Benchmarking Working Group
Internet-Draft
Intended status: Informational
Expires: January 13, 2022

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

July 12, 2021

Multiple Loss Ratio Search for Packet Throughput (MLRsearch) draft-ietf-bmwg-mlrsearch-01

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 January 13, 2022.

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 (octets).

- 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.

- Effective loss ratio: A corrected value of measured packet loss ratio chosen to avoid difficulties if SUT exhibits decreasing loss with increasing load. Maximum of packet loss ratios measured at the same duration on all loads smaller than (and including) the current one.

- Target loss ratio: A packet loss ratio value acting as an imput for search. The search is finding tight enough lower and upper bound in intended load, so that the lower bound has smaller or equal loss ratio, and upper bound has strictly larger loss ratio.

For the tighterst upper bound, the effective loss ratio is the same as packet loss ratio. For the tightest lower bound, the effective loss ratio can be higher than the packet loss ratio, but still not larger than the target loss ratio.

- 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 octets) and inter-frame gap (12 octets). 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, 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.

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:
    - Target PLRs (e.g. 0.0 and 0.005 when searching for 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 all previous trial measurement results:
  + Duration, offered load and loss ratio are tracked.
  + 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.
  + The algorithm queries the results to find best lower and upper bounds.
    - Effective loss ratios are always used.
  + The phase ends if all target loss ratios have tight enough bounds.
* Search:
  + Iterate over target loss ratios in increasing order.
  + If both upper and lower bound are in measurement results for this duration, apply bisect until the bounds are tight enough, and continue with next loss ratio.
  + 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.
  + 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.
  + The logic guarantees that a measurement is never repeated with both duration and offered load being the same.
  + 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:

        + 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).

        + 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.

                o 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.

        + 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.

4. Sample Implementation

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

4.1. 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.

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. 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.

   * DO: 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.

4.3. 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 overal 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.

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. 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.

            - Example: If a successful transaction uses 10 packets, load is given in transactions per second, byt loss ratio is calculated from packets, 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 overriden 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,
  - 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 offes the same small tolerance before it start marking a "loss".
7. For result processing, we use lower bounds and ignore upper bounds.

5.1.1. 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 immediatelly 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 quadroupled). 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.

5.2. 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.

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>.

9.2. Informative References

[FDio-CSIT-MLRsearch]
"FD.io CSIT Test Methodology - MLRsearch", February 2021, <https://docs.fd.io/csit/rls2101/report/introduction/methodology_data_plane_throughput/methodology_mlrsearch_tests.html>.

[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 (editor)
Cisco Systems

Email: vrpolak@cisco.com