Draft changes -02 to -03

- Added an Example MLRsearch Run table based on real test data, tracking MLRsearch phases, trial measurements and updated bound values.
- Default MLRsearch values got moved into FD.io CSIT implementation specific section.
Overview: Multiple Loss Ratio search (MLRsearch)

- MLRsearch discovers multiple packet throughput rates in a single search
  - With each rate associated with a distinct Packet Loss Ratio (PLR) criteria
- Provides much shorter execution times for cases when multiple rates need to be found:
  - For example in NFV benchmarking to discover both NDR and PDR throughput
    - NDR: Non-Drop Rate with PLR=0, zero packet loss
    - PDR: Partial-Drop Rate with PLR>0, non-zero packet loss
  - Instead of running separate binary searches for NDR and PDR.
Overview: Multiple Loss Ratio search (MLRsearch)

- MLRsearch execution time gets reduced even further
  - By using shorter trial durations in the intermediate steps
  - With only the final measurements conducted at the specified final trial duration.
- MLRsearch is a packet throughput search algorithm suitable for deterministic systems
  - As opposed to probabilistic systems

MLRsearch is compatible with RFC2544.
Applied to LFN FD.io CSIT Project Use Case

- LFN FD.io, Linux Foundation Networking FastData.io project with two main sub-projects
  - VPP, Vector Packet Processing, a feature rich user-mode SW data-plane
  - CSIT, Continuous System Integration and Testing, SW data-plane testing and performance benchmarking

- FD.io CSIT developed and operates CI system using following packet and bandwidth throughput benchmarking methodologies
  - MRR, Maximum Receive Rate, for discovering packet/bandwidth rate (regardless of PLR, packet-loss-ratio) sustained by DUT/SUT under specified Maximum Transmit Rate (MTR) packet load offered by traffic generator.
  - MLRsearch, this draft, for discovering zero-frame-loss (a.k.a. NDR, Non-Drop Rate) and partial-drop-rate (PDR at specified PLR) throughput.
  - PLRsearch, Probabilistic Loss Ratio search, for soak testing the throughput rate at set packet-loss-ratio.

- MLRsearch was instrumental back in January 2020 for FD.io CSIT to discover serious issues with COTS system updates
  - BIOS, Intel microcode, Linux kernel, pushed by vendors and community in order to mitigate the Spectre-Meltdown vulnerabilities.
    - CVE-2019-11091 [microarchitectural data sampling uncacheable memory (MDSUM)] aka 'RIDL'
    - CVE-2019-11135 [TSX asynchronous abort] aka 'TAA' aka 'ZombieLoad V2'
  - MRR test results did not show any significant anomalies.
  - MLRsearch tests showed substantial degradation of PDR and NDR throughput due to increased compute system jitter. Thanks to time optimized execution the problem got picked up and root caused.
Example MLRsearch Run

- Table on the right shows data from a real test run in CSIT, using the default input values as described in the draft.
- The first column is the MLRsearch phase.
- The second is the trial measurement performed
  - Aggregate bidirectional offered load in mega (10^6) packets 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.
- Black bold font signifies changed values.
- Blue bold font signifies results of the search.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Trial</th>
<th>NDR lower</th>
<th>NDR upper</th>
<th>PDR lower</th>
<th>PDR upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>init.</td>
<td>37.50</td>
<td>1.00 N/A</td>
<td>37.50 1.</td>
<td>N/A</td>
<td>37.50 1.</td>
</tr>
<tr>
<td>init.</td>
<td>10.55</td>
<td>1.00 +10.55 1.</td>
<td>37.50 1.</td>
<td>+10.55 1.</td>
<td>37.50 1.</td>
</tr>
<tr>
<td>init.</td>
<td>9.437</td>
<td>1.00 +9.437 1.</td>
<td>10.55 1.</td>
<td>+9.437 1.</td>
<td>10.55 1.</td>
</tr>
<tr>
<td>int 1</td>
<td>6.053</td>
<td>1.00 6.053 1.</td>
<td>9.437 1.</td>
<td>6.053 1.</td>
<td>9.437 1.</td>
</tr>
<tr>
<td>int 1</td>
<td>7.558</td>
<td>1.00 7.558 1.</td>
<td>9.437 1.</td>
<td>7.558 1.</td>
<td>9.437 1.</td>
</tr>
<tr>
<td>int 1</td>
<td>8.446</td>
<td>1.00 8.446 1.</td>
<td>9.437 1.</td>
<td>8.446 1.</td>
<td>9.437 1.</td>
</tr>
<tr>
<td>int 1</td>
<td>8.928</td>
<td>1.00 8.928 1.</td>
<td>9.437 1.</td>
<td>8.928 1.</td>
<td>9.437 1.</td>
</tr>
<tr>
<td>int 1</td>
<td>9.179</td>
<td>1.00 9.179 1.</td>
<td>9.179 1.</td>
<td>9.179 1.</td>
<td>9.179 1.</td>
</tr>
<tr>
<td>int 1</td>
<td>9.052</td>
<td>1.00 9.052 1.</td>
<td>9.179 1.</td>
<td>9.179 1.</td>
<td>9.179 1.</td>
</tr>
<tr>
<td>final</td>
<td>9.147</td>
<td>30.0 9.115 5.</td>
<td>9.147 30.</td>
<td>9.307 5.</td>
<td>9.401 5.</td>
</tr>
</tbody>
</table>
MLRsearch Sample Implementation

- A working implementation of MLRsearch is in Linux Foundation FD.io CSIT project.
  - Used for continuous measurements of NDR and PDR rates of:
    - FD.io VPP
    - DPDK L3fwd
    - DPDK Testpmd
  - Sample throughput results:
  - General project info:
    - [https://wiki.fd.io/view/CSIT](https://wiki.fd.io/view/CSIT)
    - [https://git.fd.io/csit/](https://git.fd.io/csit/)
- MLRsearch Python package published on PyPI:
  - [https://pypi.org/project/MLRsearch/](https://pypi.org/project/MLRsearch/)
Thanks for Comments, here some questions

- Can we move it from the named draft to BMWG workgroup one?
- Next steps?
THANK YOU!

draft-vpolak-mkonstan-bmwg-mlrsearch-03

Post IETF-107 Interim BMWG Meeting
Authors: Vratko Polák, Maciek Konstantynowicz
Presented by: Maciek Konstantynowicz