IPPM standard advancement testing
draft-ietf-ippm-metrictest-04

Abstract

This document specifies tests to determine if multiple independent instantiations of a performance metric RFC have implemented the specifications in the same way. This is the performance metric equivalent of interoperability, required to advance RFCs along the standards track. Results from different implementations of metric RFCs will be collected under the same underlying network conditions and compared using state of the art statistical methods. The goal is an evaluation of the metric RFC itself, whether its definitions are clear and unambiguous to implementors and therefore a candidate for advancement on the IETF standards track.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

The Internet Standards Process [RFC2026](#) *[RFC2026]* requires that for a IETF specification to advance beyond the Proposed Standard level, at least two genetically unrelated implementations must be shown to interoperate correctly with all features and options. This requirement can be met by supplying:

    *evidence that (at least a sub-set of) the specification has been implemented by multiple parties, thus indicating adoption by the IETF community and the extent of feature coverage.

    *evidence that each feature of the specification is sufficiently well-described to support interoperability, as demonstrated through testing and/or user experience with deployment.

In the case of a protocol specification, the notion of "interoperability" is reasonably intuitive - the implementations must successfully "talk to each other", while exercising all features and options. To achieve interoperability, two implementors need to interpret the protocol specifications in equivalent ways. In the case of IP Performance Metrics (IPPM), this definition of interoperability is only useful for test and control protocols like the One-Way Active Measurement Protocol, OWAMP [[RFC4656]](#), and the Two-Way Active Measurement Protocol, TWAMP [[RFC5357]](#).
A metric specification RFC describes one or more metric definitions, methods of measurement and a way to report the results of measurement. One example would be a way to test and report the One-way Delay that data packets incur while being sent from one network location to another, One-way Delay Metric.
In the case of metric specifications, the conditions that satisfy the "interoperability" requirement are less obvious, and there was a need for IETF agreement on practices to judge metric specification "interoperability" in the context of the IETF Standards Process. This memo provides methods which should be suitable to evaluate metric specifications for standards track advancement. The methods proposed here MAY be generally applicable to metric specification RFCs beyond those developed under the IPPM Framework [[RFC2330]](#).
Since many implementations of IP metrics are embedded in measurement systems that do not interact with one another (they were built before OWAMP and TWAMP), the interoperability evaluation called for in the IETF standards process cannot be determined by observing that

independent implementations interact properly for various protocol exchanges. Instead, verifying that different implementations give statistically equivalent results under controlled measurement conditions takes the place of interoperability observations. Even when evaluating OWAMP and TWAMP RFCs for standards track advancement, the methods described here are useful to evaluate the measurement results because their validity would not be ascertained in typical interoperability testing.

The standards advancement process aims at producing confidence that the metric definitions and supporting material are clearly worded and unambiguous, or reveals ways in which the metric definitions can be revised to achieve clarity. The process also permits identification of options that were not implemented, so that they can be removed from the advancing specification. Thus, the product of this process is information about the metric specification RFC itself: determination of the specifications or definitions that are clear and unambiguous and those that are not (as opposed to an evaluation of the implementations which assist in the process).

This document defines a process to verify that implementations (or practically, measurement systems) have interpreted the metric specifications in equivalent ways, and produce equivalent results. Testing for statistical equivalence requires ensuring identical test setups (or awareness of differences) to the best possible extent. Thus, producing identical test conditions is a core goal of the memo. Another important aspect of this process is to test individual implementations against specific requirements in the metric specifications using customized tests for each requirement. These tests can distinguish equivalent interpretations of each specific requirement.

Conclusions on equivalence are reached by two measures.

First, implementations are compared against individual metric specifications to make sure that differences in implementation are minimised or at least known.

Second, a test setup is proposed ensuring identical networking conditions so that unknowns are minimized and comparisons are simplified. The resulting separate data sets may be seen as samples taken from the same underlying distribution. Using state of the art statistical methods, the equivalence of the results is verified. To illustrate application of the process and methods defined here, evaluation of the One-way Delay Metric [RFC2679] is provided in an Appendix. While test setups will vary with the metrics to be validated, the general methodology of determining equivalent results will not. Documents defining test setups to evaluate other metrics should be developed once the process proposed here has been agreed and approved.

The metric RFC advancement process begins with a request for protocol action accompanied by a memo that documents the supporting tests and results. The procedures of [RFC2026] are expanded in[RFC5657], including sample implementation and interoperability reports. Section 3 of [morton-advance-metrics-01] can serve as a template for a metric RFC report which accompanies the protocol action request to the Area

Director, including description of the test set-up, procedures, results
for each implementation and conclusions.
Changes from WG-03 to WG-04:

     *Revisions to Appendix B code and add reference to "R" in the
      Appendix and the text of section 3.6.

Changes from WG-02 to WG-03:

     *Changes stemming from experiments that implemented this plan, in
      general.

     *Adoption of the VLAN loopback figure in the main body of the memo
      (section 3.2).

Changes from WG-01 to WG-02:

     *Clarification of the number of test streams recommended in
      section 3.2.

     *Clarifications on testing details in sections 3.3 and 3.4.

     *Spelling corrections throughout.

Changes from WG -00 to WG -01 draft

     *Discussion on merits and requirements of a distributed lab test
      using only local load generators.

     *Proposal of metrics suitable for tests using the proposed
      measurement configuration.

     *Hint on delay caused by software based L2TPv3 implementation.

     *Added an appendix with a test configuration allowing remote tests
      comparing different implementations across the network.

     *Proposal for maximum error of "equivalence", based on performance
      comparison of identical implementations. This may be useful for
      both ADK and non-ADK comparisons.

Changes from prior ID -02 to WG -00 draft

     *Incorporation of aspects of reporting to support the protocol
      action request in the Introduction and section 3.5

     *Overhaul of section 3.2 regarding tunneling: Added generic
      tunneling requirements and L2TPv3 as an example tunneling
      mechanism fulfilling the tunneling requirements. Removed and
      adapted some of the prior references to other tunneling protocols

*Softened a requirement within section 3.4 (MUST to SHOULD on
 precision) and removed some comments of the authors.

*Updated contact information of one author and added a new author.

*Added example C++ code of an Anderson-Darling two sample test
 implementation.

Changes from ID -01 to ID -02 version

*Major editorial review, rewording and clarifications on all
 contents.

*Additional text on parallel testing using VLANs and GRE or
 Pseudowire tunnels.

*Additional examples and a glossary.

Changes from ID -00 to ID -01 version

*Addition of a comparison of individual metric implementations
 against the metric specification (trying to pick up problems and
 solutions for metric advancement [morton-advance-metrics]).

*More emphasis on the requirement to carefully design and document
 the measurement setup of the metric comparison.

*Proposal of testing conditions under identical WAN network
 conditions using IP in IP tunneling or Pseudo Wires and parallel
 measurement streams.

*Proposing the requirement to document the smallest resolution at
 which an ADK test was passed by 95%. As no minimum resolution is
 specified, IPPM metric compliance is not linked to a particular
 performance of an implementation.

*Reference to RFC 2330 and RFC 2679 for the 95% confidence
 interval as preferred criterion to decide on statistical
 equivalence

*Reducing the proposed statistical test to ADK with 95%
 confidence.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in RFC 2119 [RFC2119].

Basic idea

The implementation of a standard compliant metric is expected to meet
the requirements of the related metric specification. So before
comparing two metric implementations, each metric implementation is
individually compared against the metric specification.
Most metric specifications leave freedom to implementors on non-
fundamental aspects of an individual metric (or options). Comparing
different measurement results using a statistical test with the
assumption of identical test path and testing conditions requires
knowledge of all differences in the overall test setup. Metric
specification options chosen by implementors have to be documented. It
is REQUIRED to use identical implementation options wherever possible
for any test proposed here. Calibrations proposed by metric standards
should be performed to further identify (and possibly reduce) potential
sources of errors in the test setup.
The Framework for IP Performance Metrics [RFC2330] expects that a
"methodology for a metric should have the property that it is
repeatable: if the methodology is used multiple times under identical
conditions, it should result in consistent measurements." This means an
implementation is expected to repeatedly measure a metric with
consistent results (repeatability with the same result). Small
deviations in the test setup are expected to lead to small deviations
in results only. To characterise statistical equivalence in the case of
small deviations, RFC 2330 and [RFC2679] suggest to apply a 95%
confidence interval. Quoting RFC 2679, "95 percent was chosen because
... a particular confidence level should be specified so that the
results of independent implementations can be compared."
Two different implementations are expected to produce statistically
equivalent results if they both measure a metric under the same
networking conditions. Formulating in statistical terms: separate
metric implementations collect separate samples from the same
underlying statistical process (the same network conditions). The
statistical hypothesis to be tested is the expectation that both
samples do not expose statistically different properties. This requires
careful test design:

    *The measurement test setup must be self-consistent to the largest
     possible extent. To minimize the influence of the test and
     measurement setup on the result, network conditions and paths
     MUST be identical for the compared implementations to the largest
     possible degree. This includes both the stability and non-
     ambiguity of routes taken by the measurement packets. See RFC
     2330 for a discussion on self-consistency.

    *To minimize the influence of implementation options on the
     result, metric implementations SHOULD use identical options and
     parameters for the metric under evaluation.

*The error induced by the sample size must be small enough to
   minimize its influence on the test result. This may have to be
   respected, especially if two implementations measure with
   different average probing rates.

  *The implementation with the lowest probing frequency determines
   the smallest temporal interval for which samples can be compared.

  *Repeat comparisons with several independent metric samples to
   avoid random indications of compatibility (or the lack of it).

The metric specifications themselves are the primary focus of
evaluation, rather than the implementations of metrics. The
documentation produced by the advancement process should identify which
metric definitions and supporting material were found to be clearly
worded and unambiguous, OR, it should identify ways in which the metric
specification text should be revised to achieve clarity and unified
interpretation.
The process should also permit identification of options that were not
implemented, so that they can be removed from the advancing
specification (this is an aspect more typical of protocol advancement
along the standards track).
Note that this document does not propose to base interoperability
indications of performance metric implementations on comparisons of
individual singletons. Individual singletons may be impacted by many
statistical effects while they are measured. Comparing two singletons
of different implementations may result in failures with higher
probability than comparing samples.

[3.](#) Verification of conformance to a metric specification

This section specifies how to verify compliance of two or more IPPM
implementations against a metric specification. This document only
proposes a general methodology. Compliance criteria to a specific
metric implementation need to be defined for each individual metric
specification. The only exception is the statistical test comparing two
metric implementations which are simultaneously tested. This test is
applicable without metric specific decision criteria.
Several testing options exist to compare two or more implementations:

  *Use a single test lab to compare the implementations and emulate
   the Internet with an impairment generator.

  *Use a single test lab to compare the implementations and measure
   across the Internet.

  *Use remotely separated test labs to compare the implementations
   and emulate the Internet with two "identically" configured
   impairment generators.

*Use remotely separated test labs to compare the implementations
    and measure across the Internet.

   *Use remotely separated test labs to compare the implementations
    and measure across the Internet and include a single impairment
    generator to impact all measurement flows in non discriminatory
    way.

The first two approaches work, but cause higher expenses than the other
ones (due to travel and/or shipping+installation). For the third
option, ensuring two identically configured impairment generators
requires well defined test cases and possibly identical hard- and
software.
As documented in a test report *[morton-testplan-rfc2679]*, the last
option was required to prove compatibility of two delay metric
implementations. An impairment generator is probably required when
testing compatibility of most other metrics and it therefore
RECOMMENDED to include an impairment generator in metric test set ups.

3.1. Tests of an individual implementation against a metric specification

A metric implementation MUST support the requirements classified as
"MUST" and "REQUIRED" of the related metric specification to be
compliant to the latter.
Further, supported options of a metric implementation SHOULD be
documented in sufficient detail. The documentation of chosen options is
RECOMMENDED to minimise (and recognise) differences in the test setup
if two metric implementations are compared. Further, this documentation
is used to validate and improve the underlying metric specification
option, to remove options which saw no implementation or which are
badly specified from the metric specification to be promoted to a
standard. This documentation SHOULD be made for all implementation-
relevant specifications of a metric picked for a comparison that are
not explicitly marked as "MUST" or "REQUIRED" in the RFC text. This
applies for the following sections of all metric specifications:

   *Singleton Definition of the Metric.

   *Sample Definition of the Metric.

   *Statistics Definition of the Metric. As statistics are compared
    by the test specified here, this documentation is required even
    in the case, that the metric specification does not contain a
    Statistics Definition.

   *Timing and Synchronisation related specification (if relevant for
    the Metric).

*Any other technical part present or missing in the metric
      specification, which is relevant for the implementation of the
      Metric.

RFC2330 and RFC2679 emphasise precision as an aim of IPPM metric
implementations. A single IPPM conformant implementation MUST under
otherwise identical network conditions produce precise results for
repeated measurements of the same metric.
RFC 2330 prefers the "empirical distribution function" EDF to describe
collections of measurements. RFC 2330 determines, that "unless
otherwise stated, IPPM goodness-of-fit tests are done using 5%
significance." The goodness of fit test determines by which precision
two or more samples of a metric implementation belong to the same
underlying distribution (of measured network performance events). The
goodness of fit test suggested for the metric test is the Anderson-
Darling K sample test (ADK sample test, K stands for the number of
samples to be compared) [ADK]. Please note that RFC 2330 and RFC 2679
apply an Anderson Darling goodness of fit test too.
The results of a repeated test with a single implementation MUST pass
an ADK sample test with confidence level of 95%. The conditions for
which the ADK test has been passed with the specified confidence level
MUST be documented. To formulate this differently: The requirement is
to document the set of parameters with the smallest deviation, at which
the results of the tested metric implementation pass an ADK test with a
confidence level of 95%. The minimum resolution available in the
reported results from each implementation MUST be taken into account in
the ADK test.
The test conditions which MUST be documented for a passed metric test
include:

     *The metric resolution at which a test was passed (e.g. the
      resolution of timestamps)

     *The parameters modified by an impairment generator.

     *The impairment generator parameter settings.

3.2. Test setup resulting in identical live network testing conditions

Two major issues complicate tests for metric compliance across live
networks under identical testing conditions. One is the general point
that metric definition implementations cannot be conveniently examined
in field measurement scenarios. The other one is more broadly described
as "parallelism in devices and networks", including mechanisms like
those that achieve load balancing (see [RFC4928]).
This section proposes two measures to deal with both issues. Tunneling
mechanisms can be used to avoid parallel processing of different flows
in the network. Measuring by separate parallel probe flows results in
repeated collection of data. If both measures are combined, WAN network

conditions are identical for a number of independent measurement flows, no matter what the network conditions are in detail.

Any measurement setup MUST be made to avoid the probing traffic itself to impede the metric measurement. The created measurement load MUST NOT result in congestion at the access link connecting the measurement implementation to the WAN. The created measurement load MUST NOT overload the measurement implementation itself, e.g., by causing a high CPU load or by creating imprecisions due to internal transmit (receive respectively) probe packet collisions.

Tunneling multiple flows reaching a network element on a single physical port may allow to transmit all packets of the tunnel via the same path. Applying tunnels to avoid undesired influence of standard routing for measurement purposes is a concept known from literature, see e.g. GRE encapsulated multicast probing *[GU+Duffield]*. An existing IP in IP tunnel protocol can be applied to avoid Equal-Cost Multi-Path (ECMP) routing of different measurement streams if it meets the following criteria:

> *Inner IP packets from different measurement implementations are mapped into a single tunnel with single outer IP origin and destination address as well as origin and destination port numbers which are identical for all packets.

> *An easily accessible commodity tunneling protocol allows to carry out a metric test from more test sites.

> *A low operational overhead may enable a broader audience to set up a metric test with the desired properties.

> *The tunneling protocol should be reliable and stable in set up and operation to avoid disturbances or influence on the test results.

> *The tunneling protocol should not incur any extra cost for those interested in setting up a metric test.

An illustration of a test setup with two layer 2 tunnels and two flows between two linecards of one implementation is given in Figure 1.

```
   Implementation                     ,---.        +--------+
                        +~~~~~~~~~~~/      \~~~~~~| Remote |
    +------->-----F2->-|          /        \     |->---+  |
    | +--------+       | Tunnel 1(          )    |     |  |
    | | transmit|-F1->-|         (          )    |->+  |  |
    | | LC1      |      +~~~~~~~~~|          |~~~~|  |  |  |
    | | receive |-<--+          (          )    | F1  F2 |
    | +--------+    |            |Internet |    |  |  |  |
    *-------<-----+  F2          |          |   |  |  |  |
      +--------+ |  | +~~~~~~~~~|          |~~~~|  |  |  |
      | transmit|-*   *-|         |          |   |--+<-*  |
      | LC2      |      | Tunnel 2(          )   |  |     |
      | receive |-<-F1-|          \          /   |<-*     |
      +--------+      +~~~~~~~~~~~\      /~~~~~~| Router |
                              `-+-'          +--------+
```
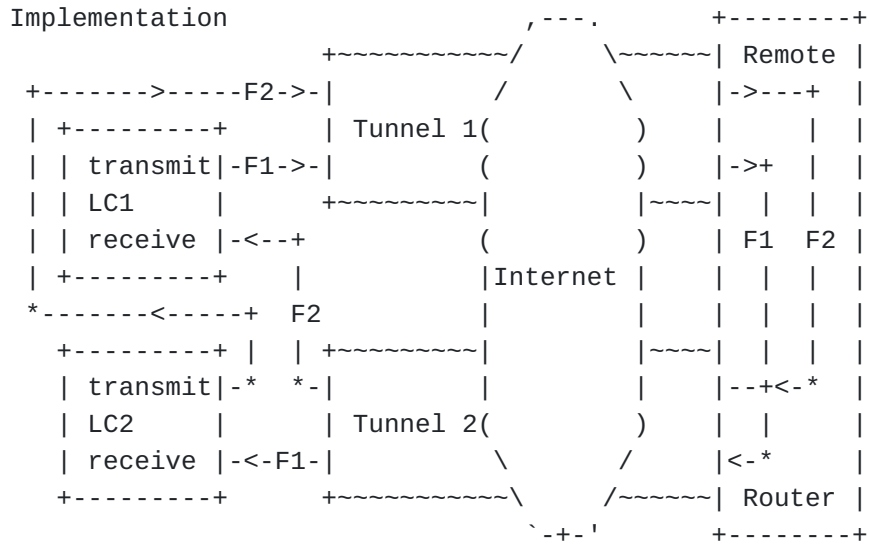
Illustration of a test setup with two layer 2 tunnels. For simplicity,
only two linecards of one implementation and two flows F between them
are shown.
[Figure 2](#) shows the network elements required to set up layer 2 tunnels
as shown by figure 1.


```
    Implementation

    +-----+                    ,---.
    | LC1 |                   /     \
    +-----+                  /       \             +------+
       |         +-------+ (           ) +-------+  |Remote|
    +--------+   |       | |           | |       |  |      |
    |Ethernet|   | Tunnel| |Internet |  | Tunnel|  |      |
    |Switch  |--| Head  |--|           |--| Head  |--|      |
    +--------+   | Router| |           | | Router|  |      |
       |         |       | (           ) |       |  |Router|
    +-----+     +-------+  \         /  +-------+  +------+
    | LC2 |                 \       /
    +-----+                  `-+-'
```

Illustration of a hardware setup to realise the test setup illustrated
by figure 1 with layer 2 tunnels or Pseudowires.
The test set up successfully used during a [delay metric test](#) *[morton-
testplan-rfc2679]* is given as an example in figure 3. Note that the
shown set up allows a metric test between two remote sites.

```
     +----+  +----+                                +----+  +----+
     |LC10|  |LC11|              ,---.              |LC20|  |LC21|
     +----+  +----+             /     \   +-------+ +----+  +----+
       | V10  | V11            /       \  | Tunnel|  | V20   | V21
       |      |               (         ) | Head  |  |       |
     +--------+  +------+ |            |  | Router|__+----------+
     |Ethernet|  |Tunnel| |Internet |  +---B---+  |Ethernet   |
     |Switch  |--|Head  |-|          |  |       |  |Switch     |
     +-+--+---+  |Router| |          |  +---+---+  +--+--+----+
       |__|      +--A---+ (          )--|Option.|     |__|
                           \        /   |Impair.|
     Bridge                 \      /    |Gener. |     Bridge
     V20 to V21              `-+-?       +-------+    V10 to V11
```
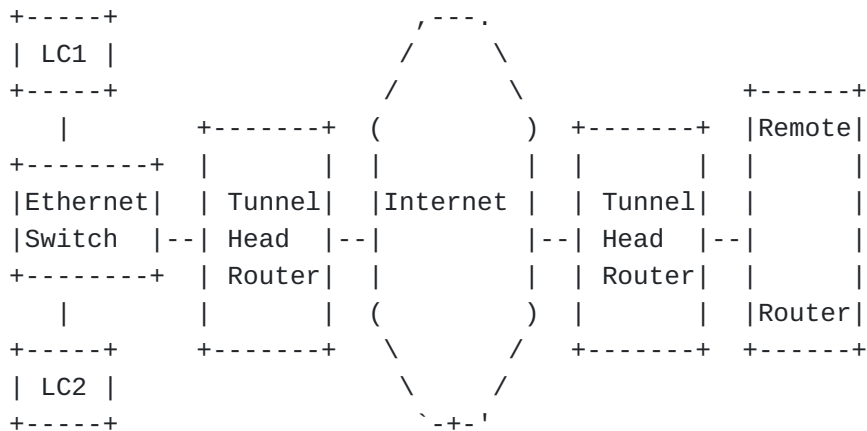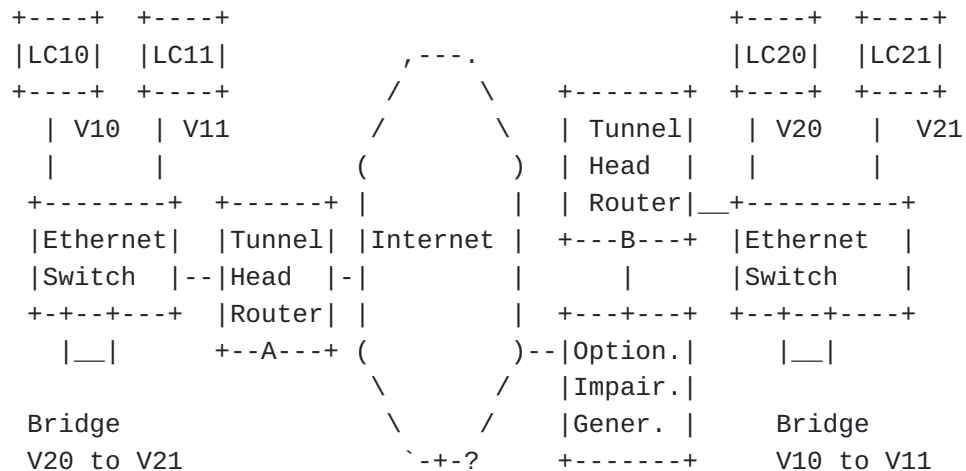
In figure 3, LC10 identify measurement clients /line cards. V10 and the
others denote VLANs. All VLANs are using the same tunnel from A to B
and in the reverse direction. The remote site VLANs are U-bridged at
the local site Ethernet switch. The measurement packets of site 1
travel tunnel A->B first, are U-bridged at site 2 and travel tunnel B-
>A second. Measurement packets of site 2 travel tunnel B->A first, are
U-bridged at site 1 and travel tunnel A->B second. So all measurement
packets pass the same tunnel segments, but in different segment order.
If tunneling is applied, two tunnels MUST carry all test traffic in
between the test site and the remote site. For example, if 802.1Q
Ethernet Virtual LANs (VLAN) are applied and the measurement streams
are carried in different VLANs, the IP tunnel or Pseudo Wires
respectively MUST be set up in physical port mode to avoid set up of
Pseudo Wires per VLAN (which may see different paths due to ECMP
routing), see RFC 4448. The remote router and the Ethernet switch shown
in figure 3 has to support 802.1Q in this set up.
The IP packet size of the metric implementation SHOULD be chosen small
enough to avoid fragmentation due to the added Ethernet and tunnel
headers. Otherwise, the impact of tunnel overhead on fragmentation and
interface MTU size MUST be understood and taken into account (see
[RFC4459]).
An Ethernet port mode IP tunnel carrying several 802.1Q VLANs each
containing measurement traffic of a single measurement system was
successfully applied when testing compatibility of two metric
implementations [morton-testplan-rfc2679].
The following headers may have to be accounted for when calculating
total packet length, if VLANs and Ethernet over L2TPv3 tunnels are
applied:

    *Ethernet 802.1Q: 22 Byte.

*L2TPv3 Header: 4-16 Byte for L2TPv3 data messages over IP; 16-28
        Byte for L2TPv3 data messages over UDP.

       *IPv4 Header (outer IP header): 20 Byte.

       *MPLS Labels may be added by a carrier. Each MPLS Label has a
        length of 4 Bytes. By the time of writing, between 1 and 4 Labels
        seems to be a fair guess of what's expectable.

The applicability of one or more of the following tunneling protocols
may be investigated by interested parties if Ethernet over L2TPv3 is
felt to be not suitable: IP in IP [RFC2003] or Generic Routing
Encapsulation (GRE) [RFC2784]. RFC 4928 [RFC4928] proposes measures how
to avoid ECMP treatment in MPLS networks.
L2TP is a commodity tunneling protocol [RFC2661]. By the time of
writing, L2TPv3 [RFC3931]is the latest version of L2TP. If L2TPv3 is
applied, software based implementations of this protocol are not
suitable for the test set up, as such implementations may cause
incalculable delay shifts.
Ethernet Pseudo Wires may also be set up on MPLS networks [RFC4448].
While there's no technical issue with this solution, MPLS interfaces
are mostly found in the network provider domain. Hence not all of the
above criteria to select a tunneling protocol are met.
Note that setting up a metric test environment isn't a plug and play
issue. Skilled networking engineers should be consulted and involved,
if a set up between remote sites is preferred.
Passing or failing an ADK test with 2 samples could be a random result
(note that [RFC2330] defines a sample as a set of singleton metric
values produced by a measurement stream, and we continue to use this
terminology here). The error margin of a statistical test is higher if
the number of samples it is based on is low (the number of samples
taken influences the so called "degree of freedom" of a statistical
test and a higher degree of freedom produces more reliable results). To
pass ADK with higher probability, the number of samples collected per
implementation under identical networking conditions SHOULD be greater
than 2. Hardware and load constraints may enforce an upper limit on the
number of simultaneous measurement streams. The ADK test allows one to
combine different samples (see section 9 [ADK]) and then to run a two
sample test between combined samples. At least 4 samples per
implementation captured under identical networking conditions is
RECOMMENDED when comparing different metric implementations by a
statistical test.
It is RECOMMENDED that tests be carried out by establishing N different
parallel measurement flows. Two or three linecards per implementation
serving to send or receive measurement flows should be sufficient to
create 4 or more parallel measurement flows. Other options are to
separate flows by DiffServ marks (without deploying any QoS in the
inner or outer tunnel) or using a single CBR flow and evaluating every
n-th singleton to belong to a specific measurement flow. Note that a

practical test indeed showed that ADK was passed with 4 samples even if a 2 sample test failed[morton-testplan-rfc2679].
Some additional guidelines to calculate and compare samples to perform a metric test are:

   *To compare different probes of a common underlying distribution
    in terms of metrics characterising a communication network
    requires to respect the temporal nature for which the assumption
    of common underlying distribution may hold. Any singletons or
    samples to be compared MUST be captured within the same time
    interval.

   *If statistical events like rates are used to characterise
    measured metrics of a time-interval, its RECOMMENDED to pick as a
    minimum 5 singletons of a relevant metric to ensure a minimum
    confidence into the reported value. The error margin of the
    determined rate depends on the number singletons (refer to
    statistical textbooks on Student's t-test). As an example, any
    packet loss measurement interval to be compared with the results
    of another implementation contains at least five lost packets to
    have some confidence that the observed loss rate wasn't caused by
    a small number of random packet drops.

   *The minimum number of singletons or samples to be compared by an
    Anderson-Darling test SHOULD be 100 per tested metric
    implementation. Note that the Anderson-Darling test detects small
    differences in distributions fairly well and will fail for high
    number of compared results (RFC2330 mentions an example with 8192
    measurements where an Anderson-Darling test always failed).

   *Generally, the Anderson-Darling test is sensitive to differences
    in the accuracy or bias associated with varying implementations
    or test conditions. These dissimilarities may result in differing
    averages of samples to be compared. An example may be different
    packet sizes, resulting in a constant delay difference between
    compared samples. Therefore samples to be compared by an Anderson
    Darling test MAY be calibrated by the difference of the average
    values of the samples. Any calibration of this kind MUST be
    documented in the test result.

3.3. Tests of two or more different implementations against a metric specification

RFC2330 expects "a methodology for a given metric [to] exhibit continuity if, for small variations in conditions, it results in small variations in the resulting measurements. Slightly more precisely, for every positive epsilon, there exists a positive delta, such that if two sets of conditions are within delta of each other, then the resulting measurements will be within epsilon of each other." A small variation

in conditions in the context of the metric test proposed here can be seen as different implementations measuring the same metric along the same path.

IPPM metric specifications however allow for implementor options to the largest possible degree. It cannot be expected that two implementors allow 100% identical options in their implementations. Testers SHOULD to the highest degree possible pick the same configurations for their systems when comparing their implementations by a metric test.

In some cases, a goodness of fit test may not be possible or show disappointing results. To clarify the difficulties arising from different implementation options, the individual options picked for every compared implementation SHOULD be documented in sufficient detail. Based on this documentation, the underlying metric specification should be improved before it is promoted to a standard. The same statistical test as applicable to quantify precision of a single metric implementation MUST be used to compare metric result equivalence for different implementations. To document compatibility, the smallest measurement resolution at which the compared implementations passed the ADK sample test MUST be documented.

For different implementations of the same metric, "variations in conditions" are reasonably expected. The ADK test comparing samples of the different implementations MAY result in a lower precision than the test for precision in the same-implementation comparison.

3.4. Clock synchronisation

Clock synchronization effects require special attention. Accuracy of one-way active delay measurements for any metrics implementation depends on clock synchronization between the source and destination of tests. Ideally, one-way active delay measurement (RFC 2679, [RFC2679]) test endpoints either have direct access to independent GPS or CDMA-based time sources or indirect access to nearby NTP primary (stratum 1) time sources, equipped with GPS receivers. Access to these time sources may not be available at all test locations associated with different Internet paths, for a variety of reasons out of scope of this document. When secondary (stratum 2 and above) time sources are used with NTP running across the same network, whose metrics are subject to comparative implementation tests, network impairments can affect clock synchronization, distort sample one-way values and their interval statistics. It is RECOMMENDED to discard sample one-way delay values for any implementation, when one of the following reliability conditions is met:

   *Delay is measured and is finite in one direction, but not the
    other.

   *Absolute value of the difference between the sum of one-way
    measurements in both directions and round-trip measurement is
    greater than X% of the latter value.

Examination of the second condition requires RTT measurement for
reference, e.g., based on TWAMP (RFC5357, [RFC 5357](#) *[RFC5357]*), in
conjunction with one-way delay measurement.
Specification of X% to strike a balance between identification of
unreliable one-way delay samples and misidentification of reliable
samples under a wide range of Internet path RTTs probably requires
further study.
An IPPM compliant metric implementation of an RFC that requires
synchronized clocks is expected to provide precise measurement results.
IF an implementation publishes a specification of its precision, such
as "a precision of 1 ms (+/- 500 us) with a confidence of 95%", then
the specification SHOULD be met over a useful measurement duration. For
example, if the metric is measured along an Internet path which is
stable and not congested, then the precision specification SHOULD be
met over durations of an hour or more.

3.5. Recommended Metric Verification Measurement Process

In order to meet their obligations under the IETF Standards Process the
IESG must be convinced that each metric specification advanced to Draft
Standard or Internet Standard status is clearly written, that there are
a sufficient number of verified equivalent implementations, and that
options that have been implemented are documented.
In the context of this document, metrics are designed to measure some
characteristic of a data network. An aim of any metric definition
should be that it should be specified in a way that can reliably
measure the specific characteristic in a repeatable way across multiple
independent implementations.
Each metric, statistic or option of those to be validated MUST be
compared against a reference measurement or another implementation by
as specified by this document.
Finally, the metric definitions, embodied in the text of the RFCs, are
the objects that require evaluation and possible revision in order to
advance to the next step on the standards track.
IF two (or more) implementations do not measure an equivalent metric as
specified by this document,
AND sources of measurement error do not adequately explain the lack of
agreement,
THEN the details of each implementation should be audited along with
the exact definition text, to determine if there is a lack of clarity
that has caused the implementations to vary in a way that affects the
correspondence of the results.
IF there was a lack of clarity or multiple legitimate interpretations
of the definition text,
THEN the text should be modified and the resulting memo proposed for
consensus and (possible) advancement along the standards track.
Finally, all the findings MUST be documented in a report that can
support advancement on the standards track, similar to those described
in [RFC5657](#). The list of measurement devices used in testing satisfies

the implementation requirement, while the test results provide
information on the quality of each specification in the metric RFC (the
surrogate for feature interoperability).
The complete process of advancing a metric specification to a standard
as defined by this document is illustrated in Figure 4.

```
   ,---.
  /     \
 ( Start )
  \     /     Implementations
   `-+-'        +-------+
    |          /|   1   `.
 +---+----+   / +-------+ `.-----------+     ,-------.
 |  RFC   | /              |Check for  |   ,' was RFC `. YES
 |        | /              |Equivalence....  clause x    ------+
 |        |/     +-------+ |under       |   `. clear?  ,'        |
 | Metric \.....|   2    ....relevant   |     `---+---'   +----+-----+
 | Metric |\    +-------+ |identical   |       No |        |Report    |
 | Metric | \             |network     |      +--+----+  |results + |
 |  ...   | \             |conditions  |      |Modify |  |Advance   |
 |        |  \ +-------+  |            |       |Spec    +--+RFC       |
 +--------+   \|   n    |.'+-----------+       +-------+  |request(?)|
              +-------+                                    +----------+
```
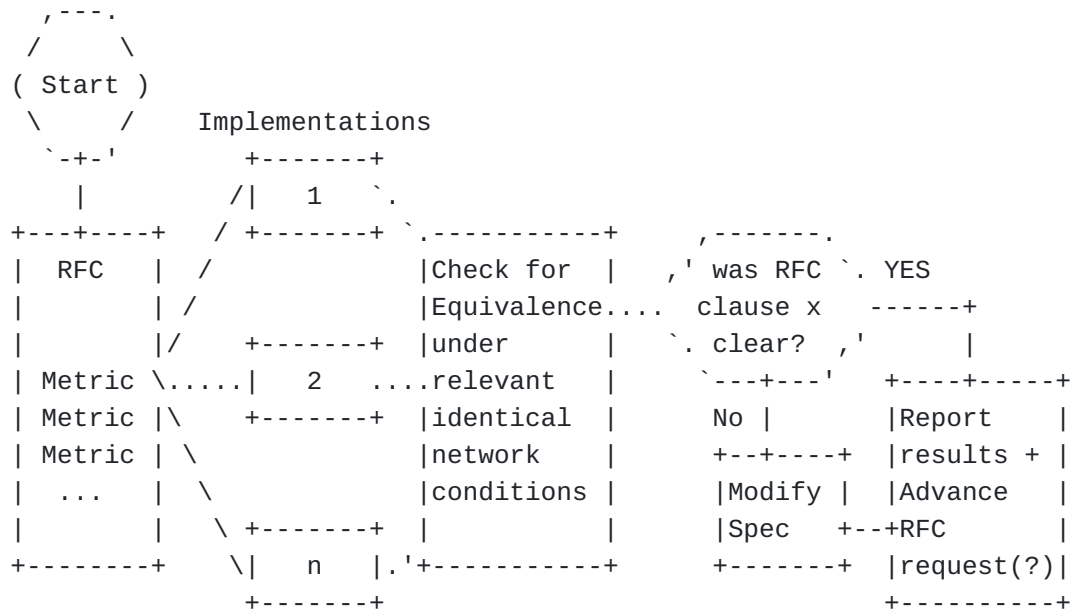
Illustration of the metric standardisation process
Any recommendation for the advancement of a metric specification MUST
be accompanied by an implementation report, as is the case with all
requests for the advancement of IETF specifications. The implementation
report needs to include the tests performed, the applied test setup,
the specific metrics in the RFC and reports of the tests performed with
two or more implementations. The test plan needs to specify the
precision reached for each measured metric and thus define the meaning
of "statistically equivalent" for the specific metrics being tested.
Ideally, the test plan would co-evolve with the development of the
metric, since that's when people have the most context in their
thinking regarding the different subtleties that can arise.
In particular, the implementation report MUST as a minimum document:

    *The metric compared and the RFC specifying it. This includes
     statements as required by the section "Tests of an individual
     implementation against a metric specification" of this document.

    *The measurement configuration and setup.

    *A complete specification of the measurement stream (mean rate,
     statistical distribution of packets, packet size or mean packet

size and their distribution), DSCP and any other measurement
stream properties which could result in deviating results.
Deviations in results can be caused also if chosen IP addresses
and ports of different implementations can result in different
layer 2 or layer 3 paths due to operation of Equal Cost Multi-
Path routing in an operational network.

*The duration of each measurement to be used for a metric
validation, the number of measurement points collected for each
metric during each measurement interval (i.e. the probe size) and
the level of confidence derived from this probe size for each
measurement interval.

*The result of the statistical tests performed for each metric
validation as required by the section "Tests of two or more
different implementations against a metric specification" of this
document.

*A parameterization of laboratory conditions and applied traffic
and network conditions allowing reproduction of these laboratory
conditions for readers of the implementation report.

*The documentation helping to improve metric specifications
defined by this section.

All of the tests for each set SHOULD be run in a test setup as
specified in the section "Test setup resulting in identical live
network testing conditions."
If a different test set up is chosen, it is RECOMMENDED to avoid
effects falsifying results of validation measurements caused by real
data networks (like parallelism in devices and networks). Data networks
may forward packets differently in the case of:

*Different packet sizes chosen for different metric
implementations. A proposed countermeasure is selecting the same
packet size when validating results of two samples or a sample
against an original distribution.

*Selection of differing IP addresses and ports used by different
metric implementations during metric validation tests. If ECMP is
applied on IP or MPLS level, different paths can result (note
that it may be impossible to detect an MPLS ECMP path from an IP
endpoint). A proposed counter measure is to connect the
measurement equipment to be compared by a NAT device, or
establishing a single tunnel to transport all measurement traffic
The aim is to have the same IP addresses and port for all
measurement packets or to avoid ECMP based local routing
diversion by using a layer 2 tunnel.

*Different IP options.

*Different DSCP.

   *If the N measurements are captured using sequential measurements
    instead of simultaneous ones, then the following factors come
    into play: Time varying paths and load conditions.

**3.6. Proposal to determine an "equivalence" threshold for each metric
evaluated**

This section describes a proposal for maximum error of "equivalence",
based on performance comparison of identical implementations. This
comparison may be useful for both ADK and non-ADK comparisons.
Each metric tested by two or more implementations (cross-implementation
testing).
Each metric is also tested twice simultaneously by the *same*
implementation, using different Src/Dst Address pairs and other
differences such that the connectivity differences of the cross-
implementation tests are also experienced and measured by the same
implementation.
Comparative results for the same implementation represent a bound on
cross-implementation equivalence. This should be particularly useful
when the metric does *not* produces a continuous distribution of
singleton values, such as with a loss metric, or a duplication metric.
Appendix A indicates how the ADK will work for 0ne-way delay, and
should be likewise applicable to distributions of delay variation.
Appendix B discusses two possible ways to perform the ADK analysis, the
R statistical language [Rtool] with ADK package [Radk] and C++ code.
Proposal: the implementation with the largest difference in homogeneous
comparison results is the lower bound on the equivalence threshold,
noting that there may be other systematic errors to account for when
comparing between implementations.
Thus, when evaluating equivalence in cross-implementation results:
Maximum_Error = Same_Implementation_Error + Systematic_Error
and only the systematic error need be decided beforehand.
In the case of ADK comparison, the largest same-implementation
resolution of distribution equivalence can be used as a limit on cross-
implementation resolutions (at the same confidence level).

4. Acknowledgements

Gerhard Hasslinger commented a first version of this document,
suggested statistical tests and the evaluation of time series
information. Matthias Wieser's thesis on a metric test resulted in new
input for this draft. Henk Uijterwaal and Lars Eggert have encouraged
and helped to orgainize this work. Mike Hamilton, Scott Bradner, David
Mcdysan and Emile Stephan commented on this draft. Carol Davids
reviewed the 01 version of the ID before it was promoted to WG draft.

5. Contributors

Scott Bradner, Vern Paxson and Allison Mankin drafted bradner-metrictest *[bradner-metrictest]*, and major parts of it are included in this document.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

This memo does not raise any specific security issues.

8. References

8.1. Normative References

[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
[RFC2026]	Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.
[RFC2679]	Almes, G., Kalidindi, S. and M. Zekauskas, "A One-way Delay Metric for IPPM", RFC 2679, September 1999.
[RFC2330]	Paxson, V., Almes, G., Mahdavi, J. and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, May 1998.
[RFC2003]	Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.
[RFC2784]	Farinacci, D., Li, T., Hanks, S., Meyer, D. and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.
[RFC2661]	Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G. and B. Palter, "Layer Two Tunneling Protocol "L2TP"", RFC 2661, August 1999.
[RFC3931]	Lau, J., Townsley, M. and I. Goyret, "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC 3931, March 2005.
[RFC4448]	Martini, L., Rosen, E., El-Aawar, N. and G. Heron, "Encapsulation Methods for Transport of Ethernet over MPLS Networks", RFC 4448, April 2006.
[RFC4459]	Savola, P., "MTU and Fragmentation Issues with In-the-Network Tunneling", RFC 4459, April 2006.
[RFC4928]	Swallow, G., Bryant, S. and L. Andersson, "Avoiding Equal Cost Multipath Treatment in MPLS Networks", BCP 128, RFC 4928, June 2007.
[RFC2680]	Almes, G., Kalidindi, S. and M. Zekauskas, "A One-way Packet Loss Metric for IPPM", RFC 2680, September 1999.
[RFC2681]	

	Almes, G., Kalidindi, S. and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, September 1999.
[RFC4656]	Shalunov, S., Teitelbaum, B., Karp, A., Boote, J. and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, September 2006.
[RFC4719]	Aggarwal, R., Townsley, M. and M. Dos Santos, "Transport of Ethernet Frames over Layer 2 Tunneling Protocol Version 3 (L2TPv3)", RFC 4719, November 2006.
[RFC5657]	Dusseault, L. and R. Sparks, "Guidance on Interoperation and Implementation Reports for Advancement to Draft Standard", BCP 9, RFC 5657, September 2009.

8.2. Informative References

[RFC5357]	Hedayat, K., Krzanowski, R., Morton, A., Yum, K. and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, October 2008.
[bradner-metrictest]	Bradner, S., Mankin, A. and V. Paxson, "Advancement of metrics specifications on the IETF Standards Track", draft -bradner-metricstest-03, (work in progress), July 2007.
[morton-advance-metrics]	Morton, A., "Problems and Possible Solutions for Advancing Metrics on the Standards Track", draft -morton-ippm-advance-metrics-00, (work in progress), July 2009.
[morton-advance-metrics-01]	Morton, A., "Lab Test Results for Advancing Metrics on the Standards Track", draft -morton-ippm-advance-metrics-01, (work in progress), June 2010.
[morton-testplan-rfc2679]	Ciavattone, L., Geib, R., Morton, A. and M. Wieser, "Test Plan and Results for Advancing RFC 2679 on the Standards Track", draft -morton-ippm-testplan-rfc2679-01, (work in progress), June 2011.
[GU+Duffield]	Gu, Y., Duffield, N., Breslau, L. and S. Sen, "GRE Encapsulated Multicast Probing: A Scalable Technique for Measuring One-Way Loss", SIGMETRICS'07 San Diego, California, USA, June 2007.
[ADK]	Scholz, F.W. and M.A. Stephens, "K-sample Anderson-Darling Tests of fit, for continuous and discrete cases", University of Washington, Technical Report No. 81, May 1986.
[Rtool]	R Development Core Team, , "R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna,

	Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/", , 2011.
[Radk]	Scholz, F., "adk: Anderson-Darling K-Sample Test and Combinations of Such Tests. R package version 1.0.", , 2008.
[Rule of thumb]	Hardy, M., "Confidence interval", March 2010.

Appendix A. An example on a One-way Delay metric validation

The text of this appendix is not binding. It is an example how parts of a One-way Delay metric test could look like. http://xml.resource.org/public/rfc/bibxml/

Appendix A.1. Compliance to Metric specification requirements

One-way Delay, Loss threshold, RFC 2679
This test determines if implementations use the same configured maximum waiting time delay from one measurement to another under different delay conditions, and correctly declare packets arriving in excess of the waiting time threshold as lost. See Section 3.5 of RFC2679, 3rd bullet point and also Section 3.8.2 of RFC2679.

(1) Configure a path with 1 sec one-way constant delay.

(2) Measure one-way delay with 2 or more implementations, using identical waiting time thresholds for loss set at 2 seconds.

(3) Configure the path with 3 sec one-way delay.

(4) Repeat measurements.

(5) Observe that the increase measured in step 4 caused all packets to be declared lost, and that all packets that arrive successfully in step 2 are assigned a valid one-way delay.

One-way Delay, First-bit to Last bit, RFC 2679
This test determines if implementations register the same relative increase in delay from one measurement to another under different delay conditions. This test tends to cancel the sources of error which may be present in an implementation. See Section 3.7.2 of RFC2679, and Section 10.2 of RFC2330.

(1) Configure a path with X ms one-way constant delay, and ideally including a low-speed link.

(2) Measure one-way delay with 2 or more implementations, using identical options and equal size small packets (e.g., 100 octet IP payload).

(3) Maintain the same path with X ms one-way delay.

(4)
     Measure one-way delay with 2 or more implementations, using
  identical options and equal size large packets (e.g., 1500 octet IP
  payload).

(5) Observe that the increase measured in steps 2 and 4 is equivalent
  to the increase in ms expected due to the larger serialization time
  for each implementation. Most of the measurement errors in each
  system should cancel, if they are stationary.

One-way Delay, RFC 2679
This test determines if implementations register the same relative
increase in delay from one measurement to another under different delay
conditions. This test tends to cancel the sources of error which may be
present in an implementation. This test is intended to evaluate
measurments in sections 3 and 4 of RFC2679.

(1) Configure a path with X ms one-way constant delay.

(2) Measure one-way delay with 2 or more implementations, using
  identical options.

(3) Configure the path with X+Y ms one-way delay.

(4) Repeat measurements.

(5) Observe that the increase measured in steps 2 and 4 is ~Y ms for
  each implementation. Most of the measurement errors in each system
  should cancel, if they are stationary.

Error Calibration, RFC 2679
This is a simple check to determine if an implementation reports the
error calibration as required in Section 4.8 of RFC2679. Note that the
context (Type-P) must also be reported.

[Appendix A.2.](#) **Examples related to statistical tests for One-way Delay**

A one way delay measurement may pass an ADK test with a timestamp
resultion of 1 ms. The same test may fail, if timestamps with a
resolution of 100 microseconds are eavluated. The implementation then
is then conforming to the metric specification up to a timestamp
resolution of 1 ms.
Let's assume another one way delay measurement comparison between
implementation 1, probing with a frequency of 2 probes per second and
implementation 2 probing at a rate of 2 probes every 3 minutes. To
ensure reasonable confidence in results, sample metrics are calculated
from at least 5 singletons per compared time interval. This means,
sample delay values are calculated for each system for identical 6
minute intervals for the whole test duration. Per 6 minute interval,
the sample metric is calculated from 720 singletons for implementation

1 and from 6 singletons for implementation 2. Note, that if outliers
are not filtered, moving averages are an option for an evaluation too.
The minimum move of an averaging interval is three minutes in this
example.
The data in table 1 may result from measuring One-Way Delay with
implementation 1 (see column Implemnt_1) and implementation 2 (see
column implemnt_2). Each data point in the table represents a (rounded)
average of the sampled delay values per interval. The resolution of the
clock is one micro-second. The difference in the delay values may
result eg. from different probe packet sizes.

Implemnt_1	Implemnt_2	Implemnt_2 - Delta_Averages
5000	6549	4997
5008	6555	5003
5012	6564	5012
5015	6565	5013
5019	6568	5016
5022	6570	5018
5024	6573	5021
5026	6575	5023
5027	6577	5025
5029	6580	5028
5030	6585	5033
5032	6586	5034
5034	6587	5035
5036	6588	5036
5038	6589	5037
5039	6591	5039
5041	6592	5040
5043	6599	5047
5046	6606	5054
5054	6612	5060

Average values of sample metrics captured during identical time
intervals are compared. This excludes random differences caused by
differing probing intervals or differing temporal distance of
singletons resulting from their Poisson distributed sending times.
In the example, 20 values have been picked (note that at least 100
values are recommended for a single run of a real test). Data must be
ordered by ascending rank. The data of Implemnt_1 and Implemnt_2 as
shown in the first two columns of table 1 clearly fails an ADK test
with 95% confidence.

The results of Implemnt_2 are now reduced by difference of the averages of column 2 (rounded to 6581 us) and column 1 (rounded to 5029 us), which is 1552 us. The result may be found in column 3 of table 1. Comparing column 1 and column 3 of the table by an ADK test shows, that the data contained in these columns passes an ADK tests with 95% confidence.

>>> Comment: Extensive averaging was used in this example, because of the vastly different sampling frequencies. As a result, the distributions compared do not exactly align with a metric in [RFC2679], but illustrate the ADK process adequately.

Appendix B. Anderson-Darling K-sample Reference and 2 sample C++ code

There are many statistical tools available, and this Appendix describes two that are familiar to the authors.

The "R tool" is a language and command-line environment for statistical computing and plotting [Rtool]. With the optional "adk" package installed [Radk], it can perform individual and combined sample ADK computations. The user must consult the package documentation and the original paper [ADK] to interpret the results, but this is as it should be.

The C++ code below will perform a 2-sample AD comparison when compiled and presented with two column vectors in a file (using white space as separation). This version contains modifications to use the vectors and run as a stand-alone module by Wes Eddy, Sept 2011. The status of the comparison can be checked on the command line with "$ echo $?" or the last line can be replaced with a printf statement for adk_result instead.

```cpp
/* Routines for computing the Anderson-Darling 2 sample
 * test statistic.
 *
 * Implemented based on the description in
 * "Anderson-Darling K Sample Test" Heckert, Alan and
 * Filliben, James, editors, Dataplot Reference Manual,
 * Chapter 15 Auxiliary, NIST, 2004.
 * Official Reference by 2010
 * Heckert, N. A. (2001). Dataplot website at the
 * National Institute of Standards and Technology:
 * http://www.itl.nist.gov/div898/software/dataplot.html/
 * June 2001.
 */

#include <iostream>
#include <fstream>
#include <vector>
#include <sstream>

using namespace std;



int main() {
    vector<double> vec1, vec2;
    double adk_result;
    static int k, val_st_z_samp1, val_st_z_samp2,
               val_eq_z_samp1, val_eq_z_samp2,
               j, n_total, n_sample1, n_sample2, L,
               max_number_samples, line, maxnumber_z;
    static int column_1, column_2;
    static double adk, n_value, z, sum_adk_samp1,
                  sum_adk_samp2, z_aux;
    static double H_j, F1j, hj, F2j, denom_1_aux, denom_2_aux;
    static bool next_z_sample2, equal_z_both_samples;
    static int stop_loop1, stop_loop2, stop_loop3,old_eq_line2,
               old_eq_line1;

    static double adk_criterium = 1.993;

    /* vec1 and vec2 to be initialised with sample 1 and
     * sample 2 values in ascending order */
    while (!cin.eof()) {
        double f1, f2;
        cin >> f1;
        cin >> f2;
        vec1.push_back(f1);
        vec2.push_back(f2);
    }
```

```
k = 2;
n_sample1 = vec1.size() - 1;
n_sample2 = vec2.size() - 1;

// -1 because vec[0] is a dummy value
n_total = n_sample1 + n_sample2;

/* value equal to the line with a value = zj in sample 1.
 * Here j=1, so the line is 1.
 */
val_eq_z_samp1 = 1;

/* value equal to the line with a value = zj in sample 2.
 * Here j=1, so the line is 1.
 */
val_eq_z_samp2 = 1;

/* value equal to the last line with a value < zj
 * in sample 1. Here j=1, so the line is 0.
 */
val_st_z_samp1 = 0;

/* value equal to the last line with a value < zj
 * in sample 1. Here j=1, so the line is 0.
 */
val_st_z_samp2 = 0;

sum_adk_samp1 = 0;
sum_adk_samp2 = 0;
j = 1;

// as mentioned above, j=1
equal_z_both_samples = false;

next_z_sample2 = false;

//assuming the next z to be of sample 1
stop_loop1 = n_sample1 + 1;

// + 1 because vec[0] is a dummy, see n_sample1 declaration
stop_loop2 = n_sample2 + 1;
stop_loop3 = n_total + 1;

/* The required z values are calculated until all values
 * of both samples have been taken into account. See the
 * lines above for the stoploop values. Construct required
 * to avoid a mathematical operation in the While condition
 */
while (((stop_loop1 > val_eq_z_samp1)
        || (stop_loop2 > val_eq_z_samp2)) && stop_loop3 > j)
```

```
{
  if(val_eq_z_samp1 < n_sample1+1)
   {
 /* here, a preliminary zj value is set.
  * See below how to calculate the actual zj.
  */
        z = vec1[val_eq_z_samp1];

 /* this while sequence calculates the number of values
  * equal to z.
  */
        while ((val_eq_z_samp1+1 < n_sample1)
                && z == vec1[val_eq_z_samp1+1] )
                {
                val_eq_z_samp1++;
                }
        }
        else
        {
        val_eq_z_samp1 = 0;
        val_st_z_samp1 = n_sample1;

// this should be val_eq_z_samp1 - 1 = n_sample1
        }

if(val_eq_z_samp2 < n_sample2+1)
        {
        z_aux = vec2[val_eq_z_samp2];;

/* this while sequence calculates the number of values
 * equal to z_aux
 */

        while ((val_eq_z_samp2+1 < n_sample2)
                && z_aux == vec2[val_eq_z_samp2+1] )
                {
                val_eq_z_samp2++;
                }

/* the smaller of the two actual data values is picked
 * as the next zj.
 */

   if(z > z_aux)
                {
                z = z_aux;
                next_z_sample2 = true;
                }
        else
```

```
                     {
                     if (z == z_aux)
                     {
                     equal_z_both_samples = true;
                     }

/* This is the case, if the last value of column1 is
 * smaller than the remaining values of column2.
 */
                        if (val_eq_z_samp1 == 0)
                        {
                        z = z_aux;
                        next_z_sample2 = true;
                        }
                  }
            }
        else
            {
          val_eq_z_samp2 = 0;
          val_st_z_samp2 = n_sample2;

// this should be val_eq_z_samp2 - 1 = n_sample2

            }

 /* in the following, sum j = 1 to L is calculated for
  * sample 1 and sample 2.
  */
        if (equal_z_both_samples)
            {

          /* hj is the number of values in the combined sample
           * equal to zj
           */
              hj = val_eq_z_samp1 - val_st_z_samp1
             + val_eq_z_samp2 - val_st_z_samp2;

          /* H_j is the number of values in the combined sample
           * smaller than zj plus one half the the number of
           * values in the combined sample equal to zj
           * (that's hj/2).
           */
             H_j = val_st_z_samp1 + val_st_z_samp2
                    + hj / 2;

          /* F1j is the number of values in the 1st sample
           * which are less than zj plus one half the number
           * of values in this sample which are equal to zj.
           */
```

```
        F1j = val_st_z_samp1 + (double)
            (val_eq_z_samp1 - val_st_z_samp1) / 2;

    /* F2j is the number of values in the 1st sample
     * which are less than zj     plus one half the number
     * of values in this sample which are equal to zj.
     */
        F2j = val_st_z_samp2 + (double)
            (val_eq_z_samp2 - val_st_z_samp2) / 2;

    /* set the line of values equal to zj to the
     * actual line of the last value picked for zj.
     */
        val_st_z_samp1 = val_eq_z_samp1;

    /* Set the line of values equal to zj to the actual
     * line of the last value picked for zjof each
     * sample. This is required as data smaller than zj
     * is accounted differently than values equal to zj.
     */
        val_st_z_samp2 = val_eq_z_samp2;

    /* next the lines of the next values z, ie. zj+1
     * are addressed.
     */
     val_eq_z_samp1++;

    /* next the lines of the next values z, ie.
     * zj+1 are addressed
     */
        val_eq_z_samp2++;
        }
else
        {

    /* the smaller z value was contained in sample 2,
     * hence this value is the zj to base the following
     * calculations on.
     */
                if (next_z_sample2)
                {

    /* hj is the number of values in the combined
     * sample equal to zj, in this case these are
     * within sample 2 only.
     */
                hj = val_eq_z_samp2 - val_st_z_samp2;

    /* H_j is the number of values in the combined sample
     * smaller than zj plus one half the the number of
```

```
           * values in the combined sample equal to zj
           * (that's hj/2).
           */

                        H_j = val_st_z_samp1 + val_st_z_samp2
                   + hj / 2;

        /* F1j is the number of values in the 1st sample which
         * are less than zj plus one half the number of values in
         * this sample which are equal to zj.
         * As val_eq_z_samp2 < val_eq_z_samp1, these are the
         * val_st_z_samp1 only.
         */
                   F1j = val_st_z_samp1;

        /* F2j is the number of values in the 1st sample which
         * are less than zj plus one half the number of values in
         * this sample which are equal to zj. The latter are from
         * sample 2 only in this case.
         */

            F2j = val_st_z_samp2 + (double)
                  (val_eq_z_samp2 - val_st_z_samp2) / 2;

        /* Set the line of values equal to zj to the actual line
         * of the last value picked for zj of sample 2 only in
         * this case.
         */
                        val_st_z_samp2 = val_eq_z_samp2;

        /* next the line of the next value z, ie. zj+1 is
         * addressed. Here, only sample 2 must be addressed.
         */

           val_eq_z_samp2++;
                           if (val_eq_z_samp1 == 0)
                           {
                           val_eq_z_samp1 = stop_loop1;
                           }
                  }

/* the smaller z value was contained in sample 2,
 * hence this value is the zj to base the following
 * calculations on.
 */

           else
           {

/* hj is the number of values in the combined
```

```
         * sample equal to zj, in this case these are
         * within sample 1 only.
         */
                    hj = val_eq_z_samp1 - val_st_z_samp1;

    /* H_j is the number of values in the combined
     * sample smaller than zj plus one half the the number
     * of values in the combined sample equal to zj
     * (that's hj/2).
     */

        H_j = val_st_z_samp1 + val_st_z_samp2
              + hj / 2;

    /* F1j is the number of values in the 1st sample which
     * are less than zj plus, in this case these are within
     * sample 1 only one half the number of values in this
     * sample which are equal to zj. The latter are from
     * sample 1 only in this case.
     */

        F1j = val_st_z_samp1 + (double)
              (val_eq_z_samp1 - val_st_z_samp1) / 2;

    /* F2j is the number of values in the 1st sample which
     * are less than zj plus one half the number of values
     * in this sample which are equal to zj. As
     * val_eq_z_samp1 < val_eq_z_samp2, these are the
     * val_st_z_samp2 only.
     */

                    F2j = val_st_z_samp2;

    /* Set the line of values equal to zj to the actual line
     * of the last value picked for zj of sample 1 only in
     * this case
     */

        val_st_z_samp1 = val_eq_z_samp1;

    /* next the line of the next value z, ie. zj+1 is
     * addressed. Here, only sample 1 must be addressed.
     */
                    val_eq_z_samp1++;

                    if (val_eq_z_samp2 == 0)
                            {
                            val_eq_z_samp2 = stop_loop2;
                            }
                    }
```

```
            }

        denom_1_aux = n_total * F1j - n_sample1 * H_j;
        denom_2_aux = n_total * F2j - n_sample2 * H_j;

        sum_adk_samp1 = sum_adk_samp1 + hj
                * (denom_1_aux * denom_1_aux) /
                                  (H_j * (n_total - H_j)
                - n_total * hj / 4);
        sum_adk_samp2 = sum_adk_samp2 + hj
          * (denom_2_aux * denom_2_aux) /
                          (H_j * (n_total - H_j)
        - n_total * hj / 4);

        next_z_sample2 = false;
        equal_z_both_samples = false;

  /* index to count the z. It is only required to prevent
   * the while slope to execute endless
   */
        j++;
        }

  // calculating the adk value is the final step.
  adk_result = (double) (n_total - 1) / (n_total
        * n_total * (k - 1))
        * (sum_adk_samp1 / n_sample1
        + sum_adk_samp2 / n_sample2);


  /* if(adk_result <= adk_criterium)
   * adk_2_sample test is passed
   */
  return adk_result <= adk_criterium;
}
```

ADK	Anderson-Darling K-Sample test, a test used to check whether two samples have the same statistical distribution.
ECMP	Equal Cost Multipath, a load balancing mechanism evaluating MPLS labels stacks, IP addresses and ports.
EDF	The "Empirical Distribution Function" of a set of scalar measurements is a function $F(x)$ which for any x gives the fractional proportion of the total measurements that were smaller than or equal as x.
Metric	

	A measured quantity related to the performance and reliability of the Internet, expressed by a value. This could be a singleton (single value), a sample of single values or a statistic based on a sample of singletons.
OWAMP	One-way Active Measurement Protocol, a protocol for communication between IPPM measurement systems specified by IPPM.
OWD	One-Way Delay, a performance metric specified by IPPM.
Sample metric	A sample metric is derived from a given singleton metric by evaluating a number of distinct instances together.
Singleton metric	A singleton metric is, in a sense, one atomic measurement of this metric.
Statistical metric	A 'statistical' metric is derived from a given sample metric by computing some statistic of the values defined by the singleton metric on the sample.
TWAMP	Two-way Active Measurement Protocol, a protocol for communication between IPPM measurement systems specified by IPPM.

Authors' Addresses

Ruediger Geib editor Geib Deutsche Telekom Heinrich Hertz Str. 3-7 Darmstadt, 64295 Germany Phone: +49 6151 58 12747 EMail: Ruediger.Geib@telekom.de

Al Morton Morton AT&T Labs 200 Laurel Avenue South Middletown, NJ 07748 USA Phone: +1 732 420 1571 EMail: acmorton@att.com URI: http://home.comcast.net/~acmacm/

Reza Fardid Fardid Cariden Technologies 888 Villa Street, Suite 500 Mountain View, CA 94041 USA EMail: rfardid@cariden.com

Alexander Steinmitz Steinmitz Deutsche Telekom Memmelsdorfer Str. 209b Bamberg, 96052 Germany EMail: Alexander.Steinmitz@telekom.de