Benchmarking Working Group
Internet-Draft
Intended status: Informational
Expires: April 25, 2019

M. Konstantynowicz, Ed.
V. Polak, Ed.
Cisco Systems
October 22, 2018

Multiple Loss Ratio Search for Packet Throughput (MLRsearch)
draft-vpolak-mkonstan-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 MLSsearch 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 chalenge and define a common (standard?) way to evaluate NFV packet throughput performance that takes into account varying characteristics of NFV systems under test.

Status of This Memo

Table of Contents

1. Terminology

- NDR - Non-Drop Rate, a packet throughput metric with Packet Loss Ratio equal zero (a zero packet loss), expressed in packets-per-second (pps). NDR packet throughput has an associated metric oftentimes referred to as NDR bandwidth expressed in bits-per-second (bps), and calculated as a product of:

  * NDR packet rate for specific packet (frame) size, and

  * Packet (L2 frame size) size in bits plus any associated L1 overhead.

- PLR - Packet Loss Ratio, a packet loss metric calculated as a ratio of (packets_transmitted - packets_received) to packets_transmitted, over the test trial duration.

- PDR - Partial-Drop Rate, a packet throughput metric with Packet Loss Ratio greater than zero (a non-zero packet loss), expressed in packets-per-second (pps). PDR packet throughput has an associated metric oftentimes referred to as PDR bandwidth expressed in bits-per- second (bps), and calculated as a product of:

  * PDR packet rate for specific packet (frame) size, and

  * Packet (L2 frame size) size in bits plus any associated L1 overhead.

2. MLRsearch Background

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

Two popular names for particular PLR criteria are Non-Drop Rate (NDR, with PLR=0, zero packet loss) and Partial Drop Rate (PDR, with PLR>0, non-zero packet loss). MLRsearch discovers NDR and PDR in a single search reducing required execution time compared to separate binary searches for NDR and PDR. 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 standard NDR/PDR binary search, while guaranteeing the same or similar results. (TODO: Specify "standard" in the previous sentence.)

If needed, MLRsearch can be easily adopted to discover more throughput rates with different pre-defined PLRs.

Unless otherwise noted, all throughput rates are _always_ bi-directional aggregates of two equal (symmetric) uni-directional packet rates received and reported by an external traffic generator.

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.
- *Initial phase*:
  - Uses link rate as a starting transmit rate and discovers the Maximum Receive Rate (MRR) used as an input to the first intermediate phase.
- *Intermediate phases*:
  - Start with initial trial duration (in the first phase) and converge geometrically towards the final trial duration (in the final phase).
  - Track two values for NDR and two for PDR.
    - The values are called (NDR or PDR) lower_bound and upper_bound.
    - Each value comes from a specific trial measurement (most recent for that transmit rate), and as such the value is associated with that measurement's duration and loss.
    - A bound can be invalid, for example if NDR lower_bound has been measured with nonzero loss.
    - Invalid bounds are not real boundaries for the searched value, but are needed to track interval widths.
    - Valid bounds are real boundaries for the searched value.
    - Each non-initial phase ends with all bounds valid.
  - Start with a large (lower_bound, upper_bound) interval width and geometrically converge towards the width goal (measurement resolution) of the phase. Each phase halves the previous width goal.

   * Use 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 doubling the interval width. It is a variant of "exponential search"_.

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

o *Final phase* is 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:

o In general MLRsearch is likely to execute more search trials overall, but less trials at a set final duration.

o In well behaving cases it greatly reduces (>50%) the overall duration compared to a single PDR (or NDR) binary search duration, while finding multiple drop rates.

o In all cases MLRsearch yields the same or similar results to binary search.

o Note: both binary search and MLRsearch are susceptible to reporting non-repeatable results across multiple runs for very bad behaving cases.

Caveats:

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

4. Sample Implementation

Following is a brief description of a sample MLRsearch implementation based on the open-source code running in FD.io CSIT project as part of a Continuous Integration / Continuous Development (CI/CD) framework.

4.1. Input Parameters

1. *maximum_transmit_rate* - maximum packet transmit rate to be used by external traffic generator, limited by either the actual Ethernet link rate or traffic generator NIC model capabilities. Sample defaults: 2 * 14.88 Mpps for 64B 10GE link rate, 2 * 18.75 Mpps for 64B 40GE NIC maximum rate.

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. Default: 2 * 10 kpps (could be higher).

3. *final_trial_duration* - required trial duration for final rate measurements. Default: 30 sec.

4. *initial_trial_duration* - trial duration for initial MLRsearch phase. Default: 1 sec.

5. *final_relative_width* - required measurement resolution expressed as (lower_bound, upper_bound) interval width relative to upper_bound. Default: 0.5%.

6. *packet_loss_ratio* - maximum acceptable PLR search criteria for PDR measurements. Default: 0.5%.

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. Default (2). (Value chosen based on limited experimentation to date. More experimentation needed to arrive to clearer guidelines.)

4.2. Initial phase

1. First trial measures at maximum rate and discovers MRR. a. _in_: trial_duration = initial_trial_duration. b. _in_: offered_transmit_rate = maximum_transmit_rate. c. _do_: single trial. d. _out_: measured loss ratio. e. _out_: mrr = measured receive rate.

2. Second trial measures at MRR and discovers MRR2. a. _in_: trial_duration = initial_trial_duration. b. _in_: offered_transmit_rate = MRR. c. _do_: single trial. d. _out_: measured loss ratio. e. _out_: mrr2 = measured receive rate.

3. Third trial measures at MRR2. a. _in_: trial_duration = initial_trial_duration. b. _in_: offered_transmit_rate = MRR2. c. _do_: single trial. d. _out_: measured loss ratio.

4.3. Non-initial phases

1. Main loop: a. _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_strial_duration and final_trial_duration. b. _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. c. _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 have lower_bound = MRR2, uper_bound = MRR. Note that the initial phase is likely to create intervals with invalid bounds. d. _do_: According to the procedure described in point 2, either exit the phase (by jumping to 1.g.), or prepare new transmit rate to measure with. e. _do_: Perform the trial measurement at the new transmit rate and trial_duration, compute its loss ratio. f. _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.c.), taking the updated intervals as new input. g. _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 1.d.):

   * 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.

* If interval width is higher than the current phase goal:

    + Else, _if_ NDR interval does not meet the current phase width goal, prepare for internal search. The new transmit rate is (NDR lower bound + NDR upper bound) / 2.

    + Else, _if_ PDR interval does not meet the current phase width goal, prepare for internal search. The new transmit rate is (PDR lower bound + PDR upper bound) / 2.

* 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.

* _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. Known Implementations

The only known working implementatin of MLRsearch is in Linux Foundation FD.io CSIT project. https://wiki.fd.io/view/CSIT. https://git.fd.io/csit/.

5.1. FD.io CSIT Implementation Deviations

This document so far has been describing a simplified version of MLRsearch algorithm. The full algorithm as implemented 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 current upper bound never update a valid upper bound, even if drop ratio is low. Measurements below 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_).

6. IANA Considerations

..

7. Security Considerations

..

8. Acknowledgements

..

9. 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>.

Authors' Addresses

Maciek Konstantynowicz (editor)
Cisco Systems

Email: mkonstan@cisco.com

Vratko Polak (editor)
Cisco Systems

Email: vrpolak@cisco.com