Benchmarking Methodology Working Group
Internet-Draft
Intended status: Informational
Expires: September 6, 2018

B. Balarajah
C. Rossenhoevel
EANTC AG

March 5, 2018

# Benchmarking Methodology for Network Security Device Performance
draft-balarajah-bmwg-ngfw-performance-02

## 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

## Copyright Notice

Table of Contents

## 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 devise 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.

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

Figure 1: Testbed Setup - Option 1

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

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 | AD | 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 though 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 t 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 recommend 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 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

    + iThe 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

    + iAttributes 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 detains 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.

- TLS Handshake Rate
  This key performance indicator will measure the average TLS 1.2 or higher session formation rate within the sustaining period.

- 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.

- 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.

- 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.

- 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 date from the DUT/SUT. TTFB SHOULD be expressed in millisecond.

- 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

To determine the average throughput performance of the DUT/SUT when using application traffic mix defined in Section 7.1.3.3.

#### 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. 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 finalyzed and updated in future draft release)
The following equation MUST be used to calculate the deviation of application transaction time or TTLB.

   max. dev = max((avg_latency - min_latency),(max_latency - avg_latency)) / (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))

d. Maximum value of TCP connect time must be less than Xms (The value for "X" will be finalyzed 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 algorithmn can be implemented using the parameter; Resolution =0.01* 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 finalyzed 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 finalyzed and updated in future draft release).
The following equation MUST be used to calculate the deviation of application transaction time or TTLB.

    max. dev = max((avg_latency - min_latency),(max_latency -
    avg_latency)) / (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 finalyzed 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 finalyzed 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 algorithmn can be implemented using the parameter; Resolution =0.01* "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 test procedures will be added in the future release.

### 7.4. HTTP Transactions Per Second

#### 7.4.1. Objective

Determine maximum and average HTTP transacton 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

Determine the maximum and average SSL/TLS handshake rate supported by the DUT/SUT.

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

## 7.9. HTTPS Transaction Per Second

### 7.9.1. Objective

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

Test parameters and test 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 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 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. An Appendix

Details about NetSecOPEN traffic mix will be added in next draft release.

## 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 L. Huang, Ed.
Internet-Draft R. Gu, Ed.
Intended status: Informational China Mobile
Expires: September 11, 2017 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

## Copyright Notice

Table of Contents

## 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 encironment 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.

```
                           ________________________________________
                          |                                        |
   -----------------      |   ----------------    ---------------- |
   |Physical tester|------|---|DUT (VSWITCH) |----|Virtual tester| |
   -----------------      |   ----------------    ---------------- |
                          |  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.

```
    ______________________________________________________________
   |                                                              |
   |   ----------------    ----------------     ----------------  |
   |   |Virtual tester|----|DUT (VSWITCH) |-----|Virtual tester|  |
   |   ----------------    ----------------     ----------------  |
   |  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 | Maxmum 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 |
|---|---|---|
| | 50% of NIC speed | |
| 64 | 75% | |
| | 90% | |
| | 50% of NIC speed | |
| 128 | 75% | |
| | 90% | |
| ~ | ~ | ~ |
| | 50% of NIC speed | |
| 1500 | 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 |
|---|---|---|
| | 50% of NIC speed | |
| 64 | 75% | |
| | 90% | |
| | 50% of NIC speed | |
| 128 | 75% | |
| | 90% | |
| ~ | ~ | ~ |
| | 50% of NIC speed | |
| 1500 | 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 vitual 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.

```
                          ________________________________________
                         |                                        |
   -----------------     |   ----------------   ----------------  |
   |Physical tester|------|---|DUT (VSWITCH) |---|  echo server |  |
   -----------------     |   ----------------   ----------------  |
                         |  Server                                |
                         |________________________________________|
```

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

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

Email: gurong@chinamobile.com

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

Email: bob@iometrix.com

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

Email: Brooks.Hickman@spirent.com

Internet-Draft
Network Working Group
Intended Status: Informational
Expires: August 25, 2018

Bhuvaneswaran Vengainathan
Anton Basil
Veryx Technologies
Mark Tassinari
Hewlett-Packard
Vishwas Manral
Nano Sec
Sarah Banks
VSS Monitoring
February 25, 2018

# Benchmarking Methodology for SDN Controller Performance
draft-ietf-bmwg-sdn-controller-benchmark-meth-08

## Abstract

This document defines the methodologies for benchmarking control plane performance of SDN controllers. SDN controller is a core component in software-defined networking architecture that controls the network behavior. Terminology related to benchmarking SDN controllers is described in the companion terminology documentI-D sdn-controller-benchmark-term. SDN controllers have been implemented with many varying designs in order to achieve their intended network functionality. Hence, the authors have taken the approach of considering an SDN controller as a black box, defining the methodology in a manner that is agnostic to protocols and network services supported by controllers. The intent of this document is to provide a standard mechanism to measure the performance of all controller implementations.

## Status of this Memo

Table of Contents

## 1. Introduction

This document provides generic methodologies for benchmarking SDN controller performance. An SDN controller may support many northbound and southbound protocols, implement a wide range of applications, and work solely, or as a group to achieve the desired functionality. This document considers an SDN controller as a black box, regardless of design and implementation. The tests defined in the document can be used to benchmark SDN controller for performance, scalability, reliability and security independent of northbound and southbound protocols. These tests can be performed on an SDN controller running as a virtual machine (VM) instance or on a bare metal server.  This document is intended for those who want to measure the SDN controller performance as well as compare various SDN controllers performance.

Conventions used in this document

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.

## 2. Scope

This document defines methodology to measure the networking metrics of SDN controllers. For the purpose of this memo, the SDN controller is a function that manages and controls Network Devices. Any SDN controller without a control capability is out of scope for this memo. The tests defined in this document enable benchmarking of SDN Controllers in two ways; as a standalone controller and as a cluster of homogeneous controllers. These tests are recommended for execution in lab environments rather than in live network deployments. Performance benchmarking of a federation of controllers, set of SDN controllers managing different domains, is beyond the scope of this document.

## 3. Test Setup

The tests defined in this document enable measurement of an SDN controller’s performance in standalone mode and cluster mode. This section defines common reference topologies that are later referred to in individual tests.

## 3.1. Test setup - Controller working in Standalone Mode

```
          +-----------------------------------------------------------+
          |               Application Plane Test Emulator             |
          |                                                           |
          |        +-----------------+      +-------------+           |
          |        |   Application   |      |   Service   |           |
          |        +-----------------+      +-------------+           |
          |                                                           |
          +-----------------------------+(I2)-------------------------+
                                        |
                                        | (Northbound interfaces)
                       +-------------------------------+
                       |       +----------------+      |
                       |       | SDN Controller |      |
                       |       +----------------+      |
                       |                               |
                       |    Device Under Test (DUT)    |
                       +-------------------------------+
                                        | (Southbound interfaces)
                                        |
          +-----------------------------+(I1)-------------------------+
          |                                                           |
          |             +-----------+     +-----------+               |
          |             |  Network  |     |  Network  |               |
          |             | Device 2  |--..-| Device n-1|               |
          |             +-----------+     +-----------+               |
          |                     /    \   /    \                       |
          |                    /      \ /      \                      |
          |                l0 /        X        \ ln                  |
          |                  /        / \        \                    |
          |               +-----------+  +-----------+                |
          |               |  Network  |  |  Network  |                |
          |               |  Device 1 |..|  Device n |                |
          |               +-----------+  +-----------+                |
          |                     |              |                      |
          |           +---------------+  +---------------+            |
          |           | Test Traffic  |  | Test Traffic  |            |
          |           |  Generator    |  |  Generator    |            |
          |           |    (TP1)      |  |    (TP2)      |            |
          |           +---------------+  +---------------+            |
          |                                                           |
          |              Forwarding Plane Test Emulator               |
          +-----------------------------------------------------------+
```

Figure 1

## 3.2. Test setup - Controller working in Cluster Mode

```
          +-----------------------------------------------------------+
          |               Application Plane Test Emulator             |
          |                                                           |
          |        +-----------------+      +-------------+           |
          |        |   Application   |      |   Service   |           |
          |        +-----------------+      +-------------+           |
          |                                                           |
          +-----------------------------+(I2)-------------------------+
                                        |
                                        | (Northbound interfaces)
           +-------------------------------------------------------+
           |                                                       |
           |  ------------------             ------------------      |
           | | SDN Controller 1 | <--E/W--> | SDN Controller n |     |
           |  ------------------             ------------------      |
           |                                                       |
           |                    Device Under Test (DUT)              |
           +-------------------------------------------------------+
                                        | (Southbound interfaces)
                                        |
          +-----------------------------+(I1)-------------------------+
          |                                                           |
          |             +-----------+     +-----------+               |
          |             |  Network  |     |  Network  |               |
          |             | Device 2  |--..-| Device n-1|               |
          |             +-----------+     +-----------+               |
          |                     /    \   /    \                       |
          |                    /      \ /      \                      |
          |                l0 /        X        \ ln                  |
          |                  /        / \        \                    |
          |               +-----------+  +-----------+                |
          |               |  Network  |  |  Network  |                |
          |               |  Device 1 |..|  Device n |                |
          |               +-----------+  +-----------+                |
          |                     |              |                      |
          |           +---------------+  +---------------+            |
          |           | Test Traffic  |  | Test Traffic  |            |
          |           |  Generator    |  |  Generator    |            |
          |           |    (TP1)      |  |    (TP2)      |            |
          |           +---------------+  +---------------+            |
          |                                                           |
          |              Forwarding Plane Test Emulator               |
          +-----------------------------------------------------------+
```

Figure 2

## 4. Test Considerations

### 4.1. Network Topology

The test cases SHOULD use Leaf-Spine topology with at least 1 Network Device in the topology for benchmarking. The test traffic generators TP1 and TP2 SHOULD be connected to the first and the last leaf Network Device. If a test case uses test topology with 1 Network Device, the test traffic generators TP1 and TP2 SHOULD be connected to the same node. However to achieve a complete performance characterization of the SDN controller, it is recommended that the controller be benchmarked for many network topologies and a varying number of Network Devices. This document includes a sample test topology, defined in Section 10 - Appendix A for reference. Further, care should be taken to make sure that a loop prevention mechanism is enabled either in the SDN controller, or in the network when the topology contains redundant network paths.

### 4.2. Test Traffic

Test traffic is used to notify the controller about the asynchronous arrival of new flows. The test cases SHOULD use frame sizes of 128, 512 and 1508 bytes for benchmarking. Tests using jumbo frames are optional.

### 4.3. Test Emulator Requirements

The Test Emulator SHOULD time stamp the transmitted and received control messages to/from the controller on the established network connections. The test cases use these values to compute the controller processing time.

### 4.4. Connection Setup

There may be controller implementations that support unencrypted and encrypted network connections with Network Devices. Further, the controller may have backward compatibility with Network Devices running older versions of southbound protocols. It may be useful to measure the controller performance with one or more applicable connection setup methods defined below. For cases with encrypted communications between the controller and the switch, key management and key exchange MUST take place before any performance or benchmark measurements.

1. Unencrypted connection with Network Devices, running same protocol version.

2. Unencrypted connection with Network Devices, running different protocol versions.
   Example:
   a. Controller running current protocol version and switch running older protocol version
   b. Controller running older protocol version and switch running current protocol version
3. Encrypted connection with Network Devices, running same protocol version
4. Encrypted connection with Network Devices, running different protocol versions.
   Example:
   a. Controller running current protocol version and switch running older protocol version
   b. Controller running older protocol version and switch running current protocol version

## 4.5. Measurement Point Specification and Recommendation

The measurement accuracy depends on several factors including the point of observation where the indications are captured. For example, the notification can be observed at the controller or test emulator. The test operator SHOULD make the observations/ measurements at the interfaces of test emulator unless it is explicitly mentioned otherwise in the individual test. In any case, the locations of measurement points MUST be reported.

## 4.6. Connectivity Recommendation

The SDN controller in the test setup SHOULD be connected directly with the forwarding and the management plane test emulators to avoid any delays or failure introduced by the intermediate devices during benchmarking tests. When the controller is implemented as a virtual machine, details of the physical and logical connectivity MUST be reported.

## 4.7. Test Repeatability

To increase the confidence in measured result, it is recommended that each test SHOULD be repeated a minimum of 10 times.

Test Reporting

Each test has a reporting format that contains some global and identical reporting components, and some individual components that are specific to individual tests. The following test configuration parameters and controller settings parameters MUST be reflected in the test report.

Test Configuration Parameters:

1. Controller name and version
2. Northbound protocols and versions
3. Southbound protocols and versions
4. Controller redundancy mode (Standalone or Cluster Mode)
5. Connection setup (Unencrypted or Encrypted)
6. Network Device Type (Physical or Virtual or Emulated)
7. Number of Nodes
8. Number of Links
9. Dataplane Test Traffic Type
10. Controller System Configuration (e.g., Physical or Virtual Machine, CPU, Memory, Caches, Operating System, Interface Speed, Storage)
11. Reference Test Setup (e.g., Section 3.1 etc.,)

Controller Settings Parameters:

1. Topology re-discovery timeout
2. Controller redundancy mode (e.g., active-standby etc.,)
3. Controller state persistence enabled/disabled

To ensure the repeatability of test, the following capabilities of test emulator SHOULD be reported

1. Maximum number of Network Devices that the forwarding plane emulates
2. Control message processing time (e.g., Topology Discovery Messages)

One way to determine the above two values are to simulate the required control sessions and messages from the control plane.

## 5. Benchmarking Tests

### 5.1. Performance

#### 5.1.1. Network Topology Discovery Time

Objective:

The time taken by controller(s) to determine the complete network topology, defined as the interval starting with the first discovery message from the controller(s) at its Southbound interface, ending with all features of the static topology determined.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document in combination with Appendix A.

Prerequisite:

1. The controller MUST support network discovery.
2. Tester should be able to retrieve the discovered topology information either through the controller’s management interface, or northbound interface to determine if the discovery was successful and complete.
3. Ensure that the controller’s topology re-discovery timeout has been set to the maximum value to avoid initiation of re-discovery process in the middle of the test.

Procedure:

1. Ensure that the controller is operational, its network applications, northbound and southbound interfaces are up and running.
2. Establish the network connections between controller and Network Devices.
3. Record the time for the first discovery message (Tm1) received from the controller at forwarding plane test emulator interface I1.
4. Query the controller every 3 seconds to obtain the discovered network topology information through the northbound interface or the management interface and compare it with the deployed network topology information.
5. Stop the trial when the discovered topology information matches the deployed network topology, or when the discovered topology information return the same details for 3 consecutive queries.
6. Record the time last discovery message (Tmn) sent to controller from the forwarding plane test emulator interface (I1) when the trial completed successfully. (e.g., the topology matches).

Measurement:

Topology Discovery Time Tr1 = Tmn-Tm1.

```
                                        Tr1 + Tr2 + Tr3 .. Trn
Average Topology Discovery Time (TDm) = -----------------------
                                             Total Trials
```

```
                                         SUM[SQUAREOF(Tri-TDm)]
   Topology Discovery Time Variance (TDv)  ----------------------
                                             Total Trials -1
```

Reporting Format:

The Topology Discovery Time results MUST be reported in the format of a table, with a row for each successful iteration. The last row of the table indicates the Topology Discovery Time variance and the previous row indicates the average Topology Discovery Time.

If this test is repeated with varying number of nodes over the same topology, the results SHOULD be reported in the form of a graph. The X coordinate SHOULD be the Number of nodes (N), the Y coordinate SHOULD be the average Topology Discovery Time.

5.1.2. Asynchronous Message Processing Time

Objective:

The time taken by controller(s) to process an asynchronous message, defined as the interval starting with an asynchronous message from a network device after the discovery of all the devices by the controller(s), ending with a response message from the controller(s) at its Southbound interface.

Reference Test Setup:

This test SHOULD use one of the test setup described in section 3.1 or section 3.2 of this document in combination with Appendix A.

Prerequisite:

1. The controller MUST have successfully completed the network topology discovery for the connected Network Devices.

Procedure:

1. Generate asynchronous messages from every connected Network Device, to the SDN controller, one at a time in series from the forwarding plane test emulator for the trial duration.
2. Record every request transmit time (T1) and the corresponding response received time (R1) at the forwarding plane test emulator interface (I1) for every successful message exchange.

Measurement:

```
                                         (R1-T1) + (R2-T2)..(Rn-Tn)
   Asynchronous Message Processing Time Tr1 = -----------------------
                                                     Nrx

    Where Nrx is the total number of successful messages exchanged

                                                Tr1 + Tr2 + Tr3..Trn
   Average Asynchronous Message Processing Time = --------------------
                                                   Total Trials


   Asynchronous Message Processing Time Variance (TAMv) =

                                               SUM[SQUAREOF(Tri-TAMm)]
                                               ----------------------
                                                   Total Trials -1
```

Where TAMm is the Average Asynchronous Message Processing Time.

Reporting Format:

The Asynchronous Message Processing Time results MUST be reported in the format of a table with a row for each iteration. The last row of the table indicates the Asynchronous Message Processing Time variance and the previous row indicates the average Asynchronous Message Processing Time.

The report should capture the following information in addition to the configuration parameters captured in section 5.

- Successful messages exchanged (Nrx)

- Percentage of unsuccessful messages exchanged, computed using the formula (1 - Nrx/Ntx) * 100), Where Ntx is the total number of messages transmitted to the controller.

If this test is repeated with varying number of nodes with same topology, the results SHOULD be reported in the form of a graph. The X coordinate SHOULD be the Number of nodes (N), the Y coordinate SHOULD be the average Asynchronous Message Processing Time.

### 5.1.3. Asynchronous Message Processing Rate

Objective:

Measure the number of responses to asynchronous messages (such as new flow arrival notification message, etc.) for which the controller(s) performed processing and replied with a valid and productive (non-trivial) response message

This test will measure two benchmarks on Asynchronous Message Processing Rate using a single procedure. The two benchmarks are (see section 2.3.1.3 of [I-D.sdn-controller-benchmark-term]):

1. Loss-free Asynchronous Message Processing Rate

2. Maximum Asynchronous Message Processing Rate

Here two benchmarks are determined through a series of trials where the number of messages are sent to the controller(s), and the responses from the controller(s) are counted over the trial duration. The message response rate and the message loss ratio are calculated for each trial.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document in combination with Appendix A.

Prerequisite:

1. The controller(s) MUST have successfully completed the network topology discovery for the connected Network Devices.
2. Choose and record the Trial Duration (Td), the sending rate step-size (STEP), the tolerance on equality for two consecutive trials (P%),and the maximum possible message sending rate (Ntx1/Td).

Procedure:

1. Generate asynchronous messages continuously at the maximum possible rate on the established connections from all the emulated/simulated Network Devices for the given trial Duration (Td).
2. Record the total number of responses received from the controller (Nrx1) as well as the number of messages sent (Ntx1) to the controller within the trial duration (Td).

3. Calculate the Asynchronous Message Processing Rate (Tr1) and the Message Loss Ratio (Lr1). Ensure that the controller(s) have returned to normal operation.
4. Repeat the trial by reducing the asynchronous message sending rate used in last trial by the STEP size.
5. Continue repeating the trials and reducing the sending rate until both the maximum value of Nrxn and the Nrxn corresponding to zero loss ratio have been found.
6. The trials corresponding to the benchmark levels MUST be repeated using the same asynchronous message rates until the responses received from the controller are equal (+/-P%) for two consecutive trials.
7. Record the number of responses received from the controller (Nrxn) as well as the number of messages sent (Ntxn) to the controller in the last trial.

Measurement:

$$\text{Asynchronous Message Processing Rate Trn} = \frac{Nrxn}{Td}$$

Maximum Asynchronous Message Processing Rate = MAX(Trn) for all n

$$\text{Asynchronous Message Loss Ratio Lrn} = 1 - \frac{Nrxn}{Ntxn}$$

Loss-free Asynchronous Message Processing Rate = MAX(Trn) given Lrn=0

Reporting Format:

The Asynchronous Message Processing Rate results MUST be reported in the format of a table with a row for each trial.

The table should report the following information in addition to the configuration parameters captured in section 5, with columns:

- Offered rate (Ntxn/Td)

- Asynchronous Message Processing Rate (Trn)

- Loss Ratio (Lr)

- Benchmark at this iteration (blank for none, Maximum, Loss-Free)

The results MAY be presented in the form of a graph. The X axis SHOULD be the Offered rate, and dual Y axes would represent Asynchronous Message Processing Rate and Loss Ratio, respectively.

If this test is repeated with varying number of nodes over same topology, the results SHOULD be reported in the form of a graph. The X axis SHOULD be the Number of nodes (N), the Y axis SHOULD be the Asynchronous Message Processing Rate. Both the Maximum and the Loss-Free Rates should be plotted for each N.

#### 5.1.4. Reactive Path Provisioning Time

Objective:

The time taken by the controller to setup a path reactively between source and destination node, defined as the interval starting with the first flow provisioning request message received by the controller(s) at its Southbound interface, ending with the last flow provisioning response message sent from the controller(s) at its Southbound interface.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document in combination with Appendix A. The number of Network Devices in the path is a parameter of the test that may be varied from 2 to maximum discovery size in repetitions of this test.

Prerequisite:

1. The controller MUST contain the network topology information for the deployed network topology.
2. The controller should have the knowledge about the location of destination endpoint for which the path has to be provisioned. This can be achieved through dynamic learning or static provisioning.
3. Ensure that the default action for ’flow miss’ in Network Device is configured to ’send to controller’.
4. Ensure that each Network Device in a path requires the controller to make the forwarding decision while paving the entire path.

Procedure:

1. Send a single traffic stream from the test traffic generator TP1 to test traffic generator TP2.

2. Record the time of the first flow provisioning request message sent to the controller (Tsf1) from the Network Device at the forwarding plane test emulator interface (I1).
3. Wait for the arrival of first traffic frame at the Traffic Endpoint TP2 or the expiry of trial duration (Td).
4. Record the time of the last flow provisioning response message received from the controller (Tdf1) to the Network Device at the forwarding plane test emulator interface (I1).

Measurement:

```
    Reactive Path Provisioning Time Tr1 = Tdf1-Tsf1.

                                            Tr1 + Tr2 + Tr3 .. Trn
    Average Reactive Path Provisioning Time = -----------------------
                                            Total Trials

                                              SUM[SQUAREOF(Tri-TRPm)]
   Reactive Path Provisioning Time Variance(TRPv) ---------------------
                                                     Total Trials -1
```

Where TRPm is the Average Reactive Path Provisioning Time.

Reporting Format:

The Reactive Path Provisioning Time results MUST be reported in the format of a table with a row for each iteration. The last row of the table indicates the Reactive Path Provisioning Time variance and the previous row indicates the Average Reactive Path Provisioning Time.

The report should capture the following information in addition to the configuration parameters captured in section 5.

- Number of Network Devices in the path

## 5.1.5. Proactive Path Provisioning Time

Objective:

The time taken by the controller to setup a path proactively between source and destination node, defined as the interval starting with the first proactive flow provisioned in the controller(s) at its Northbound interface, ending with the last flow provisioning response message sent from the controller(s) at its Southbound interface.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document in combination with Appendix A.

Prerequisite:

1. The controller MUST contain the network topology information for the deployed network topology.
2. The controller should have the knowledge about the location of destination endpoint for which the path has to be provisioned. This can be achieved through dynamic learning or static provisioning.
3. Ensure that the default action for flow miss in Network Device is ’drop’.

Procedure:

1. Send a single traffic stream from test traffic generator TP1 to TP2.
2. Install the flow entries to reach from test traffic generator TP1 to the test traffic generator TP2 through controller’s northbound or management interface.
3. Wait for the arrival of first traffic frame at the test traffic generator TP2 or the expiry of trial duration (Td).
4. Record the time when the proactive flow is provisioned in the Controller (Tsf1) at the management plane test emulator interface I2.
5. Record the time of the last flow provisioning message received from the controller (Tdf1) at the forwarding plane test emulator interface I1.

Measurement:

```
    Proactive Flow Provisioning Time Tr1 = Tdf1-Tsf1.

                                            Tr1 + Tr2 + Tr3 .. Trn
    Average Proactive Path Provisioning Time = -----------------------
                                             Total Trials

                                             SUM[SQUAREOF(Tri-TPPm)]
   Proactive Path Provisioning Time Variance(TPPv) --------------------
                                                  Total Trials -1
```

Where TPPm is the Average Proactive Path Provisioning Time.

Reporting Format:

The Proactive Path Provisioning Time results MUST be reported in the format of a table with a row for each iteration. The last row of the table indicates the Proactive Path Provisioning Time variance and the previous row indicates the Average Proactive Path Provisioning Time.

The report should capture the following information in addition to the configuration parameters captured in section 5.

- Number of Network Devices in the path

### 5.1.6. Reactive Path Provisioning Rate

Objective:

The maximum number of independent paths a controller can concurrently establish per second between source and destination nodes reactively, defined as the number of paths provisioned per second by the controller(s) at its Southbound interface for the flow provisioning requests received for path provisioning at its Southbound interface between the start of the test and the expiry of given trial duration.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document in combination with Appendix A.

Prerequisite:

1. The controller MUST contain the network topology information for the deployed network topology.
2. The controller should have the knowledge about the location of destination addresses for which the paths have to be provisioned. This can be achieved through dynamic learning or static provisioning.
3. Ensure that the default action for ’flow miss’ in Network Device is configured to ’send to controller’.
4. Ensure that each Network Device in a path requires the controller to make the forwarding decision while provisioning the entire path.

Procedure:

1. Send traffic with unique source and destination addresses from test traffic generator TP1.
2. Record total number of unique traffic frames (Ndf) received at the test traffic generator TP2 within the trial duration (Td).

Measurement:

```
                                        Ndf
    Reactive Path Provisioning Rate Tr1 = ------
                                        Td

                                            Tr1 + Tr2 + Tr3 .. Trn
    Average Reactive Path Provisioning Rate = -----------------------
                                            Total Trials

                                             SUM[SQUAREOF(Tri-RPPm)]
   Reactive Path Provisioning Rate Variance(RPPv) --------------------
                                                   Total Trials -1
```

Where RPPm is the Average Reactive Path Provisioning Rate.

Reporting Format:

The Reactive Path Provisioning Rate results MUST be reported in the format of a table with a row for each iteration. The last row of the table indicates the Reactive Path Provisioning Rate variance and the previous row indicates the Average Reactive Path Provisioning Rate.

The report should capture the following information in addition to the configuration parameters captured in section 5.

- Number of Network Devices in the path

- Offered rate

## 5.1.7. Proactive Path Provisioning Rate

Objective:

Measure the maximum number of independent paths a controller can concurrently establish per second between source and destination nodes proactively, defined as the number of paths provisioned per second by the controller(s) at its Southbound interface for the paths requested in its Northbound interface between the start of the test and the expiry of given trial duration. The measurement is based on dataplane observations of successful path activation

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document in combination with Appendix A.

Prerequisite:

1. The controller MUST contain the network topology information for the deployed network topology.

2. The controller should have the knowledge about the location of destination addresses for which the paths have to be provisioned. This can be achieved through dynamic learning or static provisioning.

3. Ensure that the default action for flow miss in Network Device is 'drop'.

Procedure:

1. Send traffic continuously with unique source and destination addresses from test traffic generator TP1.

2. Install corresponding flow entries to reach from simulated sources at the test traffic generator TP1 to the simulated destinations at test traffic generator TP2 through controller's northbound or management interface.

3. Record total number of unique traffic frames received Ndf) at the test traffic generator TP2 within the trial duration (Td).

Measurement:

```
                                         Ndf
    Proactive Path Provisioning Rate Tr1 = ------
                                         Td

                                            Tr1 + Tr2 + Tr3 .. Trn
    Average Proactive Path Provisioning Rate = -----------------------
                                            Total Trials
```

```
                                                 SUM[SQUAREOF(Tri-PPPm)]
   Proactive Path Provisioning Rate Variance(PPPv) --------------------
                                                      Total Trials -1
```

Where PPPm is the Average Proactive Path Provisioning Rate.

Reporting Format:

The Proactive Path Provisioning Rate results MUST be reported in the format of a table with a row for each iteration. The last row of the table indicates the Proactive Path Provisioning Rate variance and the previous row indicates the Average Proactive Path Provisioning Rate.

The report should capture the following information in addition to the configuration parameters captured in section 5.

- Number of Network Devices in the path

- Offered rate

## 5.1.8. Network Topology Change Detection Time

Objective:

The amount of time required for the controller to detect any changes in the network topology, defined as the interval starting with the notification message received by the controller(s) at its Southbound interface, ending with the first topology rediscovery messages sent from the controller(s) at its Southbound interface.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document in combination with Appendix A.

Prerequisite:

1. The controller MUST have successfully discovered the network topology information for the deployed network topology.

2. The periodic network discovery operation should be configured to twice the Trial duration (Td) value.

Procedure:

1. Trigger a topology change event by bringing down an active Network Device in the topology.

2. Record the time when the first topology change notification is sent to the controller (Tcn) at the forwarding plane test emulator interface (I1).

3. Stop the trial when the controller sends the first topology re-discovery message to the Network Device or the expiry of trial duration (Td).

4. Record the time when the first topology re-discovery message is received from the controller (Tcd) at the forwarding plane test emulator interface (I1)

Measurement:

Network Topology Change Detection Time Tr1 = Tcd-Tcn.

```
                                                     Tr1 + Tr2 + Tr3 .. Trn
 Average Network Topology Change Detection Time = ------------------
                                                      Total Trials

Network Topology Change Detection Time Variance(NTDv) =

                                          SUM[SQUAREOF(Tri-NTDm)]
                                          -----------------------
                                              Total Trials -1
```

Where NTDm is the Average Network Topology Change Detection Time.

Reporting Format:

The Network Topology Change Detection Time results MUST be reported in the format of a table with a row for each iteration. The last row of the table indicates the Network Topology Change Detection Time variance and the previous row indicates the average Network Topology Change Time.

## 5.2. Scalability

### 5.2.1. Control Session Capacity

Objective:

Measure the maximum number of control sessions the controller can maintain, defined as the number of sessions that the controller can accept from network devices, starting with the first control session, ending with the last control session that the controller(s) accepts at its Southbound interface.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document in combination with Appendix A.

Procedure:

1. Establish control connection with controller from every Network Device emulated in the forwarding plane test emulator.
2. Stop the trial when the controller starts dropping the control connections.
3. Record the number of successful connections established with the controller (CCn) at the forwarding plane test emulator.

Measurement:

Control Sessions Capacity = CCn.

Reporting Format:

The Control Session Capacity results MUST be reported in addition to the configuration parameters captured in section 5.

### 5.2.2. Network Discovery Size

Objective:

Measure the network size (number of nodes, links and hosts) that a controller can discover, defined as the size of a network that the controller(s) can discover, starting from a network topology given by the user for discovery, ending with the topology that the controller(s) could successfully discover.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document in combination with Appendix A.

Prerequisite:

1. The controller MUST support automatic network discovery.
2. Tester should be able to retrieve the discovered topology information either through controller's management interface or northbound interface.

Procedure:

1. Establish the network connections between controller and network nodes.
2. Query the controller for the discovered network topology information and compare it with the deployed network topology information.
3. If the comparison is successful, increase the number of nodes by 1 and repeat the trial.
   If the comparison is unsuccessful, decrease the number of nodes by 1 and repeat the trial.
4. Continue the trial until the comparison of step 3 is successful.
5. Record the number of nodes for the last trial (Ns) where the topology comparison was successful.

Measurement:

Network Discovery Size = Ns.

Reporting Format:

The Network Discovery Size results MUST be reported in addition to the configuration parameters captured in section 5.

### 5.2.3. Forwarding Table Capacity

Objective:

Measure the maximum number of flow entries a controller can manage in its Forwarding table.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document in combination with Appendix A.

Prerequisite:

1. The controller Forwarding table should be empty.
2. Flow Idle time MUST be set to higher or infinite value.
3. The controller MUST have successfully completed network topology discovery.
4. Tester should be able to retrieve the forwarding table information either through controller’s management interface or northbound interface.

Procedure:

Reactive Flow Provisioning Mode:

1. Send bi-directional traffic continuously with unique source and/or destination addresses from test traffic generators TP1 and TP2 at the asynchronous message processing rate of controller.
2. Query the controller at a regular interval (e.g., 5 seconds) for the number of learned flow entries from its northbound interface.
3. Stop the trial when the retrieved value is constant for three consecutive iterations and record the value received from the last query (Nrp).

Proactive Flow Provisioning Mode:

1. Install unique flows continuously through controller’s northbound or management interface until a failure response is received from the controller.
2. Record the total number of successful responses (Nrp).

Note:

Some controller designs for proactive flow provisioning mode may require the switch to send flow setup requests in order to generate flow setup responses. In such cases, it is recommended to generate bi-directional traffic for the provisioned flows.

Measurement:

Proactive Flow Provisioning Mode:

Max Flow Entries = Total number of flows provisioned (Nrp)

Reactive Flow Provisioning Mode:

Max Flow Entries = Total number of learned flow entries (Nrp)

Forwarding Table Capacity = Max Flow Entries.

Reporting Format:

The Forwarding Table Capacity results MUST be tabulated with the following information in addition to the configuration parameters captured in section 5.

- Provisioning Type (Proactive/Reactive)

## 5.3. Security

### 5.3.1. Exception Handling

Objective:

Determine the effect of handling error packets and notifications on performance tests. The impact MUST be measured for the following performance tests

a. Path Provisioning Rate

b. Path Provisioning Time

c. Network Topology Change Detection Time

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document in combination with Appendix A.

Prerequisite:

1. This test MUST be performed after obtaining the baseline measurement results for the above performance tests.

2. Ensure that the invalid messages are not dropped by the intermediate devices connecting the controller and Network Devices.

Procedure:

1. Perform the above listed performance tests and send 1% of messages from the Asynchronous Message Processing Rate as invalid messages from the connected Network Devices emulated at the forwarding plane test emulator.
2. Perform the above listed performance tests and send 2% of messages from the Asynchronous Message Processing Rate as invalid messages from the connected Network Devices emulated at the forwarding plane test emulator.

Note:

Invalid messages can be frames with incorrect protocol fields or any form of failure notifications sent towards controller.

Measurement:

Measurement MUST be done as per the equation defined in the corresponding performance test measurement section.

Reporting Format:

The Exception Handling results MUST be reported in the format of table with a column for each of the below parameters and row for each of the listed performance tests.

- Without Exceptions

- With 1% Exceptions

- With 2% Exceptions

### 5.3.2. Denial of Service Handling

Objective:

Determine the effect of handling DoS attacks on performance and scalability tests the impact MUST be measured for the following tests:

a. Path Provisioning Rate

b. Path Provisioning Time

c. Network Topology Change Detection Time

d. Network Discovery Size

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document in combination with Appendix A.

Prerequisite:

This test MUST be performed after obtaining the baseline measurement results for the above tests.

Procedure:

1. Perform the listed tests and launch a DoS attack towards controller while the trial is running.

Note:

DoS attacks can be launched on one of the following interfaces.

a. Northbound (e.g., Query for flow entries continuously on northbound interface)
b. Management (e.g., Ping requests to controller’s management interface)
c. Southbound (e.g., TCP SYN messages on southbound interface)

Measurement:

Measurement MUST be done as per the equation defined in the corresponding test’s measurement section.

Reporting Format:

The DoS Attacks Handling results MUST be reported in the format of table with a column for each of the below parameters and row for each of the listed tests.

- Without any attacks

- With attacks

The report should also specify the nature of attack and the interface.

## 5.4. Reliability

### 5.4.1. Controller Failover Time

Objective:

The time taken to switch from an active controller to the backup controller, when the controllers work in redundancy mode and the active controller fails, defined as the interval starting with the active controller bringing down, ending with the first re-discovery message received from the new controller at its Southbound interface.

Reference Test Setup:

The test SHOULD use the test setup described in section 3.2 of this document in combination with Appendix A.

Prerequisite:

1. Master controller election MUST be completed.
2. Nodes are connected to the controller cluster as per the Redundancy Mode (RM).
3. The controller cluster should have successfully completed the network topology discovery.
4. The Network Device MUST send all new flows to the controller when it receives from the test traffic generator.
5. Controller should have learned the location of destination (D1) at test traffic generator TP2.

Procedure:

1. Send uni-directional traffic continuously with incremental sequence number and source addresses from test traffic generator TP1 at the rate that the controller processes without any drops.
2. Ensure that there are no packet drops observed at the test traffic generator TP2.
3. Bring down the active controller.
4. Stop the trial when a first frame received on TP2 after failover operation.

5. Record the time at which the last valid frame received (T1) at test traffic generator TP2 before sequence error and the first valid frame received (T2) after the sequence error at TP2

Measurement:

Controller Failover Time = (T2 - T1)

Packet Loss = Number of missing packet sequences.

Reporting Format:

The Controller Failover Time results MUST be tabulated with the following information.

- Number of cluster nodes
- Redundancy mode
- Controller Failover Time
- Packet Loss
- Cluster keep-alive interval

### 5.4.2. Network Re-Provisioning Time

Objective:

The time taken to re-route the traffic by the Controller, when there is a failure in existing traffic paths, defined as the interval starting from the first failure notification message received by the controller, ending with the last flow re-provisioning message sent by the controller at its Southbound interface.

Reference Test Setup:

This test SHOULD use one of the test setup described in section 3.1 or section 3.2 of this document in combination with Appendix A.

Prerequisite:

1. Network with the given number of nodes and redundant paths MUST be deployed.

2. Ensure that the controller MUST have knowledge about the location of test traffic generators TP1 and TP2.
3. Ensure that the controller does not pre-provision the alternate path in the emulated Network Devices at the forwarding plane test emulator.

Procedure:

1. Send bi-directional traffic continuously with unique sequence number from TP1 and TP2.
2. Bring down a link or switch in the traffic path.
3. Stop the trial after receiving first frame after network re-convergence.
4. Record the time of last received frame prior to the frame loss at TP2 (TP2-Tlfr) and the time of first frame received after the frame loss at TP2 (TP2-Tffr). There must be a gap in sequence numbers of these frames
5. Record the time of last received frame prior to the frame loss at TP1 (TP1-Tlfr) and the time of first frame received after the frame loss at TP1 (TP1-Tffr).

Measurement:

```
    Forward Direction Path Re-Provisioning Time (FDRT)
                                              = (TP2-Tffr - TP2-Tlfr)

    Reverse Direction Path Re-Provisioning Time (RDRT)
                                              =  (TP1-Tffr - TP1-Tlfr)

    Network Re-Provisioning Time = (FDRT+RDRT)/2

    Forward Direction Packet Loss = Number of missing sequence frames
    at TP1

    Reverse Direction Packet Loss = Number of missing sequence frames
    at TP2
```

Reporting Format:

The Network Re-Provisioning Time results MUST be tabulated with the following information.

- Number of nodes in the primary path

- Number of nodes in the alternate path

- Network Re-Provisioning Time

- Forward Direction Packet Loss

- Reverse Direction Packet Loss

## 6. References

### 6.1. Normative References

[RFC2119]  S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[RFC8174]  B. Leiba, "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", RFC 8174, May 2017.

[I-D.sdn-controller-benchmark-term]  Bhuvaneswaran.V, Anton Basil, Mark.T, Vishwas Manral, Sarah Banks, "Terminology for Benchmarking SDN Controller Performance", draft-ietf-bmwg-sdn-controller-benchmark-term-08 (Work in progress), February 25, 2018

### 6.2. Informative References

[OpenFlow Switch Specification]  ONF,"OpenFlow Switch Specification" Version 1.4.0 (Wire Protocol 0x05), October 14, 2013.

## 7. IANA Considerations

This document does not have any IANA requests.

## 8. Security Considerations

Benchmarking tests described in this document are limited to the performance characterization of controllers 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.

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

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

## 9. Acknowledgments

The authors would like to thank the following individuals for providing their valuable comments to the earlier versions of this document: Al Morton (AT&T), Sandeep Gangadharan (HP), M. Georgescu (NAIST), Andrew McGregor (Google), Scott Bradner , Jay Karthik (Cisco), Ramakrishnan (Dell), Khasanov Boris (Huawei), Brian Castelli (Spirent)

This document was prepared using 2-Word-v2.0.template.dot.

## Appendix A.                 Example Test Topology

### A.1. Leaf-Spine Topology

```
                             +------+    +------+
                             |  SDN |    | SDN  |   (Spine)
                             | Node |..  | Node |
                             +------+    +------+
                                  / \    / \
                                 /   \  /   \
                             l1 /     /      \ ln
                                 /     / \     \
                           +--------+    +-------+
                           |  SDN   |    |  SDN  |
                           |  Node  |..  |  Node | (Leaf)
                           +--------+    +-------+
```

## Appendix B. Benchmarking Methodology using OpenFlow Controllers

This section gives an overview of OpenFlow protocol and provides test methodology to benchmark SDN controllers supporting OpenFlow southbound protocol.

### B.1. Protocol Overview

OpenFlow is an open standard protocol defined by Open Networking Foundation (ONF)[ OpenFlow Switch Specification], used for programming the forwarding plane of network switches or routers via a centralized controller.

### B.2. Messages Overview

OpenFlow protocol supports three messages types namely controller-to-switch, asynchronous and symmetric.

Controller-to-switch messages are initiated by the controller and used to directly manage or inspect the state of the switch. These messages allow controllers to query/configure the switch (Features, Configuration messages), collect information from switch (Read-State message), send packets on specified port of switch (Packet-out message), and modify switch forwarding plane and state (Modify-State, Role-Request messages etc.).

Asynchronous messages are generated by the switch without a controller soliciting them. These messages allow switches to update controllers to denote an arrival of new flow (Packet-in), switch state change (Flow-Removed, Port-status) and error (Error).

Symmetric messages are generated in either direction without solicitation. These messages allow switches and controllers to set up connection (Hello), verify for liveness (Echo) and offer additional functionalities (Experimenter).

### B.3. Connection Overview

OpenFlow channel is used to exchange OpenFlow message between an OpenFlow switch and an OpenFlow controller. The OpenFlow channel connection can be setup using plain TCP or TLS. By default, a switch establishes single connection with SDN controller. A switch may establish multiple parallel connections to single controller (auxiliary connection) or multiple controllers to handle controller failures and load balancing.

## B.4. Performance Benchmarking Tests

### B.4.1. Network Topology Discovery Time

Procedure:

```
      Network Devices               OpenFlow                 SDN
                                   Controller            Application
            |                            |                           |
            |                            |<Initialize controller     |
            |                            |app.,NB and SB interfaces> |
            |                            |                           |
            |<Deploy network with        |                           |
            | given no. of OF switches>  |                           |
            |                            |                           |
            |    OFPT_HELLO Exchange     |                           |
            |<-------------------------->|                           |
            |                            |                           |
            |    PACKET_OUT with LLDP    |                           |
            |      to all switches       |                           |
       (Tm1)|<---------------------------|                           |
            |                            |                           |
            |         PACKET_IN with LLDP|                           |
            |          rcvd from switch-1|                           |
            |--------------------------->|                           |
            |                            |                           |
            |         PACKET_IN with LLDP|                           |
            |          rcvd from switch-2|                           |
            |--------------------------->|                           |
            |            .               |                           |
            |            .               |                           |
            |                            |                           |
            |         PACKET_IN with LLDP|                           |
            |          rcvd from switch-n|                           |
       (Tmn)|--------------------------->|                           |
            |                            |                           |
            |                            |      <Wait for the expiry |
            |                            |     of Trial duration (Td)>|
            |                            |                           |
            |                            |   Query the controller for|
            |                            |   discovered n/w topo.(Di)|
            |                            |<--------------------------|
            |                            |                           |
            |                            |   <Compare the discovered |
            |                            |    & offered n/w topology>|
            |                            |                           |
```

Legend:

```
         NB: Northbound
         SB: Southbound
         OF: OpenFlow
         Tm1: Time of reception of first LLDP message from controller
         Tmn: Time of last LLDP message sent to controller
```

Discussion:

The Network Topology Discovery Time can be obtained by calculating the time difference between the first PACKET_OUT with LLDP message received from the controller (Tm1) and the last PACKET_IN with LLDP message sent to the controller (Tmn) when the comparison is successful.

B.4.2. Asynchronous Message Processing Time

Procedure:

```
         Network Devices            OpenFlow                   SDN
                                   Controller              Application
            |                             |                            |
            |PACKET_IN with single        |                            |
            |OFP match header             |                            |
        (T0)|---------------------------->|                            |
            |                             |                            |
            | PACKET_OUT with single OFP  |                            |
            |             action header   |                            |
        (R0)|<----------------------------|                            |
            |          .                  |                            |
            |          .                  |                            |
            |          .                  |                            |
            |                             |                            |
            |PACKET_IN with single OFP    |                            |
            |match header                 |                            |
        (Tn)|---------------------------->|                            |
            |                             |                            |
            | PACKET_OUT with single OFP  |                            |
            |              action header|                            |
        (Rn)|<----------------------------|                            |
            |                             |                            |
            |<Wait for the expiry of      |                            |
            |Trial duration>               |                           |
            |                             |                            |
            |<Record the number of        |                            |
            |PACKET_INs/PACKET_OUTs       |                            |
            |Exchanged (Nrx)>             |                            |
            |                             |                            |
```

Legend:

```
         T0,T1, ..Tn are PACKET_IN messages transmit timestamps.
         R0,R1, ..Rn are PACKET_OUT messages receive timestamps.
         Nrx : Number of successful PACKET_IN/PACKET_OUT message
   exchanges
```

Discussion:

The Asynchronous Message Processing Time will be obtained by sum of ((R0-T0),(R1-T1)..(Rn - Tn))/ Nrx.

B.4.3. Asynchronous Message Processing Rate

Procedure:

```
         Network Devices           OpenFlow                      SDN
                                  Controller                Application
            |                             |                            |
            |PACKET_IN with single OFP    |                            |
            |match headers                |                            |
            |---------------------------->|                            |
            |                             |                            |
            | PACKET_OUT with single      |                            |
            |          OFP action headers|                            |
            |<----------------------------|                            |
            |                             |                            |
            |             .               |                            |
            |             .               |                            |
            |             .               |                            |
            |                             |                            |
            |PACKET_IN with single OFP    |                            |
            |match headers                |                            |
            |---------------------------->|                            |
            |                             |                            |
            | PACKET_OUT with single      |                            |
            |          OFP action headers|                            |
            |<----------------------------|                            |
            |                             |                            |
            |<Repeat the steps until the  |                            |
            |expiry of Trial Duration>    |                            |
            |                             |                            |
            |<Record the number of OFP   |                            |
      (Ntx1)|match headers sent>         |                            |
            |                            |                            |
            |<Record the number of OFP   |                            |
      (Nrx1)|action headers rcvd>        |                            |
            |                            |                            |
```

Note: The Ntx1 on initial trials should be greater than Nrx1 and repeat the trials until the Nrxn for two consecutive trials equeal to (+/-P%).

Discussion:

This test will measure two benchmarks using single procedure. 1) The Maximum Asynchronous Message Processing Rate will be obtained by calculating the maximum PACKET OUTs (Nrxn) received from the controller(s) across n trials. 2) The Loss-free Asynchronous Message Processing Rate will be obtained by calculating the maximum PACKET OUTs received from controller (s) when Loss Ratio equals zero. The loss ratio is obtained by 1 - Nrxn/Ntxn

## B.4.4. Reactive Path Provisioning Time

Procedure:

```
        Test Traffic     Test Traffic      Network Devices    OpenFlow
       Generator TP1    Generator TP2                         Controller
             |             |                       |                   |
             |             |G-ARP (D1)             |                   |
             |             |---------------------->|                   |
             |             |                       |                   |
             |             |                       |PACKET_IN(D1)      |
             |             |                       |------------------>|
             |             |                       |                   |
             |Traffic (S1,D1)                      |                   |
       (Tsf1)|----------------------------------->|                   |
             |             |                       |                   |
             |             |                       |                   |
             |             |                       |                   |
             |             |                       |PACKET_IN(S1,D1)   |
             |             |                       |------------------>|
             |             |                       |                   |
             |             |                       |  FLOW_MOD(D1)     |
             |             |                       |<------------------|
             |             |                       |                   |
             |             |Traffic (S1,D1)        |                   |
             |       (Tdf1)|<----------------------|                   |
            |             |                      |                   |
```

Legend:

```
         G-ARP: Gratuitous ARP message.
         Tsf1: Time of first frame sent from TP1
         Tdf1: Time of first frame received from TP2
```

Discussion:

The Reactive Path Provisioning Time can be obtained by finding the time difference between the transmit and receive time of the traffic (Tsf1-Tdf1).

B.4.5. Proactive Path Provisioning Time

Procedure:

```
   Test Traffic  Test Traffic    Network Devices OpenFlow        SDN
   Generator TP1 Generator TP2                Controller  Application
        |             |               |                 |              |
        |             |G-ARP (D1)     |                 |              |
        |             |-------------->|                 |              |
        |             |               |                 |              |
        |             |               |PACKET_IN(D1)    |              |
        |             |               |---------------->|              |
        |             |               |                 |              |
        |Traffic (S1,D1)              |                 |              |
   Tsf1)|---------------------------->|                 |              |
        |             |               |                 |              |
        |             |               |                 | <Install flow|
        |             |               |                 |  for S1,D1>  |
        |             |               |                 |              |
        |             |               |  FLOW_MOD(D1)   |              |
        |             |               |<----------------|              |
        |             |               |                 |              |
        |             |Traffic (S1,D1)|                 |              |
        |       (Tdf1)|<--------------|                 |              |
        |             |               |                 |              |
```

Legend:

```
         G-ARP: Gratuitous ARP message.
         Tsf1: Time of first frame sent from TP1
         Tdf1: Time of first frame received from TP2
```

Discussion:

The Proactive Path Provisioning Time can be obtained by finding the time difference between the transmit and receive time of the traffic (Tsf1-Tdf1).

## B.4.6. Reactive Path Provisioning Rate

Procedure:

```
       Test Traffic     Test Traffic   Network Devices        OpenFlow
      Generator TP1    Generator TP2                        Controller
            |             |                    |                       |
            |             |                    |                       |
            |             |                    |                       |
            |             |G-ARP (D1..Dn)      |                       |
            |             |--------------------|                       |
            |             |                    |                       |
            |             |                    |PACKET_IN(D1..Dn)      |
            |             |                    |---------------------->|
            |             |                    |                       |
            |Traffic (S1..Sn,D1..Dn)           |                       |
            |--------------------------------->|                       |
            |             |                    |                       |
            |             |                    |PACKET_IN(S1.Sn,D1.Dn)|
            |             |                    |---------------------->|
            |             |                    |                       |
            |             |                    |        FLOW_MOD(S1)   |
            |             |                    |<----------------------|
            |             |                    |                       |
            |             |                    |        FLOW_MOD(D1)   |
            |             |                    |<----------------------|
            |             |                    |                       |
            |             |                    |        FLOW_MOD(S2)   |
            |             |                    |<----------------------|
            |             |                    |                       |
            |             |                    |        FLOW_MOD(D2)   |
            |             |                    |<----------------------|
            |             |                    |              .        |
            |             |                    |              .        |
            |             |                    |                       |
            |             |                    |        FLOW_MOD(Sn)   |
            |             |                    |<----------------------|
            |             |                    |                       |
            |             |                    |        FLOW_MOD(Dn)   |
            |             |                    |<----------------------|
            |             |                    |                       |
            |             | Traffic (S1..Sn,    |                      |
            |             |             D1..Dn)|                      |
            |             |<-------------------|                      |
            |             |                    |                      |
            |             |                    |                      |
```

Legend:

```
         G-ARP: Gratuitous ARP
         D1..Dn: Destination Endpoint 1, Destination Endpoint 2 ....
                 Destination Endpoint n
         S1..Sn: Source Endpoint 1, Source Endpoint 2 .., Source
   Endpoint n
```

Discussion:

The Reactive Path Provisioning Rate can be obtained by finding the total number of frames received at TP2 after the trial duration.

B.4.7. Proactive Path Provisioning Rate

Procedure:

```
   Test Traffic  Test Traffic   Network Devices  OpenFlow       SDN
   Generator TP1 Generator TP2                  Controller  Application
        |             |               |                |              |
        |             |G-ARP (D1..Dn) |                |              |
        |             |-------------->|                |              |
        |             |               |                |              |
        |             |               |PACKET_IN(D1.Dn)|              |
        |             |               |--------------->|              |
        |             |               |                |              |
        |Traffic (S1..Sn,D1..Dn)      |                |              |
   Tsf1)|---------------------------->|                |              |
        |             |               |                |              |
        |             |               |                | <Install flow|
        |             |               |                |  for S1,D1>  |
        |             |               |                |              |
        |             |               |                |       .      |
        |             |               |                | <Install flow|
        |             |               |                |  for Sn,Dn>  |
        |             |               |                |              |
        |             |               |  FLOW_MOD(S1)  |              |
        |             |               |<---------------|              |
        |             |               |                |              |
        |             |               |  FLOW_MOD(D1)  |              |
        |             |               |<---------------|              |
        |             |               |       .        |              |
        |             |               |  FLOW_MOD(Sn)  |              |
        |             |               |<---------------|              |
        |             |               |                |              |
        |             |               |  FLOW_MOD(Dn)  |              |
        |             |               |<---------------|              |
        |             |               |                |              |
        |             |Traffic (S1.Sn,|                |              |
        |             |         D1.Dn)|                |              |
        |       (Tdf1)|<--------------|                |              |
        |             |               |                |              |
```

Legend:

```
         G-ARP: Gratuitous ARP
         D1..Dn: Destination Endpoint 1, Destination Endpoint 2 ....
                 Destination Endpoint n
         S1..Sn: Source Endpoint 1, Source Endpoint 2 .., Source
   Endpoint n
```

Discussion:

The Proactive Path Provisioning Rate can be obtained by finding the total number of frames received at TP2 after the trial duration

B.4.8. Network Topology Change Detection Time

Procedure:

```
       Network Devices              OpenFlow                   SDN
                                   Controller              Application
             |                           |                          |
             |                           |     <Bring down a link in |
             |                           |                switch S1>|
             |                           |                          |
         T0  |PORT_STATUS with link down |                          |
             | from S1                   |                          |
             |-------------------------->|                          |
             |                           |                          |
             |First PACKET_OUT with LLDP |                          |
             |to OF Switch               |                          |
         T1  |<--------------------------|                          |
             |                           |                          |
             |                           |       <Record time of 1st |
             |                           |   PACKET_OUT with LLDP T1>|
```

Discussion:

The Network Topology Change Detection Time can be obtained by finding the difference between the time the OpenFlow switch S1 sends the PORT_STATUS message (T0) and the time that the OpenFlow controller sends the first topology re-discovery message (T1) to OpenFlow switches.

## B.5. Scalability

### B.5.1. Control Sessions Capacity

Procedure:

```
         Network Devices                       OpenFlow
                                              Controller
            |                                       |
            |    OFPT_HELLO Exchange for Switch 1   |
            |<------------------------------------->|
            |                                       |
            |    OFPT_HELLO Exchange for Switch 2   |
            |<------------------------------------->|
            |                  .                    |
            |                  .                    |
            |                  .                    |
            |    OFPT_HELLO Exchange for Switch n   |
            |X<----------------------------------->X|
            |                                       |
```

Discussion:

The value of Switch n-1 will provide Control Sessions Capacity.

### B.5.2. Network Discovery Size

Procedure:

```
       Network Devices            OpenFlow                  SDN
                                 Controller             Application
            |                          |                           |
            |                          |     <Deploy network with  |
            |                          |given no. of OF switches N>|
            |                          |                           |
            |    OFPT_HELLO Exchange   |                           |
            |<------------------------>|                           |
            |                          |                           |
            |    PACKET_OUT with LLDP  |                           |
            |       to all switches       |                            |
            |<----------------------------|                            |
            |                             |                            |
            |          PACKET_IN with LLDP|                            |
            |           rcvd from switch-1|                            |
            |---------------------------->|                            |
            |                             |                            |
            |          PACKET_IN with LLDP|                            |
            |           rcvd from switch-2|                            |
            |---------------------------->|                            |
            |             .               |                            |
            |             .               |                            |
            |                             |                            |
            |          PACKET_IN with LLDP|                            |
            |           rcvd from switch-n|                            |
            |---------------------------->|                            |
            |                             |                            |
            |                             |      <Wait for the expiry  |
            |                             |    of Trial duration (Td)>|
            |                             |                            |
            |                             |   Query the controller for|
            |                             |   discovered n/w topo.(N1)|
            |                             |<---------------------------|
            |                             |                            |
            |                             |   <If N1==N repeat Step 1  |
            |                             |with N+1 nodes until N1<N >|
            |                             |                            |
            |                             |   <If N1<N repeat Step 1   |
            |                             | with N=N1 nodes once and   |
            |                             | exit>                      |
            |                             |                            |
```

Legend:

         n/w topo: Network Topology
         OF: OpenFlow

Discussion:

The value of N1 provides the Network Discovery Size value. The trial duration can be set to the stipulated time within which the user expects the controller to complete the discovery process.

### B.5.3. Forwarding Table Capacity

Procedure:

```
   Test Traffic      Network Devices      OpenFlow            SDN
   Generator TP1                         Controller        Application
        |                  |                   |                 |
        |                  |                   |                 |
        |G-ARP (H1..Hn)    |                   |                 |
        |----------------->|                   |                 |
        |                  |                   |                 |
        |                  |PACKET_IN(D1..Dn)  |                 |
        |                  |------------------>|                 |
        |                  |                   |                 |
        |                  |                   |<Wait for 5 secs>|
        |                  |                   |                 |
        |                  |                   |  <Query for FWD |
        |                  |                   |          entry> |(F1)
        |                  |                   |                 |
        |                  |                   |<Wait for 5 secs>|
        |                  |                   |                 |
        |                  |                   |  <Query for FWD |
        |                  |                   |          entry> |(F2)
        |                  |                   |                 |
        |                  |                   |<Wait for 5 secs>|
        |                  |                   |                 |
        |                  |                   |  <Query for FWD |
        |                  |                   |          entry> |(F3)
        |                  |                   |                 |
        |                  |                   | <Repeat Step 2  |
        |                  |                   |until F1==F2==F3>|
        |                  |                   |                 |
```

Legend:

G-ARP: Gratuitous ARP
H1..Hn: Host 1 .. Host n
FWD: Forwarding Table

Discussion:

Query the controller forwarding table entries for multiple times until the three consecutive queries return the same value. The last value retrieved from the controller will provide the Forwarding Table Capacity value. The query interval is user configurable. The 5 seconds shown in this example is for representational purpose.

## B.6. Security

### B.6.1. Exception Handling

Procedure:

```
   Test Traffic  Test Traffic   Network Devices  OpenFlow        SDN
   Generator TP1 Generator TP2                  Controller  Application
       |          |                   |                 |            |
       |          |G-ARP (D1..Dn)     |                 |            |
       |          |------------------>|                 |            |
       |          |                   |                 |            |
       |          |                   |PACKET_IN(D1..Dn)|            |
       |          |                   |---------------->|            |
       |          |                   |                 |            |
       |Traffic (S1..Sn,D1..Dn)       |                 |            |
       |----------------------------->|                 |            |
       |          |                   |                 |            |
       |          |                   |PACKET_IN(S1..Sa,|            |
       |          |                   |          D1..Da)|            |
       |          |                   |---------------->|            |
       |          |                   |                 |            |
       |          |                   |PACKET_IN(Sa+1.. |            |
       |          |                   |.Sn,Da+1..Dn)    |            |
       |          |                   |(1% incorrect OFP|            |
       |          |                   |    Match header)|            |
       |          |                   |---------------->|            |
       |          |                   |                 |            |
       |          |                   | FLOW_MOD(D1..Dn)|            |
       |          |                   |<----------------|            |
       |          |                   |                 |            |
       |          |                   | FLOW_MOD(S1..Sa)|            |
       |          |                   |      OFP headers|            |
       |          |                   |<----------------|            |
       |          |                   |                 |            |
       |          |Traffic (S1..Sa,   |                 |            |
       |          |            D1..Da)|                 |            |
       |          |<------------------|                 |            |
       |          |                   |                 |            |
       |          |                   |                 |  <Wait for |
       |          |                   |                 |      Test  |
       |          |                   |                 |   Duration>|
       |          |                   |                 |            |
       |          |                   |                 |  <Record Rx|
       |          |                   |                 |   frames at|
       |          |                   |                 |  TP2 (Rn1)>|
       |          |                   |                 |            |
       |          |                   |                 |    <Repeat |
       |          |                   |                 | Step1 with |
       |          |                   |                 |2% incorrect|
       |          |                   |                 | PACKET_INs>|
       |          |                   |                 |            |
       |          |                   |                 |  <Record Rx|
       |          |                   |                 |   frames at|
       |          |                   |                 |  TP2 (Rn2)>|
       |          |                   |                 |            |
```

Legend:

```
         G-ARP: Gratuitous ARP
         PACKET_IN(Sa+1..Sn,Da+1..Dn): OpenFlow PACKET_IN with wrong
               version number
         Rn1: Total number of frames received at Test Port 2 with
              1% incorrect frames
         Rn2: Total number of frames received at Test Port 2 with
              2% incorrect frames
```

Discussion:

The traffic rate sent towards OpenFlow switch from Test Port 1 should be 1% higher than the Path Programming Rate. Rn1 will provide the Path Provisioning Rate of controller at 1% of incorrect frames handling and Rn2 will provide the Path Provisioning Rate of controller at 2% of incorrect frames handling.

The procedure defined above provides test steps to determine the effect of handling error packets on Path Programming Rate. Same procedure can be adopted to determine the effects on other performance tests listed in this benchmarking tests.

B.6.2. Denial of Service Handling

Procedure:

```
   Test Traffic  Test Traffic   Network Devic    OpenFlow        SDN
   Generator TP1 Generator TP2                  Controller  Application
        |          |                   |                 |            |
        |          |G-ARP (D1..Dn)     |                 |            |
        |          |------------------>|                 |            |
        |          |                   |                 |            |
        |          |                   |PACKET_IN(D1..Dn)|            |
        |          |                   |---------------->|            |
        |          |                   |                 |            |
        |Traffic (S1..Sn,D1..Dn)       |                 |            |
        |----------------------------->|                 |            |
        |          |                   |                 |            |
        |          |                   |PACKET_IN(S1..Sn,|            |
        |          |                   |          D1..Dn)|            |
        |          |                   |---------------->|            |
        |          |                   |                 |            |
        |          |                   |TCP SYN Attack   |            |
        |          |                   |from a switch    |            |
        |          |                   |---------------->|            |
        |          |                   |                 |            |
        |          |                   |FLOW_MOD(D1..Dn) |            |
        |          |                   |<----------------|            |
        |          |                   |                 |            |
        |          |                   | FLOW_MOD(S1..Sn)|            |
        |          |                   |      OFP headers|            |
        |          |                   |<----------------|            |
        |          |                   |                 |            |
        |          |Traffic (S1..Sn,   |                 |            |
        |          |            D1..Dn)|                 |            |
        |          |<------------------|                 |            |
        |          |                   |                 |            |
        |          |                   |                 |  <Wait for |
        |          |                   |                 |      Test  |
        |          |                   |                 |   Duration>|
        |          |                   |                 |            |
        |          |                   |                 |  <Record Rx|
        |          |                   |                 |   frames at|
        |          |                   |                 |  TP2 (Rn1)>|
        |          |                   |                 |            |
```

Legend:

         G-ARP: Gratuitous ARP

Discussion:

TCP SYN attack should be launched from one of the emulated/simulated OpenFlow Switch. Rn1 provides the Path Programming Rate of controller uponhandling denial of service attack.

The procedure defined above provides test steps to determine the effect of handling denial of service on Path Programming Rate. Same procedure can be adopted to determine the effects on other performance tests listed in this benchmarking tests.

## B.7. Reliability

### B.7.1. Controller Failover Time

Procedure:

```
   Test Traffic  Test Traffic  Network Device    OpenFlow        SDN
   Generator TP1 Generator TP2                  Controller  Application
       |             |             |                 |               |
       |             |G-ARP (D1)   |                 |               |
       |             |------------>|                 |               |
       |             |             |                 |               |
       |             |             |PACKET_IN(D1)    |               |
       |             |             |---------------->|               |
       |             |             |                 |               |
       |Traffic (S1..Sn,D1)        |                 |               |
       |-------------------------->|                 |               |
       |             |             |                 |               |
       |             |             |                 |               |
       |             |             |PACKET_IN(S1,D1) |               |
       |             |             |---------------->|               |
       |             |             |                 |               |
       |             |             |FLOW_MOD(D1)     |               |
       |             |             |<----------------|               |
       |             |             |FLOW_MOD(S1)     |               |
       |             |             |<----------------|               |
       |             |             |                 |               |
       |             |Traffic (S1,D1)|               |               |
       |             |<------------|                 |               |
       |             |             |                 |               |
       |             |             |PACKET_IN(S2,D1) |               |
       |             |             |---------------->|               |
       |             |             |                 |               |
       |             |             |FLOW_MOD(S2)     |               |
       |             |             |<----------------|               |
       |             |             |                 |               |
       |             |             |PACKET_IN(Sn-1,D1)|              |
       |             |             |---------------->|               |
       |             |             |                 |               |
       |             |             |PACKET_IN(Sn,D1) |               |
       |             |             |---------------->|               |
       |             |             |       .         |               |
       |             |             |       .         |<Bring down the|
       |             |             |       .         |active control-|
       |             |             |                 |       ler>    |
       |             |             |  FLOW_MOD(Sn-1) |               |
       |             |             |    <-X----------|               |
       |             |             |                 |               |
       |             |             |FLOW_MOD(Sn)     |               |
       |             |             |<----------------|               |
       |             |             |                 |               |
       |             |Traffic (Sn,D1)|               |               |
       |             |<------------|                 |               |
       |             |             |                 |               |
       |             |             |                 |<Stop the test |
       |             |             |                 |after recv.    |
       |             |             |                 |traffic upon   |
       |             |             |                 | failure>      |
```

Legend:

         G-ARP: Gratuitous ARP.

Discussion:

The time difference between the last valid frame received before the traffic loss and the first frame received after the traffic loss will provide the controller failover time.

If there is no frame loss during controller failover time, the controller failover time can be deemed negligible.

## B.7.2. Network Re-Provisioning Time

Procedure:

```
   Test Traffic  Test Traffic   Network Devices  OpenFlow        SDN
   Generator TP1 Generator TP2                 Controller   Application
       |             |               |                 |              |
       |             |G-ARP (D1)     |                 |              |
       |             |-------------->|                 |              |
       |             |               |                 |              |
       |             |               |PACKET_IN(D1)    |              |
       |             |               |---------------->|              |
       |             G-ARP (S1)      |                 |              |
       |---------------------------->|                 |              |
       |             |               |                 |              |
       |             |               |PACKET_IN(S1)    |              |
       |             |               |---------------->|              |
       |             |               |                 |              |
       |Traffic (S1,D1,Seq.no (1..n))|                 |              |
       |---------------------------->|                 |              |
       |             |               |                 |              |
       |             |               |PACKET_IN(S1,D1) |              |
       |             |               |---------------->|              |
       |             |               |                 |              |
       |             |Traffic (D1,S1,|                 |              |
       |             | Seq.no (1..n))|                 |              |
       |             |-------------->|                 |              |
       |             |               |                 |              |
       |             |               |PACKET_IN(D1,S1) |              |
       |             |               |---------------->|              |
       |             |               |                 |              |
       |             |               |FLOW_MOD(D1)     |              |
       |             |               |<----------------|              |
       |             |               |                 |              |
       |             |               |FLOW_MOD(S1)     |              |
       |             |               |<----------------|              |
       |             |               |                 |              |
       |             |Traffic (S1,D1,|                 |              |
       |             |     Seq.no(1))|                 |              |
       |             |<--------------|                 |              |
       |             |               |                 |              |
       |             |Traffic (S1,D1,|                 |              |
       |             |     Seq.no(2))|                 |              |
       |             |<--------------|                 |              |
       |             |               |                 |              |
       |             |               |                 |              |
       |    Traffic (D1,S1,Seq.no(1))|                 |              |
       |<----------------------------|                 |              |
       |             |               |                 |              |
       |    Traffic (D1,S1,Seq.no(2))|                 |              |
       |<----------------------------|                 |              |
       |             |               |                 |              |
       |    Traffic (D1,S1,Seq.no(x))|                 |              |
       |<----------------------------|                 |              |
       |             |               |                 |              |
       |             |Traffic (S1,D1,|                 |              |
       |             |     Seq.no(x))|                 |              |
       |             |<--------------|                 |              |
       |             |               |                 |              |
       |             |               |                 |              |
       |             |               |                 |  <Bring down |
       |             |               |                 | the switch in|
       |             |               |                 |active traffic|
       |             |               |                 |       path>  |
       |             |               |                 |              |
       |             |               |PORT_STATUS(Sa)  |              |
       |             |               |----------------->|              |
       |             |               |                  |              |
       |             |Traffic (S1,D1,|                  |              |
       |             |   Seq.no(n-1))|                  |              |
       |             |  X<-----------|                  |              |
       |             |               |                  |              |
       |  Traffic (D1,S1,Seq.no(n-1))|                  |              |
       |    X------------------------|                  |              |
       |             |               |                  |              |
       |             |               |                  |              |
       |             |               |FLOW_MOD(D1)      |              |
       |             |               |<-----------------|              |
       |             |               |                  |              |
       |             |               |FLOW_MOD(S1)      |              |
       |             |               |<-----------------|              |
       |             |               |                  |              |
       |    Traffic (D1,S1,Seq.no(n))|                  |              |
       |<----------------------------|                  |              |
       |             |               |                  |              |
       |             |Traffic (S1,D1,|                  |              |
       |             |     Seq.no(n))|                  |              |
       |             |<--------------|                  |              |
       |             |               |                  |              |
       |             |               |                  |<Stop the test|
       |             |               |                  |  after recv. |
       |             |               |                  |  traffic upon|
       |             |               |                  |   failover>  |
```

Legend:

G-ARP: Gratuitous ARP message.
Seq.no: Sequence number.
Sa: Neighbor switch of the switch that was brought down.

Discussion:

The time difference between the last valid frame received before the traffic loss (Packet number with sequence number x) and the first frame received after the traffic loss (packet with sequence number n) will provide the network path re-provisioning time.

Note that the trial is valid only when the controller provisions the alternate path upon network failure.

## Authors' Addresses

Bhuvaneswaran Vengainathan
Veryx Technologies Inc.
1 International Plaza, Suite 550
Philadelphia
PA 19113

Email: bhuvaneswaran.vengainathan@veryxtech.com

Anton Basil
Veryx Technologies Inc.
1 International Plaza, Suite 550
Philadelphia
PA 19113

Email: anton.basil@veryxtech.com

Mark Tassinari
Hewlett-Packard,
8000 Foothills Blvd,
Roseville, CA 95747

Email: mark.tassinari@hpe.com

Vishwas Manral
Nano Sec,
CA

Email: vishwas.manral@gmail.com

Sarah Banks
VSS Monitoring
930 De Guigne Drive,
Sunnyvale, CA

Email: sbanks@encrypted.net

Internet-Draft
Network Working Group
Intended Status: Informational
Expires: August 25, 2018

Bhuvaneswaran Vengainathan
Anton Basil
Veryx Technologies
Mark Tassinari
Hewlett-Packard
Vishwas Manral
Nano Sec
Sarah Banks
VSS Monitoring
February 25, 2018

# Terminology for Benchmarking SDN Controller Performance
draft-ietf-bmwg-sdn-controller-benchmark-term-09

## Abstract

This document defines terminology for benchmarking an SDN controller’s control plane performance. It extends the terminology already defined in RFC 7426 for the purpose of benchmarking SDN controllers. The terms provided in this document help to benchmark SDN controller’s performance independent of the controller’s supported protocols and/or network services. A mechanism for benchmarking the performance of SDN controllers is defined in the companion methodology document I-D sdn-controller-benchmark-meth. These two documents provide a standard mechanism to measure and evaluate the performance of various controller implementations.

## Status of this Memo

## Copyright Notice

## Table of Contents

## 1. Introduction

Software Defined Networking (SDN) is a networking architecture in which network control is decoupled from the underlying forwarding function and is placed in a centralized location called the SDN controller. The SDN controller provides an abstraction of the underlying network and offers a global view of the overall network to applications and business logic. Thus, an SDN controller provides the flexibility to program, control, and manage network behaviour dynamically through standard interfaces. Since the network controls are logically centralized, the need to benchmark the SDN controller performance becomes significant. This document defines terms to benchmark various controller designs for performance, scalability, reliability and security, independent of northbound and southbound protocols.

Conventions used in this document

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.

## 2. Term Definitions

### 2.1. SDN Terms

The terms defined in this section are extensions to the terms defined in [RFC7426] "Software-Defined Networking (SDN): Layers and Architecture Terminology". This RFC should be referred before attempting to make use of this document.

#### 2.1.1. Flow

Definition:
The definition of Flow is same as microflows defined in [RFC4689] Section 3.1.5.

Discussion:
A flow can be set of packets having same source address, destination address, source port and destination port, or any of these combinations.

Measurement Units:
N/A

See Also:
   None

### 2.1.2. Northbound Interface

Definition:
   The definition of northbound interface is same the Service Interface defined in [RFC7426].

Discussion:
   The northbound interface allows SDN applications and orchestration systems to program and retrieve the network information through the SDN controller.

Measurement Units:
   N/A

See Also:
   None

### 2.1.3. Southbound Interface

Definition:
   The southbound interface is the application programming interface provided by the SDN controller to interact with the SDN nodes.

Discussion:
   Southbound interface enables controller to interact with the SDN nodes in the network for dynamically defining the traffic forwarding behaviour.

Measurement Units:
   N/A

See Also:
   None

### 2.1.4. Controller Forwarding Table

Definition:
   A controller forwarding table contains flow entries learned in one of two ways: first, entries could be learned from traffic received through the data plane, or second, these entries could be statically provisioned on the controller and distributed to devices via the southbound interface.

Discussion:
The controller forwarding table has an aging mechanism which will be applied only for dynamically learned entries.

Measurement Units:
N/A

See Also:
None

## 2.1.5. Proactive Flow Provisioning Mode

Definition:
Controller programming flows in Network Devices based on the flow entries provisioned through controller’s northbound interface.

Discussion:
Network orchestration systems and SDN applications can define the network forwarding behaviour by programming the controller using proactive flow provisioning. The controller can then program the Network Devices with the pre-provisioned entries.

Measurement Units:
N/A

See Also:
None

## 2.1.6. Reactive Flow Provisioning Mode

Definition:
Controller programming flows in Network Devices based on the traffic received from Network Devices through controller’s southbound interface

Discussion:
The SDN controller dynamically decides the forwarding behaviour based on the incoming traffic from the Network Devices. The controller then programs the Network Devices using Reactive Flow Provisioning.

Measurement Units:
N/A

See Also:
None

## 2.1.7. Path

Definition:
Refer to Section 5 in [RFC2330]

Discussion:
None

Measurement Units:
N/A

See Also:
None

## 2.1.8. Standalone Mode

Definition:
Single controller handling all control plane functionalities without redundancy, or the ability to provide high availability and/or automatic failover.

Discussion:
In standalone mode, one controller manages one or more network domains.

Measurement Units:
N/A

See Also:
None

## 2.1.9. Cluster/Redundancy Mode

Definition:
A group of 2 or more controllers handling all control plane functionalities.

Discussion:
In cluster mode, multiple controllers are teamed together for the purpose of load sharing and/or high availability. The controllers in the group may work in active/standby (master/slave) or active/active (equal) mode depending on the intended purpose.

Measurement Units:

N/A

See Also:
None

### 2.1.10. Asynchronous Message

Definition:
Any message from the Network Device that is generated for network events.

Discussion:
Control messages like flow setup request and response message is classified as asynchronous message. The controller has to return a response message. Note that the Network Device will not be in blocking mode and continues to send/receive other control messages.

Measurement Units:
N/A

See Also:
None

### 2.1.11. Test Traffic Generator

Definition:
Test Traffic Generator is an entity that generates/receives network traffic.

Discussion:
Test Traffic Generator typically connects with Network Devices to send/receive real-time network traffic.

Measurement Units:
N/A

See Also:
None

### 2.1.12. Leaf-Spine Topology

Definition:
   Leaf-Spine is a two layered network topology, where a series of leaf switches, form the access layer, are fully meshed to a series of spine switches that form the backbone layer.

Discussion:
   In Leaf-Spine Topology, every leaf switch is connected to each of the spine switches in the topology.

Measurement Units:
   N/A

See Also:
   None

## 2.2. Test Configuration/Setup Terms

### 2.2.1. Number of Network Devices

Definition:
   The number of Network Devices present in the defined test topology.

Discussion:
   The Network Devices defined in the test topology can be deployed using real hardware or emulated in hardware platforms.

Measurement Units:
   N/A

See Also:
   None

### 2.2.2. Trial Repetition

Definition:
   The number of times the test needs to be repeated.

Discussion:
   The test needs to be repeated for multiple iterations to obtain a reliable metric. It is recommended that this test SHOULD be performed for at least 10 iterations to increase the confidence in measured result.

Measurement Units:
   N/A

See Also:
   None

### 2.2.3. Trial Duration

Definition:
Defines the duration of test trials for each iteration.

Discussion:
Trial duration forms the basis for stop criteria for benchmarking tests. Trials not completed within this time interval is considered as incomplete.

Measurement Units:
seconds

See Also:
None

### 2.2.4. Number of Cluster nodes

Definition:
   Defines the number of controllers present in the controller cluster.

Discussion:
   This parameter is relevant when testing the controller performance in clustering/teaming mode. The number of nodes in the cluster MUST be greater than 1.

Measurement Units:
   N/A

See Also:
   None

## 2.3. Benchmarking Terms

This section defines metrics for benchmarking the SDN controller. The procedure to perform the defined metrics is defined in the accompanying methodology document[I-D.sdn-controller-benchmark-meth]

## 2.3.1. Performance

### 2.3.1.1. Network Topology Discovery Time

Definition:

The time taken by controller(s) to determine the complete network topology, defined as the interval starting with the first discovery message from the controller(s) at its Southbound interface, ending with all features of the static topology determined.

Discussion:

Network topology discovery is key for the SDN controller to provision and manage the network. So it is important to measure how quickly the controller discovers the topology to learn the current network state. This benchmark is obtained by presenting a network topology (Tree, Mesh or Linear) with the given number of nodes to the controller and wait for the discovery process to complete. It is expected that the controller supports network discovery mechanism and uses protocol messages for its discovery process.

Measurement Units:

milliseconds

See Also:

None

### 2.3.1.2. Asynchronous Message Processing Time

Definition:

The time taken by controller(s) to process an asynchronous message, defined as the interval starting with an asynchronous message from a network device after the discovery of all the devices by the controller(s), ending with a response message from the controller(s) at its Southbound interface.

Discussion:

For SDN to support dynamic network provisioning, it is important to measure how quickly the controller responds to an event triggered from the network. The event could be any notification messages generated by an Network Device upon arrival of a new flow, link down etc. This benchmark is obtained by sending asynchronous messages from every connected Network Devices one at a time for the defined trial duration. This test assumes that the controller will respond to the received asynchronous message.

Measurement Units:
   milliseconds

See Also:
   None

### 2.3.1.3. Asynchronous Message Processing Rate

Definition:
   The number responses to asynchronous messages (such as new flow arrival notification message, etc.) for which the controller(s) performed processing and replied with a valid and productive (non-trivial) response message.

Discussion:
   As SDN assures flexible network and agile provisioning, it is important to measure how many network events the controller can handle at a time. This benchmark is obtained by sending asynchronous messages from every connected Network Device at the rate that the controller processes (without dropping them). This test assumes that the controller responds to all the received asynchronous messages (the messages can be designed to elicit individual responses).

   When sending asynchronous messages to the controller(s) at high rates, some messages or responses may be discarded or corrupted and require retransmission to controller(s). Therefore, a useful qualification on Asynchronous Message Processing Rate is whether the in-coming message count equals the response count in each trial. This is called the Loss-free Asynchronous Message Processing Rate.

   Note that several of the early controller benchmarking tools did not consider lost messages, and instead report the maximum response rate. This is called the Maximum Asynchronous Message Processing Rate.

   To characterize both the Loss-free and Maximum Rates, a test could begin the first trial by sending asynchronous messages to the controller(s) at the maximum possible rate and record the message reply rate and the message loss rate. The message sending rate is then decreased by the step-size. The message reply rate and the message loss rate are recorded. The test ends with a trial where the controller(s) processes the all asynchronous messages sent without loss. This is the Loss-free Asynchronous Message Processing Rate.

The trial where the controller(s) produced the maximum response rate is the Maximum Asynchronous Message Processing Rate. Of course, the first trial could begin at a low sending rate with zero lost responses, and increase until the Loss-free and Maximum Rates are discovered.

Measurement Units:
Messages processed per second.

See Also:
None

### 2.3.1.4. Reactive Path Provisioning Time

Definition:
The time taken by the controller to setup a path reactively between source and destination node, defined as the interval starting with the first flow provisioning request message received by the controller(s), ending with the last flow provisioning response message sent from the controller(s) at its Southbound interface.

Discussion:
As SDN supports agile provisioning, it is important to measure how fast that the controller provisions an end-to-end flow in the dataplane. The benchmark is obtained by sending traffic from a source endpoint to the destination endpoint, finding the time difference between the first and the last flow provisioning message exchanged between the controller and the Network Devices for the traffic path.

Measurement Units:
milliseconds.

See Also:
None

### 2.3.1.5. Proactive Path Provisioning Time

Definition:
The time taken by the controller to proactively setup a path between source and destination node, defined as the interval starting with the first proactive flow provisioned in the controller(s) at its Northbound interface, ending with the last flow provisioning command message sent from the controller(s) at its Southbound interface.

Discussion:

For SDN to support pre-provisioning of traffic path from application, it is important to measure how fast that the controller provisions an end-to-end flow in the dataplane. The benchmark is obtained by provisioning a flow on controller's northbound interface for the traffic to reach from a source to a destination endpoint, finding the time difference between the first and the last flow provisioning message exchanged between the controller and the Network Devices for the traffic path.

Measurement Units:
milliseconds.

See Also:
None

### 2.3.1.6. Reactive Path Provisioning Rate

Definition:
The maximum number of independent paths a controller can concurrently establish per second between source and destination nodes reactively, defined as the number of paths provisioned per second by the controller(s) at its Southbound interface for the flow provisioning requests received for path provisioning at its Southbound interface between the start of the trial and the expiry of given trial duration.

Discussion:
For SDN to support agile traffic forwarding, it is important to measure how many end-to-end flows that the controller could setup in the dataplane. This benchmark is obtained by sending traffic each with unique source and destination pairs from the source Network Device and determine the number of frames received at the destination Network Device.

Measurement Units:
Paths provisioned per second.

See Also:
None

### 2.3.1.7. Proactive Path Provisioning Rate

Definition:
Measure the maximum number of independent paths a controller can concurrently establish per second between source and destination nodes proactively, defined as the number of paths provisioned per

second by the controller(s) at its Southbound interface for the paths provisioned in its Northbound interface between the start of the trial and the expiry of given trial duration.

Discussion:

For SDN to support pre-provisioning of traffic path for a larger network from the application, it is important to measure how many end-to-end flows that the controller could setup in the dataplane. This benchmark is obtained by sending traffic each with unique source and destination pairs from the source Network Device. Program the flows on controller’s northbound interface for traffic to reach from each of the unique source and destination pairs and determine the number of frames received at the destination Network Device.

Measurement Units:

Paths provisioned per second.

See Also:

None

### 2.3.1.8. Network Topology Change Detection Time

Definition:

The amount of time required for the controller to detect any changes in the network topology, defined as the interval starting with the notification message received by the controller(s) at its Southbound interface, ending with the first topology rediscovery messages sent from the controller(s) at its Southbound interface.

Discussion:

In order to for the controller to support fast network failure recovery, it is critical to measure how fast the controller is able to detect any network-state change events. This benchmark is obtained by triggering a topology change event and measuring the time controller takes to detect and initiate a topology re-discovery process.

Measurement Units:

milliseconds

See Also:

None

### 2.3.2. Scalability

#### 2.3.2.1. Control Sessions Capacity

Definition:

Measure the maximum number of control sessions the controller can maintain, defined as the number of sessions that the controller can accept from network devices, starting with the first control session, ending with the last control session that the controller(s) accepts at its Southbound interface.

Discussion:

Measuring the controller’s control sessions capacity is important to determine the controller’s system and bandwidth resource requirements. This benchmark is obtained by establishing control session with the controller from each of the Network Device until it fails. The number of sessions that were successfully established will provide the Control Sessions Capacity.

Measurement Units:

N/A

See Also:

None

#### 2.3.2.2. Network Discovery Size

Definition:

Measure the network size (number of nodes and links) that a controller can discover, defined as the size of a network that the controller(s) can discover, starting from a network topology given by the user for discovery, ending with the topology that the controller(s) could successfully discover.

Discussion:

For optimal network planning, it is key to measure the maximum network size that the controller can discover. This benchmark is obtained by presenting an initial set of Network Devices for discovery to the controller. Based on the initial discovery, the number of Network Devices is increased or decreased to determine the maximum nodes that the controller can discover.

Measurement Units:

N/A

See Also:
None

#### 2.3.2.3. Forwarding Table Capacity

Definition:
The maximum number of flow entries that a controller can manage in its Forwarding table.

Discussion:
It is significant to measure the capacity of controller’s Forwarding Table to determine the number of flows that controller could forward without flooding/dropping. This benchmark is obtained by continuously presenting the controller with new flow entries through reactive or proactive flow provisioning mode until the forwarding table becomes full. The maximum number of nodes that the controller can hold in its Forwarding Table will provide Forwarding Table Capacity.

Measurement Units:
Maximum number of flow entries managed.

See Also:
None

### 2.3.3. Security

#### 2.3.3.1. Exception Handling

Definition:
To determine the effect of handling error packets and notifications on performance tests.

Discussion:
This benchmark test is to be performed after obtaining the baseline performance of the performance tests defined in Section 2.3.1. This benchmark determines the deviation from the baseline performance due to the handling of error or failure messages from the connected Network Devices.

Measurement Units:
N/A

See Also:
None

#### 2.3.3.2. Denial of Service Handling

Definition:
   To determine the effect of handling denial of service (DoS) attacks
   on performance and scalability tests.

Discussion:
   This benchmark test is to be performed after obtaining the baseline
   performance of the performance and scalability tests defined in
   section 2.3.1 and section 2.3.1. This benchmark determines the
   deviation from the baseline performance due to the handling of
   denial of service attacks on controller.

Measurement Units:
   Deviation of baseline metrics while handling Denial of Service
   Attacks.

See Also:
   None

### 2.3.4. Reliability

#### 2.3.4.1. Controller Failover Time

Definition:
   The time taken to switch from an active controller to the backup
   controller, when the controllers work in redundancy mode and the
   active controller fails, defined as the interval starting with the
   active controller bringing down, ending with the first re-discovery
   message received from the new controller at its Southbound
   interface.

Discussion:
   This benchmark determine the impact of provisioning new flows when
   controllers are teamed and the active controller fails.

Measurement Units:
   milliseconds.

See Also:
   None

#### 2.3.4.2. Network Re-Provisioning Time

Definition:

The time taken to re-route the traffic by the Controller, when there is a failure in existing traffic paths, defined as the interval starting from the first failure notification message received by the controller, ending with the last flow re-provisioning message sent by the controller at its Southbound interface .

Discussion:

This benchmark determines the controller’s re-provisioning ability upon network failures. This benchmark test assumes the following:

1. Network topology supports redundant path between source and destination endpoints.
2. Controller does not pre-provision the redundant path.

Measurement Units:

milliseconds.

See Also:

None

## 3. Test Setup

This section provides common reference topologies that are later referred to in individual tests defined in the companion methodology document.

## 3.1. Test setup - Controller working in Standalone Mode

```
          +-----------------------------------------------------------+
          |               Application Plane Test Emulator             |
          |                                                           |
          |        +-----------------+      +-------------+           |
          |        |   Application   |      |   Service   |           |
          |        +-----------------+      +-------------+           |
          |                                                           |
          +-----------------------------+(I2)-------------------------+
                                        |
                                        | (Northbound interface)
                       +-------------------------------+
                       |       +----------------+      |
                       |       | SDN Controller |      |
                       |       +----------------+      |
                       |                               |
                       |    Device Under Test (DUT)    |
                       +-------------------------------+
                                        | (Southbound interface)
                                        |
          +-----------------------------+(I1)-------------------------+
          |                                                           |
          |             +-----------+     +-----------+               |
          |             |  Network  |     |  Network  |               |
          |             | Device 2  |--..-| Device n-1|               |
          |             +-----------+     +-----------+               |
          |                     /    \   /    \                       |
          |                    /      \ /      \                      |
          |                l0 /        X        \ ln                  |
          |                  /        / \        \                    |
          |               +-----------+  +-----------+                |
          |               |  Network  |  |  Network  |                |
          |               |  Device 1 |..|  Device n |                |
          |               +-----------+  +-----------+                |
          |                     |              |                      |
          |           +---------------+  +---------------+            |
          |           | Test Traffic  |  | Test Traffic  |            |
          |           |  Generator    |  |  Generator    |            |
          |           |    (TP1)      |  |    (TP2)      |            |
          |           +---------------+  +---------------+            |
          |                                                           |
          |              Forwarding Plane Test Emulator               |
          +-----------------------------------------------------------+
```

Figure 1

## 3.2. Test setup - Controller working in Cluster Mode

```
          +-----------------------------------------------------------+
          |               Application Plane Test Emulator             |
          |                                                           |
          |        +-----------------+      +-------------+           |
          |        |   Application   |      |   Service   |           |
          |        +-----------------+      +-------------+           |
          |                                                           |
          +-----------------------------+(I2)-------------------------+
                                        |
                                        | (Northbound interface)
           +---------------------------------------------------------+
           |                                                         |
           |  ------------------             ------------------      |
           | | SDN Controller 1 | <--E/W--> | SDN Controller n |     |
           |  ------------------             ------------------      |
           |                                                         |
           |                    Device Under Test (DUT)              |
           +---------------------------------------------------------+
                                        | (Southbound interface)
                                        |
          +-----------------------------+(I1)-------------------------+
          |                                                           |
          |             +-----------+     +-----------+               |
          |             |  Network  |     |  Network  |               |
          |             | Device 2  |--..-| Device n-1|               |
          |             +-----------+     +-----------+               |
          |                     /    \   /    \                       |
          |                    /      \ /      \                      |
          |                l0 /        X        \ ln                  |
          |                  /        / \        \                    |
          |               +-----------+  +-----------+                |
          |               |  Network  |  |  Network  |                |
          |               |  Device 1 |..|  Device n |                |
          |               +-----------+  +-----------+                |
          |                     |              |                      |
          |           +---------------+  +---------------+            |
          |           | Test Traffic  |  | Test Traffic  |            |
          |           |  Generator    |  |  Generator    |            |
          |           |    (TP1)      |  |    (TP2)      |            |
          |           +---------------+  +---------------+            |
          |                                                           |
          |              Forwarding Plane Test Emulator               |
          +-----------------------------------------------------------+
```

Figure 2

## 4. Test Coverage

| Lifecycle | Speed | Scalability | Reliability |
|---|---|---|---|
| Setup | 1. Network Topolo- -gy Discovery Time<br>2. Reactive Path Provisioning Time<br>3. Proactive Path Provisioning Time<br>4. Reactive Path Provisioning Rate<br>5. Proactive Path Provisioning Rate | 1. Network Discovery Size | |
| Operational | 1. Maximum Asynchronous Message Proces- -sing Rate<br>2. Loss-Free Asynchronous Message Proces- -sing Rate<br>3. Asynchronous Message Proces- -sing Time | 1. Control Sessions Capacity<br>2. Forwarding Table Capacity | 1. Network Topology Change Detection Time<br>2. Exception Handling<br>3. Denial of Service Handling<br>4. Network Re- Provisioning Time |
| Tear Down | | | 1. Controller Failover Time |

## 5. References

### 5.1. Normative References

[RFC7426] E. Haleplidis, K. Pentikousis, S. Denazis, J. Hadi Salim, D. Meyer, O. Koufopavlou "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, January 2015.

[RFC4689] S. Poretsky, J. Perser, S. Erramilli, S. Khurana "Terminology for Benchmarking Network-layer Traffic Control Mechanisms", RFC 4689, October 2006.

[RFC2330] V. Paxson, G. Almes, J. Mahdavi, M. Mathis, "Framework for IP Performance Metrics", RFC 2330, May 1998.

[RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[RFC8174] B. Leiba, "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", RFC 8174, May 2017.

[I-D.sdn-controller-benchmark-meth] Bhuvaneswaran.V, Anton Basil, Mark.T, Vishwas Manral, Sarah Banks "Benchmarking Methodology for SDN Controller Performance", draft-ietf-bmwg-sdn-controller-benchmark-meth-08 (Work in progress), February 25, 2018

### 5.2. Informative References

[OpenFlow Switch Specification] ONF,"OpenFlow Switch Specification" Version 1.4.0 (Wire Protocol 0x05), October 14, 2013.

## 6. IANA Considerations

This document does not have any IANA requests.

## 7. Security Considerations

Security issues are not discussed in this memo.

## 8. Acknowledgements

The authors would like to acknowledge Al Morton (AT&T) for the significant contributions to the earlier versions of this document. The authors would like to thank the following individuals for providing their valuable comments to the earlier versions of this document: Sandeep Gangadharan (HP), M. Georgescu (NAIST), Andrew McGregor (Google), Scott Bradner , Jay Karthik (Cisco), Ramakrishnan (Dell), Khasanov Boris (Huawei).

## 9. Authors' Addresses

Bhuvaneswaran Vengainathan
Veryx Technologies Inc.
1 International Plaza, Suite 550
Philadelphia
PA 19113

Email: bhuvaneswaran.vengainathan@veryxtech.com

Anton Basil
Veryx Technologies Inc.
1 International Plaza, Suite 550
Philadelphia
PA 19113

Email: anton.basil@veryxtech.com

Mark Tassinari
Hewlett-Packard,
8000 Foothills Blvd,
Roseville, CA 95747

Email: mark.tassinari@hpe.com

Vishwas Manral
Nano Sec,CA

Email: vishwas.manral@gmail.com

Sarah Banks
VSS Monitoring
930 De Guigne Drive,
Sunnyvale, CA

Email: sbanks@encrypted.net

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

Table of Contents

## 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.

```
                                    +----------------+
                                    | SDN Controller |
                                    +----------------+
                                        /         \
                                       /           \
                                      /             \
                                     /               \
                Domain A            /                 \          Domain B
    +--------------------------------+              +----------------------------
-----+
    |                                |              |
     |
    |  +---------+    +----------+   |              |  +---------+     +---------
-+   |
    |  | vSwitch |    | vSwitch  |   |              |  | pSwitch |     | vSwitch
 |   |
    |  +---------+    +----------+   |              |  +---------+     +---------
-+   |
    |                                |              |
     |
    |  +---------+    +----------+   |              |  +---------+     +---------
-+   |
    |  | pSwitch |    | pSwitch  |   |              |  | vSwitch |     | pSwitch
 |   |
    |  +---------+    +----------+   |              |  +---------+     +---------
-+   |
    |              ...               |              |               ...
     |
    +--------------------------------+              +----------------------------
-----+
```

## 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 enought 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

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

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: jisu.park@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/

ï»¿ INTERNET-DRAFT Kishore Tiruveedhula
Intended Status: Informational Sudhin Jacob
Expires: October 7,2018 Juniper Networks
February 26,2018

# Benchmarking Methodology for EVPN and PBB-EVPN
draft-kishjac-bmwg-evpntest-09

## Abstract

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

## Status of this Memo

## Copyright and License Notice

Contents

## 1. Introduction

EVPN is defined in RFC7432 which describes procedures for BGP MPLS-based Ethernet VPNs(EVPN).This document defines the methodologies for benchmarking performance of EVPN. The scope of this document is to provide methodologies for benchmarking EVPN data, control plane MAC learning, MAC flush ,MAC aging, convergence, high availability, scale.
The methodologies defined for EVPN can be used for benchmarking the performance of PBB-EVPN.PBB-EVPN is defined in RFC 7623.It is being deployed in provider network. The difference between PBB-EVPN and EVPN is the former learns the customer MAC in data plane the later learns in control plane.

Conventions used in this document

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

## 1.1 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 Running in SA mode:
 Figure 1




         +-------------+
         |             |
         |             |
         |    R1(SHPE1)|
         |             | + sub interfaces
         |             | +----------------------------------+
         +-----+-------+    Router Tester (IXIA/Spirent) sending
                       |            layer 2 bi directional traffic which acts as
 sender/receiver
               |
               |
               |
          +-------------+
          |    ++       |
          |             |
          |    RR/P     |
          |             |
          |             |
      +---+-------------+----+
      |                      |
      |                      |
      |                      |
+-----+-------+        +-----+-------+
|             |        |             |
|     MHPE1(DUT)       |             |
|             |        |    MHPE2    |
|             |        |             |
|             |        |             |
|             |        +------+------+
+-------------+               |
              |               |
              |     +---------+
      +-------+-------------+
      |                     |
      |         CE1         |
      |   bridge domain     | sub interfaces
      |                     +--------------------------------+
      |                     |
      +---------------------+ Router Tester (IXIA/Spirent) Sending bi
          directional layer 2 traffic with different VLAN acts as sender/receive
r
```

## 3. Network

The network consists of 5 routers and 2 traffic generator ports. The traffic generator ports are connected to R1 and CE, these RT port will be sending uni directional or bi directional for different vlans depends on the test scenario.R1 is also termed as SHPE3 which is a single homed router is running EVPN services. DUT(MHPE1) and MHPE2 are running multihoming EVPN services, the CE acts as a bridge which will send the layer 2 traffic to both DUT and MHPE2. RR is the router reflector which is also acting as provider router. All four routers(MHPE1,MHPE2,RR,R1) except CE are running MPLS,BGP emulating a Service provider scenario. CE is a dual home connected to DUT and MHPE2.The testing will be done on DUT in order to bench mark the EVPN service. DUT and the MHPE2 are running EVPN with SA/AA, In AA EVPN service there will be LAG running between interfaces of CE, DUT and MHPE2 .The DUT and other PE’s will be running N EVI’s (EVPN instances) on <X> sub interfaces.

## 4. PBB-EVPN Network Setup

The network consists of 5 routers and 2 traffic generator ports. The traffic generator ports are connected to R1 and CE, these ports will be sending bi directional or uni directional layer 2 traffic for different vlans depending up on the various test scenarios.
R1 is also termed as SHPE3 which is a single homed router is running PBB-EVPN services. DUT(MHPE1) and MHPE2 are running multihoming PBB-EVPN services ,the CE acts as a bridge which will send the layer 2 traffic to both DUT and MHPE2. RR is the router reflector which is also acting as provider router. All four routers(MHPE1,MHPE2,RR,R1) except CE are running MPLS,BGP emulating a Service provider scenarios. CE is a dual home connected to DUT and MHPE2.The testing will be done on DUT in order to bench mark the PBB-EVPN service. DUT and the MHPE2 are running PBB-EVPN with SA/AA, In AA PBB-EVPN service there will be LAG running between the interfaces of CE, DUT and MHPE2. The DUT and other PE’s will be running <X> EVI’s (PBB-EVPN instances) on <X> sub interfaces.

## 5 Test Cases

The following tests are conducted to measure the time taken to learn the <X> number of MAC’s locally in EVI is "T"sec. 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 time taken will be "T’". The data plane MAC learning can be measured using the parameters defined in RFC 2889 section 5.8.

### 5.1.1 To Record the time taken to learn the MAC address in DUT

Objective:

To Record the time taken to learn the MAC address locally and time taken to send these local learned MAC routes to peers.

a. Send <X> unicast frames from CE to MHPE1(DUT) working in SA mode with different source and destination address. Measure the time taken to learn these MAC in forwarding table and in control plane. The data plane learning is measured using RFC 2889 section 5.8. Sending frames to the limit of bridge domain of particular EVI. Measure the time taken to learn all <X> MAC in data plane/hardware. The Range of MAC is known from RT and this is verified in DUT.

b. Measure the time taken to send these <X> type 2 routes from DUT to its peers.

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.

Measurement

The DUT EVPN MAC table must learn the <X> MACs in data plane in T Time frame. The DUT must send <X> type 2 routes to remote router in T’ Time frame. Repeat the test and plot the data. The data plane measurement is taken by considering DUT as black box the range of X MAC is known from RT and the same is learned in DUT, the time to learn that is measured.

The test is repeated for "N" times and the value is taken by averaging the values.

PBB-EVPN To Record the time taken to learn the MAC address in DUT

Objective:

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

a. Send <X> unicast frames from CE to MHPE1(DUT) working in SA mode with different source and destination address. Measure the time taken to learn these MACs in forwarding table. The data plane learning is measured using RFC 2889 section 5.8.Sending frames to the limit of bridge domain of particular EVI. Measure the time taken to learn all <X> MAC in data plane/hardware. The Range of MAC is known from RT and this is verified in DUT.

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.

Measurement

The DUT MAC table must learn the <X> MACs in data plane in T time frame. Repeat the test and plot the data. The data plane measurement is taken   by considering DUT as black box the range of <X> MAC is known from RT and the same is learned in DUT, the time to learn <X> MAC is measured. The test is repeated for "N" times and the value is taken by averaging the values.

### 5.1.2. To Record the time taken to learn remote MACs in DUT which is advertised by remote peer

Objective:

Send <X> frames with different SA and DA to R1 from RT Measure the time taken to learn these <X> MACs from remote peer in DUT and program the EVPN MAC address table. The DUT and MHPE2 are running SA mode.

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.

Measurement:

The DUT MAC table must learn the <X> MAC address in T time frame. Repeat these test and plot the data. The test is repeated for "N" times and the value is taken by averaging the values.

### PBB-EVPN To Record the time taken to learn <X> MAC’s from remote peer by DUT.

Objective:

Send <X> frames with different SA and DA to R1 from RT. Measure the time taken to learn these <X>  MACs from remote peer and program the MAC address table of the DUT. DUT and MHPE2 are running SA mode.

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 wont be any traffic flow from CE to DUT during this test. The BGP must be in established state.

Measurement:

The DUT MAC table must learn the <X> MAC address in T time frame. Repeat these test and plot the data. The test is repeated for "N" times and the value is taken by averaging the values.

## 5.1.3. To Record the time taken to flush the local entry due to CE link Failure and measure the relearning rate of MACs

Objective:

Send <X> frames with different SA and DA to DUT from CE using traffic generator. Wait till the MHPE2 learns all <X> MAC address. Then fail the MHPE2 CE link and measure the time taken to flush these <X> MACs from the EVPN MAC table and the time taken to relearn it. The DUT and MHPE2 are running SA mode. In this scenario MHPE2 is the Designated forwarder which learns mac and advertises to 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. 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.

Measurement:

Measure the time taken for flushing these <X> MAC address. Measure the time taken to relearn the <X> MACs in DUT. Repeat the test and plot the data.

PBB-EVPN To Record the time taken to flush the local entry due to CE link failu re

Objective:

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

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

Measurement:

The DUT MAC table must learn the <X> MAC address and measure the time taken for flushing these X MAC address. Measure the time taken to relearn these <X> MACs in DUT. Repeat the test and plot the data.

## 5.1.4. To Record the time taken by DUT to flush MACs learned from R1 during R1 traffic generator link failure

Objective:

Send <X> frames with 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.

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.

Measurement:

Measure the time taken to flush <X> remote MACs from EVPN MAC table of DUT. Repeat the test and plot the data.

PBB-EVPN To Record the time taken by DUT to flush MACs learned from R1 during R1 traffic generator link failure

Objective:

Send <X> frames with 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.

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.

Measurement:

Measure the time taken to flush <X> remote MACs from PBB-EVPN MAC table of DUT. Repeat the test and plot the data.

5.1.5. To measure the MAC ageing time.

Objective:

Send <X> frames with 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 ageing. The DUT and MHPE2 are running SA mode.

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.

Measurement:

Measure the time taken to flush <X> MAC address due to ageing. Repeat the test and plot the data.

PBB-EVPN To measure the MAC ageing time.

Objective:

Send X frames with 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 ageing. The DUT and MHPE2 running in SA mode

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 uni directional flowing from CE to DUT.

Measurement:

Measure the time taken to flush <X> MAC address due to ageing. Repeat the test and plot the data.

5.1.6. To Record the time taken by DUT to age X routes learned from remote PE after stopping the traffic at remote PE.

Objective:

Send X frames with different SA and DA to DUT from R1 using traffic generator. Stop the traffic at remote PE R1. Due to MAC ageing 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

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 wont be any traffic from CE.

Measurement:

Measure the time taken to flush <X> remote MACs learned in DUT EVPN MAC table due to ageing. Repeat the test and plot the data.

PBB-EVPN To Record the time taken by DUT to age <X> MAC from remote PE after stopping the traffic at remote PE.

Objective:

Send <X> frames with 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.

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.

Measurement:

Measure the time taken to flush the <X> remote MACs from DUT PBB-EVPN MAC table due to ageing. Repeat the test and plot the data.

### 5.1.7. To Record the time taken by DUT to learn routes from local and remote.

Objective:

Send <X> frames with 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.

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.

Measurement:

Measure the time taken to learn <2X> MAC address in DUT EVPN MAC table. Repeat the test and plot the data.

PBB-EVPN To Record the time taken by DUT to learn <X> MACs from local and <X> from remote.

Objective:

Send X frames with 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.

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.

Measurement:

Measure the time taken to learn 2X MAC address table in DUT PBB-EVPN MAC table. Repeat the test and plot the data.

## 5.2 High Availability

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

Objective:

Send <X> frames from CE to DUT from traffic generator with different SA and DA. Send <X> frames from traffic generator to R1 with 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 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.

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 and plot the graph.

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

Objective:

Send <X> frames to DUT with different SA and DA from CE using the traffic generator. Send <X> frames from traffic generator to R1 with 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 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.

Measurement:

There should be 0 packet loss which is the ideal case, No change in the DF role. There should not be any withdraw of routes from DUT.

## 5.3 ARP/ND Scaling

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

EVPN: To Record the ARP/ND scale of the DUT with gateway IRB configured.

Objective:

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.

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.

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

6.1. To Scale the DUT to N EVI and clear BGP in DUT without traffic.

Objective:

The DUT, MHPE2 and R1 are scaled to "N" EVI. Clear BGP neighbors of the DUT. Once the adjacency is established in DUT. Measure the routes received from remote routers MHPE2 and R1 for <N> EVIs in the 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. Once the BGP is established. Record the DUT EVPN table. For MHPE,DUT ESI must be configured per IFD/Interface.

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.

PBB-EVPN To Scale the DUT to N PBB-EVPN instances and clear BGP in DUT without traffic.

Objective:

The DUT, MHPE2 and R1 are scaled to "N" PBB-EVI. Clear BGP neighbors of the DUT. Once the adjacency is established in DUT. Measure the routes received from remote routers MHPE2 and R1 for <N> EVIs in the DUT.

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 is established. Record the DUT PBB-EVPN table. For MH PE ESI must be configured per IFD/Interface.

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.

## 6.2. To Scale the DUT to N EVI and clear BGP in DUT with traffic. Measure the convergence time

Objective:

Scale <N> EVI’s in DUT,R1 and MHPE2.Send F frames to DUT from CE using traffic generator with different SA and DA for N EVI’s. Send <F> frames from traffic generator to R1 with different SA and DA.
There will be <2F> 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 all <2F> MAC address in DUT MAC table.

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.

Measurement:

The DUT must learn all <2F> MAC address. Measure the time taken to learn 2F MAC in DUT, measure the flood traffic time "t" of DUT

PBB-EVPN To Scale the DUT to "N" PBB-EVPN instances and clear BGP in DUT with traffic. Measure the convergence time

Objective:

Scale <N> PBB-EVI’s in DUT,R1 and MHPE2.Send <F> frames to DUT from CE using traffic generator with different SA and DA for N EVI’s. Send <F> frames from traffic generator to R1 with different SA and DA. There will be <2F> 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 all <2F> MAC address in DUT PBB-MAC table.

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.

Measurement:

The DUT must learn all 2F MAC address. Measure the time taken to learn 2F MAC in DUT, measure the flood traffic time "t" of DUT

## 7. Soak Test

### 7.1. To Scale the DUT to N EVI in DUT with traffic and run the set up for 24hrs

Objective:

Scale <N> EVIs in DUT,R1 and MHPE2.Send F frames to DUT from CE using traffic generator with different SA and DA for N EVI’s. Send <F> frames from traffic generator to R1 with different SA and DA. There will be <2F> 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.
.

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.

Measurement:

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

PBB-EVPN To Scale the DUT to N PBB-EVPN instances in DUT with traffic and run the set up for 24hrs

Objective:

Scale <N> PBB-EVI’s in DUT,R1 and MHPE2.Send <F> frames to DUT from CE using traffic generator with different SA and DA for N EVI’s. Send <F> frames from traffic generator to R1 with different SA and DA. There will be <2F> 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.

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.

Measurement:

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

## 8. Acknowledgments

We would like to thank Fioccola Giuseppe of Telecom Italia reviewing our draft and commenting it. We would like to thank Sarah Banks, the work group chair for guiding us in this draft.

## 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, June 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, June 1999,<http://www.rfc-editor.org/info/rfc2544>.

[RFC2889]  R.Mandeville and J. Perser "Benchmarking Methodology for LAN Switching Devices"

### 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, <http://www.rfc-editor.org/info/rfc7432>.

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

## Authors’ Addresses

## Appendix A. Appendix

## Authors’ Addresses

Sudhin Jacob (editor)
Juniper Networks
Bangalore
India

Phone: +91 8061212543
Email: sjacob@juniper.net
sudhinjacob@rediffmail.com

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: August 9, 2018

A. Morton
AT&T Labs

February 5, 2018

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

## Abstract

Fundamental Benchmarking Methodologies for Network Interconnect Devices of interest to the IETF are defined in RFC 2544. This memo updates the provisions 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", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

## Status of This Memo

## Copyright Notice

Table of Contents

## 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.

## 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 recover 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 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 reduced for the next trial.

@@@@ Should a particular search algorithm be included?

@@@@ Should the search include trial repetition whenever frame loss is observed, to avoid the effects of background loss (un-related to buffer overflow)?

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:

- Average (Benchmark)
- Minimum
- Maximum
- 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
```

Corrected DUT Buffer Time =

```
                               Measured Throughput
   Implied DUT Buffer Time * --------------------------
                             Max Theoretical Frame Rate
```

## 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 Klozik of the VSPERF project for many contributions to the testing [VSPERF-b2b].

## 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>.

## 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>.

[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)>.

## 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

BMWG
Internet-Draft
Intended status: Informational
Expires: September 3, 2018

R. Rosa, Ed.
C. Rothenberg
UNICAMP

March 2, 2018

# VNF Benchmarking Methodology
draft-rosa-bmwg-vnfbench-01

## Abstract

This document describes a common methodology for benchmarking Virtualized Network Functions (VNFs) in general-purpose hardware. Specific cases of benchmarking methodologies for particular VNFs can be derived from this document. An open source reference implementation called Gym is reported as a running code embodiment of the proposed methodology for VNFs.

## Status of This Memo

## Copyright Notice

Table of Contents

## 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 [ETS14a], intrinsic considerations for analysis when addressing performance. This document stands as a ground methodology guide for VNF benchmarking. It addresses the state-of-the-art publications and the current developments in similar standardization efforts (e.g., [ETS14c] and [RFC8204]) towards bechmarking VNFs.

## 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 [ETS14b]. 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.

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 on-boarding and managing the life cycle of a VNF instance.

VNF-FG: Virtualized Network Function Forwarding Graph - an ordered list of VNFs creating a service chain.

## 3. Scope

This document assumes VNFs as black boxes when defining VNF benchmarking methodologies. White box approaches are assumed and analysed as a particular case under proper considerations of internal VNF instrumentation.

## 4. Considerations

VNF benchmarking considerations are defined in [RFC8172]. Additionally, VNF pre-deployment testing considerations are well explored in [ETS14c].

### 4.1. VNF Testing Methods

Following the ETSI’s model in [ETS14c], 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 components in varied settings.

```
                              +---------------+
                              |    Manager    |
                Control       | (Coordinator) |
                Interface     +---+-------+---+
             +--------+-----------+       +-------------------+
             |        |                                       |
             |        |   +-------------------------+         |
             |        |   |    System Under Test    |         |
             |        |   |                         |         |
             |        |   |    +-----------------+  |         |
             |     +--+------- +                 |  |         |
             |     |           |       VNF       |  |         |
             |     |           |                 |  |         |
             |     |           +----.---------.--+  |         |
       +-----+---+ |  Monitor  |    :         :     |   +-----+----+
       | Agent   | |{listeners}|----^---------V--+  |   |  Agent   |
       |(Sender) | |           |    Execution    |  |   |(Receiver)|
       |         | |           |   Environment   |  |   |          |
       |{Probers}| +-----------|                 |  |   |{Probers} |
       +-----.---+        |    +----.---------.--+  |   +-----.----+
             :            +---------^---------V-----+         :
             V                      :         :               :
             :................>.....:         :............>..:
             Stimulus Traffic Flow
```

Figure 1: Generic VNF Benchmarking Setup

Agent -- executes active stimulus using probers, benchmarking tools, to benchmark and collect network and system performance metrics. While a single Agent is capable of performing localized benchmarks (e.g., stress tests on CPU, memory, disk I/O), the interaction among distributed Agents enable the generation and collection of end-to-end metrics (e.g., frame loss rate, latency). In a deployment scenario, one Agent can create the benchmark stimuli and the other end be the VNF itself where, for example, one-way latency is evaluated. A prober defines a software/hardware-based tool able to generate traffic specific to a VNF (e.g., sipp) or generic to multiple VNFs (e.g., pktgen). An Agent can be defined by a physical or virtual network function.

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, monitor observes particular properties according to NFVI PoPs and VNFs capabilities. A listener defines one or more interfaces for the extraction of particular metrics monitored in a target VNF and/or execution environment. Logically, a Monitor is defined by as a virtual network function.

Manager -- in a VNF benchmarking deployment scenario, 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 parset benchmark outputs to construct a VNF performance profile, 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 by a physical or virtual network function.

Virtualized Network Function (VNF) -- consists of one or more software components 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 targetting specific technologies.

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 VNF performance profiles.

## 4.3. Deployment Scenarios

A VNF benchmark deployment scenario establishes the physical and/or virtual instantiation of components defined in a VNF benchmarking setup.

Based on a generic VNF benchmarking setup, the following considerations hold for deployment scenarios:

o Components can be composed in a single entity and defined as black or white boxes. For instance, Manager and Agent could jointly define a software entity to perform a VNF benchmark and present results.

- 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.

- 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.

- 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.

- All benchmarking components defined in a deployment scenario must perform the synchronization of clocks to an international time standard.

## 4.4. Influencing Aspects

In general, VNF benchmarks must capture relevant causes of performance variability. Examples of VNF performance influencing aspects can be observed in:

Deployment Scenario Topology: The orchestrated 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 benchmarking stimulus for a VNF and its characteristics of packets format, disposition, 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 benchmark metrics.

## 5. Methodology

Portability as a intrinsic characteristic of VNFs, allow them to be deployed in multiple environments, enabling, even parallel, 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.

### 5.1. General Description

For the sake of clarity and generalization of VNF benchmarking tests, consider the following definitions.

VNF Benchmarking Layout (VNF-BL) -- a setup that specifies a method of how to measure a VNF Performance Profile. The specification includes structural and functional instructions, and variable parameters at different abstractions (e.g., topology of the deployment scenario, benchmarking target metrics, parameters of benchmarking components). VNF-BL may be specific to a VNF or applicable to several VNF types. A VNF-BL can be used to elaborate a VNF benchmark deployment scenario aiming the extraction of particular VNF performance metrics.

VNF Performance Profile: (VNF-PP) -- defines a mapping between VNF allocated capabilities (e.g., cpu, memory) and the VNF performance metrics (e.g., throughput, latency between in/out ports) obtained in a benchmarking test elaborated based on a VNF-BL. Logically, packet processing metrics are presented in a specific format addressing statistical significance where a correspondence among VNF parameters and the delivery of a measured/qualified VNF performance exists.

#### 5.1.1. Configurations

In addition to a VNF-BL, all the items listed below, added their associated, and not limited to, settings must be contained in annotations describing a VNF benchmark deployment scenario. Ideally, any person in possession of such annotations and the necessary/ associated skeleton of hardware and software components should be able to reproduce the same deployment scenario and VNF benchmarking test.

VNF: type, model, version/release, allocated resources, specific parameters, technology requirements, software details.

Execution Environment: type, model, version/release, available resources, technology capabilities, software details.

Agents: toolset of available probers and related benchmarking metrics, workload, traffic formats, virtualization layer (if existent), hardware capabilities (if existent).

Monitors: toolset of available listeners and related monitoring metrics, monitoring target (VNF and/or execution environment), virtualization layer (if existent), hardware capabilities (if existent).

Manager: utilized procedures during the benchmark test, set of events and settings exchanged with Agents/Monitors, established sequence of possible states triggered in the target VNF.

### 5.1.2. Testing Procedures

Consider the following definitions:

Trial: Consists in a single process or iteration to obtain VNF benchmarking metrics as a singular measurement.

Test: Defines strict parameters for benchmarking components perform one or more trials.

Method: Consists of a VNF-BL targeting one or more Tests to achieve VNF benchmarking measurements. A Method explicits ranges of parameter values for the configuration of benchmarking components realized in a Test.

The following sequence of events compose basic general procedures that must be performed for the execution of a VNF benchmarking test.

1. The sketch of a VNF benchmarking setup must be defined to later be translated into a deployment scenario. Such sketch must contain all the structural and functional settings composing a VNF-BL. 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 existent), must be started and configured to execute the benchmark stimuli and retrieve expected/target metrics captured during and at the end of the VNF benchmarking test. One or more trials realize the measurement of VNF performance metrics.

4. Output results from each obtained benchmarking test must be received by Manager. In an automated or manual process, intended metrics to be extracted defined in the VNF-BL must compose a VNF-PP, resulting in a VNF benchmark report.

## 5.2. 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 detain 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-BL.

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, internal events of the VNF might trigger variated behaviors activating functionalities associated with elasticity, such as load balancing. In this terms, a detailed characterization of a VNF must be specified and be contained in the VNF-PP and benchmarking report.

On Failures: Similarly to the case before, benchmarking setups of VNF must also capture the dynamics involved in the VNF behavior. In case of failures, a VNF would restart itself and possibly result in a off-line period. A VNF-PP and benchmarking report must clearly 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 targeted VNF performance metrics.

## 6. VNF Benchmark Report

On the extraction of VNF and execution environment performance metrics various trials must be performed for statistical significance of the obtained benchmarking results. Each trial must be executed following a particular deployment scenario composed by a VNF-BL.

A VNF Benchmarking Report correlates structural and functional parameters of VNF-BL with targeted/extracted VNF benchmarking metrics of the obtained VNF-PP.

A VNF performance profile must address the combined set of classified items in the 3x3 Matrix Coverage defined in [RFC8172].

## 7. Open Source Reference Implementation

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 compounded in multiple practical ways for multiple 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 the 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 Layout (VNF-BL) examples on the vIMS and OVS targets as described in [Rosa-b] and [Rosa-c].

## 8. Security Considerations

TBD

## 9. IANA Considerations

This document does not require any IANA actions.

## 10. Acknowledgement

The authors would like to thank the support of Ericsson Research, Brazil.

## 11. References

### 11.1. Normative References

[ETS14a] 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>.

[ETS14b] 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>.

[ETS14c] 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>.

[RFC1242] S. Bradner, "Benchmarking Terminology for Network Interconnection Devices", July 1991, <https://www.rfc-editor.org/info/rfc1242>.

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

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

## 11.2. Informative References

[Gym] "Gym Home Page", <https://github.com/intrig-unicamp/gym>.

[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>.

## Authors' Addresses

Raphael Vicente Rosa (editor)
University of Campinas
Av. Albert Einstein, 400
Campinas, Sao Paulo 13083-852
Brazil

Email: rvrosa@dca.fee.unicamp.br
URI: https://intrig.dca.fee.unicamp.br/raphaelvrosa/

Christian Esteve Rothenberg
University of Campinas
Av. Albert Einstein, 400
Campinas, Sao Paulo  13083-852
Brazil

Email: chesteve@dca.fee.unicamp.br
URI:   http://www.dca.fee.unicamp.br/~chesteve/

```
BMWG                                                         S. Kommu
Internet-Draft                                                 VMware
Intended status: Informational                                J. Rapp
Expires: July 2018                                             VMware
                                                      January 2, 2018
```

# Considerations for Benchmarking Network Virtualization Platforms
draft-skommu-bmwg-nvp-01.txt

## 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. The purpose of this document is to distinguish and highlight benchmarking consideration
s

when testing and evaluating virtual networking components in the data center.

## Table of Contents

## 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 with their own benchmarks to showcase why a particular solution is better than another. Hence, the need for a vendor and product agnostic way to benchmark multivendor 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 scale limitations of VLANs. Overlays such as VXLAN were designed t
o
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 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 nvo3 working group < https://datatracker.ietf.org/wg/nvo3/documents/> for more information.

Modern application architectures, such as Micro-services, are going beyond the three tier app models such as web, app and db. Benchmarks MUST consider whether the proposed solution is able to scale up to the demands of such applications and not just a three-tier architecture.

## 2. Conventions used in this document

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

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying significance described in RFC 2119.

## 3. Definitions

### 3.1. System Under Test (SUT)

Traditional hardware based networking devices generally use the device under test (DUT) model of testing. In this model, apart fro
m
any allowed configuration, the DUT is a black box from a testing perspective. This method works for hardware based networking devices since the device itself is not influenced by any other components outside the DUT.

Virtual networking components cannot leverage DUT model of testing as the DUT is not just the virtual device but includes the hardware components that were used to host the virtual device

Hence SUT model MUST be used instead of the traditional device unde
r
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 may also work with higher level TCP segments such as TSO. In contrast, all physical switches and routers, including the ones that act as initiators for NVOs, work with L2/L3 packets.

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 does not focus on Network Function Virtualization.

Network Function Virtualization (NFV) focuses on being independent of 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.

Typical NFV implementations emulate in software, the characteristic
s
and features of physical switches. They are similar to any physica
l
L2/L3 switch from the perspective of the packet size, which is typically enforced based on the maximum transmission unit used.

Network Virtualization platforms on the other hand, are 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 verses ~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.

```
   NPV (1 segment)                NFV (40 packets)

   Segment 1                      Packet 1
     +-------------------------+    +-------------------------+
     | Headers                 |    | Headers                 |
     | +---------------------+ |    | +---------------------+ |
     | | Pay Load - upto 64K | |    | | Pay Load < 1500     | |
     | +---------------------+ |    | +---------------------+ |
     +-------------------------+    +-------------------------+

                                  Packet 2
                                    +-------------------------+
                                    | Headers                 |
                                    | +---------------------+ |
                                    | | Pay Load < 1500     | |
                                    | +---------------------+ |
                                    +-------------------------+

                                                .
                                                .
                                                .
                                                .

                                  Packet 40
                                    +-------------------------+
                                    | Headers                 |
                                    | +---------------------+ |
                                    | | Pay Load < 1500     | |
                                    | +---------------------+ |
                                    +-------------------------+
```

Figure 1  Payload NPV vs NFV

Hence, normal benchmarking methods are not relevant to the NVPs.

Instead, newer methods that take into account the built in advantages of TCP provided optimizations MUST be used for testing Network Virtualization Platforms.

## 3.3. Micro-services

Traditional monolithic application architectures such as the three tier web, app and db architectures are hitting scale and deployment limits for the modern use cases.

Micro-services make use of classic unix style of small app with single responsibility.

These small apps are designed with the following characteristics:

Each application only does one thing - like unix tools

Small enough that you could rewrite instead of maintain

Embedded with a simple web container

Packaged as a single executable

Installed as daemons

Each of these applications are completely separate

Interact via uniform interface

REST (over HTTP/HTTPS) being the most common

With Micro-services architecture, a single web app of the three tie
r

application model could now have 100s of smaller apps dedicated to do just one job.

These 100s of small one responsibility only services will MUST be secured into their own segment - hence pushing the scale boundaries of the overlay from both simple segmentation perspective and also from a security perspective

## 4. Scope

This document does not address Network Function Virtualization has been covered already by previous IETF documents (https://datatracker.ietf.org/doc/draft-ietf-bmwg-virtual-net/?include_text=1) the focus of this document is Network Virtualization Platform where the network functions are an intrinsi
c

part of the hypervisor’s TCP stack, working closer to the application layer and leveraging performance optimizations such TSO/RSS provided by the TCP stack and the underlying hardware.

### 4.1. Virtual Networking for Datacenter Applications

While virtualization is growing beyond the datacenter, this documen
t

focuses on the virtual networking for east-west traffic within the datacenter applications only. For example, in a three tier app suc
h

web, app and db, this document focuses on the east-west traffic between web and app. It does not address north-south web traffic accessed from outside the datacenter. A future document would address north-south traffic flows.

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

## 4.2. Interaction with Physical Devices

Virtual network components cannot be tested independent of other components within the system.  Example, unlike a physical router or a firewall, where the tests can be focused directly solely on the device, when testing a virtual router or firewall, multiple other devices may become part of the system under test.  Hence the characteristics of these other traditional networking switches and routers, LB, FW etc. MUST be considered.

! Hashing method used

! Over-subscription rate

! Throughput available

! Latency characteristics

## 5. Interaction with Physical Devices

In virtual environments, System Under Test (SUT) may often share resources and reside on the same Physical hardware with other components involved in the tests.  Hence SUT MUST be clearly defined.  In this tests, a single hypervisor may host multiple servers, switches, routers, firewalls etc.,

Intra host testing:  Intra host testing helps in reducing the numbe
r

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 canno
t

be completely ruled out.  For example, unlike in physical infrastructures, logical routing or distributed firewall MUST NOT b
e

benchmarked independent of logical switching. System Under Test definition MUST include all components involved with that particula
r

test.

```
+---------------------------------------------------+
| System Under Test                                 |
| +-----------------------------------------------+ |
| | Hyper-Visor                                   | |
| |                                               | |
| |                +-------------+                | |
| |                |     NVP     |                | |
| | +-----+        |    Switch/  |        +-----+ | |
| | | VM1 |<------>|   Router/   |<------>| VM2 | | |
| | +-----+   VW   |  Fire Wall/ |   VW   +-----+ | |
| |                |     etc.,   |                | |
| |                +-------------+                | |
| | Legend                                        | |
| | VM: Virtual Machine                           | |
| | VW: Virtual Wire                              | |
| +-----------------------_-----------------------+ |
+---------------------------------------------------+
```

Figure 2  Intra-Host System Under Test

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 the logical switch component but also any other devices that are part of the physical data center fabric that connects the two hypervisors. System Under Test MUST be well defined to help with repeatability of tests.  System Under Test definition in the case o
f

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

```
     +---------------------------------------------------+
     | System Under Test                                 |
     | +-----------------------------------------------+ |
     | | Hyper-Visor                                   | |
     | |                +-------------+                | |
     | |                |     NVP     |                | |
     | | +-----+        |   Switch/   |        +-----+ | |
     | | | VM1 |<------>|   Router/   |<------>| VM2 | | |
     | | +-----+   VW   |  Fire Wall/ |   VW   +-----+ | |
     | |                |     etc.,   |                | |
     | |                +-------------+                | |
     | +------------------------_----------------------+ |
     |                          ^                        |
     |                          | Network Cabling        |
     |                          v                        |
     | +-----------------------------------------------+ |
     | |       Physical Networking Components          | |
     | |     switches, routers, firewalls etc.,        | |
     | +-----------------------------------------------+ |
     |                          ^                        |
     |                          | Network Cabling        |
     |                          v                        |
     | +-----------------------------------------------+ |
     | | Hyper-Visor                                   | |
     | |                +-------------+                | |
     | |                |     NVP     |                | |
     | | +-----+        |   Switch/   |        +-----+ | |
     | | | VM1 |<------>|   Router/   |<------>| VM2 | | |
     | | +-----+   VW   |  Fire Wall/ |   VW   +-----+ | |
     | |                |     etc.,   |                | |
     | |                +-------------+                | |
     | +------------------------_----------------------+ |
     +---------------------------------------------------+
     Legend
     VM: Virtual Machine
     VW: Virtual Wire
```

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
e
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
n

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 offered.  Hence, all physical components of the physical server running the hypervisor that hosts the virtual components MUST be documented along 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 som
e
of those changes are designed to improve performance.  Please refer to Appendix A for a partial list of parameters to document.

## 5.1. Server Architecture Considerations

When testing physical networking components, the approach taken is to consider the device as a black-box.  With virtual infrastructure
,
this approach would no longer help as the virtual networking components are an intrinsic part of the hypervisor they are running on and are directly impacted by the server architecture used. Server hardware components define the capabilities of the virtual networking components.  Hence, server architecture MUST be documented in detail to help with repeatability of tests.  And the entire hardware and software components become the SUT.

### 5.1.1. 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
s
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 wit
h
any other physical NIC related configuration.

Virtual network components work closer to the application layer the
n
the physical networking components.  Hence 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.

### 5.1.2. 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 independent of the Logical Switch.

### 5.1.3. 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.  Parameters such as MAC addresses used etc.,

When testing NPV 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. Varianc
e

between the tests SHOULD be documented.

### 5.1.4. Tunnel encap/decap outside the hypervisor

Logical network components may also have performance impact based o
n

the functionality available within the physical fabric.  Physical fabric that supports NVO encap/decap is one such case that has considerable impact on the performance.  Any such functionality tha
t

exists on the physical fabric MUST be part of the test result documentation to ensure repeatability of tests. In this case SUT MUST include the physical fabric

### 5.1.5. SUT Hypervisor Profile

Physical networking equipment has well defined physical resource characteristics such as type and number of ASICs/SoCs used, amount of 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.  Example, CPU usage when running logical router.

CPU usage changes based on the type of hardware available within th
e

physical server.  For example, TCP Segmentation Offload greatly reduces CPU usage by offloading the segmentation process to the NIC card on the sender side.  Receive side scaling offers similar benefit on the receive side.  Hence, availability and status of suc
h

hardware 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

* CPU - type, speed, available instruction sets (e.g. AES-NI)

* Memory - type, amount

* Storage - type, amount

* NIC Cards - type, number of ports, offloads available/used, drivers, firmware (if applicable), HW revision

* Libraries such as DPDK if available and used

* Number and type of VMs used for testing and

    o vCPUs

    o RAM

    o Storage

    o Network Driver

    o Any prioritization of VM resources

    o Operating System type, version and kernel if applicable

    o TCP Configuration Changes - if any

    o MTU

* Test tool

    o Workload type

    o Protocol being tested

    o Number of threads

    o Version of tool

* For inter-hypervisor tests,

    o Physical network devices that are part of the test

        ! Note: For inter-hypervisor tests, system unde
r test
        is no longer only the virtual component that i
s being
        tested but the entire fabric that connects the

virtual components become part of the system u
nder
test.

## 6. Security Considerations

Benchmarking activities as described in this memo are limited to technology characterization of a Device Under Test/System Under Tes
t
(DUT/SUT) using controlled stimuli in a laboratory environment, wit
h
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.

## 7. IANA Considerations

No IANA Action is requested at this time.

## 8. 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 advantag
e
of the TCP optimizations or needed to accurately benchmark performance of the Network Virtualization Platforms

## 9. References

### 9.1. 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/

[nv03] IETF, WG, Network Virtualization Overlays, <
https://datatracker.ietf.org/wg/nvo3/documents/>

## 9.2. Informative References

[1]     A. Morton " Considerations for Benchmarking Virtual Network
        Functions and Their Infrastructure", draft-ietf-bmwg-virtua
l-

        net-03, < https://datatracker.ietf.org/doc/draft-ietf-bmwg-
        virtual-net/?include_text=1>

## 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

### 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)

## Author's 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

## Copyright Notice

Table of Contents

## 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 attrubute.

- 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

                        +---+-+-+-+----+-+-+-+-+
                        | S |L|L| |    |L| |L| |
                        | E |2|2|V| E  |3|M|U|6|
                        | R |V|C|P| V  |V|V|B|P|
                        | V |P|K|L| P  |P|P|G|E|
                        | I |N|T|S| N  |N|N|P| |
                        | C +-+-+-+-+-+-+-+-+-+
                        | E | L2SVC |  L3SVC  |
                        +---+-------+---------+
                        |   |      BGP        |
                        | I +-----------------+
                        | N |    Transport    |
                        | F +-----------------+
                        | R |      IGP        |
                        | A +-----------------+
                        |   |    Bridging     |
                        +---+---------+-------+
                        | T |         | Active|
                        | O | Logical |Standby|
                        | P +---------+ M-Home|
                        | O | Physical| L/B   |
                        +---+-+-+-+-+-+-+-+-+-+
                        | N |I| | | | |C| | |I|
                        | O |N|F|Q| | |G|N|N|S|
                        | D |T|A|O|F|G|S|S|S|S|
                        | E |F|B|S|W|R|S|R|F|U|
                        +---+-+-+-+-+-+-+-+-+-+
```

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

- QOS - Quality of Services. Traffic queuing, buffering, and congestion management technologies are used in this level

- 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.

- GR - Graceful Restart per protocol. It needs to cooperate with adjacent node

- 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.

- 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.

- NSF - Non-Stop Forwarding for layer 2 traffic, including layer 2 protocols such as spanning tree state

- 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:

- 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 funtionalities 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 cients 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
  - Labled 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 genrator.

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 avrage 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, maxiumum 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 particular 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 arpnp-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

- bgp-hold-time-distrib, list of top hold-time values and their respective percentage out of all peers.
- bgp-as-path-distrib, list of top as-path lengths and their respective percentage of all BGP paths
- bgp-community-distrib, list of top community size and their respective percentage out of all BGP paths
- mpls-sig, list, MPLS signaling protocol, rsvp or ldp
- rsvp-lsp-count-ingress, uint32, total ingress lsp count
- rsvp-lsp-count-transit, uint32, total transit lsp count
- rsvp-lsp-count-egress, uint32, total egress lsp count
- ldp-fec-count, uint32, total forwarding equivalence class
- rsvp-lsp-protection, list, link-node, link, frr
- ospf-interface-type, list, point-to-point, broadcast, non-broadcast multi-access
- 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
- ospf-route-count, list, total OSPF routes in both backbone and non-backbone areas
- isis-lsp-distrib, list, similar to ospf-lsa-distrib
- 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.

- rib-learning-time, uint32 in seconds. This indicates show quickly the route processor learns routing objects either locally and remotely
- 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.
- 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
- 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.

- 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.
- cpu-util, total CPU utilization of the controllers in stead state
- cpu-util-peak, Peak CPU utilization on the controller in event of failure, and convergence
- mem-util, total memory utilization of the controllers in steady state
- mem-util-peak, total memory utilization on the controller in event of failure and convergence
- processes, list of top processes running on the controllers with their CPU and memory utilization.
- lc-cpu-util, top CPU utilization on the line cards
- lc-cpu-util-peak, maximum peak CPU utilization among all line cards in event of failure and convergence
- lc-mem-util, top memory utilization on the line cards
- lc-mem-util-peak, maximum peak memory utilization among all line cards in event of failure and convergence
- 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
- traffic performance, in both pps and bps. It is measured the rate of traffic received by pumping oversubscribed traffic at ingress
- 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
- 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