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

L. Huang, Ed.
R. Gu, Ed.
China Mobile
Bob. Mandeville
Iometrix
Brooks. Hickman
Spirent Communications
April 28, 2015

Benchmarking Methodology for Virtualization Network Performance
draft-huang-bmwg-virtual-network-performance-01

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 October 30, 2015.

Table of Contents

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

1. Introduction

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

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

2. Terminology

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

3. Test Considerations

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

While in a virtual test environment where both the DUT as well as the test tool itself are 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 is difficult to list the exact combination and parameter settings, the following table attempts to give the most common example how to calibrate a tester before testing a DUT (VSWITCH) 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.

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

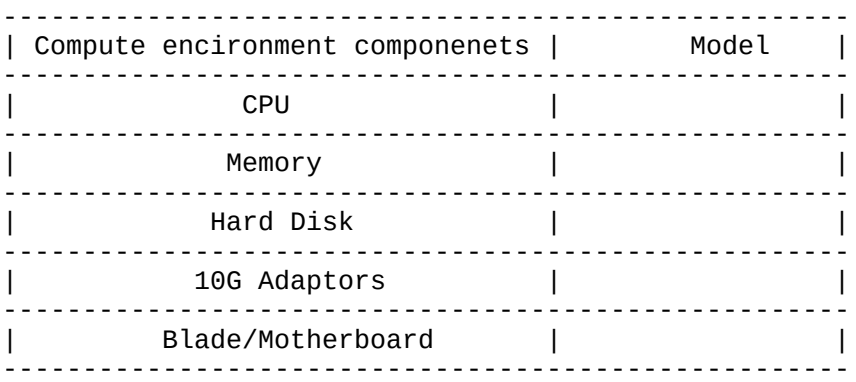


Figure 2: Compute Environment

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

1. One or more VM tester(s) need to be created for both traffic generation and analysis.

2. vSwitch has an influence on performance penalty due to extra 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.

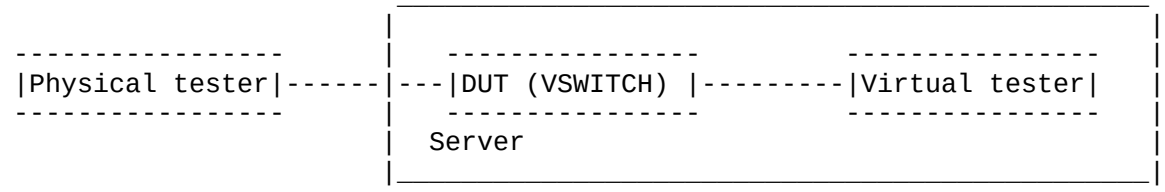


Figure 3: test model A

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

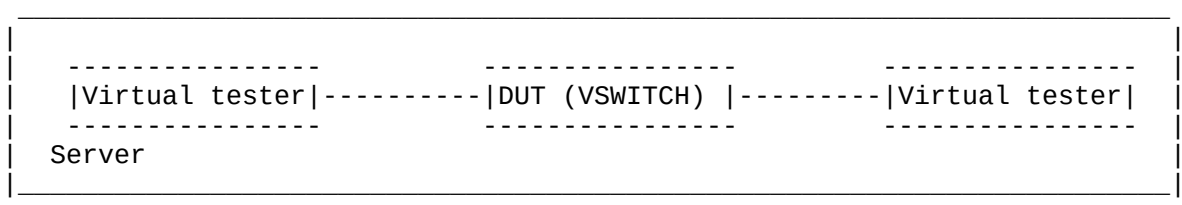


Figure 4: test model B

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

the server. 10GE switch and 1GE switch are used for test traffic and management respectively.

This test setup is also available in the VxLAN measurement.

6. Benchmarking Tests

6.1. Throughput

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

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

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

6.1.1. Objectives

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

6.1.2. Configuration parameters

Network parameters should be defined as follows:

a) the number of virtual tester (VMs)

b) the number of vNIC of virtual tester

c) the CPU type of the server

d) vCPU allocated for virtual tester (VMs)

e) memory allocated for virtual tester (VMs)

f) the number and rate of server NIC

6.1.3. Test parameters

a) test repeated times

b) test frame length

6.1.4. Test process

1. Configure the VM tester to offer traffic to the V-Switch.

2. Increase the number of vCPU in the tester until the traffic has no packet loss.

3. Record the max throughput on VSwitch.

4. Change the frame length and repeat from step1 to step4.

6.1.5. Test result format

Byte	Throughput (Gbps)
64	
128	0.46
256	0.84
512	1.56
1024	2.88
1518	4.00

Figure 5: test result format

6.2. Frame loss rate

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

6.2.1. Objectives

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

6.2.2. Configuration parameters

Network parameters should be defined as follows:

a) the number of virtual tester (VMs)

b) the number of vNIC of virtual tester

c) the CPU type of the server

d) vCPU allocated for virtual tester (VMs)

e) memory allocated for virtual tester (VMs)

f) the number and rate of server NIC

6.2.3. Test parameters

a) test repeated times

b) test frame length

c) test frame rate

6.2.4. Test process

1. Configure the VM tester to offer traffic to the V-Switch with the input frame changing from the maximum rate to the rate with no frame loss at reducing 10% intervals according to RFC 2544.

2. Record the input frame count and output count on VSwitch.

3. Calculate the frame loss percentage under different frame rate.

4. Change the frame length and repeat from step1 to step4.

6.2.5. Test result format

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

Figure 6: test result format

6.3. CPU consumption

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

6.3.1. Objectives

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

6.3.2. Configuration parameters

Network parameters should be defined as follows:

a) the number of virtual tester (VMs)

b) the number of vNIC of virtual tester

c) the CPU type of the server

d) vCPU allocated for virtual tester (VMs)

e) memory allocated for virtual tester (VMs)

f) the number and rate of server NIC

6.3.3. Test parameters

a) test repeated times

b) test frame length

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 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).

4. Change the frame length and repeat from step1 to step4.

6.3.5. Test result format

Byte	Throughput(Gbps)	Server CPU(MHZ)	VM CPU(MHz)
64			
128	0.46	6395	5836
256	0.84	6517	6143
512	1.56	6668	6099
1024	2.88	6280	5726
1518	4.00	6233	5441

Figure 7: test result format

6.4. MEM consumption

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

6.4.1. Objectives

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

6.4.2. Configuration parameters

Network parameters should be defined as follows:

a) the number of virtual tester (VMs)

b) the number of vNIC of virtual tester

c) the CPU type of the server

d) vCPU allocated for virtual tester (VMs)

e) memory allocated for virtual tester (VMs)

f) the number and rate of server NIC

6.4.3. Test parameters

a) test repeated times

b) test frame length

6.4.4. Test process

1. Configure the VM tester to offer traffic to the V-Switch with 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).

4. Change the frame length and repeat from step1 to step4.

6.4.5. Test result format

Byte	Throughput(Gbps)	Host Memory	VM Memory
64			
128	0.46	3040	696
256	0.84	3042	696
512	1.56	3041	696
1024	2.88	3043	696
1518	4.00	3045	696

Figure 8: test result format

6.5. Latency

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

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

We use the traffic model A as the time delay test model by substituting the virtual tester with the echo server, which is used to echo the traffic.Thus the delay time equals to half of the time value between the traffic transmitting by the physical tester and the traffic receiving by the physical tester.

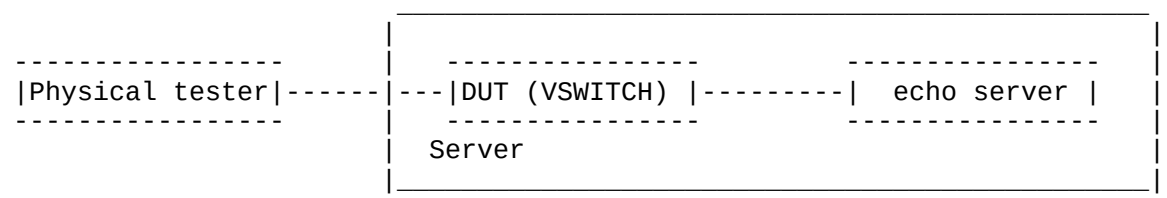


Figure 9: time delay test model

6.5.1. Objectives

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

6.5.2. Configuration parameters

Network parameters should be defined as follows:

a) the number of virtual tester (VMs)

b) the number of vNIC of virtual tester

c) the CPU type of the server

d) vCPU allocated for virtual tester (VMs)

e) memory allocated for virtual tester (VMs)

f) the number and rate of server NIC

6.5.3. Test parameters

a) test repeated times

b) test frame length

6.5.4. Test process

1. Configure the physical tester to offer traffic to the V-Switch with the traffic value of throughput tested in 6.1.

2. Under the same throughput, record the time of transmitting the traffic and receiving the traffic by the physical tester with and without the DUT.

3. Calculate the time difference value between receiving and transmitting the traffic..

4. Calculate the time delay with time difference value with and without the DUT.

5. Change the frame length and repeat from step1 to step4.

6.5.5. Test result format

Byte	Time delay(Gbps)
64	
128	
256	
512	
1024	
1518	

Figure 10: test result format

7. Security Considerations

None.

8. IANA Considerations

None.

9. Normative References

[RFC1242] Bradner, S., "Benchmarking terminology for network interconnection devices", RFC 1242, July 1991.

[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.

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

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

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

Email: bob@iometrix.com

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

Email: Brooks.Hickman@spirent.com

Internet Engineering Task Force
INTERNET-DRAFT, Intended Status: Informational
Expires December 23,2017
June 21, 2017

L. Avramov
Google
J. Rapp
VMware

Data Center Benchmarking Methodology
draft-ietf-bmwg-dcbench-methodology-18

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. A pre-requisite to this publication is the terminology document [draft-ietf-bmwg-dcbench-terminology]. Many of these terms and methods may be applicable beyond this publication's scope as the technologies originally applied in the data center are deployed elsewhere.

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."

Copyright Notice

Table of Contents

1. Introduction 3
1.1. Requirements Language 5
1.2. Methodology format and repeatability recommendation 5
2. Line Rate Testing 5
2.1 Objective 5
2.2 Methodology 5
2.3 Reporting Format 6
3. Buffering Testing 7
3.1 Objective 7
3.2 Methodology 7
3.3 Reporting format 10
4 Microburst Testing 11
4.1 Objective 11
4.2 Methodology 11
4.3 Reporting Format 12
5. Head of Line Blocking 13
5.1 Objective 13
5.2 Methodology 13
5.3 Reporting Format 15
6. Incast Stateful and Stateless Traffic 15
6.1 Objective 15
6.2 Methodology 15
6.3 Reporting Format 17
7. Security Considerations 17
8. IANA Considerations 17
9. References 18
9.1. Normative References 19
9.2. Informative References 19
9.2. Acknowledgements 20
Authors' Addresses 20

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 (server to server inside the data center) in one data center and north-south (outside of the data center to server) in another, while others 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 these can coexist in a single cluster and flow through a single network device simultaneously. Benchmarking of

network devices have long used [RFC1242], [RFC2432], [RFC2544], [RFC2889] and [RFC3918] which have largely been focused around various latency attributes and Throughput [RFC2889] of the Device Under Test (DUT) being benchmarked. These standards are good at measuring theoretical Throughput, forwarding rates and latency under testing conditions; however, they do not represent real traffic patterns that may affect these networking devices.

Currently, typical data center networking devices are characterized by:

-High port density (48 ports of more)

-High speed (up to 100 GB/s currently per port)

-High throughput (line rate on all ports for Layer 2 and/or Layer 3)

-Low latency (in the microsecond or nanosecond range)

-Low amount of buffer (in the MB range per networking device)

-Layer 2 and Layer 3 forwarding capability (Layer 3 not mandatory)

This document provides a methodology for benchmarking Data Center physical network equipment DUT including congestion scenarios, switch buffer analysis, microburst, head of line blocking, while also using a wide mix of traffic conditions. The terminology document [draft-ietf-bmwg-dcbench-terminology] is a pre-requisite.

1.1. Requirements Language

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

1.2. Methodology format and repeatability recommendation

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

-Objective

-Methodology

-Reporting Format

For each test methodology described, it is critical to obtain repeatability in the results. The recommendation is to perform enough iterations of the given test and to make sure the result is consistent. This is especially important for section 3, as the buffering testing has been historically the least reliable. The number of iterations SHOULD be explicitly reported. The relative standard deviation SHOULD be below 10%.

2. Line Rate Testing

2.1 Objective

Provide a maximum rate test for the performance values for Throughput, latency and jitter. It is meant to provide the tests to perform, 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 section 15 compliant] and also in a full mesh type of DUT test [2889/3918 section 16 compliant].

For all tests, the test traffic generator sending rate MUST be less than or equal to 99.98% of the nominal value of Line Rate (with no further PPM adjustment to account for interface clock tolerances), to ensure stressing the DUT in reasonable worst case conditions (see RFC [draft-ietf-bmwg-dcbench-terminology] section 5 for more details -- note to RFC Editor, please replace all [draft-ietf-bmwg-dcbench-

terminology] references in this document with the future RFC number of that draft). 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 SHOULD be captured during this test in % of line rate.

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

The test MUST provide the statistics of minimum, average and maximum of the jitter distribution, 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 Z.

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 [draft-ietf-bmwg-dcbench-terminology] section 4.

-number of ports used

-reading for "Throughput received in percentage of bandwidth", while sending 99.98% of nominal value of Line Rate on each port, for each packet size from 64 bytes to 9216 bytes. As guidance, an increment of 64 byte packet size between each iteration being ideal, a 256 byte and 512 bytes being are also often 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].

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

-For packet drops, they MUST be expressed as a count of packets 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 trials, with proper documentation in the report but SHOULD be conducted at the same time.

-The methodology makes an assumption that the DUT has at least nine ports, as certain methodologies require that number of ports or more.

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 SoC (Switch-on-Chip), 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 inflection 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 condition. 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 inflection point.

1) Measure the highest buffer efficiency

The tests described in this section have iterations called "first iteration", "second iteration" and, "last iteration". The idea is to show the first two iterations so the reader understands the logic on how to keep incrementing the iterations. The last iteration shows the end state of the variables.

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 inflection 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 inflection 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 inflection point multiplied by the frame size.

When the B value is found to provide the largest buffer size, then size B allows the highest buffer efficiency.

2) Measure maximum port buffer size

The tests described in this section have iterations called "first

iteration", "second iteration" and, "last iteration". The idea is to show the first two iterations so the reader understands the logic on how to keep incrementing the iterations. The last iteration shows the end state of the variables.

At fixed packet size B determined in procedure 1), for a fixed default Differentiated Services Code Point (DSCP)/Class of Service (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 DSCP/COS values of traffic and then using Multicast type of traffic, in order to find if there is any DSCP/COS impact on the buffer size.

3) Measure maximum port pair buffer sizes

The tests described in this section have iterations called "first iteration", "second iteration" and, "last iteration". The idea is to show the first two iterations so the reader understands the logic on how to keep incrementing the iterations. The last iteration shows the end state of the variables.

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 DSCP/COS values of traffic and then using Multicast type of traffic.

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

The tests described in this section have iterations called "first iteration", "second iteration" and, "last iteration". The idea is to show the first two iterations so the reader understands the logic on how to keep incrementing the iterations. The last iteration shows the end state of the variables.

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 DSCP/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

- The repeatability of the test needs to be indicated: number of iterations of the same test and percentage of variation between results for each of the tests (min, max, avg)

The percentage of variation is a metric providing a sense of how big the difference between the measured value and the previous ones.

For example, for a latency test where the minimum latency is measured, the percentage of variation of the minimum latency will indicate by how much this value has varied between the current test executed and the previous one.

PV=((x2-x1)/x1)*100 where x2 is the minimum latency value in the current test and x1 is the minimum latency value obtained in the previous test.

The same formula is used for max and avg variations measured.

4 Microburst Testing

4.1 Objective

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

This test provides additional methodology to the other RFC tests:

-All bursts should be send with 100% intensity. Note: intensity is defined in [draft-ietf-bmwg-dcbench-terminology] section 6.1.1

-All ports of the DUT must be used for this test

-All ports are recommended to be testes simultaneously

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 send bursts of 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 sent with an intensity of 100% (intensity is defined in [draft-ietf-bmwg-dcbench-terminology] section 6.1.1), meaning the burst of packets will be sent with a minimum inter-packet gap. The amount of packet contained in the burst will be trial 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. Intensity of microburst is defined in [draft-ietf-bmwg-dcbench-terminology].

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 number 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

- The repeatability of the test needs to be indicated: number of iterations of the same test and percentage of variation between results (min, max, avg)

5. Head of Line Blocking

5.1 Objective

Head-of-line blocking (HOLB) 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.

Here are the differences between this HOLB test and RFC 2889:

-This HOLB starts with 8 ports in two groups of 4, instead of 4 RFC 2889

-This HOLB shifts all the port numbers by one in a second iteration of the test, this is new compared to RFC 2889. The shifting port numbers continue until all ports are the first in the group. The purpose is to make sure to have tested all permutations to cover differences of behavior in the SoC of the DUT

-Another test in this HOLB expands the group of ports, such that traffic is divided among 4 ports instead of two (25% instead of 50% per port)

-Section 5.3 adds additional reporting requirements from Congestion Control in RFC 2889

5.2 Methodology

In order to cause congestion in the form of 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 on the flow for the first egress port which is not over-subscribed.

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

The tests described in this section have iterations called "first iteration", "second iteration" and, "last iteration". The idea is to show the first two iterations so the reader understands the logic on how to keep incrementing the iterations. The last iteration shows the end state of the variables.

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

The tests described in this section have iterations called "first iteration", "second iteration" and, "last iteration". The idea is to

show the first two iterations so the reader understands the logic on how to keep incrementing the iterations. The last iteration shows the end state of the variables.

The traffic from ingress split across 4 egress ports (100/4=25%).

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 in accordance with the HOLB test in section 5

- Percent of traffic loss

- The repeatability of the test needs to be indicated: number of iteration of the same test and percentage of variation between results (min, max, avg)

6. Incast Stateful and Stateless Traffic

6.1 Objective

The objective of this test is to measure the values for TCP Goodput [1] 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. Stateful flows are created by generating TCP traffic and, stateless flows are created using UDP type of traffic.

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.

The tests described bellow have iterations called "first iteration", "second iteration" and, "last iteration". The idea is to show the first two iterations so the reader understands the logic on how to keep incrementing the iterations. The last iteration shows the end state of the variables.

For example:

Stateful Traffic port variation (TCP traffic):

TCP traffic needs to be generated in this section. 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 Port

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

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

Stateless Traffic port variation (UDP traffic):

UDP traffic needs to be generated for this test. During Iterations, the 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 Port

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

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

6.3 Reporting Format

The report MUST include the following:

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

- Stateful flow goodput

- Stateless flow latency

- The repeatability of the test needs to be indicated: number of iterations of the same test and percentage of variation between results (min, max, avg)

7. Security Considerations

Benchmarking activities as described in this memo are limited to technology characterization using controlled stimuli in a laboratory environment, with dedicated address space and the 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.

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

8. IANA Considerations

NO IANA Action is requested at this time.

9. References

9.1. Normative References

[RFC1242] Bradner, S. "Benchmarking Terminology for Network Interconnection Devices", BCP 14, RFC 1242, DOI 10.17487/RFC1242, July 1991, <http://www.rfc-editor.org/info/rfc1242>

[RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", BCP 14, RFC 2544, DOI 10.17487/RFC2544, March 1999, <http://www.rfc-editor.org/info/rfc2544>

9.2. Informative References

[draft-ietf-bmwg-dcbench-terminology] Avramov L. and Rapp J., "Data Center Benchmarking Terminology", April 2017, RFC "draft-ietf-bmwg-dcbench-terminology", Date [to be fixed when the RFC is published and 1 to be replaced by the RFC number

[RFC2889] Mandeville R. and Perser J., "Benchmarking Methodology for LAN Switching Devices", RFC 2889, August 2000, <http://www.rfc-editor.org/info/rfc2889>

[RFC3918] Stopp D. and Hickman B., "Methodology for IP Multicast Benchmarking", RFC 3918, October 2004, <http://www.rfc-editor.org/info/rfc3918>

[RFC 6985] A. Morton, "IMIX Genome: Specification of Variable Packet Sizes for Additional Testing", RFC 6985, July 2013, <http://www.rfc-editor.org/info/rfc6985>

[1] Yanpei Chen, Rean Griffith, Junda Liu, Randy H. Katz, Anthony D. Joseph, "Understanding TCP Incast Throughput Collapse in Datacenter Networks, "http://yanpeichen.com/professional/usenixLoginIncastReady.pdf"

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

[RFC2432] Dubray, K., "Terminology for IP Multicast Benchmarking", BCP 14, RFC 2432, DOI 10.17487/RFC2432, October 1998, <http://www.rfc-editor.org/info/rfc2432>

9.2. Acknowledgements

The authors would like to thank Alfred Morton and Scott Bradner for their reviews and feedback.

Authors' Addresses

Lucien Avramov
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
United States
Phone: +1 408 774 9077
Email: lucien.avramov@gmail.com

Jacob Rapp
VMware
3401 Hillview Ave
Palo Alto, CA
United States
Phone: +1 650 857 3367
Email: jrapp@vmware.com

Internet Engineering Task Force
INTERNET-DRAFT, Intended status: Informational
Expires: December 24,2017
June 22, 2017

L. Avramov
Google
J. Rapp
VMware

Data Center Benchmarking Terminology
draft-ietf-bmwg-dcbench-terminology-19

Abstract

The purpose of this informational document is to establish definitions and describe measurement techniques for data center benchmarking, as well as it is to introduce new terminologies applicable to performance evaluations of data center network equipment. This document establishes the important concepts for benchmarking network switches and routers in the data center and, is a pre-requisite to the test methodology publication [draft-ietf-bmwg-dcbench-methodology]. Many of these terms and methods may be applicable to network equipment beyond this publication's scope as the technologies originally applied in the data center are deployed elsewhere.

Table of Contents

1. Introduction 3
1.1. Requirements Language 4
1.2. Definition format 4
2. Latency 4
2.1. Definition 4
2.2 Discussion 6
2.3 Measurement Units 6
3 Jitter 6
3.1 Definition 6
3.2 Discussion 7
3.3 Measurement Units 7
4 Physical Layer Calibration 7
4.1 Definition 7
4.2 Discussion 8
4.3 Measurement Units 8
5 Line rate 8
5.1 Definition 8
5.2 Discussion 9
5.3 Measurement Units 10
6 Buffering 11
6.1 Buffer 11
6.1.1 Definition 11
6.1.2 Discussion 12
6.1.3 Measurement Units 12
6.2 Incast 13
6.2.1 Definition 13
6.2.2 Discussion 14
6.2.3 Measurement Units 14
7 Application Throughput: Data Center Goodput 14
7.1. Definition 14
7.2. Discussion 14
7.3. Measurement Units 15
8. Security Considerations 16
9. IANA Considerations 16
10. References 16
10.1. Normative References 16
10.2. Informative References 17
10.3. Acknowledgments 17

Authors' Addresses 17

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 (server to server inside the data center) in one data center and north-south (outside of the data center to server) 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. One or more of these may coexist in a single cluster and flow through a single network device simultaneously. 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 they do not represent real traffic patterns that may affect these networking devices. The data center networking devices covered are switches and routers.

Currently, typical data center networking devices are characterized by:

-High port density (48 ports of more)

-High speed (up to 100 GB/s currently per port)

-High throughput (line rate on all ports for Layer 2 and/or Layer 3)

-Low latency (in the microsecond or nanosecond range)

-Low amount of buffer (in the MB range per networking device)

-Layer 2 and Layer 3 forwarding capability (Layer 3 not mandatory)

The following document 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. The test methodologies are defined in [draft-ietf-bmwg-dcbench-methodology].

1.1. Requirements Language

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

1.2. Definition format

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

Definition: The specific definition for the term.

Discussion: A brief discussion about the term, its 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 Device Under Test (DUT). Latency is measured in units of time (seconds, milliseconds, microseconds and so on). The purpose of measuring latency is to understand the impact of adding a device in the communication path.

The Latency interval can be assessed between different combinations of events, regardless of the type of switching device (bit forwarding aka cut-through, or store-and-forward type of device). [RFC1242] defined Latency differently for each of these types of devices.

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. [RFC1242] Latency for bit forwarding devices uses these events.

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. [RFC1242] Latency for bit forwarding devices uses these events.

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.

To accommodate both types of network devices and hybrids of the two types that have emerged, switch Latency measurements made according to this document MUST be measured with the FILO events. 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 (or RFC 1242 Latency for bit forwarding devices) MAY be used. In all cases, the event combination used in Latency measurement MUST be reported.

2.2 Discussion

As mentioned in section 2.1, FILO is the most important measuring definition.

Not all DUTs are exclusively cut-through or store-and-forward. Data Center DUTs are frequently store-and-forward for smaller packet sizes and then adopting a cut-through behavior. The change of behavior happens at specific larger packet sizes. The value of the packet size for the behavior to change MAY be configurable depending on the DUT manufacturer. FILO covers all scenarios: Store-and-forward or cut-through. The threshold of behavior change does not matter for benchmarking since FILO covers both possible scenarios.

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 because LIFO removes the serialization delay. 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 follows:

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

2) FIFO MAY be used for certain applications able to proceed the data as the first bits arrive, as for example for a Field-Programmable Gate Array (FPGA)

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

3 Jitter

3.1 Definition

Jitter in the data center context is synonymous with the common term Delay variation. It is derived from multiple measurements of one-way delay, as described in RFC 3393. The mandatory definition of Delay Variation is the Packet Delay Variation (PDV) from section 4.2 of [RFC5481]. When considering a stream of packets, the delays of all packets are subtracted from the minimum delay over all packets in the stream. This facilitates assessment of the range of delay variation

(Max - Min), or a high percentile of PDV (99th percentile, for robustness against outliers).

When First-bit to Last-bit timestamps are used for Delay measurement, then Delay Variation MUST be measured using packets or frames of the same size, since the definition of latency includes the serialization time for each packet. Otherwise if using First-bit to First-bit, the size restriction does not apply.

3.2 Discussion

In addition to PDV Range and/or a high percentile of PDV, Inter-Packet Delay Variation (IPDV) as defined in section 4.1 of [RFC5481] (differences between two consecutive packets) MAY be used for the purpose of determining how packet spacing has changed during transfer, for example, to see if packet stream has become closely-spaced or "bursty". However, the Absolute Value of IPDV SHOULD NOT be used, as this collapses the "bursty" and "dispersed" sides of the IPDV distribution together.

3.3 Measurement Units

The measurement of delay variation is expressed in units of seconds. A PDV histogram MAY be provided for the population of packets measured.

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 tests 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 Link Layer Discovery Protocol, 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 measured, therefore 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 4.1 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 "nominal value of Line Rate" 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.

Line rate SHOULD be measured in frames per second.

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. These adjustments SHOULD be used to compensate the test equipment in order to not send more than the 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 allow a software-controlled +/- 100 PPM "offset" (clock-rate adjustment) to compensate for normal variations in the clock speed of DUTs. 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)

Minimum_Gap represents the inter frame gap. This formula "scales up" or "scales down" to represent 1 GB Ethernet, or 10 GB Ethernet and so on.

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 be 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 B (byte); KB (kilobyte), MB (megabyte) or GB (gigabyte). When the buffer size is expressed it SHOULD be defined by a size metric stated 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 (class of service) or dscp (differentiated services code point) 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 byte frames is 18 MB.

Port Buffer Size: The port buffer size is the amount of buffer for a single ingress port, egress port or combination of ingress and egress buffering location for a single port. The reason for mentioning the three locations for the port buffer is because the DUT buffering scheme can be unknown or untested, and so knowing the buffer location helps clarify the buffer architecture and consequently 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: In most cases, this is the same as the Port Buffer Size. In certain switch architecture called SoC (switch on chip), there is a port buffer and a shared buffer pool available for all ports. The Maximum Port Buffer Size , in terms of an SoC buffer, represents the sum of the port buffer and the maximum value of shared buffer allowed for this port, defined in terms of B (byte), KB (kilobyte), MB (megabyte), or GB (gigabyte). 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 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 B frame size packets. The DUT has a Maximum Buffer Size of 18 MB at 1500 B 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

The above definitions are not meant to comment on the ideal sizing of a buffer, rather on how to measure it. A larger buffer is not necessarily better and can cause issues with buffer bloat.

6.1.2 Discussion

When measuring buffering on a DUT, it is important to understand the behavior for each and all ports. This provides data for the total amount of buffering available on the switch. The terms of buffer efficiency here helps one understand the optimum packet size for the buffer, or the real volume of the buffer available for a specific packet size. This section does not discuss how to conduct the test methodology; instead, it 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 traffic patterns. It measures the number of ingress and egress ports and the level of synchronization attributed, as defined in this section. Typically in the data center it would refer to many different ingress server ports (many), sending traffic to a common uplink (many-to-one), or multiple uplinks (many-to-many). This pattern is generalized for any network as many incoming ports sending traffic to one or few uplinks.

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 Device Under Test (DUT), then the frames L1 and L2 have a synchronous arrival times. This is called Incast regardless of in many-to-one (simpler form) or, many-to-many.

Asynchronous arrival time: Any condition not defined by synchronous arrival time.

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 TCP.

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

6.2.2 Discussion

In this scenario, buffers are solicited on the DUT. In an 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 case, 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 captured by the Goodput [4]. Goodput is the application-level throughput. For standard TCP applications, a very small loss can have a dramatic effect on application throughput. [RFC2647] has a definition of Goodput; the definition in this publication is a variance.

Goodput is the number of bits per unit of time forwarded to the correct destination interface of the DUT, minus any bits retransmitted.

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

The Goodput, G, is then measured by the following formula:

G=(S/F) x V bytes per second

-S represents the payload bytes, which does not include packet or TCP headers

-F is the frame size

-V is the speed of the media in 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 1500B when the standard MTU (Maximum Transmission Unit) is used. Each packet requires 20B of IP header information and 20B of TCP header information; therefore 1460B are available per packet for the file transfer. Linux based systems are further limited to 1448B as they also carry a 12B timestamp. Finally, the date is transmitted in this example over Ethernet which adds a 26B overhead per packet.

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

Please note: This example does not take into consideration the 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 following information:

-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. 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.

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

9. IANA Considerations

NO IANA Action is requested at this time.

10. References

10.1. Normative References

[draft-ietf-bmwg-dcbench-methodology] Avramov L. and Rapp J., "Data Center Benchmarking Methodology", RFC "draft-ietf-bmwg-dcbench-methodology", DATE (to be updated once published)

[RFC1242] Bradner, S. "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, July 1991, <http://www.rfc-editor.org/info/rfc1242>

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

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

[RFC5481] , Morton, A., "Packet Delay Variation Applicability Statement", BCP 14, RFC 5481, March 2009, <http://www.rfc-editor.org/info/rfc5481>

10.2. Informative References

[RFC2889] Mandeville R. and Perser J., "Benchmarking Methodology for LAN Switching Devices", RFC 2889, August 2000, <http://www.rfc-editor.org/info/rfc2889>

[RFC3918] Stopp D. and Hickman B., "Methodology for IP Multicast Benchmarking", RFC 3918, October 2004, <http://www.rfc-editor.org/info/rfc3918>

[4] Yanpei Chen, Rean Griffith, Junda Liu, Randy H. Katz, Anthony D. Joseph, "Understanding TCP Incast Throughput Collapse in Datacenter Networks, "http://yanpeichen.com/professional/usenixLoginIncastReady.pdf"

[RFC2432] Dubray, K., "Terminology for IP Multicast Benchmarking", BCP 14, RFC 2432, DOI 10.17487/RFC2432, October 1998, <http://www.rfc-editor.org/info/rfc2432>

[RFC2647] Newman D. ,"Benchmarking Terminology for Firewall Performance" BCP 14, RFC 2647, August 1999, <http://www.rfc-editor.org/info/rfc2647>

10.3. Acknowledgments

The authors would like to thank Alfred Morton, Scott Bradner, Ian Cox, Tim Stevenson for their reviews and feedback.

Authors' Addresses

Lucien Avramov
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
United States
Phone: +1 408 774 9077
Email: lucien.avramov@gmail.com

Jacob Rapp
VMware
3401 Hillview Ave
Palo Alto, CA 94304
United States

Phone: +1 650 857 3367
Email: jrapp@vmware.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 2, 2017

W. Cerveny
Arbor Networks
R. Bonica
R. Thomas
Juniper Networks
March 1, 2017

Benchmarking The Neighbor Discovery Protocol
draft-ietf-bmwg-ipv6-nd-06

Abstract

This document provides benchmarking procedures for Neighbor Discovery Protocol (NDP). It also proposes metrics by which an NDP implementation's scaling capabilities can be measured.

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 September 2, 2017.

Copyright Notice

Table of Contents

1. Introduction 2
2. Test Setup 4
2.1. Device Under Test (DUT) 4
2.1.1. Interfaces 4
2.1.2. Neighbor Discovery Protocol (NDP) 4
2.1.3. Routing 5
2.2. Tester 5
2.2.1. Interfaces 5
2.2.2. Neighbor Discovery Protocol (NDP) 6
2.2.3. Routing 6
2.2.4. Test Traffic 6
2.2.5. Counters 7
3. Tests 8
3.1. Baseline Test 8
3.1.1. Procedure 8
3.1.2. Baseline Test Procedure Flow Chart 8
3.1.3. Results 10
3.2. Scaling Test 10
3.2.1. Procedure 10
3.2.2. Scaling Test Procedure Flow Chart 11
3.2.3. Results 13
4. Measurements Explicitly Excluded 14
4.1. DUT CPU Utilization 14
4.2. Malformed Packets 14
5. IANA Considerations 14
6. Security Considerations 14
7. Acknowledgments 15
8. Normative References 15
Authors' Addresses 15

1. Introduction

When an IPv6 node forwards a packet, it executes the following procedure:

- Identifies the outbound interface and IPv6 next-hop
- Queries a local Neighbor Cache (NC) to determine the IPv6 next-hop's link-layer address

- Encapsulates the packet in a link-layer header. The link-layer header includes the IPv6 next-hop's link-layer address
- Forwards the packet to the IPv6 next-hop

IPv6 nodes use the Neighbor Discovery Protocol (NDP) [RFC4861] to maintain the NC. Operational experience [RFC6583] shows that when an implementation cannot maintain a sufficiently complete NC, its ability to forward packets is impaired.

NDP, like any other protocol, consumes processing, memory, and bandwidth resources. Its ability to maintain a sufficiently complete NC depends upon the availability of the above-mentioned resources.

This document provides benchmarking procedures for NDP. Benchmarking procedures include a Baseline Test and an NDP Scaling Test. In both tests, the Device Under Test (DUT) is an IPv6 router. Two physical links (A and B) connect the DUT to a Tester. The Tester sends traffic through Link A to the DUT. The DUT forwards that traffic, through Link B, back to the Tester.

The above-mentioned traffic stream contains one or more interleaved flows. An IPv6 Destination Address uniquely identifies each flow. Or, said another way, every packet within a flow has the same IPv6 Destination Address.

In the Baseline Test, the traffic stream contains exactly one flow. Because every packet in the stream has the same IPv6 Destination Address, the DUT can forward the entire stream using exactly one NC entry. NDP is exercised minimally and no packet loss should be observed.

The NDP Scaling Test is identical to the Baseline Test, except that the traffic stream contains many flows. In order to forward the stream without loss, the DUT must maintain one NC entry for each flow. If the DUT cannot maintain one NC entry for each flow, packet loss will be observed and attributed to NDP scaling limitations.

This document proposes an NDP scaling metric, called NDP-MAX-NEIGHBORS. NDP-MAX-NEIGHBORS is the maximum number of neighbors to which an IPv6 node can send traffic during periods of high NDP activity.

The procedures described herein reveal how many IPv6 neighbors an NDP implementation can discover. They also provide a rough estimate of the time required to discover those neighbors. However, that estimate does not reflect the maximum rate at which the

implementation can discover neighbors. Maximum rate discovery is a topic for further exploration.

The test procedures described herein assume that NDP does not compete with other applications for resources on the DUT. When NDP competes for resources, its scaling characteristics may differ from those reported by the benchmarks described, and may vary over time.

2. Test Setup

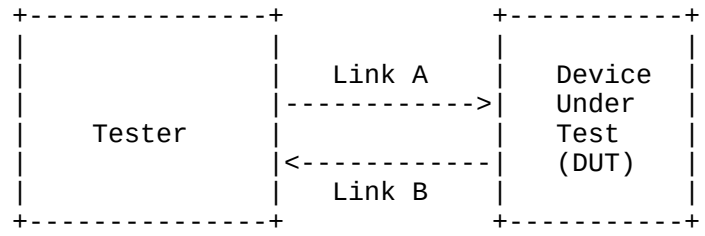


Figure 1: Test Setup

The DUT is an IPv6 router. Two links (A and B) connect the DUT to the Tester. Link A capabilities must be identical to Link B capabilities. For example, if the interface to Link A is a 10 Gigabit Ethernet port, the interface to Link B must also be a 10 Gigabit Ethernet port.

2.1. Device Under Test (DUT)

2.1.1. Interfaces

DUT interfaces are numbered as follows:

- Link A - 2001:2:0:0::2/64
- Link B- 2001:2:0:1::1/64

Both DUT interfaces should be configured with a 1500-byte MTU. However, if they cannot support a 1500-byte MTU, they may be configured with a 1280-byte MTU.

2.1.2. Neighbor Discovery Protocol (NDP)

NDP is enabled on both DUT interfaces. Therefore, the DUT emits both solicited and unsolicited Router Advertisement (RA) messages. The DUT emits an RA message at least once every 600 seconds and no more frequently than once every 200 seconds.

When the DUT sends an RA message, it includes the following information:

- Router Lifetime - 1800 seconds
- Reachable Time - 0 seconds
- Retrans Time - 0 seconds
- Source Link Layer Address - Link layer address of DUT interface
- M-bit is clear (0)
- O-bit is clear (0)

The above-mentioned values are chosen because they are the default values specified in RFC 4861.

NDP manages the NC. Each NC entry represents an on-link neighbor and is identified by the neighbor's on-link unicast IP address. As per RFC 4861, each NC entry needs to be refreshed periodically. NDP refreshes NC entries by exchanging Neighbor Solicitation (NS) and Neighbor Advertisement (NA) messages.

No static NC entries are configured on the DUT.

2.1.3. Routing

The DUT maintains a direct route to 2001:2:0:0/64 through Link A. It also maintains a direct route to 2001:2:0:1/64 through Link B. No static routes or dynamic routing protocols are configured on the DUT.

2.2. Tester

2.2.1. Interfaces

Interfaces are numbered as follows:

- Link A - 2001:2:0:0::1/64
- Link B - Multiple addresses are configured on Link B. These addresses are drawn sequentially from the 2001:2:0:1::/64 address block. The first address is 2001:2:0:1::2/64. Subsequent addresses are 2001:2:0:1::3/64, 2001:2:0:1::4/64, 2001:2:0:1::5/64, et cetera. The number of configured addresses should be the expected value of NDP-MAX-NEIGHBORS times 1.1.

Both Tester interfaces should be configured with a 1500-byte MTU. However, if they cannot support a 1500-byte MTU, they may be configured with a 1280-byte MTU.

2.2.2. Neighbor Discovery Protocol (NDP)

NDP is enabled on both Tester interfaces. Therefore, upon initiation, the Tester sends Router Solicitation (RS) messages and waits for Router Advertisement (RA) messages. The Tester also exchanges Neighbor Solicitation (NS) and Neighbor Advertisement (NA) messages with the DUT.

No static NC entries are configured on the Tester.

2.2.3. Routing

The Tester maintains a direct route to 2001:2:0:0/64 through Link A. It also maintains a direct route to 2001:2:0:1/64 through Link B. No static routes or dynamic routing protocols are configured on the Tester.

2.2.4. Test Traffic

The Tester sends a stream of test traffic through Link A to the DUT. The test traffic stream contains one or more interleaved flows. Flows are numbered 1 through N, sequentially.

Within each flow, each packet contains an IPv6 header and each IPv6 header contains the following information:

- Version - 6
- Traffic Class - 0
- Flow Label - 0
- Payload Length - 0
- Next Header - IPv6-NoNxt (59)
- Hop Limit - 255
- Source Address - 2001:2:0:0::1
- Destination Address - The first 64 bits of the Destination Address are 2001:2:0:1::. The next 64 are uniquely associated with the flow. Every packet in the first flow carries the Destination address 2001:2:0:1::2. Every subsequent flow has an IP address

one greater than the last (i.e., 2001:2:0:1::3, 2001:2:0:1::4, etc.)

In order to avoid link congestion, test traffic is offered at a rate not to exceed 50% of available link bandwidth. In order to avoid burstiness and buffer occupancy, every packet in the stream is exactly 40 bytes long (i.e., the length of an IPv6 header with no IPv6 payload). Furthermore, the gap between packets is identical.

During the course of a test, the number of flows that the test stream contains may increase. When this occurs, the rate at which test traffic is offered remains constant. For example, assume that a test stream is offered at a rate of 1,000 packets per second. This stream contains two flows, each contributing 500 packets per second to the 1,000 packet per second aggregate. When a third stream is added to the flow, all three streams must contribute 333 packets per second in order to maintain the 1,000 packet per second limit. (As in this example, rounding error is acceptable.)

The DUT attempts to forward every packet in the test stream through Link B to the Tester. It does this because:

- Every packet in the test stream has a destination address drawn from the 2001:2:0:1::/64 address block
- The DUT has a direct route to 2001:2:0:1/64 through Link B

2.2.5. Counters

On the Tester, two counters are configured for each flow. One counter, configured on Link A, increments when the Tester sends a packet belonging to the flow. The other counter, configured on Link B, increments when the Tester receives packet from the flow. In order for a packet to be associated with a flow, the following conditions must all be true:

- The IPv6 Destination Address must be that of the flow
- The IPv6 Next Header must be IPv6-NoNxt (59)

The following counters also are configured on both Tester Interfaces:

- RS packets sent
- RA packets received
- NS packets sent

- NS packets received
- NA packets sent
- NA packets received
- Total packets sent
- Total packets received

3. Tests

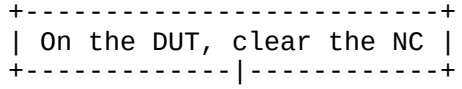
3.1. Baseline Test

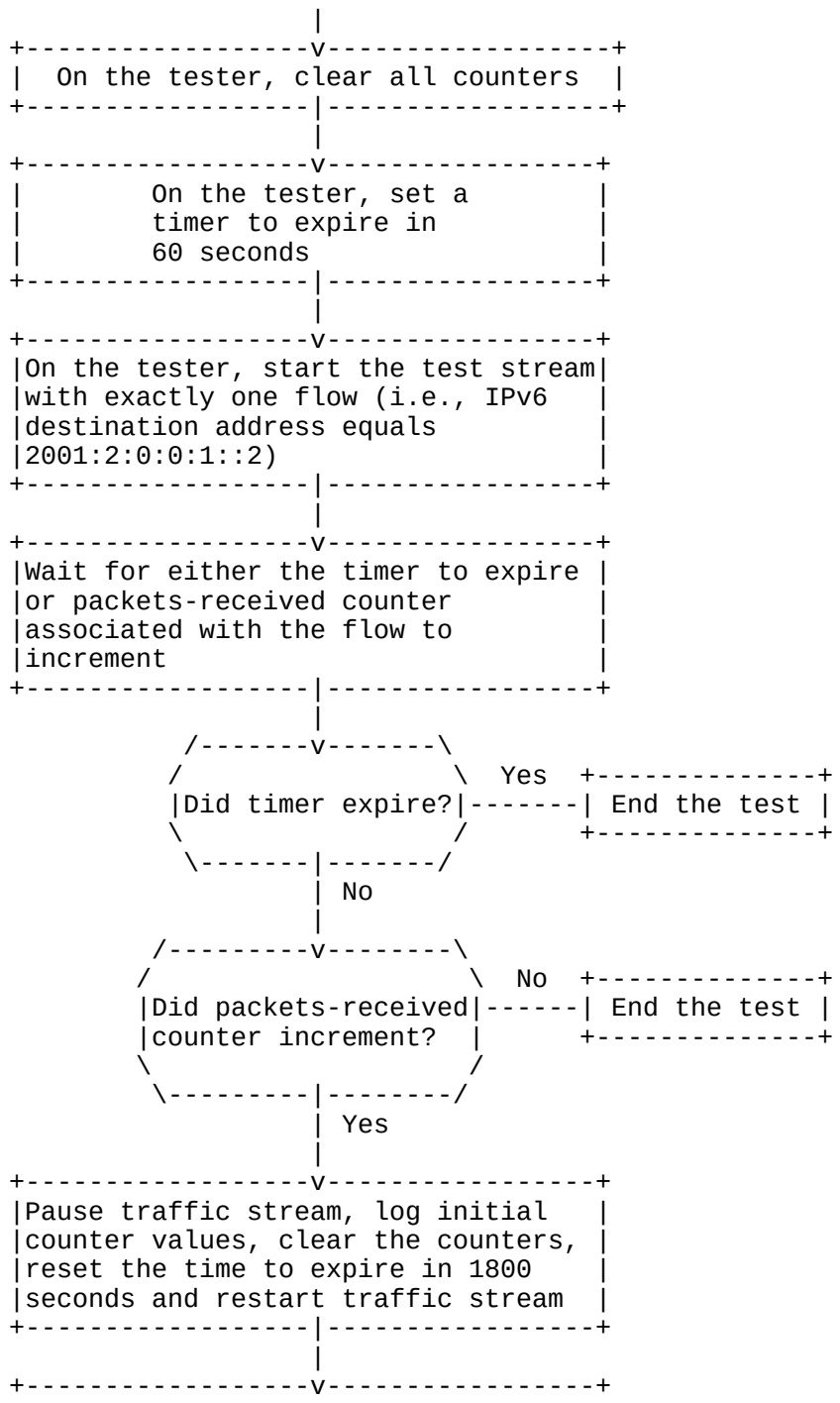
The purpose of the Baseline Test is to ensure that the DUT can forward every packet in the test stream, without loss, when NDP is minimally exercised and not operating near its scaling limit.

3.1.1. Procedure

- On the DUT, clear the NC
- On the Tester, clear all counters
- On the Tester, set a timer to expire in 60 seconds
- On the Tester, start the test stream with exactly one flow (i.e., IPv6 Destination Address equals 2001:2:0:1::2)
- Wait for either the timer to expire or the packets-received counter associated with the flow to increment
- If the timer expires, stop the test stream and end the test
- If the packets-received counter increments, pause the traffic stream, log the initial counter values, clear the counters, reset the timer to expire in 1800 seconds and restart the traffic stream
- When the timer expires, stop the test stream, wait sufficient time for any queued packets to exit, log the final counter values and end the test

3.1.2. Baseline Test Procedure Flow Chart




On the tester, clear all counters
On the tester, set a timer to expire in 60 seconds
On the tester, start the test stream with exactly one flow (i.e., IPv6 destination address equals 2001:2:0:0:1::2)
Wait for either the timer to expire or packets-received counter associated with the flow to increment
Did timer expire?
Yes
End the test
No
Did packets-received counter increment?
No
End the test
Yes
Pause traffic stream, log initial counter values, clear the counters, reset the time to expire in 1800 seconds and restart traffic stream

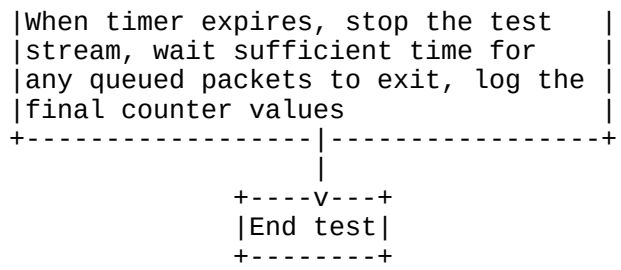


Figure 2: Baseline Test Procedure Flow Chart

3.1.3. Results

The log contains initial and final values for the following counters:

- packets-sent
- packets-received

The initial values of packets-sent and packets-received may be equal to one another. If these values are identical, none of the initial packets belonging to the flow were lost. However, if the initial value of packets-sent is greater than the initial value of packets-received, initial packets were lost. This loss of initial packets is acceptable.

The final values of packets-sent and packets-received should be equal to one another. If they are not, an error has occurred. Because this error is likely to affect Scaling Test results, the error must be corrected before the Scaling Test is executed.

3.2. Scaling Test

The purpose of the Scaling Test is to discover the number of neighbors to which an IPv6 node can send traffic during periods of high NDP activity. We call this number NDP-MAX-NEIGHBORS.

3.2.1. Procedure

Execute the following procedure:

- On the DUT, clear the NC
- On the Tester, clear all counters
- On the Tester, set a timer to expire in 60 seconds

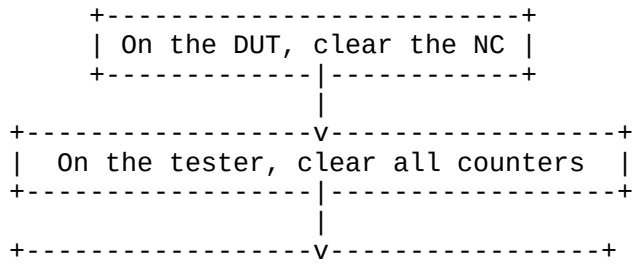
- On the Tester, start the test stream with exactly one flow (i.e., IPv6 Destination Address equals 2001:2:0:1::2)
- Wait for either the timer to expire or the packets-received counter associated with the flow to increment
- If the timer expires, stop the test stream and end the test
- If the packets-received counter increments, proceed as described below:

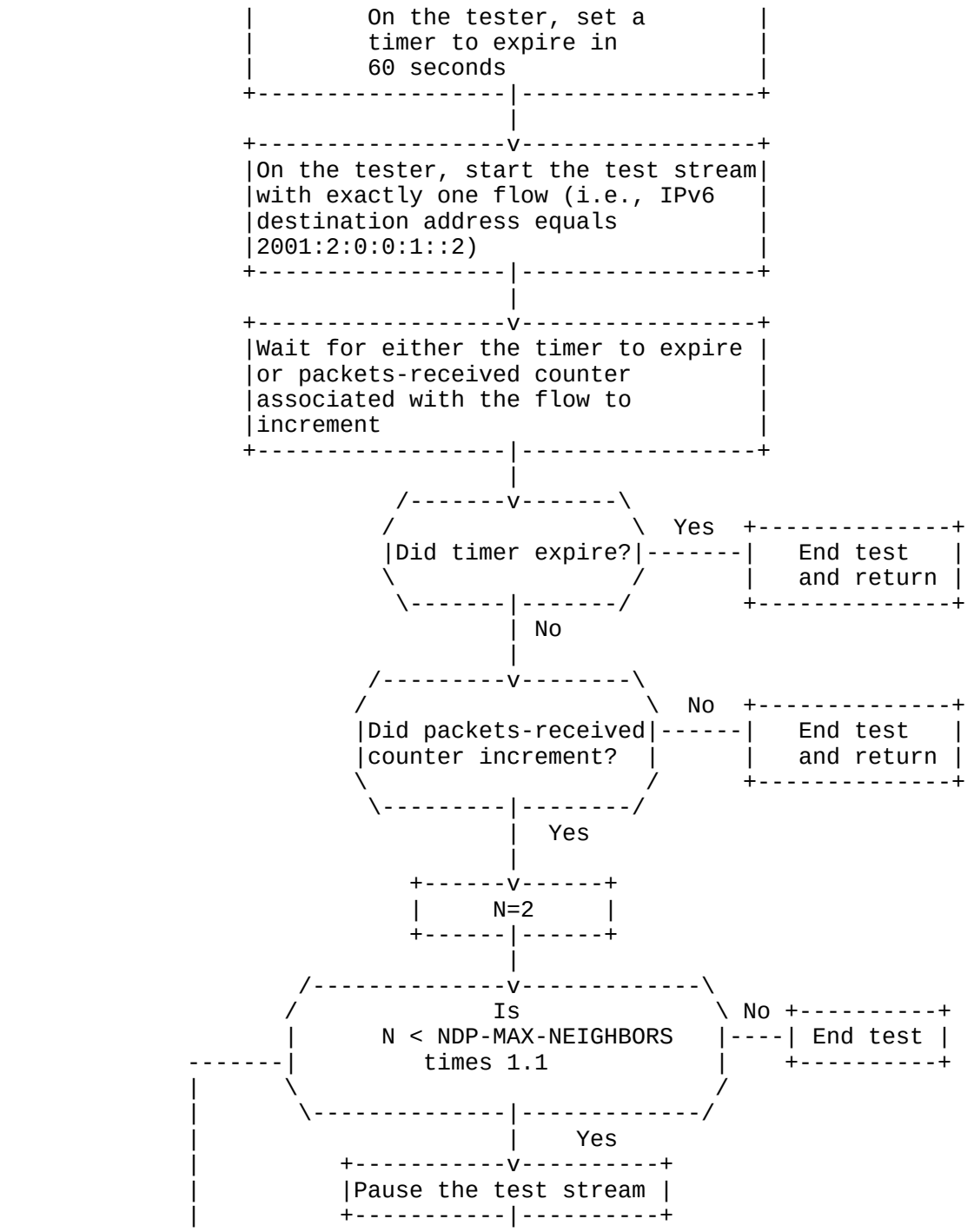
Execute the following procedure N times, starting at 2 and ending at the number of expected value of NDP-MAX-NEIGHBORS times 1.1.

- Pause the test stream
- Log the time and the value of N minus one
- Clear the packets-sent and packets-received counters associated with the previous flow (i.e., N minus one)
- Reset the timer to expire in 60 seconds
- Add the next flow to the test stream (i.e.,IPv6 Destination Address is a function of N)
- Restart the test stream
- Wait for either the timer to expire or the packets-received counter associated with the new flow to increment

After the above described procedure had been executed N times, clear the timer and reset it to expire in 1800 seconds. When the timer expires, stop the stream, log all counters and end the test (after waiting sufficient time for any queued packets to exit).

3.2.2. Scaling Test Procedure Flow Chart




On the tester, set a timer to expire in 60 seconds
On the tester, start the test stream with exactly one flow (i.e., IPv6 destination address equals 2001:2:0:0:1::2)
Wait for either the timer to expire or packets-received counter associated with the flow to increment
Did timer expire?
Yes
End test and return
No
Did packets-received counter increment?
No
End test and return
Yes
N=2
Is N < NDP-MAX-NEIGHBORS times 1.1
No
End test
Yes
Pause the test stream

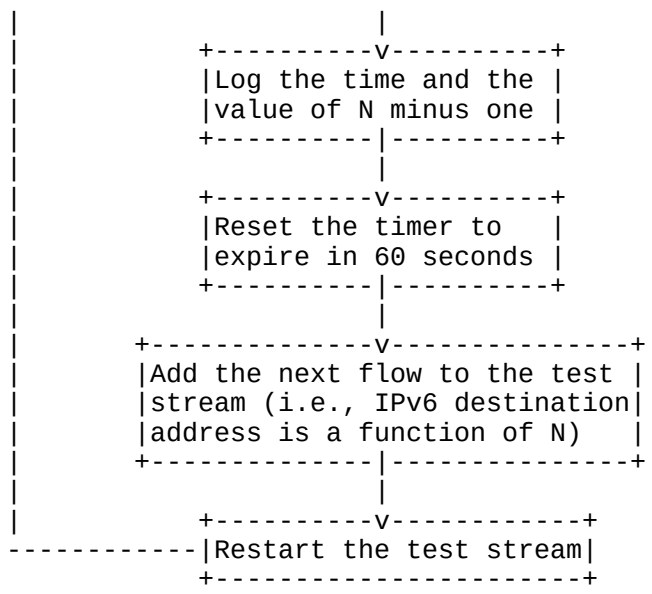


Figure 3: Scaling Test Procedure Flow Chart

3.2.3. Results

The test report includes the following:

- A description of the DUT (make, model, processor, memory, interfaces)
- Rate at which the Tester offers test traffic to the DUT (measured in packets per second)
- A log that records the time at which each flow was introduced to the test stream and the final value of all counters
- The expected value of NDP-MAX-NEIGHBORS
- The actual value of NDP-MAX-NEIGHBORS

NDP-MAX-NEIGHBORS is equal to the number of counter pairs where packets-sent is equal to packets-received. Two counters are members of a pair if they are both associated with the same flow. If packets-sent is equal to packets-recieved for every counter pair, the test should be repeated with a larger expected value of NDP-MAX-NEIGHBORS.

If an implementation abides by the recommendation of Section 7.1 of RFC 6583, for any given counter pair, packets-received will either be equal to zero or packets-sent.

The log documents the time at which each flow was introduced to the test stream. This log reveals the effect of NC size to the time required to discover a new IPv6 neighbor.

4. Measurements Explicitly Excluded

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

4.1. DUT CPU Utilization

This measurement relies on the DUT to provide utilization information, which is not externally observable (not black-box). However, some testing organizations may find the CPU utilization is useful auxiliary information specific to the DUT model, etc.

4.2. Malformed Packets

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

5. IANA Considerations

This document makes no request of IANA.

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

6. 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.

7. Acknowledgments

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.

8. Normative References

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

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

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

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

Reji Thomas
Juniper Networks
Elnath-Exora Business Park Survey
Bangalore, KA 560103
India

Email: rejithomas@juniper.net

Network Working Group
Internet Draft
Intended status: Informational
Expires: April 2016

M. Georgescu
NAIST
G. Lencse
Szechenyi Istvan University
October 15, 2015

Benchmarking Methodology for IPv6 Transition Technologies
draft-ietf-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. More specifically, this document targets IPv6 transition technologies that employ encapsulation or translation mechanisms, as dual-stack nodes can be very well tested using the recommendations of RFC2544 and RFC5180. The methodology also includes a tentative metric for benchmarking load 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 April 15, 2016.

Copyright Notice

Table of Contents

1. Introduction....3
   1.1. IPv6 Transition Technologies....4
2. Conventions used in this document....5
3. Test Setup....5
   3.1. Single translation Transition Technologies....6
   3.2. Encapsulation/Double translation Transition Technologies..6
4. Test Traffic....7
   4.1. Frame Formats and Sizes....7
      4.1.1. Frame Sizes to Be Used over Ethernet....8
   4.2. Protocol Addresses....8
   4.3. Traffic Setup....8
5. Modifiers....9
6. Benchmarking Tests....9
   6.1. Throughput....9
   6.2. Latency....9
   6.3. Packet Delay Variation....9
      6.3.1. PDV....9
      6.3.2. IPDV....10
   6.4. Frame Loss Rate....11
   6.5. Back-to-back Frames....11
   6.6. System Recovery....12
   6.7. Reset....12
7. Additional Benchmarking Tests for Stateful IPv6 Transition Technologies....12

7.1. Concurrent TCP Connection Capacity.....12
7.2. Maximum TCP Connection Establishment Rate.....12
8. DNS Resolution Performance.....13
8.1. Test and Traffic Setup.....13
8.2. Benchmarking DNS Resolution Performance.....14
9. Scalability.....15
9.1. Test Setup.....16
9.1.1. Single Translation Transition Technologies.....16
9.1.2. Encapsulation/Double Translation Transition Technologies.....16
9.2. Benchmarking Performance Degradation.....17
10. Summarizing function and repeatability.....18
11. Security Considerations.....18
12. IANA Considerations.....19
13. Conclusions.....19
14. References.....19
14.1. Normative References.....19
14.2. Informative References.....20
15. Acknowledgements.....20
Appendix A. Theoretical Maximum Frame Rates.....21

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.

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.

The document also includes an approach to quantify load scalability. Load scalability can be defined as a system’s ability to gracefully accommodate higher loads. Because poor scalability usually leads to poor performance, the proposed approach is to quantify the load scalability by measuring the performance degradation created by a higher number of network flows.

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 production network transitioning to IPv6 as being constructed using the following IP domains:

- Domain A: IPvX specific domain
- Core domain: which may be IPvY specific or dual-stack(IPvX and IPvY)
- Domain B: IPvX specific domain

Note: X,Y are part of the {4,6} set.

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

1. Single Translation: In this case, the production network is assumed to have only two domains, Domain A and the Core domain. The core domain is assumed to be IPvY specific. IPvX packets are translated to IPvY at the edge between Domain A and the Core domain.
2. Dual-stack: the core domain devices implement both IP protocols
3. Encapsulation: The production network is assumed to have all three domains, Domains A and B are IPvX specific, while the core domain is IPvY specific. An encapsulation mechanism is used to traverse the core domain. The IPvX packets are encapsulated to IPvY packets at the edge between Domain A and the Core domain. Subsequently, the IPvY packets are decapsulated at the edge between the Core domain and Domain B.
4. Double translation: The production network is assumed to have all three domains, Domains A and B are IPvX specific, while the core domain is IPvY specific. A translation mechanism is employed for the traversal of the core network. The IPvX packets are translated to IPvY packets at the edge between Domain A and the Core domain. Subsequently, the IPvY packets are translated back to IPvX at the edge between the Core domain and Domain B.

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 translation, Encapsulation and Double translation 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 addresses 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.

Although these terms are usually associated with protocol requirements, in this doc the terms are requirements for users and systems that intend to implement the test conditions and claim conformance with this specification.

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] and [RFC5180] can be applied here as well. However, the Device under test (DUT) setup options should be explained in the context of the targeted categories of IPv6 transition technologies: Single translation, Double translation and Encapsulation 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 test setups presented in this memo, dynamic routing SHOULD be employed. 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. Moreover, the "Trial description" recommendations presented in [RFC2544] (Section 23) are valid for this memo as well.

In terms of route setup, the recommendations of [RFC2544] Section 13 are valid for this document as well assuming that an IPv6 version of the routing packets shown in appendix C.2.6.2 is used.

3.1. Single translation Transition Technologies

For the evaluation of Single translation 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.

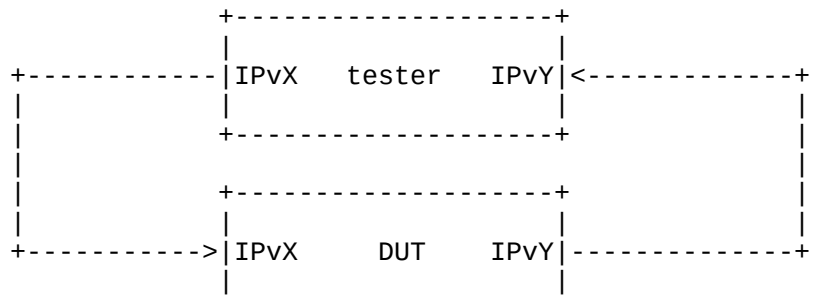


Figure 1. Test setup 1

3.2. Encapsulation/Double translation Transition Technologies

For evaluating the performance of Encapsulation and Double translation transition technologies, a dual DUT setup (see Figure 2) SHOULD be employed. The tester creates a network flow of IPvX packets. The first DUT 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 second DUT and forwarded to the tester.

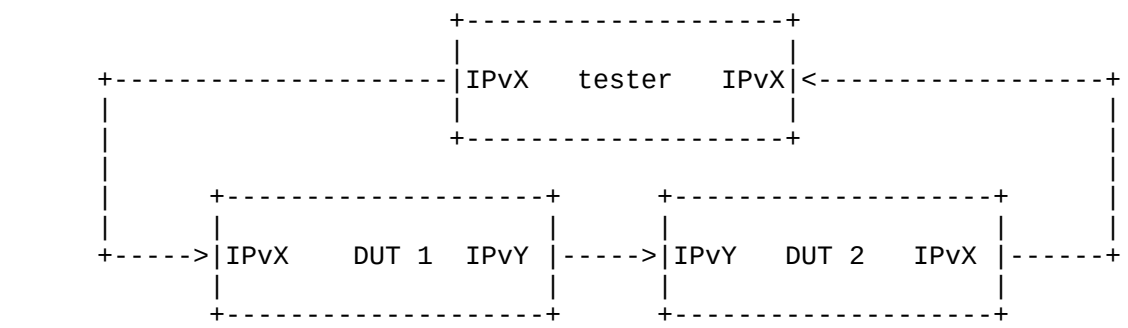


Figure 2. Test setup 2

One of the limitations of the dual DUT setup is the inability to reflect asymmetries in behavior between the DUTs. Considering this, additional performance tests SHOULD be performed using the single DUT setup.

Note: For encapsulation IPv6 transition technologies, in the single DUT setup, in order to test the decapsulation efficiency, the tester SHOULD be able to send IPvX packets encasulated as IPvY.

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 instance, the single translation 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 method for the Ethernet, as well as a calculation example are detailed in Appendix A. The details of the media employed for the benchmarking tests MUST be noted in all test reports.

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. To be more specific, the minimum IPv6 MTU size (1280 bytes) plus the encapsulation/translation overhead is the RECOMMENDED value for the physical interfaces as well as virtual ones.

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 IPvX/IPvY traffic on Ethernet links: 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.

4.2. 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. Proposed IPvX/IPvY translation algorithms such as IP/ICMP translation [RFC6145] do not support the use of extension headers.

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 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. Packet Delay Variation

Considering two of the metrics presented in [RFC5481], Packet Delay Variation (PDV) and Inter Packet Delay Variation (IPDV), it is RECOMMENDED to measure PDV. For a fine grain analysis of delay variation, IPDV measurements MAY be performed as well.

6.3.1. PDV

Objective: To determine the Packet Delay Variation as defined in [RFC5481].

Procedure: As described by [RFC2544], first determine the throughput for the DUT at each of the listed frame sizes. Send a stream of frames at a particular frame size through the DUT at the determined throughput rate to a specific destination. The stream SHOULD be at least 60 seconds in duration. Measure the One-way delay as described by [RFC3393] for all frames in the stream. Calculate the PDV of the stream using the formula:

PDV=D99.9thPercentile - Dmin

Where: D99.9thPercentile - the 99.9th Percentile (as it was described in [RFC5481]) of the One-way delay for the stream

Dmin - the minimum One-way delay in the stream

As recommended in [RFC 2544], the test MUST be repeated at least 20 times with the reported value being the average of the recorded values. Moreover, the margin of error from the average MAY be evaluated following the formula:

$$MoE = alpha * \frac{StDev}{sqrt(N)}$$

Where: alpha - critical value; the recommended value is 2.576 for a 99% level of confidence
StDev - standard deviation
N - number of test iterations

Reporting Format: The PDV results SHOULD be reported in a table with a row for each of the tested frame sizes and columns for the frame size and the applied frame rate for the tested media types. A column for the margin of error values MAY as well be displayed. Following the recommendations of [RFC5481], the RECOMMENDED units of measurement are milliseconds.

6.3.2. IPDV

Objective: To determine the Inter Packet Delay Variation as defined in [RFC5481].

Procedure: As described by [RFC2544], first determine the throughput for the DUT at each of the listed frame sizes. Send a stream of frames at a particular frame size through the DUT at the determined throughput rate to a specific destination. The stream SHOULD be at least 60 seconds in duration. Measure the One-way delay as described by [RFC3393] for all frames in the stream. Calculate the IPDV for each of the frames using the formula:

IPDV(i)=D(i) - D(i-1)

Where: D(i) - the One-way delay of the i th frame in the stream

D(i-1) - the One-way delay of i-1 th frame in the stream

Given the nature of IPDV, reporting a single number might lead to over-summarization. In this context, the report for each measurement SHOULD include 3 values: Dmin, Davg, and Dmax

Where: Dmin - the minimum One-way delay in the stream

Davg - the average One-way delay of the stream

Dmax - the maximum One-way delay in the stream

As recommended in RFC 2544, the test MUST be repeated at least 20 times.

Reporting format: The average of the 3 proposed values SHOULD be reported. The IPDV results SHOULD be reported in a table with a row for each of the tested frame sizes. The columns SHOULD include the frame size and associated frame rate for the tested media types and sub-columns for the three proposed reported values. A column for the margin of error values MAY as well be displayed. Following the recommendations of [RFC5481], the RECOMMENDED units of measurement are milliseconds.

6.4. 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].

6.5. 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].

6.6. 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].

6.7. 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 [RFC6201].

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].

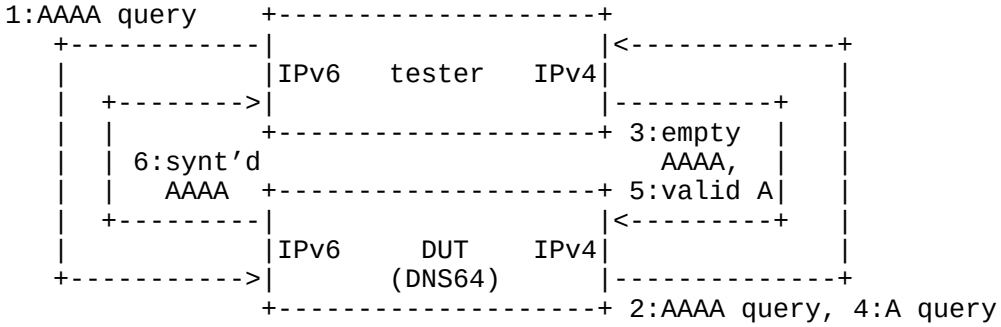
Reporting Format: As described by [RFC3511].

8. DNS Resolution Performance

This section describes benchmarking tests dedicated to DNS64 (see [RFC6147]), used as DNS support for single translation technologies such as NAT64.

8.1. Test and Traffic Setup

The test setup follows the setup proposed for single translation IPv6 transition technologies in Figure 1.



The test traffic SHOULD follow the following steps.

1. Query for the AAAA record of a domain name (from client to DNS64 server)

2. Query for the AAAA record of the same domain name (from DNS64 server to authoritative DNS server)

3. Empty AAAA record answer (from authoritative DNS server to DNS64 server)

4. Query for the A record of the same domain name (from DNS64 server to authoritative DNS server)

5. Valid A record answer (from authoritative DNS server to DNS64 server)

6. Synthesized AAAA record answer (from DNS64 server to client)

The tester plays the role of DNS client as well as authoritative DNS server.

Please note that:

- If the DNS64 server implements caching and there is a cache hit then step 1 is followed by step 6 (and steps 2 through 5 are omitted).
- If the domain name has an AAAA record then it is returned in step 3 by the authoritative DNS server, steps 4 and 5 are omitted, and the DNS64 server does not synthesizes an AAAA record, but returns the received AAAA record to the client.
- As for the IP version used between the tester and the DUT, IPv6 MUST be used between the client and the DNS64 server (as a DNS64 server provides service for an IPv6-only client), but either IPv4 or IPv6 MAY be used between the DNS64 server and the authoritative DNS server.

8.2. Benchmarking DNS Resolution Performance

Objective: To determine DNS64 performance by means of the number of successfully processed DNS requests per second.

Procedure: Send a specific number of DNS queries at a specific rate to the DUT and then count the replies received in time (within a predefined timeout period from the sending time of the corresponding query, having the default value 1 second) from the DUT. If the count of sent queries is equal to the count of received replies, the rate of the queries is raised and the test is rerun. If fewer replies are received than queries were sent, the rate of the queries is reduced and the test is rerun.

The number of processed DNS queries per second is the fastest rate at which the count of DNS replies sent by the DUT is equal to the number of DNS queries sent to it by the test equipment.

The test SHOULD be repeated at least 20 times and the average and margin of error (as described by Section 6.3.1) of the number of processed DNS queries per second SHOULD be calculated.

Details and parameters:

1. Caching
First, all the DNS queries MUST contain different domain names (or domain names MUST NOT be repeated before the cache of the DUT is exhausted). Then new tests MAY be executed with 10%, 20%, 30%, etc.

domain names which are repeated (early enough to be still in the cache).

2. Existence of AAAA record
First, all the DNS queries MUST contain domain names which do not have an AAAA record and have exactly one A record.
Then new tests MAY be executed with 10%, 20%, 30%, etc. domain names which have an AAAA record.

Please note that the two conditions above are orthogonal, thus all their combinations are possible and MAY be tested. The testing with 0% repeated DNS names and with 0% existing AAAA record is REQUIRED and the other combinations are OPTIONAL.

Reporting format: The primary result of the DNS64/DNS46 test is the average of the number of processed DNS queries per second measured with the above mentioned "0% + 0% combination". The average SHOULD be complemented with the margin of error to show the stability of the result. If optional tests are done, the average and margin of error pairs MAY be presented in a two dimensional table where the dimensions are the proportion of the repeated domain names and the proportion of the DNS names having AAAA records. The two table headings SHOULD contain these percentage values. Alternatively, the results MAY be presented as the corresponding two dimensional graph, too. In this case the graph SHOULD show the average values with the margin of error as error bars. From both the table and the graph, one dimensional excerpts MAY be made at any given fixed percentage value of the other dimension. In this case, the fixed value MUST be given together with a one dimensional table or graph.

9. 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.

9.1. Test Setup

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

9.1.1. Single Translation Transition Technologies

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

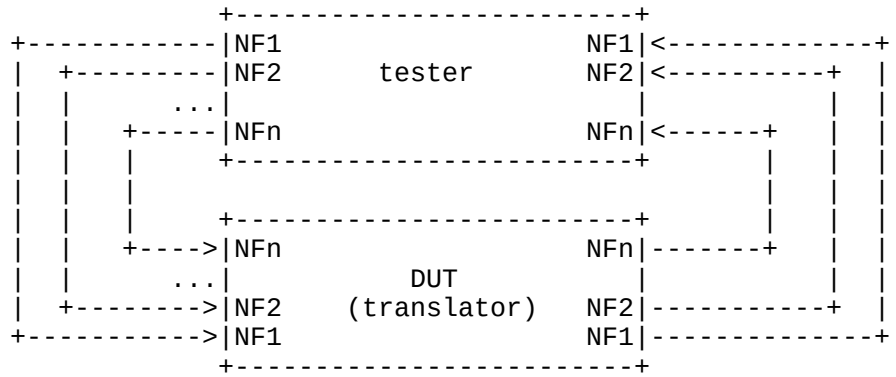


Figure 3. Test setup 3

9.1.2. Encapsulation/Double Translation Transition Technologies

Similarly, for the encapsulation/double translation technologies a multi-flow setup is recommended. Considering a multipoint-to-point scenario, for most transition technologies, one of the edge nodes is designed to support more than one connecting devices. Hence, the recommended test setup is a n:1 design, where n is the number of "client" DUTs connected to the same "server" DUT (See Figure 4).

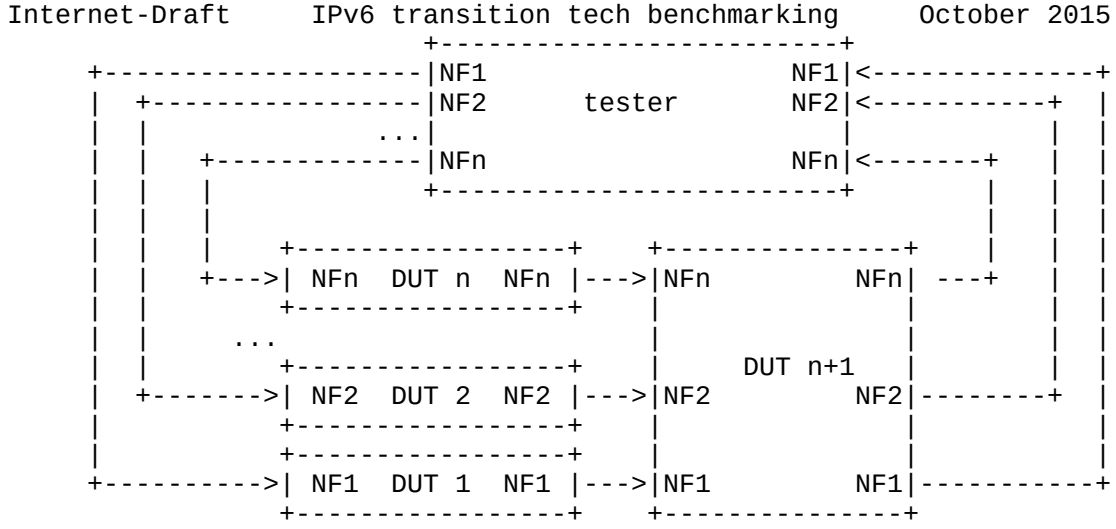


Figure 4. Test setup 4

This test setup can help to quantify the scalability of the "server" device. However, for testing the scalability of the "client" DUTs additional recommendations are needed.
For encapsulation transition technologies a m:n setup can be created, where m is the number of flows applied to the same "client" device and n the number of "client" devices connected to the same "server" device.
For the translation based transition technologies the "client" devices can be separately tested with n network flows using the test setup presented in Figure 3.

9.2. 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:

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

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.

10. Summarizing function and repeatability

To ensure the stability of the benchmarking scores obtained using the tests presented in Sections 6-9, multiple test iterations are recommended. Following the recommendations of RFC2544, the average was chosen to be the summarizing function for the reported values. While median can be an alternative summarizing function, a rationale for using one or the other is needed.

The median can be useful for summarizing especially when outliers are not a desired quantity. However, in the overall performance of a network device the outliers can represent a malfunction or misconfiguration in the DUT, which should be taken into account.

The average is a more inclusive summarizing function. Moreover, as underlined in [DeNijs], the average is less exposed to statistical uncertainty. These reasons make it the RECOMMENDED summarizing function for the results of different test iterations, unless stated otherwise.

To express the repeatability of the benchmarking tests through a number, the Margin of error (MoE) can be used. Of course, other functions, such as standard error could be employed as well. The advantage the MoE has is expressing an associated confidence interval by using the alpha parameter.

The recommended formula for calculating the MoE is presented in Section 6.3.1.

11. 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.

12. 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.

13. 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 load scalability by quantifying the performance degradation associated with network growth.

14. References

14.1. Normative References

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

[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.

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

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

[RFC3393] Demichelis, C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, November 2002.

[RFC3511] Hickman, B., Newman, D., Tadjudin, S. and T. Martin, "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.

[RFC5481] Morton, A., and B. Claise, "Packet Delay Variation Applicability Statement", RFC 5481, March 2009.

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

14.2. Informative References

[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.

[DeNijs] De Nijs, R., and Thomas Levin Klausen. "On the expected difference between mean and median." Electronic Journal of Applied Statistical Analysis 6.1 (2013): 110-117.

15. Acknowledgements

The authors 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. We would also like to thank Scott Bradner, Al Morton and Fred Baker for their detailed review of the draft and very helpful suggestions. Other helpful comments and suggestions were offered by Bhuvaneswaran Vengainathan, Andrew McGregor, Nalini Elkins, Kaname Nishizuka, Yasuhiro Ohara, Masataka Mawatari,Kostas Pentikousis and Bela Almasi. 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 Ethernet as example media. The formula takes 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.

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

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

Gabor Lencse
Szechenyi Istvan University
Egyetem ter 1.
Gyor
Hungary

Phone: +36 20 775 8267
Email: lencse@sze.hu

Internet-Draft
Network Working Group
Intended Status: Informational
Expires: April 18, 2016

Bhuvaneswaran Vengainathan
Anton Basil
Veryx Technologies
Mark Tassinari
Hewlett-Packard
Vishwas Manral
Ionos Corp
Sarah Banks
VSS Monitoring
October 19, 2015

Benchmarking Methodology for SDN Controller Performance
draft-ietf-bmwg-sdn-controller-benchmark-meth-00

Abstract

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

Table of Contents

1. Introduction 3
2. Scope 4
3. Test Setup 4
   3.1. Test setup - Controller working in Standalone Mode 4
   3.2. Test setup - Controller working in Cluster Mode 5
4. Test Considerations 6
   4.1. Network Topology 6
   4.2. Test Traffic 7
   4.3. Connection Setup 7
   4.4. Measurement Point Specification and Recommendation 7
   4.5. Connectivity Recommendation 8
   4.6. Test Repeatability 8
5. Benchmarking Tests 9
   5.1. Performance 9
      5.1.1. Network Topology Discovery Time 9
      5.1.2. Asynchronous Message Processing Time 10
      5.1.3. Asynchronous Message Processing Rate 11
      5.1.4. Reactive Path Provisioning Time 13
      5.1.5. Proactive Path Provisioning Time 14
      5.1.6. Reactive Path Provisioning Rate 16
      5.1.7. Proactive Path Provisioning Rate 17
      5.1.8. Network Topology Change Detection Time 18
   5.2. 6.2 Scalability..... 20
      5.2.1. Control Session Capacity 20
      5.2.2. Network Discovery Size 20
      5.2.3. 6.2.3 Forwarding Table Capacity 21
   5.3. 6.3 Security 23
      5.3.1. 6.3.1 Exception Handling 23
      5.3.2. Denial of Service Handling 24

5.4. Reliability 26
5.4.1. Controller Failover Time 26
5.4.2. Network Re-Provisioning Time 27
6. References 29
6.1. Normative References 29
6.2. Informative References 29
7. IANA Considerations 30
8. Security Considerations 30
9. Acknowledgments 30
Appendix A. Example Test Topologies 31
A.1. Leaf-Spine Topology - Three Tier Network Architecture . 31
A.2. Leaf-Spine Topology - Two Tier Network Architecture ... 31
Appendix B. Benchmarking Methodology using OpenFlow Controllers 32
B.1. Protocol Overview 32
B.2. Messages Overview 32
B.3. Connection Overview 32
B.4. Performance Benchmarking Tests 33
B.4.1. Network Topology Discovery Time 33
B.4.2. Asynchronous Message Processing Time 34
B.4.3. Asynchronous Message Processing Rate 35
B.4.4. Reactive Path Provisioning Time 36
B.4.5. Proactive Path Provisioning Time 37
B.4.6. Reactive Path Provisioning Rate 38
B.4.7. Proactive Path Provisioning Rate 39
B.4.8. Network Topology Change Detection Time 40
B.5. Scalability 41
B.5.1. Control Sessions Capacity 41
B.5.2. Network Discovery Size 41
B.5.3. Forwarding Table Capacity 42
B.6. Security 44
B.6.1. Exception Handling 44
B.6.2. Denial of Service Handling 45
B.7. Reliability 47
B.7.1. Controller Failover Time 47
B.7.2. Network Re-Provisioning Time 48
Authors' Addresses 51

1. Introduction

This document provides generic methodologies for benchmarking SDN controller performance. An SDN controller may support many northbound and southbound protocols, implement a wide range of applications, and work solely, or as a group to achieve the desired functionality. This document considers an SDN controller as a black box, regardless of design and implementation. The tests defined in the document can be used to benchmark SDN controller for performance, scalability, reliability and security independent of

northbound and southbound protocols. These tests can be performed on an SDN controller running as a virtual machine (VM) instance or on a bare metal server. This document is intended for those who want to measure the SDN controller performance as well as compare various SDN controllers performance.

Conventions used in this document

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

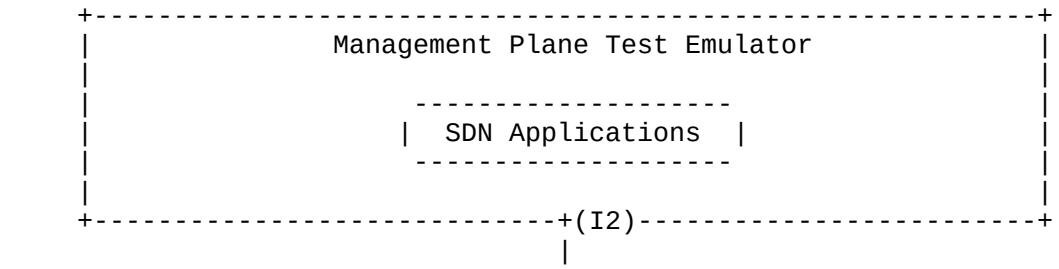
2. Scope

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

3. Test Setup

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

3.1. Test setup - Controller working in Standalone Mode



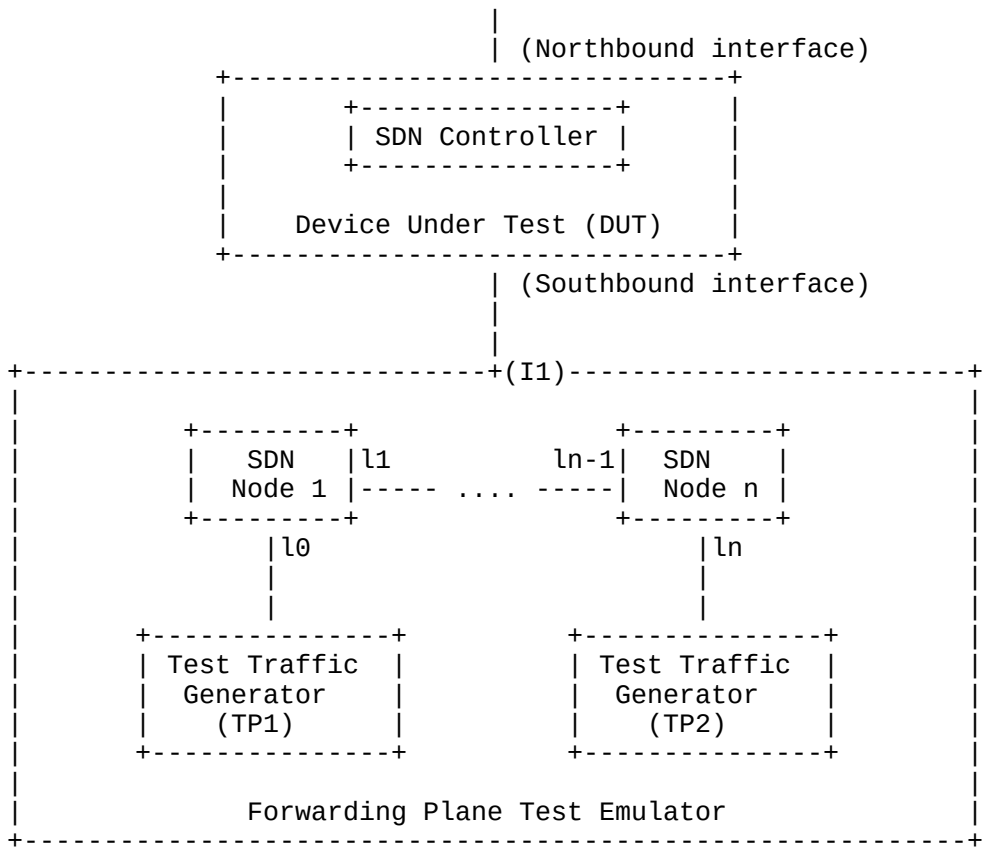
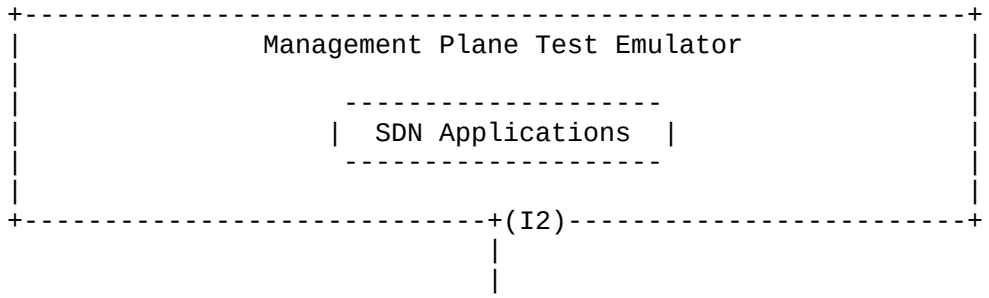


Figure 1

3.2. Test setup - Controller working in Cluster Mode



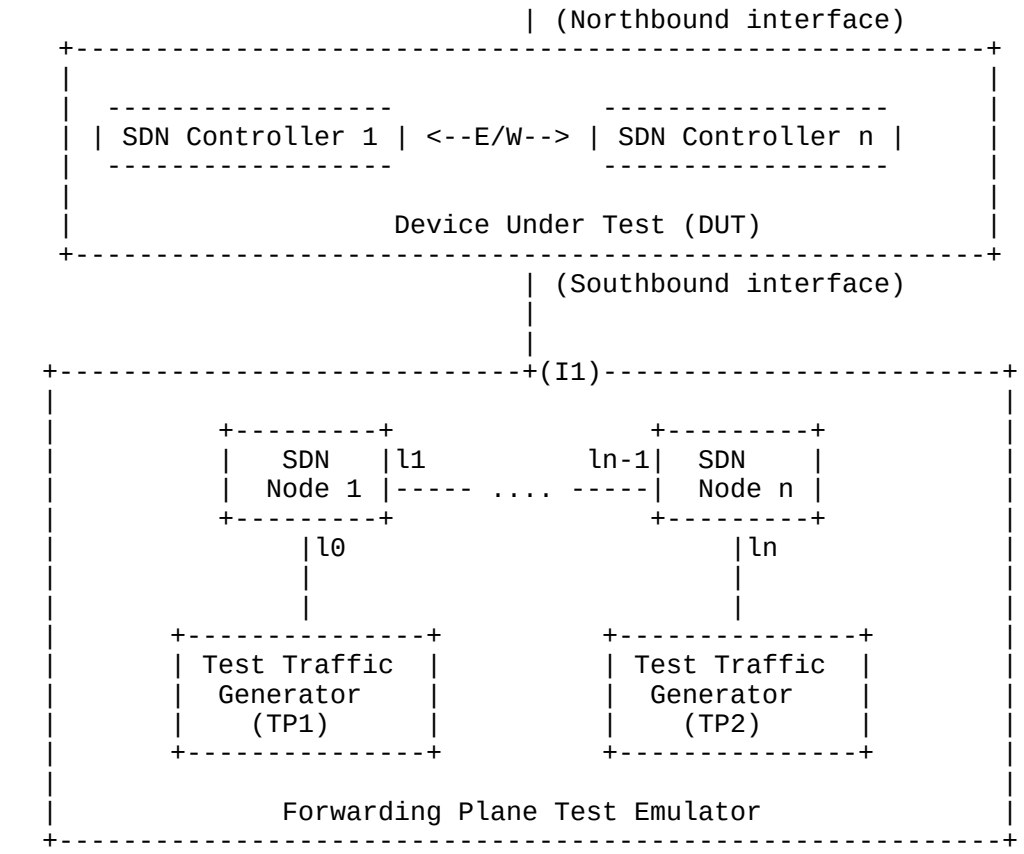


Figure 2

4. Test Considerations

4.1. Network Topology

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

controller, or in the network when the topology contains redundant network paths.

4.2. Test Traffic

Test traffic is used to notify the controller about the arrival of new flows. The test cases SHOULD use multiple frame sizes as recommended in RFC2544 for benchmarking.

4.3. 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 one or more applicable connection setup methods defined below.

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

4.4. Measurement Point Specification and Recommendation

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

4.5. Connectivity Recommendation

The SDN controller in the test setup SHOULD be connected directly with the forwarding and the management plane test emulators to avoid any delays or failure introduced by the intermediate devices during benchmarking tests.

4.6. Test Repeatability

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

Test Reporting

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

Test Configuration Parameters:

1. Controller name and version
2. Northbound protocols and versions
3. Southbound protocols and versions
4. Controller redundancy mode (Standalone or Cluster Mode)
5. Connection setup (Unencrypted or Encrypted)
6. Network 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
11. Controller System Configuration (e.g., CPU, Memory, Operating System, Interface Speed etc.,)
12. Reference Test Setup (e.g., Section 3.1 etc.,)

Controller Settings Parameters:

1. Topology re-discovery timeout
2. Controller redundancy mode (e.g., active-standby etc.,)

5. Benchmarking Tests

5.1. Performance

5.1.1. Network Topology Discovery Time

Objective:

Measure the time taken by the SDN controller to discover the network topology (nodes and links), expressed in milliseconds.

Reference Test Setup:

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

Prerequisite:

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

Procedure:

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

Measurement:

Topology Discovery Time Tr1 = Tmn-Tm1.

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

Reporting Format:

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

5.1.2. Asynchronous Message Processing Time

Objective:

Measure the time taken by the SDN controller to process an asynchronous message, expressed in milliseconds.

Reference Test Setup:

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

Prerequisite:

1. The controller MUST have completed the network topology discovery for the connected SDN nodes.

Procedure:

1. Generate asynchronous messages from every connected SDN node, to the SDN controller, one at a time in series from the forwarding plane test emulator for the test duration.

2. Record every request transmit (T1) timestamp and the corresponding response (R1) received timestamp at the forwarding plane test emulator interface (I1) for every successful message exchange.

Measurement:

$$\text{Asynchronous Message Processing Time } Tr1 = \frac{(R1-T1) + (R2-T2)..(Rn-Tn)}{Nrx}$$

Where Nrx is the total number of successful messages exchanged

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

Reporting Format:

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

The report should capture the following information in addition to the configuration parameters captured in section 5. - Successful messages exchanged (Nrx)

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

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 Asynchronous Message Processing Time.

5.1.3. Asynchronous Message Processing Rate

Objective:

To measure the maximum rate of asynchronous 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.

Reference Test Setup:

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

Prerequisite:

1. The controller MUST have completed the network topology discovery for the connected SDN nodes.

Procedure:

1. Generate asynchronous messages continuously at the maximum possible rate on the established connections from all the connected SDN nodes in the forwarding plane test emulator for the Test Duration (Td).
2. Record the total number of responses received from the controller (Nrx) as well as the number of messages sent(Ntx) to the controller within the test duration(Td) at the forwarding plane test emulator interface (I1).

Measurement:

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

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

$$\text{Loss Ratio} = (Ntx - Nrx)/100.$$

Reporting Format:

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

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

- Offered rate (Ntx)

- Loss Ratio

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 Asynchronous 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 Asynchronous Message Processing Rate.

5.1.4. Reactive Path Provisioning Time

Objective:

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

Reference Test Setup:

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

Prerequisite:

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

Procedure:

1. Send a single traffic stream from the test traffic generator TP1 to test traffic generator TP2.
2. Record the time of the first flow provisioning request message sent to the controller (Tsf1) from the SDN node at the forwarding plane test emulator interface (I1).

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

Measurement:

Reactive Path Provisioning Time Tr1 = Tdf1-Tsf1.

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

Reporting Format:

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

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

- Number of SDN nodes in the path

5.1.5. Proactive Path Provisioning Time

Objective:

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

Reference Test Setup:

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

Prerequisite:

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

2. The controller should have the knowledge about the location of destination endpoint for which the path has to be provisioned. This can be achieved through dynamic learning or static provisioning.
3. Ensure that the default action for flow miss in SDN node is 'drop'.

Procedure:

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

Measurement:

Proactive Flow Provisioning Time Tr1 = Tdf1-Tsf1.

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

Reporting Format:

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

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

- Number of SDN nodes in the path

5.1.6. Reactive Path Provisioning Rate

Objective:

Measure the maximum number of independent paths a controller can concurrently establish between source and destination nodes reactively within the test duration, expressed in paths per second.

Reference Test Setup:

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

Prerequisite:

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

Procedure:

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

Measurement:

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

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

Reporting Format:

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

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

- Number of SDN nodes in the path

- Offered rate

5.1.7. Proactive Path Provisioning Rate

Objective:

Measure the maximum number of independent paths a controller can concurrently establish between source and destination nodes proactively within the test duration, expressed in paths per second.

Reference Test Setup:

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

Prerequisite:

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

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

3. Ensure that the default action for flow miss in SDN node is ’drop’.

Procedure:

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

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

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

Measurement:

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

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

Reporting Format:

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

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

- Number of SDN nodes in the path

- Offered rate

5.1.8. Network Topology Change Detection Time

Objective:

Measure the time taken by the controller to detect any changes in the network topology, expressed in milliseconds.

Reference Test Setup:

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

Prerequisite:

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

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

Procedure:

1. Trigger a topology change event by bringing down an active SDN node in the topology.

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

3. Stop the test when the controller sends the first topology re-discovery message to the SDN node or the expiry of test interval (Td).

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

Measurement:

Network Topology Change Detection Time Tr1 = Tcd-Tcn.

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

Reporting Format:

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

5.2. 6.2 Scalability

5.2.1. Control Session Capacity

Objective:

Measure the maximum number of control sessions that the controller can maintain.

Reference Test Setup:

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

Procedure:

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

Measurement:

Control Sessions Capacity = CCn.

Reporting Format:

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

5.2.2. Network Discovery Size

Objective:

Measure the network size (number of nodes, links, and hosts) that a controller can discover.

Reference Test Setup:

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

Prerequisite:

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

Procedure:

1. Establish the network connections between controller and network nodes.
2. Query the controller for the discovered network topology information and compare it with the deployed network topology information.
3. 3a. Increase the number of nodes by 1 when the comparison is successful and repeat the test.
4. 3b. Decrease the number of nodes by 1 when the comparison fails and repeat the test.
5. Continue the test until the comparison of step 3b is successful.
6. Record the number of nodes for the last iteration (Ns) where the topology comparison was successful.

Measurement:

Network Discovery Size = Ns.

Reporting Format:

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

5.2.3. 6.2.3 Forwarding Table Capacity

Objective:

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

Reference Test Setup:

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

Prerequisite:

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

Procedure:

Reactive Flow Provisioning Mode:

1. Send bi-directional traffic continuously with unique source and/or destination addresses from test traffic generators TP1 and TP2 at the asynchronous message processing rate of controller.
2. Query the controller at a regular interval (e.g., 5 seconds) for the number of learnt 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 Flow Provisioning Mode:

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

Note:

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

Measurement:

Proactive Flow Provisioning Mode:

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

Reactive Flow Provisioning Mode:

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

Forwarding Table Capacity = Max Flow Entries.

Reporting Format:

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

- Provisioning Type (Proactive/Reactive)

5.3. 6.3 Security

5.3.1. 6.3.1 Exception Handling

Objective:

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

a. Path Provisioning Rate

b. Path Provisioning Time

c. Network Topology Change Detection Time

Reference Test Setup:

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

Prerequisite:

1. This test MUST be performed after obtaining the baseline measurement results for the above performance tests.
2. Ensure that the invalid messages are not dropped by the intermediate devices connecting the controller and SDN nodes.

Procedure:

1. Perform the above listed performance tests and send 1% of messages from the Asynchronous Message Processing Rate as invalid messages

from the connected SDN nodes emulated at the forwarding plane test emulator.

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

Note:

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

Measurement:

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

Reporting Format:

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

- Without Exceptions
- With 1% Exceptions
- With 2% Exceptions

5.3.2. Denial of Service Handling

Objective:

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

a. Path Provisioning Rate

b. Path Provisioning Time

c. Network Topology Change Detection Time

d. Network Discovery Size

Reference Test Setup:

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

Prerequisite:

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

Procedure:

1. Perform the listed tests and launch a DoS attack towards controller while the 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 MUST be done as per the equation defined in the corresponding test's measurement section.

Reporting Format:

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

- Without any attacks

- With attacks

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

5.4. Reliability

5.4.1. Controller Failover Time

Objective:

Measure the time taken to switch from an active controller to the backup controller, when the controllers work in redundancy mode and the active controller fails.

Reference Test Setup:

The test SHOULD use the test setup described in section 3.2 of this document.

Prerequisite:

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

Procedure:

1. Send uni-directional traffic continuously with incremental sequence number and source addresses from test traffic generator TP1 at the rate that the controller processes without any drops.
2. Ensure that there are no packet drops observed at the test traffic generator TP2.
3. Bring down the active controller.
4. Stop the test when a first frame received on TP2 after failover operation.
5. Record the time at which the last valid frame received (T1) at test traffic generator TP2 before sequence error and the first valid frame received (T2) after the sequence error at TP2

Measurement:

Controller Failover Time = (T2 - T1)

Packet Loss = Number of missing packet sequences.

Reporting Format:

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

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

5.4.2. Network Re-Provisioning Time

Objective:

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

Reference Test Setup:

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

Prerequisite:

1. Network with the given number of nodes and redundant paths MUST be deployed.
2. Ensure that the controller MUST have knowledge about the location of test traffic generators TP1 and TP2.
3. Ensure that the controller does not pre-provision the alternate path in the emulated SDN nodes at the forwarding plane test emulator.

Procedure:

1. Send bi-directional traffic continuously with unique sequence number from TP1 and TP2.
2. Bring down a link or switch in the traffic path.

3. Stop the test after receiving first frame after network re-convergence.
4. Record the time of last received frame prior to the frame loss at TP2 (TP2-Tlfr) and the time of first frame received after the frame loss at TP2 (TP2-Tffr).
5. Record the time of last received frame prior to the frame loss at TP1 (TP1-Tlfr) and the time of first frame received after the frame loss at TP1 (TP1-Tffr).

Measurement:

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

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

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

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

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

Reporting Format:

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

- Number of nodes in the primary path

- Number of nodes in the alternate path

- Network Re-Provisioning Time

- Forward Direction Packet Loss

- Reverse Direction Packet Loss

6. References

6.1. Normative References

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

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

[RFC6241] R. Enns, M. Bjorklund, J. Schoenwaelder, A. Bierman, "Network Configuration Protocol (NETCONF)",RFC 6241, July 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.sdn-controller-benchmark-term] Bhuvaneswaran.V, Anton Basil, Mark.T, Vishwas Manral, Sarah Banks, "Terminology for Benchmarking SDN Controller Performance", draft-ietf-bmwg-sdn-controller-benchmark-term-00 (Work in progress), October 19, 2015

6.2. Informative References

[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-09 (Work in progress), March 6, 2015

[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

7. IANA Considerations

This document does not have any IANA requests.

8. Security Considerations

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

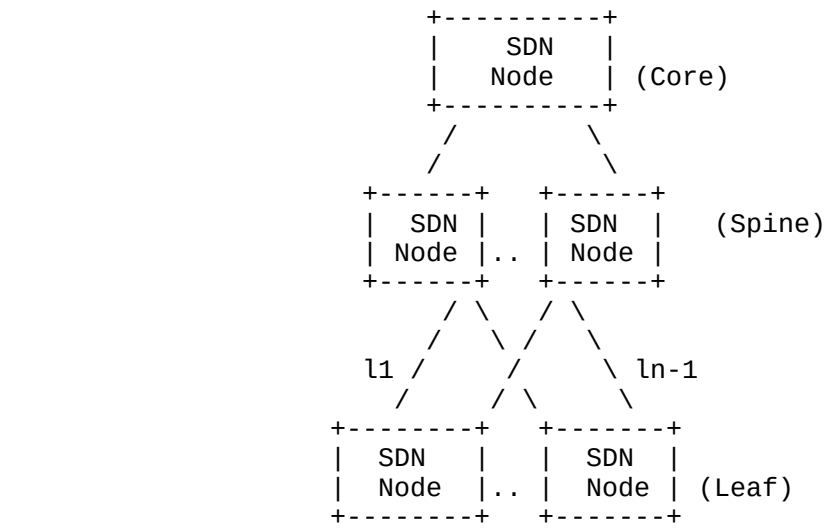
9. Acknowledgments

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

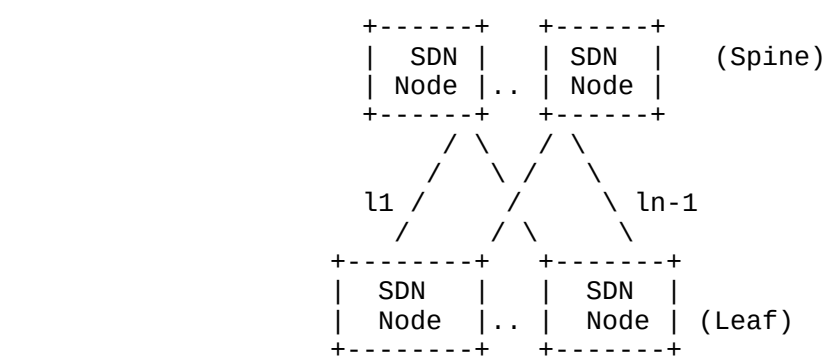
This document was prepared using 2-Word-v2.0.template.dot.

Appendix A. Example Test Topologies

A.1. Leaf-Spine Topology - Three Tier Network Architecture



A.2. Leaf-Spine Topology - Two Tier Network Architecture



Appendix B. Benchmarking Methodology using OpenFlow Controllers

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

B.1. Protocol Overview

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

B.2. Messages Overview

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

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

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

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

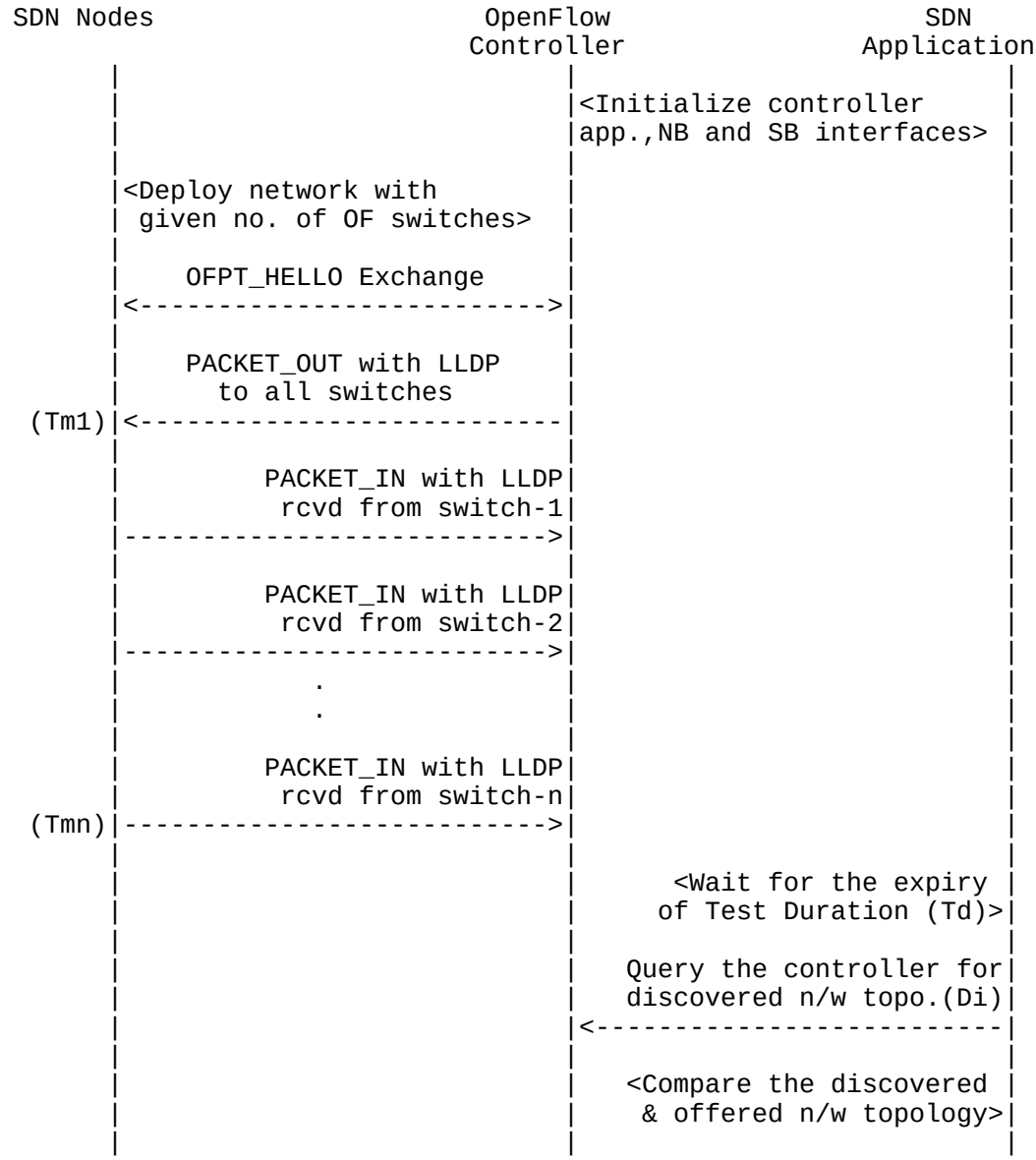
B.3. Connection Overview

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

B.4. Performance Benchmarking Tests

B.4.1. Network Topology Discovery Time

Procedure:



Legend:

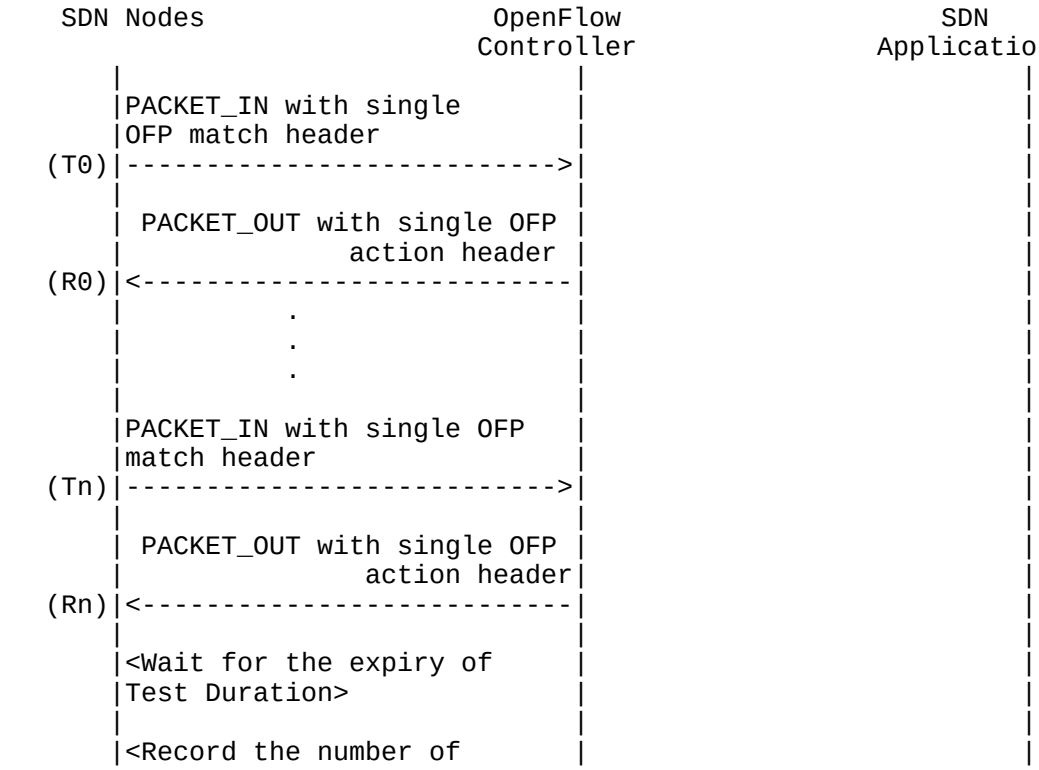
NB: Northbound
SB: Southbound
OF: OpenFlow
Tm1: Time of reception of first LLDP message from controller
Tmn: Time of last LLDP message sent to controller

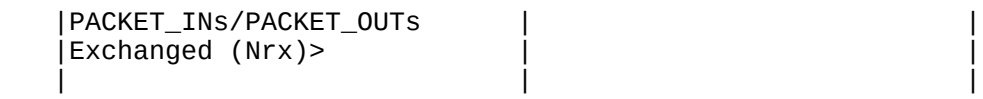
Discussion:

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

B.4.2. Asynchronous Message Processing Time

Procedure:





Legend:

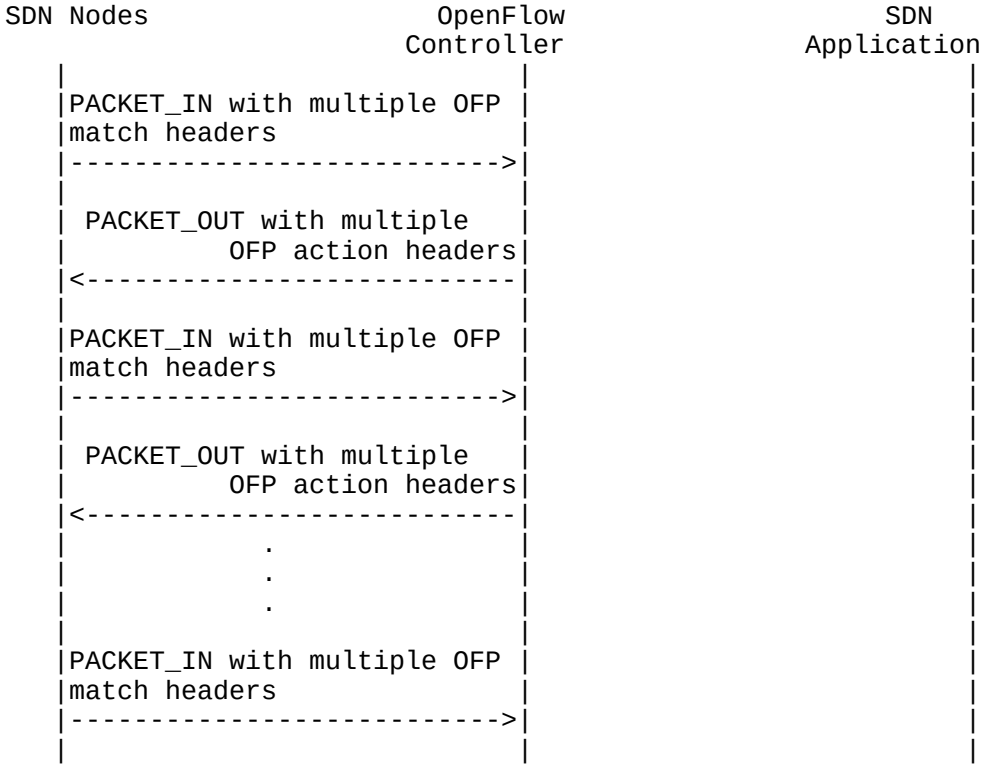
T0,T1, ..Tn are PACKET_IN messages transmit timestamps.
R0,R1, ..Rn are PACKET_OUT messages receive timestamps.
Nrx : Number of successful PACKET_IN/PACKET_OUT message exchanges

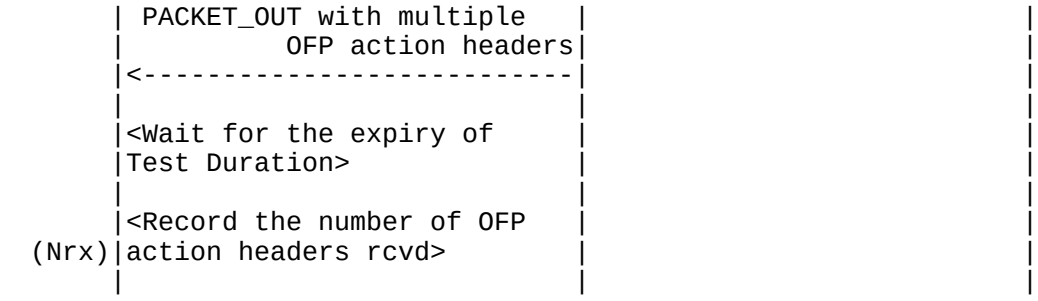
Discussion:

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

B.4.3. Asynchronous Message Processing Rate

Procedure:



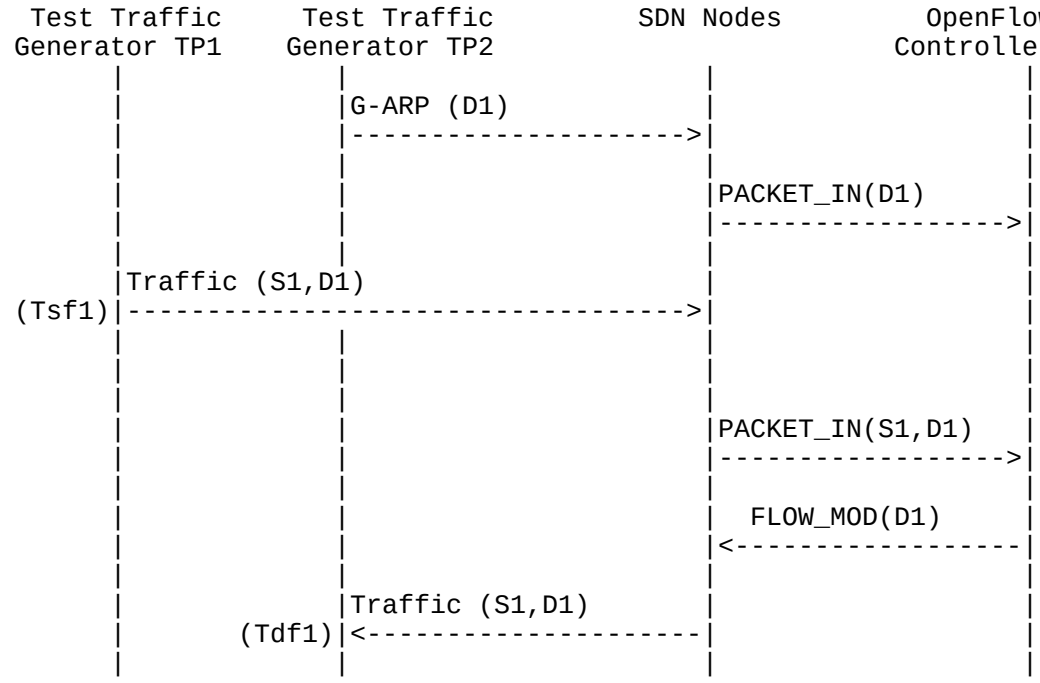


Discussion:

The Asynchronous Message Processing Rate will be obtained by calculating the number of OFP action headers received in all PACKET_OUT messages during the test duration.

B.4.4. Reactive Path Provisioning Time

Procedure:



Legend:

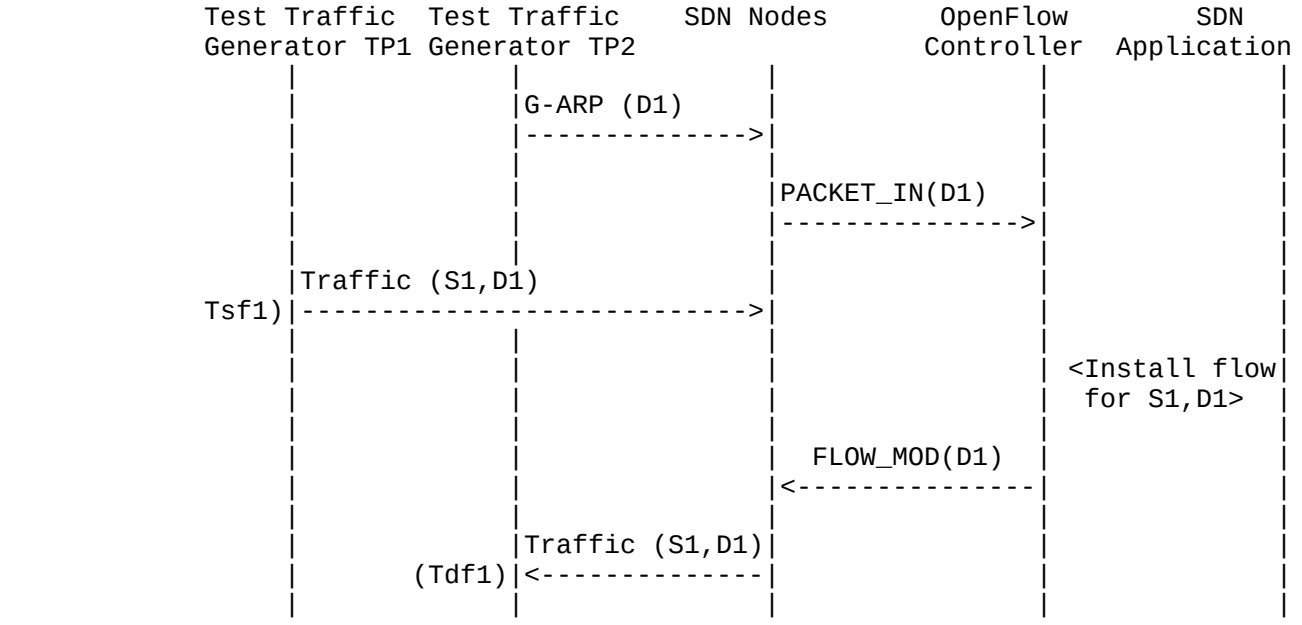
G-ARP: Gratuitous ARP message.
Tsf1: Time of first frame sent from TP1
Tdf1: Time of first frame received from TP2

Discussion:

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

B.4.5. Proactive Path Provisioning Time

Procedure:



Legend:

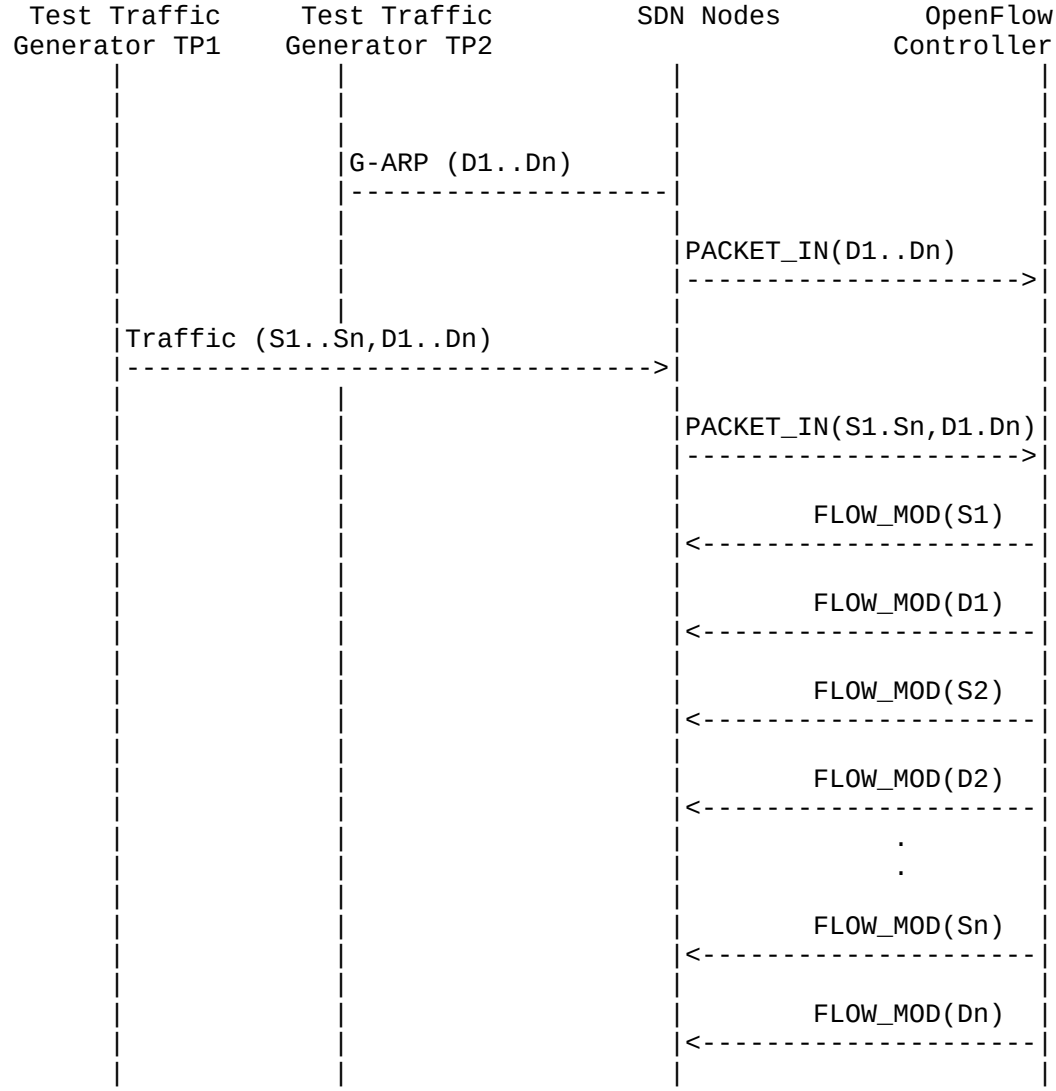
G-ARP: Gratuitous ARP message.
Tsf1: Time of first frame sent from TP1
Tdf1: Time of first frame received from TP2

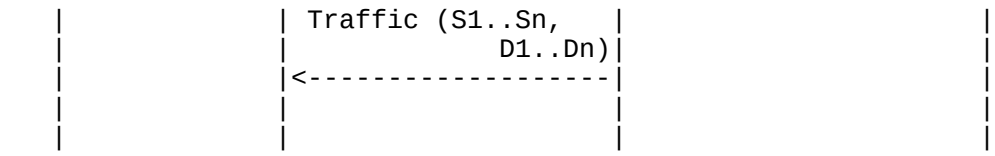
Discussion:

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

B.4.6. Reactive Path Provisioning Rate

Procedure:





Legend:

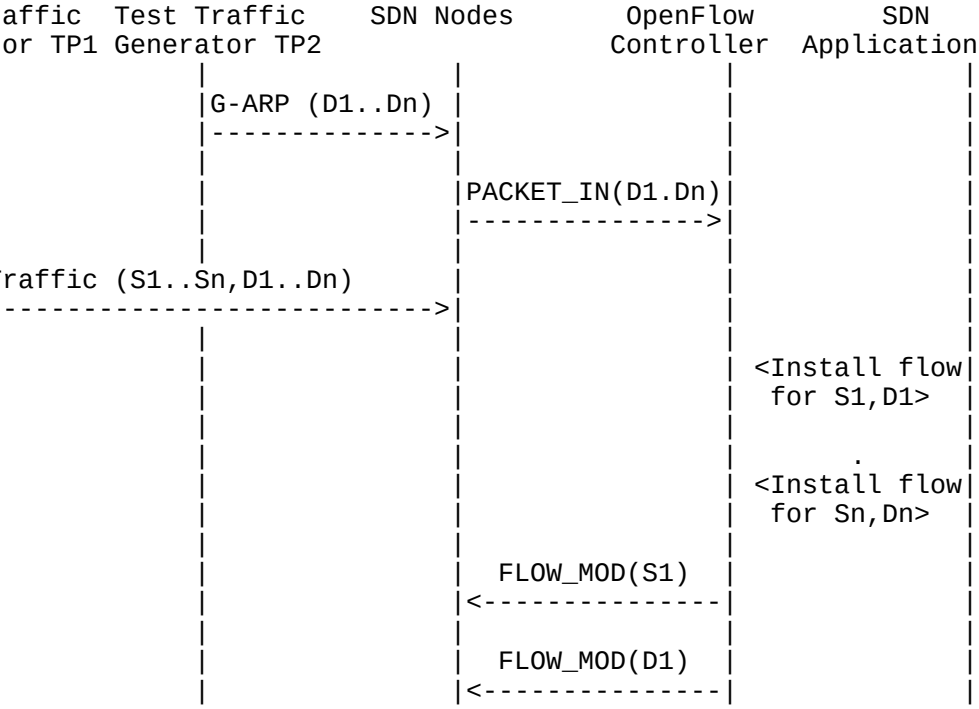
G-ARP: Gratuitous ARP
D1..Dn: Destination Endpoint 1, Destination Endpoint 2 ....
Destination Endpoint n
S1..Sn: Source Endpoint 1, Source Endpoint 2 .., Source Endpoint n

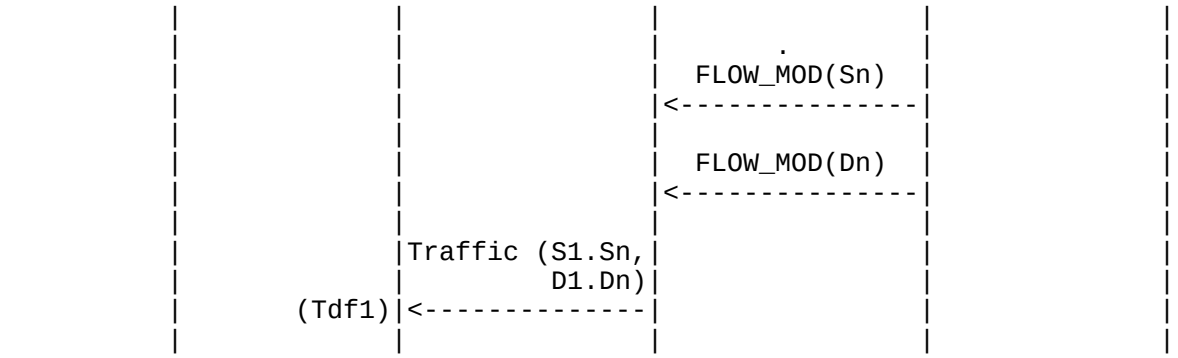
Discussion:

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

B.4.7. Proactive Path Provisioning Rate

Procedure:





Legend:

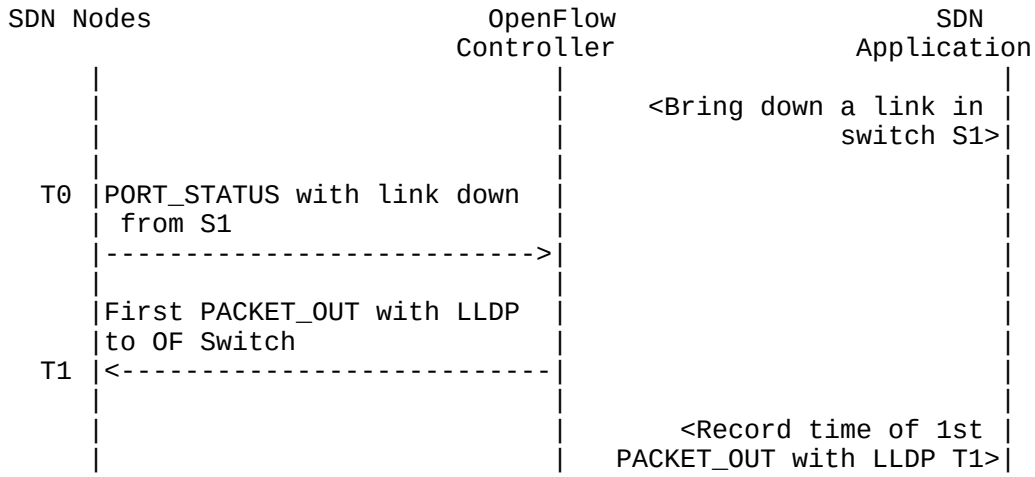
G-ARP: Gratuitous ARP
D1..Dn: Destination Endpoint 1, Destination Endpoint 2 ....
Destination Endpoint n
S1..Sn: Source Endpoint 1, Source Endpoint 2 .., Source Endpoint n

Discussion:

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

B.4.8. Network Topology Change Detection Time

Procedure:



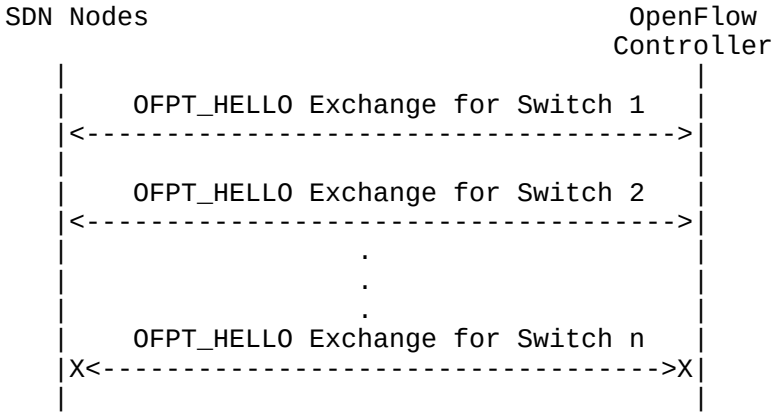
Discussion:

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

B.5. Scalability

B.5.1. Control Sessions Capacity

Procedure:

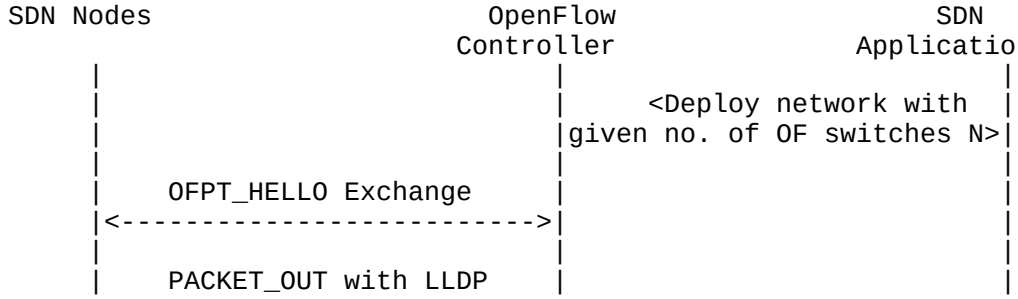


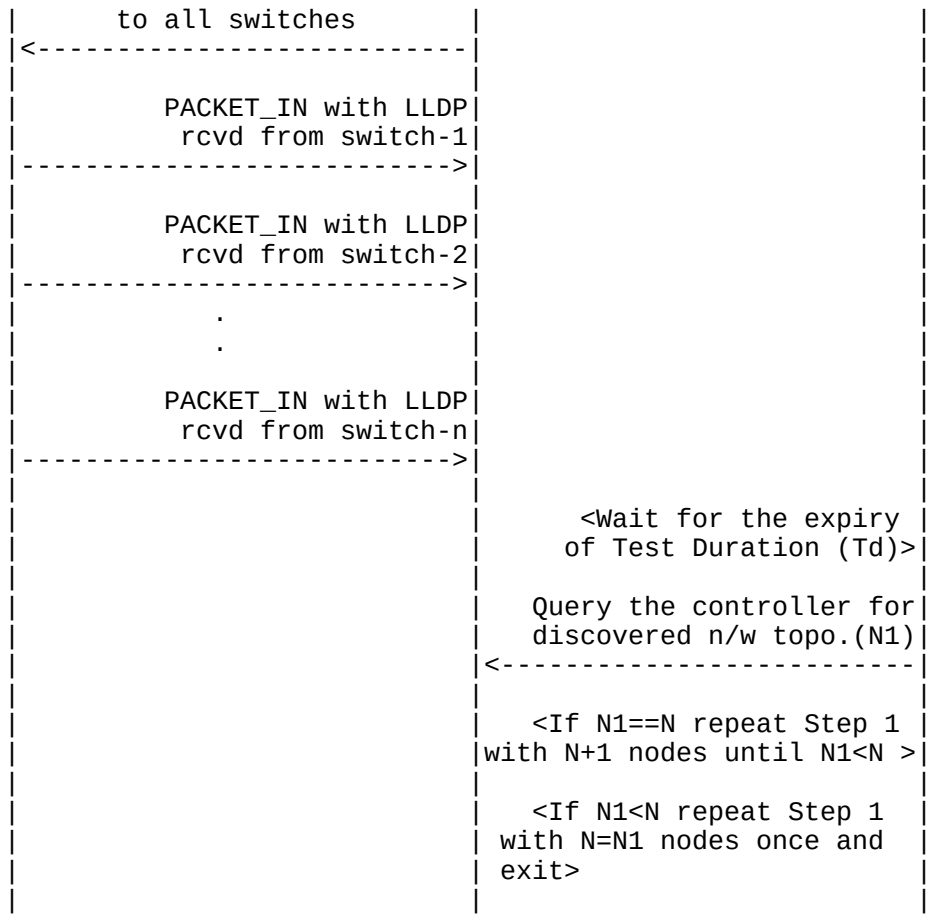
Discussion:

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

B.5.2. Network Discovery Size

Procedure:





Legend:

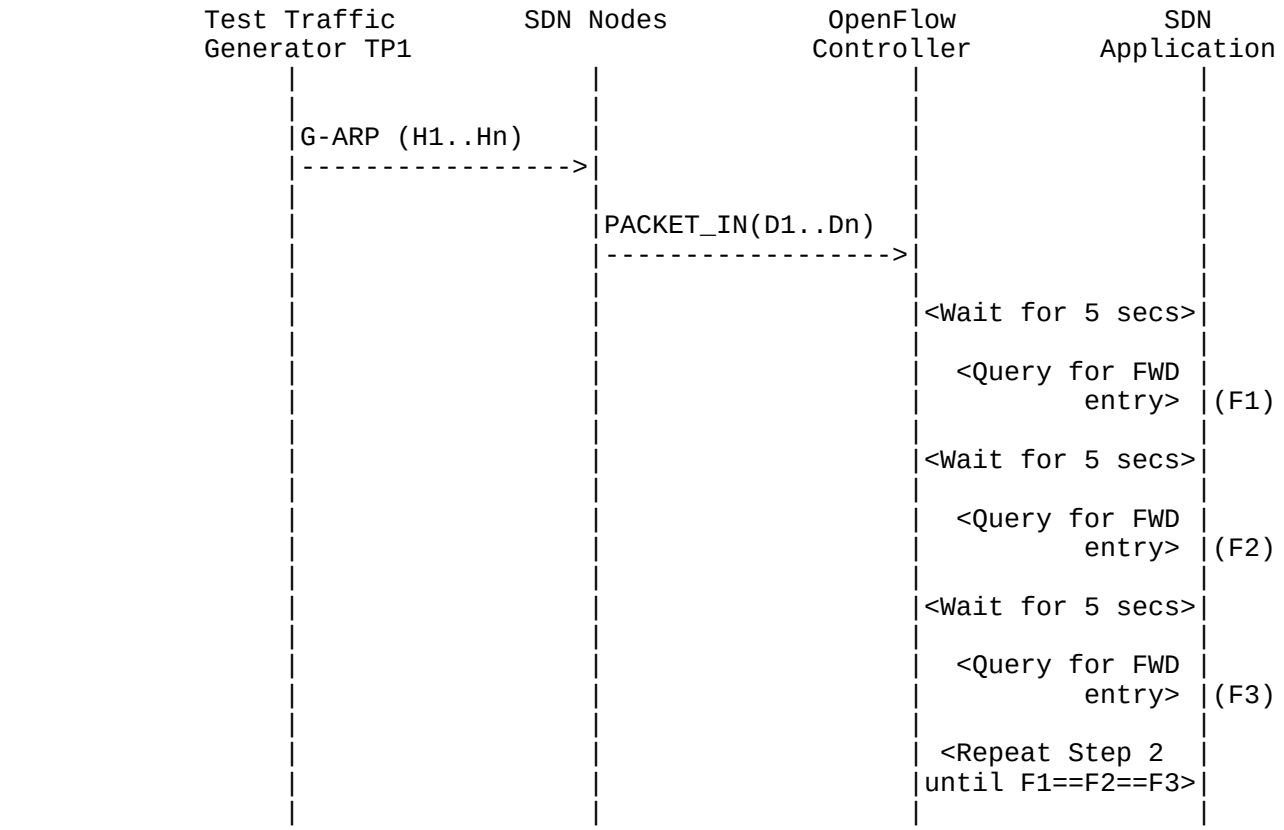
n/w topo: Network Topology
OF: OpenFlow

Discussion:

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

B.5.3. Forwarding Table Capacity

Procedure:



Legend:

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

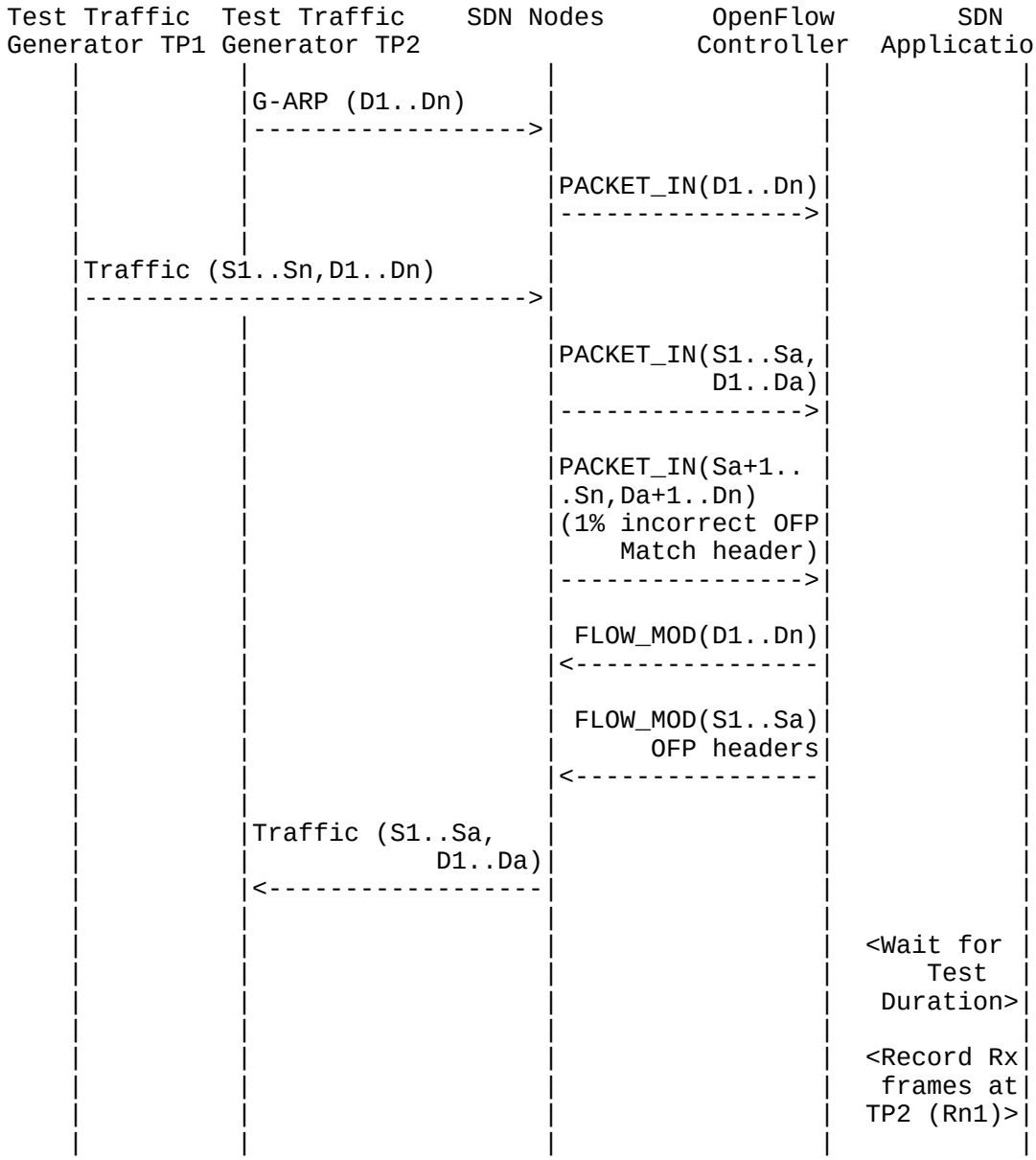
Discussion:

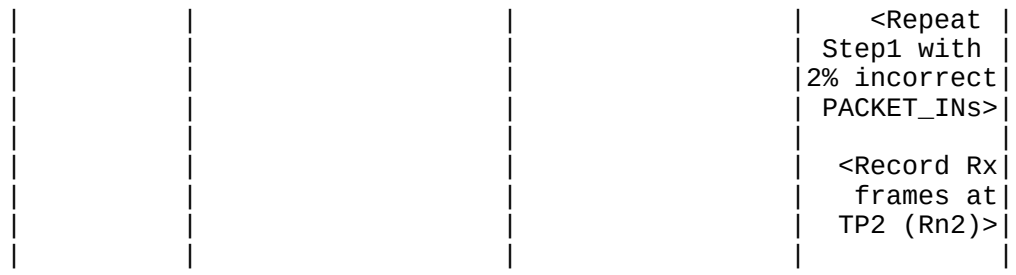
Query the controller forwarding table entries for multiple times until the three consecutive queries return the same value. The last value retrieved from the controller will provide the Forwarding Table Capacity value. The query interval is user configurable. The 5 seconds shown in this example is for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

B.6. Security

B.6.1. Exception Handling

Procedure:





Legend:

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

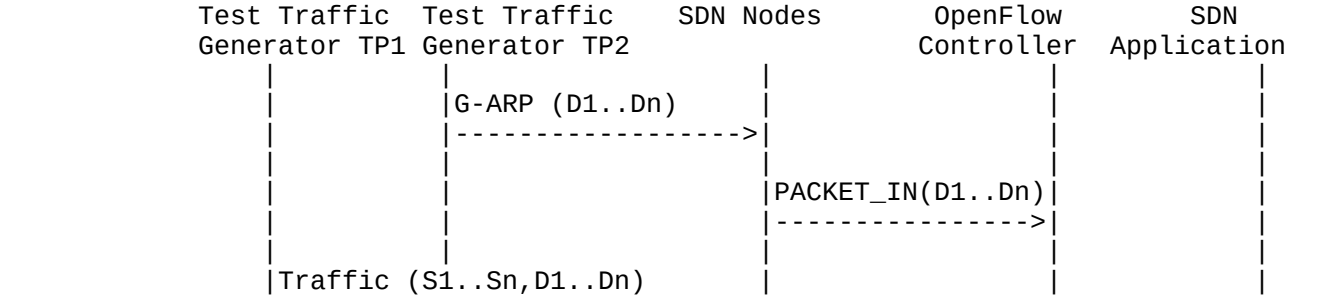
Discussion:

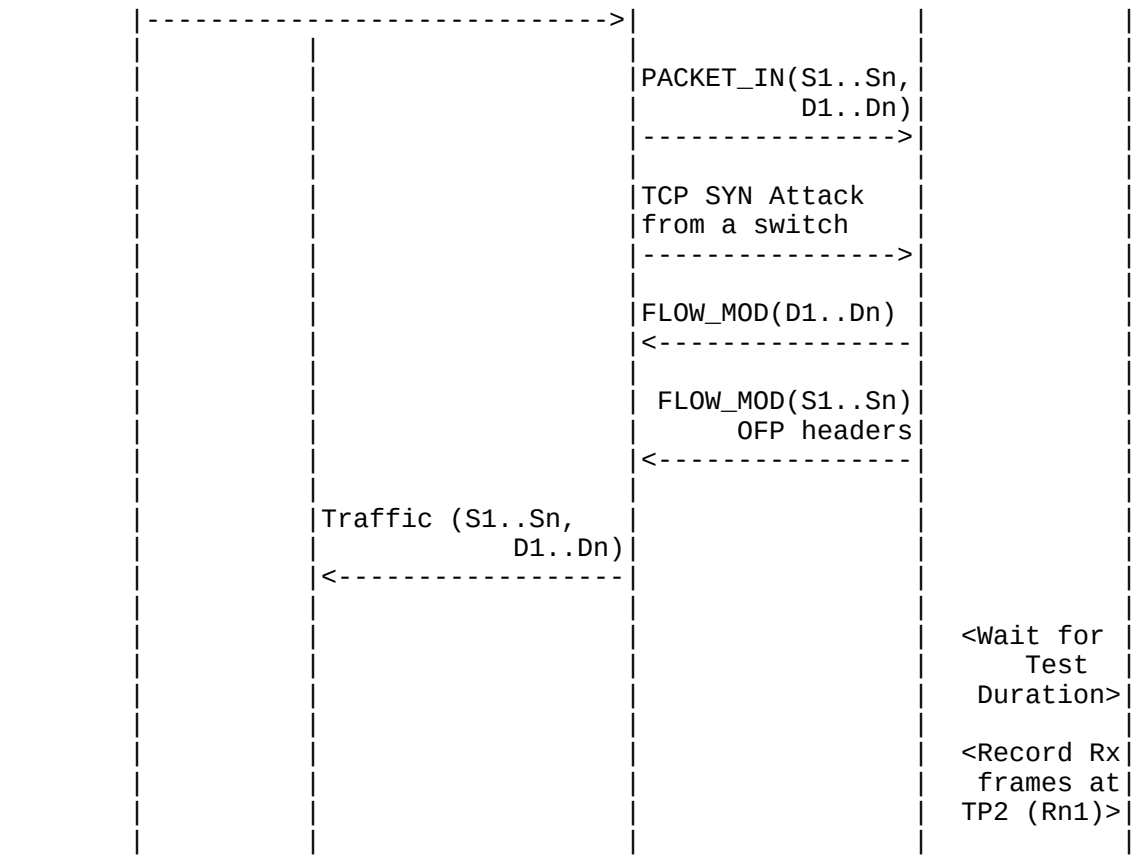
The traffic rate sent towards OpenFlow switch from Test Port 1 should be 1% higher than the Path Programming Rate. Rn1 will provide the Path Provisioning Rate of controller at 1% of incorrect frames handling and Rn2 will provide the Path Provisioning Rate of controller at 2% of incorrect frames handling.

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

B.6.2. Denial of Service Handling

Procedure:





Legend:

G-ARP: Gratuitous ARP

Discussion:

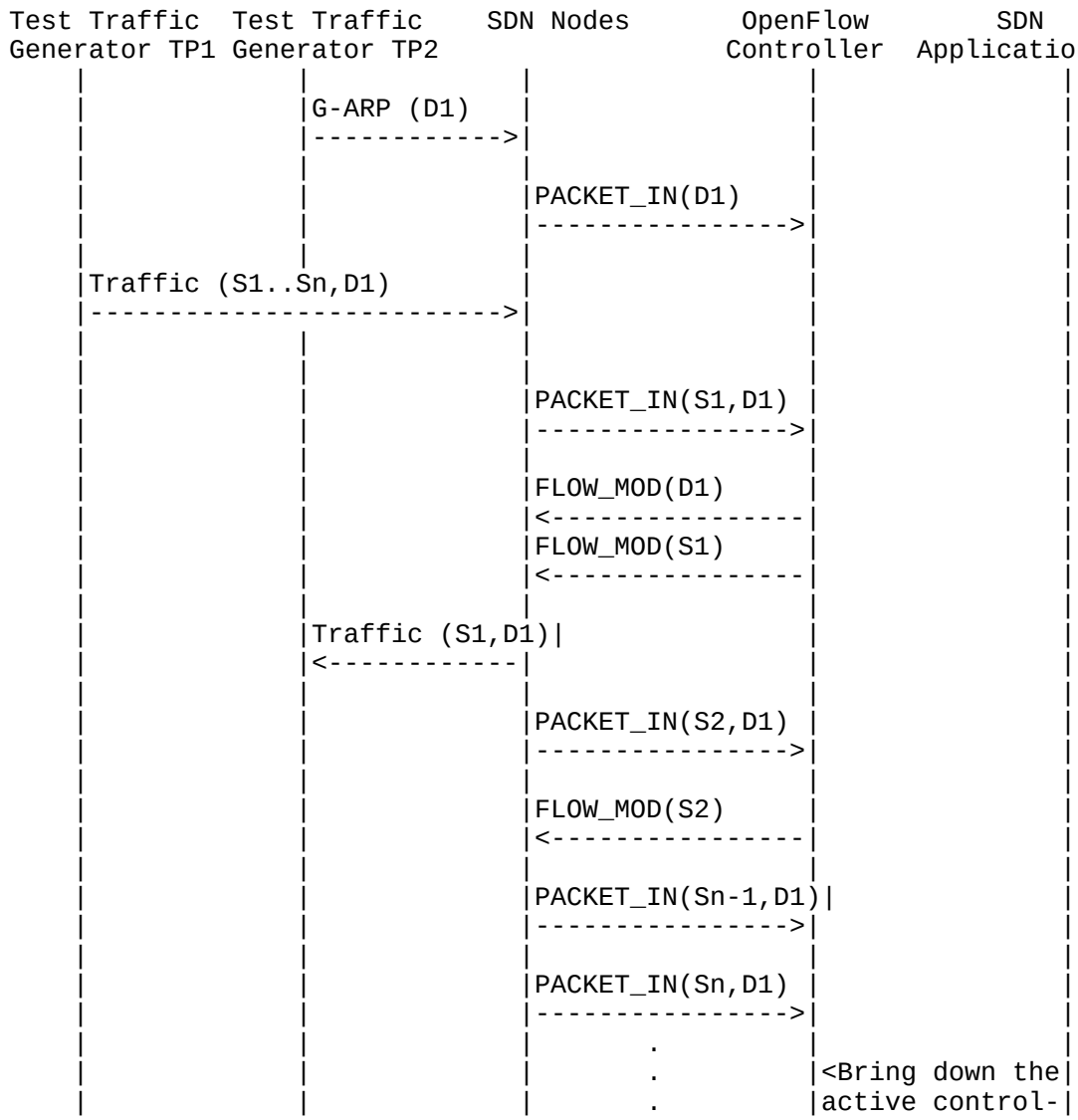
TCP SYN attack should be launched from one of the emulated/simulated OpenFlow Switch. Rn1 provides the Path Programming Rate of controller uponhandling denial of service attack.

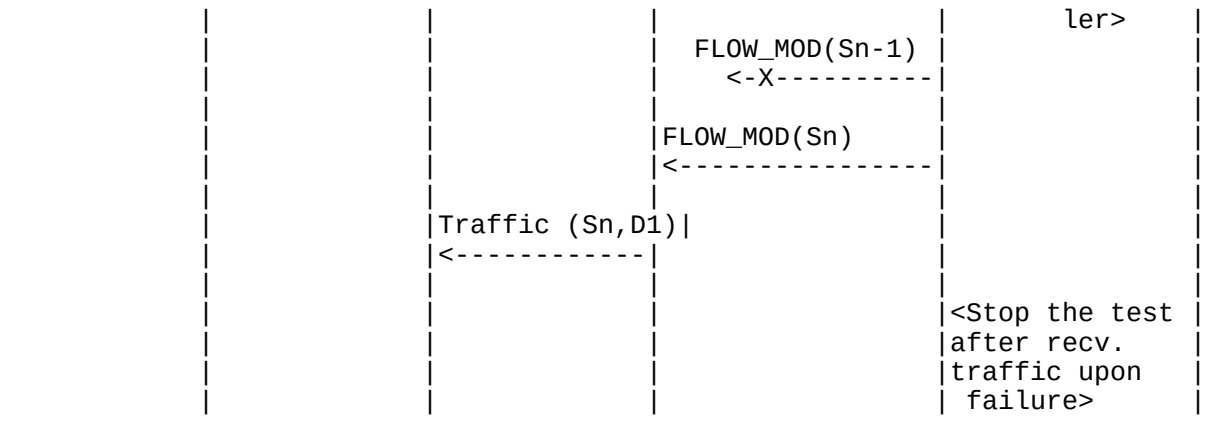
The procedure defined above provides test steps to determine the effect of handling denial of service on Path Programming Rate. Same procedure can be adopted to determine the effects on other performance tests listed in this benchmarking tests.

B.7. Reliability

B.7.1. Controller Failover Time

Procedure:





Legend:

G-ARP: Gratuitous ARP.

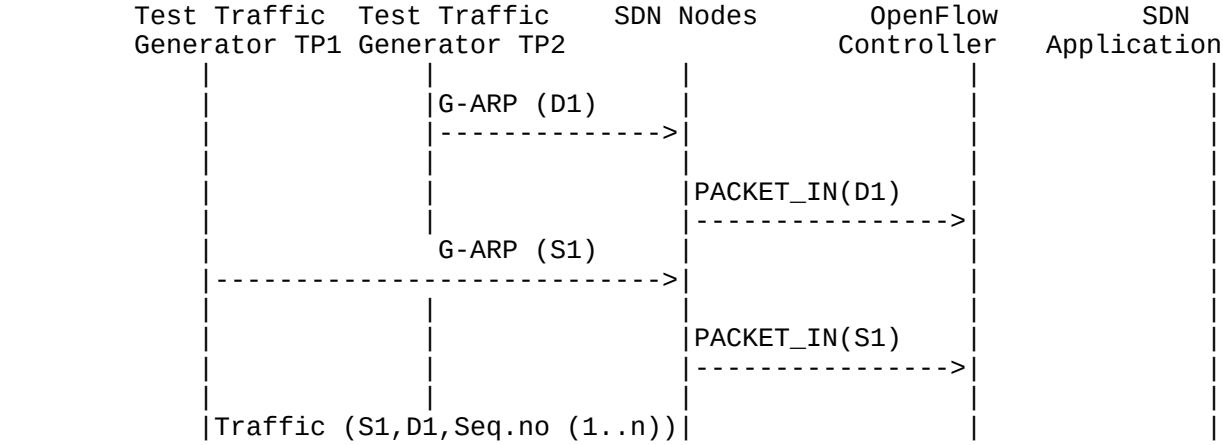
Discussion:

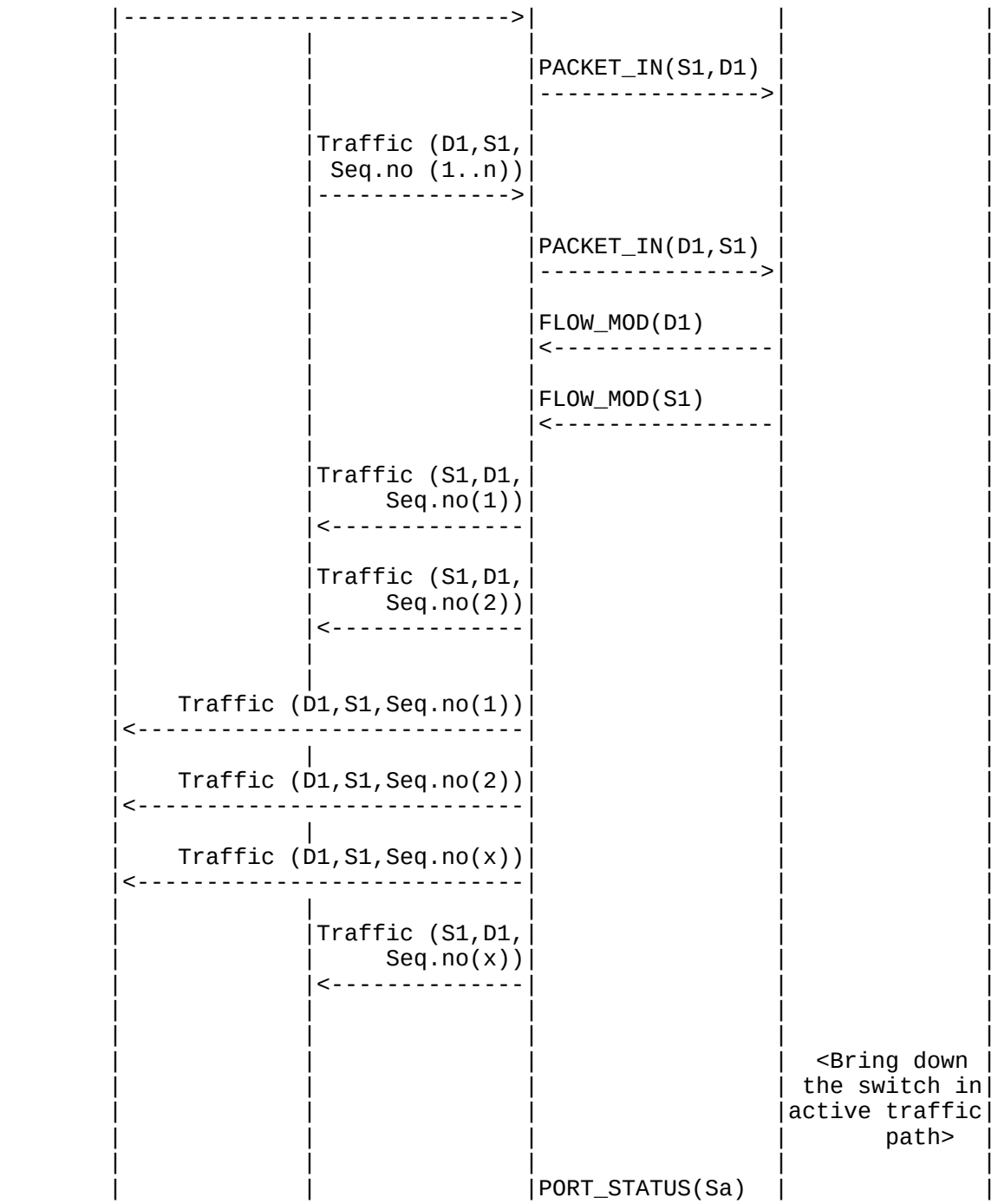
The time difference between the last valid frame received before the traffic loss and the first frame received after the traffic loss will provide the controller failover time.

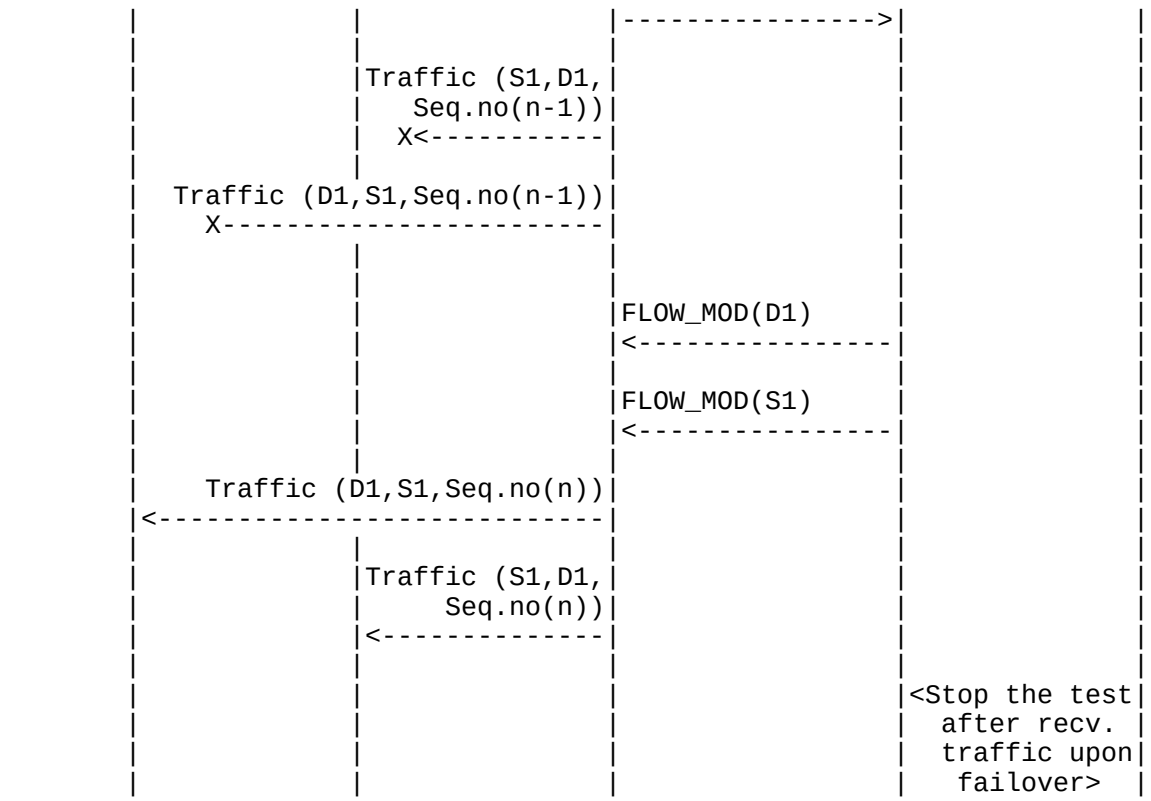
If there is no frame loss during controller failover time, the controller failover time can be deemed negligible.

B.7.2. Network Re-Provisioning Time

Procedure:




PACKET_IN(S1,D1)
Traffic (D1,S1,
Seq.no (1..n))
PACKET_IN(D1,S1)
FLOW_MOD(D1)
FLOW_MOD(S1)
Traffic (S1,D1,
Seq.no(1))
Traffic (S1,D1,
Seq.no(2))
Traffic (D1,S1,Seq.no(1))
Traffic (D1,S1,Seq.no(2))
Traffic (D1,S1,Seq.no(x))
Traffic (S1,D1,
Seq.no(x))
<Bring down
the switch in
active traffic
path>
PORT_STATUS(Sa)



Legend:

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

Discussion:

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

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

Authors' Addresses

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

Email: bhuvaneswaran.vengainathan@veryxtech.com

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

Email: anton.basil@veryxtech.com

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

Email: mark.tassinari@hp.com

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

Email: vishwas@ionosnetworks.com

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

Email: sbanks@encrypted.net

Internet-Draft
Network Working Group
Intended Status: Informational
Expires: April 18, 2016

Bhuvaneswaran Vengainathan
Anton Basil
Veryx Technologies
Mark Tassinari
Hewlett-Packard
Vishwas Manral
Ionos Corp
Sarah Banks
VSS Monitoring
October 19, 2015

Terminology for Benchmarking SDN Controller Performance
draft-ietf-bmwg-sdn-controller-benchmark-term-00

Abstract

This document defines terminology for benchmarking an SDN Controller's performance. The terms provided in this document help to benchmark SDN controller's performance independent of the controller's supported protocols and/or network services. A mechanism for benchmarking the performance of SDN controllers is defined in the companion methodology document. These two documents provide a standard mechanism to measure and evaluate the performance of various controller implementations.

Table of Contents

1. Introduction 3
2. Term Definitions 4
2.1. SDN Terms 4
2.1.1. SDN Node 4
2.1.2. SDN Application 4
2.1.3. Flow 4
2.1.4. Northbound Interface 5
2.1.5. Southbound Interface 5
2.1.6. Controller Forwarding Table 5
2.1.7. Proactive Flow Provisioning Mode 6
2.1.8. Reactive Flow Provisioning Mode 6
2.1.9. Path 7
2.1.10. Standalone Mode 7
2.1.11. Cluster/Redundancy Mode 7
2.1.12. Asynchronous Message 8
2.1.13. Test Traffic Generator 8
2.2. Test Configuration/Setup Terms 9
2.2.1. Number of SDN Nodes 9
2.2.2. Test Iterations 9
2.2.3. Test Duration 9
2.2.4. Number of Cluster nodes 10
2.3. Benchmarking Terms 10
2.3.1. Performance 10
2.3.1.1. Network Topology Discovery Time 10
2.3.1.2. Asynchronous Message Processing Time..... 11
2.3.1.3. Asynchronous Message Processing Rate..... 11
2.3.1.4. Reactive Path Provisioning Time 12
2.3.1.5. Proactive Path Provisioning Time 12

2.3.1.6. Reactive Path Provisioning Rate 13
2.3.1.7. Proactive Path Provisioning Rate 13
2.3.1.8. Network Topology Change Detection Time 13
2.3.2. Scalability 14
2.3.2.1. Control Sessions Capacity 14
2.3.2.2. Network Discovery Size 14
2.3.2.3. Forwarding Table Capacity 15
2.3.3. Security 15
2.3.3.1. Exception Handling 15
2.3.3.2. Denial of Service Handling 16
2.3.4. Reliability 16
2.3.4.1. Controller Failover Time 16
2.3.4.2. Network Re-Provisioning Time 17
3. Test Coverage 17
4. References 18
4.1. Normative References 18
4.2. Informative References 19
5. IANA Considerations 19
6. Security Considerations 19
7. Acknowledgements 19
8. Authors' Addresses 19

1. Introduction

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

Conventions used in this document

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

2. Term Definitions

2.1. SDN Terms

2.1.1. SDN Node

Definition:
An SDN node is a simulated/emulated entity that forwards data in a software defined environment.

Discussion:
An SDN node can be an emulated/simulated switch, router, gateway, or any network service appliance that supports standardized or proprietary programmable interface.

Measurement Units:
N/A

See Also:
None

2.1.2. SDN Application

Definition:
Any business logic that alter the network behaviour dynamically through controller’s northbound interface.

Discussion:
SDN application can be any business application, cloud orchestration system, network services orchestration etc.,
Measurement Units:
N/A

See Also:
None

2.1.3. Flow

Definition:
A flow is a uni-directional sequence of packets having common properties derived from the data contained in the packet.

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

Measurement Units:
N/A

See Also:
None

2.1.4. Northbound Interface

Definition:
The northbound interface is the application programming interface provided by the SDN controller for the SDN services and applications to interact with the SDN controller.

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

Measurement Units:
N/A

See Also:
None

2.1.5. Southbound Interface

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

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

Measurement Units:
N/A

See Also:
None

2.1.6. Controller Forwarding Table

Definition:
A controller forwarding table contains flow entries learned in one of two ways: first, entries could be learned from traffic received through the data plane, or, second, these entries could be

statically provisioned on the controller, and distributed to devices via the southbound interface.

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

Measurement Units:
N/A

See Also:
None

2.1.7. Proactive Flow Provisioning Mode

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

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

Measurement Units:
N/A

See Also:
None

2.1.8. Reactive Flow Provisioning Mode

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

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

Measurement Units:
N/A

See Also:
None

2.1.9. Path

Definition:
A path is a sequence of SDN nodes and links traversed by a flow.

Discussion:
As defined in RFC 2330, path is a sequence of the form < h0, l1, h1, ..., ln, hn >, where n >=0, h0 and hn is a Host, h1...hn-1 is an SDN Node, each li is a link between hi-1 and hi. A pair <li, hi> is termed a 'hop'. Note that path is a unidirectional concept.

Measurement Units:
N/A

See Also:
None

2.1.10. Standalone Mode

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

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

Measurement Units:
N/A

See Also:
None

2.1.11. Cluster/Redundancy Mode

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

Discussion:
In cluster mode, multiple controllers are teamed together for the purpose of load sharing and/or high availability. The controllers in

the group may work in active/standby (master/slave) or active/active (equal) mode depending on the intended purpose.

Measurement Units:
N/A

See Also:
None

2.1.12. Asynchronous Message

Definition:
Any message from the SDN node that is generated for network events.

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

Measurement Units:
N/A

See Also:
None

2.1.13. Test Traffic Generator

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

Discussion:
Test Traffic Generator can be an entity that interfaces with SDN Nodes to send/receive real-time network traffic.

Measurement Units:
N/A

See Also:
None

2.2. Test Configuration/Setup Terms

2.2.1. Number of SDN Nodes

Definition:
The number of SDN nodes present in the defined test topology.

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

Measurement Units:
N/A

See Also:
None

2.2.2. Test Iterations

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

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

Measurement Units:
N/A

See Also:
None

2.2.3. Test Duration

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

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

Measurement Units:
seconds

See Also:
None

2.2.4. Number of Cluster nodes

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

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

Measurement Units:
N/A

See Also:
None

2.3. Benchmarking Terms

This section defines metrics for benchmarking the SDN controller. The procedure to perform the defined metrics is defined in the accompanying methodology document.

2.3.1. Performance

2.3.1.1. Network Topology Discovery Time

Definition:
To measure the time taken to discover the network topology - nodes and links by a controller.

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

Measurement Units:
milliseconds

See Also:
None

2.3.1.2. Asynchronous Message Processing Time

Definition:
To measure the time taken by the controller to process an asynchronous message.

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

Measurement Units:
milliseconds

See Also:
None

2.3.1.3. Asynchronous Message Processing Rate

Definition:
To measure the maximum number of asynchronous messages that a controller can process within the test duration.

Discussion:
As SDN assures flexible network and agile provisioning, it is important to measure how many network events that the controller can handle at a time. This benchmark is obtained by sending asynchronous messages from every connected SDN nodes at full connection capacity for the given test duration. This test assumes that the controller will respond to all the received asynchronous messages.

Measurement Units:
Messages processed per second.

See Also:
None

2.3.1.4. Reactive Path Provisioning Time

Definition:
The time taken by the controller to setup a path reactively between source and destination node, expressed in milliseconds.

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

Measurement Units:
milliseconds.

See Also:
None

2.3.1.5. Proactive Path Provisioning Time

Definition:
The time taken by the controller to setup a path proactively between source and destination node, expressed in milliseconds.

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

Measurement Units:
milliseconds.

See Also:
None

2.3.1.6. Reactive Path Provisioning Rate

Definition:
Measure the maximum number of independent paths a controller can concurrently establish between source and destination nodes reactively within the test duration, expressed in paths per second.

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

Measurement Units:
Paths provisioned per second.

See Also:
None

2.3.1.7. Proactive Path Provisioning Rate

Definition:
Measure the maximum number of independent paths a controller can concurrently establish between source and destination nodes proactively within the test duration, expressed in paths per second.

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

Measurement Units:
Paths provisioned per second.

See Also:
None

2.3.1.8. Network Topology Change Detection Time

Definition:

The amount of time required for the controller to detect any changes in the network topology.

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

Measurement Units:
milliseconds

See Also:
None

2.3.2. Scalability

2.3.2.1. Control Sessions Capacity

Definition:
To measure the maximum number of control sessions the controller can maintain.

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

Measurement Units:
N/A

See Also:
None

2.3.2.2. Network Discovery Size

Definition:
To measure the network size (number of nodes, links and hosts) that a controller can discover.

Discussion:

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

Measurement Units:
N/A

See Also:
None

2.3.2.3. Forwarding Table Capacity

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

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

Measurement Units:
Maximum number of flow entries managed.

See Also:
None

2.3.3. Security

2.3.3.1. Exception Handling

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

Discussion:
This benchmark test is to be performed after obtaining the baseline performance of the performance tests defined in Section 2.3.1. This

benchmark determines the deviation from the baseline performance due to the handling of error or failure messages from the connected SDN nodes.

Measurement Units:
N/A

See Also:
None

2.3.3.2. Denial of Service Handling

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

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

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

See Also:
None

2.3.4. Reliability

2.3.4.1. Controller Failover Time

Definition:
The time taken to switch from an active controller to the backup controller, when the controllers work in redundancy mode and the active controller fails.

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

Measurement Units:
milliseconds.

See Also:
None

2.3.4.2. Network Re-Provisioning Time

Definition:
The time taken to re-route the traffic by the Controller, when there is a failure in existing traffic paths.

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

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

Measurement Units:
milliseconds.

See Also:
None

3. Test Coverage

	Speed	Scalability	Reliability
Setup	1. Network Topology Discovery 2. Reactive Path Provisioning Time 3. Proactive Path Provisioning Time 4. Reactive Path Provisioning Rate 5. Proactive Path Provisioning Rate	1. Network Discovery Size	

Operational	1. Asynchronous Message Proces-sing Rate 2. Asynchronous Message Proces-sing Time	1. Control Sessions Capacity 2. Forwarding Table Capacity	1. Network Topology Change Detection Time 2. Exception Handling 3. Denial of Service Handling 4. Network Re-Provisioning Time
Tear Down			1. Controller Failover Time

4. References

4.1. Normative References

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

[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.sdn-controller-benchmark-meth] Bhuvaneswaran.V, Anton Basil,

Mark.T, Vishwas Manral, Sarah Banks "Benchmarking Methodology for SDN Controller Performance", draft-ietf-bmwg-sdn-controller-benchmark-meth-00 (Work in progress), October 19, 2015

4.2. 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

5. IANA Considerations

This document does not have any IANA requests.

6. Security Considerations

Security issues are not discussed in this memo.

7. Acknowledgements

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

8. Authors' Addresses

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

Email: bhuvaneswaran.vengainathan@veryxtech.com

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

Philadelphia
PA 19113

Email: anton.basil@veryxtech.com

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

Email: mark.tassinari@hp.com

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

Email: vishwas@ionosnetworks.com

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

Email: sbanks@encrypted.net

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 17, 2017

A. Morton
AT&T Labs

March 16, 2017

Considerations for Benchmarking Virtual Network Functions and Their Infrastructure
draft-ietf-bmwg-virtual-net-05

Abstract

The 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 general purpose hardware.

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 September 17, 2017.

Copyright Notice

Table of Contents

1. Introduction 2
2. Scope 3
3. Considerations for Hardware and Testing 4
3.1. Hardware Components 4
3.2. Configuration Parameters 5
3.3. Testing Strategies 6
3.4. Attention to Shared Resources 7
4. Benchmarking Considerations 7
4.1. Comparison with Physical Network Functions 7
4.2. Continued Emphasis on Black-Box Benchmarks 8
4.3. New Benchmarks and Related Metrics 8
4.4. Assessment of Benchmark Coverage 9
4.5. Power Consumption 12
5. Security Considerations 12
6. IANA Considerations 12
7. Acknowledgements 12
8. Version history 13
9. References 13
9.1. Normative References 14
9.2. Informative References 14
Author's Address 15

1. Introduction

The Benchmarking Methodology Working Group (BMWG) has traditionally conducted laboratory characterization of dedicated physical implementations of internetworking functions (or physical network functions, PNFs). 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.

In the context of Virtualized Network Functions (VNF), the supporting Infrastructure requires general-purpose computing systems, storage systems, networking systems, virtualization support systems (such as hypervisors), and management systems for the virtual and physical resources. There will be many potential suppliers of Infrastructure systems and significant flexibility in configuring the systems for best performance. There are also many potential suppliers of VNFs, adding to the combinations possible in this environment. The separation of hardware and software suppliers has a profound implication on benchmarking activities: much more of the internal configuration of the black-box device under test (DUT) must now be specified and reported with the results, to foster both repeatability and comparison testing at a later time.

Consider the following User Story as further background and motivation:

"I'm designing and building my NFV Infrastructure platform. The first steps were easy because I had a small number of categories of VNFs to support and the VNF vendor gave HW recommendations that I followed. Now I need to deploy more VNFs from new vendors, and there are different hardware recommendations. How well will the new VNFs perform on my existing hardware? Which among several new VNFs in a given category are most efficient in terms of capacity they deliver? And, when I operate multiple categories of VNFs (and PNFs) *concurrently* on a hardware platform such that they share resources, what are the new performance limits, and what are the software design choices I can make to optimize my chosen hardware platform? Conversely, what hardware platform upgrades should I pursue to increase the capacity of these concurrently operating VNFs?"

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 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

At the time of this writing, BMWG is considering 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 VNFs instantiated and hosted in general-purpose hardware, using bare metal hypervisors [BareMetal] or other isolation environments such as Linux containers. An essential consideration is benchmarking physical and virtual network functions in the same way when possible, thereby allowing direct comparison. Benchmarking combinations of physical and virtual devices and functions in a System Under Test is another topic of keen interest.

A clearly related goal: the benchmarks for the capacity of a general-purpose platform 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 continued non-goal is any form of architecture development related to NFV and associated technologies in BMWG, consistent with all chartered work since BMWG began 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. The System Under Test (SUT) is composed of the hardware platform components, the VNFs installed, and many other supporting systems. It is critical to document all aspects of the SUT to foster repeatability.

3.1. Hardware Components

New Hardware components will become part of the test set-up.

1. High volume server platforms (general-purpose, 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.

The components above are subjects for development of specialized benchmarks which are focused on the special demands of network function deployment.

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 general-purpose platform to ensure repeatability and foster future comparisons, including but clearly not limited-to the following:

- number of server blades (shelf occupation)
- CPUs
- caches
- memory
- storage system
- I/O

as well as configurations that support the devices which host the VNF itself:

- Hypervisor (or other forms of virtual function hosting)
- Virtual Machine (VM)
- Infrastructure Virtual Network (which interconnects Virtual Machines with physical network interfaces, or with each other through virtual switches, for example)

and finally, the VNF itself, with items such as:

- specific function being implemented in VNF
- reserved resources for each function (e.g., CPU pinning and Non-Uniform Memory Access, NUMA node assignment)
- number of VNFs (or sub-VNF components, each with its own VM) in the service function chain (see section 1.1 of [RFC7498] for a definition of service function chain)
- number of physical interfaces and links transited in the service function chain

In the physical device benchmarking context, most of the corresponding infrastructure configuration choices were determined by the vendor. Although the platform itself is now one of the configuration variables, it is important to maintain emphasis on the networking benchmarks and capture the platform variables as input factors.

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 number of VM instances, the resources the VMs require, and the set-up/tear-down of network paths that support VM connectivity. Requests for and instantiation of new VMs, along with Releases for VMs hosting VNFs that are no longer needed would be an normal operational condition. In other words, benchmarking should include scenarios with production life cycle management of VMs and their VNFs and network connectivity in-progress, including VNF scaling up/down operations, as well as static configurations.

5. All physical things can fail, and benchmarking efforts can also examine recovery aided by the virtual architecture with different approaches to resiliency.

6. The sheer number of test conditionas and configuration combinations encourage increased efficiency, including automated testing arrangements, combination sub-sampling through an understanding of inter-relationships, and machine-readable test results.

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.

Note: The term "tester" has traditionally referred to devices dedicated to testing in BMWG literature. In this new context, "tester" additionally refers to functions dedicated to testing, which may be either virtual or physical. "Tester" has never referred to the individuals performing the tests.

The shared-resource aspect of test design remains one of the critical challenges to overcome in a way to produce useful results. Benchmarking set-ups may designate isolated resources for the DUT and other critical support components (such as the host/kernel) as the first baseline step, and add other loading processes. The added complexity of each set-up leads to shared-resource testing scenarios, where the characteristics of the competing load (in terms of memory, storage, and CPU utilization) will directly affect the benchmarking results (and variability of the results), but the results should reconcile with the baseline.

The physical test device remains a solid foundation to compare with results using combinations of physical and virtual test functions, or results using only virtual testers when necessary to assess virtual interfaces and other virtual functions.

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 VNFs and system implementations with their physical counterparts, identical benchmarks must be used. Since BMWG has already developed specifications for many network functions, 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 scale/capacity

with a given physical device, or whether some limit of scale was reached before the VNFs could achieve the comparable level. Again, implementation based-on different hypervisors or other virtual function hosting remain as critical factors in performance assessment.

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 observed in the dataplane). Examples include CPU/Core utilization, Network utilization, Storage utilization, and Memory Comitted/used. These "white-box" metrics provide one view of the resource footprint of a VNF. Note: The resource utilization metrics do not easily match the 3x4 Matrix, described in Section 4.4 below.

However, external observations remain essential as the basis for Benchmarks. Internal observations with fixed specification and interpretation may be provided in parallel (as auxilliary metrics), 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 general-purpose 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.

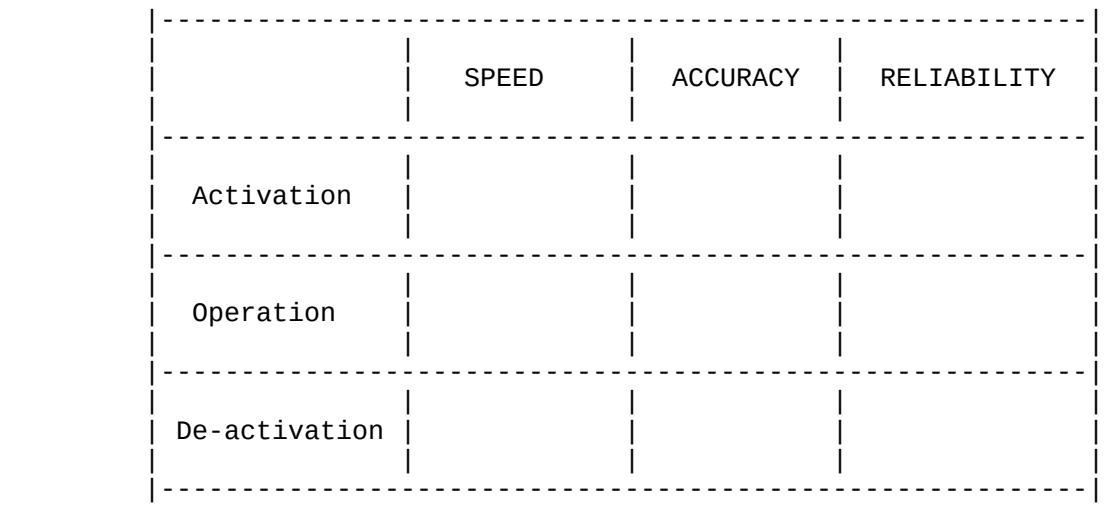
- Time to create a virtual network in the general-purpose infrastructure: This is a somewhat simplified version of existing benchmarks for convergence time, in that the process is initiated by a request from (centralized or distributed) control, rather than inferred from network events (link failure). The successful response time would remain dependent on dataplane observations to confirm that the network is ready to perform.

- Effect of verification measurements on performance: A complete VNF, or something as simple as a new policy to implement in a VNF, is implemented. The action to verify instantiation of the VNF or policy could affect performance during normal operation.

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 life cycle stage and the performance criteria they were designed to assess. The table below (derived from [X3.102]) provides a way to organize benchmarks such that there is a clear indication of coverage for the intersection of life cycle 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 life cycle 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 in BMWG [I-D.ietf-bmwg-sdn-controller-benchmark-meth], we discovered that metrics on measured size, capacity, or scale do not easily match one of the three columns above. Following discussion, this was resolved in two ways:

- Add a column, Scale, for use when categorizing and assessing the coverage of benchmarks (without measured results). Examples of this use are found in [I-D.ietf-bmwg-sdn-controller-benchmark-meth] and [I-D.vsperf-bmwg-vswitch-opnfv]. This is the 3x4 Matrix.

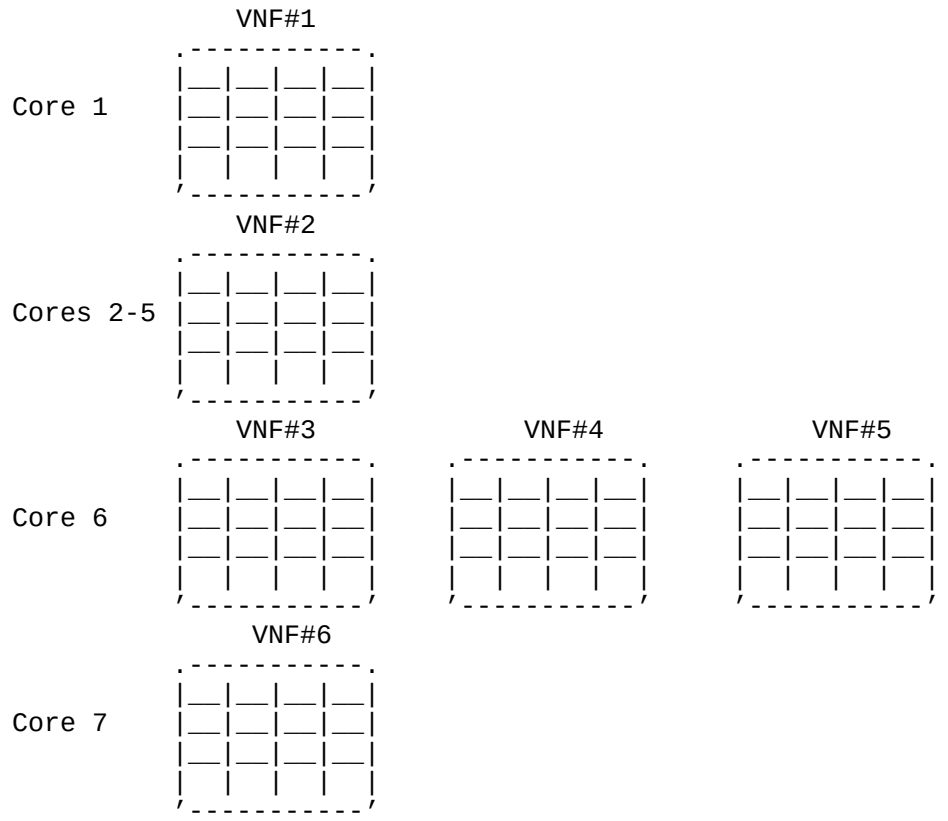
- If using the matrix to report results in an organized way, keep size, capacity, and scale metrics separate from the 3x3 matrix and incorporate them in the report with other qualifications of the results.

Note: The resource utilization (e.g., CPU) metrics do not fit in the Matrix. They are not benchmarks, and omitting them confirms their

status as auxilliary metrics. Resource assignments are configuration parameters, and these are reported seperately.

This approach encourages use of the 3x3 matrix to organize reports of results, where the capacity at which the various metrics were measured could 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).

For example, results for each VM and VNF could appear in the 3x3 matrix, organized to illustrate resource occupation (CPU Cores) in a particular physical computing system, as shown below.



The combination of tables above could be built incrementally, beginning with VNF#1 and one Core, then adding VNFs according to their supporting core assignments. X-Y plots of critical benchmarks would also provide insight to the effect of increased HW utilization. All VNFs might be of the same type, or to match a production environment there could be VNFs of multiple types and categories. In

this figure, VNFs #3-#5 are assumed to require small CPU resources, while VNF#2 requires 4 cores to perform its function.

4.5. Power Consumption

Although there is incomplete work to benchmark physical network function power consumption in a meaningful way, the desire to measure the physical infrastructure supporting the virtual functions only adds to the need. Both maximum power consumption and dynamic power consumption (with varying load) would be useful. The IPMI standard [IPMI2.0] has been implemented by many manufacturers, and supports measurement of instantaneous energy consumption.

To assess the instantaneous energy consumption of virtual resources, it may be possible to estimate the value using an overall metric based on utilization readings, according to [I-D.krishnan-nfvrg-policy-based-rm-nfviaas].

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. Further development was encouraged by Barry Constantine's comments following the IETF-92 BMWG session: the session itself was an affirmation for this memo. There have been many interesting contributions from Maryam Tahhan, Marius Georgescu, Jacob Rapp, Saurabh Chattopadhyay, and others.

8. Version history

(This section should be removed by the RFC Editor.)

version 05: Address IESG & Last Call Comments (editorial)

Version 03 & 04: address mininal comments and few WGLC comments

Version 02:

New version history section.

Added Memory in section 3.2, configuration.

Updated ACKs and References.

Version 01:

Addressed Ramki Krishnan's comments on section 4.5, power, see that section (7/27 message to the list). Addressed Saurabh Chattopadhyay's 7/24 comments on VNF resources and other resource conditions and their effect on benchmarking, see section 3.4. Addressed Marius Georgescu's 7/17 comments on the list (sections 4.3 and 4.4).

AND, comments from the extended discussion during IETF-93 BMWG session:

Section 4.2: VNF footprint and auxilliary metrics (Maryam Tahhan),
Section 4.3: Verification affect metrics (Ramki Krishnan);
Section 4.4: Auxilliary metrics in the Matrix (Maryam Tahhan, Scott Bradner, others)

9. References

9.1. 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, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

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

[RFC4689] Poretsky, S., Perser, J., Erramilli, S., and S. Khurana, "Terminology for Benchmarking Network-layer Traffic Control Mechanisms", RFC 4689, DOI 10.17487/RFC4689, October 2006, <http://www.rfc-editor.org/info/rfc4689>.

[RFC4737] Morton, A., Ciavattone, L., Ramachandran, G., Shalunov, S., and J. Perser, "Packet Reordering Metrics", RFC 4737, DOI 10.17487/RFC4737, November 2006, <http://www.rfc-editor.org/info/rfc4737>.

[RFC7498] Quinn, P., Ed. and T. Nadeau, Ed., "Problem Statement for Service Function Chaining", RFC 7498, DOI 10.17487/RFC7498, April 2015, <http://www.rfc-editor.org/info/rfc7498>.

9.2. Informative References

[BareMetal]
Popek, Gerald J.; Goldberg, Robert P. , , "Formal requirements for virtualizable third generation architectures". Communications of the ACM. 17 (7): 412-421. doi:10.1145/361011.361073.", 1974.

[I-D.ietf-bmwg-sdn-controller-benchmark-meth]
Vengainathan, B., Basil, A., Tassinari, M., Manral, V., and S. Banks, "Benchmarking Methodology for SDN Controller Performance", draft-ietf-bmwg-sdn-controller-benchmark-meth-03 (work in progress), January 2017.

[I-D.krishnan-nfvrg-policy-based-rm-nfviaas]
Krishnan, R., Figueira, N., Krishnaswamy, D., Lopez, D., Wright, S., Hinrichs, T., Krishnaswamy, R., and A. Yerra, "NFVIaaS Architectural Framework for Policy Based Resource Placement and Scheduling", draft-krishnan-nfvrg-policy-based-rm-nfviaas-06 (work in progress), March 2016.

[I-D.vsperf-bmwg-vswitch-opnfv]
Tahhan, M., O'Mahony, B., and A. Morton, "Benchmarking Virtual Switches in OPNFV", draft-vsperf-bmwg-vswitch-opnfv-02 (work in progress), March 2016.

[IPMI2.0] "Intelligent Platform Management Interface, v2.0 with latest Errata", http://www.intel.com/content/www/us/en/servers/ipmi/ipmi-intelligent-platform-mgt-interface-spec-2nd-gen-v2-0-spec-update.html, April 2015.

[RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <http://www.rfc-editor.org/info/rfc1242>.

[RFC5481] Morton, A. and B. Claise, "Packet Delay Variation Applicability Statement", RFC 5481, DOI 10.17487/RFC5481, March 2009, <http://www.rfc-editor.org/info/rfc5481>.

[RFC6049] Morton, A. and E. Stephan, "Spatial Composition of Metrics", RFC 6049, DOI 10.17487/RFC6049, January 2011, <http://www.rfc-editor.org/info/rfc6049>.

[X3.102] ANSI X3.102, , "ANSI Standard on Data Communications, User-Oriented Data Communications Framework", 1983.

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/

Network Working Group T. Kim
Internet-Draft E. Paik
Intended status: Informational KT
Expires: April 21, 2016 October 19, 2015

Considerations for Benchmarking High Availability of NFV Infrastructure
draft-kim-bmwg-ha-nfvi-00

Abstract

This documents lists additional considerations and strategies for benchmarking high availability of NFV infrastructure when network functions are virtualized and performed in NFV infrastructure.

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 April 21, 2016.

Copyright Notice

Table of Contents

1. Introduction 2
2. Considerations for Benchmarking High Availability of NFV Infrastructure 3
2.1. Definitions for High Availability Benchmarking Test . . . 3
2.2. Configuration Parameters for Benchmarking Test 3
3. High Availability Benchmarking test strategies 4
3.1. Single Point of Failure Check 4
3.2. Failover Time Check 5
4. Security Considerations 6
5. IANA Considerations 6
6. Normative References 6
Authors' Addresses 6

1. Introduction

As both amount and variety of traffic massively increase, operators are adopting SDN and NFV, the new paradigm of networking, in order to secure scalability and flexibility. Service provider and venders are developing SDN and NFV solutions and VNF(Virtual Network Function) to reduce CAPEX and OPEX, focusing on the increment of the scalability and flexibility of the network with programmable networking.

To replace the legacy network devices with VNFs and to select the fittest one from various products of vender, operators want to ensure the availability and resiliency of the VNF products and their infrastructures. There also exist fears on the immeasurable failures.

Among VNFs, vEPC is getting many attentions and some telecommunications company already deployed vEPC partially. Currently in 4G mobile communication, the availability reaches 99.9999%; downtime being 3 seconds per year. Therefore, VNFs like vEPC (virtual Evolved Packet Core) must guarantee the 6-nines to replace hardware dedicated network functions. From the telecommunication company's point of view, the availability is the most important feature, and the benchmarking tests for the high availability of VNFs and NFV infrastructure are also important. This document investigates considerations for high availability of NFV Infrastructure benchmarking test.

2. Considerations for Benchmarking High Availability of NFV Infrastructure

This section defines and lists considerations which must be addressed to benchmark the high availability of VNFs from the NFV infrastructure perspective.

2.1. Definitions for High Availability Benchmarking Test

Generally, availability is defined as follows, where MTBF stands for Mean Time Between Failure) and MTTR stands for Mean Time To Recovery.

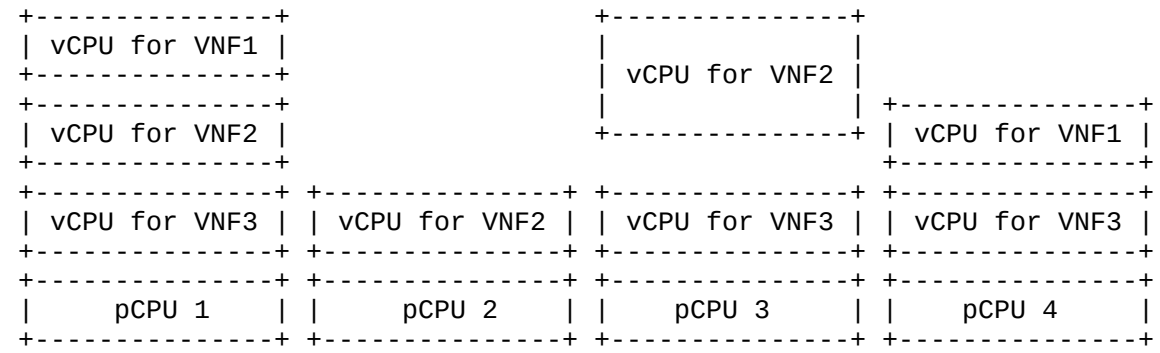
Availability : MTBF / (MTBF + MTTR)

A failover procedure is as follows.

Failure -> Detection -> Isolatation -> Recovery, therefore the time to take failover starts from the time when a failure happens.

2.2. Configuration Parameters for Benchmarking Test

- Types of VNFs; depending on the type of VNF, followings are different.
  1. What kind of operations they do
  2. How many CPUs, MEMs, Storages they need
  3. What kind of traffic pattern they usually face
- The specification of the physical machine which VMs
- The mapping ratio of hardware resources to VMs(virtual machine) where VNF runs, such as vCPU:pCPU (virtual CPU to physical CPU), vMEM:pMEM (virtual memory to physical memory), vNICs as shown below.
- Types of hypervisor and the different limitations of their roles.
- Cloud Design Pattern of NFVI
- The composition of network functions in VNFs : for example, sometimes in vEPC implementations, PGW(Packet Data Network Gateway) and SGW(Serving Gateway) are combined or PGW+SGW+MME.



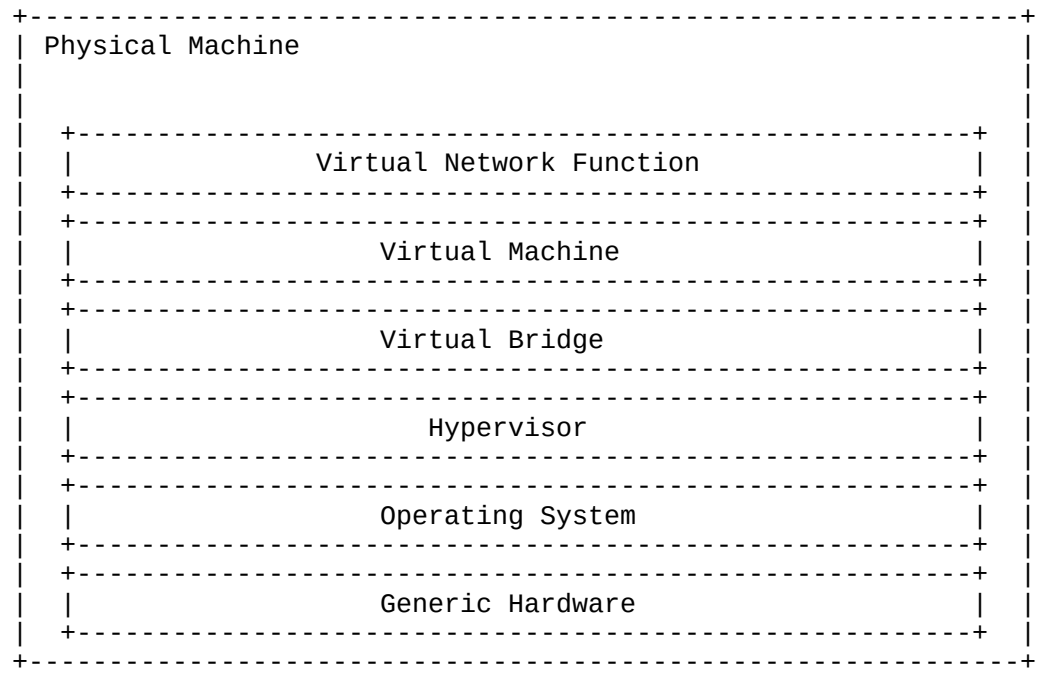
3. High Availability Benchmarking test strategies

This section discusses benchmarking test strategies for high availability of NFV infrastructure. For the continuity of the services, these two must be checked.

3.1. Single Point of Failure Check

All devices and software have potential failures, therefore, redundancy is mandatory. First, the redundancy implementation of every sing point of NFV infrastructure must be tested as shown below.

- Hardware
  - Power supply
  - CPU
  - MEM
  - Storage
  - Network :NICs, ports, LAN cable ..
- Software
  - The redundancy of VNFs
  - The redundancy of VNFs path
  - The redundancy of OvS
  - The redundancy of vNICs
  - The redundancy of VMs



3.2. Failover Time Check

Even though the components of NFV infrastructure are redundant, failover time can be long. For example, when a failure happens, the VNF with failure stops and should be replaced by backup VNF but the time to be shifted to the new VNF can be varied with the VNF; stateless or stateful. Namely, redundancy does not guarantees high availability and short failover time is required to reach high availability. This section discusses strategy about measuring failover time.

In order to measure the failover time presicely, the time when failure happens must be defined. Followings are three different criteria which is the time when failure happens.

1. The time starts when failure actually happens

2. The time starts when failure detected by manager or controller

3. The time starts when failure event alerts to the operator

As the actual operations in VNFs and NFV infrastructure start to be changed when failure happens, the precise time of the failure happened must be the 1. When measuring the failover time, it starts

from the time when the failures happens at a point in NFV infrastructure or VNF itself.

4. Security Considerations

TBD.

5. IANA Considerations

No IANA Action is requested at this time.

6. Normative References

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

Authors' Addresses

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

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

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

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

NFVRG
Internet-Draft
Intended status: Informational
Expires: April 21, 2016

R. Rosa
C. Rothenberg
Unicamp
R. Szabo
Ericsson

October 19, 2015

VNF Benchmark-as-a-Service
draft-rorosz-nfvrg-vbaas-00

Abstract

Network Function Virtualization (NFV) promises the delivery of flexible chaining of Virtualized Network Functions (VNFs) with portability, elasticity, guaranteed performance, reliability among others. Service providers expect similar behavior from NFV as their current appliances based infrastructure, however, with the ease and flexibility of operation and cost effectiveness. Managing for predictable performance in virtualized environments is far from straightforward. Considering resource provisioning, selection of an NFV Infrastructure Point of Presence to host a VNF or allocation of resources (e.g., virtual CPUs, memory) needs to be done over virtualized (abstracted and simplified) resource views. In order to support orchestration decisions, we argue that a framework for VNF benchmarking is need.

Table of Contents

1. Introduction 2
2. Terms and Definitions 3
3. Motivation 4
4. Problem Statement and Challenges 5
5. Proposed Approach 6
6. Use Case: Unify Modelling 11
7. Related drafts and open source projects 16
8. IANA Considerations 18
9. Security Considerations 18
10. Acknowledgement 18
11. Informative References 18
Authors' Addresses 20

1. Introduction

Network Function Virtualization (NFV) pursues delivering Virtualized Network Functions (VNFs) in NFV Infrastructure (NFVI) where requirements of portability, performance, elasticity, among others, are demanded by carriers and network operators [ETS14a]. Particularly, in the short term, performance and elasticity are important factors to realize NFV concepts [BVLS15]. Associated to these goals VNFs need to be deployed with predictable performance, taking into account the variable processing overheads of virtualization and the underlying available NFVI resources.

In essence, NFV Management and Orchestration (MANO) plane manages infrastructure and network services resources [ETS14c]. This involves taking into account performance considerations for VNF-FGs allocation (e.g., fulfilling NFVI resources by orchestration demands). Furthermore, when attempting to manage service assurance of embedded VNF-FGs [PSPL15] MANO tasks may depend on analytics

measurements for scaling and migration purposes. As a consequence Network Function Virtualization Orchestrator (NFVO) needs a coherent understanding of performance vs. resource needs for VNFs specific to NFVI Points of Presence (PoPs).

In what follows we motivate the need for a VNF benchmarking, define related terms and introduce a framework proposal.

2. Terms and Definitions

The reader is assumed to be familiar with the terminology as defined in the European Telecommunications Standards Institute (ETSI) NFV document [ETS14b]. Some of these terms, and others commonly used in this document, are defined below.

NFV: Network Function Virtualization - The principle of separating network functions from the hardware they run on by using virtual hardware abstraction.

NFVI PoP: NFV Infrastructure Point of Presence - Any combination of virtualized compute, storage and network resources.

NFVI: NFV Infrastructure - Collection of NFVI PoPs under one orchestrator.

VIM: Virtualized Infrastructure Manager - functional block that is responsible for controlling and managing the NFVI compute, storage and network resources, usually within one operator’s Infrastructure Domain (e.g. NFVI-PoP).

NFVO: NFV Orchestrator - functional block that manages the Network Service (NS) lifecycle and coordinates the management of NS lifecycle, VNF lifecycle (supported by the VNFM) and NFVI resources (supported by the VIM) to ensure an optimized allocation of the necessary resources and connectivity

VNF: Virtualized Network Function - a software-based network function.

VNFD: Virtualised Network Function Descriptor - configuration template that describes a VNF in terms of its deployment and operational behaviour, and is used in the process of VNF on-boarding and managing the lifecycle of a VNF instance.

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

MANO: Management and Orchestration - In the ETSI NFV framework [ETS14c], this is the global entity responsible for management and orchestration of NFV lifecycle.

VBaaS: VNF Benchmark-as-a-Service - is a VNF Profile extraction service of a Virtualized Infrastructure Manager (VIM) overseeing an NFVI PoP.

VNF-BP: VNF Benchmarking Profile - the specification how to measure a VNF Profile. VNF-BP may be specific to a VNF or applicable to several VNF types. The specification includes structural and functional instructions, and variable parameters (metrics) at different abstractions (e.g., vCPU, memory, bandwidth, delay; session, transaction, tenants, etc.).

VNF Profile: is a mapping between virtualized resources (e.g., vCPU, memory) and VNF performance (e.g., bandwidth, delay between in/out or ports) at a given NFVI PoP. An orchestration function can use the VNF Profile to select host for a VNF and to allocate the desired resources to deliver a given (predictable) service quality.

3. Motivation

Let us take an example where a VNF Developer creates a new code base for a VNF. She is able to optimize her VNF for multiple execution environments and offers these as a set of different but equivalent implementations of VNF1 for Service Providers (SPs) (see Figure 1). Orchestration managers (e.g., NFVO) must know adequate infrastructure resources to allocate when deploying VNF1, specially if it belongs to a network service (VNF-FG) with a target SLA, e.g., min throughput or max latency. However, instances of VNF1 optimized for heterogeneous NFVI PoPs may need different resources/settings for the same behavior (performance). Orchestrator (NFVO) needs to take into account all these requirements to embed VNFs and allocate proper infrastructure resources in accordance with network service intents (SLAs).

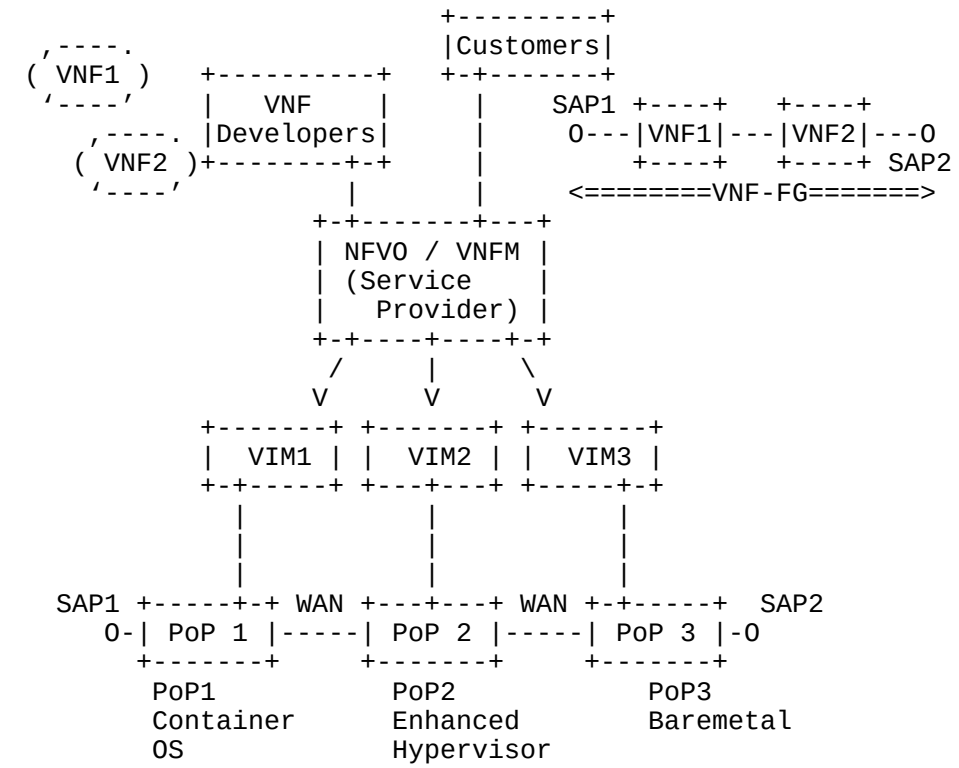


Figure 1: NFV MANO: Customers and VNF Developers

4. Problem Statement and Challenges

Previous section motivates NFVO needs of knowledge about correlations between VNFs performance and NFVI PoP resources.

Suppose that specifications, based on developers' definitions, exist in VNF Descriptors [ETS14d] describing performance associated to specific NFVI PoP resources. Such VNF, when deployed, may present oscillations in its performance because of changes in the underlying infrastructure (due to varying customers workloads, traffic engineering, scaling/migration, etc). Therefore, we believe we need descriptions on how to benchmark VNFs' resources and performance characteristics (e.g., VNF Profiles) on demand and in a flexible process supporting different NFVI PoPs, which could create service level specification associations.

Some of the associated challenges are:

Consistency: Will the VNF deployed at a NVFI PoP deliver the performance given in the VNF Profile for the NFVI PoP under different workloads? The performance of a VNF may not only depend on the resources allocated to the VNF but on the overall workload at the target NFVI PoP.

Stability: Benchmark measurements need to present consistent results when applied over different NFVI PoPs. How express benchmarking descriptions to fit in different virtualized environments? How to uniformly handle service/resource definitions and metrics?

Goodness: A given VIM / NFVI PoP may virtualize (i.e., hide) certain resource details; hence a benchmark evaluation might be executed in a different resource set of the NFVI PoP unlike the final production environment. How good benchmark results can correspond to actual workloads in virtualized environments?

VNF benchmarking service description: How to define the VNF benchmarking process at the abstraction of the corresponding component (e.g., VIM)? How to offer benchmarking as a service to be consumed by the different stakeholders?

VNF benchmarking versus monitoring: How can a VNF benchmarking process complement continuous monitoring of NFVI? When does a VNF benchmarking process surpass in performance and costs the execution of continuous monitoring process? And for which type of NFVI PoPs and/or VNFs is it necessary/sufficient?

5. Proposed Approach

Definition: VNF Benchmarking as a Service (VBaaS) -- a standalone service that can be hosted by orchestration managers (e.g., NFVO) to report a VNF Profile. VBaas might take inputs consisting of VNF-FGs determining VNFs to be evaluated under specific NFVI targets. As output, VBaaS returns a Service Graph based on the VNF-Benchmark Profile (VNF-BP) containing different parameters used to evaluate all required candidates (PoPs) for the specified VNF. Such interactions are abstracted by the VBaaS API. A VBaaS Service Graph is deployed by orchestration managers (NFVO), VBaaS receives benchmarking results and builds VNF Profiles. VNF performance values or VNF resource values can be provided, and VBaaS can only complement the VNF-FG missing parts to build VNF-BPs, or report an existing up-to-date VNF Profile for pre-defined values.

In comparison with Figure 1, the VBaaS approach introduces two information bases, a VBaaS component and an API to interact with NFVO, as shown in Figure 2. On one hand, the Network Function

Information Base (NF-IB) holds the VNF Profiles and on the other hand the VBaaS-IB holds the VNF Benchmark Profiles (VNF-BPs).

Definition: VNF Profile -- is a mapping between virtualized resources of a certain NFVI PoP (e.g., virtual machine with vCPU, memory) and VNF performance (e.g., throughput, delay, packet loss between in/out or ports).

An orchestration function can use the VNF Profile to select hosts for VNFs in a VNF-FG and to allocate the necessary resources to deliver a given (predictable) service quality (SLA).

Figure 2 shows an example VNF1 and VNF2, which are implemented by a VNF Developer, who provides metrics for VBaaS build VNF-BPs. The VNF-BP is then used by VBaaS when NFVO needs to evaluate the available NFVI-PoPs for different resources configuration. In this case, NFVO instantiates the VNF-FG (VBaaS Service Graph) as defined in VNF1-BP; configures the variable parameters in the configuration template, e.g., CPU and memory values and handles the configuration file to the measurement controller (part of the VNF-FG definition) to execute the benchmarking. The measurement controller make the measurement agents execute the benchmark tests (the execution environment might also be monitored by the agents) and reports the results back to the VBaaS. As a result, resource and performance associations for the VNF is created for the evaluated the NFVI PoPs into the NF-IB. Figure 2 shows, as an example, that VNF1 needs 2CPU (cores) and 8GByte of memory to handle up 10Mbps input rate with transaction delay less than 200ms at NFVI PoP1.

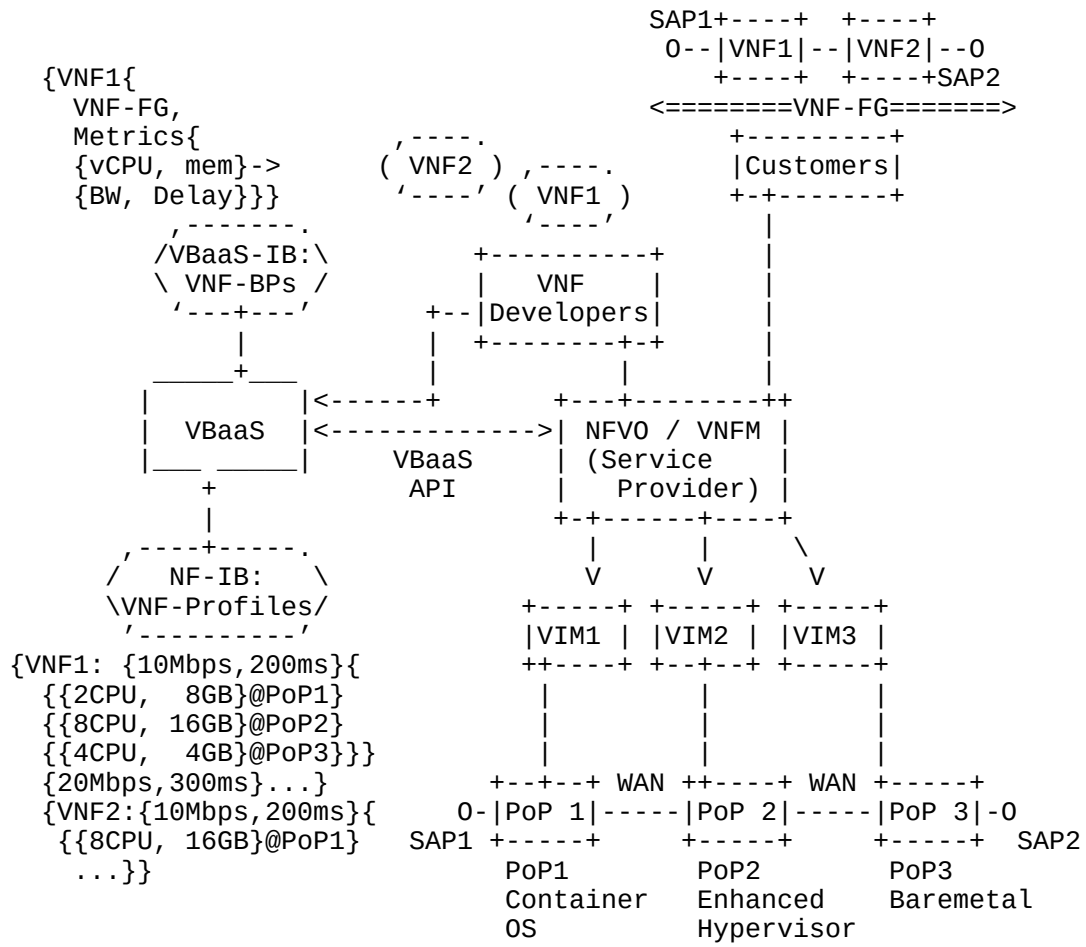


Figure 2: NFV MANO and VBaaS Approach

On the left side of the NFVO in Figure 2 is the resulting NF-IB by the application of VBaaS-IB profiles on the extraction of VNFs performance characteristics in different NFVI PoPs. Basically, a VNF profile corresponds to performance metrics in different environments. For example, VNF1 to offer 10Mbps bandwidth with 200ms latency needs (if deployed) in PoP1 2CPU (cores) and 8GB of memory, meanwhile in PoP2 requires 8CPU and 16GB. Similar results can be presented for VNF2, if catalogued as a VNF profile. In this scenario, VBaaS can be seen featuring VNFs in different PoPs in an on-demand composition of profiles, for example, to be used in provisioning VNF-FGs containing VNF1 or VNF2, attending PoPs 1, 2 or 3 resources consumption. Such profiles present performance metrics of VNFs in different NFVI PoPs

when, for example, NFVOs need to deploy particular VNF-FGs that contain elements already certified in NF-IB profiles.

Definition: VNF Benchmarking Profile (VNF-BPs) -- the specification of how to measure the VNF Profile for a given VNF. The specification includes structural (e.g., measurement components defined as a VNF-FG) and functional (e.g., measurement process definitions and configurable parameters) instructions, and variable parameters (metrics) at different abstractions (e.g., vCPU, memory, bandwidth, delay; session, transaction, tenants, etc.).

Note: a VNF-BP may be specific to a VNF or applicable to several VNF types.

Figure 3 presents details of VBaaS API regarding the interactions with NFVO and the construction of VBaaS Service Graphs based on VNF-BPs. As a VNF-FG (deployment request), NFVO demands VNF1 allocated to a specific PoP1 and the remaining virtualized view of the current resources for the VNF-FG deployment (e.g., interconnected PoPs). VBaaS (Manager Service) looks for VNF1-BP in VBaaS-IB, complements the requested VNF-FG with benchmark instances (described below) to perform evaluations and monitor resources in VNF1 and PoP1. NFVO receiving a VNF-FG reply (deployment order of a VBaaS Service Graph) instantiates it as a normal provisioning process of network services. Benchmark reports of the deployed VNF-FG are sent to VBaaS and it defines the construction of the VNF-Profile associated with that evaluation.

We believe in the existence of VBaaS instances (VNFs) which compose a VBaaS Service Graph to perform probing and monitoring activities on selected targets (VNF in a specific NFVI PoP). Those are defined as Manager, Monitor and Agent, described in details below.

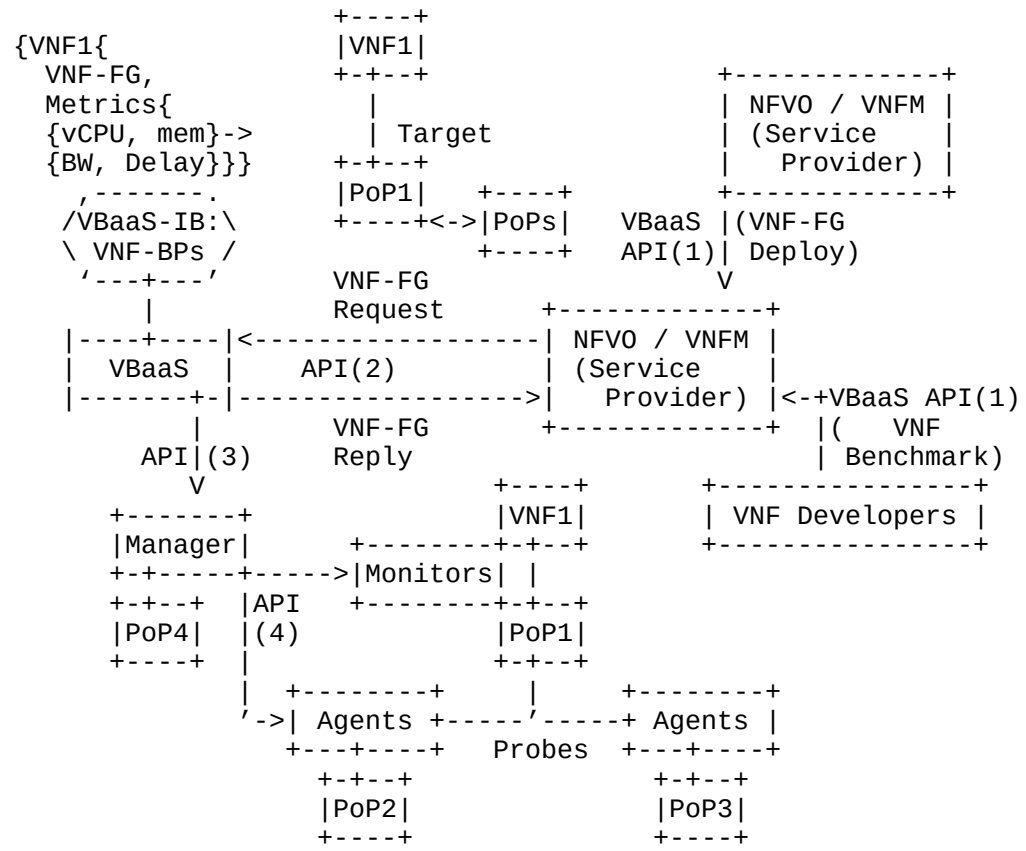


Figure 3: VBaaS APIs

Agent -- executes active probes using benchmark tools to collect network and system performance metrics by interacting with other agents. While a single Agent is capable of performing localized benchmarks (e.g., Stress tests on CPU, memory, I/O), the interaction among distributed agents enable the generation and collection of end-to-end metrics (e.g., packet loss, delay). Consider here the possibility of one end be the VNF itself where, for example, one-way packet delay is evaluated. For instance, it can be used for probe measurements of throughput and latency in a multi-point distributed topology. Agent's APIs are open and extensible (e.g., to plug-and-bench new tools and metrics) and receive configuration and run-time commands from the Manager for synchronization purposes (e.g., test duration/repetition) with other Agent and Monitor instances.

Monitor -- when possible it is placed inside the target VNF and NFVI PoP to perform active and passive monitoring and metric collection based on benchmarks evaluated according to Agents' workloads. For instance, to monitor CPU utilization of NFVI PoP considering packet length traffic variations sent by Agents. Different from the active approach of Agents that can be seen as generic benchmarking VNFs, monitors observe particular properties according to NFVI PoPs and VNFs capabilities. Monitors can plug-and-bench new tools/metrics and maintain an interface with Manager to synchronize its activities with Agents. Besides, they can be instantiated, if allowed, inside the target VNF (e.g., as a plug-in process in a virtualized environment) to check internal behaviours, alongside the VNF deployment environment, i.e., NFVI PoP, or in both places.

Manager -- is responsible for (i) the coordination and synchronization of activities between agents and monitors, (ii) collecting all benchmark raw results, and (iii) aggregating the inputs to construct a profile report that correlates different metrics as required by the VNF-BP. Therefore, it executes the main configuration, operation and management actions to deliver the results as specified by NFVI PoPs or VNF-BPs (e.g., instantiation of agents and monitors activities along- side tools and metrics configuration). Manager uses APIs to read and set configuration parameters for benchmarking instances such as metric parameters (e.g., bandwidth or packet loss probes) according to the VBaaS process. APIs are also used to receive benchmark instructions from VBaaS and send the reports of finished benchmark procedures.

The benchmarking process, however, can be offered as a service (see API (1) in right side of Figure 3). For example, a NFVO may be able to handle a VBaaS-IB and offer a VNF Benchmarking as a Service to its client, e.g., another NFVO. Similarly, a second NFVO may store benchmark measurements in its NF-IB so it can report it to multiple clients if it shares resources among other NFVOs. Alternatively, an NFVO may offer to it's developers the VNF Benchmarking as a Service, to assess the performance of a VNF in the operator's environment. This may be part of a DevOps primitive (e.g., VNFs continuous integration).

6. Use Case: Unify Modelling

Unify looks for a generic description of compute and network resources as a yang model named virtualizer [Szab15]. It allows the aggregation of resources (e.g., nodes capabilities and NF instances) and their abstractions in a structure nicknamed Big Switch with Big Software (BiS-BiS) (joint software and network resource abstraction

with a joint control API), which allows recursive control plane structuring for multi-domain hierarchies. For example, supported network functions of a joint compute and network virtualization (a BiS-BiS node) can be described with their associated resources and performance mapping, e.g., {cpu, memory, storage} -> {bandwidth, delay} (see Figure 5 based on virtualizer xml example in [Szab15]).

In this example, the interface between NFVOs (Producer and Consumer of VBaaS API (1)), as shown in Figure 3, is described by the UNIFY virtualization model [Szab15]. Figure 4 shows an example netconf message exchange over the domain virtualization model. First the consumer reads in (1) and (2) the virtualization view, which includes the supported NFs reported in the capabilities section. Next in (3) the consumer requests a capability report of NF1 at a given virtualized node (UUID001), by defining the required delay and bandwidth performance parameters of the NF1 (see Figure 5. This is a dimensioning request according to [ETS14t], i.e., the VBaaS service must evaluate and provide cpu, memory and storage values at the given virtualized node to deliver the given network performance characteristics. The start of the VBaaS process is acknowledged by a (4) rpc-ok. Once the VBaaS process completes a notification is sent (5) to the consumer, which reads in (6) and (7) the updates to the NF1 capability report, i.e., the missing cpu, memory and storage parameters (see Figure 6).

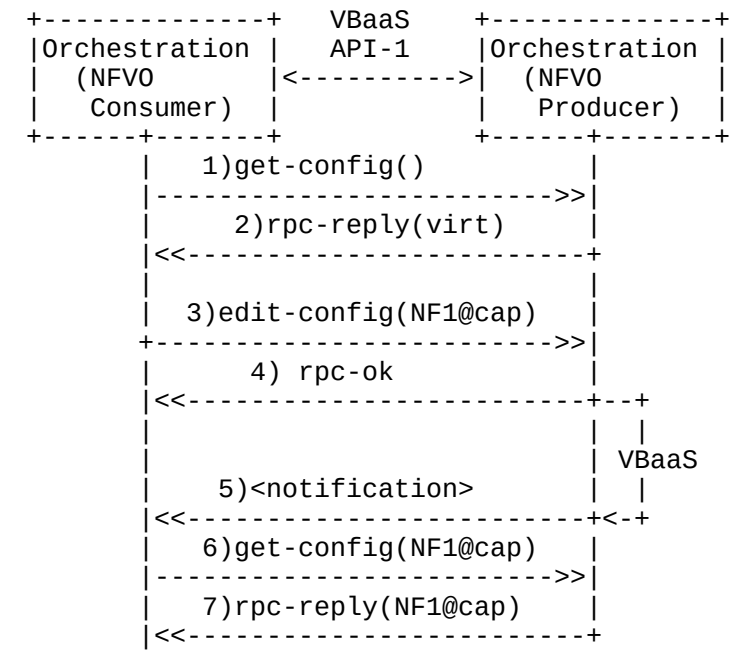


Figure 4: Sequence Diagram of VBaaS API (1) - Multi-Domain Interaction

```
<virtualizer xmlns="http://fp7-unify.eu/framework/virtualizer">
    <id>UUID001</id>
    <name>Single node simple VBaaS request</name>
    <nodes>
        <node>
            <id>UUID11</id>
            <name>single Bis-Bis node</name>
            <type>BisBis</type>
            <ports>
                ...
            </ports>
            <resources>
                <cpu>12</cpu>
                <mem>64 GB</mem>
                <storage>20 TB</storage>
            </resources>
            <capabilities>
              <supported_NFs>
                      <node>
                          <id>NF1</id>
                          <name>vFirewall</name>
```

```
                            <type>Stateful virtual firewall C</type>
                            <ports>
                                <port>
                                    <id>1</id>
                                    <name>in</name>
                                    <port_type>port-abstract</port_type>
                                    <capability>...</capability>
                                </port>
                                <port>
                                    <id>2</id>
                                    <name>out</name>
                                    <port_type>port-abstract</port_type>
                                    <capability>...</capability>
                                </port>
                            </ports>
                            <resources>
                                        <cpu> ? </cpu>
                                        <mem> ? </mem>
                                        <storage> ? </storage>
                                    </resources>
                        <links>
                                        <link>
                                            <id>int0</id>
                                            <name>internal horizontal</name>
                                            <src>../../ports/port[id=1]</src>
                                            <dst>../../ports/port[id=2]</dst>
                                            <resources>
                                                <delay> 20 us </delay>
                                                <bandwidth> 10 GB </bandwidth>
                                            </resources>
                                        </link>
                                    </links>
                                </node>
                            </supported_NFs>
                        </capabilities>
                    </node>
                </nodes>
            </virtualizer>
```

Figure 5: VBaaS_Request(NF-FG-1)

```
            <virtualizer xmlns="http://fp7-unify.eu/framework/virtualizer">
                <id>UUID001</id>
                <name>Single node simple VBaaS report</name>
                <nodes>
                    <node>
```

```
              <id>UUID11</id>
              <name>single Bis-Bis node</name>
              <type>BisBis</type>
              <capabilities>
                <supported_NFs>
                        <node>
                            <id>NF1</id>
                            <name>vFirewall</name>
                            <type>Stateful virtual firewall C</type>
                            <ports>
                                <port>
                                    <id>1</id>
                                    <name>in</name>
                                    <port_type>port-abstract</port_type>
                                    <capability>...</capability>
                                </port>
                                <port>
                                    <id>2</id>
                                    <name>out</name>
                                    <port_type>port-abstract</port_type>
                                    <capability>...</capability>
                                </port>
                            </ports>
                            <resources>
                                        <cpu>2</cpu>
                                        <mem>8 GB</mem>
                                        <storage>20 GB</storage>
                                    </resources>
                        <links>
                                        <link>
                                            <id>int0</id>
                                            <name>internal horizontal</name>
                                            <src>../../ports/port[id=1]</src>
                                            <dst>../../ports/port[id=2]</dst>
                                            <resources>
                                                <delay> 20 us </delay>
                                                <bandwidth> 10 GB </bandwidth>
                                            </resources>
                                        </link>
                                </links>
                        </node>
                </supported_NFs>
              </capabilities>
          </node>
      </nodes>
  </virtualizer>
```

Figure 6: VBaaS_Request(NF-FG-2)

Following the above example and the terminology of [ETS14t], the different tasks of performance verification, benchmarking and dimension can be mapped as follows:

Verification: Both {cpu, memory, storage} and {delay, bandwidth} parameters are provided and the VBaaS system verifies if the given association is correct or not. In the case of a failure the entry might be removed or updated with correct values.

Benchmarking: Where {cpu, memory, storage} parameters are provided and the VBaaS service fills-in the corresponding {delay, bandwidth} performance parameters. Note, such request might create more entries, like an entry with minimal delay and an entry with maximum bandwidth.

Dimensioning: Where {delay, bandwidth} performance parameters are provided and the VBaaS service fills-in the corresponding {cpu, memory, storage} resource parameters (as shown in the above example).

7. Related drafts and open source projects

Definetly, VBaaS intersects IETF Benchmarking Methodology Work Group (BMWG) goals. For example, in [Mor15], "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure" presents relevant statements concerning, for example, % of utilization of NFVI PoPs resources; noisy behavior caused by VNFs operations and misbehavior in NFVI PoPs; long-term tests which represent service continuity guarantees for VNFs; and considerations on how shared resources dependencies may affect consistent benchmark results. All these NFV benchmarking aspects are relevant for VBaaS, as well other current [Tah15] and future recommendations from BMWG.

The main DevOps Principles for Software Defined Infrastructures (SDIs) (e.g., NFV and SDN), well explained in [Meir15], are associated with the main design concepts of VBaaS. By definition, DevOps principles look for deployment with repeatable and reliable processes (VNF-BPs), as well modular, e.g., VNF Profiles assisting orchestration decisions. Besides, it also stands for development and testing procedures to assist VNF Developers (VBaaS API) specify modular applications/components (VNF-BPs) against production-like systems. In addition, DevOps intends to define ways of monitor and validate operational quality of services before and after services deployment, e.g., respectively applying VNF-BPs in multiple NFVI PoPs and over instantiated VNFs. All these aspects illustrate how VBaaS can help in amplifying the cycle between VNFs continuous changes of

development and Operators desires for infrastructure stability and reliability, therefore, helping the consolidation of DevOps principles in SDIs.

In OPNFV (Open Platform for NFV) community, projects dedicated to VIM and NFVI tests are being developed, such as vSwitchPerf, Pharos, Functest, among others. Specifically, Yardstick [Yard15] is closely related with VBaaS, because it stands for a common definition of tests to verify the infrastructure compliance when running VNF applications. While VBaaS is still building its concepts, Yardstick already defined requirements, methodologies (including metrics for compute, network and storage domains), besides functional code. The specification of yaml files to define structures like scenarios, contexts and runners, as a Yardstick nomenclature to compose tests, represents associations with VNF-BPs concepts. As Yardstick future work intends to apply tests in compute and storage domains, and store tests results, it is on its way the possible intersection with VBaaS concepts, like VNF Profiles.

T-NOVA [TNOVA15] pursues the definition of a marketplace for VNFs (and turn them instanced as a service). Regarding the description of capabilities for the allocation of VNFs, T-NOVA intends to allow network services and functions, by a variety of developers, to be published and brokered/traded, allowing customers to browse and select services that best match their needs (negotiating SLAs and billing models). One of the main marketplace functions is the publication of resources and network function advertisements in order to compose and package NFV services. VBaaS can enhance VNFs capabilities publication and certification models of their performance when constructing a diverse set of VNF-Profiles. Consequently, VNF flavors can be defined by an enhanced set of VBaaS capabilities for smarter placements, e.g., exposing VBaaS interfaces for developers evolve VNF-Profiles.

Unify [Szab15] presents a recursive compute and network joint virtualization and programming view of generic infrastructures (e.g., data centers), considering NFV as an example. Big Switch with Big Software (BiS-BiS) represents the abstraction of such virtualization. The draft presents an Yang model to describe a Network Function Forwarding Graph (NF-FG) representing NFV resources. Complementary, as part of main Unify goals, problem statements and challenges about the unification of carrier and cloud networks [Csas15] show the need for a high level of programmability beyond policy and service descriptions. As presented, VBaaS can contribute for such abstractions since it detaches from monolithic views of infrastructure resources allowing the recursive definitions of VBaaS Service Graphs and, consequently, the decomposition of benchmark tasks in multiple domains. Besides, as stated by Unify measurements

and analytics challenges, VBaaS poses as a solution of tools for planning, testing, and reliability assurance of NFVIs.

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

TBD

10. Acknowledgement

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

This work is partially supported by FP7 UNIFY, a research project partially funded by the European Community under the Seventh Framework Program (grant agreement no. 619609). The views expressed here are those of the authors only. The European Commission is not liable for any use that may be made of the information in this document.

11. Informative References

[BVLS15] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations", Communications Magazine, IEEE, vol. 53, no. 2, pp. 90-97 , February 2015.

[Csas15] R. Szabo, A. Csaszar, K. Pentikousis, M. Kind, D. Daino, Z. Qiang, H. Woesner, "Unifying Carrier and Cloud Networks: Problem Statement and Challenges", March 2015, <https://tools.ietf.org/html/draft-unify-nfvrg-challenges-01>.

[ETS14a] ETSI, "Architectural Framework - ETSI GS NFV 002 V1.2.1", Dec 2014, <http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01-_60/gs_NFV002v010201p.pdf>.

[ETS14b] ETSI, "Terminology for Main Concepts in NFV - ETSI GS NFV 003 V1.2.1", Dec 2014, <http://www.etsi.org/deliver/etsi_gs/NFV/001_099-/003/01.02.01_60/gs_NFV003v010201p.pdf>.

[ETS14c] ETSI, "Management and Orchestration - ETSI GS NFV-MAN 001 V1.1.1", Dec 2014, <http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf>.

[ETS14d] ETSI, "Virtual Network Functions Architecture - ETSI GS NFV-SWA 001 V1.1.1", Dec 2014, <http://www.etsi.org/deliver/etsi_gs/NFV-SWA/001_099/001/01.01.01_60/gs_NFV-SWA001v010101p.pdf>.

[ETS14t] ETSI, "NFV Pre-deployment Testing - ETSI GS NFV TST001 V0.0.13", Sept 2015, <http://docbox.etsi.org/ISG/NFV/Open/DRAFTS/TST001_-_Pre-deployment_Validation/NFV-TST001v0013.zip>.

[Meir15] C. Meirosu, A. Manzalini, J. Kim, R. Steinert, S. Sharma, G. Marchetto, I. Papafili, K. Pentikousis, S. Wright, "DevOps for Software-Defined Telecom Infrastructures", July 2015, <http://www.ietf.org/id/draft-unify-nfvrg-devops-02.txt>.

[Mor15] A. Morton, "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure", February 2015, <http://tools.ietf.org/html/draft-morton-bmwg-virtual-net-03>.

[PSPL15] E. Paik and M-K. Shin and S. Pack and S. Lee, "Resource Management in Service Chaining", March 2015, <http://tools.ietf.org/html/draft-lee-nfvrg-resource-management-service-chain-01>.

[Szab15] R. Szabo, Z. Qiang, M. Kind, "Towards recursive virtualization and programming for network and cloud resources", July 2015, <https://tools.ietf.org/html/draft-unify-nfvrg-recursive-programming-01>.

[Tah15] M. Tahhan, B. O'Mahony, A. Morton, "Benchmarking Virtual Switches in OPNFV", July 2015, <https://tools.ietf.org/html/draft-vsperf-bmwg-vswitch-opnfv-00>.

[TNOVA15] T-NOVA, "T-NOVA Results", July 2015, <http://www.t-nova.eu/results/>.

[Yard15] OPNFV, "Project: Yardstick - Infrastructure Verification", July 2015, <https://wiki.opnfv.org/yardstick>.

Authors' Addresses

Raphael Vicente Rosa
University of Campinas
Irinyi Jozsef u. 4-20
Budapest 1117
Hungary

Email: raphaelvrosa@dca.fee.unicamp.br
URI: http://www.intrig.dca.fee.unicamp.br/

Christian Esteve Rothenberg
University of Campinas
Av. Albert Einstein, 400
Campinas 13083-852
Brasil

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

Robert Szabo
Ericsson Research, Hungary
Irinyi Jozsef u. 4-20
Budapest 1117
Hungary

Email: robert.szabo@ericsson.com
URI: http://www.ericsson.com/

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 20, 2016

S. Tsuchiya, Ed.
Cisco Systems

October 18, 2015

Benchmarking for CoPP
draft-shishio-bmwg-copp-00

Abstract

COntrol Plane Policing (CoPP) which is defined as RFC6192 to protect router's control plane from undesired or malicious traffic.

This document provides methodology to confirm implementation of CoPP.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

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 20, 2016.

Copyright Notice

Table of Contents

1. Introduction 3
2. Test setup 3
2.1. Topology 3
2.2. Network 4
3. Test procedure 4
3.1. Attack Packet 4
3.1.1. Typical 4
3.1.2. Routing Protocol from trust networkl 4
3.1.3. Routing Protocol from untrust networkl 4
3.1.4. Control Packet from trust network 4
3.1.5. Control Packet from un-trust network 4
3.1.6. Management Packet from trust network 5
3.1.7. Management Packet from un-trust network 5
3.1.8. undefined packets 5
4. Test Result 5
5. Acknowledgements 5
6. IANA Considerations 5
7. Security Considerations 6
8. References 6
8.1. Normative References 6
8.2. Informative References 6
Author’s Address 6

1. Introduction

COntrol Plane Policing (CoPP) which is defined as RFC6192 to protect router's control plane from undesired or malicious traffic. Some modern router implemented this RFC6192 mechanism as the default. Also some router can support RFC6192 by configuration.

Ethernet based service has been widely deployed for both consumer and business . In some case , service provider has to converge customer network directly without CPE. There is ARP/NDP and broadcast/ multicast protocol in the segment , therefore service provider becomes carefully for protection of router's control plane.

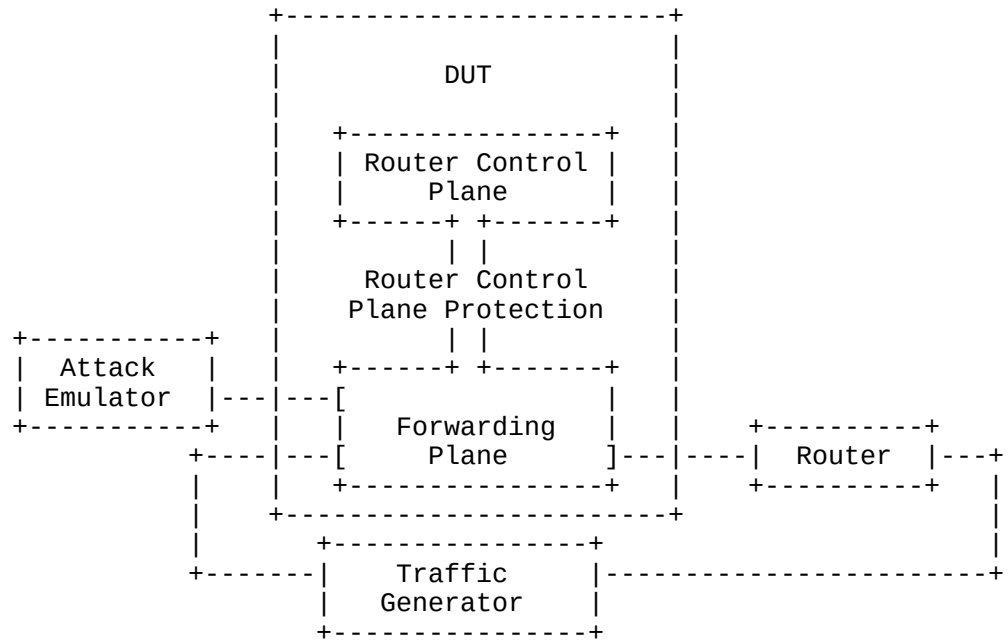
This document provides methodology to confirm implementation of CoPP.

2. Test setup

2.1. Topology

The DUT need to at least 3 interfaces to connect Attack Emulator/ Uplink Router/Traffic Generator.

Basic network topology is here



2.2. Network

Configure applicable network parameter on both DUT and uplink router such as IGP/BGP. Send traffic from traffic generator to transit between DUT and Router. If needed configure DUT to protect Control plane. section-3.1 of RFC6192 would be reference.

3. Test procedure

Attack emulator sends packets to DUT to confirm influence of CoPP. section-3.1 of RFC6192 describes example of CoPP. It completely depends on network environment but it can be categorized in the following section.

3.1. Attack Packet

3.1.1. Typical

fragment/ip option/ ICMP ttl exceed

Expect Action: drop/rate-limit

3.1.2. Routing Protocol from trust networkl

BGP/OSPF

Expect Action: accept but rate-limit would be prefered

3.1.3. Routing Protocol from untrust networkl

BGP/OSPF

Expect Action: drop

3.1.4. Control Packet from trust network

ARP/NDP/ICMP/ICMPv6

Expect Action: accept but rate-limit would be prefered

3.1.5. Control Packet from un-trust network

ARP/NDP/ICMP/ICMPv6

Expect Action: drop

3.1.6. Management Packet from trust network

NTP/SSH/Telnet/Radius/DNS/DHCP

Expect Action: accept/rate-limit

3.1.7. Management Packet from un-trust network

NTP/SSH/Telnet/Radius/DNS/DHCP

Expect Action: drop

3.1.8. undefined packets

IPX/Apple talk

Expect Action: drop

The section will be update more detail(src ip/dst ip and packet type)

Attack duration must be higher than routing protocol hold timer between DUT and router. Transit packets should be non drop.

4. Test Result

Test Result report needs these information.

Attack	Attack rate[pps/bps]	Attack duration time	Loss of packets on traffic generator
-	-	-	-

5. Acknowledgements

TBD

6. IANA Considerations

No IANA Action is requested at this time.

7. Security Considerations

There is no additional consideration from RFC6192.

8. References

8.1. Normative References

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

[RFC6192] Dugal, D., Pignataro, C., and R. Dunn, "Protecting the Router Control Plane", RFC 6192, DOI 10.17487/RFC6192, March 2011, <http://www.rfc-editor.org/info/rfc6192>.

8.2. Informative References

Author's Address

Shishio Tsuchiya (editor)
Cisco Systems
Midtown Tower, 9-7-1,Akasaka
Minato-Ku, Tokyo 107-6227
Japan

Phone: +81 3 6434 6543
Email: shtsuchi@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 16, 2016

M. Tahhan
B. O'Mahony
Intel
A. Morton
AT&T Labs

October 14, 2015

Benchmarking Virtual Switches in OPNFV
draft-vsperf-bmwg-vswitch-opnfv-01

Abstract

This memo describes the progress of the Open Platform for NFV (OPNFV) project on virtual switch performance "VSWITCHPERF". This project intends to build on the current and completed work of the Benchmarking Methodology Working Group in IETF, by referencing existing literature. The Benchmarking Methodology Working Group has traditionally conducted laboratory characterization of dedicated physical implementations of internetworking functions. Therefore, this memo begins to describe the additional considerations when virtual switches are implemented in general-purpose hardware. The expanded tests and benchmarks are also influenced by the OPNFV mission to support virtualization of the "telco" infrastructure.

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 April 16, 2016.

Table of Contents

1. Introduction 3
2. Scope 3
3. Benchmarking Considerations 4
3.1. Comparison with Physical Network Functions 4
3.2. Continued Emphasis on Black-Box Benchmarks 4
3.3. New Configuration Parameters 4
3.4. Flow classification 6
3.5. Benchmarks using Baselines with Resource Isolation . . . 7
4. VSWITCHPERF Specification Summary 8
5. 3x3 Matrix Coverage 16
5.1. Speed of Activation 17
5.2. Accuracy of Activation section 17
5.3. Reliability of Activation 17
5.4. Scale of Activation 17
5.5. Speed of Operation 17
5.6. Accuracy of Operation 17
5.7. Reliability of Operation 17
5.8. Scalability of Operation 18
5.9. Summary 18
6. Security Considerations 18
7. IANA Considerations 19
8. Acknowledgements 19
9. References 19
9.1. Normative References 19
9.2. Informative References 21
Authors' Addresses 21

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. Now, Network Function Virtualization (NFV) has the goal to transform how internetwork functions are implemented, and therefore has garnered much attention.

This memo describes the progress of the Open Platform for NFV (OPNFV) project on virtual switch performance characterization, "VSWITCHPERF". This project intends to build on the current and completed work of the Benchmarking Methodology Working Group in IETF, by referencing existing literature. For example, currently the most often referenced RFC is [RFC2544] (which depends on [RFC1242]) and foundation of the benchmarking work in OPNFV is common and strong.

See https://wiki.opnfv.org/characterize_vswitch_performance_for_telco_nfv_use_cases for more background, and the OPNFV website for general information: https://www.opnfv.org/

The authors note that OPNFV distinguishes itself from other open source compute and networking projects through its emphasis on existing "telco" services as opposed to cloud-computing. There are many ways in which telco requirements have different emphasis on performance dimensions when compared to cloud computing: support for and transfer of isochronous media streams is one example.

Note also that the move to NFV Infrastructure has resulted in many new benchmarking initiatives across the industry, and the authors are currently doing their best to maintain alignment with many other projects, and this Internet Draft is evidence of the efforts.

2. Scope

The primary purpose and scope of the memo is to inform BMWG of work-in-progress that builds on the body of extensive literature and experience. Additionally, once the initial information conveyed here is received, this memo may be expanded to include more detail and commentary from both BMWG and OPNFV communities, under BMWG’s chartered work to characterize the NFV Infrastructure (a virtual switch is an important aspect of that infrastructure).

3. Benchmarking Considerations

This section highlights some specific considerations (from [I-D.ietf-bmwg-virtual-net])related to Benchmarks for virtual switches. The OPNFV project is sharing its present view on these areas, as they develop their specifications in the Level Test Design (LTD) document.

3.1. Comparison with Physical Network Functions

To compare the performance of virtual designs and implementations with their physical counterparts, identical benchmarks are needed. BMWG has developed specifications for many network functions this memo re-uses existing benchmarks through references, and expands them during development of new methods. A key configuration aspect is the number of parallel cores required to achieve comparable performance with a given physical device, or whether some limit of scale was reached before the cores could achieve the comparable level.

It’s unlikely that the virtual switch will be the only application running on the SUT, so CPU utilization, Cache utilization, and Memory footprint should also be recorded for the virtual implementations of internetworking functions.

3.2. Continued Emphasis on Black-Box Benchmarks

External observations remain essential as the basis for Benchmarks. Internal observations with fixed specification and interpretation will be provided in parallel to assist the development of operations procedures when the technology is deployed.

3.3. New Configuration Parameters

A key consideration when conducting any sort of benchmark is trying to ensure the consistency and repeatability of test results. When benchmarking the performance of a vSwitch there are many factors that can affect the consistency of results, one key factor is matching the various hardware and software details of the SUT. This section lists some of the many new parameters which this project believes are critical to report in order to achieve repeatability.

Hardware details including:

- Platform details
- Processor details
- Memory information (type and size)

- Number of enabled cores
- Number of cores used for the test
- Number of physical NICs, as well as their details (manufacturer, versions, type and the PCI slot they are plugged into)
- NIC interrupt configuration
- BIOS version, release date and any configurations that were modified
- CPU microcode level
- Memory DIMM configurations (quad rank performance may not be the same as dual rank) in size, freq and slot locations
- PCI configuration parameters (payload size, early ack option...)
- Power management at all levels (ACPI sleep states, processor package, OS...)

Software details including:

- OS parameters and behavior (text vs graphical no one typing at the console on one system)
- OS version (for host and VNF)
- Kernel version (for host and VNF)
- GRUB boot parameters (for host and VNF)
- Hypervisor details (Type and version)
- Selected vSwitch, version number or commit id used
- vSwitch launch command line if it has been parameterised
- Memory allocation to the vSwitch
- which NUMA node it is using, and how many memory channels
- DPDK or any other SW dependency version number or commit id used
- Memory allocation to a VM - if it’s from Hugpages/elsewhere

- VM storage type: snapshot/independent persistent/independent non-persistent
- Number of VMs
- Number of Virtual NICs (vNICs), versions, type and driver
- Number of virtual CPUs and their core affinity on the host
- Number vNIC interrupt configuration
- Thread affinitization for the applications (including the vSwitch itself) on the host
- Details of Resource isolation, such as CPUs designated for Host/ Kernel (isolcpu) and CPUs designated for specific processes (taskset). - Test duration. - Number of flows.

Test Traffic Information:

- Traffic type - UDP, TCP, IMIX / Other
- Packet Sizes
- Deployment Scenario

3.4. Flow classification

Virtual switches group packets into flows by processing and matching particular packet or frame header information, or by matching packets based on the input ports. Thus a flow can be thought of a sequence of packets that have the same set of header field values or have arrived on the same port. Performance results can vary based on the parameters the vSwitch uses to match for a flow. The recommended flow classification parameters for any vSwitch performance tests are: the input port, the source IP address, the destination IP address and the Ethernet protocol type field. It is essential to increase the flow timeout time on a vSwitch before conducting any performance tests that do not measure the flow setup time. Normally the first packet of a particular stream will install the flow in the virtual switch which adds an additional latency, subsequent packets of the same flow are not subject to this latency if the flow is already installed on the vSwitch.

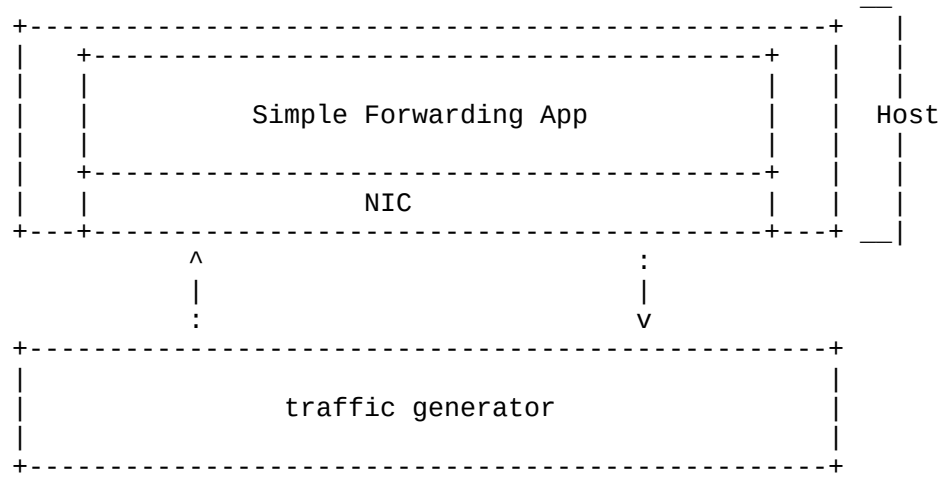
3.5. Benchmarks using Baselines with Resource Isolation

This outline describes measurement of baseline with isolated resources at a high level, which is the intended approach at this time.

1. Baselines:

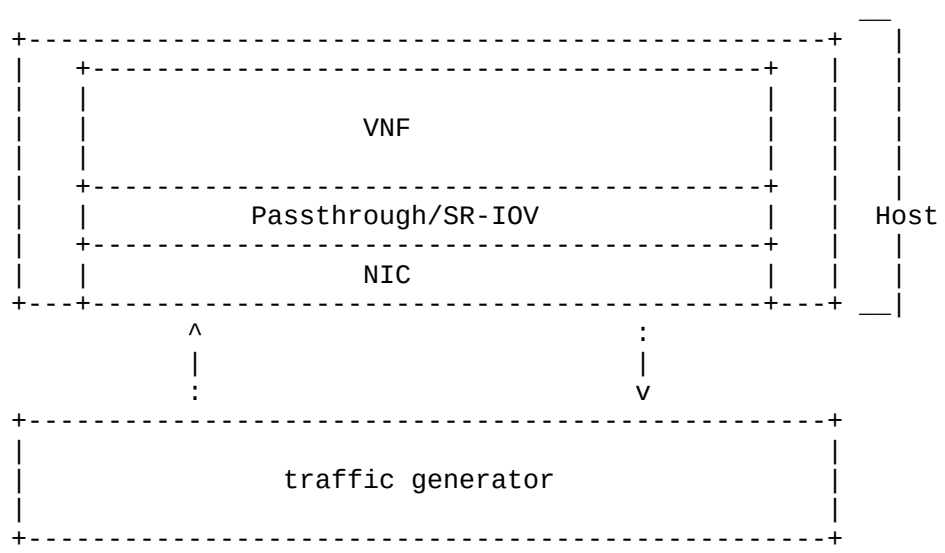
   * Optional: Benchmark platform forwarding capability without a vswitch or VNF for at least 72 hours (serves as a means of platform validation and a means to obtain the base performance for the platform in terms of its maximum forwarding rate and latency).

Benchmark platform forwarding capability



   * Benchmark VNF forwarding capability with direct connectivity (vSwitch bypass, e.g., SR/IOV) for at least 72 hours (serves as a means of VNF validation and a means to obtain the base performance for the VNF in terms of its maximum forwarding rate and latency). The metrics gathered from this test will serve as a key comparison point for vSwitch bypass technologies performance and vSwitch performance.

Benchmark VNF forwarding capability



* Benchmarking with isolated resources alone, with other resources (both HW&SW) disabled Example, vSw and VM are SUT

* Benchmarking with isolated resources alone, leaving some resources unused

* Benchmark with isolated resources and all resources occupied

2. Next Steps

* Limited sharing

* Production scenarios

* Stressful scenarios

4. VSWITCHPERF Specification Summary

The overall specification in preparation is referred to as a Level Test Design (LTD) document, which will contain a suite of performance tests. The base performance tests in the LTD are based on the pre-existing specifications developed by BMWG to test the performance of physical switches. These specifications include:

o [RFC2544] Benchmarking Methodology for Network Interconnect Devices

- [RFC2889] Benchmarking Methodology for LAN Switching
- [RFC6201] Device Reset Characterization
- [RFC5481] Packet Delay Variation Applicability Statement

Some of the above/newer RFCs are being applied in benchmarking for the first time, and represent a development challenge for test equipment developers. Fortunately, many members of the testing system community have engaged on the VSPERF project, including an open source test system.

In addition to this, the LTD also re-uses the terminology defined by:

- [RFC2285] Benchmarking Terminology for LAN Switching Devices
- [RFC5481] Packet Delay Variation Applicability Statement

Specifications to be included in future updates of the LTD include:

- [RFC3918] Methodology for IP Multicast Benchmarking
- [RFC4737] Packet Reordering Metrics

As one might expect, the most fundamental internetworking characteristics of Throughput and Latency remain important when the switch is virtualized, and these benchmarks figure prominently in the specification.

When considering characteristics important to "telco" network functions, we must begin to consider additional performance metrics. In this case, the project specifications have referenced metrics from the IETF IP Performance Metrics (IPPM) literature. This means that the [RFC2544] test of Latency is replaced by measurement of a metric derived from IPPM’s [RFC2679], where a set of statistical summaries will be provided (mean, max, min, etc.). Further metrics planned to be benchmarked include packet delay variation as defined by [RFC5481] , reordering, burst behaviour, DUT availability, DUT capacity and packet loss in long term testing at Throughput level, where some low-level of background loss may be present and characterized.

Tests have been (or will be) designed to collect the metrics below:

- Throughput Tests to measure the maximum forwarding rate (in frames per second or fps) and bit rate (in Mbps) for a constant load (as defined by RFC1242) without traffic loss.

- Packet and Frame Delay Distribution Tests to measure average, min and max packet and frame delay for constant loads.

- Packet Delay Tests to understand latency distribution for different packet sizes and over an extended test run to uncover outliers.

- Scalability Tests to understand how the virtual switch performs as the number of flows, active ports, complexity of the forwarding logic's configuration... it has to deal with increases.

- Stream Performance Tests (TCP, UDP) to measure bulk data transfer performance, i.e. how fast systems can send and receive data through the switch.

- Control Path and Datapath Coupling Tests, to understand how closely coupled the datapath and the control path are as well as the effect of this coupling on the performance of the DUT (example: delay of the initial packet of a flow).

- CPU and Memory Consumption Tests to understand the virtual switch's footprint on the system, usually conducted as auxiliary measurements with benchmarks above. They include: CPU utilization, Cache utilization and Memory footprint.

Future/planned test specs include:

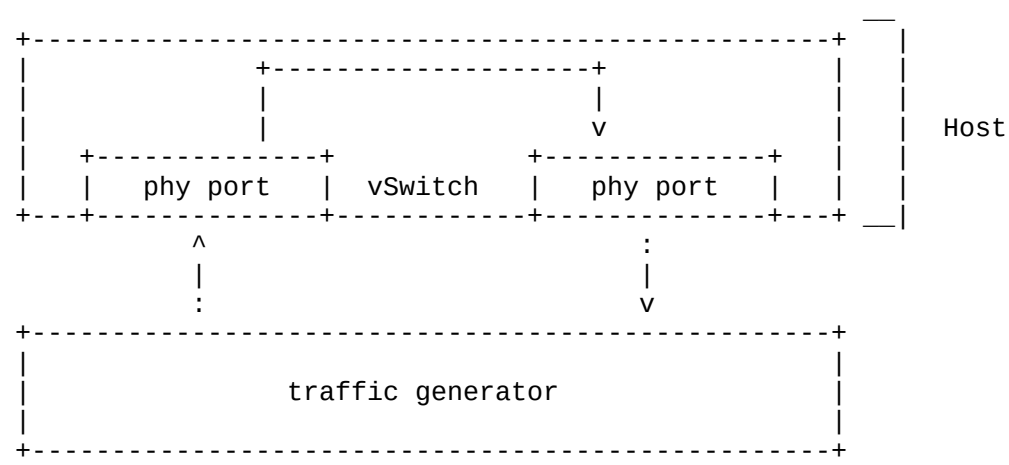
- Request/Response Performance Tests (TCP, UDP) which measure the transaction rate through the switch.

- Noisy Neighbour Tests, to understand the effects of resource sharing on the performance of a virtual switch.

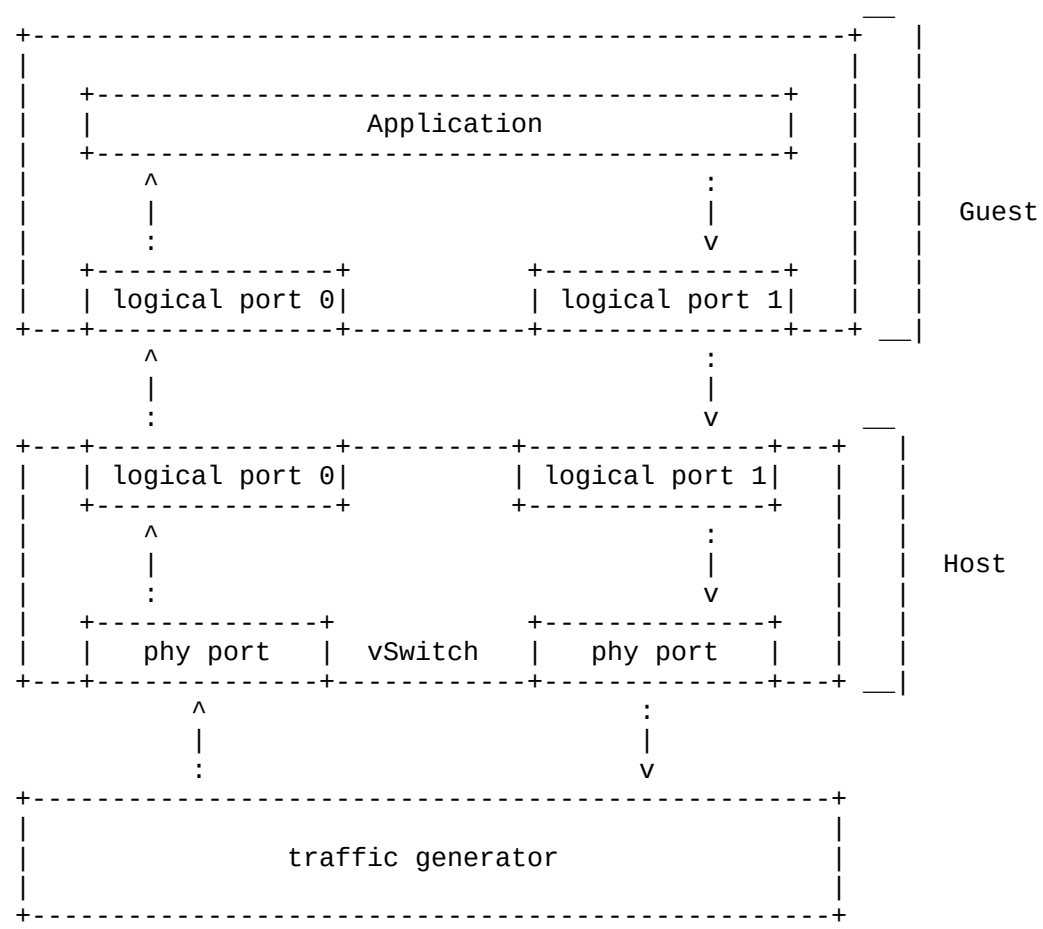
- Tests derived from examination of ETSI NFV Draft GS IFA003 requirements [IFA003] on characterization of acceleration technologies applied to vswitches.

The flexibility of deployment of a virtual switch within a network means that the BMWG IETF existing literature needs to be used to characterize the performance of a switch in various deployment scenarios. The deployment scenarios under consideration include:

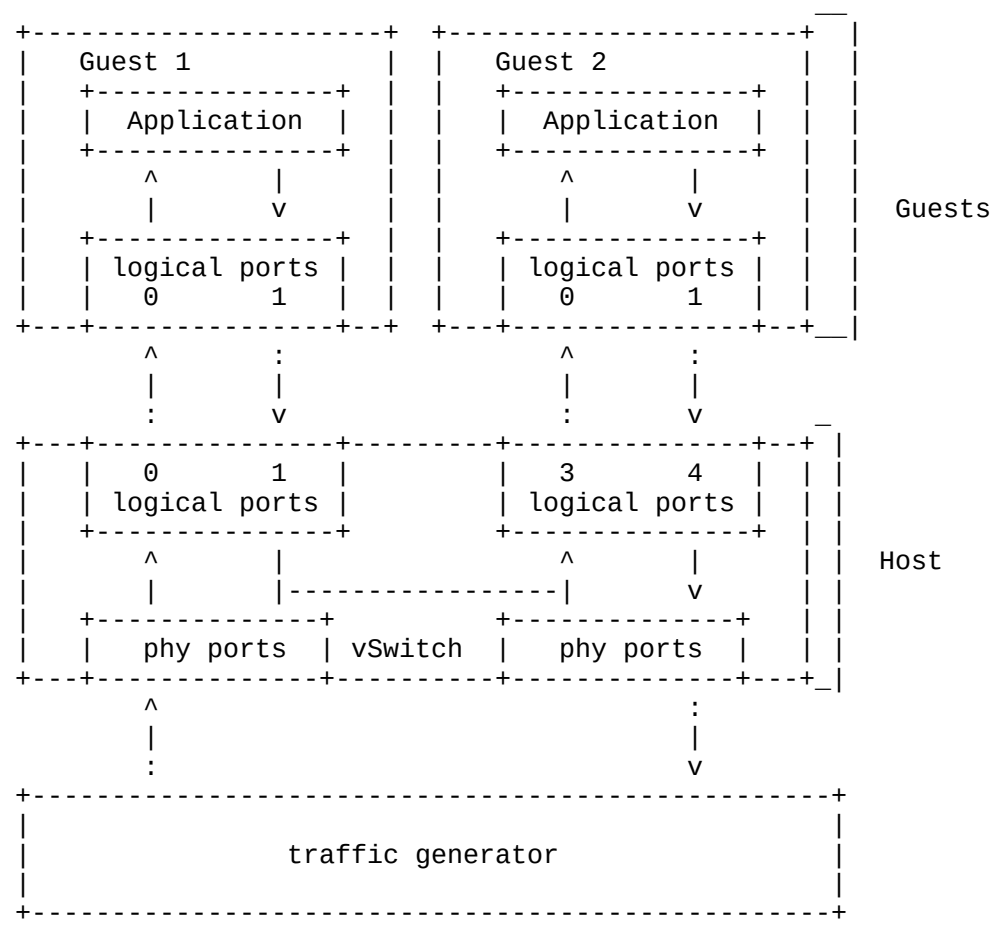
Physical port to virtual switch to physical port



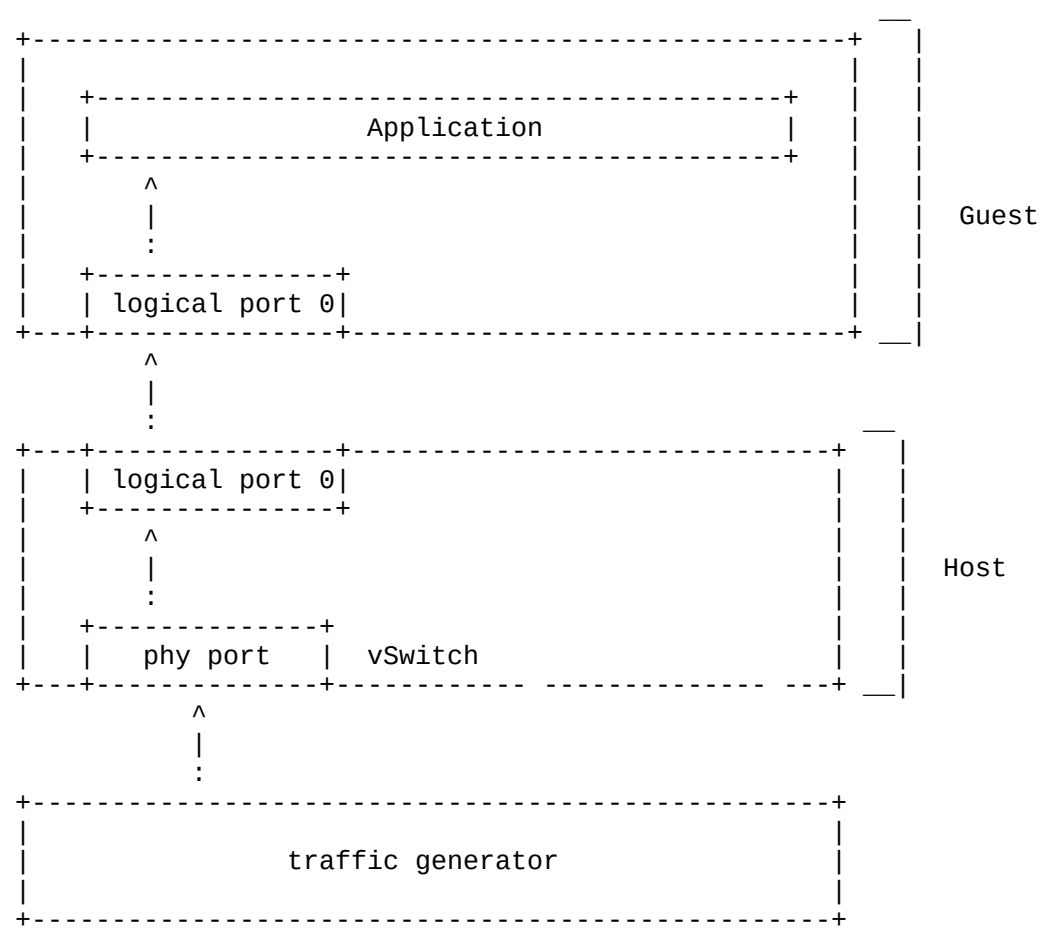
Physical port to virtual switch to VNF to virtual switch to physical port



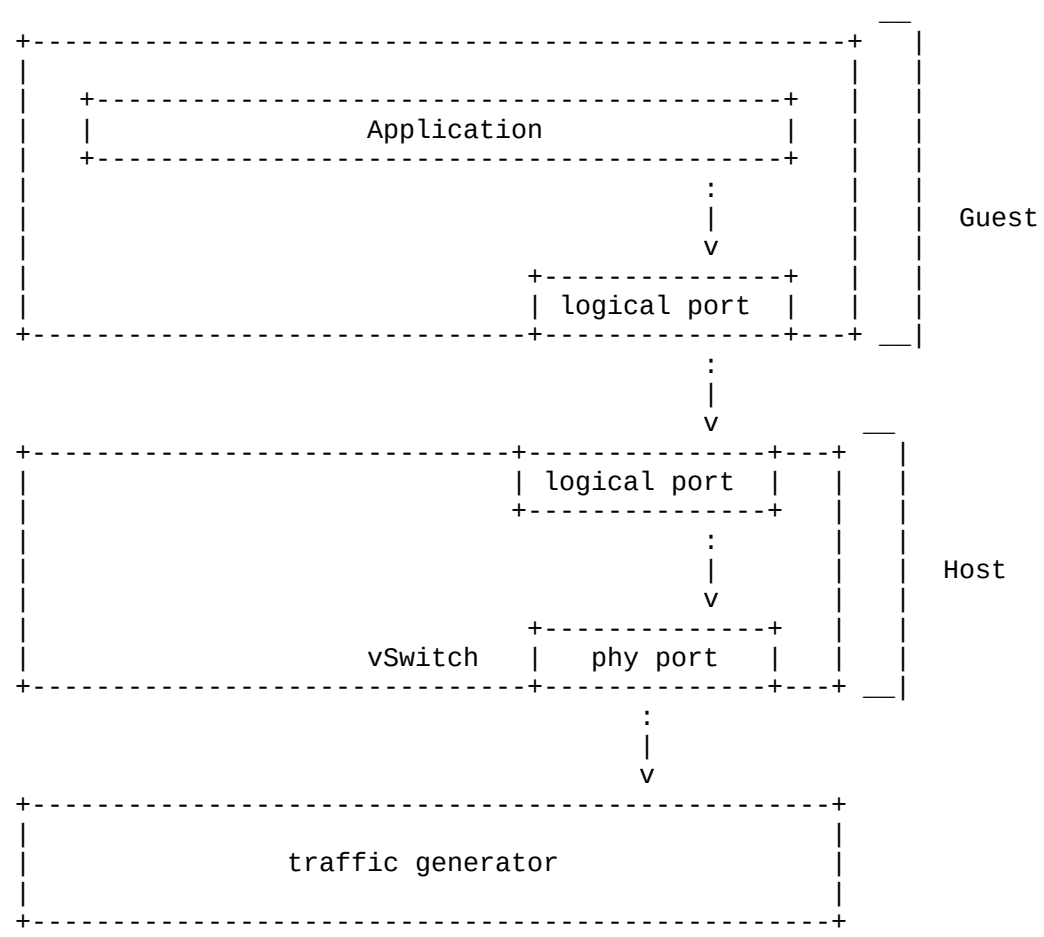
Physical port to virtual switch to VNF to virtual switch to VNF to virtual switch to physical port



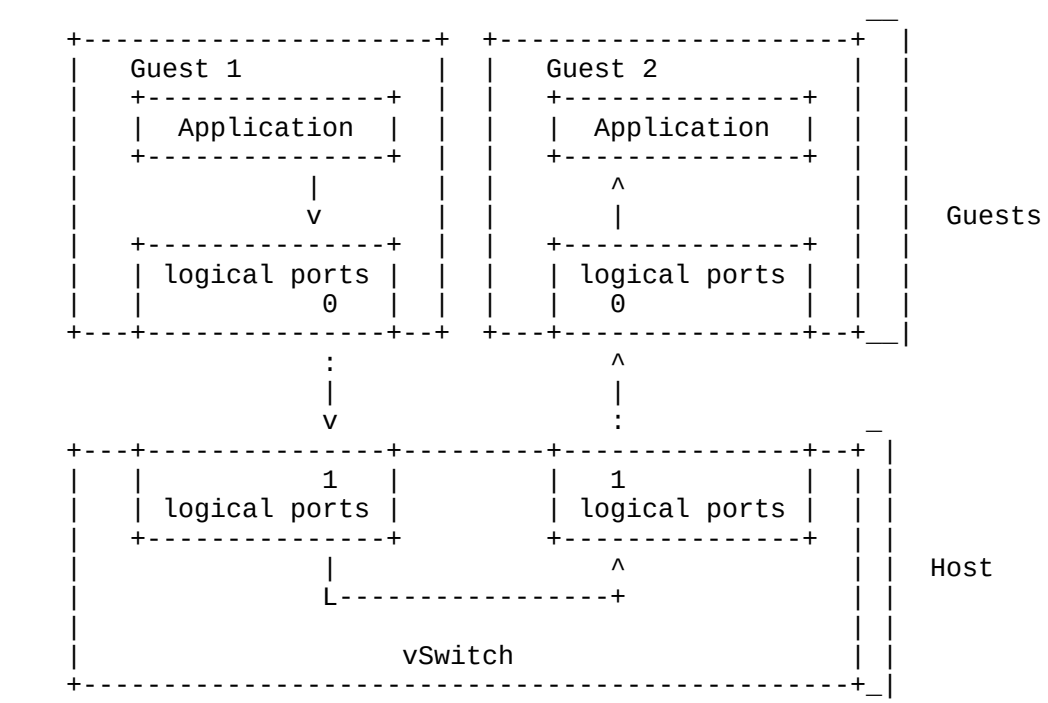
Physical port to virtual switch to VNF



VNF to virtual switch to physical port



VNF to virtual switch to VNF



A set of Deployment Scenario figures is available on the VSPERF Test Methodology Wiki page [TestTopo].

5. 3x3 Matrix Coverage

This section organizes the many existing test specifications into the "3x3" matrix (introduced in [I-D.ietf-bmwg-virtual-net]). Because the LTD specification ID names are quite long, this section is organized into lists for each occupied cell of the matrix (not all are occupied, also the matrix has grown to 3x4 to accommodate scale metrics when displaying the coverage of many metrics/benchmarks).

The tests listed below assess the activation of paths in the data plane, rather than the control plane.

A complete list of tests with short summaries is available on the VSPERF "LTD Test Spec Overview" Wiki page [LTDoverV].

5.1. Speed of Activation

- Activation.RFC2889.AddressLearningRate
- PacketLatency.InitialPacketProcessingLatency

5.2. Accuracy of Activation section

- CPDP.Coupling.Flow.Addition

5.3. Reliability of Activation

- Throughput.RFC2544.SystemRecoveryTime
- Throughput.RFC2544.ResetTime

5.4. Scale of Activation

- Activation.RFC2889.AddressCachingCapacity

5.5. Speed of Operation

- Throughput.RFC2544.PacketLossRate
- CPU.RFC2544.0PacketLoss
- Throughput.RFC2544.PacketLossRateFrameModification
- Throughput.RFC2544.BackToBackFrames
- Throughput.RFC2889.MaxForwardingRate
- Throughput.RFC2889.ForwardPressure
- Throughput.RFC2889.BroadcastFrameForwarding

5.6. Accuracy of Operation

- Throughput.RFC2889.ErrorFramesFiltering
- Throughput.RFC2544.Profile

5.7. Reliability of Operation

- Throughput.RFC2889.Soak
- Throughput.RFC2889.SoakFrameModification

- o PacketDelayVariation.RFC3393.Soak

5.8. Scalability of Operation

- o Scalability.RFC2544.0PacketLoss
- o MemoryBandwidth.RFC2544.0PacketLoss.Scalability

5.9. Summary

	SPEED	ACCURACY	RELIABILITY	SCALE
Activation	X	X	X	X
Operation	X	X	X	X
De-activation				

6. 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.

7. IANA Considerations

No IANA Action is requested at this time.

8. Acknowledgements

The authors acknowledge

9. References

9.1. 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, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2285] Mandeville, R., "Benchmarking Terminology for LAN Switching Devices", RFC 2285, DOI 10.17487/RFC2285, February 1998, <http://www.rfc-editor.org/info/rfc2285>.

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, DOI 10.17487/RFC2330, May 1998, <http://www.rfc-editor.org/info/rfc2330>.

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

[RFC2679] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Delay Metric for IPPM", RFC 2679, DOI 10.17487/RFC2679, September 1999, <http://www.rfc-editor.org/info/rfc2679>.

[RFC2680] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Packet Loss Metric for IPPM", RFC 2680, DOI 10.17487/RFC2680, September 1999, <http://www.rfc-editor.org/info/rfc2680>.

[RFC2681] Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, DOI 10.17487/RFC2681, September 1999, <http://www.rfc-editor.org/info/rfc2681>.

[RFC2889] Mandeville, R. and J. Perser, "Benchmarking Methodology for LAN Switching Devices", RFC 2889, DOI 10.17487/RFC2889, August 2000, <http://www.rfc-editor.org/info/rfc2889>.

[RFC3393] Demichelis, C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, DOI 10.17487/RFC3393, November 2002, <http://www.rfc-editor.org/info/rfc3393>.

[RFC3432] Raisanen, V., Grotefeld, G., and A. Morton, "Network performance measurement with periodic streams", RFC 3432, DOI 10.17487/RFC3432, November 2002, <http://www.rfc-editor.org/info/rfc3432>.

[RFC3918] Stopp, D. and B. Hickman, "Methodology for IP Multicast Benchmarking", RFC 3918, DOI 10.17487/RFC3918, October 2004, <http://www.rfc-editor.org/info/rfc3918>.

[RFC4689] Poretsky, S., Perser, J., Erramilli, S., and S. Khurana, "Terminology for Benchmarking Network-layer Traffic Control Mechanisms", RFC 4689, DOI 10.17487/RFC4689, October 2006, <http://www.rfc-editor.org/info/rfc4689>.

[RFC4737] Morton, A., Ciavattone, L., Ramachandran, G., Shalunov, S., and J. Perser, "Packet Reordering Metrics", RFC 4737, DOI 10.17487/RFC4737, November 2006, <http://www.rfc-editor.org/info/rfc4737>.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <http://www.rfc-editor.org/info/rfc5357>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <http://www.rfc-editor.org/info/rfc5905>.

[RFC6201] Asati, R., Pignataro, C., Calabria, F., and C. Olvera, "Device Reset Characterization", RFC 6201, DOI 10.17487/RFC6201, March 2011, <http://www.rfc-editor.org/info/rfc6201>.

9.2. Informative References

[I-D.ietf-bmwg-virtual-net]
Morton, A., "Considerations for Benchmarking Virtual Network Functions and Their Infrastructure", draft-ietf-bmwg-virtual-net-01 (work in progress), September 2015.

[IFA003] "https://docbox.etsi.org/ISG/NFV/Open/Drafts/IFA003_Acceleration_-_vSwitch_Spec/".

[LTDoverV]
"LTD Test Spec Overview https://wiki.opnfv.org/wiki/vswitchperf_test_spec_review".

[RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <http://www.rfc-editor.org/info/rfc1242>.

[RFC5481] Morton, A. and B. Claise, "Packet Delay Variation Applicability Statement", RFC 5481, DOI 10.17487/RFC5481, March 2009, <http://www.rfc-editor.org/info/rfc5481>.

[RFC6049] Morton, A. and E. Stephan, "Spatial Composition of Metrics", RFC 6049, DOI 10.17487/RFC6049, January 2011, <http://www.rfc-editor.org/info/rfc6049>.

[RFC6248] Morton, A., "RFC 4148 and the IP Performance Metrics (IPPM) Registry of Metrics Are Obsolete", RFC 6248, DOI 10.17487/RFC6248, April 2011, <http://www.rfc-editor.org/info/rfc6248>.

[RFC6390] Clark, A. and B. Claise, "Guidelines for Considering New Performance Metric Development", BCP 170, RFC 6390, DOI 10.17487/RFC6390, October 2011, <http://www.rfc-editor.org/info/rfc6390>.

[TestTopo]
"Test Topologies https://wiki.opnfv.org/vsperf/test_methodology".

Authors' Addresses

Maryam Tahhan
Intel

Email: maryam.tahhan@intel.com

Billy O'Mahony
Intel

Email: billy.o.mahony@intel.com

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/