNF-PP structure to record arbitrary metrics, enables automation of this case.

#### 6.5. Elasticity and Flexibility

Having software based network functions and the possibility of a VNF to be composed by multiple components (VNFCs), internal events of the VNF might trigger changes in VNF behavior, e.g., activating functionalities associated with elasticity such as automated scaling. These state changes and triggers (e.g. the VNF's scaling state) must be captured in the benchmarking results (VNF-PP) to provide a detailed characterization of the VNF's performance behavior in different states.

#### 6.6. Handling Configurations

As described in [RFC8172], does the sheer number of test conditions and configuration combinations create a challenge for VNF benchmarking. As suggested, machine readable output formats, as they are presented in this document, will allow automated benchmarking procedures to optimize the tested configurations. Approaches for this are, e.g., machine learning-based configuration space sub-sampling methods, such as [Peu-c].

#### 6.7. White Box VNF

A benchmarking setup must be able to define scenarios with and without monitoring components inside the VNFs and/or the hosting container or VM. If no monitoring solution is available from within the VNFs, the benchmark is following the black-box concept. If, in contrast, those additional sources of information from within the VNF are available, VNF-PPs must be able to handle these additional VNF performance metrics.

### 7. Open Source Reference Implementation

Currently, technical motivating factors in favor of the automation of VNF benchmarking methodologies comprise: (i) the facility to run high-fidelity and commodity traffic generators by software; (ii) the existent means to construct synthetic traffic workloads purely by software (e.g., handcrafted pcap files); (iii) the increasing availability of datasets containing actual sources of production traffic able to be reproduced in benchmarking tests; (iv) the existence of a myriad of automating tools and open interfaces to

programmatically manage VNFs; (v) the varied set of orchestration platforms enabling the allocation of resources and instantiation of VNFs through automated machineries based on well-defined templates; (vi) the ability to utilize a large tool set of software components to compose pipelines that mathematically analyze benchmarking metrics in automated ways.

In simple terms, the enlisted factors above justify that network softwarization enables the automation of VNF benchmarking methodologies. There exists an open source reference implementation that is built to demonstrate the concepts and methodology of this document in order to automate the benchmarking of Virtualized Network Functions.

### 7.1. Gym

The software, named Gym, is a framework for automated benchmarking of Virtualized Network Functions (VNFs). It was coded following the initial ideas presented in a 2015 scientific paper entitled "VBaaS: VNF Benchmark-as-a-Service" [Rosa-a]. Later, the evolved design and prototyping ideas were presented at IETF/IRTF meetings seeking impact into NFVRG and BMWG.

Gym was built to receive high-level test descriptors and execute them to extract VNFs profiles, containing measurements of performance metrics - especially to associate resources allocation (e.g., vCPU) with packet processing metrics (e.g., throughput) of VNFs. From the original research ideas [Rosa-a], such output profiles might be used by orchestrator functions to perform VNF lifecycle tasks (e.g., deployment, maintenance, tear-down).

In [Rosa-b] Gym was utilized to benchmark a decomposed IP Multimedia Subsystem VNF. And in [Rosa-c], a virtual switch (Open vSwitch - OVS) was the target VNF of Gym for the analysis of VNF benchmarking automation. Such articles validated Gym as a prominent open source reference implementation for VNF benchmarking tests. Such articles set important contributions as discussion of the lessons learned and the overall NFV performance testing landscape, included automation.

Gym stands as one open source reference implementation that realizes the VNF benchmarking methodologies presented in this document. Gym is released as open source tool under Apache 2.0 license [gym].

### 7.2. Related work: tng-bench

Another software that focuses on implementing a framework to benchmark VNFs is the "5GTANGO VNF/NS Benchmarking Framework" also called "tng-bench" (previously "son-profile") and was developed as

part of the two European Union H2020 projects SONATA NFV and 5GTANGO [tango]. Its initial ideas were presented in [Peu-a] and the system design of the end-to-end prototype was presented in [Peu-b].

Tng-bench aims to be a framework for the end-to-end automation of VNF benchmarking processes. Its goal is to automate the benchmarking process in such a way that VNF-PPs can be generated without further human interaction. This enables the integration of VNF benchmarking into continuous integration and continuous delivery (CI/CD) pipelines so that new VNF-PPs are generated on-the-fly for every new software version of a VNF. Those automatically generated VNF-PPs can then be bundled with the VNFs and serve as inputs for orchestration systems, fitting to the original research ideas presented in [Rosa-a] and [Peu-a].

Following the same high-level VNF testing purposes as Gym, namely: Comparability, repeatability, configurability, and interoperability, tng-bench specifically aims to explore description approaches for VNF benchmarking experiments. In [Peu-b] a prototype specification for VNF-BDs is presented which not only allows to specify generic, abstract VNF benchmarking experiments, it also allows to describe sets of parameter configurations to be tested during the benchmarking process, allowing the system to automatically execute complex parameter studies on the SUT, e.g., testing a VNF's performance under different CPU, memory, or software configurations.

Tng-bench was used to perform a set of initial benchmarking experiments using different VNFs, like a Squid proxy, an Nginx load balancer, and a Socat TCP relay in [Peu-b]. Those VNFs have not only been benchmarked in isolation, but also in combined setups in which up to three VNFs were chained one after each other. These experiments were used to test tng-bench for scenarios in which composed VNFs, consisting of multiple VNF components (VNFCs), have to be benchmarked. The presented results highlight the need to benchmark composed VNFs in end-to-end scenarios rather than only benchmark each individual component in isolation, to produce meaningful VNF-PPs for the complete VNF.

Tng-bench is actively developed and released as open source tool under Apache 2.0 license [tng-bench]. A larger set of example benchmarking results of various VNFs is available in [Peu-d].

## 8. Security Considerations

Benchmarking tests described in this document are limited to the performance characterization of VNFs in a lab environment with isolated network.

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

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

## 9. IANA Considerations

This document registers one URI in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

URI: urn:ietf:params:xml:ns:yang:ietf-vnf-bd  
Registrant Contact: The BMWG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-vnf-pp  
Registrant Contact: The BMWG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-vnf-br  
Registrant Contact: The BMWG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

Figure 3

This document registers three YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registration is requested:

```
name:      ietf-vnf-bd
namespace: urn:ietf:params:xml:ns:yang:ietf-vnf-bd
prefix:    vnf-bd
reference:  RFC CCCC

name:      ietf-vnf-pp
namespace: urn:ietf:params:xml:ns:yang:ietf-vnf-pp
prefix:    vnf-pp
reference:  RFC CCCC

name:      ietf-vnf-br
namespace: urn:ietf:params:xml:ns:yang:ietf-vnf-br
prefix:    vnf-br
reference:  RFC CCCC
```

Figure 4

## 10. YANG Modules

The following sections contain the YANG modules defined by this document.

### 10.1. VNF-Benchmarking Descriptor

```
module vnf-bd {
  namespace "urn:ietf:params:xml:ns:yang:vnf-bd";
  prefix "vnf-bd";

  organization "IETF/BMWG";
  contact "Raphael Vicente Rosa <raphaelvrosa@gmail.com>,
  Manuel Peuster <peuster@mail.uni-paderborn.de>";

  description "Yang module for a VNF Benchmarking
  Descriptor (VNF-BD).";

  revision "2019-08-13" {
    description "V0.3: Reviewed proceedings,
    tool - not VNF specific";
    reference "";
  }

  revision "2019-03-13" {
    description "V0.2: Reviewed role, policies, connection-points,
    lifecycle workflows, resources";
    reference "";
  }

  revision "2019-02-28" {
```



```
    description "V0.1: First release";
    reference "";
}

typedef workflows {
    type enumeration {
        enum create {
            description "When calling the create workflow.";
        }
        enum configure {
            description "When calling the configure workflow.";
        }
        enum start {
            description "When calling the start workflow.";
        }
        enum stop {
            description "When calling the stop workflow.";
        }
        enum delete {
            description "When calling the delete workflow.";
        }
        enum custom {
            description "When calling a custom workflow.";
        }
    }
    description "Defines basic life cycle workflows for a
node in a scenario.";
}

grouping node_requirements {
    container resources {
        container cpu {
            leaf vcpus {
                type uint32;
                description "The number of cores to be allocated
for a node.";
            }
            leaf cpu_bw {
                type string;
                description "The CPU bandwidth (CFS limit in 0.01-1.0)";
            }
            leaf pinning {
                type string;
                description "The list of CPU cores, separated by comma,
that a node must be pinned to.";
            }
        }
        description "The node CPU resources that must
be allocated for a benchmarking Test.";
    }
}
```

```
}
container memory {
  leaf size {
    type uint32;
    description "The memory allocation size.";
  }
  leaf unit {
    type string;
    description "The memory unit.";
  }
  description "The node memory resources
that must be allocated for a benchmarking
Test.";
}
container storage {
  leaf size {
    type uint32;
    description "The storage allocation size.";
  }
  leaf unit {
    type string;
    description "The storage unit.";
  }
  leaf volumes {
    type string;
    description "Volumes to be allocated by
a node storage.
A volume defines a mapping of an outside storage
partition inside the node storage system.
Volumes must be separated by comma and be defined
using a colon to separate the node internal and external
references of storage system paths.";
  }
  description "The node storage resources
that must be allocated for a benchmarking Test.";
}

description "The set of resources that must be allocated
for a node in a benchmarking Test.";
}

description "'The grouping determining the
resource requirements for a node in a scenario.'"
}

grouping connection_points {
  leaf id {
```

```
    type string;
    description "The connection-point
    unique identifier";
  }
  leaf interface {
    type string;
    description "The name of the node interface
    associated with the connection-point.";
  }
  leaf type {
    type string;
    description "The type of the network the
    connection-point interface is attached to.";
  }
  leaf address {
    type string;
    description "The Network address of the
    connection-point. It can be specified as a
    Ethernet MAC address, a IPv4 address or an IPv6 address.";
  }
  description "A connections-point of a node.";
}

grouping nodes {
  leaf id {
    type string;
    description "The unique identifier of a node
    in a scenario.";
  }
  leaf type {
    type string;
    description "The type of a node.";
  }
  leaf image {
    type string;
    description "The name of the image to be used to instantiate
    a node.";
  }
  leaf format {
    type string;
    description "The node format (e.g., container, process, VM).";
  }
  leaf role {
    type string;
    description "The role of the node in the Test scenario.
    The role must be one of: manager, agent, monitor, sut.";
  }
}
```

```
uses node_requirements;

list connection_points {
  key "id";
  uses connection_points;
  description "The list of connection points of a node.";
}

list relationships {
  key "name";
  leaf name {
    type string;
    description "Name of the relationship.";
  }
  leaf type {
    type string;
    description "Type of the relationship.";
  }
  leaf target {
    type string;
    description "Target of the relationship.";
  }

  description "Relationship of a node with the other
scenario components.";
}

list lifecycle {
  key "workflow";
  leaf workflow {
    type workflows;
    description "The type of the Workflow.";
  }
  leaf name {
    type string;
    description "The workflow name.";
  }
}

list parameters {
  key "input";
  leaf input {
    type string;
    description "The name of the parameter.";
  }
  leaf value {
    type string;
    description "The value of the parameter";
  }
}
```

```
    }

    description "The list of parameters to be
    applied to the node workflow.";
  }

  leaf-list implementation {
    type string;
    description "The workflow implementation.";
  }

  description "The life cycle workflows to be
  applied to this node.";
}

description "The specification of a node to be used
in a scenario for a benchmarking Test.";
}

grouping link {
  leaf id {
    type string;
    description "The link unique identifier.";
  }
  leaf name {
    type string;
    description "The name of the link.";
  }
  leaf type {
    type string;
    description "The type of the link.";
  }
  leaf network {
    type string;
    description "The network the link belongs to.";
  }
  leaf-list connection_points {
    type leafref {
      path "../../nodes/connection_points/id";
    }
    description "Reference to the connection points of nodes
    the link is adjacent.";
  }
  description "A link between nodes in a scenario.";
}

grouping scenario {
  list nodes {
```

```
    key "id";
    uses nodes;
    description "The list of nodes that must be
instantiated in a scenario in order to enable
a benchmarking Test.";
}

list links {
    key "id";
    uses link;
    description "The list of links among nodes that must be
instantiated in a scenario in order to enable
a benchmarking Test.";
}

list policies {
    key "name";
    leaf name {
        type string;
        description "The name of the policy.";
    }
    leaf type {
        type string;
        description "The type of the policy";
    }
    leaf targets {
        type string;
        description "The targets of the policy.
Uuid of nodes and/or links separated by comma.";
    }
    leaf action {
        type string;
        description "The action of the policy";
    }
}

description "Definition of policies to be
utilized on the instantiation of the scenario.
A policy is defined by a name, it type,
the targets (nodes and/or links) to which it must
be applied to, and the proper action that
realizes the policy.";
}

description "Describes the deployment of all
involved functional components mandatory for
the execution of a benchmarking Test.";
}
```

```
grouping tool {
  leaf id {
    type uint32;
    description "The unique identifier of a tool.
    This information specifies how a tool can be
    identified in a list of probers/listeners of an
    Agent/Monitor.";
  }
  leaf instances {
    type uint32;
    description "The number of the tool instances that
    must be executed in parallel.";
  }
  leaf name {
    type string;
    description "The name of a tool.";
  }
  list parameters {
    key "input";
    leaf input {
      type string;
      description "The input key of a parameter";
    }
    leaf value {
      type string;
      description "The value of a parameter";
    }
    description "List of parameters for the execution
    of the tool. Each tool detains the proper set of running
    parameters that must be utilized to realize a benchmarking
    test.";
  }
}

container sched {
  leaf from {
    type uint32;
    default 0;
    description "The initial time (in seconds)
    of the execution of the tool.";
  }

  leaf until {
    type uint32;
    description "The final/maximum time (in seconds)
    of the execution of the tool summed all its instances
    repeat, duration and interval parameters.";
  }
}
```

```
    leaf duration {
      type uint32;
      description "The total duration (in seconds) of the execution
        of each instance of the tool.";
    }

    leaf interval {
      type uint32;
      description "The interval (in seconds) to be awaited
        among each one of the instances of the
        execution of the tool.";
    }

    leaf repeat {
      type uint32;
      description "The number of times the tool must be executed.";
    }

    description "The scheduling parameters of a tool.
      Each Agent/Monitor must utilize the scheduling parameters
      to perform the execution of its tools (probers/listeners)
      accordingly.";
  }

  description "A tool to be used in a benchmarking test.
    A tool can be inferred as a prober or a listener.";
}

grouping component {
  leaf uuid {
    type string;
    description "A unique identifier";
  }
  leaf name {
    type string;
    description "The name of component";
  }
}

description "A generic component.";
}

grouping agent {
  uses component;

  list probers {
    key "id";
    uses tool;
    description "Defines a list of the Prober(s)";
  }
}
```



```
        that must be used in a benchmarking test.";
    }
    description "An Agent defined by its uuid,
name and the mandatory list of probers to be used
by a benchmarking test.";
}

grouping monitor {
    uses component;

    list listeners {
        key "id";
        uses tool;
        description "Defines a list of the Listeners(s)
that must used in a benchmarking test.";
    }
    description "A Monitor defined by its uuid,
name and the mandatory list of probers to be used
by a benchmarking test.";
}

grouping proceedings {
    list agents {
        key "uuid";
        uses agent;
        description "Defines a list containing the
Agent(s) needed for a VNF-BD test.";
    }

    list monitors {
        key "uuid";
        uses monitor;
        description "Defines a list containing the
Monitor(s) needed for a VNF-BD test.";
    }
    description "Information utilized by a Manager
component to execute a benchmarking test.";
}

grouping vnf-bd {

    container experiments {
        leaf trials {
            type uint32;
            default 1;
            description "Number of trials.
A trial is a single process or iteration
to obtain VNF performance metrics from
```

```
        benchmarking the VNF-BD proceedings.";
    }
    leaf tests {
        type uint32;
        default 1;
        description "Number of tests.
        Each test defines unique structural
        and functional parameters (e.g., configurations,
        resource assignment) for benchmarked components
        to perform one or multiple Trials.
        Each Test must be executed following a
        particular scenario.";
    }
    description "Defines the number of trials and tests
    the VNF-BD must execute.";
}

container scenario {
    uses scenario;
    description "Scenarios defined by this VNF-BD.
    A scenario contains all information needed to describe
    the deployment of all involved functional components
    mandatory for the execution of a benchmarking Test.";
}

container proceedings {
    uses proceedings;
    description "Proceedings of VNF-BD.
    The proceedings are utilized by the Manager component
    to execute a benchmarking Test. It consists of
    agent(s)/monitor(s) settings, detailing their
    prober(s)/listener(s) specification and
    running parameters.";
}

description "A single VNF-BD.
A VNF-BD contains all required definitions and
requirements to deploy, configure, execute, and
reproduce VNF benchmarking tests.";
}

uses vnf-bd;
}
```

Figure 5

## 10.2. VNF Performance Profile

```
module vnf-pp {
  namespace "urn:ietf:params:xml:ns:yang:vnf-pp";
  prefix "vnf-pp";

  organization "IETF/BMWG";
  contact "Raphael Vicente Rosa <raphaelvrosa@gmail.com>,
  Manuel Peuster <peuster@mail.uni-paderborn.de>";

  description "Yang module for a VNF Performance Profile (VNF-PP).";

  revision "2019-10-15" {
    description "Reviewed VNF-PP structure -
    defines reports, snapshots, evaluations";
    reference "";
  }

  revision "2019-08-13" {
    description "V0.1: First release";
    reference "";
  }

  grouping tuple {
    description "A tuple used as key-value.";
    leaf key {
      type string;
      description "Tuple key.";
    }

    leaf value {
      type string;
      description "Tuple value.";
    }
  }

  grouping metric {
    leaf name {
      type string;
      description "The metric name";
    }

    leaf unit {
      type string;
      description "The unit of the metric value(s).";
    }
  }
}
```

```
leaf type {
  type string;
  mandatory true;
  description "The data type encoded in the value.
  It must refer to a known variable type, i.e.,
  string, float, uint, etc.";
}

choice value {
  case scalar {
    leaf scalar {
      type string;
      mandatory true;
      description "A single scalar value.";
    }
  }
  case vector {
    leaf-list vector {
      type string;
      min-elements 1;
      description "A list of scalar values";
    }
  }
  case series {
    list series {
      key "key";
      uses tuple;
      description "A list of key/values,
      e.g., a timeseries.";
    }
  }
}

mandatory true;
description "Value choice: scalar, vector, series.
A metric can only contain a value with one of them.";

description "A metric that holds the recorded benchmarking
results, can be a single value (scalar), a list of values
(vector), or a list of key/value
data (series), e.g., for timeseries.";

}

grouping evaluation {
  leaf id {
    type string;
    description "The evaluation
```

```
        unique identifier.";
    }

    leaf instance {
        type uint32;
        description "The unique identifier of the
parallel instance of the prober/listener that
was executed and created the evaluation.";
    }

    leaf repeat {
        type uint32;
        description "The unique identifier of the
prober/listener repeatition instance
was executed and created the evaluation.";
    }

    container source {

        leaf id {
            type string;
            description "The unique identifier of the source
of the evaluation,
i.e., the prober/listener unique identifier.";
        }

        leaf name {
            type string;
            description "The name of the source of the evaluation,
i.e., the prober/listener name.";
        }

        leaf type {
            type string;
            description "The type of the source of the evaluation,
i.e., one of prober or listener, that was used to obtain
it.";
        }

        leaf version {
            type string;
            description "The version of the tool interfacing
the prober/listener that was used to obtain
the evaluation.";
        }

        leaf call {
            type string;
        }
    }
}
```

```
        description "The full call of the tool realized by
        the source of the evaluation that performed
        the acquisition of the metrics.";
    }

    description "The details regarding the
    source of the evaluation.";
}

container timestamp {

    leaf start {
        type string;
        description "Time (date, hour, minute, second)
        when the evaluation started";
    }

    leaf stop {
        type string;
        description "Time (date, hour, minute, second)
        when the evaluation stopped";
    }

    description "Timestamps of the procedures
    that realized the extraction of the evaluation.";
}

list metrics {
    key "name";
    uses metric;
    description "List of metrics obtained
    from a single evaluation.";
}

leaf error {
    type string;
    description "Error, if existent,
    when obtaining evaluation.";
}

description "The set of metrics and their source
associated with a single Trial.";
}

grouping snapshot {
    leaf id {
        type string;
```

```
    description "The snapshot
    unique identifier.";
}

leaf trial {
    type uint32;
    description "The identifier of the trial
    when the snapshot was obtained.";
}

container origin {

    leaf id {
        type string;
        description "The unique identifier of the
        component of the origin of the snapshot,
        i.e., the agent or monitor unique identifier.";
    }

    leaf role {
        type string;
        description "The role of the component,
        origin of the snapshot, i.e.,
        one of agent or monitor.";
    }

    leaf host {
        type string;
        description "The hostname where the
        source of the snapshot was placed.";
    }

    description "The detailed origin of
    the snapshot.";
}

list evaluations {
    key "id";
    uses evaluation;
    description "The list of evaluations
    contained in a single snapshot Test.";
}

leaf timestamp {
    type string;
    description "Time (date, hour, minute, second)
    when the snapshot was created.";
```

```
    }

    leaf error {
      type string;
      description "Error, if existent,
when obtaining the snapshot.";
    }

    description "The set of evaluations and their origin
output of the execution of a single trial.";
  }

  grouping report {
    leaf id {
      type string;
      description "The report unique identifier.";
    }

    leaf test {
      type uint32;
      description "The identifier of the Test
when the snapshots were obtained.";
    }

    list snapshots {
      key "id";
      uses snapshot;
      description "List of snapshots contained
in a single report.";
    }

    leaf timestamp {
      type string;
      description "Time (date, hour, minute, second)
when the report was created.";
    }

    leaf error {
      type string;
      description "Error, if existent,
when obtaining the report.";
    }

    description "The set of snapshots output
of a single Test.";
  }
```



```
grouping header {
  leaf id {
    type string;
    description "Unique identifier of the VNF-PP.";
  }
  leaf name {
    type string;
    description "Name of the VNF-PP.";
  }
  leaf version {
    type string;
    description "Version of the VNF-PP.";
  }
  leaf description {
    type string;
    description "Description of the VNF-PP.";
  }
  leaf timestamp {
    type string;
    description "Time (date, hour, minute, second)
when the VNF-PP was created.";
  }

  description "The header content of a VNF-PP.";
}

grouping vnf-pp {

  uses header;

  list reports {
    key "id";
    uses report;
    description "List of the reports of a VNF-PP.";
  }

  description "A single VNF-PP.";
}

uses vnf-pp;
}
```

Figure 6

## 10.3. VNF Benchmarking Report

```
module vnf-br {
  namespace "urn:ietf:params:xml:ns:yang:vnf-br";
  prefix "vnf-br";

  import vnf-bd {
    prefix "vnfbd";
    revision-date 2020-10-08;
  }

  import vnf-pp {
    prefix "vnfpp";
    revision-date 2020-10-08;
  }

  organization "IETF/BMWG";
  contact "Raphael Vicente Rosa <raphaelvrosa@gmail.com>,
  Manuel Peuster <peuster@mail.uni-paderborn.de>";
  description "Yang model for a VNF Benchmark Report (VNF-BR).";

  revision "2020-09-09" {
    description "V0.2: Review the structure
    and the grouping/leaf descriptions.";
    reference "";
  }

  revision "2020-09-09" {
    description "V0.1: First release";
    reference "";
  }

  grouping variable {
    leaf name {
      type string;
      description "The name of the variable.";
    }
    leaf path {
      type string;
      description "The VNF-BD YANG path of the
      variable.";
    }
    leaf type {
      type string;
      description "The type of the
      variable values.";
    }
    leaf class {
```

```
        type string;
        description "The class of the
        variable (one of resource, stimulus,
        configuration).";
    }
    leaf-list values {
        type string;
        description "The list of values
        of the variable.";
    }
}

grouping output {
    leaf id {
        type string;
        description "The output unique identifier.";
    }
    list variables {
        key "name";
        leaf name { type string; }
        leaf value { type string; }
        description "The list of instance of variables
        from VNF-BR:inputs utilized by a VNF-BD to
        generate a VNF-PP.";
    }
}

container vnfbd {
    uses vnfbd:vnf-bd;
    description "The VNF-BD that was executed
    to generate a output.";
}

container vnfpp {
    uses vnfpp:vnf-pp;
    description "The output VNF-PP of the
    execution of a VNF-BD.";
}
}

grouping vnf {
    leaf id {
        type string;
        description "The VNF unique identifier.";
    }
    leaf name {
        type string;
        description "The VNF name.";
    }
}
```

```
    leaf version {
      type string;
      description "The VNF version.";
    }
    leaf author {
      type string;
      description "The author of the VNF.";
    }
    leaf description {
      type string;
      description "The description of the VNF.";
    }
    description "The details of the VNF SUT.";
  }

  grouping header {
    leaf id {
      type string;
      description "The unique identifier of the VNF-BR ";
    }
    leaf name {
      type string;
      description "The name of the VNF-BR.";
    }
    leaf version {
      type string;
      description "The VNF-BR version.";
    }
    leaf author {
      type string;
      description "The VNF-BR author.";
    }
    leaf description {
      type string;
      description "The description of the VNF-BR.";
    }
  }

  container vnf {
    uses vnf;
    description "The VNF-BR target SUT VNF.";
  }

  container environment {
    leaf name {
      type string;
      description "The environment name";
    }
    leaf description {
```

```
    type string;
    description "A description
of the environment";
}
leaf deploy {
    type boolean;
    description "Defines if (True) the environment enables
the automated deployment by an orchestrator platform.";
}
container orchestrator {
    leaf name {
        type string;
        description "Name of the orchestrator
platform.";
    }

    leaf type {
        type string;
        description "The type of the orchestrator
platform.";
    }

    leaf description {
        type string;
        description "The description of the
orchestrator platform.";
    }

    list parameters {
        key "input";
        leaf input {
            type string;
            description "The name of the parameter";
        }
        leaf value {
            type string;
            description "The value of the parameter";
        }
    }

    description "List of orchestrator
input parameters.";
}

description "The specification of the orchestration platform
settings of a VNF-BR.";
}

description "The environment settings of a VNF-BR.";
```

```
    }

    description "Defines the content of a VNF-BR header.";
}

grouping vnf-br {
    description "Grouping for a single vnf-br.";

    uses header;

    container inputs {
        list variables {
            key "name";
            uses variable;
            description "The list of
            input variables.";
        }

        container vnfbd {
            uses vnfbd:vnf-bd;
            description "The input VNF-BD.";
        }

        description "The inputs needed to
        realize a VNF-BR.";
    }

    list outputs {
        key "id";
        uses output;
        description "The list of outputs
        of a VNF-BR.";
    }

    container timestamp {
        leaf start {
            type string;
            description "Time (date, hour, minute, second)
            of when the VNF-BR realization started";
        }

        leaf stop {
            type string;
            description "Time (date, hour, minute, second)
            of when the VNF-BR realization stopped";
        }

        description "Timestamps of the procedures that
```

```
    realized the realization of a VNF-BR.";
  }

  leaf error {
    type string;
    description "The VNF-BR error,
if occurred during its realization.";
  }
}

uses vnf-br;
}
```

Figure 7

## 11. Acknowledgement

The authors would like to thank the support of Ericsson Research, Brazil. Parts of this work have received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. H2020-ICT-2016-2 761493 (5GTANGO: <https://5gtango.eu>).

## 12. References

### 12.1. Normative References

- [ETSI14a] ETSI, "Architectural Framework - ETSI GS NFV 002 V1.2.1", Dec 2014, <[http://www.etsi.org/deliver/etsi\\_gs/NFV/001\\_099/002/01.02.01-\\_60/gs\\_NFV002v010201p.pdf](http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01-_60/gs_NFV002v010201p.pdf)>.
- [ETSI14b] ETSI, "Terminology for Main Concepts in NFV - ETSI GS NFV 003 V1.2.1", Dec 2014, <[http://www.etsi.org/deliver/etsi\\_gs/NFV/001\\_099-/003/01.02.01\\_60/gs\\_NFV003v010201p.pdf](http://www.etsi.org/deliver/etsi_gs/NFV/001_099-/003/01.02.01_60/gs_NFV003v010201p.pdf)>.
- [ETSI14c] ETSI, "NFV Pre-deployment Testing - ETSI GS NFV TST001 V1.1.1", April 2016, <[http://docbox.etsi.org/ISG/NFV/Open/DRAFTS/TST001\\_-\\_Pre-deployment\\_Validation/NFV-TST001v0015.zip](http://docbox.etsi.org/ISG/NFV/Open/DRAFTS/TST001_-_Pre-deployment_Validation/NFV-TST001v0015.zip)>.
- [ETSI14d] ETSI, "Network Functions Virtualisation (NFV); Virtual Network Functions Architecture - ETSI GS NFV SWA001 V1.1.1", December 2014, <[https://docbox.etsi.org/ISG/NFV/Open/Publications\\_pdf/Specs-Reports/NFV-SWA%20001v1.1.1%20-%20GS%20-%20Virtual%20Network%20Function%20Architecture.pdf](https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-SWA%20001v1.1.1%20-%20GS%20-%20Virtual%20Network%20Function%20Architecture.pdf)>.

- [ETSI14e] ETSI, "Report on CI/CD and Devops - ETSI GS NFV TST006 V0.0.9", April 2018,  
<[https://docbox.etsi.org/isg/nfv/open/drafts/TST006\\_CICD\\_and\\_Devops\\_report](https://docbox.etsi.org/isg/nfv/open/drafts/TST006_CICD_and_Devops_report)>.
- [ETSI19f] ETSI, "Specification of Networking Benchmarks and Measurement Methods for NFVI - ETSI GS NFV-TST 009 V3.2.1", June 2019,  
<[https://docbox.etsi.org/ISG/NFV/Open/Publications\\_pdf/Specs-Reports/NFV-TST%20009v3.2.1%20-%20GS%20-%20NFVI\\_Benchmarks.pdf](https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-TST%20009v3.2.1%20-%20GS%20-%20NFVI_Benchmarks.pdf)>.
- [ETSI19g] ETSI, "NFVI Compute and Network Metrics Specification - ETSI GS NFV-TST 008 V3.2.1", March 2019,  
<[https://docbox.etsi.org/ISG/NFV/Open/Publications\\_pdf/Specs-Reports/NFV-TST%20008v3.2.1%20-%20GS%20-%20NFVI%20Compute%20and%20Nwk%20Metrics%20-%20Spec.pdf](https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-TST%20008v3.2.1%20-%20GS%20-%20NFVI%20Compute%20and%20Nwk%20Metrics%20-%20Spec.pdf)>.
- [RFC1242] S. Bradner, "Benchmarking Terminology for Network Interconnection Devices", July 1991,  
<<https://www.rfc-editor.org/info/rfc1242>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC8172] A. Morton, "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure", July 2017,  
<<https://www.rfc-editor.org/info/rfc8172>>.
- [RFC8204] M. Tahhan, B. O'Mahony, A. Morton, "Benchmarking Virtual Switches in the Open Platform for NFV (OPNFV)", September 2017, <<https://www.rfc-editor.org/info/rfc8204>>.

## 12.2. Informative References

- [gym] "Gym Framework Source Code",  
<<https://github.com/intrig-unicamp/gym>>.
- [Peu-a] M. Peuster, H. Karl, "Understand Your Chains: Towards Performance Profile-based Network Service Management", Fifth European Workshop on Software Defined Networks (EWSDN) , 2016,  
<<http://ieeexplore.ieee.org/document/7956044>>.



- [Peu-b] M. Peuster, H. Karl, "Profile Your Chains, Not Functions: Automated Network Service Profiling in DevOps Environments", IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) , 2017, <<http://ieeexplore.ieee.org/document/8169826/>>.
- [Peu-c] M. Peuster, H. Karl, "Understand your chains and keep your deadlines: Introducing time-constrained profiling for NFV", IEEE/IFIP 14th International Conference on Network and Service Management (CNSM) , 2018, <<https://ris.uni-paderborn.de/record/6016>>.
- [Peu-d] M. Peuster and S. Schneider and H. Karl, "The Softwarised Network Data Zoo", IEEE/IFIP 15th International Conference on Network and Service Management (CNSM) , 2019, <<https://sndzoo.github.io/>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [Rosa-a] R. V. Rosa, C. E. Rothenberg, R. Szabo, "VBaaS: VNF Benchmark-as-a-Service", Fourth European Workshop on Software Defined Networks , Sept 2015, <<http://ieeexplore.ieee.org/document/7313620>>.
- [Rosa-b] R. Rosa, C. Bertoldo, C. Rothenberg, "Take your VNF to the Gym: A Testing Framework for Automated NFV Performance Benchmarking", IEEE Communications Magazine Testing Series , Sept 2017, <<http://ieeexplore.ieee.org/document/8030496>>.
- [Rosa-c] R. V. Rosa, C. E. Rothenberg, "Taking Open vSwitch to the Gym: An Automated Benchmarking Approach", IV Workshop pre-IETF/IRTF, CSBC Brazil, July 2017, <<https://intrig.dca.fee.unicamp.br/wp-content/plugins/papercite/pdf/rosa2017taking.pdf>>.
- [tango] "5GTANGO: Development and validation platform for global industry-specific network services and apps", <<https://5gtango.eu>>.
- [tng-bench] "5GTANGO VNF/NS Benchmarking Framework", <<https://github.com/sonata-nfv/tng-sdk-benchmark>>.

## Authors' Addresses

Raphael Vicente Rosa (editor)  
University of Campinas  
Av. Albert Einstein, 400  
Campinas, Sao Paulo 13083-852  
Brazil

Email: [rvrosa@dca.fee.unicamp.br](mailto:rvrosa@dca.fee.unicamp.br)  
URI: <https://intrig.dca.fee.unicamp.br/raphaelvrosa/>

Christian Esteve Rothenberg  
University of Campinas  
Av. Albert Einstein, 400  
Campinas, Sao Paulo 13083-852  
Brazil

Email: [chesteve@dca.fee.unicamp.br](mailto:chesteve@dca.fee.unicamp.br)  
URI: <http://www.dca.fee.unicamp.br/~chesteve/>

Manuel Peuster  
Paderborn University  
Warburgerstr. 100  
Paderborn 33098  
Germany

Email: [manuel.peuster@upb.de](mailto:manuel.peuster@upb.de)  
URI: <https://peuster.de>

Holger Karl  
Paderborn University  
Warburgerstr. 100  
Paderborn 33098  
Germany

Email: [holger.karl@upb.de](mailto:holger.karl@upb.de)  
URI: <https://cs.uni-paderborn.de/cn/>

i»¿INTERNET-DRAFT

Kommu	BMWG	S.
Mware	Internet-Draft	V
Rapp	Intended status: Informational	J.
Mware	Expires: Sep 2019	V
2019		Mar 11,

Considerations for Benchmarking Network Virtualization Platforms  
draft-skommu-bmwg-nvp-03.txt

#### Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

This Internet-Draft will expire on September 11, 2019.

#### Copyright Notice

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

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



## Abstract

Current network benchmarking methodologies are focused on physical networking components and do not consider the actual application layer traffic patterns and hence do not reflect the traffic that virtual networking components work with when using network virtualization overlays (NVO3). The purpose of this document is to distinguish and highlight benchmarking considerations when testing and evaluating virtual networking components in the data center.

## Table of Contents

3	1. Introduction .....	
4	2. Conventions used in this document .....	
4	3. Definitions .....	
4	3.1. System Under Test (SUT) .....	
5	3.2. Network Virtualization Platform .....	
6	3.3. Microservices .....	
7	4. Scope .....	
7	4.1.1. Scenario 1 .....	
7	4.1.2. Scenario 2 .....	
7	4.1.3. Learning .....	
7	4.1.4. Flow Optimization .....	
7	4.1.5. Out of scope .....	
8	4.2. Virtual Networking for Datacenter Applications .....	
8	4.3. Interaction with Physical Devices .....	
9	5. NVP Benchmarking Considerations .....	
2	5.1. Learning .....	1
2	5.2. Traffic Flow Optimizations .....	1
2	5.2.1. Fast Path .....	1
2	5.2.2. Dedicated cores / Co-processors .....	1
3	5.2.3. Prioritizing and de-prioritizing active flows ....	1
3	5.3. Server Architecture Considerations .....	1
3	5.3.1. NVE Component considerations .....	1
7	5.3.2. Frame format/sizes within the Hypervisor .....	1
7	5.3.3. Baseline testing with Logical Switch .....	1
7	5.3.4. Repeatability .....	1

7	5.3.5. Tunnel encap/decap outside the Hypervisor .....	1
8	5.3.6. SUT Hypervisor Profile .....	1
0	5.4. Benchmarking Tools Considerations .....	2
0	5.4.1. Considerations for NVE .....	2
0	5.4.2. Considerations for Split-NVE .....	2
0	6. Control Plane Scale Considerations .....	2
1	6.1.1. VM Events .....	2
2	6.1.2. Scale .....	2
2	6.1.3. Control Plane Performance at Scale .....	2

7. Security Considerations .....	2
8. IANA Considerations .....	2
9. Conclusions .....	2
10. References .....	2
10.1. Normative References .....	2
10.2. Informative References .....	2
11. Acknowledgments .....	2
Appendix A. Partial List of Parameters to Document .....	2
A.1. CPU .....	2
A.2. Memory .....	2
A.3. NIC .....	2
A.4. Hypervisor .....	2
A.5. Guest VM .....	2
A.6. Overlay Network Physical Fabric .....	2
A.7. Gateway Network Physical Fabric .....	2
A.8. Metrics .....	2

## 1. Introduction

Datacenter virtualization that includes both compute and network virtualization is growing rapidly as the industry continues to look for ways to improve productivity, flexibility and at the same time cut costs. Network virtualization is comparatively new and expected

to grow tremendously similar to compute virtualization. There are multiple vendors and solutions out in the market. Each vendor often

has their own recommendations on how to benchmark their solutions thus making it difficult to perform an apples-to-apples comparison between different solutions. Hence, the need for a vendor, product and cloud agnostic way to benchmark network virtualization solutions

to help with comparison and make informed decisions when it comes to selecting the right network virtualization solution.

Applications traditionally have been segmented using VLANs and ACLs between the VLANs. This model does not scale because of the 4K scalability

limitations of VLANs. Overlays such as VXLAN were designed to address the limitations of VLANs.

With VXLAN, applications are segmented based on VXLAN encapsulation (specifically the VNI field in the VXLAN header), which is similar

to VLAN ID in the 802.1Q VLAN tag, however without the 4K scale limitations of VLANs. For a more detailed discussion on this subject

ct

please refer RFC 7364 'Problem Statement: Overlays for Network Virtualization'.

VXLAN is just one of several Network Virtualization Overlays (NVO). Some of the others include STT, Geneve and NVGRE. STT and Geneve have expanded on the capabilities of VXLAN. Please refer IETF's nv

o3

Kommu & Rapp

Expires Sep 11, 2019

[Page 3]



r working group < <https://datatracker.ietf.org/wg/nvo3/documents/> > fo  
d more information.

f Modern application architectures, such as Micro-services, because o  
f IP based connectivity within the app, place high demands on the  
networking and security when compared to the traditional three tier  
r app models such as web, app and db. Benchmarks MUST consider whethe  
the proposed solution is able to scale up to the demands of such  
applications and not just a three-tier architecture.

The benchmarks will be utilizing the various terminology and  
definitions from the NVO3 working group including RFC 8014 and RFC  
8394.

## 2. Conventions used in this document

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

## 3. Definitions

### 3.1. System Under Test (SUT)

m Traditional hardware based networking devices generally use the  
device under test (DUT) model of testing. In this model, apart fro  
any allowed configuration, the DUT is a black box from a testing  
es perspective. This method works for hardware based networking devic  
since the device itself is not influenced by any other components  
outside the DUT.

as Virtual networking components cannot leverage DUT model of testing  
the DUT is not just the virtual device but includes the hardware  
components that were used to host the virtual device

Hence System Under Test (SUT) model MUST be used instead of the  
traditional device under test

With SUT model, the virtual networking component along with all  
software and hardware components that host the virtual networking  
component MUST be considered as part of the SUT.

Virtual networking components, because of their dependency on the underlying hardware and other software components, may end up leveraging NIC offload benefits, such as TCP Segmentation Offload (TSO), Large Receive Offload (LRO) and Rx / Tx Filters. Such underlying hardware and software level features, even though they may

not be part of virtual networking stack itself, MUST be considered and documented. Note: Physical switches and routers, including those ones that act as initiators for NVOs, work with L2/L3 packets and may not be able to leverage TCP enhancements such as TSO.

Please refer to section 5 Figure 1 for a visual representation of System Under Test in the case of Intra-Host testing and section 5 Figure 2 for System Under Test in the case of Inter-Host testing.

### 3.2. Network Virtualization Platform

This document focuses on the Network Virtualization Overlay platform as outlined in RFC 8014 and use cases from RFC 8394.

Network Virtualization platforms, function closer to the application layer and are able to work with not only L2/L3 packets but also segments that leverage TCP optimizations such as Large Segment Offload (LSO).

NVPs leverage TCP stack optimizations such as TCP Segmentation Offload (TSO) and Large Receive Offload (LRO) that enables NVPs to work with much larger payloads of up to 64K unlike their counterparts such as NFVs.

Because of the difference in the payload, which translates into one operation per 64K of payload in NVP versus ~40 operations for the same amount of payload in NFV because of having to divide it to MTU sized packets, results in considerable difference in performance between NFV and NVP.

Please refer to figure 1 for a pictorial representation of this primary difference between NPV and NFV for a 64K payload segment/packet running on network set to 1500 bytes MTU.

Note: Payload sizes in figure 1 are approximates.

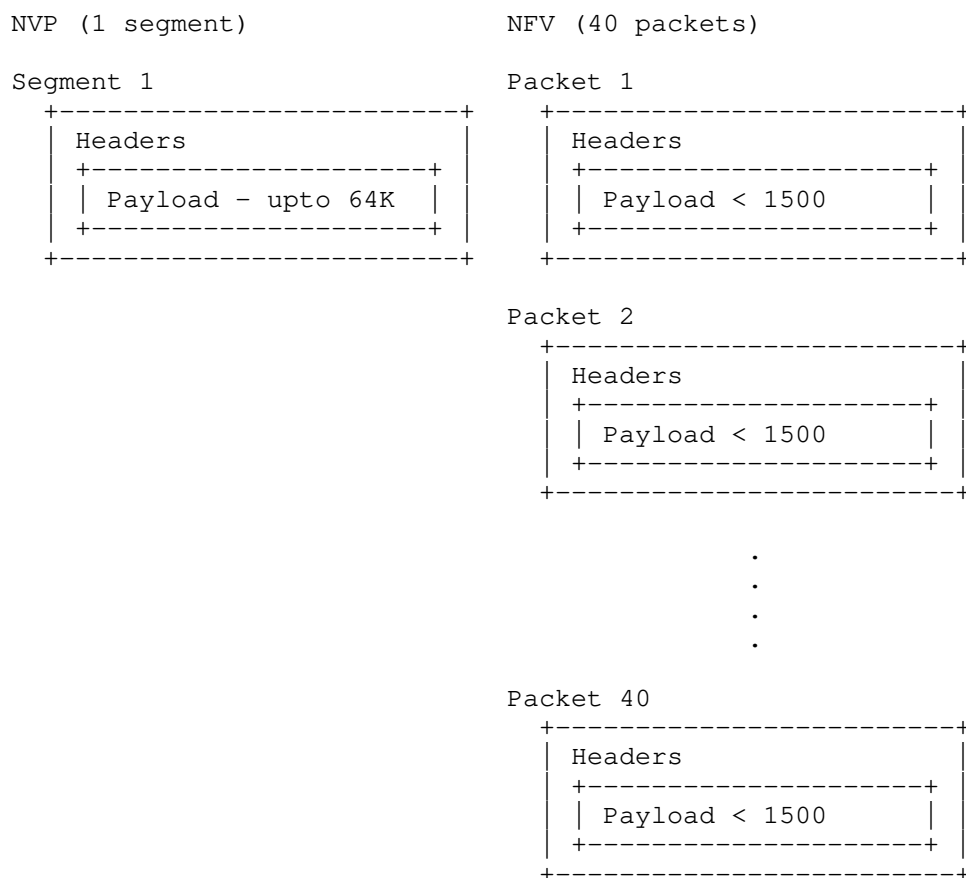


Figure 1! Payload NPV vs NFV

Hence, normal benchmarking methods are not relevant to the NVPs.

Instead, newer methods that leverage TCP optimizations MUST be used for testing Network Virtualization Platforms.

### 3.3. Microservices

Moving from traditional monolithic application architectures such as the three tier web, app and db architectures to microservices model open up networking and security stacks to new scale and performance

related challenges. At a high level, in a microservices model, a traditional monolithic app that may use few IPs is broken down into 100s of individual one-responsibility-only applications where each application has connectivity and security related requirements. These 100s of small one-responsibility-only micro-services need the

own IP and also secured into their own segments, hence pushing the scale boundaries of the overlay from both simple segmentation perspective and also from a security perspective.

For more details regarding microservices, please refer to wiki on microservices: <https://en.wikipedia.org/wiki/Microservices>

#### 4. Scope

Focus of this document is the Network Virtualization Platform in two separate scenarios as outlined in RFC 8014 section 4, Network Virtualization Edge (NVE) and RFC 8394 section 1.1 Split-NVE and the associated learning phase:

##### 4.1.1. Scenario 1

RFC 8014 Section 4.1 "NVE Co-located with server hypervisor": Where the entire NVE functionality will typically be implemented as part of the hypervisor and/or virtual switch on the server.

##### 4.1.2. Scenario 2

RFC 8394 Section 1.1 "Split-NVE: A type of NVE (Network Virtualization Edge) where the functionalities are split across an end device supporting virtualization and an external network device

##### 4.1.3. Learning

Address learning rate is a key contributor to the overall performance of SUT specially in microservices type of use cases where a large amount of end-points are created and destroyed on demand.

##### 4.1.4. Flow Optimization

There are several flow optimization algorithms that are designed to help improve latency or throughput. These optimizations MUST be documented.

##### 4.1.5. Out of scope

This document does not address Network Function Virtualization which has been covered already by previous IETF documents

([https://datatracker.ietf.org/doc/draft-ietf-bmwg-virtual-net/?include\\_text=1](https://datatracker.ietf.org/doc/draft-ietf-bmwg-virtual-net/?include_text=1)).

of Network Function Virtualization (NFV) focuses on being independent networking hardware while providing the same functionality. In the case of NFV, traditional benchmarking methodologies recommended by IETF may be used. Considerations for Benchmarking Virtual Network Functions and Their Infrastructure IETF document addresses benchmarking NFVs.

s Typical NFV implementations emulate in software, the characteristic  
l and features of physical switches. They are similar to any physical L2/L3 switch from the perspective of the packet size, which is typically enforced based on the maximum transmission unit used.

#### 4.2. Virtual Networking for Datacenter Applications

ic This document focuses on the virtual networking for east-west traffic  
er within on-prem datacenter and/or cloud. For example, in a three tier app such web, app and db, this document focuses on the east-west traffic between web and app.

ed This document addresses scale requirements for modern application architectures such as Micro-services to consider whether the proposed solution is able to scale up to the demands of micro-services application models that basically have 100s of small services communicating on some standard ports such as http/https using protocols such as REST.

#### 4.3. Interaction with Physical Devices

a Virtual network components MUST NOT be tested independent of other components within the system. Example, unlike a physical router or firewall, where the tests can be focused solely on the device, when testing a virtual router or firewall, multiple other devices may become part of the SUT. Hence the characteristics of these other traditional networking switches and routers, LB, FW etc. MUST be considered.

- o Hashing method used
- o Over-subscription rate
- o Throughput available
- o Latency characteristics

## 5. NVP Benchmarking Considerations

In virtual environments, the SUT may often share resources and reside on the same physical hardware with other components involved in the tests. Hence SUT MUST be clearly documented. In these tests, a single hypervisor may host multiple servers, switches, routers, firewalls etc.

Intra host testing: Intra host testing helps in reducing the number of components involved in a test. For example, intra host testing would help focus on the System Under Test, logical switch and the hardware that is running the hypervisor that hosts the logical switch, and eliminate other components. Because of the nature of virtual infrastructures and multiple elements being hosted on the same physical infrastructure, influence from other components cannot be completely ruled out. For example, unlike in physical infrastructures, logical routing or distributed firewall MUST NOT be benchmarked independent of logical switching. System Under Test definition MUST include all components involved with that particular test.

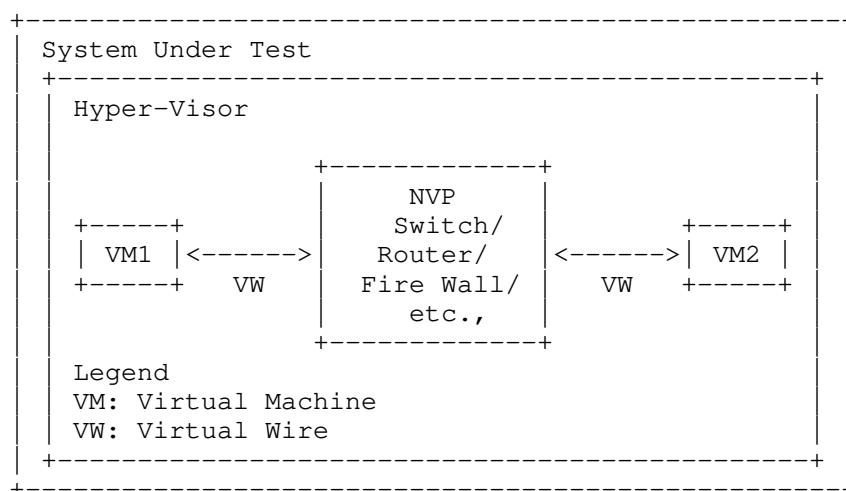


Figure 2! Intra-Host System Under Test

In the above figure, we only address the NVE co-located with the hypervisor.

Inter-host testing: Inter-host testing helps in profiling the underlying network interconnect performance. For example, when testing Logical Switching, inter host testing would not only test t

f logical switch component but also any other devices that are part o  
f the physical data center fabric that connects the two hypervisors.  
f System Under Test MUST be well defined to help with repeatability o  
tests. System Under Test definition in the case of inter host  
testing, MUST include all components, including the underlying  
network fabric.

Figure 2 is a visual representation of system under test for inter-  
host testing.





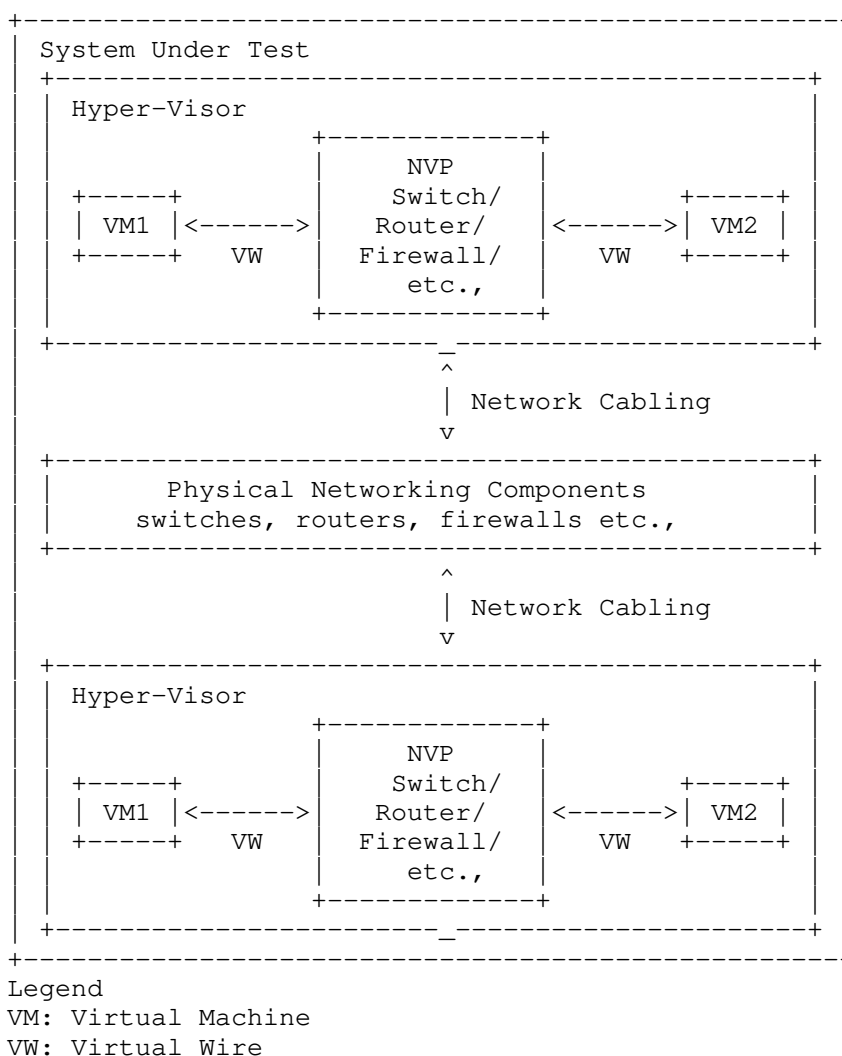


Figure 3! Inter-Host System Under Test

Virtual components have a direct dependency on the physical infrastructure that is hosting these resources. Hardware characteristics of the physical host impact the performance of the virtual components. The components that are being tested and the impact of the other hardware components within the hypervisor on th

performance of the SUT MUST be documented. Virtual component performance is influenced by the physical hardware components withi

n the hypervisor. Access to various offloads such as TCP segmentatio  
offload, may have significant impact on performance. Firmware and  
driver differences may also significantly impact results based on  
whether the specific driver leverages any hardware level offloads  
d offered. Packet processing could be executed on shared or dedicate  
cores on the main processor or via a dedicated co-processor or  
embedded processor on NIC.  
Hence, all physical components of the physical server running the  
ng hypervisor that hosts the virtual components MUST be documented alo  
with the firmware and driver versions of all the components used to  
help ensure repeatability of test results. For example, BIOS  
configuration of the server MUST be documented as some of those  
changes are designed to improve performance. Please refer to  
Appendix A for a partial list of parameters to document.

#### 5.1. Learning

SUT needs to learn all the addresses before running any tests.  
Address learning rate MUST be considered in the overall performance  
metrics because address learning rate has a high impact in  
ts microservices based use cases where there is huge churn of end poin  
as they are created and destroyed on demand. In these cases, both  
the throughput at stable state, and the time taken to get to stable  
state MUST be tested and documented.

#### 5.2. Traffic Flow Optimizations

Several mechanisms are employed to optimize traffic flows. Followi  
ng are some examples:

##### 5.2.1. Fast Path

A single flow may go through various switching, routing and  
firewalling decisions. While in the standard model, every single  
packet has to go through the entire process/pipeline, some  
optimizations help make this decision for the first packet, store t  
he final state for that packet, and leverage it to skip the process fo  
r rest of the packets that are part of the same flow.

##### 5.2.2. Dedicated cores / Co-processors

Packet processing is a CPU intensive workload. Some NVE's may use  
dedicated cores or a co-processor primarily for packet processing  
instead of sharing the cores used for the actual workloads. Such  
cases MUST be documented. Tests MUST be performed with both shared

and dedicated cores. Results and differences in results MUST be documented.

#### 5.2.3. Prioritizing and de-prioritizing active flows

Certain algorithms may prioritize or de-prioritize traffic flows based on purely their network characteristics such as the length of the flow. For example, de-prioritize a long-lived flow. This could

result in changing the performance of a flow over a period of time. Such optimizations MUST be documented, and tests MUST consist of long-lived flows to help capture the change in performance for such flows. Tests MUST note the point at which performance changes.

#### 5.3. Server Architecture Considerations

When testing physical networking components, the approach taken is

to consider the device as a black-box. With virtual infrastructure, this approach would no longer help as the virtual networking components are an intrinsic part of the hypervisor they are running on and are directly impacted by the server architecture used. Server

hardware components define the capabilities of the virtual networki

ng components. Hence, server architecture MUST be documented in detai

l to help with repeatability of tests. And the entire hardware and software components become the SUT.

##### 5.3.1. NVE Component considerations

###### 5.3.1.1. NVE co-located

Components of NVE co-located may be hypervisor based or offloaded entirely to the NIC card or a hybrid model. In the case of hypervisor-based model, they may be running in user space or kernel space. Further, they may use dedicated cores, shared cores or in some cases dedicated co-processors. All the components and the process used MUST be documented.

###### 5.3.1.2. NVE split

NVE split scenario generally has three primary components as documented per RFC 8394.

"tNVE: Terminal-side NVE. The portion of Split-NVE functionalitie

s located on the end device supporting virtualization. The tNVE interacts with a Tenant System through an internal interface in the end device." tNVE may be made of either hypervisor controlled components such as hypervisor provided switches or NVE controlled

components where the network functionality is not provided by the hypervisor. In either case, the components used MUST be documented

"nNVE: Network-side NVE. The portion of Split-NVE functionalities located on the network device that is directly or indirectly connected to the end device that contains the corresponding NVE. Th

nNVE normally performs encapsulation to and decapsulation from the overlay network." All the functionality provided by the nNVE MUST documented.

"External NVE: The physical network device that contains the nNVE. Networking device hardware specs MUST be documented. Please use Appendix A for an example of the specs that MUST be documented.

In either case, NVE co-located or NVE split all the components MUST be documented. Where possible, individual components MUST be tested independent of the entire system. For example, where possible, hypervisor provided switching functionality MUST be tested independent of the NVE.

Per RFC 8014, "For the split-NVE case, protocols will be needed that allow the hypervisor and NVE to negotiate and set up the necessary state so that traffic sent across the access link between a server and the NVE can be associated with the correct virtual network instance." Supported VM lifecycle events, from RFC 8394 section 2, MUST be documented as part of the benchmark process. This process MUST also include how the hypervisor and the external NVE have signaled each other to reach an agreement. Example, see section 2.

of RFC 8394 "VM creation event". The process used to update agreement status MUST also be documented.



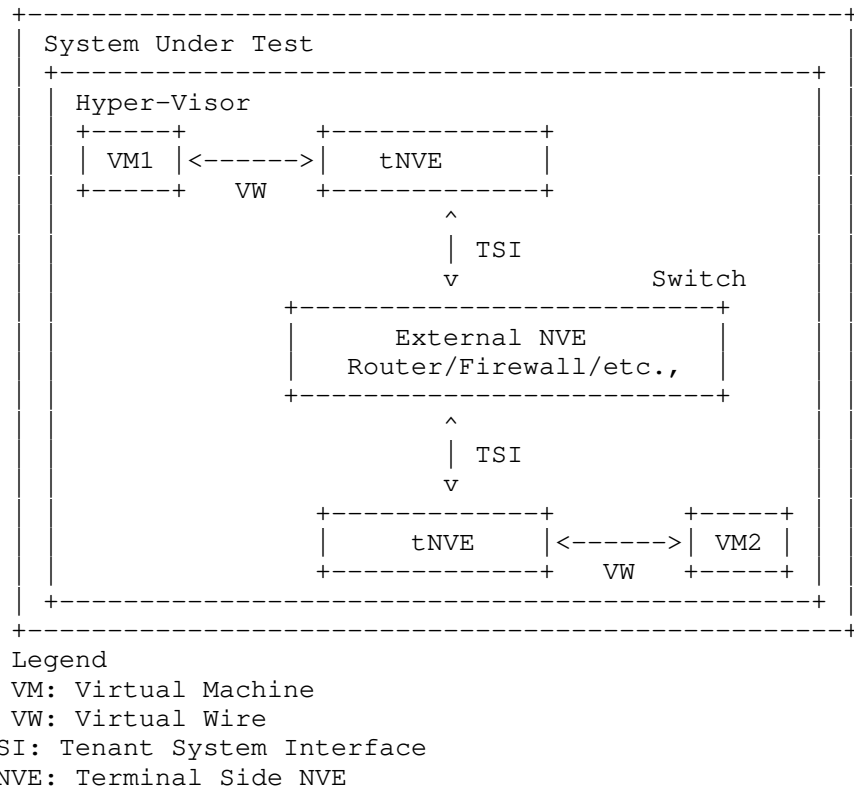


Figure 4 NVE Split collocated - System Under Test

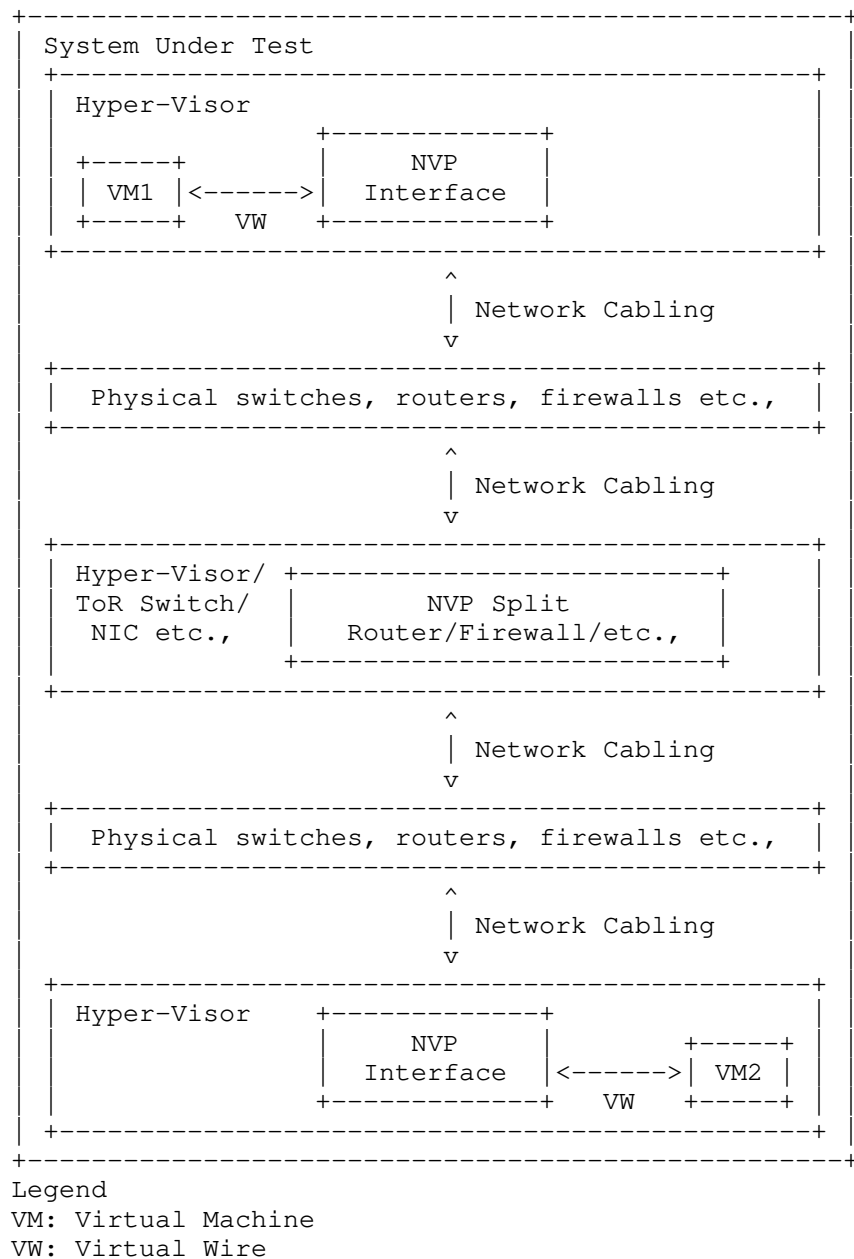


Figure 5 NVE Split not collocated - System Under Test

### 5.3.2. Frame format/sizes within the Hypervisor

Maximum Transmission Unit (MTU) limits physical network component's frame sizes. The most common max supported MTU for physical device is 9000. However, 1500 MTU is the standard. Physical network testing and NFV uses these MTU sizes for testing. However, the virtual networking components that live inside a hypervisor, may work with much larger segments because of the availability of hardware and software based offloads. Hence, the normal smaller packets based testing is not relevant for performance testing of virtual networking components. All the TCP related configuration such as TSO size, number of RSS queues MUST be documented along with any other physical NIC related configuration.

NVE co-located may have a different performance profile when compared with NVE split because, the NVE co-located may have access to offloads that may not be available when the packet has to traverse the physical link. Such differences MUST be documented.

### 5.3.3. Baseline testing with Logical Switch

Logical switch is often an intrinsic component of the test system along with any other hardware and software components used for testing. Also, other logical components cannot be tested independently of the Logical Switch.

### 5.3.4. Repeatability

To ensure repeatability of the results, in the physical network component testing, much care is taken to ensure the tests are conducted with exactly the same parameters. Example parameters such as MAC addresses used.

When testing NVP components with an application layer test tool, there may be a number of components within the system that may not be available to tune or to ensure they maintain a desired state. Example: housekeeping functions of the underlying Operating System.

Hence, tests MUST be repeated a number of times and each test case MUST be run for at least 2 minutes if test tool provides such an option. Results SHOULD be derived from multiple test runs. Variance between the tests SHOULD be documented.

### 5.3.5. Tunnel encap/decap outside the Hypervisor

Logical network components may also have performance impact based on the functionality available within the physical fabric. Physical



a fabric that supports NVO encap/decap is one such case that may have  
on different performance profile. Any such functionality that exists  
o the physical fabric MUST be part of the test result documentation t  
ensure repeatability of tests. In this case SUT MUST include the  
physical fabric if its being used for encap/decap operations.

#### 5.3.6. SUT Hypervisor Profile

of Physical networking equipment has well defined physical resource  
characteristics such as type and number of ASICs/SoCs used, amount

memory, type and number of processors etc., Virtual networking  
components performance is dependent on the physical hardware that  
hosts the hypervisor. Hence the physical hardware usage, which is  
part of SUT, for a given test MUST be documented, for example, CPU  
usage when running logical router.

he CPU usage, changes based on the type of hardware available within t  
physical server. For example, TCP Segmentation Offload greatly  
reduces CPU usage by offloading the segmentation process to the NIC  
it card on the sender side. Receive side scaling offers similar benef  
re on the receive side. Hence, availability and status of such hardwa  
MUST be documented along with actual CPU/Memory usage when the  
virtual networking components have access to such offload capable  
hardware.

Following is a partial list of components that MUST be documented  
both in terms of what is available and also what is used by the SUT

- o CPU - type, speed, available instruction sets (e.g. AES-NI)
- o Memory - type, amount
- o Storage - type, amount
- o NIC Cards -
  - \* Type
  - \* number of ports
  - \* offloads available/used - following is a partial list o  
f possible features
    - o TCP Segmentation Offload
    - o Large Receive Offload

- o Checksum Offloads
  - o Receive Side Scaling
  - o Other Queuing Mechanisms
    - \* drivers, firmware (if applicable)
    - \* HW revision
  - o Libraries such as DPDK if available and used
  - o Number and type of VMs used for testing and
    - \* vCPUs
    - \* RAM
    - \* Storage
    - \* Network Driver
    - \* Any prioritization of VM resources
    - \* Operating System type, version and kernel if applicable
    - \* TCP Configuration Changes - if any
    - \* MTU
  - o Test tool
    - \* Workload type
    - \* Protocol being tested
    - \* Number of threads
    - \* Version of tool
  - o For inter-hypervisor tests,
    - \* Physical network devices that are part of the test
- o Note: For inter-hypervisor tests, system under test is no longer only the virtual component that is being

virtual components become part of the system under test.

#### 5.4. Benchmarking Tools Considerations

##### 5.4.1. Considerations for NVE

Virtual network components in NVE work closer to the application layer than the physical networking components, which enables the virtual network components to take advantage of TCP optimizations such as TCP Segmentation Offload (TSO) and Large Receive Offload (LRO). Because of this optimizations, virtual network components work with type and size of segments that are often not the same type and size that the physical network works with. Hence, testing virtual network components MUST be done with application layer segments instead of the physical network layer packets. Testing MUST be done with application layer testing tools such as iperf, netperf etc.,

##### 5.4.2. Considerations for Split-NVE

In the case of Split-NVE, since they may not leverage any TCP related optimizations, typical network test tools focused on packet processing MUST be used. However, the tools used MUST be able to leverage Receive Side Scaling (RSS).

#### 6. Control Plane Scale Considerations

For a holistic approach to performance testing, control plane performance must also be considered. While the previous sections focused on performance tests after the SUT has come to a steady state, the following section focusses on tests to measure the time taken to bring the SUT to steady state.

In a physical network infrastructure world view, this could be various stages such as boot up time, time taken to apply configuration, BGP convergence time etc., In a virtual infrastructure world, this involves lot more components which may also be distributed across multiple hosts. Some of the components are:

- o VM Creation Event
- o VM Migration Event
- i How many total VMs can the SUT support

- o What is the rate at which the SUT would allow creation of VM

Please refer to section 2 of RFC 8394 for various VM events and the definitions. In the following section we further clarify some of the terms used in the above RFC.

#### VM Creation

For the purposes of NVP control plane testing, VM Creation event is when a VM starts participating for the first time on a NVP provided network. This involves various actions on the tNVE and NVP. Please refer to 2.1 "VM Creation Event" of RFC 8394 for more details.

In order to rule out any Hypervisor imposed limitations, System Under Test must first be profiled and baselined with-out the use of NVP components. For the purposes of baselining control plane, the VM used may have very small footprint such as DSL Linux which runs in 16MB RAM.

Once a baseline has been established for a single HV, a similar exercise MUST be done on multiple HVs to establish a baseline for the entire hypervisor domain. However, it may not be practical to have physical hosts and hence nested hosts may be used for this purpose

#### 6.1.1. VM Events

Performance of various control plane activities which are associated with the System Under Test, MUST BE documented.

- o VM Creation: Time taken to join the VMs to the SUT provided network
- o Policy Realization: Time taken for policy realization on the VM
- o VM Migration: Time taken to migrate a VM from one SUT provided network to another SUT provided network

For the test itself, the following process could be used:

- 1 API to call to join VM on the SUT provided network
- 2 Loop while incrementing a timer - till the VM comes online on the SUT provided network

Similarly, policy realization and VM migration may also be tested with a check on whether the VM is available or not available based on the type of policy that is applied.

#### 6.1.2. Scale

SUT must also be tested to determine the maximum scale supported. Scale can be multi-faceted such as the following:

- o Total # of VMs per Host
- o Total # of VMs per one SUT Domain
- o Total # of Hosts per one SUT Domain
- o Total # of Logical Switches per one SUT Domain
  - \* Total # of VMs per one SUT provided Logical Switch
    - o Per Host
    - o Per SUT Domain
- o Total # of Logical Routers per one SUT Domain
  - \* Total # of Logical Switches per one Logical Router
  - \* Total # of VMs on a single Logical Router
- o Total # of Firewall Sections
- o Total # of Firewall Rules per Section
- o Total # of Firewall Rules applied per VM
- o Total # of Firewall Rules applied per Host
- o Total # of Firewall Rules per SUT

#### 6.1.3. Control Plane Performance at Scale

Benchmarking MUST also test and document the control performance at scale. That is,

- o Total # VMs that can be created in parallel
  - \* How long does the action take
- o Total # of VMs that can be migrated in parallel
  - \* How long does the action take

- o Total amount of time taken to apply 1 firewall across the entire VMs under a SUT
- o Time taken to apply 1000s rules on a SUT

## 7. Security Considerations

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

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

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

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

## 8. IANA Considerations

No IANA Action is requested at this time.

## 9. Conclusions

Network Virtualization Platforms, because of their proximity to the application layer and since they can take advantage of TCP stack optimizations, do not function on packets/sec basis. Hence, traditional benchmarking methods, while still relevant for Network Function Virtualization, are not designed to test Network Virtualization Platforms. Also, advances in application architectures such as micro-services, bring new challenges and need benchmarking not just around throughput and latency but also around scale. New benchmarking methods that are designed to take advantage

of the TCP optimizations or needed to accurately benchmark performance of the Network Virtualization Platforms

## 10. References

### 10.1. Normative References

- [RFC7364] T. Narten, E. Gray, D. Black, L. Fang, L. Kreeger, M. Napierala, 'Problem Statement: Overlays for Network Virtualization', RFC 7364, October 2014, <https://datatracker.ietf.org/doc/rfc7364/>
- [RFC8014] D. Black, J. Hudson, L. Kreeger, M. Lasserre, T. Narten 'Architecture for Data-Center Network Virtualization over Layer 3 (NVO3)', RFC 8014, December 2016, <https://tools.ietf.org/html/rfc8014>
- [RFC8394] Y. Li, D. Eastlake 3rd, L. Kreeger, T. Narten, D. Black 'Split Network Virtualization Edge (Split-NVE) Control Plane Requirements', RFC 8394, May 2018, <https://tools.ietf.org/html/rfc8394>
- [nv03] IETF, WG, Network Virtualization Overlays, <<https://datatracker.ietf.org/wg/nv03/documents/>>

### 10.2. Informative References

- [RFC8172] A. Morton 'Considerations for Benchmarking Virtual Network Functions and Their Infrastructure', RFC 8172, July 2017, <https://tools.ietf.org/html/rfc8172>

## 11. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.





## Appendix A. Partial List of Parameters to Document

### A.1. CPU

- CPU Vendor
- CPU Number
- CPU Architecture
- # of Sockets (CPUs)
- # of Cores
- Clock Speed (GHz)
- Max Turbo Freq. (GHz)
- Cache per CPU (MB)
- # of Memory Channels
- Chipset
- Hyperthreading (BIOS Setting)
- Power Management (BIOS Setting)
- VT-d
- Shared vs Dedicated packet processing
- User space vs Kernel space packet processing

### A.2. Memory

- Memory Speed (MHz)
- DIMM Capacity (GB)
- # of DIMMs
- DIMM configuration
- Total DRAM (GB)

#### A.3. NIC

Vendor  
Model  
Port Speed (Gbps)  
Ports  
PCIe Version  
PCIe Lanes  
Bonded  
Bonding Driver  
Kernel Module Name  
Driver Version  
VXLAN TSO Capable  
VXLAN RSS Capable  
Ring Buffer Size RX  
Ring Buffer Size TX

#### A.4. Hypervisor

Hypervisor Name  
Version/Build  
Based on  
Hotfixes/Patches  
OVS Version/Build  
IRQ balancing  
vCPUs per VM  
Modifications to HV

Modifications to HV TCP stack

Number of VMs

IP MTU

Flow control TX (send pause)

Flow control RX (honor pause)

Encapsulation Type

#### A.5. Guest VM

Guest OS & Version

Modifications to VM

IP MTU Guest VM (Bytes)

Test tool used

Number of NetPerf Instances

Total Number of Streams

Guest RAM (GB)

#### A.6. Overlay Network Physical Fabric

Vendor

Model

# and Type of Ports

Software Release

Interface Configuration

Interface/Ethernet MTU (Bytes)

Flow control TX (send pause)

Flow control RX (honor pause)

#### A.7. Gateway Network Physical Fabric

Vendor

Model

# and Type of Ports

Software Release

Interface Configuration

Interface/Ethernet MTU (Bytes)

Flow control TX (send pause)

Flow control RX (honor pause)

#### A.8. Metrics

Drops on the virtual infrastructure

Drops on the physical underlay infrastructure

#### Authors' Addresses

Samuel Kommu  
VMware  
3401 Hillview Ave  
Palo Alto, CA, 94304  
  
Email: skommu@vmware.com

Jacob Rapp  
VMware  
3401 Hillview Ave  
Palo Alto, CA, 94304  
  
Email: jrapp@vmware.com

Benchmarking Workgroup  
Internet-Draft  
Intended status: Informational  
Expires: May 19, 2018

S. Wu  
Juniper Networks  
November 15, 2017

Network Service Layer Abstract Model  
draft-xwu-bmwg-nslam-00

Abstract

While the networking technologies have evolved over the years, the layered approach has been dominant in many network solutions. Each layer may have multiple interchangeable, competing alternatives that deliver a similar set of functionality. In order to provide an objective benchmarking data among various implementations, the need arises for a common abstract model for each network service layer, with a set of required and optional specifications in respective layers. Many overlay and or underlay solutions can be described using these models.

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 <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on May 19, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
1.1. Terminology . . . . .	3
1.2. Purpose of The Document . . . . .	3
1.3. Conventions Used in This Document . . . . .	3
2. Network Service Framework . . . . .	3
2.1. Node . . . . .	4
2.2. Topology . . . . .	5
2.3. Infrastructure . . . . .	7
2.4. Services . . . . .	8
3. Service Models . . . . .	9
3.1. Layer 2 Ethernet Service Model . . . . .	9
3.2. Layer 3 Service Model . . . . .	10
3.3. Infrastructure Service Model . . . . .	10
3.4. Node Level Features . . . . .	11
3.5. Common Service Specification . . . . .	11
3.6. Comment Network Events . . . . .	12
4. Use of Network Service Layer Abstract Model . . . . .	12
5. Acknowledgements . . . . .	13
6. IANA Considerations . . . . .	13
7. Security Considerations . . . . .	13
8. References . . . . .	13
8.1. Normative References . . . . .	13
8.2. Informative References . . . . .	14
Author's Address . . . . .	14

## 1. Introduction

This document provides a reference model for common network service framework. The main purpose is to abstract service model for each network layer with a small set of key specifications. This is essential to characterize the capability and capacity of a production network, a target network design. A complete service model mainly includes

Infrastructure - devices, links, and other equipment.

Services - network applications provisioned. It is often defined as device configuration and or resource allocation.

Capacity - A set of objects dynamically created for both control and forwarding planes, such as routes, traffic, subscribers and

etc. In some cases, the amount and types of traffic may impact control plane objects, such as multicast or ethernet networks.

Performance Metrics - infrastructure resource utilization.

### 1.1. Terminology

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

### 1.2. Purpose of The Document

Many efforts to YANG model and OpenConfig collaboration are well under way. This document specifies a higher layer abstraction that reuses a small subset of YANG keywords for service description purpose. It SHALL NOT be used for production provisioning purpose. Instead, it can be adopted for design spec, capacity planning, product benchmarking and test setup.

The specification described in this document SHALL be used for outline service requirements from customer perspective, instead of network implementation mechanism from operators perspective.

### 1.3. Conventions Used in This Document

Descriptive terms can quickly become overloaded. For consistency, the following definitions are used.

- o Node - The name for an attribute.
- o Brackets "[" and "]" enclose list keys.
- o Abbreviations before data node names: "rw" means configuration data (read-write), and "ro" means state data (read-only).
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

## 2. Network Service Framework

The network service layer abstract model is illustrated by Figure 1. This shows a stack of components to enable end-to-end services. Not all components are required for a given service. A common use case is to pick one component as target service with a detailed profile, with the remaining components as supporting technologies using default profiles.

## Network Service Layer Abstract Model

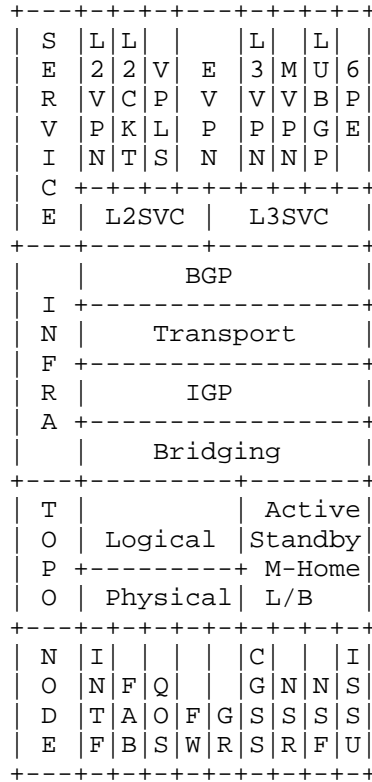


Figure 1

## 2.1. Node

A network node or a network device processes and forwards traffic based on predefined or dynamically learned policies. This section covers standalone features like the following:

- o INTF - Network interfaces that provides internal or external connectivity to adjacent devices. Only physical properties of the interfaces are of concern in this level. The interfaces can be physical or logical, wired or wireless.
- o FAB - Fabric capacity. It provides redundancy and cross connect within the same network device among various linecards or interfaces



- o QOS - Quality of Services. Traffic queuing, buffering, and congestion management technologies are used in this level
- o FW - Firewall filters or access control list. This commonly refers to stateless packet inspection and filtering. Stateful firewall is out of scope of this document. Number of filters daisy chained on a given protocol family, number of terms within same filter, and depth of packet inspection can all affect speed and latency of traffic forwarding. It also provides necessary security protection of node, where protocol traffic may be affected.
- o GR - Graceful Restart per protocol. It needs to cooperate with adjacent node
- o CGSS - Controller Graceful / Stateful Switchover. A network device often has two redundant controllers to minimize the disruption in event of catastrophic failure on the primary controller. This is accomplished via real time state synchronization from the primary to the backup controller. It, however, should be used along with either NSR or NSF to achieve optimal redundancy.
- o NSR - Non-Stop Routing - hitless failover of route processor. It maintains an active copy of route information base (RIB) as well as state for protocol exchange so that the protocol adjacency is not reset.
- o NSF - Non-Stop Forwarding for layer 2 traffic, including layer 2 protocols such as spanning tree state
- o ISSU - In Service Software Upgrade - Sub-second traffic loss in many modern routing platforms. The demand for this feature continues to grow from the field. Some study shows that the downtime due to software upgrade is greater than that caused by unplanned outages.

## 2.2. Topology

Placement of network devices and corresponding links plays an important role for optimal traffic forwarding. There are two types of topologies:

- o Physical Topology - Actual physical connectivity via fiber, coax, cat5 or even wireless. That could be a ring, bus, star or matrix topology. Even though all can be modeled using point-to-point connections.

- o Logical Topology - With aggregated ethernet, extended dot1q tunneling, or VxLAN, a logical or virtual topology can be easily created spanning across geography boundaries. Recent development of multi-chassis, virtual-chassis, node-slicing technologies, and multiple logical units within a single physical node have enabled logical topology deployment more flexible and agile.

With various topology, the following functionalities need to be taken into consideration for feature design and validation.

- o Active-Standby - 1:1 or 1:n support. The liveness detection is essential to trigger failover.
- o M-Home - Multi-homing support. A customer edge (CE) device can be homed to 2 or more Provider Edge (PE) devices at the same time. This is a common redundancy design in layer 2 service offering
- o L/B - Load Balancing - When multiple diverse paths exist for a given destination, it is important to achieve load balancing based on multiple criteria, such as per packet, or per prefix. Sometimes, cascading effect can make issues more complex and harder to resolve

The topology, regardless of physical or virtual, can be better depicted using a collection of nodes and point-to-point links. Some broadcast network, or ring topology, can also be abstracted using same collection of point-to-point links. For example, in a wireless LAN network, each client is a node with wireless LAN NIC as its physical interface. The access point is the node, at which all WLAN clients terminate with airwaves. The Service Set Identifier (SSID) on this access point can be considered as part of broadcast domain with many pseudo-ports taking airwave terminations from clients.

The default link id can use srcnode-dstnode-linkseq to uniquely identify a link in this topology. If this is a link connecting two ports on the same node, it can use link-id of srcnode-srcnode-linkseq-portseq. Additional attributes of the node can be added with proper placement for auto topology diagram.

## Network Topology Definition

```

node-id-1 {
  maker: maker_name,
  model: model_name,
  controller: controller-type,
  mgmt_ip: mgmt_ip_address,
  links: {
    link-id-1 {
      name: link_name,
      connector: 'sfpp',
      attr: ['10G', 'Ethernet'],
      node_dst: destination node-id,
      link_seq: sequence number for links between the node pair
      ...
    }
    ...
  }
}
node-id-2 {
  ...
}

```

Figure 2

## 2.3. Infrastructure

Network infrastructure here refers to a list of protocols and policies for a data center network, an enterprise network, or a core backbone in a service provider network.

- o Bridging - Spanning Tree Protocol (STP) and its various flavors, 802.1q tunneling, Q-in-Q, VRRP and etc
- o IGP - Interior Gateway Protocol - some common choices are OSPF, IS-IS, RIP, RIPng. For multicast, choices are PIM and its various flavors including MSDP, Bootstrap, DVMRP
- o Transport - Tunnel technologies including
  - \* MPLS - Multi-Protocol Label Switching - most commonly used in service provider network
  - + LDP - Label distribution protocol - including mLDP and LDP Tunneling through RSVP LSPs
  - + RSVP - Resource Reservation Protocol - including P2MP and its various features like Fast ReRoute - FRR.

- \* IPsec - IPsec Tunnel with AH or ESP
- \* GRE - Generic Routing Encapsulation (GRE) tunnels provides a flexible direct adjacency between two remote routers
- \* VxLAN - In data center interconnect (DCI) solutions, VxLAN encapsulation provides data plane for layer 2 frames
- o BGP - Define families and their sub-SAFI deployed, as well as route reflector topology.

## 2.4. Services

Previous sections mostly outline an operator's implementation of the network, while customers may not necessarily care about these. This section defines service profiles from customer's view.

- o Layer 2 Services
  - \* Layer 2 VPN - RFC 6624
  - \* Martini Layer 2 Circuit - RFC 4906
  - \* Virtual Private LAN Services - RFC 4761
  - \* Ethernet VPN - RFC 7432
- o Layer 3 Services
  - \* Type 5 Route for EVPN - draft-ietf-bess-evpn-prefix-advertisement-05
  - \* Layer 3 VPN - RFC 4364
  - \* Labeled Unicast BGP - RFC 3107
  - \* Draft Rosen MVPN - RFC 6037
  - \* NG MVPN - RFC 6513
  - \* 6PE - RFC 4798

In next section, an abstract model is proposed to identify key metrics for both layer 2 and layer 3 model

### 3. Service Models

A service model is a high level abstraction of network deployment from bottom up. It defines a set of common key characteristics of customer traffic profile in both control and forwarding planes. The network itself should be considered as a blackbox and deliver the services regardless of types of network equipment vendor or network technologies.

The abstraction removes some details like specific IP address assignment, and favors address range and its distribution. The goal is to describe aggregated network behavior instead of granular network element configuration. It is up to implementation to map aggregated metrics to actual configuration for the network devices, protocol emulator and traffic generator.

A single network may be comprised of multiple instances of service models defined below.

#### 3.1. Layer 2 Ethernet Service Model

The metrics outlined below are for layer 2 network services typically within a data center, data center interconnect, metro ethernet, or layer 2 domain over WAN or even inter-carrier.

- o service-type: identityref, ELAN, ELINE, ETREE
- o sites-per-instance: uint32, an average number of sites a layer 2 instance may span across
- o global-mac-count: uint32, Global MAC from all attachment circuits, local and remote. This is probably the most important metric that determines the capacity requirements in layer 2 for both control and forwarding planes
- o interface-mac-max: uint32, maximum number of locally learned MAC addresses per logical interface, aka attachment circuit
- o single-home-segments: uint32, number of single homed ethernet segments per service instance
- o multi-home-segments: uint32, number of multi homed ethernet segments per layer 2 service instance
- o service-instance-count: uint32, total number of layer 2 service instances. Typically, one customer is
- o traffic-type: list, {known-unicast: %, multicast: %, broadcast: %, unknown-unicast: %}
- o traffic-frame-size: list, predefined mixture of traffic frame size distribution
- o traffic-load: speed of traffic being sent towards the network. This can be defined as frame per second (fps), or actual speed in bps. This is particularly important whenever some component along

- forwarding path is implemented in software, the throughput might be affected significantly at high speed
- o traffic-flow: A distribution of flows. This may affect efficient use of load-balancing techniques and resource consumption. More details discussed in later section of this document.
  - o layer3-gateway-count: uint32, number of layer 2 service instances that also provide layer 3 gateway service
  - o arnpnp-table-size: uint32. This is only relevant with presence of layer 3 gateway

Integrated routing and bridging (IRB) and EVPN Type 5 route have blurred boundaries between layer 2 and layer 3 services.

### 3.2. Layer 3 Service Model

This section outlines traffic type, layer 3 protocol families, layer 3 prefixes distribution, layer 3 traffic flow and packet size distributions.

- o proto-family: protocol family are defined with three sub-attributes. The list may grow as the complexity
  - \* proto - list: inet, inet6, iso
  - \* type - list: unicast, mcast, segment, labeled
  - \* vpn - list, true, false
- o prefix-count, uint32, total unique prefixes
- o prefix-distrib, list of prefix length size and percentage. This could be a distribution pattern, such as uniform, random. Or simply top representation of prefix lengths
- o bgp-path-count, uint32, total BGP paths
- o bgp-path-distrib, top representation of number of paths per prefix
- o traffic-frame-size, similar to traffic-frame-size in layer 2 model. The focus is on the MTU size on each protocol interfaces and the impact of fragmentation
- o traffic-flow, similar to traffic-flow in layer 2 model, it focuses on a set of labels, source and destination addresses as well as ports
- o traffic-load, similar to traffic-load in layer 2 model
- o ifl-count, uint32,
- o vpn-count, uint32,

### 3.3. Infrastructure Service Model

- o bgp-peer-ext-count, uint32, number of eBGP peers
- o bgp-peer-int-count, uint32, number of iBGP peers
- o bgp-path-mtu, list, true or false. Larger path mtu helps convergence

- o bgp-hold-time-distrib, list of top hold-time values and their respective percentage out of all peers.
- o bgp-as-path-distrib, list of top as-path lengths and their respective percentage of all BGP paths
- o bgp-community-distrib, list of top community size and their respective percentage out of all BGP paths
- o mpls-sig, list, MPLS signaling protocol, rsvp or ldp
- o rsvp-lsp-count-ingress, uint32, total ingress lsp count
- o rsvp-lsp-count-transit, uint32, total transit lsp count
- o rsvp-lsp-count-egress, uint32, total egress lsp count
- o ldp-fec-count, uint32, total forwarding equivalence class
- o rsvp-lsp-protection, list, link-node, link, frr
- o ospf-interface-type, list, point-to-point, broadcast, non-broadcast multi-access
- o ospf-lsa-distrib, list. OSPF Link Statement Advertisement distribution is comprised of those for core router in backbone area, and internal router in non-Backbone areas. A common modeling can include number of LSAs per OSPF LSA type
- o ospf-route-count, list, total OSPF routes in both backbone and non-backbone areas
- o isis-lsp-distrib, list, similar to ospf-lsa-distrib
- o isis-route-count, list, total IS-IS routes in both level-1 and level-2 areas

TODO: bridging, OAM, EOAM, BFD and etc

### 3.4. Node Level Features

TODO: node level feature set

### 3.5. Common Service Specification

For most network services, regardless of layer 2 or layer 3, protocol families, the following needs to be considered when measuring network capacity and baseline.

- o rib-learning-time, uint32 in seconds. This indicates how quickly the route processor learns routing objects either locally and remotely
- o fib-learning-time. In large routing system, forwarding engine residing on separate hardware from controller, takes additional time to install all forwarding entries learned by controller.
- o convergence-time, this is could be as a result of many events, such as uplink failure, ae member link failure, fast reroute, local repair, upstream node failure and etc
- o multihome-failover-time, this refers to traffic convergence in a topology where a customer edge (CE) device is connected to two or more provider edge (PE) devices.

- o issu-dark-window-size. Unlike NSR, the goal of ISSU is not zero packet loss. Instead, there will be a few seconds, or in some cases, sub-second dark window where it sees both total packet loss for both transit and or host bound protocol traffic.
- o cpu-util, total CPU utilization of the controllers in stead state
- o cpu-util-peak, Peak CPU utilization on the controller in event of failure, and convergence
- o mem-util, total memory utilization of the controllers in steady state
- o mem-util-peak, total memory utilization on the controller in event of failure and convergence
- o processes, list of top processes running on the controllers with their CPU and memory utilization.
- o lc-cpu-util, top CPU utilization on the line cards
- o lc-cpu-util-peak, maximum peak CPU utilization among all line cards in event of failure and convergence
- o lc-mem-util, top memory utilization on the line cards
- o lc-mem-util-peak, maximum peak memory utilization among all line cards in event of failure and convergence
- o throughout, in both pps and bps. This is measured with zero packet loss. For virtualized environment, throughput is sometimes measured with a small loss tolerance given the nature of shared resource
- o traffic performance, in both pps and bps. It is measured the rate of traffic received by pumping oversubscribed traffic at ingress
- o latency in us. this is more important within a local data center environment rather than DCI over wide area network. Use of extensive firewall filter or access control lists may affect latency
- o Out of Order Packet - This can happen in intra-node or over ECMP where different paths have large latency/delay variations.

The list of metrics can be used for network monitoring during network resiliency test. This is to understand how quickly a network service can restore during various events and failures

### 3.6. Comment Network Events

TODO: A list of events to be defined to characterize network resiliency. These attributes require the provider networks have diverse paths and node redundancy built-in. These numbers affect service level agreement and network availability

## 4. Use of Network Service Layer Abstract Model

The primary goal is to characterize and document a complex network using a simplified service model. While eliminating many details such as address assignment, actual route or mac entries, it retains a



set of key network information, including services, scale, and performance profiles. This can be used to validate how well each underlying solution performs when delivering same set of services.

The model can also be used to build a virtualized topology with both static and dynamic scale closely resemble to a real network. This eases network design and benchmarking, and helps capacity planning by studying the impact with changes to a specific dimension.

## 5. Acknowledgements

The authors appreciate and acknowledge comments from Al Morton and others based on initial discussions.

## 6. IANA Considerations

This memo includes no request to IANA.

All drafts are required to have an IANA considerations section (see Guidelines for Writing an IANA Considerations Section in RFCs [RFC5226] for a guide). If the draft does not require IANA to do anything, the section contains an explicit statement that this is the case (as above). If there are no requirements for IANA, the section will be removed during conversion into an RFC by the RFC Editor.

## 7. Security Considerations

All drafts are required to have a security considerations section. See RFC 3552 [RFC3552] for a guide.

## 8. References

### 8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <<https://www.rfc-editor.org/info/rfc3552>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <<https://www.rfc-editor.org/info/rfc5226>>.

## 8.2. Informative References

- [L2SM] B. Wu et al, "A Yang Data Model for L2VPN Service Delivery", 2017, <<https://www.ietf.org/id/draft-ietf-l2sm-l2vpn-service-model-02.txt>>.
- [RFC8049] Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8049, DOI 10.17487/RFC8049, February 2017, <<https://www.rfc-editor.org/info/rfc8049>>.

## Author's Address

Sean Wu  
Juniper Networks  
2251 Corporate Park Dr.  
Suite #200  
Herndon, VA 20171  
US

Phone: +1 571 203 1898  
Email: [xwu@juniper.net](mailto:xwu@juniper.net)