

Benchmarking Working Group  
Internet-Draft  
Intended status: Informational  
Expires: 8 September 2022

M. Konstantynowicz, Ed.  
V. Polak  
Cisco Systems  
7 March 2022

Multiple Loss Ratio Search for Packet Throughput (MLRsearch)  
draft-ietf-bmwg-mlrsearch-02

## Abstract

TOD0: Update after all sections are ready.

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-Draft

Multiple Loss Ratio Search

March 2022

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 8 September 2022.

## Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Revised BSD License.

## Table of Contents

<a href="#">1.</a>	Terminology . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Intentions of this document . . . . .	<a href="#">5</a>
<a href="#">3.</a>	<a href="#">RFC2544</a> . . . . .	<a href="#">5</a>
<a href="#">3.1.</a>	Throughput search . . . . .	<a href="#">5</a>
<a href="#">4.</a>	Problems . . . . .	<a href="#">6</a>
<a href="#">4.1.</a>	Repeatability and Comparability . . . . .	<a href="#">6</a>
<a href="#">4.2.</a>	Non-Zero Target Loss Ratios . . . . .	<a href="#">6</a>
<a href="#">5.</a>	Solution ideas . . . . .	<a href="#">7</a>
<a href="#">5.1.</a>	Short duration trials . . . . .	<a href="#">8</a>
<a href="#">5.2.</a>	FRMOL as reasonable start . . . . .	<a href="#">8</a>
<a href="#">5.3.</a>	Non-zero loss ratios . . . . .	<a href="#">8</a>
<a href="#">5.4.</a>	Concurrent ratio search . . . . .	<a href="#">9</a>
<a href="#">5.5.</a>	Load selection heuristics and shortcuts . . . . .	<a href="#">9</a>
<a href="#">6.</a>	Non-compliance with <a href="#">RFC2544</a> . . . . .	<a href="#">9</a>
<a href="#">7.</a>	Additional Requirements . . . . .	<a href="#">10</a>
<a href="#">7.1.</a>	TOD0: Search Stop Criteria . . . . .	<a href="#">10</a>
<a href="#">7.2.</a>	Reliability of Test Equipment . . . . .	<a href="#">10</a>
<a href="#">7.2.1.</a>	Very late frames . . . . .	<a href="#">10</a>
<a href="#">8.</a>	MLRsearch Background . . . . .	<a href="#">11</a>
<a href="#">9.</a>	MLRsearch Overview . . . . .	<a href="#">13</a>

<a href="#">10.</a>	Sample Implementation . . . . .	<a href="#">16</a>
<a href="#">10.1.</a>	Input Parameters . . . . .	<a href="#">16</a>
<a href="#">10.2.</a>	Initial Phase . . . . .	<a href="#">17</a>
<a href="#">10.3.</a>	Non-Initial Phases . . . . .	<a href="#">18</a>
<a href="#">11.</a>	FD.io CSIT Implementation . . . . .	<a href="#">22</a>

<a href="#">11.1.</a>	Additional details . . . . .	<a href="#">22</a>
<a href="#">11.1.1.</a>	FD.io CSIT Input Parameters . . . . .	<a href="#">24</a>
<a href="#">11.2.</a>	Example MLRsearch Run . . . . .	<a href="#">25</a>
<a href="#">12.</a>	IANA Considerations . . . . .	<a href="#">27</a>
<a href="#">13.</a>	Security Considerations . . . . .	<a href="#">27</a>
<a href="#">14.</a>	Acknowledgements . . . . .	<a href="#">27</a>
<a href="#">15.</a>	References . . . . .	<a href="#">27</a>
<a href="#">15.1.</a>	Normative References . . . . .	<a href="#">27</a>
<a href="#">15.2.</a>	Informative References . . . . .	<a href="#">28</a>
	Authors' Addresses . . . . .	<a href="#">28</a>

## [1.](#) Terminology

TODO: Update after most other sections are updated.

- \* TODO: The current text uses Throughput for the zero loss ratio load. Is the capital T needed/useful?
- \* DUT and SUT: see the definitions in <https://gerrit.fd.io/r/c/csit/+/35545>
- \* Traffic Generator (TG) and Traffic Analyzer (TA): see <https://datatracker.ietf.org/doc/html/rfc6894#section-4> TODO: Maybe there is an earlier RFC?
- \* Overall search time: the time it takes to find all required loads within their precision goals, starting from zero trials measured at given DUT configuration and traffic profile.
- \* TODO: traffic profile?
- \* Intended load: <https://datatracker.ietf.org/doc/html/rfc2285#section-3.5.1>
- \* Offered load: <https://datatracker.ietf.org/doc/html/rfc2285#section-3.5.2>

- \* Maximum offered load (MOL): see <https://datatracker.ietf.org/doc/html/rfc2285#section-3.5.3>
- \* Forwarding rate at maximum offered load (FRMOL) <https://datatracker.ietf.org/doc/html/rfc2285#section-3.6.2>
- \* Trial Loss Count: the number of frames transmitted minus the number of frames received. Negative count is possible, e.g. when SUT duplicates some frames.

- \* Trial Loss Ratio: ratio of frames received relative to frames transmitted over the trial duration. For bi-directional throughput tests, the aggregate ratio is calculated, based on the aggregate number of frames transmitted and received. If the trial loss count is negative, its absolute value MUST be used to keep compliance with [RFC2544](https://datatracker.ietf.org/doc/html/rfc2544).
- \* Safe load: any value, such that trial measurement at this (or lower) intended load is correctly handled by both TG and TA, regardless of SUT behavior. Frequently, it is not known what the safe load is.
- \* Max load (TODO rename?): Maximal intended load to be used during search. Benchmarking team decides which value is low enough to guarantee values reported by TG and TA are reliable. It has to be a safe load, but it can be lower than a safe load estimate for added safety. See the subsection on unreliable test equipment below. This value MUST NOT be higher than MOL, which itself MUST NOT be higher than Maximum Frame Rate <https://datatracker.ietf.org/doc/html/rfc2544#section-20>
- \* Min load: Minimal intended load to be used during search. Benchmarking team decides which value is high enough to guarantee the trial measurement results are valid. E.g. considerable overall search time can be saved by declaring SUT faulty if min load trial shows too high loss rate. Zero frames per second is a valid min load value
- \* Effective loss ratio: a corrected value of trial loss ratio chosen

to avoid difficulties if SUT exhibits decreasing loss ratio with increasing load. It is the maximum of trial loss ratios measured at the same duration on all loads smaller than (and including) the current one.

- \* Target loss ratio: a loss ratio value acting as an input for the search. The search is finding tight enough lower and upper bounds in intended load, so that the measurement at the lower bound has smaller or equal trial loss ratio, and upper bound has strictly larger trial loss ratio. For the tightest upper bound, the effective loss ratio is the same as trial loss ratio at that upper bound load. For the tightest lower bound, the effective loss ratio can be higher than the trial loss ratio at that lower bound, but still not larger than the target loss ratio.
- \* TODO: Search algorithm.
- \* TODO: Precision goal.

- \* TODO: Define a "benchmarking group".
- \* TODO: Upper and lower bound.
- \* TODO: Valid and invalid bound?
- \* TODO: Interval and interval width?

TODO: Mention NIC/PCI bandwidth/pps limits can be lower than bandwidth of medium.

## [2.](#) Intentions of this document

The intention of this document is to provide recommendations for: \* optimizing search for multiple target loss ratios at once, \* speeding up the overall search time, \* improve search results repeatability and comparability.

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

## [3.](#) [RFC2544](#)

### [3.1.](#) Throughput search

It is useful to restate the key requirements of [RFC2544](#) using the new terminology (see section Terminology).

The following sections of [RFC2544](#) are of interest for this document.

- \* <https://datatracker.ietf.org/doc/html/rfc2544#section-20> Mentions the max load SHOULD not be larger than the theoretical maximum rate for the frame size on the media.
- \* <https://datatracker.ietf.org/doc/html/rfc2544#section-23> Lists the actions to be done for each trial measurement, it also mentions loss rate as an example of trial measurement results. This document uses loss count instead, as that is the quantity that is easier for the current test equipment to measure, e.g. it is not affected by the real traffic duration. TODO: Time uncertainty again.
- \* <https://datatracker.ietf.org/doc/html/rfc2544#section-24> Mentions "full length trials" leading to the Throughput found, as opposed to shorter trial durations, allowed in an attempt to "minimize the length of search procedure". This document talks about "final trial duration" and aims to "optimize overall search time".

- \* <https://datatracker.ietf.org/doc/html/rfc2544#section-26.1> with <https://www.rfc-editor.org/errata/eid422> finally states requirements for the search procedure. It boils down to "increase intended load upon zero trial loss and decrease intended load upon non-zero trial loss".

No additional constraints are placed on the load selection, and there is no mention of an exit condition, e.g. when there is enough trial measurements to proclaim the largest load with zero trial loss (and final trial duration) to be the Throughput found.

## [4.](#) Problems

### [4.1.](#) 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 (how likely it is for a repeated Throughput search to end up with a value less than the precision goal away from the first value).

Depending on SUT behavior, different benchmark groups can report significantly different Throughput values, even when using identical SUT and test equipment, just because of minor differences in their search algorithm (e.g. different max load value).

While repeatability can be addressed by repeating the search several times, the differences in the comparability scenario may be systematic, e.g. seeming like a bias in one or both benchmark groups.

MLRsearch algorithm does not really help with the repeatability problem. This document RECOMMENDS to repeat a selection of "important" tests ten times, so users can ascertain the repeatability of the results.

TODO: How to report? Average and standard deviation?

Following MLRsearch algorithm leaves less freedom for the benchmark groups to encounter the comparability problem, although more research is needed to determine the effect of MLRsearch's tweakable parameters.

#### [4.2.](#) Non-Zero Target Loss Ratios

<https://datatracker.ietf.org/doc/html/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.

New "software DUTs" (traffic forwarding programs running on commercial-off-the-shelf compute server hardware) frequently exhibit quite low repeatability of Throughput results per above definition.

This is due to, in general, throughput rates of software DUTs (programs) being sensitive to server resource allocation by OS during runtime, as well as any interrupts or blocking of software threads involved in packet processing.

To deal with this, this document recommends discovery of multiple throughput rates of interest for software DUTs that run on general purpose COTS servers (with x86, AArch64 Instruction Set Architectures): \* throughput rate with target of zero packet loss ratio. \* at least one throughput rate with target of non-zero packet loss ratio.

In our experience, the higher the target loss ratio is, the better is the repeatability of the corresponding load found.

TOD0: Define a good name for a load corresponding to a specific non-zero target loss ratio, while keeping Throughput for the load corresponding to zero target loss ratio.

This document RECOMMENDS the benchmark groups to search for corresponding loads to at least one non-zero target loss ratio. This document does not suggest any particular non-zero target loss ratio value to search the corresponding load for.

## [5.](#) Solution ideas

This document gives several independent ideas on how to lower the (average) overall search time, while remaining unconditionally compliant with [RFC2544](#) (and adding some of extensions).

This document also specifies one particular way to combine all the ideas into a single search algorithm class (single logic with few tweakable parameters).

Little to no research has been done into the question of which combination of ideas achieves the best compromise with respect to overall search time, high repeatability and high comparability.

TOD0: How important it is to discuss particular implementation



choices, especially when motivated by non-deterministic SUT behavior?

### 5.1. Short duration trials

<https://datatracker.ietf.org/doc/html/rfc2544#section-24> already mentions the possibility of using shorter duration for trials that are not part of "final determination".

Obviously, the upper and lower bound from a smaller duration trial can be used as the initial upper and lower bound for the final determination.

MLRsearch makes it clear a re-measurement is always needed (new trial measurement with the same load but longer duration). It also specifies what to do if the longer trial is no longer a valid bound (TODO define?), e.g. start an external search. Additionally one halving can be saved during the shorter duration search.

### 5.2. FRMOL as reasonable start

TODO expand: Overall search ends with "final determination" search, preceded by "shorter duration search" preceded by "bound initialization", where the bounds can be considerably different from min and max load.

For SUTs with high repeatability, the FRMOL is usually a good approximation of Throughput. But for less repeatable SUTs, forwarding rate (TODO define) is frequently a bad approximation to Throughput, therefore halving and other robust-to-worst-case approaches have to be used. Still, forwarding rate at FRMOL load can be a good initial bound.

### 5.3. Non-zero loss ratios

See the "Popularity of non-zero target loss ratios" section above.

TODO: Define "trial measurement result classification criteria", or keep reusing long phrases without definitions?

A search for a load corresponding to a non-zero target loss rate is very similar to a search for Throughput, just the criterion when to increase or decrease the intended load for the next trial measurement uses the comparison of trial loss ratio to the target loss ratio (instead of comparing loss count to zero) Any search algorithm that works for Throughput can be easily used also for non-zero target loss rates, perhaps with small modifications in places where the measured forwarding rate is used.

Note that it is possible to search for multiple loss ratio goals if needed.

#### [5.4.](#) Concurrent ratio search

A single trial measurement result can act as an upper bound for a lower target loss ratio, and as a lower bound for a higher target loss ratio at the same time. This is an example of how it can be advantageous to search for all loss ratio goals "at once", or at least "reuse" trial measurement result done so far.

Even when a search algorithm is fully deterministic in load selection while focusing on a single loss ratio and trial duration, the choice of iteration order between target loss ratios and trial durations can affect the obtained results in subtle ways. MLRsearch offers one particular ordering.

#### [5.5.](#) Load selection heuristics and shortcuts

Aside of the two heuristics already mentioned (FRMOL based initial bounds and saving one halving when increasing trial duration), there are other tricks that can save some overall search time at the cost of keeping the difference between final lower and upper bound intentionally large (but still within the precision goal).

TODO: Refer implementation subsections on: \* Uneven splits. \* Rounding the interval width up. \* Using old invalid bounds for interval width guessing.

The impact on overall duration is probably small, and the effect on result distribution maybe even smaller. TODO: Is the two-liner above useful at all?

### [6.](#) Non-compliance with [RFC2544](#)

It is possible to achieve even faster search times by abandoning some requirements and suggestions of [RFC2544](#), mainly by reducing the wait times at start and end of trial.

Such results are therefore no longer compliant with [RFC2544](#) (or at least not unconditionally), but they may still be useful for internal usage, or for comparing results of different DUTs achieved with an identical non-compliant algorithm.

TODO: Refer to the subsection with CSIT customizations.

Internet-Draft

Multiple Loss Ratio Search

March 2022

## [7.](#) Additional Requirements

[RFC2544](#) can be understood as having a number of implicit requirements. They are made explicit in this section (as requirements for this document, not for [RFC2544](#)).

Recommendations on how to properly address the implicit requirements are out of scope of this document.

### [7.1.](#) TODO: Search Stop Criteria

TODO: Mention the timeout parameter?

### [7.2.](#) Reliability of Test Equipment

Both TG and TA MUST be able to handle correctly every intended load used during the search.

On TG side, the difference between Intended Load and Offered Load MUST be small.

TODO: How small? Difference of one packet may not be measurable due to time uncertainties.

TODO expand: time uncertainty.

To ensure that, max load (see Terminology) has to be set to low enough value. Benchmark groups MAY list the max load value used, especially if the Throughput value is equal (or close) to the max load.

Solutions (even problem formulations) for the following open problems are outside of the scope of this document: \* Detecting when the test equipment operates above its safe load. \* Finding a large but safe load value. \* Correcting any result affected by max load value not being a safe load.

#### [7.2.1.](#) Very late frames

[RFC2544](https://datatracker.ietf.org/doc/html/rfc2544) requires quite conservative time delays see <https://datatracker.ietf.org/doc/html/rfc2544#section-23> to prevent frames buffered in one trial measurement to be counted as received in a subsequent trial measurement.

However, for some SUTs it may still be possible to buffer enough frames, so they are still sending them (perhaps in bursts) when the next trial measurement starts. Sometimes, this can be detected as a negative trial loss count, e.g. TA receiving more frames than TG has sent during this trial measurement. Frame duplication is another way of causing the negative trial loss count.

<https://datatracker.ietf.org/doc/html/rfc2544#section-10> recommends to use sequence numbers in frame payloads, but generating and verifying them requires test equipment resources, which may be not plenty enough to suport at high loads. (Using low enough max load would work, but frequently that would be smaller than SUT's sctual Throughput.)

[RFC2544](https://datatracker.ietf.org/doc/html/rfc2544) does not offer any solution to the negative loss problem, except implicitly treating negative trial loss counts the same way as positive trial loss counts.

This document also does not offer any practical solution.

Instead, this document SUGGESTS the search algorithm to take any precaution necessary to avoid very late frames.

This document also REQUIRES any detected duplicate frames to be counted as additional lost frames. This document also REQUIRES, any negative trial loss ratio to be treated as positive trial loss ratio of the same absolute value.

!!! Nothing below is up-to-date with draft v02. !!!

## [8.](#) MLRsearch Background

TODO: Old section, probably obsoleted by preceding section(s).

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, each rate is associated with a distinct Packet Loss Ratio (PLR) criterion.

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 with NDR and PDR pair as an example.

Similarly to other throughput search approaches like binary search, MLRsearch is effective for SUTs/DUTs with PLR curve that is non-decreasing with growing offered load. It may not be as effective for SUTs/DUTs with abnormal PLR curves, although it will always converge to some value.

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:

- \* 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.
- \* SUT/DUT packet processing capacity is the same in both directions, resulting in the same packet loss under load.

MLRsearch can be applied even without those assumptions, but in that case the aggregate loss ratio is less useful as a metric.

MLRsearch can be used for network transactions consisting of more than just one packet, or anything else that has intended load as input and loss ratio as output (duration as input is optional). This text uses mostly packet-centric language.

## [9.](#) MLRsearch Overview

The main properties of MLRsearch:

- \* 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:

- o Target PLRs (e.g. 0.0 and 0.005 when searching for NDR and PDR).
  - o Final trial duration.
  - o Measurement resolution.
- \* Initial Phase:
- Measure MRR over initial trial duration.
  - Measured MRR is used as an input to the first intermediate phase.
- \* Multiple Intermediate Phases:
- Trial duration:
    - o Start with initial trial duration in the first intermediate phase.
    - o Converge geometrically towards the final trial duration.
  - Track all previous trial measurement results:
    - o Duration, offered load and loss ratio are tracked.
    - o Effective loss ratios are tracked.

- + While in practice, real loss ratios can decrease with increasing load, effective loss ratios never decrease. This is achieved by sorting results by load, and using the effective loss ratio of the previous load if the current loss ratio is smaller than that.
- o The algorithm queries the results to find best lower and upper bounds.
- + Effective loss ratios are always used.

- o The phase ends if all target loss ratios have tight enough bounds.
- Search:
  - o Iterate over target loss ratios in increasing order.
  - o If both upper and lower bound are in measurement results for this duration, apply bisection until the bounds are tight enough, and continue with next loss ratio.
  - o If a bound is missing for this duration, but there exists a bound from the previous duration (compatible with the other bound at this duration), re-measure at the current duration.
  - o If a bound in one direction (upper or lower) is missing for this duration, and the previous duration does not have a compatible bound, compute the current "interval size" from the second tightest bound in the other direction (lower or upper respectively) for the current duration, and choose next offered load for external search.
  - o The logic guarantees that a measurement is never repeated with both duration and offered load being the same.
  - o The logic guarantees that measurements for higher target loss ratio iterations (still within the same phase duration) do not affect validity and tightness of bounds for previous target loss ratio iterations (at the same duration).
- Use of internal and external searches:
  - o External search:
    - + It is a variant of "exponential search".

- + The "interval size" is multiplied by a configurable constant (powers of two work well with the subsequent internal search).



- o Internal search:
  - + A variant of binary search that measures at offered load between the previously found bounds.
  - + The interval does not need to be split into exact halves, if other split can get to the target width goal faster.
    - \* The idea is to avoid returning interval narrower than the current width goal. See sample implementation details, below.
- \* Final Phase:
  - Executed with the final test trial duration, and the final width goal that determines resolution of the overall search.
- \* Intermediate Phases together with the Final Phase are called Non-Initial Phases.
- \* The returned bounds stay within prescribed min\_rate and max\_rate.
  - When returning min\_rate or max\_rate, the returned bounds may be invalid.
    - o E.g. upper bound at max\_rate may come from a measurement with loss ratio still not higher than the target loss ratio.

The main benefits of MLRsearch vs. binary search include:

- \* In general, MLRsearch is likely to execute more trials overall, but likely less trials at a set final trial duration.
- \* 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.
- \* In all cases MLRsearch yields the same or similar results to binary search.
- \* Note: both binary search and MLRsearch are susceptible to reporting non-repeatable results across multiple runs for very bad behaving cases.

#### Caveats:

- \* Worst case MLRsearch can take longer than a binary search, e.g. in case of drastic changes in behaviour for trials at varying durations.
  - Re-measurement at higher duration can trigger a long external search. That never happens in binary search, which uses the final duration from the start.

### [10.](#) Sample Implementation

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

#### [10.1.](#) Input Parameters

1. *\*max\_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. *\*min\_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\_ratios\** - list of maximum acceptable PLR search criteria.
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.

Internet-Draft

Multiple Loss Ratio Search

March 2022

## [10.2.](#) 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.
  - \* D0: single trial.
  - \* OUT: measured loss ratio.
  - \* OUT: MRR = measured receive rate. Received rate is computed as intended load multiplied by pass ratio (which is one minus loss ratio). This is useful when loss ratio is computed from a different metric than intended load. For example, intended load can be in transactions (multiple packets each), but loss ratio is computed on level of packets, not transactions.
  - \* Example: If MTR is 10 transactions per second, and each transaction has 10 packets, and receive rate is 90 packets per second, then loss rate is 10%, and MRR is computed to be 9 transactions per second.

If MRR is too close to MTR, MRR is set below MTR so that interval width is equal to the width goal of the first intermediate phase. If MRR is less than min\_rate, min\_rate is used.

2. Second trial measures at MRR and discovers MRR2.
  - \* IN: trial\_duration = initial\_trial\_duration.
  - \* IN: offered\_transmit\_rate = MRR.
  - \* D0: single trial.
  - \* OUT: measured loss ratio.
  - \* OUT: MRR2 = measured receive rate. If MRR2 is less than

min\_rate, min\_rate is used. If loss ratio is less or equal to the smallest target loss ratio, MRR2 is set to a value 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.
- \* OUT: MRR3 = measured receive rate. If MRR3 is less than min\_rate, min\_rate is used. If step 3 is not skipped, the first trial measurement is forgotten. This is done because in practice (if MRR2 is above MRR), external search from MRR and MRR2 is likely to lead to a faster intermediate phase than a bisect between MRR2 and MTR.

### [10.3.](#) Non-Initial Phases

#### 1. Main phase 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: Measurement results from the previous phase (previous duration).
4. Internal target ratio loop:
  1. IN: Target loss ratio for this iteration of ratio loop.
  2. IN: Measurement results from all previous ratio loop iterations of current phase (current duration).
  3. DO: According to the procedure described in point 2:
    1. either exit the phase (by jumping to 1.5),

2. or exit loop iteration (by continuing with next target loss ratio, jumping to 1.4.1),
  3. or calculate new transmit rate to measure with.
4. DO: Perform the trial measurement at the new transmit rate and current trial duration, compute its loss ratio.
5. DO: Add the result and go to next iteration (1.4.1), including the added trial result in 1.4.2.
5. OUT: Measurement results from this phase.
6. OUT: In the final phase, bounds for each target loss ratio are extracted and returned.
  1. If a valid bound does not exist, use min\_rate or max\_rate.
2. New transmit rate (or exit) calculation (for point 1.4.3):
  1. If the previous duration has the best upper and lower bound, select the middle point as the new transmit rate.
    1. See 2.5.3. below for the exact splitting logic.

2. This can be a no-op if interval is narrow enough already, in that case continue with 2.2.
3. Discussion, assuming the middle point is selected and measured:
  1. Regardless of loss rate measured, the result becomes either best upper or best lower bound at current duration.
  2. So this condition is satisfied at most once per iteration.
  3. This also explains why previous phase has double width goal:
    1. We avoid one more bisection at previous phase.
    2. At most one bound (per iteration) is re-measured with current duration.

3. Each re-measurement can trigger an external search.
4. Such surprising external searches are the main hurdle in achieving low overall search durations.
5. Even without 1.1, there is at most one external search per phase and target loss ratio.
6. But without 1.1 there can be two re-measurements, each coming with a risk of triggering external search.
2. If the previous duration has one bound best, select its transmit rate. In deterministic case this is the last measurement needed this iteration.
3. If only upper bound exists in current duration results:

1. This can only happen for the smallest target loss ratio.
2. If the upper bound was measured at min\_rate, exit the whole phase early (not investigating other target loss ratios).
3. Select new transmit rate using external search:
  1. For computing previous interval size, use:
    1. second tightest bound at current duration,
    2. or tightest bound of previous duration, if compatible and giving a more narrow interval,
    3. or target interval width if none of the above is available.
    4. In any case increase to target interval width if smaller.
  2. Quadruple the interval width.
  3. Use min\_rate if the new transmit rate is lower.
4. If only lower bound exists in current duration results:
  1. If the lower bound was measured at max\_rate, exit this iteration (continue with next lowest target loss ratio).

2. Select new transmit rate using external search:
  1. For computing previous interval size, use:
    1. second tightest bound at current duration,
    2. or tightest bound of previous duration, if compatible and giving a more narrow interval,
    3. or target interval width if none of the above is available.

4. In any case increase to target interval width if smaller.
2. Quadruple the interval width.
3. Use max\_rate if the new transmit rate is higher.
5. The only remaining option is both bounds in current duration results.
  1. This can happen in two ways, depending on how the lower bound was chosen.
    1. It could have been selected for the current loss ratio, e.g. in re-measurement (2.2) or in initial bisect (2.1).
    2. It could have been found as an upper bound for the previous smaller target loss ratio, in which case it might be too low.
    3. The algorithm does not track which one is the case, as the decision logic works well regardless.
  2. Compute "extending down" candidate transmit rate exactly as in 2.3.
  3. Compute "bisecting" candidate transmit rate:
    1. Compute the current interval width from the two bounds.
    2. Express the width as a (float) multiple of the target width goal for this phase.

3. If the multiple is not higher than one, it means the width goal is met. Exit this iteration and continue with next higher target loss ratio.
4. If the multiple is two or less, use half of that for



new width if the lower subinterval.

5. Round the multiple up to nearest even integer.
  6. Use half of that for new width if the lower subinterval.
  7. Example: If lower bound is 2.0 and upper bound is 5.0, and width goal is 1.0, the new candidate transmit rate will be 4.0. This can save a measurement when 4.0 has small loss. Selecting the average (3.5) would never save a measurement, giving more narrow bounds instead.
4. If either candidate computation want to exit the iteration, do as bisecting candidate computation says.
  5. The remaining case is both candidates wanting to measure at some rate. Use the higher rate. This prefers external search down narrow enough interval, competing with perfectly sized lower bisect subinterval.

## [11.](#) 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](#)].

### [11.1.](#) 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. 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_`).

## 3. Intended count.

- \* The number of packets to send during the trial should be equal to the intended load multiplied by the duration.
  - Also multiplied by a coefficient, if loss ratio is calculated from a different metric.
    - o Example: If a successful transaction uses 10 packets, load is given in transactions per second, but loss ratio is calculated from packets, so the coefficient to get intended count of packets is 10.
- \* But in practice that does not work.
  - It could result in a fractional number of packets,
  - so it has to be rounded in a way traffic generator chooses,
  - which may depend on the number of traffic flows and traffic generator worker threads.

## 4. Attempted count. As the real number of intended packets is not known exactly, the computation uses the number of packets traffic generator reports as sent. Unless overridden by the next point.

## 5. Duration stretching.

- \* In some cases, traffic generator may get overloaded, causing it to take significantly longer (than duration) to send all packets.
- \* The implementation uses an explicit stop,

Internet-Draft

Multiple Loss Ratio Search

March 2022

- causing lower attempted count in those cases.
  - \* The implementation tolerates some small difference between attempted count and intended count.
    - 10 microseconds worth of traffic is sufficient for our tests.
  - \* If the difference is higher, the unsent packets are counted as lost.
    - This forces the search to avoid the regions of high duration stretching.
    - The final bounds describe the performance of not just SUT, but of the whole system, including the traffic generator.
6. Excess packets.
- \* In some test (e.g. using TCP flows) Traffic generator reacts to packet loss by retransmission. Usually, such packet loss is already affecting loss ratio. If a test also wants to treat retransmissions due to heavily delayed packets also as a failure, this is once again visible as a mismatch between the intended count and the attempted count.
  - \* The CSIT implementation simply looks at absolute value of the difference, so it offers the same small tolerance before it starts marking a "loss".
7. For result processing, we use lower bounds and ignore upper bounds.

#### [11.1.1.1.](#) FD.io CSIT Input Parameters

1. `*max_rate*` - Typical values: 2 \* 14.88 Mpps for 64B 10GE link rate, 2 \* 18.75 Mpps for 64B 40GE NIC (specific model).
2. `*min_rate*` - Value: 2 \* 9001 pps (we reserve 9000 pps for latency measurements).
3. `*final_trial_duration*` - Value: 30.0 seconds.

4. `*initial_trial_duration*` - Value: 1.0 second.
5. `*final_relative_width*` - Value: 0.005 (0.5%).

6. `*packet_loss_ratios*` - Value: 0.0, 0.005 (0.0% for NDR, 0.5% for PDR).
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: 600.0 (seconds).
9. `*expansion_coefficient*` - Width multiplier for external search. Value: 4.0 (interval width is quadrupled). Value of 2.0 is best for well-behaved SUTs, but value of 4.0 has been found to decrease overall search time for worse-behaved SUT configurations, contributing more to the overall set of different SUT configurations tested.

### [11.2.](#) Example MLRsearch Run

The following list describes a search from a real test run in CSIT (using the default input values as above).

\* Initial phase, trial duration 1.0 second.

Measurement 1, intended load 18750000.0 pps (MTR), measured loss ratio 0.7089514628479618 (valid upper bound for both NDR and PDR).

Measurement 2, intended load 5457160.071600716 pps (MRR), measured loss ratio 0.018650817320118702 (new tightest upper bounds).

Measurement 3, intended load 5348832.933500009 pps (slightly less than MRR2 in preparation for first intermediate phase target interval width), measured loss ratio 0.00964383362905351 (new tightest upper

bounds).

\* First intermediate phase starts, trial duration still 1.0 seconds.

Measurement 4, intended load 4936605.579021453 pps (no lower bound, performing external search downwards, for NDR), measured loss ratio 0.0 (valid lower bound for both NDR and PDR).

Measurement 5, intended load 5138587.208637197 pps (bisecting for NDR), measured loss ratio 0.0 (new tightest lower bounds).

Measurement 6, intended load 5242656.244044665 pps (bisecting), measured loss ratio 0.013523745379347257 (new tightest upper bounds).

\* Both intervals are narrow enough.

\* Second intermediate phase starts, trial duration 5.477225575051661 seconds.

Measurement 7, intended load 5190360.904111567 pps (initial bisect for NDR), measured loss ratio 0.0023533920869969953 (NDR upper bound, PDR lower bound).

Measurement 8, intended load 5138587.208637197 pps (re-measuring NDR lower bound), measured loss ratio 1.2080222912800403e-06 (new tightest NDR upper bound).

\* The two intervals have separate bounds from now on.

Measurement 9, intended load 4936605.381062318 pps (external NDR search down), measured loss ratio 0.0 (new valid NDR lower bound).

Measurement 10, intended load 5036583.888432355 pps (NDR bisect), measured loss ratio 0.0 (new tightest NDR lower bound).

Measurement 11, intended load 5087329.903232804 pps (NDR bisect), measured loss ratio 0.0 (new tightest NDR lower bound).

\* NDR interval is narrow enough, PDR interval not ready yet.

Measurement 12, intended load 5242656.244044665 pps (re-measuring PDR upper bound), measured loss ratio 0.0101174866190136 (still valid PDR

upper bound).

- \* Also PDR interval is narrow enough, with valid bounds for this duration.
- \* Final phase starts, trial duration 30.0 seconds.

Measurement 13, intended load 5112894.3238511775 pps (initial bisect for NDR), measured loss ratio 0.0 (new tightest NDR lower bound).

Measurement 14, intended load 5138587.208637197 (re-measuring NDR upper bound), measured loss ratio 2.030389804256833e-06 (still valid PDR upper bound).

- \* NDR interval is narrow enough, PDR interval not yet.

Measurement 15, intended load 5216443.04126728 pps (initial bisect for PDR), measured loss ratio 0.005620871287975237 (new tightest PDR upper bound).

Measurement 16, intended load 5190360.904111567 (re-measuring PDR lower bound), measured loss ratio 0.0027629971184465604 (still valid PDR lower bound).

- \* PDR interval is also narrow enough.
- \* Returning bounds:
- \* NDR\_LOWER = 5112894.3238511775 pps; NDR\_UPPER = 5138587.208637197 pps;
- \* PDR\_LOWER = 5190360.904111567 pps; PDR\_UPPER = 5216443.04126728 pps.

## [12.](#) IANA Considerations

No requests of IANA.

## [13.](#) 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.

## 14. Acknowledgements

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

## 15. References

### 15.1. 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>>.

### 15.2. 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](https://s3-docs.fd.io/csit/rls2110/report/introduction/methodology_data_plane_throughput/methodology_data_plane_throughput.html#mlrsearch-tests)>.

[PyPI-MLRsearch]

"MLRsearch 0.4.0, Python Package Index", April 2021,  
<<https://pypi.org/project/MLRsearch/0.4.0/>>.

#### Authors' Addresses

Maciek Konstantynowicz (editor)  
Cisco Systems  
Email: mkonstan@cisco.com

Vratko Polak  
Cisco Systems  
Email: vrpolak@cisco.com