Benchmarking Working Group
Internet-Draft
Intended status: Informational
Expires: September 7, 2020

M. Konstantynowicz, Ed.
V. Polak, Ed.
Cisco Systems

March 06, 2020

Probabilistic Loss Ratio Search for Packet Throughput (PLRsearch)
draft-vpolak-bmwg-plrsearch-03

Abstract

This document addresses challenges while applying methodologies described in [RFC2544] to benchmarking software based NFV (Network Function Virtualization) data planes over an extended period of time, sometimes referred to as "soak testing". Packet throughput search approach proposed by this document assumes that system under test is probabilistic in nature, and not deterministic.

Status of This Memo

Copyright Notice

Table of Contents

1. Motivation

Network providers are interested in throughput a networking system can sustain.

[RFC2544] assumes loss ratio is given by a deterministic function of offered load. But NFV software systems are not deterministic enough. This makes deterministic algorithms (such as Binary Search per [RFC2544] and [draft-vpolak-mkonstan-bmwg-mlrsearch] with single trial) to return results, which when repeated show relatively high standard deviation, thus making it harder to tell what "the throughput" actually is.

We need another algorithm, which takes this indeterminism into account.

2. Relation To RFC2544

The aim of this document is to become an extension of [RFC2544] suitable for benchmarking networking setups such as software based NFV systems.

3. Terms And Assumptions

Due to the indeterministic nature of certain NFV systems that are the targetted by PLRsearch algorithm, existing network benchmarking terms are explicated and a number of new terms and assumptions are introduced.

3.1. Device Under Test

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 that is outside the scope of this document.

3.2. System Under Test

System under test (SUT) is a part of the whole test setup whose performance is to be benchmarked. The complete methodology contains other parts, whose performance is either already established, or not affecting the benchmarking result.

3.3. SUT Configuration

Usually, system under test allows different configurations, affecting its performance. The rest of this document assumes a single configuration has been chosen.

3.4. SUT Setup

Similarly to [RFC2544], it is assumed that the system under test has been updated with all the packet forwarding information it needs, before the trial measurements (see below) start.

3.5. Network Traffic

Network traffic is a type of interaction between system under test and the rest of the system (traffic generator), used to gather information about the system under test performance. PLRsearch is applicable only to areas where network traffic consists of packets.

3.6. Packet

Unit of interaction between traffic generator and the system under test. Term "packet" is used also as an abstraction of Ethernet frames.

3.6.1. Packet Offered

Packet can be offered, which means it is sent from traffic generator to the system under test.

Each offered packet is assumed to become received or lost in a short time.

3.6.2. Packet Received

Packet can be received, which means the traffic generator verifies it has been processed. Typically, when it is succesfully sent from the system under test to traffic generator.

It is assumed that each received packet has been caused by an offered packet, so the number of packets received cannot be larger than the number of packets offered.

3.6.3. Packet Lost

Packet can be lost, which means sent but not received in a timely manner.

It is assumed that each lost packet has been caused by an offered packet, so the number of packets lost cannot be larger than the number of packets offered.

Usually, the number of packets lost is computed as the number of packets offered, minus the number of packets received.

3.6.4. Other Packets

PLRsearch is not considering other packet behaviors known from networking (duplicated, reordered, greatly delayed), assuming the

test specification reclassifies those behaviors to fit into the first three categories.

3.7. Traffic Profile

Usually, the performance of the system under test depends on a "type" of a particular packet (for example size), and "composition" if the network traffic consists of a mixture of different packet types.

Also, some systems under test contain multiple "ports" packets can be offered to and received from.

All such qualities together (but not including properties of trial measurements) are called traffic profile.

Similarly to system under test configuration, this document assumes only one traffic profile has been chosen for a particular test.

3.8. Traffic Generator

Traffic generator is the part of the whole test setup, distinct from the system under test, responsible both for offering packets in a highly predictable manner (so the number of packets offered is known), and for counting received packets in a precise enough way (to distinguish lost packets from tolerably delayed packets).

Traffic generator must offer only packets compatible with the traffic profile, and only count similarly compatible packets as received.

Criteria defining which received packets are compatible are left for test specification to decide.

3.9. Offered Load

Offered load is an aggregate rate (measured in packets per second) of network traffic offered to the system under test, the rate is kept constant for the duration of trial measurement.

3.10. Trial Measurement

Trial measurement is a process of stressing (previously setup) system under test by offering traffic of a particular offered load, for a particular duration.

After that, the system has a short time to become idle, while the traffic generator decides how many packets were lost.

After that, another trial measurement (possibly with different offered load and duration) can be immediately performed. Traffic generator should ignore received packets caused by packets offered in previous trial measurements.

3.11. Trial Duration

Duration for which the traffic generator was offering packets at constant offered load.

In theory, care has to be taken to ensure the offered load and trial duration predict integer number of packets to offer, and that the traffic generator really sends appropriate number of packets within precisely enough timed duration. In practice, such consideration do not change PLRsearch result in any significant way.

3.12. Packet Loss

Packet loss is any quantity describing a result of trial measurement.

It can be loss count, loss rate or loss ratio. Packet loss is zero (or non-zero) if either of the three quantities are zero (or non-zero, respecively).

3.12.1. Loss Count

Number of packets lost (or delayed too much) at a trial measurement by the system under test as determined by packet generator. Measured in packets.

3.12.2. Loss Rate

Loss rate is computed as loss count divided by trial duration. Measured in packets per second.

3.12.3. Loss Ratio

Loss ratio is computed as loss count divided by number of packets offered. Measured as a real (in practice rational) number between zero or one (including).

3.13. Trial Order Independent System

Trial order independent system is a system under test, proven (or just assumed) to produce trial measurement results that display trial order independence.

That means when a pair of consequent trial measurements are performed, the probability to observe a pair of specific results is the same, as the probability to observe the reversed pair of results whe performing the reversed pair of consequent measurements.

PLRsearch assumes the system under test is trial order independent.

In practice, most system under test are not entirely trial order independent, but it is not easy to devise an algorithm taking that into account.

3.14. Trial Measurement Result Distribution

When a trial order independent system is subjected to repeated trial measurements of constant duration and offered load, Law of Large Numbers implies the observed loss count frequencies will converge to a specific probability distribution over possible loss counts.

This probability distribution is called trial measurement result distribution, and it depends on all properties fixed when defining it. That includes the system under test, its chosen configuration, the chosen traffic profile, the offered load and the trial duration.

As the system is trial order independent, trial measurement result distribution does not depend on results of few initial trial measurements, of any offered load or (finite) duration.

3.15. Average Loss Ratio

Probability distribution over some (finite) set of states enables computation of probability-weighted average of any quantity evaluated on the states (called the expected value of the quantity).

Average loss ratio is simply the expected value of loss ratio for a given trial measurement result distribution.

3.16. Duration Independent System

Duration independent system is a trial order independent system, whose trial measurement result distribution is proven (or just assumed) to display practical independence from trial duration. See definition of trial duration for discussion on practical versus theoretical.

The only requirement is for average loss ratio to be independent of trial duration.

In theory, that would necessitate each trial measurement result distribution to be a binomial distribution. In practice, more distributions are allowed.

PLRsearch assumes the system under test is duration independent, at least for trial durations typically chosen for trial measurements initiated by PLRsearch.

3.17. Load Regions

For a duration independent system, trial measurement result distribution depends only on offered load.

It is convenient to name some areas of offered load space by possible trial results.

3.17.1. Zero Loss Region

A particular offered load value is said to belong to zero loss region, if the probability of seeing non-zero loss trial measurement result is exactly zero, or at least practically indistinguishable from zero.

3.17.2. Guaranteed Loss Region

A particular offered load value is said to belong to guaranteed loss region, if the probability of seeing zero loss trial measurement result (for non-negligible count of packets offered) is exactly zero, or at least practically indistinguishable from zero.

3.17.3. Non-Deterministic Region

A particular offered load value is said to belong to non-deterministic region, if the probability of seeing zero loss trial measurement result (for non-negligible count of packets offered) is practically distinguishable from both zero and one.

3.17.4. Normal Region Ordering

Although theoretically the three regions can be arbitrary sets, this document assumes they are intervals, where zero loss region contains values smaller than non-deterministic region, which in turn contains values smaller than guaranteed loss region.

3.18. Deterministic System

A hypothetical duration independent system with normal region ordering, whose non-deterministic region is extremely narrow (only present due to "practical distinguishibility" and cases when the expected number of packets offered is not and integer).

A duration independent system which is not deterministic is called non- deterministic system.

3.19. Througphput

Throughput is the highest offered load provably causing zero packet loss for trial measurements of duration at least 60 seconds.

For duration independent systems with normal region ordering, the throughput is the highest value within the zero loss region.

3.20. Deterministic Search

Any algorithm that assumes each measurement is a proof of the offered load belonging to zero loss region (or not) is called deterministic search.

This definition includes algorithms based on "composite measurements" which perform multiple trial measurements, somehow re-classifying results pointing at non-deterministic region.

Binary Search is an example of deterministic search.

Single run of a deterministic search launched against a deterministic system is guaranteed to find the throughput with any prescribed precision (not better than non-deterministic region width).

Multiple runs of a deterministic search launched against a non-deterministic system can return varied results within non-deterministic region. The exact distribution of deterministic search results depends on the algorithm used.

3.21. Probabilistic Search

Any algorithm which performs probabilistic computations based on observed results of trial measurements, and which does not assume that non-deterministic region is practically absent, is called probabilistic search.

A probabilistic search algorithm, which would assume that non-deterministic region is practically absent, does not really need to

perform probabilistic computations, so it would become a deterministic search.

While probabilistic search for estimating throughput is possible, it would need a careful model for boundary between zero loss region and non-deterministic region, and it would need a lot of measurements of almost surely zero loss to reach good precision.

3.22. Loss Ratio Function

For any duration independent system, the average loss ratio depends only on offered load (for a particular test setup).

Loss ratio function is the name used for the function mapping offered load to average loss ratio.

This function is initially unknown.

3.23. Target Loss Ratio

Input parameter of PLRsearch. The average loss ratio the output of PLRsearch aims to achieve.

3.24. Critical Load

Aggregate rate of network traffic, which would lead to average loss ratio exactly matching target loss ratio, if used as the offered load for infinite many trial measurement.

3.25. Critical Load Estimate

Any quantitative description of the possible critical load PLRsearch is able to give after observing finite amount of trial measurements.

3.26. Fitting Function

Any function PLRsearch uses internally instead of the unknown loss ratio function. Typically chosen from small set of formulas (shapes) with few parameters to tweak.

3.27. Shape of Fitting Function

Any formula with few undetermined parameters.

3.28. Parameter Space

A subset of Real Coordinate Space. A point of parameter space is a vector of real numbers. Fitting function is defined by shape (a formula with parameters) and point of parameter space (specifying values for the parameters).

4. Abstract Algorithm

4.1. High level description

PLRsearch accepts some input arguments, then iteratively performs trial measurements at varying offered loads (and durations), and returns some estimates of critical load.

PLRsearch input arguments form three groups.

First group has a single argument: measurer. This is a callback (function) accepting offered load and duration, and returning the measured loss count.

Second group consists of load related arguments required for measurer to work correctly, typically minimal and maximal load to offer. Also, target loss ratio (if not hardcoded) is a required argument.

Third group consists of time related arguments. Typically the duration for the first trial measurement, duration increment per subsequent trial measurement, and total time for search. Some PLRsearch implementation may use estimation accuracy parameters as an exit condition instead of total search time.

The returned quantities should describe the final (or best) estimate of critical load. Implementers can chose any description that suits their users, typically it is average and standard deviation, or lower and upper boundary.

4.2. Main Ideas

The search tries to perform measurements at offered load close to the critical load, because measurement results at offered loads far from the critical load give less information on precise location of the critical load. As virtually every trial measurement result alters the estimate of the critical load, offered loads vary as they approach the critical load.

The only quantity of trial measurement result affecting the computation is loss count. No latency (or other information) is taken into account.

PLRsearch uses Bayesian Inference, computed using numerical integration, which takes long time to get reliable enough results. Therefore it takes some time before the most recent measurement result starts affecting subsequent offered loads and critical rate estimates.

During the search, PLRsearch spawns few processes that perform numerical computations, the main process is calling the measurer to perform trial measurements, without any significant delays between them. The durations of the trial measurements are increasing linearly, as higher number of trial measurement results take longer to process.

4.2.1. Trial Durations

[RFC2544] motivates the usage of at least 60 second duration by the idea of the system under test slowly running out of resources (such as memory buffers).

Practical results when measuring NFV software systems show that relative change of trial duration has negligible effects on average loss ratio, compared to relative change in offered load.

While the standard deviation of loss ratio usually shows some effects of trial duration, they are hard to model. So PLRsearch assumes SUT is duration independent, and chooses trial durations only based on numeric integration requirements.

4.2.2. Target Loss Ratio

(TODO: Link to why we think 1e-7 is acceptable loss ratio.)

4.3. PLRsearch Building Blocks

Here we define notions used by PLRsearch which are not applicable to other search methods, nor probabilistic systems under test in general.

4.3.1. Bayesian Inference

PLRsearch uses a fixed set of fitting function shapes, and uses Bayesian inference to track posterior distribution on each fitting function parameter space.

Specifically, the few parameters describing a fitting function become the model space. Given a prior over the model space, and trial duration results, a posterior distribution is computed, together with quantities describing the critical load estimate.

Likelihood of a particular loss count is computed using Poisson distribution of average loss rate given by the fitting function (at specific point of parameter space).

Side note: Binomial Distribution is a better fit compared to Poisson distribution (acknowledging that the number of packets lost cannot be higher than the number of packets offered), but the difference tends to be relevant only in high loss region. Using Poisson distribution lowers the impact of measurements in high loss region, thus helping the algorithm to converge towards critical load faster.

4.3.2. Iterative Search

The idea PLRsearch is to iterate trial measurements, using Bayesian inference to compute both the current estimate of the critical load and the next offered load to measure at.

The required numerical computations are done in parallel with the trial measurements.

This means the result of measurement "n" comes as an (additional) input to the computation running in parallel with measurement "n+1", and the outputs of the computation are used for determining the offered load for measurement "n+2".

Other schemes are possible, aimed to increase the number of measurements (by decreasing their duration), which would have even higher number of measurements run before a result of a measurement affects offered load.

4.3.3. Fitting Functions

To make the space of possible loss ratio functions more tractable the algorithm uses only few fitting function shapes for its predicitons. As the search algorithm needs to evaluate the function also far away from the critical load, the fitting function have to be reasonably behaved for every positive offered load, specifically cannot cannot predict non-positive packet loss ratio.

4.3.4. Measurement Impact

Results from trials far from the critical load are likely to affect the critical load estimate negatively, as the fitting functions do not need to be good approximations there. This is true mainly for guaranteed loss region, as in zero loss region even badly behaved fitting function predicts loss count to be "almost zero", so seeing a measurement confirming the loss has been zero indeed has small impact.

Discarding some results, or "suppressing" their impact with ad-hoc methods (other than using Poisson distribution instead of binomial) is not used, as such methods tend to make the overall search unstable. We rely on most of measurements being done (eventually) near the critical load, and overweighting far-off measurements (eventually) for well-behaved fitting functions.

4.3.5. Fitting Function Coefficients Distribution

To accomodate systems with different behaviours, a fitting function is expected to have few numeric parameters affecting its shape (mainly affecting the linear approximation in the critical region).

The general search algorithm can use whatever increasing fitting function, some specific functions are described later.

It is up to implementer to chose a fitting function and prior distribution of its parameters. The rest of this document assumes each parameter is independently and uniformly distributed over a common interval. Implementers are to add non-linear transformations into their fitting functions if their prior is different.

4.3.6. Exit Condition

Exit condition for the search is either the standard deviation of the critical load estimate becoming small enough (or similar), or overal search time becoming long enough.

The algorithm should report both average and standard deviation for its critical load posterior.

4.3.7. Integration

The posterior distributions for fitting function parameters are not be integrable in general.

The search algorithm utilises the fact that trial measurement takes some time, so this time can be used for numeric integration (using suitable method, such as Monte Carlo) to achieve sufficient precision.

4.3.8. Optimizations

After enough trials, the posterior distribution will be concentrated in a narrow area of the parameter space. The integration method should take advantage of that.

Even in the concentrated area, the likelihood can be quite small, so the integration algorithm should avoid underflow errors by some means, for example by tracking the logarithm of the likelihood.

4.3.9. Offered Load Selection

The simplest rule is to set offered load for next trial measurememnt equal to the current average (both over posterio and over fitting function shapes) of the critical load estimate.

Contrary to critical load estimate computation, heuristic algorithms affecting offered load selection do not introduce instability, and can help with convergence speed.

4.3.10. Trend Analysis

If the reported averages follow a trend (maybe without reaching equilibrium), average and standard deviation COULD refer to the equilibrium estimates based on the trend, not to immediate posterior values.

But such post-processing is discouraged, unless a clear reason for the trend is known. Frequently, presence of such a trend is a sign of some of PLRsearch assumption being violated (usually trial order independence or duration independence).

It is RECOMMENDED to report any trend quantification together with direct critical load estimate, so users can draw their own conclusion. Alternatively, trend analysis may be a part of exit conditions, requiring longer searches for systems displaying trends.

5. Known Implementations

The only known working implementation of PLRsearch is in Linux Foundation FD.io CSIT open-source project [FDio-CSIT-PLRsearch].

5.1. FD.io CSIT Implementation Specifics

The search receives min_rate and max_rate values, to avoid measurements at offered loads not supporeted by the traffic generator.

The implemented tests cases use bidirectional traffic. The algorithm stores each rate as bidirectional rate (internally, the algorithm is agnostic to flows and directions, it only cares about overall counts of packets sent and packets lost), but debug output from traffic generator lists unidirectional values.

5.1.1. Measurement Delay

In a sample implemenation in FD.io CSIT project, there is roughly 0.5 second delay between trials due to restrictons imposed by packet traffic generator in use (T-Rex).

As measurements results come in, posterior distribution computation takes more time (per sample), although there is a considerable constant part (mostly for inverting the fitting functions).

Also, the integrator needs a fair amount of samples to reach the region the posterior distribution is concentrated at.

And of course, speed of the integrator depends on computing power of the CPUs the algorithm is able to use.

All those timing related effects are addressed by arithmetically increasing trial durations with configurable coefficients (currently 5.1 seconds for the first trial, each subsequent trial being 0.1 second longer).

5.1.2. Rounding Errors and Underflows

In order to avoid them, the current implementation tracks natural logarithm (instead of the original quantity) for any quantity which is never negative. Logarithm of zero is minus infinity (not supported by Python), so special value "None" is used instead. Specific functions for frequent operations (such as "logarithm of sum of exponentials") are defined to handle None correctly.

5.1.3. Fitting Functions

Current implementation uses two fitting functions. In general, their estimates for critical rate differ, which adds a simple source of systematic error, on top of posterior dispersion reported by integrator. Otherwise the reported stdev of critical rate estimate is unrealistically low.

Both functions are not only increasing, but also convex (meaning the rate of increase is also increasing).

As Primitive Function to any positive function is an increasing function, and Primitive Function to any increasing function is convex function; both fitting functions were constructed as double Primitive Function to a positive function (even though the intermediate increasing function is easier to describe).

As not any function is integrable, some more realistic functions (especially with respect to behavior at very small offered loads) are not easily available.

Both fitting functions have a "central point" and a "spread", varied by simply shifting and scaling (in x-axis, the offered load direction) the function to be doubly integrated. Scaling in y-axis (the loss rate direction) is fixed by the requirement of transfer rate staying nearly constant in very high offered loads.

In both fitting functions (as they are a double Primitive Function to a symmetric function), the "central point" turns out to be equal to the aforementioned limiting transfer rate, so the fitting function parameter is named "mrr", the same quantity CSIT Maximum Receive Rate tests are designed to measure.

Both fitting functions return logarithm of loss rate, to avoid rounding errors and underflows. Parameters and offered load are not given as logarithms, as they are not expected to be extreme, and the formulas are simpler that way.

Both fitting functions have several mathematically equivalent formulas, each can lead to an overflow or underflow in different places. Overflows can be eliminated by using different exact formulas for different argument ranges. Underflows can be avoided by using approximate formulas in affected argument ranges, such ranges have their own formulas to compute. At the end, both fitting function implementations contain multiple "if" branches, discontinuities are a possibility at range boundaries.

5.1.3.1. Stretch Function

The original function (before applying logarithm) is Primitive Function to Logistic Function. The name "stretch" is used for related a function in context of neural networks with sigmoid activation function.

Formula for stretch fitting function: average loss rate (r) computed from offered load (b), mrr parameter (m) and spread parameter (a), given as InputForm of Wolfram language:

```
r = (a*(1 + E^(m/a))*Log[(E^(b/a) + E^(m/a))/(1 + E^(m/a))])/E^(m/a)
```

5.1.3.2. Erf Function

The original function is double Primitive Function to Gaussian Function. The name "erf" comes from error function, the first primitive to Gaussian.

Formula for erf fitting function: average loss rate (r) computed from offered load (b), mrr parameter (m) and spread parameter (a), given as InputForm of Wolfram language:

```
 r = ((a*(E^(-((b - m)^2/a^2)) - E^(-(m^2/a^2))))/Sqrt[Pi] + m*Erfc[m/a]
     + (b - m)*Erfc[(-b + m)/a])/(1 + Erf[m/a])
```

5.1.4. Prior Distributions

The numeric integrator expects all the parameters to be distributed (independently and) uniformly on an interval (-1, 1).

As both "mrr" and "spread" parameters are positive and not dimensionless, a transformation is needed. Dimentionality is inherited from max_rate value.

The "mrr" parameter follows a Lomax Distribution with alpha equal to one, but shifted so that mrr is always greater than 1 packet per second.

The "stretch" parameter is generated simply as the "mrr" value raised to a random power between zero and one; thus it follows a Reciprocal Distribution.

5.1.5. Integrator

After few measurements, the posterior distribution of fitting function arguments gets quite concentrated into a small area. The integrator is using Monte Carlo with Importance Sampling where the biased distribution is Bivariate Gaussian distribution, with deliberately larger variance. If the generated sample falls outside (-1, 1) interval, another sample is generated.

The the center and the covariance matrix for the biased distribution is based on the first and second moments of samples seen so far (within the computation), with the following additional features designed to avoid hyper-focused distributions.

Each computation starts with the biased distribution inherited from the previous computation (zero point and unit covariance matrix is used in the first computation), but the overal weight of the data is set to the weight of the first sample of the computation. Also, the center is set to the first sample point. When additional samples come, their weight (including the importance correction) is compared to the weight of data seen so far (within the computation). If the new sample is more than one e-fold more impactful, both weight values (for data so far and for the new sample) are set to (geometric) average if the two weights. Finally, the actual sample generator

uses covariance matrix scaled up by a configurable factor (8.0 by default).

This combination showed the best behavior, as the integrator usually follows two phases. First phase (where inherited biased distribution or single big sasmples are dominating) is mainly important for locating the new area the posterior distribution is concentrated at. The second phase (dominated by whole sample population) is actually relevant for the critical rate estimation.

5.1.6. Offered Load Selection

First two measurements are hardcoded to happen at the middle of rate interval and at max_rate. Next two measurements follow MRR-like logic, offered load is decreased so that it would reach target loss ratio if offered load decrease lead to equal decrease of loss rate.

Basis for offered load for next trial measurements is the integrated average of current critical rate estimate, averaged over fitting function.

There is one workaround implemented, aimed at reducing the number of consequent zero loss measurements. The workaround first stores every measurement result which loss ratio was the targed loss ratio or higher. Sorted list (called lossy loads) of such results is maintained.

When a sequence of one or more zero loss measurement results is encountered, a smallest of lossy loads is drained from the list. If the estimate average is smaller than the drained value, a weighted average of this estimate and the drained value is used as the next offered load. The weight of the drained value doubles with each additional consecutive zero loss results.

This behavior helps the algorithm with convergence speed, as it does not need so many zero loss result to get near critical load. Using the smallest (not drained yet) of lossy loads makes it sure the new offered load is unlikely to result in big loss region. Draining even if the estimate is large enough helps to discard early measurements when loss hapened at too low offered load. Current implementation adds 4 copies of lossy loads and drains 3 of them, which leads to fairly stable behavior even for somewhat inconsistent SUTs.

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

To be added.

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

[draft-vpolak-mkonstan-bmwg-mlrsearch]
"Multiple Loss Ratio Search for Packet Throughput (MLRsearch)", February 2020, <https://tools.ietf.org/html/draft-vpolak-mkonstan-bmwg-mlrsearch>.

[FDio-CSIT-PLRsearch]
"FD.io CSIT Test Methodology - PLRsearch", February 2020, <https://docs.fd.io/csit/rls2001/report/introduction/methodology_data_plane_throughput/methodology_plrsearch.html>.

Authors' Addresses

Maciek Konstantynowicz (editor)
Cisco Systems

Email: mkonstan@cisco.com

Vratko Polak (editor)
Cisco Systems

Email: vrpolak@cisco.com