Benchmarking Working Group
Internet-Draft
Intended status: Informational
Expires: 17 September 2025

M. Konstantynowicz
V. Polak
Cisco Systems
16 March 2025

Multiple Loss Ratio Search (MLRsearch)
draft-ietf-bmwg-mlrsearch-10

Abstract

This document ~~proposes~~ specifies extensions to "Benchmarking Methodology for Network Interconnect Devices" (RFC 2544) throughput search by defining a new methodology called Multiple Loss Ratio search (MLRsearch). MLRsearch aims to minimize search duration, support multiple loss ratio searches, and enhance result repeatability and comparability.

~~The primary reason for extending RFC 2544~~MLRsearch is motivated by the pressing need to address the challenges of evaluating and testing ~~the~~ the various data plane solutions, especially ~~of~~in software-based networking systems.

~~To give users more freedom,~~ MLRsearch provides additional configuration options such as allowing multiple short trials per load instead of one large trial, tolerating a certain percentage of trial results with higher loss, and supporting the search for multiple goals with varying loss ratios.

Commenté [MB1]: This may trigger automatically a comment whether we change (update or amend) any of RFC2544 text.

Do we?

Commenté [MB2]: The abstract should self-contained. Hence the need to expand the RFC title.

Commenté [MB3]: What is meant here? What is specific to these systems?

Do we need to have this mention at this stage?

Commenté [MB4]: Too detailed for an abstract. Can be mentioned in an overview/introduction section

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 17 September 2025.

Table of Contents

~~1. Requirements Language~~

~~The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.~~

2. Introduction~~Purpose and Scope~~

~~The purpose of t~~This document ~~is to~~ describes the Multiple Loss Ratio search (MLRsearch) methodology, optimized for determining data plane throughput in software-based networking ~~devices and~~ functions. Such functions can be physical (e.g., a device) or logical (e.g., Virtual Network Function).

Commenté [MB5]: Move after the intro

Commenté [MB6]: Should be defined.

Not sure what is specific as any networking device is a software-based device. Even hardware, it is not more than frozen software 😉

Applying the vanilla [RFC2544] throughput bisection method to software DUTs results in several problems:

* Binary search takes too long as most trials are done far from the eventually found throughput.

* The required final trial duration and pauses between trials prolong the overall search duration.

* Software DUTs show noisy trial results, leading to a big spread of possible discovered throughput values.

* Throughput requires a loss of exactly zero frames, but the industry frequently allows for low but non-zero losses.

* The definition of throughput is not clear when trial results are inconsistent.

To address these problems, the MLRsearch test methodology employs the following enhancements:

* Allow multiple short trials instead of one big trial per load.

  - Optionally, tolerate a percentage of trial results with higher loss.

* Allow searching for multiple Search Goals, with differing loss ratios.

  - Any trial result can affect each Search Goal in principle.

* Insert multiple coarse targets for each Search Goal, earlier ones need to spend less time on trials.

  - Earlier targets also aim for lesser precision.

  - Use Forwarding Rate (FR) at maximum offered load [RFC2285] (Section 3.6.2) to initialize bounds.

* ~~Take care~~Be caution ~~-~~when dealing with inconsistent trial results.

  - Reported throughput is smaller than the smallest load with high loss.

  - Smaller load candidates are measured first.

* Apply several load selection heuristics to save even more time by trying hard to avoid unnecessarily narrow bounds.

Some of these enhancements are formalized as MLRsearch specification~~,~~. ~~the~~ The remaining enhancements are treated as implementation details, thus, achieving high comparability without limiting future improvements.

MLRsearch configuration options ~~are flexible enough to~~ support both conservative settings and aggressive settings. Conservative enough settings lead to results unconditionally compliant with [RFC2544],

Commenté [MB7]: Can we have an explicit reference for the method?

Commenté [MB8]: Expand

Commenté [MB9]: Can we have a public reference to share here?

Commenté [MB10]: What is meant here?

Commenté [MB11]: Can we expand on this one?

Commenté [MB12]: There is no such section in the document.

Do you meant Section 3.6.2 of [RFC2285]?

If so, please update accordingly.

Idem for all similar occurrences in the document. Thanks.

Commenté [MB13]: Maximizing means ?

Commenté [MB14]: Which ones?

Commenté [MB15]: Where? In this document?

Commenté [MB16]: Where those are defined? Please add a pointe to the apropriate section.

Commenté [MB17]: «flexible» is ambiguous. Simply, state what we do.

but without much improvement on search duration and repeatability. Conversely, aggressive settings lead to shorter search durations and better repeatability, but the results are not compliant with [RFC2544].

This document does not ~~No~~ change or obsolete any part of [RFC2544] ~~is intended to be obsoleted by this document~~.

1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

xx

xx

3. ~~Identified~~ Overview of RFC 2544 Problems

This ~~chapter~~ section describes the problems affecting usability of various performance testing methodologies, mainly a binary search for [RFC2544] unconditionally compliant throughput.

3.1. Long Search Duration

The emergence of software DUTs, with frequent software updates, and a number of different frame processing modes and configurations, has increased both the number of performance tests required to verify ~~thea~~ DUT update and the frequency of running those tests. This makes the overall test execution time even more important than before.

The ~~current [RFC2544]~~ throughput definition per [RFC2544] restricts the potential for time-efficiency improvements. A more generalized throughput concept could enable further enhancements while maintaining the precision of simpler methods.

The bisection method, when used in a manner unconditionally compliant with [RFC2544], is excessively slow.

This is because a significant amount of time is spent on trials with loads that, in retrospect, are far from the final determined throughput.

[RFC2544] does not specify any stopping condition for throughput search, so users already ~~have an access to~~have access to a limited trade-off between search duration and achieved precision. However, each of the full 60-second trials doubles the precision.~~, s~~ As such,~~e~~ not many trials can be removed without a substantial loss of precision.

3.2. DUT in SUT

[RFC2285] defines:

Commenté [MB18]: Add pointers where this is further elaborated.

Commenté [MB19]: List the set of terms/definitions used in this document.
I guess we should at least leverage terms defined in 2544/1242.

Commenté [MB20]: Also, please add a statement that the convention used in bmwg are followed here as well (def, discussion, etc.)

Commenté [MB21]: Is this really new?

Commenté [MB22]: Won't age well

Commenté [MB23]: Concretely, be affirmative if we provide an elaborated def, otherwise this statement can be removed.

Commenté [MB24]: Can we have a reference?

Commenté [MB25]: Define «users».

Commenté [MB26]: Can we include a reminder of the 2544 search basics? (no need to be verbose, though)?

DUT as:

* The network frame forwarding device to which stimulus is offered and response measured (Section 3.1.1 of [RFC2285] ~~(Section 3.1.1)~~).

SUT as:

* The collective set of network devices as a single entity to which stimulus is offered and response measured (Section 3.1.2 of [RFC2285] ~~(Section 3.1.2)~~.

Section 19 of [RFC2544] ~~(Section 19)~~ specifies a test setup with an external tester stimulating the networking system, treating it either as a single DUT, or as a system of devices, ~~an~~ a SUT.

In the case of software networking, the SUT consists of not only the DUT as a software program processing frames, but also of server hardware and operating system functions, with that server hardware resources shared across all programs including the operating system.

Given that ~~the~~ a SUT is a shared multi-tenant environment encompassing the DUT and other components, the DUT might inadvertently experience interference from the operating system or other software operating on the same server.

Some of this interference can be mitigated. For instance, pinning DUT program threads to specific CPU cores and isolating those cores can prevent context switching.

Despite taking all feasible precautions, some adverse effects may still impact the DUT's network performance. In this document, these effects are collectively referred to as SUT noise, even if the effects are not as unpredictable as what other engineering disciplines call noise.

A DUT can also exhibit fluctuating performance itself, for reasons not related to the rest of SUT. For example, this can be due to pauses in execution as needed for internal stateful processing. In many cases this may be an expected per-design behavior, as it would be observable even in a hypothetical scenario where all sources of SUT noise are eliminated. Such behavior affects trial results in a way similar to SUT noise. As the two ~~phenomenons~~ phenomena are hard to distinguish, in this document the term 'noise' is used to encompass both the internal performance fluctuations of the DUT and the genuine noise of the SUT.

A simple model of SUT performance consists of an idealized noiseless performance~~,~~ and additional noise effects. For a specific SUT, the noiseless performance is assumed to be constant, with all observed performance variations being attributed to noise. The impact of the noise can vary in time, sometimes wildly, even within a single trial. The noise can sometimes be negligible, but frequently it lowers the observed SUT performance as observed in trial results.

Commenté [MB27]: Double check

Commenté [MB28]: This makes assumptions on the software architecture. We need to make sure this is generic enough.

For example, what is a server? Etc.

Does it applies to container, microservice, SF a la RFC7665, VNF a la ETSI, etc.?

Commenté [MB29]: Such as?

Commenté [MB30]: If many? Or do we assume there are always many?

In this simple model, a SUT does not have a single performance value, it has a spectrum. 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. The worse a possible performance value is, the more rarely it is seen in a trial. Therefore, the extreme noiseful end of the SUT spectrum is not observable among trial results. Furthermore, the extreme noiseless end of the SUT spectrum is unlikely to be observable, this time because some small noise effects are very likely to occur multiple times during a trial.

Unless specified otherwise, this document's focus is on the potentially observable ends of the SUT performance spectrum, as opposed to the extreme ones.

When focusing on the DUT, the benchmarking effort should ideally aim to eliminate only the SUT noise from SUT measurements. However, this is currently not feasible in practice, as there are no realistic enough models that would be capable to distinguish SUT noise from DUT fluctuations ~~(at least based on authors' experience and~~based on the available literature at the time of writing).

Assuming a well-constructed SUT, the DUT is likely its primary bottleneck. In this case, ~~we can define~~ the DUT's ideal noiseless performance is defined as the noiseless end of the SUT performance spectrum.
That is true for throughput. Other performance metrics, such as latency, may require additional considerations.

Note that by this definition, DUT noiseless performance also minimizes the impact of DUT fluctuations, as much as ~~realistically~~ possible for a given trial duration.

The MLRsearch methodology aims to solve the DUT in SUT problem by estimating the noiseless end of the SUT performance spectrum using a limited number of trial results.

~~Any i~~Improvements to the throughput search algorithm, aimed at better dealing with software networking SUT and DUT setups, should employ strategies recognizing the presence of SUT noise, allowing the discovery of (proxies for) DUT noiseless performance at different levels of sensitivity to SUT noise.

3.3. Repeatability and Comparability

[RFC2544] does not suggest ~~to repeat~~repeating throughput search. ~~And f~~Also, note that from ~~just~~ simply one discovered throughput value, it cannot be determined how repeatable that value is. Poor repeatability then leads to poor comparability, as different benchmarking teams may obtain varying throughput values for the same SUT, exceeding the expected differences from search precision.

Repeatability is important also when the test procedure is kept the same, but SUT is varied in small

Commenté [MB31]: I don't parse this one. Please reword.

Commenté [MB32]: As we need to reflect the view of the WG/IETF, not only authors

Commenté [MB33]: That is?

Commenté [MB34]: Please avoid «we..» constructs.

Commenté [MB35]: Can we cite an example?

Commenté [MB36]: ?

ways. For example, during development of software-based DUTs, repeatability is needed to detect small regressions.

[RFC2544] throughput requirements (60 seconds trial and no tolerance of a single frame loss) affect the throughput results ~~in the following way~~ as follows:~~.~~ The SUT behavior close to the noiseful end of its performance spectrum consists of rare occasions of significantly low performance, but the long trial duration makes those occasions not so rare on the trial level. Therefore, 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 causing poor throughput repeatability.

The repeatability problem can be better addressed by defining a search procedure that identifies a consistent level of performance, even if it does not meet the strict definition of throughput in [RFC2544].

According to the SUT performance spectrum model, better repeatability will be at the noiseless end of the spectrum. Therefore, solutions to the DUT in SUT problem will help also with the repeatability problem.

Conversely, any alteration to [RFC2544] throughput search that improves repeatability should be considered as less dependent on the SUT noise.

An alternative option is to simply run a search multiple times, and report some statistics (e.g.,~~e.g.~~ average and standard deviation). This can be used for a subset of tests deemed more important, but it makes the search duration problem even more pronounced.

Commenté [MB37]: What about at some other representative percentiles?

3.4. Throughput with Non-Zero Loss

Section 3.17 of [RFC1242] ~~(Section 3.17)~~ defines throughput as: The maximum rate at which none of the offered frames are dropped by the device.

Then, it says: Since even the loss of one frame in a data stream can cause significant delays while waiting for the ~~higher level~~higher-level protocols to time out, it is useful to know the actual maximum data rate that the device can support.

However, many benchmarking teams accept a low, non-zero loss ratio as the goal for their load search.

Motivations are many:

* ~~Modern p~~Networking protocols tolerate frame loss better, compared to the time when [RFC1242] and [RFC2544] were specified.

Commenté [MB38]: 1242 was also modern at the time they were published 😉

This can be easily stale. Let's avoid that

* Deployed t~~T~~rial approaches ~~s~~ ~~nowadays~~ require sending way more frames within the same duration, increasing the chance of a small SUT performance fluctuation being enough to cause frame loss.

Commenté [MB39]: Won't age well.

* Small bursts of frame loss caused by noise have otherwise smaller impact on the average frame loss ratio observed in the trial, as during other parts of the same trial the SUT may work more closely to its noiseless performance, thus perhaps lowering the Trial Loss Ratio below the Goal Loss Ratio value.

* If an approximation of the SUT noise impact on the Trial Loss Ratio is known, it can be set as the Goal Loss Ratio.

~~* For more information, see an earlier draft [Lencze-Shima] (Section 5) and references there.~~

Regardless of the validity of all similar motivations, support for non-zero loss goals makes ~~any~~ a search algorithm more user-friendly. [RFC2544] throughput is not user-friendly in this regard.

Furthermore, allowing users to specify multiple loss ratio values, and enabling a single search to find all relevant bounds, significantly enhances the usefulness of the search algorithm.

Searching for multiple Search Goals also helps to describe the SUT performance spectrum better than the result of a single Search Goal. For example, the repeated wide gap between zero and non-zero loss loads indicates that the noise has a large impact on the observed performance, which ~~is~~ might not be evident from a single goal load search procedure result.

It is easy to modify the vanilla bisection to find a lower bound for the load that satisfies a non-zero Goal Loss Ratio. But it is not that obvious how to search for multiple goals at once, hence the support for multiple Search Goals remains a problem.

There does not seem to be a consensus on which ratio value is the best. For users, performance of higher protocol layers is important, for ~~example~~example, goodput of TCP connection (TCP throughput, [RFC6349]), but relationship between goodput and loss ratio is not simple. ~~See~~Refer to [Lencze-Kovacs-Shima] for examples of various corner cases, Section 3 of [RFC6349] ~~Section 3~~ for loss ratios acceptable for an accurate measurement of TCP throughput, and [Ott-Mathis-Semke-Mahdavi] for models and calculations of TCP performance in presence of packet loss.

3.5. Inconsistent Trial Results

While performing throughput search by executing a sequence of measurement trials, there is a risk of encountering inconsistencies between trial results.

Examples include, but are not limited to:

* A trial at the same load (same or different trial duration) results in a different Trial Loss Ratio.

* A trial at a larger load (same or different trial duration)

Commenté [MB40]: Please split. Too long

Commenté [MB41]: Help readers find where to look for an authoritative definition.

Commenté [MB42]: Help readers find where to look for an authoritative definition.

Commenté [MB43]: We can't claim that

Commenté [MB44]: Among?

Also, indicate «at the time of writing».

results in a lower Trial Loss Ratio.

The plain bisection never encounters inconsistent trials. But [RFC2544] hints about the possibility of inconsistent trial results, in two places in its text. The first place is Section 24, where full trial durations are required, presumably because they can be inconsistent with the results from short trial durations. The second place is Section 26.3, where two successive zero-loss trials are recommended, presumably because after one zero-loss trial there can be a subsequent inconsistent non-zero-loss trial.

~~Any~~ A robust throughput search algorithm needs to decide how to continue the search in the 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 Goal Loss Ratio values (and other possible repeatability enhancements), while being precise enough to force a specific way to resolve trial result inconsistencies. But until such a definition is agreed upon, the correct way to handle inconsistent trial results remains an open problem.

Relevant Lower Bound is the MLRsearch term that addresses this problem.

4. MLRsearch Specification

MLRsearch specification ~~describes all~~provides the technical definitions needed for evaluating whether a particular test procedure complies with MLRsearch specification.

Some terms used in the specification are capitalized. It is just a stylistic choice for this document, reminding the reader this term is introduced, defined or explained elsewhere in the document. See Index (Section 12) for list of such terms. Lowercase variants are equally valid.

Each per term subsection contains a short *Definition* paragraph containing a minimal definition and all strict requirements, followed by *Discussion* paragraphs focusing on important consequences and recommendations. Requirements ~~on the way~~about how other components can use the defined quantity are also ~~present~~ included in the discussion ~~paragraphs~~.

~~Other text in this section discusses document structure and non-authoritative summaries.~~

4.1. Overview

MLRsearch Specification describes a set of abstract system components, acting as functions with specified inputs and outputs.

A test procedure is said to comply with MLRsearch Specification if it

Commenté [MB45]: ??

Commenté [MB46]: ??

Commenté [MB47]: Please move this to the terminology section where we can group all conventions used in the document.

Commenté [MB48]: Not sure this brings much

can be conceptually divided into analogous components, each satisfying requirements for the corresponding MLRsearch component. Any such compliant test procedure is called a MLRsearch Implementation.

The Measurer component is tasked to perform Trials, the Controller component is tasked to select Trial Durations and Loads, the Manager component is tasked to pre-configure ~~everything~~ involved entities and to produce the test report. The test report explicitly states Search Goals (as Controller inputs) and corresponding Goal Results (Controller outputs).

The Manager calls ~~the~~ a Controller once, the Controller keeps calling the Measurer until all stopping conditions are met.

The part where a Controller calls the Measurer is called the Search. Any activity ~~done~~ undertaken by the Manager before it calls the Controller (or after Controller returns) is not considered to be part of the Search.

MLRsearch Specification prescribes regular search results and recommends their stopping conditions. Irregular search results are also allowed, they may have different requirements and stopping conditions.

Search results are based on Load Classification. When measured enough, ~~any~~ a chosen Load can either achieve or fail each Search Goal (separately), thus becoming a Lower Bound or an Upper Bound for that Search Goal~~, respectively~~.

For repeatability and comparability ~~reasons~~purposes, it is important that all implementations of MLRsearch classify the Load equivalently, based on all Trials measured at that Load.

When the Relevant Lower Bound is close enough to Relevant Upper Bound according to Goal Width, the Regular Goal Result is found. Search stops when all Regular Goal Results are found, or when some Search Goals are proven to have only Irregular Goal Results.

4.1.1. Behavior Correctness

MLRsearch Specification by itself does not guarantee that the Search ends in finite time, as the freedom the Controller has for Load selection also allows for clearly deficient choices.

~~Although the authors believe that any MLRsearch Implementation that aims to shorten the Search Duration (with fixed Controller Input) will necessarily also become good at repeatability and comparability, any attempts to prove such claims are outside of the scope of this document.~~

For deeper insights on these matters, ~~see~~ refer to [FDio-CSIT-MLRsearch].

The primary MLRsearch Implementation, used as the prototype for this

Commenté [MB49]: Invoke?

Maybe better to clarify what is actually meant by «calls».

Commenté [MB50]: Is there only one? Always?

Include a provision to have many

Commenté [MB51]: Does this also cover «abort» (before completion» to handle some error conditions? Or this is more a «stop execution»?

Commenté [MB52]: Do we have taxonomoy/means to make that equivalence easy to put in place?

Commenté [MB53]: I suggest we be factual and avoid use of «believe» and so on.

specification, is [PyPI-MLRsearch].

4.2. Quantities

MLRsearch specification uses a number of specific quantities, some of them can be expressed in several different units.

~~In general,~~ MLRsearch specification does not require particular units to be used, but it is REQUIRED for the test report to state all the units. For example, ratio quantities can be dimensionless numbers between zero and ~~one, but~~one but may be expressed as percentages instead.

For convenience, a group of quantities can be treated as a composite quantity. One constituent of a composite quantity is called an attribute, and a group of attribute values is called an instance of that composite quantity.

Some attributes ~~are~~ may be ~~not independent~~depend on ~~from~~ others, and they can be calculated from other attributes. Such ~~quantites~~quantities are called derived quantities.

4.2.1. Current and Final Values

Some ~~quantites~~quantities are defined ~~in a way~~so ~~that~~ it is possible to ~~allows computing~~ compute their values in the middle of ~~the~~ a Search. Other quantities are specified ~~in a way~~so that ~~allows~~ their values can ~~to~~ be computed only after ~~the~~a Search ends. ~~And s~~Some quantities are important only after ~~the~~ a Search ended, but their values are computable also before ~~the~~ a Search ends.

For a quantity that is computable before ~~the~~ a Search ends, the adjective *current* is used to mark a value of that quantity available before the Search ends. When such value is relevant for the search result, the adjective *final* is used to denote the value of that quantity at the end of the Search.

If a time evolution of such a dynamic quantity is guided by configuration quantities, those adjectives can be used to distinguish quantities. For ~~example~~example, if the current value of "duration" (dynamic quantity) increases from "initial duration" to "final duration" (configuration quantities), all the quoted names denote separate but related ~~quantites~~quantities. As the naming suggests, the final value od "duration" is expected to be equal to "final duration" value.

4.3. Existing Terms

This specification relies on the following three documents that should be consulted before attempting to make use of this document:

* ~~RFC 1242~~ "Benchmarking Terminology for Network Interconnect Devices" [RFC1242] contains basic term definitions.

Commenté [MB54]: «S» is used in the previous section,

Please pick one form and be consistent through the document.

Commenté [MB55]: Please check

* ~~RFC 2285~~ "Benchmarking Terminology for LAN Switching Devices" [RFC2285] adds
more terms and discussions, describing some known network benchmarking situations in a more precise way.

* ~~RFC 2544~~ "Benchmarking Methodology for Network Interconnect Devices" [RFC2544] contains discussions of a number of terms and additional
methodology requirements.

Definitions of some central terms from above documents are copied and discussed in the following subsections.

4.3.1. SUT

Defined in [RFC2285] (Section 3.1.2) as follows.

Definition:

The collective set of network devices to which stimulus is offered as a single entity and response measured.

Discussion:

An SUT consisting of a single network device is also allowed.

4.3.2. DUT

Defined in Section 3.1.1 of [RFC2285] ~~(Section 3.1.1)~~ as follows.

Definition:

The network forwarding device to which stimulus is offered and response measured.

Discussion:

DUT, as a sub-component of SUT, is only indirectly mentioned in MLRsearch ~~specification, but~~specification but is of key relevance for its motivation.

4.3.3. Trial

A trial is the part of the test described in Section 23 of [RFC2544] ~~(Section 23)~~.

Definition:

A particular test consists of multiple trials. Each trial returns one piece of information, for example the loss rate at a particular input frame rate. Each trial consists of a number of phases:

a) If the DUT is a router, send the routing update to the "input" port and pause two seconds to be sure that the routing has settled.

b) Send the "learning frames" to the "output" port and wait 2 seconds to be sure that the learning has settled. Bridge learning frames are frames with source addresses that are the same as the destination

Commenté [MB56]: I would delete.

Commenté [MB57]: Please move this to a terminology section suggested above

Commenté [MB58]: Do we need to include this?

I would only introduce deviation from bas specs.

Commenté [MB59]: This reasons about «device», should we say that we extends this to «function»?

Commenté [MB60]: Idem as SUT

addresses used by the test frames. Learning frames for other protocols are used to prime the address resolution tables in the DUT. The formats of the learning frame that should be used are shown in the Test Frame Formats document.

c) Run the test trial.

d) Wait for two seconds for any residual frames to be received.

e) Wait for at least five seconds for the DUT to restabilize.

Discussion:

The traffic is sent only in phase c) and received in phases c) and d).

The definition describes some traits, and it is not clear whether all of them are required, or some of them are only recommended.

Trials are the only stimuli the SUT is expected to experience during the Search.

For the purposes of the MLRsearch specification, it is ~~ALLOWED~~ allowed for the test procedure to deviate from the [RFC2544] description, but any such deviation MUST be described explicitly in the test report.

In some discussion paragraphs, it is useful to consider the traffic as sent and received by a tester, as implicitly defined in Section 6 of [RFC2544] ~~(Section 6)~~.

An example of deviation from [RFC2544] is using shorter wait times, compared to those described in phases a), b), d) and e).

~~The~~ [RFC2544] ~~document itself~~ seems to be treating phase b) as any type of configuration that cannot be configured only once (by Manager, before Search starts), as some crucial SUT state could time-out during the Search. ~~This document~~It is ~~RECOMMENDS~~ RECOMMENDED ~~to understand~~understanding "learning frames" to be any such time-sensitive per-trial configuration method, with bridge MAC learning being only one ~~possibe~~possible example. Appendix C.2.4.1 of [RFC2544] ~~(Section C.2.4.1)~~ lists another example: ARP with wait time 5 seconds.

4.4. Trial Terms

This section defines new and redefine existing terms for quantities relevant as inputs or outputs of a Trial, as used by the Measurer component. This includes also any derived quantities related to one trial result.

4.4.1. Trial Duration

Definition:

Commenté [MB61]: Is there any aspect new to MLRS?

Commenté [MB62]: Not a normative langiage

Commenté [MB63]: Not a normative term

Trial Duration is the intended duration of the phase c) of a Trial.

Discussion:

While any positive real value may be provided, some Measurer implementations MAY limit possible values, e.g., ~~e.g.~~ by rounding down to nearest integer in seconds. In that case, it is RECOMMENDED to give such inputs to the Controller so that the Controller only proposes the accepted values.

4.4.2. Trial Load

Definition:

Trial Load is the per-interface Intended Load for a Trial.

Discussion:

For test report purposes, it is assumed that this is a constant load by default, as specified in Section 3.4 of [RFC1242] ~~(Section 3.4)~~.

Trial Load MAY be ~~only~~ an average load, e.g., when the traffic is intended to be bursty, e.g. as suggested in Section 21 of [RFC2544] ~~(Section 21)~~.
In the case of a non-constant load, the test report MUST explicitly mention how exactly non-constant the traffic is.

Trial Load is equivalent to the quantities defined as constant load (Section 3.4 of [RFC1242] ~~(Section 3.4)~~, data rate ~~of~~ (Section 14 of [RFC2544] ~~(Section 14)~~, and Intended Load (Section 3.5.1 of [RFC2285] ~~(Section 3.5.1)~~, in the sense that all three definitions specify that this value applies to one (input or output) interface.

Similarly to Trial Duration, some Measurers may limit the possible values of trial load. Contrary to trial duration, the test report is not ~~REQUIRED~~ required to document such behavior. This is because~~, as in practice~~ the load differences are negligible (and frequently undocumented) in practice.

It is ~~ALLOWED~~ allowed to combine Trial Load and Trial Duration values in a way that would not be possible to achieve using any integer number of data frames.

If a particular Trial Load value is not tied to a single Trial, e.g.,~~e.g.~~ if there are no Trials yet or if there are multiple Trials, this document uses a shorthand *Load*.

For test report purposes, multi-interface aggregate load MAY be reported, and is understood as the same quantity expressed using different units. From the report it MUST be clear whether a particular Trial Load value is per ~~one~~ an interface, a set of interfaces (e.g., all logical interfaces bound to the same port), or an aggregate

Commenté [MB64]: Does this cover also recurrences? See, e.g., draft-ietf-netmod-schedule-yang-05 - A Common YANG Data Model for Scheduling or draft-ietf-opsawg-scheduling-oam-tests-00?

Commenté [MB65]: To?

Commenté [MB66]: Please fix all similar ones in the doc

Commenté [MB67]: An example of an example :-)

Please reword.

Commenté [MB68]: Can we also cover load percentiles?

The avg may not be representative to stress functions with anti-ddos guards, for example.

Commenté [MB69]: Inappropriate use of normative language

over all interfaces. This implies that there is a known and constant coefficient between single-interface and multi-interface load values. The single-interface value is ~~still~~ the primary one, as most other documents deal with single-interface ~~quantites~~quantities only.

The last paragraph also applies to other terms related to Load.

4.4.3. Trial Input

Definition:

Trial Input is a composite quantity, consisting of two attributes: Trial Duration and Trial Load.

Discussion:

When talking about multiple Trials, it is common to say "Trial Inputs" to denote all corresponding Trial Input instances.

A Trial Input instance acts as the input for one call of the Measurer component.

Contrary to other composite quantities, MLRsearch Implementations are ~~NOT ALLOWED~~MUST NOT ~~to~~ add optional attributes here. This improves interoperability between various implementations of ~~the~~ a Controller and ~~the~~ a Measurer.

~~Please~~ Nnote that both attributes are *intended* quantities, as only those can be fully controlled by the Controller. The actual offered quantities, as realized by the Measurer, can be different (and must be different if not multiplying into integer number of frames), but questions around those offered quantities are generally outside of the scope of this document.

4.4.4. Traffic Profile

Definition:

Traffic Profile is a composite quantity containing all attributes other than Trial Load and Trial Duration, that are needed for unique determination of the Trial to be performed.

Discussion:

All the attributes are assumed to be constant during the search, and the composite is configured on the Measurer by the Manager before the Search starts. This is why the traffic profile is not part of the Trial Input.

~~As a consequence~~Therefore, implementations of the Manager and the Measurer must be aware of their common set of capabilities, so that Traffic Profile instance uniquely defines the traffic during the Search. ~~The important fact is that n~~None of those capabilities have to be known by the Controller implementations.

The Traffic Profile SHOULD contain some specific quantities defined

Commenté [MB70]: The causality effect may not be evident for the subset case, at least.

Commenté [MB71]: Which ones?

Commenté [MB72]: Can we provide an example how to make that?

elsewhere. For ~~example~~example, Section 9 of [RFC2544] ~~(Section 9)~~ governs data link frame
sizes as defined in Section 3.5 of [RFC1242] ~~(Section 3.5)~~.

~~Several more~~More specific quantities may be ~~RECOMMENDED~~recommended, depending on
media type. For example, Appendix C of [RFC2544] ~~(Appendix C)~~ lists frame formats
and protocol addresses, as recommended in Section 8 of [RFC2544] ~~(Section 8)~~ and
Section 12 of [RFC2544] ~~(Section 12)~~.

Depending on a SUT configuration ~~(~~, e.g.~~,~~ when testing specific protocols~~)~~,
additional attributes ~~MUST~~ may be included in the traffic profile and in
the test report.

Example: [RFC8219] (Section 5.3) introduces traffic setups consisting of a mix of IPv4 and IPv6 traffic - the implied traffic profile ~~therefore~~therefore, must include an attribute for their percentage.

Other traffic properties that need to be somehow specified in Traffic Profile, if they apply to the test scenario, include:

* bidirectional traffic from Section 14 of [RFC2544] ~~(Section 14)~~,

* fully meshed traffic from Section 3.3.3 of [RFC2285] ~~(Section 3.3.3)~~, or

* modifiers from Section 11 of [RFC2544] ~~(Section 11)~~.

4.4.5. Trial Forwarding Ratio

Definition:

The Trial Forwarding Ratio is a dimensionless ~~floating point~~floating-point value.
It MUST range between 0.0 and 1.0, both inclusive.

It is calculated
by dividing the number of frames successfully forwarded by the SUT by the total number of frames expected to be forwarded during the trial.

Discussion:

For most Traffic Profiles, "expected to be forwarded" means "intended to get transmitted from tester towards SUT". Only if this is not the case, the test report ~~MUST~~ SHOULD describe the Traffic Profile in a way that
implies how Trial Forwarding Ratio should be calculated.

Trial Forwarding Ratio MAY be expressed in other units (~~e.g.~~e.g., as a Percentage) in the test report.

Note that, contrary to Load terms, frame counts used to compute Trial Forwarding Ratio are generally aggregates over all SUT output

Commenté [MB73]: This is too vague. Unless we reword top better reflect the requirement, I don't think we can use the normative language here

Commenté [MB74]: Inappropriate use of normative language

Commenté [MB75]: Idem as above. MUST is not appropriate here.

Commenté [MB76]: MUST is an absolute requirement (i.e., there is no exception):

1. **MUST** This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.

SHOULD This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

interfaces, as most test procedures verify all ~~outgoung~~outgoing frames.

For example, in a test with symmetric bidirectional traffic, if one direction is forwarded without losses, but the opposite direction does not forward at all, the trial forwarding ratio would be 0.5 (50%).

4.4.6. Trial Loss Ratio

Definition:

The Trial Loss Ratio is equal to one minus the Trial Forwarding Ratio.

Discussion:

100% minus the Trial Forwarding Ratio, when expressed as a percentage.

This is almost identical to Frame Loss Rate (Section 3.6 of [RFC1242] ~~(Section 3.6)~~). The only ~~minor~~ differences are that Trial Loss Ratio does not need to be expressed as a percentage~~,~~ and Trial Loss Ratio is explicitly based on aggregate frame counts.

4.4.7. Trial Forwarding Rate

Definition:

The Trial Forwarding Rate is a derived quantity, calculated by multiplying the Trial Load by the Trial Forwarding Ratio.

Discussion:

~~It is important to note that while similar, t~~This quantity is not identical to the Forwarding Rate as defined in Section 3.6.1 if [RFC2285] ~~(Section 3.6.1)~~. Concretely, ~~The~~ the latter is based on frame counts on one output interface only, so each output interface can have different forwarding rate, whereas the Trial Forwarding Rate is based on frame counts aggregated over all SUT output interfaces, while ~~still~~still being a multiple of Load.

Consequently, for symmetric bidirectional Traffic Profiles, the Trial Forwarding Rate value is equal to the arithmetic average of [RFC2285] Forwarding Rate values across both output interfaces.

Given that Trial Forwarding Rate is a quantity based on Load, ~~it is ALLOWED to express~~ this quantity may be expressed using multi-interface values in test Report (~~,~~ e.g., as sum of per-interface forwarding rate values).

Commenté [MB77]: Should we call for more granularity to be provided/characterized?

a mis en forme : Retrait : Gauche : 1,25 cm

a mis en forme : Retrait : Gauche : 1,25 cm

Commenté [MB78]: For all sections, please indent so that we separate the def/discussion vs. description.

Commenté [MB79]: Do we have an authoritative reference where this is defined? If not, please add an definition entry early in the terminology section.

Commenté [MB80]: Why both?

4.4.8. Trial Effective Duration

Definition:

Trial Effective Duration is a time quantity related to the trial, by default equal to the Trial Duration.

Discussion:

This is an optional feature. If the Measurer does not return any Trial Effective Duration value, the Controller MUST use the Trial Duration value instead.

Trial Effective Duration may be any time quantity chosen by the Measurer to be used for time-based decisions in the Controller.

The test report MUST explain how the Measurer computes the returned Trial Effective Duration values, if they are not always equal to the Trial Duration.

This feature can be beneficial for users who wish to manage the overall search duration, rather than solely the traffic portion of it. An approach is ~~Simply~~ to measure the duration of the whole trial (including all wait times) and use that as the Trial Effective Duration.

This is also a way for the Measurer to inform the Controller about its surprising behavior, for example, ~~for example~~ when rounding the Trial Duration value.

4.4.9. Trial Output

Definition:

Trial Output is a composite quantity. The ~~REQUIRED~~ required attributes to characterize the output are Trial Loss Ratio, Trial Effective Duration, and Trial Forwarding Rate.

Discussion:

Trial Output instance refer to the output of a given trial, when several such trials are in place~~When talking about multiple trials, it is common to say "Trial Outputs" to denote all corresponding Trial Output instances~~.

Implementations may provide additional ~~(~~optional~~)~~ attributes. The Controller implementations ~~MUST~~ SHOULD ignore values of any optional attribute they are not familiar with, except when passing Trial Output instances to the Manager.

Example of an optional attribute: The aggregate number of frames expected to be forwarded during the trial, especially if it is not ~~just~~ (a rounded-down value) implied by Trial Load and Trial Duration.

While Section 3.5.2 of [RFC2285] ~~(Section 3.5.2)~~ requires the Offered Load value to be

Commenté [MB81]: For the periodic/recurrences, does it cover only one recurrence or from start to last independent of in-between execution periods?

Commenté [MB82]: It is obvious, but should we say «positive»?

Commenté [MB83]: To be defined early in the terminology section

Commenté [MB84]: As we have an exception

reported for forwarding rate measurements, it is not ~~REQUIRED~~ required in MLRsearch Specification, as search results do not depend on it.

4.4.10. Trial Result

Definition:

Trial Result is a composite quantity, consisting of the Trial Input and the Trial Output.

Discussion:

Trial Result instance refers to ~~When talking about multiple trials, it is common to say~~ "Trial Results" of a trial, when multiple ~~to denote all corresponding Trial Result instances~~trials are into effect.

While implementations SHOULD NOT include additional attributes with independent values, they MAY include derived quantities.

Commenté [MB85]: Can we include a short sentence to explain the risk if not followed?

4.5. Goal Terms

This section defines new terms for quantities relevant (directly or indirectly) for inputs and outputs of the Controller component.

Several goal attributes are defined before introducing the main composite quantity: the Search Goal.

Contrary to other sections, definitions in subsections of this section are necessarily vague, as their fundamental meaning is to act as coefficients in formulas for Controller Output, which are not defined yet.

The discussions ~~here~~ in this section relate the attributes to concepts mentioned in ~~chapter Identified Problems (~~Section 3~~)~~, but even these discussion paragraphs are short, informal, and mostly referencing later sections, where the impact on search results is discussed after introducing the complete set of auxiliary terms.

4.5.1. Goal Final Trial Duration

Definition:

Minimal value for Trial Duration that ~~has to~~must be reached. The value MUST be positive.

Discussion:

Some Trials have to be at least this long to allow a Load to be classified as a Lower Bound. The Controller ~~is allowed to~~may choose shorter durations, results of those may be enough for classification as an Upper Bound.

Commenté [MB86]: I don't parse this.

It is RECOMMENDED for all search goals to share the same Goal Final Trial Duration value. Otherwise, Trial Duration values larger than

the Goal Final Trial Duration may occur, weakening the assumptions the Load Classification Logic (Section 6.1) is based on.

4.5.2. Goal Duration Sum

Definition:

A threshold value for a particular sum of Trial Effective Duration values. The value MUST be positive.

Discussion:

Informally, this prescribes the sufficient ~~amount~~number of trials performed at a specific Trial Load and Goal Final Trial Duration during the search.

If the Goal Duration Sum is larger than the Goal Final Trial Duration, multiple trials may be needed to be performed at the same load.

~~See section MLRsearch Compliant with TST009 (~~Refer to Section 4.9.3~~) of this document~~ for an example where the possibility of multiple trials at the same load is intended.

A Goal Duration Sum value shorter than the Goal Final Trial Duration (of the same goal) could save some search time, but is NOT RECOMMENDED, as the time savings come at the cost of decreased repeatability.

In practice, the Search can spend less than Goal Duration Sum measuring a Load value when the results are particularly one-sided, but ~~also~~also, the Search can spend more than Goal Duration Sum measuring a Load when the results are balanced and include trials shorter than Goal Final Trial Duration.

4.5.3. Goal Loss Ratio

Definition:

A threshold value for Trial Loss Ratio values. The value MUST be non-negative and smaller than one.

Discussion:

A trial with Trial Loss Ratio larger than this value signals the SUT may be unable to process this Trial Load well enough.

See Throughput with Non-Zero Loss (Section 3.4) for reasons why users may want to set this value above zero.

Since multiple trials may be needed for one Load value, the Load Classification ~~is generally~~may be more complicated than mere comparison of Trial Loss Ratio to Goal Loss Ratio.

Commenté [MB87]: I like this, but we should be consistent and mention it when appropriate for all other metrics

4.5.4. Goal Exceed Ratio

Definition:

A threshold value for a particular ratio of sums of Trial Effective Duration values. The value MUST be non-negative and smaller than one.

Discussion:

Informally, up to this proportion of Trial Results with Trial Loss Ratio above Goal Loss Ratio is tolerated at a Lower Bound. This is the full impact if every Trial was measured at Goal Final Trial Duration. The actual full logic is more complicated, as shorter Trials are allowed.

For explainability reasons, the RECOMMENDED value for exceed ratio is 0.5 (50%), as in practice that value leads to the smallest variation in overall Search Duration.

~~See Exceed Ratio and Multiple Trials (~~Refer to Section 5.4~~)~~ ~~section~~ for more details.

4.5.5. Goal Width

Definition:

A threshold value for deciding whether two Trial Load values are close enough. This is an OPTIONAL attribute. If present, the value MUST be positive.

Discussion:

Informally, this acts as a stopping condition, controlling the precision of the search result. The search stops if every goal has reached its precision.

Implementations without this attribute MUST ~~give~~ provide the Controller with other ~~ways~~ means to control the search stopping conditions.

Absolute load difference and relative load difference are two popular choices, but implementations may choose a different way to specify width.

The test report MUST make it clear what specific quantity is used as Goal Width.

It is RECOMMENDED to set the Goal Width (as relative difference) value to a value no lower than the Goal Loss Ratio. ~~If the reason is not obvious, see the details in Generalized Throughput (~~Refer to Section 5.6 for more elaboration on the reasoning~~)~~.

4.5.6. Goal Initial Trial Duration

Definition:

Minimal value for Trial Duration suggested to use for this goal. If present, this value MUST be positive.

Discussion:

This is an example of an ~~OPTIONAL~~ optional Search Goal ~~some implementations may support~~.

A typical ~~The reasonable~~ default value is equal to the Goal Final Trial Duration value.

Informally, this is the shortest Trial Duration the Controller should select when focusing on the goal.

Note that shorter Trial Duration values can still be used, for ~~Example~~example, selected while focusing on a different Search Goal. Such results MUST be still accepted by the Load Classification logic.

Goal Initial Trial Duration is ~~just a way~~a mechanism for ~~the~~ a user to discourage trials with Trial Duration values deemed as too unreliable for a particular SUT and a given ~~this~~ Search Goal.

4.5.7. Search Goal

Definition:

The Search Goal is a composite quantity consisting of several attributes, some of them are required.

Required attributes: ~~-~~ Goal Final Trial Duration, ~~-~~ Goal Duration Sum ~~-~~ Goal Loss Ratio, and ~~-~~ Goal Exceed Ratio

Optional attributes: ~~-~~ Goal Initial Trial Duration ~~-~~ and Goal Width

Discussion:

Implementations MAY add their own attributes. Those additional attributes may be required by ~~the~~ an implementation even if they are not required by MLRsearch specification. ~~But~~ However, it is RECOMMENDED for those implementations to support missing attributes by providing ~~reasonable~~typical default values.

For example, implementations with Goal Initial Trial Durations may also require users to specify "how quickly" should Trial Durations increase.

~~See~~ Refer to Compliance (Section 4.9) for important Search Goal instances.

4.5.8. Controller Input

Commenté [MB88]: Listing the attributes this way allows to easily classify mandatory/optional. However, this not followed in previous. Please pick your favorite approach and use it in a consistent manner in the document.

Commenté [MB89]: I guess I understand what is meant here, but I think this should be reworded to avoid what can be seen as inconsistency: do not support vs. support a default.

Definition:

Controller Input is a composite quantity required as an input for the Controller. The only REQUIRED attribute is a list of Search Goal instances.

Discussion:

MLRsearch Implementations MAY use additional attributes. Those additional attributes may be required by ~~the~~ an implementation even if they are not required by MLRsearch ~~specification~~Specification.

Formally, the Manager does not apply any Controller configuration apart from one Controller Input instance.

For example, Traffic Profile is configured on the Measurer by the Manager, without explicit assistance of the Controller.

The order of Search Goal instances in a list SHOULD NOT have a big impact on Controller Output, but MLRsearch Implementations MAY base their behavior on the order of Search Goal instances in a list.

4.5.8.1. Max Load

Definition:

Max Load is an optional attribute of Controller Input. It is the maximal value the Controller is allowed to use for Trial Load values.

Discussion:

Max Load is an example of an optional attribute (outside the list of Search Goals) required by some implementations of MLRsearch.

In theory, each search goal could have its own Max Load value, but as all Trial Results are possibly affecting all Search Goals, it makes more sense for a single Max Load value to apply to all Search Goal instances.

While Max Load is a frequently used configuration parameter, already governed (as maximum frame rate) by [RFC2544] (Section 20) and (as maximum offered load) by [RFC2285] (Section 3.5.3), some implementations may detect or discover it (instead of requiring a user-supplied value).

In MLRsearch specification, one reason for listing the Relevant Upper Bound (Section 4.7.1) as a required attribute is that it makes the search result independent of Max Load value.

Given that Max Load is a quantity based on Load, it is ~~ALLOWED~~ allowed to express this quantity using multi-interface values in test report, e.g.,~~e.g.~~ as sum of per-interface maximal loads.

4.5.8.2. Min Load

Definition:

Min Load is an optional attribute of Controller Input. It is the minimal value the Controller is allowed to use for Trial Load values.

Discussion:

Min Load is another example of an optional attribute required by some implementations of MLRsearch. Similarly to Max Load, it makes more sense to prescribe one common value, as opposed to using a different value for each Search Goal.

Min Load is mainly useful for saving time by failing early, arriving at an Irregular Goal Result when Min Load gets classified as an Upper Bound.

For implementations, it is RECOMMENDED to require Min Load to be non-zero and large enough to result in at least one frame being forwarded even at shortest allowed Trial Duration, so that Trial Loss Ratio is always well-defined, and the implementation can apply relative Goal Width safely.

Given that Min Load is a quantity based on Load, it is ~~ALLOWED~~ allowed to express this quantity using multi-interface values in test report, e.g., ~~e.g.~~ as sum of per-interface minimal loads.

4.6. Auxiliary Terms

While the terms defined in this section are not strictly needed when formulating MLRsearch requirements, they simplify the language used in discussion paragraphs and explanation sections~~chapters~~.

4.6.1. Trial Classification

When one Trial Result instance is compared to one Search Goal instance, several relations can be named using short adjectives.

As trial results do not affect each other, this *Trial Classification* does not change during ~~the~~ a Search.

4.6.1.1. High-Loss Trial

A trial with Trial Loss Ratio larger than a Goal Loss Ratio value is called a *high-loss trial*, with respect to given Search Goal (or lossy trial, if Goal Loss Ratio is zero).

4.6.1.2. Low-Loss Trial

If a trial is not high-loss, it is called a *low-loss trial* (or zero-loss trial, if Goal Loss Ratio is zero).

4.6.1.3. Short Trial

A trial with Trial Duration shorter than the Goal Final Trial Duration is called a *short trial* (with respect to the given Search Goal).

4.6.1.4. Full-Length Trial

A trial that is not short is called a *full-length* trial.

Note that this includes Trial Durations larger than Goal Final Trial Duration.

4.6.1.5. Long Trial

A trial with Trial Duration longer than the Goal Final Trial Duration is called a *long trial*.

4.6.2. Load Classification

When a set of all Trial Result instances, performed so far at one Trial Load, is compared to one Search Goal instance, their relation can be named using the concept of a bound.

In general, such bounds are a current quantity, even though cases of a Load changing its classification more than once during the Search is rare in practice.

4.6.2.1. Upper Bound

Definition:

A Load value is called an Upper Bound if and only if it is classified as such by Appendix A~~: Load Classification (Section 10)~~ algorithm for the given Search Goal at the current moment of the Search.

Discussion:

In more detail, the set of all Trial Result instances performed so far at the Trial Load (and any Trial Duration) is certain to fail to uphold all the requirements of the given Search Goal, mainly the Goal Loss Ratio in combination with the Goal Exceed Ratio. ~~Here~~ In this context, "certain to fail" relates to any possible results within the time remaining till Goal Duration Sum.

One search goal can have multiple different Trial Load values classified as its Upper Bounds. While search progresses and more trials are measured, any load value can become an Upper Bound in principle.

Moreover, a load can stop being an Upper Bound, but that can only happen when more than Goal Duration Sum of trials are measured (e.g.,~~e.g.~~ because another Search Goal needs more trials at this load). In practice, the load becomes a Lower Bound (~~see next subsection~~Section 4.6.2.2), and we say the previous Upper Bound got Invalidated.

4.6.2.2. Lower Bound

Definition:

A Load value is called a Lower Bound if and only if it is classified as such by Appendix A~~: Load Classification (Section 10)~~ algorithm for the given Search Goal at the current moment of the search.

Discussion:

In more detail, the set of all Trial Result instances performed so far at the Trial Load (and any Trial Duration) is certain to uphold all the requirements of the given Search Goal, mainly the Goal Loss Ratio in combination with the Goal Exceed Ratio. Here "certain to uphold" relates to any possible results within the time remaining till Goal Duration Sum.

One search goal can have multiple different Trial Load values classified as its Lower Bounds. As search progresses and more trials are measured, any load value can become a Lower Bound in principle.

No load can be both an Upper Bound and a Lower Bound for the same Search goal at the same time, but it is possible for a larger load to be a Lower Bound while a smaller load is an Upper Bound.

Moreover, a load can stop being a Lower Bound, but that can only happen when more than Goal Duration Sum of trials are measured (e.g.,~~e.g.~~ because another Search Goal needs more trials at this load). In that case, the load becomes an Upper Bound, and we say the previous Lower Bound got Invalidated.

4.6.2.3. Undecided

Definition:

A Load value is called Undecided if it is currently neither an Upper Bound nor a Lower Bound.

Discussion:

A Load value that has not been measured so far is Undecided.

It is possible for a Load to transition from an Upper Bound to Undecided by adding Short Trials with Low-Loss results. That is yet another reason for users to avoid using Search Goal instances with ~~diferent~~different Goal Final Trial Duration values.

4.7. Result Terms

Before defining the full structure of a Controller Output, it is useful to define the composite quantity, called Goal Result. The following subsections define its attribute first, before describing the Goal Result quantity.

There is a correspondence between Search Goals and Goal Results. Most of the following subsections refer to a given Search Goal, when defining their terms. Conversely, at the end of the search, each

Search Goal instance has its corresponding Goal Result instance.

4.7.1. Relevant Upper Bound

Definition:

The Relevant Upper Bound is the smallest Trial Load value classified as an Upper Bound for a given Search Goal at the end of the Search.

Discussion:

If no measured load had enough High-Loss Trials, the Relevant Upper Bound MAY be non-existent. For example, when Max Load is classified as a Lower Bound.

Conversely, when Relevant Upper Bound does exist, it is not affected by Max Load value.

Given that Relevant Upper Bound is a quantity based on Load, it is ~~ALLOWED~~ allowed to express this quantity using multi-interface values in test report, e.g., ~~e.g.~~ as sum of per-interface loads.

4.7.2. Relevant Lower Bound

Definition:

The Relevant Lower Bound is the largest Trial Load value among those smaller than the Relevant Upper Bound, that got classified as a Lower Bound for a given Search Goal at the end of the search.

Discussion:

If no load had enough Low-Loss Trials, the Relevant Lower Bound MAY be non-existent.

Strictly speaking, if the Relevant Upper Bound does not exist, the Relevant Lower Bound also does not exist. In a typical case, Max Load is classified as a Lower Bound, making it impossible to increase

the Load to continue the search for an Upper Bound. Thus, it is not clear whether a larger value would be found for a Relevant Lower Bound if larger Loads were possible.

Given that Relevant Lower Bound is a quantity based on Load, it is ~~ALLOWED~~ allowed to express this quantity using multi-interface values in test report, e.g.,~~e.g.~~ as sum of per-interface loads.

4.7.3. Conditional Throughput

Definition:

Conditional Throughput is a value computed at the Relevant Lower Bound according to algorithm defined in Appendix B~~: Conditional Throughput (Section 11)~~.

Discussion:

The Relevant Lower Bound is defined only at the end of the Search, and so is the Conditional Throughput. But the algorithm can be applied at any time on any Lower Bound load, so the final Conditional Throughput value may appear sooner than at the end of ~~the~~ a Search.

Informally, the Conditional Throughput should be a typical Trial Forwarding Rate, expected to be seen at the Relevant Lower Bound of ~~the~~ a given Search Goal.

But frequently it is only a conservative estimate thereof, as MLRsearch Implementations tend to stop measuring more Trials as soon as they confirm the value cannot get worse than this estimate within the Goal Duration Sum.

This value is RECOMMENDED to be used when evaluating repeatability and comparability of different MLRsearch Implementations.

~~See~~ Refer to ~~Generalized Throughput (~~Section 5.6~~)~~ for more details.

Given that Conditional Throughput is a quantity based on Load, it is ~~ALLOWED~~ allowed to express this quantity using multi-interface values in a test report, e.g.~~,~~ as sum of per-interface ~~foerwarding~~forwarding rates.

4.7.4. Goal Results

MLRsearch specification is based on a set of requirements for a "regular" result. But in practice, it is not always possible for such result instance to exist, so also "irregular" results need to be supported.

4.7.4.1. Regular Goal Result

Definition:

Regular Goal Result is a composite quantity consisting of several attributes. Relevant Upper Bound and Relevant Lower Bound are REQUIRED attributes.~~,~~ Conditional Throughput is a RECOMMENDED attribute. Stopping conditions for the corresponding Search Goal MUST be satisfied.

Discussion:

Both relevant bounds MUST exist.

If ~~the~~ an implementation offers Goal Width as a Search Goal attribute, the distance between the Relevant Lower Bound and the Relevant Upper Bound MUST NOT be larger than the Goal Width,

Implementations MAY add their own attributes.

Test report MUST display Relevant Lower Bound. Displaying Relevant Upper Bound is ~~not REQUIRED, but it is~~ RECOMMENDED, especially if the implementation does not use Goal Width.

4.7.4.2. Irregular Goal Result

Commenté [MB90]: To do what.?

I'm afraid we need to explicit the meaning here.

Commenté [MB91]: Isn't this redundant with listing the bounds as required in the previous definition?

Definition:

Irregular Goal Result is a composite quantity. No attributes are required.

Discussion:

It is RECOMMENDED to report any useful quantity even if it does not satisfy all the requirements. For ~~example~~example, if Max Load is classified as a Lower Bound, it is fine to report it as an "effective" Relevant Lower Bound (although not a real one, as that requires Relevant Upper Bound which does not exist in this case), and compute Conditional Throughput for it. In this case, only the missing Relevant Upper Bound signals this result instance is irregular.

Similarly, if both ~~revevant~~relevant bounds exist, it is RECOMMENDED to include them as Irregular Goal Result attributes, and let the Manager decide if their distance is too far for users' purposes.

If test report displays some Irregular Goal Result attribute values, they MUST be clearly marked as ~~comming~~coming from irregular results.

The implementation MAY define additional attributes.

4.7.4.3. Goal Result

Definition:

Goal Result is a composite quantity. Each instance is either a Regular Goal Result or an Irregular Goal Result.

Discussion:

The Manager MUST be able to distinguish whether the instance is regular or not.

4.7.5. Search Result

Definition:

The Search Result is a single composite object that maps each Search Goal instance to a corresponding Goal Result instance.

Discussion:

Alternatively, the Search Result can be implemented as an ordered list of the Goal Result instances, matching the order of Search Goal instances.

Commenté [MB92]: To what?

The Search Result ~~(as a mapping)~~ MUST map from all the Search Goal instances present in the associated Controller Input.

Identical Goal Result instances MAY be listed for different Search Goals, but their status as regular or irregular may be different. For ~~example~~example, if two goals differ only in Goal Width value, and the

relevant bound values are close enough according to only one of them.

4.7.6. Controller Output

Definition:

The Controller Output is a composite quantity returned from the Controller to the Manager at the end of the search. The Search Result instance is its only ~~REQUIRED~~ required attribute.

Discussion:

MLRsearch Implementation MAY return additional data in the Controller Output, e.g., ~~for example~~ number of trials performed and the total Search Duration.

4.8. MLRsearch Architecture

MLRsearch architecture consists of three main system components: the Manager, the Controller, and the Measurer.

The architecture also implies the presence of other components~~.~~ such as the SUT and the tester (as a sub-component of the Measurer).

Communication ~~Protocols~~ protocols ~~of communication~~and interfaces between components are ~~generally~~ left unspecified. For example, when MLRsearch ~~specification~~ Specification mentions "Controller calls Measurer", it is possible that the Controller notifies the Manager to call the Measurer indirectly instead. ~~This~~In doing so, ~~way~~ the Measurer Implementations can be fully independent from the Controller implementations, ~~e.g.~~e.g., developed in different programming languages.

4.8.1. Measurer

Definition:

The Measurer is an ~~abstract system component~~functional element that when called with a Trial Input (Section 4.4.3) instance, performs one Trial (Section 4.3.3~~), and~~) and returns a Trial Output (Section 4.4.9) instance.

Discussion:

This definition assumes the Measurer is already initialized. In practice, there may be additional steps before the Search, e.g., ~~e.g.~~ when the Manager configures the traffic profile (either on the Measurer or on its tester sub-component directly) and performs a warm-up (if the tester or the test procedure requires one).

It is the responsibility of the Measurer implementation to uphold any

Commenté [MB93]: I guess these should be introduced before the attributes as these components are used in the description.

Please reconsider the flow of the document.

Commenté [MB94]: Aha, this answers a comment I made earlier :-)

Let's save cycles for other readers and move all this section early in the document.

requirements and assumptions present in MLRsearch specification, e.g.~~e.g.~~
Trial Forwarding Ratio not being larger than one.

Implementers have some freedom. For ~~example~~example, [RFC2544] (Section 10)
gives some suggestions (but not requirements) related to duplicated or reordered frames. Implementations are RECOMMENDED to document their behavior related to such freedoms in as detailed a way as possible.

It is RECOMMENDED to benchmark the test equipment first, e.g.,~~e.g.~~ connect
sender and receiver directly (without any SUT in the path), find a load value that guarantees the Offered Load is not too far from the Intended ~~Load, and~~Load and use that value as the Max Load value.

When
testing the real SUT, it is RECOMMENDED to turn any ~~big~~ severe ~~difference~~deviation
between the Intended Load and the Offered Load into increased Trial Loss Ratio.

Neither of the two recommendations are made into mandatory requirements,
because it is not easy to ~~tell~~ provide guidance about when the difference is ~~big~~ severe enough, in
a way that would be disentangled from other Measurer freedoms.

For a ~~simple example of a~~ sample situation where the Offered Load cannot
keep up with the Intended Load, and the consequences on MLRsearch result, ~~see~~ refer to~~Hard Performance Limit (~~Section 5.6.1~~)~~.

4.8.2. Controller

Definition:

The Controller is an ~~abstract system component~~functional element that when called once
with a Controller Input instance repeatedly computes Trial Input instance for the Measurer, obtains corresponding Trial Output instances, and eventually returns a Controller Output instance.

Commenté [MB95]: Till a stop?

Discussion:

Informally, the Controller has big freedom in selection of Trial Inputs, and the implementations want to achieve all the Search Goals in the shortest average time.

The Controller's role in optimizing the overall Search Duration distinguishes MLRsearch algorithms from simpler search procedures.

Informally, each implementation can have different stopping conditions. Goal Width is only one example. In practice, implementation details do not matter, as long as Goal Result instances are regular.

4.8.3. Manager

Definition:

The Manager is ~~an abstract system component~~functional element that is ~~reponsible~~responsible for ~~configuring~~ provisioning other components, calling ~~the~~ a Controller component once, and for creating the test report following the reporting format as defined in Section 26 of [RFC2544] ~~(Section 26)~~.

Discussion:

The Manager initializes the SUT, the Measurer (and the tester if independent from Measurer) with their intended configurations before calling the Controller.

Note that Section 7 of [RFC2544] ~~(Section 7)~~ already puts requirements on SUT setups:

It is expected that ~~all of~~all the tests will be run without changing the ~~configuration or~~ setup of the DUT in any way other than that required to do the specific test. For example, it is not acceptable to change the size of frame handling buffers between tests of frame handling rates or to disable all but one transport protocol when testing the throughput of that protocol.

It is REQUIRED for the test report to encompass all the SUT configuration details, ~~perhaps~~ including by describing a "default" configuration common for most tests and only describe configuration changes if required by a specific test.

For example, Section 5.1.1 of [RFC5180] ~~(Section 5.1.1)~~ recommends testing jumbo frames if SUT can forward them, even though they are outside the scope of the 802.3 IEEE standard. In this case, it is ~~fair~~ acceptable for the SUT default configuration to not support jumbo frames, and only enable this support when testing jumbo traffic profiles, as the handling of jumbo frames typically has different packet buffer requirements and potentially higher processing overhead. ~~Ideally, n~~Non-jumbo frame sizes should also be tested on the jumbo-enabled setup.

The Manager does not need to be able to tweak any Search Goal attributes, but it MUST report all applied attribute values even if not tweaked.

In principle, there should be a "user" (human or automated) that "starts" or "calls" the Manager and receives the report. The Manager MAY be able to be called more than once ~~whis~~ this way, thus triggering multiple independent Searches.

Commenté [MB96]: This answers a comment I have earlier. Please move all these details to be provided early.

Commenté [MB97]: Should there be a mode where conditional calls are invoked? Or more generally to instruct some dependency?

4.9. Compliance

This section discusses compliance relations between MLRsearch and

other test procedures.

4.9.1. Test Procedure Compliant with MLRsearch

Any networking measurement setup that could be understood as consisting of abstract components satisfying requirements for the Measurer, the Controller, and the Manager, ~~is considered to be~~is compliant with MLRsearch specification.

These components can be seen as abstractions present in any testing procedure. For example, there can be a single component acting both as the Manager and the Controller, but ~~as long as~~if values of required attributes of Search Goals and Goal Results are visible in the test report, the Controller Input instance and Controller Output instance are implied.

For example, any setup for conditionally (or unconditionally) compliant [RFC2544] throughput testing can be understood as a MLRsearch architecture, ~~as long as~~if there is enough data to reconstruct the Relevant Upper Bound. ~~See~~ Refer to ~~the next subsection~~Section 4.9.2 for an equivalent Search Goal.

Any test procedure that can be understood as one call to the Manager of MLRsearch architecture is said to be compliant with MLRsearch Specification.

4.9.2. MLRsearch Compliant with RFC 2544~~RFC2544~~

The following Search Goal instance makes the corresponding Search Result unconditionally compliant with Section 24 of [RFC2544]~~ (Section 24).~~:

* Goal Final Trial Duration = 60 seconds
* Goal Duration Sum = 60 seconds
* Goal Loss Ratio = 0%
* Goal Exceed Ratio = 0%

~~The latter two attributes,~~ Goal Loss Ratio and Goal Exceed Ratio attributes~~,~~ are enough to make the Search Goal conditionally compliant. Adding ~~the first attribute,~~ Goal Final Trial Duration~~,~~ makes the Search Goal unconditionally compliant.

The second attribute (Goal Duration Sum) ~~only~~ prevents MLRsearch from repeating zero-loss Full-Length Trials.

The presence of other Search Goals does not affect the compliance of this Goal Result. The Relevant Lower Bound and the Conditional Throughput are in this case equal to each other, and the value is the [RFC2544] throughput.

Non-zero exceed ratio is not strictly disallowed, but it could

Commenté [MB98]: Not related but triggered by this, can we have at the end of the document a table with all the default values/recommended for the various attributes defined in the document?

needlessly prolong the search when Low-Loss short trials are present.

4.9.3. MLRsearch Compliant with TST009

One of the alternatives to [RFC2544] is Binary search with loss verification as described in Section 12.3.3 of [TST009] ~~(Section 12.3.3)~~.

~~The idea ther~~The rationale of such search ~~e~~ is to repeat high-loss trials, hoping for zero loss on second try, so the results are closer to the noiseless end of performance ~~sprectum~~spectrum, thus more repeatable and comparable.

Only the variant with "z = infinity" is achievable with MLRsearch.

For example, for "max(r) = 2" variant, the following Search Goal instance should be used to get compatible Search Result:

* Goal Final Trial Duration = 60 seconds
* Goal Duration Sum = 120 seconds
* Goal Loss Ratio = 0%
* Goal Exceed Ratio = 50%

If the first ~~60s~~ 60 seconds trial has zero loss, it is enough for MLRsearch to stop measuring at that load, as even a second high-loss trial would still fit within the exceed ratio.

But if the first trial is high-loss, MLRsearch needs to perform also the second trial to classify that load. Goal Duration Sum is twice as long as Goal Final Trial Duration, so third full-length trial is never needed.

5. Further Explanations

Commenté [MB99]: Please consider that a more explicit title that reflects the content.

This ~~chapter~~ section provides further explanations of MLRsearch behavior, mainly in comparison to a simple bisection for [RFC2544] Throughput.

5.1. Binary Search

A typical binary search implementation for [RFC2544] tracks only the two tightest bounds. To start, the search needs both Max Load and Min Load values. Then, one trial is used to confirm Max Load is an Upper Bound, and one trial to confirm Min Load is a Lower Bound.

Then, next Trial Load is chosen as the mean of the current tightest upper bound and the current tightest lower bound, and becomes a new tightest bound depending on the Trial Loss Ratio.

After some number of trials, the tightest lower bound becomes the throughput, but [RFC2544] does not specify when, if ever, the search should stop. In practice, the search stops either at some distance between the tightest upper bound and the tightest lower bound, or

after some number of Trials.

For a given pair of Max Load and Min Load values, there is one-to-one correspondence between number of Trials and final distance between the tightest bounds. Thus, the search always takes the same time, assuming initial bounds are confirmed.

5.2. Stopping Conditions and Precision

MLRsearch specification requires listing both Relevant Bounds for each Search Goal, and the difference between the bounds implies whether the result precision is achieved. Therefore, it is not necessary to report the specific stopping condition used.

MLRsearch Implementations may use Goal Width to allow direct control of result precision and ~~, and~~ indirect control of the Search Duration.

Other MLRsearch Implementations may use different stopping ~~conditions;~~conditions: for example, based on the Search Duration, trading off precision control for duration control.

Due to various possible time optimizations, there is no ~~longer a~~ strict correspondence between the Search Duration and Goal Width values. In practice, noisy SUT performance increases both average search time and its variance.

5.3. Loss Ratios and Loss Inversion

The most ~~obvious~~ difference between MLRsearch and [RFC2544] binary search is in the goals of the search. [RFC2544] has a single goal, based on classifying a single full-length trial as either zero-loss or non-zero-loss. MLRsearch supports searching for multiple Search Goals at once, usually differing in their Goal Loss Ratio values.

Commenté [MB100]: We don't need to say it if it is obvious 😉

5.3.1. Single Goal and Hard Bounds

Each bound in [RFC2544] simple binary search is "hard", in the sense that all further Trial Load values are smaller than any current upper bound and larger than any current lower bound.

This is also possible for MLRsearch Implementations, when the search is started with only one Search Goal instance.

5.3.2. Multiple Goals and Loss Inversion

MLRsearch supports multiple Search Goals, making the search procedure more complicated compared to binary search with single goal, but most of the complications do not affect the final results much. Except for one phenomenon: Loss Inversion.

Commenté [MB101]: Specification?

Depending on Search Goal attributes, Load Classification results may be resistant to small amounts of Inconsistent Trial Results (Section 3.5). ~~But~~ However, for larger amounts, a Load that is classified as an Upper Bound for one Search Goal may still be a Lower Bound for another Search Goal. ~~And, d~~Due to this other goal, MLRsearch will

probably perform subsequent Trials at Trial Loads even larger than the original value.

This introduces questions any many-goals search algorithm has to address. For example: What to do when all such larger load trials happen to have zero loss? Does it mean the earlier upper bound was not real? Does it mean the later Low-Loss trials are not considered a lower bound?

The situation where a smaller Load is classified as an Upper Bound, while a larger Load is classified as a Lower Bound (for the same search goal), is called Loss Inversion.

Conversely, only single-goal search algorithms can have hard bounds that shield them from Loss Inversion.

5.3.3. Conservativeness and Relevant Bounds

MLRsearch is conservative when dealing with Loss Inversion: the Upper Bound is considered real, and the Lower Bound is considered to be a fluke, at least when computing the final result.

This is formalized using definitions of Relevant Upper Bound (Section 4.7.1) and Relevant Lower Bound (Section 4.7.2).

The Relevant Upper Bound (for specific goal) is the smallest Load classified as an Upper Bound. But the Relevant Lower Bound is not simply the largest among Lower Bounds. It is the largest Load among Loads that are Lower Bounds while also being smaller than the Relevant Upper Bound.

With these definitions, the Relevant Lower Bound is always smaller than the Relevant Upper Bound (if both exist), and the two relevant bounds are used analogously as the two tightest bounds in the binary search. When they meet the stopping conditions, the Relevant Bounds are used in the output.

5.3.4. Consequences

The consequence of the way the Relevant Bounds are defined is that every Trial Result can have an impact on any current Relevant Bound larger than that Trial Load, namely by becoming a new Upper Bound.

This also applies when that Load is measured before another Load gets enough measurements to become a current Relevant Bound.

This also implies that if the SUT tested (or the Traffic Generator used) needs a warm-up, it should be warmed up before starting the Search, otherwise the first few measurements could become unjustly limiting.

For MLRsearch Implementations, it means it is better to measure at smaller Loads first, so bounds found earlier are less likely to get invalidated later.

5.4. Exceed Ratio and Multiple Trials

The idea of performing multiple Trials at the same Trial Load comes

from a model where some Trial Results (those with high Trial Loss Ratio) are affected by infrequent effects, causing ~~poor~~ unsatisfactory repeatability of [RFC2544] Throughput results. ~~See~~ Refer to Section 3.2 for ~~the~~ a discussion about noiseful and noiseless ends of the SUT performance spectrum ~~in section DUT in SUT (Section 3.2)~~. Stable results are closer to the noiseless end of the SUT performance spectrum, so MLRsearch may need to allow some frequency of high-loss trials to ignore the rare but big effects near the noiseful end.

For MLRsearch to perform such Trial Result filtering, it needs a configuration option to tell how frequent the "infrequent" big loss can be. This option is called the Goal Exceed Ratio (Section 4.5.4). It tells MLRsearch what ratio of trials (more specifically, what ratio of Trial Effective Duration seconds) can have a Trial Loss Ratio (Section 4.4.6) larger than the Goal Loss Ratio (Section 4.5.3) and still be classified as a Lower Bound (Section 4.6.2.2).

Zero exceed ratio means all Trials must have a Trial Loss Ratio equal to or lower than the Goal Loss Ratio.

When more than one Trial is intended to classify a Load, MLRsearch also needs something that controls the number of trials needed. Therefore, each goal also has an attribute called Goal Duration Sum.

The meaning of a Goal Duration Sum (Section 4.5.2) is that when a Load has (Full-Length) Trials whose Trial Effective Durations when summed up give a value at least as big as the Goal Duration Sum value, the Load is guaranteed to be classified either as an Upper Bound or a Lower Bound for that Search Goal instance.

5.5. Short Trials and Duration Selection

MLRsearch requires each ~~Searcg~~Search Goal to specify its Goal Final Trial Duration.

Section 24 of [RFC2544] already anticipates possible time savings when Short Trials are used.

Any MLRsearch Implementation may include its own configuration options which control when and how MLRsearch chooses to use short trial durations.

While MLRsearch Implementations are free to use any logic to select Trial Input values, comparability between MLRsearch Implementations is only assured when the Load Classification logic handles any possible set of Trial Results in the same way.

The presence of Short Trial Results complicates the Load Classification logic, see more details ~~in Load Classification Logic (~~Section 6.1~~) chapter~~.

While the Load Classification algorithm is designed to avoid any unneeded Trials, for explainability reasons it is recommended for users to use such Controller Input instances that lead to all Trial Duration values selected by Controller to be the same, e.g., ~~e.g.~~ by

Commenté [MB102]: Or other similar terms, but not poor thing. Please consider the same change in other parts of the document.

Commenté [MB103]: We may say that how this is exposed to a user/manager is implmentation specific.

setting any Goal Initial Trial Duration to be a single value also used in all Goal Final Trial Duration attributes.

5.6. Generalized Throughput

~~Due to the fact that~~Because a testing equipment takes the Intended Load as an input parameter for a Trial measurement, any load search algorithm needs to deal with Intended Load values internally.

But in the presence of Search Goals with a non-zero Goal Loss Ratio (Section 4.5.3), the Load usually does not match the user's intuition of what a throughput is. The forwarding rate as defined in Section 3.6.1 of [RFC2285] ~~(Section 3.6.1)~~ is better, but it is not obvious how to generalize it for Loads with multiple Trials and a non-zero Goal Loss Ratio.

The best example is also the main motivation: hard performance limit.

Commenté [MB104]: Not sure to parse this.

5.6.1. Hard Performance Limit

Even if bandwidth of ~~the~~ a medium allows higher traffic forwarding performance, the SUT interfaces may have their additional own limitations, ~~e.g.~~ e.g., a specific frames-per-second limit on the NIC (a common ~~occurence~~occurrence).

~~Ideally,~~ ~~t~~Those limitations should be known and provided as Max Load (Section 4.5.8.1). But if Max Load is set larger than what the interface can receive or transmit, there will be a "hard limit" behavior observed in Trial Results.

Commenté [MB105]: We may say that some implementation may expose their capabilities using IPFIX/YANG, but such exposure is out of scope.

~~Imagine~~ Consider that the hard limit is at hundred million frames per second (100 Mfps), Max Load is larger, and the Goal Loss Ratio is 0.5%. If DUT has no additional losses, 0.5% Trial Loss Ratio will be achieved at Relevant Lower Bound of 100.5025 Mfps. But it is not intuitive to report SUT performance as a value that is larger than the known hard limit. ~~We~~ There is a need of a generalization of [RFC2544] throughput, different from ~~just~~ solely the Relevant Lower Bound.

MLRsearch defines one such generalization, the Conditional Throughput (Section 4.7.3). It is the Trial Forwarding Rate from one of the Full-Length Trials performed at the Relevant Lower Bound. The algorithm to determine which trial exactly is in Appendix B: Conditional Throughput (Section 11).

In the hard limit example, 100.5025 Mfps Load will still have only 100.0 Mfps forwarding rate, nicely confirming the known limitation.

5.6.2. Performance Variability

With non-zero Goal Loss Ratio, and without hard performance limits, Low-Loss trials at the same Load may achieve different Trial Forwarding Rate values just due to DUT performance variability.

By comparing the best case (all Relevant Lower Bound trials have zero loss) and the worst case (all Trial Loss Ratios at Relevant Lower Bound are equal to the Goal Loss Ratio), we find the possible Conditional Throughput values may have up to the Goal Loss Ratio relative difference.

Setting the Goal Width below the Goal Loss Ratio may cause the Conditional Throughput for a larger Goal Loss Ratio to become smaller than a Conditional Throughput for a goal with a lower Goal Loss Ratio, which is counter-intuitive, considering they come from the same Search. ~~Therefore~~Therefore, it is RECOMMENDED to set the Goal Width to a value no lower than the Goal Loss Ratio of the higher-loss Search Goal.

Despite this variability, in practice Conditional Throughput behaves better than Relevant Lower Bound for comparability purposes, especially if deterministic Load selection is likely to produce exactly the same Relevant Lower Bound value across multiple runs.

6. MLRsearch Logic and Example

This section uses informal language to describe two ~~pieces~~ aspects of MLRsearch logic~~,~~ : Load Classification and Conditional Throughput, reflecting formal pseudocode representation ~~present~~ provided in Appendi~~ces~~x A~~:~~ ~~Load Classification (Section 10)~~ and Appendix B~~: Conditional Throughput (Section 11)~~. This is followed by example search.

The logic ~~as described here~~ is equivalent but not identical to the pseudocode on appendices. The pseudocode is designed to be short and frequently combines multiple operation into one expression. The logic as described ~~here~~ in this section lists each operation separately and uses more intuitive names for ~~te~~ the intermediate values.

6.1. Load Classification Logic

Note: For explanation clarity variables are ~~tagged~~tagged as (I)nput, (T)emporary, (O)utput.

Commenté [MB106]: Move this to the terminology/convention section

* Take all Trial Result instances (I) measured at a given load.
* Full-length high-loss sum (T) is the sum of Trial Effective Duration values of all full-length high-loss trials (I).
* Full-length low-loss sum (T) is the sum of Trial Effective Duration values of all full-length low-loss trials (I).
* Short high-loss sum is the sum (T) of Trial Effective Duration values of all short high-loss trials (I).
* Short low-loss sum is the sum (T) of Trial Effective Duration values of all short low-loss trials (I).
* Subceed ratio (T) is One minus the Goal Exceed Ratio (I).
* Exceed coefficient (T) is the Goal Exceed Ratio divided by the

subceed ratio.

* Balancing sum (T) is the short low-loss sum multiplied by the exceed coefficient.

* Excess sum (T) is the short high-loss sum minus the balancing sum.

* Positive excess sum (T) is the maximum of zero and excess sum.

* Effective high-loss sum (T) is the full-length high-loss sum plus the positive excess sum.

* Effective full sum (T) is the effective high-loss sum plus the full-length low-loss sum.

* Effective whole sum (T) is the larger of the effective full sum and the Goal Duration Sum.

* Missing sum (T) is the effective whole sum minus the effective full sum.

* Pessimistic high-loss sum (T) is the effective high-loss sum plus the missing sum.

* Optimistic exceed ratio (T) is the effective high-loss sum divided by the effective whole sum.

* Pessimistic exceed ratio (T) is the pessimistic high-loss sum divided by the effective whole sum.

* The load is classified as an Upper Bound (O) if the optimistic exceed ratio is larger than the Goal Exceed Ratio.

* The load is classified as a Lower Bound (O) if the pessimistic exceed ratio is not larger than the Goal Exceed Ratio.

* The load is classified as undecided (O), otherwise.

6.2. Conditional Throughput Logic

~~Note: For explanation clarity variables are taged as (I)nput, (T)emporary, (O)utput.~~

* Take all Trial Result instances (I) measured at a given Load.

* Full-length high-loss sum (T) is the sum of Trial Effective Duration values of all full-length high-loss trials (I).

* Full-length low-loss sum (T) is the sum of Trial Effective Duration values of all full-length low-loss trials (I).

* Full-length sum (T) is the full-length high-loss sum (I) plus the full-length low-loss sum (I).

* Subceed ratio (T) is One minus the Goal Exceed Ratio (I) is called.

* Remaining sum (T) initially is full-lengths sum multiplied by

subceed ratio.

* Current loss ratio (T) initially is 100%.

* For each full-length trial result, sorted in increasing order by Trial Loss Ratio:

  - If remaining sum is not larger than zero, exit the loop.

  - Set current loss ratio to this trial's Trial Loss Ratio (I).

  - Decrease the remaining sum by this trial's Trial Effective Duration (I).

* Current forwarding ratio (T) is One minus the current loss ratio.

* Conditional Throughput (T) is the current forwarding ratio multiplied by the Load value.

This shows that Conditional Throughput is partially related to Load Classification. If a Load is classified as a Relevant Lower Bound for a Search Goal instance, the Conditional Throughput comes from a Trial Result that is guaranteed to have Trial Loss Ratio no larger than the Goal Loss Ratio. The converse is not true if Goal Width is smaller than the Goal Loss Ratio, as in that case it is possible for the Conditional Throughput to be larger than the Relevant Upper Bound.

6.3. SUT Behaviors

In ~~DUT in SUT (~~Section 3.2~~)~~, the notion of noise has been introduced. ~~In t~~This section ~~we rely on~~uses new terms~~ defined since then~~ to describe possible SUT behaviors more precisely.

From measurement point of view, noise is visible as inconsistent trial results. See Inconsistent Trial Results (Section 3.5) for general points and Loss Ratios and Loss Inversion (Section 5.3) for specifics when comparing different Load values.

Load Classification and Conditional Throughput apply to a single Load value, but even the set of Trial Results measured at that Trial Load value may appear inconsistent.

As MLRsearch aims to save time, it executes only a small number of Trials, getting only a limited amount of information about SUT behavior. It is useful to introduce an "SUT expert" point of view to contrast with that limited information.

6.3.1. Expert Predictions

Imagine that before the Search starts, a human expert had unlimited time to measure SUT and obtain all reliable information about it. The information is not perfect, as there is still random noise influencing SUT. But the expert is familiar with possible noise events, even the rare ones, and thus the expert can do probabilistic predictions about future Trial Outputs.

When several outcomes are possible, the expert can ~~asses~~assess probability
of each outcome.

6.3.2. Exceed Probability

When the Controller selects new Trial Duration and Trial Load, and just before the Measurer starts performing the Trial, the SUT expert can envision possible Trial Results.

With respect to a particular Search Goal instance, the possibilities can be summarized into a single number: Exceed Probability. It is the probability (according to the expert) that the measured Trial Loss Ratio will be higher than the Goal Loss Ratio.

6.3.3. Trial Duration Dependence

When comparing Exceed Probability values for the same Trial Load value but different Trial Duration values, there are several patterns that commonly occur in practice.

6.3.3.1. Strong Increase

Exceed Probability is very low at short durations but very high at full-length. This SUT behavior is undesirable, and may hint at faulty SUT, e.g. SUT leaks resources and is unable to sustain the desired performance.

But this behavior is also seen when SUT uses large amount of buffers. This is the main reasons users may want to set large Goal Final Trial Duration.

6.3.3.2. Mild Increase

Short trials have lower ~~exceed~~exceeded probability, but the difference is not
as high. This behavior is quite common if the noise contains infrequent but large loss spikes, as the more performant parts of a full-length trial are unable to compensate for all the frame loss from a less performant part.

6.3.3.3. Independence

Short trials have basically the same Exceed Probability as full-length trials. This is possible only if loss spikes are small (so other parts can compensate) and if Goal Loss Ratio is more than zero (~~otherwise~~otherwise, other parts cannot compensate at all).

6.3.3.4. Decrease

Short trials have larger Exceed Probability than full-length trials. This can be possible only for non-zero Goal Loss Ratio, for example if SUT needs to "warm up" to best performance within each trial. Not ~~commonly~~commonly seen in practice.

6.4. Example Search

Commenté [MB107]: We may move this section to an appendix

The following example Search is related to one hypothetical run of a

Search test procedure that has been started with multiple Search Goals. Several points in time are chosen, ~~in order to~~to show how the logic works, with specific sets of Trial Result available. The trial results themselves are not very realistic, as the intention is to show several corner cases of the logic.

In all Trials, the Effective Trial Duration is equal to Trial Duration.

Only one Trial Load is in focus, its value is one million frames per second. Trial Results at other Trial Loads are not mentioned, as the parts of logic present here do not depend on those. In practice, Trial Results at other Load values would be present, e.g.,~~e.g.~~ MLRsearch will look for a Lower Bound smaller than any Upper Bound found.

In all points in time, only one Search Goal instance is marked as "in focus". That explains Trial Duration of the new Trials, but is ~~otherwise~~otherwise, unrelated to the logic applied.

MLRsearch Implementations are not required to "focus" on one goal at time, but this example is useful to show a load can be classified ~~also~~also, for goals not "in focus".

6.4.1. Example Goals

The following four Search Goal instances are selected for the example Search. Each goal has a readable name and dense code, the code is useful to show Search Goal attribute values.

As the variable "exceed coefficient" does not depend on trial results, it is also precomputed here.

Goal 1:

name: RFC2544
Goal Final Trial Duration: 60s
Goal Duration Sum: 60s
Goal Loss Ratio: 0%
Goal Exceed Ratio: 0%
exceed coefficient: 0% / (100% / 0%) = 0.0
code: 60f60d0l0e

Goal 2:

name: TST009
Goal Final Trial Duration: 60s
Goal Duration Sum: 120s
Goal Loss Ratio: 0%
Goal Exceed Ratio: 50%
exceed coefficient: 50% / (100% - 50%) = 1.0
code: 60f120d0l50e

Goal 3:

name: 1s final
Goal Final Trial Duration: 1s
Goal Duration Sum: 120s

Goal Loss Ratio: 0.5%
Goal Exceed Ratio: 50%
exceed coefficient: 50% / (100% - 50%) = 1.0
code: 1f120d.5l50e

Goal 4:

name: 20% exceed
Goal Final Trial Duration: 60s
Goal Duration Sum: 60s
Goal Loss Ratio: 0.5%
Goal Exceed Ratio: 20%
exceed coefficient: 20% / (100% - 20%) = 0.25
code: 60f60d0.5l20e

The first two goals are important for compliance reasons, the other two cover less frequent cases.

6.4.2. Example Trial Results

The following six sets of trial results are selected for the example Search. The sets are defined as points in time, describing which Trial Results were added since the previous point.

Each point has a readable name and dense code, the code is useful to show Trial Output attribute values and number of times identical results were added.

Point 1:

name: first short good
goal in focus: 1s final (1f120d.5l50e)
added Trial Results: 59 trials, each 1 second and 0% loss
code: 59x1s0l

Point 2:

name: first short bad
goal in focus: 1s final (1f120d.5l50e)
added Trial Result: one trial, 1 second, 1% loss
code: 59x1s0l+1x1s1l

Point 3:

name: last short bad
goal in focus: 1s final (1f120d.5l50e)
added Trial Results: 59 trials, 1 second each, 1% loss each
code: 59x1s0l+60x1s1l

Point 4:

name: last short good
goal in focus: 1s final (1f120d.5l50e)
added Trial Results: one trial 1 second, 0% loss
code: 60x1s0l+60x1s1l

Point 5:

name: first long bad
goal in focus: TST009 (60f120d0l50e)
added Trial Results: one trial, 60 seconds, 0.1% loss
code: 60x1s0l+60x1s1l+1x60s.1l

Point 6:

name: first long good
goal in focus: TST009 (60f120d0l50e)
added Trial Results: one trial, 60 seconds, 0% loss
code: 60x1s0l+60x1s1l+1x60s.1l+1x60s0l

Comments on point in time naming:

* When a name contains "short", it means the added trial had Trial Duration of 1 second, which is Short Trial for 3 of the Search Goals, but it is a Full-Length Trial for the "1s final" goal.

* Similarly, "long" in name means the added trial had Trial Duration of 60 seconds, which is Full-Length Trial for 3 goals but Long Trial for the "1s final" goal.

* When a name contains "good" it means the added trial is Low-Loss Trial for all the goals.

* When a name contains "short bad" it means the added trial is High-Loss Trial for all the goals.

* When a name contains "long bad", it means the added trial is a High-Loss Trial for goals "RFC2544" and "TST009", but it is a Low-Loss Trial for the two other goals.

6.4.3. Load Classification Computations

This section shows how Load Classification logic is applied by listing all temporary values at the specific time point.

6.4.3.1. Point 1

This is the "first short good" point. Code for available results is: 59x1s0l

Goal name	RFC2544	TST009	1s final	20% exceed
Goal code	60f60d0l0e	60f120d0l50e	1f120d.5l50e	60f60d0.5l20e
Full-length high-loss sum	0s	0s	0s	0s
Full-length low-loss sum	0s	0s	59s	0s
Short high-loss sum	0s	0s	0s	0s
Short low-loss sum	59s	59s	0s	59s

Balancing sum	0s	59s	0s	14.75s
Excess sum	0s	-59s	0s	-14.75s
Positive excess sum	0s	0s	0s	0s
Effective high-loss sum	0s	0s	0s	0s
Effective full sum	0s	0s	59s	0s
Effective whole sum	60s	120s	120s	60s
Missing sum	60s	120s	61s	60s
Pessimistic high-loss sum	60s	120s	61s	60s
Optimistic exceed ratio	0%	0%	0%	0%
Pessimistic exceed ratio	100%	100%	50.833%	100%
Classification Result	Undecided	Undecided	Undecided	Undecided

Table 1: XXXX

Commenté [MB108]: Please add a table legend. Idem for all tables

This is the last point in time where all goals have this load as Undecided.

6.4.3.2. Point 2

This is the "first short bad" point. Code for available results is: 59x1s0l+1x1s1l

Goal name	RFC2544	TST009	1s final	20% exceed
Goal code	60f60d0l0e	60f120d0l50e	1f120d.5l50e	60f60d0.5l20e
Full-length high-loss sum	0s	0s	1s	0s
Full-length low-loss sum	0s	0s	59s	0s
Short high-loss sum	1s	1s	0s	1s
Short low-loss sum	59s	59s	0s	59s

Balancing sum	0s	59s	0s	14.75s
Excess sum	1s	-58s	0s	-13.75s
Positive excess sum	1s	0s	0s	0s
Effective high-loss sum	1s	0s	1s	0s
Effective full sum	1s	0s	60s	0s
Effective whole sum	60s	120s	120s	60s
Missing sum	59s	120s	60s	60s
Pessimistic high-loss sum	60s	120s	61s	60s
Optimistic exceed ratio	1.667%	0%	0.833%	0%
Pessimistic exceed ratio	100%	100%	50.833%	100%
Classification Result	Upper Bound	Undecided	Undecided	Undecided

Table 2

Due to zero Goal Loss Ratio, RFC2544 goal must have mild or strong increase of exceed probability, so the one lossy trial would be lossy even if measured at 60 second duration. Due to zero exceed ratio, one High-Loss Trial is enough to preclude this Load from becoming a Lower Bound for RFC2544. That is why this Load is classified as an Upper Bound for RFC2544 this early.

This is an example how significant time can be saved, compared to 60-second trials.

6.4.3.3. Point 3

This is the "last short bad" point. Code for available trial results is: 59x1s0l+60x1s1l

Goal name	RFC2544	TST009	1s final	20% exceed
Goal code	60f60d0l0e	60f120d0l50e	1f120d.5l50e	60f60d0.5l20e
Full-length high-loss sum	0s	0s	60s	0s
Full-length	0s	0s	59s	0s

low-loss sum				
Short high-loss sum	60s	60s	0s	60s
Short low-loss sum	59s	59s	0s	59s
Balancing sum	0s	59s	0s	14.75s
Excess sum	60s	1s	0s	45.25s
Positive excess sum	60s	1s	0s	45.25s
Effective high-loss sum	60s	1s	60s	45.25s
Effective full sum	60s	1s	119s	45.25s
Effective whole sum	60s	120s	120s	60s
Missing sum	0s	119s	1s	14.75s
Pessimistic high-loss sum	60s	120s	61s	60s
Optimistic exceed ratio	100%	0.833%	50%	75.417%
Pessimistic exceed ratio	100%	100%	50.833%	100%
Classification Result	Upper Bound	Undecided	Undecided	Upper Bound

Table 3

This is the last point for "1s final" goal to have this Load still Undecided. Only one 1-second trial is missing within the 120-second Goal Duration Sum, but its result will decide the classification result.

The "20% exceed" started to classify this load as an Upper Bound somewhere between points 2 and 3.

6.4.3.4. Point 4

This is the "last short good" point. Code for available trial results is: 60x1s0l+60x1s1l

Goal name	RFC2544	TST009	1s final	20% exceed

Goal code	60f60d0l0e	60f120d0l50e	1f120d.5l50e	60f60d0.5l20e
Full-length high-loss sum	0s	0s	60s	0s
Full-length low-loss sum	0s	0s	60s	0s
Short high-loss sum	60s	60s	0s	60s
Short low-loss sum	60s	60s	0s	60s
Balancing sum	0s	60s	0s	15s
Excess sum	60s	0s	0s	45s
Positive excess sum	60s	0s	0s	45s
Effective high-loss sum	60s	0s	60s	45s
Effective full sum	60s	0s	120s	45s
Effective whole sum	60s	120s	120s	60s
Missing sum	0s	120s	0s	15s
Pessimistic high-loss sum	60s	120s	60s	60s
Optimistic exceed ratio	100%	0%	50%	75%
Pessimistic exceed ratio	100%	100%	50%	100%
Classification Result	Upper Bound	Undecided	Lower Bound	Upper Bound

Table 4

The one missing trial for "1s final" was Low-Loss, half of trial results are Low-Loss which exactly matches 50% exceed ratio. This shows time savings are not guaranteed.

6.4.3.5. Point 5

This is the "first long bad" point. Code for available trial results is: 60x1s0l+60x1s1l+1x60s.1l

Goal name	RFC2544	TST009	1s final	20% exceed

Goal code	60f60d0l0e	60f120d0l50e	1f120d.5l50e	60f60d0.5l20e
Full-length high-loss sum	60s	60s	60s	0s
Full-length low-loss sum	0s	0s	120s	60s
Short high-loss sum	60s	60s	0s	60s
Short low-loss sum	60s	60s	0s	60s
Balancing sum	0s	60s	0s	15s
Excess sum	60s	0s	0s	45s
Positive excess sum	60s	0s	0s	45s
Effective high-loss sum	120s	60s	60s	45s
Effective full sum	120s	60s	180s	105s
Effective whole sum	120s	120s	180s	105s
Missing sum	0s	60s	0s	0s
Pessimistic high-loss sum	120s	120s	60s	45s
Optimistic exceed ratio	100%	50%	33.333%	42.857%
Pessimistic exceed ratio	100%	100%	33.333%	42.857%
Classification Result	Upper Bound	Undecided	Lower Bound	Lower Bound

Table 5

As designed for TST009 goal, one Full-Length High-Loss Trial can be tolerated. 120s worth of 1-second trials is not useful, as this is allowed when Exceed Probability does not depend on Trial Duration.

As Goal Loss Ratio is zero, it is not ~~really possible~~possible for 60-second trials to compensate for losses seen in 1-second results. But Load Classification logic does not have that knowledge hardcoded, so optimistic exceed ratio is still only 50%.

But the 0.1% Trial Loss Ratio is lower than "20% exceed" Goal Loss

Ratio, so this unexpected Full-Length Low-Loss trial changed the classification result of this Load to Lower Bound.

6.4.3.6. Point 6

This is the "first long good" point. Code for available trial results is: 60x1s0l+60x1s1l+1x60s.1l+1x60s0l

Goal name	RFC2544	TST009	1s final	20% exceed
Goal code	60f60d0l0e	60f120d0l50e	1f120d.5l50e	60f60d0.5l20e
Full-length high-loss sum	60s	60s	60s	0s
Full-length low-loss sum	60s	60s	180s	120s
Short high-loss sum	60s	60s	0s	60s
Short low-loss sum	60s	60s	0s	60s
Balancing sum	0s	60s	0s	15s
Excess sum	60s	0s	0s	45s
Positive excess sum	60s	0s	0s	45s
Effective high-loss sum	120s	60s	60s	45s
Effective full sum	180s	120s	240s	165s
Effective whole sum	180s	120s	240s	165s
Missing sum	0s	0s	0s	0s
Pessimistic high-loss sum	120s	60s	60s	45s
Optimistic exceed ratio	66.667%	50%	25%	27.273%
Pessimistic exceed ratio	66.667%	50%	25%	27.273%
Classification Result	Upper Bound	Lower Bound	Lower Bound	Lower Bound

Table 6

This is the Low-Loss Trial the "TST009" goal was waiting for. This Load is now classified for all ~~goals,~~goals; the search may end. Or, more realistically, it can focus on larger load only, as the three goals will want an Upper Bound (unless this Load is Max Load).

6.4.4. Conditional Throughput Computations

At the end of the hypothetical search, "RFC2544" goal has this load classified as an Upper Bound, so it is not eligible for Conditional Throughput calculations. But the remaining three goals ~~calssify~~classify this Load as a Lower Bound, and if we assume it has also ~~became~~become the Relevant Lower Bound, we can compute Conditional Throughput values for all three goals.

As a reminder, the Load value is one million frames per second.

6.4.4.1. Goal 2

The Conditional Throughput is computed from sorted list of Full-Length Trial results. As TST009 Goal Final Trial Duration is 60 seconds, only two of 122 Trials are considered Full-Length Trials. One has Trial Loss Ratio of 0%, the other of 0.1%.

* Full-length high-loss sum is 60 seconds.
* Full-length low-loss sum is 60 seconds.
* Full-length is 120 seconds.
* Subceed ratio is 50%.
* Remaining sum initially is 0.5x12s = 60 seconds.
* Current loss ratio initially is 100%.
* For first result (duration 60s, loss 0%):
  - Remaining sum is larger than zero, not exiting the loop.
  - Set current loss ratio to this trial's Trial Loss Ratio which is 0%.
  - Decrease the remaining sum by this trial's Trial Effective Duration.
  - New remaining sum is 60s - 60s = 0s.
* For second result (duration 60s, loss 0.1%):
* Remaining sum is not larger than zero, exiting the loop.
* Current forwarding ratio was most recently set to 0%.
* Current forwarding ratio is one minus the current loss ratio, so 100%.

* Conditional Throughput is the current forwarding ratio multiplied by the Load value.

* Conditional Throughput is one million frames per second.

6.4.4.2. Goal 3

The "1s final" has Goal Final Trial Duration of 1 second, so all 122 Trial Results are considered Full-Length Trials. They are ordered like this:

60 1-second 0% loss trials,
1 60-second 0% loss trial,
1 60-second 0.1% loss trial,
60 1-second 1% loss trials.

The result does not depend on the order of 0% loss trials.

* Full-length high-loss sum is 60 seconds.

* Full-length low-loss sum is 180 seconds.

* Full-length is 240 seconds.

* Subceed ratio is 50%.

* Remaining sum initially is 0.5x240s = 120 seconds.

* Current loss ratio initially is 100%.

* For first 61 results (duration varies, loss 0%):

  - Remaining sum is larger than zero, not exiting the loop.

  - Set current loss ratio to this trial's Trial Loss Ratio which is 0%.

  - Decrease the remaining sum by this trial's Trial Effective Duration.

  - New remaining sum varies.

* After 61 trials, we have subtracted 60x1s + 1x60s from 120s, remaining 0s.

* For 62-th result (duration 60s, loss 0.1%):

  - Remaining sum is not larger than zero, exiting the loop.

* Current forwarding ratio was most recently set to 0%.

* Current forwarding ratio is one minus the current loss ratio, so 100%.

* Conditional Throughput is the current forwarding ratio multiplied by the Load value.

* Conditional Throughput is one million frames per second.

6.4.4.3. Goal 4

The Conditional Throughput is computed from sorted list of Full-Length Trial results. As "20% exceed" Goal Final Trial Duration is 60 seconds, only two of 122 Trials are considered Full-Length Trials. One has Trial Loss Ratio of 0%, the other of 0.1%.

* Full-length high-loss sum is 60 seconds.

* Full-length low-loss sum is 60 seconds.

* Full-length is 120 seconds.

* Subceed ratio is 80%.

* Remaining sum initially is 0.8x120s = 96 seconds.

* Current loss ratio initially is 100%.

* For first result (duration 60s, loss 0%):

  - Remaining sum is larger than zero, not exiting the loop.

  - Set current loss ratio to this trial's Trial Loss Ratio which is 0%.

  - Decrease the remaining sum by this trial's Trial Effective Duration.

  - New remaining sum is 96s - 60s = 36s.

* For second result (duration 60s, loss 0.1%):

  - Remaining sum is larger than zero, not exiting the loop.

  - Set current loss ratio to this trial's Trial Loss Ratio which is 0.1%.

  - Decrease the remaining sum by this trial's Trial Effective Duration.

  - New remaining sum is 36s - 60s = -24s.

* No more trials (~~and also~~and remaining sum is not larger than zero), exiting loop.

* Current forwarding ratio was most recently set to 0.1%.

* Current forwarding ratio is one minus the current loss ratio, so 99.9%.

* Conditional Throughput is the current forwarding ratio multiplied by the Load value.

* Conditional Throughput is 999 thousand frames per second.

Due to stricter Goal Exceed Ratio, this Conditional Throughput is smaller than Conditional Throughput of the other two goals.

7. IANA Considerations

This document does not make any ~~No~~ request~~s of~~ to IANA.

8. 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 an "opaque~~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.

xx

9. Acknowledgements

Special wholehearted gratitude and thanks to the late Al Morton for his thorough reviews filled with very specific feedback and constructive guidelines. Thank You Al for the close collaboration over the years, Your Mentorship, Your continuous unwavering encouragement full of empathy and energizing positive attitude. Al, You are dearly missed.

Thanks to Gabor Lencse, Giuseppe Fioccola and BMWG contributors for good discussions and thorough reviews, guiding and helping us to improve the clarity and formality of this document.

Many thanks to Alec Hothan of the OPNFV NFVbench project for a thorough review and numerous useful comments and suggestions in the earlier versions of this document.

10. Appendix A: Load Classification

This ~~section~~ appendix specifies how to perform the Load Classification.

Any Trial Load value can be classified, according to a given Search Goal (Section 4.5.7) instance.

The algorithm uses (some subsets of) the set of all available Trial Results from Trials measured at a given Load at the end of the Search.

a mis en forme : Anglais (États-Unis)

Commenté [MB109]: Some more elaboration is needed

Commenté [MB110]: Why?

We can accept some relax rule in controlled environnement, but this not acceptable in deployement. I would adjust accordingly.

Commenté [MB111]: I would some text to basically say that the benchmarking results should be adequately protected and guards top prevent leaks to unauthorized entities.
Otherwise, the benchmark results can be used by attacker to better adjust their attacks and perform attacks that would lead to DDoS a node of the DUT in a live network, infer the limitation of a DUT that can be used for overflow attacks, etc.

Also, we can say that the benchmark is agnostic to trafic and does not manipulate real traffic. As such, Privacy is not a concern.

Commenté [MB112]: Move after references

The block at the end of this appendix holds pseudocode which computes two values, stored in variables named optimistic_is_lower and pessimistic_is_lower.

The pseudocode ~~happens to be~~is a valid Python code.

Commenté [MB113]: Where is that python code?

If values of both variables are computed to be true, the Load in question is classified as a Lower Bound according to the given Search Goal instance. If values of both variables are false, the Load is classified as an Upper Bound. Otherwise, the load is classified as Undecided.

Some variable names are shortened ~~in order to~~to fit expressions in one line. Namely, variables holding sum quantities end in _s instead of _sum, and variables holding effective quantities start in effect_ instead of effective_.

The pseudocode expects the following variables to hold the following values:

* goal_duration_s: The Goal Duration Sum value of the given Search Goal.

* goal_exceed_ratio: The Goal Exceed Ratio value of the given Search Goal.

* full_length_low_loss_s: Sum of Trial Effective Durations across Trials with Trial Duration at least equal to the Goal Final Trial Duration and with Trial Loss Ratio not higher than the Goal Loss Ratio (across Full-Length Low-Loss Trials).

* full_length_high_loss_s: Sum of Trial Effective Durations across Trials with Trial Duration at least equal to the Goal Final Trial Duration and with Trial Loss Ratio higher than the Goal Loss Ratio (across Full-Length High-Loss Trials).

* short_low_loss_s: Sum of Trial Effective Durations across Trials with Trial Duration shorter than the Goal Final Trial Duration and with Trial Loss Ratio not higher than the Goal Loss Ratio (across Short Low-Loss Trials).

* short_high_loss_s: Sum of Trial Effective Durations across Trials with Trial Duration shorter than the Goal Final Trial Duration and with Trial Loss Ratio higher than the Goal Loss Ratio (across Short High-Loss Trials).

The code works correctly also when there are no Trial Results at a given Load.

```
exceed_coefficient = goal_exceed_ratio / (1.0 - goal_exceed_ratio)
balancing_s = short_low_loss_s * exceed_coefficient
positive_excess_s = max(0.0, short_high_loss_s - balancing_s)
effect_high_loss_s = full_length_high_loss_s + positive_excess_s
effect_full_length_s = full_length_low_loss_s + effect_high_loss_s
effect_whole_s = max(effect_full_length_s, goal_duration_s)
quantile_duration_s = effect_whole_s * goal_exceed_ratio
pessimistic_high_loss_s = effect_whole_s - full_length_low_loss_s
pessimistic_is_lower = pessimistic_high_loss_s <= quantile_duration_s
```

Commenté [MB114]: May display this a table for better readabiulity

```
   optimistic_is_lower = effect_high_loss_s <= quantile_duration_s
```

11. Appendix B: Conditional Throughput

This section specifies an example how to compute Conditional Throughput, as referred to in ~~section Conditional Throughput (~~Section 4.7.3~~)~~.

Any Load value can be used as the basis for the following computation, but only the Relevant Lower Bound (at the end of the Search) leads to the value called the Conditional Throughput for a given Search Goal.

The algorithm uses (some subsets of) the set of all available Trial Results from Trials measured at a given Load at the end of the Search.

The block at the end of this appendix holds pseudocode which computes a value stored as variable conditional_throughput.

The pseudocode ~~happens to~~is a ~~be~~ valid Python code.

Some variable names are shortened in order to fit expressions in one line. Namely, variables holding sum quantities end in _s instead of _sum, and variables holding effective quantities start in effect_ instead of effective_.

The pseudocode expects the following variables to hold the following values:

* goal_duration_s: The Goal Duration Sum value of the given Search Goal.

* goal_exceed_ratio: The Goal Exceed Ratio value of the given Search Goal.

* full_length_low_loss_s: Sum of Trial Effective Durations across Trials with Trial Duration at least equal to the Goal Final Trial Duration and with Trial Loss Ratio not higher than the Goal Loss Ratio (across Full-Length Low-Loss Trials).

* full_length_high_loss_s: Sum of Trial Effective Durations across Trials with Trial Duration at least equal to the Goal Final Trial Duration and with Trial Loss Ratio higher than the Goal Loss Ratio (across Full-Length High-Loss Trials).

* full_length_trials: An iterable of all Trial Results from Trials with Trial Duration at least equal to the Goal Final Trial Duration (all Full-Length Trials), sorted by increasing Trial Loss Ratio. One item trial is a composite with the following two attributes available:

  - trial.loss_ratio: The Trial Loss Ratio as measured for this Trial.

  - trial.effect_duration: The Trial Effective Duration of this Trial.

The code works correctly only when there if there is at least one Trial ~~Result~~ Result measured at the given Load.

```
full_length_s = full_length_low_loss_s + full_length_high_loss_s
whole_s = max(goal_duration_s, full_length_s)
remaining = whole_s * (1.0 - goal_exceed_ratio)
quantile_loss_ratio = None
for trial in full_length_trials:
    if quantile_loss_ratio is None or remaining > 0.0:
        quantile_loss_ratio = trial.loss_ratio
        remaining -= trial.effect_duration
    else:
        break
else:
    if remaining > 0.0:
        quantile_loss_ratio = 1.0
conditional_throughput = intended_load * (1.0 - quantile_loss_ratio)
```

Commenté [MB115]: Please use <CODE BEGINS> and <CODE ENDS> markers

12. Index

* Intended Load: defined in [RFC2285] (Section 3.5.1).

* Irregular Goal Result: defined in Irregular Goal Result (Section 4.7.4.2).

* Load: introduced in Trial Load (Section 4.4.2).

* Load Classification: Introduced in Overview (Section 4.1), defined in Load Classification (Section 4.6.2), discussed in Load Classification Logic (Section 6.1).

* Loss Inversion: Situation introduced in Inconsistent Trial Results (Section 3.5), defined in Loss Ratios and Loss Inversion (Section 5.3).

* Low-Loss Trial: defined in Low-Loss Trial (Section 4.6.1.2).

* Lower Bound: defined in Lower Bound (Section 4.6.2.2).

* Manager: introduced in Overview (Section 4.1), defined in Manager (Section 4.8.3).

* Max Load: defined in Max Load (Section 4.5.8.1).

* Measurer: introduced in Overview (Section 4.1), defined in ~~Meaurer~~Measurer (Section 4.8.1).

* Min Load: defined in Min Load (Section 4.5.8.2).

* MLRsearch Specification: introduced in Purpose and Scope (Section 2) and in Overview (Section 4.1), defined in Test Procedure Compliant with MLRsearch (Section 4.9.1).

* MLRsearch Implementation: defined in Test Procedure Compliant with MLRsearch (Section 4.9.1).

* Offered Load: defined in [RFC2285] (Section 3.5.2).

* Regular Goal Result: defined in Regular Goal Result (Section 4.7.4.1).

* Relevant Bound: Relevant Lower Bound or Relevant Upper Bound.

* Relevant Bounds: Relevant Lower Bound and Relevant Upper Bound.

* Relevant Lower Bound: defined in Relevant Lower Bound (Section 4.7.2), discussed in Conservativeness and Relevant Bounds (Section 5.3.3).

* Relevant Upper Bound: defined in Relevant Upper Bound (Section 4.7.1).

* Search: defined in Overview (Section 4.1).

* Search Duration: introduced in Purpose and Scope (Section 2) and in Long Search Duration (Section 3.1), discussed in Stopping

Conditions and Precision (Section 5.2).

* Search Goal: defined in Search Goal (Section 4.5.7).

* Search Result: defined in Search Result (Section 4.7.5).

* Short Trial: defined in Short Trial (Section 4.6.1.3).

* Throughput: defined in Section 3.17 of [RFC1242] (~~Section 3.17~~), Methodology specified in Section 26.1 of [RFC2544] (~~Section 26.1~~).

* Trial: defined in Trial (Section 4.3.3).

* Trial Duration: defined in Trial Duration (Section 4.4.1).

* Trial Effective Duration: defined in Trial Effective Duration (Section 4.4.8).

* Trial Forwarding Rate: defined in Trial Forwarding Rate (Section 4.4.7).

* Trial Forwarding Ratio: defined in Trial Forwarding Ratio (Section 4.4.5).

* Trial Input: defined in Trial Input (Section 4.4.3).

* Trial Loss Ratio: defined in Trial Loss Ratio (Section 4.4.6).

* Trial Load: defined in Trial Load (Section 4.4.2).

* Trial Output: defined in Trial Output (Section 4.4.9).

* Trial Result: defined in Trial Result (Section 4.4.10).

* Undecided: defined in Undecided (Section 4.6.2.3).

* Upper Bound: defined in Upper Bound (Section 4.6.2.1).

13. References

13.1. Normative References

[RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <https://www.rfc-editor.org/info/rfc1242>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2285] Mandeville, R., "Benchmarking Terminology for LAN Switching Devices", RFC 2285, DOI 10.17487/RFC2285, February 1998, <https://www.rfc-editor.org/info/rfc2285>.

[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>.

[RFC5180] Popoviciu, C., Hamza, A., Van de Velde, G., and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, DOI 10.17487/RFC5180, May 2008, <https://www.rfc-editor.org/info/rfc5180>.

[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>.

[RFC8219] Georgescu, M., Pislaru, L., and G. Lencse, "Benchmarking Methodology for IPv6 Transition Technologies", RFC 8219, DOI 10.17487/RFC8219, August 2017, <https://www.rfc-editor.org/info/rfc8219>.

Commenté [MB116]: Please move to information, as this was provided only as an example

Commenté [MB117]: Idem as the other entry

13.2. Informative References

[FDio-CSIT-MLRsearch]
"FD.io CSIT Test Methodology - MLRsearch", October 2023, <https://csit.fd.io/cdocs/methodology/measurements/data_plane_throughput/mlr_search/>.

[Lencze-Kovacs-Shima]
"Gaming with the Throughput and the Latency Benchmarking Measurement Procedures of RFC 2544", n.d., <http://dx.doi.org/10.11601/ijates.v9i2.288>.

[Lencze-Shima]
"An Upgrade to Benchmarking Methodology for Network Interconnect Devices", n.d., <https://datatracker.ietf.org/doc/html/draft-lencse-bmwg-rfc2544-bis-00>.

Commenté [MB118]: This was expired since 2020. Please remove. Idem for all similar entries

[Ott-Mathis-Semke-Mahdavi]
"The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", n.d., <https://www.cs.cornell.edu/people/egs/cornellonly/syslunch/fall02/ott.pdf>.

[PyPI-MLRsearch]
"MLRsearch 1.2.1, Python Package Index", October 2023, <https://pypi.org/project/MLRsearch/1.2.1/>.

[RFC6349] Constantine, B., Forget, G., Geib, R., and R. Schrage, "Framework for TCP Throughput Testing", RFC 6349, DOI 10.17487/RFC6349, August 2011, <https://www.rfc-editor.org/info/rfc6349>.

[TST009] "TST 009", n.d., <https://www.etsi.org/deliver/etsi_gs/NFV-TST/001_099/009/03.04.01_60/gs_NFV-TST009v030401p.pdf>.

Authors' Addresses

Maciek Konstantynowicz
Cisco Systems

Email: mkonstan@cisco.com

Vratko Polak
Cisco Systems
Email: vrpolak@cisco.com