draft-ietf-bmwg-mlrsearch-05

IETF-118 Prague, BMWG Meeting

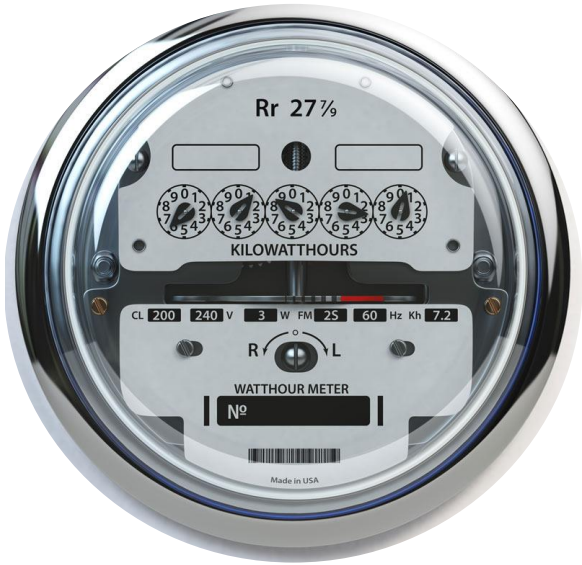
Authors: Vratko Polák, Maciek Konstantynowicz

MLRsearch Update

- draft-ietf-bmwg-mlrsearch-05 posted on 23rd of October 2023
  - The content now focuses on MLRsearch Specification and additional explanations.
  - Descriptions of search optimizations not required by Specification are simplified or omitted.
- BMWG next steps
  - Parts of the draft ready for a thorough BMWG review process, parts are almost ready.
  - BMWG help needed to drive the draft to completion and WG LC.
  - See Work Status slide at the end.

MLRsearch Topics

- Problem Statement
- Proposed Approach
- MLRsearch Specification
  - API Requirements
  - Goal Result
  - Conditional Throughput (with Example)
- MLRsearch Work Status as of -05

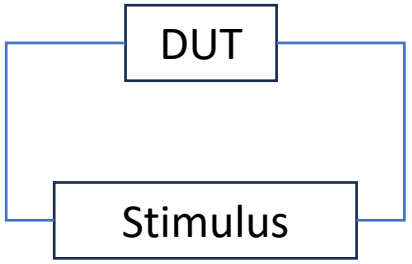

Rr 27 7/9
KILOWATTHOURS
CL 200 240 V 3 W FM 2S 60 Hz Kh 7.2
R L
WATTHOUR METER
№
Made in USA

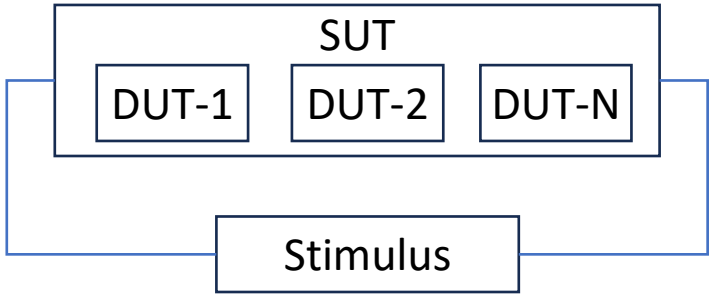
Problem Statement 1/5
Long Search Duration

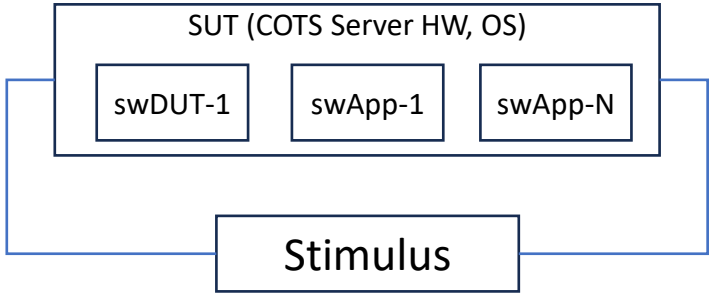
- Reducing test duration is critical for software networking CI/CD pipelines.
- Vanilla bisection is slow, because most trials spend time quite far away from the eventual throughput.
- Users can trade-off between search duration and achieved precision
  - There is no explicit stopping condition for throughput search [RFC2544].
  - But the logarithmic nature of bisection causes relatively big precision sacrifice for even small reductions in search duration.

Problem Statement 2/5
DUT in SUT

- Observe the DUT and SUT definitions in [RFC2285] and also in [RFC2544].
- With SW networking, SUT includes swDUT[(1)], as well as, Operating System (OS) and HW functions shared with other Applications
  - swDUT is nested within the SUT. SUT is multi-tenanted.
- DUT is subject to interference from the OS and other Applications (SW threads) running on the same SUT
  - In general effect of this is hard to predict. In the draft it is referred to as SUT noise.
- DUT can exhibit fluctuating performance itself e.g. due to pauses in execution needed for thread synchronization for internal stateful processing
  - This affects trial results in a way similar to SUT noise.
  - Draft uses the word noise as a shorthand covering both cases: this and genuine SUT noise.
- In the proposed model, SUT has a spectrum of observable performance results, not a single performance value
  - One end of the spectrum is the idealized noiseless performance value. The other end can be called a noiseful performance.
  - In practice, trial results close to the noiseful end of the spectrum happen only rarely, as the worse the performance value is, the more rarely it is seen in a trial.
  - Similarly, the extreme noiseless end of SUT spectrum is unlikely to be observable, due to small noise effects likely to occur multiple times during a trial.







(1) swDUT, software-based networking DUT.

Problem Statement 3/5
Repeatability and Comparability

- [RFC2544] does not suggest to repeat throughput search
  - From just one discovered throughput value, it cannot be determined how repeatable the value is.
  - Poor repeatability is also the main cause of poor comparability.
- [RFC2544] throughput requirements[(1)] affect the throughput results
  - The binary search results tend to wander away from the noiseless end of SUT performance spectrum, more frequently and more widely than shorter trials would, thus resulting in poor throughput repeatability.
- According to the SUT performance spectrum model, better repeatability will be at the noiseless end of the spectrum
  - The repeatability problem can be addressed by defining a search procedure which reports more stable results, even if they can no longer be called throughput in [RFC2544] sense.
  - Solutions to the DUT in SUT problem will help also with the repeatability problem.

[1] 60 seconds trial and no tolerance of a single frame loss.

Problem Statement 4/5
Throughput with Non-Zero Loss

- [RFC1242] (section 3.17) defines throughput as:
  - “The maximum rate at which none of the offered frames are dropped by the device.”
  - “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.
- Many benchmarking teams settle with small, non-zero loss ratio as the goal for their load search of swDUTs
  - Modern protocols tolerate frame loss better, compared to the time when [RFC1242] and [RFC2544] were specified.
  - Trials nowadays send way more frames within the same duration, increasing the chance of small SUT performance fluctuations being enough to cause frame loss.
  - If an approximation of the SUT noise impact on the trial loss ratio is known, it can be set as the goal loss ratio.
- Support for non-zero loss goals makes any search algorithm more user friendly
  - [RFC2544] throughput is not user friendly in this regard.
- If users are allowed to specify the goal loss ratio value, the usefulness is enhanced even more if users can specify multiple loss ratio values
  - Especially when a single search can find all relevant bounds.
  - Searching for multiple search goals also helps to describe the SUT performance spectrum better than a single search goal result.
  - E.g. repeated wide gap between zero and non-zero loss loads indicates the noise has a large impact on the observed performance, which is not evident from a single goal load search procedure result.

Problem Statement 5/5
Inconsistent Trial Results

- While performing throughput search by executing a sequence of measurement trials, there is a risk of encountering inconsistencies between trial results
  - The plain bisection never encounters inconsistent trials.
  - But [RFC2544] hints about possibility of inconsistent trial results
    - Section 24, where full trial durations are required, presumably because they can be inconsistent with results from shorter trial durations.
    - Section 26.3, where two successive zero-loss trials are recommended, presumably because after one zero-loss trial there can be subsequent inconsistent non-zero-loss trial.
- Any robust throughput search algorithm needs to decide how to continue the search in presence of such inconsistencies
  - Definitions of throughput in [RFC1242] and [RFC2544] are not specific enough to imply a unique way of handling such inconsistencies.
- Ideally, there will be a definition of a new quantity which both generalizes throughput for non-zero-loss, while being precise enough to force a specific way to resolve trial result inconsistencies.

Proposed Approach

- Long Search Duration
  - The main factor improving the overall search time is the introduction of preceding targets.
  - Less impactful time savings are achieved by pre-initial trials, halving mode and smart splitting in bisecting mode.
- DUT in SUT
  - In the proposed model, SUT does not have a single performance value, it has a spectrum of observable results
  - One end of the spectrum is the idealized noiseless performance value. The other end can be called a noiseful performance.
  - DUT in SUT performance measurement problem is reduced to estimating the noiseless end of SUT performance spectrum from a limited number of trial results.
  - Note: Please pay attention to the new MLRsearch parameter Exceed Ratio.
- Repeatability and Comparability
  - Multiple trials with noise tolerance enhancement, using non-zero goal Exceed Ratio value, increases result stability.
  - This allows MLRsearch to achieve benefits of Binary Search with Loss Verification
    - Recommended in [RFC9004] (section 6.2)
    - Specified in [TST009] (section 12.3.3).
- Throughput with Non-Zero Loss
  - Configurable loss ratio in MLRsearch search goals are there in direct support for non-zero-loss conditional throughput.
  - In practice the conditional throughput results' stability increases with higher loss ratio goals.
- Inconsistent Trial Results
  - In several places, MLRsearch is "conservative" when handling (potentially) inconsistent results.
  - This includes the requirement for the relevant lower bound to be smaller than any upper bound, the unequal handling of good and bad short trials, and preference to lower load when choosing the winner among candidates.

MLRsearch Specification

- MLRsearch specification requirements are formulated mostly as requirements for APIs between library components:
  - Manager: Configures everything, calls the controller, creates the test report.
  - Controller: Repeatedly calls the measurer until all search goals are achieved.
  - Measurer: Performs one trial when called.
- Controller assigns a goal result to every search goal, see next slides.
  - The goal result attribute definitions give a precise meaning to the search goal attributes.
  - The strictness of the definitions ensures comparability between MLRsearch implementations.
- MLRsearch Python library version 1.2.1 is an example of an implementation satisfying all requirements and recommendations.
  - Many optimizations in that library have negligible impact on comparability, thus they are not described in the draft.
  - This makes other implementations (including future versions of the Python library) free to evolve their optimizations.

API requirements (and search goal)

- Measurer performs one trial each time it is called, turning (intended) duration and (intended) load into trial loss ratio.
  - Forwarding rate is computed from the trial loss ratio and the trial load.
  - Optionally, measurer can also return trial offered duration, e.g. including time with no traffic.
- Controller is called only once, turning inputs (list of search goals) into outputs (goal result for each search goal).
- Search goal attributes:
  - Final trial duration : Intended duration for trials at the end of the search.
    - Trials shorter than this value are called "short", otherwise "long", according to this search goal.
  - (Trial) Duration sum: The amount of long trials sufficient to classify a load as an upper of lower bound for this search goal.
  - (Frame) Loss ratio: Decides whether a trial result is good or bad for this search goal.
  - (Loss) Exceed ratio: What portion of trials may be bad at a lower bound.
    - Also used to select which of the trial results will be turned into the conditional throughput.
  - Optional attributes: initial trial duration, width, ...

Goal Result (and load classification)

- A goal result consists of several attributes, all rely on load classification.
- Required attribute: Relevant upper bound.
  - Smallest load that got classified as an upper bound according to this search goal.
- Required attribute: Relevant lower bound.
  - Largest load that got classified as a lower bound according to this search goal.
    - Among loads smaller than the relevant upper bound.
- Optional attribute: Conditional throughput.
  - Forwarding rate of one of trials at the relevant lower bound. See next slides for selection logic.
- Irregular goal result is allowed, but otherwise unspecified.
  - A goal result is regular if it satisfies all requirements.
  - Example of an irregular result: When there is no upper bound due to SUT having zero loss at max load.
- Load classification: Whether a load is a lower bound or an upper bound when search ends.
  - The classification conditions are strict, as loosening them harms comparability.
  - The classification conditions and complicated, especially in presence of short trials.
  - This is the only requirement that has to remain complicated due to optimizations in the Python library.

Conditional Throughput Definition

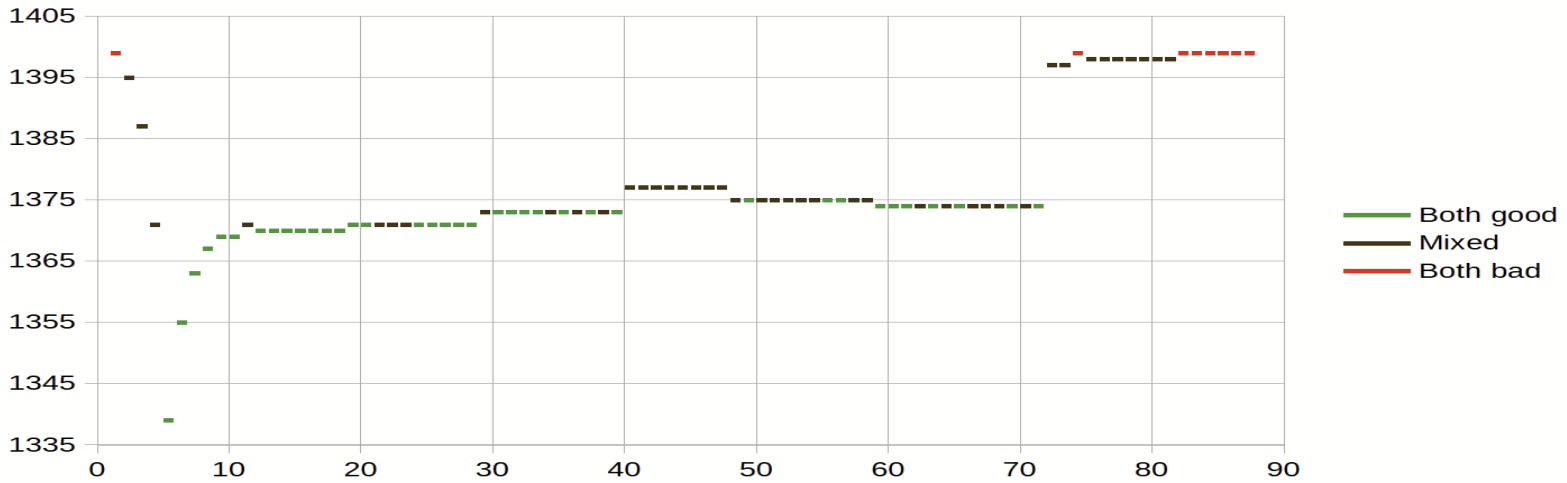
- Forwarding rate at a quantile (based on exceed ratio) among of all expected long trials at the relevant lower bound.
  - Pessimistic: If some trials are not measured yet, assume they would get 100% loss.
  - With exceed ratio 50%, the quantile is the median.
  - In practice, search stops as soon as the quantile is good, returning the worst good trial.
  - Conditional throughput accuracy relates the goal loss ratio to the goal width.
  - Not a required attribute of goal result, as even better quantities may be found, especially when short trial results are present.
- Draft-04 definition was different: Average forwarding rate over measured good long trials at relevant lower bound.
  - This can lead to unintuitive results, example on the next slide.
  - Was a required attribute of the goal result.

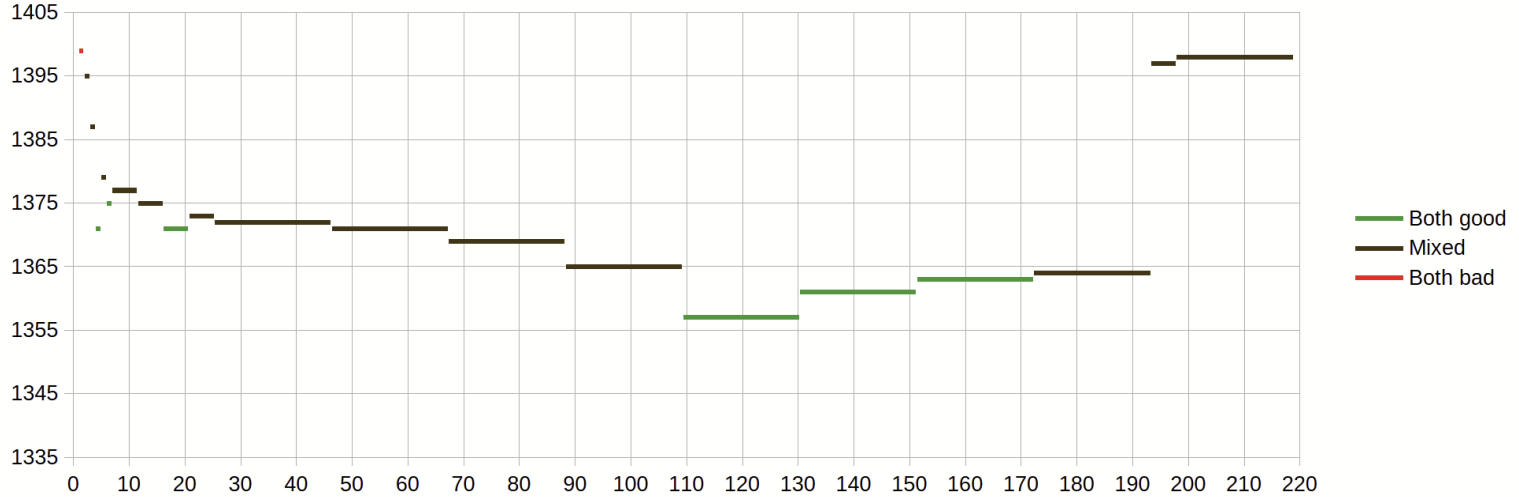
Conditional Throughput Example

- Two search goals (‰ is per mille, 1‰ = 0.1%):
  - NDR: 1s final trial duration, 7s goal duration sum, 0‰ goal loss ratio, 50% goal exceed ratio.
  - PDR: 1s final trial duration, 7s goal duration sum, 5‰ goal loss ratio, 50% goal exceed ratio.
- Search examines the offered load of 1 Mpps as a possible lower bound for both goals.
- Sorted sequence of trial loss ratios after each trial measurement (new result is underlined, median of 7 is bold):
  1. 0‰: NDR undecided, PDR undecided.
  2. 0‰, 7‰: NDR undecided, PDR undecided.
  3. 0‰, 0‰, 7‰: NDR undecided, PDR undecided.
  4. 0‰, 0‰, 5‰, **7‰**: NDR undecided, PDR undecided.
  5. 0‰, 0‰, 0‰, **5‰**, 7‰: NDR undecided, PDR lower bound.
  6. 0‰, 0‰, 0‰, **0‰**, 5‰, 7‰: NDR lower bound, PDR lower bound.
- Row 6 conditional throughput (worst case median of 7): NDR 0‰ (1 Mpps), PDR 0‰ (1 Mpps).
  - Row 5 would give draft-05 conditional throughput 5‰ (995 kpps) so accuracy is 5‰.
    - Row 6 happened just because the same load was interesting also for NDR.
  - Row 6 draft-04 conditional throughput (average of all good): NDR 0‰ (1 Mpps), PDR 1‰ (999 kpps).

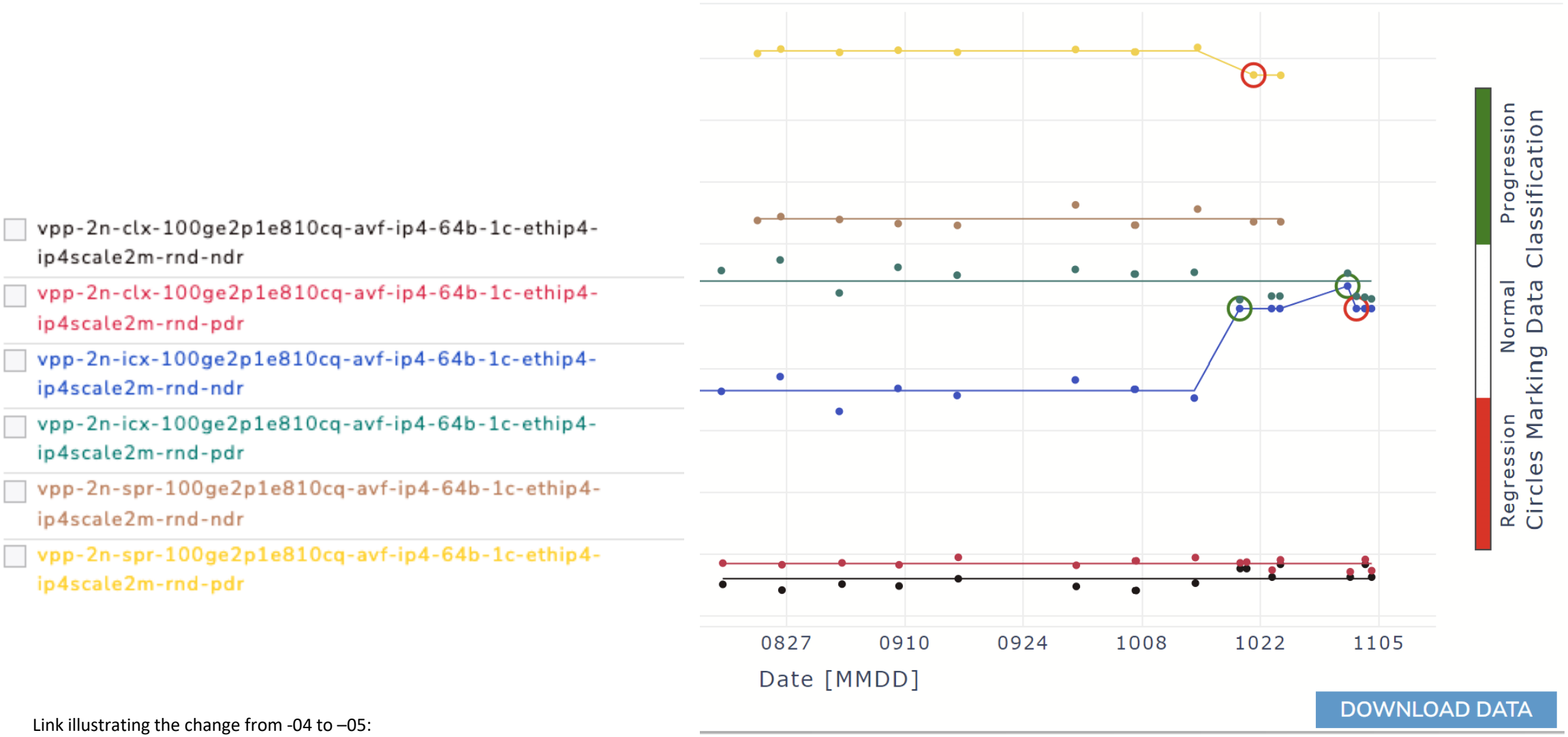
Search examples

- Recommended-like settings:
  - NDR:
    - 1s initial trial duration
    - 1s final trial duration
    - 20s goal duration sum
    - 0‰ goal loss ratio
    - 50% goal exceed ratio
  - PDR:
    - 1s initial trial duration
    - 1s final trial duration
    - 20s goal duration sum
    - 5‰ goal loss ratio
    - 50% goal exceed ratio
- RFC 2544-like settings:
  - NDR:
    - 1s initial trial duration
    - 20s final trial duration
    - 20s goal duration sum
    - 0‰ goal loss ratio
    - 0% goal exceed ratio.
  - PDR:
    - 1s initial trial duration
    - 20s final trial duration
    - 20s goal duration sum
    - 5‰ goal loss ratio
    - 0% goal exceed ratio.





Changes from draft -04 to draft-05: Results



Link illustrating the change from -04 to –05:
Shortcut: https://bit.ly/3QGCpnB
Actual: https://csit.fd.io/trending/#eNrllU0KwjAQhU9TNzLQRGvduFB7D4npaAv9GZMo6ulNRRiFCqVIF3aRhPAm85KPB7GuNrizWKyCaBPEm0DGeeqnYLae-uVCBLICXVxBhOERJQlcilCfQF0OkNMcFvM9CA3osmbnh9WqQFmCqVKoUtP0ktumV3p2H41ZoezGylc7rlcGFR_wniw5tG8urbfi4oNRJdr8jnzCP4d17dmwJPSnjbvRm_p6apw8K34NksYDkjqBzPWgiWy3-4dE9gVJ4wHZLZGWzJCJbLf7h0T2BUnjAcmJjJJJVZvy-XtHyQOx2BxY

MLRsearch Work Status

- Final and ready for review:
  - Problem statement.
  - MLRsearch specification.
    - Except impact of short trials on conditional throughput.
  - Recommended values for goal attributes.
- Not final and under editing:
  - Conditional throughput: When there is an excess of bad short results.
  - Explanations: Currently too verbose.
  - Appendices: The code got formatted wrong.
  - Typos and grammar, everywhere.
  - Missing references.
- Open points for adding and for discussion:
  - MLRsearch library should allow the measurer to add extra data (e.g. telemetry), visible in conditional throughput.
  - MLRsearch specification can recommend more ideas for useful implementations.
  - More clarity on reporting the behaviour of the measurer.

THANK YOU !

draft-ietf-bmwg-mlrsearch-05

IETF-118 Prague, BMWG Meeting

Authors: Vratko Polák, Maciek Konstantynowicz