Workgroup: Benchmarking Working Group
Internet-Draft: draft-ietf-bmwg-mlrsearch-03
Published: 9 November 2022
Intended Status: Informational
Expires: 13 May 2023
A M. Konstantynowicz V. Polak
uCisco Systems Cisco Systems
t
h
o
r
s
:

Multiple Loss Ratio Search

Abstract

This document proposes improvements to [RFC2544] throughput search by defining a new methodology called Multiple Loss Ratio search (MLRsearch). The main objectives for MLRsearch are to minimize the total test duration, search for multiple loss ratios and improve results repeatibility and comparability.

The main motivation behind MLRsearch is the new set of challenges and requirements posed by testing Network Function Virtualization (NFV) systems and other software based network data planes.

MLRsearch offers several ways to address these challenges, giving user configuration options to select their way.

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 13 May 2023.

Table of Contents

1. Purpose and Scope
2. Problems
  2.1. Long Test Duration
  2.2. DUT within SUT
  2.3. Repeatability and Comparability
  2.4. Throughput with Non-Zero Loss
  2.5. Inconsistent Trial Results
3. MLRsearch Approach
  3.1. Terminology
  3.2. Description
  3.3. Enhancement: Multiple trials per load
4. How the problems are addressed
5. IANA Considerations
6. Security Considerations
7. Acknowledgements
8. References
  8.1. Normative References
  8.2. Informative References
Authors' Addresses

1. Purpose and Scope

The purpose of this document is to describe Multiple Loss Ratio search (MLRsearch), a throughput search methodology optimized for software DUTs.

Applying vanilla [RFC2544] throughput bisection to software DUTs results in a number of problems:

* Binary search takes too long as most of trials are done far from the eventually found throughput.

* The required final trial duration and pauses between trials also prolong the overall search duration.

* Software DUTs show noisy trial results (noisy neighbor problem), leading to big spread of possible discovered throughput values.

* Throughput requires loss of exactly zero packets, but the industry frequently allows for small but non-zero losses.

* The definition of throughput is not clear when trial results are inconsistent.

MLRsearch aims to address these problems by applying the following set of enhancements:

- Allow searching with multiple loss ratio goals.
  - Each trial result can affect any search goal in principle (trial reuse).
- Multiple phases within one loss ratio goal search, middle ones need to spend less time on trials.
  - Middle phases also aim at lesser precision.
  - Use Forwarding Rate (FR) at maximum offered load [RFC2285] (section 3.6.2) to initialize the first middle phase.
- Take care when dealing with inconsistent trial results.
  - Loss ratios goals are handled in an order that precludes any interference from later trials to earlier goals.
- Apply several load selection heuristics to save even more time by trying hard to avoid unnecessarily narrow intervals.

MLRsearch configuration options are flexible enough to support both conservative settings (unconditionally compliant with [RFC2544], but longer search duration and worse repeatability) and aggressive settings (shorter search duration and better repeatability but not compliant with [RFC2544]).

No part of [RFC2544] is intended to be obsoleted by this document.

2. Problems

2.1. Long Test Duration

Emergence of software DUTs, with frequent software updates and a number of different packet processing modes and configurations, drives the requirement of continuous test execution and bringing down the test execution time.

In the context of characterising particular DUT's network performance, this calls for improving the time efficiency of throughput search. A vanilla bisection (at 60sec trial duration for unconditional [RFC2544] compliance) is slow, because most trials spend time quite far from the eventual throughput.

[RFC2544] does not specify any stopping condition for throughput search, so users can trade-off between search duration and precision goal. But, due to exponential behavior of bisection, small improvement in search duration needs relatively big sacrifice in the result precision.

2.2. DUT within SUT

[RFC2285] defines: - *DUT* as - The network forwarding device to which stimulus is offered and response measured [RFC2285] (section 3.1.1). - *SUT* as - The collective set of network devices to which stimulus

is offered as a single entity and response measured [RFC2285] (section 3.1.2).

[RFC2544] specifies a test setup with an external tester stimulating the networking system, treating it either as a single DUT, or as a system of devices, an SUT.

In case of software networking, the SUT consists of a software program processing packets (device of interest, the DUT), running on a server hardware and using operating system functions as appropriate, with server hardware resources shared across all programs and the operating system.

DUT is effectively "nested" within SUT.

Due to a shared multi-tenant nature of SUT, DUT is subject to interference (noise) coming from the operating system and any other software running on the same server. Some sources of noise can be eliminated (e.g. by pinning DUT program threads to specific CPU cores and isolating those cores to avoid context switching). But some noise remains after all such reasonable precautions are applied. This noise does negatively affect DUT's network performance. We refer to it as an *SUT noise*.

DUT can also exhibit fluctuating performance itself, e.g. while performing some "stop the world" internal stateful processing. In many cases this may be an expected per-design behavior, as it would be observable even in a hypothetical scenario where all sources of SUT noise are eliminated. Such behavior affects trial results in a way similar to SUT noise. We use *noise* as a shorthand covering both *DUT fluctuations* and genuine SUT noise.

A simple model of SUT performance consists of a baseline *noiseless performance*, and an additional noise. The baseline is assumed to be constant (enough). The noise varies in time, sometimes wildly. The noise can sometimes be negligible, but frequently it lowers the observed SUT performance in a trial.

In this model, SUT does not have a single performance value, it has a spectrum. One end of the spectrum is the noiseless baseline, the other end is a *noiseful performance*. In practice, trial results close to the noiseful end of the spectrum happen only rarely. The worse performance, the more rarely it is seen.

Focusing on DUT, the benchmarking effort should aim at eliminating only the SUT noise from SUT measurement. But that is not really possible, as there are no realistic enough models able to distinguish SUT noise from DUT fluctuations.

However, assuming that a well-constructed SUT has the DUT as its performance bottleneck, the "DUT noiseless performance" can be defined as the noiseless end of SUT performance spectrum. (At least for throughput. For other quantities such as latency there will be an additive difference.) By this definition, DUT noiseless performance also minimizes the impact of DUT fluctuations.

In this document, we reduce the "DUT within SUT" problem to estimating the noiseless end of SUT performance spectrum from a limited number of trial results.

Any improvements to throughput search algorithm, aimed for better dealing with software networking SUT and DUT setup, should employ strategies recognizing the presence of SUT noise, and allow discovery of (proxies for) DUT noiseless performance at different levels of sensitivity to SUT noise.

2.3. Repeatability and Comparability

[RFC2544] does not suggest to repeat throughput search, and from just one throughput value, it cannot be determined how repeatable that value is. In practice, poor repeatability is also the main cause of poor comparability, e.g. different benchmarking teams can test the same DUT but get different throughput values.

[RFC2544] throughput requirements (60s trial, no tolerance to single frame loss) force the search to converge around the noiseful end of SUT performance spectrum. As that end is affected by rare trials of significantly low performance, the resulting throughput repeatability is poor.

The repeatability problem is the problem of defining a search procedure which reports more stable results (even if they can no longer be called "throughput" in [RFC2544] sense). According to baseline (noiseless) and noiseful model, better repeatability will be at the noiseless end of the spectrum. Therefore, solutions to the "DUT within SUT" problem will help also with the repeatability problem.

Conversely, any alteration to [RFC2544] throughput search that improves repeatability should be considered as less dependent on the SUT noise.

An alternative option is to simply run a search multiple times, and report some statistics (e.g. average and standard deviation). This can be used for "important" tests, but it makes the search duration problem even bigger.

2.4. Throughput with Non-Zero Loss

[RFC1242] (section 3.17) defines throughput as: The maximum rate at which none of the offered frames are dropped by the device.

and then it says: Since even the loss of one frame in a data stream can cause significant delays while waiting for the higher level protocols to time out, it is useful to know the actual maximum data rate that the device can support.

Contrary to that, many benchmarking teams settle with non-zero (small) loss ratio as the goal for a "throughput rate".

Motivations are many: modern protocols tolerate frame loss better; trials nowadays send way more frames within the same duration; impact of rare noise bursts is smaller as the baseline performance can compensate somewhat by keeping the loss ratio below the goal; if

SUT noise with "ideal DUT" is known, it can be set as the loss ratio goal.

Regardless of validity of any and all similar motivations, support for non-zero loss goals makes any search algorithm more user-friendly. [RFC2544] throughput is not friendly in this regard.

Searching for multiple loss ratio goals also helps to describe the SUT performance better than a single goal result. Repeated wide gap between zero and non-zero loss loads indicates the noise has a large impact on the overall SUT performance.

It is easy to modify the vanilla bisection to find a lower bound for intended load that satisfies a non-zero-loss goal, but it is not that obvious how to search for multiple goals at once, hence the support for multiple loss goals remains a problem.

2.5. Inconsistent Trial Results

While performing throughput search by executing a sequence of measurement trials, there is a risk of encountering inconsistencies between trial results.

The plain bisection never encounters inconsistent trials. But [RFC2544] hints about possibility if inconsistent trial results in two places. The first place is section 24 where full trial durations are required, presumably because they can be inconsistent with results from shorter trial durations. The second place is section 26.3 where two successive zero-loss trials are recommended, presumably because after one zero-loss trial there can be subsequent inconsistent non-zero-loss trial.

Examples include:

* a trial at the same load (same or different trial duration) results in a different packet loss ratio.

* a trial at higher load (same or different trial duration) results in a smaller packet loss ratio.

Any robust throughput search algorithm needs to decide how to continue the search in presence of such inconsistencies. Definitions of throughput in [RFC1242] and [RFC2544] are not specific enough to imply a unique way of handling such inconsistencies.

Ideally, there will be a definition of a quantity which both generalizes throughput for non-zero-loss (and other possible repeatibility enhancements), while being precise enough to force a specific way to resolve trial inconsistencies. But until such definition is agreed upon, the correct way to handle inconsistent trial results remains an open problem.

3. MLRsearch Approach

The following description intentionally leaves out some important implementation details. This is both to hide complexity that is not important for overall understanding, and to allow future improvements in the implementation.

3.1. Terminology

*_trial duration_: Amount of time over which frames are transmitted towards SUT and DUT in a single measurement step.

  -**MLRsearch input parameter** for final MLRsearch measurements.

*_loss ratio_: Ratio of the count of frames lost to the count of frames transmitted over a trial duration, a.k.a. packet loss ratio. Related to packet loss rate [RFC1242] (section 3.6). In MLRsearch loss ratio can mean either a trial result or a goal:

  -_trial loss ratio_: Loss ratio measured during a trial.

  -_loss ratio goal_: **MLRsearch input parameter**.

    oIf _trial loss ratio_ is smaller or equal to this, the trial **satisfies** the loss ratio goal.

*_load_: Constant offered load stimulating the SUT and DUT. Consistent with offered load [RFC2285] (section 3.5.2).

  -MLRsearch works with intended load instead, as it cannot deal with situations where the offered load is considerably different than intended load.

*_throughput_: The maximum load at which none of the offered frames are dropped by the SUT and DUT. Consistent with [RFC1242] (section 3.17).

*_conditional throughput_: The forwarding rate measured at the maximum load at which a list of specified conditions are met i.e. loss ratio goal and trial duration.

  -Throughput is then a special case of conditional throughput for zero loss ratio goal and long enough trial duration.

  -Conditional throughput is aligned with forwarding rate (FR) [RFC2285] (section 3.6.1), adding trial duration to offered load required when reporting FR.

*_lower bound_: One of values tracked by MLRsearch during the search runtime. It is specific to the current trial duration and current loss ratio goal. It represents a load value with at least one trial result available. If the trial satisfies the current loss ratio goal, it is a _valid_ bound (else _invalid_).

*_upper bound_: One of values tracked by MLRsearch during the search runtime. It is specific to the current trial duration and current loss ratio goal. It represents a load value with at least one trial result available. If the trial satisfies the current loss ratio goal, it is an _invalid_ bound (else _valid_).

*_interval_: The span between lower and upper bound loads.

*_precision goal_: **MLRsearch input parameter**, acting as a search stop condition, given as either absolute or relative width goal. An interval meets precision goal if:

- The difference of upper and lower bound loads (in pps) is not more than the absolute width goal.

- The difference as above, divided by upper bound load (in pps) is not more than the relative width goal.

3.2. Description

The MLRsearch approach to address the identified problems is based on the following main strategies:

*MLRsearch main inputs include the following search goals and parameters:

- One or more **loss ratio goals**.

  oe.g. a zero-loss goal and one (or more) non-zero-loss goals.

- **Target trial duration** condition governing required trial duration for final measurements.

- **Target precision** condition governing how close final lower and upper bound load values must be to each other for final measurements.

*Search is executed as a sequence of phases:

- *Initial phase* initializes bounds for the first middle phase.

- *Middle phase*s narrow down the bounds, using shorter trial durations and lower precision goals. Several middle phases can precede each final phase.

- *Final phase* (one per loss ratio goal) finds bounds matching input goals and parameters to serve as the overal search output.

*Each search phase produces its *ending* upper bound and lower bound:

- Initial phase may produce invalid bounds.

- Middle and final phases produce valid bounds.

- Middle or final phases needs at least two values to act as *starting* bounds (may be invalid).

- Each phase may perform several trial measurements, until phase's ending conditions are all met.

- Trial results from previous phases may be re-used.

* Initial phase establishes the starting values for bounds, using forwarding rates (FR) [RFC2285] (section 3.6.1) from a few trials of minimal duration, as follows:
  - 1st trial is done at *maximum offered load (MOL)* [RFC2285] (section 3.5.3), resulting in Forwarding rate at maximum offered load (FRMOL) [RFC2285] (section 3.6.2).
  - 2nd trial is done at *FRMOL*, resulting in forwarding rate at FRMOL (FRFRMOL), newly defined here.
  - 3rd trial is done at *FRFRMOL*, so its results are available for the next phase.
  - By default, FRMOL is used as an upper bound, FRFRMOL as a lower bound.
    - Adjustments may apply here for some cases e.g. when 2nd trial got zero loss or if FRFRMOL is too close to FRMOL.

* Middle phases are producing ending bounds by improving upon starting bounds:
  - Each middle phase uses the same loss ratio goal as the final phase it precedes.
    - Called *current loss ratio goal* for upper and lower bound purposes.
  - Each middle phase has its own *current trial duration* and *current precision goal* parameters, computed from MLRsearch input parameters. As phases progress, these parameters approach MLRsearch main input values.
    - Current trial duration starts from a configurable minimum (e.g. 1 sec) and increases in a geometric sequence.
    - Current precision goal always allows twice as wide intervals as the following phase.
  - The starting bounds are usually the ending bounds from the preceding phase.
    - Unless there are many previous trial results that are more promising.
  - Each middle phase operates in a sequence of four actions:
    1. Perform trial at the load between the starting bounds. - Depending on the trial result this becomes the first new valid upper or lower bound for current phase.
    2. Re-measure at the remaining starting lower or upper (respectively) bound.

3. If that did not result in a valid bound, start an *external search*. - That is a variant of exponential search.

   - The "growth" is given by input parameter *expansion_coefficient*. - This action ends when a new valid bound is found.

   - Or if an already existing valid bound becomes close enough.

4. Repeatedly bisect the current interval until the bounds are close enough.

* Final search phase operates in exactly the same way as middle phases. There are two reasons why it is named differently:

  - The current trial duration and current precision goal within the phase are equal to the target trial duration and target precision input parameters.

  - The forwarding rates of the ending bounds become the output of MLRsearch.

    - Specifically, the forwarding rates of the final lower bounds are the conditional throughput values per given loss ratio goals.

3.3. Enhancement: Multiple trials per load

An enhancement of MLRsearch is to introduce a *noise tolerance* input parameter. The idea is to perform several medium-length trials (instead of a single long trial) and tolerate a configurable fraction of them to not-satisfy the loss ratio goal.

MLRsearch implementation with this enhancement exists in FD.io CSIT project and test results of VPP and DPDK (testpmd, l3fwd) DUTs look promising.

This enhancement would make the description of MLRsearch approach considerably more complicated, so this document version only describes MLRsearch without this enhancement.

4. How the problems are addressed

Configurable loss ratio goals are in direct support for non-zero-loss conditional througput. In practice the conditional throughput results' stability increases with higher loss ratio goals.

Multiple trials with noise tolerance enhancement will also indirectly increase result stability and it will allow MLRsearch to add all the benefits of Binary Search with Loss Verification, as recommended in [RFC9004] (section 6.2) and specified in [TST009] (section 12.3.3).

The main factor improving the overall search time is the introduction of middle phases. The full implementation can bring a

large number of heuristics related to how exactly should the next trial load be chosen, but the impact of those is not as big.

The Description subsection lacks any details on how to handle inconsistent trial results. In practice, there tend to be a three-way trade-off between i) short overall search time, ii) result stability and iii) how simple the definition of the returned conditional throughput can be. The third one is important for comparability between different MLRsearch implementations.

5. IANA Considerations

No requests of IANA.

6. 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.

7. Acknowledgements

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

8. References

8.1. 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>.

[RFC2285] Mandeville, R., "Benchmarking Terminology for LAN Switching Devices", RFC 2285, DOI 10.17487/RFC2285, February 1998, <https://www.rfc-editor.org/info/rfc2285>.

[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>.

[RFC9004] Morton, A., "Updates for the Back-to-Back Frame Benchmark in RFC 2544", RFC 9004, DOI 10.17487/RFC9004, May 2021, <https://www.rfc-editor.org/info/rfc9004>.

8.2. Informative References

[FDio-CSIT-MLRsearch] "FD.io CSIT Test Methodology - MLRsearch", November 2021, <https://s3-docs.fd.io/csit/rls2110/report/introduction/methodology_data_plane_throughput/methodology_data_plane_throughput.html#mlrsearch-tests>.

[PyPI-MLRsearch] "MLRsearch 0.3.0, Python Package Index", April 2021, <https://pypi.org/project/MLRsearch/0.3.0/>.

[TST009] "TST 009", n.d., <https://www.etsi.org/deliver/etsi_gs/NFV-TST/001_099/009/03.04.01_60/gs_NFV-TST009v030401p.pdf>.

Authors' Addresses

Maciek Konstantynowicz
Cisco Systems

Email: mkonstan@cisco.com

Vratko Polak
Cisco Systems

Email: vrpolak@cisco.com