

Network Working Group  
Internet-Draft  
Updates: 2544 (if approved)  
Intended status: Informational  
Expires: May 20, 2021

A. Morton  
AT&T Labs  
November 16, 2020

Updates for the Back-to-back Frame Benchmark in RFC 2544  
draft-ietf-bmwg-b2b-frame-03

Abstract

Fundamental Benchmarking Methodologies for Network Interconnect Devices of interest to the IETF are defined in RFC 2544. This memo updates the procedures of the test to measure the Back-to-back frames Benchmark of RFC 2544, based on further experience.

This memo updates Section 26.4 of RFC 2544.

Requirements Language

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

Status of This Memo

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

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

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

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

## Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	2
2. Scope and Goals . . . . .	3
3. Motivation . . . . .	4
4. Prerequisites . . . . .	6
5. Back-to-back Frames . . . . .	7
5.1. Preparing the list of Frame sizes . . . . .	7
5.2. Test for a Single Frame Size . . . . .	8
5.3. Test Repetition and Benchmark . . . . .	9
5.4. Benchmark Calculations . . . . .	9
6. Reporting . . . . .	10
7. Security Considerations . . . . .	11
8. IANA Considerations . . . . .	12
9. Acknowledgements . . . . .	12
10. References . . . . .	12
10.1. Normative References . . . . .	12
10.2. Informative References . . . . .	13
Author's Address . . . . .	14

## 1. Introduction

The IETF's fundamental Benchmarking Methodologies are defined in [RFC2544], supported by the terms and definitions in [RFC1242], and [RFC2544] actually obsoletes an earlier specification, [RFC1944]. Over time, the benchmarking community has updated [RFC2544] several times, including the Device Reset Benchmark [RFC6201], and the important Applicability Statement [RFC6815] concerning use outside the Isolated Test Environment (ITE) required for accurate benchmarking. Other specifications implicitly update [RFC2544], such as the IPv6 Benchmarking Methodologies in [RFC5180].

Recent testing experience with the Back-to-back Frame test and Benchmark in Section 26.4 of [RFC2544] indicates that an update is warranted [OPNFV-2017] [VSPERF-b2b]. In particular, analysis of the results indicates that buffer size matters when compensating for interruptions of software packet processing, and this finding increases the importance of the Back-to-back frame characterization described here. This memo describes additional rationale and provides the updated method.

[RFC2544] (which Obsoletes [RFC1944]) provides its own Requirements Language consistent with [RFC2119], since [RFC1944] pre-dates [RFC2119] and all three memos share common authorship. Today, [RFC8174] clarifies the usage of Requirements Language, so the requirements presented in this memo are expressed in [RFC8174] terms, and intended for those performing/reporting laboratory tests to improve clarity and repeatability, and for those designing devices that facilitate these tests.

## 2. Scope and Goals

The scope of this memo is to define an updated method to unambiguously perform tests, measure the benchmark(s), and report the results for Back-to-back Frames (presently described Section 26.4 of [RFC2544]).

The goal is to provide more efficient test procedures where possible, and to expand reporting with additional interpretation of the results. The tests described in this memo address the cases in which the maximum frame rate of a single ingress port cannot be transferred loss-free to an egress port (for some frame sizes of interest).

[RFC2544] Benchmarks rely on test conditions with constant frame sizes, with the goal of understanding what network device capability has been tested. Tests with the smallest size stress the header processing capacity, and tests with the largest size stress the overall bit processing capacity. Tests with sizes in-between may determine the transition between these two capacities. However, conditions simultaneously sending multiple frame sizes, such as those described in [RFC6985], MUST NOT be used in Back-to-back Frame testing.

Section 3 of [RFC8239] describes buffer size testing for physical networking devices in a Data Center. The [RFC8239] methods measure buffer latency directly with traffic on multiple ingress ports that overload an egress port on the Device Under Test (DUT) and are not subject to the revised calculations presented in this memo. Likewise, the methods of [RFC8239] SHOULD be used for test cases where the egress port buffer is the known point of overload.

### 3. Motivation

Section 3.1 of [RFC1242] describes the rationale for the Back-to-back Frames Benchmark. To summarize, there are several reasons that devices on a network produce bursts of frames at the minimum allowed spacing; and it is, therefore, worthwhile to understand the Device Under Test (DUT) limit on the length of such bursts in practice. Also, [RFC1242] states:

"Tests of this parameter are intended to determine the extent of data buffering in the device."

After this test was defined, there have been occasional discussions of the stability and repeatability of the results, both over time and across labs. Fortunately, the Open Platform for Network Function Virtualization (OPNFV) VSPERF project's Continuous Integration (CI) [VSPERF-CI] testing routinely repeats Back-to-back Frame tests to verify that test functionality has been maintained through development of the test control programs. These tests were used as a basis to evaluate stability and repeatability, even across lab set-ups when the test platform was migrated to new DUT hardware at the end of 2016.

When the VSPERF CI results were examined [VSPERF-b2b], several aspects of the results were considered notable:

1. Back-to-back Frame Benchmark was very consistent for some fixed frame sizes, and somewhat variable for other frame sizes.
2. The number of Back-to-back Frames with zero loss reported for large frame sizes was unexpectedly long (translating to 30 seconds of buffer time), and no explanation or measurement limit condition was indicated. It was important that the buffering time calculations were part of the referenced testing and analysis[VSPERF-b2b], because the calculated buffer times of 30 seconds for some frame sizes were clearly wrong or highly suspect. On the other hand, a result expressed only as a large number of Back-to-back Frames does not permit such an easy comparison with reality.
3. Calculation of the extent of buffer time in the DUT helped to explain the results observed with all frame sizes (for example, tests with some frame sizes cannot exceed the frame header processing rate of the DUT and thus no buffering occurs; therefore, the results depended on the test equipment and not the DUT).

4. It was found that a better estimate of the DUT buffer time could be calculated using measurements of both the longest burst in frames without loss and results from the Throughput tests conducted according to Section 26.1 of [RFC2544]. It is apparent that the DUT's frame processing rate empties the buffer during a trial and tends to increase the "implied" buffer size estimate (measured according to Section 26.4 of [RFC2544] because many frames have departed the buffer when the burst of frames ends). A calculation using the Throughput measurement can reveal a "corrected" buffer size estimate.

Further, if the Throughput tests of Section 26.1 of [RFC2544] are conducted as a prerequisite test, the number of frame sizes required for Back-to-back Frame Benchmarking can be reduced to one or more of the small frame sizes, or the results for large frame sizes can be noted as invalid in the results if tested anyway (these are the larger frame sizes for which the back-to-back frame rate cannot exceed the frame header processing rate of the DUT and little or no buffering occurs).

The material below provides the details of the calculation to estimate the actual buffer storage available in the DUT, using results from the Throughput tests for each frame size, and the maximum theoretical frame rate for the DUT links (which constrain the minimum frame spacing).

In reality, there are many buffers and packet header processing steps in a typical DUT. The simplified model used in these calculations for the DUT includes a packet header processing function with limited rate of operation, as shown below:

```

      |----- DUT -----|
Generator -> Ingress -> Buffer -> HeaderProc -> Egress -> Receiver
```

So, in the back2back frame testing:

1. The Ingress burst arrives at Max Theoretical Frame Rate, and initially the frames are buffered.
2. The packet header processing function (HeaderProc) operates at the "Measured Throughput" (Section 26.1 of [RFC2544]), removing frames from the buffer (this is the best approximation we have).
3. Frames that have been processed are clearly not in the buffer, so the Corrected DUT buffer time equation (Section 5.4) estimates and removes the frames that the DUT forwarded on Egress during the burst. We define buffer time as the number of Frames

occupying the buffer divided by the Maximum Theoretical Frame Rate (on ingress) for the Frame size under test.

4. A helpful concept is the buffer filling rate, which is the difference between the Max Theoretical Frame Rate (ingress) and the Measured Throughput (HeaderProc on egress). If the actual buffer size in frames was known, the time to fill the buffer during a measurement can be calculated using the filling rate as a check on measurements. However, the Buffer in the model represents many buffers of different sizes in the DUT data path.

Knowledge of approximate buffer storage size (in time or bytes) may be useful to estimate whether frame losses will occur if DUT forwarding is temporarily suspended in a production deployment, due to an unexpected interruption of frame processing (an interruption of duration greater than the estimated buffer would certainly cause lost frames). In Section 5, the calculations for the correct buffer time use the combination of offered load at Max Theoretical Frame Rate and header processing speed at 100% of Measured Throughput. Other combinations are possible, such as changing the percent of measured Throughput to account for other processes reducing the header processing rate.

The presentation of OPNFV VSPERF evaluation and development of enhanced search algorithms [VSPERF-BSLV] was discussed at IETF-102. The enhancements are intended to compensate for transient interrupts that may cause loss at near-Throughput levels of offered load. Subsequent analysis of the results indicates that buffers within the DUT can compensate for some interrupts, and this finding increases the importance of the Back-to-back frame characterization described here.

#### 4. Prerequisites

The Test Setup MUST be consistent with Figure 1 of [RFC2544], or Figure 2 when the tester's sender and receiver are different devices. Other mandatory testing aspects described in [RFC2544] MUST be included, unless explicitly modified in the next section.

The ingress and egress link speeds and link layer protocols MUST be specified and used to compute the maximum theoretical frame rate when respecting the minimum inter-frame gap.

The test results for the Throughput Benchmark conducted according to Section 26.1 of [RFC2544] for all [RFC2544]-RECOMMENDED frame sizes MUST be available to reduce the tested frame size list, or to note invalid results for individual frame sizes (because the burst length may be essentially infinite for large frame sizes).

Note that:

- o the Throughput and the Back-to-back Frame measurement configuration traffic characteristics (unidirectional or bi-directional, and number of flows generated) MUST match.
- o the Throughput measurement MUST be under zero-loss conditions, according to Section 26.1 of [RFC2544].

The Back-to-back Benchmark described in Section 3.1 of [RFC1242] MUST be measured directly by the tester, where buffer size is inferred from Back-to-back Frame bursts and associated packet loss measurements. Therefore, sources of packet loss that are unrelated to consistent evaluation of buffer size SHOULD be identified and removed or mitigated. Example sources include:

- o On-path active components that are external to the DUT
- o Operating system environment interrupting DUT operation
- o Shared resource contention between the DUT and other off-path component(s) impacting DUT's behaviour, sometimes called the "noisy neighbour" problem with virtualized network functions.

Mitigations applicable to some of the sources above are discussed in Section 5.2, with the other measurement requirements described below in Section 5.

## 5. Back-to-back Frames

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

The Procedure follows.

### 5.1. Preparing the list of Frame sizes

From the list of RECOMMENDED Frame sizes (Section 9 of [RFC2544]), select the subset of Frame sizes whose measured Throughput (during prerequisite testing) was less than the maximum theoretical Frame Rate of the DUT/test-set-up. These are the only Frame sizes where it is possible to produce a burst of frames that cause the DUT buffers to fill and eventually overflow, producing one or more discarded frames.

## 5.2. Test for a Single Frame Size

Each trial in the test requires the tester to send a burst of frames (after idle time) with the minimum inter-frame gap, and to count the corresponding frames forwarded by the DUT.

The duration of the trial **MUST** be at least 2 seconds, to allow DUT buffers to deplete.

If all frames have been received, the tester increases the length of the burst according to the search algorithm and performs another trial.

If the received frame count is less than the number of frames in the burst, then the limit of DUT processing and buffering may have been exceeded, and the burst length is determined by the search algorithm for the next trial (the burst length is typically reduced, but see below).

Classic search algorithms have been adapted for use in benchmarking, where the search requires discovery of a pair of outcomes, one with no loss and another with loss, at load conditions within the acceptable tolerance or accuracy. Conditions encountered when benchmarking the Infrastructure for Network Function Virtualization require algorithm enhancement. Fortunately, the adaptation of Binary Search, and an enhanced Binary Search with Loss Verification have been specified in clause 12.3 of [TST009]. These algorithms can easily be used for Back-to-back Frame benchmarking by replacing the Offered Load level with burst length in frames. [TST009] Annex B describes the theory behind the enhanced Binary Search with Loss Verification algorithm.

There is also promising work-in-progress that may prove useful in Back-to-back Frame benchmarking. [I-D.vpolak-mkonstan-bmwg-mlrsearch] and [I-D.vpolak-bmwg-plrsearch] are two such examples.

Either the [TST009] Binary Search or Binary Search with Loss Verification algorithms **MUST** be used, and input parameters to the algorithm(s) **MUST** be reported.

The tester usually imposes a (configurable) minimum step size for burst length, and the step size **MUST** be reported with the results (as this influences the accuracy and variation of test results).

The original Section 26.4 of [RFC2544] definition is stated below:



The Back-to-back Frame value is the longest burst of frames that the DUT can successfully process and buffer without frame loss, as determined from the series of trials.

### 5.3. Test Repetition and Benchmark

On this topic, Section 26.4 of [RFC2544] requires:

The trial length MUST be at least 2 seconds and SHOULD be repeated at least 50 times with the average of the recorded values being reported.

Therefore, the Benchmark for Back-to-back Frames is the average of burst length values over repeated tests to determine the longest burst of frames that the DUT can successfully process and buffer without frame loss. Each of the repeated tests completes an independent search process.

In this update, the test MUST be repeated N times (the number of repetitions is now a variable that must be reported), for each frame size in the subset list, and each Back-to-back Frame value made available for further processing (below).

### 5.4. Benchmark Calculations

For each Frame size, calculate the following summary statistics for longest Back-to-back Frame values over the N tests:

- o Average (Benchmark)
- o Minimum
- o Maximum
- o Standard Deviation

Further, calculate the Implied DUT Buffer Time and the Corrected DUT Buffer Time in seconds, as follows:

Implied DUT Buffer Time =

Average num of Back-to-back Frames / Max Theoretical Frame Rate

The formula above is simply expressing the Burst of Frames in units of time.

The next step is to apply a correction factor that accounts for the DUT's frame forwarding operation during the test (assuming the simple

model of the DUT composed of a buffer and a forwarding function, described in Section 3).

Corrected DUT Buffer Time =

$$= \text{Implied DUT Buffer Time} - \left( \text{Implied DUT Buffer Time} * \frac{\text{Measured Throughput}}{\text{Max Theoretical Frame Rate}} \right)$$

where:

1. The "Measured Throughput" is the [RFC2544] Throughput Benchmark for the frame size tested, as augmented by methods including the Binary Search with Loss Verification algorithm in [TST009] where applicable, and MUST be expressed in Frames per second in this equation.
2. The "Max Theoretical Frame Rate" is a calculated value for the interface speed and link layer technology used, and MUST be expressed in Frames per second in this equation.

The term on the far right in the formula for Corrected DUT Buffer Time accounts for all the frames in the Burst that were transmitted by the DUT \*while the Burst of frames were sent in\*. So, these frames are not in the Buffer and the Buffer size is more accurately estimated by excluding them.

## 6. Reporting

The back-to-back results SHOULD be reported in the format of a table with a row for each of the tested frame sizes. There SHOULD be columns for the frame size and for the resultant average frame count for each type of data stream tested.

The number of tests Averaged for the Benchmark, N, MUST be reported.

The Minimum, Maximum, and Standard Deviation across all complete tests SHOULD also be reported (they are referred to as "Min,Max,StdDev" in the table below).

The Corrected DUT Buffer Time SHOULD also be reported.

If the tester operates using a limited maximum burst length in frames, then this maximum length SHOULD be reported.

Frame Size, octets	Ave B2B Length, frames	Min,Max,StdDev	Corrected Buff Time, Sec
64	26000	25500,27000,20	0.00004

#### Back-to-Back Frame Results

Static and configuration parameters:

Number of test repetitions, N

Minimum Step Size (during searches), in frames.

If the tester has a specific (actual) frame rate of interest (less than the Throughput rate), it is useful to estimate the buffer time at that actual frame rate:

$$\begin{aligned} \text{Actual Buffer Time} &= \\ &= \text{Corrected DUT Buffer Time} * \frac{\text{Max Theoretical Frame Rate}}{\text{Actual Frame Rate}} \end{aligned}$$

and report this value, properly labeled.

#### 7. Security Considerations

Benchmarking activities as described in this memo are limited to technology characterization using controlled stimuli in a laboratory environment, with dedicated address space and the other constraints of[RFC2544].

The benchmarking network topology will be an independent test setup and MUST NOT be connected to devices that may forward the test traffic into a production network, or misroute traffic to the test management network. See [RFC6815].

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

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

## 8. IANA Considerations

This memo makes no requests of IANA.

## 9. Acknowledgements

Thanks to Trevor Cooper, Sridhar Rao, and Martin Klokik of the VSPERF project for many contributions to the testing [VSPERF-b2b]. Yoshiaki Itou has also investigated the topic, and made useful suggestions. Maciek Konstantyowicz and Vratko Polak also provided many comments and suggestions based on extensive integration testing and resulting search algorithm proposals - the most up-to-date feedback possible. Tim Carlin also provided comments and support for the draft. Warren Kumari's review improved readability in several key passages.

## 10. References

### 10.1. Normative References

- [RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <<https://www.rfc-editor.org/info/rfc1242>>.
- [RFC1944] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 1944, DOI 10.17487/RFC1944, May 1996, <<https://www.rfc-editor.org/info/rfc1944>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <<https://www.rfc-editor.org/info/rfc2544>>.
- [RFC5180] Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, DOI 10.17487/RFC5180, May 2008, <<https://www.rfc-editor.org/info/rfc5180>>.
- [RFC6201] Asati, R., Pignataro, C., Calabria, F., and C. Olvera, "Device Reset Characterization", RFC 6201, DOI 10.17487/RFC6201, March 2011, <<https://www.rfc-editor.org/info/rfc6201>>.

- [RFC6815] Bradner, S., Dubray, K., McQuaid, J., and A. Morton, "Applicability Statement for RFC 2544: Use on Production Networks Considered Harmful", RFC 6815, DOI 10.17487/RFC6815, November 2012, <<https://www.rfc-editor.org/info/rfc6815>>.
- [RFC6985] Morton, A., "IMIX Genome: Specification of Variable Packet Sizes for Additional Testing", RFC 6985, DOI 10.17487/RFC6985, July 2013, <<https://www.rfc-editor.org/info/rfc6985>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

## 10.2. Informative References

- [I-D.vpolak-bmwg-plrsearch]  
Konstantynowicz, M. and V. Polak, "Probabilistic Loss Ratio Search for Packet Throughput (PLRsearch)", draft-vpolak-bmwg-plrsearch-03 (work in progress), March 2020.
- [I-D.vpolak-mkonstan-bmwg-mlrsearch]  
Konstantynowicz, M. and V. Polak, "Multiple Loss Ratio Search for Packet Throughput (MLRsearch)", draft-vpolak-mkonstan-bmwg-mlrsearch-03 (work in progress), March 2020.
- [OPNFV-2017]  
Cooper, T., Morton, A., and S. Rao, "Dataplane Performance, Capacity, and Benchmarking in OPNFV", June 2017, <<https://wiki.opnfv.org/download/attachments/10293193/VSPERF-Dataplane-Perf-Cap-Bench.pptx?api=v2>>.
- [RFC8239] Avramov, L. and J. Rapp, "Data Center Benchmarking Methodology", RFC 8239, DOI 10.17487/RFC8239, August 2017, <<https://www.rfc-editor.org/info/rfc8239>>.
- [TST009] Morton, R. A., "ETSI GS NFV-TST 009 V3.2.1 (2019-06), "Network Functions Virtualisation (NFV) Release 3; Testing; Specification of Networking Benchmarks and Measurement Methods for NFVI"", June 2019, <[https://www.etsi.org/deliver/etsi\\_gs/NFV-TST/001\\_099/009/03.01.01\\_60/gs\\_NFV-TST009v030101p.pdf](https://www.etsi.org/deliver/etsi_gs/NFV-TST/001_099/009/03.01.01_60/gs_NFV-TST009v030101p.pdf)>.

## [VSPERF-b2b]

Morton, A., "Back2Back Testing Time Series (from CI)",  
June 2017, <[https://wiki.opnfv.org/display/vsperf/  
Traffic+Generator+Testing#TrafficGeneratorTesting-  
AppendixB:Back2BackTestingTimeSeries\(fromCI\)](https://wiki.opnfv.org/display/vsperf/Traffic+Generator+Testing#TrafficGeneratorTesting-AppendixB:Back2BackTestingTimeSeries(fromCI))>.

## [VSPERF-BSLV]

Morton, A. and S. Rao, "Evolution of Repeatability in  
Benchmarking: Fraser Plugfest (Summary for IETF BMWG)",  
July 2018,  
<[https://datatracker.ietf.org/meeting/102/materials/  
slides-102-bmwg-evolution-of-repeatability-in-  
benchmarking-fraser-plugfest-summary-for-ietf-bmwg-00](https://datatracker.ietf.org/meeting/102/materials/slides-102-bmwg-evolution-of-repeatability-in-benchmarking-fraser-plugfest-summary-for-ietf-bmwg-00)>.

## [VSPERF-CI]

Tahhan, M., "OPNFV VSPERF CI", June 2019,  
<<https://wiki.opnfv.org/display/vsperf/VSPERF+CI>>.

## Author's Address

Al Morton  
AT&T Labs  
200 Laurel Avenue South  
Middletown,, NJ 07748  
USA

Phone: +1 732 420 1571  
Fax: +1 732 368 1192  
Email: [acmorton@att.com](mailto:acmorton@att.com)

Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: August 6, 2021

S. Jacob, Ed.  
K. Tiruveedhula  
Juniper Networks  
February 2, 2021

Benchmarking Methodology for EVPN and PBB-EVPN  
draft-ietf-bmwg-evpntest-07

## Abstract

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

## Status of This Memo

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

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

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

This Internet-Draft will expire on August 6, 2021.

## Copyright Notice

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

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

## Table of Contents

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

## 1. Introduction

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



EVPN data and control planes, MAC learning, MAC flushing, MAC aging, convergence, high availability, and scale.

### 1.1. Requirements Language

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

### 1.2. Terminologies

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

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

**AA:** All Active mode

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

**DF:** Designated Forwarder

**DUT:** Device under test.

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

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

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

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

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

**IRB:** Integrated routing and bridging interface

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

MHPE2: Multi homed Provider Edge router 2.

MHPE1: Multi homed Provider Edge router 1.

SHPE3: Single homed Provider Edge Router 3.

PE: Provider Edge device.

P: Provider Router.

RR: Route Reflector.

RT: Traffic Generator.

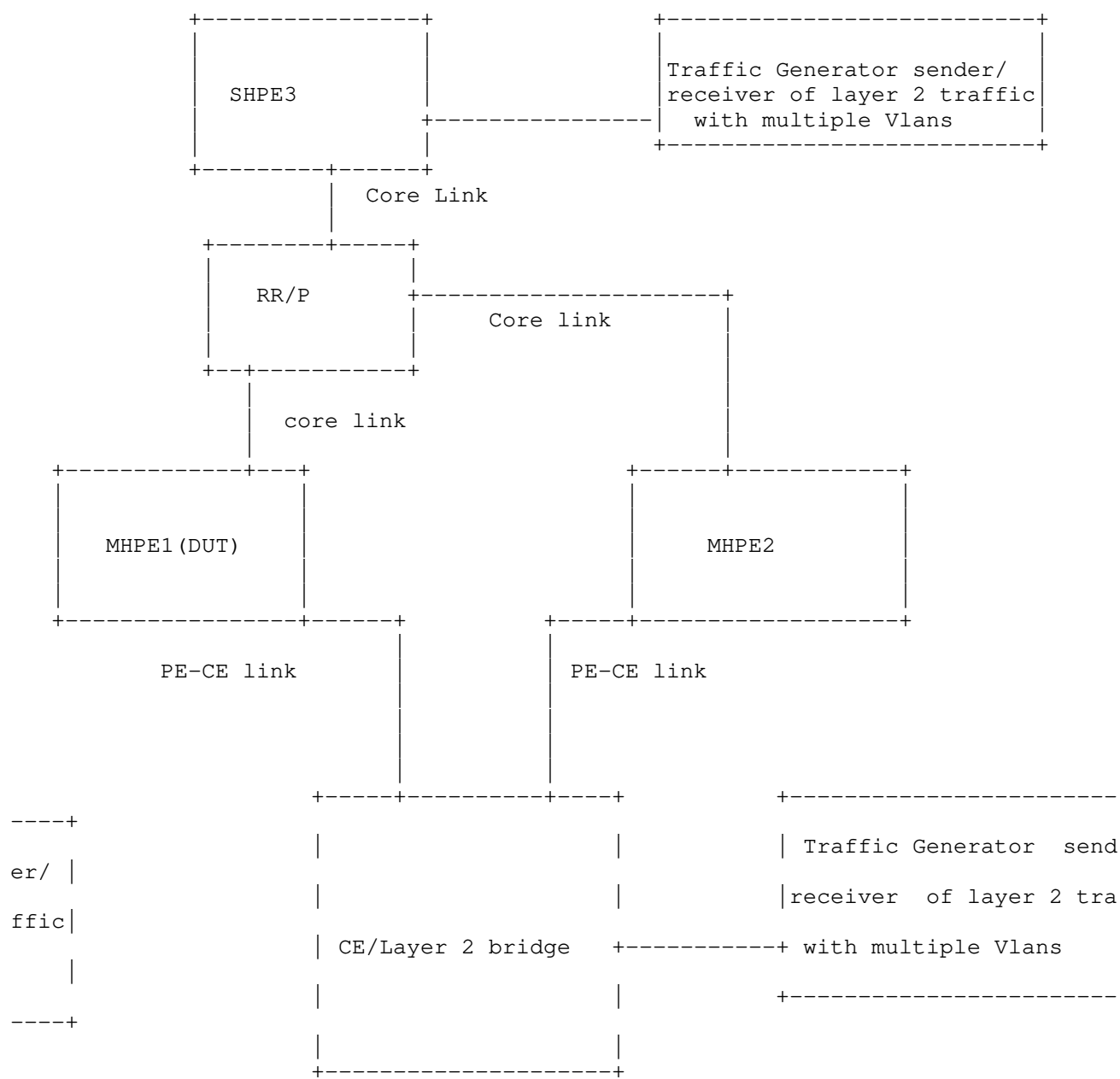
Sub Interface: Each physical Interfaces is subdivided in to a set of Logical units.

SA: Single Active

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

## 2. Test Topology

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



Topology 1

Test Setup

Figure 1

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

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

Figure 2

#### Test Setup Configurations:

SHPE3 is configured with Interior Gateway protocols like OSPF or IS-IS for underlay, LDP for MPLS support, Interior Border Gateway with

EVPN address family for overlay support. This router must be configured with N EVPN/PBB-EVPN instances for testing. Traffic generator is connected to this router for sending and receiving traffic.

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

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

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

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

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

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

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

### 3. Test Cases for EVPN Benchmarking

#### 3.1. Data Plane MAC Learning

Objective:

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

Topology : Topology 1

Procedure:

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

Confirm the DUT is up and running with EVPN.

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

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

The DUT will learn these X MAC in data plane.

Measurement :

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

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

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

### 3.2. Control Plane MAC Learning

Objective:

Measure the time taken to learn the control plane MAC.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

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

Ensure the frames must be destined to one EVPN instance.

The DUT will learn these X MAC in control plane.

Measurement :

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

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

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

Objective:

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

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

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

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

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

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

The DUT and MHPE2 are running SA mode.

Measurement :

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



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

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

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

Objective:

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

Topology : Topology 1

Procedure:

confirm the DUT is up and running with EVPN.

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

Bring down the link between SHPE3 and traffic generator.

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

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

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

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

Measurement :

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

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

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

### 3.5. MAC Aging

Objective:

To measure the MAC aging time.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

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

Ensure these X MAC addresses are learned in DUT.

Then stop the traffic.

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

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

The DUT and MHPE2 are running SA mode.

Measurement :

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

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

### 3.6. Remote MAC Aging

Objective:

Measure the control plane learned MAC aging time.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

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

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

Then stop the traffic.

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

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

The DUT and MHPE2 are running SA mode.

Measurement :

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

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

### 3.7. Control and Data plane MAC Learning

Objective:

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

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

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

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

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

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

DUT and MHPE2 are running in SA mode.

Measurement :

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

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

### 3.8. High Availability.

Objective:

Measure traffic loss during routing engine fail over.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

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

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

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

Ensure the DUT learn 2X MAC.

Then do a routing engine fail-over.

Measurement :

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

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

### 3.9. ARP/ND Scale

Measure the DUT scaling limit of ARP/ND.

Objective:

Measure the ARP/ND scale of the DUT.

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

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

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

The value of X must be increased at a incremental value of 5% of X, till the limit is reached. The limit is where the DUT cant learn any more type 2 MAC+ip/MAC+ipv6. The test must be separately conducted for arp and ND.

Measurement :

Measure the scale limit of type 2 MAC+ip/MAC+ipv6 route which DUT can learn. The test is repeated for N times and the values are collected. The scale limit is calculated by averaging the values obtained by N samples for both MAC+ip and MAC+ipv6. N is an

arbitrary number to get a sufficient sample. The scale value obtained by each sample be  $v_1, v_2, \dots, v_n$ . The measurement is carried out using external server which polls the DUT using automated scripts to find the scale limit of MAC+ipv4/MAC+ipv6.

Scale limit for MAC+ip =  $(v_1 + v_2 + \dots + v_n) / N$

Scale limit for MAC+ipv6 =  $(v_1 + v_2 + \dots + v_n) / N$

### 3.10. Scaling of Services

Objective:

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

Topology : Topology 1

Procedure:

Confirm the DUT is up and running with EVPN.

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

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

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

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

Measurement :

There should not be any loss of route types 1,2,3 and 4 in DUT. DUT must relearn all type 1, 2, 3 and 4 from remote routers. The DUT must be subjected to various values of N to find the optimal scale limit. The scope of the test is find out the maximum evpn instance that a DUT can hold. The measurement is carried out using external server which polls the DUT using automated scripts to find the scale limit of EVPN instances.

### 3.11. Scale Convergence

Objective:

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

Topology : Topology 1

#### Procedure:

Confirm the DUT is up and running with EVPN.

Scale N EVIs in DUT, SHPE3 and MHPE2.

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

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

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

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

Then clear the BGP neighbors in the DUT.

Once the BGP session is in established state in DUT.

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

#### Measurement :

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

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

### 3.12. SOAK Test.

#### Objective:

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

Topology : Topology 1

#### Procedure:

Confirm the DUT is up and running with EVPN.

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

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

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

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

The DUT must run with traffic for 24 hours.

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

Measurement :

Take the hourly reading of CPU, process memory. There should not be any leak, crashes, CPU spikes. The CPU spike is determined as the CPU usage which shoots at 40 to 50 percent of the average usage. The average value vary from device to device. Memory leak is determined by increase usage of the memory for EVPN process. The expectation is under steady state the memory usage for EVPN process should not increase. The measurement is carried out using external server which polls the DUT using automated scripts which captures the CPU usage and process memory.

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

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

Objective:

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

Topology : Topology 1

Procedure:

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



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

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

The DUT will learn these X MAC in data plane.

Measurement :

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

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

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

#### 4.2. Data Plane Remote MAC Learning

Objective:

To Record the time taken to learn the remote MAC.

Topology : Topology 1

Procedure:

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

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

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

The DUT will learn these X MAC in data plane.

Measurement :

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

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

#### 4.3. MAC Flush-Local Link Failure

Objective:

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

Topology : Topology 1

Procedure:

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

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

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

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

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

The DUT and MHPE2 are running SA mode.

Measurement :

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

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

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

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

Objective:

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

Topology : Topology 1

Procedure:

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

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

Bring down the link between SHPE3 and traffic generator.

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

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

Measure time taken by the DUT to relearn these X MAC

Measurement :

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

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

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

#### 4.5. MAC Aging

Objective:

Measure the MAC aging time.

Topology : Topology 1

Procedure:

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

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

Ensure these X MAC addresses are learned in DUT.

Then stop the traffic.

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

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

The DUT and MHPE2 are running SA mode.

Measurement :

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

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

#### 4.6. Remote MAC Aging.

Objective:

Measure the remote MAC aging time.

Topology : Topology 1

Procedure:

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

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

Ensure these X MAC addresses are learned in DUT.

Then stop the traffic.

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

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

The DUT and MHPE2 are running SA mode.

Measurement :

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

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

#### 4.7. Local and Remote MAC Learning

Objective:

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

Topology : Topology 1

Procedure:

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

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

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

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

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

DUT and MHPE2 are running in SA mode.

Measurement :

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

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

#### 4.8. High Availability

Objective:

Measure traffic loss during routing engine failover.

Topology : Topology 1

Procedure:

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

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

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

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

Ensure the DUT learn 2X MAC.

Then do a routing engine fail-over.

Measurement :

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

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

#### 4.9. Scale

Objective:

Measure the scale limit of DUT for PBB-EVPN.

Topology : Topology 1

Procedure:

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

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

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

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

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

Measurement :

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

#### 4.10. Scale Convergence

Objective:

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

Topology : Topology 1

Procedure:

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

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

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

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

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

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

Then clear the BGP neighbors in the DUT.

Once the BGP session is in established state in DUT.

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

Measurement :

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

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



#### 4.11. Soak Test

##### Objective:

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

Topology : Topology 1

##### Procedure:

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

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

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

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

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

The DUT must run with traffic for 24 hours.

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

##### Measurement :

Take the hourly reading of CPU, process memory. There should not be any leak, crashes, CPU spikes. The CPU spike is determined as the CPU usage which shoots at 40 to 50 percent of the average usage. The average value vary from device to device. Memory leak is determined by increase usage of the memory for PBB-EVPN process. The expectation is under steady state the memory usage for PBB-EVPN process should not increase. The measurement is carried out using external server which polls the DUT using automated scripts which captures the CPU usage and process memory.

#### 5. Acknowledgments

We would like to thank Fioccola Giuseppe of Telecom Italia reviewing our draft and commenting it. We would like to thank Sarah Banks for

guiding and mentoring us. We take the opportunity to thank Al for reviewing our draft and gave us valuable comments.

## 6. IANA Considerations

This memo includes no request to IANA.

## 7. Security Considerations

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

## 8. References

### 8.1. Normative References

- [RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <<https://www.rfc-editor.org/info/rfc2544>>.
- [RFC2899] Ginoza, S., "Request for Comments Summary RFC Numbers 2800-2899", RFC 2899, DOI 10.17487/RFC2899, May 2001, <<https://www.rfc-editor.org/info/rfc2899>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

### 8.2. Informative References

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

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

## Appendix A. Appendix

### Authors' Addresses

Sudhin Jacob (editor)  
Juniper Networks  
Bangalore  
India

Phone: +91 8061212543  
Email: [sjacob@juniper.net](mailto:sjacob@juniper.net)

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

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

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

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

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

Abstract

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

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

Status of This Memo

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

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

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

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

## Copyright Notice

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

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

## Table of Contents

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

## 1. Terminology

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

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

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

## 2. MLRsearch Background

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

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

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

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

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

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

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

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

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

### 3. MLRsearch Overview

The main properties of MLRsearch:

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



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

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

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

The main benefits of MLRsearch vs. binary search include:

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

Caveats:

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

#### 4. Sample Implementation

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

##### 4.1. Input Parameters

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

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

#### 4.2. Initial Phase

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

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

3. Third trial measures at MRR2.

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

4.3. Non-Initial Phases

1. Main loop:

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

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

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

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

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

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

## 5. FD.io CSIT Implementation

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

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

### 5.1. Additional details

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

#### 1. Logarithmic transmit rate.

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

#### 2. Optimistic maximum rate.

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

3. Pessimistic minimum rate.

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

4. Conservative interval updates.

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

5. Ensure sufficient interval width.

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

6. Timeout for bad cases.

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

7. Pessimistic external search.

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



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

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

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

#### 5.2. Example MLRsearch Run

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

# Internet-DraftMultiple Loss Ratio Search for Packet ThroughFebruary 2021

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

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

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

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

## 6. IANA Considerations

No requests of IANA.

## 7. Security Considerations

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

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

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

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

## 8. Acknowledgements

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

## 9. References

### 9.1. Normative References

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

### 9.2. Informative References

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

## Authors' Addresses

Maciek Konstantynowicz (editor)  
Cisco Systems

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

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

Vratko Polak (editor)  
Cisco Systems

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

Benchmarking Methodology Working Group  
Internet-Draft  
Intended status: Informational  
Expires: August 26, 2021

B. Balarajah  
C. Rossenhoevel  
EANTC AG  
B. Monkman  
NetSecOPEN  
February 22, 2021

Benchmarking Methodology for Network Security Device Performance  
draft-ietf-bmwg-ngfw-performance-06

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on August 26, 2021.

Copyright Notice

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

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

## Table of Contents

1. Introduction . . . . .	4
2. Requirements . . . . .	4
3. Scope . . . . .	4
4. Test Setup . . . . .	4
4.1. Test Bed Configuration . . . . .	4
4.2. DUT/SUT Configuration . . . . .	6
4.2.1. Security Effectiveness Configuration . . . . .	12
4.3. Test Equipment Configuration . . . . .	12
4.3.1. Client Configuration . . . . .	12
4.3.2. Backend Server Configuration . . . . .	15
4.3.3. Traffic Flow Definition . . . . .	16
4.3.4. Traffic Load Profile . . . . .	17
5. Test Bed Considerations . . . . .	17
6. Reporting . . . . .	18
6.1. Introduction . . . . .	18
6.2. Detailed Test Results . . . . .	20
6.3. Benchmarks and Key Performance Indicators . . . . .	20
7. Benchmarking Tests . . . . .	22
7.1. Throughput Performance with Application Traffic Mix . . . . .	22
7.1.1. Objective . . . . .	22
7.1.2. Test Setup . . . . .	22
7.1.3. Test Parameters . . . . .	22
7.1.4. Test Procedures and Expected Results . . . . .	24
7.2. TCP/HTTP Connections Per Second . . . . .	25
7.2.1. Objective . . . . .	25
7.2.2. Test Setup . . . . .	25
7.2.3. Test Parameters . . . . .	25
7.2.4. Test Procedures and Expected Results . . . . .	27
7.3. HTTP Throughput . . . . .	28
7.3.1. Objective . . . . .	28
7.3.2. Test Setup . . . . .	28
7.3.3. Test Parameters . . . . .	28
7.3.4. Test Procedures and Expected Results . . . . .	30
7.4. HTTP Transaction Latency . . . . .	31
7.4.1. Objective . . . . .	31
7.4.2. Test Setup . . . . .	31

7.4.3.	Test Parameters . . . . .	32
7.4.4.	Test Procedures and Expected Results . . . . .	33
7.5.	Concurrent TCP/HTTP Connection Capacity . . . . .	34
7.5.1.	Objective . . . . .	34
7.5.2.	Test Setup . . . . .	34
7.5.3.	Test Parameters . . . . .	34
7.5.4.	Test Procedures and Expected Results . . . . .	36
7.6.	TCP/HTTPS Connections per Second . . . . .	37
7.6.1.	Objective . . . . .	37
7.6.2.	Test Setup . . . . .	38
7.6.3.	Test Parameters . . . . .	38
7.6.4.	Test Procedures and Expected Results . . . . .	39
7.7.	HTTPS Throughput . . . . .	40
7.7.1.	Objective . . . . .	40
7.7.2.	Test Setup . . . . .	41
7.7.3.	Test Parameters . . . . .	41
7.7.4.	Test Procedures and Expected Results . . . . .	43
7.8.	HTTPS Transaction Latency . . . . .	44
7.8.1.	Objective . . . . .	44
7.8.2.	Test Setup . . . . .	44
7.8.3.	Test Parameters . . . . .	44
7.8.4.	Test Procedures and Expected Results . . . . .	46
7.9.	Concurrent TCP/HTTPS Connection Capacity . . . . .	47
7.9.1.	Objective . . . . .	47
7.9.2.	Test Setup . . . . .	47
7.9.3.	Test Parameters . . . . .	47
7.9.4.	Test Procedures and Expected Results . . . . .	49
8.	IANA Considerations . . . . .	50
9.	Security Considerations . . . . .	50
10.	Contributors . . . . .	50
11.	Acknowledgements . . . . .	50
12.	References . . . . .	51
12.1.	Normative References . . . . .	51
12.2.	Informative References . . . . .	51
Appendix A.	Test Methodology – Security Effectiveness Evaluation	52
A.1.	Test Objective . . . . .	52
A.2.	Test Bed Setup . . . . .	52
A.3.	Test Parameters . . . . .	52
A.3.1.	DUT/SUT Configuration Parameters . . . . .	52
A.3.2.	Test Equipment Configuration Parameters . . . . .	52
A.4.	Test Results Validation Criteria . . . . .	53
A.5.	Measurement . . . . .	53
A.6.	Test Procedures and Expected Results . . . . .	54
A.6.1.	Step 1: Background Traffic . . . . .	54
A.6.2.	Step 2: CVE Emulation . . . . .	54
Appendix B.	DUT/SUT Classification . . . . .	55
Authors' Addresses	. . . . .	55



## 1. Introduction

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

## 2. Requirements

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

## 3. Scope

This document provides testing terminology and testing methodology for modern and next-generation network security devices. It covers the validation of security effectiveness configurations of network security devices, followed by performance benchmark testing. This document focuses on advanced, realistic, and reproducible testing methods. Additionally, it describes test bed environments, test tool requirements, and test result formats.

## 4. Test Setup

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

### 4.1. Test Bed Configuration

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

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

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

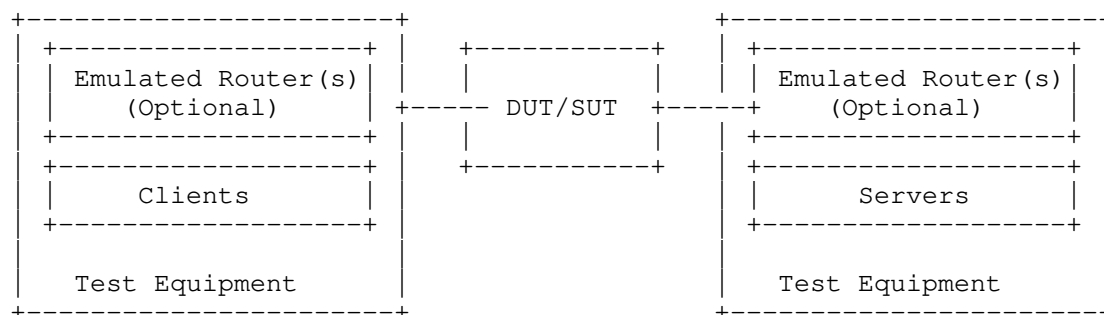


Figure 1: Test Bed Setup - Option 1

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

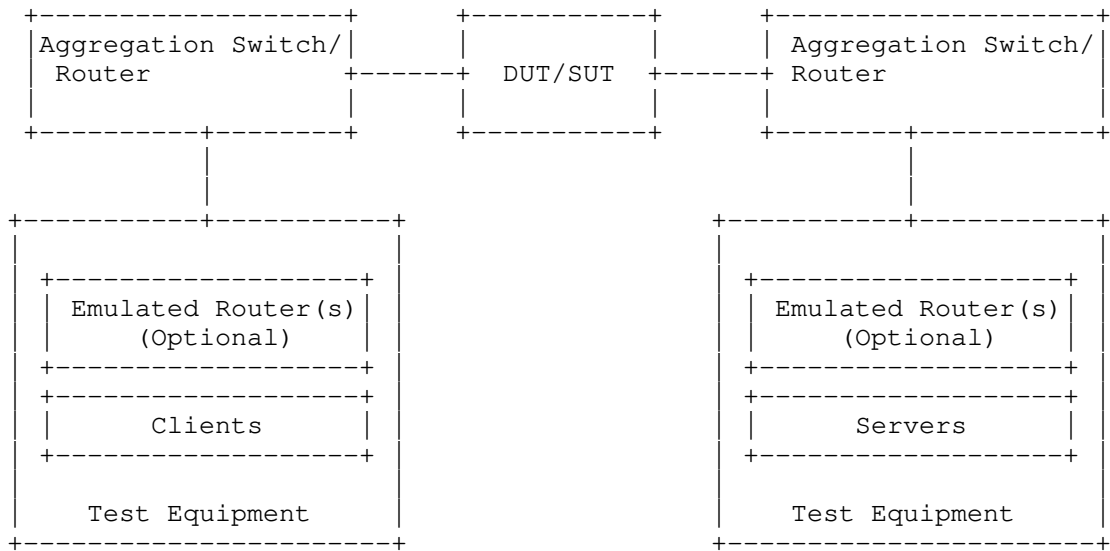


Figure 2: Test Bed Setup - Option 2

4.2. DUT/SUT Configuration

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

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

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

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

Table 1: NGFW Security Features

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

Table 2: NGIDS/NGIPS Security Features

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

DUT/SUT Features	Description
SSL Inspection	DUT/SUT intercepts and decrypts inbound HTTPS traffic between servers and clients. Once the content inspection has been completed, DUT/SUT encrypts the HTTPS traffic with ciphers and keys used by the clients and servers.
IDS/IPS	DUT/SUT detects and blocks exploits targeting known and unknown vulnerabilities across the monitored network.
Anti-Malware	DUT/SUT detects and prevents the transmission of malicious executable code and any associated communications across the monitored network.

	This includes data exfiltration as well as command and control channels.
Anti-Spyware	Anti-Spyware is a subcategory of Anti Malware. Spyware transmits information without the user's knowledge or permission. DUT/SUT detects and block initial infection or transmission of data.
Anti-Botnet	DUT/SUT detects traffic to or from botnets.
Anti-Evasion	DUT/SUT detects and mitigates attacks that have been obfuscated in some manner.
Web Filtering	DUT/SUT detects and blocks malicious website including defined classifications of website across the monitored network.
DLP	DUT/SUT detects and blocks the transmission of Personally Identifiable Information (PII) and specific files across the monitored network
Certificate Validation	DUT/SUT validates certificates used in encrypted communications across the monitored network.
Logging and Reporting	DUT/SUT logs and reports all traffic at the flow level across the monitored.
Application Identification	DUT/SUT detects known applications as defined within the traffic mix selected across the monitored network.

Table 3: Security Feature Description

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

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

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

The Access Control Rules (ACL) defined in Table 4 MUST be configured from top to bottom in the correct order as shown in the table. This is due to ACL types listed in specificity decreasing order, with "block" first, followed by "allow", representing typical ACL based security policy. The ACL entries SHOULD be configured with routable IP subnets by the DUT/SUT. (Note: There will be differences between how security vendors implement ACL decision making.) The configured ACL MUST NOT block the security and measurement traffic used for the benchmarking tests.

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

Table 4: DUT/SUT Access List



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

#### 4.2.1. Security Effectiveness Configuration

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

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

#### 4.3. Test Equipment Configuration

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

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

##### 4.3.1. Client Configuration

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

## 4.3.1.1. TCP Stack Attributes

The TCP stack SHOULD use a congestion control algorithm at client and server endpoints. The default IPv4 and IPv6 MSS segments size SHOULD be set to 1460 bytes and 1440 bytes respectively and a TX and RX initial receive windows of 64 KByte. Client initial congestion window SHOULD NOT exceed 10 times the MSS. Delayed ACKs are permitted and the maximum client delayed ACK SHOULD NOT exceed 10 times the MSS before a forced ACK. Up to 3 retries SHOULD be allowed before a timeout event is declared. All traffic MUST set the TCP PSH flag to high. The source port range SHOULD be in the range of 1024 - 65535. Internal timeout SHOULD be dynamically scalable per RFC 793. The client SHOULD initiate and close TCP connections. TCP connections MUST be closed via FIN.

## 4.3.1.2. Client IP Address Space

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

- o The IP blocks SHOULD consist of multiple unique, discontinuous static address blocks.
- o A default gateway is permitted.
- o The IPv4 Type of Service (ToS) byte or IPv6 traffic class should be set to '00' or '000000' respectively.

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

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

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

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

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

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

(Option 1) 100 % IPv4, no IPv6

(Option 2) 80 % IPv4, 20% IPv6

(Option 3) 50 % IPv4, 50% IPv6

(Option 4) 20 % IPv4, 80% IPv6

(Option 5) no IPv4, 100% IPv6

Note: The IANA has assigned IP address range for the testing purpose as described in Section 8.

#### 4.3.1.3. Emulated Web Browser Attributes

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

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

For encrypted traffic, the following attributes SHALL define the negotiated encryption parameters. The test clients MUST use TLSv1.2 or higher. TLS record size MAY be optimized for the HTTPS response object size up to a record size of 16 KByte. The client endpoint SHOULD send TLS Extension Server Name Indication (SNI) information when opening a security tunnel. Each client connection MUST perform a full handshake with server certificate and MUST NOT use session reuse or resumption.

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

1. ECHDE-ECDSA-AES128-GCM-SHA256 with Prime256v1 (Signature Hash Algorithm: `ecdsa_secp256r1_sha256` and Supported group: `secp256r1`)
2. ECDHE-RSA-AES128-GCM-SHA256 with RSA 2048 (Signature Hash Algorithm: `rsa_pkcs1_sha256` and Supported group: `secp256`)
3. ECDHE-ECDSA-AES256-GCM-SHA384 with Secp521 (Signature Hash Algorithm: `ecdsa_secp384r1_sha384` and Supported group: `secp521r1`)

4.    ECDHE-RSA-AES256-GCM-SHA384 with RSA 4096 (Signature Hash  
      Algorithm: rsa\_pkcs1\_sha384 and Supported group: secp256)

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

#### 4.3.2.    Backend Server Configuration

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

##### 4.3.2.1.    TCP Stack Attributes

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

##### 4.3.2.2.    Server Endpoint IP Addressing

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

- o    The server IP blocks SHOULD consist of unique, discontinuous static address blocks with one IP per Server Fully Qualified Domain Name (FQDN) endpoint per test port.
- o    A default gateway is permitted. The IPv4 ToS byte and IPv6 traffic class bytes should be set to '00' and '000000' respectively.
- o    The server IP addresses SHOULD be distributed between IPv4 and IPv6 with a ratio identical to the clients distribution ratio.

Note: The IANA has assigned IP address range for the testing purpose as described in Section 8.

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

The server pool for HTTP SHOULD listen on TCP port 80 and emulate HTTP version 1.1 with persistence. The Server MUST advertise server type in the Server response header [RFC2616]. For HTTPS server, TLS 1.2 or higher MUST be used with a maximum record size of 16 KByte and

MUST NOT use ticket resumption or Session ID reuse. The server MUST listen on port TCP 443. The server SHALL serve a certificate to the client. The HTTPS server MUST check Host SNI information with the FQDN if the SNI is in use. Cipher suite and key size on the server side MUST be configured similar to the client side configuration described in Section 4.3.1.3.

#### 4.3.3. Traffic Flow Definition

This section describes the traffic pattern between client and server endpoints. At the beginning of the test, the server endpoint initializes and will be ready to accept connection states including initialization of the TCP stack as well as bound HTTP and HTTPS servers. When a client endpoint is needed, it will initialize and be given attributes such as a MAC and IP address. The behavior of the client is to sweep through the given server IP space, sequentially generating a recognizable service by the DUT. Thus, a balanced, mesh between client endpoints and server endpoints will be generated in a client port server port combination. Each client endpoint performs the same actions as other endpoints, with the difference being the source IP of the client endpoint and the target server IP pool. The client MUST use the server's IP address or Fully Qualified Domain Names (FQDN) in Host Headers [RFC2616]. For TLS the client MAY use Server Name Indication (SNI).

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

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

#### 4.3.4. Traffic Load Profile

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

1. During the Init phase, test bed devices including the client and server endpoints should negotiate layer 2-3 connectivity such as MAC learning and ARP. Only after successful MAC learning or ARP/ND resolution SHALL the test iteration move to the next phase. No measurements are made in this phase. The minimum RECOMMEND time for Init phase is 5 seconds. During this phase, the emulated clients SHOULD NOT initiate any sessions with the DUT/SUT, in contrast, the emulated servers should be ready to accept requests from DUT/SUT or from emulated clients.
2. In the ramp up phase, the test equipment SHOULD start to generate the test traffic. It SHOULD use a set approximate number of unique client IP addresses actively to generate traffic. The traffic SHOULD ramp from zero to desired target objective. The target objective will be defined for each benchmarking test. The duration for the ramp up phase MUST be configured long enough, so that the test equipment does not overwhelm the DUT/SUT's stated performance metrics namely; connections per second, throughput, concurrent TCP connections, and application transactions per second. No measurements are made in this phase.
3. Sustain phase starts when all required clients (connections) are active and operating at their desired load condition. In the sustain phase, the test equipment SHOULD continue generating traffic to constant target value for a constant number of active clients. The minimum RECOMMENDED time duration for sustain phase is 300 seconds. This is the phase where measurements occur.
4. In the ramp down/close phase, no new connections are established, and no measurements are made. The time duration for ramp up and ramp down phase SHOULD be the same.
5. The last phase is administrative and will occur when the test equipment merges and collates the report data.

#### 5. Test Bed Considerations

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

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

Test bed reference pre-tests help to ensure that the maximum desired traffic generator aspects such as throughput, transaction per second, connection per second, concurrent connection, and latency.

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

## 6. Reporting

This section describes how the final report should be formatted and presented. The final test report MAY have two major sections; Introduction and detailed test results sections.

### 6.1. Introduction

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

1. The time and date of the execution of the test MUST be prominent.
2. Summary of test bed software and Hardware details
  - A. DUT/SUT Hardware/Virtual Configuration
    - + This section SHOULD clearly identify the make and model of the DUT/SUT
    - + The port interfaces, including speed and link information MUST be documented.

- + If the DUT/SUT is a Virtual Network Function (VNF), host (server) hardware and software details, interface acceleration type such as DPDK and SR-IOV used CPU cores, used RAM, and the resource sharing (e.g. Pinning details and NUMA Node) configuration MUST be documented. The virtual components such as Hypervisor, virtual switch version MUST be also documented.
- + Any additional hardware relevant to the DUT/SUT such as controllers MUST be documented

B. DUT/SUT Software

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

C. DUT/SUT Enabled Features

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

D. Test equipment hardware and software

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

E. Key test parameters

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



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

### 3. Results Summary / Executive Summary

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

### 6.2. Detailed Test Results

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

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

### 6.3. Benchmarks and Key Performance Indicators

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

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

- o TCP Connections Per Second  
The average number of successfully established TCP connections per second between hosts across the DUT/SUT, or between hosts and the DUT/SUT. The TCP connection must be initiated via a TCP 3 way handshake (SYN, SYN/ACK, ACK). Then the TCP session data is sent. The TCP session MUST be closed via either a TCP 3 way close (FIN, FIN/ACK, ACK), or a TCP 4 way close (FIN, ACK, FIN, ACK), and not by a RST.
- o Application Transactions Per Second  
The average number of successfully completed transactions per second. For a particular transaction to be considered successful, all data must have been transferred in its entirety. In case of HTTP(S) transaction, it must have a valid status code, and the appropriate FIN, FIN/ACK sequence must have been completed.
- o TLS Handshake Rate  
The average number of successfully established TLS connections per second between hosts across the DUT/SUT, or between hosts and the DUT/SUT.
- o Throughput  
The number of bits per second of allowed traffic a DUT/SUT can be observed to transmit to the correct destination interface(s) in response to a specified offered load (defined in [RFC2647]). The throughput benchmarking tests defined in Section 7 SHOULD measure the average throughput value. This document recommends presenting the throughput value in Gbit/s rounded to two places of precision with a more specific Kbit/s in parenthesis.
- o Time to First Byte (TTFB)  
TTFB is the elapsed time between the start of sending the TCP SYN packet from the client and the client receiving the first packet of application data from the server or DUT/SUT. The benchmarking tests HTTP Transaction Latency (Section 7.4) and HTTPS Transaction Latency (Section 7.8) measure the minimum, average and maximum TTFB. The value SHOULD be expressed in millisecond.
- o URL Response time / Time to Last Byte (TTLB)  
URL Response time / TTLB is the elapsed time between the start of sending the TCP SYN packet from the client and the client receiving the last packet of application data from the server or DUT/SUT. The benchmarking tests HTTP Transaction Latency (Section 7.4) and HTTP Transaction Latency (Section 7.8) measure the minimum, average and maximum TTLB. The value SHOULD be expressed in millisecond.

## 7. Benchmarking Tests

### 7.1. Throughput Performance with Application Traffic Mix

#### 7.1.1. Objective

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

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

Name of applications and layer 7 protocols

Percentage of emulated traffic for each application and layer 7 protocols

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

Used object sizes for each application and layer 7 protocols

#### 7.1.2. Test Setup

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

#### 7.1.3. Test Parameters

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

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

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

#### 7.1.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

Initial throughput: 10% of the "Target throughput"

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

#### 7.1.3.3. Traffic Profile

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

#### 7.1.3.4. Test Results Validation Criteria

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

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

Note: Criteria a. and b. above are synonymous with the zero-packet loss criteria for [RFC2544] Throughput, and recognize the additional complexity of application layer performance.

#### 7.1.3.5. Measurement

Following KPI metrics MUST be reported for this benchmarking test:

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

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

Optional KPIs: TCP Connections Per Second and TLS Handshake Rate

#### 7.1.4. Test Procedures and Expected Results

The test procedures are designed to measure the throughput performance of the DUT/SUT at the sustaining period of traffic load profile. The test procedure consists of three major steps. This test procedure MAY be repeated multiple times with different IP types; IPv4 only, IPv6 only and IPv4 and IPv6 mixed traffic distribution.

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

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

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

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

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

Configure test equipment to generate traffic at the "Target throughput" rate defined in the parameter table. The test equipment SHOULD follow the traffic load profile definition as described in Section 4.3.4. The test equipment SHOULD start to measure and record all specified KPIs and the frequency of measurements SHOULD be less than 2 seconds. Continue the test until all traffic profile phases are completed.

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

#### 7.1.4.3. Step 3: Test Iteration

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

### 7.2. TCP/HTTP Connections Per Second

#### 7.2.1. Objective

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

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

#### 7.2.2. Test Setup

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

#### 7.2.3. Test Parameters

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

##### 7.2.3.1. DUT/SUT Configuration Parameters

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

##### 7.2.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

Initial connections per second: 10% of "Target connections per second" (an optional parameter for documentation)

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

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

#### 7.2.3.3. Test Results Validation Criteria

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

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

#### 7.2.3.4. Measurement

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

#### 7.2.4. Test Procedures and Expected Results

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

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

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

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

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

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

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

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

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

The test equipment SHOULD start to measure and record all specified KPIs and the frequency of measurements SHOULD be less than 2 seconds. Continue the test until all traffic profile phases are completed.



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

#### 7.2.4.3. Step 3: Test Iteration

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

### 7.3. HTTP Throughput

#### 7.3.1. Objective

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

#### 7.3.2. Test Setup

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

#### 7.3.3. Test Parameters

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

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

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

##### 7.3.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

Initial Throughput: 10% of "Target Throughput" (an optional parameter for documentation)

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

RECOMMENDED HTTP response object size: 1, 16, 64, 256 KByte, and mixed objects defined in the table

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

Table 4: Mixed Objects

#### 7.3.3.3. Test Results Validation Criteria

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

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

Note: Criteria a. above is synonymous with the zero-packet loss criteria for [RFC2544] Throughput, and recognize the additional complexity of application layer performance.

#### 7.3.3.4. Measurement

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

#### 7.3.4. Test Procedures and Expected Results

The test procedure is designed to measure HTTP throughput of the DUT/SUT. The test procedure consists of three major steps. This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution and HTTP response object sizes.

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

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

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

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

The measured KPIs during the sustain phase MUST meet the test results validation criteria "a" defined in Section 7.3.3.3.

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

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

Configure test equipment to establish the target objective ("Target throughput") defined in the parameters table. The test equipment SHOULD start to measure and record all specified KPIs and the frequency of measurements SHOULD be less than 2 seconds. Continue the test until all traffic profile phases are completed.

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

#### 7.3.4.3. Step 3: Test Iteration

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

### 7.4. HTTP Transaction Latency

#### 7.4.1. Objective

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

Test iterations MUST be performed with different HTTP response object sizes in two different scenarios. One with a single transaction and the other with multiple transactions within a single TCP connection. For consistency both the single and multiple transaction test MUST be configured with HTTP 1.1.

Scenario 1: The client MUST negotiate HTTP 1.1 and close the connection with FIN immediately after completion of a single transaction (GET and RESPONSE).

Scenario 2: The client MUST negotiate HTTP 1.1 and close the connection FIN immediately after completion of 10 transactions (GET and RESPONSE) within a single TCP connection.

#### 7.4.2. Test Setup

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

#### 7.4.3. Test Parameters

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

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

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

##### 7.4.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

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

Initial objective for scenario 1: 10% of Target objective for scenario 1" (an optional parameter for documentation)

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

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

HTTP 1.1 with GET command requesting a single object. The RECOMMENDED object sizes are 1, 16, and 64 KByte. For each test iteration, client MUST request a single HTTP response object size.

#### 7.4.3.3. Test Results Validation Criteria

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

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

#### 7.4.3.4. Measurement

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

#### 7.4.4. Test Procedures and Expected Results

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

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

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

Configure traffic load profile of the test equipment to establish "Initial objective" as defined in the parameters Section 7.4.3.2.

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

The DUT/SUT SHOULD reach the "Initial objective" before the sustain phase. The measured KPIs during the sustain phase MUST meet the test results validation criteria a, b, c, d, e and f defined in Section 7.4.3.3.

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

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

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

The test equipment SHOULD start to measure and record all specified KPIs and the frequency of measurement SHOULD be less than 2 seconds. Continue the test until all traffic profile phases are completed.

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

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

### 7.5. Concurrent TCP/HTTP Connection Capacity

#### 7.5.1. Objective

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

#### 7.5.2. Test Setup

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

#### 7.5.3. Test Parameters

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

#### 7.5.3.1. DUT/SUT Configuration Parameters

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

#### 7.5.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

Initial concurrent connection: 10% of "Target concurrent connection" (an optional parameter for documentation)

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

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

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

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

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

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



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

#### 7.5.3.3. Test Results Validation Criteria

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

- a. Number of failed Application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transaction) of total attempted transactions.
- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections.
- c. During the sustain phase, traffic SHOULD be forwarded constantly.

#### 7.5.3.4. Measurement

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

#### 7.5.4. Test Procedures and Expected Results

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

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

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

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

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

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

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

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

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

The test equipment SHOULD start to measure and record KPIs defined in Section 7.5.3.4. The frequency of measurement SHOULD be less than 2 seconds. Continue the test until all traffic profile phases are completed.

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

#### 7.5.4.3. Step 3: Test Iteration

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

### 7.6. TCP/HTTPS Connections per Second

#### 7.6.1. Objective

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

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

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

#### 7.6.2. Test Setup

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

#### 7.6.3. Test Parameters

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

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

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

##### 7.6.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

Initial connections per second: 10% of "Target connections per second" (an optional parameter for documentation)

RECOMMENDED ciphers and keys defined in Section 4.3.1.3

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

#### 7.6.3.3. Test Results Validation Criteria

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

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

#### 7.6.3.4. Measurement

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

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

#### 7.6.4. Test Procedures and Expected Results

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

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

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

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

The DUT/SUT SHOULD reach the "Initial connections per second" before the sustain phase. The measured KPIs during the sustain phase MUST

meet the test results validation criteria a, b, c, and d defined in Section 7.6.3.3.

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

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

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

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

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement SHOULD be less than 2 seconds. Continue the test until all traffic profile phases are completed.

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

#### 7.6.4.3. Step 3: Test Iteration

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

### 7.7. HTTPS Throughput

#### 7.7.1. Objective

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

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

#### 7.7.2. Test Setup

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

#### 7.7.3. Test Parameters

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

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

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

##### 7.7.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

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

Initial Throughput: 10% of "Target Throughput" (an optional parameter for documentation)

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

RECOMMENDED ciphers and keys defined in Section 4.3.1.3

RECOMMENDED HTTPS response object size: 1, 16, 64, 256 KByte, and mixed objects defined in the table below.

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

Table 5: Mixed Objects

#### 7.7.3.3. Test Results Validation Criteria

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

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

Note: Criteria a. above is synonymous with the zero-packet loss criteria for [RFC2544] Throughput, and recognize the additional complexity of application layer performance.

#### 7.7.3.4. Measurement

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

#### 7.7.4. Test Procedures and Expected Results

The test procedure consists of three major steps. This test procedure MAY be repeated multiple times with different IPv4 and IPv6 traffic distribution and HTTPS response object sizes.

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

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

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

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

The measured KPIs during the sustain phase MUST meet the test results validation criteria "a" defined in Section 7.7.3.3.

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

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

Configure test equipment to establish the target objective ("Target throughput") defined in the parameters table. The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement SHOULD be less than 2 seconds. Continue the test until all traffic profile phases are completed.

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

##### 7.7.4.3. Step 3: Test Iteration

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



## 7.8. HTTPS Transaction Latency

### 7.8.1. Objective

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

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

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

### 7.8.2. Test Setup

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

### 7.8.3. Test Parameters

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

#### 7.8.3.1. DUT/SUT Configuration Parameters

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

#### 7.8.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

RECOMMENDED cipher suites and key sizes defined in Section 4.3.1.3

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

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

Initial objective for scenario 1: 10% of Target objective for scenario 1" (an optional parameter for documentation)

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

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

HTTPS 1.1 with GET command requesting a single object. The RECOMMENDED object sizes are 1, 16, and 64 KByte. For each test iteration, client MUST request a single HTTPS response object size.

#### 7.8.3.3. Test Results Validation Criteria

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

- a. Number of failed Application transactions (receiving any HTTP response code other than 200 OK) MUST be less than 0.001% (1 out of 100,000 transactions) of attempt transactions.
- b. Number of Terminated TCP connections due to unexpected TCP RST sent by DUT/SUT MUST be less than 0.001% (1 out of 100,000 connections) of total initiated TCP connections
- c. During the sustain phase, traffic should be forwarded at a constant rate.
- d. Concurrent TCP connections MUST be constant during steady state and any deviation of concurrent TCP connections SHOULD be less than 10%. This confirms the DUT opens and closes TCP connections almost at the same rate
- e. After ramp up the DUT MUST achieve the "Target objective" defined in the parameter Section 7.8.3.2 and remain in that state for the entire test duration (sustain phase).

#### 7.8.3.4. Measurement

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

#### 7.8.4. Test Procedures and Expected Results

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

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

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

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

The DUT/SUT SHOULD reach the "Initial objective" before the sustain phase. The measured KPIs during the sustain phase MUST meet the test results validation criteria a, b, c, d, e and f defined in Section 7.8.3.3.

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

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

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

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement SHOULD be less than 2 seconds. Continue the test until all traffic profile phases are completed.

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

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

## 7.9. Concurrent TCP/HTTPS Connection Capacity

### 7.9.1. Objective

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

### 7.9.2. Test Setup

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

### 7.9.3. Test Parameters

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

#### 7.9.3.1. DUT/SUT Configuration Parameters

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

#### 7.9.3.2. Test Equipment Configuration Parameters

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

Client IP address range defined in Section 4.3.1.2

Server IP address range defined in Section 4.3.2.2

Traffic distribution ratio between IPv4 and IPv6 defined in Section 4.3.1.2

RECOMMENDED cipher suites and key sizes defined in Section 4.3.1.3

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

Initial concurrent connections: 10% of "Target concurrent connections" (an optional parameter for documentation)

Connections per second during ramp up phase: 50% of maximum connections per second measured in benchmarking test TCP/HTTPS Connections per second (Section 7.6)

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

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

The client MUST perform HTTPS transaction with persistence and each client can open multiple concurrent TCP connections per server endpoint IP.

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

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

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

#### 7.9.3.3. Test Results Validation Criteria

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

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

#### 7.9.3.4. Measurement

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

#### 7.9.4. Test Procedures and Expected Results

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

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

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

Configure test equipment to establish "initial concurrent TCP connections" defined in Section 7.9.3.2. Except ramp up time, the traffic load profile SHOULD be defined as described in Section 4.3.4.

During the sustain phase, the DUT/SUT SHOULD reach the "Initial concurrent TCP connections". The measured KPIs during the sustain phase MUST meet the test results validation criteria "a" and "b" defined in Section 7.9.3.3.

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

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

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

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

The test equipment SHOULD start to measure and record KPIs defined in Section 7.9.3.4. The frequency of measurement SHOULD be less than 2 seconds. Continue the test until all traffic profile phases are completed.

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

#### 7.9.4.3. Step 3: Test Iteration

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

### 8. IANA Considerations

The IANA has allocated 2001:0200::/48 for IPv6 testing, which is a 48-bit prefix from the [RFC4733] pool. For IPv4 testing, the IP subnet 198.18.0.0/15 has been assigned to the BMWG by the IANA. This assignment was made to minimize the chance of conflict in case a testing device were to be accidentally connected to part of the Internet. The specific use of the IPv4 addresses is detailed in [RFC2544] Appendix C.

### 9. Security Considerations

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

### 10. Contributors

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

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

### 11. Acknowledgements

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

Anand Vijayan, Baski Mohan, Chao Guo, Chris Brown, Chris Marshall, Jay Lindenauer, Michael Shannon, Mike Deichman, Ray Vinson, Ryan Riese, Tim Carlin, and Toulnay Orkun

## 12. References

### 12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

### 12.2. Informative References

- [RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <<https://www.rfc-editor.org/info/rfc2544>>.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <<https://www.rfc-editor.org/info/rfc2616>>.
- [RFC2647] Newman, D., "Benchmarking Terminology for Firewall Performance", RFC 2647, DOI 10.17487/RFC2647, August 1999, <<https://www.rfc-editor.org/info/rfc2647>>.
- [RFC3511] Hickman, B., Newman, D., Tadjudin, S., and T. Martin, "Benchmarking Methodology for Firewall Performance", RFC 3511, DOI 10.17487/RFC3511, April 2003, <<https://www.rfc-editor.org/info/rfc3511>>.
- [RFC4733] Schulzrinne, H. and T. Taylor, "RTP Payload for DTMF Digits, Telephony Tones, and Telephony Signals", RFC 4733, DOI 10.17487/RFC4733, December 2006, <<https://www.rfc-editor.org/info/rfc4733>>.
- [RFC6815] Bradner, S., Dubray, K., McQuaid, J., and A. Morton, "Applicability Statement for RFC 2544: Use on Production Networks Considered Harmful", RFC 6815, DOI 10.17487/RFC6815, November 2012, <<https://www.rfc-editor.org/info/rfc6815>>.



## Appendix A. Test Methodology - Security Effectiveness Evaluation

### A.1. Test Objective

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

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

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

### A.2. Test Bed Setup

The same Test bed MUST be used for security effectiveness test and as well as for benchmarking test cases defined in Section 7.

### A.3. Test Parameters

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

#### A.3.1. DUT/SUT Configuration Parameters

DUT/SUT configuration Parameters MUST conform to the requirements defined in Section 4.2. The same DUT configuration MUST be used for Security effectiveness test and as well as for benchmarking test cases defined in Section 7. The DUT/SUT MUST be configured in inline mode and all detected attack traffic MUST be dropped and the session Should be reset

#### A.3.2. Test Equipment Configuration Parameters

Test equipment configuration parameters MUST conform to the requirements defined in Section 4.3. The same Client and server IP ranges MUST be configured as used in the benchmarking test cases. In

addition, the following parameters MUST be documented for this benchmarking test:

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

#### A.4. Test Results Validation Criteria

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

- a. Number of failed Application transaction in the background traffic MUST be less than 0.01% of attempted transactions
- b. Number of Terminated TCP connections of the background traffic (due to unexpected TCP RST sent by DUT/SUT) MUST be less than 0.01% of total initiated TCP connections in the background traffic
- c. During the sustain phase, traffic should be forwarded at a constant rate
- d. False positive MUST NOT occur in the background traffic

#### A.5. Measurement

Following KPI metrics MUST be reported for this test scenario:

Mandatory KPIs:

- o Blocked CVEs: It should be represented in the following ways:
  - \* Number of blocked CVEs out of total CVEs
  - \* Percentage of blocked CVEs

- o Unblocked CVEs: It should be represented in the following ways:
  - \* Number of unblocked CVEs out of total CVEs
  - \* Percentage of unblocked CVEs
- o Background traffic behavior: it should represent one of the followings ways:
  - \* No impact (traffic transmission at a constant rate)
  - \* Minor impact (e.g. small spikes- +/- 100 Mbit/s)
  - \* Heavily impacted (e.g. large spikes and reduced the background throughput > 100 Mbit/s)
- o DUT/SUT reporting accuracy: DUT/SUT MUST report all detected vulnerabilities.

Optional KPIs:

- o List of unblocked CVEs

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

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

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

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

The DUT/SUT MUST reach the target objective (throughput) in sustain phase. The measured KPIs during the sustain phase MUST meet the test results validation criteria a, b, c and d defined in Appendix A.4.

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

##### A.6.2. Step 2: CVE Emulation

While generating the background traffic (in sustain phase), send the CVE traffic as defined in the parameter section.

The test equipment SHOULD start to measure and record all specified KPIs. The frequency of measurement MUST be less than 2 seconds. Continue the test until all CVEs are sent.

The measured KPIs MUST meet all the test results validation criteria a, b, c, and d defined in Appendix A.4.

In addition, the DUT/SUT SHOULD report the vulnerabilities correctly.

#### Appendix B. DUT/SUT Classification

This document attempts to classify the DUT/SUT in four different four different categories based on its maximum supported firewall throughput performance number defined in the vendor datasheet. This classification MAY help user to determine specific configuration scale (e.g., number of ACL entries), traffic profiles, and attack traffic profiles, scaling those proportionally to DUT/SUT sizing category.

The four different categories are Extra Small, Small, Medium, and Large. The RECOMMENDED throughput values for the following categories are:

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

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

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

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

#### Authors' Addresses

Balamuhunthan Balarajah  
Berlin  
Germany

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

Carsten Rossenhoevel  
EANTC AG  
Salzufer 14  
Berlin 10587  
Germany

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

Brian Monkman  
NetSecOPEN  
417 Independence Court  
Mechanicsburg, PA 17050  
USA

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

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 20, 2021

V. Vassilev  
Lightside Instruments AS  
February 16, 2021

A YANG Data Model for Network Interconnect Tester Management  
draft-vassilev-bmwg-network-interconnect-tester-05

## Abstract

This document introduces new YANG model for use in network interconnect testing containing modules of traffic generator and traffic analyzer.

## Status of This Memo

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

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

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

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

## Copyright Notice

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

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

## Table of Contents

1. Introduction . . . . .	2
1.1. Terminology . . . . .	2
1.1.1. Definitions and Acronyms . . . . .	2
1.1.2. Tree Diagram . . . . .	3
1.2. Problem Statement . . . . .	3
1.3. Objectives . . . . .	3
1.4. Solution . . . . .	4
2. Using the network interconnect tester model . . . . .	5
3. Traffic Generator Module Tree Diagram . . . . .	5
4. Traffic Analyzer Module Tree Diagram . . . . .	6
5. Traffic Generator Module YANG . . . . .	7
6. Traffic Analyzer Module YANG . . . . .	15
7. IANA Considerations . . . . .	22
7.1. URI Registration . . . . .	22
7.2. YANG Module Name Registration . . . . .	22
8. Security Considerations . . . . .	22
8.1. ietf-traffic-generator.yang . . . . .	23
8.2. ietf-traffic-analyzer.yang . . . . .	23
9. References . . . . .	23
9.1. Normative References . . . . .	23
9.2. Informative References . . . . .	24
Appendix A. Examples . . . . .	24
A.1. Basic Test Program . . . . .	25
A.2. Generating RFC2544 Testframes . . . . .	26
Author's Address . . . . .	26

## 1. Introduction

There is a need for standard mechanism to allow the specification and implementation of the transactions part of network tests. The mechanism should allow the control and monitoring of the data plane traffic in a transactional way. This document defines two YANG modules for test traffic generator and analyzer.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) defined in RFC 8342.

## 1.1. Terminology

## 1.1.1. Definitions and Acronyms

DUT: Device Under Test

TA: Traffic Analyzer

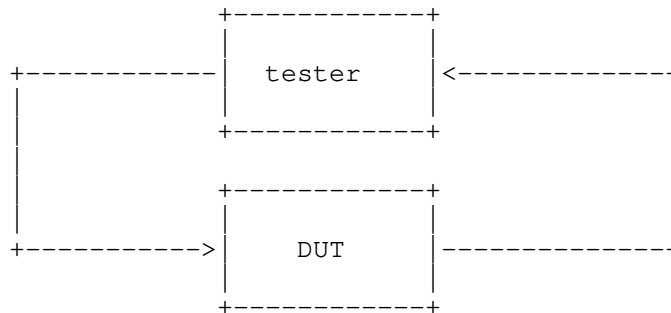
TG: Traffic Generator

### 1.1.2. Tree Diagram

For a reference to the annotations used in tree diagrams included in this document, please see YANG Tree Diagrams [RFC8340].

### 1.2. Problem Statement

Network interconnect tests require active network elements part of the tested network that generate test traffic and network elements that analyze the test traffic at one or more points of its path. A network interconnect tester is a device that can either generate test traffic, analyze test traffic or both. Here is a figure borrowed from [RFC2544] representing the horseshoe test setup topology consisting of a single tester and a single DUT connected in a network interconnect loop.



This document attempts to address the problem of defining YANG model of a network interconnect tester that can be used for development of vendor independent network interconnect tests and utilize the advantages of transactional management using standard protocols like NETCONF.

### 1.3. Objectives

This section describes some of the design objectives for the model. It should:

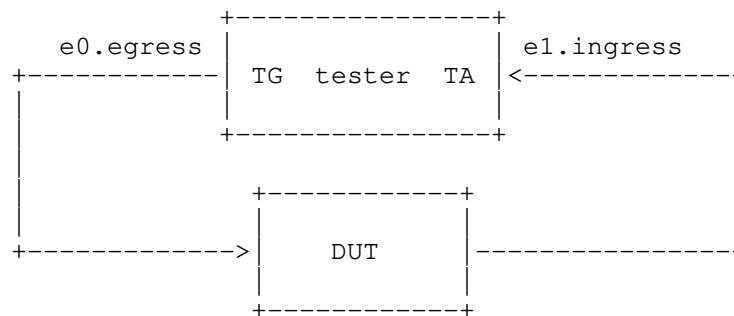
- o provide means to specify the generated traffic as streams of cyclic sequence of bursts with configurable frame size, frame data, interframe gap and interburst gap.
- o have a mandatory single stream mode and optional multi stream mode.



- o provide means for configuration of traffic streams with static frame data where frames with identical frame data are sent during the lifetime of the stream.
- o provide means for configuration of traffic streams with dynamic frame data where frames contain fields with dynamic data like generation time and sequence number.
- o allow third parties to augment the base module with alternative dynamic fields of frame data extensions.
- o provide means for realtime synchronization and orchestration of the generated streams.
- o provide counters for received test traffic frames and octets.
- o provide latency statistic in the case of test traffic with dynamic frame data that includes timestamp.
- o provide sequence number errors in the case of test traffic with dynamic frame data that includes sequence number.

#### 1.4. Solution

The proposed model splits the design into 2 modules - 1) Traffic Generator module (TG), 2) Traffic Analyzer module (TA). The modules are implemented as augmentations of the ietf-interfaces [RFC8343] module adding configuration and state data that models the functionality of a network interconnect tester. The TA and TG modules concept is illustrated with the following diagram of a tester with two interfaces (named e0 and e1) connected in a loop with single DUT:



## 2. Using the network interconnect tester model

Basic example of how the model can be used in transactional network test program to control the testers part of a network and report counter statistics and timing measurement data is presented in Appendix A. All example cases present the configuration and state data from a single test trial. The search algorithm logic that operates to control the trial configuration is outside the scope of this document. One of the examples demonstrates the use of the [RFC2544] defined testframe packet.

## 3. Traffic Generator Module Tree Diagram

```

module: ietf-traffic-generator
  augment /if:interfaces/if:interface:
    +--rw traffic-generator {egress-direction}?
      +--rw (type)?
        +--:(single-stream)
          +--rw testframe-type?      identityref
          +--rw frame-size           uint32
          +--rw frame-data?          string
          +--rw interframe-gap       uint32
          +--rw interburst-gap?      uint32
          +--rw frames-per-burst?    uint32
          +--rw src-mac-address?     yang:mac-address {ethernet}?
          +--rw dst-mac-address?     yang:mac-address {ethernet}?
          +--rw ether-type?          uint16 {ethernet}?
        +--:(multi-stream)
          +--rw streams
            +--rw stream* [id]
              +--rw id               uint32
              +--rw testframe-type?  identityref
              +--rw frame-size       uint32
              +--rw frame-data?      string
              +--rw interframe-gap   uint32
              +--rw interburst-gap?  uint32
              +--rw frames-per-burst uint32
              +--rw frames-per-stream uint32
              +--rw interstream-gap  uint32
              +--rw src-mac-address?
                | yang:mac-address {ethernet}?
              +--rw dst-mac-address?
                | yang:mac-address {ethernet}?
              +--rw ether-type?      uint16 {ethernet}?
          +--rw realtime-epoch?
            | yang:date-and-time {realtime-epoch}?
          +--rw total-frames?      uint64
      +--rw traffic-generator-ingress {ingress-direction}?

```

```

+--rw (type)?
|   +--:(single-stream)
|   |   +--rw testframe-type?      identityref
|   |   +--rw frame-size           uint32
|   |   +--rw frame-data?         string
|   |   +--rw interframe-gap       uint32
|   |   +--rw interburst-gap?     uint32
|   |   +--rw frames-per-burst?   uint32
|   |   +--rw src-mac-address?    yang:mac-address {ethernet}?
|   |   +--rw dst-mac-address?    yang:mac-address {ethernet}?
|   |   +--rw ether-type?         uint16 {ethernet}?
|   +--:(multi-stream)
|   |   +--rw streams
|   |   |   +--rw stream* [id]
|   |   |   |   +--rw id           uint32
|   |   |   |   +--rw testframe-type? identityref
|   |   |   |   +--rw frame-size   uint32
|   |   |   |   +--rw frame-data?  string
|   |   |   |   +--rw interframe-gap uint32
|   |   |   |   +--rw interburst-gap? uint32
|   |   |   |   +--rw frames-per-burst? uint32
|   |   |   |   +--rw frames-per-stream uint32
|   |   |   |   +--rw interstream-gap uint32
|   |   |   |   +--rw src-mac-address?
|   |   |   |   |   yang:mac-address {ethernet}?
|   |   |   |   +--rw dst-mac-address?
|   |   |   |   |   yang:mac-address {ethernet}?
|   |   |   |   +--rw ether-type?
|   |   |   |   |   uint16 {ethernet}?
|   +--rw realtime-epoch?
|   |   yang:date-and-time {realtime-epoch}?
+--rw total-frames?      uint64

```

#### 4. Traffic Analyzer Module Tree Diagram

```

module: ietf-traffic-analyzer
augment /if:interfaces/if:interface:
+--rw traffic-analyzer! {ingress-direction}?
|   +--rw filter! {filter}?
|   |   +--rw type      identityref
|   |   +--rw ether-type? uint16
|   +--ro state
|   |   +--ro pkts?      yang:counter64
|   |   +--ro octets?    yang:counter64
|   |   +--ro idle-octets? yang:counter64 {idle-octets-counter}?
|   |   +--ro errors?    yang:counter64
|   +--ro testframe-stats
|   |   +--ro testframe-pkts? yang:counter64

```

```

    +---ro sequence-errors?    yang:counter64
    +---ro payload-errors?     yang:counter64
    +---ro latency
        +---ro samples?       uint64
        +---ro min?           uint64
        +---ro max?           uint64
        +---ro average?       uint64
        +---ro latest?        uint64
    +---ro capture {capture}?
        +---ro frame* [sequence-number]
            +---ro sequence-number      uint64
            +---ro timestamp?           yang:date-and-time
            +---ro length?              uint32
            +---ro preceding-interframe-gap?  uint32
            +---ro data?                string
+---rw traffic-analyzer-egress! {egress-direction}?
    +---rw filter! {filter}?
    |   +---rw type      identityref
    +---ro state
        +---ro pkts?          yang:counter64
        +---ro octets?        yang:counter64
        +---ro idle-octets?    yang:counter64 {idle-octets-counter}?
        +---ro errors?        yang:counter64
        +---ro testframe-stats
            +---ro testframe-pkts?    yang:counter64
            +---ro sequence-errors?    yang:counter64
            +---ro payload-errors?     yang:counter64
            +---ro latency
                +---ro samples?       uint64
                +---ro min?           uint64
                +---ro max?           uint64
                +---ro average?       uint64
                +---ro latest?        uint64
        +---ro capture {capture}?
            +---ro frame* [sequence-number]
                +---ro sequence-number      uint64
                +---ro timestamp?           yang:date-and-time
                +---ro length?              uint32
                +---ro preceding-interframe-gap?  uint32
                +---ro data?                string

```

## 5. Traffic Generator Module YANG

<CODE BEGINS> file "ietf-traffic-generator@2021-02-16.yang"

```

module ietf-traffic-generator {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-traffic-generator";

```

```
prefix tg;

import ietf-interfaces {
  prefix if;
  reference
    "RFC 8343: A YANG Data Model For Interface Management";
}
import ietf-yang-types {
  prefix yang;
  reference
    "RFC 6991: Common YANG Data Types";
}
import iana-if-type {
  prefix ianaift;
  reference
    "RFC 7224: IANA Interface Type YANG Module";
}

organization
  "IETF Benchmarking Methodology Working Group";
contact
  "WG Web:  <http://tools.ietf.org/wg/bmwg/>
  WG List:  <mailto:bmwg@ietf.org>

  Editor:   Vladimir Vassilev
            <mailto:vladimir@lightside-instruments.com>";
description
  "This module contains a collection of YANG definitions for
  description and management of network interconnect testers.

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

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).
  This version of this YANG module is part of RFC XXXX; see
  the RFC itself for full legal notices.";

revision 2021-02-16 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for
    Network Interconnect Tester Management";
```

```
}

feature egress-direction {
  description
    "The device can generate traffic in the egress direction.";
}

feature ingress-direction {
  description
    "The device can generate traffic in the ingress direction.";
}

feature multi-stream {
  description
    "The device can generate multi-stream traffic.";
}

feature ethernet {
  description
    "The device can generate ethernet traffic.";
}

feature realtime-epoch {
  description
    "The device can generate traffic precisely
    at configured realtime epoch.";
}

identity testframe-type {
  description
    "Base identity for all testframe types.";
}

identity static {
  base testframe-type;
  description
    "Identity for static testframe.
    The frame data and size are constant.";
}

identity dynamic {
  base testframe-type;
  description
    "Identity to be used as base for dynamic
    testframe type identities defined
    in external modules.

    When used itself it identifies dynamic testframe
```

```
        where the last 18 octets of the payload contain
        incrementing sequence number field (8 octets)
        followed by timestamp field in the
        IEEE 1588-2008 format (10 octets). If frame data is defined
        for the last 18 octets of the payload it will be ignored
        and overwritten with dynamic data according to this
        specification.";
    }

    grouping common-data {
        description
            "Common configuration data.";
        leaf realtime-epoch {
            if-feature "realtime-epoch";
            type yang:date-and-time;
            description
                "If this leaf is present the stream generation will start
                at the specified realtime epoch.";
        }
        leaf total-frames {
            type uint64;
            description
                "If this leaf is present the traffic generation will stop
                after the specified number of frames are generated.";
        }
    }

    grouping burst-data {
        description
            "Generated traffic burst parameters.";
        leaf testframe-type {
            type identityref {
                base tg:testframe-type;
            }
            default "tg:static";
            description
                "In case of dynamic testframes this leaf specifies
                the dynamic testframe identity.";
        }
        leaf frame-size {
            type uint32;
            mandatory true;
            description
                "Size of the frames generated. For example for
                ethernet interfaces the following definition
                applies:

                Ethernet frame-size in octets includes:
```

```
* Destination Address (6 octets),
* Source Address (6 octets),
* Frame Type (2 octets),
* Data (min 46 octets or 42 octets + 4 octets 802.1Q tag),
* CRC Checksum (4 octets).

Ethernet frame-size does not include:
* Preamble (dependent on MAC configuration
             by default 7 octets),
* Start of frame delimiter (1 octet)

Minimum standard ethernet frame-size is 64 bytes but
generators might support smaller sizes for validation.";
}
leaf frame-data {
  type string {
    pattern '([0-9A-F]{2})*';
  }
  must 'string-length(.)<=(../frame-size*2)';
  description
    "The raw frame data specified as hexadecimal string.
    The specified data can be shorter then the ../frame-size
    value specifying only the header or the header and the
    payload with or without the 4 byte CRC Checksum
    in the case of a Ethernet frame.";
}
leaf interframe-gap {
  type uint32;
  mandatory true;
  description
    "Length of the idle period between generated frames.
    For example for ethernet interfaces the following
    definition applies:

    Ethernet interframe-gap between transmission of frames
    known as the interframe gap (IFG). A brief recovery time
    between frames allows devices to prepare for
    reception of the next frame. The minimum
    interframe gap is 96 bit times (12 octet times) (the time it
    takes to transmit 96 bits (12 octets) of raw data on the
    medium). However the preamble (7 octets) and start of
    frame delimiter (1 octet) are considered a constant gap that
    should be included in the interframe-gap. Thus the minimum
    value for standard ethernet transmission should be considered
    20 octets.";
}
leaf interburst-gap {
  type uint32;
```



```
        description
        "Similar to the interframe-gap but takes place between
        any two bursts of the stream.";
    }
    leaf frames-per-burst {
        type uint32;
        description
        "Number of frames contained in a burst";
    }
}

grouping multi-stream-data {
    description
    "Multi stream traffic generation parameters.";
    container streams {
        description
        "Non-presence container holding the configured stream list.";
        list stream {
            key "id";
            description
            "Each stream repeats a burst until frames-per-stream
            count is reached followed by interstream-gap delay.";
            leaf id {
                type uint32;
                description
                "Number specifying the order of the stream.";
            }
            uses burst-data;
            leaf frames-per-stream {
                type uint32;
                mandatory true;
                description
                "The count of frames to be generated before
                generation of the next stream is started.";
            }
            leaf interstream-gap {
                type uint32;
                mandatory true;
                description
                "Idle period after the last frame of the last burst.";
            }
        }
    }
}

grouping ethernet-data {
    description
    "Ethernet frame data specific parameters.";
```

```
reference
  "IEEE 802-2014 Clause 9.2";
leaf src-mac-address {
  type yang:mac-address;
  description
    "Source Address field of the generated Ethernet packet.";
}
leaf dst-mac-address {
  type yang:mac-address;
  description
    "Destination Address field of the generated Ethernet packet.";
}
leaf ether-type {
  type uint16;
  description
    "Length/Type field of the generated Ethernet packet.";
}
}

augment "/if:interfaces/if:interface" {
  description
    "Traffic generator augmentations of ietf-interfaces.";
  container traffic-generator {
    if-feature "egress-direction";
    description
      "Traffic generator for egress direction.";
    choice type {
      description
        "Choice of the type of the data model of the generator.
        Single or multi stream.";
      case single-stream {
        uses burst-data;
      }
      case multi-stream {
        uses multi-stream-data;
      }
    }
    uses common-data;
  }
  container traffic-generator-ingress {
    if-feature "ingress-direction";
    description
      "Traffic generator for ingress direction.";
    choice type {
      description
        "Choice of the type of the data model of the generator.
        Single or multi stream.";
      case single-stream {
```

```
        uses burst-data;
    }
    case multi-stream {
        uses multi-stream-data;
    }
}
uses common-data;
}
}

augment "/if:interfaces/if:interface/tg:traffic-generator/tg:type/"
+ "tg:single-stream" {
    when "derived-from-or-self(..if:type, 'ianaift:ethernetCsmacd')" {
        description
            "Ethernet interface type.";
    }
    if-feature "ethernet";
    description
        "Ethernet specific augmentation for egress
        single stream generator type.";
    uses ethernet-data;
}

augment "/if:interfaces/if:interface/tg:traffic-generator/"
+ "tg:type/tg:multi-stream/tg:streams/tg:stream" {
    when "derived-from-or-self(..../if:type, "
    + "'ianaift:ethernetCsmacd')" {
        description
            "Ethernet interface type.";
    }
    if-feature "ethernet";
    description
        "Ethernet specific augmentation for egress
        multi stream generator type.";
    uses ethernet-data;
}

augment "/if:interfaces/if:interface/tg:traffic-generator-ingress/"
+ "tg:type/tg:single-stream" {
    when "derived-from-or-self(..if:type, 'ianaift:ethernetCsmacd')" {
        description
            "Ethernet interface type.";
    }
    if-feature "ethernet";
    description
        "Ethernet specific augmentation for ingress
        single stream generator type.";
    uses ethernet-data;
}
```

```
}  
  
augment "/if:interfaces/if:interface/tg:traffic-generator-ingress/"  
  + "tg:type/tg:multi-stream/tg:streams/tg:stream" {  
    when "derived-from-or-self(..../if:type,"  
      + "'ianaift:ethernetCsmacd')" {  
      description  
        "Ethernet interface type.";  
    }  
    if-feature "ethernet";  
    description  
      "Ethernet specific augmentation for ingress  
      multi stream generator type.";  
    uses ethernet-data;  
  }  
}
```

<CODE ENDS>

## 6. Traffic Analyzer Module YANG

<CODE BEGINS> file "ietf-traffic-analyzer@2021-02-16.yang"

```
module ietf-traffic-analyzer {  
  yang-version 1.1;  
  namespace "urn:ietf:params:xml:ns:yang:ietf-traffic-analyzer";  
  prefix ta;  
  
  import ietf-interfaces {  
    prefix if;  
    reference  
      "RFC 8343: A YANG Data Model For Interface Management";  
  }  
  import ietf-yang-types {  
    prefix yang;  
    reference  
      "RFC 6991: Common YANG Data Types";  
  }  
  
  organization  
    "IETF Benchmarking Methodology Working Group";  
  contact  
    "WG Web:  <http://tools.ietf.org/wg/bmwg/>  
    WG List:  <mailto:bmwg@ietf.org>  
  
    Editor:    Vladimir Vassilev  
              <mailto:vladimir@lightside-instruments.com>";  
  description
```

"This module contains a collection of YANG definitions for description and management of network interconnect testers.

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

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.";

```
revision 2021-02-16 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for
    Network Interconnect Tester Management";
}

feature egress-direction {
  description
    "The device can analyze traffic from the egress direction.";
}

feature ingress-direction {
  description
    "The device can generate traffic from the ingress direction.";
}

feature filter {
  description
    "This feature indicates that the device implements
    filter that can specify a subset of packets to be
    analyzed when filtering is enabled.";
}

feature idle-octets-counter {
  description
    "This feature indicates that the device implements
    idle-octets counter that accumulates the time
    the link is not utilized. The minimum required
    idle gaps are not counted as idle octets.";
}
```

```
feature capture {
  description
    "This feature indicates that the device implements
    packet capture functionality.";
}

identity filter {
  description
    "Base filter identity.";
}

identity ethernet {
  base ta:filter;
  description
    "Ethernet packet fields filter.";
}

grouping statistics-data {
  description
    "Analyzer statistics.";
  leaf pkts {
    type yang:counter64;
    description
      "Total number of packets analyzed.";
  }
  leaf octets {
    type yang:counter64;
    description
      "This counter is identical with the in-octets/out-octets
      counters defined in RFC8343 except that it counts the
      octets since the analyzer was created.";
  }
  leaf idle-octets {
    if-feature "idle-octets-counter";
    type yang:counter64;
    description
      "Total accumulated period with no frame transmission
      taking place measured in octets at the current link
      speed. Octets not counted in ../octets but not idle are
      for example layer 1 framing octets - for Ethernet links
      7+1 preamble octets per packet.";
  }
  leaf errors {
    type yang:counter64;
    description
      "Count of packets with errors.
      Not counted in the pkts or captured.
      For example packets with CRC error.";
```

```
}
container testframe-stats {
  description
    "Statistics for testframe packets containing
    either sequence number, payload checksum,
    timestamp or any combination of these features.";
  leaf testframe-pkts {
    type yang:counter64;
    description
      "Total count of detected testframe packets.";
  }
  leaf sequence-errors {
    type yang:counter64;
    description
      "Total count of testframe packets with
      unexpected sequence number. After each sequence
      error the expected next sequence number is
      updated.";
  }
  leaf payload-errors {
    type yang:counter64;
    description
      "Total count of testframe packets with
      payload errors.";
  }
}
container latency {
  description
    "Latency statistics.";
  leaf samples {
    type uint64;
    description
      "Total count of packets used for estimating
      the latency statistics. Ideally
      samples=../testframe-stats.";
  }
  leaf min {
    type uint64;
    units "nanoseconds";
    description
      "Minimum measured latency.";
  }
  leaf max {
    type uint64;
    units "nanoseconds";
    description
      "Maximum measured latency.";
  }
  leaf average {
```

```
        type uint64;
        units "nanoseconds";
        description
            "The sum of all sampled latencies divided
             by the number of samples.";
    }
    leaf latest {
        type uint64;
        units "nanoseconds";
        description
            "Latency of the latest sample.";
    }
}
}
}

grouping capture-data {
    description
        "Grouping with statistics and data
         of one or more captured frame.";
    container capture {
        if-feature "capture";
        description
            "Statistics and data of
             one or more captured frames.";
        list frame {
            key "sequence-number";
            description
                "Statistics and data of a captured frame.";
            leaf sequence-number {
                type uint64;
                description
                    "Incremental counter of frames captured.";
            }
            leaf timestamp {
                type yang:date-and-time;
                description
                    "Timestamp of the moment the frame was captured.";
            }
            leaf length {
                type uint32;
                description
                    "Frame length. Ideally the data captured will be
                     of the same length but can be shorter
                     depending on implementation limitations.";
            }
            leaf preceding-interframe-gap {
                type uint32;
            }
        }
    }
}
```



```
        units "nanoseconds";
        description
            "Measured delay between the reception of the previous
            frame was completed and the reception of the current
            frame was started.";
    }
    leaf data {
        type string {
            pattern '([0-9A-F]{2})*';
        }
        description
            "Raw data of the captured frame.";
    }
}

grouping filter-data {
    description
        "Grouping with a filter container specifying the filtering
        rules for processing only a specific subset of the
        frames.";
    container filter {
        if-feature "filter";
        presence "When present packets are
            filtered before analyzed according
            to the filter type";
        description
            "Contains the filtering rules for processing only
            a specific subset of the frames.";
        leaf type {
            type identityref {
                base ta:filter;
            }
            mandatory true;
            description
                "Type of the applied filter. External modules can
                define alternative filter type identities.";
        }
    }
}

augment "/if:interfaces/if:interface" {
    description
        "Traffic analyzer augmentations of ietf-interfaces.";
    container traffic-analyzer {
        if-feature "ingress-direction";
        presence "Enables the traffic analyzer for ingress traffic.";
    }
}
```

```
    description
      "Traffic analyzer for ingress direction.";
    uses filter-data;
    container state {
      config false;
      description
        "State data.";
      uses statistics-data;
      uses capture-data;
    }
  }
  container traffic-analyzer-egress {
    if-feature "egress-direction";
    presence "Enables the traffic analyzer for egress traffic.";
    description
      "Traffic analyzer for egress direction.";
    uses filter-data;
    container state {
      config false;
      description
        "State data.";
      uses statistics-data;
      uses capture-data;
    }
  }
}

augment "/if:interfaces/if:interface/ta:traffic-analyzer/ta:filter" {
  when "derived-from-or-self(ta:type, 'ta:ethernet')";
  description
    "Ethernet frame specific filter type.";
  leaf ether-type {
    type uint16;
    description
      "The Ethernet Type (or Length) value
       defined by IEEE 802.";
    reference
      "IEEE 802-2014 Clause 9.2";
  }
}
}
```

<CODE ENDS>

## 7. IANA Considerations

This document registers two URIs and two YANG modules.

### 7.1. URI Registration

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-traffic-generator  
URI: urn:ietf:params:xml:ns:yang:ietf-traffic-analyzer

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

### 7.2. YANG Module Name Registration

This document registers two YANG module in the YANG Module Names registry YANG [RFC6020].

name: ietf-traffic-generator  
namespace: urn:ietf:params:xml:ns:yang:ietf-traffic-generator  
prefix: tg  
reference: RFC XXXX

name: ietf-traffic-analyzer  
namespace: urn:ietf:params:xml:ns:yang:ietf-traffic-analyzer  
prefix: ta  
reference: RFC XXXX

## 8. Security Considerations

The YANG modules defined in this document are designed to be accessed via the NETCONF protocol RFC 6241 [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory to implement secure transport is SSH RFC 6242 [RFC6242]. The NETCONF access control model RFC 6536 [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of data nodes defined in this YANG module which are writable/creatable/deletable (i.e. config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g. edit-config) to these data nodes without proper protection can have a negative effect

on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

#### 8.1. ietf-traffic-generator.yang

The ietf-traffic-generator YANG module controls a stateless traffic generator which is intended to be used for testing and verification purposes but can be used for malicious purposes like generating network traffic part of a Denial-of-Service (DoS) attack. This should be taken into consideration when granting write access to the following container and descendant data nodes:

- o /if:interfaces/if:interface/tg:traffic-generator

#### 8.2. ietf-traffic-analyzer.yang

The ietf-traffic-analyzer YANG module controls a traffic analyzer which is designed for use in testing and verification but can be used for reading information contained in packets sent and received on any of the interfaces on systems that implement the capture feature. This should be taken into consideration when granting read access to the following container and descendant data nodes:

- o /if:interfaces/if:interface/ta:traffic-analyzer/ta:capture

### 9. References

#### 9.1. Normative References

- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<https://www.rfc-editor.org/info/rfc6536>>.

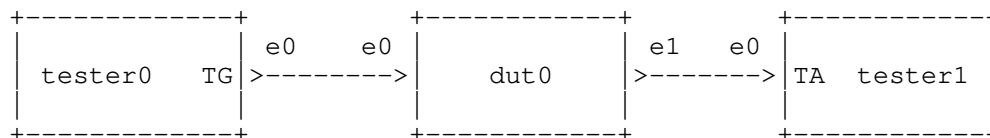
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", RFC 7224, DOI 10.17487/RFC7224, May 2014, <<https://www.rfc-editor.org/info/rfc7224>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.

## 9.2. Informative References

- [IEEE1588] IEEE, "IEEE 1588-2008", 2008.
- [IEEE802.3-2014] IEEE WG802.3 - Ethernet Working Group, "IEEE 802.3-2014", 2014.
- [RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <<https://www.rfc-editor.org/info/rfc2544>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.

## Appendix A. Examples

The following topology will be used for the examples in this section:



## A.1. Basic Test Program

This pseudo code program orchestrates a network test and shows how the model can be used:

```
#Connect to network
net=connect("topology.xml")

# Configure DUTs and enable traffic-analyzers
net.node("dut0").edit( \
    "create /interfaces/interface[name='e0'] -- type=ethernetCsmacd")
net.node("dut0").edit(
    "create /interfaces/interface[name='e1'] -- type=ethernetCsmacd")
net.node("dut0").edit(
    "create /flows/flow[id='t0'] -- match/in-port=e0 "
    "actions/action[order='0']/output-action/out-port=e1")

net.node("tester1").edit(
    "create /interfaces/interface[name='e0']/traffic-analyzer")
net.commit()

#Get network state - before
before=net.get()

# Start traffic
net.node("tester0").edit(
    "create /interfaces/interface[name='e0']/traffic-generator -- "
    "frame-size=64 interframe-gap=20")

net.commit()

time.sleep(60)

# Stop traffic
net.node("tester1").edit("delete /interfaces/interface[name='e0']/\"
    "traffic-generator")
net.commit()

#Get network state - after
after=net.get()

#Report
sent_pkts=delta("tester0",before,after,
    "/interfaces/interface[name='e0']/statistics/out-unicast-pkts")

received_pkts=delta("tester1",before,after,
    "/interfaces/interface[name='e0']/statistics/in-unicast-pkts")
```

```
latency_max=absolute(after,  
  "/interfaces/interface[name='e0']/traffic-analyzer/state/"  
  "testframe-stats/latency/max")  
  
#Cleanup  
net.node("tester1").edit(  
  "delete /interfaces/interface/traffic-analyzer")  
net.node("dut0").edit("delete /flows")  
net.node("dut0").edit("delete /interfaces")  
net.commit()
```

#### A.2. Generating RFC2544 Testframes

In sec. C.2.6.4 Test Frames a detailed format is specified. The frame-data leaf allows full control over the generated frames payload.

```
...  
net.node("tester1").edit(  
  "merge /interfaces/interface[name='e0']/"  
  "traffic-generator -- frame-data="  
  "6CA96F00000026CA96F000000108004500"  
  "002ED4A5000000A115816C0000201C000"  
  "0202C0200007001A00000010203040506"  
  "0708090A0B0C0D0E0F101112")  
...
```

#### Author's Address

Vladimir Vassilev  
Lightside Instruments AS

Email: vladimir@lightside-instruments.com