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

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

February 09, 2021

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

Abstract

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

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

Status of This Memo

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

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

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

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

Table of Contents

1. Terminology

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

- Packet size: same as frame size, both terms used interchangeably.

- Device Under Test (DUT): In software networking, "device" denotes a specific piece of software tasked with packet processing. Such device is surrounded with other software components (such as operating system kernel). It is not possible to run devices without also running the other components, and hardware resources are shared between both. For purposes of testing, the whole set of hardware and software components is called "system under test" (SUT). As SUT is the part of the whole test setup performance of which can be measured by [RFC2544] methods, this document uses SUT instead of [RFC2544] DUT. Device under test (DUT) can be re-introduced when analysing test results using whitebox techniques, but this document sticks to blackbox testing.

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

- Bi-directional throughput tests: involve packets/frames flowing in both transmit and receive directions over every tested interface of SUT/DUT. Packet flow metrics are measured per direction, and can be reported as aggregate for both directions and/or separately for each measured direction. In most cases bi-directional tests use the same (symmetric) load in both directions.

- Uni-directional throughput tests: involve packets/frames flowing in only one direction, i.e. either transmit or receive direction, over every tested interface of SUT/DUT. Packet flow metrics are measured and are reported for measured direction.

- Packet Loss Ratio (PLR): ratio of packets received relative to packets transmitted over the test trial duration, calculated using formula: PLR = (pkts_transmitted - pkts_received) / pkts_transmitted. For bi-directional throughput tests aggregate PLR is calculated based on the aggregate number of packets transmitted and received.

- Packet Throughput Rate: maximum packet offered load DUT/SUT forwards within the specified Packet Loss Ratio (PLR). In many cases the rate depends on the frame size processed by DUT/SUT. Hence packet throughput rate MUST be quoted with specific frame size as received by DUT/SUT during the measurement. For bi-directional tests, packet throughput rate should be reported as aggregate for both directions. Measured in packets-per-second (pps) or frames-per-second (fps), equivalent metrics.

- Bandwidth Throughput Rate: a secondary metric calculated from packet throughput rate using formula: bw_rate = pkt_rate * (frame_size + L1_overhead) * 8, where L1_overhead for Ethernet includes preamble (8 Bytes) and inter-frame gap (12 Bytes). For bi-directional tests, bandwidth throughput rate should be reported as aggregate for both directions. Expressed in bits-per-second (bps).

- Non Drop Rate (NDR): maximum packet/bandwith throughput rate sustained by DUT/SUT at PLR equal zero (zero packet loss) specific to tested frame size(s). MUST be quoted with specific packet size as received by DUT/SUT during the measurement. Packet NDR measured in packets-per-second (or fps), bandwidth NDR expressed in bits-per-second (bps).

- Partial Drop Rate (PDR): maximum packet/bandwith throughput rate sustained by DUT/SUT at PLR greater than zero (non-zero packet loss) specific to tested frame size(s). MUST be quoted with specific packet size as received by DUT/SUT during the measurement. Packet PDR measured in packets-per-second (or fps), bandwidth PDR expressed in bits-per-second (bps).

- Maximum Receive Rate (MRR): packet/bandwidth rate regardless of PLR sustained by DUT/SUT under specified Maximum Transmit Rate (MTR) packet load offered by traffic generator. MUST be quoted with both specific packet size and MTR as received by DUT/SUT during the measurement. Packet MRR measured in packets-per-second (or fps), bandwidth MRR expressed in bits-per-second (bps).

- Trial: a single measurement step. See [RFC2544] section 23.

- Trial duration: amount of time over which packets are transmitted in a single measurement step.

2. MLRsearch Background

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

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

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

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

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

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

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

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

- 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.

3. 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:

      + PLRs associated with NDR and PDR.

      + Final trial duration.

      + 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:

      + Start with initial trial duration in the first intermediate phase.

      + Converge geometrically towards the final trial duration.

    * Track two values for NDR and two for PDR:

      + The values are called lower_bound and upper_bound.

      + Each value comes from a specific trial measurement:

        - Most recent for that transmit rate.

        - As such the value is associated with that measurement's duration and loss.

      + A bound can be valid or invalid:

        - Valid lower_bound must conform with PLR search criteria.

        - Valid upper_bound must not conform with PLR search criteria.

        - Example of invalid NDR lower_bound is if it has been measured with non-zero loss.

- Invalid bounds are not real boundaries for the searched value:

  - They are needed to track interval widths.

- Valid bounds are real boundaries for the searched value.

- Each non-initial phase ends with all bounds valid.

- Bound can become invalid if it re-measured at a longer trial duration in a sub-sequent phase.

* Search:

  + Start with a large (lower_bound, upper_bound) interval width, that determines measurement resolution.

  + Geometrically converge towards the width goal of the phase.

  + Each phase halves the previous width goal.

    - First measurement of the next phase will be internal search which always gives a valid bound and brings the width to the new goal.

    - Only one bound then needs to be re-measured with new duration.

* Use of internal and external searches:

  + External search:

    - Measures at transmit rates outside the (lower_bound, upper_bound) interval.

    - Activated when a bound is invalid, to search for a new valid bound by multiplying (for example doubling) the interval width.

    - It is a variant of "exponential search".

  + Internal search:

    - A "binary search" that measures at transmit rates within the (lower_bound, upper_bound) valid interval, halving the interval width.

- 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 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.

4. Sample Implementation

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

4.1. Input Parameters

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

4. *initial_trial_duration* - trial duration for initial MLRsearch phase.

5. *final_relative_width* - required measurement resolution expressed as (lower_bound, upper_bound) interval width relative to upper_bound.

6. *packet_loss_ratio* - maximum acceptable PLR search criterion for PDR measurements.

7. *number_of_intermediate_phases* - number of phases between the initial phase and the final phase. Impacts the overall MLRsearch duration. Less phases are required for well behaving cases, more phases may be needed to reduce the overall search duration for worse behaving cases.

4.2. Initial Phase

1. First trial measures at configured maximum transmit rate (MTR) and discovers maximum receive rate (MRR).

   * IN: trial_duration = initial_trial_duration.

   * IN: offered_transmit_rate = maximum_transmit_rate.

   * DO: single trial.

   * OUT: measured loss ratio.

   * OUT: MRR = measured receive rate. If loss ratio is zero, MRR is set below MTR so that interval width is equal to the width goal of the first intermediate phase.

2. Second trial measures at MRR and discovers MRR2.

   * IN: trial_duration = initial_trial_duration.

   * IN: offered_transmit_rate = MRR.

   * DO: single trial.

   * OUT: measured loss ratio.

   * OUT: MRR2 = measured receive rate. If loss ratio is zero, MRR2 is set above MRR so that interval width is equal to the width goal of the first intermediate phase. MRR2 could end up being equal to MTR (for example if both measurements so far

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

3. Third trial measures at MRR2.

   * IN: trial_duration = initial_trial_duration.

   * IN: offered_transmit_rate = MRR2.

   * DO: single trial.

   * OUT: measured loss ratio.

4.3. Non-Initial Phases

1. Main loop:

   1. IN: trial_duration for the current phase. Set to initial_trial_duration for the first intermediate phase; to final_trial_duration for the final phase; or to the element of interpolating geometric sequence for other intermediate phases. For example with two intermediate phases, trial_duration of the second intermediate phase is the geometric average of initial_trial_duration and final_trial_duration.

   2. IN: relative_width_goal for the current phase. Set to final_relative_width for the final phase; doubled for each preceding phase. For example with two intermediate phases, the first intermediate phase uses quadruple of final_relative_width and the second intermediate phase uses double of final_relative_width.

   3. IN: ndr_interval, pdr_interval from the previous main loop iteration or the previous phase. If the previous phase is the initial phase, both intervals are formed by a (correctly ordered) pair of MRR2 and MRR. Note that the initial phase is likely to create intervals with invalid bounds.

   4. DO: According to the procedure described in point 2., either exit the phase (by jumping to 1.7.), or calculate new transmit rate to measure with.

   5. DO: Perform the trial measurement at the new transmit rate and trial_duration, compute its loss ratio.

   6. DO: Update the bounds of both intervals, based on the new measurement. The actual update rules are numerous, as NDR

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

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

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

    1. If there is an invalid bound then prepare for external search:

        + IF the most recent measurement at NDR lower_bound transmit rate had the loss higher than zero, then the new transmit rate is NDR lower_bound decreased by two NDR interval widths.

        + Else, IF the most recent measurement at PDR lower_bound transmit rate had the loss higher than PLR, then the new transmit rate is PDR lower_bound decreased by two PDR interval widths.

        + Else, IF the most recent measurement at NDR upper_bound transmit rate had no loss, then the new transmit rate is NDR upper_bound increased by two NDR interval widths.

        + Else, IF the most recent measurement at PDR upper_bound transmit rate had the loss lower or equal to PLR, then the new transmit rate is PDR upper_bound increased by two PDR interval widths.

    2. Else, if interval width is higher than the current phase goal:

        + IF NDR interval does not meet the current phase width goal, prepare for internal search. The new transmit rate is a in the middle of NDR lower_bound and NDR upper_bound.

        + IF PDR interval does not meet the current phase width goal, prepare for internal search. The new transmit rate is a in the middle of PDR lower_bound and PDR upper_bound.

3. Else, if some bound has still only been measured at a lower duration, prepare to re-measure at the current duration (and the same transmit rate). The order of priorities is:

   + NDR lower_bound,

   + PDR lower_bound,

   + NDR upper_bound,

   + PDR upper_bound.

4. Else, do not prepare any new rate, to exit the phase. This ensures that at the end of each non-initial phase all intervals are valid, narrow enough, and measured at current phase trial duration.

5. FD.io CSIT Implementation

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

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

5.1. Additional details

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

1. Logarithmic transmit rate.

   * In order to better fit the relative width goal, the interval doubling and halving is done differently.

   * For example, the middle of 2 and 8 is 4, not 5.

2. Optimistic maximum rate.

   * The increased rate is never higher than the maximum rate.

   * Upper bound at that rate is always considered valid.

3. Pessimistic minimum rate.

    * The decreased rate is never lower than the minimum rate.

    * If a lower bound at that rate is invalid, a phase stops refining the interval further (until it gets re-measured).

4. Conservative interval updates.

    * Measurements above the current upper bound never update a valid upper bound, even if drop ratio is low.

    * Measurements below the current lower bound always update any lower bound if drop ratio is high.

5. Ensure sufficient interval width.

    * Narrow intervals make external search take more time to find a valid bound.

    * If the new transmit increased or decreased rate would result in width less than the current goal, increase/decrease more.

    * This can happen if the measurement for the other interval makes the current interval too narrow.

    * Similarly, take care the measurements in the initial phase create wide enough interval.

6. Timeout for bad cases.

    * The worst case for MLRsearch is when each phase converges to intervals way different than the results of the previous phase.

    * Rather than suffer total search time several times larger than pure binary search, the implemented tests fail themselves when the search takes too long (given by argument _timeout_).

7. Pessimistic external search.

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

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

5.1.1. FD.io CSIT Input Parameters

1. *maximum_transmit_rate* - Typical values: 2 * 14.88 Mpps for 64B 10GE link rate, 2 * 18.75 Mpps for 64B 40GE NIC (specific model).

2. *minimum_transmit_rate* - Value: 2 * 10 kpps (traffic generator limitation).

3. *final_trial_duration* - Value: 30 seconds.

4. *initial_trial_duration* - Value: 1 second.

5. *final_relative_width* - Value: 0.005 (0.5%).

6. *packet_loss_ratio* - Value: 0.005 (0.5%).

7. *number_of_intermediate_phases* - Value: 2. The value has been chosen based on limited experimentation to date. More experimentation needed to arrive to clearer guidelines.

8. *timeout* - Limit for the overall search duration (for one search). If MLRsearch oversteps this limit, it immediatelly declares the test failed, to avoid wasting even more time on a misbehaving SUT. Value: 600 (seconds).

9. *doublings* - Number of dublings when computing new interval width in external search. Value: 2 (interval width is quadroupled). Value of 1 is best for well-behaved SUTs, but value of 2 has been found to decrease overall search time for worse-behaved SUT configurations, contributing more to the overall set of different SUT configurations tested.

5.2. Example MLRsearch Run

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

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

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

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

6. IANA Considerations

No requests of IANA.

7. Security Considerations

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

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

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

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

8. Acknowledgements

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

9. References

9.1. Normative References

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Informative References

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

[PyPI-MLRsearch]
"MLRsearch 0.3.0, Python Package Index", February 2020, <https://pypi.org/project/MLRsearch/0.3.0/>.

Authors' Addresses

Maciek Konstantynowicz (editor)
Cisco Systems

Email: mkonstan@cisco.com

Vratko Polak (editor)
Cisco Systems

Email: vrpolak@cisco.com