Stream: Internet Engineering Task Force (IETF)
RFC: 9971
Category: Informational
Published: May 2026
ISSN: 2070-1721
Authors: M. Konstantynowicz, *Cisco Systems*
V. Polak, *Cisco Systems*

RFC 9971
Multiple Loss Ratio Search

Abstract

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

MLRsearch is motivated by the pressing need to address the challenges of evaluating and testing the various data plane solutions, especially in software-based networking systems based on Commercial Off-the-Shelf (COTS) CPU hardware vs. purpose-built Application-Specific Integrated Circuit (ASIC), Network Processing Unit (NPU), and Field-Programmable Gate Array (FPGA) hardware.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9971.

Copyright Notice

Table of Contents

1. Introduction

This document describes the Multiple Loss Ratio search (MLRsearch) methodology, optimized for determining data plane throughput in software-based networking functions running on commodity systems with generic CPUs (vs. purpose-built ASICs, NPUs, and FPGAs). Such network

functions can be deployed on a dedicated physical appliance (e.g., a standalone hardware device) or as virtual appliance (e.g., a Virtual Network Function running on shared servers in the compute cloud).

This document tightly couples terminology and methodology aspects. Instead of a separate terminology section, the subsections of "MLRsearch Specification" (Section 4) act as a list of newly defined terms. If a term appears with the first letter capitalized, it likely refers to a specific term defined in an eponymous subsection of "MLRsearch Specification" (Section 4).

For first-time readers, the information in "MLRsearch Specification" (Section 4) might feel dense and lacking motivation. Subsequent sections provide explanations, making "MLRsearch Specification" (Section 4) more approachable on repeated reads.

1.1. Purpose

The purpose of this document is to describe the Multiple Loss Ratio search (MLRsearch) methodology, optimized for determining data plane throughput in software-based networking devices and functions.

Applying the Binary Search (Section 2.1) to software Devices Under Test (DUTs) results in several problems:

- Binary search takes a long time, 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 best practices frequently allow for tolerance of low but non-zero losses ([Y.1564], test-equipment manuals).
- The definition of throughput is not clear when trial results are inconsistent (e.g., when successive trials at the same -- or even a higher -- offered load yield different loss ratios, the classical throughput metric in [RFC1242] and [RFC2544] can no longer be pinned to a single, unambiguous value.)

To address these problems, early MLRsearch implementations employed the following enhancements:

1. Allow multiple short trials instead of one big trial per load.
   - Optionally, tolerate a percentage of trial results with higher loss.
2. Allow searching for multiple Search Goals (Section 4.6.7), with differing loss ratios.
   - Any trial result can affect each Search Goal in principle.

3. 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 at Maximum Offered Load (FRMOL), as defined in Section 3.6.2 of [RFC2285], to initialize bounds.
4. Clarify handling of inconsistent trial results.
   - Reported throughput should be smaller than the smallest load with high loss.
   - Measure smaller load candidates first.
5. Apply several time-saving load selection heuristics that deliberately prevent the bounds from narrowing unnecessarily.

Enhancements 1, 2, and partly 4 are formalized as the MLRsearch Specification within this document; other implementation details are out the scope.

The remaining enhancements are treated as implementation details, thus achieving high comparability without limiting future improvements.

MLRsearch configuration supports both conservative settings and aggressive settings. Results unconditionally compliant with [RFC2544] are possible with conservative enough settings but without much improvement on search duration and repeatability -- see "MLRsearch Compliant with RFC 2544" (Section 4.10.2). Conversely, aggressive settings lead to shorter search durations and better repeatability, but the results are not compliant with [RFC2544]. Exact settings are not specified, but see the discussion in "Overview of RFC 2544 Problems" (Section 2) for the impact of different settings on result quality.

This document does not change or obsolete any part of [RFC2544].

1.2. Positioning Within BMWG Methodologies

The Benchmarking Methodology Working Group (BMWG) produces recommendations (RFCs) that describe various benchmarking methodologies for use in a controlled laboratory environment. A large number of these benchmarks are based on the terminology from [RFC1242] and the foundational methodology from [RFC2544]. A common pattern has emerged where BMWG documents reference the methodology of [RFC2544] and augment it with specific requirements for testing particular network systems or protocols, without modifying the core benchmark definitions.

While BMWG documents are formal recommendations, they are widely treated as industry norms to ensure the comparability of results between different labs. The set of benchmarks defined in [RFC2544], in particular, became a de facto standard for performance testing. In this context, the MLRsearch Specification formally defines a new class of benchmarks that fits within the wider framework of [RFC2544]; see Section 4.1 ("Scope").

A primary consideration in the design of MLRsearch is the trade-off between configurability and comparability. The methodology's flexibility, especially the ability to define various sets of Search Goals, in supporting both single-goal and multiple-goal benchmarks in a unified way is powerful for detailed characterization and internal testing. However, this same flexibility is detrimental to inter-lab comparability unless a specific, common set of Search Goals is agreed upon.

Therefore, MLRsearch should not be seen as a direct extension nor a replacement for the [RFC2544] Throughput benchmark. Instead, this document provides a foundational methodology that future BMWG documents can use to define new, specific, and comparable benchmarks by mandating particular Search Goal configurations. For operators of existing test procedures, it is worth noting that many test setups measuring [RFC2544] Throughput can be adapted to produce results compliant with the MLRsearch Specification, often without affecting Trials, merely by augmenting the content of the final test report.

2. Overview of RFC 2544 Problems

This section describes the problems affecting usability of various performance testing methodologies, mainly the Binary Search (Section 2.1) for unconditionally compliant [RFC2544] throughput.

2.1. Binary Search

While [RFC2544] offers some flexibility when searching for throughput, a particular algorithm is frequently used as a starting point, as it is the simplest one among those that offer reasonable effectivity.

This algorithm is based on (balanced) binary search over sorted arrays but does not have a specific name when searching for throughput. "Trial duration" (Section 24 of [RFC2544]) mentions binary search only in quotes, without providing specifics. In this document, we call that algorithm the Binary Search, as that is the title of Section 12.3.2 of [TST009], which describes a variant of it.

Here is a simplified description of the algorithm:

- Initialize the lower-bound variable to a line-rate value.
- Initialize the upper-bound variable to a loss-free load value.
- Compute a midpoint, the arithmetic mean of current bound values.
- Run a single 60-second trial at the midpoint (for [RFC2544] unconditional compliance).
- If loss is zero, set the lower-bound to a midpoint value; else, set the upper bound to a midpoint value.
- Repeat (computing new midpoint) until the gap between the bounds meets the desired precision.
- Return the final lower-bound value as the throughput.

The description in [TST009] has two more requirements (stopping condition and rounding, both based on the Offered Load Step Size Parameter), but those are not required in this document.

Small modifications related to initial bound values are also allowed.

The load values currently held in the two variables are called "tightest bounds", especially when discussing older trial results (logically still bounds).

2.2. Long Search Duration

The proliferation 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 the DUT update and the frequency of running those tests. This makes the overall test execution time even more important than before.

The definition of throughput test methodology per [RFC2544] restricts the potential for time-efficiency improvements. The Binary Search (Section 2.1), when used in a manner unconditionally compliant with [RFC2544], is excessively slow due to two main factors.

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

Second, [RFC2544] does not specify any stopping condition for throughput search, so users of testing equipment implementing the procedure already have access to a limited trade-off between search duration and achieved precision, as each one of the full 60-second trials halves the interval of possible results.

As such, not many trials can be removed without a substantial loss of precision.

2.3. DUT in SUT

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

[RFC2285] defines these terms as follows.

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

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

For software-based data plane forwarding running on commodity x86/ARM CPUs, the SUT comprises not only the forwarding application itself, the DUT, but also the entire execution environment: host hardware, firmware and kernel/hypervisor services, as well as any other software workloads that share the same CPUs, memory, and I/O resources.

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

Some of this interference can be mitigated. For instance, in multi-core CPU systems, 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 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 it frequently lowers the observed SUT performance as observed in trial results.

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, and 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 minor noise events almost always occur during each trial, nudging the measured performance slightly below the theoretical maximum.

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 (based on the available literature at the time of writing).

If the SUT execution environments and any co-resident workloads place only negligible demands on SUT shared resources, so that the DUT remains the principal performance limiter, the DUT's ideal noiseless performance is defined as the noiseless end of the SUT performance spectrum.

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.

Improvements to the throughput search algorithm, aimed at better dealing with software networking SUT and DUT setups, should adopt methods that explicitly model SUT-generated noise, deriving surrogate metrics that approximate the (proxies for) DUT noiseless performance across a range of SUT noise-tolerance levels.

2.4. Repeatability and Comparability

[RFC2544] does not suggest repeating throughput search. Also, note that from simply one discovered throughput value, it cannot be determined how repeatable that value is. Unsatisfactory repeatability then leads to unacceptable comparability, as different benchmarking teams may obtain varying throughput values for the same SUT, exceeding the expected differences from search precision. Repeatability is also important when the test procedure is kept the same, but SUT is varied in small ways. For example, during development of software-based DUTs, repeatability is needed to detect small regressions.

[RFC2544] throughput requirements (60-second trial and no tolerance of a single frame loss) affect the throughput result 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 (Section 2.1) results tend to spread away from the noiseless end of SUT performance spectrum more frequently and more widely than shorter trials would, thus causing unacceptable 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 test methodology 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 also help 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., average and standard deviation and/or percentiles like p95).

This can be used for a subset of tests deemed more important, but it makes the search duration problem even more pronounced.

2.5. Throughput with Non-Zero Loss

Section 3.17 of [RFC1242] 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 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.

There are many motivations:

- Networking protocols tolerate frame loss better, compared to the time when [RFC1242] and [RFC2544] were specified.
- Increased link speeds require trials sending more frames within the same duration, increasing the chance of a small SUT performance fluctuation being enough to cause frame loss.
- Because noise-related drops usually arrive in small bursts, their impact on the trial's overall frame loss ratio is diluted by the longer intervals in which the SUT operates close to its noiseless performance; consequently, the averaged Trial Loss Ratio can still end up below the specified 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 (see definitions of Trial and Goal terms in "Trial Terms" (Section 4.5) and "Goal Terms" (Section 4.6)).

For more information, see Section 5 of [Lencze-Shima] (and the references there) for a few synthetic examples, confirming that each protocol and application can have different realistic loss ratio values.

Regardless of the validity of all similar motivations, support for non-zero loss goals makes a search algorithm applicable for a wider range of use cases than the approach defined in [RFC2544].

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 the noise has a large impact on the observed performance, which is not evident from the result of a single goal load search procedure.

It is easy to modify the Binary Search (Section 2.1) to find a lower bound for the load that satisfies a non-zero Goal Loss Ratio. But how to search for multiple goals at once is not that obvious; hence, the support for multiple Search Goals remains a problem.

At the time of writing, there does not seem to be a consensus in the industry on which loss ratio value is the best. For users, performance of higher protocol layers is important, for example, goodput of TCP connection (TCP throughput [RFC6349]), but the relationship between goodput and loss ratio is not simple. Refer to [Lencze-Kovacs-Shima] for examples of various corner cases, Section 3 of [RFC6349] 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.

2.6. 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) results in a lower Trial Loss Ratio.

The Binary Search (Section 2.1) 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 of [RFC2544], 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 of [RFC2544], 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.

A robust throughput search algorithm needs to decide how to continue the search in the presence of such inconsistencies. Definitions of throughput and its test methodology 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 that 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.

3. 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.

This document is categorized as an Informational RFC. While it does not mandate the adoption of the MLRsearch methodology, it uses the normative language of BCP 14 [RFC2119] [RFC8174] to provide an unambiguous specification. This ensures that if a test procedure or test report claims compliance with the MLRsearch Specification, it **MUST** adhere to all the absolute requirements defined herein. The use of normative language is intended to promote repeatable and comparable results among those who choose to implement this methodology.

4. MLRsearch Specification

This section provides all technical definitions needed for evaluating whether a particular test procedure complies with the MLRsearch Specification.

Some terms used in the specification are capitalized. This is a stylistic choice for this document, reminding the reader that the term is introduced, defined, or explained elsewhere in the document. Lowercase variants are equally valid.

This document does not separate terminology from methodology. Terms are fully specified and discussed in their own subsections, under sections with the word "Terms" in the title. This way, the list of terms is visible in the table of contents.

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 about how other components can use the defined quantity are also included in the discussion.

4.1. Scope

This document specifies the Multiple Loss Ratio search (MLRsearch) methodology. The MLRsearch Specification details a new class of benchmarks by listing all terminology definitions and methodology requirements. The definitions support "multi-goal" benchmarks, with "single-goal" as a subset.

The normative scope of this specification includes:

- The terminology for all required quantities and their attributes.
- An abstract architecture consisting of functional components (Manager, Controller, and Measurer) and the requirements for their inputs and outputs.

- The required structure and attributes of the Controller Input, including one or more Search Goal instances.
- The required logic for Load Classification, which determines whether a given Trial Load qualifies as a Lower Bound or an Upper Bound for a Search Goal.
- The required structure and attributes of the Controller Output, including a Goal Result for each Search Goal.

4.1.1. Relationship to RFC 2544

The MLRsearch Specification is an independent methodology and does not change or obsolete any part of [RFC2544].

This specification permits deviations from the Trial procedure as described in [RFC2544]. Any deviation from the procedure in [RFC2544] must be documented explicitly in the Test Report, and such variations remain outside the scope of the original benchmarks in [RFC2544].

A specific single-goal MLRsearch benchmark can be configured to be compliant with [RFC2544] Throughput, and most procedures reporting [RFC2544] Throughput can be adapted to also satisfy MLRsearch requirements for a specific search goal.

4.1.2. Applicability of Other Specifications

Methodology extensions from other BMWG documents that specify details for testing particular DUTs, configurations, or protocols (e.g., by defining a particular Traffic Profile) are considered orthogonal to MLRsearch and are applicable to a benchmark conducted using MLRsearch methodology.

4.1.3. Out of Scope

The following aspects are explicitly out of the normative scope of this document:

- This specification does not mandate or recommend any single, universal Search Goal configuration for all use cases. The selection of Search Goal parameters is left to the operator of the test procedure or may be defined by future specifications.
- The internal heuristics or algorithms used by the Controller to select Trial Input values (e.g., the load selection strategy) are considered implementation details.
- The potential for, and the effects of, interference between different Search Goal instances within a multiple-goal search are considered outside the normative scope of this specification.

4.2. Architecture Overview

Although the normative text only references terminology that has already been introduced, explanatory passages sometimes profit from terms that are defined later in the document. To keep the initial read-through clear, this informative section offers a concise, top-down sketch of the complete MLRsearch architecture.

The architecture is modelled as a set of abstract, interacting components. Information exchange between components is expressed in an imperative-programming style: One component "calls" another, supplying inputs (arguments) and receiving outputs (return values). This notation is purely conceptual; actual implementations need not exchange explicit messages. When the text contrasts alternative behaviors, it refers to the different implementations of the same component.

A test procedure is considered compliant with the MLRsearch Specification if it can be conceptually decomposed into the abstract components defined herein and if each component satisfies the requirements defined for its corresponding MLRsearch element.

The Measurer component is tasked to perform Trials, the Controller component is tasked to select Trial Durations and Loads, and the Manager component is tasked to preconfigure involved entities and to produce the Test Report. The Test Report explicitly states Search Goals (as Controller Input) and corresponding Goal Results (Controller Output).

This constitutes one benchmark (single-goal or multi-goal). Repeated or slightly differing benchmarks are realized by calling the Controller once for each benchmark.

The Manager calls a Controller once, and the Controller then invokes the Measurer repeatedly until the Controller decides it has enough information to return outputs.

The part during which the Controller invokes the Measurer is termed the "Search". Any work the Manager performs, either before invoking the Controller or after Controller returns, falls outside the scope of the Search.

The MLRsearch Specification prescribes Regular Goal Results (Section 4.8.4.1) and recommends corresponding search completion conditions. Irregular Goal Results (Section 4.8.4.2) are also allowed; they have different requirements, and their corresponding stopping conditions are out of scope. The Search Result is the combination of regular and irregular results for each goal.

Search Results are based on Load Classification. When measured enough, 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.

When the Relevant Lower Bound is close enough to the 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.

4.2.1. Test Report

A primary responsibility of the Manager is to produce a Test Report, which serves as the final and formal output of the test procedure.

This document does not provide a single, complete, normative definition for the structure of the Test Report. For example, the Test Report may contain results for a single benchmark, or it could aggregate results of many benchmarks.

Instead, normative requirements for the content of the Test Report are specified throughout this document in conjunction with the definitions of the quantities and procedures to which they apply. Readers should note that any clause requiring a value to be "reported" or "stated in the test report" constitutes a normative requirement on the content of this final artifact.

Even where not stated explicitly, the "Reporting format" paragraphs in [RFC2544] are still requirements on Test Reports if they apply to an MLRsearch benchmark.

4.2.2. Behavior Correctness

The 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.

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

The primary MLRsearch implementation, used as the prototype for this specification, is [PyPI-MLRsearch].

4.3. Quantities

The MLRsearch Specification uses a number of specific quantities; some of them can be expressed in several different units.

In general, the 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 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. A group of attribute values is called an instance of that composite quantity.

Some attributes may depend on others and can be calculated from other attributes. Such quantities are called derived quantities.

4.3.1. Current and Final Values

Some quantities are defined in a way that makes it possible to compute their values in the middle of a Search. Other quantities are specified so that their values can be computed only after a Search ends. Some quantities are important only after a Search ends, but their values are also computable before a Search ends.

For a quantity that is computable before 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, 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 quantities. As the naming suggests, the final value of "duration" is expected to be equal to the "final duration" value.

4.4. Existing Terms

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

- "Benchmarking Terminology for Network Interconnection Devices" [RFC1242] contains basic term definitions.
- "Benchmarking Terminology for LAN Switching Devices" [RFC2285] includes more terms and discussions, and it describes some known network benchmarking situations in a more precise way.
- "Benchmarking Methodology for Network Interconnect Devices" [RFC2544] contains discussions about terms and additional methodology requirements.

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

4.4.1. SUT

SUT is defined in Section 3.1.2 of [RFC2285] as follows.

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

Discussion:
: A SUT consisting of a single network device is allowed by this definition.

: In software-based networking, SUT may comprise a multitude of networking applications and the entire host hardware and software execution environment.

: SUT is the only entity that can be benchmarked directly, even though only the performance of some sub-components are of interest.

4.4.2. DUT

DUT is defined in Section 3.1.1 of [RFC2285] as follows.

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

Discussion:

Contrary to SUT, the DUT stimulus and response are frequently initiated and observed only indirectly, on different parts of SUT.

DUT, as a sub-component of SUT, is only indirectly mentioned in the MLRsearch Specification but is of key relevance for its motivation. The device can represent a software-based networking function running on commodity x86/ARM CPUs (vs. purpose-built ASICs, NPUs, and FPGAs).

A well-designed SUT should have the primary DUT as their performance bottleneck. The ways to achieve that are outside the scope of the MLRsearch Specification.

4.4.3. Trial

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

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 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).

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

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].

The definition describes some traits, not using capitalized verbs to signify the strength of the requirements. For the purposes of the MLRsearch Specification, the test procedure **MAY** deviate from the description in [RFC2544], but any such deviation **MUST** be described explicitly in the Test Report. It is still **RECOMMENDED** to not deviate from the description, as any deviation weakens comparability.

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

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

Some methodologies describe recurring tests. If those are based on Trials, they are treated as multiple independent Trials.

4.5. Trial Terms

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

4.5.1. Trial Duration

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

Discussion:
The value **MUST** be positive.

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

4.5.2. Trial Load

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

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

For specification purposes, it is assumed that this is a constant load by default, as specified in Section 3.4 of [RFC1242]. Informally, Traffic Load is a single number that can "scale" any traffic pattern as long as the intuition of load intended against a single interface can be applied.

It **MAY** be possible to use a Trial Load value to describe non-constant traffic (using average load when the traffic consists of repeated bursts of frames, e.g., as suggested in Section 21 of [RFC2544]). In the case of a non-constant load, the Test Report **MUST** explicitly mention exactly how non-constant the traffic is and how it reacts to a Traffic Load value. But the rest of the MLRsearch Specification assumes that is not the case, to avoid discussing corner cases (e.g., which values are possible within medium limitations).

Similarly, traffic patterns where different interfaces are subject to different loads **MAY** be described by a single Trial Load value (e.g., using the largest load among interfaces), but again, the Test Report **MUST** explicitly describe how the traffic pattern reacts to the Traffic Load value, and this specification does not discuss all the implications of that approach.

In the common case of bidirectional traffic, as described in "Bidirectional traffic" (Section 14 of [RFC2544]), Trial Load is the data rate per direction, half of the aggregate data rate.

Traffic patterns where a single Trial Load does not describe their scaling cannot be used for MLRsearch benchmarks.

Similarly to Trial Duration, some Measurers **MAY** limit the possible values of Trial Load. Contrary to Trial Duration, documenting such behavior in the test report is **OPTIONAL**. This is because the load differences are negligible (and frequently undocumented) in practice.

The Controller **MAY** select the 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., if there are no Trials yet or if there are multiple Trials, this document uses a shorthand "Load".

The test report **MAY** present the aggregate load across multiple interfaces, treating it as the same quantity expressed using different units. Each reported Trial Load value **MUST** unambiguously state whether it refers to (i) a single interface, (ii) a specified subset of interfaces (such as all logical interfaces mapped to one physical port), or (iii) the total across every interface. For any aggregate load value, the report **MUST** also give the fixed conversion factor that links the per-interface and multi-interface load values.

The per-interface value remains the primary unit, consistent with the prevailing practice described in [RFC1242], [RFC2544], and [RFC2285].

The previous paragraph also applies to other terms related to Load. For example, tests with symmetric bidirectional traffic can report load-related values as "bidirectional load" (double of "unidirectional load").

4.5.3. 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 **MUST NOT** add optional attributes into Trial Input. This improves interoperability between various implementations of a Controller and a Measurer.

Note 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 an integer number of frames), but questions around those offered quantities are generally outside of the scope of this document.

4.5.4. Traffic Profile

Definition:

Traffic Profile is a composite quantity containing all attributes other than Trial Load and Trial Duration that are needed for the 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.

Specification of traffic properties included in the Traffic Profile is the responsibility of the Manager, but the specific configuration mechanisms are outside of the scope of this document.

Informally, implementations of the Manager and the Measurer must be aware of their common set of capabilities, so that the Traffic Profile instance uniquely defines the traffic during the Search. Typically, Manager and Measurer implementations are tightly integrated.

Integration efforts between independent Manager and Measurer implementations are outside of the scope of this document. An example standardization effort is described in [Vassilev].

Examples of traffic properties include:

- Data link frame size:
  - Fixed sizes as listed in Section 3.5 of [RFC1242] and in Section 9 of [RFC2544]
  - Internet Mix (IMIX) mixed sizes as defined in [RFC6985]
- Frame formats and protocol addresses:
  - Sections 8 and 12 of [RFC2544] and Appendix C of [RFC2544]

- Symmetric bidirectional traffic:
  - Section 14 of [RFC2544]

Other traffic properties that need to somehow be specified in Traffic Profile, and **MUST** be mentioned in Test Report if they apply to the benchmark, include:

- bidirectional traffic from Section 14 of [RFC2544],
- fully meshed traffic from Section 3.3.3 of [RFC2285],
- modifiers from Section 11 of [RFC2544], and
- IP version mixing from Section 5.3 of [RFC8219].

4.5.5. Trial Forwarding Ratio

Definition:
The Trial Forwarding Ratio is a dimensionless 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 received by a SUT from the tester". This **SHOULD** be the default interpretation. However, if this is not the case, the test report **MUST** describe the Traffic Profile in a detail sufficient to imply how the Trial Forwarding Ratio should be calculated.

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

Note that, contrary to Load terms, frame counts used to compute the Trial Forwarding Ratio are generally aggregates over all SUT output interfaces, as most test procedures verify all outgoing frames. The procedure for [RFC2544] Throughput counts received frames, so it implies bidirectional counts for bidirectional traffic, even though the final value is the "rate" that is still per-interface. 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%).

In future extensions, more general ways to compute the Trial Forwarding Ratio may be allowed, but the current MLRsearch Specification relies on this specific averaged counters approach.

4.5.6. 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 in Section 3.6 of [RFC1242]. 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 averaged frame counts when more than one data stream is present.

4.5.7. Trial Forwarding Rate

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

Discussion:
This quantity differs from the forwarding rate described in Section 3.6.1 of [RFC2285]. Under the method described in [RFC2285], each output interface is measured separately, so every interface may report a distinct rate. The Trial Forwarding Rate, by contrast, uses a single set of frame counts and therefore yields one value that represents the whole system while still preserving the direct link to the per-interface load.

When the Traffic Profile is symmetric and bidirectional, as defined in Section 14 of [RFC2544], the Trial Forwarding Rate is numerically equal to the arithmetic average of the individual per-interface forwarding rates that would be produced by the procedure described in [RFC2285].

For more complex traffic patterns, such as many-to-one, as mentioned in "Partially meshed traffic" (Section 3.3.2 of [RFC2285]), the meaning of Trial Forwarding Rate is less straightforward. For example, if two input interfaces receive one million frames per second (fps) each and a single interface outputs 1.4 million frames per second (fps), the Trial Load is 1 million fps, the Trial Loss Ratio is 30%, and the Trial Forwarding Rate is 0.7 million fps.

Because this rate is anchored to the Load defined for one interface, a test report **MAY** show it either as the single averaged figure just described or as the sum of the separate per-interface forwarding rates. For the example above, the aggregate Trial Forwarding Rate is 1.4 million fps.

4.5.8. Trial Effective Duration

Definition:
The Trial Effective Duration is a time quantity related to a Trial. By default, it is 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.

The Trial Effective Duration may be any positive 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 time-critical benchmarks designed to manage the overall search duration, rather than solely the traffic portion of it. An approach is 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, when rounding the Trial Duration value.

4.5.9. Trial Output

Definition:
Trial Output is a composite quantity consisting of several attributes. The required attributes are Trial Loss Ratio, Trial Effective Duration, and Trial Forwarding Rate.

Discussion:
When referring to more than one trial, the plural term "Trial Outputs" is used to collectively describe multiple Trial Output instances.

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

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

While Section 3.5.2 of [RFC2285] requires the Offered Load value to be reported for forwarding rate measurements, it is not required in the MLRsearch Specification, as search results do not depend on it.

4.5.10. Trial Result

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

Discussion:
When referring to more than one trial, the plural term "Trial Results" is used to collectively describe multiple Trial Result instances.

4.6. 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 in this section relate the attributes to concepts mentioned in "Overview of RFC 2544 Problems" (Section 2), but these discussion paragraphs are short and informal, and they mostly reference later sections, where the impact on search results is discussed after introducing the complete set of auxiliary terms.

4.6.1. Goal Final Trial Duration

Definition:
: This is the minimal value for Trial Duration that must be reached. The value **MUST** be positive.

Discussion:
: Certain trials must reach this minimum duration before a load can be classified as a lower bound.

  The Controller may choose shorter durations; results of those may be enough for classification as an Upper Bound.

  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.

4.6.2. Goal Duration Sum

Definition:
: This is a threshold value for a particular sum of Trial Effective Duration values. The value **MUST** be positive.

Discussion:
: Informally, this prescribes the sufficient 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.

  Refer to "MLRsearch Compliant with TST009" (Section 4.10.3) 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 the Goal Duration Sum measuring a Load value when the results are particularly one-sided, but also, the Search can spend more than the Goal Duration Sum measuring a Load when the results are balanced and include trials shorter than the Goal Final Trial Duration.

4.6.3. Goal Loss Ratio

Definition:
This is a threshold value for Trial Loss Ratio values. The value **MUST** be non-negative and smaller than one.

Discussion:
A trial with the 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 2.5) 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 may be more complicated than the mere comparison of the Trial Loss Ratio to the Goal Loss Ratio.

4.6.4. Goal Exceed Ratio

Definition:
This is 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 the Trial Results with the Trial Loss Ratio above the 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 the exceed ratio is 0.5 (50%), as in practice that value leads to the smallest variation in overall Search Duration.

Refer to "Exceed Ratio and Multiple Trials" (Section 5.4) for more details.

4.6.5. Goal Width

Definition:
This is 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** provide the Controller with other 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 the Goal Width.

It is **RECOMMENDED** to express the Goal Width as a relative difference and set it to a value not lower than the Goal Loss Ratio.

Refer to "Generalized Throughput" (Section 5.6) for more elaboration on the reasoning.

4.6.6. Goal Initial Trial Duration

Definition:
This is the minimal value for the Trial Duration suggested to use for this goal. If present, this value **MUST** be positive.

Discussion:
This is an example of an optional Search Goal.

A typical 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, selected while focusing on a different Search Goal. Such results **MUST** be still accepted by the Load Classification logic.

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

4.6.7. Search Goal

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

The required attributes are Goal Final Trial Duration, Goal Duration Sum, Goal Loss Ratio, and Goal Exceed Ratio.

The optional attributes are Goal Initial Trial Duration and Goal Width.

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

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

Refer to "Compliance" (Section 4.10) for important Search Goal settings.

4.6.8. Controller Input

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 an implementation even if they are not required by the MLRsearch 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.

4.6.8.1. 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.

If the Max Load value is provided, the Controller **MUST NOT** select Trial Load values larger than that value.

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 Section 20 of [RFC2544] and (as maximum offered load) by Section 3.5.3 of [RFC2285], some implementations may detect or discover it (instead of requiring a user-supplied value).

In the MLRsearch Specification, one reason for listing the Relevant Upper Bound (Section 4.8.1) as a required attribute is that it makes the search result independent of the Max Load value.

Given that Max Load is a quantity based on Load, Test Report **MAY** express this quantity using multi-interface values, as the sum of per-interface maximal loads.

4.6.8.2. 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.

If the Min Load value is provided, the Controller **MUST NOT** select Trial Load values smaller than that value.

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 the shortest allowed Trial Duration, so that the Trial Loss Ratio is always well-defined and the implementation can apply the relative Goal Width safely.

Given that Min Load is a quantity based on Load, Test Report **MAY** express this quantity using multi-interface values, as the sum of per-interface minimal loads.

4.7. 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.

4.7.1. 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 a Search.

4.7.1.1. High-Loss Trial

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

4.7.1.2. Low-Loss Trial

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

4.7.1.3. Short Trial

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

4.7.1.4. Full-Length Trial

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

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

4.7.1.5. Long Trial

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

4.7.2. 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.

4.7.2.1. Upper Bound

Definition:

> A Load value is called an Upper Bound if and only if it is classified as such by a load classification code (Appendix A) 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. In this context, "certain to fail" relates to any possible results within the time remaining till the 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 a Goal Duration Sum of trials are measured (e.g., because another Search Goal needs more trials at this load). Informally, the previous Upper Bound got invalidated. In practice, the Load frequently becomes a Lower Bound (Section 4.7.2.2) instead.

4.7.2.2. Lower Bound

Definition:

A Load value is called a Lower Bound if and only if it is classified as such by a load classification code (Appendix A) 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 the 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 a Goal Duration Sum of trials are measured (e.g., because another Search Goal needs more trials at this load). Informally, the previous Lower Bound got invalidated. In practice, the Load frequently becomes an Upper Bound (Section 4.7.2.1) instead.

4.7.2.3. 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 different Goal Final Trial Duration values.

4.8. 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.

4.8.1. 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 the Relevant Upper Bound does exist, it is not affected by the Max Load value.

Given that the Relevant Upper Bound is a quantity based on Load, Test Report **MAY** express this quantity using multi-interface values, as the sum of per-interface loads.

4.8.2. 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 the Relevant Lower Bound is a quantity based on Load, Test Report **MAY** express this quantity using multi-interface values, as the sum of per-interface loads.

4.8.3. Conditional Throughput

Definition:
Conditional Throughput is a value computed at the Relevant Lower Bound according to the algorithm defined in "Conditional Throughput Code" (Appendix B).

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 a Search.

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

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

Given that Conditional Throughput is a quantity based on Load, Test Report **MAY** express this quantity using multi-interface values, as the sum of per-interface forwarding rates.

4.8.4. Goal Results

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

4.8.4.1. 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.

Discussion:
Implementations **MAY** add their own attributes.

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

In general, stopping conditions for the corresponding Search Goal **MUST** be satisfied to produce a Regular Goal Result. Specifically, if 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.

For stopping conditions, refer to "Goal Width" (Section 4.6.5) and "Stopping Conditions and Precision" (Section 5.2).

4.8.4.2. Irregular Goal Result

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, 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 a 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 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 Test Report purposes.

If Test Report displays some Irregular Goal Result attribute values, they **MUST** be clearly marked as coming from irregular results.

The implementation **MAY** define additional attributes, for example, explicit flags for expected situations, so the Manager logic can be simpler.

4.8.4.3. 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 of distinguishing whether the instance is regular or not.

4.8.5. Search Result

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

Discussion:
As an alternative to mapping, the Search Result may be represented as an ordered list of Goal Result instances that appears in the exact sequence of their corresponding Search Goal instances.

When the Search Result is expressed as a mapping, it **MUST** contain an entry for every Search Goal instance supplied in the 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, if two goals differ only in the Goal Width value, and the relevant bound values are close enough according to only one of them.

4.8.6. 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 attribute.

Discussion:
The MLRsearch implementation **MAY** return additional data in the Controller Output, e.g., the number of trials performed and the total Search Duration.

4.9. Architecture Terms

The MLRsearch architecture consists of three main system components: the Manager, the Controller, and the Measurer. The components were introduced in "Architecture Overview" (Section 4.2), and the following subsections finalize their definitions using terms from previous sections.

Note that 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 and interfaces between components are left unspecified. For example, when the MLRsearch Specification mentions "Controller calls Measurer", it is possible that the Controller notifies the Manager to call the Measurer indirectly instead. In doing so, the Measurer implementations can be fully independent from the Controller implementations, e.g., developed in different programming languages.

4.9.1. Measurer

Definition:
The Measurer is a functional element that, when called with a Trial Input (Section 4.5.3) instance, performs one Trial (Section 4.4.3) and returns a Trial Output (Section 4.5.9) instance.

Discussion:
This definition assumes the Measurer is already initialized. In practice, there may be additional steps before the Search, 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 requirements and assumptions present in the MLRsearch Specification, e.g., the Trial Forwarding Ratio not being larger than one.

Implementers have some freedom. For example, Section 10 of [RFC2544] 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., connect the 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 use that value as the Max Load value. When testing the real SUT, it is **RECOMMENDED** to turn any severe deviation between the Intended Load and the Offered Load into the increased Trial Loss Ratio.

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

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

4.9.2. Controller

Definition:
The Controller is a functional element that, upon receiving a Controller Input instance, repeatedly generates Trial Input instances for the Measurer and collects the corresponding Trial Output instances. This cycle continues until the stopping conditions are met, at which point, the Controller produces a final Controller Output instance and terminates.

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.

4.9.3. Manager

Definition:
The Manager is a functional element that is responsible for provisioning other components, calling a Controller component once, and for creating the test report following the reporting format as defined in Section 26 of [RFC2544].

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] already puts requirements on SUT setups:

> It is expected that all of 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, including the description of a "default" configuration common for most tests and configuration changes if required by a specific test.

For example, Section 5.1.1 of [RFC5180] 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 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. 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.

A "user" -- human or automated -- invokes the Manager once to launch a single Search and receive its report. Every new invocation is treated as a fresh, independent Search; how the system behaves across multiple calls (for example, combining or comparing their results) is explicitly out of scope for this document.

4.10. Compliance

This section discusses compliance relations between MLRsearch and other test procedures.

4.10.1. Test Procedure Compliant with MLRsearch

Any networking measurement setup that could be understood as consisting of functional elements satisfying requirements for the Measurer, the Controller, and the Manager is compliant with the MLRsearch Specification.

These components can be seen as abstractions present in any testing procedure. For example, there can be a single component acting as both the Manager and the Controller, but 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 an MLRsearch architecture if there is enough data to reconstruct the Relevant Upper Bound.

Refer to "MLRsearch Compliant with RFC 2544" (Section 4.10.2) for an equivalent Search Goal.

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

4.10.2. MLRsearch Compliant with RFC 2544

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

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

The Goal Loss Ratio and Goal Exceed Ratio attributes are enough to make the Search Goal conditionally compliant. Adding Goal Final Trial Duration makes the Search Goal unconditionally compliant.

The Goal Duration Sum prevents MLRsearch from repeating zero-loss Full-Length Trials.

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

A non-zero exceed ratio is not strictly disallowed, but it could needlessly prolong the search when Low-Loss short trials are present.

4.10.3. 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].

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

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

For example, for the "max(r) = 2" variant, the following Search Goal instance should be used to get a 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 60-second 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 also needs to perform the second trial to classify that load. The Goal Duration Sum is twice as long as the Goal Final Trial Duration, so a third full-length trial is never needed.

5. Methodology Rationale and Design Considerations

This section explains the "why" behind MLRsearch. Building on the normative specification in "MLRsearch Specification" (Section 4), it contrasts MLRsearch with the classic single-ratio Binary Search (Section 2.1) in [RFC2544] and walks through the key design choices: search mechanics, stopping-rule precision, loss inversion for multiple goals, exceed-ratio handling, short-trial strategies, and the generalized throughput concept. Together, these considerations show how the methodology reduces test time, supports multiple loss ratios, and improves repeatability.

5.1. Binary Search Commonalities

A typical search implementation for [RFC2544], such as Binary Search (Section 2.1), tracks only the two tightest bounds (in variables "lower-bound" and "upper-bound"). 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 is used to confirm Min Load is a Lower Bound.

Then, 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 a one-to-one correspondence between the number of Trials and the 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

The 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 indirect control of the Search Duration.

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

Due to various possible time optimizations, there is no 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 biggest difference between MLRsearch and Binary Search (Section 2.1) 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.

5.3.1. Single Goal and Hard Bounds

Each bound in Binary Search (Section 2.1) 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

The MLRsearch Specification supports multiple Search Goals, making the search procedure more complicated compared to Binary Search (Section 2.1) with a single goal, but most of the complications do not affect the final results much, except for one phenomenon: Loss Inversion.

Depending on Search Goal attributes, Load Classification results may be resistant to small amounts of Inconsistent Trial Results (Section 2.6). 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. 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, such as: 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 the definitions of Relevant Upper Bound (Section 4.8.1) and Relevant Lower Bound (Section 4.8.2).

The Relevant Upper Bound (for a 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 (Section 2.1). 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, this means that 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 a high Trial Loss Ratio) are affected by infrequent effects, causing unsatisfactory repeatability of [RFC2544] Throughput results. Refer to "DUT in SUT" (Section 2.3) for a discussion about noiseful and noiseless ends of the SUT performance spectrum. 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 4.6.4). It tells MLRsearch what ratio of trials (more specifically, what ratio of Trial Effective Duration seconds) can have a Trial Loss Ratio (Section 4.5.6) larger than the Goal Loss Ratio (Section 4.6.3) and still be classified as a Lower Bound (Section 4.7.2.2).

A 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 4.6.2) 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 as either an Upper Bound or a Lower Bound for that Search Goal instance.

5.5. Short Trials and Duration Selection

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

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

An MLRsearch implementation **MAY** expose configuration parameters that decide whether, when, and how short trial durations are used. The exact heuristics and controls are left to the discretion of the implementer.

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).

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 the Controller to be the same, e.g., by setting any Goal Initial Trial Duration to be a single value also used in all Goal Final Trial Duration attributes.

5.6. Generalized Throughput

Because 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 4.6.3), 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] is better, but it is not obvious how to generalize it for Loads with multiple Trials and a non-zero Goal Loss Ratio.

The clearest illustration -- and the chief reason for adopting a generalized throughput definition -- is the presence of a hard performance limit.

5.6.1. Hard Performance Limit

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

Those limitations should be known and provided as Max Load (Section 4.6.8.1).

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.

Consider that the hard limit is at one 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 the Relevant Lower Bound of 100.5025 Mfps.

Reporting a throughput that exceeds the SUT's verified hard limit is counterintuitive. Accordingly, the [RFC2544] Throughput metric should be generalized -- rather than relying solely on the Relevant Lower Bound -- to reflect realistic, limit-aware performance.

MLRsearch defines one such generalization, the Conditional Throughput (Section 4.8.3). It is the Trial Forwarding Rate from one of the Full-Length Trials performed at the Relevant Lower Bound. For the algorithm to determine exactly which trial, see "Conditional Throughput Code" (Appendix B).

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

5.6.2. Performance Variability

With a non-zero Goal Loss Ratio, and without hard performance limits, Low-Loss trials at the same Load may achieve different Trial Forwarding Rate values simply 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 the Relevant Lower Bound are equal to the Goal Loss Ratio), one can prove that Conditional Throughput values may have a relative difference up to the Goal Loss Ratio.

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 counterintuitive, considering they come from the same Search. 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.

Although Conditional Throughput can fluctuate from one run to the next, it still offers a more discriminating basis for comparison than the Relevant Lower Bound -- particularly when deterministic load selection yields the same Lower Bound value across multiple runs.

6. MLRsearch Logic and Example

This section uses informal language to describe two aspects of MLRsearch logic: Load Classification and Conditional Throughput, reflecting formal pseudocode representation provided in "Load Classification Code" (Appendix A) and "Conditional Throughput Code" (Appendix B). This is followed by an example search.

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

6.1. Load Classification Logic

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

- Collect Trial Results:
  - Take all Trial Result instances (I) measured at a given load.
- Aggregate Trial Durations:
  - 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).
- Derive goal-based ratios:
  - Subceed ratio (T) is one minus the Goal Exceed Ratio (I).
  - Exceed coefficient (T) is the Goal Exceed Ratio divided by the subceed ratio.
- Balance short-trial effects:
  - 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 the excess sum.
- Compute effective duration totals:
  - 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.
- Estimate exceed ratios:
  - 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.
- Classify the Load:
  - 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

- Collect Trial Results:
  - Take all Trial Result instances (I) measured at a given Load.
- Sum of Full-Length Durations:
  - 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).
- Derive initial thresholds:
  - Subceed ratio (T) is one minus the Goal Exceed Ratio (I) is called.
  - Remaining sum (T) is initially the full-length sum multiplied by the subceed ratio.
  - Current loss ratio (T) is initially 100%.
- Iterate through ordered trials:
  - For each full-length trial result, sorted in increasing order by Trial Loss Ratio:
    - If the remaining sum is not larger than zero, exit the loop.
    - Set the current loss ratio to this trial's Trial Loss Ratio (I).
    - Decrease the remaining sum by this trial's Trial Effective Duration (I).
- Compute Conditional Throughput:
  - Current forwarding ratio (T) is one minus the current loss ratio.
  - Conditional Throughput (T) is the current forwarding ratio multiplied by the Load value.

6.2.1. Conditional Throughput and Load Classification

Conditional Throughput and results of Load Classification overlap but are not identical.

- When a load is marked as a Relevant Lower Bound, its Conditional Throughput is taken from a trial whose loss ratio never exceeds the Goal Loss Ratio.
- The reverse is not guaranteed: If the Goal Width is narrower than the Goal Loss Ratio, Conditional Throughput can still end up higher than the Relevant Upper Bound.

6.3. SUT Behaviors

In "DUT in SUT" (Section 2.3), the notion of noise is introduced. This section uses new terms to describe possible SUT behaviors more precisely.

From a measurement point of view, noise is visible as inconsistent trial results. See "Inconsistent Trial Results" (Section 2.6) 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 a "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 assess the probability of each outcome.

6.3.2. Exceed Probability

When the Controller selects a 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 a 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 reason users may want to set a large Goal Final Trial Duration.

6.3.3.2. Mild Increase

Short trials are slightly less likely to exceed the loss-ratio limit, but the improvement is modest. This mild benefit is typical when noise is dominated by rare, large loss spikes: During a full-length trial, the good-performing periods cannot fully offset the heavy frame loss that occurs in the brief low-performing bursts.

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 the Goal Loss Ratio is more than zero (otherwise, other parts cannot compensate at all).

6.3.3.4. Decrease

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

7. IANA Considerations

This document has no IANA actions.

8. Security Considerations

Benchmarking activities as described in this document are limited to the 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" basis, relying solely on measurements observable external to the DUT/SUT.

The DUT/SUT **SHOULD NOT** include features that serve only to boost benchmark scores -- such as a dedicated "fast-track" test mode that is never used in normal operation.

Any implications for network security arising from the DUT/SUT **SHOULD** be identical in the lab and in production networks.

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, and your continuous unwavering encouragement full of empathy and an energizing positive attitude. Al, you are dearly missed.

Thanks to Gábor Lencse, Giuseppe Fioccola, Carsten Rossenhövel, 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 the Linux Foundation FastData I/O (FD.io) project, specifically committers and contributors to the two core projects of FD.io: VPP and CSIT. It was there where the need for MLRsearch originally came up, and it was in FD.io CSIT labs where the first MLRsearch open-source code got prototyped and then over the years productized. Thanks also goes to Alec Hothan of the OPNFV NFVbench project for a thorough review and numerous useful comments and suggestions in the earlier draft versions of this document.

We are equally indebted to Mohamed Boucadair for a very thorough and detailed AD review, for providing many good comments and suggestions, for helping us make this document complete.

Our appreciation is also extended to Shawn Emery, Yoshifumi Nishida, David Dong, Nabeel Cocker, Lars Eggert, Jen Linkova, Mike Bishop, and Éric Vyncke for their reviews and valuable comments.

10. References

10.1. 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>.

[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>.

10.2. 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] Lencse, G., Kovács, Á., and K. Shima, "Gaming with the Throughput and the Latency Benchmarking Measurement Procedures of RFC 2544", International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems, vol. 9, no. 2, pp. 10-17, DOI 10.11601/ijates.v9i2.288, June 2020, <https://doi.org/10.11601/ijates.v9i2.288>.

[Lencze-Shima] Lencse, G. and K. Shima, "An Upgrade to Benchmarking Methodology for Network Interconnect Devices", Work in Progress, Internet-Draft, draft-lencse-bmwg-rfc2544-bis-00, 20 May 2020, <https://datatracker.ietf.org/doc/html/draft-lencse-bmwg-rfc2544-bis-00>.

[Ott-Mathis-Semke-Mahdavi] Mathis, M., Semke, J., Mahdavi, J., and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", ACM SIGCOMM Computer Communication Review, vol. 27, no. 3, pp. 67-82, July 1997, <https://www.cs.cornell.edu/people/egs/cornellonly/syslunch/fall02/ott.pdf>.

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

[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>.

[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>.

[RFC6985] Morton, A., "IMIX Genome: Specification of Variable Packet Sizes for Additional Testing", RFC 6985, DOI 10.17487/RFC6985, July 2013, <https://www.rfc-editor.org/info/rfc6985>.

[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>.

[TST009] ETSI, "Network Functions Virtualisation (NFV) Release 3; Testing; Specification of Networking Benchmarks and Measurement Methods for NFVI", ETSI GS NFV-TST 009 V3.4.1, December 2020, <https://www.etsi.org/deliver/etsi_gs/NFV-TST/001_099/009/03.04.01_60/gs_NFV-TST009v030401p.pdf>.

[Vassilev] Vassilev, V., "A YANG Data Model for Network Tester Management", Work in Progress, Internet-Draft, draft-ietf-bmwg-network-tester-cfg-10, 7 April 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-bmwg-network-tester-cfg-10>.

[Y.1564] ITU-T, "Ethernet service activation test methodology", ITU-T Recommendation Y.1564, February 2016, <https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-Y.1564-201602-I!!PDF-E&type=items>.

Appendix A. Load Classification Code

This appendix specifies how to perform the Load Classification.

Any Trial Load value can be classified, according to a given Search Goal (Section 4.6.7) 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.

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

Although presented as pseudocode, the listing is syntactically valid Python and can be executed without modification.

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 to fit expressions in one line. Namely, variables holding sum quantities end in "*s" instead of "_sum", and variables holding effective quantities start with "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 the Trial Duration at least equal to the Goal Final Trial Duration and with the 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 the Trial Duration at least equal to the Goal Final Trial Duration and with the 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 the Trial Duration shorter than the Goal Final Trial Duration and with the 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 the Trial Duration shorter than the Goal Final Trial Duration and with the Trial Loss Ratio higher than the Goal Loss Ratio (across Short High-Loss Trials).

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

```
<CODE BEGINS>
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
optimistic_is_lower = effect_high_loss_s <= quantile_duration_s

<CODE ENDS>
```

Appendix B. Conditional Throughput Code

This section specifies an example of how to compute Conditional Throughput, as referred to in "Conditional Throughput" (Section 4.8.3).

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 that computes a value stored as the `conditional_throughput` variable.

Although presented as pseudocode, the listing is syntactically valid Python and can be executed without modification.

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 with "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 the Trial Duration at least equal to the Goal Final Trial Duration and with the 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 the Trial Duration at least equal to the Goal Final Trial Duration and with the 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 the Trial Duration at least equal to the Goal Final Trial Duration (all Full-Length Trials), sorted by increasing the 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 is at least one Trial Result measured at a given Load.

```
<CODE BEGINS>
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)

<CODE ENDS>
```

Appendix C. Example Search

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, 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 the 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., MLRsearch will look for a Lower Bound smaller than any Upper Bound found.

At any given moment, exactly one Search Goal is designated as in focus. This designation affects only the Trial Duration chosen for new trials; it does not alter the rest of the decision logic.

An MLRsearch implementation is free to evaluate several goals simultaneously -- the "focus" mechanism is optional and appears here only to show that a load can still be classified against goals that are not currently in focus.

C.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.

C.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 the 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 a Trial Duration of 1 second, which is a Short Trial for 3 of the Search Goals, but it is a Full-Length Trial for the "1s final" goal.
- Similarly, when a name contains "long", it means the added trial had a Trial Duration of 60 seconds, which is a Full-Length Trial for 3 goals but a Long Trial for the "1s final" goal.
- When a name contains "good", it means the added trial is a Low-Loss Trial for all the goals.
- When a name contains "short bad", it means the added trial is a 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.

C.3. Load Classification Computations

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

C.3.1. Point 1

This is the "first short good" point. The 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

Goal name	RFC2544	TST009	1s final	20% exceed
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

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

C.3.2. Point 2

This is the "first short bad" point. The 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

Goal name	RFC2544	TST009	1s final	20% exceed
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, the "RFC2544" goal must have a mild or strong increase of exceed probability, so the one lossy trial would be lossy even if measured at a 60-second duration. Due to zero exceed ratio, one High-Loss Trial is enough to preclude this Load from becoming a Lower Bound for the "RFC2544" goal. That is why this Load is classified as an Upper Bound for the "RFC2544" goal this early.

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

C.3.3. Point 3

This is the "last short bad" point. The 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 low-loss sum	0s	0s	59s	0s
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

Goal name	RFC2544	TST009	1s final	20% exceed
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 the "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" goal started to classify this load as an Upper Bound somewhere between points 2 and 3.

C.3.4. Point 4

This is the "last short good" point. The 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

Goal name	RFC2544	TST009	1s final	20% exceed
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 a 50% exceed ratio. This shows time savings are not guaranteed.

C.3.5. Point 5

This is the "first long bad" point. The 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%

Goal name	RFC2544	TST009	1s final	20% exceed
Classification Result	Upper Bound	Undecided	Lower Bound	Lower Bound

Table 5

As designed for the "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 the Goal Loss Ratio is zero, it is not 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 the optimistic exceed ratio is still only 50%.

But the 0.1% Trial Loss Ratio is lower than the "20% exceed" Goal Loss Ratio, so this unexpected Full-Length Low-Loss Trial changed the classification result of this Load to Lower Bound.

C.3.6. Point 6

This is the "first long good" point. The 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%

Goal name	RFC2544	TST009	1s final	20% exceed
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; the search may end. Or more realistically, it can focus on larger loads only, as the three goals will want an Upper Bound (unless this Load is Max Load).

C.4. Conditional Throughput Computations

At the end of this hypothetical search, the "RFC2544" goal labels the load as an Upper Bound, making it ineligible for Conditional Throughput calculations. By contrast, the other three goals treat the same load as a Lower Bound; if it is also accepted as their Relevant Lower Bound, Conditional Throughput values can be computed for each of them.

(The load under discussion is one million frames per second.)

C.4.1. Goal 2

The Conditional Throughput is computed from a sorted list of Full-Length Trial results. As the "TST009" Goal Final Trial Duration is 60 seconds, only two of 122 Trials are considered Full-Length Trials. One has a 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 is initially 0.5x12s = 60 seconds.
- Current loss ratio is initially 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 loss 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.

C.4.2. Goal 3

The "1s final" has a 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, and
- 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 is initially 0.5x240s = 120 seconds.
- Current loss ratio is initially 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, duration of 60x1s + 1x60s has been subtracted from 120s, leaving 0s.
- For 62-th result (duration 60s, loss 0.1%):
  - Remaining sum is not larger than zero, exiting the loop.
- Current loss 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.

C.4.3. Goal 4

The Conditional Throughput is computed from a 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 a 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 is initially 0.8x120s = 96 seconds.
- Current loss ratio is initially 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 remaining sum is not larger than zero), exiting loop.
- Current loss 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 a stricter Goal Exceed Ratio, this Conditional Throughput is smaller than Conditional Throughput of the other two goals.

Authors' Addresses

Maciek Konstantynowicz
Cisco Systems
Email: mkonstan@cisco.com

Vratko Polak
Cisco Systems
Email: vrpolak@cisco.com