Internet-Draft
Network Working Group
Intended Status: Informational
Expires: March 2015

Bhuvaneswaran Vengainathan
Anton Basil
Veryx Technologies
Vishwas Manral
Ionos Corp
Mark Tassinari
Hewlett-Packard
September 26, 2014

# Benchmarking Methodology for SDN Controller Performance
## draft-bhuvan-bmwg-of-controller-benchmarking-01

## Abstract

This document defines the metrics and methodologies for measuring performance of SDN controllers. SDN controllers have been implemented with many varying designs, in order to achieve their intended network functionality. Hence, in this document the authors take the approach of considering an SDN controller as a black box, defining the metrics 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

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

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

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

This Internet-Draft will expire on March 26, 2015.

Table of Contents

## 1. Introduction

This document provides generic metrics and methodologies for benchmarking SDN controller performance. An SDN controller may support many northbound and southbound protocols, implement wide range of applications and work as standalone 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 various controller designs 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", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

## 2. Terminology

SDN Node:
An SDN node is a physical or virtual entity that forwards data in a software defined environment.

Flow:
A flow is a traffic stream having same source and destination address. The address could be MAC or IP or combination of both.

Learning Rate:
The rate at which the controller learns the new source addresses from the received traffic without dropping.

Controller Forwarding Table:
A controller forwarding table contains flow records for the flows configured in the data path.

Northbound Interface:
Northbound interface is the application programming interface provided by the SDN controller for communication with SDN services and applications.

Southbound Interface:
Southbound interface is the application programming interface provided by the SDN controller for communication with the SDN nodes.

Proactive Flow Provisioning:
Proactive flow provisioning is the pre-provisioning of flow entries into the controller's forwarding table through controller's northbound interface or management interface.

Reactive Flow Provisioning:
Reactive flow provisioning is the dynamic provisioning of flow entries into the controller's forwarding table based on traffic forwarded by the SDN nodes through controller's southbound interface.

Path:
A path is the route taken by a flow while traversing from a source node to destination node.

Standalone Mode:
Single controller handling all control plane functionalities.

Cluster/Redundancy Mode:
Group of controllers handling all control plane functionalities .

Synchronous Message:
Any message from the SDN node that triggers a response message from the controller e.g., Keepalive request and response message, flow setup request and response message etc.,

## 3. Scope

This document defines a number of tests to measure the networking aspects of SDN controllers. These tests are recommended for execution in lab environments rather than in real time deployments.

## 4. Test Setup

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

### 4.1 SDN Network - Controller working in Standalone Mode

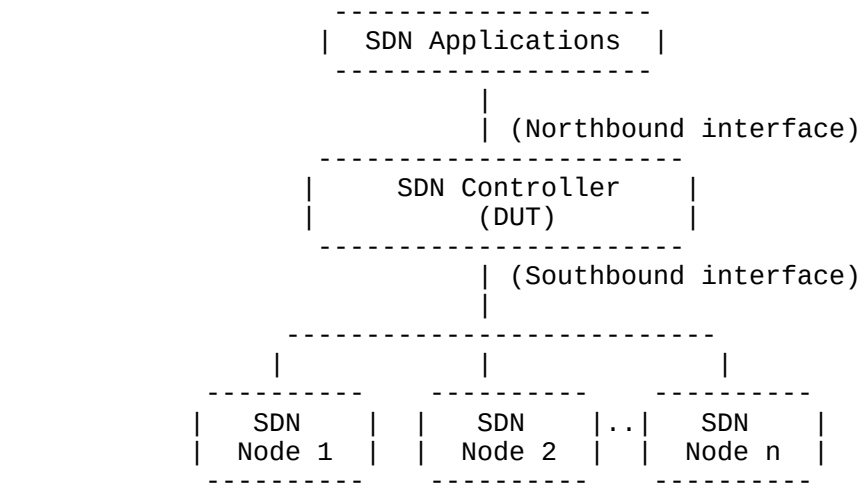



Figure 1

### 4.2 SDN Network - Controller working in Cluster Mode

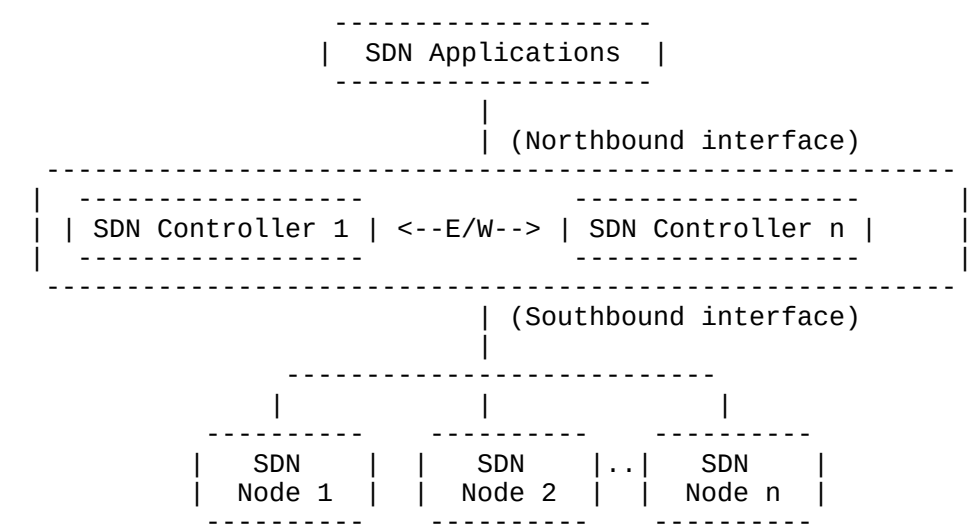



Figure 2

4.3 SDN Network with Traffic Endpoints (TE) - Controller working in Standalone Mode

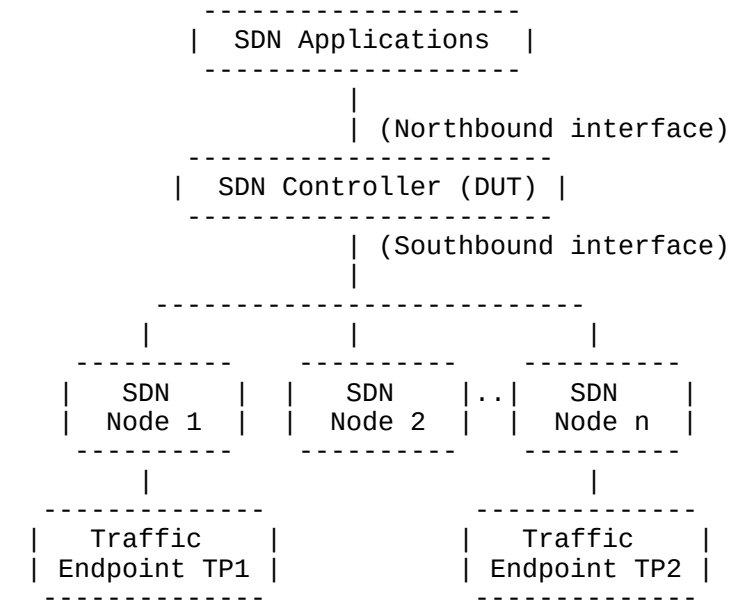
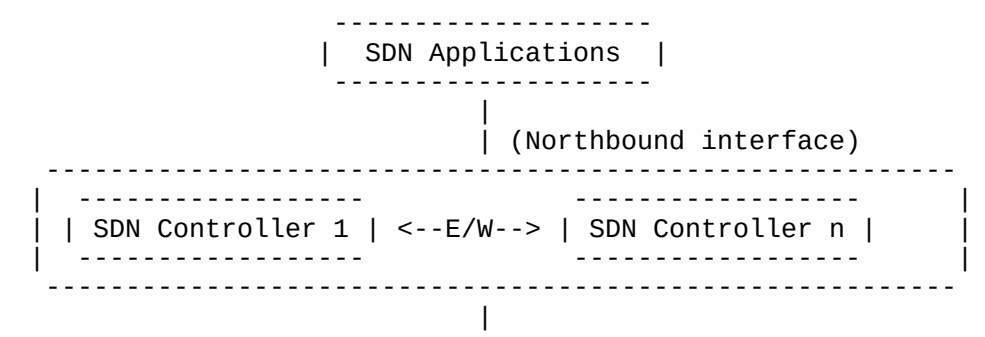



Figure 3

4.4 SDN Network with Traffic Endpoints (TE) - Controller working in Cluster Mode



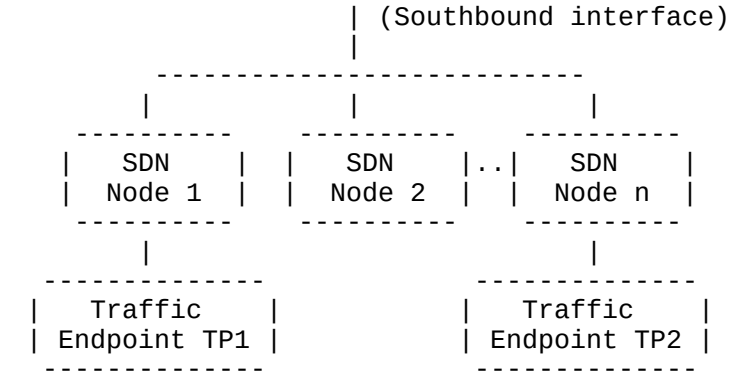



Figure 4

4.5 SDN Node with Traffic Endpoints (TE) - Controller working in Standalone Mode

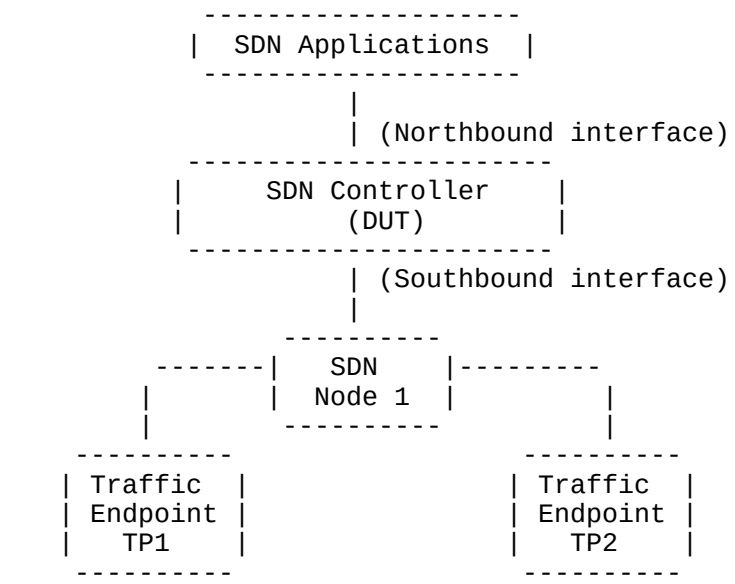



Figure 5

4.6 SDN Node with Traffic Endpoints (TE) - Controller working in Cluster Mode

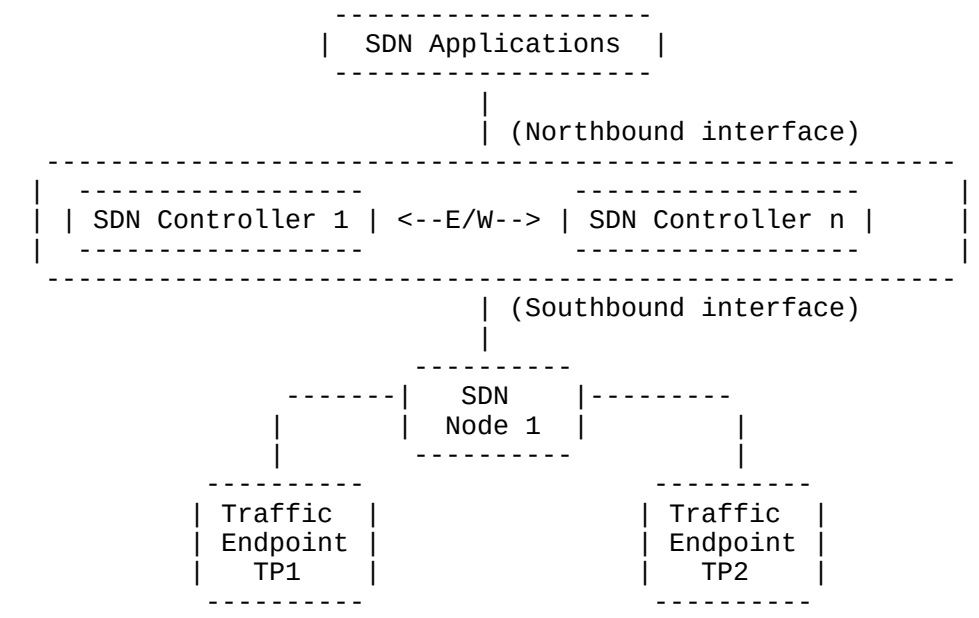



Figure 6

5. Test Considerations

5.1 Network Topology

The network SHOULD be deployed with SDN nodes interconnected in either fully meshed, tree or linear topology. Care should be taken to make sure that the loop prevention mechanism is enabled either in the SDN controller or in the network. To get complete performance characterization of SDN controller, it is recommended that the controller be benchmarked for many network topologies. These network topologies can be deployed using real hardware or emulated in hardware platforms.

5.2 Test Traffic

Test traffic can be used to notify the controller about the arrival of new flows or generate notifications/events towards controller. In either case, it is recommended that at least five different frame sizes and traffic types be used, depending on the intended network deployment.

### 5.3 Connection Setup

There may be controller implementations that support unencrypted and encrypted network connections with SDN nodes. Further, the controller may have backward compatibility with SDN nodes running older versions of southbound protocols. It is recommended that the controller performance be measured with the applicable connection setup methods.

1. Unencrypted connection with SDN nodes, running same protocol version.
2. Unencrypted connection with SDN nodes, running different (previous) protocol versions.
3. Encrypted connection with SDN nodes,running same protocol version
4. Encrypted connection with SDN nodes, running different (previous)protocol versions.

### 5.4 Measurement Accuracy

The measurement accuracy depends on the point of observation where the indications are captured. For example, the notification can be observed at the ingress or egress point of the SDN node. If it is observed at the egress point of the SDN node, the measurement includes the latency within the SDN node also. It is recommended to make observation at the ingress point of the SDN node unless it is explicitly mentioned otherwise in the individual test.

### 5.5 Real World Scenario

Benchmarking tests discussed in the document are to be performed on a "black-box" basis, relying solely on measurements observable external to the controller. The network deployed and the test parameters should be identical to the deployment scenario to obtain value added measures.

## 6. Test Reporting

Each test has a reporting format which is specific to individual test. In addition, the following configuration parameters SHOULD be reflected in the test report.

1. Controller name and version
2. Northbound protocols and version
3. Southbound protocols and version
4. Controller redundancy mode (Standalone or Cluster Mode)
5. Connection setup (Unencrypted or Encrypted)
6. Network Topology (Mesh or Tree or Linear)
7. SDN Node Type (Physical or Virtual or Emulated)
8. Number of Nodes
9. Number of Links
10. Test Traffic Type

# 7. Benchmarking Tests

## 7.1 Performance

### 7.1.1 Network Topology Discovery Time

Objective:
To measure the time taken to discover the network topology- nodes and its connectivity by a controller, expressed in milliseconds.

Setup Parameters:
The following parameters MUST be defined:

Network setup parameters:
Number of nodes (N) - Defines the number of nodes present in the defined network topology

Test setup parameters:
Test Iterations (Tr) - Defines the number of times the test needs to be repeated. The recommended value is 3.
Test Interval (To)- Defines the maximum time for the test to complete, expressed in milliseconds.

Test Setup:
The test can use one of the test setup described in section 4.1 and 4.2 of this document.

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

Procedure:
1. Initialize the controller - network applications, northbound and southbound interfaces.
2. Deploy the network with the given number of nodes using mesh or linear topology.
3. Initialize the network connections between controller and network nodes.
4. Record the time for the first discovery message exchange between the controller and the network node (Tm1).
5. Query the controller continuously for the discovered network topology information and compare it with the deployed network topology information.
6. Stop the test when the discovered topology information is matching with the deployed network topology or the expiry of test interval (To).

7. Record the time last discovery message exchange between the controller and the network node (Tmn) when the test completed successfully.

Note: While recording the Tmn value, it is recommended that the messages that are used for aliveness check or session management be ignored.

Measurement:
Topology Discovery Time Tr1 = Tmn-Tm1.

$$\text{Average Topology Discovery Time} = \frac{Tr1 + Tr2 + Tr3 \ldots Trn}{\text{Total Test Iterations}}$$

Note:
1. To increase the certainty of measured result, it is recommended that this test be performed several times with same number of nodes using same topology.
2. To get the full characterization of a controller's topology discovery functionality
   a. Perform the test with varying number of nodes using same topology
   b. Perform the test with same number of nodes using different topologies.

Reporting Format:
The Topology Discovery Time results SHOULD be reported in the format of a table, with a row for each iteration. The last row of the table 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.

If this test is repeated with same number of nodes over different topologies,the results SHOULD be reported in the form of a graph. The X coordinate SHOULD be the Topology Type, the Y coordinate SHOULD be the average Topology Discovery Time.

### 7.1.2 Synchronous Message Processing Time

Objective:
To measure the time taken by the controller to process a synchronous message, expressed in milliseconds.

Setup Parameters:
The following parameters MUST be defined:

Network setup parameters:
Number of nodes (N) - Defines the number of nodes present in the defined network topology

Test setup parameters:
Test Iterations (Tr) - Defines the number of times the test needs to be repeated. The recommended value is 3.
Test Duration (Td) - Defines the duration of test iteration, expressed in seconds. The recommended value is 5 seconds.

Test Setup:
The test can use one of the test setup described in section 4.1 and 4.2 of this document.

Prerequisite:
1. The controller should have completed the network topology discovery for the connected nodes.

Procedure:
1. Generate a synchronous message from every connected nodes one at a time and wait for the response before generating the next message.
2. Record total number of messages sent to the controller by all nodes (Ntx) and the responses received from the controller (Nrx) within the test duration (Td).

Measurement:

$$\text{Synchronous Message Processing Time Tr1} = \frac{Td}{Nrx}$$

$$\text{Average Synchronous Message Processing Time} = \frac{Tr1 + Tr2 + Tr3..Trn}{\text{Total Test Iterations}}$$

Note:

1. The above test measures the controller's message processing time at lower traffic rate. To measure the controller's message processing time at full connection rate, apply the same measurement equation with the Td and Nrx values obtained from Synchronous Message Processing Rate test (defined in Section 7.1.3).
2. To increase the certainty of measured result, it is recommended that this test be performed several times with same number of nodes using same topology.
3. To get the full characterization of a controller's synchronous message processing time
   a. Perform the test with varying number of nodes using same topology
   b. Perform the test with same number of nodes using different topologies.

Reporting Format:

The Synchronous Message Processing Time results SHOULD be reported in the format of a table with a row for each iteration. The last row of the table indicates the average Synchronous Message Processing Time.

The report should capture the following information in addition to the configuration parameters captured in section 6.
- Offered rate (Ntx)

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 Synchronous Message Processing Time.

If this test is repeated with same number of nodes using different topologies, the results SHOULD be reported in the form of a graph. The X coordinate SHOULD be the Topology Type, the Y coordinate SHOULD be the average Synchronous Message Processing Time.

### 7.1.3 Synchronous Message Processing Rate

Objective:

To measure the maximum number of synchronous messages (session aliveness check message, new flow arrival notification message etc.) a controller can process within the test duration, expressed in messages processed per second.

Setup Parameters:
The following parameters MUST be defined:

Network setup parameters:
Number of nodes (N) - Defines the number of nodes present in the defined network topology.

Test setup parameters:
Test Iterations (Tr) - Defines the number of times the test needs to be repeated. The recommended value is 3.
Test Duration (Td) - Defines the duration of test iteration, expressed in seconds. The recommended value is 5 seconds.

Test Setup:
The test can use one of the test setup described in section 4.1 and 4.2 of this document.

Prerequisite:
1. The controller should have completed the network topology discovery for the connected nodes.

Procedure:
1. Generate synchronous messages from all the connected nodes at the full connection capacity for the Test Duration (Td).
2. Record total number of messages sent to the controller by all nodes (Ntx) and the responses received from the controller (Nrx) within the test duration (Td).

Measurement:

$$\text{Synchronous Message Processing Rate Tr1} = \frac{Nrx}{Td}$$

$$\text{Average Synchronous Message Processing Rate} = \frac{Tr1 + Tr2 + Tr3..Trn}{\text{Total Test Iterations}}$$

Note:
1. To increase the certainty of measured result, it is recommended that this test be performed several times with same number of nodes using same topology.
2. To get the full characterization of a controller’s synchronous message processing rate
   a. Perform the test with varying number of nodes using same topology.
   b. Perform the test with same number of nodes using different topologies.

Reporting Format:
The Synchronous Message Processing Rate results SHOULD be reported in the format of a table with a row for each iteration. The last row of the table indicates the average Synchronous Message Processing Rate.

The report should capture the following information in addition to the configuration parameters captured in section 6.
- Offered rate (Ntx)

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 coordinate SHOULD be the Number of nodes (N), the Y coordinate SHOULD be the average Synchronous Message Processing Rate.

If this test is repeated with same number of nodes over different topologies,the results SHOULD be reported in the form of a graph. The X coordinate SHOULD be the Topology Type, the Y coordinate SHOULD be the average Synchronous Message Processing Rate.

### 7.1.4 Path Provisioning Time

Objective:
To measure the time taken by the controller to setup a path between source and destination node, expressed in milliseconds.

Setup Parameters:
The following parameters MUST be defined:

Network setup parameters:
Number of nodes (N) - Defines the number of nodes present in the defined network topology
Number of data path nodes (Ndp) - Defines the number of nodes present in the path between source and destination node.

Test setup parameters:
Test Iterations (Tr) - Defines the number of times the test needs to be repeated. The recommended value is 3.
Test Interval (To) - Defines the maximum time for the test to complete, expressed in milliseconds.

Test Setup:
The test can use one of the test setups described in section 4.3 and 4.4 of this document.

Prerequisite:

1. The controller should contain the network topology information for the deployed network topology.
2. The network topology information can be learnt through dynamic Topology Discovery Mechanism or static configuration.
3. The controller should have learnt about the location of source/destination endpoint for which the path has to be provisioned. This can be achieved through dynamic learning or static provisioning.
4. The SDN Node should send all new flows to the controller when it receives.

Procedure:

Reactive Path Provisioning:

1. Send traffic with source as source endpoint address and destination as destination endpoint address from TP1.
2. Record the time for the first frame sent to the source SDN node (Tsf1).
3. Wait for the arrival of first frame from the destination node or the expiry of test interval (To).
4. Record the time when the first frame received from the destination SDN node (Tdf1).

Proactive Path Provisioning:

1. Send traffic with source as source endpoint address and destination as destination endpoint address from TP1.
2. Install the flow with the learnt source and destination address through controller’s northbound or management interface.
3. Record the time when a successful response for the flow installation is received (Tp) from the controller.
4. Wait for the arrival of first frame from the destination node or the expiry of test interval (To).
5. Record the time when the first frame received from the destination node (Tdf1).

Measurement:

Reactive Path Provisioning:

Flow Provisioning Time Tr1 = Tdf1-Tsf1.

Proactive Path Provisioning:

Path Provisioning Time Tr1 = Tdf1-Tp.

$$\text{Average Path Provisioning Time} = \frac{Tr1 + Tr2 + Tr3 \,..\, Trn}{\text{Total Test Iterations}}$$

Note:
1. To increase the certainty of measured result,it is recommended that this test be performed several times with same number of nodes using same topology.
2. To get the full characterization of a controller's path provisioning time
   a. Perform the test with varying number of nodes using same topology
   b. Perform the test with same number of nodes using different topologies.

Reporting Format:
The Path Provisioning Time results SHOULD be reported in the format of a table with a row for each iteration. The last row of the table indicates the average Path Provisioning Time.

The report should capture the following information in addition to the configuration parameters captured in section 6.
- Number of data path nodes

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 Path Provisioning Time.

If this test is repeated with same number of nodes using different topologies, the results SHOULD be reported in the form of a graph. The X coordinate SHOULD be the Topology Type, the Y coordinate SHOULD be the average Path Provisioning Time.

### 7.1.5 Path Provisioning Rate

Objective:
To measure the maximum number of paths a controller can setup between sources and destination node within the test duration, expressed in paths per second.

Setup Parameters:
The following parameters MUST be defined:

Network setup parameters:
Number of nodes (N) - Defines the number of nodes present in the defined network topology.

Test setup parameters:
Test Iterations (Tr) - Defines the number of times the test needs to be repeated. The recommended value is 3.
Test Duration (Td)- Defines the duration of test iteration, expressed in seconds. The recommended value is 5 seconds.

Test Setup:
The test can use one of the test setup described in section 4.3 and 4.4 of this document.

Prerequisite:

1. The controller should contain the network topology information for the deployed network topology.
2. The network topology information can be learnt through dynamic Topology Discovery Mechanism or static configuration.
3. The controller should have learnt about the location of source/destination endpoints for which the paths have to be provisioned. This can be achieved through dynamic learning or static provisioning.
4. The SDN Node should send all new flows to the controller when it receives.

Procedure:
Reactive Path Provisioning:

1. Send traffic at the individual node's synchronous message processing rate with unique source and/or destination addresses from test port TP1.
2. Record total number of unique frames received by the destination node (Ndf) within the test duration (Td).

Proactive Path Provisioning:

1. Send traffic continuously with unique source and destination addresses from the source node.
2. Install flows with the learnt source and destination addresses through controller's northbound or management interface.
3. Record total number of unique frames received from the destination node (Ndf) within the test duration (Td).

Measurement:
Proactive/Reactive Path Provisioning:

$$\text{Path Provisioning Rate Tr1} = \frac{\text{Ndf}}{\text{Td}}$$

$$\text{Average Path Provisioning Rate} = \frac{\text{Tr1} + \text{Tr2} + \text{Tr3} \ldots \text{Trn}}{\text{Total Test Iterations}}$$

Note:
1. To increase the certainty of measured result,it is recommended that this test be performed several times with same number of nodes using same topology.
2. To get the full characterization of a controller's path provisioning rate
   a. Perform the test with varying number of nodes using same topology
   b. Perform the test with same number of nodes using different topologies.

Reporting Format:
The Path Provisioning Rate results SHOULD be reported in the format of a table with a row for each iteration. The last row of the table indicates the average Path Provisioning Rate.

The report should capture the following information in addition to the configuration parameters captured in section 6.
- Number of Nodes in the path
- Provisioning Type (Proactive/Reactive)
- Offered rate

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 Path Provisioning Rate.

If this test is repeated with same number of nodes using different topologies, the results SHOULD be reported in the form of a graph. The X coordinate SHOULD be the Topology Type, the Y coordinate SHOULD be the average Path Provisioning Rate.

### 7.1.6 Network Topology Change Detection Time

Objective:
To measure the time taken by the controller to detect any changes in the network topology, expressed in milliseconds.

Setup Parameters:
The following parameters MUST be defined:

Network setup parameters:
Number of nodes (N) - Defines the number of nodes present in the defined network topology

Test setup parameters:
Test Iterations (Tr) - Defines the number of times the test needs to be repeated. The recommended value is 3.
Test Interval (To) - Defines the maximum time for the test to complete,expressed in milliseconds. Test not completed within this time interval is considered as incomplete.

Test Setup:
The test can use one of the test setup described in section 4.1 and 4.2 of this document.

Prerequisite:
1. The controller should have discovered the network topology information for the deployed network topology.
2. The periodic network discovery operation should be configured to twice the Test Interval (To) value.

Procedure:
1. Trigger a topology change event through one of the operation (e.g., Add a new node or bring down an existing node or a link).
2. Record the time when the first topology change notification is sent to the controller (Tcn).
3. Stop the test when the controller sends the first topology re-discovery message to the SDN node or the expiry of test interval (To).
4. Record the time when the first topology re-discovery message is received from the controller (Tcd).

Measurement:
Network Topology Change Detection Time Tr1 = Tcd-Tcn.

$$\text{Average Network Topology Change Detection Time} = \frac{Tr1 + Tr2 + Tr3 \,..\, Trn}{\text{Total Test Iterations}}$$

Note:
1. To increase the certainty of measured result,it is recommended that this test be performed several times with same number of nodes using same topology.

Reporting Format:
The Network Topology Change Detection Time results SHOULD be reported in the format of a table with a row for each iteration. The last row of the table indicates the average Network Topology Change Time.

## 7.2 Scalability

### 7.2.1 Network Discovery Size

Objective:
To measure the network size (number of nodes) that a controller can discover within a stipulated time.

Setup Parameters:
The following parameters MUST be defined:

Network setup parameters:
Number of nodes (N) - Defines the initial number of nodes present in the defined network topology

Test setup parameters:
Network Discovery Time (Tnd) - Defines the stipulated time acceptable by the user, expressed in seconds.

Test Setup:
The test can use one of the test setup described in section 4.1 and 4.2 of this document.

Prerequisite:
1. The controller should support automatic network discovery.
2. Tester should be able to retrieve the discovered topology information either through controller's management interface or northbound interface.
3. Controller should be operational.
4. Network with the given number of nodes and intended topology (Mesh or Linear or Tree) should be deployed.

Procedure:
1. Initialize 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 after the expiry of Network Discovery Time (Tnd).
3. Increase the number of nodes by 1 when the comparison is successful and repeat the test.
4. Decrease the number of nodes by 1 when the comparison fails and repeat the test.
5. Continue the test until the comparison of step 4 is successful.
6. Record the number of nodes for the last iteration (Ns) where the topology comparison was successful.

Measurement:

Network Discovery Size = Ns.

Note:
This test may be performed with different topologies to obtain the controller's scalability factor for various network topologies.

Reporting Format:
The Network Discovery Size results SHOULD be reported in addition to the configuration parameters captured in section 6.

### 7.2.2 Flow Scalable Limit

Objective:
To measure the maximum number of flow entries a controller can manage in its Forwarding table.

Setup Parameters:
The following parameters MUST be defined:

Test Setup:
The test can use one of the test setups described in section 4.5 and 4.6 of this document.

Prerequisite:
1. The controller Forwarding table should be empty.
2. Flow Idle time should be set to higher or infinite value.
3. The controller should have 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 Path Provisioning:
1. Send bi-directional traffic continuously with unique source and/or destination addresses from test ports TP1 and TP2 at the learning rate of controller.
2. Query the controller at a regular interval (e.g., 5 seconds) for the number of flow entries from its northbound interface.
3. Stop the test when the retrieved value is constant for three consecutive iterations and record the value received from the last query (Nrp).

Proactive Path Provisioning:
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 path provisioning 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 Path Provisioning:

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

Reactive Path Provisioning:

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

Flow Scalable Limit = Max Flow Entries.

Reporting Format:
The Flow Scalable Limit results SHOULD be tabulated with the following information in addition to the configuration parameters captured in section 6.
- Provisioning Type (Proactive/Reactive)

### 7.3 Security

#### 7.3.1 Exception Handling

Objective:
To determine the effect of handling error packets and notifications on performance tests. The impact SHOULD be measured for the following performance tests
a. Path Programming Rate
b. Path Programming Time
c. Network Topology Change Detection Time

Prerequisite:
This test should be performed after obtaining the baseline measurement results for the above performance tests.

Procedure:

1. Perform the above listed performance tests and send 1% of messages from the Synchronous Message Processing Rate as invalid messages from the connected nodes.
2. Perform the above listed performance tests and send 2% of messages from the Synchronous Message Processing Rate as invalid messages from the connected nodes.

Note:

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

Measurement:

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

Reporting Format:

The Exception Handling results SHOULD 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

### 7.3.2 Denial of Service Handling

Objective:

To determine the effect of handling DoS attacks on performance and scalability tests The impact SHOULD be measured for the following tests

a. Path Programming Rate
b. Path Programming Time
c. Network Topology Change Detection Time
d. Network Discovery Size

Prerequisite:

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

Procedure:

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

Note:
DoS attacks can be launched on one of the following interfaces.
a. Northbound (e.g., Sending a huge number of requests on northbound interface)
b. Management (e.g., Ping requests to controller's management interface)
c. Southbound (e.g., TCP SYNC messages on southbound interface)

Measurement:
Measurement should be done as per the equation defined in the corresponding test's measurement section.

Reporting Format:
The DoS Attacks Handling results SHOULD 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.

## 7.4 Reliability

### 7.4.1 Controller Failover Time

Objective:
To compute the time taken to switch from one controller to another when the controllers are teamed and the active controller fails.

Setup Parameters:
The following parameters MUST be defined:

Controller setup parameters:
Number of cluster nodes (CN) - Defines the number of member nodes present in the cluster.
Redundancy Mode (RM) - Defines the controller clustering mode e.g., Active - Standby or Active - Active.

Test Setup:
The test can use the test setup described in section 4.4 of this document.

Prerequisite:

1. Master controller election should be completed.
2. Nodes are connected to the controller cluster as per the Redundancy Mode (RM).
3. The controller cluster should have completed the network topology discovery.
4. The SDN Node should send all new flows to the controller when it receives.

Procedure:

1. Send bi-directional traffic continuously with unique source and/or destination addresses from test ports TP1 and TP2 at the rate that the controller processes without any drops.
2. Bring down the active controller.
3. Stop the test when a first frame received on TP2 after failover operation.
4. Record the test duration (Td), total number of frames sent (Nsnt) on TP1 and number of frames received (Nrvd) on TP2.

Measurement:

Controller Failover Time = ((Td/Nrvd) - (Td/Nsnt))
Packet Loss = Nsnt - Nrvd

Reporting Format:

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

- Number of cluster nodes
- Redundancy mode
- Controller Failover
- Time Packet Loss

### 7.4.2 Network Re-Provisioning Time

Objective:

To compute the time taken to re-route the traffic by the controller when there is a failure in existing traffic paths.

Setup Parameters:

Same setup parameters as defined in the Path Programming Rate performance test (Section 7.1.5).

Prerequisite:

Network with the given number of nodes and intended topology (Mesh or Tree) with redundant paths should be deployed.

Procedure:

1. Perform the test procedure mentioned in Path Programming Rate test (Section 7.1.5).
2. Send bi-directional traffic continuously with unique sequence number for one particular traffic endpoint.
3. Bring down a link or switch in the traffic path.
4. Stop the test after receiving first frame after network re-convergence (timeline).
5. 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).
6. 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 SHOULD 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

## 8. Test Coverage

| | Performance | Scalability | Reliablity |
|---|---|---|---|
| Setup | 1. Network Topology Discovery<br>2. Path Provisioning Time<br>3. Path Provisioning Rate | 1. Network Discovery Size | |
| Operational | 1. Synchronous Message Processing Rate<br>2. Synchronous Message Processing Time | 1. Flow Scalable Limit | 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 |

## 9. References

### 9.1 Normative References

[RFC6241] R. Enns, M. Bjorklund, J. Schoenwaelder, A. Bierman, "Network Configuration Protocol (NETCONF)",RFC 6241, June 2011.

[RFC6020] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010

[RFC5440] JP. Vasseur, JL. Le Roux, "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, March 2009.

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

[I-D.i2rs-architecture] A. Atlas, J. Halpern, S. Hares, D. Ward, T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-05 (Work in progress), July 20,2014.

## 9.2 Informative References

[OpenContrail] Ankur Singla, Bruno Rijsman, "OpenContrail Architecture Documentation", http://opencontrail.org/opencontrail-architecture-documentation

[OpenDaylight] OpenDaylight Controller:Architectural Framework, https://wiki.opendaylight.org/view/OpenDaylight_Controller

## 10. IANA Considerations

This document does not have any IANA requests.

## 11. Security Considerations

Benchmarking tests described in this document are limited to the performance characterization of controller in lab environment with isolated network and dedicated address space.

## 12. Acknowledgements

The authors would like to acknowledge the following individuals for their help and participation of the compilation of this document: Al Morton (AT&T), Brain Castelli (Spirent), Sandeep Gangadharan(HP), Sarah Banks (VSS Monitoring) who made significant suggestions to the current and earlier versions of this document.

## 13. 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

Vishwas Manral
Ionos Corp,
4100 Moorpark Ave,
San Jose, CA

Email: vishwas@ionosnetworks.com

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

Email: mark.tassinari@hp.com

Internet Engineering Task Force
Internet-Draft, Intended status: Informational
Expires April 2015
October 17, 2014

L. Avramov
Cisco Systems
J. Rapp
Hewlett-Packard

# Data Center Benchmarking Methodology
draft-bmwg-dcbench-methodology-03

## Abstract

The purpose of this informational document is to establish test and evaluation methodology and measurement techniques for physical network equipment in the data center.

## Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

## Copyright Notice

Table of Contents



## 1. Introduction

Traffic patterns in the data center are not uniform and are constantly changing. They are dictated by the nature and variety of applications utilized in the data center. It can be largely east-west traffic flows in one data center and north-south in another, while some may combine both. Traffic patterns can be bursty in nature and contain many-to-one, many-to-many, or one-to-many flows. Each flow may also be small and latency sensitive or large and throughput sensitive while containing a mix of UDP and TCP traffic. All of which can coexist in a single cluster and flow through a single network device all at the same time. Benchmarking of network devices have long used RFC1242, RFC2432, RFC2544, RFC2889 and RFC3918. These benchmarks have largely been focused around various latency attributes and max throughput of the Device Under Test [DUT] being

benchmarked. These standards are good at measuring theoretical max throughput, forwarding rates and latency under testing conditions however, they do not represent real traffic patterns that may affect these networking devices.

The following provides a methodology for benchmarking Data Center DUT including congestion scenarios, switch buffer analysis, microburst, head of line blocking, while also using a wide mix of traffic conditions.

### 1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [6].

### 1.2. Methodology format and repeatability recommendation

The format used for each section of this document is the following:

-Objective

-Methodology

-Reporting Format

MUST: minimum test for the scenario described

SHOULD: recommended test for the scenario described

MAY: ideal test for the scenario described

For each test methodology described, it is key to obtain repeatability of the results. The recommendation is to perform enough iterations of the given test to make sure the result is accurate, this is especially important for section 3) as the buffering testing has been historically the least reliable.

## 2. Line Rate Testing

### 2.1 Objective

Provide at maximum rate test for the performance values for throughput, latency and jitter. It is meant to provide the tests to run and methodology to verify that a DUT is capable of forwarding packets at line rate under non-congested conditions.

### 2.2 Methodology

A traffic generator SHOULD be connected to all ports on the DUT. Two tests MUST be conducted: a port-pair test [RFC 2544/3918 compliant] and also in a full mesh type of DUT test [RFC 2889/3918 compliant].

For all tests, the percentage of traffic per port capacity sent MUST be 99.98% at most, with no PPM adjustment to ensure stressing the DUT

in worst case conditions. Tests results at a lower rate MAY be provided for better understanding of performance increase in terms of latency and jitter when the rate is lower than 99.98%. The receiving rate of the traffic needs to be captured during this test in % of line rate.

The test MUST provide the latency values for minimum, average and maximum, for the exact same iteration of the test.

The test MUST provide the jitter values for minimum, average and maximum, for the exact same iteration of the test.

Alternatively when a traffic generator CAN NOT be connected to all ports on the DUT, a snake test MUST be used for line rate testing, excluding latency and jitter as those became then irrelevant. The snake test consists in the following method: -connect the first and last port of the DUT to a traffic generator-connect back to back sequentially all the ports in between: port 2 to 3, port 4 to 5 etc to port n-2 to port n-1; where n is the total number of ports of the DUT-configure port 1 and 2 in the same vlan X, port 3 and 4 in the same vlan Y, etc. port n-1 and port n in the same vlan ZZZ. This snake test provides a capability to test line rate for Layer 2 and Layer 3 RFC 2544/3918 in instance where a traffic generator with only two ports is available. The latency and jitter are not to be considered with this test.

## 2.3 Reporting Format

The report MUST include:

-physical layer calibration information as defined into (Placeholder for definitions draft)

-number of ports used

-reading for throughput received in percentage of bandwidth, while sending 99.98% of port capacity on each port, across packet size from 64 byte all the way to 9216. As guidance, an increment of 64 byte packet size between each iteration being ideal, a 256 byte and 512 bytes being also often time used, the most common packets sizes order for the report is: 64b,128b,256b,512b,1024b,1518b,4096,8000,9216b.

The pattern for testing can be expressed using RFC 6985 [IMIX Genome: Specification of Variable Packet Sizes for Additional Testing]

-throughput needs to be expressed in % of total transmitted frames

-for packet drops, they MUST be expressed in packet count value and SHOULD be expressed in % of line rate

-for latency and jitter, values expressed in unit of time [usually microsecond or nanosecond] reading across packet size from 64 bytes to 9216 bytes

-for latency and jitter, provide minimum, average and maximum values. if different iterations are done to gather the minimum, average and maximum, it SHOULD be specified in the report along with a justification on why the information could not have been gathered at the same test iteration

-for jitter, a histogram describing the population of packets measured per latency or latency buckets is RECOMMENDED

-The tests for throughput, latency and jitter MAY be conducted as individual independent events, with proper documentation in the report but SHOULD be conducted at the same time.

## 3. Buffering Testing

### 3.1 Objective

To measure the size of the buffer of a DUT under typical|many|multiple conditions. Buffer architectures between multiple DUTs can differ and include egress buffering, shared egress buffering switch-on-chip [SoC], ingress buffering or a combination. The test methodology covers the buffer measurement regardless of buffer architecture used in the DUT.

### 3.2 Methodology

A traffic generator MUST be connected to all ports on the DUT.

The methodology for measuring buffering for a data-center switch is based on using known congestion of known fixed packet size along with maximum latency value measurements. The maximum latency will increase until the first packet drop occurs. At this point, the maximum latency value will remain constant. This is the point of inflexion of this maximum latency change to a constant value. There MUST be multiple ingress ports receiving known amount of frames at a known fixed size, destined for the same egress port in order to create a known congestion event. The total amount of packets sent from the oversubscribed port minus one, multiplied by the packet size

represents the maximum port buffer size at the measured inflexion point.

1) Measure the highest buffer efficiency

First iteration: ingress port 1 sending line rate to egress port 2, while port 3 sending a known low amount of over subscription traffic (1% recommended) with a packet size of 64 bytes to egress port 2. Measure the buffer size value of the number of frames sent from the port sending the oversubscribed traffic up to the inflexion point multiplied by the frame size.

Second iteration: ingress port 1 sending line rate to egress port 2, while port 3 sending a known low amount of over subscription traffic (1% recommended) with same packet size 65 bytes to egress port 2. Measure the buffer size value of the number of frames sent from the port sending the oversubscribed traffic up to the inflexion point multiplied by the frame size.

Last iteration: ingress port 1 sending line rate to egress port 2, while port 3 sending a known low amount of over subscription traffic (1% recommended) with same packet size B bytes to egress port 2. Measure the buffer size value of the number of frames sent from the port sending the oversubscribed traffic up to the inflexion point multiplied by the frame size..

When the B value is found to provide the highest buffer size, this is the highest buffer efficiency

2) Measure maximum port buffer size

At fixed packet size B determined in 3.2.1, for a fixed default COS value of 0 and for unicast traffic proceed with the following:

First iteration: ingress port 1 sending line rate to egress port 2, while port 3 sending a known low amount of over subscription traffic (1% recommended) with same packet size to the egress port 2. Measure the buffer size value by multiplying the number of extra frames sent by the frame size.

Second iteration: ingress port 2 sending line rate to egress port 3, while port 4 sending a known low amount of over subscription traffic (1% recommended) with same packet size to the egress port 3. Measure the buffer size value by multiplying the number of extra frames sent by the frame size.

Last iteration: ingress port N-2 sending line rate traffic to egress port N-1, while port N sending a known low amount of over

subscription traffic (1% recommended) with same packet size to the egress port N Measure the buffer size value by multiplying the number of extra frames sent by the frame size.

This test series MAY be repeated using all different COS values of traffic and then using Multicast type of traffic, in order to find if there is any COS impact on the buffer size.

3) Measure maximum port pair buffer sizes

First iteration: ingress port 1 sending line rate to egress port 2; ingress port 3 sending line rate to egress port 4 etc. Ingress port N-1 and N will respectively over subscribe at 1% of line rate egress port 2 and port 3. Measure the buffer size value by multiplying the number of extra frames sent by the frame size for each egress port.

Second iteration: ingress port 1 sending line rate to egress port 2; ingress port 3 sending line rate to egress port 4 etc. Ingress port N-1 and N will respectively over subscribe at 1% of line rate egress port 4 and port 5. Measure the buffer size value by multiplying the number of extra frames sent by the frame size for each egress port.

Last iteration: ingress port 1 sending line rate to egress port 2; ingress port 3 sending line rate to egress port 4 etc. Ingress port N-1 and N will respectively over subscribe at 1% of line rate egress port N-3 and port N-2. Measure the buffer size value by multiplying the number of extra frames sent by the frame size for each egress port.

This test series MAY be repeated using all different COS values of traffic and then using Multicast type of traffic.

4) Measure maximum DUT buffer size with many to one ports

First iteration: ingress ports 1,2,... N-1 sending each [(1/[N-1])*99.98]+[1/[N-1]] % of line rate per port to the N egress port.

Second iteration: ingress ports 2,... N sending each [(1/[N-1])*99.98]+[1/[N-1]] % of line rate per port to the 1 egress port.

Last iteration: ingress ports N,1,2...N-2 sending each [(1/[N-1])*99.98]+[1/[N-1]] % of line rate per port to the N-1 egress port.

This test series MAY be repeated using all different COS values of traffic and then using Multicast type of traffic.

Unicast traffic and then Multicast traffic SHOULD be used in order to determine the proportion of buffer for documented selection of tests.

Also the COS value for the packets SHOULD be provided for each test iteration as the buffer allocation size MAY differ per COS value. It is RECOMMENDED that the ingress and egress ports are varied in a random, but documented fashion in multiple tests to measure the buffer size for each port of the DUT.

### 3.3 Reporting format

The report MUST include:

- The packet size used for the most efficient buffer used, along with COS value

- The maximum port buffer size for each port

- The maximum DUT buffer size

- The packet size used in the test

- The amount of over subscription if different than 1%

- The number of ingress and egress ports along with their location on the DUT.

## 4 Microburst Testing

### 4.1 Objective

To find the maximum amount of packet bursts a DUT can sustain under various configurations.

### 4.2 Methodology

A traffic generator MUST be connected to all ports on the DUT. In order to cause congestion, two or more ingress ports MUST bursts packets destined for the same egress port. The simplest of the setups would be two ingress ports and one egress port (2-to-1).

The burst MUST be measure with an intensity of 100%, meaning the burst of packets will be sent with a minimum inter-packet gap. The amount of packet contained in the burst will be variable and increase until there is a non-zero packet loss measured. The aggregate amount of packets from all the senders will be used to calculate the maximum amount of microburst the DUT can sustain.

It is RECOMMENDED that the ingress and egress ports are varied in multiple tests to measure the maximum microburst capacity.

The intensity of a microburst MAY be varied in order to obtain the microburst capacity at various ingress rates.

It is RECOMMENDED that all ports on the DUT will be tested simultaneously and in various configurations in order to understand all the combinations of ingress ports, egress ports and intensities.

An example would be:

First Iteration: N-1 Ingress ports sending to 1 Egress Ports

Second Iterations: N-2 Ingress ports sending to 2 Egress Ports

Last Iterations: 2 Ingress ports sending to N-2 Egress Ports

### 4.3 Reporting Format

The report MUST include:

- The maximum value of packets received per ingress port with the maximum burst size obtained with zero packet loss

- The packet size used in the test

- The number of ingress and egress ports along with their location on the DUT

## 5. Head of Line Blocking

### 5.1 Objective

Head-of-line blocking (HOL blocking) is a performance-limiting phenomenon that occurs when packets are held-up by the first packet ahead waiting to be transmitted to a different output port. This is defined in RFC 2889 section 5.5. Congestion Control. This section expands on RFC 2889 in the context of Data Center Benchmarking

The objective of this test is to understand the DUT behavior under head of line blocking scenario and measure the packet loss.

### 5.2 Methodology

In order to cause congestion, head of line blocking, groups of four ports are used. A group has 2 ingress and 2 egress ports. The first

ingress port MUST have two flows configured each going to a different egress port. The second ingress port will congest the second egress port by sending line rate. The goal is to measure if there is loss for the first egress port which is not not oversubscribed.

A traffic generator MUST be connected to at least eight ports on the DUT and SHOULD be connected using all the DUT ports.

1) Measure two groups with eight DUT ports

First iteration: measure the packet loss for two groups with consecutive ports

The first group is composed of: ingress port 1 is sending 50% of traffic to egress port 3 and ingress port 1 is sending 50% of traffic to egress port 4. Ingress port 2 is sending line rate to egress port 4. Measure the amount of traffic loss for the traffic from ingress port 1 to egress port 3.

The second group is composed of: ingress port 5 is sending 50% of traffic to egress port 7 and ingress port 5 is sending 50% of traffic to egress port 8. Ingress port 6 is sending line rate to egress port 8. Measure the amount of traffic loss for the traffic from ingress port 5 to egress port 7.

Second iteration: repeat the first iteration by shifting all the ports from N to N+1

the first group is composed of: ingress port 2 is sending 50% of traffic to egress port 4 and ingress port 2 is sending 50% of traffic to egress port 5. Ingress port 3 is sending line rate to egress port 5. Measure the amount of traffic loss for the traffic from ingress port 2 to egress port 4.

the second group is composed of: ingress port 6 is sending 50% of traffic to egress port 8 and ingress port 6 is sending 50% of traffic to egress port 9. Ingress port 7 is sending line rate to egress port 9. Measure the amount of traffic loss for the traffic from ingress port 6 to egress port 8.

Last iteration: when the first port of the first group is connected on the last DUT port and the last port of the second group is connected to the seventh port of the DUT

Measure the amount of traffic loss for the traffic from ingress port N to egress port 2 and from ingress port 4 to egress port 6.

2) Measure with N/4 groups with N DUT ports

First iteration: Expand to fully utilize all the DUT ports in increments of four. Repeat the methodology of 1) with all the group of ports possible to achieve on the device and measure for each port group the amount of traffic loss.

Second iteration: Shift by +1 the start of each consecutive ports of groups

Last iteration: Shift by N-1 the start of each consecutive ports of groups and measure the traffic loss for each port group.

### 5.3 Reporting Format

For each test the report MUST include:

- The port configuration including the number and location of ingress and egress ports located on the DUT

- If HOLB was observed

- Percent of traffic loss

## 6. Incast Stateful and Stateless Traffic

### 6.1 Objective

The objective of this test is to measure the effect of TCP Goodput and latency with a mix of large and small flows. The test is designed to simulate a mixed environment of stateful flows that require high rates of goodput and stateless flows that require low latency.

### 6.2 Methodology

In order to simulate the effects of stateless and stateful traffic on the DUT there MUST be multiple ingress ports receiving traffic destined for the same egress port. There also MAY be a mix of stateful and stateless traffic arriving on a single ingress port. The simplest setup would be 2 ingress ports receiving traffic destined to the same egress port.

One ingress port MUST be maintaining a TCP connection trough the ingress port to a receiver connected to an egress port. Traffic in the TCP stream MUST be sent at the maximum rate allowed by the traffic generator. At the same time the TCP traffic is flowing

through the DUT the stateless traffic is sent destined to a receiver on the same egress port. The stateless traffic MUST be a microburst of 100% intensity.

It is RECOMMENDED that the ingress and egress ports are varied in multiple tests to measure the maximum microburst capacity.

The intensity of a microburst MAY be varied in order to obtain the microburst capacity at various ingress rates.

It is RECOMMENDED that all ports on the DUT be used in the test.

For example:

Stateful Traffic port variation:

During Iterations number of Egress ports MAY vary as well.

First Iteration: 1 Ingress port receiving stateful TCP traffic and 1 Ingress port receiving stateless traffic destined to 1 Egress Ports

Second Iteration: 2 Ingress port receiving stateful TCP traffic and 1 Ingress port receiving stateless traffic destined to 1 Egress Ports

Last Iteration: N-2 Ingress port receiving stateful TCP traffic and 1 Ingress port receiving stateless traffic destined to 1 Egress Ports

Stateless Traffic port variation:

During Iterations number of Egress ports MAY vary as well. First Iteration: 1 Ingress port receiving stateful TCP traffic and 1 Ingress port receiving stateless traffic destined to 1 Egress Ports

Second Iteration: 1 Ingress port receiving stateful TCP traffic and 2 Ingress port receiving stateless traffic destined to 1 Egress Ports

Last Iteration: 1 Ingress port receiving stateful TCP traffic and N-2 Ingress port receiving stateless traffic destined to 1 Egress Ports

## 6.3 Reporting Format

The report MUST include the following:

- Number of ingress and egress ports along with designation of stateful or stateless.

- Stateful goodput

- Stateless latency

## 7. References

## 7.1. Normative References

[1] Bradner, S. "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, July 1991.

[2] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, March 1999.

## 7.2. Informative References

[3] Mandeville R. and Perser J., "Benchmarking Methodology for LAN Switching Devices", RFC 2889, August 2000.

[4] Stopp D. and Hickman B., "Methodology for IP Multicast Benchmarking", BCP 26, RFC 3918, October 2004.

## 7.3. URL References

[5] Yanpei Chen, Rean Griffith, Junda Liu, Randy H. Katz, Anthony D. Joseph, "Understanding TCP Incast Throughput Collapse in Datacenter Networks", http://www.eecs.berkeley.edu/~ychen2/professional/TCPIncastWREN2009.pdf ".

## Authors' Addresses

Lucien Avramov
Cisco Systems
170 West Tasman drive
San Jose, CA 95134
United States
Phone: +1 408 526 7686
Email: lavramov@cisco.com

Jacob Rapp
Hewlett-Packard
3000 Hanover Street
Palo Alto, CA
United States
Phone: +1 650 857 3367
Email: jacob.h.rapp@hp.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: July 8, 2015

W. Cerveny
Arbor Networks
R. Bonica
Juniper Networks

January 4, 2015

# Benchmarking Neighbor Discovery
# draft-cerveny-bmwg-ipv6-nd-06

## Abstract

This document is a benchmarking instantiation of RFC 6583: "Operational Neighbor Discovery Problems" [RFC6583]. It describes a general testing procedure and measurements that can be performed to evaluate how the problems described in RFC 6583 may impact the functionality or performance of intermediate nodes.

## Requirements Language

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

## Status of This Memo

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

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

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

This Internet-Draft will expire on July 8, 2015.

## Copyright Notice

Table of Contents



## 1. Introduction

This document is a benchmarking instantiation of RFC 6583: "Operational Neighbor Discovery Problems" [RFC6583]. It describes a general testing procedure and measurements that can be performed to evaluate how the problems described in RFC 6583 may impact the functionality or performance of intermediate nodes.

## 2. Terminology

Intermediate Node A router, switch, firewall or any other device which separates end-nodes. The tests in this document can be completed with any intermediate node which maintains a neighbor cache, although not all measurements and performance characteristics may apply.

Neighbor Cache The neighbor cache is a database which correlates the link-layer address and the adjacent interface with an IPv6 address.

Neighbor Discovery See Section 1 of RFC 4861 [RFC4861]

Scanner Network The network from which the scanning tested is connected.

Scanning Interface The interface from which the scanning activity is conducted.

Stale Entry Time This is the duration for which a neighbor cache entry marked "Reachable" will continue to be marked "Reachable" if an update for the address is not received.

Target Network The network for which the scanning tests is targeted.

Target Network Destination Interface The interface that resides on the target network, which is primarily used to measure DUT performance while the scanning activity is occurring.

## 3. Overview of Relevant NDP and Intermediate Node Behavior

In a traditional network, an intermediate node must support a mapping between a connected node’s IP address and the connected node’s link-layer address and interface the node is connected to. With IPv4, this process is handled by ARP [RFC0826]. With IPv6, this process is handled by NDP and is documented in [RFC4861]. With IPv6, when a packet arrives on one of an intermediate node’s interfaces and the destination address is determined to be reachable via an adjacent network:

1. The intermediate node first determines if the destination IPv6 address is present in its neighbor cache.

2. If the address is present in the neighbor cache, the intermediate node forwards the packet to the destination node using the appropriate link-layer address and interface.

3. If the destination IPv6 address is not in the intermediate node's neighbor cache:

    1. An entry for the IPv6 address is added to the neighbor cache and the entry is marked "INCOMPLETE".

    2. The intermediate node sends a neighbor solicitation packet to the solicited-node multicast address on the interface considered on-link.

    3. If a solicited neighbor advertisement for the IPv6 address is received by the intermediate node, the neighbor cache entry is marked "REACHABLE" and remains in this state for 30 seconds.

    4. If a neighbor advertisement is not received, the intermediate node will continue sending neighbor solicitation packets every second until either a neighbor solicitation is received or the maximum number of solicitations has been sent. If a neighbor advertisement is not received in this period, the entry can be discarded.

There are two scenarios where a neighbor cache can grow to a very large size:

1. There are a large number of real nodes connected via an intermediate node's interface and a large number of these nodes are sending and receiving traffic simultaneously.

2. There are a large number of addresses for which a scanning activity is occuring and no real node will respond to the neighbor solicitation. This scanning activity can be unintentional or malicious. In addition to maintaining the "INCOMPLETE" neighbor cache entry, the intermediate node must send a neighbor solicitation packet every second for the maximum number of socicitations. With today's network link bandwidths, a scanning event could cause a lot of entries to be added to the neighbor cache and solicited for in the time that it takes for a neighbor cache entry to be discarded.

An intermediate node's neighbor cache is of a finite size and can only accommodate a specific number of entries, which can be limited by available memory or a preset operating system limit. If the maximum number of entries in a neighbor cache is reached, the intermediate node must either drop an existing entry to make space for the new entry or deny the new IP address to MAC address/ interface mapping with an entry in the neighbor cache. In an extreme

case, the intermediate node's memory may become exhausted, causing the intermediate node to crash or begin paging memory.

At the core of the neighbor discovery problems presented in RFC 6583 [RFC6583], unintentional or malicious IPv6 traffic can transit the intermediate node that resembles an IP address scan similar to an IPv4-based network scan. Unlike IPv4 networks, an IPv6 end network is typically configured with a /64 address block, allowing for upwards of 2**64 addresses. When a network node attempts to scan all the addresses in a /64 address block directly attached to the intermediate node, it is possible to create a huge amount of state in the intermediate node's neighbor cache, which may stress processing or memory resources.

Section 7.1 of RFC 6583 recommends how intermediate nodes should behave when the neighbor cache is exceeded. Section 6 of RFC 6583 [RFC6583] recommends how damage from an IPv6 address scan may be mitigated. Section 6.2 of RFC 6583 [RFC6583] discusses queue tuning.

## 4. Test Setup

The network needs to minimally have two subnets: one from which the scanner(s) source their scanning activity and the other which is the target network of the address scans.

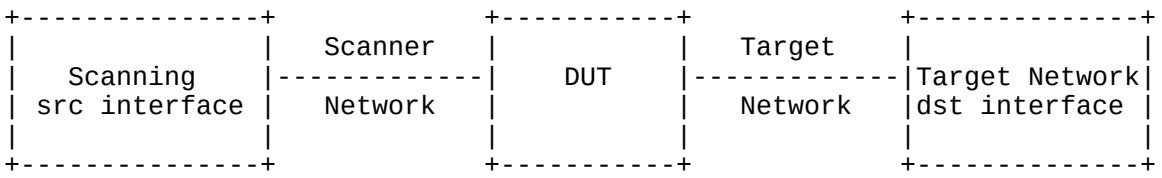
It is assumed that the latency for all network segments is neglible. By default, the target network's subnet shall be 64-bits in length, although some tests may involve increasing the prefix length.

Although packet size shouldn't have a direct impact, packet per second (pps) rates will have an impact. Smaller packet sizes should be utilized to facilitate higher packet per second rates.

For purposes of this test, the packet type being sent by the scanning device isn't important, although most scanning applications might want to send packets that would elicit responses from nodes within a subnet (such as an ICMPv6 echo request). Since it is not intended that responses be evoked from the target network node, such packets aren't necessary.

At the beginning of each test the intermediate node should be initialized. Minimally, the neighbor cache should be cleared.

Basic format of test network. Note that optional "non-participating network" is a third network not related to the scanner or target network.




### 4.1. Testing Interfaces

Two tester interfaces are configured for most tests:

- Scanning source (src) interface: This is the interface from which test packets are sourced. This interface sources traffic to destination IPv6 addresses on the target network from a single link-local address, similar to how an adjacent intermediate node would transit traffic through the intermediate node.

- Target network destination (dst) interface: This interface responds to neighbor solicitations as appropriate and confirms when an intermediate node has forwarded a packet to the interface for consumption. Where appropriate, the target network destination interface will respond to neighbor solicitations with a unique link-layer address per IPv6 address solicited.

## 5. Modifiers (Variables)

### 5.1. Frequency of NDP Triggering Packets

The frequency of NDP triggering packets can be as high as the maximum packet per second rate that the scanner network will support (or is rated for). However, it may not be necessary to send packets at a particularly high rate. In fact, a non-benchmarking goal of testing could be to identify if the DUT is able to withstand scans at rates which otherwise would not impact the performance of the DUT.

Optimistically, the scanning rate should be incremented until the DUT’s performance begins deteriorating. Depending on the software and system being used to implement the scanning, it may be challenging to achieve a sufficient rate. Where this maximum threshold cannot be determined, the test results should note the highest rate tested and that DUT performance deterioration was not noticed at this rate.

The lowest rate tested should be the rate for which packets can be expected to have an impact on the DUT -- this value is of course, subjective.

# 6. Tests

## 6.1. Stale Entry Time Determination

This test determines the time interval when the intermediate node (DUT) identifies an address as stale.

RFC 4861, section 6.3.2 [RFC4861] states that an address can be marked "stale" at a random value between 15 and 45 seconds (as defined via constants in the RFC). This test confirms what value is being used by the intermediate node. Note that RFC 4861 states that this random time can be changed "at least every few hours."

### 6.1.1. General Testing Procedure

1. Send a packet from the scanning source interface to an address in target network. Observe that the intermediate node sends a neighbor solicitation to the solicited-node multicast address on the target network, for which tester destination interface should respond with a neighbor advertisement. The intermediate node should create an entry in neighbor cache for the address, marking the address as "reachable". As this point, the packet should be forwarded to the tester destination interface.

2. After the neighbor advertisement from the destination tester interface in step one, no more neighbor advertisements from the tester destination interface should be allowed.

3. Continue sending packets from the scanning source interface to the same address in the target network.

4. Note the time at which the DUT no longer sends packets. The stale timer value will be the period of time between when the DUT received the first neighbor advertisement above and the point at which the DUT no longer forwards packets for this flow to the tester destination interface.

## 6.2. Neighbor Cache Exhaustion Determination

Discover the point at which the neighbor cache is exhausted and evaluate intermediate node behavior when this threshold is reached. If possible, the stale timer value should be locked down to a large value. A side-effect of this test is to confirm that intermediate node behaves correctly; in particular, it shouldn’t crash.

Note that some intermediate nodes may restrict the frequency of allowed neighbor discovery packets transmitted. The maximum allowed packets per second must either be set to a value which doesn't impact the outcome of the test must allow for this restriction.

#### 6.2.1. General Testing Procedure

1. At a very fast rate, send packets incrementally to valid unique addresses in the target network, within stale entry time period. Simultaneously, send packets for addresses previously added to the neighbor cache. The neighbor cache has been exhausted when previously added addresses must be re-discovered with a neighbor solicitation (within the stale entry time period).

2. Observe what happens when one address greater than the maximum neighbor cache size ("n") is reached. When "n+1" is reached, if either the first or most recent cache entry are dropped, this may be acceptable.

3. Confirm intermediate node doesn't crash when "n+1" is reached.

### 6.3. Dropped Flows Per Second

This test determines the rate that which flows are dropped once the neighbor cache size is exceeded. The metric for this test is the number of flows which are dropped in a minute.

#### 6.3.1. General Testing Procedure

1. Send packets incrementally to unique valid addresses in the target network, within stale entry time period. The number of unique valid addresses may be as high as the size of the neighbor cache, but may be the number of nodes that would be expected in a deployed network. Continue sending packets to previously cached addresses.

2. Send packets incrementally to unique invalid addresses (addresses without valid node in target network), until the intermediate node crashes, packets are no longer accepted or existing flows to unique valid addresses are dropped.

## 7. Measurements Explicitly Excluded

These are measurements which aren't recommended because of the itemized reasons below:

### 7.1. DUT CPU Utilization

This measurement relies on the DUT to provide utilization information, which is subjective.

### 7.2. Malformed Packets

This benchmarking test is not intended to test DUT behavior in the presence of malformed packets.

## 8. DUT Initialization

At the beginning of each test, the neighbor cache of the DUT should be initialized.

## 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

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 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.

## 11. Acknowledgements

Helpful comments and suggestions were offered by Al Morton, Joel Jaeggli, Nalini Elkins, Scott Bradner, and Ram Krishnan, on the BMWG e-mail list and at BMWG meetings. Precise grammatical corrections and suggestions were offered by Ann Cerveny.

## 12. References

### 12.1. Normative References

[RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware", STD 37, RFC 826, November 1982.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, March 1999.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[RFC5180] Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, May 2008.

[RFC6583] Gashinsky, I., Jaeggli, J., and W. Kumari, "Operational Neighbor Discovery Problems", RFC 6583, March 2012.

### 12.2. Informative References

[RFC7048] Nordmark, E. and I. Gashinsky, "Neighbor Unreachability Detection Is Too Impatient", RFC 7048, January 2014.

## Authors' Addresses

Bill Cerveny
Arbor Networks
2727 South State Street
Ann Arbor, MI 48104
USA

Email: wcerveny@arbor.net

Ron Bonica
Juniper Networks
2251 Corporate Park Drive
Herndon, VA 20170
USA

Email: rbonica@juniper.net

Internet Engineering Task Force
Internet-Draft, Intended status: Informational
Expires: April 20, 2015
October 17, 2014

L. Avramov
Cisco Systems
J. Rapp
Hewlett-Packard

# Data Center Benchmarking Definitions and Metrics
draft-dcbench-def-02

Abstract

The purpose of this informational document is to establish definitions, discussion and measurement techniques for data center benchmarking. Also, it is to introduce new terminologies applicable to data center performance evaluations. The purpose of this document is not to define the test methodology, but rather establish the important concepts when one is interested in benchmarking network equipment in the data center.

Table of Contents

## 1. Introduction

Traffic patterns in the data center are not uniform and are contently changing. They are dictated by the nature and variety of applications utilized in the data center. It can be largely east-west traffic flows in one data center and north-south in another, while some may combine both. Traffic patterns can be bursty in nature and contain many-to-one, many-to-many, or one-to-many flows. Each flow may also be small and latency sensitive or large and throughput sensitive while containing a mix of UDP and TCP traffic. All of which can coexist in a single cluster and flow through a single network device all at the same time. Benchmarking of network devices have long used RFC1242, RFC2432, RFC2544, RFC2889 and RFC3918. These benchmarks have largely been focused around various latency attributes and max throughput of the Device Under Test being benchmarked. These standards are good at measuring theoretical max throughput, forwarding rates and latency under testing conditions, but to not represent real traffic patterns that may affect these networking devices.

The following defines a set of definitions, metrics and terminologies including congestion scenarios, switch buffer analysis and redefines basic definitions in order to represent a wide mix of traffic conditions.

### 1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [6].

### 1.2. Definition format

Term to be defined. (e.g., Latency)

Definition: The specific definition for the term.

Discussion: A brief discussion about the term, it’s application and any restrictions on measurement procedures.

Measurement Units: Methodology for the measure and units used to report measurements of this term, if applicable.

## 2. Latency

### 2.1. Definition

Latency is a the amount of time it takes a frame to transit the DUT.

The Latency interval can be assessed between different combinations of events, irrespectively of the type of switching device (bit forwarding aka cut-through or store forward type of device)

Traditionally the latency measurement definitions are:

FILO (First In Last Out) The time interval starting when the end of the first bit of the input frame reaches the input port and ending when the last bit of the output frame is seen on the output port

FIFO (First In First Out) The time interval starting when the end of the first bit of the input frame reaches the input port and ending when the start of the first bit of the output frame is seen on the output port

LILO (Last In Last Out) The time interval starting when the last bit of the input frame reaches the input port and the last bit of the output frame is seen on the output port

LIFO (Last In First Out) The time interval starting when the last bit of the input frame reaches the input port and ending when the

first bit of the output frame is seen on the output port.

Another possibility to summarize the four different definitions above is to refer to the bit position as they normally occur: input to output.

FILO is FL (First bit Last bit)  FIFO is FF (First bit First bit)  LILO is LL (Last bit Last bit)  LIFO is LF (Last bit First bit)

This definition explained in this section in context of data center switching benchmarking is in lieu of the previous definition of Latency defined in RFC 1242, section 3.8 and is quoted here:

For store and forward devices: The time interval starting when the last bit of the input frame reaches the input port and ending when the first bit of the output frame is seen on the output port.

For bit forwarding devices: The time interval starting when the end of the first bit of the input frame reaches the input port and ending when the start of the first bit of the output frame is seen on the output port.

## 2.2 Discussion

FILO is the most important measuring definition. Any type of switches MUST be measured with the FILO mechanism: FILO will include the latency of the switch and the latency of the frame as well as the serialization delay. It is a picture of the 'whole' latency going through the DUT. For applications, which are latency sensitive and can function with initial bytes of the frame, FIFO MAY be an additional type of measuring to supplement FILO.

LIFO mechanism can be used with store forward type of switches but not with cut-through type of switches, as it will provide negative latency values for larger packet sizes. Therefore this mechanism MUST NOT be used when comparing latencies of two different DUTs.

## 2.3 Measurement Units

The measuring methods to use for benchmarking purposes are as follow:

1) FILO MUST be used as a measuring method, as this will include the latency of the packet; and today the application commonly need to read the whole packet to process the information and take an action.

2) FIFO MAY be used for certain applications able to proceed data as

the first bits arrive (FPGA for example)

3) LIFO MUST not be used, because it subtracts the latency of the packet; unlike all the other methods.

# 3 Jitter

## 3.1 Definition

The definition of Jitter is covered extensively in RFC 3393. This definition is not meant to replace that definition, but it is meant to provide guidance of use for data center network devices.

The use of Jitter is in according with the variation delay definition from RFC 3393:

The second meaning has to do with the variation of a metric (e.g., delay) with respect to some reference metric (e.g., average delay or minimum delay). This meaning is frequently used by computer scientists and frequently (but not always) refers to variation in delay.

Even with the reference to RFC 3393, there are many definitions of "jitter" possible. The one selected for Data Center Benchmarking is closest to RFC 3393.

## 3.2 Discussion

Jitter can be measured in different scenarios:-packet to packet delay variation-delta between min and max packet delay variation for all packets sent.

## 3.3 Measurement Units

The jitter MUST be measured when sending packets of the same size. Jitter MUST be measured as packet to packet delay variation and delta between min and max packet delay variation of all packets sent. A histogram MAY be provided as a population of packets measured per latency or latency buckets.

# 4 Physical Layer Calibration

## 4.1 Definition

The calibration of the physical layer consists of defining and measuring the latency of the physical devices used to perform test on

the DUT.

It includes the list of all physical layer components used as listed here after:

-type of device used to generate traffic / measure traffic

-type of line cards used on the traffic generator

-type of transceivers on traffic generator

-type of transceivers on DUT

-type of cables

-length of cables

-software name, and version of traffic generator and DUT

-list of enabled features on DUT MAY be provided and is recommended [especially the control plane protocols such as LLDP, Spanning-Tree etc.]. A comprehensive configuration file MAY be provided to this effect.

### 4.2 Discussion

Physical layer calibration is part of the end to end latency, which should be taken into acknowledgment while evaluating the DUT. Small variations of the physical components of the test may impact the latency being measure so they MUST be described when presenting results.

### 4.3 Measurement Units

It is RECOMMENDED to use all cables of : the same type, the same length, when possible using the same vendor. It is a MUST to document the cables specifications on section [4.1s] along with the test results. The test report MUST specify if the cable latency has been removed from the test measures or not. The accuracy of the traffic generator measure MUST be provided [this is usually a value in the 20ns range for current test equipment].

## 5 Line rate

### 5.1 Definition

The transmit timing, or maximum transmitted data rate is controlled by the "transmit clock" in the DUT. The receive timing (maximum ingress data rate) is derived from the transmit clock of the connected interface.

The line rate or physical layer frame rate is the maximum capacity to send frames of a specific size at the transmit clock frequency of the DUT.

The term port capacity term defines the maximum speed capability for the given port; for example 1GE, 10GE, 40GE, 100GE etc.

The frequency ("clock rate") of the transmit clock in any two connected interfaces will never be precisely the same, therefore a tolerance is needed, this will be expressed by Parts Per Million (PPM) value. The IEEE standards allow a specific +/- variance in the transmit clock rate, and Ethernet is designed to allow for small, normal variations between the two clock rates. This results in a tolerance of the line rate value when traffic is generated from a testing equipment to a DUT.

## 5.2 Discussion

For a transmit clock source, most Ethernet switches use "clock modules" (also called "oscillator modules") that are sealed, internally temperature-compensated, and very accurate. The output frequency of these modules is not adjustable because it is not necessary. Many test sets, however, offer a software-controlled adjustment of the transmit clock rate, which should be used to compensate the test equipment to not send more than line rate of the DUT.

To allow for the minor variations typically found in the clock rate of commercially-available clock modules and other crystal-based oscillators, Ethernet standards specify the maximum transmit clock rate variation to be not more than +/- 100 PPM (parts per million) from a calculated center frequency. Therefore a DUT must be able to accept frames at a rate within +/- 100 PPM to comply with the standards.

Very few clock circuits are precisely +/- 0.0 PPM because:

1.The Ethernet standards allow a maximum of +/- 100 PPM (parts per million) variance over time. Therefore it is normal for the frequency of the oscillator circuits to experience variation over time and over a wide temperature range, among external factors.

2.The crystals or clock modules, usually have a specific +/- PPM variance that is significantly better than +/- 100 PPM. Often times this is +/- 30 PPM or better in order to be considered a "certification instrument".

When testing an Ethernet switch throughput at "line rate", any specific switch will have a clock rate variance. If a test set is running +1 PPM faster than a switch under test, and a sustained line rate test is performed, a gradual increase in latency and eventually packet drops as buffers fill and overflow in the switch can be observed. Depending on how much clock variance there is between the two connected systems, the effect may be seen after the traffic stream has been running for a few hundred microseconds, a few milliseconds, or seconds. The same low latency and no-packet-loss can be demonstrated by setting the test set link occupancy to slightly less than 100 percent link occupancy. Typically 99 percent link occupancy produces excellent low-latency and no packet loss. No Ethernet switch or router will have a transmit clock rate of exactly +/- 0.0 PPM. Very few (if any) test sets have a clock rate that is precisely +/- 0.0 PPM.

Test set equipment manufacturers are well-aware of the standards, and allows a software-controlled +/- 100 PPM "offset" (clock-rate adjustment) to compensate for normal variations in the clock speed of "devices under test". This offset adjustment allows engineers to determine the approximate speed the connected device is operating, and verify that it is within parameters allowed by standards.

## 5.3 Measurement Units

"Line Rate" CAN be measured in terms of "Frame Rate":

Frame Rate = Transmit-Clock-Frequency / (Frame-Length*8 + Minimum_Gap + Preamble + Start-Frame Delimiter)

Example for 1 GB Ethernet speed with 64-byte frames: Frame Rate = 1,000,000,000 /(64*8 + 96 + 56 + 8) Frame Rate = 1,000,000,000 / 672 Frame Rate = 1,488,095.2 frames per second.

Considering the allowance of +/- 100 PPM, a switch may "legally" transmit traffic at a frame rate between 1,487,946.4 FPS and 1,488,244 FPS. Each 1 PPM variation in clock rate will translate to a 1.488 frame-per-second frame rate increase or decrease.

In a production network, it is very unlikely to see precise line rate over a very brief period. There is no observable difference between

dropping packets at 99% of line rate and 100% of line rate. -Line rate CAN measured at 100% of line rate with a -100PPM adjustment. -Line rate SHOULD be measured at 99,98% with 0 PPM adjustment.-The PPM adjustment SHOULD only be used for a line rate type of measurement

# 6 Buffering

## 6.1 Buffer

### 6.1.1 Definition

Buffer Size: the term buffer size, represents the total amount of frame buffering memory available on a DUT. This size is expressed in Byte; KB (kilobytes), MB (megabytes) or GB (gigabyte). When the buffer size is expressed it SHOULD be defined by a size metric defined above. When the buffer size is expressed, an indication of the frame MTU used for that measurement is also necessary as well as the cos or dscp value set; as often times the buffers are carved by quality of service implementation. (please refer to the buffer efficiency section for further details).

Example: Buffer Size of DUT when sending 1518 bytes frames is 18 Mb.

Port Buffer Size: the port buffer size is the amount of buffer a single ingress port, egress port or combination of ingress and egress buffering location for a single port. The reason of mentioning the three locations for the port buffer is, that the DUT buffering scheme can be unknown or untested, and therefore the indication of where the buffer is located helps understand the buffer architecture and therefore the total buffer size. The Port Buffer Size is an informational value that MAY be provided from the DUT vendor. It is not a value that is tested by benchmarking. Benchmarking will be done using the Maximum Port Buffer Size or Maximum Buffer Size methodology.

Maximum Port Buffer Size: this is in most cases the same as the Port Buffer Size. In certain switch architecture called SoC (switch on chip), there is a concept of port buffer and shared buffer pool available for all ports. Maximum Port Buffer, defines the scenario of a SoC buffer, where this amount in B (byte), KB (kilobyte), MB (megabyte) or GB (gigabyte) would represent the sum of the port buffer along with the maximum value of shared buffer this given port can take. The Maximum Port Buffer Size needs to be expressed along with the frame MTU used for the measurement and the cos or dscp bit value set for the test.

Example: a DUT has been measured to have 3KB of port buffer for 1518

frame size packets and a total of 4.7 MB of maximum port buffer for 1518 frame size packets and a cos of 0.

Maximum DUT Buffer Size: this is the total size of Buffer a DUT can be measured to have. It is most likely different than the Maximum Port Buffer Size. It CAN also be different from the sum of Maximum Port Buffer Size. The Maximum Buffer Size needs to be expressed along with the frame MTU used for the measurement and along with the cos or dscp value set during the test.

Example: a DUT has been measured to have 3KB of port buffer for 1518 frame size packets and a total of 4.7 MB of maximum port buffer for 1518 frame size packets. The DUT has a Maximum Buffer Size of 18 MB at 1500 bytes and a cos of 0.

Burst: The burst is a fixed number of packets sent over a percentage of linerate of a defined port speed. The amount of frames sent are evenly distributed across the interval T. A constant C, can be defined to provide the average time between two consecutive packets evenly spaced.

Microburst: it is a burst. A microburst is when packet drops occur when there is not sustained or noticeable congestion upon a link or device. A characterization of microburst is when the Burst is not evenly distributed over T, and is less than the constant C [C= average time between two consecutive packets evenly spaced out].

Intensity of Microburst: this is a percentage, representing the level of microburst between 1 and 100%. The higher the number the higher the microburst is. I=[1-[ (TP2-Tp1)+(Tp3-Tp2)+....(TpN-Tp(n-1) ] / Sum(packets)]]*100

### 6.1.3 Discussion

When measuring buffering on a DUT, it is important to understand what the behavior is for each port, and also for all ports as this will provide an evidence of the total amount of buffering available on the switch. The terms of buffer efficiency here helps one understand what is the optimum packet size for the buffer to be used, or what is the real volume of buffer available for a specific packet size. This section does not discuss how to conduct the test methodology, it rather explains the buffer definitions and what metrics should be provided for a comprehensive data center device buffering benchmarking.

### 6.1.3 Measurement Units

When Buffer is measured:-the buffer size MUST be measured-the port

buffer size MAY be provided for each port-the maximum port buffer size MUST be measured-the maximum DUT buffer size MUST be measured-the intensity of microburst MAY be mentioned when a microburst test is performed-the cos or dscp value set during the test SHOULD be provided

## 6.2 Incast

### 6.2.1 Definition

The term Incast, very commonly utilized in the data center, refers to the traffic pattern of many-to-one or many-to-many conversations. Typically in the data center it would refer to many different ingress server ports(many), sending traffic to a common uplink (one), or multiple uplinks (many). This pattern is generalized for any network as many incoming ports sending traffic to one or few uplinks. It can also be found in many-to-many traffic patterns.

Synchronous arrival time: When two, or more, frames of respective sizes L1 and L2 arrive at their respective one or multiple ingress ports, and there is an overlap of the arrival time for any of the bits on the DUT, then the frames L1 and L2 have a synchronous arrival times. This is called incast.

Asynchronous arrival time: Any condition not defined by synchronous.

Percentage of synchronization: this defines the level of overlap [amount of bits] between the frames L1,L2..Ln.

Example: two 64 bytes frames, of length L1 and L2, arrive to ingress port 1 and port 2 of the DUT. There is an overlap of 6.4 bytes between the two where L1 and L2 were at the same time on the respective ingress ports. Therefore the percentage of synchronization is 10%.

Stateful type traffic defines packets exchanged with a stateful protocol such as for example TCP.

Stateless type traffic defines packets exchanged with a stateless protocol such as for example UDP.

### 6.2.2 Discussion

In this scenario, buffers are solicited on the DUT. In a ingress buffering mechanism, the ingress port buffers would be solicited along with Virtual Output Queues, when available; whereas in an

egress buffer mechanism, the egress buffer of the one outgoing port would be used.

In either cases, regardless of where the buffer memory is located on the switch architecture; the Incast creates buffer utilization.

When one or more frames having synchronous arrival times at the DUT they are considered forming an incast.

### 6.2.3 Measurement Units

It is a MUST to measure the number of ingress and egress ports. It is a MUST to have a non null percentage of synchronization, which MUST be specified.

## 7 Application Throughput: Data Center Goodput

### 7.1. Definition

In Data Center Networking, a balanced network is a function of maximal throughput ’and’ minimal loss at any given time. This is defined by the Goodput. Goodput is the application-level throughput. It is measured in bytes / second. Goodput is the measurement of the actual payload of the packet being sent.

### 7.2. Discussion

In data center benchmarking, the goodput is a value that SHOULD be measured. It provides a realistic idea of the usage of the available bandwidth. A goal in data center environments is to maximize the goodput while minimizing the loss.

### 7.3. Measurement Units

When S is the total bytes received from all senders [not inclusive of packet headers or TCP headers - it’s the payload] and Ft is the Finishing Time of the last sender; the Goodput G is then measured by the following formula: G= S / Ft bytes per second

Example: a TCP file transfer over HTTP protocol on a 10Gb/s media. The file cannot be transferred over Ethernet as a single continuous stream. It must be broken down into individual frames of 1500 bytes when the standard MTU [Maximum Transmission Unit] is used. Each packet requires 20 bytes of IP header information and 20 bytes of TCP

header information, therefore 1460 byte are available per packet for the file transfer. Linux based systems are further limited to 1448 bytes as they also carry a 12 byte timestamp. Finally, the date is transmitted in this example over Ethernet which adds a 26 byte overhead per packet.

G= 1460/1526 x 10 Gbit/s which is 9.567 Gbit/s or 1.196 Gigabytes per second.

Please note: this example does not take into consideration additional Ethernet overhead, such as the interframe gap (a minimum of 96 bit times), nor collisions (which have a variable impact, depending on the network load).

When conducting Goodput measurements please document in addition to the 4.1 section:

-the TCP Stack used

-OS Versions

-NIC firmware version and model

For example, Windows TCP stacks and different Linux versions can influence TCP based tests results.

## 8. References

### 3.1. Normative References

[1] Bradner, S. "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, July 1991.

[2] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, March 1999.

### 3.2. Informative References

[3] Mandeville R. and Perser J., "Benchmarking Methodology for LAN Switching Devices", RFC 2889, August 2000.

[4] Stopp D. and Hickman B., "Methodology for IP Multicast Benchmarking", BCP 26, RFC 3918, October 2004.

### 3.3. URL References

[5] Yanpei Chen, Rean Griffith, Junda Liu, Randy H. Katz, Anthony D.

Joseph, "Understanding TCP Incast Throughput Collapse in Datacenter Networks", http://www.eecs.berkeley.edu/~ychen2/professional/TCPIncastWREN2009.pdf ".

## 3.4. Acknowledgments

The authors would like to thank Ian Cox and Tim Stevenson for their reviews and feedback.

## Authors' Addresses

Lucien Avramov
Cisco Systems
170 West Tasman drive
San Jose, CA 95134
United States
Phone: +1 408 526 7686
Email: lavramov@cisco.com

Jacob Rapp
Hewlett-Packard Company
3000 Hanover Street
Palo Alto, CA 94304
United States
Phone: +1 650 857 3367
Email: jacob.h.rapp@hp.com

Network Working Group
Internet Draft
Intended status: Informational
Expires: September 2015

M. Georgescu
NAIST

March 6, 2015

# Benchmarking Methodology for IPv6 Transition Technologies
## draft-georgescu-bmwg-ipv6-tran-tech-benchmarking-00.txt

## Abstract

There are benchmarking methodologies addressing the performance of network interconnect devices that are IPv4- or IPv6-capable, but the IPv6 transition technologies are outside of their scope. This document provides complementary guidelines for evaluating the performance of IPv6 transition technologies. The methodology also includes a tentative metric for benchmarking scalability.

## Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on September 6, 2015.

## Copyright Notice

Table of Contents

# 1. Introduction

The methodologies described in [RFC2544] and [RFC5180] help vendors and network operators alike analyze the performance of IPv4 and IPv6-capable network devices. The methodology presented in [RFC2544] is mostly IP version independent, while [RFC5180] contains complementary recommendations which are specific to the latest IP version, IPv6. However, [RFC5180] does not cover IPv6 transition technologies.

IPv6 is not backwards compatible, which means that IPv4-only nodes cannot directly communicate with IPv6-only nodes. To solve this issue, IPv6 transition technologies have been proposed and implemented, many of which are still in development.

This document presents benchmarking guidelines dedicated to IPv6 transition technologies. The benchmarking tests can provide insights about the performance of these technologies, which can act as useful feedback for developers, as well as for network operators going through the IPv6 transition process.

## 1.1. IPv6 Transition Technologies

Two of the basic transition technologies, dual IP layer (also known as dual stack) and encapsulation, are presented in [RFC4213]. IPv4/IPv6 Translation is presented in [RFC6144]. Most of the transition technologies employ at least one variation of these mechanisms. Some of the more complex ones (e.g. DSLite [RFC6333]) are using all three. In this context, a generic classification of the transition technologies can prove useful.

Tentatively, we can consider a basic production IP-based network as being constructed using the following components:

- a Customer Edge (CE) segment
- a Core network segment
- a Provider Edge (PE) segment

According to the technology used for the core network traversal the transition technologies can be categorized as follows:

1. Single-stack: either IPv4 or IPv6 is used to traverse the core network, and translation is used at one of the edges
2. Dual-stack: the core network devices implement both IP protocols

3. Encapsulation-based: an encapsulation mechanism is used to traverse the core network; CE nodes encapsulate the IPvX packets in IPvY packets, while PE nodes are responsible for the decapsulation process.

4. Translation-based: a translation mechanism is employed for the traversal of the core network; CE nodes translate IPvX packets to IPvY packets and PE nodes translate the packets back to IPvX.

The performance of Dual-stack transition technologies can be fully evaluated using the benchmarking methodologies presented by [RFC2544] and [RFC5180]. Consequently, this document focuses on the other 3 categories: Single-stack, Encapsulation-based, and Translation-based transition technologies.

Another important aspect by which the IPv6 transition technologies can be categorized is their use of stateful or stateless mapping algorithms. The technologies that use stateful mapping algorithms (e.g. Stateful NAT64 [RFC6146]) create dynamic correlations between IP addreesses or {IP address, transport protocol, transport port number} tuples, which are stored in a state table. For ease of reference, the IPv6 transition technologies which employ stateful mapping algorithms will be called stateful IPv6 transition technologies. The efficiency with which the state table is managed can be an important performance indicator for these technologies. Hence, for the stateful IPv6 transition technologies additional benchmarking tests are RECOMMENDED.

## 2. Conventions used in this document

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

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 [RFC2119] significance.

## 3. Test Setup

The test environment setup options recommended for IPv6 transition technologies benchmarking are very similar to the ones presented in Section 6 of [RFC2544]. In the case of the tester setup, the options presented in [RFC2544] can be applied here as well. However, the Device under test (DUT) setup options should be explained in the context of the 3 targeted categories of IPv6 transition technologies: Single-stack, Encapsulation-based and Translation-based transition technologies.

Although both single tester and sender/receiver setups are applicable to this methodology, the single tester setup will be used to describe the DUT setup options.

For the simple test setups described in the next two subsections, static routing MAY be employed. However, for more complex test setups (e.g. scalability test setups) dynamic routing is a more reasonable choice. However, the presence of routing and management frames can represent unwanted background data that can affect the benchmarking result. To that end, the procedures defined in [RFC2544] (Sections 11.2 and 11.3) related to routing and management frames SHOULD be used here as well.

## 3.1. Single-stack Transition Technologies

For the evaluation of Single-stack transition technologies a single DUT setup (see Figure 1) SHOULD be used. The DUT is responsible for translating the IPvX packets into IPvY packets. In this context, the tester device should be configured to support both IPvX and IPvY.

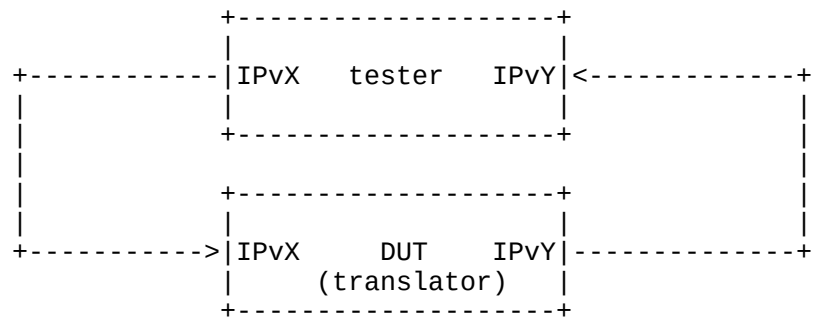



Figure 1. Test setup 1

## 3.2. Encapsulation/Translation Based Transition Technologies

For evaluating the performance of Encapsulation-based and Translation-based transition technologies a dual DUT setup (see Figure 2) SHOULD be employed. The tester creates a network flow of IPvX packets. The DUT CE is responsible for the encapsulation or translation of IPvX packets into IPvY packets. The IPvY packets are decapsulated/translated back to IPvX packets by the DUT PE and forwarded to the tester.

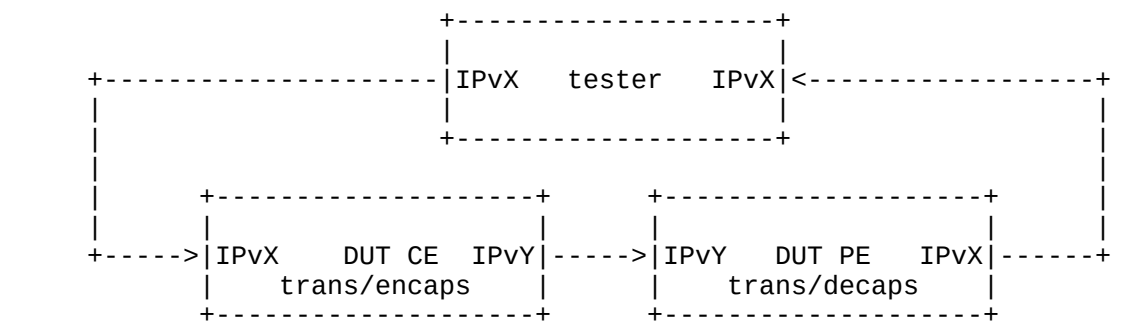



Figure 2. Test setup 2

In the case of translation based transition technology, the DUT CE and DUT PE machines MAY be tested separately as well. These tests can represent a fine grain performance analysis of the IPvX to IPvY translation direction versus the IPvY to IPvX translation direction. The tests SHOULD follow the test setup presented in Figure 1.

## 4. Test Traffic

The test traffic represents the experimental workload and SHOULD meet the requirements specified in this section. The requirements are dedicated to unicast IP traffic. Multicast IP traffic is outside of the scope of this document.

### 4.1. Frame Formats and Sizes

[RFC5180] describes the frame size requirements for two commonly used media types: Ethernet and SONET (Synchronous Optical Network). [RFC2544] covers also other media types, such as token ring and FDDI. The two documents can be referred for the dual-stack transition technologies. For the rest of the transition technologies the frame overhead introduced by translation or encapsulation MUST be considered.

The encapsulation/translation process generates different size frames on different segments of the test setup. For example, the single-stack transition technologies will create different frame sizes on the receiving segment of the test setup, as IPvX packets are translated to IPvY. This is not a problem if the bandwidth of the employed media is not exceeded. To prevent exceeding the limitations imposed by the media, the frame size overhead needs to be taken into account when calculating the maximum theoretical frame rates. The calculation methods for the two media types, Ethernet and SONET, as well as a calculation example are detailed in Appendix A.

In the context of frame size overhead MTU recommendations are needed in order to avoid frame loss due to MTU mismatch between the virtual encapsulation/translation interfaces and the physical network interface controllers (NICs). To avoid this situation, the larger MTU between the physical NICs and virtual encapsulation/translation interfaces SHOULD be set for all interfaces of the DUT and tester.

#### 4.1.1. Frame Sizes to Be Used over Ethernet

Based on the recommendations of [RFC5180], the following frame sizes SHOULD be used for benchmarking Ethernet traffic: 64, 128, 256, 512, 1024, 1280, 1518, 1522, 2048, 4096, 8192 and 9216.

The theoretical maximum frame rates considering an example of frame overhead are presented in Appendix A1.

#### 4.1.2. Frame Sizes to Be Used over SONET

Based on the recommendations of [RFC5180], the frame sizes for SONET traffic SHOULD be: 47, 64, 128, 256, 512, 1024, 1280, 1518, 2048, 4096 bytes.

An example of theoretical maximum frame rates calculation is shown in Appendix A2.

### 4.2. Protocol Addresses

The selected protocol addresses should follow the recommendations of [RFC5180](Section 5) for IPv6 and [RFC2544](Section 12) for IPv4.

Note: testing traffic with extension headers might not be possible for the transition technologies which employ translation.

### 4.3. Traffic Setup

Following the recommendations of [RFC5180], all tests described SHOULD be performed with bi-directional traffic. Uni-directional traffic tests MAY also be performed for a fine grained performance assessment.

Because of the simplicity of UDP, UDP measurements offer a more reliable basis for comparison than other transport layer protocols. Consequently, for the benchmarking tests described in Section 6 of this document UDP traffic SHOULD be employed.

Considering that the stateful transition technologies need to manage the state table for each connection, a connection-oriented transport

layer protocol needs to be used with the test traffic. Consequently, TCP test traffic SHOULD be employed for the tests described in Section 7 of this document.

## 5. Modifiers

The idea of testing under different operational conditions was first introduced in [RFC2544](Section 11) and represents an important aspect of benchmarking network elements, as it emulates to some extent the conditions of a production environment. [RFC5180] describes complementary testing conditions specific to IPv6. Their recommendations can be referred for IPv6 transition technologies testing as well.

## 6. Benchmarking Tests

The benchmarking tests condition described in [RFC2544] (Sections 24, 25, 26) are also recommended here. The following sub-sections contain the list of all recommended benchmarking tests.

### 6.1. Throughput

Objective: To determine the DUT throughput as defined in [RFC1242].

Procedure: As described by [RFC2544].

Reporting Format: As described by [RFC2544].

### 6.2. Latency

Objective: To determine the latency as defined in [RFC1242].

Procedure: As described by [RFC2544].

Reporting Format: As described by [RFC2544].

### 6.3. Frame Loss Rate

Objective: To determine the frame loss rate, as defined in [RFC1242], of a DUT throughout the entire range of input data rates and frame sizes.

Procedure: As described by [RFC2544].

Reporting Format: As described by [RFC2544].

### 6.4. Back-to-back Frames

Objective: To characterize the ability of a DUT to process back-to-back frames as defined in [RFC1242].

Procedure: As described by [RFC2544].

Reporting Format: As described by [RFC2544].

### 6.5. System Recovery

Objective: To characterize the speed at which a DUT recovers from an overload condition.

Procedure: As described by [RFC2544].

Reporting Format: As described by [RFC2544].

### 6.6. Reset

Objective: To characterize the speed at which a DUT recovers from a device or software reset.

Procedure: As described by [RFC2544].

Reporting Format: As described by [RFC2544].

## 7. Additional Benchmarking Tests for Stateful IPv6 Transition Technologies

This section describes additional tests dedicated to the stateful IPv6 transition technologies. For the tests described in this section the DUT devices SHOULD follow the test setup and test parameters recommendations presented in [RFC3511] (Sections 4, 5).

In addition to the IPv4/IPv6 transition function a network node can have a firewall function. This document is targeting only the network devices that do not have a firewall function, as this function can be benchmarked using the recommendations of [RFC3511]. Consequently, only the tests described in [RFC3511] (Sections 5.2, 5.3) are RECOMMENDED. Namely, the following additional tests SHOULD be performed:

### 7.1. Concurrent TCP Connection Capacity

Objective: To determine the maximum number of concurrent TCP connections supported through or with the DUT, as defined in [RFC 2647]. This test is supposed to find the maximum number of entries the DUT can store in its state table.

Procedure: As described by [RFC3511].

Reporting Format: As described by [RFC3511].

### 7.2. Maximum TCP Connection Establishment Rate

Objective: To determine the maximum TCP connection establishment rate through or with the DUT, as defined by RFC [2647]. This test is expected to find the maximum rate at which the DUT can update its connection table.

Procedure: As described by [RFC3511].

Reporting Format: As described by [RFC3511].

## 8. Scalability

Scalability has been often discussed; however, in the context of network devices, a formal definition or a measurement method has not yet been approached.

Scalability can be defined as the ability of each transition technology to accommodate network growth.

Poor scalability usually leads to poor performance. Considering this, scalability can be measured by quantifying the network performance degradation while the network grows.

The following subsections describe how the test setups can be modified to create network growth and how the associated performance degradation can be quantified.

### 8.1. Test Setup

The test setups defined in Section 3 have to be modified to create network growth.

#### 8.1.1. Single-stack Transition Technologies

In the case of single-stack transition technologies the network growth can be generated by increasing the number of network flows generated by the tester machine (see Figure 3).

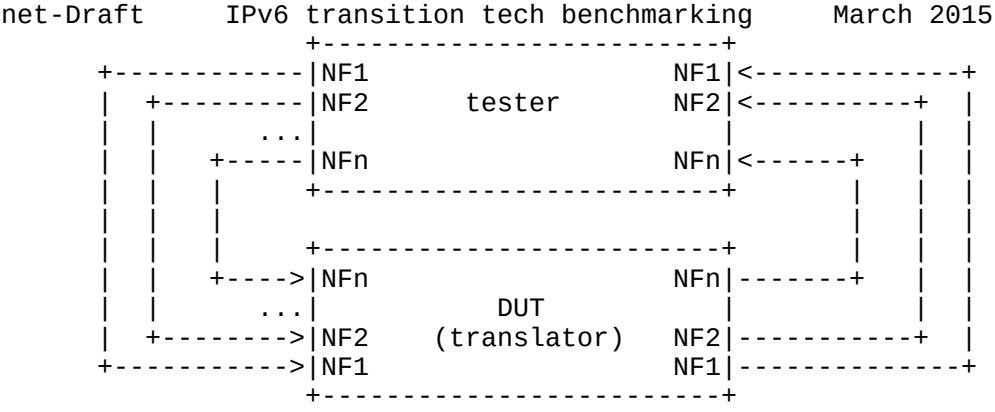



Figure 3. Test setup 3

8.1.2. Encapsulation/Translation Transition Technologies

Similarly, for the encapsulation/translation based technologies a multi-flow setup is recommended. For most transition technologies, the provider edge device is designed to support more than one customer edge network. Hence, the recommended test setup is a n:1 design, where n is the number of CE DUTs connected to the same PE DUT (See Figure 4).

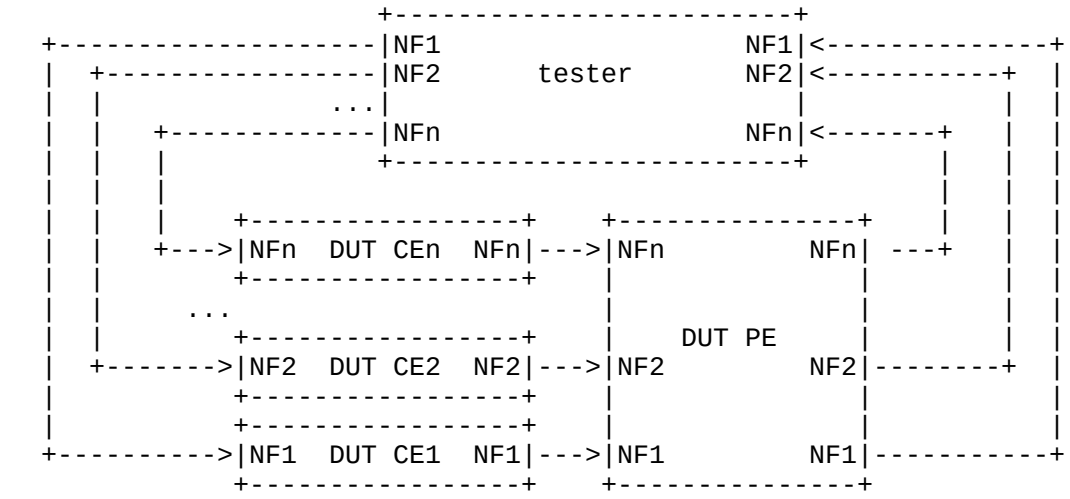



Figure 4. Test setup 4

This test setup can help to quantify the scalability of the PE device. However, for testing the scalability of the DUT CEs additional recommendations are needed.
For encapsulation based transition technologies a m:n setup can be created, where m is the number of flows applied to the same CE

device and n the number of CE devices connected to the same PE device.

For the translation based transition technologies the CE devices can be separately tested with n network flows using the test setup presented in Figure 3.

### 8.2. Benchmarking Performance Degradation

Objective: To quantify the performance degradation introduced by n parallel network flows.

Procedure: First the benchmarking tests presented in Section 6 have to be performed for one network flow.

The same tests have to be repeated for n network flows. The performance degradation of the X benchmarking dimension SHOULD be calculated as relative performance change between the 1-flow results and the n-flow results, using the following formula:

$$Xpd = \frac{Xn - X1}{X1} \times 100$$

where: X1 - result for 1-flow
Xn - result for n-flows

Reporting Format: The performance degradation SHOULD be expressed as a percentage. The number of tested parallel flows n MUST be clearly specified. For each of the performed benchmarking tests, there SHOULD be a table containing a column for each frame size. The table SHOULD also state the applied frame rate.

## 9. 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 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.

## 10. IANA Considerations

The IANA has allocated the prefix 2001:0002::/48 [RFC5180] for IPv6 benchmarking. For IPv4 benchmarking, the 198.18.0.0/15 prefix was reserved, as described in [RFC6890]. The two ranges are sufficient for benchmarking IPv6 transition technologies.

## 11. Conclusions

The methodologies described in [RFC2544] and [RFC5180] can be used for benchmarking the performance of IPv4-only, IPv6-only and dual-stack supporting network devices. This document presents complementary recommendations dedicated to IPv6 transition technologies. Furthermore, the methodology includes a tentative approach for benchmarking scalability by quantifying the performance degradation associated with network growth.

## 12. References

### 12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2234] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

[RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005.

[RFC6144] Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", RFC 6144, April 2011.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

[RFC6333] Cotton, M., Vegoda, L., Bonica, R., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC6890, April 2013.

### 12.2. Informative References

[RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", [RFC1242], July 1991.

[RFC2544] Bradner, S., McQuaid, J., "Benchmarking Methodology for Network Interconnect Devices", [RFC2544], March 1999.

[RFC2647] Newman, D., "Benchmarking Terminology for Firewall Devices", [RFC2647], August 1999.

[RFC3511] Hickman, B., Newman, D., Tadjudin, S., Martin, T., "Benchmarking Methodology for Firewall Performance", [RFC3511], April 2003.

[RFC5180] Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, May 2008.

## 13. Acknowledgments

The author would like to thank Professor Youki Kadobayashi for his constant feedback and support. The thanks should be extended to the NECOMA project members for their continuous support. Helpful comments and suggestions were offered by Scott Bradner, Al Morton, Bhuvaneswaran Vengainathan, Andrew McGregor, Nalini Elkins, Kaname Nishizuka and Yasuhiro Ohara. A special thank you to the RFC Editor Team for their thorough editorial review and helpful suggestions. This document was prepared using 2-Word-v2.0.template.dot.

# Appendix A. Theoretical Maximum Frame Rates

This appendix describes the recommended calculation formulas for the theoretical maximum frame rates to be employed over two types of commonly used media. The formulas take into account the frame size overhead created by the encapsulation or the translation process. For example, the 6in4 encapsulation described in [RFC4213] adds 20 bytes of overhead to each frame.

## A.1. Ethernet

Considering X to be the frame size and O to be the frame size overhead created by the encapsulation on translation process, the maximum theoretical frame rate for Ethernet can be calculated using the following formula:

```
            Line Rate (bps)
      ------------------------------
      (8bits/byte)*(X+O+20)bytes/frame
```

The calculation is based on the formula recommended by RFC5180 in Appendix A1. As an example, the frame rate recommended for testing a 6in4 implementation over 10Mb/s Ethernet with 64 bytes frames is:

```
            10,000,000(bps)
      ------------------------------      = 12,019 fps
      (8bits/byte)*(64+20+20)bytes/frame
```

The complete list of recommended frame rates for 6in4 encapsulation can be found in the following table:

| Frame size (bytes) | 10 Mb/s (fps) | 100 Mb/s (fps) | 1000 Mb/s (fps) | 10000 Mb/s (fps) |
|---|---|---|---|---|
| 64 | 12,019 | 120,192 | 1,201,923 | 12,019,231 |
| 128 | 7,440 | 74,405 | 744,048 | 7,440,476 |
| 256 | 4,223 | 42,230 | 422,297 | 4,222,973 |
| 512 | 2,264 | 22,645 | 226,449 | 2,264,493 |
| 1024 | 1,175 | 11,748 | 117,481 | 1,174,812 |
| 1280 | 947 | 9,470 | 94,697 | 946,970 |
| 1518 | 802 | 8,023 | 80,231 | 802,311 |
| 1522 | 800 | 8,003 | 80,026 | 800,256 |
| 2048 | 599 | 5,987 | 59,866 | 598,659 |
| 4096 | 302 | 3,022 | 30,222 | 302,224 |
| 8192 | 152 | 1,518 | 15,185 | 151,846 |
| 9216 | 135 | 1,350 | 13,505 | 135,048 |

A.2. SONET

Similarly for SONET, if X is the target frame size and O the frame size overhead, the recommended formula for calculating the maximum theoretical frame rate is:

```
            Line Rate (bps)
      ------------------------------
      (8bits/byte)*(X+O+1)bytes/frame
```

The calculation formula is based on the recommendation of RFC5180 in Appendix A2.

As an example, the frame rate recommended for testing a 6in4 implementation over a 10Mb/s PoS interface with 64 bytes frames is:

```
            10,000,000(bps)
      ------------------------------      = 14,706 fps
      (8bits/byte)*(64+20+1)bytes/frame
```

The complete list of recommended frame rates for 6in4 encapsulation can be found in the following table:

| Frame size (bytes) | 10 Mb/s (fps) | 100 Mb/s (fps) | 1000 Mb/s (fps) | 10000 Mb/s (fps) |
|---|---|---|---|---|
| 47 | 18,382 | 183,824 | 1,838,235 | 18,382,353 |
| 64 | 14,706 | 147,059 | 1,470,588 | 14,705,882 |
| 128 | 8,389 | 83,893 | 838,926 | 8,389,262 |
| 256 | 4,513 | 45,126 | 451,264 | 4,512,635 |
| 512 | 2,345 | 23,452 | 234,522 | 2,345,216 |
| 1024 | 1,196 | 11,962 | 119,617 | 1,196,172 |
| 2048 | 604 | 6,042 | 60,416 | 604,157 |
| 4096 | 304 | 3,036 | 30,362 | 303,619 |

## Authors' Addresses

Marius Georgescu
Nara Institute of Science and Technology (NAIST)
Takayama 8916-5
Nara
Japan

Phone: +81 743 72 5216
Email: liviumarius-g@is.naist.jp

Network Working Group
Internet Draft
Intended status: Informational
Expires: March 2015

M. Georgescu
NAIST

September 24, 2014

# IPv6 Transition Technologies Benchmarking Methodology
## draft-georgescu-ipv6-transition-tech-benchmarking-00.txt

## Status of this Memo



## Copyright Notice

Abstract

There are benchmarking methodologies addressing the performance of network interconnect devices which are IPv4 or IPv6-capable. However, the IPv6 transition technologies are outside of their scope. This document provides complementary guidelines for evaluating the performance of IPv6 transition technologies. The methodology also includes a tentative metric for benchmarking scalability.

Table of Contents

# 1. Introduction

The methodologies described in [RFC2544] and [RFC5180] help vendors and network operators alike analyze the performance of IPv4 and IPv6-capable network devices. The methodology presented in [RFC2544] is mostly IP version independent, while [RFC5180] contains complementary recommendations which are specific to the latest IP version, IPv6. However, [RFC5180] does not cover IPv6 transition technologies.

IPv6 is not backwards compatible, which means that IPv4-only nodes cannot directly communicate with IPv6-only nodes. To solve this issue, IPv6 transition technologies have been proposed and implemented, many of which are still in development.

This document presents benchmarking guidelines dedicated to IPv6 transition technologies. The benchmarking tests can provide insights about the performance of these technologies, which can act as useful feedback for developers, as well as for network operators going through the IPv6 transition process.

## 1.1. IPv6 transition technologies

Two of the basic transition technologies dual IP layer (also known as dual stack) and encapsulation are presented in [RFC4213]. IPv4/IPv6 Translation is presented in [RFC6144]. Most of the transition technologies employ at least one variation of these mechanisms. Some of the more complex ones (e.g. DSLite [RFC6333]) are using all three. In this context, a generic classification of the transition technologies can prove useful.

Tentatively, we can consider a basic production IP-based network as being constructed using the following components:

- a Customer Edge (CE) segment
- a Core network segment
- a Provider Edge (PE) segment

According to the technology used for the core network traversal the transition technologies can be categorized as follows:

1. Single-stack: either IPv4 or IPv6 is used to traverse the core network and translation is used at one of the edges
2. Dual-stack: the core network devices implement both IP protocols

3. Encapsulation-based: an encapsulation mechanism is used to traverse the core network; CE nodes encapsulate the IPvX packets in IPvY packets, while PE nodes are responsible for the decapsulation process.

4. Translation-based: a translation mechanism is employed for the traversal of the network core; CE nodes translate IPvX packets to IPvY packets and PE nodes translate the packets back to IPvX.

The performance of Dual-stack transition technologies can be very well evaluated using the benchmarking methodology presented by [RFC2544] and [RFC5180]. Consequently the focus of this document is represented by the other 3 categories: Single-stack, Encapsulation-based and Translation-based transition technologies.

## 2. Conventions used in this document

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

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 [RFC2119] significance.

## 3. Test environment setup

The test environment setup options recommended for IPv6 transition technologies benchmarking are very similar to the ones presented in Section 6 of [RFC2544]. In the case of the tester setup, the options presented in [RFC2544] can be applied here as well. However, the Device under test (DUT) setup options should be explained in the context of the 3 targeted categories of IPv6 transition technologies: Single-stack, Encapsulation-based and Translation-based transition technologies.

Although both single tester and sender/receiver setups are applicable to this methodology, the single tester setup will be used to describe the DUT setup options.

### 3.1. Single-stack transition technologies

For the evaluation of Single-stack transition technologies a single DUT setup (see Figure 1) SHOULD be used. The DUT is responsible for translating the IPvX packets into IPvY packets. In this context, the tester device should be configured to support both IPvX and IPvY.

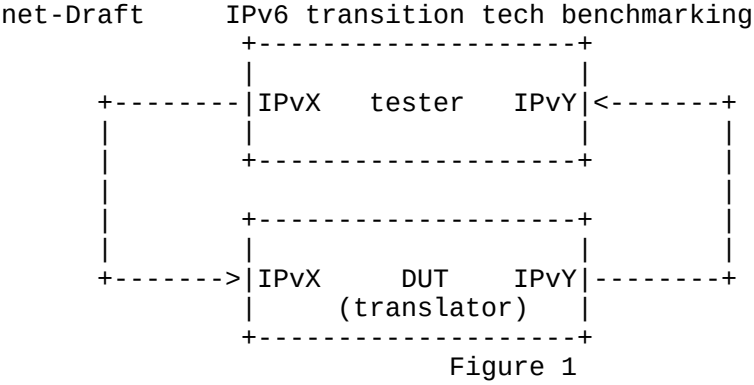



Figure 1

### 3.2. Encapsulation/Translation based transition technologies

For evaluating the performance of Encapsulation-based and Translation-based transition technologies a dual DUT setup (see Figure 2) SHOULD be employed. The tester creates a network flow of IPvX packets. The DUT CE is responsible for the encapsulation or translation of IPvX packets into IPvY packets. The IPvY packets are decapsulated/translated back to IPvX packets by the DUT PE and forwarded to the tester.

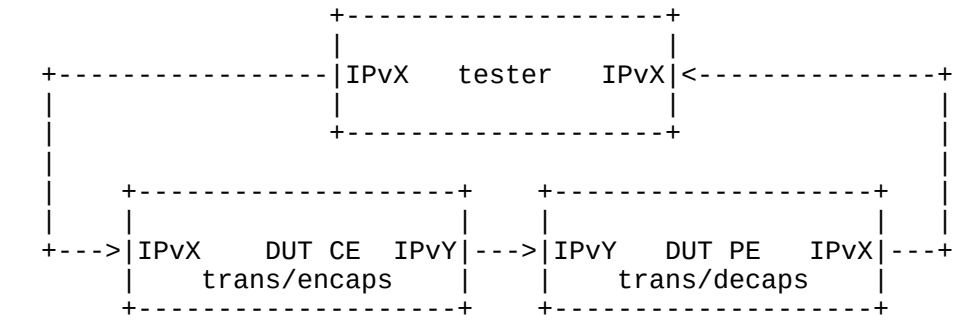



Figure 2

## 4. Test traffic

The test traffic represents the experimental workload and SHOULD meet the requirements specified in this section. The requirements are dedicated to unicast IP traffic.

### 4.1. Frame formats and sizes

[RFC5180] describes the frame size requirements for two commonly used media types: Ethernet and SONET (Synchronous Optical Network).

[RFC2544] covers also other media types, such as token ring and FDDI. The two documents can be referred for the dual-stack transition technologies. For the rest of the transition technologies the frame overhead introduced by translation or encapsulation MUST be considered.

The encapsulation/translation process generates different size frames on different segments of the test setup. For example, the single-stack transition technologies will create different frame sizes on the receiving segment of the test setup, as IPvX packets are translated to IPvY. This is not a problem if the bandwidth of the employed media is not exceeded. To prevent exceeding the limitations imposed by the media, the frame size overhead needs to be taken into account when calculating the maximum theoretical frame rates. The calculation methods for the two media types, Ethernet and SONET, as well as a calculation example are detailed in Appendix A.

#### 4.1.1. Frame sizes to be used over Ethernet

Based on the recommendations of [RFC5180], the following frame sizes SHOULD be used for benchmarking Ethernet traffic: 64, 128, 256, 512, 1024, 1280, 1518, 1522, 2048, 4096, 8192 and 9216.

The theoretical maximum frame rates considering an example of frame overhead are presented in Appendix A1.

#### 4.1.2. Frame sizes to be used over SONET

Based on the recommendations of [RFC5180], the frame sizes for SONET traffic SHOULD be: 47, 64, 128, 256, 512, 1024, 1280, 1518, 2048, 4096 bytes.

An example of theoretical maximum frame rates calculation is shown in Appendix A2.

### 4.2. Protocol addresses

The selected protocol addresses should follow the recommendations of [RFC5180](Section 5) for IPv6 and [RFC2544](Section 12) for IPv4.

Note: testing traffic with extension headers might not be possible for the transition technologies which employ translation.

### 4.3. Traffic setup

Following the recommendations of [RFC5180], all tests described SHOULD be performed with bi-directional traffic. Uni-directional traffic tests MAY also be performed for a fine grained performance assessment.

## 5. Modifiers

The idea of testing under different operational conditions was first introduced in [RFC2544](Section 11) and represents an important aspect of benchmarking network elements, as it emulates to some extent the conditions of a production environment. [RFC5180] describes complementary testing conditions specific to IPv6. Their recommendations can be referred for IPv6 transition technologies testing as well.

## 6. Benchmarking tests

The benchmarking tests condition described in [RFC2544] (Sections 24, 25, 26) are also recommended here. The following sub-sections contain the list of all recommended benchmarking tests.

### 6.1. Throughput

Objective: To determine the DUT throughput as defined in [RFC1242].

Procedure: As described by [RFC2544].

Reporting Format: As described by [RFC2544].

### 6.2. Latency

Objective: To determine the latency as defined in [RFC1242].

Procedure: As described by [RFC2544].

Reporting Format: As described by [RFC2544].

### 6.3. Frame loss rate

Objective: To determine the frame loss rate, as defined in [RFC1242], of a DUT throughout the entire range of input data rates and frame sizes.

Procedure: As described by [RFC2544].

Reporting Format: As described by [RFC2544].

### 6.4. Back-to-back frames

Objective: To characterize the ability of a DUT to process back-to-back frames as defined in [RFC1242].

Procedure: As described by [RFC2544].

Reporting Format: As described by [RFC2544].

### 6.5. System recovery

Objective: To characterize the speed at which a DUT recovers from an overload condition.

Procedure: As described by [RFC2544].

Reporting Format: As described by [RFC2544].

### 6.6. Reset

Objective: To characterize the speed at which a DUT recovers from a device or software reset.

Procedure: As described by [RFC6201].

Reporting Format: As described by [RFC6201].

## 7. Scalability

Scalability has been often discussed, however, in the context of network devices, a formal definition or a measurement method have not been approached yet.

Scalability can be defined as the ability of each transition technology to accommodate network growth.

Poor scalability usually leads to poor performance. Considering this, scalability can be measured by quantifying the network performance degradation while the network grows.

### 7.1. Test setup

The test setups defined in Section 3 have to be modified to create network growth.

#### 7.1.1. Single-stack transition technologies

In the case of single-stack transition technologies the network growth can be generated by increasing the number of network flows generated by the tester machine (see Figure 3).

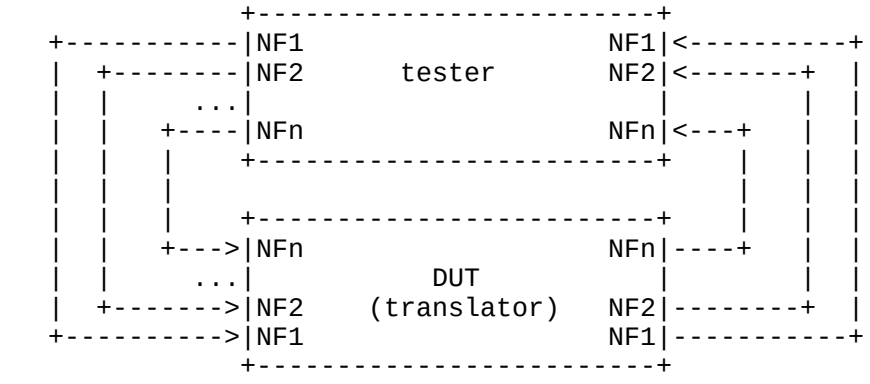



Figure 3

7.1.2. Encapsulation/Translation transition technologies

Similarly, for the encapsulation/translation based technologies a multi-flow setup is recommended. As for most transition technologies the provider edge device is designed to support more than one customer edge network, the recommended test setup is a n:1 design, where n is the number of CE DUTs connected to the same PE DUT (See Figure 4).

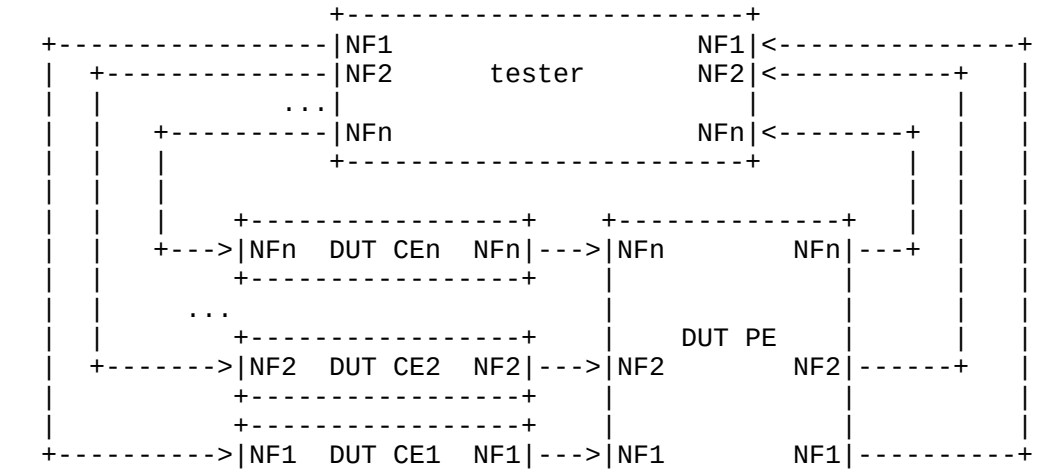



Figure 4

7.2. Benchmarking performance degradation

Objective: To quantify the performance degradation introduced by n parallel network flows.

Procedure: First the benchmarking tests presented in Section 6 have to be performed for one network flow.

The same tests have to be repeated for n-network flows. The performance degradation of the X benchmarking dimension SHOULD be calculated as relative performance change between the 1-flow results and the n-flow results, using the following formula:

$$Xpd = \frac{Xn - X1}{X1} \times 100$$ , where: X1 - result for 1-flow
Xn - result for n-flows

Reporting Format: The performance degradation SHOULD be expressed as a percentage. The number of tested parallel flows n MUST be clearly specified. For each of the performed benchmarking tests there SHOULD be a table containing a column for each frame size, stating also the applied frame rate.

## 8. Security Considerations

The benchmarking methodology described in this document MUST be used in conjunction with a controlled experimental environment.

The benchmarking environment MUST be isolated and the generated traffic MUST NOT be forwarded into production networks.

Given the isolated nature of the experimental environment, no other security considerations are required.

## 9. IANA Considerations

The IANA has allocated the prefix 2001:0002::/48 [RFC5180] for IPv6 benchmarking. For IPv4 benchmarking, the 198.18.0.0/15 prefix was reserved, as described in [RFC6890]. The two ranges are sufficient for benchmarking IPv6 transition technologies.

## 10. Conclusions

The methodologies described in [RFC2544] and [RFC5180] can be used for benchmarking the performance of IPv4-only, IPv6-only and dual-stack supporting network devices. This document presents complementary recommendations dedicated to IPv6 transition technologies. Furthermore, the methodology includes a tentative approach for benchmarking scalability by quantifying the performance degradation associated with network growth.

# 11. References

## 11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2234] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

[RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005.

[RFC6144] Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", RFC 6144, April 2011.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

[RFC6333] Cotton, M., Vegoda, L., Bonica, R., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC6890, April 2013.

## 11.2. Informative References

[RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", [RFC1242], July 1991.

[RFC2544] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", [RFC1242], July 1991.

[RFC5180] Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, May 2008.

[RFC6201] Asati, R., Pignataro, C., Calabria, F., and C. Olvera, "Device Reset Characterization", RFC 6201, March 2011.

# 12. Acknowledgments

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

# Appendix A. Theoretical maximum frame rates

This appendix describes the recommended calculation formulas for the theoretical maximum frame rates to be employed over two types of commonly used media. The formulas take into account the frame size overhead created by the encapsulation or the translation process. For example, the 6in4 encapsulation described in [RFC4213] adds 20 bytes of overhead to each frame.

## A.1. Ethernet

Considering X to be the frame size and O to be the frame size overhead created by the encapsulation on translation process, the maximum theoretical frame rate for Ethernet can be calculated using the following formula:

```
            Line Rate (bps)
      ------------------------------
      (8bits/byte)*(X+O+20)bytes/frame
```

The calculation is based on the formula recommended by RFC5180 in Appendix A1. As an example, the frame rate recommended for testing a 6in4 implementation over 10Mb/s Ethernet with 64 bytes frames is:

```
            10,000,000(bps)
      ------------------------------      = 12,019 fps
      (8bits/byte)*(64+20+20)bytes/frame
```

The complete list of recommended frame rates for 6in4 encapsulation can be found in the following table:

| Frame size (bytes) | 10 Mb/s (fps) | 100 Mb/s (fps) | 1000 Mb/s (fps) | 10000 Mb/s (fps) |
|---|---|---|---|---|
| 64 | 12,019 | 120,192 | 1,201,923 | 12,019,231 |
| 128 | 7,440 | 74,405 | 744,048 | 7,440,476 |
| 256 | 4,223 | 42,230 | 422,297 | 4,222,973 |
| 512 | 2,264 | 22,645 | 226,449 | 2,264,493 |
| 1024 | 1,175 | 11,748 | 117,481 | 1,174,812 |
| 1280 | 947 | 9,470 | 94,697 | 946,970 |
| 1518 | 802 | 8,023 | 80,231 | 802,311 |
| 1522 | 800 | 8,003 | 80,026 | 800,256 |
| 2048 | 599 | 5,987 | 59,866 | 598,659 |
| 4096 | 302 | 3,022 | 30,222 | 302,224 |
| 8192 | 152 | 1,518 | 15,185 | 151,846 |
| 9216 | 135 | 1,350 | 13,505 | 135,048 |

A.2. SONET

Similarly for SONET, if X is the target frame size and O the frame size overhead, the recommended formula for calculating the maximum theoretical frame rate is:

```
            Line Rate (bps)
      ------------------------------
      (8bits/byte)*(X+O+1)bytes/frame
```

The calculation formula is based on the recommendation of RFC5180 in Appendix A2.

As an example, the frame rate recommended for testing a 6in4 implementation over a 10Mb/s PoS interface with 64 bytes frames is:

```
            10,000,000(bps)
      ------------------------------      = 14,706 fps
      (8bits/byte)*(64+20+1)bytes/frame
```

The complete list of recommended frame rates for 6in4 encapsulation can be found in the following table:

| Frame size (bytes) | 10 Mb/s (fps) | 100 Mb/s (fps) | 1000 Mb/s (fps) | 10000 Mb/s (fps) |
|---|---|---|---|---|
| 47 | 18,382 | 183,824 | 1,838,235 | 18,382,353 |
| 64 | 14,706 | 147,059 | 1,470,588 | 14,705,882 |
| 128 | 8,389 | 83,893 | 838,926 | 8,389,262 |
| 256 | 4,513 | 45,126 | 451,264 | 4,512,635 |
| 512 | 2,345 | 23,452 | 234,522 | 2,345,216 |
| 1024 | 1,196 | 11,962 | 119,617 | 1,196,172 |
| 2048 | 604 | 6,042 | 60,416 | 604,157 |
| 4096 | 304 | 3,036 | 30,362 | 303,619 |

Authors' Addresses

Marius Georgescu
Nara Institute of Science and Technology (NAIST)
Takayama 8916-5
Nara
Japan

Phone: +81 743 72 5216
Email: liviumarius-g@is.naist.jp

BMWG
Internet-Draft
Intended status: Informational
Expires: April 30, 2015

L. Huang, Ed.
R. Gu, Ed.
D. Liu
China Mobile
Bob. Mandeville
Iometrix
Brooks. Hickman
Spirent Communications
Guang. Zhang
IXIA

October 27, 2014

# Benchmarking Methodology for Virtualization Network Performance
## draft-huang-bmwg-virtual-network-performance-00

## Abstract

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

## Status of This Memo

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

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

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

This Internet-Draft will expire on April 30, 2015.

## Copyright Notice

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents



## 1. Introduction

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. Terminology

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’t 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 VM based, it’s quite a different story. Just like the DUT VM, tester in VM shape will have its own performance peak under various conditions. Just like the DUT VM, a VM based tester will have its own performance characteristics.

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’s 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) under the same condition.

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.

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 VM addition.

3. VNIC and its type is needed in the test setup to once again accommodate performance penalty when DUT (VSWITCH) is created.

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

## 4. Key Performance Indicators

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

a) Throughput under various frame sizes: forwarding performance under various frame sizes is a key performance indicator of interest.

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

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

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

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

## 5. Test Setup

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

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

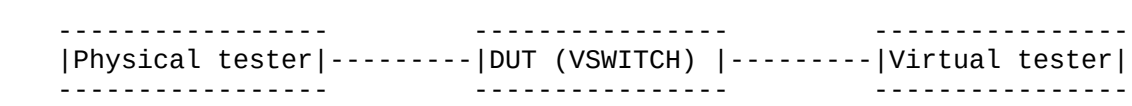



Figure 2: test model A

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

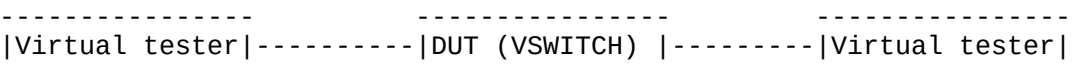



Figure 3: test model B

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) |
|---|---|
| 0 | 0 |
| 128 | 0.46 |
| 256 | 0.84 |
| 512 | 1.56 |
| 1024 | 2.88 |
| 1518 | 4.00 |

Figure 4: test result format

## 6.2. 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.

### 6.2.1. Objectives

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

### 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

6.2.4. Test process

1. Configure the VM 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 CPU load value of server in the condition of shutting down and bypassing the DUT (VSWITCH), respectively.

3. Calculate the increase of the CPU load value due to establishing the DUT (VSWITCH).

6.2.5. Test result format

| Byte | Throughput(GE) | Server CPU(MHZ) | VM CPU(MHz) |
|---|---|---|---|
| 0 | 0 | 515 | 3042 |
| 128 | 0.46 | 6395 | 3040 |
| 256 | 0.84 | 6517 | 3042 |
| 512 | 1.56 | 6668 | 3041 |
| 1024 | 2.88 | 6280 | 3043 |
| 1450 | 4.00 | 6233 | 3045 |

test result format

6.3. 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.

#### 6.3.1. Objectives

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

#### 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

#### 6.3.4. Test process

1. Configure the VM 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 memory consumption value of server in the condition of shutting down and bypassing the DUT (VSWITCH), respectively.

3. Calculate the increase of the memory consumption value due to establishing the DUT (VSWITCH).

#### 6.3.5. Test result format

| Byte | Throughput(GE) | Host Memory | VM Memory | |
|---|---|---|---|---|
| 0 | 0 | 3042 | 696 | |
| 128 | 0.46 | 3040 | 696 | |
| 256 | 0.84 | 3042 | 696 | |
| 512 | 1.56 | 3041 | 696 | |
| 1024 | 2.88 | 3043 | 696 | |
| 1450 | 4.00 | 3045 | 696 | |

test result format

## 6.4. 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.

In the future, we may consider some other ways to have a better synchronization of clock to improve the accuracy of the test.

### 6.4.1. 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.

### 6.4.2. 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 the traffic value of throughput tested in 6.1.

2. Under the same throughput, record the latency value of server in the condition of shutting down and bypassing the DUT (VSWITCH), respectively.

3. Calculate the increase of the latency value due to establishing the DUT (VSWITCH).

#### 6.4.5. Test result format

TBD

## 7. Security Considerations

None.

## 8. IANA Considerations

None.

## 9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

## Authors' Addresses

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

Email: huanglu@chinamobile.com

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

Email: gurong@chinamobile.com

Dapeng Liu
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

Guang Zhang
IXIA

Email: GZhang@ixiacom.com

Benchmarking Working Group
Internet Draft
Intended status: Informational
Expires: September 9, 2015

Sarah Banks
VSS Monitoring
Fernando Calabria
Cisco
Gery Czirjak
Ramdas Machat
Juniper
March 9, 2015

# ISSU Benchmarking Methodology
# draft-ietf-bmwg-issu-meth-00

## Abstract

Modern forwarding devices attempt to minimize any control and data plane disruptions while performing planned software changes, by implementing a technique commonly known as In Service Software Upgrade (ISSU) This document specifies a set of common methodologies and procedures designed to characterize the overall behavior of a Device Under Test (DUT), subject to an ISSU event.

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

Table of Contents

## 1. Introduction

As required by most Service Provider (SP) network operators, ISSU functionality has been implemented by modern forwarding devices to upgrade or downgrade from one software version to another with a goal of eliminating the downtime of the router and/or the outage of service. However, it is noted that while most operators desire complete elimination of downtime, minimization of downtime and service degradation is often the expectation.

The ISSU operation may apply in terms of an atomic version change of the entire system software or it may be applied in a more modular sense such as for a patch or maintenance upgrade. The procedure described herein may be used to verify either approach, as may be supported by the vendor hardware and software.

In support of this document, the desired behavior for an ISSU operation can be summarized as follows:

- The software is successfully migrated, from one version to a successive version or vice versa.

- There are no control plane interruptions throughout the process. That is, the upgrade/downgrade could be accomplished while the device remains "in service". It is noted however, that most service providers will still undertake such actions in a maintenance window (even in redundant environments) to minimize any risk.

- Interruptions to the forwarding plane are minimal to none.

- The total time to accomplish the upgrade is minimized, again to reduce potential network outage exposure (e.g. an external failure

event might impact the network as it operates with reduced redundancy).

This document provides a set of procedures to characterize a given forwarding device's ISSU behavior quantitatively, from the perspective of meeting the above expectations.

Different hardware configurations may be expected to be benchmarked, but a typical configuration for a forwarding device that supports ISSU consists of at least one pair of Routing Processors (RP's) that operate in a redundant fashion, and single or multiple Forwarding Engines (Line Cards) that may or may not be redundant, as well as fabric cards or other components as applicable. This does not preclude the possibility that a device in question can perform ISSU functions through the operation of independent process components, which may be upgraded without impact to the overall operation of the device. As an example, perhaps the software module involved in SNMP functions can be upgraded without impacting other operations.

The concept of a multi-chassis deployment may also be characterized by the current set of proposed methodologies, but the implementation specific details (i.e. process placement and others) are beyond the scope of the current document.

Since most modern forwarding devices, where ISSU would be applicable, do consist of redundant RP's and hardware-separated control plane and data plane functionality, this document will focus on methodologies which would be directly applicable to those platforms. It is anticipated that the concepts and approaches described herein may be readily extended to accommodate other device architectures as well.

## 2. Conventions used in this document

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

In this document, the characters ">>" preceding an indented line(s) indicates a compliance requirement statement using the key words listed above. This convention aids reviewers in quickly identifying or finding the explicit compliance requirements of this RFC.

## 3. Generic ISSU Process, phased approach

ISSU may be viewed as the behavior of a device when exposed to a planned change in its software functionality. This may mean changes to the core operating system, separate processes or daemons or even of firmware logic in programmable hardware devices (e.g. CPLD/FPGA). The goal of an ISSU implementation is to permit such actions with minimal or no disruption to the primary operation of the device in question.

ISSU may be user initiated through direct interaction with the device or activated through some automated process on a management system or even on the device itself. For the purposes of this document, we will focus on the model where the ISSU action is initiated by direct user intervention.

The ISSU process can be viewed as a series of different phases or activities, as defined below. For each of these phases, the test operator MUST record the outcome as well as any relevant observations (defined further in the present document). Note that, a given vendor implementation may or may not permit the abortion of the in-progress ISSU at particular stages. There may also be certain restrictions as to ISSU availability given certain functional configurations (for example, ISSU in the presence of Bidirectional Failure Detection (BFD) [RFC 5880] may not be supported). It is incumbent upon the test operator to ensure that the DUT is appropriately configured to provide the appropriate test environment. As with any properly orchestrated test effort, the test plan document should reflect these and other relevant details and SHOULD be written with close attention to the expected production-operating environment. The combined analysis of the results of each phase will characterize the overall ISSU process with the main goal of being able to identify and quantify any disruption in service (from the data and control plane perspective) allowing operators to plan their maintenance activities with greater precision.

### 3.1. Software Download

In this first phase, the requested software package may be downloaded to the router and is typically stored onto a device. The downloading of software may be performed automatically by the device as part of the upgrade process, or it may be initiated separately. Such separation allows an administrator to download the new code inside or outside of a maintenance window; it is anticipated that downloading new code and saving it to disk on the router will not impact operations. In the case where the software can be downloaded outside of the actual upgrade process, the administrator SHOULD do

so; downloading software can skew timing results based on factors that are often not comparative in nature. Internal compatibility verification may be performed by the software running on the DUT, to verify the checksum of the files downloaded as well as any other pertinent checks. Depending upon vendor implementation, these mechanisms may extend to include verification that the downloaded module(s) meet a set of identified pre-requisites such as hardware or firmware compatibility or minimum software requirements. Where such mechanisms are made available by the product, they should be verified, by the tester, with the perspective of avoiding operational issues in production. Verification should include both positive verification (ensuring that an ISSU action should be permitted) as well as negative tests (creation of scenarios where the verification mechanisms would report exceptions).

### 3.2. Software Staging

In this second phase, the requested software package is loaded in the pertinent components of a given forwarding device (typically the RP in standby state). Internal compatibility verification may be performed by the software running on the DUT, as part of the upgrade process itself, to verify the checksum of the files downloaded as well as any other pertinent checks. Depending upon vendor implementation, these mechanisms may extend to include verification that the downloaded module(s) meet a set of identified pre-requisites such as hardware or firmware compatibility or minimum software requirements. Where such mechanisms are made available by the product, they should be verified, by the tester (again with the perspective of avoiding operational issues in production). In this case, the execution of these checks is within scope of the upgrade time, and SHOULD be included in the testing results. Once the new software is downloaded to the pertinent components of the DUT, the upgrade begins and the DUT begins to prepare itself for upgrade. Depending on the vendor implementation, it is expected that redundant hardware pieces within the DUT are upgraded, including the backup or secondary RP.

### 3.3. Upgrade Run

In this phase, a switchover of RPs may take place, where one RP is now upgraded with the new version of software. More importantly, the "Upgrade Run" phase is where the internal changes made to information and state stored on the router, on disk and in memory, are either migrated to the "new" version of code, or transformed/rebuilt to meet the standards of the new version of code, and pushed onto the appropriate pieces of hardware. It is within this phase that any outage(s) on the control or forwarding

plane may be expected to be observed. This is the critical phase of the ISSU, where the control plane should not be impacted and any interruptions to the forwarding plane should be minimal to none. For some implementations, the above two steps may be concatenated into one monolithic operation. In such case, the calculation of the respective ISSU time intervals may need to be adapted accordingly.

If any control or data plane interruptions are observed within this stage, they should be recorded as part of the results document.

### 3.4. Upgrade Acceptance

In this phase, the new version of software MUST be running in all the physical nodes of the logical forwarding device. (RP's and LC's as applicable). At this point, configuration control is returned to the operator and normal device operation i.e. outside of ISSU-oriented operation, is resumed.

## 4. Test Methodology

As stated by https://tools.ietf.org/html/rfc6815, the Test Topology Setup must be part of an ITE (Isolated Test Environment)

The reporting of results MUST take into account the repeatability considerations from Section 4 of [RFC2544]. It is RECOMMENDED to perform multiple trials and report average results. The results are reported in a simple statement including the measured frame loss and ISSU impact times.

### 4.1. Test Topology

The hardware configuration of the DUT (Device Under test) SHOULD be identical to the one expected to be or currently deployed in production in order for the benchmark to have relevance. This would include the number of RP's, hardware version, memory and initial software release, any common chassis components, such as fabric hardware in the case of a fabric-switching platform and the specific LC's (version, memory, interfaces type, rate etc.)

For the Control and Data plane, differing configuration approached MAY be utilized. The recommended approach relies on "mimicking" the existing production data and control plane information, in order to emulate all the necessary Layer1 through Layer3 communications and, if appropriate, the upper layer characteristics of the network, as well as end to end traffic/communication pairs. In other words,

design a representative load model of the production environment and deploy a collapsed topology utilizing test tools and/or external devices, where the DUT will be tested. Note that, the negative impact of ISSU operations is likely to impact scaled, dynamic topologies to a greater extent than simpler, static environments. As such, this methodology (based upon production environments) is advised for most test scenarios.

The second, more simplistic approach is to deploy an ITE "Isolated Testing Environment" as described in some of the existing standards for benchmarking methodologies (e.g. RFC2544/RFC6815) in which end-points are "directly" connected to the DUT. In this manner control plane information is kept to a minimum (only connected interfaces) and only a basic data plane of sources and destinations is applied. If this methodology is selected, care must be taken to understand that the systemic behavior of the ITE may not be identical to that experienced by a device in a production network role. That is, control plane validation may be minimal to none if this methodology.

## 4.2. Load Model

In consideration of the defined test topology, a load model must be developed to exercise the DUT while the ISSU event is introduced. This applied load should be defined in such a manner as to provide a granular, repeatable verification of the ISSU impact on transit traffic. Sufficient traffic load (rate) should be applied to permit timing extrapolations at a minimum granularity of 100 milliseconds e.g. 100Mbps for a 10Gbps interface. The use of steady traffic streams rather than bursty loads is preferred to simplify analysis.

The traffic should be patterned to provide a broad range of source and destination pairs, which resolve to a variety of FIB (forwarding information base) prefix lengths. If the production network environment includes multicast traffic or VPN’s (L2, L3 or IPSec) it is critical to include these in the model.

For mixed protocol environments (e.g. IPv4 and IPv6), frames SHOULD be distributed between the different protocols. The distribution SHOULD approximate the network conditions of deployment. In all cases, the details of the mixed protocol distribution MUST be included in the reporting.

The feature, protocol timing and other relevant configurations should be matched to the expected production environment. Deviations from the production templates may be deemed necessary by the test operator (for example, certain features may not support ISSU or the test bed may not be able to accommodate such). However, the impact

of any such divergence should be clearly understood and the differences MUST be recorded in the results documentation. It is recommended that an NMS system be deployed, preferably similar to that utilized in production. This will allow for monitoring of the DUT while it is being tested both in terms of supporting the system resource impact analysis as well as from the perspective of detecting interference with non-transit (management) traffic as a result of the ISSU operation. Additionally, a DUT management session other than snmp-based, typical of usage in production, should be established to the DUT and monitored for any disruption. It is suggested that the actual test exercise be managed utilizing direct console access to the DUT, if at all possible to avoid the possibility that a network interruption impairs execution of the test exercise.

All in all, the load model should attempt to simulate the production network environment to the greatest extent possible in order to maximize the applicability of the results generated.

## 5. ISSU Test Methodology

As previously described, for the purposes of this test document, the ISSU process is divided into three main phases. The following methodology assumes that a suitable test topology has been constructed per section 4. A description of the methodology to be applied for each of the above phases follows:

### 5.1. Pre-ISSU recommended verifications

1. Verify that enough hardware and software resources are available to complete the Load operation (enough disk space).

2. Verify that the redundancy states between RPs and other nodes are as expected (e.g. redundancy on, RP’s synchronized).

3. Verify that the device, if running NSR capable routing protocols, is in a "ready" state; that is, that the sync between RPs is complete and the system is ready for failover, if necessary.

4. Gather a configuration snapshot of the device and all of its applicable components.

5. Verify that the node is operating in a "steady" state (that is, no critical or maintenance function is being currently performed).

6. Note any other operational characteristics that the tester may deem applicable to the specific implementation deployed.

5.2. Software Staging

1. Establish all relevant protocol adjacencies and stabilize routing within the test topology. In particular, ensure that the scaled levels of the dynamic protocols are dimensioned as specified by the test topology plan.

2. Clear relevant logs and interface counters to simplify analysis. If possible, set logging timestamps to a highly granular mode. If the topology includes management systems, ensure that the appropriate polling levels have been applied, sessions established and that the responses are per expectation.

3. Apply the traffic loads as specified in the load model previously developed for this exercise.

4. Document an operational baseline for the test bed with relevant data supporting the above steps (include all relevant load characteristics of interest in the topology e.g. routing load, traffic volumes, memory and CPU utilization)

5. Note the start time (T0) and begin the code change process utilizing the appropriate mechanisms as expected to be used in production (e.g. active download with TFTP/FTP/SCP/etc. or direct install from local or external storage facility). In order to ensure that ISSU process timings are not skewed by the lack of a network wide synchronization source, the use of a network NTP source is encouraged.

6. Take note of any logging information and command line interface (CLI) prompts as needed (this detail will be vendor-specific) Respond to any DUT prompts in a timely manner.

7. Monitor the DUT for the reload of secondary RP to the new software level. Once the secondary has stabilized on the new code, note the completion time. The duration of these steps will be recorded as "T1".

8. Review system logs for any anomalies, check that relevant dynamic protocols have remained stable and note traffic loss if any. Verify

that deployed management systems have not identified any unexpected behavior.

### 5.3. Upgrade Run

The following assumes that the software load step and upgrade step are discretely controllable. If not, maintain the afore-mentioned timer and monitor for completion of the ISSU as described below.

1. Note the start time and initiate the actual upgrade procedure

2. Monitor the operation of the secondary route processor while it initializes with the new software and assumes mastership of the DUT. At this point, pay particular attention to any indications of control plane disruption, traffic impact or other anomalous behavior. Once the DUT has converged upon the new code and returned to normal operation note the completion time and log the duration of this step as T2.

3. Review the syslog data in the DUT and neighboring devices for any behavior, which would be disruptive in a production environment (linecard reloads, control plane flaps etc.). Examine the traffic generators for any indication of traffic loss over this interval. If the Test Set reported any traffic loss, note the number of frames lost as "TP_frames". If the test set also provides outage duration, note this as TP_time (alternatively this may be calculated as TP/offered pps (packets per second) load).

4. Verify the DUT status observations as per any NMS systems managing the DUT and its neighboring devices. Document the observed CPU and memory statistics both during the ISSU upgrade event and after and ensure that memory and CPU have returned to an expected (previously baselined) level.

### 5.4. Post ISSU verification

The following describes a set of post-ISSU verification tasks that are not directly part of the ISSU process, but are recommended for execution in order to validate a successful upgrade:

1. Configuration delta analysis

   Examine the post-ISSU configurations to determine if any changes have occurred either through process error or due to differences in the implementation of the upgraded code.

2. Exhaustive control plane analysis

Review the details of the RIB and FIB to assess whether any unexpected changes have been introduced in the forwarding paths.

3. Verify that both RPs are up and that the redundancy mechanism for the control plane is enabled and fully synchronized.

4. Verify that no control plane (protocol) events or flaps were detected.

5. Verify that no L1 and or L2 interface flaps were observed.

6. Document the hitless operation or presence of an outage based upon the counter values provided by the Test Set.

### 5.5. ISSU under negative stimuli

As an OPTIONAL Test Case, the operator may want to perform an ISSU test while the DUT is under stress by introducing route churn to any or all of the involved phases of the ISSU process.

One approach relies on the operator to gather statistical information from the production environment and determine a specific number of routes to flap every 'fixed' or 'variable' interval. Alternatively, the operator may wish to simply pre-select a fixed number of prefixes to flap. As an example, an operator may decide to flap 1% of all the BGP routes every minute and restore them 1 minute afterwards. The tester may wish to apply this negative stimulus throughout the entire ISSU process or most importantly, during the run phase. It is important to ensure that these routes, which are introduced solely for stress proposes, must not overlap the ones (per the Load Model) specifically leveraged to calculate the TP (recorded outage). Furthermore, there SHOULD NOT be 'operator induced' control plane - protocol adjacency flaps for the duration of the test process as it may adversely affect the characterization of the entire test exercise. For example, triggering IGP adjacency events may force re-computation of underlying routing tables with attendant impact to the perceived ISSU timings. While not recommended, if such trigger events are desired by the test operator, care should be taken to avoid the introduction of unexpected anomalies within the test harness.

## 6. ISSU Abort and Rollback

Where a vendor provides such support, the ISSU process could be aborted for any reason by the operator. However, the end results and behavior may depend on the specific phase where the process was aborted. While this is implementation dependent, as a general

recommendation, if the process is aborted during the "Software Download" or "Software Staging" phases, no impact to service or device functionality should be observed. In contrast, if the process is aborted during the "Upgrade Run" or "Upgrade Accept" phases, the system may reload and revert back to the previous software release and as such, this operation may be service affecting. Where vendor support is available, the abort/rollback functionality should be verified and the impact, if any, quantified generally following the procedures provided above.

## 7. Final Report - Data Presentation - Analysis

All ISSU impact results are summarized in a simple statement describing the "ISSU Disruption Impact" including the measured frame loss and impact time, where impact time is defined as the time frame determined per the TP reported outage. These are considered to be the primary data points of interest.

However, the entire ISSU operational impact should also be considered in support of planning for maintenance and as such additional reporting points are included.

| | |
|---|---|
| Software download/secondary update | T1 |
| Upgrade/Run | T2 |
| ISSU Traffic Disruption (Frame Loss) | TP_frames |
| ISSU Traffic Impact Time (milliseconds) | TP Time |
| ISSU Housekeeping Interval | T |

(Time for both RP’s up on new code and fully synced - Redundancy restored)

| | |
|---|---|
| Total ISSU Maintenance Window | T4 (sum of T1+T2+T3) |

The results reporting MUST provide the following information:

DUT hardware and software detail

Test Topology definition and diagram (especially as related to the ISSU operation)

Load Model description including protocol mixes and any divergence from the production environment

Time Results as per above

Anomalies Observed during ISSU

Anomalies Observed in post-ISSU analysis

It is RECOMMENDED that the following parameters be reported in these units:

| Parameter | Units or Examples |
|---|---|
| Traffic Load | Frames per second and bits per Second |
| Disruption (average) | Frames |
| Impact Time (average) | Milliseconds |
| Number of trials | Integer count |
| Protocols | IPv4, IPv6, MPLS, etc. |
| Frame Size | Octets |
| Port Media | Ethernet, Gigabit Ethernet (GbE), Packet over SONET (POS), etc. |
| Port Speed | 10 Gbps, 1 Gbps, 100 Mbps, etc. |
| Interface Encap. | Ethernet, Ethernet VLAN, PPP, High-Level Data Link Control(HDLC),etc. |
| Number of Prefixes | Integer count |
| flapped (ON Interval) | (Optional # of prefixes / Time (minutes) |
| flapped (OFF Interval) | (Optional # of prefixes / Time (minutes) |

Document any configuration deltas, which are observed after the ISSU upgrade has taken effect. Note differences, which are driven by changes in the patch or release level as well as items, which

are aberrant changes due to software faults. In either of these cases, any unexpected behavioral changes should be analyzed and a determination made as to the impact of the change (be it functional variances or operational impacts to existing scripts or management mechanisms.

### 7.1. Data collection considerations

When a DUT is undergoing an ISSU operation, it’s worth noting that the DUT’s data collection and reporting of data, such as counters, interface statistics, log messages, etc., might not be accurate. As such, one SHOULD NOT rely on the DUTs data collection methods, but rather, SHOULD use the test tools and equipment to collect data used for reporting in Section 7. Care and consideration should be paid in testing or adding new test cases, such that the desired data can be collected from the test tools themselves, or other external equipment, outside of the DUT itself.

## 8. Security Considerations

This Applicability Statement intends to help preserve the security of the Internet by clarifying that the scope of this document and other BMWG memos are all limited to testing in a laboratory ITE, thus avoiding accidental Denial-of-Service attacks or congestion due to high traffic volume test streams. All benchmarking activities 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.

Further, benchmarking is performed on a "black-box" basis, relying solely on measurements observable external to the device under test/ system under test (DUT/SUT).

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

## 9. IANA Considerations

There are no IANA actions required by this memo.

## 10. References

### 10.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2234] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

### 10.2. Informative References

[3] Faber, T., Touch, J. and W. Yue, "The TIME-WAIT state in TCP and Its Effect on Busy Servers", Proc. Infocom 1999 pp. 1573-1583.

[Fab1999] Faber, T., Touch, J. and W. Yue, "The TIME-WAIT state in TCP and Its Effect on Busy Servers", Proc. Infocom 1999 pp. 1573-1583.

## 11. Acknowledgments

The authors wish to thanks Vibin Thomas for his valued review and feedback.

Authors' Addresses

Sarah Banks
VSS Monitoring
Email: sbanks@encrypted.net

Fernando Calabria
Cisco Systems
Email: fcalabri@cisco.com

Gery Czirjak
Juniper Networks
Email: gczirjak@juniper.net

Ramdas Machat
Juniper Networks
Email: rmachat@juniper.net

Network Working Group
Internet Draft
Intended status: Informational
Expires: August 2015
February 23, 2015

B. Constantine
JDSU
R. Krishnan
Brocade Communications

# Traffic Management Benchmarking
## draft-ietf-bmwg-traffic-management-03.txt

Status of this Memo

Abstract

This framework describes a practical methodology for benchmarking the traffic management capabilities of networking devices (i.e. policing, shaping, etc.). The goal is to provide a repeatable test method that objectively compares performance of the device's traffic management capabilities and to specify the means to benchmark traffic management with representative application traffic.

Table of Contents

# 1. Introduction

Traffic management (i.e. policing, shaping, etc.) is an increasingly important component when implementing network Quality of Service (QoS).

There is currently no framework to benchmark these features although some standards address specific areas which are described in Section 1.1.

This draft provides a framework to conduct repeatable traffic management benchmarks for devices and systems in a lab environment.

Specifically, this framework defines the methods to characterize the capacity of the following traffic management features in network devices; classification, policing, queuing / scheduling, and traffic shaping.

This benchmarking framework can also be used as a test procedure to assist in the tuning of traffic management parameters before service activation. In addition to Layer 2/3 (Ethernet / IP) benchmarking, Layer 4 (TCP) test patterns are proposed by this draft in order to more realistically benchmark end-user traffic.

## 1.1. Traffic Management Overview

In general, a device with traffic management capabilities performs the following functions:

- Traffic classification: identifies traffic according to various configuration rules (for example IEEE 802.1Q Virtual LAN (VLAN), Differential Services Code Point (DSCP) etc.) and marks this traffic internally to the network device. Multiple external priorities (DSCP, 802.1p, etc.) can map to the same priority in the device.
- Traffic policing: limits the rate of traffic that enters a network device according to the traffic classification. If the traffic exceeds the provisioned limits, the traffic is either dropped or remarked and forwarded onto to the next network device
- Traffic Scheduling: provides traffic classification within the network device by directing packets to various types of queues and applies a dispatching algorithm to assign the forwarding sequence of packets
- Traffic shaping: a traffic control technique that actively buffers and smooths the output rate in an attempt to adapt bursty traffic to the configured limits
- Active Queue Management (AQM):
  AQM involves monitoring the status of internal queues and proactively dropping (or remarking) packets, which causes hosts using congestion-aware protocols to back-off and in turn alleviate queue congestion [AQM-RECO]. On the other hand, classic traffic management techniques reactively drop (or remark) packets based on queue full condition. The benchmarking scenarios for AQM are different and is outside of the scope of this testing framework.

The following diagram is a generic model of the traffic management capabilities within a network device. It is not intended to represent all variations of manufacturer traffic management capabilities, but provide context to this test framework.

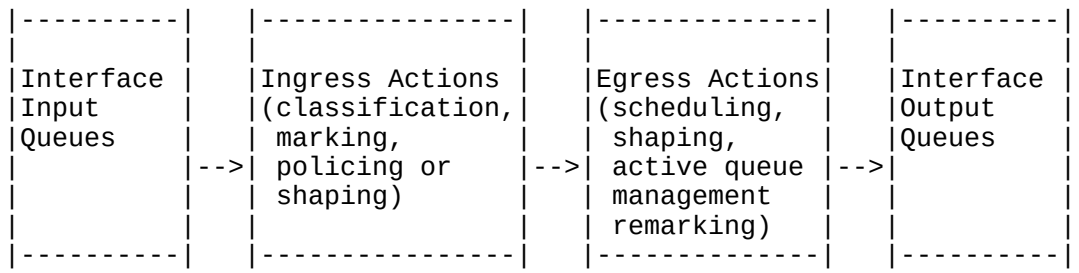


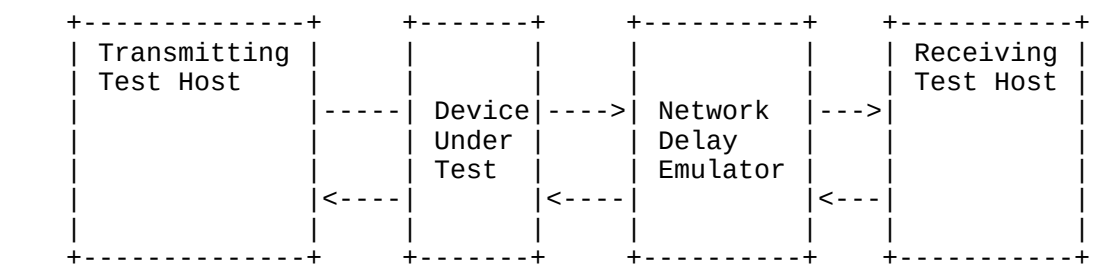

Figure 1: Generic Traffic Management capabilities of a Network Device

Ingress actions such as classification are defined in [RFC4689] and include IP addresses, port numbers, DSCP, etc. In terms of marking, [RFC2697] and [RFC2698] define a single rate and dual rate, three color marker, respectively.

The Metro Ethernet Forum (MEF) specifies policing and shaping in terms of Ingress and Egress Subscriber/Provider Conditioning Functions in MEF12.1 [MEF-12.1]; Ingress and Bandwidth Profile attributes in MEF10.2 [MEF-10.2] and MEF 26 [MEF-26].

## 1.2 Lab Configuration and Testing Overview

The following is the description of the lab set-up for the traffic management tests:




As shown in the test diagram, the framework supports uni-directional and bi-directional traffic management tests (where the transmitting and receiving roles would be reversed on the return path).

This testing framework describes the tests and metrics for each of the following traffic management functions:
- Policing
- Queuing / Scheduling
- Shaping

The tests are divided into individual and rated capacity tests. The individual tests are intended to benchmark the traffic management functions according to the metrics defined in Section 4. The capacity tests verify traffic management functions under the load of many simultaneous individual tests and their flows.

This involves concurrent testing of multiple interfaces with the specific traffic management function enabled, and increasing load to the capacity limit of each interface.

As an example: a device is specified to be capable of shaping on all of its egress ports. The individual test would first be conducted to benchmark the specified shaping function against the metrics defined in section 4. Then the capacity test would be executed to test the shaping function concurrently on all interfaces and with maximum traffic load.

The Network Delay Emulator (NDE) is required for TCP stateful tests in order to allow TCP to utilize a significant size TCP window in its control loop.

Also note that the Network Delay Emulator (NDE) SHOULD be passive in nature such as a fiber spool. This is recommended to eliminate the potential effects that an active delay element (i.e. test impairment generator) may have on the test flows. In the case where a fiber spool is not practical due to the desired latency, an active NDE MUST be independently verified to be capable of adding the configured delay without loss. In other words, the DUT would be removed and the NDE performance benchmarked independently.

Note the NDE SHOULD be used in "full pipe" delay mode. Most NDEs allow for per flow delay actions, emulating QoS prioritization. For this framework, the NDE’s sole purpose is simply to add delay to all packets (emulate network latency). So to benchmark the performance of the NDE, maximum offered load should be tested against the following frame sizes: 128, 256, 512, 768, 1024, 1500,and 9600 bytes. The delay accuracy at each of these packet sizes can then be used to calibrate the range of expected Bandwidth Delay Product (BDP) for the TCP stateful tests.

## 2. Conventions used in this document

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

The following acronyms are used:

AQM: Active Queue Management

BB: Bottleneck Bandwidth

BDP: Bandwidth Delay Product

BSA: Burst Size Achieved

CBS: Committed Burst Size

CIR: Committed Information Rate

DUT: Device Under Test

EBS: Excess Burst Size

EIR: Excess Information Rate

NDE: Network Delay Emulator

SP: Strict Priority Queuing

QL: Queue Length

QoS: Quality of Service

RTH: Receiving Test Host

RTT: Round Trip Time

SBB: Shaper Burst Bytes

SBI: Shaper Burst Interval

SR: Shaper Rate

SSB: Send Socket Buffer

Tc: CBS Time Interval

Te: EBS Time Interval

Ti Transmission Interval

TTH: Transmitting Test Host

TTP: TCP Test Pattern

TTPET: TCP Test Pattern Execution Time

## 3. Scope and Goals

The scope of this work is to develop a framework for benchmarking and testing the traffic management capabilities of network devices in the lab environment. These network devices may include but are not limited to:

- Switches (including Layer 2/3 devices)
- Routers
- Firewalls
- General Layer 4-7 appliances (Proxies, WAN Accelerators, etc.)

Essentially, any network device that performs traffic management as defined in section 1.1 can be benchmarked or tested with this framework.

The primary goal is to assess the maximum forwarding performance deemed to be within the provisioned traffic limits that a network device can sustain without dropping or impairing packets, or compromising the accuracy of multiple instances of traffic management functions. This is the benchmark for comparison between devices.

Within this framework, the metrics are defined for each traffic management test but do not include pass / fail criterion, which is not within the charter of BMWG. This framework provides the test methods and metrics to conduct repeatable testing, which will provide the means to compare measured performance between DUTs.

As mentioned in section 1.2, these methods describe the individual tests and metrics for several management functions. It is also within scope that this framework will benchmark each function in terms of overall rated capacity. This involves concurrent testing of multiple interfaces with the specific traffic management function enabled, up to the capacity limit of each interface.

It is not within scope of this of this framework to specify the procedure for testing multiple configurations of traffic management functions concurrently. The multitudes of possible combinations is almost unbounded and the ability to identify functional "break points" would be almost impossible.

However, section 6.4 provides suggestions for some profiles of concurrent functions that would be useful to benchmark. The key requirement for any concurrent test function is that tests MUST produce reliable and repeatable results.

Also, it is not within scope to perform conformance testing. Tests defined in this framework benchmark the traffic management functions according to the metrics defined in section 4 and do not address any conformance to standards related to traffic management. The current specifications don't specify exact behavior or implementation and the specifications that do exist (cited in section 1.1) allow implementations to vary w.r.t. short term rate accuracy and other factors. This is a primary driver for this framework: to provide an objective means to compare vendor traffic management functions.

Another goal is to devise methods that utilize flows with congestion-aware transport (TCP) as part of the traffic load and still produce repeatable results in the isolated test environment. This framework will derive stateful test patterns (TCP or application layer) that can also be used to further benchmark the performance of applicable traffic management techniques such as queuing / scheduling and traffic shaping. In cases where the network device is stateful in nature (i.e. firewall, etc.), stateful test pattern traffic is important to test along with stateless, UDP traffic in specific test scenarios (i.e. applications using TCP transport and UDP VoIP, etc.)

As mentioned earlier in the document, repeatability of test results is critical, especially considering the nature of stateful TCP traffic. To this end, the stateful tests will use TCP test patterns to emulate applications. This framework also provides guidelines for application modeling and open source tools to achieve the repeatable stimulus. And finally, TCP metrics from [RFC6349] MUST be measured for each stateful test and provide the means to compare each repeated test.

## 4. Traffic Benchmarking Metrics

The metrics to be measured during the benchmarks are divided into two (2) sections: packet layer metrics used for the stateless traffic testing and TCP layer metrics used for the stateful traffic testing.

### 4.1. Metrics for Stateless Traffic Tests

Stateless traffic measurements require that sequence number and time-stamp be inserted into the payload for lost packet analysis. Delay analysis may be achieved by insertion of timestamps directly into the packets or timestamps stored elsewhere (packet captures). This framework does not specify the packet format to carry sequence number or timing information.

However,[RFC4737] and [RFC4689] provide recommendations for sequence tracking along with definitions of in-sequence and out-of-order packets.

The following are the metrics that MUST be measured during the stateless traffic benchmarking components of the tests:

- Burst Size Achieved (BSA): for the traffic policing and network queue tests, the tester will be configured to send bursts to test either the Committed Burst Size (CBS) or Excess Burst Size (EBS) of a policer or the queue / buffer size configured in the DUT. The Burst Size Achieved metric is a measure of the actual burst size received at the egress port of the DUT with no lost packets. As an example, the configured CBS of a DUT is 64KB and after the burst test, only a 63 KB can be achieved without packet loss. Then 63KB is the BSA. Also, the average Packet Delay Variation (PDV see below) as experienced by the packets sent at the BSA burst size should be recorded. This metric shall be reported in units of bytes, KBytes, or MBytes.

- Lost Packets(LP): For all traffic management tests, the tester will transmit the test packets into the DUT ingress port and the number of packets received at the egress port will be measured. The difference between packets transmitted into the ingress port and received at the egress port is the number of lost packets as measured at the egress port. These packets must have unique identifiers such that only the test packets are measured. For cases where multiple flows are transmitted from ingress to egress port (e.g. IP conversations), each flow must have sequence numbers within the test packets stream.

[RFC6703] and [RFC2680] describe the need to establish the time threshold to wait before a packet is declared as lost, and this threshold MUST be reported with the results. This metric shall be reported as an integer number which cannot be negative. (see: http://tools.ietf.org/html/rfc6703#section-4.1)

- Out of Order (OOO): in additions to the LP metric, the test packets must be monitored for sequence. [RFC4689] defines the general function of sequence tracking, as well as definitions for in-sequence and out-of-order packets. Out-of-order packets will be counted per [RFC4737]. This metric shall be reported as an integer number which cannot be negative.

- Packet Delay (PD): the Packet Delay metric is the difference between the timestamp of the received egress port packets and the packets transmitted into the ingress port and specified in [RFC1242]. The transmitting host and receiving host time must be in time sync using NTP , GPS, etc. This metric SHALL be reported as a real number of seconds, where a negative measurement usually indicates a time synchronization problem between test devices.

- Packet Delay Variation (PDV): the Packet Delay Variation metric is the variation between the timestamp of the received egress port packets and specified in [RFC5481]. Note that per [RFC5481], this PDV is the variation of one-way delay across many packets in the traffic flow. Per the measurement formula in [RFC5481], select the high percentile of 99% and units of measure will be a real number of seconds (negative is not possible for PDV and would indicate a measurement error).

- Shaper Rate (SR): The SR represents the average DUT output rate (bps) over the test interval. The Shaper Rate is only applicable to the traffic shaping tests.

- Shaper Burst Bytes (SBB): A traffic shaper will emit packets in different size "trains"; these are frames "back-to-back", respect the mandatory inter-frame gap. This metric characterizes the method by which the shaper emits traffic. Some shapers transmit larger bursts per interval, and a burst of 1 packet would apply to the extreme case of a shaper sending a CBR stream of single packets. This metric SHALL be reported in units of bytes, KBytes, or MBytes. Shaper Burst Bytes is only applicable to thetraffic shaping tests.

- Shaper Burst Interval(SBI): the SBI is the time between shaper emitted bursts and is measured at the DUT egress port. This metric shall be reported as an real number of seconds. Shaper Burst Interval is only applicable to the traffic shaping tests,

## 4.2. Metrics for Stateful Traffic Tests

The stateful metrics will be based on [RFC6349] TCP metrics and MUST include:

- TCP Test Pattern Execution Time (TTPET): [RFC6349] defined the TCP Transfer Time for bulk transfers, which is simply the measured time to transfer bytes across single or concurrent TCP connections. The TCP test patterns used in traffic management tests will include bulk transfer and interactive applications. The interactive patterns include instances such as HTTP business applications, database applications, etc. The TTPET will be the measure of the time for a single execution of a TCP Test Pattern (TTP). Average, minimum, and maximum times will be measured or calculated and expressed as a real number of seconds.

An example would be an interactive HTTP TTP session which should take 5 seconds on a GigE network with 0.5 millisecond latency. During ten (10) executions of this TTP, the TTPET results might be: average of 6.5 seconds, minimum of 5.0 seconds, and maximum of 7.9 seconds.

- TCP Efficiency: after the execution of the TCP Test Pattern, TCP Efficiency represents the percentage of Bytes that were not retransmitted.

$$\text{TCP Efficiency \%} = \frac{\text{Transmitted Bytes} - \text{Retransmitted Bytes}}{\text{Transmitted Bytes}} \times 100$$

Transmitted Bytes are the total number of TCP Bytes to be transmitted including the original and the retransmitted Bytes. These retransmitted bytes should be recorded from the sender’s TCP/IP stack

perspective, to avoid any misinterpretation that a reordered packet is a retransmitted packet (as may be the case with packet decode interpretation).

- Buffer Delay: represents the increase in RTT during a TCP test versus the baseline DUT RTT (non congested, inherent latency). RTT and the technique to measure RTT (average versus baseline) are defined in [RFC6349]. Referencing [RFC6349], the average RTT is derived from the total of all measured RTTs during the actual test sampled at every second divided by the test duration in seconds.

$$\text{Average RTT during transfer} = \frac{\text{Total RTTs during transfer}}{\text{Transfer duration in seconds}}$$

$$\text{Buffer Delay \%} = \frac{\text{Average RTT during Transfer} - \text{Baseline RTT}}{\text{Baseline RTT}} \text{ X } 100$$

Note that even though this was not explicitly stated in [RFC6349], retransmitted packets should not be used in RTT measurements.

Also, the test results should record the average RTT in millisecond across the entire test duration and number of samples.

## 5. Tester Capabilities

The testing capabilities of the traffic management test environment are divided into two (2) sections: stateless traffic testing and stateful traffic testing

### 5.1. Stateless Test Traffic Generation

The test device MUST be capable of generating traffic at up to the link speed of the DUT. The test device must be calibrated to verify that it will not drop any packets. The test device’s inherent PD and PDV must also be calibrated and subtracted from the PD and PDV metrics. The test device must support the encapsulation to be tested such as IEEE 802.1Q VLAN, IEEE 802.1ad Q-in-Q, Multiprotocol Label Switching (MPLS), etc. Also, the test device must allow control of the classification techniques defined in [RFC4689] (i.e. IP address, DSCP, TOS, etc classification).

The open source tool "iperf" can be used to generate stateless UDP traffic and is discussed in Appendix A. Since iperf is a software based tool, there will be performance limitations at higher link speeds (e.g. GigE, 10 GigE, etc.). Careful calibration of any test environment using iperf is important. At higher link speeds, it is recommended to use hardware based packet test equipment.

#### 5.1.1 Burst Hunt with Stateless Traffic

A central theme for the traffic management tests is to benchmark the specified burst parameter of traffic management function, since burst

parameters of SLAs are specified in bytes. For testing efficiency, it is recommended to include a burst hunt feature, which automates the manual process of determining the maximum burst size which can be supported by a traffic management function.

The burst hunt algorithm should start at the target burst size (maximum burst size supported by the traffic management function) and will send single bursts until it can determine the largest burst that can pass without loss. If the target burst size passes, then the test is complete. The hunt aspect occurs when the target burst size is not achieved; the algorithm will drop down to a configured minimum burst size and incrementally increase the burst until the maximum burst supported by the DUT is discovered. The recommended granularity of the incremental burst size increase is 1 KB.

Optionally for a policer function and if the burst size passes, the burst should be increased by increments of 1 KB to verify that the policer is truly configured properly (or enabled at all).

## 5.2. Stateful Test Pattern Generation

The TCP test host will have many of the same attributes as the TCP test host defined in [RFC6349]. The TCP test device may be a standard computer or a dedicated communications test instrument. In both cases, it must be capable of emulating both a client and a server.

For any test using stateful TCP test traffic, the Network Delay Emulator (NDE function from the lab set-up diagram) must be used in order to provide a meaningful BDP. As referenced in section 2, the target traffic rate and configured RTT MUST be verified independently using just the NDE for all stateful tests (to ensure the NDE can delay without loss).

The TCP test host MUST be capable to generate and receive stateful TCP test traffic at the full link speed of the DUT. As a general rule of thumb, testing TCP Throughput at rates greater than 500 Mbps may require high performance server hardware or dedicated hardware based test tools.

The TCP test host MUST allow adjusting both Send and Receive Socket Buffer sizes. The Socket Buffers must be large enough to fill the BDP for bulk transfer TCP test application traffic.

Measuring RTT and retransmissions per connection will generally require a dedicated communications test instrument. In the absence of dedicated hardware based test tools, these measurements may need to be conducted with packet capture tools, i.e. conduct TCP Throughput tests and analyze RTT and retransmissions in packet captures.

The TCP implementation used by the test host MUST be specified in the test results (e.g. TCP New Reno, TCP options supported, etc.). Additionally, the test results SHALL provide specific congestion control algorithm details, as per [RFC3148].

While [RFC6349] defined the means to conduct throughput tests of TCP bulk transfers, the traffic management framework will extend TCP test execution into interactive TCP application traffic. Examples include email, HTTP, business applications, etc. This interactive traffic is bi-directional and can be chatty, meaning many turns in traffic communication during the course of a transaction (versus the relatively uni-directional flow of bulk transfer applications).

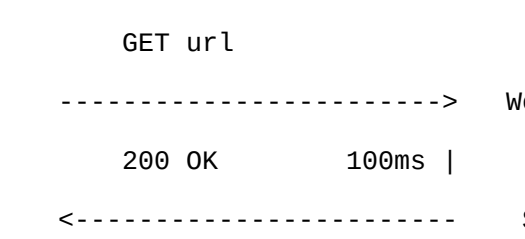
The test device must not only support bulk TCP transfer application traffic but MUST also support chatty traffic. A valid stress test SHOULD include both traffic types. This is due to the non-uniform, bursty nature of chatty applications versus the relatively uniform nature of bulk transfers (the bulk transfer smoothly stabilizes to equilibrium state under lossless conditions).

While iperf is an excellent choice for TCP bulk transfer testing, the netperf open source tool provides the ability to control the client and server request / response behavior. The netperf-wrapper tool is a Python wrapper to run multiple simultaneous netperf instances and aggregate the results. Appendix A provides an overview of netperf / netperf-wrapper and another open source application emulation tools, iperf. As with any software based tool, the performance must be qualified to the link speed to be tested. Hardware-based test equipment should be considered for reliable results at higher links speeds (e.g. 1 GigE, 10 GigE).

### 5.2.1. TCP Test Pattern Definitions

As mentioned in the goals of this framework, techniques are defined to specify TCP traffic test patterns to benchmark traffic management technique(s) and produce repeatable results. Some network devices such as firewalls, will not process stateless test traffic which is another reason why stateful TCP test traffic must be used.

An application could be fully emulated up to Layer 7, however this framework proposes that stateful TCP test patterns be used in order to provide granular and repeatable control for the benchmarks. The following diagram illustrates a simple Web Browsing application (HTTP).




In this example, the Client Web Browser (Client) requests a URL and then the Web Server delivers the web page content to the Client (after a Server delay of 100 millisecond). This asynchronous, "request/response" behavior is intrinsic to most TCP based applications such as Email (SMTP), File Transfers (FTP and SMB),

Database (SQL), Web Applications (SOAP), REST, etc. The impact to the network elements is due to the multitudes of Clients and the variety of bursty traffic, which stresses traffic management functions. The actual emulation of the specific application protocols is not required and TCP test patterns can be defined to mimic the application network traffic flows and produce repeatable results.

Application modeling techniques have been proposed in "3GPP2 C.R1002-0 v1.0" and provides examples to model the behavior of HTTP, FTP, and WAP applications at the TCP layer. The models have been defined with various mathematical distributions for the Request/Response bytes and inter-request gap times. The model definition format described in this work are the basis for the guidelines provides in Appendix B and are also similar to formats used by network modeling tools. Packet captures can also be used to characterize application traffic and specify some of the test patterns listed in Appendix B.

This framework does not specify a fixed set of TCP test patterns, but does provide test cases that SHOULD be performed in Appendix B. Some of these examples reflect those specified in "draft-ietf-bmwg-ca-bench-meth-04" which suggests traffic mixes for a variety of representative application profiles. Other examples are simply well-known application traffic types such as HTTP.

## 6. Traffic Benchmarking Methodology

The traffic benchmarking methodology uses the test set-up from section 2 and metrics defined in section 4.

Each test SHOULD compare the network device’s internal statistics (available via command line management interface, SNMP, etc.) to the measured metrics defined in section 4. This evaluates the accuracy of the internal traffic management counters under individual test conditions and capacity test conditions that are defined in each subsection.

From a device configuration standpoint, scheduling and shaping functionality can be applied to logical ports such Link Aggregation (LAG). This would result in the same scheduling and shaping configuration applied to all the member physical ports. The focus of this draft is only on tests at a physical port level.

The following sections provide the objective, procedure, metrics, and reporting format for each test. For all test steps, the following global parameters must be specified:

Test Runs (Tr). Defines the number of times the test needs to be run to ensure accurate and repeatable results. The recommended value is 3.

Test Duration (Td). Defines the duration of a test iteration, expressed in seconds. The recommended minimum value is 60 seconds.

The variability in the test results MUST be measured between Test Runs and if the variation is characterized as a significant portion of the measured values, the next step may be to revise the methods to achieve better consistency.

## 6.1. Policing Tests

A policer is defined as the entity performing the policy function. The intent of the policing tests is to verify the policer performance (i.e. CIR-CBS and EIR-EBS parameters). The tests will verify that the network device can handle the CIR with CBS and the EIR with EBS and will use back-back packet testing concepts from [RFC2544] (but adapted to burst size algorithms and terminology). Also MEF-14,19, 37 provide some basis for specific components of this test. The burst hunt algorithm defined in section 5.1.1 can also be used to automate the measurement of the CBS value.

The tests are divided into two (2) sections; individual policer tests and then full capacity policing tests. It is important to benchmark the basic functionality of the individual policer then proceed into the fully rated capacity of the device. This capacity may include the number of policing policies per device and the number of policers simultaneously active across all ports.

### 6.1.1 Policer Individual Tests

Objective:
Test a policer as defined by [RFC4115] or MEF 10.2, depending upon the equipment’s specification. In addition to verifying that the policer allows the specified CBS and EBS bursts to pass, the policer test MUST verify that the policer will remark or drop excess, and pass traffic at the specified CBS/EBS values.

Test Summary:
Policing tests should use stateless traffic. Stateful TCP test traffic will generally be adversely affected by a policer in the absence of traffic shaping. So while TCP traffic could be used, it is more accurate to benchmark a policer with stateless traffic.

As an example for [RFC4115], consider a CBS and EBS of 64KB and CIR and EIR of 100 Mbps on a 1GigE physical link (in color-blind mode). A stateless traffic burst of 64KB would be sent into the policer at the GigE rate. This equates to approximately a 0.512 millisecond burst time (64 KB at 1 GigE). The traffic generator must space these bursts to ensure that the aggregate throughput does not exceed the CIR. The Ti between the bursts would equal CBS * 8 / CIR = 5.12 millisecond in this example.

Test Metrics:
The metrics defined in section 4.1 (BSA, LP, OOS, PD, and PDV) SHALL be measured at the egress port and recorded.

Procedure:

1. Configure the DUT policing parameters for the desired CIR/EIR and CBS/EBS values to be tested

2. Configure the tester to generate a stateless traffic burst equal to CBS and an interval equal to Ti (CBS in bits / CIR)

3. Compliant Traffic Step: Generate bursts of CBS + EBS traffic into the policer ingress port and measure the metrics defined in section 4.1 (BSA, LP. OOS, PD, and PDV) at the egress port and across the entire Td (default 60 seconds duration)

4. Excess Traffic Test: Generate bursts of greater than CBS + EBS limit traffic into the policer ingress port and verify that the policer only allowed the BSA bytes to exit the egress. The excess burst MUST be recorded and the recommended value is 1000 bytes. Additional tests beyond the simple color-blind example might include: color-aware mode, configurations where EIR is greater than CIR, etc.

Reporting Format:
The policer individual report MUST contain all results for each CIR/EIR/CBS/EBS test run and a recommended format is as follows:

**********
Test Configuration Summary: Tr, Td

DUT Configuration Summary: CIR, EIR, CBS, EBS

The results table should contain entries for each test run, (Test #1 to Test #Tr).

Compliant Traffic Test: BSA, LP, OOS, PD, and PDV

Excess Traffic Test: BSA
**********

### 6.1.2 Policer Capacity Tests

Objective:
The intent of the capacity tests is to verify the policer performance in a scaled environment with multiple ingress customer policers on multiple physical ports. This test will benchmark the maximum number of active policers as specified by the device manufacturer.

Test Summary:
The specified policing function capacity is generally expressed in terms of the number of policers active on each individual physical port as well as the number of unique policer rates that are utilized. For all of the capacity tests, the benchmarking test procedure and report format described in Section 6.1.1 for a single policer MUST be applied to each of the physical port policers.

As an example, a Layer 2 switching device may specify that each of the 32 physical ports can be policed using a pool of policing service policies. The device may carry a single customer's traffic on each physical port and a single policer is instantiated per physical port. Another possibility is that a single physical port may carry multiple customers, in which case many customer flows would be policed concurrently on an individual physical port (separate policers per customer on an individual port).

Test Metrics:
The metrics defined in section 4.1 (BSA, LP, OOS, PD, and PDV) SHALL be measured at the egress port and recorded.

The following sections provide the specific test scenarios, procedures, and reporting formats for each policer capacity test.

6.1.2.1 Maximum Policers on Single Physical Port Test

Test Summary:
The first policer capacity test will benchmark a single physical port, maximum policers on that physical port.

Assume multiple categories of ingress policers at rates r1, r2,...rn. There are multiple customers on a single physical port. Each customer could be represented by a single tagged vlan, double tagged vlan, VPLS instance etc. Each customer is mapped to a different policer. Each of the policers can be of rates r1, r2,..., rn.

An example configuration would be
- Y1 customers, policer rate r1
- Y2 customers, policer rate r2
- Y3 customers, policer rate r3
...
- Yn customers, policer rate rn

Some bandwidth on the physical port is dedicated for other traffic non customer traffic); this includes network control protocol traffic. There is a separate policer for the other traffic. Typical deployments have 3 categories of policers; there may be some deployments with more or less than 3 categories of ingress policers.

Test Procedure:
1. Configure the DUT policing parameters for the desired CIR/EIR and CBS/EBS values for each policer rate (r1-rn) to be tested

2. Configure the tester to generate a stateless traffic burst equal to CBS and an interval equal to TI (CBS in bits/CIR) for each customer stream (Y1 - Yn). The encapsulation for each customer must also be configured according to the service tested (VLAN, VPLS, IP mapping, etc.).

3. Compliant Traffic Step: Generate bursts of CBS + EBS traffic into the policer ingress port for each customer traffic stream and measure the metrics defined in section 4.1 (BSA, LP, OOS, PD, and PDV) at the egress port for each stream and across the entire Td (default 30 seconds duration)

4. Excess Traffic Test: Generate bursts of greater than CBS + EBS limit traffic into the policer ingress port for each customer traffic stream and verify that the policer only allowed the BSA bytes to exit the egress for each stream. The excess burst MUST recorded and the recommended value is 1000 bytes.

Reporting Format:
The policer individual report MUST contain all results for each CIR/EIR/CBS/EBS test run, per customer traffic stream.

A recommended format is as follows:

**********
Test Configuration Summary: Tr, Td

Customer traffic stream Encapsulation: Map each stream to VLAN, VPLS, IP address

DUT Configuration Summary per Customer Traffic Stream: CIR, EIR, CBS, EBS

The results table should contain entries for each test run, (Test #1 to Test #Tr).

Customer Stream Y1-Yn (see note), Compliant Traffic Test: BSA, LP, OOS, PD, and PDV

Customer Stream Y1-Yn (see note), Excess Traffic Test: BSA
**********

Note: For each test run, there will be a two (2) rows for each customer stream, the compliant traffic result and the excess traffic result.

6.1.2.2 Single Policer on All Physical Ports

Test Summary:
The second policer capacity test involves a single Policer function per physical port with all physical ports active. In this test, there is a single policer per physical port. The policer can have one of the rates r1, r2,.., rn. All the physical ports in the networking device are active.

Procedure:
The procedure is identical to 6.1.1, the configured parameters must be reported per port and the test report must include results per measured egress port

#### 6.1.2.3 Maximum Policers on All Physical Ports

Finally the third policer capacity test involves a combination of the first and second capacity test, namely maximum policers active per physical port and all physical ports are active.

Procedure:
Uses the procedural method from 6.1.2.1 and the configured parameters must be reported per port and the test report must include per stream results per measured egress port.

### 6.2. Queue and Scheduler Tests

Queues and traffic Scheduling are closely related in that a queue's priority dictates the manner in which the traffic scheduler transmits packets out of the egress port.

Since device queues / buffers are generally an egress function, this test framework will discuss testing at the egress (although the technique can be applied to ingress side queues).

Similar to the policing tests, the tests are divided into two sections; individual queue/scheduler function tests and then full capacity tests.

#### 6.2.1 Queue/Scheduler Individual Tests Overview

The various types of scheduling techniques include FIFO, Strict Priority (SP), Weighted Fair Queueing (WFQ) along with other variations. This test framework recommends to test at a minimum of three techniques although it is the discretion of the tester to benchmark other device scheduling algorithms.

#### 6.2.1.1 Queue/Scheduler with Stateless Traffic Test

Objective:
Verify that the configured queue and scheduling technique can handle stateless traffic bursts up to the queue depth.

Test Summary:
A network device queue is memory based unlike a policing function, which is token or credit based. However, the same concepts from section 6.1 can be applied to testing network device queues.

The device's network queue should be configured to the desired size in KB (queue length, QL) and then stateless traffic should be transmitted to test this QL.

A queue should be able to handle repetitive bursts with the transmission gaps proportional to the bottleneck bandwidth. This gap is referred to as the transmission interval (Ti). Ti can be defined for the traffic bursts and is based off of the QL and Bottleneck Bandwidth (BB) of the egress interface.

Ti = QL * 8 / BB

Note that this equation is similar to the Ti required for transmission into a policer (QL = CBS, BB = CIR). Also note that the burst hunt algorithm defined in section 5.1.1 can also be used to automate the measurement of the queue value.

The stateless traffic burst shall be transmitted at the link speed and spaced within the Ti time interval. The metrics defined in section 4.1 shall be measured at the egress port and recorded; the primary result is to verify the BSA and that no packets are dropped.

The scheduling function must also be characterized to benchmark the device’s ability to schedule the queues according to the priority. An example would be 2 levels of priority including SP and FIFO queueing. Under a flow load greater the egress port speed, the higher priority packets should be transmitted without drops (and also maintain low latency), while the lower priority (or best effort) queue may be dropped.

Test Metrics:
The metrics defined in section 4.1 (BSA, LP, OOS, PD, and PDV) SHALL be measured at the egress port and recorded.

Procedure:

1. Configure the DUT queue length (QL) and scheduling technique (FIFO, SP, etc) parameters

2. Configure the tester to generate a stateless traffic burst equal to QL and an interval equal to Ti (QL in bits/BB)

3. Generate bursts of QL traffic into the DUT and measure the metrics defined in section 4.1 (LP, OOS, PD, and PDV) at the egress port and across the entire Td (default 30 seconds duration)

Report Format:
The Queue/Scheduler Stateless Traffic individual report MUST contain all results for each QL/BB test run and a recommended format is as follows:

**********
Test Configuration Summary: Tr, Td

DUT Configuration Summary: Scheduling technique, BB and QL

The results table should contain entries for each test run as follows,

(Test #1 to Test #Tr).

- LP, OOS, PD, and PDV
**********

6.2.1.2 Testing Queue/Scheduler with Stateful Traffic

Objective:
Verify that the configured queue and scheduling technique can handle stateful traffic bursts up to the queue depth.

Test Background and Summary:
To provide a more realistic benchmark and to test queues in layer 4 devices such as firewalls, stateful traffic testing is recommended for the queue tests. Stateful traffic tests will also utilize the Network Delay Emulator (NDE) from the network set-up configuration in section 2.

The BDP of the TCP test traffic must be calibrated to the QL of the device queue. Referencing [RFC6349], the BDP is equal to:

BB * RTT / 8 (in bytes)

The NDE must be configured to an RTT value which is large enough to allow the BDP to be greater than QL. An example test scenario is defined below:

- Ingress link = GigE
- Egress link = 100 Mbps (BB)
- QL = 32KB

RTT(min) = QL * 8 / BB and would equal 2.56 millisecond (and the BDP = 32KB)

In this example, one (1) TCP connection with window size / SSB of 32KB would be required to test the QL of 32KB. This Bulk Transfer Test can be accomplished using iperf as described in Appendix A.

Two types of TCP tests MUST be performed: Bulk Transfer test and Micro Burst Test Pattern as documented in Appendix B. The Bulk Transfer Test only bursts during the TCP Slow Start (or Congestion Avoidance) state, while the Micro Burst test emulates application layer bursting which may occur any time during the TCP connection.

Other tests types SHOULD include: Simple Web Site, Complex Web Site, Business Applications, Email, SMB/CIFS File Copy (which are also documented in Appendix B).

Test Metrics:
The test results will be recorded per the stateful metrics defined in section 4.2, primarily the TCP Test Pattern Execution Time (TTPET), TCP Efficiency, and Buffer Delay.

Procedure:

1. Configure the DUT queue length (QL) and scheduling technique (FIFO, SP, etc) parameters

2. Configure the tester* to generate a profile of emulated of an application traffic mixture

- The application mixture MUST be defined in terms of percentage of the total bandwidth to be tested

- The rate of transmission for each application within the mixture MUST be also be configurable

* The tester MUST be capable of generating a precise TCP test patterns for each application specified, to ensure repeatable results.

3. Generate application traffic between the ingress (client side) and egress (server side) ports of the DUT and measure application throughput the metrics (TTPET, TCP Efficiency, and Buffer Delay),

   per application stream and at the ingress and egress port (across the entire Td, default 60 seconds duration).

Concerning application measurements, a couple of items require clarification. An application session may be comprised of a single TCP connection or multiple TCP connections.

For the single TCP connection application sessions, the application thoughput / metrics have a 1-1 relationship to the TCP connection measurements.

If an application session (i.e. HTTP-based application) utilizes multiple TCP connections, then all of the TCP connections are aggregated in the application throughput measurement / metrics for that application.

Then there is the case of mulitlple instances of an application session (i.e. multiple FTPs emulating multiple clients). In this situation, the test should measure / record each FTP application session independently, tabulating the minimum, maximum, and average for all FTP sessions.

Finally, application throughput measurements are based on Layer 4 TCP throughput and do not include bytes retransmitted. The TCP Efficiency metric MUST be measured during the test and provides a measure of "goodput" during each test.

Reporting Format:
The Queue/Scheduler Stateful Traffic individual report MUST contain all results for each traffic scheduler and QL/BB test run and a recommended format is as follows:

**********
Test Configuration Summary: Tr, Td

DUT Configuration Summary: Scheduling technique, BB and QL

Application Mixture and Intensities: this is the percent configured of each application type

The results table should contain entries for each test run with minimum, maximum, and average per application session as follows, (Test #1 to Test #Tr)

- Per Application Throughout (bps) and TTPET
- Per Application Bytes In and Bytes Out
- Per Application TCP Efficiency, and Buffer Delay
**********

### 6.2.2 Queue / Scheduler Capacity Tests

Objective:
The intent of these capacity tests is to benchmark queue/scheduler performance in a scaled environment with multiple queues/schedulers active on multiple egress physical ports. This test will benchmark the maximum number of queues and schedulers as specified by the device manufacturer. Each priority in the system will map to a separate queue.

Test Metrics:
The metrics defined in section 4.1 (BSA, LP, OOS, PD, and PDV) SHALL be measured at the egress port and recorded.

The following sections provide the specific test scenarios, procedures, and reporting formats for each queue / scheduler capacity test.

#### 6.2.2.1 Multiple Queues / Single Port Active

For the first scheduler / queue capacity test, multiple queues per port will be tested on a single physical port. In this case, all the queues (typically 8) are active on a single physical port. Traffic from multiple ingress physical ports are directed to the same egress physical port which will cause oversubscription on the egress physical port.

There are many types of priority schemes and combinations of priorities that are managed by the scheduler. The following sections specify the priority schemes that should be tested.

6.2.2.1.1 Strict Priority on Egress Port

Test Summary:
For this test, Strict Priority (SP) scheduling on the egress physical port should be tested and the benchmarking methodology specified in section 6.2.1.1 and 6.2.1.2 (procedure, metrics, and reporting format) should be applied here. For a given priority, each ingress physical port should get a fair share of the egress physical port bandwidth.

Since this is a capacity test, the configuration and report results format from 6.2.1.1 and 6.2.1.2 MUST also include:

Configuration:
- The number of physical ingress ports active during the test
- The classication marking (DSCP, VLAN, etc.) for each physical ingress port
- The traffic rate for stateful traffic and the traffic rate / mixture for stateful traffic for each physical ingress port

Report results:
- For each ingress port traffic stream, the achieved throughput rate and metrics at the egress port

6.2.2.1.2 Strict Priority + Weighted Fair Queue (WFQ) on Egress Port

Test Summary:
For this test, Strict Priority (SP) and Weighted Fair Queue (WFQ) should be enabled simultaneously in the scheduler but on a single egress port. The benchmarking methodology specified in Section

6.2.1.1 and 6.2.1.2 (procedure, metrics, and reporting format) should be applied here. Additionally, the egress port bandwidth sharing among weighted queues should be proportional to the assigned weights. For a given priority, each ingress physical port should get a fair share of the egress physical port bandwidth.

Since this is a capacity test, the configuration and report results format from 6.2.1.1 and 6.2.1.2 MUST also include:

Configuration:
- The number of physical ingress ports active during the test
- The classication marking (DSCP, VLAN, etc.) for each physical ingress port
- The traffic rate for stateful traffic and the traffic rate / mixture for stateful traffic for each physical ingress port

Report results:
- For each ingress port traffic stream, the achieved throughput rate and metrics at each queue of the egress port queue (both the SP and WFQ queue).

Example:
- Egress Port SP Queue: throughput and metrics for ingress streams 1-n
- Egress Port WFQ Queue: throughput and metrics for ingress streams 1-n

6.2.2.2 Single Queue per Port / All Ports Active

Test Summary:
Traffic from multiple ingress physical ports are directed to the same egress physical port, which will cause oversubscription on the egress physical port. Also, the same amount of traffic is directed to each egress physical port.

The benchmarking methodology specified in Section 6.2.1.1 and 6.2.1.2 (procedure, metrics, and reporting format) should be applied here. Each ingress physical port should get a fair share of the egress physical port bandwidth. Additionally, each egress physical port should receive the same amount of traffic.

Since this is a capacity test, the configuration and report results format from 6.2.1.1 and 6.2.1.2 MUST also include:

Configuration:
- The number of ingress ports active during the test
- The number of egress ports active during the test
- The classication marking (DSCP, VLAN, etc.) for each physical ingress port
- The traffic rate for stateful traffic and the traffic rate / mixture for stateful traffic for each physical ingress port

Report results:
- For each egress port, the achieved throughput rate and metrics at the egress port queue for each ingress port stream.

Example:
- Egress Port 1: throughput and metrics for ingress streams 1-n
- Egress Port n: throughput and metrics for ingress streams 1-n

6.2.2.3 Multiple Queues per Port, All Ports Active

Traffic from multiple ingress physical ports are directed to all queues of each egress physical port, which will cause oversubscription on the egress physical ports. Also, the same amount of traffic is directed to each egress physical port.

The benchmarking methodology specified in Section 6.2.1.1 and 6.2.1.2 (procedure, metrics, and reporting format) should be applied here. For a given priority, each ingress physical port should get a fair share of the egress physical port bandwidth. Additionally, each egress physical port should receive the same amount of traffic.

Since this is a capacity test, the configuration and report results format from 6.2.1.1 and 6.2.1.2 MUST also include:

Configuration:
- The number of physical ingress ports active during the test
- The classication marking (DSCP, VLAN, etc.) for each physical ingress port
- The traffic rate for stateful traffic and the traffic rate / mixture for stateful traffic for each physical ingress port

Report results:
- For each egress port, the achieved throughput rate and metrics at each egress port queue for each ingress port stream.

Example:
- Egress Port 1, SP Queue: throughput and metrics for ingress streams 1-n
- Egress Port 2, WFQ Queue: throughput and metrics for ingress streams 1-n
.
.
- Egress Port n, SP Queue: throughput and metrics for ingress streams 1-n
- Egress Port n, WFQ Queue: throughput and metrics for ingress streams 1-n

## 6.3. Shaper tests

A traffic shaper is memory based like a queue, but with the added intelligence of an active traffic scheduler. The same concepts from section 6.2 (Queue testing) can be applied to testing network device shaper.

Again, the tests are divided into two sections; individual shaper benchmark tests and then full capacity shaper benchmark tests.

### 6.3.1 Shaper Individual Tests Overview

A traffic shaper generally has three (3) components that can be configured:

- Ingress Queue bytes
- Shaper Rate, bps
- Burst Committed (Bc) and Burst Excess (Be), bytes

The Ingress Queue holds burst traffic and the shaper then meters traffic out of the egress port according to the Shaper Rate and Bc/Be parameters. Shapers generally transmit into policers, so the idea is for the emitted traffic to conform to the policer’s limits.

6.3.1.1 Testing Shaper with Stateless Traffic

Objective:
Test a shaper by transmitting stateless traffic bursts into the shaper ingress port and verifying that the egress traffic is shaped according to the shaper traffic profile.

Test Summary:
The stateless traffic must be burst into the DUT ingress port and not exceed the Ingress Queue. The burst can be a single burst or multiple bursts. If multiple bursts are transmitted, then the Ti (Time interval) must be large enough so that the Shaper Rate is not exceeded. An example will clarify single and multiple burst test cases.

In the example, the shaper's ingress and egress ports are both full duplex Gigabit Ethernet. The Ingress Queue is configured to be 512,000 bytes, the Shaper Rate (SR) = 50 Mbps, and both Bc/Be configured to be 32,000 bytes. For a single burst test, the transmitting test device would burst 512,000 bytes maximum into the ingress port and then stop transmitting.

If a multiple burst test is to be conducted, then the burst bytes divided by the time interval between the 512,000 byte bursts must not exceed the Shaper Rate. The time interval (Ti) must adhere to a similar formula as described in section 6.2.1.1 for queues, namely:

Ti = Ingress Queue x 8 / Shaper Rate

For the example from the previous paragraph, Ti between bursts must be greater than 82 millisecond (512,000 bytes x 8 / 50,000,000 bps). This yields an average rate of 50 Mbps so that an Input Queue would not overflow.

Test Metrics:
- The metrics defined in section 4.1 (LP, OOS, PDV, SR, SBB, SBI) SHALL be measured at the egress port and recorded.

Procedure:
1. Configure the DUT shaper ingress queue length (QL) and shaper egress rate parameters (SR, Bc, Be) parameters

2. Configure the tester to generate a stateless traffic burst equal to QL and an interval equal to Ti (QL in bits/BB)

3. Generate bursts of QL traffic into the DUT and measure the metrics defined in section 4.1 (LP, OOS, PDV, SR, SBB, SBI) at the egress port and across the entire Td (default 30 seconds duration)

Report Format:
The Shaper Stateless Traffic individual report MUST contain all results for each QL/SR test run and a recommended format is as follows:
**********
Test Configuration Summary: Tr, Td

DUT Configuration Summary: Ingress Burst Rate, QL, SR

The results table should contain entries for each test run as follows,(Test #1 to Test #Tr).

- LP, OOS, PDV, SR, SBB, SBI
**********

6.3.1.2 Testing Shaper with Stateful Traffic

Objective:
Test a shaper by transmitting stateful traffic bursts into the shaper ingress port and verifying that the egress traffic is shaped according to the shaper traffic profile.

Test Summary:
To provide a more realistic benchmark and to test queues in layer 4 devices such as firewalls, stateful traffic testing is also recommended for the shaper tests. Stateful traffic tests will also utilize the Network Delay Emulator (NDE) from the network set-up configuration in section 2.

The BDP of the TCP test traffic must be calculated as described in section 6.2.2. To properly stress network buffers and the traffic shaping function, the cumulative TCP window should exceed the BDP which will stress the shaper. BDP factors of 1.1 to 1.5 are recommended, but the values are the discretion of the tester and should be documented.

The cumulative TCP Window Sizes* (RWND at the receiving end & CWND at the transmitting end) equates to:

TCP window size* for each connection x number of connections

* as described in section 3 of [RFC6349], the SSB MUST be large enough to fill the BDP

Example, if the BDP is equal to 256 Kbytes and a connection size of 64Kbytes is used for each connection, then it would require four (4) connections to fill the BDP and 5-6 connections (over subscribe the BDP) to stress test the traffic shaping function.

Two types of TCP tests MUST be performed: Bulk Transfer test and Micro Burst Test Pattern as documented in Appendix B. The Bulk Transfer Test only bursts during the TCP Slow Start (or Congestion Avoidance) state, while the Micro Burst test emulates application layer bursting which may any time during the TCP connection.

Other tests types SHOULD include: Simple Web Site, Complex Web Site, Business Applications, Email, SMB/CIFS File Copy (which are also documented in Appendix B).

Test Metrics:
The test results will be recorded per the stateful metrics defined in section 4.2, primarily the TCP Test Pattern Execution Time (TTPET), TCP Efficiency, and Buffer Delay.

Procedure:
1. Configure the DUT shaper ingress queue length (QL) and shaper egress rate parameters (SR, Bc, Be) parameters

2. Configure the tester* to generate a profile of emulated of an application traffic mixture

   - The application mixture MUST be defined in terms of percentage of the total bandwidth to be tested

   - The rate of transmission for each application within the mixture MUST be also be configurable

*The tester MUST be capable of generating precise TCP test patterns for each application specified, to ensure repeatable results.

3. Generate application traffic between the ingress (client side) and egress (server side) ports of the DUT and measure the metrics (TTPET, TCP Efficiency, and Buffer Delay) per application stream and at the ingress and egress port (across the entire Td, default 30 seconds duration).

Reporting Format:
The Shaper Stateful Traffic individual report MUST contain all results for each traffic scheduler and QL/SR test run and a recommended format is as follows:

**********
Test Configuration Summary: Tr, Td

DUT Configuration Summary: Ingress Burst Rate, QL, SR

Application Mixture and Intensities: this is the percent configured of each application type

The results table should contain entries for each test run with minimum, maximum, and average per application session as follows, (Test #1 to Test #Tr)

- Per Application Throughout (bps) and TTPET
- Per Application Bytes In and Bytes Out
- Per Application TCP Efficiency, and Buffer Delay

**********

### 6.3.2 Shaper Capacity Tests

Objective:
The intent of these scalability tests is to verify shaper performance in a scaled environment with shapers active on multiple queues on multiple egress physical ports. This test will benchmark the maximum number of shapers as specified by the device manufacturer.

The following sections provide the specific test scenarios, procedures, and reporting formats for each shaper capacity test.

#### 6.3.2.1 Single Queue Shaped, All Physical Ports Active

Test Summary:
The first shaper capacity test involves per port shaping, all physical ports active. Traffic from multiple ingress physical ports are directed to the same egress physical port and this will cause oversubscription on the egress physical port. Also, the same amount of traffic is directed to each egress physical port.

The benchmarking methodology specified in Section 6.3.1 (procedure, metrics, and reporting format) should be applied here. Since this is a capacity test, the configuration and report results format from 6.3.1 MUST also include:

Configuration:
- The number of physical ingress ports active during the test
- The classication marking (DSCP, VLAN, etc.) for each physical ingress port
- The traffic rate for stateful traffic and the traffic rate / mixture for stateful traffic for each physical ingress port
- The shaped egress ports shaper parameters (QL, SR, Bc, Be)

Report results:
- For each active egress port, the achieved throughput rate and shaper metrics for each ingress port traffic stream

Example:
- Egress Port 1: throughput and metrics for ingress streams 1-n
- Egress Port n: throughput and metrics for ingress streams 1-n

#### 6.3.2.2 All Queues Shaped, Single Port Active

Test Summary:
The second shaper capacity test is conducted with all queues actively shaping on a single physical port. The benchmarking methodology

described in per port shaping test (previous section) serves as the foundation for this. Additionally, each of the SP queues on the egress physical port is configured with a shaper. For the highest priority queue, the maximum amount of bandwidth available is limited by the bandwidth of the shaper. For the lower priority queues, the maximum amount of bandwidth available is limited by the bandwidth of the shaper and traffic in higher priority queues.

The benchmarking methodology specified in Section 6.3.1 (procedure, metrics, and reporting format) should be applied here. Since this is a capacity test, the configuration and report results format from 6.3.1 MUST also include:

Configuration:
- The number of physical ingress ports active during the test
- The classication marking (DSCP, VLAN, etc.) for each physical ingress port
- The traffic rate for stateful traffic and the traffic rate/mixture for stateful traffic for each physical ingress port
- For the active egress port, each shaper queue parameters (QL, SR, Bc, Be)

Report results:
- For each queue of the active egress port, the achieved throughput rate and shaper metrics for each ingress port traffic stream

Example:
- Egress Port High Priority Queue: throughput and metrics for ingress streams 1-n
- Egress Port Lower Priority Queue: throughput and metrics for ingress streams 1-n

6.3.2.3 All Queues Shaped, All Ports Active

Test Summary:
And for the third shaper capacity test (which is a combination of the tests in the previous two sections),all queues will be actively shaping and all physical ports active.

The benchmarking methodology specified in Section 6.3.1 (procedure, metrics, and reporting format) should be applied here. Since this is a capacity test, the configuration and report results format from 6.3.1 MUST also include:

Configuration:
- The number of physical ingress ports active during the test
- The classication marking (DSCP, VLAN, etc.) for each physical ingress port
- The traffic rate for stateful traffic and the traffic rate / mixture for stateful traffic for each physical ingress port
- For each of the active egress ports, shaper port and per queue parameters(QL, SR, Bc, Be)

Report results:
- For each queue of each active egress port, the achieved throughput rate and shaper metrics for each ingress port traffic stream

Example:
- Egress Port 1 High Priority Queue: throughput and metrics for ingress streams 1-n
- Egress Port 1 Lower Priority Queue: throughput and metrics for ingress streams 1-n
.
- Egress Port n High Priority Queue: throughput and metrics for ingress streams 1-n
- Egress Port n Lower Priority Queue: throughput and metrics for ingress streams 1-n

6.4 Concurrent Capacity Load Tests

As mentioned in the scope of this document, it is impossible to specify the various permutations of concurrent traffic management functions that should be tested in a device for capacity testing. However, some profiles are listed below which may be useful to test under capacity as well:

- Policers on ingress and queuing on egress
- Policers on ingress and shapers on egress (not intended for a flow to be policed then shaped, these would be two different flows tested at the same time)
- etc.

The test procedures and reporting formatting from the previous sections may be modified to accommodate the capacity test profile.

Appendix A: Open Source Tools for Traffic Management Testing

This framework specifies that stateless and stateful behaviors SHOULD both be tested. Four (4) open source tools that can be used are iperf, netperf (with netperf-wrapper), uperf, and TMIX to accomplish many of the tests proposed in this framework.

Iperf can generate UDP or TCP based traffic; a client and server must both run the iperf software in the same traffic mode. The server is set up to listen and then the test traffic is controlled from the client. Both uni-directional and bi-directional concurrent testing are supported.

The UDP mode can be used for the stateless traffic testing. The target bandwidth, packet size, UDP port, and test duration can be controlled. A report of bytes transmitted, packets lost, and delay variation are provided by the iperf receiver.

The TCP mode can be used for stateful traffic testing to test bulk transfer traffic. The TCP Window size (which is actually the SSB), the number of connections, the packet size, TCP port and the test duration can be controlled. A report of bytes transmitted and throughput achieved are provided by the iperf sender.

Netperf is a software application that provides network bandwidth testing between two hosts on a network. It supports Unix domain sockets, TCP, SCTP, DLPI and UDP via BSD Sockets. Netperf provides a number of predefined tests e.g. to measure bulk (unidirectional) data transfer or request response performance http://en.wikipedia.org/wiki/Netperf). Netperf-wrapper is a Python script that runs multiple simultaneous netperf instances and aggregate the results.

uperf uses a description (or model) of an application mixture and the tool generates the load according to the model desciptor. uperf is more flexible than Netperf in it's ability to generate request / response application behavior within a single TCP connection. The application model descriptor can be based off of empirical data, but currently the import of packet captures is not directly supported.

Tmix is another application traffic emulation tool and uses packet captures directly to create the traffic profile. The packet trace is 'reverse compiled' into a source-level characterization, called a connection vector, of each TCP connection present in the trace. While most widely used in ns2 simulation environment, TMix also runs on Linux hosts.

Iperf, Netperf-wrapper, uperf, and Tmix's traffic generation parameters facilitate the emulation of the TCP test patterns which are discussed in Appendix B.

## Appendix B: Stateful TCP Test Patterns

This framework recommends at a minimum the following TCP test patterns since they are representative of real world application traffic (section 5.2.1 describes some methods to derive other application-based TCP test patterns).

- Bulk Transfer: generate concurrent TCP connections whose aggregate number of in-flight data bytes would fill the BDP. Guidelines from [RFC6349] are used to create this TCP traffic pattern.

- Micro Burst: generate precise burst patterns within a single or multiple TCP connections(s). The idea is for TCP to establish equilibrium and then burst application bytes at defined sizes. The test tool must allow the burst size and burst time interval to be configurable.

- Web Site Patterns: The HTTP traffic model from "3GPP2 C.R1002-0 v1.0" is referenced (Table 4.1.3.2.1) to develop these TCP test patterns. In summary, the HTTP traffic model consists of the following parameters:
    - Main object size (Sm)
    - Embedded object size (Se)
    - Number of embedded objects per page (Nd)
    - Client processing time (Tcp)
    - Server processing time (Tsp)

Web site test patterns are illustrated with the following examples:

- Simple Web Site: mimic the request / response and object download behavior of a basic web site (small company).
- Complex Web Site: mimic the request / response and object download behavior of a complex web site (ecommerce site).

Referencing the HTTP traffic model parameters , the following table was derived (by analysis and experimentation) for Simple and Complex Web site TCP test patterns:

| Parameter | Simple Web Site | Complex Web Site |
|---|---|---|
| Main object size (Sm) | Ave. = 10KB<br>Min. = 100B<br>Max. = 500KB | Ave. = 300KB<br>Min. = 50KB<br>Max. = 2MB |
| Embedded object size (Se) | Ave. = 7KB<br>Min. = 50B<br>Max. = 350KB | Ave. = 10KB<br>Min. = 100B<br>Max. = 1MB |
| Number of embedded objects per page (Nd) | Ave. = 5<br>Min. = 2<br>Max. = 10 | Ave. = 25<br>Min. = 10<br>Max. = 50 |
| Client processing time (Tcp)* | Ave. = 3s<br>Min. = 1s<br>Max. = 10s | Ave. = 10s<br>Min. = 3s<br>Max. = 30s |
| Server processing time (Tsp)* | Ave. = 5s<br>Min. = 1s<br>Max. = 15s | Ave. = 8s<br>Min. = 2s<br>Max. = 30s |

* The client and server processing time is distributed across the transmission / receipt of all of the main and embedded objects

To be clear, the parameters in this table are reasonable guidelines for the TCP test pattern traffic generation. The test tool can use fixed parameters for simpler tests and mathematical distributions for more complex tests. However, the test pattern must be repeatable to ensure that the benchmark results can be reliably compared.

- Inter-active Patterns: While Web site patterns are inter-active to a degree, they mainly emulate the downloading of various complexity web sites. Inter-active patterns are more chatty in nature since there is alot of user interaction with the servers. Examples include business applications such as Peoplesoft, Oracle and consumer applications such as Facebook, IM, etc. For the inter-active patterns, the packet capture technique was used to characterize some business applications and also the email application.

In summary, an inter-active application can be described by the following parameters:

- Client message size (Scm)
- Number of Client messages (Nc)
- Server response size (Srs)
- Number of server messages (Ns)
- Client processing time (Tcp)
- Server processing Time (Tsp)
- File size upload (Su)*
- File size download (Sd)*

* The file size parameters account for attachments uploaded or downloaded and may not be present in all inter-active applications

Again using packet capture as a means to characterize, the following table reflects the guidelines for Simple Business Application, Complex Business Application, eCommerce, and Email Send / Receive:

| Parameter | Simple Biz. App. | Complex Biz. App | eCommerce* | Email |
|---|---|---|---|---|
| Client message size (Scm) | Ave. = 450B<br>Min. = 100B<br>Max. = 1.5KB | Ave. = 2KB<br>Min. = 500B<br>Max. = 100KB | Ave. = 1KB<br>Min. = 100B<br>Max. = 50KB | Ave. = 200B<br>Min. = 100B<br>Max. = 1KB |
| Number of client messages (Nc) | Ave. = 10<br>Min. = 5<br>Max. = 25 | Ave. = 100<br>Min. = 50<br>Max. = 250 | Ave. = 20<br>Min. = 10<br>Max. = 100 | Ave. = 10<br>Min. = 5<br>Max. = 25 |
| Client processing time (Tcp)** | Ave. = 10s<br>Min. = 3s<br>Max. = 30s | Ave. = 30s<br>Min. = 3s<br>Max. = 60s | Ave. = 15s<br>Min. = 5s<br>Max. = 120s | Ave. = 5s<br>Min. = 3s<br>Max. = 45s |
| Server response size (Srs) | Ave. = 2KB<br>Min. = 500B<br>Max. = 100KB | Ave. = 5KB<br>Min. = 1KB<br>Max. = 1MB | Ave. = 8KB<br>Min. = 100B<br>Max. = 50KB | Ave. = 200B<br>Min. = 150B<br>Max. = 750B |
| Number of server messages (Ns) | Ave. = 50<br>Min. = 10<br>Max. = 200 | Ave. = 200<br>Min. = 25<br>Max. = 1000 | Ave. = 100<br>Min. = 15<br>Max. = 500 | Ave. = 15<br>Min. = 5<br>Max. = 40 |
| Server processing time (Tsp)** | Ave. = 0.5s<br>Min. = 0.1s<br>Max. = 5s | Ave. = 1s<br>Min. = 0.5s<br>Max. = 20s | Ave. = 2s<br>Min. = 1s<br>Max. = 10s | Ave. = 4s<br>Min. = 0.5s<br>Max. = 15s |
| File size upload (Su) | Ave. = 50KB<br>Min. = 2KB<br>Max. = 200KB | Ave. = 100KB<br>Min. = 10KB<br>Max. = 2MB | Ave. = N/A<br>Min. = N/A<br>Max. = N/A | Ave. = 100KB<br>Min. = 20KB<br>Max. = 10MB |
| File size download (Sd) | Ave. = 50KB<br>Min. = 2KB<br>Max. = 200KB | Ave. = 100KB<br>Min. = 10KB<br>Max. = 2MB | Ave. = N/A<br>Min. = N/A<br>Max. = N/A | Ave. = 100KB<br>Min. = 20KB<br>Max. = 10MB |

 * eCommerce used a combination of packet capture techniques and reference traffic flows from "SPECweb2009" (need proper reference)
** The client and server processing time is distributed across the transmission / receipt of all of messages. Client processing time consists mainly of the delay between user interactions (not machine processing).

And again, the parameters in this table are the guidelines for the TCP test pattern traffic generation. The test tool can use fixed parameters for simpler tests and mathematical distributions for more complex tests. However, the test pattern must be repeatable to ensure that the benchmark results can be reliably compared.

- SMB/CIFS File Copy: mimic a network file copy, both read and write. As opposed to FTP which is a bulk transfer and is only flow controlled via TCP, SMB/CIFS divides a file into application blocks and utilizes application level handshaking in addition to TCP flow control.

In summary, an SMB/CIFS file copy can be described by the following parameters:

- Client message size (Scm)
- Number of client messages (Nc)
- Server response size (Srs)
- Number of Server messages (Ns)
- Client processing time (Tcp)
- Server processing time (Tsp)
- Block size (Sb)

The client and server messages are SMB control messages. The Block size is the data portion of th file transfer.

Again using packet capture as a means to characterize the following table reflects the guidelines for SMB/CIFS file copy:

| Parameter | SMB File Copy |
|---|---|
| Client message size (Scm) | Ave. = 450B<br>Min. = 100B<br>Max. = 1.5KB |
| Number of client messages (Nc) | Ave. = 10<br>Min. = 5<br>Max. = 25 |
| Client processing time (Tcp) | Ave. = 1ms<br>Min. = 0.5ms<br>Max. = 2 |
| Server response size (Srs) | Ave. = 2KB<br>Min. = 500B<br>Max. = 100KB |
| Number of server messages (Ns) | Ave. = 10<br>Min. = 10<br>Max. = 200 |

```
 Server processing Ave. = 1ms
    time (Tsp)       Min. = 0.5ms
                     Max. = 2ms
    Block            Ave. = N/A
     Size (Sb)*      Min. = 16KB
                     Max. = 128KB
```

*Depending upon the tested file size, the block size will be transferred n number of times to complete the example. An example would be a 10 MB file test and 64KB block size. In this case 160 blocks would be transferred after the control channel is opened between the client and server.

## 7. Security Considerations

Documents of this type do not directly affect the security of the Internet or of corporate networks as long as benchmarking is not performed on devices or systems connected to production networks.

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 document does not REQUIRE an IANA registration for ports dedicated to the TCP testing described in this document.

## 9. Acknowledgments

We would like to thank Al Morton for his continuous review and invaluable input to the document. We would also like to thank Scott Bradner for providing guidance early in the drafts conception in the area of benchmarking scope of traffic management functions. Additionally, we would like to thank Tim Copley for this original input and David Taht, Gory Erg, Toke Hoiland-Jorgensen for their review and input for the AQM group. And for the formal reviews of this document, we would like to thank Gilles Forget, Vijay Gurbani, Reinhard Schrage, and Bhuvaneswaran Vengainathan

## 10. References

### 10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC2119, March 1997.

[RFC1242] S. Bradner, "Benchmarking Terminology for Network Interconnection Devices," RFC1242 July 1991

[RFC2544] S. Bradner, "Benchmarking Methodology for Network Interconnect Devices," RFC2544 March 1999

[RFC3148] M. Mathis et al., A Framework for Defining Empirical Bulk Transfer Capacity Metrics," RFC3148 July 2001

[RFC5481] A. Morton et al., "Packet Delay Variation Applicability Statement," RFC5481 March 2009

[RFC6703] A. Morton et al., "Reporting IP Network Performance Metrics: Different Points of View." RFC 6703 August 2012

[RFC2680] G. Almes et al., "A One-way Packet Loss Metric for IPPM," RFC2680 September 1999

[RFC2697] J. Heinanen et al., "A Single Rate Three Color Marker," RFC2697, September 1999

[RFC2698] J. Heinanen et al., "A Two Rate Three Color Marker, " RFC2698, September 1999

[RFC4689] S. Poretsky et al., "Terminology for Benchmarking Network-layer Traffic Control Mechanisms," RFC4689, October 2006

[RFC4737] A. Morton et al., "Packet Reordering Metrics," RFC4737, February 2006

[RFC4115] O. Aboul-Magd et al., "A Differentiated Service Two-Rate, Three-Color Marker with Efficient Handling of in-Profile Traffic." RFC4115 July 2005

[RFC6349] Barry Constantine et al., "Framework for TCP Throughput Testing," RFC6349, August 2011

[AQM-RECO] Fred Baker et al., "IETF Recommendations Regarding Active Queue Management," August 2014, https://datatracker.ietf.org/doc/draft-ietf-aqm-recommendation/

[MEF-10.2] "MEF 10.2: Ethernet Services Attributes Phase 2," October 2009, http://metroethernetforum.org/PDF_Documents/technical-specifications/MEF10.2.pdf

[MEF-12.1] "MEF 12.1: Carrier Ethernet Network Architecture Framework -- Part 2: Ethernet Services Layer - Base Elements," April 2010, https://www.metroethernetforum.org/Assets/Technical_Specifications/PDF/MEF12.1.pdf

[MEF-26] "MEF 26: External Network Network Interface (ENNI) - Phase 1,"January 2010, http://www.metroethernetforum.org /PDF_Documents/technical-specifications/MEF26.pdf

10.2. Informative References

Authors' Addresses

Barry Constantine
JDSU, Test and Measurement Division
Germantown, MD 20876-7100, USA
Phone: +1 240 404 2227
Email: barry.constantine@jdsu.com

Ram Krishnan
Brocade Communications
San Jose, 95134, USA
Phone: +001-408-406-7890
Email: ramk@brocade.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 6, 2015

A. Morton
AT&T Labs

February 2, 2015

# Considerations for Benchmarking Virtual Network Functions and Their Infrastructure
## draft-morton-bmwg-virtual-net-03

## Abstract

Benchmarking Methodology Working Group has traditionally conducted laboratory characterization of dedicated physical implementations of internetworking functions. This memo investigates additional considerations when network functions are virtualized and performed in commodity off-the-shelf hardware.

NOTES:

3.4 Added inter-actions/dependencies within resource domains

4.3 Added new metrics for characterization: PDV, reordering, mean delay, etc.

4.4 Resolved the question of capacity and the 3x3 Matrix

## Requirements Language

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

## Status of This Memo

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

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

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

This Internet-Draft will expire on August 6, 2015.



Table of Contents



## 1. Introduction

Benchmarking Methodology Working Group (BMWG) has traditionally conducted laboratory characterization of dedicated physical implementations of internetworking functions. The Black-box Benchmarks of Throughput, Latency, Forwarding Rates and others have served our industry for many years. [RFC1242] and [RFC2544] are the cornerstones of the work.

An emerging set of service provider and vendor development goals is to reduce costs while increasing flexibility of network devices, and drastically accelerate their deployment. Network Function Virtualization (NFV) has the promise to achieve these goals, and therefore has garnered much attention. It now seems certain that some network functions will be virtualized following the success of cloud computing and virtual desktops supported by sufficient network path capacity, performance, and widespread deployment; many of the same techniques will help achieve NFV.

See http://www.etsi.org/technologies-clusters/technologies/nfv for more background, for example, the white papers there may be a useful starting place. The Performance and Portability Best Practices [NFV.PER001] are particularly relevant to BMWG. There are currently work-in-progress documents available in the Open Area http://docbox.etsi.org/ISG/NFV/Open/Latest_Drafts/ including drafts describing Infrastructure aspects and service quality.

## 2. Scope

BMWG will consider the new topic of Virtual Network Functions and related Infrastructure to ensure that common issues are recognized from the start, using background materials from industry and SDOs (e.g., IETF, ETSI NFV).

This memo investigates additional methodological considerations necessary when benchmarking VNF instantiated and hosted in commodity off-the-shelf (COTS) hardware. An essential consideration is benchmarking both physical and virtual network functions, thereby allowing direct comparison.

A clearly related goal: the benchmarks for the capacity of COTS to host a plurality of VNF instances should be investigated. Existing networking technology benchmarks will also be considered for adaptation to NFV and closely associated technologies.

A non-goal is any overlap with traditional computer benchmark development and their specific metrics (SPECmark suites such as SPECCPU).

A colossal non-goal is any form of architecture development related to NFV and associated technologies in BMWG, as has been the case since BMWG began work in 1989.

## 3. Considerations for Hardware and Testing

This section lists the new considerations which must be addressed to benchmark VNF(s) and their supporting infrastructure.

### 3.1. Hardware Components

New Hardware devices will become part of the test set-up.

1. High volume server platforms (COTS, possibly with virtual technology enhancements).

2. Storage systems with large capacity, high speed, and high reliability.

3. Network Interface ports specially designed for efficient service of many virtual NICs.

4. High capacity Ethernet Switches.

Labs conducting comparisons of different VNFs may be able to use the same hardware platform over many studies, until the steady march of innovations overtakes their capabilities (as happens with the lab’s traffic generation and testing devices today).

### 3.2. Configuration Parameters

It will be necessary to configure and document the settings for the entire COTS platform, including:

- number of server blades (shelf occupation)
- CPUs
- caches
- storage system
- I/O

as well as configurations that support the devices which host the VNF itself:

- Hypervisor
- Virtual Machine
- Infrastructure Virtual Network

and finally, the VNF itself, with items such as:

- specific function being implemented in VNF
- number of VNF components in the service function chain
- number of physical interfaces and links transited in the service function chain

### 3.3. Testing Strategies

The concept of characterizing performance at capacity limits may change. For example:

1. It may be more representative of system capacity to characterize the case where Virtual Machines (VM, hosting the VNF) are operating at 50% Utilization, and therefore sharing the "real" processing power across many VMs.

2. Another important case stems from the need for partitioning functions. A noisy neighbor (VM hosting a VNF in an infinite loop) would ideally be isolated and the performance of other VMs would continue according to their specifications.

3. System errors will likely occur as transients, implying a distribution of performance characteristics with a long tail (like latency), leading to the need for longer-term tests of each set of configuration and test parameters.

4. The desire for Elasticity and flexibility among network functions will include tests where there is constant flux in the VM instances. Requests for new VMs and Releases for VMs hosting VNFs no longer needed would be an normal operational condition.

5. All physical things can fail, and benchmarking efforts can also examine recovery aided by the virtual architecture with different approaches to resiliency.

### 3.4. Attention to Shared Resources

Since many components of the new NFV Infrastructure are virtual, test set-up design must have prior knowledge of inter-actions/dependencies within the various resource domains in the System Under Test (SUT). For example, a virtual machine performing the role of a traditional tester function such as generating and/or receiving traffic should avoid sharing any SUT resources with the Device Under Test DUT. Otherwise, the results will have unexpected dependencies not encountered in physical device benchmarking. The shared-resource

aspect of test design remains one of the critical challenges to overcome in a reasonable way to produce useful results.

## 4. Benchmarking Considerations

This section discusses considerations related to Benchmarks applicable to VNFs and their associated technologies.

### 4.1. Comparison with Physical Network Functions

In order to compare the performance of virtual designs and implementations with their physical counterparts, identical benchmarks must be used. Since BMWG has developed specifications for many network functions already, there will be re-use of existing benchmarks through references, while allowing for the possibility of benchmark curation during development of new methodologies. Consideration should be given to quantifying the number of parallel VNFs required to achieve comparable performance with a given physical device, or whether some limit of scale was reached before the VNFs could achieve the comparable level.

### 4.2. Continued Emphasis on Black-Box Benchmarks

When the network functions under test are based on Open Source code, there may be a tendency to rely on internal measurements to some extent, especially when the externally-observable phenomena only support an inference of internal events (such as routing protocol convergence). However, external observations remain essential as the basis for Benchmarks. Internal observations with fixed specification and interpretation may be provided in parallel, to assist the development of operations procedures when the technology is deployed, for example. Internal metrics and measurements from Open Source implementations may be the only direct source of performance results in a desired dimension, but corroborating external observations are still required to assure the integrity of measurement discipline was maintained for all reported results.

A related aspect of benchmark development is where the scope includes multiple approaches to a common function under the same benchmark. For example, there are many ways to arrange for activation of a network path between interface points and the activation times can be compared if the start-to-stop activation interval has a generic and unambiguous definition. Thus, generic benchmark definitions are preferred over technology/protocol specific definitions where possible.

### 4.3. New Benchmarks and Related Metrics

There will be new classes of benchmarks needed for network design and assistance when developing operational practices (possibly automated management and orchestration of deployment scale). Examples follow in the paragraphs below, many of which are prompted by the goals of increased elasticity and flexibility of the network functions, along with accelerated deployment times.

Time to deploy VNFs: In cases where the COTS hardware is already deployed and ready for service, it is valuable to know the response time when a management system is tasked with "standing-up" 100’s of virtual machines and the VNFs they will host.

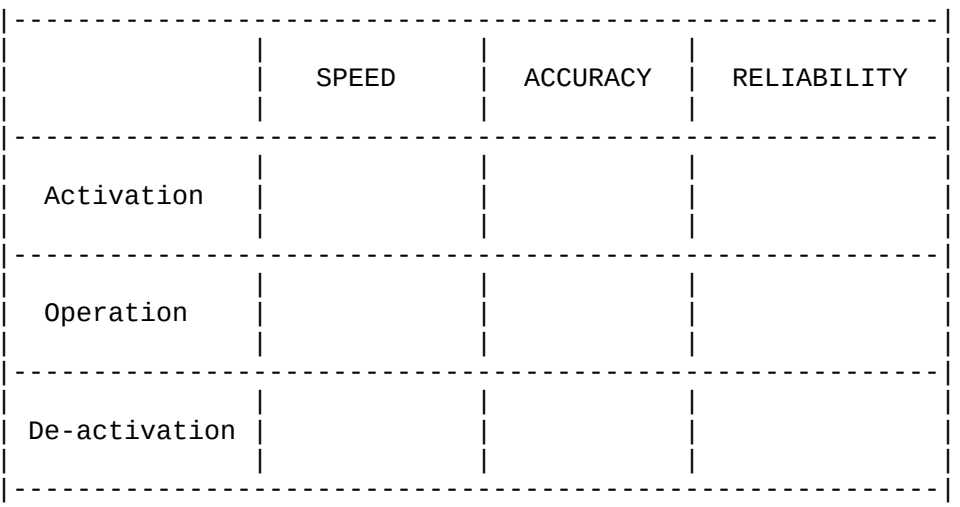
Time to migrate VNFs: In cases where a rack or shelf of hardware must be removed from active service, it is valuable to know the response time when a management system is tasked with "migrating" some number of virtual machines and the VNFs they currently host to alternate hardware that will remain in-service.

Time to create a virtual network in the COTS infrastructure: This is a somewhat simplified version of existing benchmarks for convergence time, in that the process is initiated by a request from (centralized or distributed) control, rather than inferred from network events (link failure). The successful response time would remain dependent on dataplane observations to confirm that the network is ready to perform.

Also, it appears to be valuable to measure traditional packet transfer performance metrics during the assessment of traditional and new benchmarks, including metrics that may be used to support service engineering such as the Spatial Composition metrics found in [RFC6049]. Examples include Mean one-way delay in section 4.1 of [RFC6049], Packet Delay Variation (PDV) in [RFC5481], and Packet Reordering [RFC4737] [RFC4689].

### 4.4. Assessment of Benchmark Coverage

It can be useful to organize benchmarks according to their applicable lifecycle stage and the performance criteria they intend to assess. The table below provides a way to organize benchmarks such that there is a clear indication of coverage for the intersection of lifecycle stages and performance criteria.




For example, the "Time to deploy VNFs" benchmark described above would be placed in the intersection of Activation and Speed, making it clear that there are other potential performance criteria to benchmark, such as the "percentage of unsuccessful VM/VNF stand-ups" in a set of 100 attempts. This example emphasizes that the Activation and De-activation lifecycle stages are key areas for NFV and related infrastructure, and encourage expansion beyond traditional benchmarks for normal operation. Thus, reviewing the benchmark coverage using this table (sometimes called the 3x3 matrix) can be a worthwhile exercise in BMWG.

In one of the first applications of the 3x3 matrix on BMWG, we discovered that metrics on measured size, capacity, or scale do not easily match one of the three columns above. There are three possibilities to resolve this:

- Add a column, Scaleability, but then it would be expected to have metrics in most of the Activation, Operation, and De-activation functions (which may not be the case).

- Include Scalability under Reliability: This fits the user perspective of the 3x3 matrix because the size or capacity of a device contributes to the likelihood that a request will be blocked, or that operation will be un-reliable when operating in an overload state.

- Keep size, capacity, and scale metrics separate from the 3x3 matrix.

After some discussion, including some of the original developers of the 3x3 matrix, it is suggested to keep capacity metrics separate from the 3x3 matrix and list them separately. This approach encourages use of the 3x3 matrix to organize reports of results, where the capacity at which the various metrics were measured would be included in the title of the matrix (and results for multiple capacities would result in separate 3x3 matrices, if there were sufficient measurements/results to organize in that way).

## 5. Security Considerations

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

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

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

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

## 6. IANA Considerations

No IANA Action is requested at this time.

## 7. Acknowledgements

The author acknowledges an encouraging conversation on this topic with Mukhtiar Shaikh and Ramki Krishnan in November 2013. Bhavani Parise and Ilya Varlashkin have provided useful suggestions to expand these considerations. Bhuvaneswaran Vengainathan has already tried the 3x3 matrix with SDN controller draft, and contributed to many discussions. Scott Bradner quickly pointed out shared resource dependencies in an early vSwitch measurement proposal, and the topic was included here as a key consideration.

# 8. References

## 8.1. Normative References

[NFV.PER001] "Network Function Virtualization: Performance and Portability Best Practices", Group Specification ETSI GS NFV-PER 001 V1.1.1 (2014-06), June 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

[RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, March 1999.

[RFC2679] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Delay Metric for IPPM", RFC 2679, September 1999.

[RFC2680] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Packet Loss Metric for IPPM", RFC 2680, September 1999.

[RFC2681] Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, September 1999.

[RFC3393] Demichelis, C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, November 2002.

[RFC3432] Raisanen, V., Grotefeld, G., and A. Morton, "Network performance measurement with periodic streams", RFC 3432, November 2002.

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

[RFC4737] Morton, A., Ciavattone, L., Ramachandran, G., Shalunov, S., and J. Perser, "Packet Reordering Metrics", RFC 4737, November 2006.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, October 2008.

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

## 8.2. Informative References

[RFC1242] Bradner, S., "Benchmarking terminology for network interconnection devices", RFC 1242, July 1991.

[RFC5481] Morton, A. and B. Claise, "Packet Delay Variation Applicability Statement", RFC 5481, March 2009.

[RFC6049] Morton, A. and E. Stephan, "Spatial Composition of Metrics", RFC 6049, January 2011.

[RFC6248] Morton, A., "RFC 4148 and the IP Performance Metrics (IPPM) Registry of Metrics Are Obsolete", RFC 6248, April 2011.

[RFC6390] Clark, A. and B. Claise, "Guidelines for Considering New Performance Metric Development", BCP 170, RFC 6390, October 2011.

## 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
URI: http://home.comcast.net/~acmacm/