Network Working Group
Internet-Draft
Updates: 2330 (if approved)
Intended status: Informational
Expires: January 1, 2019

A. Morton
AT&T Labs
J. Fabini
TU Wien
N. Elkins
Inside Products, Inc.
M. Ackermann
Blue Cross Blue Shield of Michigan
V. Hegde
Consultant
June 30, 2018

# IPv6, IPv4 and Coexistence Updates for IPPM's Active Metric Framework
draft-ietf-ippm-2330-ipv6-06

## Abstract

This memo updates the IP Performance Metrics (IPPM) Framework RFC 2330 with new considerations for measurement methodology and testing. It updates the definition of standard-formed packets in RFC 2330 to include IPv6 packets, deprecates the definition of minimal IP packet, and augments distinguishing aspects of packets, referred to as Type-P for test packets in RFC 2330. This memo identifies that IPv4-IPv6 co-existence can challenge measurements within the scope of the IPPM Framework. Example use cases include, but are not limited to IPv4-IPv6 translation, NAT, or protocol encapsulation. IPv6 header compression and use of IPv6 over Low-Power Wireless Area Networks (6LoWPAN) are considered and excluded from the standard-formed packet evaluation.

## 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.

Copyright Notice



Table of Contents

## 1. Introduction

The IETF IP Performance Metrics (IPPM) working group first created a framework for metric development in [RFC2330]. This framework has stood the test of time and enabled development of many fundamental metrics. It has been updated in the area of metric composition [RFC5835], and in several areas related to active stream measurement of modern networks with reactive properties [RFC7312].

The IPPM framework [RFC2330] recognized (in section 13) that many aspects of IP packets can influence its processing during transfer across the network.

In Section 15 of [RFC2330], the notion of a "standard-formed" packet is defined. However, the definition was never updated to include IPv6, as the original authors originally desired to do.

In particular, IPv6 Extension Headers and protocols which use IPv6 header compression are growing in use. This memo seeks to provide the needed updates.

## 2. Scope

The purpose of this memo is to expand the coverage of IPPM metrics to include IPv6, and to highlight additional aspects of test packets and make them part of the IPPM performance metric framework.

The scope is to update key sections of [RFC2330], adding considerations that will aid the development of new measurement methodologies intended for today's IP networks. Specifically, this memo expands the Type-P examples in section 13 of [RFC2330] and expands the definition (in section 15 of [RFC2330]) of a standard-formed packet to include IPv6 header aspects and other features.

Other topics in [RFC2330] which might be updated or augmented are deferred to future work. This includes the topics of passive and various forms of hybrid active/passive measurements.

## 3. Packets of Type-P

A fundamental property of many Internet metrics is that the measured value of the metric depends on characteristics of the IP packet(s) used to make the measurement. Potential influencing factors include IP header fields and their values, but also higher-layer protocol headers and their values. Consider an IP-connectivity metric: one obtains different results depending on whether one is interested in connectivity for packets destined for well-known TCP ports or unreserved UDP ports, or those with invalid IPv4 checksums, or those with TTL or Hop Limit of 16, for example. In some circumstances these distinctions will result in special treatment of packets in intermediate nodes and end systems (for example, if Diffserv [RFC2474], ECN [RFC3168], Router Alert [RFC6398], Hop-by-hop extensions [RFC7045], or Flow Labels [RFC6437] are used, or in the presence of firewalls or RSVP reservations).

Because of this distinction, we introduce the generic notion of a "packet of Type-P", where in some contexts P will be explicitly defined (i.e., exactly what type of packet we mean), partially defined (e.g., "with a payload of B octets"), or left generic. Thus we may talk about generic IP-Type-P-connectivity or more specific IP-port-HTTP-connectivity. Some metrics and methodologies may be fruitfully defined using generic Type-P definitions which are then made specific when performing actual measurements.

Whenever a metric's value depends on the type of the packets involved in the metric, the metric's name will include either a specific type or a phrase such as "Type-P". Thus we will not define an "IP-connectivity" metric but instead an "IP-Type-P-connectivity" metric and/or perhaps an "IP-port-HTTP-connectivity" metric. This naming convention serves as an important reminder that one must be conscious of the exact type of traffic being measured.

If the information constituting Type-P at the Source is found to have changed at the Destination (or at a measurement point between the Source and Destination, as in [RFC5644]), then the modified values MUST be noted and reported with the results. Some modifications occur according to the conditions encountered in transit (such as congestion notification) or due to the requirements of segments of the Source to Destination path. For example, the packet length will change if IP headers are converted to the alternate version/address family, or if optional Extension Headers are added or removed. Even header fields like TTL/Hop Limit that typically change in transit may be relevant to specific tests. For example Neighbor Discovery Protocol (NDP) [RFC4861] packets are transmitted with Hop Limit value set to 255, and the validity test specifies that the Hop Limit MUST have a value of 255 at the receiver, too. So, while other tests may intentionally exclude the TTL/Hop Limit value from their Type-P definition, for this particular test the correct Hop Limit value is of high relevance and MUST be part of the Type-P definition.

Local policies in intermediate nodes based on examination of IPv6 Extension Headers may affect measurement repeatability. If intermediate nodes follow the recommendations of [RFC7045], repeatability may be improved to some degree.

A closely related note: it would be very useful to know if a given Internet component (like host, link, or path) treats equally a class C of different types of packets. If so, then any one of those types of packets can be used for subsequent measurement of the component. This suggests we devise a metric or suite of metrics that attempt to determine class C (a designation which has no relationship to address assignments, of course).

Load balancing over parallel paths is one particular example where such a class C would be more complex to determine in IPPM measurements. Load balancers and routers often use flow identifiers, computed as hashes of (specific parts of) the packet header, for deciding among the available parallel paths a packet will traverse. Packets with identical hashes are assigned to the same flow and forwarded to the same resource in the load balancer's (or router's) pool. The presence of a load balancer on the measurement path, as well as the specific headers and fields that are used for the forwarding decision, are not known when measuring the path as a black-box. Potential assessment scenarios include the measurement of one of the parallel paths, and the measurement of all available parallel paths that the load balancer can use. Knowledge of a load balancer's flow definition (alternatively: its class C specific treatment in terms of header fields in scope of hash operations) is therefore a prerequisite for repeatable measurements. A path may have more than one stage of load balancing, adding to class C definition complexity.

## 4. Standard-Formed Packets

Unless otherwise stated, all metric definitions that concern IP packets include an implicit assumption that the packet is *standard-formed*. A packet is standard-formed if it meets all of the following REQUIRED criteria:

+ It includes a valid IP header: see below for version-specific criteria.

+ It is not an IP fragment.

+ The Source and Destination addresses correspond to the intended Source and Destination, including Multicast Destination addresses.

+ If a transport header is present, it contains a valid checksum and other valid fields.

For an IPv4 ([RFC0791] and updates) packet to be standard-formed, the following additional criteria are REQUIRED:

- The version field is 4

- The Internet Header Length (IHL) value is >= 5; the checksum is correct.

- Its total length as given in the IPv4 header corresponds to the size of the IPv4 header plus the size of the payload.

- Either the packet possesses sufficient TTL to travel from the Source to the Destination if the TTL is decremented by one at each hop, or it possesses the maximum TTL of 255.

- It does not contain IP options unless explicitly noted.

For an IPv6 ([RFC8200] and updates) packet to be standard-formed, the following criteria are REQUIRED:

- The version field is 6.

- Its total length corresponds to the size of the IPv6 header (40 octets) plus the length of the payload as given in the IPv6 header.

- The payload length value for this packet (including Extension Headers) conforms to the IPv6 specifications.

- Either the packet possesses sufficient Hop Limit to travel from the Source to the Destination if the Hop Limit is decremented by one at each hop, or it possesses the maximum Hop Limit of 255.

- Either the packet does not contain IP Extension Headers, or it contains the correct number and type of headers as specified in the packet, and the headers appear in the standard-conforming order (Next Header).

- All parameters used in the header and Extension Headers are found in the IANA Registry of Internet Protocol Version 6 (IPv6) Parameters, specified in [IANA-6P].

Two mechanisms require some discussion in the context of standard-formed packets, namely IPv6 over Low-Power Wireless Area Networks (6LowPAN, [RFC4944]) and Robust Header Compression (ROHC, [RFC3095]). IPv6 over Low-Power Wireless Area Networks (6LowPAN), as defined in [RFC4944] and updated by [RFC6282] with header compression and [RFC6775] with neighbor discovery optimizations, proposes solutions for using IPv6 in resource-constrained environments. An adaptation layer enables the transfer of IPv6 packets over networks having a MTU smaller than the minimum IPv6 MTU. Fragmentation and re-assembly of

IPv6 packets, as well as the resulting state that would be stored in intermediate nodes, poses substantial challenges to measurements. Likewise, ROHC operates statefully in compressing headers on subpaths, storing state in intermediate hosts. The modification of measurement packets' Type-P by ROHC and 6LowPAN, as well as requirements with respect to the concept of standard-formed packets for these two protocols requires substantial work. Because of these reasons we consider ROHC and 6LowPAN packets to be out of the scope for the standard-formed packet evaluation.

The topic of IPv6 Extension Headers brings current controversies into focus as noted by [RFC6564] and [RFC7045]. However, measurement use cases in the context of the IPPM framework like in-situ OAM [I-D.ietf-ippm-ioam-data] in enterprise environments can benefit from inspection, modification, addition or deletion of IPv6 extension headers in hosts along the measurement path.

[RFC8250] endorses the use of IPv6 Destination Option for measurement purposes, consistent with other approved IETF specifications.

The following additional considerations apply when IPv6 Extension Headers are present:

- o Extension Header inspection: Some intermediate nodes may inspect Extension Headers or the entire IPv6 packet while in transit. In exceptional cases, they may drop the packet or route via a sub-optimal path, and measurements may be unreliable or unrepeatable. The packet (if it arrives) may be standard-formed, with a corresponding Type-P.

- o Extension Header modification: In Hop-by-Hop headers, some TLV encoded options may be permitted to change at intermediate nodes while in transit. The resulting packet may be standard-formed, with a corresponding Type-P.

- o Extension Header insertion or deletion: Although such behavior is not endorsed by current standards, it is possible that Extension Headers could be added to, or removed from the header chain. The resulting packet may be standard-formed, with a corresponding Type-P. This point simply encourages measurement system designers to be prepared for the unexpected, and to notify users when such events occur. There are issues with Extension Header insertion and deletion of course, such as exceeding the path MTU due to insertion, etc.

- o A change in packet length (from the corresponding packet observed at the Source) or header modification is a significant factor in

Internet measurement, and REQUIRES a new Type-P to be reported with the test results.

It is further REQUIRED that if a packet is described as having a "length of B octets", then 0 <= B <= 65535; and if B is the payload length in octets, then B <= (65535-IP header size in octets, including any Extension Headers). The jumbograms defined in [RFC2675] are not covered by the above length analysis, but if the IPv6 Jumbogram Payload Hop-by-Hop Option Header is present, then a packet with corresponding length MUST be considered standard-formed. In practice, the path MTU will restrict the length of standard-formed packets that can successfully traverse the path. Path MTU Discovery for IP version 6 (PMTUD, [RFC8201]) or Packetization Layer Path MTU Discovery (PLPMTUD, [RFC4821]) is recommended to prevent fragmentation.

So, for example, one might imagine defining an IP connectivity metric as "IP-type-P-connectivity for standard-formed packets with the IP Diffserv field set to 0", or, more succinctly, "IP-type-P-connectivity with the IP Diffserv Field set to 0", since standard-formed is already implied by convention. Changing the contents of a field, such as the Diffserv Code Point, ECN bits, or Flow Label may have a profound affect on packet handling during transit, but does not affect a packet's status as standard-formed. Likewise, the addition, modification, or deletion of extension headers may change the handling of packets in transit hosts.

[RFC2330] defines the "minimal IP packet from A to B" as a particular type of standard-formed packet often useful to consider. When defining IP metrics no packet smaller or simpler than this can be transmitted over a correctly operating IP network. However, the concept of the minimal IP packet has not been employed (since typical active measurement systems employ a transport layer and a payload) and its practical use is limited. Therefore, this memo deprecates the concept of the "minimal IP packet from A to B".

## 5. NAT, IPv4-IPv6 Transition and Compression Techniques

This memo adds the key considerations for utilizing IPv6 in two critical conventions of the IPPM Framework, namely packets of Type-P and standard-formed packets. The need for co-existence of IPv4 and IPv6 has originated transitioning standards like the Framework for IPv4/IPv6 Translation in [RFC6144] or IP/ICMP Translation Algorithms in [RFC7915] and [RFC7757].

The definition and execution of measurements within the context of the IPPM Framework is challenged whenever such translation mechanisms are present along the measurement path. In particular use cases like

IPv4-IPv6 translation, NAT, protocol encapsulation, or IPv6 header compression may result in modification of the measurement packet's Type-P along the path. All these changes MUST be reported. Example consequences include, but are not limited to:

- o Modification or addition of headers or header field values in intermediate nodes. IPv4-IPv6 transitioning or IPv6 header compression mechanisms may result in changes of the measurement packets' Type-P, too. Consequently, hosts along the measurement path may treat packets differently because of the Type-P modification. Measurements at observation points along the path may also need extra context to uniquely identify a packet.

- o Network Address Translators (NAT) on the path can have unpredictable impact on latency measurement (in terms of the amount of additional time added), and possibly other types of measurements. It is not usually possible to control this impact (as testers may not have any control of the underlying network or middleboxes). There is a possibility that stateful NAT will lead to unstable performance for a flow with specific Type-P, since state needs to be created for the first packet of a flow, and state may be lost later if the NAT runs out of resources. However, this scenario does not invalidate the Type-P for testing - for example the purpose of a test might be exactly to quantify the NAT's impact on delay variation. The presence of NAT may mean that the measured performance of Type-P will change between the source and the destination. This can cause an issue when attempting to correlate measurements conducted on segments of the path that include or exclude the NAT. Thus, it is a factor to be aware of when conducting measurements.

- o Variable delay due to internal state. One side effect of changes due to IPv4-IPv6 transitioning mechanisms is the variable delay that intermediate nodes spend for header modifications. Similar to NAT the allocation of internal state and establishment of context within intermediate nodes may cause variable delays, depending on the measurement stream pattern and position of a packet within the stream. For example the first packet in a stream will typically trigger allocation of internal state in an intermediate IPv4-IPv6 transition host. Subsequent packets can benefit from lower processing delay due to the existing internal state. However, large inter-packet delays in the measurement stream may result in the intermediate host deleting the associated state and needing to re-establish it on arrival of another stream packet. It is worth noting that this variable delay due to internal state allocation in intermediate nodes can be an explicit use case for measurements.

- o Variable delay due to packet length. IPv4-IPv6 transitioning or header compression mechanisms modify the length of measurement packets. The modification of the packet size may or may not change the way how the measurement path treats the packets.

## 6. Security Considerations

The security considerations that apply to any active measurement of live paths are relevant here as well. See [RFC4656] and [RFC5357].

When considering privacy of those involved in measurement or those whose traffic is measured, the sensitive information available to potential observers is greatly reduced when using active techniques which are within this scope of work. Passive observations of user traffic for measurement purposes raise many privacy issues. We refer the reader to the privacy considerations described in the Large Scale Measurement of Broadband Performance (LMAP) Framework [RFC7594], which covers active and passive techniques.

## 7. IANA Considerations

This memo makes no requests of IANA.

## 8. Acknowledgements

The authors thank Brian Carpenter for identifying the lack of IPv6 coverage in IPPM's Framework, and for listing additional distinguishing factors for packets of Type-P. Both Brian and Fred Baker discussed many of the interesting aspects of IPv6 with the co-authors, leading to a more solid first draft: thank you both. Thanks to Bill Jouris for an editorial pass through the pre-00 text. As we completed our journey, Nevil Brownlee, Mike Heard, Spencer Dawkins, Warren Kumari, and Suresh Krishnan all contributed useful suggestions.

## 9. References

### 9.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[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>.

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, DOI 10.17487/RFC2330, May 1998, <https://www.rfc-editor.org/info/rfc2330>.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, DOI 10.17487/RFC2675, August 1999, <https://www.rfc-editor.org/info/rfc2675>.

[RFC3095] Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L-E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T., and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, DOI 10.17487/RFC3095, July 2001, <https://www.rfc-editor.org/info/rfc3095>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

[RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, DOI 10.17487/RFC4656, September 2006, <https://www.rfc-editor.org/info/rfc4656>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <https://www.rfc-editor.org/info/rfc4821>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <https://www.rfc-editor.org/info/rfc4944>.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <https://www.rfc-editor.org/info/rfc5357>.

[RFC5644] Stephan, E., Liang, L., and A. Morton, "IP Performance Metrics (IPPM): Spatial and Multicast", RFC 5644, DOI 10.17487/RFC5644, October 2009, <https://www.rfc-editor.org/info/rfc5644>.

[RFC5835] Morton, A., Ed. and S. Van den Berghe, Ed., "Framework for Metric Composition", RFC 5835, DOI 10.17487/RFC5835, April 2010, <https://www.rfc-editor.org/info/rfc5835>.

[RFC6144] Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", RFC 6144, DOI 10.17487/RFC6144, April 2011, <https://www.rfc-editor.org/info/rfc6144>.

[RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <https://www.rfc-editor.org/info/rfc6282>.

[RFC6398] Le Faucheur, F., Ed., "IP Router Alert Considerations and Usage", BCP 168, RFC 6398, DOI 10.17487/RFC6398, October 2011, <https://www.rfc-editor.org/info/rfc6398>.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <https://www.rfc-editor.org/info/rfc6437>.

[RFC6564] Krishnan, S., Woodyatt, J., Kline, E., Hoagland, J., and M. Bhatia, "A Uniform Format for IPv6 Extension Headers", RFC 6564, DOI 10.17487/RFC6564, April 2012, <https://www.rfc-editor.org/info/rfc6564>.

[RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <https://www.rfc-editor.org/info/rfc6775>.

[RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <https://www.rfc-editor.org/info/rfc7045>.

[RFC7312] Fabini, J. and A. Morton, "Advanced Stream and Sampling Framework for IP Performance Metrics (IPPM)", RFC 7312, DOI 10.17487/RFC7312, August 2014, <https://www.rfc-editor.org/info/rfc7312>.

[RFC7757] Anderson, T. and A. Leiva Popper, "Explicit Address Mappings for Stateless IP/ICMP Translation", RFC 7757, DOI 10.17487/RFC7757, February 2016, <https://www.rfc-editor.org/info/rfc7757>.

[RFC7915] Bao, C., Li, X., Baker, F., Anderson, T., and F. Gont, "IP/ICMP Translation Algorithm", RFC 7915, DOI 10.17487/RFC7915, June 2016, <https://www.rfc-editor.org/info/rfc7915>.

[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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <https://www.rfc-editor.org/info/rfc8201>.

[RFC8250] Elkins, N., Hamilton, R., and M. Ackermann, "IPv6 Performance and Diagnostic Metrics (PDM) Destination Option", RFC 8250, DOI 10.17487/RFC8250, September 2017, <https://www.rfc-editor.org/info/rfc8250>.

## 9.2. Informative References

[I-D.ietf-ippm-ioam-data] Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., Chang, R., daniel.bernier@bell.ca, d., and J. Lemon, "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data-03 (work in progress), June 2018.

[IANA-6P] IANA, "IANA Internet Protocol Version 6 (IPv6) Parameters", Internet Assigned Numbers Authority https://www.iana.org/assignments/ipv6-parameters, January 2018.

[RFC7594] Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A Framework for Large-Scale Measurement of Broadband Performance (LMAP)", RFC 7594, DOI 10.17487/RFC7594, September 2015, <https://www.rfc-editor.org/info/rfc7594>.

Authors' Addresses

Al Morton
AT&T Labs
200 Laurel Avenue South
Middletown, NJ 07748
USA

Phone: +1 732 420 1571
Fax: +1 732 368 1192
Email: acmorton@att.com
URI: http://home.comcast.net/˜acmacm/

Joachim Fabini
TU Wien
Gusshausstrasse 25/E389
Vienna 1040
Austria

Phone: +43 1 58801 38813
Fax: +43 1 58801 38898
Email: Joachim.Fabini@tuwien.ac.at
URI: http://www.tc.tuwien.ac.at/about-us/staff/joachim-fabini/

Nalini Elkins
Inside Products, Inc.
Carmel Valley, CA 93924
USA

Email: nalini.elkins@insidethestack.com

Michael S. Ackermann
Blue Cross Blue Shield of Michigan

Email: mackermann@bcbsm.com

Vinayak Hegde
Consultant
Brahma Sun City, Wadgaon-Sheri
Pune, Maharashtra 411014
INDIA

Phone: +91 9449834401
Email: vinayakh@gmail.com
URI: http://www.vinayakhegde.com

Network Working Group G. Fioccola, Ed.
Internet-Draft A. Capello
Intended status: Experimental M. Cociglio
Expires: June 10, 2018 L. Castaldelli
Telecom Italia
M. Chen
L. Zheng
Huawei Technologies
G. Mirsky
ZTE
T. Mizrahi
Marvell
December 7, 2017

# Alternate Marking method for passive and hybrid performance monitoring
draft-ietf-ippm-alt-mark-14

## Abstract

This document describes a method to perform packet loss, delay and jitter measurements on live traffic. This method is based on Alternate Marking (Coloring) technique. A report is provided in order to explain an example and show the method applicability. This technology can be applied in various situations as detailed in this document and could be considered passive or hybrid depending on the application.

## 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.

## Status of This Memo

Copyright Notice



Table of Contents

# 1. Introduction

Nowadays, most Service Providers' networks carry traffic with contents that are highly sensitive to packet loss [RFC7680], delay [RFC7679], and jitter [RFC3393].

In view of this scenario, Service Providers need methodologies and tools to monitor and measure network performances with an adequate accuracy, in order to constantly control the quality of experience perceived by their customers. On the other hand, performance monitoring provides useful information for improving network management (e.g. isolation of network problems, troubleshooting, etc.).

A lot of work related to OAM, that includes also performance monitoring techniques, has been done by Standards Developing Organizations(SDOs): [RFC7276] provides a good overview of existing OAM mechanisms defined in IETF, ITU-T and IEEE. Considering IETF, a lot of work has been done on fault detection and connectivity verification, while a minor effort has been dedicated so far to performance monitoring. The IPPM WG has defined standard metrics to measure network performance; however, the methods developed in this WG mainly refer to focus on active measurement techniques. More recently, the MPLS WG has defined mechanisms for measuring packet loss, one-way and two-way delay, and delay variation in MPLS networks[RFC6374], but their applicability to passive measurements has some limitations, especially for pure connection-less networks.

The lack of adequate tools to measure packet loss with the desired accuracy drove an effort to design a new method for the performance monitoring of live traffic, easy to implement and deploy. The effort led to the method described in this document: basically, it is a passive performance monitoring technique, potentially applicable to any kind of packet based traffic, including Ethernet, IP, and MPLS, both unicast and multicast. The method addresses primarily packet loss measurement, but it can be easily extended to one-way delay and delay variation measurements as well.

The method has been explicitly designed for passive measurements but it can also be used with active probes. Passive measurements are usually more easily understood by customers and provide a much better accuracy, especially for packet loss measurements.

RFC 7799 [RFC7799] defines passive and hybrid methods of measurement. In particular, Passive Methods of Measurement are based solely on observations of an undisturbed and unmodified packet stream of interest; Hybrid Methods are Methods of Measurement that use a combination of Active Methods and Passive Methods.

Taking into consideration these definitions, Alternate Marking Method could be considered Hybrid or Passive depending on the case. In case the marking method is obtained by changing existing field values of the packets (e.g. DSCP field), the technique is Hybrid. In case the marking field is dedicated, reserved and is included in the protocol specification Alternate Marking technique can be considered as Passive (e.g. RFC6374 Synonymous Flow Label or OAM Marking Bits in BIER Header).

The advantages of the method described in this document are:

o easy implementation: it can be implemented or by using features already available on major routing platforms as described in Section 5.1 or by applying an optimized implementation of the method for both legacy and newest technologies;

o low computational effort: the additional load on processing is negligible;

o accurate packet loss measurement: single packet loss granularity is achieved with a passive measurement;

o potential applicability to any kind of packet/frame -based traffic: Ethernet, IP, MPLS, etc., both unicast and multicast;

o robustness: the method can tolerate out of order packets and it's not based on "special" packets whose loss could have a negative impact;

o flexibility: all the timestamp formats are allowed, because they are managed out-of-band. The format (the Network Time Protocol (NTP) RFC 5905 [RFC5905] or the IEEE 1588 Precision Time Protocol (PTP) [IEEE-1588]) depends on the precision you want;

o no interoperability issues: the features required to experiment and test the method (as described in Section 5.1) are available on all current routing platforms. Both a centarlized or distributed solution can be used to harvest data from the routers.

The method doesn't raise any specific need for protocol extension, but it could be further improved by means of some extension to existing protocols. Specifically, the use of DiffServ bits for coloring the packets could not be a viable solution in some cases: a standard method to color the packets for this specific application could be beneficial.

## 2.  Overview of the method

In order to perform packet loss measurements on a production traffic flow, different approaches exist.  The most intuitive one consists in numbering the packets, so that each router that receives the flow can immediately detect a packet missing.  This approach, though very simple in theory, is not simple to achieve: it requires the insertion of a sequence number into each packet and the devices must be able to extract the number and check it in real time.  Such a task can be difficult to implement on live traffic: if UDP is used as the transport protocol, the sequence number is not available; on the other hand, if a higher layer sequence number (e.g. in the RTP header) is used, extracting that information from each packet and process it in real time could overload the device.

An alternate approach is to count the number of packets sent on one end, the number of packets received on the other end, and to compare the two values.  This operation is much simpler to implement, but requires that the devices performing the measurement are in sync: in order to compare two counters it is required that they refer exactly to the same set of packets.  Since a flow is continuous and cannot be stopped when a counter has to be read, it can be difficult to determine exactly when to read the counter.  A possible solution to overcome this problem is to virtually split the flow in consecutive blocks by inserting periodically a delimiter so that each counter refers exactly to the same block of packets.  The delimiter could be for example a special packet inserted artificially into the flow. However, delimiting the flow using specific packets has some limitations.  First, it requires generating additional packets within the flow and requires the equipment to be able to process those packets.  In addition, the method is vulnerable to out of order reception of delimiting packets and, to a lesser extent, to their loss.

The method proposed in this document follows the second approach, but it doesn’t use additional packets to virtually split the flow in blocks.  Instead, it "marks" the packets so that the packets belonging to the same block will have the same color, whilst consecutive blocks will have different colors.  Each change of color represents a sort of auto-synchronization signal that guarantees the consistency of measurements taken by different devices along the path (see also [I-D.cociglio-mboned-multicast-pm] and [I-D.tempia-opsawg-p3m], where this technique was introduced).

Figure 1 represents a very simple network and shows how the method can be used to measure packet loss on different network segments: by enabling the measurement on several interfaces along the path, it is possible to perform link monitoring, node monitoring or end-to-end monitoring. The method is flexible enough to measure packet loss on any segment of the network and can be used to isolate the faulty element.

```
                          Traffic flow
      =========================================================>
        +------+       +------+       +------+       +------+
    ---<>  R1  <>-----<>  R2  <>-----<>  R3  <>-----<>  R4  <>---
        +------+       +------+       +------+       +------+
        .              .      .              .       .      .
        .              .      .              .       .      .
        .              <------>              <------->      .
        .          Node Packet Loss      Link Packet Loss   .
        .                                                   .
        <--------------------------------------------------->
                         End-to-End Packet loss
```

Figure 1: Available measurements

## 3. Detailed description of the method

This section describes in detail how the method operate. A special emphasis is given to the measurement of packet loss, that represents the core application of the method, but applicability to delay and jitter measurements is also considered.

### 3.1. Packet loss measurement

The basic idea is to virtually split traffic flows into consecutive blocks: each block represents a measurable entity unambiguously recognizable by all network devices along the path. By counting the number of packets in each block and comparing the values measured by different network devices along the path, it is possible to measure packet loss occurred in any single block between any two points.

As discussed in the previous section, a simple way to create the blocks is to "color" the traffic (two colors are sufficient) so that packets belonging to different consecutive blocks will have different colors. Whenever the color changes, the previous block terminates and the new one begins. Hence, all the packets belonging to the same block will have the same color and packets of different consecutive blocks will have different colors. The number of packets in each block depends on the criterion used to create the blocks:

- if the color is switched after a fixed number of packets, then each block will contain the same number of packets (except for any losses);

- if the color is switched according to a fixed timer, then the number of packets may be different in each block depending on the packet rate.

The following figure shows how a flow looks like when it is split in traffic blocks with colored packets.

A: packet with A coloring
B: packet with B coloring

```
         |           |           |           |           |
         |           |    Traffic flow       |           |
 ------------------------------------------------------------------->
  BBBBBBB AAAAAAAAAAA BBBBBBBBBBB AAAAAAAAAAA BBBBBBBBBBB AAAAAAA
 ------------------------------------------------------------------->
    ...  |  Block 5  |  Block 4  |  Block 3  |  Block 2  |  Block 1
         |           |           |           |           |
```

Figure 2: Traffic coloring

Figure 3 shows how the method can be used to measure link packet loss between two adjacent nodes.

Referring to the figure, let's assume we want to monitor the packet loss on the link between two routers: router R1 and router R2. According to the method, the traffic is colored alternatively with two different colors, A and B.  Whenever the color changes, the transition generates a sort of square-wave signal, as depicted in the following figure.

```
Color A   ----------+           +-----------+           +----------
                    |           |           |           |
Color B             +-----------+           +-----------+
           Block n        ...      Block 3     Block 2     Block 1
         <---------> <---------> <---------> <---------> <--------->

                              Traffic flow
         ===========================================================>
Color ...AAAAAAAAAAA BBBBBBBBBBB AAAAAAAAAAA BBBBBBBBBBB AAAAAAA...
         ===========================================================>
```

Figure 3: Computation of link packet loss

Traffic coloring could be done by R1 itself or it is already done before. R1 needs two counters, C(A)R1 and C(B)R1, on its egress interface: C(A)R1 counts the packets with color A and C(B)R1 counts those with color B. As long as traffic is colored A, only counter C(A)R1 will be incremented, while C(B)R1 is not incremented; vice versa, when the traffic is colored as B, only C(B)R1 is incremented. C(A)R1 and C(B)R1 can be used as reference values to determine the packet loss from R1 to any other measurement point down the path. Router R2, similarly, will need two counters on its ingress interface, C(A)R2 and C(B)R2, to count the packets received on that interface and colored with color A and B respectively. When an A block ends, it is possible to compare C(A)R1 and C(A)R2 and calculate the packet loss within the block; similarly, when the successive B block terminates, it is possible to compare C(B)R1 with C(B)R2, and so on for every successive block.

Likewise, by using two counters on R2 egress interface it is possible to count the packets sent out of R2 interface and use them as reference values to calculate the packet loss from R2 to any measurement point down R2.

Using a fixed timer for color switching offers a better control over the method: the (time) length of the blocks can be chosen large enough to simplify the collection and the comparison of measures taken by different network devices. It's preferable to read the value of the counters not immediately after the color switch: some packets could arrive out of order and increment the counter associated to the previous block (color), so it is worth waiting for some time. A safe choice is to wait L/2 time units (where L is the duration for each block) after the color switch, to read the still counter of the previous color, so the possibility to read a running counter instead of a still one is minimized. The drawback is that the longer the duration of the block, the less frequent the measurement can be taken.

The following table shows how the counters can be used to calculate the packet loss between R1 and R2. The first column lists the sequence of traffic blocks while the other columns contain the counters of A-colored packets and B-colored packets for R1 and R2. In this example, we assume that the values of the counters are reset to zero whenever a block ends and its associated counter has been read: with this assumption, the table shows only relative values, that is the exact number of packets of each color within each block. If the values of the counters were not reset, the table would contain cumulative values, but the relative values could be determined simply by difference from the value of the previous block of the same color.

The color is switched on the basis of a fixed timer (not shown in the table), so the number of packets in each block is different.

| Block | C(A)R1 | C(B)R1 | C(A)R2 | C(B)R2 | Loss |
|---|---|---|---|---|---|
| 1 | 375 | 0 | 375 | 0 | 0 |
| 2 | 0 | 388 | 0 | 388 | 0 |
| 3 | 382 | 0 | 381 | 0 | 1 |
| 4 | 0 | 377 | 0 | 374 | 3 |
| ... | ... | ... | ... | ... | ... |
| 2n | 0 | 387 | 0 | 387 | 0 |
| 2n+1 | 379 | 0 | 377 | 0 | 2 |

Table 1: Evaluation of counters for packet loss measurements

During an A block (blocks 1, 3 and 2n+1), all the packets are A-colored, therefore the C(A) counters are incremented to the number seen on the interface, while C(B) counters are zero. Vice versa, during a B block (blocks 2, 4 and 2n), all the packets are B-colored: C(A) counters are zero, while C(B) counters are incremented.

When a block ends (because of color switching) the relative counters stop incrementing and it is possible to read them, compare the values measured on router R1 and R2 and calculate the packet loss within that block.

For example, looking at the table above, during the first block (A-colored), C(A)R1 and C(A)R2 have the same value (375), which corresponds to the exact number of packets of the first block (no loss). Also during the second block (B-colored) R1 and R2 counters have the same value (388), which corresponds to the number of packets of the second block (no loss). During blocks three and four, R1 and R2 counters are different, meaning that some packets have been lost: in the example, one single packet (382-381) was lost during block three and three packets (377-374) were lost during block four.

The method applied to R1 and R2 can be extended to any other router and applied to more complex networks, as far as the measurement is enabled on the path followed by the traffic flow(s) being observed.

It’s worth mentioning two different strategies that can be used when implementing the method:

- o flow-based: the flow-based strategy is used when only a limited number of traffic flows need to be monitored. According to this strategy, only a subset of the flows is colored. Counters for packet loss measurements can be instantiated for each single flow, or for the set as a whole, depending on the desired granularity. A relevant problem with this approach is the necessity to know in advance the path followed by flows that are subject to measurement. Path rerouting and traffic load-balancing increase the issue complexity, especially for unicast traffic. The problem is easier to solve for multicast traffic where load balancing is seldom used and static joins are frequently used to force traffic forwarding and replication.

- o link-based: measurements are performed on all the traffic on a link by link basis. The link could be a physical link or a logical link. Counters could be instantiated for the traffic as a whole or for each traffic class (in case it is desired to monitor each class separately), but in the second case a couple of counters is needed for each class.

As mentioned, the flow-based measurement requires the identification of the flow to be monitored and the discovery of the path followed by the selected flow. It is possible to monitor a single flow or multiple flows grouped together, but in this case measurement is consistent only if all the flows in the group follow the same path. Moreover if a measurement is performed by grouping many flows, it is not possible to determine exactly which flow was affected by packets loss. In order to have measures per single flow it is necessary to configure counters for each specific flow. Once the flow(s) to be monitored have been identified, it is necessary to configure the monitoring on the proper nodes. Configuring the monitoring means configuring the rule to intercept the traffic and configuring the counters to count the packets. To have just an end-to-end monitoring, it is sufficient to enable the monitoring on the first and the last hop routers of the path: the mechanism is completely transparent to intermediate nodes and independent from the path followed by traffic flows. On the contrary, to monitor the flow on a hop-by-hop basis along its whole path it is necessary to enable the monitoring on every node from the source to the destination. In case the exact path followed by the flow is not known a priori (i.e. the flow has multiple paths to reach the destination) it is necessary to enable the monitoring system on every path: counters on interfaces traversed by the flow will report packet count, counters on other interfaces will be null.

### 3.1.1. Coloring the packets

The coloring operation is fundamental in order to create packet blocks. This implies choosing where to activate the coloring and how to color the packets.

In case of flow-based measurements, the flow to monitor can be defined by a set of selection rules (e.g. headers fields) used to match a subset of the packets; in this way it is possible to control the number of involved nodes, the path followed by the packets and the size of the flows. it is possible, in general, to have multiple coloring nodes or a single coloring node that is easier to manage and doesn't rise any risk of conflict. Coloring in multiple nodes can be done and the requirement is that the coloring must change periodically between the nodes according to the timing considerations in Section 3.2; so every node, that is designated as a measurement point along the path, should be able to identify unambiguously the colored packets. Furthermore [I-D.fioccola-ippm-multipoint-alt-mark] generalizes the coloring for multipoint to multipoint flow. In addition, it can be advantageous to color the flow as close as possible to the source because it allows an end-to-end measure if a measurement point is enabled on the last-hop router as well.

For link-based measurements, all traffic needs to be colored when transmitted on the link. If the traffic had already been colored, then it has to be re-colored because the color must be consistent on the link. This means that each hop along the path must (re-)color the traffic; the color is not required to be consistent along different links.

Traffic coloring can be implemented by setting a specific bit in the packet header and changing the value of that bit periodically. How to choose the marking field depends on the application and is out of scope here. However some applications are reported in Section 5.

### 3.1.2. Counting the packets

For flow-based measurements, assuming that the coloring of the packets is performed only by the source nodes, the nodes between source and destination (included) have to count the colored packets that they receive and forward: this operation can be enabled on every router along the path or only on a subset, depending on which network segment is being monitored (a single link, a particular metro area, the backbone, the whole path). Since the color switches periodically between two values, two counters (one for each value) are needed: one counter for packets with color A and one counter for packets with color B. For each flow (or group of flows) being monitored and for every interface where the monitoring is active, a couple of counters is needed. For example, in order to monitor separately 3 flows on a router with 4 interfaces involved, 24 counters are needed (2 counters for each of the 3 flows on each of the 4 interfaces). Furthermore [I-D.fioccola-ippm-multipoint-alt-mark] generalizes the counting for multipoint to multipoint flow.

In case of link-based measurements the behaviour is similar except that coloring and counting operations are performed on a link by link basis at each endpoint of the link.

Another important aspect to take into consideration is when to read the counters: in order to count the exact number of packets of a block the routers must perform this operation when that block has ended: in other words, the counter for color A must be read when the current block has color B, in order to be sure that the value of the counter is stable. This task can be accomplished in two ways. The general approach suggests to read the counters periodically, many times during a block duration, and to compare these successive readings: when the counter stops incrementing means that the current block has ended and its value can be elaborated safely. Alternatively, if the coloring operation is performed on the basis of a fixed timer, it is possible to configure the reading of the counters according to that timer: for example, reading the counter for color A every period in the middle of the subsequent block with color B is a safe choice. A sufficient margin should be considered between the end of a block and the reading of the counter, in order to take into account any out-of-order packets.

#### 3.1.3. Collecting data and calculating packet loss

The nodes enabled to perform performance monitoring collect the value of the counters, but they are not able to directly use this information to measure packet loss, because they only have their own samples. For this reason, an external Network Management System (NMS) can be used to collect and elaborate data and to perform packet loss calculation. The NMS compares the values of counters from different nodes and can calculate if some packets were lost (even a single packet) and also where packets were lost.

The value of the counters needs to be transmitted to the NMS as soon as it has been read. This can be accomplished by using SNMP or FTP and can be done in Push Mode or Polling Mode. In the first case, each router periodically sends the information to the NMS, in the latter case it is the NMS that periodically polls routers to collect information. In any case, the NMS has to collect all the relevant values from all the routers within one cycle of the timer.

it would be also possible to use a protocol to exchange values of counters between the two endpoints in order to let them perform the packet loss calculation for each traffic direction.

A possible approach for the performance measurement architecture is explained in [I-D.chen-ippm-coloring-based-ipfpm-framework], while [I-D.chen-ippm-ipfpm-report] introduces new information elements of IPFIX (RFC 7011 [RFC7011]).

## 3.2. Timing aspects

This document introduces two color switching method: one is based on fixed number of packet, the other is based on fixed timer. But the method based on fixed timer is preferable because is more deterministic, and will be considered in the rest of the dcoument.

By considering the clock error between network devices R1 and R2, they must be synchronized to the same clock reference with an accuracy of +/- L/2 time units, where L is the time duration of the block. So each colored packet can be assigned to the right batch by each router. This is because the minimum time distance between two packets of the same color but belonging to different batches is L time units.

In practice, there are also out of order at batch boundaries, strictly related to the delay between measurement points. This means that, without considering clock error, we wait L/2 after color switching to be sure to take a still counter.

In summary we need to take into account two contributions: clock error between network devices and the interval we need to wait to avoid out of order because of network delay.

The following figure explains both issues.

```
...BBBBBBBBB | AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA | BBBBBBBB...
             |<======================================>|
             |                   L                    |
...=========>|<==================><==================>|<==========...
             |       L/2                   L/2        |
             |<===>|                             |<===>|
               d   |                             |  d
                   |<===========================>|
                    available counting interval
```

Figure 4: Timing aspects

It is assumed that all network devices are synchronized to a common reference time with an accuracy of +/- A/2. Thus, the difference between the clock values of any two network devices is bounded by A.

The guardband d is given by:

d = A + D_max - D_min,

where A is the clock accuracy, D_max is an upper bound on the network delay between the network devices, and D_min is a lower bound on the delay.

The available counting interval is L - 2d that must be > 0.

The condition that must be satisfied and is a requirement on the synchronization accuracy is:

d < L/2.

### 3.3. One-way delay measurement

The same principle used to measure packet loss can be applied also to one-way delay measurement. There are three alternatives, as described hereinafter.

#### 3.3.1. Single marking methodology

The alternation of colors can be used as a time reference to calculate the delay. Whenever the color changes (that means that a new block has started) a network device can store the timestamp of the first packet of the new block; that timestamp can be compared with the timestamp of the same packet on a second router to compute packet delay. Considering Figure 2, R1 stores a timestamp TS(A1)R1 when it sends the first packet of block 1 (A-colored), a timestamp TS(B2)R1 when it sends the first packet of block 2 (B-colored) and so on for every other block. R2 performs the same operation on the receiving side, recording TS(A1)R2, TS(B2)R2 and so on. Since the timestamps refer to specific packets (the first packet of each block) we are sure that timestamps compared to compute delay refer to the same packets. By comparing TS(A1)R1 with TS(A1)R2 (and similarly TS(B2)R1 with TS(B2)R2 and so on) it is possible to measure the delay between R1 and R2. In order to have more measurements, it is possible to take and store more timestamps, referring to other packets within each block.

In order to coherently compare timestamps collected on different routers, the clocks on the network nodes must be in sync. Furthermore, a measurement is valid only if no packet loss occurs and if packet misordering can be avoided, otherwise the first packet of a block on R1 could be different from the first packet of the same block on R2 (f.i. if that packet is lost between R1 and R2 or it arrives after the next one).

The following table shows how timestamps can be used to calculate the delay between R1 and R2. The first column lists the sequence of blocks while other columns contain the timestamp referring to the first packet of each block on R1 and R2. The delay is computed as a difference between timestamps. For the sake of simplicity, all the values are expressed in milliseconds.

| Block | TS(A)R1 | TS(B)R1 | TS(A)R2 | TS(B)R2 | Delay R1-R2 |
|---|---|---|---|---|---|
| 1 | 12.483 | - | 15.591 | - | 3.108 |
| 2 | - | 6.263 | - | 9.288 | 3.025 |
| 3 | 27.556 | - | 30.512 | - | 2.956 |
| | - | 18.113 | - | 21.269 | 3.156 |
| ... | ... | ... | ... | ... | ... |
| 2n | 77.463 | - | 80.501 | - | 3.038 |
| 2n+1 | - | 24.333 | - | 27.433 | 3.100 |

Table 2: Evaluation of timestamps for delay measurements

The first row shows timestamps taken on R1 and R2 respectively and referring to the first packet of block 1 (which is A-colored). Delay can be computed as a difference between the timestamp on R2 and the timestamp on R1. Similarly, the second row shows timestamps (in milliseconds) taken on R1 and R2 and referring to the first packet of block 2 (which is B-colored). Comparing timestamps taken on different nodes in the network and referring to the same packets (identified using the alternation of colors) it is possible to measure delay on different network segments.

For the sake of simplicity, in the above example a single measurement is provided within a block, taking into account only the first packet of each block. The number of measurements can be easily increased by considering multiple packets in the block: for instance, a timestamp could be taken every N packets, thus generating multiple delay measurements. Taking this to the limit, in principle the delay could be measured for each packet, by taking and comparing the corresponding timestamps (possible but impractical from an implementation point of view).

#### 3.3.1.1. Mean delay

As mentioned before, the method previously exposed for measuring the delay is sensitive to out of order reception of packets. In order to overcome this problem, a different approach has been considered: it is based on the concept of mean delay. The mean delay is calculated by considering the average arrival time of the packets within a single block. The network device locally stores a timestamp for each packet received within a single block: summing all the timestamps and dividing by the total number of packets received, the average arrival time for that block of packets can be calculated. By subtracting the average arrival times of two adjacent devices it is possible to calculate the mean delay between those nodes. When computing the mean delay, measurement error could be augmented by accumulating measurement error of a lot of packets. This method is robust to out of order packets and also to packet loss (only a small error is introduced). Moreover, it greatly reduces the number of timestamps (only one per block for each network device) that have to be collected by the management system. On the other hand, it only gives one measure for the duration of the block (f.i. 5 minutes), and it doesn't give the minimum, maximum and median delay values (RFC 6703 [RFC6703]). This limitation could be overcome by reducing the duration of the block (f.i. from 5 minutes to a few seconds), that implicates an highly optimized implementation of the method.

By summing the mean delays of the two directions of a path, it is also possible to measure the two-way mean delay (round-trip delay).

### 3.3.2. Double marking methodology

The Single marking methodology for one-way delay measurement is sensitive to out of order reception of packets. The first approach to overcome this problem is described before and is based on the concept of mean delay. But the limitation of mean delay is that it doesn't give information about the delay values distribution for the duration of the block. Additionally it may be useful to have not only the mean delay but also the minimum, maximum and median delay values and, in wider terms, to know more about the statistic distribution of delay values. So in order to have more information about the delay and to overcome out of order issues, a different approach can be introduced: it is based on double marking methodology.

Basically, the idea is to use the first marking to create the alternate flow and, within this colored flow, a second marking to select the packets for measuring delay/jitter. The first marking is needed for packet loss and mean delay measurement. The second marking creates a new set of marked packets that are fully identified over the network, so that a network device can store the timestamps of these packets; these timestamps can be compared with the timestamps of the same packets on a second router to compute packet delay values for each packet. The number of measurements can be easily increased by changing the frequency of the second marking. But the frequency of the second marking must be not too high in order to avoid out of order issues. Between packets with the second marking there should be a security time gap (e.g. this gap could be, at the minimum, the mean network delay calculated with the previous methodology) to avoid out of order issues and also to have a number of measurement packets that is rate independent. If a second marking packet is lost, the delay measurement for the considered block is corrupted and should be discarded.

Mean delay is calculated on all the packets of a sample and is a simple computation to be performed for single marking method. In some cases the mean delay measure is not sufficient to characterize the sample, and more statistics of delay extent data are needed, e.g. percentiles, variance and median delay values. The conventional range (maximum-minimum) should be avoided for several reasons, including stability of the maximum delay due to the influence by outliers. RFC 5481 [RFC5481] Section 6.5 highlights how the 99.9th percentile of delay and delay variation is more helpful to performance planners. To overcome this drawback the idea is to couple the mean delay measure for the entire batch with double marking method, where a subset of batch packets are selected for extensive delay calculation by using a second marking. In this way it is possible to perform a detailed analysis on these double marked packets. Please note that there are classic algorithms for median and variance calculation, but are out of the scope of this document. The comparison between the mean delay for the entire batch and the mean delay on these double marked packets gives an useful information since it is possible to understand if the double marking measurements are actually representative of the delay trends.

### 3.4. Delay variation measurement

Similarly to one-way delay measurement (both for single marking and double marking), the method can also be used to measure the inter-arrival jitter. We refer to the definition in RFC 3393 [RFC3393]. The alternation of colors, for single marking method, can be used as a time reference to measure delay variations. In case of double marking, the time reference is given by the second marked packets.

Considering the example depicted in Figure 2, R1 stores a timestamp TS(A)R1 whenever it sends the first packet of a block and R2 stores a timestamp TS(B)R2 whenever it receives the first packet of a block. The inter-arrival jitter can be easily derived from one-way delay measurement, by evaluating the delay variation of consecutive samples.

The concept of mean delay can also be applied to delay variation, by evaluating the average variation of the interval between consecutive packets of the flow from R1 to R2.

## 4. Considerations

This section highlights some considerations about the methodology.

### 4.1. Synchronization

The Alternate Marking technique does not require a strong synchronization, especially for packet loss and two-way delay measurement. Only one-way delay measurement requires network devices to have synchronized clocks.

The color switching is the reference for all the network devices, and the only requirement to be achieved is that all network devices have to recognize the right batch along the path.

If the length of the measurement period is L time units, then all network devices must be synchronized to the same clock reference with an accuracy of +/- L/2 time units (without considering network delay). This level of accuracy guarantees that all network devices consistently match the color bit to the correct block. For example, if the color is toggeled every second (L = 1 second), then clocks must be synchronized with an accuracy of +/- 0.5 second to a common time reference.

This synchronization requirement can be satisfied even with a relatively inaccurate synchronization method. This is true for packet loss and two-way delay measurement, instead, for one-way delay measurement clock synchronization must be accurate.

Therefore, a system that uses only packet loss and two-way delay measurement does not require synchronization. This is because the value of the clocks of network devices does not affect the computation of the two-way delay measurement.

## 4.2. Data Correlation

Data Correlation is the mechanism to compare counters and timestamps for packet loss, delay and delay variation calculation. It could be performed in several ways depending on the alternate marking application and use case.

- A possibility is to use a centralized solution using Network Management System (NMS) to correlate data;

- Another possibility is to define a protocol based distributed solution, by defining a new protocol or by extending the existing protocols (e.g. RFC6374, TWAMP, OWAMP) in order to communicate the counters and timestamps between nodes.

In the following paragraphs an example data correlation mechanism is explained and could be use independently of the adopted solutions.

When data is collected on the upstream and downstream node, e.g., packet counts for packet loss measurement or timestamps for packet delay measurement, and periodically reported to or pulled by other nodes or NMS, a certain data correlation mechanism SHOULD be in use to help the nodes or NMS to tell whether any two or more packet counts are related to the same block of markers, or any two timestamps are related to the same marked packet.

The alternate marking method described in this document literally split the packets of the measured flow into different measurement blocks, in addition a Block Number could be assigned to each of such measurement block. The BN is generated each time a node reads the data (packet counts or timestamps), and is associated with each packet count and timestamp reported to or pulled by other nodes or NMS. The value of BN could be calculated as the modulo of the local time (when the data are read) and the interval of the marking time period.

When the nodes or NMS see, for example, same BNs associated with two packet counts from an upstream and a downstream node respectively, it considers that these two packet counts corresponding to the same block, i.e. that these two packet counts belong to the same block of markers from the upstream and downstream node. The assumption of this BN mechanism is that the measurement nodes are time synchronized. This requires the measurement nodes to have a certain time synchronization capability (e.g., the Network Time Protocol (NTP) RFC 5905 [RFC5905], or the IEEE 1588 Precision Time Protocol (PTP) [IEEE-1588]). Synchronization aspects are further discussed in Section 4.

### 4.3. Packet Re-ordering

Due to ECMP, packet re-ordering is very common in IP network. The accuracy of marking based PM, especially packet loss measurement, may be affected by packet re-ordering. Take a look at the following example:

```
Block   :    1     |    2     |    3     |    4     |    5     |...
--------|---------|----------|----------|----------|----------|---
Node R1 : AAAAAAA | BBBBBBB | AAAAAAA | BBBBBBB | AAAAAAA |...
Node R2 : AAAAABB | AABBBBA | AAABAAA | BBBBBBA | ABAAABA |...
```

Figure 5: Packet Reordering

In Figure 5 the packet stream for Node R1 isn't being reordered, and can be safely assigned to interval blocks, but the packet stream for Node R2 is being reordered, so, looking at the packet with the marker of "B" in block 3, there is no safe way to tell whether the packet belongs to block 2 or block 4.

In general there is the need to assign packets with the marker of "B" or "A" to the right interval blocks. Most of the packet re-ordering occur at the edge of adjacent blocks, and they are easy to handle if the interval of each block is sufficient large. Then, it can assume that the packets with different marker belong to the block that they are more close to. If the interval is small, it is difficult and sometime impossible to determine to which block a packet belongs.

To choose a proper interval is important and how to choose a proper interval is out of the scope of this document. But an implementation SHOULD provide a way to configure the interval and allow a certain degree of packet re-ordering.

## 5. Applications, implementation and deployment

The methodology described in the previous sections can be applied in various situations. Basically Alternate Marking technique could be used in many cases for performance measurement. The only requirement is to select and mark the flow to be monitored; in this way packets are batched by the sender and each batch is alternately marked such that can be easily recognized by the receiver.

Some recent alternate marking method applications are listed below:

o IP flow performance measurement (IPFPM): this application of marking method is described in [I-D.chen-ippm-coloring-based-ipfpm-framework]. As an example, in this document, the last reserved bit of the Flag field of the IPv4 header is proposed to be used for marking, while a solution for IPv6 could be to leverage the IPv6 extension header for marking.

- OAM Passive Performance Measurement: In [I-D.ietf-bier-mpls-encapsulation] two OAM bits from Bit Index Explicit Replication (BIER) Header are reserved for the passive performance measurement marking method. [I-D.ietf-bier-pmmm-oam] details the measurement for multicast service over BIER domain. In addition, the alternate marking method could also be used in a Service Function Chaining (SFC) domain. Lastly the application of the marking method to Network Virtualization Overlays (NVO3) protocols is considered by [I-D.ietf-nvo3-encap].

- RFC6374 Use Case: RFC6374 [RFC6374] uses the LM packet as the packet accounting demarcation point. Unfortunately this gives rise to a number of problems that may lead to significant packet accounting errors in certain situations. [I-D.ietf-mpls-flow-ident] discusses the desired capabilities for MPLS flow identification in order to perform a better in-band performance monitoring of user data packets. A method of accomplishing identification is Synonymous Flow Labels (SFL) introduced in [I-D.bryant-mpls-sfl-framework], while [I-D.ietf-mpls-rfc6374-sfl] describes RFC6374 performance measurements with SFL.

- active performance measurement: [I-D.fioccola-ippm-alt-mark-active] describes how to extend the existing Active Measurement Protocol, in order to implement alternate marking methodology. [I-D.fioccola-ippm-rfc6812-alt-mark-ext] describes an extension to the Cisco SLA Protocol Measurement-Type UDP-Measurement.

An example of implementation and deployment is explained in the next section, just to clarify how the method can work.

### 5.1. Report on the operational experiment

The method described in this document, also called PNPM (Packet Network Performance Monitoring), has been invented and engineered in Telecom Italia.

It is important to highlight that the general description of the methodology in this document is a consequence of the operational experiment. The foundational elements of the technique have been tested and the lessons learnt from the operational experiment inspired the formalization of the Alternate Marking Method as detailed in the previous sections.

The methodology is experimented in Telecom Italia's network and is applied to multicast IPTV channels or other specific traffic flows with high QoS requirements (i.e. Mobile Backhauling traffic realized with a VPN MPLS).

This technology has been employed by leveraging functions and tools available on IP routers and it's currently being used to monitor packet loss in some portions of the Telecom Italia's network. The application of the method to delay measurement has also been evaluated in Telecom Italia's labs.

This Section describes how the experiment has been executed, in particular how the features currently available on existing routing platforms can be used to apply the method, in order to give an example of implementation and deployment.

The operational test, here described, uses the flow-based strategy, as defined in Section 3. Instead the link-based strategy could be applied to physical link or a logical link (e.g. Ethernet VLAN or a MPLS PW).

The implementation of the method leverages the available router functions, since the experiment has been done by a Service Provider (as Telecom Itlaia is) on its own network. So, with current router implementations, only QoS related fields and features offer the required flexibility to set bits in the packet header. In case a Service Provider only uses the three most significant bits of the DSCP field (corresponding to IP Precedence) for QoS classification and queuing, it is possible to use the two less significant bits of the DSCP field (bit 0 and bit 1) to implement the method without affecting QoS policies. That is the approach used for the experiment. One of the two bits (bit 0) could be used to identify flows subject to traffic monitoring (set to 1 if the flow is under monitoring, otherwise it is set to 0), while the second (bit 1) can be used for coloring the traffic (switching between values 0 and 1, corresponding to color A and B) and creating the blocks.

The experiment considers a flow as all the packets sharing the same source IP address or the same destination IP address, depending on the direction. In practice, once the flow has been defined, coloring the traffic using the DSCP field can be implemented by configuring on the router output interface an access list that intercepts the flow(s) to be monitored and applies to them a policy that sets the DSCP field accordingly. Since traffic coloring has to be switched between the two values over time, the policy needs to be modified periodically: an automatic script is used to perform this task on the basis of a fixed timer. The automatic script is loaded on board of the router and automatizes the basic operations that are needed to realize the methodology.

After the traffic is colored using the DSCP field, all the routers on the path can perform the counting. For this purpose an access-list that matches specific DSCP values can be used to count the packets of the flow(s) being monitored. The same access-list can be installed on all the routers of the path. In addition, network flow monitoring, such as provided by IPFIX (RFC 7011 [RFC7011]), can be used to recognize timestamps of first/last packet of a batch in order to enable one of the alternatives to measure the delay as detailed in Section 3.3.

In the Telecom Italia's experiment the timer is set to 5 minutes, so the sequence of actions of the script is also executed every 5 minutes. This value has showed to be a good compromise between measurement frequency and stability of the measurement (i.e. possibility to collect all the measures referring to the same block).

For this expertiment, both counters and any other data are collected by using the automatic script that sends out these to a Network Management System (NMS). The NMS is responsible for packet loss calculation, performed by comparing the values of counters from the routers along the flow(s) path. 5 minutes timer for color switching is a safe choice for reading the counters and is also coherent with the reporting window of the NMS.

Note that the use of the DSCP field for marking implies that the method in this case works reliably only within a single management and operation domain.

Lastly, the Telecom Italia experiment scales up to 1000 flows monitored together on a single router, while an implementation on dedicated hardware scales more, but it was tested only in labs for now.

#### 5.1.1. Metric transparency

Since a Service Provider application is described here, the method can be applied to end-to-end services supplied to Customers. So it is important to highlight that the method MUST be transparent outside the Service Provider domain.

In Telecom Italia's implementation the source node colors the packets with a policy that is modified periodically via an automatic script in order to alternate the DSCP field of the packets. The nodes between source and destination (included) have to count with an access-list the colored packets that they receive and forward.

Moreover the destination node has an important role: the colored packets are intercepted and a policy restores and sets the DSCP field of all the packets to the initial value. In this way the metric is transparent because outside the section of the network under monitoring the traffic flow is unchanged.

In such a case, thanks to this restoring technique, network elements outside the Alternate Marking monitoring domain (e.g. the two Provider Edge nodes of the Mobile Backhauling VPN MPLS) are totally anaware that packets were marked. So this restoring technique makes Alternate Marking completely transparent outside its monitoring domain.

## 6. Hybrid measurement

The method has been explicitly designed for passive measurements but it can also be used with active measurements. In order to have both end to end measurements and intermediate measurements (hybrid measurements) two end points can exchanges artificial traffic flows and apply alternate marking over these flows. In the intermediate points artificial traffic is managed in the same way as real traffic and measured as specified before. So the application of marking method can simplify also the active measurement, as explained in [I-D.fioccola-ippm-alt-mark-active].

## 7. Compliance with RFC6390 guidelines

RFC6390 [RFC6390] defines a framework and a process for developing Performance Metrics for protocols above and below the IP layer (such as IP-based applications that operate over reliable or datagram transport protocols).

This document doesn’t aim to propose a new Performance Metric but a new method of measurement for a few Performance Metrics that have already been standardized. Nevertheless, it’s worth applying [RFC6390] guidelines to the present document, in order to provide a more complete and coherent description of the proposed method. We used a subset of the Performance Metric Definition template defined by [RFC6390].

- o Metric name and description: as already stated, this document doesn’t propose any new Performance Metric. On the contrary, it describes a novel method for measuring packet loss [RFC7680]. The same concept, with small differences, can also be used to measure delay [RFC7679], and jitter [RFC3393]. The document mainly describes the applicability to packet loss measurement.

o Method of Measurement or Calculation: according to the method described in the previous sections, the number of packets lost is calculated by subtracting the value of the counter on the source node from the value of the counter on the destination node. Both counters must refer to the same color. The calculation is performed when the value of the counters is in a steady state. The steady state is an intrinsic characteristic of the marking method counters because the alternation of color makes the counters associated to each color still one at a time for the duration of a marking period.

o Units of Measurement: the method calculates and reports the exact number of packets sent by the source node and not received by the destination node.

o Measurement Points: the measurement can be performed between adjacent nodes, on a per-link basis, or along a multi-hop path, provided that the traffic under measurement follows that path. In case of a multi-hop path, the measurements can be performed both end-to-end and hop-by-hop.

o Measurement Timing: the method have a constraint on the frequency of measurements. This is detailed in Section 3.2, where it is specified that the marking period and the guardband interval are strictly related each other to avoid out of order issues. That is because, in order to perform a measure, the counter must be in a steady state and this happens when the traffic is being colored with the alternate color. As an example in the experiment of the method the time interval is set to 5 minutes, while other optimized implementations can also use a marking period of a few seconds.

o Implementation: the experiment of the method uses two encodings of the DSCP field to color the packets; this enables the use of policy configurations on the router to color the packets and accordingly configure the counter for each color. The path followed by traffic being measured should be known in advance in order to configure the counters along the path and be able to compare the correct values.

o Verification: both in the Lab and in the operational network the methodology has been tested and experimented for packet loss and delay measurements by using traffic generators together with precision test instruments and network emulators.

o Use and Applications: the method can be used to measure packet loss with high precision on live traffic; moreover, by combining end-to-end and per-link measurements, the method is useful to pinpoint the single link that is experiencing loss events.

- Reporting Model: the value of the counters has to be sent to a centralized management system that perform the calculations; such samples must contain a reference to the time interval they refer to, so that the management system can perform the correct correlation; the samples have to be sent while the corresponding counter is in a steady state (within a time interval), otherwise the value of the sample should be stored locally.

- Dependencies: the values of the counters have to be correlated to the time interval they refer to; moreover, as far the experiment of the method is based on DSCP values, there are significant dependencies on the usage of the DSCP field: it must be possible to rely on unused DSCP values without affecting QoS-related configuration and behavior; moreover, the intermediate nodes must not change the value of the DSCP field not to alter the measurement.

- Organization of Results: the method of measurement produces singletons.

- Parameters: currently, the main parameter of the method is the time interval used to alternate the colors and read the counters.

## 8. Security Considerations

This document specifies a method to perform measurements in the context of a Service Provider's network and has not been developed to conduct Internet measurements, so it does not directly affect Internet security nor applications which run on the Internet. However, implementation of this method must be mindful of security and privacy concerns.

There are two types of security concerns: potential harm caused by the measurements and potential harm to the measurements.

- Harm caused by the measurement: the measurements described in this document are passive, so there are no new packets injected into the network causing potential harm to the network itself and to data traffic. Nevertheless, the method implies modifications on the fly to a header or encapsulation of the data packets: this must be performed in a way that doesn't alter the quality of service experienced by packets subject to measurements and that preserve stability and performance of routers doing the measurements. One of the main security threats in OAM protocols is network reconnaissance; an attacker can gather information about the network performance by passively eavesdropping to OAM messages. The advantage of the methods described in this document is that the marking bits are the only information that is exchanged between the network devices. Therefore, passive eavesdropping to data plane traffic does not allow attackers to gain information about the network performance.

o Harm to the measurement: the measurements could be harmed by routers altering the marking of the packets, or by an attacker injecting artificial traffic. Authentication techniques, such as digital signatures, may be used where appropriate to guard against injected traffic attacks. Since the measurement itself may be affected by routers (or other network devices) along the path of IP packets intentionally altering the value of marking bits of packets, as mentioned above, the mechanism specified in this document can be applied just in the context of a controlled domain, and thus the routers (or other network devices) are locally administered and this type of attack can be avoided. In addition, an attacker can't gain information about network performance from a single monitoring point, and must use synchronized monitoring points at multiple points on the path, because they have to do the same kind of measurement and aggregation that Service Providers using Alternate Marking must do.

The privacy concerns of network measurement are limited because the method only relies on information contained in the header or encapsulation without any release of user data.Although information in the header or encapsulation is metadata that can be used to compromise the privacy of users, the limited marking technique in this document seems unlikely to substantially increase the existing privacy risks from header or encapsulation metadata.It might be theoretically possible to modulate the marking to serve as a covert channel, but it would have a very low data rate if it is to avoid adversely affecting the measurement systems that monitor the marking.

Delay attacks are another potential threat in the context of this document. Delay measurement is performed using a specific packet in each block, marked by a dedicated color bit. Therefore, a man-in-the-middle attacker can selectively induce synthetic delay only to delay-colored packets, causing systematic error in the delay measurements. As discussed in previous sections, the methods described in this document rely on an underlying time synchronization protocol. Thus, by attacking the time protocol an attacker can potentially compromise the integrity of the measurement. A detailed discussion about the threats against time protocols and how to mitigate them is presented in RFC 7384 [RFC7384].

## 9. IANA Considerations

There are no IANA actions required.

## 10. Acknowledgements

The previous IETF drafts about this technique were: [I-D.cociglio-mboned-multicast-pm] and [I-D.tempia-opsawg-p3m].

The authors would like to thank Alberto Tempia Bonda, Domenico Laforgia, Daniele Accetta and Mario Bianchetti for their contribution to the definition and the implementation of the method.

The authors would also thank Spencer Dawkins, Carlos Pignataro, Brian Haberman and Eric Vyncke for their assistance and their detailed and precious reviews.

## 11. References

### 11.1. Normative References

[IEEE-1588] IEEE 1588-2008, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", July 2008.

[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>.

[RFC3393] Demichelis, C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, DOI 10.17487/RFC3393, November 2002, <https://www.rfc-editor.org/info/rfc3393>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC7679] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Delay Metric for IP Performance Metrics (IPPM)", STD 81, RFC 7679, DOI 10.17487/RFC7679, January 2016, <https://www.rfc-editor.org/info/rfc7679>.

[RFC7680] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Loss Metric for IP Performance Metrics (IPPM)", STD 82, RFC 7680, DOI 10.17487/RFC7680, January 2016, <https://www.rfc-editor.org/info/rfc7680>.

[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>.

## 11.2. Informative References

[I-D.bryant-mpls-sfl-framework]
Bryant, S., Chen, M., Li, Z., Swallow, G., Sivabalan, S., and G. Mirsky, "Synonymous Flow Label Framework", draft-bryant-mpls-sfl-framework-05 (work in progress), June 2017.

[I-D.chen-ippm-coloring-based-ipfpm-framework]
Chen, M., Zheng, L., Mirsky, G., Fioccola, G., and T. Mizrahi, "IP Flow Performance Measurement Framework", draft-chen-ippm-coloring-based-ipfpm-framework-06 (work in progress), March 2016.

[I-D.chen-ippm-ipfpm-report]
Chen, M., Zheng, L., and G. Mirsky, "IP Flow Performance Measurement Report", draft-chen-ippm-ipfpm-report-01 (work in progress), April 2016.

[I-D.cociglio-mboned-multicast-pm]
Cociglio, M., Capello, A., Bonda, A., and L. Castaldelli, "A method for IP multicast performance monitoring", draft-cociglio-mboned-multicast-pm-01 (work in progress), October 2010.

[I-D.fioccola-ippm-alt-mark-active]
Fioccola, G., Clemm, A., Bryant, S., Cociglio, M., Chandramouli, M., and A. Capello, "Alternate Marking Extension to Active Measurement Protocol", draft-fioccola-ippm-alt-mark-active-01 (work in progress), March 2017.

[I-D.fioccola-ippm-multipoint-alt-mark]
Fioccola, G., Cociglio, M., Sapio, A., and R. Sisto, "Multipoint Alternate Marking method for passive and hybrid performance monitoring", draft-fioccola-ippm-multipoint-alt-mark-01 (work in progress), October 2017.

[I-D.fioccola-ippm-rfc6812-alt-mark-ext]
Fioccola, G., Clemm, A., Cociglio, M., Chandramouli, M., and A. Capello, "Alternate Marking Extension to Cisco SLA Protocol RFC6812", draft-fioccola-ippm-rfc6812-alt-mark-ext-01 (work in progress), March 2016.

[I-D.ietf-bier-mpls-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication in MPLS and non-MPLS Networks", draft-ietf-bier-mpls-encapsulation-12 (work in progress), October 2017.

[I-D.ietf-bier-pmmm-oam]
Mirsky, G., Zheng, L., Chen, M., and G. Fioccola, "Performance Measurement (PM) with Marking Method in Bit Index Explicit Replication (BIER) Layer", draft-ietf-bier-pmmm-oam-03 (work in progress), October 2017.

[I-D.ietf-mpls-flow-ident]
Bryant, S., Pignataro, C., Chen, M., Li, Z., and G. Mirsky, "MPLS Flow Identification Considerations", draft-ietf-mpls-flow-ident-05 (work in progress), July 2017.

[I-D.ietf-mpls-rfc6374-sfl]
Bryant, S., Chen, M., Li, Z., Swallow, G., Sivabalan, S., Mirsky, G., and G. Fioccola, "RFC6374 Synonymous Flow Labels", draft-ietf-mpls-rfc6374-sfl-01 (work in progress), December 2017.

[I-D.ietf-nvo3-encap]
Boutros, S., Ganga, I., Garg, P., Manur, R., Mizrahi, T., Mozes, D., Nordmark, E., Smith, M., Aldrin, S., and I. Bagdonas, "NVO3 Encapsulation Considerations", draft-ietf-nvo3-encap-01 (work in progress), October 2017.

[I-D.tempia-opsawg-p3m]
Capello, A., Cociglio, M., Castaldelli, L., and A. Bonda, "A packet based method for passive performance monitoring", draft-tempia-opsawg-p3m-04 (work in progress), February 2014.

[RFC5481] Morton, A. and B. Claise, "Packet Delay Variation Applicability Statement", RFC 5481, DOI 10.17487/RFC5481, March 2009, <https://www.rfc-editor.org/info/rfc5481>.

[RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, DOI 10.17487/RFC6374, September 2011, <https://www.rfc-editor.org/info/rfc6374>.

[RFC6390] Clark, A. and B. Claise, "Guidelines for Considering New Performance Metric Development", BCP 170, RFC 6390, DOI 10.17487/RFC6390, October 2011, <https://www.rfc-editor.org/info/rfc6390>.

[RFC6703] Morton, A., Ramachandran, G., and G. Maguluri, "Reporting IP Network Performance Metrics: Different Points of View", RFC 6703, DOI 10.17487/RFC6703, August 2012, <https://www.rfc-editor.org/info/rfc6703>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <https://www.rfc-editor.org/info/rfc7011>.

[RFC7276] Mizrahi, T., Sprecher, N., Bellagamba, E., and Y. Weingarten, "An Overview of Operations, Administration, and Maintenance (OAM) Tools", RFC 7276, DOI 10.17487/RFC7276, June 2014, <https://www.rfc-editor.org/info/rfc7276>.

[RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, DOI 10.17487/RFC7384, October 2014, <https://www.rfc-editor.org/info/rfc7384>.

[RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <https://www.rfc-editor.org/info/rfc7799>.

## Authors' Addresses

Giuseppe Fioccola (editor)
Telecom Italia
Via Reiss Romoli, 274
Torino 10148
Italy

Email: giuseppe.fioccola@telecomitalia.it

Alessandro Capello
Telecom Italia
Via Reiss Romoli, 274
Torino 10148
Italy

Email: alessandro.capello@telecomitalia.it

Mauro Cociglio
Telecom Italia
Via Reiss Romoli, 274
Torino 10148
Italy

Email: mauro.cociglio@telecomitalia.it

Luca Castaldelli
Telecom Italia
Via Reiss Romoli, 274
Torino 10148
Italy

Email: luca.castaldelli@telecomitalia.it

Mach(Guoyi) Chen
Huawei Technologies

Email: mach.chen@huawei.com

Lianshu Zheng
Huawei Technologies

Email: vero.zheng@huawei.com

Greg Mirsky
ZTE
USA

Email: gregimirsky@gmail.com

Tal Mizrahi
Marvell
6 Hamada st.
Yokneam
Israel

Email: talmi@marvell.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2020

A. Morton
AT&T Labs
M. Bagnulo
UC3M
P. Eardley
BT
K. D'Souza
AT&T Labs

March 9, 2020

# Initial Performance Metrics Registry Entries
draft-ietf-ippm-initial-registry-16

## Abstract

This memo defines the set of Initial Entries for the IANA Performance Metrics Registry. The set includes: UDP Round-trip Latency and Loss, Packet Delay Variation, DNS Response Latency and Loss, UDP Poisson One-way Delay and Loss, UDP Periodic One-way Delay and Loss, ICMP Round-trip Latency and Loss, and TCP round-trip Latency and Loss.

## 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.

## Status of This Memo

Copyright Notice



Table of Contents

# 1. Introduction

This memo proposes an initial set of entries for the Performance Metrics Registry. It uses terms and definitions from the IPPM literature, primarily [RFC2330].

Although there are several standard templates for organizing specifications of performance metrics (see [RFC7679] for an example of the traditional IPPM template, based to large extent on the Benchmarking Methodology Working Group's traditional template in [RFC1242], and see [RFC6390] for a similar template), none of these templates were intended to become the basis for the columns of an IETF-wide registry of metrics. While examining aspects of metric specifications which need to be registered, it became clear that none of the existing metric templates fully satisfies the particular needs of a registry.

Therefore, [I-D.ietf-ippm-metric-registry] defines the overall format for a Performance Metrics Registry. Section 5 of [I-D.ietf-ippm-metric-registry] also gives guidelines for those requesting registration of a Metric, that is the creation of entry(s) in the Performance Metrics Registry: "In essence, there needs to be evidence that a candidate Registered Performance Metric has significant industry interest, or has seen deployment, and there is agreement that the candidate Registered Performance Metric serves its intended purpose." The process in [I-D.ietf-ippm-metric-registry] also requires that new entries are administered by IANA through Specification Required policy, which will ensure that the metrics are tightly defined.

## 2. Scope

This document defines a set of initial Performance Metrics Registry entries. Most are Active Performance Metrics, which are based on RFCs prepared in the IPPM working group of the IETF, according to their framework [RFC2330] and its updates.

## 3. Registry Categories and Columns

This memo uses the terminology defined in [I-D.ietf-ippm-metric-registry].

This section provides the categories and columns of the registry, for easy reference. An entry (row) therefore gives a complete description of a Registered Metric.

```
Legend:
 Registry Categories and Columns, shown as
                                             Category
                                             ------------------
                                             Column |  Column |

Summary
------------------------------------------------------------------------
Identifier | Name | URI | Desc. | Reference | Change Controller | Ver |

Metric Definition
-----------------------------------------
Reference Definition | Fixed Parameters |

Method of Measurement
---------------------------------------------------------------------
Reference | Packet       | Traffic | Sampling     | Run-time   | Role |
Method    | Stream       | Filter  | Distribution | Parameters |      |
          | Generation   |
Output
-----------------------------------------
Type | Reference   | Units | Calibration |
     | Definition  |       |             |

Administrative Information
------------------------------------
Status |Requester | Rev | Rev.Date |

Comments and Remarks
--------------------
```

## 4. UDP Round-trip Latency and Loss Registry Entries

This section specifies an initial registry entry for the UDP Round-trip Latency, and another entry for UDP Round-trip Loss Ratio.

Note: Each Registry entry only produces a "raw" output or a statistical summary. To describe both "raw" and one or more statistics efficiently, the Identifier, Name, and Output Categories can be split and a single section can specify two or more closely-related metrics. For example, this section specifies two Registry entries with many common columns. See Section 7 for an example specifying multiple Registry entries with many common columns.

All column entries beside the ID, Name, Description, and Output Reference Method categories are the same, thus this section proposes two closely-related registry entries. As a result, IANA is also asked to assign a corresponding URL to each Named Metric.

## 4.1. Summary

This category includes multiple indexes to the registry entry: the element ID and metric name.

### 4.1.1. ID (Identifier)

IANA is asked to assign different numeric identifiers to each of the two Named Metrics.

### 4.1.2. Name

RTDelay_Active_IP-UDP-Periodic_RFCXXXXsec4_Seconds_95Percentile

RTLoss_Active_IP-UDP-Periodic_RFCXXXXsec4_Percent_LossRatio

### 4.1.3. URI

URL: https://www.iana.org/ ... <name>

### 4.1.4. Description

RTDelay: This metric assesses the delay of a stream of packets exchanged between two hosts (which are the two measurement points), and the Output is the Round-trip delay for all successfully exchanged packets expressed as the 95th percentile of their conditional delay distribution.

RTLoss: This metric assesses the loss ratio of a stream of packets exchanged between two hosts (which are the two measurement points), and the Output is the Round-trip loss ratio for all successfully exchanged packets expressed as a percentage.

### 4.1.5. Change Controller

IETF

### 4.1.6. Version (of Registry Format)

1.0

### 4.2. Metric Definition

This category includes columns to prompt the entry of all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters.

#### 4.2.1. Reference Definition

Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, September 1999.

[RFC2681]

Section 2.4 of [RFC2681] provides the reference definition of the singleton (single value) Round-trip delay metric. Section 3.4 of [RFC2681] provides the reference definition expanded to cover a multi-singleton sample. Note that terms such as singleton and sample are defined in Section 11 of [RFC2330].

Note that although the [RFC2681] definition of "Round-trip-Delay between Src and Dst" is directionally ambiguous in the text, this metric tightens the definition further to recognize that the host in the "Src" role will send the first packet to "Dst", and ultimately receive the corresponding return packet from "Dst" (when neither are lost).

Finally, note that the variable "dT" is used in [RFC2681] to refer to the value of Round-trip delay in metric definitions and methods. The variable "dT" has been re-used in other IPPM literature to refer to different quantities, and cannot be used as a global variable name.

Morton, A., "Round-trip Packet Loss Metrics", RFC 6673, August 2012.

[RFC6673]

Both delay and loss metrics employ a maximum waiting time for received packets, so the count of lost packets to total packets sent is the basis for the loss ratio calculation as per Section 6.1 of [RFC6673].

#### 4.2.2. Fixed Parameters

Type-P as defined in Section 13 of [RFC2330]:

- o IPv4 header values:

   * DSCP: set to 0

   * TTL: set to 255

   * Protocol: set to 17 (UDP)

- IPv6 header values:

   * DSCP: set to 0

   * Hop Count: set to 255

   * Next Header: set to 17 (UDP)

   * Flow Label: set to zero

   * Extension Headers: none

- UDP header values:

   * Checksum: the checksum MUST be calculated and the non-zero checksum included in the header

- UDP Payload

   * total of 100 bytes

Other measurement parameters:

- Tmax: a loss threshold waiting time

   * 3.0, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 4 (see section 9.3 of [RFC6020]) and with resolution of 0.0001 seconds (0.1 ms), with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905].

### 4.3. Method of Measurement

This category includes columns for references to relevant sections of the RFC(s) and any supplemental information needed to ensure an unambiguous methods for implementations.

#### 4.3.1. Reference Method

The methodology for this metric is defined as Type-P-Round-trip-Delay-Poisson-Stream in section 2.6 of RFC 2681 [RFC2681] and section 3.6 of RFC 2681 [RFC2681] using the Type-P and Tmax defined under Fixed Parameters. However, the Periodic stream will be generated according to [RFC3432].

The reference method distinguishes between long-delayed packets and lost packets by implementing a maximum waiting time for packet arrival. Tmax is the waiting time used as the threshold to declare a packet lost. Lost packets SHALL be designated as having undefined delay, and counted for the RTLoss metric.

The calculations on the delay (RTT) SHALL be performed on the conditional distribution, conditioned on successful packet arrival within Tmax. Also, when all packet delays are stored, the process which calculates the RTT value MUST enforce the Tmax threshold on stored values before calculations. See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

The reference method requires some way to distinguish between different packets in a stream to establish correspondence between sending times and receiving times for each successfully-arriving packet. Sequence numbers or other send-order identification MUST be retained at the Src or included with each packet to disambiguate packet reordering if it occurs.

If a standard measurement protocol is employed, then the measurement process will determine the sequence numbers or timestamps applied to test packets after the Fixed and Runtime parameters are passed to that process. The chosen measurement protocol will dictate the format of sequence numbers and time-stamps, if they are conveyed in the packet payload.

Refer to Section 4.4 of [RFC6673] for expanded discussion of the instruction to "send a Type-P packet back to the Src as quickly as possible" in Section 2.6 of RFC 2681 [RFC2681]. Section 8 of [RFC6673] presents additional requirements which MUST be included in the method of measurement for this metric.

### 4.3.2. Packet Stream Generation

This section gives the details of the packet traffic which is the basis for measurement. In IPPM metrics, this is called the Stream, and can easily be described by providing the list of stream parameters.

Section 3 of [RFC3432] prescribes the method for generating Periodic streams using associated parameters.

incT the nominal duration of inter-packet interval, first bit to first bit, with value 0.0200, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 4 (see section 9.3 of [RFC6020]) and with resolution of 0.0001 seconds (0.1 ms).

dT the duration of the interval for allowed sample start times, with value 1.0, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 4 (see section 9.3 of [RFC6020]) and with resolution of 0.0001 seconds (0.1 ms).

NOTE: an initiation process with a number of control exchanges resulting in unpredictable start times (within a time interval) may be sufficient to avoid synchronization of periodic streams, and therefore a valid replacement for selecting a start time at random from a fixed interval.

The T0 parameter will be reported as a measured parameter. Parameters incT and dT are Fixed Parameters.

#### 4.3.3. Traffic Filtering (observation) Details

NA

#### 4.3.4. Sampling Distribution

NA

#### 4.3.5. Run-time Parameters and Data Format

Run-time Parameters are input factors that must be determined, configured into the measurement system, and reported with the results for the context to be complete.

Src the IP address of the host in the Src Role (format ipv4-address-no-zone value for IPv4, or ipv6-address-no-zone value for IPv6, see Section 4 of [RFC6991])

Dst the IP address of the host in the Dst Role (format ipv4-address-no-zone value for IPv4, or ipv6-address-no-zone value for IPv6, see section 4 of [RFC6991])

T0 a time, the start of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330]. When T0 is "all-zeros", a start time is unspecified and Tf is to be interpreted as the Duration of the measurement interval. The start time is controlled through other means.

Tf a time, the end of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of

[RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330]. When T0 is "all-zeros", a end time date is ignored and Tf is interpreted as the Duration of the measurement interval.

#### 4.3.6. Roles

Src launches each packet and waits for return transmissions from Dst.

Dst waits for each packet from Src and sends a return packet to Src.

### 4.4. Output

This category specifies all details of the Output of measurements using the metric.

#### 4.4.1. Type

Percentile -- for the conditional distribution of all packets with a valid value of Round-trip delay (undefined delays are excluded), a single value corresponding to the 95th percentile, as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

The percentile = 95, meaning that the reported delay, "95Percentile", is the smallest value of Round-trip delay for which the Empirical Distribution Function (EDF), F(95Percentile) >= 95% of the singleton Round-trip delay values in the conditional distribution. See section 11.3 of [RFC2330] for the definition of the percentile statistic using the EDF.

LossRatio -- the count of lost packets to total packets sent is the basis for the loss ratio calculation as per Section 6.1 of [RFC6673].

#### 4.4.2. Reference Definition

For all outputs ---

T0 the start of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330].

Tf the end of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of

[RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330].

TotalPkts the count of packets sent by the Src to Dst during the measurement interval.

For

RTDelay_Active_IP-UDP-Periodic_RFCXXXXsec4_Seconds_95Percentile:

95Percentile The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns).

For

RTLoss_Active_IP-UDP-Periodic_RFCXXXXsec4_Percent_LossRatio:

Percentile The numeric value of the result is expressed in units of lost packets to total packets times 100%, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.0000000001.

#### 4.4.3. Metric Units

The 95th Percentile of Round-trip Delay is expressed in seconds.

The Round-trip Loss Ratio is expressed as a percentage of lost packets to total packets sent.

#### 4.4.4. Calibration

Section 3.7.3 of [RFC7679] provides a means to quantify the systematic and random errors of a time measurement. In-situ calibration could be enabled with an internal loopback at the Source host that includes as much of the measurement system as possible, performs address manipulation as needed, and provides some form of isolation (e.g., deterministic delay) to avoid send-receive interface contention. Some portion of the random and systematic error can be characterized this way.

When a measurement controller requests a calibration measurement, the loopback is applied and the result is output in the same format as a normal measurement with additional indication that it is a calibration result.

Both internal loopback calibration and clock synchronization can be used to estimate the available accuracy of the Output Metric Units. For example, repeated loopback delay measurements will reveal the portion of the Output result resolution which is the result of system noise, and thus inaccurate.

### 4.5. Administrative items

#### 4.5.1. Status

Current

#### 4.5.2. Requester

This RFC number

#### 4.5.3. Revision

1.0

#### 4.5.4. Revision Date

YYYY-MM-DD

### 4.6. Comments and Remarks

None.

## 5. Packet Delay Variation Registry Entry

This section gives an initial registry entry for a Packet Delay Variation metric.

### 5.1. Summary

This category includes multiple indexes to the registry entries, the element ID and metric name.

#### 5.1.1. ID (Identifier)

<insert numeric identifier, an integer>

#### 5.1.2. Name

OWPDV_Active_IP-UDP-Periodic_RFCXXXXsec5_Seconds_95Percentile

#### 5.1.3. URI

URL: https://www.iana.org/ ... <name>

#### 5.1.4. Description

An assessment of packet delay variation with respect to the minimum delay observed on the periodic stream, and the Output is expressed as the 95th percentile of the packet delay variation distribution.

#### 5.1.5. Change Controller

IETF

#### 5.1.6. Version (of Registry Format)

1.0

### 5.2. Metric Definition

This category includes columns to prompt the entry of all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters.

#### 5.2.1. Reference Definition

Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, May 1998. [RFC2330]

Demichelis, C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, November 2002. [RFC3393]

Morton, A. and B. Claise, "Packet Delay Variation Applicability Statement", RFC 5481, March 2009. [RFC5481]

Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010. [RFC5905]

See sections 2.4 and 3.4 of [RFC3393]. Singleton delay differences measured are referred to by the variable name "ddT" (applicable to all forms of delay variation). However, this metric entry specifies the PDV form defined in section 4.2 of [RFC5481], where the singleton PDV for packet i is referred to by the variable name "PDV(i)".

### 5.2.2. Fixed Parameters

- o IPv4 header values:
  - * DSCP: set to 0
  - * TTL: set to 255
  - * Protocol: set to 17 (UDP)
- o IPv6 header values:
  - * DSCP: set to 0
  - * Hop Count: set to 255
  - * Next Header: set to 17 (UDP)
  - * Flow Label: set to zero
  - * Extension Headers: none
- o UDP header values:
  - * Checksum: the checksum MUST be calculated and the non-zero checksum included in the header
- o UDP Payload
  - * total of 200 bytes

Other measurement parameters:

Tmax: a loss threshold waiting time with value 3.0, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 4 (see section 9.3 of [RFC6020]) and with resolution of 0.0001 seconds (0.1 ms), with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905].

F a selection function unambiguously defining the packets from the stream selected for the metric. See section 4.2 of [RFC5481] for the PDV form.

See the Packet Stream generation category for two additional Fixed Parameters.

## 5.3. Method of Measurement

This category includes columns for references to relevant sections of the RFC(s) and any supplemental information needed to ensure an unambiguous methods for implementations.

### 5.3.1. Reference Method

See section 2.6 and 3.6 of [RFC3393] for general singleton element calculations. This metric entry requires implementation of the PDV form defined in section 4.2 of [RFC5481]. Also see measurement considerations in section 8 of [RFC5481].

The reference method distinguishes between long-delayed packets and lost packets by implementing a maximum waiting time for packet arrival. Tmax is the waiting time used as the threshold to declare a packet lost. Lost packets SHALL be designated as having undefined delay.

The calculations on the one-way delay SHALL be performed on the conditional distribution, conditioned on successful packet arrival within Tmax. Also, when all packet delays are stored, the process which calculates the one-way delay value MUST enforce the Tmax threshold on stored values before calculations. See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

The reference method requires some way to distinguish between different packets in a stream to establish correspondence between sending times and receiving times for each successfully-arriving packet. Sequence numbers or other send-order identification MUST be retained at the Src or included with each packet to disambiguate packet reordering if it occurs.

If a standard measurement protocol is employed, then the measurement process will determine the sequence numbers or timestamps applied to test packets after the Fixed and Runtime parameters are passed to that process. The chosen measurement protocol will dictate the format of sequence numbers and time-stamps, if they are conveyed in the packet payload.

### 5.3.2. Packet Stream Generation

This section gives the details of the packet traffic which is the basis for measurement. In IPPM metrics, this is called the Stream, and can easily be described by providing the list of stream parameters.

Section 3 of [RFC3432] prescribes the method for generating Periodic streams using associated parameters.

incT the nominal duration of inter-packet interval, first bit to first bit, with value 0.0200, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 4 (see section 9.3 of [RFC6020]) and with resolution of 0.0001 seconds (0.1 ms).

dT the duration of the interval for allowed sample start times, with value 1.0, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 4 (see section 9.3 of [RFC6020]) and with resolution of 0.0001 seconds (0.1 ms).

NOTE: an initiation process with a number of control exchanges resulting in unpredictable start times (within a time interval) may be sufficient to avoid synchronization of periodic streams, and therefore a valid replacement for selecting a start time at random from a fixed interval.

The T0 parameter will be reported as a measured parameter. Parameters incT and dT are Fixed Parameters.

#### 5.3.3. Traffic Filtering (observation) Details

NA

#### 5.3.4. Sampling Distribution

NA

#### 5.3.5. Run-time Parameters and Data Format

Src the IP address of the host in the Src Role (format ipv4-address-no-zone value for IPv4, or ipv6-address-no-zone value for IPv6, see Section 4 of [RFC6991])

Dst the IP address of the host in the Dst Role (format ipv4-address-no-zone value for IPv4, or ipv6-address-no-zone value for IPv6, see section 4 of [RFC6991])

T0 a time, the start of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330]. When T0 is "all-zeros", a start time is unspecified and Tf is to be interpreted as the Duration of the measurement interval. The start time is controlled through other means.

Tf a time, the end of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330]. When T0 is "all-zeros", a end time date is ignored and Tf is interpreted as the Duration of the measurement interval.

#### 5.3.6. Roles

Src launches each packet and waits for return transmissions from Dst.

Dst waits for each packet from Src and sends a return packet to Src.

### 5.4. Output

This category specifies all details of the Output of measurements using the metric.

#### 5.4.1. Type

Percentile -- for the conditional distribution of all packets with a valid value of one-way delay (undefined delays are excluded), a single value corresponding to the 95th percentile, as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

The percentile = 95, meaning that the reported delay, "95Percentile", is the smallest value of one-way PDV for which the Empirical Distribution Function (EDF), F(95Percentile) >= 95% of the singleton one-way PDV values in the conditional distribution. See section 11.3 of [RFC2330] for the definition of the percentile statistic using the EDF.

#### 5.4.2. Reference Definition

T0 the start of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330].

Tf the end of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330].

95Percentile The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

#### 5.4.3. Metric Units

The 95th Percentile of one-way PDV is expressed in seconds.

#### 5.4.4. Calibration

Section 3.7.3 of [RFC7679] provides a means to quantify the systematic and random errors of a time measurement. In-situ calibration could be enabled with an internal loopback that includes as much of the measurement system as possible, performs address manipulation as needed, and provides some form of isolation (e.g., deterministic delay) to avoid send-receive interface contention. Some portion of the random and systematic error can be characterized this way.

For one-way delay measurements, the error calibration must include an assessment of the internal clock synchronization with its external reference (this internal clock is supplying timestamps for measurement). In practice, the time offsets [RFC5905] of clocks at both the source and destination are needed to estimate the systematic error due to imperfect clock synchronization (the time offsets are smoothed, thus the random variation is not usually represented in the results).

time_offset The time value of the result is expressed in units of seconds, as a signed value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

When a measurement controller requests a calibration measurement, the loopback is applied and the result is output in the same format as a normal measurement with additional indication that it is a calibration result. In any measurement, the measurement function SHOULD report its current estimate of time offset [RFC5905] as an indicator of the degree of synchronization.

Both internal loopback calibration and clock synchronization can be used to estimate the available accuracy of the Output Metric Units. For example, repeated loopback delay measurements will reveal the portion of the Output result resolution which is the result of system noise, and thus inaccurate.

### 5.5. Administrative items

#### 5.5.1. Status

Current

#### 5.5.2. Requester

This RFC number

#### 5.5.3. Revision

1.0

#### 5.5.4. Revision Date

YYYY-MM-DD

### 5.6. Comments and Remarks

Lost packets represent a challenge for delay variation metrics. See section 4.1 of [RFC3393] and the delay variation applicability statement[RFC5481] for extensive analysis and comparison of PDV and an alternate metric, IPDV.

## 6. DNS Response Latency and Loss Registry Entries

This section gives initial registry entries for DNS Response Latency and Loss from a network user's perspective, for a specific named resource. The metric can be measured repeatedly using different names. RFC 2681 [RFC2681] defines a Round-trip delay metric. We build on that metric by specifying several of the input parameters to precisely define two metrics for measuring DNS latency and loss.

Note to IANA: Each Registry "Name" below specifies a single registry entry, whose output format varies in accordance with the name.

All column entries beside the ID, Name, Description, and Output Reference Method categories are the same, thus this section proposes two closely-related registry entries. As a result, IANA is also asked to assign corresponding URLs to each Named Metric.

### 6.1. Summary

This category includes multiple indexes to the registry entries, the element ID and metric name.

#### 6.1.1. ID (Identifier)

<insert numeric identifier, an integer>

IANA is asked to assign different numeric identifiers to each of the two Named Metrics.

#### 6.1.2. Name

RTDNS_Active_IP-UDP-Poisson_RFCXXXXsec6_Seconds_Raw

RLDNS_Active_IP-UDP-Poisson_RFCXXXXsec6_Logical_Raw

#### 6.1.3. URI

URL: https://www.iana.org/ ... <name>

#### 6.1.4. Description

This is a metric for DNS Response performance from a network user's perspective, for a specific named resource. The metric can be measured repeatedly using different resource names.

RTDNS: This metric assesses the response time, the interval from the query transmission to the response.

RLDNS: This metric indicates that the response was deemed lost. In other words, the response time exceeded the maximum waiting time.

#### 6.1.5. Change Controller

IETF

#### 6.1.6. Version (of Registry Format)

1.0

### 6.2. Metric Definition

This category includes columns to prompt the entry of all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters.

#### 6.2.1. Reference Definition

Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987. (and updates)

[RFC1035]

Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, September 1999.

[RFC2681]

Section 2.4 of [RFC2681] provides the reference definition of the singleton (single value) Round-trip delay metric. Section 3.4 of [RFC2681] provides the reference definition expanded to cover a multi-singleton sample. Note that terms such as singleton and sample are defined in Section 11 of [RFC2330].

For DNS Response Latency, the entities in [RFC1035] must be mapped to [RFC2681]. The Local Host with its User Program and Resolver take the role of "Src", and the Foreign Name Server takes the role of "Dst".

Note that although the [RFC2681] definition of "Round-trip-Delay between Src and Dst at T" is directionally ambiguous in the text, this metric tightens the definition further to recognize that the host in the "Src" role will send the first packet to "Dst", and ultimately receive the corresponding return packet from "Dst" (when neither are lost).

Morton, A., "Round-trip Packet Loss Metrics", RFC 6673, August 2012.

[RFC6673]

Both response time and loss metrics employ a maximum waiting time for received responses, so the count of lost packets to total packets sent is the basis for the loss determination as per Section 4.3 of [RFC6673].

### 6.2.2. Fixed Parameters

Type-P as defined in Section 13 of [RFC2330]:

- o IPv4 header values:
  - * DSCP: set to 0
  - * TTL set to 255
  - * Protocol: set to 17 (UDP)
- o IPv6 header values:

* DSCP: set to 0

* Hop Count: set to 255

* Next Header: set to 17 (UDP)

* Flow Label: set to zero

* Extension Headers: none

o UDP header values:

* Source port: 53

* Destination port: 53

* Checksum: the checksum must be calculated and the non-zero checksum included in the header

o Payload: The payload contains a DNS message as defined in RFC 1035 [RFC1035] with the following values:

* The DNS header section contains:

+ Identification (see the Run-time column)

+ QR: set to 0 (Query)

+ OPCODE: set to 0 (standard query)

+ AA: not set

+ TC: not set

+ RD: set to one (recursion desired)

+ RA: not set

+ RCODE: not set

+ QDCOUNT: set to one (only one entry)

+ ANCOUNT: not set

+ NSCOUNT: not set

+ ARCOUNT: not set

   * The Question section contains:

      + QNAME: the Fully Qualified Domain Name (FQDN) provided as input for the test, see the Run-time column

      + QTYPE: the query type provided as input for the test, see the Run-time column

      + QCLASS: set to 1 for IN

   * The other sections do not contain any Resource Records.

Other measurement parameters:

- Tmax: a loss threshold waiting time (and to help disambiguate queries)

   * 5.0, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 4 (see section 9.3 of [RFC6020]) and with resolution of 0.0001 seconds (0.1 ms), with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905].

Observation: reply packets will contain a DNS response and may contain RRs.

### 6.3. Method of Measurement

This category includes columns for references to relevant sections of the RFC(s) and any supplemental information needed to ensure an unambiguous methods for implementations.

#### 6.3.1. Reference Method

The methodology for this metric is defined as Type-P-Round-trip-Delay-Poisson-Stream in section 2.6 of RFC 2681 [RFC2681] and section 3.6 of RFC 2681 [RFC2681] using the Type-P and Timeout defined under Fixed Parameters.

The reference method distinguishes between long-delayed packets and lost packets by implementing a maximum waiting time for packet arrival. Tmax is the waiting time used as the threshold to declare a response packet lost. Lost packets SHALL be designated as having undefined delay and counted for the RLDNS metric.

The calculations on the delay (RTT) SHALL be performed on the conditional distribution, conditioned on successful packet arrival within Tmax. Also, when all packet delays are stored, the process which calculates the RTT value MUST enforce the Tmax threshold on stored values before calculations. See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

The reference method requires some way to distinguish between different packets in a stream to establish correspondence between sending times and receiving times for each successfully-arriving reply.

DNS Messages bearing Queries provide for random ID Numbers in the Identification header field, so more than one query may be launched while a previous request is outstanding when the ID Number is used. Therefore, the ID Number MUST be retained at the Src and included with each response packet to disambiguate packet reordering if it occurs.

IF a DNS response does not arrive within Tmax, the response time RTDNS is undefined, and RLDNS = 1. The Message ID SHALL be used to disambiguate the successive queries that are otherwise identical.

Since the ID Number field is only 16 bits in length, it places a limit on the number of simultaneous outstanding DNS queries during a stress test from a single Src address.

Refer to Section 4.4 of [RFC6673] for expanded discussion of the instruction to "send a Type-P packet back to the Src as quickly as possible" in Section 2.6 of RFC 2681 [RFC2681]. However, the DNS Server is expected to perform all required functions to prepare and send a response, so the response time will include processing time and network delay. Section 8 of [RFC6673] presents additional requirements which SHALL be included in the method of measurement for this metric.

In addition to operations described in [RFC2681], the Src MUST parse the DNS headers of the reply and prepare the query response information for subsequent reporting as a measured result, along with the Round-Trip Delay.

### 6.3.2. Packet Stream Generation

This section gives the details of the packet traffic which is the basis for measurement. In IPPM metrics, this is called the Stream, and can easily be described by providing the list of stream parameters.

Section 11.1.3 of RFC 2681 [RFC2330] provides three methods to generate Poisson sampling intervals. The reciprocal of lambda is the average packet spacing, thus the Run-time Parameter is Reciprocal_lambda = 1/lambda, in seconds.

Method 3 is used, where given a start time (Run-time Parameter), the subsequent send times are all computed prior to measurement by computing the pseudo-random distribution of inter-packet send times, (truncating the distribution as specified in the Run-time Parameters), and the Src sends each packet at the computed times.

Note that Trunc is the upper limit on inter-packet times in the Poisson distribution. A random value greater than Trunc is set equal to Trunc instead.

#### 6.3.3. Traffic Filtering (observation) Details

NA

#### 6.3.4. Sampling Distribution

NA

#### 6.3.5. Run-time Parameters and Data Format

Run-time Parameters are input factors that must be determined, configured into the measurement system, and reported with the results for the context to be complete.

Src the IP address of the host in the Src Role (format ipv4-address-no-zone value for IPv4, or ipv6-address-no-zone value for IPv6, see Section 4 of [RFC6991])

Dst the IP address of the host in the Dst Role (format ipv4-address-no-zone value for IPv4, or ipv6-address-no-zone value for IPv6, see section 4 of [RFC6991])

T0 a time, the start of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330]. When T0 is "all-zeros", a start time is unspecified and Tf is to be interpreted as the Duration of the measurement interval. The start time is controlled through other means.

Tf a time, the end of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of

[RFC2330]. When T0 is "all-zeros", a end time date is ignored and Tf is interpreted as the Duration of the measurement interval.

Reciprocal_lambda average packet interval for Poisson Streams expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 4 (see section 9.3 of [RFC6020]) with resolution of 0.0001 seconds (0.1 ms), and with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905].

Trunc Upper limit on Poisson distribution expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 4 (see section 9.3 of [RFC6020]) with resolution of 0.0001 seconds (0.1 ms), and with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905] (values above this limit will be clipped and set to the limit value).

ID The 16-bit identifier assigned by the program that generates the query, and which must vary in successive queries (a list of IDs is needed), see Section 4.1.1 of [RFC1035]. This identifier is copied into the corresponding reply and can be used by the requester (Src) to match-up replies to outstanding queries.

QNAME The domain name of the Query, formatted as specified in section 4 of [RFC6991].

QTYPE The Query Type, which will correspond to the IP address family of the query (decimal 1 for IPv4 or 28 for IPv6, formatted as a uint16, as per section 9.2 of [RFC6020].

#### 6.3.6. Roles

Src launches each packet and waits for return transmissions from Dst.

Dst waits for each packet from Src and sends a return packet to Src.

### 6.4. Output

This category specifies all details of the Output of measurements using the metric.

#### 6.4.1. Type

Raw -- for each DNS Query packet sent, sets of values as defined in the next column, including the status of the response, only assigning delay values to successful query-response pairs.

### 6.4.2. Reference Definition

For all outputs:

T the time the DNS Query was sent during the measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330].

dT The time value of the round-trip delay to receive the DNS response, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]. This value is undefined when the response packet is not received at Src within waiting time Tmax seconds.

Rcode The value of the Rcode field in the DNS response header, expressed as a uint64 as specified in section 9.2 of [RFC6020]. Non-zero values convey errors in the response, and such replies must be analyzed separately from successful requests.

### 6.4.3. Metric Units

RTDNS: Round-trip Delay, dT, is expressed in seconds.

RTLDNS: the Logical value, where 1 = Lost and 0 = Received.

### 6.4.4. Calibration

Section 3.7.3 of [RFC7679] provides a means to quantify the systematic and random errors of a time measurement. In-situ calibration could be enabled with an internal loopback at the Source host that includes as much of the measurement system as possible, performs address and payload manipulation as needed, and provides some form of isolation (e.g., deterministic delay) to avoid send-receive interface contention. Some portion of the random and systematic error can be characterized this way.

When a measurement controller requests a calibration measurement, the loopback is applied and the result is output in the same format as a normal measurement with additional indication that it is a calibration result.

Both internal loopback calibration and clock synchronization can be used to estimate the available accuracy of the Output Metric Units. For example, repeated loopback delay measurements will reveal the portion of the Output result resolution which is the result of system noise, and thus inaccurate.

### 6.5. Administrative items

#### 6.5.1. Status

Current

#### 6.5.2. Requester

This RFC number

#### 6.5.3. Revision

1.0

#### 6.5.4. Revision Date

YYYY-MM-DD

### 6.6. Comments and Remarks

None

## 7. UDP Poisson One-way Delay and Loss Registry Entries

This section specifies five initial registry entries for the UDP Poisson One-way Delay, and one for UDP Poisson One-way Loss.

IANA Note: Registry "Name" below specifies multiple registry entries, whose output format varies according to the <statistic> element of the name that specifies one form of statistical summary. There is an additional metric name for the Loss metric.

All column entries beside the ID, Name, Description, and Output Reference Method categories are the same, thus this section proposes six closely-related registry entries. As a result, IANA is also asked to assign corresponding URLs to each Named Metric.

### 7.1. Summary

This category includes multiple indexes to the registry entries, the element ID and metric name.

### 7.1.1. ID (Identifier)

IANA is asked to assign different numeric identifiers to each of the six Metrics.

### 7.1.2. Name

OWDelay_Active_IP-UDP-Poisson-
Payload250B_RFCXXXXsec7_Seconds_<statistic>

where <statistic> is one of:

- o 95Percentile
- o Mean
- o Min
- o Max
- o StdDev

OWLoss_Active_IP-UDP-Poisson-
Payload250B_RFCXXXXsec7_Percent_LossRatio

### 7.1.3. URI

URL: https://www.iana.org/ ... <name>

### 7.1.4. Description

OWDelay: This metric assesses the delay of a stream of packets exchanged between two hosts (or measurement points), and reports the <statistic> One-way delay for all successfully exchanged packets based on their conditional delay distribution.

where <statistic> is one of:

- o 95Percentile
- o Mean
- o Min
- o Max
- o StdDev

OWLoss: This metric assesses the loss ratio of a stream of packets exchanged between two hosts (which are the two measurement points), and the Output is the One-way loss ratio for all successfully received packets expressed as a percentage.

## 7.2. Metric Definition

This category includes columns to prompt the entry of all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters.

### 7.2.1. Reference Definition

For Delay:

Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Delay Metric for IP Performance Metrics (IPPM)", STD 81, RFC 7679, DOI 10.17487/RFC7679, January 2016, <http://www.rfc-editor.org/info/rfc7679>.

[RFC7679]

Morton, A., and Stephan, E., "Spatial Composition of Metrics", RFC 6049, January 2011.

[RFC6049]

Section 3.4 of [RFC7679] provides the reference definition of the singleton (single value) One-way delay metric. Section 4.4 of [RFC7679] provides the reference definition expanded to cover a multi-value sample. Note that terms such as singleton and sample are defined in Section 11 of [RFC2330].

Only successful packet transfers with finite delay are included in the sample, as prescribed in section 4.1.2 of [RFC6049].

For loss:

Almes, G., Kalidini, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Loss Metric for IP Performance Metrics (IPPM)", RFC 7680, DOI 10.17487/RFC7680, January 2016, <http://www.rfc-editor.org/info/rfc7680>.

Section 2.4 of [RFC7680] provides the reference definition of the singleton (single value) one-way loss metric. Section 3.4 of [RFC7680] provides the reference definition expanded to cover a multi-singleton sample. Note that terms such as singleton and sample are defined in Section 11 of [RFC2330].

### 7.2.2. Fixed Parameters

Type-P:

- o IPv4 header values:
  * DSCP: set to 0
  * TTL: set to 255
  * Protocol: Set to 17 (UDP)
- o IPv6 header values:
  * DSCP: set to 0
  * Hop Count: set to 255
  * Next Header: set to 17 (UDP)
  * Flow Label: set to zero
  * Extension Headers: none
- o UDP header values:
  * Checksum: the checksum MUST be calculated and the non-zero checksum included in the header
- o UDP Payload: TWAMP Test Packet Formats, Section 4.1.2 of [RFC5357]
  * Security features in use influence the number of Padding octets.
  * 250 octets total, including the TWAMP format type, which MUST be reported.

Other measurement parameters:

Tmax: a loss threshold waiting time with value 3.0, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 4 (see section 9.3 of [RFC6020]) and with resolution of 0.0001 seconds (0.1 ms), with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905].

See the Packet Stream generation category for two additional Fixed Parameters.

## 7.3. Method of Measurement

This category includes columns for references to relevant sections of the RFC(s) and any supplemental information needed to ensure an unambiguous methods for implementations.

### 7.3.1. Reference Method

The methodology for this metric is defined as Type-P-One-way-Delay-Poisson-Stream in section 3.6 of [RFC7679] and section 4.6 of [RFC7679] using the Type-P and Tmax defined under Fixed Parameters.

The reference method distinguishes between long-delayed packets and lost packets by implementing a maximum waiting time for packet arrival. Tmax is the waiting time used as the threshold to declare a packet lost. Lost packets SHALL be designated as having undefined delay, and counted for the OWLoss metric.

The calculations on the one-way delay SHALL be performed on the conditional distribution, conditioned on successful packet arrival within Tmax. Also, when all packet delays are stored, the process which calculates the one-way delay value MUST enforce the Tmax threshold on stored values before calculations. See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

The reference method requires some way to distinguish between different packets in a stream to establish correspondence between sending times and receiving times for each successfully-arriving packet.

Since a standard measurement protocol is employed [RFC5357], then the measurement process will determine the sequence numbers or timestamps applied to test packets after the Fixed and Runtime parameters are passed to that process. The measurement protocol dictates the format of sequence numbers and time-stamps conveyed in the TWAMP-Test packet payload.

### 7.3.2. Packet Stream Generation

This section gives the details of the packet traffic which is the basis for measurement. In IPPM metrics, this is called the Stream, and can easily be described by providing the list of stream parameters.

Section 11.1.3 of RFC 2681 [RFC2330] provides three methods to generate Poisson sampling intervals. The reciprocal of lambda is the average packet spacing, thus the Run-time Parameter is Reciprocal_lambda = 1/lambda, in seconds.

Method 3 SHALL be used, where given a start time (Run-time Parameter), the subsequent send times are all computed prior to measurement by computing the pseudo-random distribution of inter-packet send times, (truncating the distribution as specified in the Parameter Trunc), and the Src sends each packet at the computed times.

Note that Trunc is the upper limit on inter-packet times in the Poisson distribution. A random value greater than Trunc is set equal to Trunc instead.

Reciprocal_lambda average packet interval for Poisson Streams expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 4 (see section 9.3 of [RFC6020]) with resolution of 0.0001 seconds (0.1 ms), and with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905]. Reciprocal_lambda = 1 second.

Trunc Upper limit on Poisson distribution expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 4 (see section 9.3 of [RFC6020]) with resolution of 0.0001 seconds (0.1 ms), and with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905] (values above this limit will be clipped and set to the limit value). Trunc = 30.0000 seconds.

#### 7.3.3. Traffic Filtering (observation) Details

NA

#### 7.3.4. Sampling Distribution

NA

#### 7.3.5. Run-time Parameters and Data Format

Run-time Parameters are input factors that must be determined, configured into the measurement system, and reported with the results for the context to be complete.

Src the IP address of the host in the Src Role (format ipv4-address-no-zone value for IPv4, or ipv6-address-no-zone value for IPv6, see Section 4 of [RFC6991])

Dst the IP address of the host in the Dst Role (format ipv4-address-no-zone value for IPv4, or ipv6-address-no-zone value for IPv6, see section 4 of [RFC6991])

T0 a time, the start of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330]. When T0 is "all-zeros", a start time is unspecified and Tf is to be interpreted as the Duration of the measurement interval. The start time is controlled through other means.

Tf a time, the end of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330]. When T0 is "all-zeros", a end time date is ignored and Tf is interpreted as the Duration of the measurement interval.

#### 7.3.6. Roles

Src launches each packet and waits for return transmissions from Dst. This is the TWAMP Session-Sender.

Dst waits for each packet from Src and sends a return packet to Src. This is the TWAMP Session-Reflector.

### 7.4. Output

This category specifies all details of the Output of measurements using the metric.

#### 7.4.1. Type

See subsection titles below for Types.

#### 7.4.2. Reference Definition

For all output types ---

T0 the start of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330].

Tf the end of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330].

For LossRatio -- the count of lost packets to total packets sent is the basis for the loss ratio calculation as per Section 4.1 of [RFC7680].

For each <statistic>, one of the following sub-sections apply:

#### 7.4.2.1. Percentile95

The 95th percentile SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3 of [RFC3393] for details on the percentile statistic (where Round-trip delay should be substituted for "ipdv").

The percentile = 95, meaning that the reported delay, "95Percentile", is the smallest value of one-way delay for which the Empirical Distribution Function (EDF), F(95Percentile) >= 95% of the singleton one-way delay values in the conditional distribution. See section 11.3 of [RFC2330] for the definition of the percentile statistic using the EDF.

95Percentile The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

#### 7.4.2.2. Mean

The mean SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.2.2 of [RFC6049] for details on calculating this statistic, and 4.2.3 of [RFC6049].

Mean The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

### 7.4.2.3. Min

The minimum SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3.2 of [RFC6049] for details on calculating this statistic, and 4.3.3 of [RFC6049].

Min The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

### 7.4.2.4. Max

The maximum SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3.2 of [RFC6049] for a closely related method for calculating this statistic, and 4.3.3 of [RFC6049]. The formula is as follows:

```
Max = (FiniteDelay [j])

      such that for some index, j, where 1 <= j <= N
      FiniteDelay[j] >= FiniteDelay[n] for all n
```

Max The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

#### 7.4.2.5. Std_Dev

The Std_Dev SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 6.1.4 of [RFC6049] for a closely related method for calculating this statistic. The formula is the classic calculation for standard deviation of a population.

Define Population Std_Dev_Delay as follows:
(where all packets n = 1 through N have a value for Delay[n], and MeanDelay calculated as in 7.4.2.2), and SQRT[] is the Square Root function:

$$\mathrm{Std\_Dev} = \mathrm{SQRT}\left[ \frac{1}{(N)} \sum_{n=1}^{N} (\mathrm{Delay}[n] - \mathrm{MeanDelay})^{2} \right]$$

Std_Dev The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

### 7.4.3. Metric Units

The <statistic> of One-way Delay is expressed in seconds.

The One-way Loss Ratio is expressed as a percentage of lost packets to total packets sent.

### 7.4.4. Calibration

Section 3.7.3 of [RFC7679] provides a means to quantify the systematic and random errors of a time measurement. In-situ calibration could be enabled with an internal loopback that includes as much of the measurement system as possible, performs address manipulation as needed, and provides some form of isolation (e.g., deterministic delay) to avoid send-receive interface contention. Some portion of the random and systematic error can be characterized this way.

For one-way delay measurements, the error calibration must include an assessment of the internal clock synchronization with its external reference (this internal clock is supplying timestamps for measurement). In practice, the time offsets [RFC5905] of clocks at both the source and destination are needed to estimate the systematic error due to imperfect clock synchronization (the time offsets [RFC5905] are smoothed, thus the random variation is not usually represented in the results).

time_offset The time value of the result is expressed in units of seconds, as a signed value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

When a measurement controller requests a calibration measurement, the loopback is applied and the result is output in the same format as a normal measurement with additional indication that it is a calibration result. In any measurement, the measurement function SHOULD report its current estimate of time offset [RFC5905] as an indicator of the degree of synchronization.

Both internal loopback calibration and clock synchronization can be used to estimate the available accuracy of the Output Metric Units. For example, repeated loopback delay measurements will reveal the portion of the Output result resolution which is the result of system noise, and thus inaccurate.

## 7.5. Administrative items

### 7.5.1. Status

Current

### 7.5.2. Requester

This RFC number

### 7.5.3. Revision

1.0

#### 7.5.4. Revision Date

YYYY-MM-DD

### 7.6. Comments and Remarks

None

## 8. UDP Periodic One-way Delay and Loss Registry Entries

This section specifies five initial registry entries for the UDP Periodic One-way Delay, and one for UDP Periodic One-way Loss.

IANA Note: Registry "Name" below specifies multiple registry entries, whose output format varies according to the <statistic> element of the name that specifies one form of statistical summary. There is an additional metric name for the Loss metric.

All column entries beside the ID, Name, Description, and Output Reference Method categories are the same, thus this section proposes six closely-related registry entries. As a result, IANA is also asked to assign corresponding URLs to each Named Metric.

### 8.1. Summary

This category includes multiple indexes to the registry entries, the element ID and metric name.

#### 8.1.1. ID (Identifier)

IANA is asked to assign a different numeric identifiers to each of the six Metrics.

#### 8.1.2. Name

OWDelay_Active_IP-UDP-Periodic20m-
Payload142B_RFCXXXXsec8_Seconds_<statistic>

where <statistic> is one of:

- o 95Percentile
- o Mean
- o Min
- o Max

o StdDev

OWLoss_Active_IP-UDP-Periodic-Payload142B_RFCXXXXsec8_Percent_LossRatio

### 8.1.3. URI

URL: https://www.iana.org/ ... <name>

### 8.1.4. Description

OWDelay: This metric assesses the delay of a stream of packets exchanged between two hosts (or measurement points), and reports the <statistic> One-way delay for all successfully exchanged packets based on their conditional delay distribution.

where <statistic> is one of:

- o 95Percentile
- o Mean
- o Min
- o Max
- o StdDev

OWLoss: This metric assesses the loss ratio of a stream of packets exchanged between two hosts (which are the two measurement points), and the Output is the One-way loss ratio for all successfully received packets expressed as a percentage.

## 8.2. Metric Definition

This category includes columns to prompt the entry of all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters.

### 8.2.1. Reference Definition

For Delay:

Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Delay Metric for IP Performance Metrics (IPPM)", STD 81, RFC 7679, DOI 10.17487/RFC7679, January 2016, <http://www.rfc-editor.org/info/rfc7679>.

[RFC7679]

Morton, A., and Stephan, E., "Spatial Composition of Metrics", RFC 6049, January 2011.

[RFC6049]

Section 3.4 of [RFC7679] provides the reference definition of the singleton (single value) One-way delay metric. Section 4.4 of [RFC7679] provides the reference definition expanded to cover a multi-value sample. Note that terms such as singleton and sample are defined in Section 11 of [RFC2330].

Only successful packet transfers with finite delay are included in the sample, as prescribed in section 4.1.2 of [RFC6049].

For loss:

Almes, G., Kalidini, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Loss Metric for IP Performance Metrics (IPPM)", RFC 7680, DOI 10.17487/RFC7680, January 2016, <http://www.rfc-editor.org/info/rfc7680>.

Section 2.4 of [RFC7680] provides the reference definition of the singleton (single value) one-way loss metric. Section 3.4 of [RFC7680] provides the reference definition expanded to cover a multi-singleton sample. Note that terms such as singleton and sample are defined in Section 11 of [RFC2330].

### 8.2.2. Fixed Parameters

Type-P:

- o IPv4 header values:
  - DSCP: set to 0
  - TTL: set to 255
  - Protocol: Set to 17 (UDP)
- o IPv6 header values:
  - DSCP: set to 0
  - Hop Count: set to 255
  - Next Header: set to 17 (UDP)

   * Flow Label: set to zero

   * Extension Headers: none

- UDP header values:

   * Checksum: the checksum MUST be calculated and the non-zero checksum included in the header

- UDP Payload: TWAMP Test Packet Formats, Section 4.1.2 of [RFC5357]

   * Security features in use influence the number of Padding octets.

   * 142 octets total, including the TWAMP format (and format type MUST be reported, if used)

Other measurement parameters:

Tmax: a loss threshold waiting time with value 3.0, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 4 (see section 9.3 of [RFC6020]) and with resolution of 0.0001 seconds (0.1 ms), with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905].

See the Packet Stream generation category for two additional Fixed Parameters.

### 8.3. Method of Measurement

This category includes columns for references to relevant sections of the RFC(s) and any supplemental information needed to ensure an unambiguous methods for implementations.

#### 8.3.1. Reference Method

The methodology for this metric is defined as Type-P-One-way-Delay-Poisson-Stream in section 3.6 of [RFC7679] and section 4.6 of [RFC7679] using the Type-P and Tmax defined under Fixed Parameters. However, a Periodic stream is used, as defined in [RFC3432].

The reference method distinguishes between long-delayed packets and lost packets by implementing a maximum waiting time for packet arrival. Tmax is the waiting time used as the threshold to declare a packet lost. Lost packets SHALL be designated as having undefined delay, and counted for the OWLoss metric.

The calculations on the one-way delay SHALL be performed on the conditional distribution, conditioned on successful packet arrival within Tmax. Also, when all packet delays are stored, the process which calculates the one-way delay value MUST enforce the Tmax threshold on stored values before calculations. See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

The reference method requires some way to distinguish between different packets in a stream to establish correspondence between sending times and receiving times for each successfully-arriving packet.

Since a standard measurement protocol is employed [RFC5357], then the measurement process will determine the sequence numbers or timestamps applied to test packets after the Fixed and Runtime parameters are passed to that process. The measurement protocol dictates the format of sequence numbers and time-stamps conveyed in the TWAMP-Test packet payload.

### 8.3.2. Packet Stream Generation

This section gives the details of the packet traffic which is the basis for measurement. In IPPM metrics, this is called the Stream, and can easily be described by providing the list of stream parameters.

Section 3 of [RFC3432] prescribes the method for generating Periodic streams using associated parameters.

incT the nominal duration of inter-packet interval, first bit to first bit, with value 0.0200 expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 4 (see section 9.3 of [RFC6020]) and with resolution of 0.0001 seconds (0.1 ms), with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905].

dT the duration of the interval for allowed sample start times, with value 1.0000, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 4 (see section 9.3 of [RFC6020]) and with resolution of 0.0001 seconds (0.1 ms), with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905].

T0 the actual start time of the periodic stream, determined from T0 and dT.

NOTE: an initiation process with a number of control exchanges resulting in unpredictable start times (within a time interval) may be sufficient to avoid synchronization of periodic streams, and therefore a valid replacement for selecting a start time at random from a fixed interval.

These stream parameters will be specified as Run-time parameters.

#### 8.3.3. Traffic Filtering (observation) Details

NA

#### 8.3.4. Sampling Distribution

NA

#### 8.3.5. Run-time Parameters and Data Format

Run-time Parameters are input factors that must be determined, configured into the measurement system, and reported with the results for the context to be complete.

Src the IP address of the host in the Src Role (format ipv4-address-no-zone value for IPv4, or ipv6-address-no-zone value for IPv6, see Section 4 of [RFC6991])

Dst the IP address of the host in the Dst Role (format ipv4-address-no-zone value for IPv4, or ipv6-address-no-zone value for IPv6, see section 4 of [RFC6991])

T0 a time, the start of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330]. When T0 is "all-zeros", a start time is unspecified and Tf is to be interpreted as the Duration of the measurement interval. The start time is controlled through other means.

Tf a time, the end of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330]. When T0 is "all-zeros", a end time date is ignored and Tf is interpreted as the Duration of the measurement interval.

#### 8.3.6. Roles

Src launches each packet and waits for return transmissions from Dst. This is the TWAMP Session-Sender.

Dst waits for each packet from Src and sends a return packet to Src. This is the TWAMP Session-Reflector.

### 8.4. Output

This category specifies all details of the Output of measurements using the metric.

#### 8.4.1. Type

See subsection titles in Reference Definition for Latency Types.

#### 8.4.2. Reference Definition

For all output types ---

T0 the start of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330].

Tf the end of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330].

For LossRatio -- the count of lost packets to total packets sent is the basis for the loss ratio calculation as per Section 4.1 of [RFC7680].

For each <statistic>, one of the following sub-sections apply:

##### 8.4.2.1. Percentile95

The 95th percentile SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3 of [RFC3393] for details on the percentile statistic (where Round-trip delay should be substituted for "ipdv").

The percentile = 95, meaning that the reported delay, "95Percentile", is the smallest value of one-way delay for which the Empirical Distribution Function (EDF), F(95Percentile) >= 95% of the singleton one-way delay values in the conditional distribution. See section 11.3 of [RFC2330] for the definition of the percentile statistic using the EDF.

95Percentile The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

#### 8.4.2.2. Mean

The mean SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.2.2 of [RFC6049] for details on calculating this statistic, and 4.2.3 of [RFC6049].

Mean The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

#### 8.4.2.3. Min

The minimum SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3.2 of [RFC6049] for details on calculating this statistic, and 4.3.3 of [RFC6049].

Min The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

#### 8.4.2.4. Max

The maximum SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3.2 of [RFC6049] for a closely related method for calculating this statistic, and 4.3.3 of [RFC6049]. The formula is as follows:

```
         Max = (FiniteDelay [j])

               such that for some index, j, where 1 <= j <= N
               FiniteDelay[j] >= FiniteDelay[n] for all n
```

Max The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

#### 8.4.2.5. Std_Dev

The Std_Dev SHALL be calculated using the conditional distribution of all packets with a finite value of One-way delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3.2 of [RFC6049] for a closely related method for calculating this statistic, and 4.3.3 of [RFC6049]. The formula is the classic calculation for standard deviation of a population.

Define Population Std_Dev_Delay as follows:
(where all packets n = 1 through N have a value for Delay[n], and MeanDelay calculated as in 7.4.2.2), and SQRT[] is the Square Root function:

$$\text{Std\_Dev} = \text{SQRT}\left[ \frac{1}{(N)} \sum_{n=1}^{N} (\text{Delay}[n] - \text{MeanDelay})^2 \right]$$

Std_Dev The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

### 8.4.3. Metric Units

The <statistic> of One-way Delay is expressed in seconds, where <statistic> is one of:

- o 95Percentile
- o Mean
- o Min
- o Max
- o StdDev

The One-way Loss Ratio is expressed as a percentage of lost packets to total packets sent.

### 8.4.4. Calibration

Section 3.7.3 of [RFC7679] provides a means to quantify the systematic and random errors of a time measurement. In-situ calibration could be enabled with an internal loopback that includes as much of the measurement system as possible, performs address manipulation as needed, and provides some form of isolation (e.g., deterministic delay) to avoid send-receive interface contention. Some portion of the random and systematic error can be characterized this way.

For one-way delay measurements, the error calibration must include an assessment of the internal clock synchronization with its external reference (this internal clock is supplying timestamps for measurement). In practice, the time offsets [RFC5905] of clocks at both the source and destination are needed to estimate the systematic error due to imperfect clock synchronization (the time offsets [RFC5905] are smoothed, thus the random variation is not usually represented in the results).

time_offset The time value of the result is expressed in units of seconds, as a signed value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

When a measurement controller requests a calibration measurement, the loopback is applied and the result is output in the same format as a normal measurement with additional indication that it is a calibration result. In any measurement, the measurement function SHOULD report its current estimate of time offset [RFC5905] as an indicator of the degree of synchronization.

Both internal loopback calibration and clock synchronization can be used to estimate the available accuracy of the Output Metric Units. For example, repeated loopback delay measurements will reveal the portion of the Output result resolution which is the result of system noise, and thus inaccurate.

## 8.5. Administrative items

### 8.5.1. Status

Current

### 8.5.2. Requester

This RFC number

### 8.5.3. Revision

1.0

### 8.5.4. Revision Date

YYYY-MM-DD

### 8.6. Comments and Remarks

None.

## 9. ICMP Round-trip Latency and Loss Registry Entries

This section specifies three initial registry entries for the ICMP Round-trip Latency, and another entry for ICMP Round-trip Loss Ratio.

IANA Note: Registry "Name" below specifies multiple registry entries, whose output format varies according to the <statistic> element of the name that specifies one form of statistical summary. There is an additional metric name for the Loss metric.

All column entries beside the ID, Name, Description, and Output Reference Method categories are the same, thus this section proposes two closely-related registry entries. As a result, IANA is also asked to assign corresponding URLs to each Named Metric.

### 9.1. Summary

This category includes multiple indexes to the registry entry: the element ID and metric name.

#### 9.1.1. ID (Identifier)

IANA is asked to assign different numeric identifiers to each of the four Named Metrics.

#### 9.1.2. Name

RTDelay_Active_IP-ICMP-SendOnRcv_RFCXXXXsec9_Seconds_<statistic>

where <statistic> is one of:

- Mean
- Min
- Max

RTLoss_Active_IP-ICMP-SendOnRcv_RFCXXXXsec9_Percent_LossRatio

#### 9.1.3. URI

URL: https://www.iana.org/ ... <name>

### 9.1.4. Description

RTDelay: This metric assesses the delay of a stream of ICMP packets exchanged between two hosts (which are the two measurement points), and the Output is the Round-trip delay for all successfully exchanged packets expressed as the <statistic> of their conditional delay distribution, where <statistic> is one of:

- o Mean
- o Min
- o Max

RTLoss: This metric assesses the loss ratio of a stream of ICMP packets exchanged between two hosts (which are the two measurement points), and the Output is the Round-trip loss ratio for all successfully exchanged packets expressed as a percentage.

### 9.1.5. Change Controller

IETF

### 9.1.6. Version (of Registry Format)

1.0

## 9.2. Metric Definition

This category includes columns to prompt the entry of all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters.

### 9.2.1. Reference Definition

Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, September 1999.

[RFC2681]

Section 2.4 of [RFC2681] provides the reference definition of the singleton (single value) Round-trip delay metric. Section 3.4 of [RFC2681] provides the reference definition expanded to cover a multi-singleton sample. Note that terms such as singleton and sample are defined in Section 11 of [RFC2330].

Note that although the [RFC2681] definition of "Round-trip-Delay between Src and Dst" is directionally ambiguous in the text, this metric tightens the definition further to recognize that the host in the "Src" role will send the first packet to "Dst", and ultimately receive the corresponding return packet from "Dst" (when neither are lost).

Finally, note that the variable "dT" is used in [RFC2681] to refer to the value of Round-trip delay in metric definitions and methods. The variable "dT" has been re-used in other IPPM literature to refer to different quantities, and cannot be used as a global variable name.

Morton, A., "Round-trip Packet Loss Metrics", RFC 6673, August 2012.

[RFC6673]

Both delay and loss metrics employ a maximum waiting time for received packets, so the count of lost packets to total packets sent is the basis for the loss ratio calculation as per Section 6.1 of [RFC6673].

### 9.2.2. Fixed Parameters

Type-P as defined in Section 13 of [RFC2330]:

- o IPv4 header values:
  - * DSCP: set to 0
  - * TTL: set to 255
  - * Protocol: Set to 01 (ICMP)
- o IPv6 header values:
  - * DSCP: set to 0
  - * Hop Count: set to 255
  - * Next Header: set to 128 decimal (ICMP)
  - * Flow Label: set to zero
  - * Extension Headers: none
- o ICMP header values:
  - * Type: 8 (Echo Request)
  - * Code: 0

   * Checksum: the checksum MUST be calculated and the non-zero checksum included in the header

   * (Identifier and Sequence Number set at Run-Time)

- o ICMP Payload

   * total of 32 bytes of random info, constant per test.

Other measurement parameters:

- o Tmax: a loss threshold waiting time

   * 3.0, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 4 (see section 9.3 of [RFC6020]) and with resolution of 0.0001 seconds (0.1 ms), with lossless conversion to/from the 32-bit NTP timestamp as per section 6 of [RFC5905].

## 9.3. Method of Measurement

This category includes columns for references to relevant sections of the RFC(s) and any supplemental information needed to ensure an unambiguous methods for implementations.

### 9.3.1. Reference Method

The methodology for this metric is defined as Type-P-Round-trip-Delay-Poisson-Stream in section 2.6 of RFC 2681 [RFC2681] and section 3.6 of RFC 2681 [RFC2681] using the Type-P and Tmax defined under Fixed Parameters.

The reference method distinguishes between long-delayed packets and lost packets by implementing a maximum waiting time for packet arrival. Tmax is the waiting time used as the threshold to declare a packet lost. Lost packets SHALL be designated as having undefined delay, and counted for the RTLoss metric.

The calculations on the delay (RTD) SHALL be performed on the conditional distribution, conditioned on successful packet arrival within Tmax. Also, when all packet delays are stored, the process which calculates the RTD value MUST enforce the Tmax threshold on stored values before calculations. See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

The reference method requires some way to distinguish between different packets in a stream to establish correspondence between sending times and receiving times for each successfully-arriving packet. Sequence numbers or other send-order identification MUST be retained at the Src or included with each packet to disambiguate packet reordering if it occurs.

The measurement process will determine the sequence numbers applied to test packets after the Fixed and Runtime parameters are passed to that process. The ICMP measurement process and protocol will dictate the format of sequence numbers and other identifiers.

Refer to Section 4.4 of [RFC6673] for expanded discussion of the instruction to "send a Type-P packet back to the Src as quickly as possible" in Section 2.6 of RFC 2681 [RFC2681]. Section 8 of [RFC6673] presents additional requirements which MUST be included in the method of measurement for this metric.

### 9.3.2. Packet Stream Generation

This section gives the details of the packet traffic which is the basis for measurement. In IPPM metrics, this is called the Stream, and can easily be described by providing the list of stream parameters.

The ICMP metrics use a sending discipline called "SendOnRcv" or Send On Receive. This is a modification of Section 3 of [RFC3432], which prescribes the method for generating Periodic streams using associated parameters as defined below for this description:

incT the nominal duration of inter-packet interval, first bit to first bit

dT the duration of the interval for allowed sample start times

The incT stream parameter will be specified as a Run-time parameter, and dT is not used in SendOnRcv.

A SendOnRcv sender behaves exactly like a Periodic stream generator while all reply packets arrive with RTD < incT, and the inter-packet interval will be constant.

If a reply packet arrives with RTD >= incT, then the inter-packet interval for the next sending time is nominally RTD.

If a reply packet fails to arrive within Tmax, then the inter-packet interval for the next sending time is nominally Tmax.

If an immediate send on reply arrival is desired, then set incT=0.

#### 9.3.3. Traffic Filtering (observation) Details

NA

#### 9.3.4. Sampling Distribution

NA

#### 9.3.5. Run-time Parameters and Data Format

Run-time Parameters are input factors that must be determined, configured into the measurement system, and reported with the results for the context to be complete.

Src the IP address of the host in the Src Role (format ipv4-address-no-zone value for IPv4, or ipv6-address-no-zone value for IPv6, see Section 4 of [RFC6991])

Dst the IP address of the host in the Dst Role (format ipv4-address-no-zone value for IPv4, or ipv6-address-no-zone value for IPv6, see section 4 of [RFC6991])

incT the nominal duration of inter-packet interval, first bit to first bit, expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 4 (see section 9.3 of [RFC6020]) and with resolution of 0.0001 seconds (0.1 ms).

T0 a time, the start of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330]. When T0 is "all-zeros", a start time is unspecified and Tf is to be interpreted as the Duration of the measurement interval. The start time is controlled through other means.

Count The total count of ICMP Echo Requests to send, formatted as a uint16, as per section 9.2 of [RFC6020].

(see the Packet Stream Generation section for additional Run-time parameters)

#### 9.3.6. Roles

Src launches each packet and waits for return transmissions from Dst.

Dst waits for each packet from Src and sends a return packet to Src.

### 9.4. Output

This category specifies all details of the Output of measurements using the metric.

#### 9.4.1. Type

See subsection titles in Reference Definition for Latency Types.

LossRatio -- the count of lost packets to total packets sent is the basis for the loss ratio calculation as per Section 6.1 of [RFC6673].

#### 9.4.2. Reference Definition

For all output types ---

T0 the start of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330].

Tf the end of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330].

TotalCount the count of packets actually sent by the Src to Dst during the measurement interval.

For LossRatio -- the count of lost packets to total packets sent is the basis for the loss ratio calculation as per Section 4.1 of [RFC7680].

For each <statistic>, one of the following sub-sections apply:

##### 9.4.2.1. Mean

The mean SHALL be calculated using the conditional distribution of all packets with a finite value of Round-trip delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.2.2 of [RFC6049] for details on calculating this statistic, and 4.2.3 of [RFC6049].

Mean The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

#### 9.4.2.2. Min

The minimum SHALL be calculated using the conditional distribution of all packets with a finite value of Round-trip delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3.2 of [RFC6049] for details on calculating this statistic, and 4.3.3 of [RFC6049].

Min The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

#### 9.4.2.3. Max

The maximum SHALL be calculated using the conditional distribution of all packets with a finite value of Round-trip delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3.2 of [RFC6049] for a closely related method for calculating this statistic, and 4.3.3 of [RFC6049]. The formula is as follows:

```
Max = (FiniteDelay [j])

        such that for some index, j, where 1 <= j <= N
        FiniteDelay[j] >= FiniteDelay[n] for all n
```

Max The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

#### 9.4.3. Metric Units

The <statistic> of Round-trip Delay is expressed in seconds, where <statistic> is one of:

- Mean
- Min
- Max

The Round-trip Loss Ratio is expressed as a percentage of lost packets to total packets sent.

#### 9.4.4. Calibration

Section 3.7.3 of [RFC7679] provides a means to quantify the systematic and random errors of a time measurement. In-situ calibration could be enabled with an internal loopback at the Source host that includes as much of the measurement system as possible, performs address manipulation as needed, and provides some form of isolation (e.g., deterministic delay) to avoid send-receive interface contention. Some portion of the random and systematic error can be characterized this way.

When a measurement controller requests a calibration measurement, the loopback is applied and the result is output in the same format as a normal measurement with additional indication that it is a calibration result.

Both internal loopback calibration and clock synchronization can be used to estimate the available accuracy of the Output Metric Units. For example, repeated loopback delay measurements will reveal the portion of the Output result resolution which is the result of system noise, and thus inaccurate.

### 9.5. Administrative items

#### 9.5.1. Status

Current

#### 9.5.2. Requester

This RFC number

#### 9.5.3. Revision

1.0

#### 9.5.4. Revision Date

YYYY-MM-DD

### 9.6. Comments and Remarks

None

## 10. TCP Round-Trip Delay and Loss Registry Entries

This section specifies three initial registry entries for the Passive assessment of TCP Round-Trip Delay (RTD) and another entry for TCP Round-trip Loss Count.

IANA Note: Registry "Name" below specifies multiple registry entries, whose output format varies according to the <statistic> element of the name that specifies one form of statistical summary. There are two additional metric names for Singleton RT Delay and Packet Count metrics.

All column entries beside the ID, Name, Description, and Output Reference Method categories are the same, thus this section proposes four closely-related registry entries. As a result, IANA is also asked to assign corresponding URLs to each Named Metric.

### 10.1. Summary

This category includes multiple indexes to the registry entry: the element ID and metric name.

#### 10.1.1. ID (Identifier)

IANA is asked to assign different numeric identifiers to each of the four Named Metrics.

#### 10.1.2. Name

RTDelay_Passive_IP-TCP_RFCXXXXsec10_Seconds_<statistic>

where <statistic> is one of:

- o Mean
- o Min
- o Max

RTDelay_Passive_IP-TCP-HS_RFCXXXXsec10_Seconds_Singleton

Note that a mid-point observer only has the opportunity to compose a single RTDelay on the TCP Hand Shake.

RTLoss_Passive_IP-TCP_RFCXXXXsec10_Packet_Count

### 10.1.3. URI

URL: https://www.iana.org/ ... <name>

### 10.1.4. Description

RTDelay: This metric assesses the round-trip delay of TCP packets constituting a single connection, exchanged between two hosts. We consider the measurement of round-trip delay based on a single Observation Point [RFC7011] somewhere in the network. The Output is the Round-trip delay for all successfully exchanged packets expressed as the <statistic> of their conditional delay distribution, where <statistic> is one of:

- o Mean
- o Min
- o Max

RTLoss: This metric assesses the estimated loss count for TCP packets constituting a single connection, exchanged between two hosts. We consider the measurement of round-trip delay based on a single Observation Point [RFC7011] somewhere in the network. The Output is the estimated Loss Count for the measurement interval.

### 10.1.5. Change Controller

IETF

### 10.1.6. Version (of Registry Format)

1.0

### 10.2. Metric Definition

This category includes columns to prompt the entry of all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters.

#### 10.2.1. Reference Definitions

Although there is no RFC that describes passive measurement of Round-Trip Delay, the parallel definition for Active measurement is:

Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, September 1999.

[RFC2681]

This metric definition uses the terms singleton and sample as defined in Section 11 of [RFC2330]. (Section 2.4 of [RFC2681] provides the reference definition of the singleton (single value) Round-trip delay metric. Section 3.4 of [RFC2681] provides the reference definition expanded to cover a multi-singleton sample.)

With the Observation Point [RFC7011] (OP) typically located between the hosts participating in the TCP connection, the Round-trip Delay metric requires two individual measurements between the OP and each host, such that the Spatial Composition [RFC6049]of the measurements yields a Round-trip Delay singleton (we are extending the composition of one-way subpath delays to subpath round-trip delay).

Using the direction of TCP SYN transmission to anchor the nomenclature, host A sends the SYN and host B replies with SYN-ACK during connection establishment. The direction of SYN transfer is considered the Forward direction of transmission, from A through OP to B (Reverse is B through OP to A).

Traffic filters reduce the packet stream at the OP to a Qualified bidirectional flow of packets.

In the definitions below, Corresponding Packets are transferred in different directions and convey a common value in a TCP header field that establishes correspondence (to the extent possible). Examples may be found in the TCP timestamp fields.

For a real number, RTD_fwd, >> the Round-trip Delay in the Forward direction from OP to host B at time T' is RTD_fwd << it is REQUIRED that OP observed a Qualified Packet to host B at wire-time T', that host B received that packet and sent a Corresponding Packet back to host A, and OP observed the Corresponding Packet at wire-time T' + RTD_fwd.

For a real number, RTD_rev, >> the Round-trip Delay in the Reverse direction from OP to host A at time T'' is RTD_rev << it is REQUIRED that OP observed a Qualified Packet to host A at wire-time T'', that host A received that packet and sent a Corresponding Packet back to host B, and that OP observed the Corresponding Packet at wire-time T'' + RTD_rev.

Ideally, the packet sent from host B to host A in both definitions above SHOULD be the same packet (or, when measuring RTD_rev first, the packet from host A to host B in both definitions should be the same).

The REQUIRED Composition Function for a singleton of Round-trip Delay at time T (where T is the earliest of T' and T'' above) is:

RTDelay = RTD_fwd + RTD_rev

Note that when OP is located at host A or host B, one of the terms composing RTDelay will be zero or negligible.

When the Qualified and Corresponding Packets are a TCP-SYN and a TCP-SYN-ACK, then RTD_fwd == RTD_HS_fwd.

When the Qualified and Corresponding Packets are a TCP-SYN-ACK and a TCP-ACK, then RTD_rev == RTD_HS_rev.

The REQUIRED Composition Function for a singleton of Round-trip Delay for the connection Hand Shake:

RTDelay_HS = RTD_HS_fwd + RTD_HS_rev

The definition of Round-trip Loss Count uses the nomenclature developed above, based on observation of the TCP header sequence numbers and storing the sequence number gaps observed. Packet Losses can be inferred from:

o Out-of-order segments: TCP segments are transmitted with monotonically increasing sequence numbers, but these segments may be received out of order. Section 3 of [RFC4737] describes the notion of "next expected" sequence numbers which can be adapted to TCP segments (for the purpose of detecting reordered packets). Observation of out-of-order segments indicates loss on the path prior to the OP, and creates a gap.

o Duplicate segments: Section 2 of [RFC5560] defines identical packets and is suitable for evaluation of TCP packets to detect duplication. Observation of duplicate segments *without a corresponding gap* indicates loss on the path following the OP (because they overlap part of the delivered sequence numbers already observed at OP).

Each observation of an out-of-order or duplicate infers a singleton of loss, but composition of Round-trip Loss Counts will be conducted over a measurement interval which is synonymous with a single TCP connection.

With the above observations in the Forward direction over a measurement interval, the count of out-of-order and duplicate segments is defined as RTL_fwd. Comparable observations in the Reverse direction are defined as RTL_rev.

For a measurement interval (corresponding to a single TCP connection), T0 to Tf, the REQUIRED Composition Function for a the two single-direction counts of inferred loss is:

RTLoss = RTL_fwd + RTL_rev

### 10.2.2. Fixed Parameters

Traffic Filters:

- o IPv4 header values:
  - * DSCP: set to 0
  - * Protocol: Set to 06 (TCP)
- o IPv6 header values:
  - * DSCP: set to 0
  - * Hop Count: set to 255
  - * Next Header: set to 6 (TCP)
  - * Flow Label: set to zero
  - * Extension Headers: none
- o TCP header values:
  - * Flags: ACK, SYN, FIN, set as required

* Timestamp Option (TSopt): Set

  + Section 3.2 of [RFC7323]

### 10.3. Method of Measurement

This category includes columns for references to relevant sections of the RFC(s) and any supplemental information needed to ensure an unambiguous methods for implementations.

#### 10.3.1. Reference Methods

The foundation methodology for this metric is defined in Section 4 of [RFC7323] using the Timestamp Option with modifications that allow application at a mid-path Observation Point (OP) [RFC7011]. Further details and applicable heuristics were derived from [Strowes] and [Trammell-14].

The Traffic Filter at the OP is configured to observe a single TCP connection. When the SYN, SYN-ACK, ACK handshake occurs, it offers the first opportunity to measure both RTD_fwd (on the SYN to SYN-ACK pair) and RTD_rev (on the SYN-ACK to ACK pair). Label this singleton of RTDelay as RTDelay_HS (composed using the forward and reverse measurement pair). RTDelay_HS SHALL be treated separately from other RTDelays on data-bearing packets and their ACKs. The RTDelay_HS value MAY be used as a sanity check on other Composed values of RTDelay.

For payload bearing packets, the OP measures the time interval between observation of a packet with Sequence Number s, and the corresponding ACK with same Sequence number. When the payload is transferred from host A to host B, the observed interval is RTD_fwd.

Because many data transfers are unidirectional (say, in the Forward direction from host A to host B), it is necessary to use pure ACK packets with Timestamp (TSval) and their Timestamp value echo to perform a RTD_rev measurement. The time interval between observation of the ACK from B to A, and the corresponding packet with Timestamp echo (TSecr) is the RTD_rev.

Delay Measurement Filtering Heuristics:

If Data payloads were transferred in both Forward and Reverse directions, then the Round-Trip Time Measurement Rule in Section 4.1 of [RFC7323] could be applied. This rule essentially excludes any measurement using a packet unless it makes progress in the transfer (advances the left edge of the send window, consistent with [Strowes]).

A different heuristic from [Trammell-14] is to exclude any RTD_rev that is larger than previously observed values. This would tend to exclude Reverse measurements taken when the Application has no data ready to send, because considerable time could be added to RTD_rev from this source of error.

Note that the above Heuristic assumes that host A is sending data. Host A expecting a download would mean that this heuristic should be applied to RTD_fwd.

The statistic calculations to summarize the delay (RTDelay) SHALL be performed on the conditional distribution, conditioned on successful Forward and Reverse measurements which follow the Heuristics.

Method for Inferring Loss:

The OP tracks sequence numbers and stores gaps for each direction of transmission, as well as the next-expected sequence number as in [Trammell-14] and [RFC4737]. Loss is inferred from Out-of-order segments and Duplicate segments.

Loss Measurement Filtering Heuristics:

[Trammell-14] adds a window of evaluation based on the RTDelay.

Distinguish Re-ordered from OOO due to loss, because sequence number gap is filled during the same RTDelay window. Segments detected as re-ordered according to [RFC4737] MUST reduce the Loss Count inferred from Out-of-order segments.

Spurious (unneeded) retransmissions (observed as duplicates) can also be reduced this way, as described in [Trammell-14].

Sources of Error:

The principal source of RTDelay error is the host processing time to return a packet that defines the termination of a time interval. The heuristics above intend to mitigate these errors by excluding measurements where host processing time is a significant part of RTD_fwd or RTD_rev.

A key source of RTLoss error is observation loss, described in section 3 of [Trammell-14].

### 10.3.2. Packet Stream Generation

NA

### 10.3.3. Traffic Filtering (observation) Details

The Fixed Parameters above give a portion of the Traffic Filter. Other aspects will be supplied as Run-time Parameters (below).

### 10.3.4. Sampling Distribution

This metric requires a complete sample of all packets that qualify according to the Traffic Filter criteria.

### 10.3.5. Run-time Parameters and Data Format

Run-time Parameters are input factors that must be determined, configured into the measurement system, and reported with the results for the context to be complete.

Src the IP address of the host in the host A Role (format ipv4-address-no-zone value for IPv4, or ipv6-address-no-zone value for IPv6, see Section 4 of [RFC6991])

Dst the IP address of the host in the host B (format ipv4-address-no-zone value for IPv4, or ipv6-address-no-zone value for IPv6, see section 4 of [RFC6991])

T0 a time, the start of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330]. When T0 is "all-zeros", a start time is unspecified and Td is to be interpreted as the Duration of the measurement interval. The start time is controlled through other means.

Td Optionally, the end of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]), or the duration (see T0). The UTC Time Zone is required by Section 6.1 of [RFC2330]. Alternatively, the end of the measurement interval MAY be controlled by the measured connection, where the second pair of FIN and ACK packets exchanged between host A and B effectively ends the interval.

TTL or Hop Limit Set at desired value.

### 10.3.6. Roles

host A launches the SYN packet to open the connection, and synonymous with an IP address.

host B replies with the SYN-ACK packet to open the connection, and synonymous with an IP address.

## 10.4. Output

This category specifies all details of the Output of measurements using the metric.

### 10.4.1. Type

See subsection titles in Reference Definition for RTDelay Types.

For RTLoss -- the count of lost packets.

### 10.4.2. Reference Definition

For all output types ---

T0 the start of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330].

Tf the end of a measurement interval, (format "date-and-time" as specified in Section 5.6 of [RFC3339], see also Section 3 of [RFC6991]). The UTC Time Zone is required by Section 6.1 of [RFC2330]. The end of the measurement interval MAY be controlled by the measured connection, where the second pair of FIN and ACK packets exchanged between host A and B effectively ends the interval.

... ...

For RTDelay_HS -- the Round trip delay of the Handshake.

For RTLoss -- the count of lost packets.

For each <statistic>, one of the following sub-sections apply:

#### 10.4.2.1. Mean

The mean SHALL be calculated using the conditional distribution of all packets with a finite value of Round-trip delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.2.2 of [RFC6049] for details on calculating this statistic, and 4.2.3 of [RFC6049].

Mean The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

#### 10.4.2.2. Min

The minimum SHALL be calculated using the conditional distribution of all packets with a finite value of Round-trip delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3.2 of [RFC6049] for details on calculating this statistic, and 4.3.3 of [RFC6049].

Min The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

#### 10.4.2.3. Max

The maximum SHALL be calculated using the conditional distribution of all packets with a finite value of Round-trip delay (undefined delays are excluded), a single value as follows:

See section 4.1 of [RFC3393] for details on the conditional distribution to exclude undefined values of delay, and Section 5 of [RFC6703] for background on this analysis choice.

See section 4.3.2 of [RFC6049] for a closely related method for calculating this statistic, and 4.3.3 of [RFC6049]. The formula is as follows:

```
Max = (FiniteDelay [j])

      such that for some index, j, where 1 <= j <= N
      FiniteDelay[j] >= FiniteDelay[n] for all n
```

Max The time value of the result is expressed in units of seconds, as a positive value of type decimal64 with fraction digits = 9 (see section 9.3 of [RFC6020]) with resolution of 0.000000001 seconds (1.0 ns), and with lossless conversion to/from the 64-bit NTP timestamp as per section 6 of RFC [RFC5905]

#### 10.4.3. Metric Units

The <statistic> of Round-trip Delay is expressed in seconds, where <statistic> is one of:

- o Mean
- o Min
- o Max

The Round-trip Delay of the Hand Shake is expressed in seconds.

The Round-trip Loss Count is expressed as a number of packets.

#### 10.4.4. Calibration

Passive measurements at an OP could be calibrated against an active measurement (with loss emulation) at host A or B, where the active measurement represents the ground-truth.

### 10.5. Administrative items

#### 10.5.1. Status

Current

#### 10.5.2. Requester

This RFC number

#### 10.5.3. Revision

1.0

#### 10.5.4. Revision Date

YYYY-MM-DD

### 10.6. Comments and Remarks

None.

## 11. Security Considerations

These registry entries represent no known implications for Internet Security. Each RFC referenced above contains a Security Considerations section. Further, the LMAP Framework [RFC7594] provides both security and privacy considerations for measurements.

There are potential privacy considerations for observed traffic, particularly for passive metrics in section 10. An attacker that knows that its TCP connection is being measured can modify its behavior to skew the measurement results.

## 12. IANA Considerations

IANA is requested to populate The Performance Metrics Registry defined in [I-D.ietf-ippm-metric-registry] with the values defined in sections 4 through 10.

See the IANA Considerations section of [I-D.ietf-ippm-metric-registry] for additional requests and considerations.

## 13. Acknowledgements

The authors thank Brian Trammell for suggesting the term "Run-time Parameters", which led to the distinction between run-time and fixed parameters implemented in this memo, for identifying the IPFIX metric with Flow Key as an example, for suggesting the Passive TCP RTD metric and supporting references, and for many other productive suggestions. Thanks to Peter Koch, who provided several useful suggestions for disambiguating successive DNS Queries in the DNS Response time metric.

The authors also acknowledge the constructive reviews and helpful suggestions from Barbara Stark, Juergen Schoenwaelder, Tim Carey, Yaakov Stein, and participants in the LMAP working group. Thanks to

Michelle Cotton for her early IANA reviews, and to Amanda Barber for answering questions related to the presentation of the registry and accessibility of the complete template via URL.

## 14. References

### 14.1. Normative References

[I-D.ietf-ippm-metric-registry]
Bagnulo, M., Claise, B., Eardley, P., and A. Morton, "Registry for Performance Metrics", Internet Draft (work in progress) draft-ietf-ippm-metric-registry, 2019.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[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>.

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, DOI 10.17487/RFC2330, May 1998, <https://www.rfc-editor.org/info/rfc2330>.

[RFC2681] Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, DOI 10.17487/RFC2681, September 1999, <https://www.rfc-editor.org/info/rfc2681>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC3393] Demichelis, C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, DOI 10.17487/RFC3393, November 2002, <https://www.rfc-editor.org/info/rfc3393>.

[RFC3432] Raisanen, V., Grotefeld, G., and A. Morton, "Network performance measurement with periodic streams", RFC 3432, DOI 10.17487/RFC3432, November 2002, <https://www.rfc-editor.org/info/rfc3432>.

[RFC4737] Morton, A., Ciavattone, L., Ramachandran, G., Shalunov, S., and J. Perser, "Packet Reordering Metrics", RFC 4737, DOI 10.17487/RFC4737, November 2006, <https://www.rfc-editor.org/info/rfc4737>.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <https://www.rfc-editor.org/info/rfc5357>.

[RFC5481] Morton, A. and B. Claise, "Packet Delay Variation Applicability Statement", RFC 5481, DOI 10.17487/RFC5481, March 2009, <https://www.rfc-editor.org/info/rfc5481>.

[RFC5560] Uijterwaal, H., "A One-Way Packet Duplication Metric", RFC 5560, DOI 10.17487/RFC5560, May 2009, <https://www.rfc-editor.org/info/rfc5560>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6049] Morton, A. and E. Stephan, "Spatial Composition of Metrics", RFC 6049, DOI 10.17487/RFC6049, January 2011, <https://www.rfc-editor.org/info/rfc6049>.

[RFC6673] Morton, A., "Round-Trip Packet Loss Metrics", RFC 6673, DOI 10.17487/RFC6673, August 2012, <https://www.rfc-editor.org/info/rfc6673>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <https://www.rfc-editor.org/info/rfc7011>.

[RFC7323] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extensions for High Performance", RFC 7323, DOI 10.17487/RFC7323, September 2014, <https://www.rfc-editor.org/info/rfc7323>.

[RFC7679] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Delay Metric for IP Performance Metrics (IPPM)", STD 81, RFC 7679, DOI 10.17487/RFC7679, January 2016, <https://www.rfc-editor.org/info/rfc7679>.

[RFC7680] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Loss Metric for IP Performance Metrics (IPPM)", STD 82, RFC 7680, DOI 10.17487/RFC7680, January 2016, <https://www.rfc-editor.org/info/rfc7680>.

[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>.

[Strowes] Strowes, S., "Passively Measuring TCP Round Trip Times, Communications of the ACM, Vol. 56 No. 10, Pages 57-64", September 2013.

[Trammell-14] Trammell, B., "Inline Data Integrity Signals for Passive Measurement, In: Dainotti A., Mahanti A., Uhlig S. (eds) Traffic Monitoring and Analysis. TMA 2014. Lecture Notes in Computer Science, vol 8406. Springer, Berlin, Heidelberg https://link.springer.com/chapter/10.1007/978-3-642-54999-1_2", March 2014.

## 14.2. Informative 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>.

[RFC6390] Clark, A. and B. Claise, "Guidelines for Considering New Performance Metric Development", BCP 170, RFC 6390, DOI 10.17487/RFC6390, October 2011, <https://www.rfc-editor.org/info/rfc6390>.

[RFC6703] Morton, A., Ramachandran, G., and G. Maguluri, "Reporting IP Network Performance Metrics: Different Points of View", RFC 6703, DOI 10.17487/RFC6703, August 2012, <https://www.rfc-editor.org/info/rfc6703>.

[RFC7594] Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A Framework for Large-Scale Measurement of Broadband Performance (LMAP)", RFC 7594, DOI 10.17487/RFC7594, September 2015, <https://www.rfc-editor.org/info/rfc7594>.

Authors' Addresses

Al Morton
AT&T Labs
200 Laurel Avenue South
Middletown,, NJ 07748
USA

Phone: +1 732 420 1571
Fax: +1 732 368 1192
Email: acmorton@att.com

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
SPAIN

Phone: 34 91 6249500
Email: marcelo@it.uc3m.es
URI: http://www.it.uc3m.es

Philip Eardley
BT
Adastral Park, Martlesham Heath
Ipswich
ENGLAND

Email: philip.eardley@bt.com

Kevin D'Souza
AT&T Labs
200 Laurel Avenue South
Middletown,, NJ 07748
USA

Phone: +1 732 420 xxxx
Email: kld@att.com

Network Working Group M. Bagnulo
Internet-Draft UC3M
Intended status: Standards Track B. Claise
Expires: September 10, 2020 Cisco Systems, Inc.
P. Eardley
BT
A. Morton
AT&T Labs
A. Akhter
Consultant
March 9, 2020

# Registry for Performance Metrics
draft-ietf-ippm-metric-registry-24

Abstract

This document defines the format for the IANA Performance Metrics Registry. This document also gives a set of guidelines for Registered Performance Metric requesters and reviewers.

Table of Contents

## 1. Introduction

The IETF specifies and uses Performance Metrics of protocols and applications transported over its protocols. Performance metrics are important part of network operations using IETF protocols, and [RFC6390] specifies guidelines for their development.

The definition and use of Performance Metrics in the IETF has been fostered in various working groups (WG), most notably:

The "IP Performance Metrics" (IPPM) WG is the WG primarily focusing on Performance Metrics definition at the IETF.

The "Benchmarking Methodology" WG (BMWG) defines many Performance Metrics for use in laboratory benchmarking of inter-networking technologies.

The "Metric Blocks for use with RTCP's Extended Report Framework" (XRBLOCK) WG (concluded) specified many Performance Metrics related to "RTP Control Protocol Extended Reports (RTCP XR)" [RFC3611], which establishes a framework to allow new information to be conveyed in RTCP, supplementing the original report blocks defined in "RTP: A Transport Protocol for Real-Time Applications", [RFC3550].

The "IP Flow Information eXport" (IPFIX) concluded WG specified an IANA process for new Information Elements. Some Performance Metrics related Information Elements are proposed on regular basis.

The "Performance Metrics for Other Layers" (PMOL) a concluded WG defined some Performance Metrics related to Session Initiation Protocol (SIP) voice quality [RFC6035].

It is expected that more Performance Metrics will be defined in the future, not only IP-based metrics, but also metrics which are protocol-specific and application-specific.

Despite the importance of Performance Metrics, there are two related problems for the industry. First, ensuring that when one party requests another party to measure (or report or in some way act on) a particular Performance Metric, then both parties have exactly the same understanding of what Performance Metric is being referred to. Second, discovering which Performance Metrics have been specified, to avoid developing a new Performance Metric that is very similar, but not quite inter-operable. These problems can be addressed by creating a registry of performance metrics. The usual way in which the IETF organizes registries is with Internet Assigned Numbers

Authority (IANA), and there is currently no Performance Metrics Registry maintained by the IANA.

This document requests that IANA create and maintain a Performance Metrics Registry, according to the maintenance procedures and the Performance Metrics Registry format defined in this memo. The resulting Performance Metrics Registry is for use by the IETF and others. Although the Registry formatting specifications herein are primarily for registry creation by IANA, any other organization that wishes to create a performance metrics registry may use the same formatting specifications for their purposes. The authors make no guarantee of the registry format's applicability to any possible set of Performance Metrics envisaged by other organizations, but encourage others to apply it. In the remainder of this document, unless we explicitly say otherwise, we will refer to the IANA-maintained Performance Metrics Registry as simply the Performance Metrics Registry.

## 2. Terminology

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.

Performance Metric: A Performance Metric is a quantitative measure of performance, targeted to an IETF-specified protocol or targeted to an application transported over an IETF-specified protocol. Examples of Performance Metrics are the FTP response time for a complete file download, the DNS response time to resolve the IP address(es), a database logging time, etc. This definition is consistent with the definition of metric in [RFC2330] and broader than the definition of performance metric in [RFC6390].

Registered Performance Metric: A Registered Performance Metric is a Performance Metric expressed as an entry in the Performance Metrics Registry, administered by IANA. Such a performance metric has met all the registry review criteria defined in this document in order to be included in the registry.

Performance Metrics Registry: The IANA registry containing Registered Performance Metrics.

Proprietary Registry: A set of metrics that are registered in a proprietary registry, as opposed to Performance Metrics Registry.

Performance Metrics Experts: The Performance Metrics Experts is a group of designated experts [RFC8126] selected by the IESG to validate the Performance Metrics before updating the Performance Metrics Registry. The Performance Metrics Experts work closely with IANA.

Parameter: A Parameter is an input factor defined as a variable in the definition of a Performance Metric. A Parameter is a numerical or other specified factor forming one of a set that defines a metric or sets the conditions of its operation. All Parameters must be known in order to make a measurement using a metric and interpret the results. There are two types of Parameters: Fixed and Run-time parameters. For the Fixed Parameters, the value of the variable is specified in the Performance Metrics Registry entry and different Fixed Parameter values results in different Registered Performance Metrics. For the Run-time Parameters, the value of the variable is defined when the metric measurement method is executed and a given Registered Performance Metric supports multiple values for the parameter. Although Run-time Parameters do not change the fundamental nature of the Performance Metric's definition, some have substantial influence on the network property being assessed and interpretation of the results.

> Note: Consider the case of packet loss in the following two Active Measurement Method cases. The first case is packet loss as background loss where the Run-time Parameter set includes a very sparse Poisson stream, and only characterizes the times when packets were lost. Actual user streams likely see much higher loss at these times, due to tail drop or radio errors. The second case is packet loss as inverse of throughput where the Run-time Parameter set includes a very dense, bursty stream, and characterizes the loss experienced by a stream that approximates a user stream. These are both "loss metrics", but the difference in interpretation of the results is highly dependent on the Run-time Parameters (at least), to the extreme where we are actually using loss to infer its compliment: delivered throughput.

Active Measurement Method: Methods of Measurement conducted on traffic which serves only the purpose of measurement and is generated for that reason alone, and whose traffic characteristics are known a priori. The complete definition of Active Methods is specified in section 3.4 of[RFC7799]. Examples of Active Measurement Methods are the measurement methods for the One way delay metric defined in [RFC7679] and the one for round trip delay defined in [RFC2681].

Passive Measurement Method: Methods of Measurement conducted on network traffic, generated either from the end users or from network elements that would exist regardless whether the measurement was being conducted or not. The complete definition of Passive Methods is specified in section 3.6 of [RFC7799]. One characteristic of Passive Measurement Methods is that sensitive information may be observed, and as a consequence, stored in the measurement system.

Hybrid Measurement Method: Hybrid Methods are Methods of Measurement that use a combination of Active Methods and Passive Methods, to assess Active Metrics, Passive Metrics, or new metrics derived from the a priori knowledge and observations of the stream of interest. The complete definition of Hybrid Methods is specified in section 3.8 of [RFC7799].

## 3. Scope

This document is intended for two different audiences:

1. For those defining new Registered Performance Metrics, it provides specifications and best practices to be used in deciding which Registered Performance Metrics are useful for a measurement study, instructions for writing the text for each column of the Registered Performance Metrics, and information on the supporting documentation required for the new Performance Metrics Registry entry (up to and including the publication of one or more immutable documents such as an RFC).

2. For the appointed Performance Metrics Experts and for IANA personnel administering the new IANA Performance Metrics Registry, it defines a set of acceptance criteria against which these proposed Registered Performance Metrics should be evaluated.

In addition, this document may be useful for other organizations who are defining a Performance Metric registry of their own, and may re-use the features of the Performance Metrics Registry defined in this document.

This Performance Metrics Registry is applicable to Performance Metrics issued from Active Measurement, Passive Measurement, and any other form of Performance Metric. This registry is designed to encompass Performance Metrics developed throughout the IETF and especially for the technologies specified in the following working groups: IPPM, XRBLOCK, IPFIX, and BMWG. This document analyzes a prior attempt to set up a Performance Metrics Registry, and the reasons why this design was inadequate [RFC6248]. Finally, this document gives a set of guidelines for requesters and expert reviewers of candidate Registered Performance Metrics.

This document makes no attempt to populate the Performance Metrics Registry with initial entries; the related memo [I-D.ietf-ippm-initial-registry] proposes the initial set of regsitry entries.

## 4. Motivation for a Performance Metrics Registry

In this section, we detail several motivations for the Performance Metrics Registry.

### 4.1. Interoperability

As with any IETF registry, the primary intention is to manage registration of identifiers for use within one or more protocols. In the particular case of the Performance Metrics Registry, there are two types of protocols that will use the Performance Metrics in the Performance Metrics Registry during their operation (by referring to the Index values):

- o Control protocol: This type of protocol used to allow one entity to request another entity to perform a measurement using a specific metric defined by the Performance Metrics Registry. One particular example is the LMAP framework [RFC7594]. Using the LMAP terminology, the Performance Metrics Registry is used in the LMAP Control protocol to allow a Controller to schedule a measurement task for one or more Measurement Agents. In order to enable this use case, the entries of the Performance Metrics Registry must be sufficiently defined to allow a Measurement Agent implementation to trigger a specific measurement task upon the reception of a control protocol message. This requirement heavily constrains the type of entries that are acceptable for the Performance Metrics Registry.

- o Report protocol: This type of protocol is used to allow an entity to report measurement results to another entity. By referencing to a specific Performance Metrics Registry, it is possible to properly characterize the measurement result data being reported. Using the LMAP terminology, the Performance Metrics Registry is used in the Report protocol to allow a Measurement Agent to report measurement results to a Collector.

It should be noted that the LMAP framework explicitly allows for using not only the IANA-maintained Performance Metrics Registry but also other registries containing Performance Metrics, either defined by other organizations or private ones. However, others who are creating Registries to be used in the context of an LMAP framework are encouraged to use the Registry format defined in this document, because this makes it easier for developers of LMAP Measurement Agents (MAs) to programmatically use information found in those other Registries' entries.

### 4.2. Single point of reference for Performance Metrics

A Performance Metrics Registry serves as a single point of reference for Performance Metrics defined in different working groups in the IETF. As we mentioned earlier, there are several WGs that define Performance Metrics in the IETF and it is hard to keep track of all them. This results in multiple definitions of similar Performance Metrics that attempt to measure the same phenomena but in slightly different (and incompatible) ways. Having a registry would allow the IETF community and others to have a single list of relevant Performance Metrics defined by the IETF (and others, where appropriate). The single list is also an essential aspect of communication about Performance Metrics, where different entities that request measurements, execute measurements, and report the results can benefit from a common understanding of the referenced Performance Metric.

### 4.3. Side benefits

There are a couple of side benefits of having such a registry. First, the Performance Metrics Registry could serve as an inventory of useful and used Performance Metrics, that are normally supported by different implementations of measurement agents. Second, the results of measurements using the Performance Metrics should be comparable even if they are performed by different implementations and in different networks, as the Performance Metric is properly defined. BCP 176 [RFC6576] examines whether the results produced by independent implementations are equivalent in the context of evaluating the completeness and clarity of metric specifications. This BCP defines the standards track advancement testing for (active) IPPM metrics, and the same process will likely suffice to determine whether Registered Performance Metrics are sufficiently well specified to result in comparable (or equivalent) results. Registered Performance Metrics which have undergone such testing SHOULD be noted, with a reference to the test results.

## 5. Criteria for Performance Metrics Registration

It is neither possible nor desirable to populate the Performance Metrics Registry with all combinations of Parameters of all Performance Metrics. The Registered Performance Metrics SHOULD be:

1. interpretable by the user.

2. implementable by the software or hardware designer,

3. deployable by network operators,

4. accurate in terms of producing equivalent results, and for interoperability and deployment across vendors,

5. Operationally useful, so that it has significant industry interest and/or has seen deployment,

6. Sufficiently tightly defined, so that different values for the Run-time Parameters does not change the fundamental nature of the measurement, nor change the practicality of its implementation.

In essence, there needs to be evidence that a candidate Registered Performance Metric has significant industry interest, or has seen deployment, and there is agreement that the candidate Registered Performance Metric serves its intended purpose.

## 6. Performance Metric Registry: Prior attempt

There was a previous attempt to define a metric registry RFC 4148 [RFC4148]. However, it was obsoleted by RFC 6248 [RFC6248] because it was "found to be insufficiently detailed to uniquely identify IPPM metrics... [there was too much] variability possible when characterizing a metric exactly" which led to the RFC4148 registry having "very few users, if any".

A couple of interesting additional quotes from RFC 6248 [RFC6248] might help to understand the issues related to that registry.

1. "It is not believed to be feasible or even useful to register every possible combination of Type P, metric parameters, and Stream parameters using the current structure of the IPPM Metrics Registry."

2. "The registry structure has been found to be insufficiently detailed to uniquely identify IPPM metrics."

3. "Despite apparent efforts to find current or even future users, no one responded to the call for interest in the RFC 4148 registry during the second half of 2010."

The current approach learns from this by tightly defining each Registered Performance Metric with only a few variable (Run-time) Parameters to be specified by the measurement designer, if any. The idea is that entries in the Performance Metrics Registry stem from different measurement methods which require input (Run-time) parameters to set factors like source and destination addresses (which do not change the fundamental nature of the measurement). The downside of this approach is that it could result in a large number of entries in the Performance Metrics Registry. There is agreement that less is more in this context - it is better to have a reduced set of useful metrics rather than a large set of metrics, some with with questionable usefulness.

### 6.1. Why this Attempt Should Succeed

As mentioned in the previous section, one of the main issues with the previous registry was that the metrics contained in the registry were too generic to be useful. This document specifies stricter criteria for performance metric registration (see section 5), and imposes a group of Performance Metrics Experts that will provide guidelines to assess if a Performance Metric is properly specified.

Another key difference between this attempt and the previous one is that in this case there is at least one clear user for the Performance Metrics Registry: the LMAP framework and protocol. Because the LMAP protocol will use the Performance Metrics Registry values in its operation, this actually helps to determine if a metric is properly defined. In particular, since we expect that the LMAP control protocol will enable a controller to request a measurement agent to perform a measurement using a given metric by embedding the Performance Metrics Registry identifier in the protocol. Such a metric and method are properly specified if they are defined well-enough so that it is possible (and practical) to implement them in the measurement agent. This was the failure of the previous attempt: a registry entry with an undefined Type-P (section 13 of RFC 2330 [RFC2330]) allows implementation to be ambiguous.

## 7. Definition of the Performance Metric Registry

This Performance Metrics Registry is applicable to Performance Metrics used for Active Measurement, Passive Measurement, and any other form of Performance Measurement. Each category of measurement has unique properties, so some of the columns defined below are not applicable for a given metric category. In this case, the column(s) SHOULD be populated with the "NA" value (Non Applicable). However, the "NA" value MUST NOT be used by any metric in the following columns: Identifier, Name, URI, Status, Requester, Revision, Revision Date, Description. In the future, a new category of metrics could require additional columns, and adding new columns is a recognized form of registry extension. The specification defining the new column(s) MUST give general guidelines for populating the new column(s) for existing entries.

The columns of the Performance Metrics Registry are defined below. The columns are grouped into "Categories" to facilitate the use of the registry. Categories are described at the 7.x heading level, and columns are at the 7.x.y heading level. The Figure below illustrates this organization. An entry (row) therefore gives a complete description of a Registered Performance Metric.

Each column serves as a check-list item and helps to avoid omissions during registration and expert review.

```
=========================================================================
Legend:
    Registry Categories and Columns are shown below as:
    Category
    ------------------...
    Column |  Column |...
=========================================================================
Summary
------------------------------------------------------------------------
Identifier | Name | URI | Desc. | Reference | Change Controller | Ver |

Metric Definition
-----------------------------------------
Reference Definition | Fixed Parameters |

Method of Measurement
---------------------------------------------------------------------
Reference | Packet      | Traffic | Sampling     | Run-time    | Role |
Method    | Stream      | Filter  | Distribution | Parameters  |      |
          | Generation  |
Output
-----------------------------------------
Type | Reference  | Units | Calibration |
     | Definition |       |             |

Administrative Information
------------------------------------
Status |Requester | Rev | Rev.Date |

Comments and Remarks
--------------------
```

There is a blank template of the Registry template provided in Section 11 of this memo.

### 7.1. Summary Category

#### 7.1.1. Identifier

A numeric identifier for the Registered Performance Metric. This identifier MUST be unique within the Performance Metrics Registry.

The Registered Performance Metric unique identifier is an unbounded integer (range 0 to infinity).

The Identifier 0 should be Reserved. The Identifier values from 64512 to 65536 are reserved for private or experimental use, and the user may encounter overlapping uses.

When adding newly Registered Performance Metrics to the Performance Metrics Registry, IANA SHOULD assign the lowest available identifier to the new Registered Performance Metric.

If a Performance Metrics Expert providing review determines that there is a reason to assign a specific numeric identifier, possibly leaving a temporary gap in the numbering, then the Performance Expert SHALL inform IANA of this decision.

#### 7.1.2. Name

As the name of a Registered Performance Metric is the first thing a potential human implementor will use when determining whether it is suitable for their measurement study, it is important to be as precise and descriptive as possible. In future, users will review the names to determine if the metric they want to measure has already been registered, or if a similar entry is available as a basis for creating a new entry.

Names are composed of the following elements, separated by an underscore character "_":

MetricType_Method_SubTypeMethod_... Spec_Units_Output

- MetricType: a combination of the directional properties and the metric measured, such as and not limited to:

    RTDelay (Round Trip Delay)

    RTDNS (Response Time Domain Name Service)

    RLDNS (Response Loss Domain Name Service)

    OWDelay (One Way Delay)

RTLoss (Round Trip Loss)

OWLoss (One Way Loss)

OWPDV (One Way Packet Delay Variation)

OWIPDV (One Way Inter-Packet Delay Variation)

OWReorder (One Way Packet Reordering)

OWDuplic (One Way Packet Duplication)

OWBTC (One Way Bulk Transport Capacity)

OWMBM (One Way Model Based Metric)

SPMonitor (Single Point Monitor)

MPMonitor (Multi-Point Monitor)

- Method: One of the methods defined in [RFC7799], such as and not limited to:

  Active (depends on a dedicated measurement packet stream and observations of the stream)

  Passive (depends *solely* on observation of one or more existing packet streams)

  HybridType1 (observations on one stream that combine both active and passive methods)

  HybridType2 (observations on two or more streams that combine both active and passive methods)

  Spatial (Spatial Metric of RFC5644)

- SubTypeMethod: One or more sub-types to further describe the features of the entry, such as and not limited to:

  ICMP (Internet Control Message Protocol)

  IP (Internet Protocol)

  DSCPxx (where xx is replaced by a Diffserv code point)

  UDP (User Datagram Protocol)

TCP (Transport Control Protocol)

QUIC (QUIC transport protocol)

HS (Hand-Shake, such as TCP's 3-way HS)

Poisson (Packet generation using Poisson distribution)

Periodic (Periodic packet generation)

SendOnRcv (Sender keeps one packet in-transit by sending when previous packet arrives)

PayloadxxxxB (where xxxx is replaced by an integer, the number of octets in the Payload))

SustainedBurst (Capacity test, worst case)

StandingQueue (test of bottleneck queue behavior)

SubTypeMethod values are separated by a hyphen "-" character, which indicates that they belong to this element, and that their order is unimportant when considering name uniqueness.

- Spec: An immutable document identifier combined with a document section identifier. For RFCs, this consists of the RFC number and major section number that specifies this Registry entry in the form RFCXXXXsecY, such as RFC7799sec3. Note: the RFC number is not the Primary Reference specification for the metric definition, such as [RFC7679] for One-way Delay; it will contain the placeholder "RFCXXXXsecY" until the RFC number is assigned to the specifying document, and would remain blank in private registry entries without a corresponding RFC. Anticipating the "RFC10K" problem, the number of the RFC continues to replace RFCXXXX regardless of the number of digits in the RFC number. Anticipating Registry Entries from other standards bodies, the form of this Name Element MUST be proposed and reviewed for consistency and uniqueness by the Expert Reviewer.

- Units: The units of measurement for the output, such as and not limited to:

  Seconds

  Ratio (unitless)

Percent (value multiplied by 100%)

Logical (1 or 0)

Packets

BPS (Bits per Second)

PPS (Packets per Second)

EventTotal (for unit-less counts)

Multiple (more than one type of unit)

Enumerated (a list of outcomes)

Unitless

- Output: The type of output resulting from measurement, such as and not limited to:

  Singleton

  Raw (multiple Singletons)

  Count

  Minimum

  Maximum

  Median

  Mean

  95Percentile (95th Percentile)

  99Percentile (99th Percentile)

  StdDev (Standard Deviation)

  Variance

  PFI (Pass, Fail, Inconclusive)

  FlowRecords (descriptions of flows observed)

  LossRatio (lost packets to total packets, <=1)

An example is:

RTDelay_Active_IP-UDP-Periodic_RFCXXXXsecY_Seconds_95Percentile

as described in section 4 of [I-D.ietf-ippm-initial-registry].

Note that private registries following the format described here SHOULD use the prefix "Priv_" on any name to avoid unintended conflicts (further considerations are described in section 10). Private registry entries usually have no specifying RFC, thus the Spec: element has no clear interpretation.

#### 7.1.3. URI

The URIs column MUST contain a URL [RFC3986] that uniquely identifies and locates the metric entry so it is accessible through the Internet. The URL points to a file containing all the human-readable information for one registry entry. The URL SHALL reference a target file that is preferably HTML-formatted and contains URLs to referenced sections of HTML-ized RFCs, or other reference specifications. These target files for different entries can be more easily edited and re-used when preparing new entries. The exact form of the URL for each target file, and the target file itself, will be determined by IANA and reside on "iana.org". The major sections of [I-D.ietf-ippm-initial-registry] provide an example of a target file in HTML form (sections 4 and higher).

#### 7.1.4. Description

A Registered Performance Metric description is a written representation of a particular Performance Metrics Registry entry. It supplements the Registered Performance Metric name to help Performance Metrics Registry users select relevant Registered Performance Metrics.

#### 7.1.5. Reference

This entry gives the specification containing the candidate registry entry which was reviewed and agreed, if such an RFC or other specification exists.

#### 7.1.6. Change Controller

This entry names the entity responsible for approving revisions to the registry entry, and SHALL provide contact information (for an individual, where appropriate).

#### 7.1.7. Version (of Registry Format)

This entry gives the version number for the registry format used. Formats complying with this memo MUST use 1.0. The version number SHALL NOT change unless a new RFC is published that changes the registry format. The version number of registry entries SHALL NOT change unless the registry entry is updated (following procedures in section 8).

### 7.2. Metric Definition Category

This category includes columns to prompt all necessary details related to the metric definition, including the immutable document reference and values of input factors, called fixed parameters, which are left open in the immutable document, but have a particular value defined by the performance metric.

#### 7.2.1. Reference Definition

This entry provides a reference (or references) to the relevant section(s) of the document(s) that define the metric, as well as any supplemental information needed to ensure an unambiguous definition for implementations. The reference needs to be an immutable document, such as an RFC; for other standards bodies, it is likely to be necessary to reference a specific, dated version of a specification.

#### 7.2.2. Fixed Parameters

Fixed Parameters are Parameters whose value must be specified in the Performance Metrics Registry. The measurement system uses these values.

Where referenced metrics supply a list of Parameters as part of their descriptive template, a sub-set of the Parameters will be designated as Fixed Parameters. As an example for active metrics, Fixed Parameters determine most or all of the IPPM Framework convention "packets of Type-P" as described in [RFC2330], such as transport protocol, payload length, TTL, etc. An example for passive metrics is for RTP packet loss calculation that relies on the validation of a packet as RTP which is a multi-packet validation controlled by MIN_SEQUENTIAL as defined by [RFC3550]. Varying MIN_SEQUENTIAL values can alter the loss report and this value could be set as a Fixed Parameter.

Parameters MUST have well-defined names. For human readers, the hanging indent style is preferred, and any Parameter names and definitions that do not appear in the Reference Method Specification MUST appear in this column (or Run-time Parameters column).

Parameters MUST have a well-specified data format.

A Parameter which is a Fixed Parameter for one Performance Metrics Registry entry may be designated as a Run-time Parameter for another Performance Metrics Registry entry.

### 7.3. Method of Measurement Category

This category includes columns for references to relevant sections of the immutable document(s) and any supplemental information needed to ensure an unambiguous method for implementations.

#### 7.3.1. Reference Method

This entry provides references to relevant sections of immutable documents, such as RFC(s) (for other standards bodies, it is likely to be necessary to reference a specific, dated version of a specification) describing the method of measurement, as well as any supplemental information needed to ensure unambiguous interpretation for implementations referring to the immutable document text.

Specifically, this section should include pointers to pseudocode or actual code that could be used for an unambiguous implementation.

#### 7.3.2. Packet Stream Generation

This column applies to Performance Metrics that generate traffic as part of their Measurement Method, including but not necessarily limited to Active metrics. The generated traffic is referred as a stream and this column describes its characteristics.

Each entry for this column contains the following information:

o Value: The name of the packet stream scheduling discipline

o Reference: the specification where the parameters of the stream are defined

The packet generation stream may require parameters such as the average packet rate and distribution truncation value for streams with Poisson-distributed inter-packet sending times. In case such parameters are needed, they should be included either in the Fixed parameter column or in the run time parameter column, depending on whether they will be fixed or will be an input for the metric.

The simplest example of stream specification is Singleton scheduling (see [RFC2330]), where a single atomic measurement is conducted. Each atomic measurement could consist of sending a single packet (such as a DNS request) or sending several packets (for example, to request a webpage). Other streams support a series of atomic measurements in a "sample", with a schedule defining the timing between each transmitted packet and subsequent measurement. Principally, two different streams are used in IPPM metrics, Poisson distributed as described in [RFC2330] and Periodic as described in [RFC3432]. Both Poisson and Periodic have their own unique parameters, and the relevant set of parameters names and values should be included either in the Fixed Parameters column or in the Run-time parameter column.

### 7.3.3. Traffic Filter

This column applies to Performance Metrics that observe packets flowing through (the device with) the measurement agent i.e. that is not necessarily addressed to the measurement agent. This includes but is not limited to Passive Metrics. The filter specifies the traffic that is measured. This includes protocol field values/ ranges, such as address ranges, and flow or session identifiers.

The traffic filter itself depends on needs of the metric itself and a balance of an operator’s measurement needs and a user’s need for privacy. Mechanics for conveying the filter criteria might be the BPF (Berkley Packet Filter) or PSAMP [RFC5475] Property Match Filtering which reuses IPFIX [RFC7012]. An example BPF string for matching TCP/80 traffic to remote destination net 192.0.2.0/24 would be "dst net 192.0.2.0/24 and tcp dst port 80". More complex filter engines might be supported by the implementation that might allow for matching using Deep Packet Inspection (DPI) technology.

The traffic filter includes the following information:

Type: the type of traffic filter used, e.g. BPF, PSAMP, OpenFlow rule, etc. as defined by a normative reference

Value: the actual set of rules expressed

### 7.3.4. Sampling Distribution

The sampling distribution defines out of all the packets that match the traffic filter, which one of those are actually used for the measurement. One possibility is "all" which implies that all packets matching the Traffic filter are considered, but there may be other sampling strategies. It includes the following information:

Value: the name of the sampling distribution

Reference definition: pointer to the specification where the sampling distribution is properly defined.

The sampling distribution may require parameters. In case such parameters are needed, they should be included either in the Fixed parameter column or in the run time parameter column, depending on whether they will be fixed or will be an input for the metric.

Sampling and Filtering Techniques for IP Packet Selection are documented in the PSAMP (Packet Sampling) [RFC5475], while the Framework for Packet Selection and Reporting, [RFC5474] provides more background information. The sampling distribution parameters might be expressed in terms of the Information Model for Packet Sampling Exports, [RFC5477], and the Flow Selection Techniques, [RFC7014].

### 7.3.5. Run-time Parameters

Run-Time Parameters are Parameters that must be determined, configured into the measurement system, and reported with the results for the context to be complete. However, the values of these parameters is not specified in the Performance Metrics Registry (like the Fixed Parameters), rather these parameters are listed as an aid to the measurement system implementer or user (they must be left as variables, and supplied on execution).

Where metrics supply a list of Parameters as part of their descriptive template, a sub-set of the Parameters will be designated as Run-Time Parameters.

Parameters MUST have well defined names. For human readers, the hanging indent style is preferred, and the names and definitions that do not appear in the Reference Method Specification MUST appear in this column.

A Data Format for each Run-time Parameter MUST be specified in this column, to simplify the control and implementation of measurement devices. For example, parameters that include an IPv4 address can be encoded as a 32 bit integer (i.e. binary base64 encoded value) or ip-address as defined in [RFC6991]. The actual encoding(s) used must be explicitly defined for each Run-time parameter. IPv6 addresses and options MUST be accommodated, allowing Registered Metrics to be used in that address family. Other address families are permissable.

Examples of Run-time Parameters include IP addresses, measurement point designations, start times and end times for measurement, and other information essential to the method of measurement.

### 7.3.6. Role

In some methods of measurement, there may be several roles defined, e.g., for a one-way packet delay active measurement there is one measurement agent that generates the packets and another agent that receives the packets. This column contains the name of the Role(s) for this particular entry. In the one-way delay example above, there should be two entries in the Role registry column, one for each Role (Source and Destination). When a measurement agent is instructed to perform the "Source" Role for one-way delay metric, the agent knows that it is required to generate packets. The values for this field are defined in the reference method of measurement (and this frequently results in abbreviated role names such as "Src").

When the Role column of a registry entry defines more than one Role, then the Role SHALL be treated as a Run-time Parameter and supplied for execution. It should be noted that the LMAP framework [RFC7594] distinguishes the Role from other Run-time Parameters, and defines a special parameter "Roles" inside the registry-grouping function list in the LMAP YANG model[RFC8194].

## 7.4. Output Category

For entries which involve a stream and many singleton measurements, a statistic may be specified in this column to summarize the results to a single value. If the complete set of measured singletons is output, this will be specified here.

Some metrics embed one specific statistic in the reference metric definition, while others allow several output types or statistics.

### 7.4.1. Type

This column contains the name of the output type. The output type defines a single type of result that the metric produces. It can be the raw results (packet send times and singleton metrics), or it can be a summary statistic. The specification of the output type MUST define the format of the output. In some systems, format specifications will simplify both measurement implementation and collection/storage tasks. Note that if two different statistics are required from a single measurement (for example, both "Xth percentile mean" and "Raw"), then a new output type must be defined ("Xth percentile mean AND Raw"). See the Naming section above for a list of Output Types.

### 7.4.2. Reference Definition

This column contains a pointer to the specification(s) where the output type and format are defined.

### 7.4.3. Metric Units

The measured results must be expressed using some standard dimension or units of measure. This column provides the units.

When a sample of singletons (see Section 11 of[RFC2330] for definitions of these terms) is collected, this entry will specify the units for each measured value.

### 7.4.4. Calibration

Some specifications for Methods of Measurement include the possibility to perform an error calibration. Section 3.7.3 of [RFC7679] is one example. In the registry entry, this field will identify a method of calibration for the metric, and when available, the measurement system SHOULD perform the calibration when requested and produce the output with an indication that it is the result of a calibration method. In-situ calibration could be enabled with an internal loopback that includes as much of the measurement system as possible, performs address manipulation as needed, and provides some form of isolation (e.g., deterministic delay) to avoid send-receive interface contention. Some portion of the random and systematic error can be characterized this way.

For one-way delay measurements, the error calibration must include an assessment of the internal clock synchronization with its external reference (this internal clock is supplying timestamps for measurement). In practice, the time offsets of clocks at both the source and destination are needed to estimate the systematic error due to imperfect clock synchronization (the time offsets are smoothed, thus the random variation is not usually represented in the results).

Both internal loopback calibration and clock synchronization can be used to estimate the *available accuracy* of the Output Metric Units. For example, repeated loopback delay measurements will reveal the portion of the Output result resolution which is the result of system noise, and thus inaccurate.

### 7.5. Administrative information

#### 7.5.1. Status

The status of the specification of this Registered Performance Metric. Allowed values are 'current' and 'deprecated'. All newly defined Information Elements have 'current' status.

#### 7.5.2. Requester

The requester for the Registered Performance Metric. The requester MAY be a document, such as RFC, or person.

#### 7.5.3. Revision

The revision number of a Registered Performance Metric, starting at 0 for Registered Performance Metrics at time of definition and incremented by one for each revision.

#### 7.5.4. Revision Date

The date of acceptance or the most recent revision for the Registered Performance Metric. The date SHALL be determined by IANA and the reviewing Performance Metrics Expert.

### 7.6. Comments and Remarks

Besides providing additional details which do not appear in other categories, this open Category (single column) allows for unforeseen issues to be addressed by simply updating this informational entry.

## 8. Processes for Managing the Performance Metric Registry Group

Once a Performance Metric or set of Performance Metrics has been identified for a given application, candidate Performance Metrics Registry entry specifications prepared in accordance with Section 7 should be submitted to IANA to follow the process for review by the Performance Metric Experts, as defined below. This process is also used for other changes to the Performance Metrics Registry, such as deprecation or revision, as described later in this section.

It is desirable that the author(s) of a candidate Performance Metrics Registry entry seek review in the relevant IETF working group, or offer the opportunity for review on the working group mailing list.

### 8.1. Adding new Performance Metrics to the Performance Metrics Registry

Requests to add Registered Performance Metrics in the Performance Metrics Registry SHALL be submitted to IANA, which forwards the request to a designated group of experts (Performance Metric Experts) appointed by the IESG; these are the reviewers called for by the Specification Required [RFC8126] policy defined for the Performance Metrics Registry. The Performance Metric Experts review the request for such things as compliance with this document, compliance with other applicable Performance Metric-related RFCs, and consistency with the currently defined set of Registered Performance Metrics. The most efficient path for submission begins with preparation of an Internet Draft containing the proposed Performance Metrics Registry entry using the template in Section 11, so that the submission formatting will benefit from the normal IETF Internet Draft submission processing (including HTML-ization).

Submission to IANA may be during IESG review (leading to IETF Standards Action), where an Internet Draft proposes one or more Registered Performance Metrics to be added to the Performance Metrics Registry, including the text of the proposed Registered Performance Metric(s).

If an RFC-to-be includes a Performance Metric and a proposed Performance Metrics Registry entry, but the Performance Metric Expert review determines that one or more of the Section 5 criteria have not been met, then the proposed Performance Metrics Registry entry MUST be removed from the text. Once evidence exists that the Performance Metric meets the criteria in section 5, the proposed Performance Metrics Registry entry SHOULD be submitted to IANA to be evaluated in consultation with the Performance Metric Experts for registration at that time.

Authors of proposed Registered Performance Metrics SHOULD review compliance with the specifications in this document to check their submissions before sending them to IANA.

At least one Performance Metric Expert should endeavor to complete referred reviews in a timely manner. If the request is acceptable, the Performance Metric Experts signify their approval to IANA, and IANA updates the Performance Metrics Registry. If the request is not acceptable, the Performance Metric Experts MAY coordinate with the requester to change the request to be compliant, otherwise IANA SHALL coordinate resolution of issues on behalf of the expert. The Performance Metric Experts MAY choose to reject clearly frivolous or inappropriate change requests outright, but such exceptional circumstances should be rare.

This process should not in any way be construed as allowing the Performance Metric Experts to overrule IETF consensus. Specifically, any Registered Performance Metrics that were added to the Performance Metrics Registry with IETF consensus require IETF consensus for revision or deprecation.

Decisions by the Performance Metric Experts may be appealed as in Section 7 of [RFC8126].

### 8.2. Revising Registered Performance Metrics

A request for Revision is only permitted when the requested changes maintain backward-compatibility with implementations of the prior Performance Metrics Registry entry describing a Registered Performance Metric (entries with lower revision numbers, but the same Identifier and Name).

The purpose of the Status field in the Performance Metrics Registry is to indicate whether the entry for a Registered Performance Metric is 'current' or 'deprecated'.

In addition, no policy is defined for revising the Performance Metric entries in the IANA Registry or addressing errors therein. To be clear, changes and deprecations within the Performance Metrics Registry are not encouraged, and should be avoided to the extent possible. However, in recognition that change is inevitable, the provisions of this section address the need for revisions.

Revisions are initiated by sending a candidate Registered Performance Metric definition to IANA, as in Section 8.1, identifying the existing Performance Metrics Registry entry, and explaining how and why the existing entry should be revised.

The primary requirement in the definition of procedures for managing changes to existing Registered Performance Metrics is avoidance of measurement interoperability problems; the Performance Metric Experts must work to maintain interoperability above all else. Changes to Registered Performance Metrics may only be done in an interoperable way; necessary changes that cannot be done in a way to allow interoperability with unchanged implementations MUST result in the creation of a new Registered Performance Metric (with a new Name, replacing the RFCXXXXsecY portion of the name) and possibly the deprecation of the earlier metric.

A change to a Registered Performance Metric SHALL be determined to be backward-compatible when:

1. it involves the correction of an error that is obviously only editorial; or

2. it corrects an ambiguity in the Registered Performance Metric's definition, which itself leads to issues severe enough to prevent the Registered Performance Metric's usage as originally defined; or

3. it corrects missing information in the metric definition without changing its meaning (e.g., the explicit definition of 'quantity' semantics for numeric fields without a Data Type Semantics value); or

4. it harmonizes with an external reference that was itself corrected.

If a Performance Metric revision is deemed permissible and backward-compatible by the Performance Metric Experts, according to the rules in this document, IANA SHOULD execute the change(s) in the Performance Metrics Registry. The requester of the change is appended to the original requester in the Performance Metrics Registry. The Name of the revised Registered Performance Metric, including the RFCXXXXsecY portion of the name, SHALL remain unchanged (even when the change is the result of IETF Standards Action; the revised registry entry SHOULD reference the new immutable document, such as an RFC or for other standards bodies, it is likely to be necessary to reference a specific, dated version of a specification, in an appropriate category and column).

Each Registered Performance Metric in the Performance Metrics Registry has a revision number, starting at zero. Each change to a Registered Performance Metric following this process increments the revision number by one.

When a revised Registered Performance Metric is accepted into the Performance Metrics Registry, the date of acceptance of the most recent revision is placed into the revision Date column of the registry for that Registered Performance Metric.

Where applicable, additions to Registered Performance Metrics in the form of text Comments or Remarks should include the date, but such additions may not constitute a revision according to this process.

Older version(s) of the updated metric entries are kept in the registry for archival purposes. The older entries are kept with all fields unmodified (version, revision date) except for the status field that SHALL be changed to "Deprecated".

### 8.3. Deprecating Registered Performance Metrics

Changes that are not permissible by the above criteria for Registered Performance Metric's revision may only be handled by deprecation. A Registered Performance Metric MAY be deprecated and replaced when:

1. the Registered Performance Metric definition has an error or shortcoming that cannot be permissibly changed as in Section 8.2 Revising Registered Performance Metrics; or

2. the deprecation harmonizes with an external reference that was itself deprecated through that reference's accepted deprecation method.

A request for deprecation is sent to IANA, which passes it to the Performance Metric Experts for review. When deprecating an Performance Metric, the Performance Metric description in the Performance Metrics Registry must be updated to explain the deprecation, as well as to refer to any new Performance Metrics created to replace the deprecated Performance Metric.

The revision number of a Registered Performance Metric is incremented upon deprecation, and the revision Date updated, as with any revision.

The intentional use of deprecated Registered Performance Metrics should result in a log entry or human-readable warning by the respective application.

Names and Metric IDs of deprecated Registered Performance Metrics must not be reused.

The deprecated entries are kept with all fields unmodified, except the version, revision date, and the status field (changed to "Deprecated").

## 9. Security considerations

This draft defines a registry structure, and does not itself introduce any new security considerations for the Internet. The definition of Performance Metrics for this registry may introduce some security concerns, but the mandatory references should have their own considerations for security, and such definitions should be reviewed with security in mind if the security considerations are not covered by one or more reference standards.

The aggregated results of the performance metrics described in this registry might reveal network topology information that may be considered sensitive.  If such cases are found, then access control mechanisms should be applied.

## 10.  IANA Considerations

With the background and processes described in earlier sections, this document requests the following IANA Actions.

Editor's Note: Mock-ups of the implementation of this set of requests have been prepared with IANA's help during development of this memo, and have been captured in the Proceedings of IPPM working group sessions.  IANA is currently preparing a mock-up.  A recent version is available here: http://encrypted.net/IETFMetricsRegistry-106.html

### 10.1.  Registry Group

The new registry group SHALL be named, "PERFORMANCE METRICS Group".

Registration Procedure: Specification Required

Reference: <This RFC>

Experts: Performance Metrics Experts

Note: TBD

### 10.2.  Performance Metric Name Elements

This document specifies the procedure for Performance Metrics Name Element Registry setup.  IANA is requested to create a new set of registries for Performance Metric Name Elements called "Registered Performance Metric Name Elements".  Each Registry, whose names are listed below:

MetricType:

Method:

SubTypeMethod:

Spec:

Units:

Output:

will contain the current set of possibilities for Performance Metrics Registry Entry Names.

To populate the Registered Performance Metric Name Elements at creation, the IANA is asked to use the lists of values for each name element listed in Section 7.1.2. The Name Elements in each registry are case-sensitive.

When preparing a Metric entry for Registration, the developer SHOULD choose Name elements from among the registered elements. However, if the proposed metric is unique in a significant way, it may be necessary to propose a new Name element to properly describe the metric, as described below.

A candidate Metric Entry RFC or immutable document for IANA and Expert Review would propose one or more new element values required to describe the unique entry, and the new name element(s) would be reviewed along with the metric entry. New assignments for Registered Performance Metric Name Elements will be administered by IANA through Specification Required policy (which includes Expert Review) [RFC8126], i.e., review by one of a group of experts, the Performance Metric Experts, who are appointed by the IESG upon recommendation of the Transport Area Directors.

## 10.3. New Performance Metrics Registry

This document specifies the procedure for Performance Metrics Registry setup. IANA is requested to create a new registry for Performance Metrics called "Performance Metrics Registry". This Registry will contain the following Summary columns:

Identifier:

Name:

URI:

Description:

Reference:

Change Controller:

Version:

Descriptions of these columns and additional information found in the template for registry entries (categories and columns) are further defined in section Section 7.

The Identifier 0 should be Reserved. The Registered Performance Metric unique identifier is an unbounded integer (range 0 to infinity). The Identifier values from 64512 to 65536 are reserved for private or experimental use, and the user may encounter overlapping uses. When adding newly Registered Performance Metrics to the Performance Metrics Registry, IANA SHOULD assign the lowest available identifier to the new Registered Performance Metric. If a Performance Metrics Expert providing review determines that there is a reason to assign a specific numeric identifier, possibly leaving a temporary gap in the numbering, then the Performance Expert SHALL inform IANA of this decision.

Names starting with the prefix Priv_ are reserved for private use, and are not considered for registration. The "Name" column entries are further defined in section Section 7.

The "URI" column will have a URL to the full template of each registry entry. The Registry Entry text SHALL be HTML-ized to aid the reader, with links to reference RFCs (similar to the way that Internet Drafts are HTML-ized, the same tool can perform the function) or immutable document.

The "Reference" column will include an RFC number, an approved specification designator from another standards body, or other immutable document.

New assignments for Performance Metrics Registry will be administered by IANA through Specification Required policty (which includes Expert Review) [RFC8126], i.e., review by one of a group of experts, the Performance Metric Experts, who are appointed by the IESG upon recommendation of the Transport Area Directors, or by Standards Action. The experts can be initially drawn from the Working Group Chairs, document editors, and members of the Performance Metrics Directorate, among other sources of experts.

Extensions of the Performance Metrics Registry require IETF Standards Action. Only one form of registry extension is envisaged:

1. Adding columns, or both categories and columns, to accommodate unanticipated aspects of new measurements and metric categories.

If the Performance Metrics Registry is extended in this way, the Version number of future entries complying with the extension SHALL be incremented (either in the unit or tenths digit, depending on the degree of extension.

# 11. Blank Registry Template

This section provides a blank template to help IANA and registry entry writers.

## 11.1. Summary

This category includes multiple indexes to the registry entry: the element ID and metric name.

### 11.1.1. ID (Identifier)

<insert a numeric identifier, an integer, TBD>

### 11.1.2. Name

<insert name according to metric naming convention>

### 11.1.3. URI

URL: https://www.iana.org/ ... <name>

### 11.1.4. Description

<provide a description>

### 11.1.5. Change Controller

### 11.1.6. Version (of Registry Format)

## 11.2. Metric Definition

This category includes columns to prompt the entry of all necessary details related to the metric definition, including the immutable document reference and values of input factors, called fixed parameters.

### 11.2.1. Reference Definition

<Full bibliographic reference to an immutable doc.>

<specific section reference and additional clarifications, if needed>

### 11.2.2. Fixed Parameters

<list and specify Fixed Parameters, input factors that must be determined and embedded in the measurement system for use when needed>

### 11.3. Method of Measurement

This category includes columns for references to relevant sections of the immutable documents(s) and any supplemental information needed to ensure an unambiguous methods for implementations.

#### 11.3.1. Reference Method

<for metric, insert relevant section references and supplemental info>

#### 11.3.2. Packet Stream Generation

<list of generation parameters and section/spec references if needed>

#### 11.3.3. Traffic Filtering (observation) Details

The measured results based on a filtered version of the packets observed, and this section provides the filter details (when present).

<section reference>.

#### 11.3.4. Sampling Distribution

<insert time distribution details, or how this is diff from the filter>

#### 11.3.5. Run-time Parameters and Data Format

Run-time Parameters are input factors that must be determined, configured into the measurement system, and reported with the results for the context to be complete.

<list of run-time parameters, and their data formats>

#### 11.3.6. Roles

<lists the names of the different roles from the measurement method>

### 11.4. Output

This category specifies all details of the Output of measurements using the metric.

#### 11.4.1. Type

<insert name of the output type, raw or a selected summary statistic>

#### 11.4.2. Reference Definition

<describe the reference data format for each type of result>

#### 11.4.3. Metric Units

<insert units for the measured results, and the reference specification>.

#### 11.4.4. Calibration

<insert information on calibration>

### 11.5. Administrative items

#### 11.5.1. Status

<current or deprecated>

#### 11.5.2. Requester

<name or RFC, etc.>

#### 11.5.3. Revision

<1.0>

#### 11.5.4. Revision Date

<format YYYY-MM-DD>

### 11.6. Comments and Remarks

<Additional (Informational) details for this entry>

## 12. Acknowledgments

Thanks to Brian Trammell and Bill Cerveny, IPPM chairs, for leading some brainstorming sessions on this topic. Thanks to Barbara Stark and Juergen Schoenwaelder for the detailed feedback and suggestions. Thanks to Andrew McGregor for suggestions on metric naming. Thanks to Michelle Cotton for her early IANA review, and to Amanda Barber for answering questions related to the presentation of the registry and accessibility of the complete template via URL. Thanks to Roni

Even for his review and suggestions to generalize the procedures.
Thanks to ˜all the Area Directors for their reviews.

# 13. References

## 13.1. Normative References

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, DOI 10.17487/RFC2026, October 1996, <https://www.rfc-editor.org/info/rfc2026>.

[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>.

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, DOI 10.17487/RFC2330, May 1998, <https://www.rfc-editor.org/info/rfc2330>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC6390] Clark, A. and B. Claise, "Guidelines for Considering New Performance Metric Development", BCP 170, RFC 6390, DOI 10.17487/RFC6390, October 2011, <https://www.rfc-editor.org/info/rfc6390>.

[RFC6576] Geib, R., Ed., Morton, A., Fardid, R., and A. Steinmitz, "IP Performance Metrics (IPPM) Standard Advancement Testing", BCP 176, RFC 6576, DOI 10.17487/RFC6576, March 2012, <https://www.rfc-editor.org/info/rfc6576>.

[RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <https://www.rfc-editor.org/info/rfc7799>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

## 13.2. Informative References

[I-D.ietf-ippm-initial-registry]
Morton, A., Bagnulo, M., Eardley, P., and K. D'Souza, "Initial Performance Metrics Registry Entries", draft-ietf-ippm-initial-registry-15 (work in progress), December 2019.

[RFC2681] Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, DOI 10.17487/RFC2681, September 1999, <https://www.rfc-editor.org/info/rfc2681>.

[RFC3432] Raisanen, V., Grotefeld, G., and A. Morton, "Network performance measurement with periodic streams", RFC 3432, DOI 10.17487/RFC3432, November 2002, <https://www.rfc-editor.org/info/rfc3432>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, DOI 10.17487/RFC3611, November 2003, <https://www.rfc-editor.org/info/rfc3611>.

[RFC4148] Stephan, E., "IP Performance Metrics (IPPM) Metrics Registry", BCP 108, RFC 4148, DOI 10.17487/RFC4148, August 2005, <https://www.rfc-editor.org/info/rfc4148>.

[RFC5474] Duffield, N., Ed., Chiou, D., Claise, B., Greenberg, A., Grossglauser, M., and J. Rexford, "A Framework for Packet Selection and Reporting", RFC 5474, DOI 10.17487/RFC5474, March 2009, <https://www.rfc-editor.org/info/rfc5474>.

[RFC5475] Zseby, T., Molina, M., Duffield, N., Niccolini, S., and F. Raspall, "Sampling and Filtering Techniques for IP Packet Selection", RFC 5475, DOI 10.17487/RFC5475, March 2009, <https://www.rfc-editor.org/info/rfc5475>.

[RFC5477] Dietz, T., Claise, B., Aitken, P., Dressler, F., and G. Carle, "Information Model for Packet Sampling Exports", RFC 5477, DOI 10.17487/RFC5477, March 2009, <https://www.rfc-editor.org/info/rfc5477>.

[RFC6035] Pendleton, A., Clark, A., Johnston, A., and H. Sinnreich, "Session Initiation Protocol Event Package for Voice Quality Reporting", RFC 6035, DOI 10.17487/RFC6035, November 2010, <https://www.rfc-editor.org/info/rfc6035>.

[RFC6248] Morton, A., "RFC 4148 and the IP Performance Metrics (IPPM) Registry of Metrics Are Obsolete", RFC 6248, DOI 10.17487/RFC6248, April 2011, <https://www.rfc-editor.org/info/rfc6248>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7012] Claise, B., Ed. and B. Trammell, Ed., "Information Model for IP Flow Information Export (IPFIX)", RFC 7012, DOI 10.17487/RFC7012, September 2013, <https://www.rfc-editor.org/info/rfc7012>.

[RFC7014] D'Antonio, S., Zseby, T., Henke, C., and L. Peluso, "Flow Selection Techniques", RFC 7014, DOI 10.17487/RFC7014, September 2013, <https://www.rfc-editor.org/info/rfc7014>.

[RFC7594] Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A Framework for Large-Scale Measurement of Broadband Performance (LMAP)", RFC 7594, DOI 10.17487/RFC7594, September 2015, <https://www.rfc-editor.org/info/rfc7594>.

[RFC7679] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Delay Metric for IP Performance Metrics (IPPM)", STD 81, RFC 7679, DOI 10.17487/RFC7679, January 2016, <https://www.rfc-editor.org/info/rfc7679>.

[RFC8194] Schoenwaelder, J. and V. Bajpai, "A YANG Data Model for LMAP Measurement Agents", RFC 8194, DOI 10.17487/RFC8194, August 2017, <https://www.rfc-editor.org/info/rfc8194>.

## Authors' Addresses

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid  28911
SPAIN

Phone: 34 91 6249500
Email: marcelo@it.uc3m.es
URI:   http://www.it.uc3m.es

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Email: bclaise@cisco.com

Philip Eardley
BT
Adastral Park, Martlesham Heath
Ipswich
ENGLAND

Email: philip.eardley@bt.com

Al Morton
AT&T Labs
200 Laurel Avenue South
Middletown, NJ
USA

Email: acmorton@att.com

Aamer Akhter
Consultant
118 Timber Hitch
Cary, NC
USA

Email: aakhter@gmail.com

IP Performance Working Group M. Mathis
Internet-Draft Google, Inc
Intended status: Experimental A. Morton
Expires: March 19, 2018 AT&T Labs
September 15, 2017

# Model Based Metrics for Bulk Transport Capacity
draft-ietf-ippm-model-based-metrics-13.txt

## Abstract

We introduce a new class of Model Based Metrics designed to assess if a complete Internet path can be expected to meet a predefined Target Transport Performance by applying a suite of IP diagnostic tests to successive subpaths. The subpath-at-a-time tests can be robustly applied to critical infrastructure, such as network interconnections or even individual devices, to accurately detect if any part of the infrastructure will prevent paths traversing it from meeting the Target Transport Performance.

Model Based Metrics rely on mathematical models to specify a Targeted Suite of IP Diagnostic tests, designed to assess whether common transport protocols can be expected to meet a predetermined Target Transport Performance over an Internet path.

For Bulk Transport Capacity the IP diagnostics are built using test streams and statistical criteria for evaluating the packet transfer that mimic TCP over the complete path. The temporal structure of the test stream (bursts, etc) mimic TCP or other transport protocol carrying bulk data over a long path. However they are constructed to be independent of the details of the subpath under test, end systems or applications. Likewise the success criteria evaluates the packet transfer statistics of the subpath against criteria determined by protocol performance models applied to the Target Transport Performance of the complete path. The success criteria also does not depend on the details of the subpath, end systems or application.

## Status of This Memo

Table of Contents

## 1. Introduction

Model Based Metrics (MBM) rely on peer-reviewed mathematical models to specify a Targeted Suite of IP Diagnostic tests, designed to assess whether common transport protocols can be expected to meet a predetermined Target Transport Performance over an Internet path. This note describes the modeling framework to derive the test parameters for assessing an Internet path's ability to support a predetermined Bulk Transport Capacity.

Each test in the Targeted IP Diagnostic Suite (TIDS) measures some aspect of IP packet transfer needed to meet the Target Transport Performance. For Bulk Transport Capacity the TIDS includes IP diagnostic tests to verify that there is: sufficient IP capacity (data rate); sufficient queue space at bottlenecks to absorb and deliver typical transport bursts; and that the background packet loss ratio is low enough not to interfere with congestion control; and other properties described below. Unlike typical IPPM metrics which yield measures of network properties, Model Based Metrics nominally yield pass/fail evaluations of the ability of standard transport protocols to meet the specific performance objective over some network path.

In most cases, the IP diagnostic tests can be implemented by combining existing IPPM metrics with additional controls for generating test streams having a specified temporal structure (bursts or standing queues caused by constant bit rate streams, etc.) and statistical criteria for evaluating packet transfer. The temporal structure of the test streams mimic transport protocol behavior over the complete path; the statistical criteria models the transport protocol's response to less than ideal IP packet transfer. In control theory terms, the tests are "open loop". Note that running a test requires the coordinated activity of sending and receiving measurement points.

This note addresses Bulk Transport Capacity. It describes an alternative to the approach presented in "A Framework for Defining Empirical Bulk Transfer Capacity Metrics" [RFC3148]. Other Model Based Metrics may cover other applications and transports, such as VoIP over UDP and RTP, and new transport protocols.

This note assumes a traditional Reno TCP style self clocked, window controlled transport protocol that uses packet loss and ECN CE marks for congestion feedback. There are currently some experimental protocols and congestion control algorithms that are rate based or otherwise fall outside of these assumptions. In the future these new protocols and algorithms may call for revised models.

The MBM approach, mapping Target Transport Performance to a Targeted IP Diagnostic Suite (TIDS) of IP tests, solves some intrinsic problems with using TCP or other throughput maximizing protocols for measurement. In particular all throughput maximizing protocols (and TCP congestion control in particular) cause some level of congestion in order to detect when they have reached the available capacity limitation of the network. This self inflicted congestion obscures the network properties of interest and introduces non-linear dynamic equilibrium behaviors that make any resulting measurements useless as metrics because they have no predictive value for conditions or paths different than that of the measurement itself. In order to prevent these effects it is necessary to avoid the effects of TCP congestion control in the measurement method. These issues are discussed at length in Section 4. Readers whom are unfamiliar with basic properties of TCP and TCP-like congestion control may find it easier to start at Section 4 or Section 4.1.

A Targeted IP Diagnostic Suite does not have such difficulties. IP diagnostics can be constructed such that they make strong statistical statements about path properties that are independent of the measurement details, such as vantage and choice of measurement points.

## 1.1. Version Control

RFC Editor: Please remove this entire subsection prior to publication.

REF Editor: The reference to draft-ietf-tcpm-rack is to attribute an idea. This document should not block waiting for the completion of that one.

Please send comments about this draft to ippm@ietf.org. See http://goo.gl/02tkD for more information including: interim drafts, an up to date todo list and information on contributing.

Formatted: Fri Sep 15 15:07:50 PDT 2017

Changes since -11 draft:

- (From IESG review comments.)
- Ben Campbell: Shorten the Abstract.
- Mirja Kuhlewind: Reduced redundancy. (See message)
- MK: Mention open loop in the introduction.
- MK: Spelled out ECN and reference RFC3168.
- MK: Added a paragraph to the introduction about assuming a traditional self clocked, window controlled transport protocol.
- MK: Added language about initial window to the list at about bursts at the end of section 4.1.
- MK: Network power is defined in the terminology section.
- MK: The introduction mention coordinated activity of both endpoints.
- MK: The security section restates that some of the tests are not intended for frequent monitoring tests as the high load can impact other traffic negatively.
- MK: Restored "Informative References" section name.
- And a few minor nits.

Changes since -10 draft:

- A few more nits from various sources.
- (From IETF LC review comments.)
- David Mandelberg: design metrics to prevent DDOS.
- From Robert Sparks:
    - Remove all legacy 2119 language.
    - Fixed Xr notation inconsistency.
    - Adjusted abstract: tests are only partially specified.

    * Avoid rather than suppress the effects of congestion control
    * Removed the unnecessary, excessively abstract and unclear thought about IP vs TCP measurements.
    * Changed "thwarted" to "not fulfilled".
    * Qualified language about burst models.
    * Replaced "infinitesimal" with other language.
    * Added citations for the reordering strawman.
    * Pointed out that pseudo CBR tests depend on self clock.
    * Fixed some run on sentences.
- Update language to reflect RFC7567, AQM recommendations.
- Suggestion from Merry Mou (MIT)

Changes since -09 draft:

- Five last minute editing nits.

Changes since -08 draft:

- Language, spelling and usage nits.
- Expanded the abstract describe the models.
- Remove superfluous standards like language
- Remove superfluous "future technology" language.
- Interconnects -> network interconnections.
- Added more labels to Figure 1.
- Defined Bulk Transport.
- Clarified "implied bottleneck IP capacity"
- Clarified the history of the BTC metrics.
- Clarified stochastic vs non-stochastic test traffic generation.
- Reworked Fig 2 and 6.1 "Mimicking slowstart"
- Described the unsynchronized parallel stream failure case.
- Discussed how to measure devices that use virtual queues.
- Changed section 8.5.2 (Streaming Media) to be Passive Measurements.

Changes since -07 draft:

- Sharpened the use of "statistical criteria"
- Sharpened the definition of test_window, and removed related redundant text in several places
- Clarified "equilibrium" as "dynamic equilibrium, similar to processes observed in chemistry"
- Properly explained "Heisenberg" as "observer effect"
- Added the observation from RFC 6576 that HW and SW congestion control implementations do not generally give the same results.
- Noted that IP and application metrics differ as to how overhead is handled. MBM is explicit about how it handles overhead.
- Clarified the language and added a new reference about the problems caused by token bucket policers.

- o Added an subsection in the example that comments on some of issues that need to be mentioned in a future usage or applicability doc.
- o Updated ippm-2680-bis to RFC7680
- o Many terminology, punctuation and spelling nits.

Changes since -06 draft:

- o More language nits:

  - * "Targeted IP Diagnostic Suite (TIDS)" replaces "Targeted Diagnostic Suite (TDS)".
  - * "implied bottleneck IP capacity" replaces "implied bottleneck IP rate".
  - * Updated to ECN CE Marks.
  - * Added "specified temporal structure"
  - * "test stream" replaces "test traffic"
  - * "packet transfer" replaces "packet delivery"
  - * Reworked discussion of slowstart, bursts and pacing.
  - * RFC 7567 replaces RFC 2309.

Changes since -05 draft:

- o Wordsmithing on sections overhauled in -05 draft.
- o Reorganized the document:

  - * Relocated subsection "Preconditions".
  - * Relocated subsection "New Requirements relative to RFC 2330".
- o Addressed nits and not so nits by Ruediger Geib. (Thanks!)
- o Substantially tightened the entire definitions section.
- o Many terminology changes, to better conform to other docs :

  - * IP rate and IP capacity (following RFC 5136) replaces various forms of link data rate.
  - * subpath replaces link.
  - * target_window_size replaces target_pipe_size.
  - * implied bottleneck IP rate replaces effective bottleneck link rate.
  - * Packet delivery statistics replaces delivery statistics.

Changes since -04 draft:

- o The introduction was heavily overhauled: split into a separate introduction and overview.
- o The new shorter introduction:

  - * Is a problem statement;
  - * This document provides a framework;
  - * That it replaces TCP measurement by IP tests;

   * That the results are pass/fail.
- Added a diagram of the framework to the overview
- and introduces all of the elements of the framework.
- Renumbered sections, reducing the depth of some section numbers.
- Updated definitions to better agree with other documents:

   * Reordered section 2
   * Bulk [data] performance -> Bulk Transport Capacity, everywhere including the title.
   * loss rate and loss probability -> packet loss ratio
   * end-to-end path -> complete path
   * [end-to-end][target] performance -> Target Transport Performance
   * load test -> capacity test

## 2. Overview

This document describes a modeling framework for deriving a Targeted IP Diagnostic Suite from a predetermined Target Transport Performance. It is not a complete specification, and relies on other standards documents to define important details such as packet Type-P selection, sampling techniques, vantage selection, etc. We imagine Fully Specified - Targeted IP Diagnostic Suites (FS-TIDS), that define all of these details. We use Targeted IP Diagnostic Suite (TIDS) to refer to the subset of such a specification that is in scope for this document. This terminology is defined in Section 3.

Section 4 describes some key aspects of TCP behavior and what they imply about the requirements for IP packet transfer. Most of the IP diagnostic tests needed to confirm that the path meets these properties can be built on existing IPPM metrics, with the addition of statistical criteria for evaluating packet transfer and in a few cases, new mechanisms to implement the required temporal structure. (One group of tests, the standing queue tests described in Section 8.2, don't correspond to existing IPPM metrics, but suitable new IPPM metrics can be patterned after the existing definitions.)

Figure 1 shows the MBM modeling and measurement framework. The Target Transport Performance, at the top of the figure, is determined by the needs of the user or application, outside the scope of this document. For Bulk Transport Capacity, the main performance parameter of interest is the Target Data Rate. However, since TCP's ability to compensate for less than ideal network conditions is fundamentally affected by the Round Trip Time (RTT) and the Maximum Transmission Unit (MTU) of the complete path, these parameters must also be specified in advance based on knowledge about the intended application setting. They may reflect a specific application over a real path through the Internet or an idealized application and hypothetical path representing a typical user community. Section 5 describes the common parameters and models derived from the Target Transport Performance.

```
               Target Transport Performance
     (Target Data Rate, Target RTT and Target MTU)
                            |
                    ________V_________
                    |  mathematical  |
                    |     models     |
                    |                |
                    ------------------
   Traffic parameters |            | Statistical criteria
                      |            |
               _______V____________V____Targeted_______
              |       |   * * *    | Diagnostic Suite  |
         _____|_______V____________V________________   |
       __|____________V____________V______________  |  |
       |           IP diagnostic tests            | |  |
       |              |            |              | |  |
       | _____________V__        __V_____________ | |  |
       | |   traffic    |        |   Delivery  |  | |  |
       | |   pattern    |        |  Evaluation |  | |  |
       | |  generation  |        |             |  | |  |
       | -------v--------        ------^--------- | |  |
       |   |    v    test stream via   ^      |   | |--
       |   |  -->=======================>--   |   | |
       |   |       subpath under test         |   |-
       ----V----------------------------------V---  |
           | |  |                             | |  |
           V V  V                             V V  V
       fail/inconclusive            pass/fail/inconclusive
   (traffic generation status)           (test result)
```

Overall Modeling Framework

Figure 1

Mathematical TCP models are used to determine Traffic parameters and subsequently to design traffic patterns that mimic TCP or other transport protocol delivering bulk data and operating at the Target Data Rate, MTU and RTT over a full range of conditions, including flows that are bursty at multiple time scales. The traffic patterns are generated based on the three Target parameters of complete path and independent of the properties of individual subpaths using the techniques described in Section 6. As much as possible the test streams are generated deterministically (precomputed) to minimize the extent to which test methodology, measurement points, measurement vantage or path partitioning affect the details of the measurement traffic.

Section 7 describes packet transfer statistics and methods to test them against the statistical criteria provided by the mathematical models. Since the statistical criteria typically apply to the complete path (a composition of subpaths) [RFC6049], in situ testing requires that the end-to-end statistical criteria be apportioned as separate criteria for each subpath. Subpaths that are expected to be bottlenecks would then be permitted to contribute a larger fraction of the end-to-end packet loss budget. In compensation, subpaths that are not expected to exhibit bottlenecks must be constrained to contribute less packet loss. Thus the statistical criteria for each subpath in each test of a TIDS is an apportioned share of the end-to-end statistical criteria for the complete path which was determined by the mathematical model.

Section 8 describes the suite of individual tests needed to verify all of required IP delivery properties. A subpath passes if and only if all of the individual IP diagnostic tests pass. Any subpath that fails any test indicates that some users are likely to fail to attain their Target Transport Performance under some conditions. In addition to passing or failing, a test can be deemed to be inconclusive for a number of reasons including: the precomputed traffic pattern was not accurately generated; the measurement results were not statistically significant; and others such as failing to meet some required test preconditions. If all tests pass but some are inconclusive, then the entire suite is deemed to be inconclusive.

In Section 9 we present an example TIDS that might be representative of High Definition (HD) video, and illustrate how Model Based Metrics can be used to address difficult measurement situations, such as confirming that inter-carrier exchanges have sufficient performance and capacity to deliver HD video between ISPs.

Since there is some uncertainty in the modeling process, Section 10 describes a validation procedure to diagnose and minimize false positive and false negative results.

## 3. Terminology

Terms containing underscores (rather than spaces) appear in equations and typically have algorithmic definitions.

General Terminology:

Target: A general term for any parameter specified by or derived from the user's application or transport performance requirements.

Target Transport Performance: Application or transport performance target values for the complete path. For Bulk Transport Capacity defined in this note the Target Transport Performance includes the Target Data Rate, Target RTT and Target MTU as described below.

Target Data Rate: The specified application data rate required for an application's proper operation. Conventional Bulk Transport Capacity (BTC) metrics are focused on the Target Data Rate, however these metrics had little or no predictive value because they do not consider the effects of the other two parameters of the Target Transport Performance, the RTT and MTU of the complete paths.

Target RTT (Round Trip Time): The specified baseline (minimum) RTT of the longest complete path over which the user expects to be able to meet the target performance. TCP and other transport protocol's ability to compensate for path problems is generally proportional to the number of round trips per second. The Target RTT determines both key parameters of the traffic patterns (e.g. burst sizes) and the thresholds on acceptable IP packet transfer statistics. The Target RTT must be specified considering appropriate packets sizes: MTU sized packets on the forward path, ACK sized packets (typically header_overhead) on the return path. Note that Target RTT is specified and not measured, MBM measurements derived for a given target_RTT will be applicable to any path with a smaller RTTs.

Target MTU (Maximum Transmission Unit): The specified maximum MTU supported by the complete path the over which the application expects to meet the target performance. In this document assume a 1500 Byte MTU unless otherwise specified. If some subpath has a smaller MTU, then it becomes the Target MTU for the complete path, and all model calculations and subpath tests must use the same smaller MTU.

Targeted IP Diagnostic Suite (TIDS): A set of IP diagnostic tests designed to determine if an otherwise ideal complete path containing the subpath under test can sustain flows at a specific target_data_rate using target_MTU sized packets when the RTT of the complete path is target_RTT.

Fully Specified Targeted IP Diagnostic Suite (FS-TIDS): A TIDS together with additional specification such as measurement packet type ("type-p" [RFC2330]), etc. which are out of scope for this document, but need to be drawn from other standards documents.

Bulk Transport Capacity: Bulk Transport Capacity Metrics evaluate an Internet path's ability to carry bulk data, such as large files, streaming (non-real time) video, and under some conditions, web images and other content. Prior efforts to define BTC metrics have been based on [RFC3148], which predates our understanding of TCP and the requirements described in Section 4. In general "Bulk Transport" indicates that performance is determined by the interplay between the network, cross traffic and congestion control in the transport protocol. It excludes situations where performance is dominated by the RTT alone (e.g. transactions) or bottlenecks elsewhere, such as in the application itself.

IP diagnostic tests: Measurements or diagnostics to determine if packet transfer statistics meet some precomputed target.

traffic patterns: The temporal patterns or burstiness of traffic generated by applications over transport protocols such as TCP. There are several mechanisms that cause bursts at various time scales as described in Section 4.1. Our goal here is to mimic the range of common patterns (burst sizes and rates, etc), without tying our applicability to specific applications, implementations or technologies, which are sure to become stale.

Explicit Congestion Notification (ECN): See [RFC3168].

packet transfer statistics: Raw, detailed or summary statistics about packet transfer properties of the IP layer including packet losses, ECN Congestion Experienced (CE) marks, reordering, or any other properties that may be germane to transport performance.

packet loss ratio: As defined in [RFC7680].

apportioned: To divide and allocate, for example budgeting packet loss across multiple subpaths such that the losses will accumulate to less than a specified end-to-end loss ratio. Apportioning metrics is essentially the inverse of the process described in [RFC5835].

open loop: A control theory term used to describe a class of techniques where systems that naturally exhibit circular dependencies can be analyzed by suppressing some of the dependencies, such that the resulting dependency graph is acyclic.

Terminology about paths, etc. See [RFC2330] and [RFC7398] for existing terms and definitions.

data sender: Host sending data and receiving ACKs.

data receiver: Host receiving data and sending ACKs.

complete path: The end-to-end path from the data sender to the data receiver.

subpath: A portion of the complete path. Note that there is no requirement that subpaths be non-overlapping. A subpath can be a small as a single device, link or interface.

measurement point: Measurement points as described in [RFC7398].

test path: A path between two measurement points that includes a subpath of the complete path under test. If the measurement points are off path, the test path may include "test leads" between the measurement points and the subpath.

dominant bottleneck: The bottleneck that generally determines most of packet transfer statistics for the entire path. It typically determines a flow’s self clock timing, packet loss and ECN Congestion Experienced (CE) marking rate, with other potential bottlenecks having less effect on the packet transfer statistics. See Section 4.1 on TCP properties.

front path: The subpath from the data sender to the dominant bottleneck.

back path: The subpath from the dominant bottleneck to the receiver.

return path: The path taken by the ACKs from the data receiver to the data sender.

cross traffic: Other, potentially interfering, traffic competing for network resources (bandwidth and/or queue capacity).

Properties determined by the complete path and application. These are described in more detail in Section 5.1.

Application Data Rate: General term for the data rate as seen by the application above the transport layer in bytes per second. This is the payload data rate, and explicitly excludes transport and lower level headers (TCP/IP or other protocols), retransmissions and other overhead that is not part to the total quantity of data delivered to the application.

IP rate: The actual number of IP-layer bytes delivered through a subpath, per unit time, including TCP and IP headers, retransmits and other TCP/IP overhead. Follows from IP-type-P Link Usage [RFC5136].

IP capacity: The maximum number of IP-layer bytes that can be transmitted through a subpath, per unit time, including TCP and IP headers, retransmits and other TCP/IP overhead. Follows from IP-type-P Link Capacity [RFC5136].

bottleneck IP capacity: The IP capacity of the dominant bottleneck in the forward path. All throughput maximizing protocols estimate this capacity by observing the IP rate delivered through the bottleneck. Most protocols derive their self clocks from the timing of this data. See Section 4.1 and Appendix B for more details.

implied bottleneck IP capacity: This is the bottleneck IP capacity implied by the ACKs returning from the receiver. It is determined by looking at how much application data the ACK stream at the sender reports delivered to the data receiver per unit time at various time scales. If the return path is thinning, batching or otherwise altering the ACK timing the implied bottleneck IP capacity over short time scales might be substantially larger than the bottleneck IP capacity averaged over a full RTT. Since TCP derives its clock from the data delivered through the bottleneck, the front path must have sufficient buffering to absorb any data bursts at the dimensions (size and IP rate) implied by the ACK stream, which are potentially doubled during slowstart. If the return path is not altering the ACK stream, then the implied bottleneck IP capacity will be the same as the bottleneck IP capacity. See Section 4.1 and Appendix B for more details.

sender interface rate: The IP rate which corresponds to the IP capacity of the data sender's interface. Due to sender efficiency algorithms including technologies such as TCP segmentation offload (TSO), nearly all modern servers deliver data in bursts at full interface link rate. Today 1 or 10 Gb/s are typical.

Header_overhead: The IP and TCP header sizes, which are the portion of each MTU not available for carrying application payload. Without loss of generality this is assumed to be the size for returning acknowledgments (ACKs). For TCP, the Maximum Segment Size (MSS) is the Target MTU minus the header_overhead.

Basic parameters common to models and subpath tests are defined here are described in more detail in Section 5.2. Note that these are mixed between application transport performance (excludes headers) and IP performance (which include TCP headers and retransmissions as part of the IP payload).

Network power: The observed data rate divided by the observed RTT. Network power indicates how effectively a transport protocol is filling a network.

Window [size]: The total quantity of data carried by packets in-flight plus the data represented by ACKs circulating in the network is referred to as the window. See Section 4.1. Sometimes used with other qualifiers (congestion window, cwnd or receiver window) to indicate which mechanism is controlling the window.

pipe size: A general term for number of packets needed in flight (the window size) to exactly fill some network path or subpath. It corresponds to the window size which maximizes network power. Often used with additional qualifiers to specify which path, or under what conditions, etc.

target_window_size: The average number of packets in flight (the window size) needed to meet the Target Data Rate, for the specified Target RTT, and MTU. It implies the scale of the bursts that the network might experience.

run length: A general term for the observed, measured, or specified number of packets that are (expected to be) delivered between losses or ECN Congestion Experienced (CE) marks. Nominally one over the sum of the loss and ECN CE marking probabilities, if there are independently and identically distributed.

target_run_length: The target_run_length is an estimate of the minimum number of non-congestion marked packets needed between losses or ECN Congestion Experienced (CE) marks necessary to attain the target_data_rate over a path with the specified target_RTT and target_MTU, as computed by a mathematical model of TCP congestion control. A reference calculation is shown in Section 5.2 and alternatives in Appendix A reference target_run_length: target_run_length computed precisely by the method in Section 5.2. This is likely to be slightly more conservative than required by modern TCP implementations.

Ancillary parameters used for some tests:

derating: Under some conditions the standard models are too conservative. The modeling framework permits some latitude in relaxing or "derating" some test parameters as described in Section 5.3 in exchange for a more stringent TIDS validation procedures, described in Section 10. Models can be derated by including a multiplicative derating factor to make tests less stringent.

subpath_IP_capacity: The IP capacity of a specific subpath.

test path: A subpath of a complete path under test.

test_path_RTT: The RTT observed between two measurement points using packet sizes that are consistent with the transport protocol. This is generally MTU sized packets of the forward path, header_overhead sized packets on the return path.

test_path_pipe: The pipe size of a test path. Nominally the test_path_RTT times the test path IP_capacity.

test_window: The smallest window sufficient to meet or exceed the target_rate when operating with a pure self clock over a test path. The test_window is typically given by ceiling(target_data_rate*test_path_RTT/(target_MTU-header_overhead)) but see the discussion in Appendix B about the effects of channel scheduling on RTT. On some test paths the test_window may need to be adjusted slightly to compensate for the RTT being inflated by the devices that schedule packets.

The terminology below is used to define temporal patterns for test stream. These patterns are designed to mimic TCP behavior, as described in Section 4.1.

packet headway: Time interval between packets, specified from the start of one to the start of the next. e.g. If packets are sent with a 1 mS headway, there will be exactly 1000 packets per second.

burst headway: Time interval between bursts, specified from the start of the first packet one burst to the start of the first packet of the next burst. e.g. If 4 packet bursts are sent with a 1 mS burst headway, there will be exactly 4000 packets per second.

paced single packets: Send individual packets at the specified rate or packet headway.

paced bursts: Send bursts on a timer. Specify any 3 of: average data rate, packet size, burst size (number of packets) and burst headway (burst start to start). By default the bursts are assumed to occur at full sender interface rate, such that the packet headway within each burst is the minimum supported by the sender's interface. Under some conditions it is useful to explicitly specify the packet headway within each burst.

slowstart rate: Mimic TCP slowstart by sending 4 packet paced bursts at an average data rate equal to twice the implied bottleneck IP capacity (but not more than the sender interface rate). This is a two level burst pattern described in more detail in Section 6.1. If the implied bottleneck IP capacity is more than half of the sender interface rate, slowstart rate becomes sender interface rate.

slowstart burst: Mimic one round of TCP slowstart by sending a specified number of packets packets in a two level burst pattern that resembles slowstart.

repeated slowstart bursts: Repeat Slowstart bursts once per target_RTT. For TCP each burst would be twice as large as the prior burst, and the sequence would end at the first ECN CE mark or lost packet. For measurement, all slowstart bursts would be the same size (nominally target_window_size but other sizes might be specified), and the ECN CE marks and lost packets are counted.

The tests described in this note can be grouped according to their applicability.

Capacity tests: Capacity tests determine if a network subpath has sufficient capacity to deliver the Target Transport Performance. As long as the test stream is within the proper envelope for the Target Transport Performance, the average packet losses or ECN Congestion Experienced (CE) marks must be below the statistical criteria computed by the model. As such, capacity tests reflect parameters that can transition from passing to failing as a consequence of cross traffic, additional presented load or the actions of other network users. By definition, capacity tests also consume significant network resources (data capacity and/or queue buffer space), and the test schedules must be balanced by their cost.

Monitoring tests: Monitoring tests are designed to capture the most important aspects of a capacity test, but without presenting excessive ongoing load themselves. As such they may miss some details of the network's performance, but can serve as a useful reduced-cost proxy for a capacity test, for example to support continuous production network monitoring.

Engineering tests: Engineering tests evaluate how network algorithms (such as AQM and channel allocation) interact with TCP-style self clocked protocols and adaptive congestion control based on packet loss and ECN Congestion Experienced (CE) marks. These tests are likely to have complicated interactions with cross traffic and under some conditions can be inversely sensitive to load. For example a test to verify that an AQM algorithm causes ECN CE marks or packet drops early enough to limit queue occupancy may experience a false pass result in the presence of cross traffic. It is important that engineering tests be performed under a wide range of conditions, including both in situ and bench testing, and over a wide variety of load conditions. Ongoing monitoring is less likely to be useful for engineering tests, although sparse in situ testing might be appropriate.

## 4. Background

At the time the "Framework for IP Performance Metrics" [RFC2330] was published (1998), sound Bulk Transport Capacity (BTC) measurement was known to be well beyond our capabilities. Even when Framework for Empirical BTC Metrics [RFC3148] was published, we knew that we didn't really understand the problem. Now, by hindsight we understand why assessing BTC is such a hard problem:

- o TCP is a control system with circular dependencies - everything affects performance, including components that are explicitly not part of the test (for example, the host processing power is not in-scope of path performance tests).
- o Congestion control is a dynamic equilibrium process, similar to processes observed in chemistry and other fields. The network and transport protocols find an operating point which balances between opposing forces: the transport protocol pushing harder (raising the data rate and/or window) while the network pushes back (raising packet loss ratio, RTT and/or ECN CE marks). By design TCP congestion control keeps raising the data rate until the network gives some indication that its capacity has been exceeded by dropping packets or adding ECN CE marks. If a TCP sender accurately fills a path to its IP capacity, (e.g. the bottleneck is 100% utilized), then packet losses and ECN CE marks are mostly determined by the TCP sender and how aggressively it seeks additional capacity, and not the network itself, since the network must send exactly the signals that TCP needs to set its rate.
- o TCP's ability to compensate for network impairments (such as loss, delay and delay variation, outside of those caused by TCP itself) is directly proportional to the number of send-ACK round trip exchanges per second (i.e. inversely proportional to the RTT). As a consequence an impaired subpath may pass a short RTT local test even though it fails when the subpath is extended by an effectively perfect network to some larger RTT.
- o TCP has an extreme form of the Observer Effect (colloquially know as the Heisenberg effect). Measurement and cross traffic interact in unknown and ill defined ways. The situation is actually worse than the traditional physics problem where you can at least estimate bounds on the relative momentum of the measurement and measured particles. For network measurement you can not in

general determine even the order of magnitude of the effect. It is possible to construct measurement scenarios where the measurement traffic starves real user traffic, yielding an overly inflated measurement. The inverse is also possible: the user traffic can fill the network, such that the measurement traffic detects only minimal available capacity. You can not in general determine which scenario might be in effect, so you can not gauge the relative magnitude of the uncertainty introduced by interactions with other network traffic.

o As a consequence of the properties listed above it is difficult, if not impossible, for two independent implementations (HW or SW) of TCP congestion control to produce equivalent performance results [RFC6576] under the same network conditions,

These properties are a consequence of the dynamic equilibrium behavior intrinsic to how all throughput maximizing protocols interact with the Internet. These protocols rely on control systems based on estimated network metrics to regulate the quantity of data to send into the network. The packet sending characteristics in turn alter the network properties estimated by the control system metrics, such that there are circular dependencies between every transmission characteristic and every estimated metric. Since some of these dependencies are nonlinear, the entire system is nonlinear, and any change anywhere causes a difficult to predict response in network metrics. As a consequence Bulk Transport Capacity metrics have not fulfilled the analytic framework envisioned in [RFC2330]

Model Based Metrics overcome these problems by making the measurement system open loop: the packet transfer statistics (akin to the network estimators) do not affect the traffic or traffic patterns (bursts), which are computed on the basis of the Target Transport Performance. A path or subpath meeting the Target Transfer Performance requirements would exhibit packet transfer statistics and estimated metrics that would not cause the control system to slow the traffic below the Target Data Rate.

## 4.1. TCP properties

TCP and other self clocked protocols (e.g. SCTP) carry the vast majority of all Internet data. Their dominant bulk data transport behavior is to have an approximately fixed quantity of data and acknowledgments (ACKs) circulating in the network. The data receiver reports arriving data by returning ACKs to the data sender, the data sender typically responds by sending approximately the same quantity of data back into the network. The total quantity of data plus the data represented by ACKs circulating in the network is referred to as the window. The mandatory congestion control algorithms incrementally adjust the window by sending slightly more or less data in response to each ACK. The fundamentally important property of this system is that it is self clocked: The data transmissions are a reflection of the ACKs that were delivered by the network, the ACKs are a reflection of the data arriving from the network.

A number of protocol features cause bursts of data, even in idealized networks that can be modeled as simple queuing systems.

During slowstart the IP rate is doubled on each RTT by sending twice as much data as was delivered to the receiver during the prior RTT. Each returning ACK causes the sender to transmit twice the data the ACK reported arriving at the receiver. For slowstart to be able to fill the pipe, the network must be able to tolerate slowstart bursts up to the full pipe size inflated by the anticipated window reduction on the first loss or ECN CE mark. For example, with classic Reno congestion control, an optimal slowstart has to end with a burst that is twice the bottleneck rate for one RTT in duration. This burst causes a queue which is equal to the pipe size (i.e. the window is twice the pipe size) so when the window is halved in response to the first packet loss, the new window will be the pipe size.

Note that if the bottleneck IP rate is less that half of the capacity of the front path (which is almost always the case), the slowstart bursts will not by themselves cause significant queues anywhere else along the front path; they primarily exercise the queue at the dominant bottleneck.

Several common efficiency algorithms also cause bursts. The self clock is typically applied to groups of packets: the receiver's delayed ACK algorithm generally sends only one ACK per two data segments. Furthermore the modern senders use TCP segmentation offload (TSO) to reduce CPU overhead. The sender's software stack builds super sized TCP segments that the TSO hardware splits into MTU sized segments on the wire. The net effect of TSO, delayed ACK and other efficiency algorithms is to send bursts of segments at full sender interface rate.

Note that these efficiency algorithms are almost always in effect, including during slowstart, such that slowstart typically has a two level burst structure. Section 6.1 describes slowstart in more detail.

Additional sources of bursts include TCP's initial window [RFC6928], application pauses, channel allocation mechanisms and network devices that schedule ACKs. Appendix B describes these last two items. If the application pauses (stops reading or writing data) for some fraction of an RTT, many TCP implementations catch up to their earlier window size by sending a burst of data at the full sender interface rate. To fill a network with a realistic application, the network has to be able to tolerate sender interface rate bursts large enough to restore the prior window following application pauses.

Although the sender interface rate bursts are typically smaller than the last burst of a slowstart, they are at a higher IP rate so they potentially exercise queues at arbitrary points along the front path from the data sender up to and including the queue at the dominant bottleneck. It is known that these bursts can hurt network performance, especially in conjunction with other queue pressure, however we are not aware of any models for how frequent sender rate bursts the network should be able to tolerate at various burst sizes.

In conclusion, to verify that a path can meet a Target Transport Performance, it is necessary to independently confirm that the path can tolerate bursts at the scales that can be caused by the above mechanisms. Three cases are believed to be sufficient:

- o Two level slowstart bursts sufficient to get connections started properly.
- o Ubiquitous sender interface rate bursts caused by efficiency algorithms. We assume 4 packet bursts to be the most common case, since it matches the effects of delayed ACK during slowstart. These bursts should be assumed not to significantly affect packet transfer statistics.
- o Infrequent sender interface rate bursts that are the maximum of the full target_window_size and the initial window size (10 segments in [RFC6928]). The Target_run_length may be derated for these large fast bursts.

If a subpath can meet the required packet loss ratio for bursts at all of these scales then it has sufficient buffering at all potential bottlenecks to tolerate any of the bursts that are likely introduced by TCP or other transport protocols.

## 4.2. Diagnostic Approach

A complete path of a given RTT and MTU, which are equal to or smaller than the Target RTT and equal to or larger than the Target MTU respectively, is expected to be able to attain a specified Bulk Transport Capacity when all of the following conditions are met:

1. The IP capacity is above the Target Data Rate by sufficient margin to cover all TCP/IP overheads. This can be confirmed by the tests described in Section 8.1 or any number of IP capacity tests adapted to implement MBM.
2. The observed packet transfer statistics are better than required by a suitable TCP performance model (e.g. fewer packet losses or

ECN CE marks). See Section 8.1 or any number of low or fixed rate packet loss tests outside of MBM.
3. There is sufficient buffering at the dominant bottleneck to absorb a slowstart bursts large enough to get the flow out of slowstart at a suitable window size. See Section 8.3.
4. There is sufficient buffering in the front path to absorb and smooth sender interface rate bursts at all scales that are likely to be generated by the application, any channel arbitration in the ACK path or any other mechanisms. See Section 8.4.
5. When there is a slowly rising standing queue at the bottleneck the onset of packet loss has to be at an appropriate point (time or queue depth) and progressive [RFC7567]. See Section 8.2.
6. When there is a standing queue at a bottleneck for a shared media subpath (e.g. half duplex), there must be a suitable bounds on the interaction between ACKs and data, for example due to the channel arbitration mechanism. See Section 8.2.4.

Note that conditions 1 through 4 require capacity tests for validation, and thus may need to be monitored on an ongoing basis. Conditions 5 and 6 require engineering tests, which are best performed in controlled environments such as a bench test. They won’t generally fail due to load, but may fail in the field due to configuration errors, etc. and should be spot checked.

A tool that can perform many of the tests is available from [MBMSource].

## 4.3. New requirements relative to RFC 2330

Model Based Metrics are designed to fulfill some additional requirements that were not recognized at the time RFC 2330 was written [RFC2330]. These missing requirements may have significantly contributed to policy difficulties in the IP measurement space. Some additional requirements are:

- o IP metrics must be actionable by the ISP - they have to be interpreted in terms of behaviors or properties at the IP or lower layers, that an ISP can test, repair and verify.
- o Metrics should be spatially composable, such that measures of concatenated paths should be predictable from subpaths.
- o Metrics must be vantage point invariant over a significant range of measurement point choices, including off path measurement points. The only requirements on MP selection should be that the RTT between the MPs is below some reasonable bound, and that the effects of the "test leads" connecting MPs to the subpath under test can be can be calibrated out of the measurements. The latter might be be accomplished if the test leads are effectively ideal or their properties can be deducted from the measurements between

the MPs. While many of tests require that the test leads have at least as much IP capacity as the subpath under test, some do not, for example Background Packet Transfer Tests described in Section 8.1.3.

- o Metric measurements should be repeatable by multiple parties with no specialized access to MPs or diagnostic infrastructure. It should be possible for different parties to make the same measurement and observe the same results. In particular it is specifically important that both a consumer (or their delegate) and ISP be able to perform the same measurement and get the same result. Note that vantage independence is key to meeting this requirement.

# 5. Common Models and Parameters

## 5.1. Target End-to-end parameters

The target end-to-end parameters are the Target Data Rate, Target RTT and Target MTU as defined in Section 3. These parameters are determined by the needs of the application or the ultimate end user and the complete Internet path over which the application is expected to operate. The target parameters are in units that make sense to upper layers: payload bytes delivered to the application, above TCP. They exclude overheads associated with TCP and IP headers, retransmits and other protocols (e.g. DNS). Note that IP-based network services include TCP headers and retransmissions as part of delivered payload, and this difference is recognized in calculations below (header_overhead).

Other end-to-end parameters defined in Section 3 include the effective bottleneck data rate, the sender interface data rate and the TCP and IP header sizes.

The target_data_rate must be smaller than all subpath IP capacities by enough headroom to carry the transport protocol overhead, explicitly including retransmissions and an allowance for fluctuations in TCP's actual data rate. Specifying a target_data_rate with insufficient headroom is likely to result in brittle measurements having little predictive value.

Note that the target parameters can be specified for a hypothetical path, for example to construct TIDS designed for bench testing in the absence of a real application; or for a live in situ test of production infrastructure.

The number of concurrent connections is explicitly not a parameter to this model. If a subpath requires multiple connections in order to meet the specified performance, that must be stated explicitly and the procedure described in Section 6.4 applies.

## 5.2. Common Model Calculations

The Target Transport Performance is used to derive the target_window_size and the reference target_run_length.

The target_window_size, is the average window size in packets needed to meet the target_rate, for the specified target_RTT and target_MTU. It is given by:

target_window_size = ceiling( target_rate * target_RTT / ( target_MTU - header_overhead ) )

Target_run_length is an estimate of the minimum required number of unmarked packets that must be delivered between losses or ECN Congestion Experienced (CE) marks, as computed by a mathematical model of TCP congestion control. The derivation here follows [MSMO97], and by design is quite conservative.

Reference target_run_length is derived as follows: assume the subpath_IP_capacity is infinitesimally larger than the target_data_rate plus the required header_overhead. Then target_window_size also predicts the onset of queuing. A larger window will cause a standing queue at the bottleneck.

Assume the transport protocol is using standard Reno style Additive Increase, Multiplicative Decrease (AIMD) congestion control [RFC5681] (but not Appropriate Byte Counting [RFC3465]) and the receiver is using standard delayed ACKs. Reno increases the window by one packet every pipe_size worth of ACKs. With delayed ACKs this takes 2 Round Trip Times per increase. To exactly fill the pipe, the spacing of losses must be no closer than when the peak of the AIMD sawtooth reached exactly twice the target_window_size. Otherwise, the multiplicative window reduction triggered by the loss would cause the network to be under-filled. Following [MSMO97] the number of packets between losses must be the area under the AIMD sawtooth. They must be no more frequent than every 1 in ((3/2)*target_window_size)*(2*target_window_size) packets, which simplifies to:

target_run_length = 3*(target_window_size^2)

Note that this calculation is very conservative and is based on a number of assumptions that may not apply. Appendix A discusses these assumptions and provides some alternative models. If a different model is used, a FS-TIDS must document the actual method for computing target_run_length and ratio between alternate target_run_length and the reference target_run_length calculated above, along with a discussion of the rationale for the underlying assumptions.

These two parameters, target_window_size and target_run_length, directly imply most of the individual parameters for the tests in Section 8.

## 5.3. Parameter Derating

Since some aspects of the models are very conservative, the MBM framework permits some latitude in derating test parameters. Rather than trying to formalize more complicated models we permit some test parameters to be relaxed as long as they meet some additional procedural constraints:

- o The FS-TIDS must document and justify the actual method used to compute the derated metric parameters.
- o The validation procedures described in Section 10 must be used to demonstrate the feasibility of meeting the Target Transport Performance with infrastructure that just barely passes the derated tests.
- o The validation process for a FS-TIDS itself must be documented is such a way that other researchers can duplicate the validation experiments.

Except as noted, all tests below assume no derating. Tests where there is not currently a well established model for the required parameters explicitly include derating as a way to indicate flexibility in the parameters.

## 5.4. Test Preconditions

Many tests have preconditions which are required to assure their validity. Examples include: the presence or non-presence of cross traffic on specific subpaths; negotiating ECN; and appropriate preamble packet stream to testing to put reactive network elements into the proper states [RFC7312]. If preconditions are not properly satisfied for some reason, the tests should be considered to be inconclusive. In general it is useful to preserve diagnostic information as to why the preconditions were not met, and any test data that was collected even if it is not useful for the intended test. Such diagnostic information and partial test data may be useful for improving the test or test procedures themselves.

It is important to preserve the record that a test was scheduled, because otherwise precondition enforcement mechanisms can introduce sampling bias. For example, canceling tests due to cross traffic on subscriber access links might introduce sampling bias in tests of the rest of the network by reducing the number of tests during peak network load.

Test preconditions and failure actions must be specified in a FS-TIDS.

## 6. Generating test streams

Many important properties of Model Based Metrics, such as vantage independence, are a consequence of using test streams that have temporal structures that mimic TCP or other transport protocols running over a complete path. As described in Section 4.1, self clocked protocols naturally have burst structures related to the RTT and pipe size of the complete path. These bursts naturally get larger (contain more packets) as either the Target RTT or Target Data Rate get larger, or the Target MTU gets smaller. An implication of these relationships is that test streams generated by running self clocked protocols over short subpaths may not adequately exercise the queuing at any bottleneck to determine if the subpath can support the full Target Transport Performance over the complete path.

Failing to authentically mimic TCP's temporal structure is part of the reason why simple performance tools such as iPerf, netperf, nc, etc have the reputation of yielding false pass results over short test paths, even when some subpath has a flaw.

The definitions in Section 3 are sufficient for most test streams. We describe the slowstart and standing queue test streams in more detail.

In conventional measurement practice stochastic processes are used to eliminate many unintended correlations and sample biases. However MBM tests are designed to explicitly mimic temporal correlations caused by network or protocol elements themselves. Some portions of these systems, such as traffic arrival (test scheduling) are naturally stochastic. Other behaviors, such as back-to-back packet transmissions, are dominated by implementation specific deterministic effects. Although these behaviors always contain non-deterministic elements and might be modeled stochastically, these details typically do not contribute significantly to the overall system behavior. Furthermore, it is known that real protocols are subject to failures caused by network property estimators suffering from bias due to correlation in their own traffic. For example TCP's RTT estimator used to determine the Retransmit Time Out (RTO), can be fooled by periodic cross traffic or start-stop applications. For these reasons many details of the test streams are specified deterministically.

It may prove useful to introduce fine grained noise sources into the models used for generating test streams in an update of Model Based Metrics, but the complexity is not warranted at the time this document was written.

### 6.1. Mimicking slowstart

TCP slowstart has a two level burst structure as shown in Figure 2. The fine time structure is caused by efficiency algorithms that deliberately batch work (CPU, channel allocation, etc) to better amortize certain network and host overheads. ACKs passing through the return path typically cause the sender to transmit small bursts of data at full sender interface rate. For example TCP Segmentation Offload (TSO) and Delayed Acknowledgment both contribute to this effect. During slowstart these bursts are at the same headway as the returning ACKs, but are typically twice as large (e.g. having twice as much data) as the ACK reported was delivered to the receiver. Due to variations in delayed ACK and algorithms such as Appropriate Byte Counting [RFC3465], different pairs of senders and receivers produce slightly different burst patterns. Without loss of generality, we assume each ACK causes 4 packet sender interface rate bursts at an average headway equal to the ACK headway, and corresponding to sending at an average rate equal to twice the effective bottleneck IP rate. Each slowstart burst consists of a series of 4 packet sender interface rate bursts such that the total number of packets is the current window size (as of the last packet in the burst).

The coarse time structure is due to each RTT being a reflection of the prior RTT. For real transport protocols, each slowstart burst is twice as large (twice the window) as the previous burst but is spread out in time by the network bottleneck, such that each successive RTT exhibits the same effective bottleneck IP rate. The slowstart phase ends on the first lost packet or ECN mark, which is intended to happen after successive slowstart bursts merge in time: the next burst starts before the bottleneck queue is fully drained and the prior burst is complete.

For diagnostic tests described below we preserve the fine time structure but manipulate the coarse structure of the slowstart bursts (burst size and headway) to measure the ability of the dominant bottleneck to absorb and smooth slowstart bursts.

Note that a stream of repeated slowstart bursts has three different average rates, depending on the averaging time interval. At the finest time scale (a few packet times at the sender interface) the peak of the average IP rate is the same as the sender interface rate; at a medium timescale (a few ACK times at the dominant bottleneck) the peak of the average IP rate is twice the implied bottleneck IP capacity; and at time scales longer than the target_RTT and when the burst size is equal to the target_window_size, the average rate is equal to the target_data_rate. This pattern corresponds to repeating the last RTT of TCP slowstart when delayed ACK and sender side byte counting are present but without the limits specified in Appropriate Byte Counting [RFC3465].

```
time ==>    ( - equals one packet)

Fine time structure of the packet stream:

----  ----  ----  ----  ----

|<>| sender interface rate bursts (typically 3 or 4 packets)
|<===>| burst headway (from the ACK headway)

\____repeating sender______/
       rate bursts

Coarse (RTT level) time structure of the packet stream:

----  ----  ----  ----  ----                     ----  ----  ...

|<========================>| slowstart burst size (from the window)
|<==============================================>| slowstart headway
                                                    (from the RTT)
\__________________________/                     \__________ ...
    one slowstart burst                     Repeated slowstart bursts

Multiple levels of Slowstart Bursts
```

Figure 2

## 6.2. Constant window pseudo CBR

Implement pseudo constant bit rate by running a standard self clocked protocol such as TCP with a fixed window size. If that window size is test_window, the data rate will be slightly above the target_rate.

Since the test_window is constrained to be an integer number of packets, for small RTTs or low data rates there may not be sufficiently precise control over the data rate. Rounding the test_window up (as defined above) is likely to result in data rates that are higher than the target rate, but reducing the window by one packet may result in data rates that are too small. Also cross traffic potentially raises the RTT, implicitly reducing the rate.

Cross traffic that raises the RTT nearly always makes the test more strenuous (more demanding for the network path).

Note that Constant window pseudo CBR (and Scanned window pseudo CBR in the next section) both rely on a self clock which is at least partially derived from the properties of the subnet under test. This introduces the possibility that the subnet under test exhibits behaviors such as extreme RTT fluctuations that prevent these algorithms from accurately controlling data rates.

A FS-TIDS specifying a constant window CBR test must explicitly indicate under what conditions errors in the data rate cause tests to be inconclusive. Conventional paced measurement traffic may be more appropriate for these environments.

## 6.3. Scanned window pseudo CBR

Scanned window pseudo CBR is similar to the constant window CBR described above, except the window is scanned across a range of sizes designed to include two key events, the onset of queuing and the onset of packet loss or ECN CE marks. The window is scanned by incrementing it by one packet every 2*target_window_size delivered packets. This mimics the additive increase phase of standard Reno TCP congestion avoidance when delayed ACKs are in effect. Normally the window increases separated by intervals slightly longer than twice the target_RTT.

There are two ways to implement this test: one built by applying a window clamp to standard congestion control in a standard protocol such as TCP and the other built by stiffening a non-standard transport protocol. When standard congestion control is in effect, any losses or ECN CE marks cause the transport to revert to a window smaller than the clamp such that the scanning clamp loses control the window size. The NPAD pathdiag tool is an example of this class of algorithms [Pathdiag].

Alternatively a non-standard congestion control algorithm can respond to losses by transmitting extra data, such that it maintains the specified window size independent of losses or ECN CE marks. Such a stiffened transport explicitly violates mandatory Internet congestion control [RFC5681] and is not suitable for in situ testing. It is only appropriate for engineering testing under laboratory conditions. The Windowed Ping tool implements such a test [WPING]. The tool described in the paper has been updated.[mpingSource]

The test procedures in Section 8.2 describe how to the partition the scans into regions and how to interpret the results.

## 6.4. Concurrent or channelized testing

The procedures described in this document are only directly applicable to single stream measurement, e.g. one TCP connection or measurement stream. In an ideal world, we would disallow all performance claims based multiple concurrent streams, but this is not practical due to at least two issues. First, many very high rate link technologies are channelized and at last partially pin the flow to channel mapping to minimize packet reordering within flows. Second, TCP itself has scaling limits. Although the former problem might be overcome through different design decisions, the later problem is more deeply rooted.

All congestion control algorithms that are philosophically aligned with the standard [RFC5681] (e.g. claim some level of TCP compatibility, friendliness or fairness) have scaling limits, in the sense that as a long fast network (LFN) with a fixed RTT and MTU gets faster, these congestion control algorithms get less accurate and as a consequence have difficulty filling the network [CCscaling]. These properties are a consequence of the original Reno AIMD congestion control design and the requirement in [RFC5681] that all transport protocols have similar responses to congestion.

There are a number of reasons to want to specify performance in terms of multiple concurrent flows, however this approach is not recommended for data rates below several megabits per second, which can be attained with run lengths under 10000 packets on many paths. Since the required run length goes as the square of the data rate, at higher rates the run lengths can be unreasonably large, and multiple flows might be the only feasible approach.

If multiple flows are deemed necessary to meet aggregate performance targets then this must be stated in both the design of the TIDS and in any claims about network performance. The IP diagnostic tests must be performed concurrently with the specified number of connections. For the tests that use bursty test streams, the bursts should be synchronized across streams unless there is a priori knowledge that the applications have some explicit mechanism to stagger their own bursts. In the absences of an explicit mechanism to stagger bursts many network and application artifacts will sometimes implicitly synchronize bursts. A test that does not control burst synchronization may be prone to false pass results for some applications.

# 7. Interpreting the Results

## 7.1. Test outcomes

To perform an exhaustive test of a complete network path, each test of the TIDS is applied to each subpath of the complete path. If any subpath fails any test then a standard transport protocol running over the complete path can also be expected to fail to attain the Target Transport Performance under some conditions.

In addition to passing or failing, a test can be deemed to be inconclusive for a number of reasons. Proper instrumentation and treatment of inconclusive outcomes is critical to the accuracy and robustness of Model Based Metrics. Tests can be inconclusive if the precomputed traffic pattern or data rates were not accurately generated; the measurement results were not statistically significant; and others causes such as failing to meet some required preconditions for the test. See Section 5.4

For example consider a test that implements Constant Window Pseudo CBR (Section 6.2) by adding rate controls and detailed IP packet transfer instrumentation to TCP (e.g. [RFC4898]). TCP includes built in control systems which might interfere with the sending data rate. If such a test meets the required packet transfer statistics (e.g. run length) while failing to attain the specified data rate it must be treated as an inconclusive result, because we can not a priori determine if the reduced data rate was caused by a TCP problem or a network problem, or if the reduced data rate had a material effect on the observed packet transfer statistics.

Note that for capacity tests, if the observed packet transfer statistics meet the statistical criteria for failing (accepting hypnosis H1 in Section 7.2), the test can can be considered to have failed because it doesn't really matter that the test didn't attain the required data rate.

The really important new properties of MBM, such as vantage independence, are a direct consequence of opening the control loops in the protocols, such that the test stream does not depend on network conditions or IP packets received. Any mechanism that introduces feedback between the path's measurements and the test stream generation is at risk of introducing nonlinearities that spoil these properties. Any exceptional event that indicates that such feedback has happened should cause the test to be considered inconclusive.

One way to view inconclusive tests is that they reflect situations where a test outcome is ambiguous between limitations of the network and some unknown limitation of the IP diagnostic test itself, which may have been caused by some uncontrolled feedback from the network.

Note that procedures that attempt to search the target parameter space to find the limits on some parameter such as target_data_rate are at risk of breaking the location independent properties of Model Based Metrics, if any part of the boundary between passing and inconclusive or failing results is sensitive to RTT (which is normally the case). For example the maximum data rate for a marginal link (e.g. exhibiting excess errors) is likely to be sensitive to the test_path_RTT. The maximum observed data rate over the test path has very little value for predicting the maximum rate over a different path.

One of the goals for evolving TIDS designs will be to keep sharpening distinction between inconclusive, passing and failing tests. The criteria for for passing, failing and inconclusive tests must be explicitly stated for every test in the TIDS or FS-TIDS.

One of the goals of evolving the testing process, procedures, tools and measurement point selection should be to minimize the number of inconclusive tests.

It may be useful to keep raw packet transfer statistics and ancillary metrics [RFC3148] for deeper study of the behavior of the network path and to measure the tools themselves. Raw packet transfer statistics can help to drive tool evolution. Under some conditions it might be possible to re-evaluate the raw data for satisfying alternate Target Transport Performance. However it is important to guard against sampling bias and other implicit feedback which can cause false results and exhibit measurement point vantage sensitivity. Simply applying different delivery criteria based on a different Target Transport Performance is insufficient if the test traffic patterns (bursts, etc.) does not match the alternate Target Transport Performance.

## 7.2. Statistical criteria for estimating run_length

When evaluating the observed run_length, we need to determine appropriate packet stream sizes and acceptable error levels for efficient measurement. In practice, can we compare the empirically estimated packet loss and ECN Congestion Experienced (CE) marking ratios with the targets as the sample size grows? How large a sample is needed to say that the measurements of packet transfer indicate a particular run length is present?

The generalized measurement can be described as recursive testing: send packets (individually or in patterns) and observe the packet transfer performance (packet loss ratio or other metric, any marking we define).

As each packet is sent and measured, we have an ongoing estimate of the performance in terms of the ratio of packet loss or ECN CE mark to total packets (i.e. an empirical probability). We continue to send until conditions support a conclusion or a maximum sending limit has been reached.

We have a target_mark_probability, 1 mark per target_run_length, where a "mark" is defined as a lost packet, a packet with ECN CE mark, or other signal. This constitutes the null Hypothesis:

H0: no more than one mark in target_run_length =
   3*(target_window_size)^2 packets

and we can stop sending packets if on-going measurements support accepting H0 with the specified Type I error = alpha (= 0.05 for example).

We also have an alternative Hypothesis to evaluate: if performance is significantly lower than the target_mark_probability. Based on analysis of typical values and practical limits on measurement duration, we choose four times the H0 probability:

H1: one or more marks in (target_run_length/4) packets

and we can stop sending packets if measurements support rejecting H0 with the specified Type II error = beta (= 0.05 for example), thus preferring the alternate hypothesis H1.

H0 and H1 constitute the Success and Failure outcomes described elsewhere in the memo, and while the ongoing measurements do not support either hypothesis the current status of measurements is inconclusive.

The problem above is formulated to match the Sequential Probability Ratio Test (SPRT) [Wald45] and [Montgomery90]. Note that as originally framed the events under consideration were all manufacturing defects. In networking, ECN CE marks and lost packets are not defects but signals, indicating that the transport protocol should slow down.

The Sequential Probability Ratio Test also starts with a pair of hypothesis specified as above:

H0: p0 = one defect in target_run_length
H1: p1 = one defect in target_run_length/4

As packets are sent and measurements collected, the tester evaluates the cumulative defect count against two boundaries representing H0 Acceptance or Rejection (and acceptance of H1):

```
Acceptance line:  Xa = -h1 + s*n
Rejection line:  Xr = h2 + s*n
```

where n increases linearly for each packet sent and

```
h1 =  { log((1-alpha)/beta) }/k
h2 =  { log((1-beta)/alpha) }/k
k  =  log{ (p1(1-p0)) / (p0(1-p1)) }
s  =  [ log{ (1-p0)/(1-p1) } ]/k
```

for p0 and p1 as defined in the null and alternative Hypotheses statements above, and alpha and beta as the Type I and Type II errors.

The SPRT specifies simple stopping rules:

o  Xa < defect_count(n) < Xr: continue testing
o  defect_count(n) <= Xa: Accept H0
o  defect_count(n) >= Xr: Accept H1

The calculations above are implemented in the R-tool for Statistical Analysis [Rtool] , in the add-on package for Cross-Validation via Sequential Testing (CVST) [CVST].

Using the equations above, we can calculate the minimum number of packets (n) needed to accept H0 when x defects are observed.  For example, when x = 0:

```
Xa = 0  = -h1 + s*n
and  n = h1 / s
```

Note that the derivations in [Wald45] and [Montgomery90] differ. Montgomery's simplified derivation of SPRT may assume a Bernoulli processes, where the packet loss probabilities are independent and identically distributed, making the SPRT more accessible.  Wald's seminal paper showed that this assumption is not necessary.  It helps to remember that the goal of SPRT is not to estimate the value of the packet loss rate, but only whether or not the packet loss ratio is likely low enough (when we accept the H0 null hypothesis) yielding success; or too high (when we accept the H1 alternate hypothesis) yielding failure.

## 7.3. Reordering Tolerance

All tests must be instrumented for packet level reordering [RFC4737]. However, there is no consensus for how much reordering should be acceptable. Over the last two decades the general trend has been to make protocols and applications more tolerant to reordering (see for example [RFC4015]), in response to the gradual increase in reordering in the network. This increase has been due to the deployment of technologies such as multithreaded routing lookups and Equal Cost MultiPath (ECMP) routing. These techniques increase parallelism in network and are critical to enabling overall Internet growth to exceed Moore’s Law.

Note that transport retransmission strategies can trade off reordering tolerance vs how quickly they can repair losses vs overhead from spurious retransmissions. In advance of new retransmission strategies we propose the following strawman: Transport protocols should be able to adapt to reordering as long as the reordering extent is not more than the maximum of one quarter window or 1 mS, whichever is larger. (These values come from experience prototyping Early Retransmit [RFC5827] and related algorithms. They agree with the values being proposed for "RACK: a time-based fast loss detection algorithm" [I-D.ietf-tcpm-rack].) Within this limit on reorder extent, there should be no bound on reordering density.

By implication, recording which is less than these bounds should not be treated as a network impairment. However [RFC4737] still applies: reordering should be instrumented and the maximum reordering that can be properly characterized by the test (because of the bound on history buffers) should be recorded with the measurement results.

Reordering tolerance and diagnostic limitations, such as the size of the history buffer used to diagnose packets that are way out-of-order, must be specified in a FSTIDS.

# 8. IP Diagnostic Tests

The IP diagnostic tests below are organized according to the technique used to generate the test stream as described in Section 6. All of the results are evaluated in accordance with Section 7, possibly with additional test specific critera.

We also introduce some combined tests which are more efficient when networks are expected to pass, but conflate diagnostic signatures when they fail.

### 8.1. Basic Data Rate and Packet Transfer Tests

We propose several versions of the basic data rate and packet transfer statistics test that differ in how the data rate is controlled. The data can be paced on a timer, or window controlled (and self clocked). The first two tests implicitly confirm that sub_path has sufficient raw capacity to carry the target_data_rate. They are recommended for relatively infrequent testing, such as an installation or periodic auditing process. The third, background packet transfer statistics, is a low rate test designed for ongoing monitoring for changes in subpath quality.

#### 8.1.1. Delivery Statistics at Paced Full Data Rate

Confirm that the observed run length is at least the target_run_length while relying on timer to send data at the target_rate using the procedure described in in Section 6.1 with a burst size of 1 (single packets) or 2 (packet pairs).

The test is considered to be inconclusive if the packet transmission can not be accurately controlled for any reason.

RFC 6673 [RFC6673] is appropriate for measuring packet transfer statistics at full data rate.

#### 8.1.2. Delivery Statistics at Full Data Windowed Rate

Confirm that the observed run length is at least the target_run_length while sending at an average rate approximately equal to the target_data_rate, by controlling (or clamping) the window size of a conventional transport protocol to test_window.

Since losses and ECN CE marks cause transport protocols to reduce their data rates, this test is expected to be less precise about controlling its data rate. It should not be considered inconclusive as long as at least some of the round trips reached the full target_data_rate without incurring losses or ECN CE marks. To pass this test the network must deliver target_window_size packets in target_RTT time without any losses or ECN CE marks at least once per two target_window_size round trips, in addition to meeting the run length statistical test.

#### 8.1.3. Background Packet Transfer Statistics Tests

The background run length is a low rate version of the target target rate test above, designed for ongoing lightweight monitoring for changes in the observed subpath run length without disrupting users. It should be used in conjunction with one of the above full rate tests because it does not confirm that the subpath can support raw data rate.

RFC 6673 [RFC6673] is appropriate for measuring background packet transfer statistics.

## 8.2. Standing Queue Tests

These engineering tests confirm that the bottleneck is well behaved across the onset of packet loss, which typically follows after the onset of queuing. Well behaved generally means lossless for transient queues, but once the queue has been sustained for a sufficient period of time (or reaches a sufficient queue depth) there should be a small number of losses or ECN CE marks to signal to the transport protocol that it should reduce its window or data rate. Losses that are too early can prevent the transport from averaging at the target_data_rate. Losses that are too late indicate that the queue might not have an appropriate AQM [RFC7567] and as a consequence subject to bufferbloat [wikiBloat]. Queues without AQM have the potential to inflict excess delays on all flows sharing the bottleneck. Excess losses (more than half of the window) at the onset of loss make loss recovery problematic for the transport protocol. Non-linear, erratic or excessive RTT increases suggest poor interactions between the channel acquisition algorithms and the transport self clock. All of the tests in this section use the same basic scanning algorithm, described here, but score the link or subpath on the basis of how well it avoids each of these problems.

Some network technologies rely on virtual queues or other techniques to meter traffic without adding any queuing delay, in which case the data rate will vary with the window size all the way up to the onset of load induced packet loss or ECN CE marks. For these technologies, the discussion of queuing in Section 6.3 does not apply, but it is still necessary to confirm that the onset of losses or ECN CE marks be at an appropriate point and progressive. If the network bottleneck does not introduce significant queuing delay, modify the procedure described in Section 6.3 to start the scan at a window equal to or slightly smaller than the test_window.

Use the procedure in Section 6.3 to sweep the window across the onset of queuing and the onset of loss. The tests below all assume that the scan emulates standard additive increase and delayed ACK by incrementing the window by one packet for every 2*target_window_size packets delivered. A scan can typically be divided into three regions: below the onset of queuing, a standing queue, and at or beyond the onset of loss.

Below the onset of queuing the RTT is typically fairly constant, and the data rate varies in proportion to the window size. Once the data rate reaches the subpath IP rate, the data rate becomes fairly constant, and the RTT increases in proportion to the increase in window size. The precise transition across the start of queuing can be identified by the maximum network power, defined to be the ratio data rate over the RTT. The network power can be computed at each window size, and the window with the maximum is taken as the start of the queuing region.

If there is random background loss (e.g. bit errors, etc), precise determination of the onset of queue induced packet loss may require multiple scans. Above the onset of queuing loss, all transport protocols are expected to experience periodic losses determined by the interaction between the congestion control and AQM algorithms. For standard congestion control algorithms the periodic losses are likely to be relatively widely spaced and the details are typically dominated by the behavior of the transport protocol itself. For the stiffened transport protocols case (with non-standard, aggressive congestion control algorithms) the details of periodic losses will be dominated by how the window increase function responds to loss.

### 8.2.1. Congestion Avoidance

A subpath passes the congestion avoidance standing queue test if more than target_run_length packets are delivered between the onset of queuing (as determined by the window with the maximum network power as described above) and the first loss or ECN CE mark. If this test is implemented using a standard congestion control algorithm with a clamp, it can be performed in situ in the production internet as a capacity test. For an example of such a test see [Pathdiag].

For technologies that do not have conventional queues, use the test_window in place of the onset of queuing. i.e. A subpath passes the congestion avoidance standing queue test if more than target_run_length packets are delivered between start of the scan at test_window and the first loss or ECN CE mark.

### 8.2.2. Bufferbloat

This test confirms that there is some mechanism to limit buffer occupancy (e.g. that prevents bufferbloat). Note that this is not strictly a requirement for single stream bulk transport capacity, however if there is no mechanism to limit buffer queue occupancy then a single stream with sufficient data to deliver is likely to cause the problems described in [RFC7567], and [wikiBloat]. This may cause only minor symptoms for the dominant flow, but has the potential to make the subpath unusable for other flows and applications.

Pass if the onset of loss occurs before a standing queue has introduced more delay than than twice target_RTT, or other well defined and specified limit. Note that there is not yet a model for how much standing queue is acceptable. The factor of two chosen here reflects a rule of thumb. In conjunction with the previous test, this test implies that the first loss should occur at a queuing delay which is between one and two times the target_RTT.

Specified RTT limits that are larger than twice the target_RTT must be fully justified in the FS-TIDS.

#### 8.2.3. Non excessive loss

This test confirms that the onset of loss is not excessive. Pass if losses are equal or less than the increase in the cross traffic plus the test stream window increase since the previous RTT. This could be restated as non-decreasing total throughput of the subpath at the onset of loss. (Note that when there is a transient drop in subpath throughput and there is not already a standing queue, a subpath that passes other queue tests in this document will have sufficient queue space to hold one full RTT worth of data).

Note that token bucket policers will not pass this test, which is as intended. TCP often stumbles badly if more than a small fraction of the packets are dropped in one RTT. Many TCP implementations will require a timeout and slowstart to recover their self clock. Even if they can recover from the massive losses the sudden change in available capacity at the bottleneck wastes serving and front path capacity until TCP can adapt to the new rate [Policing].

#### 8.2.4. Duplex Self Interference

This engineering test confirms a bound on the interactions between the forward data path and the ACK return path when they share a half duplex link.

Some historical half duplex technologies had the property that each direction held the channel until it completely drained its queue. When a self clocked transport protocol, such as TCP, has data and ACKs passing in opposite directions through such a link, the behavior often reverts to stop-and-wait. Each additional packet added to the window raises the observed RTT by two packet times, once as the additional packet passes through the data path, and once for the additional delay incurred by the ACK waiting on the return path.

The duplex self interference test fails if the RTT rises by more than a fixed bound above the expected queuing time computed from the excess window divided by the subpath IP Capacity. This bound must be smaller than target_RTT/2 to avoid reverting to stop and wait behavior. (e.g. Data packets and ACKs both have to be released at least twice per RTT.)

### 8.3. Slowstart tests

These tests mimic slowstart: data is sent at twice the effective bottleneck rate to exercise the queue at the dominant bottleneck.

#### 8.3.1. Full Window slowstart test

This is a capacity test to confirm that slowstart is not likely to exit prematurely. Send slowstart bursts that are target_window_size total packets.

Accumulate packet transfer statistics as described in Section 7.2 to score the outcome. Pass if it is statistically significant that the observed number of good packets delivered between losses or ECN CE marks is larger than the target_run_length. Fail if it is statistically significant that the observed interval between losses or ECN CE marks is smaller than the target_run_length.

It is deemed inconclusive if the elapsed time to send the data burst is not less than half of the time to receive the ACKs. (i.e. It is acceptable to send data too fast, but sending it slower than twice the actual bottleneck rate as indicated by the ACKs is deemed inconclusive). The headway for the slowstart bursts should be the target_RTT.

Note that these are the same parameters as the Sender Full Window burst test, except the burst rate is at slowstart rate, rather than sender interface rate.

#### 8.3.2. Slowstart AQM test

Do a continuous slowstart (send data continuously at twice the implied IP bottleneck capacity), until the first loss, stop, allow the network to drain and repeat, gathering statistics on how many packets were delivered before the loss, the pattern of losses, maximum observed RTT and window size. Justify the results. There is not currently sufficient theory justifying requiring any particular result, however design decisions that affect the outcome of this tests also affect how the network balances between long and short flows (the "mice vs elephants" problem). The queue sojourn time for the first packet delivered after the first loss should be at least one half of the target_RTT.

This is an engineering test: It should be performed on a quiescent network or testbed, since cross traffic has the potential to change the results in ill defined ways.

### 8.4. Sender Rate Burst tests

These tests determine how well the network can deliver bursts sent at sender's interface rate. Note that this test most heavily exercises the front path, and is likely to include infrastructure may be out of scope for an access ISP, even though the bursts might be caused by ACK compression, thinning or channel arbitration in the access ISP. See Appendix B.

Also, there are a several details about sender interface rate bursts that are not fully defined here. These details, such as the assumed sender interface rate, should be explicitly stated is a FS-TIDS.

Current standards permit TCP to send full window bursts following an application pause. (Congestion Window Validation [RFC2861] and updates to support Rate-Limited Traffic [RFC7661], are not required). Since full window bursts are consistent with standard behavior, it is desirable that the network be able to deliver such bursts, otherwise application pauses will cause unwarranted losses. Note that the AIMD sawtooth requires a peak window that is twice target_window_size, so the worst case burst may be 2*target_window_size.

It is also understood in the application and serving community that interface rate bursts have a cost to the network that has to be balanced against other costs in the servers themselves. For example TCP Segmentation Offload (TSO) reduces server CPU in exchange for larger network bursts, which increase the stress on network buffer memory. Some newer TCP implementations can pace traffic at scale [TSO_pacing][TSO_fq_pacing]. It remains to be determined if and how quickly these changes will be deployed.

There is not yet theory to unify these costs or to provide a framework for trying to optimize global efficiency. We do not yet have a model for how much server rate bursts should be tolerated by the network. Some bursts must be tolerated by the network, but it is probably unreasonable to expect the network to be able to efficiently deliver all data as a series of bursts.

For this reason, this is the only test for which we encourage derating. A TIDS could include a table of pairs of derating parameters: burst sizes and how much each burst size is permitted to reduce the run length, relative to to the target_run_length.

### 8.5. Combined and Implicit Tests

Combined tests efficiently confirm multiple network properties in a single test, possibly as a side effect of normal content delivery. They require less measurement traffic than other testing strategies at the cost of conflating diagnostic signatures when they fail. These are by far the most efficient for monitoring networks that are nominally expected to pass all tests.

#### 8.5.1. Sustained Bursts Test

The sustained burst test implements a combined worst case version of all of the capacity tests above. It is simply:

Send target_window_size bursts of packets at server interface rate with target_RTT burst headway (burst start to next burst start). Verify that the observed packet transfer statistics meets the target_run_length.

Key observations:

- The subpath under test is expected to go idle for some fraction of the time, determined by the difference between the time to drain the queue at the subpath_IP_capacity, and the target_RTT. If the queue does not drain completely it may be an indication that the the subpath has insufficient IP capacity or that there is some other problem with the test (e.g. inconclusive).
- The burst sensitivity can be derated by sending smaller bursts more frequently. E.g. send target_window_size*derate packet bursts every target_RTT*derate, where "derate" is less than one.
- When not derated, this test is the most strenuous capacity test.
- A subpath that passes this test is likely to be able to sustain higher rates (close to subpath_IP_capacity) for paths with RTTs significantly smaller than the target_RTT.
- This test can be implemented with instrumented TCP [RFC4898], using a specialized measurement application at one end [MBMSource] and a minimal service at the other end [RFC0863] [RFC0864].
- This test is efficient to implement, since it does not require per-packet timers, and can make use of TSO in modern NIC hardware.
- If a subpath is known to pass the Standing Queue engineering tests (particularly that it has a progressive onset of loss at an appropriate queue depth), then the Sustained Burst Test is sufficient to assure that the subpath under test will not impair Bulk Transport Capacity at the target performance under all conditions. See Section 8.2 for a discussion of the standing queue tests.

Note that this test is clearly independent of the subpath RTT, or other details of the measurement infrastructure, as long as the measurement infrastructure can accurately and reliably deliver the required bursts to the subpath under test.

#### 8.5.2. Passive Measurements

Any non-throughput maximizing application, such as fixed rate streaming media, can be used to implement passive or hybrid (defined in [RFC7799]) versions of Model Based Metrics with some additional instrumentation and possibly a traffic shaper or other controls in the servers. The essential requirement is that the data transmission be constrained such that even with arbitrary application pauses and bursts, the data rate and burst sizes stay within the envelope defined by the individual tests described above.

If the application's serving data rate can be constrained to be less than or equal to the target_data_rate and the serving_RTT (the RTT between the sender and client) is less than the target_RTT, this constraint is most easily implemented by clamping the transport window size to serving_window_clamp, set to the test_window, computed for the actual serving path.

Under the above constraints the serving_window_clamp will limit the both the serving data rate and burst sizes to be no larger than the procedures in Section 8.1.2 and Section 8.4 or Section 8.5.1. Since the serving RTT is smaller than the target_RTT, the worst case bursts that might be generated under these conditions will be smaller than called for by Section 8.4 and the sender rate burst sizes are implicitly derated by the serving_window_clamp divided by the target_window_size at the very least. (Depending on the application behavior, the data might be significantly smoother than specified by any of the burst tests.)

In an alternative implementation the data rate and bursts might be explicitly controlled by a programmable traffic shaper or pacing at the sender. This would provide better control over transmissions but is more complicated to implement, although the required technology is available [TSO_pacing][TSO_fq_pacing].

Note that these techniques can be applied to any content delivery that can operated at a constrained data rate to inhibit TCP equilibrium behavior.

Furthermore note that Dynamic Adaptive Streaming over HTTP (DASH) is generally in conflict with passive Model Based Metrics measurement, because it is a rate maximizing protocol. It can still meet the requirement here if the rate can be capped, for example by knowing a priori the maximum rate needed to deliver a particular piece of content.

## 9. An Example

In this section we illustrate a TIDS designed to confirm that an access ISP can reliably deliver HD video from multiple content providers to all of their customers. With modern codecs, minimal HD video (720p) generally fits in 2.5 Mb/s. Due to their geographical size, network topology and modem characteristics the ISP determines that most content is within a 50 mS RTT of their users (This example RTT is a sufficient to cover the propagation delay to continental Europe or either US coast with low delay modems or somewhat smaller geographical regions if the modems require additional delay to implement advanced compression and error recovery).

2.5 Mb/s over a 50 ms path

| End-to-End Parameter | value | units |
|---|---|---|
| target_rate | 2.5 | Mb/s |
| target_RTT | 50 | ms |
| target_MTU | 1500 | bytes |
| header_overhead | 64 | bytes |
| | | |
| target_window_size | 11 | packets |
| target_run_length | 363 | packets |

Table 1

Table 1 shows the default TCP model with no derating, and as such is quite conservative. The simplest TIDS would be to use the sustained burst test, described in Section 8.5.1. Such a test would send 11 packet bursts every 50mS, and confirming that there was no more than 1 packet loss per 33 bursts (363 total packets in 1.650 seconds).

Since this number represents is the entire end-to-end loss budget, independent subpath tests could be implemented by apportioning the packet loss ratio across subpaths. For example 50% of the losses might be allocated to the access or last mile link to the user, 40% to the network interconnections with other ISPs and 1% to each internal hop (assuming no more than 10 internal hops). Then all of the subpaths can be tested independently, and the spatial composition of passing subpaths would be expected to be within the end-to-end loss budget.

### 9.1. Observations about applicability

Guidance on deploying and using MBM belong in a future document. However this example illustrates some the issues that may need to be considered.

Note that another ISP, with different geographical coverage, topology or modem technology may need to assume a different target_RTT, and as a consequence different target_window_size and target_run_length, even for the same target_data rate. One of the implications of this is that infrastructure shared by multiple ISPs, such as inter-exchange points (IXPs) and other interconnects may need to be evaluated on the basis of the most stringent target_window_size and target_run_length of any participating ISP. One way to do this might be to choose target parameters for evaluating such shared infrastructure on the basis of a hypothetical reference path that does not necessarily match any actual paths.

Testing interconnects has generally been problematic: conventional performance tests run between measurement points adjacent to either side of the interconnect are not generally useful. Unconstrained TCP tests, such as iPerf [iPerf] are usually overly aggressive due to the small RTT (often less than 1 mS). With a short RTT these tools are likely to report inflated data rates because on a short RTT these tools can tolerate very high packet loss ratios and can push other cross traffic off of the network. As a consequence these measurements are useless for predicting actual user performance over longer paths, and may themselves be quite disruptive. Model Based Metrics solves this problem. The interconnect can be evaluated with the same TIDS as other subpaths. Continuing our example, if the interconnect is apportioned 40% of the losses, 11 packet bursts sent every 50mS should have fewer than one loss per 82 bursts (902 packets).

## 10. Validation

Since some aspects of the models are likely to be too conservative, Section 5.2 permits alternate protocol models and Section 5.3 permits test parameter derating. If either of these techniques are used, we require demonstrations that such a TIDS can robustly detect subpaths that will prevent authentic applications using state-of-the-art protocol implementations from meeting the specified Target Transport Performance. This correctness criteria is potentially difficult to prove, because it implicitly requires validating a TIDS against all possible paths and subpaths. The procedures described here are still experimental.

We suggest two approaches, both of which should be applied: first, publish a fully open description of the TIDS, including what assumptions were used and and how it was derived, such that the research community can evaluate the design decisions, test them and comment on their applicability; and second, demonstrate that applications do meet the Target Transport Performance when running over a network testbed which has the tightest possible constraints that still allow the tests in the TIDS to pass.

This procedure resembles an epsilon-delta proof in calculus. Construct a test network such that all of the individual tests of the TIDS pass by only small (infinitesimal) margins, and demonstrate that a variety of authentic applications running over real TCP implementations (or other protocols as appropriate) meets the Target Transport Performance over such a network. The workloads should include multiple types of streaming media and transaction oriented short flows (e.g. synthetic web traffic).

For example, for the HD streaming video TIDS described in Section 9, the IP capacity should be exactly the header_overhead above 2.5 Mb/s, the per packet random background loss ratio should be 1/363, for a run length of 363 packets, the bottleneck queue should be 11 packets and the front path should have just enough buffering to withstand 11 packet interface rate bursts. We want every one of the TIDS tests to fail if we slightly increase the relevant test parameter, so for example sending a 12 packet burst should cause excess (possibly deterministic) packet drops at the dominant queue at the bottleneck. This network has the tightest possible constraints that can be expected to pass the TIDS, yet it should be possible for a real application using a stock TCP implementation in the vendor's default configuration to attain 2.5 Mb/s over an 50 mS path.

The most difficult part of setting up such a testbed is arranging for it to have the tightest possible constraints that still allow it to pass the individual tests. Two approaches are suggested: constraining (configuring) the network devices not to use all available resources (e.g. by limiting available buffer space or data rate); and pre-loading subpaths with cross traffic. Note that is it important that a single tightly constrained environment just barely passes all tests, otherwise there is a chance that TCP can exploit extra latitude in some parameters (such as data rate) to partially compensate for constraints in other parameters (queue space, or vice-versa).

To the extent that a TIDS is used to inform public dialog it should be fully publicly documented, including the details of the tests, what assumptions were used and how it was derived. All of the details of the validation experiment should also be published with sufficient detail for the experiments to be replicated by other researchers. All components should either be open source of fully described proprietary implementations that are available to the research community.

## 11. Security Considerations

Measurement is often used to inform business and policy decisions, and as a consequence is potentially subject to manipulation. Model Based Metrics are expected to be a huge step forward because equivalent measurements can be performed from multiple vantage points, such that performance claims can be independently validated by multiple parties.

Much of the acrimony in the Net Neutrality debate is due to the historical lack of any effective vantage independent tools to characterize network performance. Traditional methods for measuring Bulk Transport Capacity are sensitive to RTT and as a consequence often yield very different results when run local to an ISP or interconnect and when run over a customer's complete path. Neither the ISP nor customer can repeat the others measurements, leading to high levels of distrust and acrimony. Model Based Metrics are expected to greatly improve this situation.

Note that in situ measurements sometimes requires sending synthetic measurement traffic between arbitrary locations in the network, and as such are potentially attractive platforms for launching DDOS attacks. All active measurement tools and protocols must be designed to minimize the opportunities for these misuses. See the discussion in section 7 of [RFC7594].

Some of the tests described in the note are not intended for frequent network monitoring since they have the potential to cause high network loads and might adversely affect other traffic.

This document only describes a framework for designing Fully Specified Targeted IP Diagnostic Suite. Each FS-TIDS must include its own security section.

## 12. Acknowledgments

Ganga Maguluri suggested the statistical test for measuring loss probability in the target run length. Alex Gilgur and Merry Mou for helping with the statistics.

Meredith Whittaker for improving the clarity of the communications.

Ruediger Geib provided feedback which greatly improved the document.

This work was inspired by Measurement Lab: open tools running on an open platform, using open tools to collect open data. See http://www.measurementlab.net/

## 13. IANA Considerations

This document has no actions for IANA.

## 14. Informative References


[RFC0863] Postel, J., "Discard Protocol", STD 21, RFC 863, May 1983.

[RFC0864] Postel, J., "Character Generator Protocol", STD 22, RFC 864, May 1983.

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, May 1998.

[RFC2861] Handley, M., Padhye, J., and S. Floyd, "TCP Congestion Window Validation", RFC 2861, June 2000.

[RFC3148] Mathis, M. and M. Allman, "A Framework for Defining Empirical Bulk Transfer Capacity Metrics", RFC 3148, July 2001.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <http://www.rfc-editor.org/info/rfc3168>.

[RFC3465] Allman, M., "TCP Congestion Control with Appropriate Byte Counting (ABC)", RFC 3465, February 2003.

[RFC4015] Ludwig, R. and A. Gurtov, "The Eifel Response Algorithm for TCP", RFC 4015, February 2005.

[RFC4737] Morton, A., Ciavattone, L., Ramachandran, G., Shalunov, S., and J. Perser, "Packet Reordering Metrics", RFC 4737, November 2006.

[RFC4898] Mathis, M., Heffner, J., and R. Raghunarayan, "TCP Extended Statistics MIB", RFC 4898, May 2007.

[RFC5136] Chimento, P. and J. Ishac, "Defining Network Capacity", RFC 5136, February 2008.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

[RFC5827] Allman, M., Avrachenkov, K., Ayesta, U., Blanton, J., and P. Hurtig, "Early Retransmit for TCP and Stream Control Transmission Protocol (SCTP)", RFC 5827, DOI 10.17487/RFC5827, May 2010, <http://www.rfc-editor.org/info/rfc5827>.

[RFC5835] Morton, A. and S. Van den Berghe, "Framework for Metric Composition", RFC 5835, April 2010.

[RFC6049] Morton, A. and E. Stephan, "Spatial Composition of Metrics", RFC 6049, January 2011.

[RFC6576] Geib, R., Ed., Morton, A., Fardid, R., and A. Steinmitz, "IP Performance Metrics (IPPM) Standard Advancement Testing", BCP 176, RFC 6576, DOI 10.17487/RFC6576, March 2012, <http://www.rfc-editor.org/info/rfc6576>.

[RFC6673] Morton, A., "Round-Trip Packet Loss Metrics", RFC 6673, August 2012.

[RFC6928] Chu, J., Dukkipati, N., Cheng, Y., and M. Mathis, "Increasing TCP's Initial Window", RFC 6928, DOI 10.17487/RFC6928, April 2013, <http://www.rfc-editor.org/info/rfc6928>.

[RFC7312] Fabini, J. and A. Morton, "Advanced Stream and Sampling Framework for IP Performance Metrics (IPPM)", RFC 7312, August 2014.

[RFC7398] Bagnulo, M., Burbridge, T., Crawford, S., Eardley, P., and A. Morton, "A Reference Path and Measurement Points for Large-Scale Measurement of Broadband Performance", RFC 7398, February 2015.

[RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <http://www.rfc-editor.org/info/rfc7567>.

[RFC7594] Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A Framework for Large-Scale Measurement of Broadband Performance (LMAP)", RFC 7594, DOI 10.17487/RFC7594, September 2015, <http://www.rfc-editor.org/info/rfc7594>.

[RFC7661] Fairhurst, G., Sathiaseelan, A., and R. Secchi, "Updating TCP to Support Rate-Limited Traffic", RFC 7661, DOI 10.17487/RFC7661, October 2015, <http://www.rfc-editor.org/info/rfc7661>.

[RFC7680] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Loss Metric for IP Performance Metrics (IPPM)", STD 82, RFC 7680, DOI 10.17487/RFC7680, January 2016, <http://www.rfc-editor.org/info/rfc7680>.

[RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <http://www.rfc-editor.org/info/rfc7799>.

[I-D.ietf-tcpm-rack]
Cheng, Y., Cardwell, N., and N. Dukkipati, "RACK: a time-based fast loss detection algorithm for TCP", draft-ietf-tcpm-rack-02 (work in progress), March 2017.

[MSMO97] Mathis, M., Semke, J., Mahdavi, J., and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm", Computer Communications Review volume 27, number3, July 1997.

[WPING] Mathis, M., "Windowed Ping: An IP Level Performance Diagnostic", INET 94, June 1994.

[mpingSource]
Fan, X., Mathis, M., and D. Hamon, "Git Repository for mping: An IP Level Performance Diagnostic", Sept 2013, <https://github.com/m-lab/mping>.

[MBMSource]
Hamon, D., Stuart, S., and H. Chen, "Git Repository for Model Based Metrics", Sept 2013, <https://github.com/m-lab/MBM>.

[Pathdiag]
Mathis, M., Heffner, J., O'Neil, P., and P. Siemsen, "Pathdiag: Automated TCP Diagnosis", Passive and Active Measurement , June 2008.

[iPerf] Wikipedia Contributors, , "iPerf", Wikipedia, The Free Encyclopedia , cited March 2015, <http://en.wikipedia.org/w/index.php?title=Iperf&oldid=649720021>.

[Wald45] Wald, A., "Sequential Tests of Statistical Hypotheses", The Annals of Mathematical Statistics, Vol. 16, No. 2, pp. 117-186, Published by: Institute of Mathematical Statistics, Stable URL: http://www.jstor.org/stable/2235829, June 1945.

[Montgomery90] Montgomery, D., "Introduction to Statistical Quality Control - 2nd ed.", ISBN 0-471-51988-X, 1990.

[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.

[CVST] Krueger, T. and M. Braun, "R package: Fast Cross-Validation via Sequential Testing", version 0.1, 11 2012.

[AFD] Pan, R., Breslau, L., Prabhakar, B., and S. Shenker, "Approximate fairness through differential dropping", SIGCOMM Comput. Commun. Rev. 33, 2, April 2003.

[wikiBloat] Wikipedia, , "Bufferbloat", http://en.wikipedia.org/w/ index.php?title=Bufferbloat&oldid=608805474, March 2015.

[CCscaling] Fernando, F., Doyle, J., and S. Steven, "Scalable laws for stable network congestion control", Proceedings of Conference on Decision and Control, http://www.ee.ucla.edu/˜paganini, December 2001.

[TSO_pacing] Corbet, J., "TSO sizing and the FQ scheduler", LWN.net https://lwn.net/Articles/564978/, Aug 2013.

[TSO_fq_pacing] Dumazet, E. and Y. Chen, "TSO, fair queuing, pacing: three's a charm", Proceedings of IETF 88, TCPM WG https://www.ietf.org/proceedings/88/slides/slides-88-tcpm-9.pdf, Nov 2013.

[Policing] Flach, T., Papageorge, P., Terzis, A., Pedrosa, L., Cheng, Y., Karim, T., Katz-Bassett, E., and R. Govindan, "An Internet-Wide Analysis of Traffic Policing", ACM SIGCOMM , August 2016.

# Appendix A. Model Derivations

The reference target_run_length described in Section 5.2 is based on very conservative assumptions: that all excess data in flight (window) above the target_window_size contributes to a standing queue that raises the RTT, and that classic Reno congestion control with delayed ACKs are in effect. In this section we provide two alternative calculations using different assumptions.

It may seem out of place to allow such latitude in a measurement method, but this section provides offsetting requirements.

The estimates provided by these models make the most sense if network performance is viewed logarithmically. In the operational Internet, data rates span more than 8 orders of magnitude, RTT spans more than 3 orders of magnitude, and packet loss ratio spans at least 8 orders of magnitude if not more. When viewed logarithmically (as in decibels), these correspond to 80 dB of dynamic range. On an 80 dB scale, a 3 dB error is less than 4% of the scale, even though it represents a factor of 2 in untransformed parameter.

This document gives a lot of latitude for calculating target_run_length, however people designing a TIDS should consider the effect of their choices on the ongoing tussle about the relevance of "TCP friendliness" as an appropriate model for Internet capacity allocation. Choosing a target_run_length that is substantially smaller than the reference target_run_length specified in Section 5.2 strengthens the argument that it may be appropriate to abandon "TCP friendliness" as the Internet fairness model. This gives developers incentive and permission to develop even more aggressive applications and protocols, for example by increasing the number of connections that they open concurrently.

## A.1. Queueless Reno

In Section 5.2 models were derived based on the assumption that the subpath IP rate matches the target rate plus overhead, such that the excess window needed for the AIMD sawtooth causes a fluctuating queue at the bottleneck.

An alternate situation would be a bottleneck where there is no significant queue and losses are caused by some mechanism that does not involve extra delay, for example by the use of a virtual queue as done in Approximate Fair Dropping [AFD]. A flow controlled by such a bottleneck would have a constant RTT and a data rate that fluctuates in a sawtooth due to AIMD congestion control. Assume the losses are being controlled to make the average data rate meet some goal which is equal or greater than the target_rate. The necessary run length to meet the target_rate can be computed as follows:

For some value of Wmin, the window will sweep from Wmin packets to 2*Wmin packets in 2*Wmin RTT (due to delayed ACK). Unlike the queuing case where Wmin = target_window_size, we want the average of Wmin and 2*Wmin to be the target_window_size, so the average data rate is the target rate. Thus we want Wmin = (2/3)*target_window_size.

Between losses each sawtooth delivers (1/2)(Wmin+2*Wmin)(2Wmin) packets in 2*Wmin round trip times.

Substituting these together we get:

target_run_length = (4/3)(target_window_size^2)

Note that this is 44% of the reference_run_length computed earlier. This makes sense because under the assumptions in Section 5.2 the AMID sawtooth caused a queue at the bottleneck, which raised the effective RTT by 50%.

## Appendix B. The effects of ACK scheduling

For many network technologies simple queuing models don't apply: the network schedules, thins or otherwise alters the timing of ACKs and data, generally to raise the efficiency of the channel allocation algorithms when confronted with relatively widely spaced small ACKs. These efficiency strategies are ubiquitous for half duplex, wireless and broadcast media.

Altering the ACK stream by holding or thinning ACKs typically has two consequences: it raises the implied bottleneck IP capacity, making the fine grained slowstart bursts either faster or larger and it raises the effective RTT by the average time that the ACKs and data are delayed. The first effect can be partially mitigated by re-clocking ACKs once they are beyond the bottleneck on the return path to the sender, however this further raises the effective RTT.

The most extreme example of this sort of behavior would be a half duplex channel that is not released as long as the endpoint currently holding the channel has more traffic (data or ACKs) to send. Such environments cause self clocked protocols under full load to revert to extremely inefficient stop and wait behavior. The channel constrains the protocol to send an entire window of data as a single contiguous burst on the forward path, followed by the entire window of ACKs on the return path.

If a particular return path contains a subpath or device that alters the timing of the ACK stream, then the entire front path from the sender up to the bottleneck must be tested at the burst parameters implied by the ACK scheduling algorithm. The most important parameter is the Implied Bottleneck IP Capacity, which is the average rate at which the ACKs advance snd.una. Note that thinning the ACK stream (relying on the cumulative nature of seg.ack to permit discarding some ACKs) causes most TCP implementations to send interface rate bursts to offset the longer times between ACKs in order to maintain the average data rate.

Note that due to ubiquitous self clocking in Internet protocols, ill conceived channel allocation mechanisms are likely to increases the queuing stress on the front path because they cause larger full sender rate data bursts.

Holding data or ACKs for channel allocation or other reasons (such as forward error correction) always raises the effective RTT relative to the minimum delay for the path. Therefore it may be necessary to replace target_RTT in the calculation in Section 5.2 by an effective_RTT, which includes the target_RTT plus a term to account for the extra delays introduced by these mechanisms.

## Appendix C. Version Control

This section to be removed prior to publication.

Formatted: Thu Apr 7 18:12:37 PDT 2016

## Authors' Addresses

Matt Mathis
Google, Inc
1600 Amphitheater Parkway
Mountain View, California 94043
USA

Email: mattmathis@google.com

Al Morton
AT&T Labs
200 Laurel Avenue South
Middletown, NJ 07748
USA

Phone: +1 732 420 1571
Email: acmorton@att.com

Network Working Group G. Mirsky
Internet-Draft ZTE Corp.
Intended status: Standards Track I. Meilik
Expires: October 14, 2017 Broadcom
April 12, 2017

# Support of IEEE-1588 time stamp format in Two-Way Active Measurement Protocol (TWAMP)
draft-ietf-ippm-twamp-time-format-06

## Abstract

This document describes an OPTIONAL feature for active performance measurement protocols allowing use of the Precision Time Protocol time stamp format defined in IEEE-1588v2-2008, as an alternative to the Network Time Protocol that is currently used.

## Status of This Memo



## Copyright Notice

Table of Contents



# 1. Introduction

One-Way Active Measurement Protocol (OWAMP) [RFC4656] defines that only the NTP [RFC5905] format of a time stamp can be used in OWAMP-Test protocol. Two-Way Active Measurement Protocol (TWAMP) [RFC5357] adopted the OWAMP-Test packet format and extended it by adding a format for a reflected test packet. Both the sender's and reflector's packets time stamps are expected to follow the 64-bit long NTP format [RFC5905]. NTP, when used over Internet, typically achieves clock accuracy of about 5ms to 100ms. Surveys conducted recently suggest that 90% devices achieve accuracy of better than 100 ms and 99% - better than 1 sec. It should be noted that NTP synchronizes clocks on the control plane, not on data plane. Distribution of clock within a node may be supported by independent NTP domain or via interprocess communication in multiprocessor distributed system. Any of the mentioned solutions will be subject to additional queuing delays that negatively affect data plane clock accuracy.

Precision Time Protocol (PTP) [IEEE.1588.2008] has gained wide support since the development of OWAMP and TWAMP. PTP, using on-path support and other mechanisms, allows sub-microsecond clock accuracy. PTP is now supported in multiple implementations of fast forwarding engines and thus accuracy achieved by PTP is the accuracy of clock in data plane. An option to use a more accurate clock as a source of time stamps for IP performance measurements is one of this specification's advantages. Another advantage is realized by simplification of hardware in data plane. To support OWAMP or TWAMP test protocol time stamps must be converted from PTP to NTP. That requires resources, use of micro-code or additional processing elements, that are always limited. To address this, this document proposes optional extensions to Control and Test protocols to support use of IEEE-1588v2 time stamp format as optional alternative to the NTP time stamp format.

One of the goals of this specification is not only to allow end-points of a test session to use timestamp format other than NTP but to support backwards compatibility with nodes that do not yet support this extension.

### 1.1. Conventions used in this document

#### 1.1.1. Terminology

IPPM: IP Performance Measurement

NTP: Network Time Protocol

PTP: Precision Time Protocol

TWAMP: Two-Way Active Measurement Protocol

OWAMP: One-Way Active Measurement Protocol

#### 1.1.2. 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 [RFC2119].

## 2. OWAMP and TWAMP Extensions

OWAMP connection establishment follows the procedure defined in Section 3.1 of [RFC4656] and additional steps in TWAMP described in Section 3.1 of [RFC5357]. In these procedures, the Modes field has been used to identify and select specific communication capabilities. At the same time the Modes field has been recognized and used as extension mechanism [RFC6038]. The new feature requires one bit position for Server and Control-Client to negotiate which timestamp format can be used in some or all test sessions invoked with this control connection. The end-point of the test session, Session-Sender and Session-Receiver or Session-Reflector, that supports this extension MUST be capable to interpret NTP and PTPv2 timestamp formats. If the end-point does not support this extension, then the value of PTPv2 Timestamp flag MUST be 0 because it is in Must Be Zero field. If the value of PTPv2 Timestamp flags is 0, then the advertising node can use and interpret only NTP timestamp format. Implementations of OWAMP and/or TWAMP MAY provide a configuration knob to bypass the timestamp format negotiation process and to use the locally configured values instead.

Use of PTPv2 Timestamp flags is discussed in the following sub-sections. For details on the assigned values and bit positions see the Section 3.

### 2.1. Timestamp Format Negotiation in Setting Up Connection in OWAMP

In OWAMP-Test [RFC4656] the Session-Receiver and/or Fetch-Client interpret collected timestamps. Thus, the Server uses the Modes field timestamp format to indicate which formats the Session-Receiver is capable to interpret. The Control-Client inspects values set by the Server for timestamp formats and sets values in the Modes field of the Set-Up-Response message according to timestamp formats Session-Sender can use. The rules of setting timestamp flags in Modes field in server greeting and Set-Up-Response messages and interpreting them are as follows:

o If the Session-Receiver supports this extension, then the Server that establishes test sessions on its behalf MUST set PTPv2 Timestamp flag to 1 in the server greeting message per the requirement listed in Section 2. Otherwise, the PTPv2 Timestamp flag will be set to 0 to indicate that the Session-Receiver interprets only NTP format.

o If the Control-Client receives greeting message with the PTPv2 Timestamp flag set to 0, then the Session-Sender MUST use NTP format for timestamp in the test session and Control-Client SHOULD set PTPv2 Timestamp flag to 0 in accordance with [RFC4656]. If the Session-Sender cannot use NTP timestamps, then the Control-Client SHOULD close the TCP connection associated with the OWAMP-Control session.

o If the Control-Client receives greeting message with the PTPv2 Timestamp flag set to 1 and the Session-Sender can set timestamp in PTPv2 format, then the Control-Client MUST set the PTPv2 Timestamp flag to 1 in Modes field in the Set-Up-Response message and the Session-Sender MUST use PTPv2 timestamp format.

o If the Session-Sender doesn’t support this extension and can set timestamp only in NTP format, then the PTPv2 Timestamp flag in

Modes field in the Set-Up-Response message will be set to 0 as part of Must Be Zero and the Session-Sender use NTP format.

If OWAMP-Control uses Fetch-Session commands, then selection and use of one or another timestamp format is local decision for both Session-Sender and Session-Receiver.

## 2.2. Timestamp Format Negotiation in Setting Up Connection in TWAMP

In TWAMP-Test [RFC5357] the Session-Sender interprets collected timestamps. Hence, in the Modes field a Server advertises timestamp formats that the Session-Reflector can use in TWAMP-Test message. The choice of the timestamp format to be used by the Session-Sender is a local decision. The Control-Client inspects the Modes field and sets timestamp flags values to indicate which format will be used by the Session-Reflector. The rules of setting and interpreting flag values are as follows:

- Server MUST set to 1 value of PTPv2 Timestamp flag in its greeting message if Session-Reflector can set timestamp in PTPv2 format. Otherwise the PTPv2 Timestamp flag MUST be set to 0.

- If value of the PTPv2 Timestamp flag in received server greeting message equals 0, then Session-Reflector does not support this extension and will use NTP timestamp format. Control-Client SHOULD set PTPv2 Timestamp flag to 0 in Set-Up-Response message in accordance with [RFC5357].

- Control-Client MUST set PTPv2 Timestamp flag value to 1 in Modes field in the Set-Up-Response message if Server advertised ability of the Session-Reflector to use PTPv2 format for timestamps. Otherwise the flag MUST be set to 0.

- If the values of PTPv2 Timestamp flag in the Set-Up-Response message equals 0, then that means that Session-Sender can only interpret NTP timestamp format. Then the Session-Reflector MUST use NTP timestamp format. If the Session-Reflector does not support NTP format then Server and MUST close the TCP connection associated with the TWAMP-Control session.

## 2.3. OWAMP-Test and TWAMP-Test Update

Participants of a test session need to indicate which timestamp format being used. The specification is to use Z field in Error Estimate defined in Section 4.1.2 of [RFC4656]. The new interpretation of the Error Estimate is in addition to it specifying error estimate and synchronization, Error Estimate indicates format of a collected timestamp. And this specification changes the semantics of the Z bit field, the one between S and Scale fields, to be referred as Timestamp format and value MUST be set per the following:

o 0 - NTP 64 bit format of a timestamp;

o 1 - PTPv2 truncated format of a timestamp.

As result of this value of the Z field from Error Estimate, Sender Error Estimate or Send Error Estimate and Receive Error Estimate SHOULD NOT be ignored and MUST be used when calculating delay and delay variation metrics based on collected timestamps.

### 2.3.1. Consideration for TWAMP Light mode

This document does not specify how Session-Sender and Session-Reflector in TWAMP Light mode are informed of timestamp format to be used. It is assumed that, for example, configuration could be used to direct Session-Sender and Session-Reflector respectively to use timestamp format per their capabilities and rules listed in Section 2.2.

## 3. IANA Considerations

The TWAMP-Modes registry defined in [RFC5618].

IANA is requested to reserve a new PTPv2 Timestamp as follows:

| Value | Description | Semantics | Reference |
|---|---|---|---|
| TBA1 (proposed 256) | PTPv2 Timestamp Capability | bit position TBA2 (proposed 8) | This document |

Table 1: New Timestamp Capability

## 4. Security Considerations

Use of particular format of a timestamp in test session does not appear to introduce any additional security threat to hosts that communicate with OWAMP and/or TWAMP as defined in [RFC4656], [RFC5357] respectively. The security considerations that apply to any active measurement of live networks are relevant here as well. See the Security Considerations sections in [RFC4656] and [RFC5357].

## 5. Acknowledgements

The authors would like to thank Lakshmikanthan and Suchit Bansal for their insightful suggestions. The authors would like to thank David Allan for his thorough review and thoughtful comments.

## 6. Normative References

[IEEE.1588.2008]
"Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Standard 1588, March 2008.

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

[RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, DOI 10.17487/RFC4656, September 2006, <http://www.rfc-editor.org/info/rfc4656>.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <http://www.rfc-editor.org/info/rfc5357>.

[RFC5618] Morton, A. and K. Hedayat, "Mixed Security Mode for the Two-Way Active Measurement Protocol (TWAMP)", RFC 5618, DOI 10.17487/RFC5618, August 2009, <http://www.rfc-editor.org/info/rfc5618>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <http://www.rfc-editor.org/info/rfc5905>.

[RFC6038] Morton, A. and L. Ciavattone, "Two-Way Active Measurement Protocol (TWAMP) Reflect Octets and Symmetrical Size Features", RFC 6038, DOI 10.17487/RFC6038, October 2010, <http://www.rfc-editor.org/info/rfc6038>.

## Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Israel Meilik
Broadcom

Email: israel@broadcom.com

IPPM WG
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

R. Civil
Ciena Corporation
A. Morton
AT&T Labs
R. Rahman
Cisco Systems
M. Jethanandani
Xoriant Corporation
K. Pentikousis, Ed.
Travelping

July 2, 2018

# Two-Way Active Measurement Protocol (TWAMP) Data Model
draft-ietf-ippm-twamp-yang-13

## Abstract

This document specifies a data model for client and server implementations of the Two-Way Active Measurement Protocol (TWAMP). The document defines the TWAMP data model through Unified Modeling Language (UML) class diagrams and formally specifies it using a NDMA-compliant YANG model.

## Status of This Memo



## Copyright Notice

Table of Contents

# 1. Introduction

The Two-Way Active Measurement Protocol (TWAMP) [RFC5357] is used to measure network performance parameters such as latency, bandwidth, and packet loss by sending probe packets and measuring their experience in the network. To date, TWAMP implementations do not come with a standard management framework, and, as such, implementers have no choice except to provide a proprietary mechanism. This document addresses this gap by defining the model using UML [UML] class diagrams, and formally specifying a NMDA-complaint [RFC8342] TWAMP data model using YANG 1.1 [RFC7950].

## 1.1. Motivation

In current TWAMP deployments the lack of a standardized data model limits the flexibility to dynamically instantiate TWAMP-based measurements across equipment from different vendors. In large, virtualized, and dynamically instantiated infrastructures where network functions are placed according to orchestration algorithms, proprietary mechanisms for managing TWAMP measurements pose severe limitations with respect to programmability.

Two major trends call for standardizing TWAMP management aspects. First, it is expected that in the coming years large-scale and multi-vendor TWAMP deployments will become the norm. From an operations perspective, using several vendor-specific TWAMP configuration mechanisms when one standard mechanism could provide an alternative is expensive and inefficient. Second, the increasingly software-defined and virtualized nature of network infrastructures, based on dynamic service chains [NSC] and programmable control and management planes Software-Defined Networking (SDN): Layers and Architecture Terminology [RFC7426] requires a well-defined data model for TWAMP implementations. This document defines such a TWAMP data model and specifies it formally using the YANG 1.1 [RFC7950] data modeling language.

Note to RFC Editor:

Please replace the date 2018-07-02 in Section 5.2 of the draft with the date of publication of this draft as a RFC. Also, replace reference to RFC XXXX, and draft-ietf-ippm-port-twamp-test with the RFC numbers assigned to the drafts.

## 1.2. Terminology

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.

### 1.3. Document Organization

The rest of this document is organized as follows. Section 2 presents the scope and applicability of this document. Section 3 provides a high-level overview of the TWAMP data model. Section 4 details the configuration parameters of the data model and Section 5 specifies in YANG the TWAMP data model. Section 6 lists illustrative examples which conform to the YANG data model specified in this document. Appendix A elaborates these examples further.

## 2. Scope, Model, and Applicability

The purpose of this document is the specification of a vendor-independent data model for TWAMP implementations.

Figure 1 illustrates a redrawn version of the TWAMP logical model found in Section 1.2 of TWAMP [RFC5357]. The figure is annotated with pointers to the UML [UML] diagrams provided in this document and associated with the data model of the four logical entities in a TWAMP deployment, namely the TWAMP Control-Client, Server, Session-Sender and Session-Reflector. A UML [UML] Notation Guide is available in Section 5 of the said document.

As per TWAMP [RFC5357], unlabeled links in Figure 1 are left unspecified and may be proprietary protocols.

```
    [Fig. 3]                                [Fig. 4]
+----------------+                        +--------+
| Control-Client |  <-- TWAMP-Control --> | Server |
+----------------+                        +--------+
        ^                                     ^
        |                                     |
        V                                     V
+----------------+                   +-------------------+
| Session-Sender |  <-- TWAMP-Test --> | Session-Reflector |
+----------------+                   +-------------------+
    [Fig. 5]                                [Fig. 6]
```

Figure 1: Annotated TWAMP logical model

As per TWAMP [RFC5357], a TWAMP implementation may follow a simplified logical model, in which the same node acts both as Control-Client and Session-Sender, while another node acts at the same time as TWAMP Server and Session-Reflector. Figure 2 illustrates this simplified logical model and indicates the interaction between the TWAMP configuration client and server using, for instance, NETCONF [RFC6241] or RESTCONF [RFC8040].

```
o-------------------o                           o-------------------o
|   Config client   |                           |   Config client   |
o-------------------o                           o-------------------o
          ||                                              ||
 NETCONF  || RESTCONF                            NETCONF  || RESTCONF
          ||                                              ||
o-------------------o                           o-------------------o
|   Config server   |                           |   Config server   |
|    [Fig. 3, 5]    |                           |    [Fig. 4, 6]    |
+-------------------+                           +-------------------+
|   Control-Client  | <-- TWAMP-Control -->     |      Server       |
|                   |                           |                   |
|   Session-Sender  |  <-- TWAMP-Test -->       | Session-Reflector |
+-------------------+                           +-------------------+
```

Figure 2: Simplified TWAMP model and protocols

The data model defined in this document is orthogonal to the specific protocol used between the Config client and Config server to communicate the TWAMP configuration parameters.

Operational actions such as how TWAMP-Test sessions are started and stopped, how performance measurement results are retrieved, or how stored results are cleared, and so on, are not addressed by the configuration model defined in this document. As noted above, such operational actions are not part of the TWAMP specification TWAMP [RFC5357] and hence are out of scope of this document. See also Appendix B. In addition, for operational state, current work in Registry for Performance Metrics [I-D.ietf-ippm-metric-registry], can be used to develop an independent model for the performance metrics that need to be captured and retrieved.

## 3. Data Model Overview

The TWAMP data model includes four categories of configuration items.

First, global configuration items relate to parameters that are set on a per device level. For example, the administrative status of the device with respect to whether it allows TWAMP sessions and, if so, in what capacity (e.g. Control-Client, Server or both), is a typical instance of a global configuration item.

A second category includes attributes that can be configured on a per TWAMP-Control connection basis, such as the Server IP address.

A third category includes attributes related to per TWAMP-Test session attributes, for instance setting different values in the Differentiated Services Code Point (DSCP) field.

Finally, the data model includes attributes that relate to the operational state of the TWAMP implementation.

As the TWAMP data model is described in the remaining sections of this document, readers should keep in mind the functional entity grouping illustrated in Figure 1.

## 3.1. Control-Client

A TWAMP Control-Client has an administrative status field set at the device level that indicates whether the node is enabled to function as such.

Each TWAMP Control-Client is associated with zero or more TWAMP-Control connections. The main configuration parameters of each control connection are:

- o A name which can be used to uniquely identify at the Control-Client a particular control connection. This name is necessary for programmability reasons because at the time of creation of a TWAMP-Control connection not all IP and TCP port number information needed to uniquely identify the connection is available.

- o The IP address of the interface the Control-Client will use for connections.

- o The IP address of the remote TWAMP Server.

- o Authentication and encryption attributes such as KeyID, Token and the Client Initialization Vector (Client-IV); see also Section 3.1 in OWAMP [RFC4656] and Randomness Requirements for Security [RFC4086].

Each TWAMP-Control connection, in turn, is associated with zero or more TWAMP-Test sessions. For each test session, the following configuration items should be noted:

- o The test session name uniquely identifies a particular test session at the Control-Client and Session-Sender. Similar to the control connections above, this unique test session name is needed because at the time of creation of a TWAMP-Test session, for example, the source UDP port number is not known to uniquely identify the test session.

o The IP address and UDP port number of the Session-Sender on the path under test by TWAMP.

o The IP address and UDP port number of the Session-Reflector on said path.

o Information pertaining to the test packet stream, such as the test starting time, which performance metric is to be used, as defined in Registry for Performance Metrics [I-D.ietf-ippm-metric-registry], or whether the test should be repeated.

## 3.2. Server

Each TWAMP Server has an administrative status field set at the device level to indicate whether the node is enabled to function as a TWAMP Server.

Each Server is associated with zero or more TWAMP-Control connections. Each control connection is uniquely identified by the 4-tuple {Control-Client IP address, Control-Client TCP port number, Server IP address, Server TCP port}. Control connection configuration items on a TWAMP Server are read-only.

## 3.3. Session-Sender

A TWAMP Session-Sender has an administrative status field set at the device level that indicates whether the node is enabled to function as such.

There is one Session-Sender instance for each TWAMP-Test session that is initiated from the sending device. Primary configuration fields include:

o The test session name MUST be identical to the corresponding test session name on the TWAMP Control-Client (Section 3.1).

o The control connection name, which along with the test session name uniquely identify the TWAMP Session-Sender instance.

o Information pertaining to the test packet stream, such as, the number of test packets and the packet distribution to be employed; see also Network performance measurement with periodic streams [RFC3432].

### 3.4. Session-Reflector

Each TWAMP Session-Reflector has an administrative status field set at the device level to indicate whether the node is enabled to function as such.

Each Session-Reflector is associated with zero or more TWAMP-Test sessions. For each test session, the REFWAIT timeout parameter, which determines whether to discontinue the session if no packets have been received (TWAMP [RFC5357], Section 4.2), can be configured.

Read-only access to other data model parameters, such as the Sender IP address, is foreseen. Each test session can be uniquely identified by the 4-tuple mentioned in Section 3.2.

## 4. Data Model Parameters

This section defines the TWAMP data model using UML [UML] and introduces selected parameters associated with the four TWAMP logical entities. The complete TWAMP data model specification is provided in the YANG module presented in Section 5.2.

### 4.1. Control-Client

The client container (see Figure 3) holds items that are related to the configuration of the TWAMP Control-Client logical entity (recall Figure 1).

The client container includes an administrative configuration parameter (client/admin-state) that indicates whether the device is allowed to initiate TWAMP-Control connections.

```
   +-------------+
   | client      |
   +-------------+                   1..* +-----------------------+
   | admin-state |<>----------------------| mode-preference-chain |
   |             |                        +-----------------------+
   |             |  1..* +------------+   | priority              |
   |             |<>-----| key-chain  |   | mode                  |
   +-------------+       +------------+   +-----------------------+
          ^              | key-id     |
          V              | secret-key |
          |              +------------+
          | 0..*
   +------------------------+
   | ctrl-connection        |
   +------------------------+
   | name                   |
   | client-ip              |
   | server-ip              |
   | server-tcp-port        |     0..* +----------------------+
   | control-packet-dscp    |<>-------| test-session-request |
   | key-id                 |         +----------------------+
   | max-count              |         | name                 |
   | client-tcp-port   {ro} |         | sender-ip            |
   | server-start-time {ro} |         | sender-udp-port      |
   | state             {ro} |         | reflector-ip         |
   | selected-mode     {ro} |         | reflector-udp-port   |
   | token             {ro} |         | timeout              |
   | client-iv         {ro} |         | padding-length       |
   +------------------------+         | test-packet-dscp     |
                                      | start-time           |
               +-------------+ 1      | repeat               |
               | pm-reg-list |------<>| repeat-interval      |
               +-------------+        | state           {ro} |
               | pm-index    |        | sid             {ro} |
               +-------------+        +----------------------+
```

Figure 3: TWAMP Control-Client UML class diagram

The client container holds a list (mode-preference-chain) which specifies the Mode values according to their preferred order of use by the operator of this Control-Client, including the authentication and encryption Modes. Specifically, mode-preference-chain lists the mode and its corresponding priority, as a 16-bit unsigned integer. Values for the priority start with zero, the highest priority, and decreasing priority value is indicated by every increase in value by one.

Depending on the Modes available in the Server Greeting, the Control-Client MUST choose the highest priority Mode from the configured mode-preference-chain list.

Note that the list of preferred Modes may set multiple bit positions independently, such as when referring to the extended TWAMP features in Mixed Security Mode for TWAMP [RFC5618], Individual Session Control Feature for TWAMP [RFC5938], TWAMP Reflect Octets and Symmetrical Size Features [RFC6038], and IKEv2-Derived Shared Secret Key for OWAMP and TWAMP [RFC7717]. If the Control-Client cannot determine an acceptable Mode, or when the bit combinations do not make sense, e.g., both authenticated and unauthenticated bit are set, it MUST respond with zero Mode bits set in the Set-up Response message, indicating it will not continue with the control connection.

In addition, the client container holds a list named key-chain which relates key-id with the respective secret-key. Both the Server and the Control-Client use the same mappings from key-id to secret-key (in Figure 3); in order for this to work properly, key-id must be unique across all systems in the administrative domain. The Server, being prepared to conduct sessions with more than one Control-Client, uses key-id to choose the appropriate secret-key; a Control-Client would typically have different secret keys for different Servers. The secret-key is the shared secret, of type binary and the length SHOULD contain at least 128 bits of entropy. The key-id and secret-key encoding SHOULD follow Section 9.8 of YANG [RFC7950]. The derived key length (dkLen in PKCS #5: Password-Based Cryptography Specification Version 2.1 [RFC8018]) MUST be 16 octets for the AES Session-key used for encryption and 32 octets for the HMAC-SHA1 Session-key used for authentication; see also Section 6.10 of OWAMP [RFC4656].

Each client container also holds a list of control connections, where each item in the list describes a TWAMP control connection initiated by this Control-Client. There SHALL be one ctrl-connection per TWAMP-Control (TCP) connection that is to be initiated from this device.

In turn, each ctrl-connection holds a test-session-request list. Each test-session-request holds information associated with the Control-Client for this test session. This includes information associated with the Request-TW-Session/Accept-Session message exchange (see Section 3.5 of TWAMP [RFC5357]).

There SHALL be one instance of test-session-request for each TWAMP-Test session that is to be negotiated by this TWAMP-Control connection via a Request-TW-Session/Accept-Session exchange.

The Control-Client is also responsible for scheduling TWAMP-Test sessions, therefore test-session-request holds information related to these actions (e.g. pm-index, repeat-interval).

### 4.2. Server

The server container (see Figure 4) holds items that are related to the configuration of the TWAMP Server logical entity (recall Figure 1).

The server container includes an administrative configuration parameter (server/admin-state) that indicates whether the device is allowed to receive TWAMP-Control connections.

A device operating in the Server role cannot configure attributes on a per TWAMP-Control connection basis, as it has no foreknowledge of the incoming TWAMP-Control connections to be received. Consequently, any parameter that the Server might want to apply to an incoming control connection must be configured at the overall Server level and applied to all incoming TWAMP-Control connections.

```
+---------------------+
| server              |
+---------------------+
| admin-state         |    1..* +------------+
| server-tcp-port     |<>------| key-chain  |
| servwait            |        +------------+
| control-packet-dscp |        | key-id     |
| count               |        | secret-key |
| max-count           |        +------------+
| modes               |
|                     |    0..* +--------------------------+
|                     |<>------| ctrl-connection          |
+---------------------+        +--------------------------+
                               | client-ip           {ro} |
                               | client-tcp-port     {ro} |
                               | server-ip           {ro} |
                               | server-tcp-port     {ro} |
                               | state               {ro} |
                               | control-packet-dscp {ro} |
                               | selected-mode       {ro} |
                               | key-id              {ro} |
                               | count               {ro} |
                               | max-count           {ro} |
                               | salt                {ro} |
                               | server-iv           {ro} |
                               | challenge           {ro} |
                               +--------------------------+
```

Figure 4: TWAMP Server UML class diagram

Each server container holds a list named key-chain which relates key-id with the respective secret-key. As mentioned in Section 4.1, both the Server and the Control-Client use the same mapping from key-id to shared secret-key; in order for this to work properly, key-id must be unique across all the systems in the administrative domain. The Server, being prepared to conduct sessions with more than one Control-Client, uses key-id to choose the appropriate secret-key; a Control-Client would typically have different secret keys for different Servers. The key-id tells the Server which shared secret-key the Control-Client wishes to use for authentication or encryption.

Each incoming control connection active on the Server is represented by a ctrl-connection. There SHALL be one ctrl-connection per incoming TWAMP-Control (TCP) connection that is received and active on the Server. Each ctrl-connection can be uniquely identified by the 4-tuple {client-ip, client-tcp-port, server-ip, server-tcp-port}. All items in the ctrl-connection list are read-only.

### 4.3. Session-Sender

The session-sender container, illustrated in Figure 5, holds items that are related to the configuration of the TWAMP Session-Sender logical entity.

The session-sender container includes an administrative parameter (session-sender/admin-state) that controls whether the device is allowed to initiate TWAMP-Test sessions.

```
+-----------------+
| session-sender  |
+-----------------+  0..* +----------------------------+
| admin-state     |<>-----| test-session               |
+-----------------+       +----------------------------+
                          | name                       |
                          | ctrl-connection-name {ro}  |
                          | fill-mode                  |
                          | number-of-packets          |
                          | state                {ro}  |
                          | sent-packets         {ro}  |
                          | rcv-packets          {ro}  |
                          | last-sent-seq        {ro}  |
                          | last-rcv-seq         {ro}  |
                          +----------------------------+
                                       ^
                                       V
                                       | 1
                           +----------------------+
                           | packet-distribution  |
                           +----------------------+
                           | periodic /  poisson  |
                           +----------------------+
                               |            |
                    +--------------------+  |
                    | periodic-interval  |  |
                    +--------------------+  |
                                            |
                                   +---------------+
                                   | lambda        |
                                   | max-interval  |
                                   +---------------+
```

Figure 5: TWAMP Session-Sender UML class diagram

Each TWAMP-Test session initiated by the Session-Sender will be represented by an instance of a test-session object. There SHALL be one instance of test-session for each TWAMP-Test session for which packets are being sent.

### 4.4. Session-Reflector

The session-reflector container, illustrated in Figure 6, holds items that are related to the configuration of the TWAMP Session-Reflector logical entity.

The session-reflector container includes an administrative parameter (session-reflector/admin-state) that controls whether the device is allowed to respond to incoming TWAMP-Test sessions.

A device operating in the Session-Reflector role cannot configure attributes on a per-session basis, as it has no foreknowledge of what incoming sessions it will receive. As such, any parameter that the Session-Reflector might want to apply to an incoming TWAMP-Test session must be configured at the overall Session-Reflector level and are applied to all incoming sessions.

```
+-------------------+
| session-reflector |
+-------------------+
| admin-state       |
| refwait           |
+-------------------+
         ^
         V
         |
         | 0..*
+----------------------------------------+
| test-session                           |
+----------------------------------------+
| sid                               {ro} |
| sender-ip                         {ro} |
| sender-udp-port                   {ro} |
| reflector-ip                      {ro} |
| reflector-udp-port                {ro} |
| parent-connection-client-ip       {ro} |
| parent-connection-client-tcp-port {ro} |
| parent-connection-server-ip       {ro} |
| parent-connection-server-tcp-port {ro} |
| test-packet-dscp                  {ro} |
| sent-packets                      {ro} |
| rcv-packets                       {ro} |
| last-sent-seq                     {ro} |
| last-rcv-seq                      {ro} |
+----------------------------------------+
```

Figure 6: TWAMP Session-Reflector UML class diagram

Each incoming TWAMP-Test session that is active on the Session-Reflector SHALL be represented by an instance of a test-session object. All items in the test-session object are read-only.

Instances of test-session are indexed by a session identifier (sid). This value is auto-allocated by the TWAMP Server as test session requests are received, and communicated back to the Control-Client in the SID field of the Accept-Session message; see Section 4.3 of TWAMP Reflect Octets and Symmetrical Size Features [RFC6038].

When attempting to retrieve operational data for active test sessions from a Session-Reflector device, the user will not know what sessions are currently active on that device, or what SIDs have been auto-allocated for these test sessions. If the user has network access to the Control-Client device, then it is possible to read the data for this session under client/ctrl-connection/test-session-request/sid and obtain the SID (see Figure 3). The user may then use this SID value as an index to retrieve an individual session-reflector/test-session instance on the Session-Reflector device.

If the user has no network access to the Control-Client device, then the only option is to retrieve all test-session instances from the Session-Reflector device, and then pick out specific test-session instances of interest to the user. This could be problematic if a large number of test sessions are currently active on that device.

Each Session-Reflector TWAMP-Test session contains the following 4-tuple: {parent-connection-client-ip, parent-connection-client-tcp-port, parent-connection-server-ip, parent-connection-server-tcp-port}. This 4-tuple MUST correspond to the equivalent 4-tuple {client-ip, client-tcp-port, server-ip, server-tcp-port} in server/ctrl-connection. This 4-tuple allows the user to trace back from the TWAMP-Test session to the (parent) TWAMP-Control connection that negotiated this test session.

# 5. Data Model

This section formally specifies the TWAMP data model using YANG.

## 5.1. YANG Tree Diagram

This section presents a simplified graphical representation of the TWAMP data model using a YANG tree diagram. Readers should keep in mind that the limit of 72 characters per line forces us to introduce artificial line breaks in some tree diagram nodes. Tree diagrams used in this document follow the notation defined in YANG Tree Diagrams [RFC8340].

```
module: ietf-twamp
    +--rw twamp
       +--rw client {control-client}?
       |  +--rw admin-state?             boolean
       |  +--rw mode-preference-chain* [priority]
       |  |  +--rw priority    uint16
       |  |  +--rw mode?       twamp-modes
       |  +--rw key-chain* [key-id]
       |  |  +--rw key-id        string
       |  |  +--rw secret-key?   binary
       |  +--rw ctrl-connection* [name]
       |     +--rw name                    string
       |     +--rw client-ip?              inet:ip-address
       |     +--rw server-ip               inet:ip-address
       |     +--rw server-tcp-port?        inet:port-number
       |     +--rw control-packet-dscp?    inet:dscp
       |     +--rw key-id?                 string
```

```
     |      +--rw max-count-exponent?     uint8
     |      +--ro client-tcp-port?        inet:port-number
     |      +--ro server-start-time?      uint64
     |      +--ro repeat-count?           uint64
     |      +--ro state?
     |      |       control-client-connection-state
     |      +--ro selected-mode?          twamp-modes
     |      +--ro token?                  binary
     |      +--ro client-iv?              binary
     |      +--rw test-session-request* [name]
     |         +--rw name                    string
     |         +--rw sender-ip?              inet:ip-address
     |         +--rw sender-udp-port?        union
     |         +--rw reflector-ip            inet:ip-address
     |         +--rw reflector-udp-port?     inet:port-number
     |         +--rw timeout?                uint64
     |         +--rw padding-length?         uint32
     |         +--rw test-packet-dscp?       inet:dscp
     |         +--rw start-time?             uint64
     |         +--rw repeat?                 uint32
     |         +--rw repeat-interval?        uint32
     |         +--rw pm-reg-list* [pm-index]
     |         |  +--rw pm-index    uint16
     |         +--ro state?                  test-session-state
     |         +--ro sid?                    string
     +--rw server {server}?
     |  +--rw admin-state?           boolean
     |  +--rw server-tcp-port?       inet:port-number
     |  +--rw servwait?              uint32
     |  +--rw control-packet-dscp?   inet:dscp
     |  +--rw count?                 uint8
     |  +--rw max-count-exponent?    uint8
     |  +--rw modes?                 twamp-modes
     |  +--rw key-chain* [key-id]
     |  |  +--rw key-id        string
     |  |  +--rw secret-key?   binary
     |  +--ro ctrl-connection*
     |          [client-ip client-tcp-port server-ip server-tcp-port]
     |     +--ro client-ip               inet:ip-address
     |     +--ro client-tcp-port         inet:port-number
     |     +--ro server-ip               inet:ip-address
     |     +--ro server-tcp-port         inet:port-number
     |     +--ro state?                  server-ctrl-connection-state
     |     +--ro control-packet-dscp?    inet:dscp
     |     +--ro selected-mode?          twamp-modes
     |     +--ro key-id?                 string
     |     +--ro count?                  uint8
     |     +--ro max-count-exponent?     uint8
```

```
        |      +--ro salt?                  binary
        |      +--ro server-iv?             binary
        |      +--ro challenge?             binary
        +--rw session-sender {session-sender}?
        |  +--rw admin-state?    boolean
        |  +--rw test-session* [name]
        |     +--rw name                     string
        |     +--ro ctrl-connection-name?   string
        |     +--rw fill-mode?               padding-fill-mode
        |     +--rw number-of-packets       uint32
        |     +--rw (packet-distribution)?
        |     |  +--:(periodic)
        |     |  |  +--rw periodic-interval       decimal64
        |     |  +--:(poisson)
        |     |     +--rw lambda                  decimal64
        |     |     +--rw max-interval?           decimal64
        |     +--ro state?                   sender-session-state
        |     +--ro sent-packets?            uint32
        |     +--ro rcv-packets?             uint32
        |     +--ro last-sent-seq?           uint32
        |     +--ro last-rcv-seq?            uint32
        +--rw session-reflector {session-reflector}?
           +--rw admin-state?    boolean
           +--rw refwait?        uint32
           +--ro test-session*
                   [sender-ip sender-udp-port reflector-ip reflector-udp
 -port]
              +--ro sid?                                  string
              +--ro sender-ip                             inet:ip-address
              +--ro sender-udp-port
              |       dynamic-port-number
              +--ro reflector-ip                          inet:ip-address
              +--ro reflector-udp-port                    inet:port-numbe
 r
              +--ro parent-connection-client-ip?          inet:ip-address
              +--ro parent-connection-client-tcp-port?   inet:port-numbe
 r
              +--ro parent-connection-server-ip?          inet:ip-address
              +--ro parent-connection-server-tcp-port?   inet:port-numbe
 r
              +--ro test-packet-dscp?                     inet:dscp
              +--ro sent-packets?                         uint32
              +--ro rcv-packets?                          uint32
              +--ro last-sent-seq?                        uint32
              +--ro last-rcv-seq?                         uint32
```

Figure 7: YANG Tree Diagram.

5.2. YANG Module

This section presents the YANG module for the TWAMP data model defined in this document. The module imports definitions from Common YANG Data Types [RFC6991], and references NTPv4 Specification [RFC5905], Framework for IP Performance Metrics [RFC2330], Randomness Requirements for Security [RFC4086], OWAMP [RFC4656], TWAMP [RFC5357], More Features for TWAMP [RFC5618], Individual Session Control Feature [RFC5938], TWAMP Reflect Octets and Symmetrical Size Features [RFC6038], Advances Stream and Sampling Framework [RFC7312], IKEv2-Derived Shared Secret Key for OWAMP and TWAMP [RFC7717], and OWAMP and TWAMP Well-Known Port Assignments [I-D.ietf-ippm-port-twamp-test].

```
<CODE BEGINS> file "ietf-twamp@2018-07-02.yang"

module ietf-twamp {
  yang-version 1.1;
  namespace urn:ietf:params:xml:ns:yang:ietf-twamp;
  prefix ietf-twamp;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Types.";
  }

  organization
    "IETF IPPM (IP Performance Metrics) Working Group";

  contact
    "WG Web: http://tools.ietf.org/wg/ippm/
     WG List: ippm@ietf.org

     Editor: Ruth Civil
             gcivil@ciena.com
     Editor: Al Morton
             acmorton@att.com
     Editor: Reshad Rehman
             rrahman@cisco.com
     Editor: Mahesh Jethanandani
             mjethanandani@gmail.com
     Editor: Kostas Pentikousis
             k.pentikousis@travelping.com";


  description
    "This YANG module specifies a vendor-independent data
```

```
        model for the Two-Way Active Measurement Protocol (TWAMP).

        The data model covers four TWAMP logical entities, namely,
        Control-Client, Server, Session-Sender, and Session-Reflector,
        as illustrated in the annotated TWAMP logical model (Fig. 1
        of RFC XXXX).

        This YANG module uses features to indicate which of the four
        logical entities are supported by a TWAMP implementation.

        Copyright (c) 2018 IETF Trust and the persons identified as
        the document authors.  All rights reserved.
        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD
        License set forth in Section 4.c of the IETF Trust's Legal
        Provisions Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

     revision 2018-07-02 {
       description
         "Initial Revision.

         Covers RFC 5357, RFC 5618, RFC 5938, RFC 6038, RFC 7717, and
         draft-ietf-ippm-metric-registry";

       reference
         "RFC XXXX: TWAMP YANG Data Model.";
     }


     /*
      * Typedefs
      */

     typedef twamp-modes {
       type bits {
         bit unauthenticated {
           position 0;
           description
             "Unauthenticated mode, in which no encryption or
             authentication is applied in TWAMP-Control and
             TWAMP-Test. KeyID, Token, and Client-IV are not used in
             the Set-Up-Response message. See Section 3.1 of
             RFC 4656.";
```

```
          reference
            "RFC 4656: A One-way Active Measurement Protocol
             (OWAMP)";
        }
        bit authenticated {
          position 1;
          description
            "Authenticated mode, in which the Control-Client and
             Server possess a shared secret thus prohibiting
             'theft of service'. As per Section 6 of RFC 4656,
             in 'authenticated mode, the timestamp is in the clear
             and is not protected cryptographically in any way,
             while the rest of the message has the same protection
             as in encrypted mode. This mode allows one to trade off
             cryptographic protection against accuracy of
             timestamps.'";
          reference
            "RFC 4656: A One-way Active Measurement Protocol
             (OWAMP)";
        }
        bit encrypted {
          position 2;
          description
            "Encrypted mode 'makes it impossible to alter
             timestamps undetectably' [Section 6 of RFC 4656].
             See also Section 4 of RFC 7717.";
          reference
            "RFC 4656: A One-way Active Measurement Protocol
             (OWAMP)";
        }
        bit unauth-test-encrpyt-control {
          position 3;
          description
            "When using the Mixed Security Mode, the TWAMP-Test
             protocol follows the Unauthenticated mode and the
             TWAMP-Control protocol the Encrypted mode.";
          reference
            "RFC 5618: Mixed Security Mode for the Two-Way Active
             Measurement Protocol (TWAMP)";
        }
        bit individual-session-control {
          position 4;
          description
            "This mode enables individual test sessions using
             Session Identifiers.";
          reference
            "RFC 5938: Individual Session Control Feature
             for the Two-Way Active Measurement Protocol (TWAMP)";
```

```
        }
        bit reflect-octets {
          position 5;
          description
            "This mode indicates the reflect octets capability.";
          reference
            "RFC 6038: Two-Way Active Measurement Protocol (TWAMP)
             Reflect Octets and Symmetrical Size Features";
        }
        bit symmetrical-size {
          position 6;
          description
            "This mode indicates support for the symmetrical size
             sender test packet format.";
          reference
            "RFC 6038: Two-Way Active Measurement Protocol (TWAMP)
             Reflect Octets and Symmetrical Size Features";
        }
        bit IKEv2Derived {
          position 7;
          description
            "In this mode the the shared key is derived
             from an IKEv2 security association (SA).";
          reference
            "RFC 7717: IKEv2-Derived Shared Secret Key for
             the One-Way Active Measurement Protocol (OWAMP)
             and Two-Way Active Measurement Protocol (TWAMP)";
        }
      }
      description
        "Specifies the configurable TWAMP-Modes supported during a
         TWAMP-Control Connection setup between a Control-Client
         and a Server. Section 7 of RFC 7717 summarizes the
         TWAMP-Modes registry and points to their formal
         specification.";
    }

    typedef control-client-connection-state {
      type enumeration {
        enum active {
          description
            "Indicates an active TWAMP-Control connection to
             Server.";
        }
        enum idle {
          description
            "Indicates an idle TWAMP-Control connection to Server.";
        }
```

```
        }
        description
          "Indicates the Control-Client TWAMP-Control connection
           state.";
      }

      typedef test-session-state {
        type enumeration {
          enum accepted {
            value 0;
            description
              "Indicates an accepted TWAMP-Test session request.";
          }
          enum failed {
            value 1;
            description
              "Indicates a TWAMP-Test session failure due to
               some unspecified reason (catch-all).";
          }
          enum internal-error {
            value 2;
            description
              "Indicates a TWAMP-Test session failure due to
               an internal error.";
          }
          enum not-supported {
            value 3;
            description
              "Indicates a TWAMP-Test session failure because
               some aspect of the TWAMP-Test session request
               is not supported.";
          }
          enum permanent-resource-limit {
            value 4;
            description
              "Indicates a TWAMP-Test session failure due to
               permanent resource limitations.";
          }
          enum temp-resource-limit {
            value 5;
            description
              "Indicates a TWAMP-Test session failure due to
               temporary resource limitations.";
          }
        }
        description
          "Indicates the Control-Client TWAMP-Test session state.";
      }
```

```
     typedef server-ctrl-connection-state {
       type enumeration {
         enum active {
           description
             "Indicates an active TWAMP-Control connection
              to the Control-Client.";
         }
         enum servwait {
           description
             "Indicates that the TWAMP-Control connection to the
              Control-Client is in SERVWAIT as per the definition of
              Section 3.1 of RFC 5357.";
         }
       }
       description
         "Indicates the Server TWAMP-Control connection state.";
     }

     typedef sender-session-state {
       type enumeration {
         enum active {
           description
             "Indicates that the TWAMP-Test session is active.";
         }
         enum failure {
           description
             "Indicates that the TWAMP-Test session has failed.";
         }
       }
       description
         "Indicates the Session-Sender TWAMP-Test session state.";
     }

     typedef padding-fill-mode {
       type enumeration {
         enum zero {
           description
             "TWAMP-Test packets are padded with all zeros.";
         }
         enum random {
           description
             "TWAMP-Test packets are padded with pseudo-random
             numbers.";
         }
       }
       description
         "Indicates what type of packet padding is used in the
          TWAMP-Test packets.";
```

```
     }

     typedef dynamic-port-number {
       type inet:port-number {
         range 49152..65535;
       }
       description "Dynamic range for port numbers.";
     }

     /*
      * Features
      */

     feature control-client {
       description
         "Indicates that the device supports configuration of the
          TWAMP Control-Client logical entity.";
     }

     feature server {
       description
         "Indicates that the device supports configuration of the
          TWAMP Server logical entity.";
     }

     feature session-sender {
       description
         "Indicates that the device supports configuration of the
          TWAMP Session-Sender logical entity.";
     }

     feature session-reflector {
       description
         "Indicates that the device supports configuration of the
          TWAMP Session-Reflector logical entity.";
     }


     /*
      * Reusable node groups
      */

     grouping key-management {
       list key-chain {
         key key-id;
         leaf key-id {
           type string {
             length 1..80;
```

```
          }
          description
            "KeyID used for a TWAMP-Control connection. As per
             Section 3.1 of RFC 4656, KeyID is 'a UTF-8 string, up to
             80 octets in length' and is used to select which 'shared
             shared secret the [Control-Client] wishes to use to
             authenticate or encrypt'.";
          }
          leaf secret-key {
            type binary;
            description
              "The secret key corresponding to the KeyID for this
               TWAMP-Control connection.";
          }
          description
            "Relates KeyIDs with their respective secret keys
             in a TWAMP-Control connection.";
      }
      description
          "Used by the Control-Client and Server for TWAMP-Control
           key management.";
    }

    grouping maintenance-statistics {
      leaf sent-packets {
        type uint32;
        config false;
        description
          "Indicates the number of packets sent.";
      }

      leaf rcv-packets {
        type uint32;
        config false;
        description
          "Indicates the number of packets received.";
      }

      leaf last-sent-seq {
        type uint32;
        config false;
        description
          "Indicates the last sent sequence number.";
      }

      leaf last-rcv-seq {
        type uint32;
        config false;
```

```
        description
          "Indicates the last received sequence number.";
      }
      description
        "Used for TWAMP-Test maintenance statistics.";
    }

    grouping count {
      leaf count {
        type uint8 {
          range "10..31";
        }
        default 15;
        description
          "Parameter communicated to the Control-Client as part of
           the Server Greeting message and used for deriving a key
           from a shared secret as per Section 3.1 of  RFC 4656:
           MUST be a power of 2 and at least 1024. It is configured
           by providing said power. For example, configuring 20 here
           means count 2^20 = 1048576. The default is 15,
           meaning 2^15 = 32768.";
        }
     description
       "Reusable data structure for count, which is used both in the
        Server and the Control-Client.";
    }

    grouping max-count-exponent {
      leaf max-count-exponent {
        type uint8 {
          range 10..31;
        }
        default 20;
        description
          "This parameter limits the maximum Count value, which MUST
           be a power of 2 and at least 1024 as per RFC 5357. It is
           configured by providing said power. For example,
           configuring 10 here means max count 2^10 = 1024.
           The default is 20, meaning 2^20 = 1048576.

           A TWAMP Server uses this configured value in the
           Server-Greeting message sent to the Control-Client.

           A TWAMP Control-Client uses this configured value to
           prevent denial-of-service (DOS) attacks by closing the
           control connection to the Server if it 'receives a
           Server-Greeting message with Count greater that its
           maximum configured value', as per Section 6 of RFC 5357.
```

```
           Further, note that according to Section 6 of RFC 5357:

           'If an attacking system sets the maximum value in
           Count (2**32), then the system under attack would stall
           for a significant period of time while it attempts to
           generate keys.

           TWAMP-compliant systems SHOULD have a configuration
           control to limit the maximum count value. The default
           max-count-exponent value SHOULD be 15 which corresponds
           to a maximum value of 2**15 or 32768.'

           RFC 5357 does not qualify 'significant period' in terms of
           time, but it is clear that this depends on the processing
           capacity available and operators need to pay attention to
           this security consideration.";
        }
      description
        "Reusable data structure for max-count which is used both at
         the Control-Client and the Server containers.";
    }


    /*
     * Configuration data nodes
     */

    container twamp {
      description
        "TWAMP logical entity configuration grouping of four models
        which correspond to the four TWAMP logical entities
        Control-Client, Server, Session-Sender, and Session-Reflector
        as illustrated in Fig. 1 of RFC XXXX.";

      container client {
        if-feature control-client;
        description
          "Configuration of the TWAMP Control-Client logical
           entity.";

        leaf admin-state {
          type boolean;
          default true;
          description
            "Indicates whether the device is allowed to operate as a
             TWAMP Control-Client.";
        }
```

```
         list mode-preference-chain {
           key priority;
           unique mode;
           leaf priority {
             type uint16;
             description
               "Indicates the Control-Client Mode preference priority
                expressed as a 16-bit unsigned integer. Values for the
                priority start with zero, the highest priority, and
                decreasing priority value is indicated by every increase
                in value by one.";
           }
           leaf mode {
             type twamp-modes;
             description
               "The supported TWAMP Mode matching the corresponding
                priority.";

           }
           description
             "Indicates the Control-Client preferred order of use of
              the supported TWAMP Modes.

              Depending on the Modes available in the TWAMP Server
              Greeting message (see Fig. 2 of RFC 7717), the
              Control-Client MUST choose the highest priority
              Mode from the configured mode-preference-chain list.";
         }

         uses key-management;

         list ctrl-connection {
           key name;
           description
             "List of TWAMP Control-Client control connections.
              Each item in the list describes a control connection
              that will be initiated by this Control-Client";

           leaf name {
             type string;
             description
               "A unique name used as a key to identify this
                individual TWAMP-Control connection on the
                Control-Client device.";
           }
           leaf client-ip {
             type inet:ip-address;
             description
```

```
              "The IP address of the local Control-Client device,
               to be placed in the source IP address field of the
               IP header in TWAMP-Control (TCP) packets belonging
               to this control connection. If not configured, the
               device SHALL choose its own source IP address.";
          }
          leaf server-ip {
            type inet:ip-address;
            mandatory true;
            description
              "The IP address of the remote Server device, which the
               TWAMP-Control connection will be initiated to.";
          }

          leaf server-tcp-port {
            type inet:port-number;
            default 862;
            description
              "This parameter defines the TCP port number that is
               to be used by this outgoing TWAMP-Control connection.
               Typically, this is the well-known TWAMP-Control
               port number (862) as per RFC 5357 However, there are
               known realizations of TWAMP in the field that were
               implemented before this well-known port number was
               allocated. These early implementations allowed the
               port number to be configured. This parameter is
               therefore provided for backward compatibility
               reasons.";
          }

          leaf control-packet-dscp {
            type inet:dscp;
            default 0;
            description
              "The DSCP value to be placed in the IP header of
               TWAMP-Control (TCP) packets generated by this
               Control-Client.";
          }

          leaf key-id {
            type string {
              length 1..80;
            }
            description
             "Indicates the KeyID value selected for this
              TWAMP-Control connection.";
          }
```

```
          uses max-count-exponent;

          leaf client-tcp-port {
            type inet:port-number;
            config false;
            description
              "Indicates the source TCP port number used in the
               TWAMP-Control packets belonging to this control
               connection.";
          }

          leaf server-start-time {
            type uint64;
            config false;
            description
              "Indicates the Start-Time advertised by the Server in
               the Server-Start message (RFC 4656, Section 3.1),
               representing the time when the current
               instantiation of the Server started operating.
               The timestamp format follows RFC 5905
               according to Section 4.1.2 of RFC 4656.";
            reference
              "RFC 4656: OWAMP, Section 3.1 and 4.1.2,
               RFC 5905: NTPv4 Specification.";
          }

          leaf repeat-count {
            type uint64;
            config false;
            description
              "Indicates how many times the test session has been
               repeated. When a test is running, this value will be
               greater than 0. If the repeat parameter is non-zero,
               this value is smaller than or equal to the repeat
               parameter.";
          }
          leaf state {
            type control-client-connection-state;
            config false;
            description
              "Indicates the current state of the TWAMP-Control
               connection state.";
          }

          leaf selected-mode {
            type twamp-modes;
            config false;
            description
```

```
               "The TWAMP Mode that the Control-Client has chosen for
                this control connection as set in the Mode field of
                the Set-Up-Response message";
             reference
               "RFC 4656, Section 3.1.";
           }

           leaf token {
               type binary {
               length 64;
             }
             config false;
             description
               "This parameter holds the 64 octets containing the
                concatenation of a 16-octet Challenge, a 16-octet AES
                Session-key used for encryption, and a 32-octet
                HMAC-SHA1 Session-key used for authentication; see
                also the last paragraph of Section 6 in RFC 4656.

                If the Mode defined in RFC 7717 is selected
                (selected-mode), Token is limited to 16 octets.";
             reference
               "RFC 4086: Randomness Requirements for Security

                RFC 7717: IKEv2-Derived Shared Secret Key for the
                One-Way Active Measurement Protocol (OWAMP) and
                Two-Way Active Measurement Protocol (TWAMP)";
           }

           leaf client-iv {
             type binary {
               length 16;
             }
             config false;
             description
               "Indicates the Control-Client Initialization Vector
                (Client-IV), that is generated randomly by the
                Control-Client. As per RFC 4656:

                 Client-IV merely needs to be unique (i.e., it MUST
                 never be repeated for different sessions using the
                 same secret key; a simple way to achieve that without
                 the use of cumbersome state is to generate the
                 Client-IV values using a cryptographically secure
                 pseudo-random number source.

                 If the Mode defined in RFC 7717 is selected
                 (selected-mode), Client-IV is limited to 12 octets.";
```

```
            reference
              "RFC 4656: A One-way Active Measurement Protocol
               (OWAMP).

               RFC 7717: IKEv2-Derived Shared Secret Key for the
               One-Way Active Measurement Protocol (OWAMP) and
               Two-Way Active Measurement Protocol (TWAMP)";
          }

          list test-session-request {
            key name;
            description
              "Information associated with the Control-Client
               for this test session";

            leaf name {
              type string;
              description
                "A unique name to be used for identification of
                 this TWAMP-Test session on the Control-Client.";
            }

            leaf sender-ip {
              type inet:ip-address;
              description
                "The IP address of the Session-Sender device,
                 which is to be placed in the source IP address
                 field of the IP header in TWAMP-Test (UDP) packets
                 belonging to this test session. This value will be
                 used to populate the sender address field of the
                 Request-TW-Session message.

                 If not configured, the device SHALL choose its own
                 source IP address.";
            }

            leaf sender-udp-port {
              type union {
                type dynamic-port-number;
                type enumeration {
                  enum autoallocate {
                    description
                      "Indicates that the Contol-Client will
                       auto-allocate the TWAMP-Test (UDP) port number
                       from the dynamic port range.";
                  }
                }
              }
```

```
            default autoallocate;
            description
              "The UDP port number that is to be used by
               the Session-Sender for this TWAMP-Test session.
               The number is restricted to the dynamic port range.

               By default the Control-Client SHALL auto-allocate a
               UDP port number for this TWAMP-Test session.

               The configured (or auto-allocated) value is
               advertised in the Sender Port field of the
               Request-TW-session message (see Section 3.5 of
               RFC 5357). Note that in the scenario where a device
               auto-allocates a UDP port number for a session, and
               the repeat parameter for that session indicates that
               it should be repeated, the device is free to
               auto-allocate a different UDP port number when it
               negotiates the next (repeated) iteration of this
               session.";
          }

          leaf reflector-ip {
            type inet:ip-address;
            mandatory true;
            description
              "The IP address belonging to the remote
               Session-Reflector device to which the TWAMP-Test
               session will be initiated. This value will be
               used to populate the receiver address field of
               the Request-TW-Session message.";
          }

          leaf reflector-udp-port {
            type inet:port-number {
              range "862 | 49152..65535";
            }
            description
              "This parameter defines the UDP port number that
               will be used by the Session-Reflector for
               this TWAMP-Test session. The default number is
               within the dynamic port range and is to be placed
               in the Receiver Port field of the Request-TW-Session
               message. The well-known port (862) MAY be
               used.";
            reference
              "draft-ietf-ippm-port-twamp-test: OWAMP and TWAMP
               Well-Known Port Assignments.";
          }
```

```
            leaf timeout {
              type uint64;
              units seconds;
              default 2;
              description
                "The length of time (in seconds) that the
                 Session-Reflector should continue to respond to
                 packets belonging to this TWAMP-Test session after
                 a Stop-Sessions TWAMP-Control message has been
                 received.

                 This value will be placed in the Timeout field of
                 the Request-TW-Session message.";
              reference
                "RFC 5357: TWAMP, Section 3.5.";
            }

            leaf padding-length {
              type uint32 {
                range 64..4096;
              }
              description
                  "The number of padding bytes to be added to the
                   TWAMP-Test (UDP) packets generated by the
                   Session-Sender.

                   This value will be placed in the Padding Length
                   field of the Request-TW-Session message.";
              reference
                "RFC 4656, Section 3.5.";
            }

            leaf test-packet-dscp {
              type inet:dscp;
              default 0;
              description
                "The DSCP value to be placed in the IP header
                 of TWAMP-Test packets generated by the
                 Session-Sender, and in the UDP header of the
                 TWAMP-Test response packets generated by the
                 Session-Reflector for this test session.

                 This value will be placed in the Type-P Descriptor
                 field of the Request-TW-Session message";
              reference
                "RFC 5357.";
            }
```

```
            leaf start-time {
              type uint64;
              default 0;
              description
                "Time when the session is to be started
                 (but not before the TWAMP Start-Sessions command
                 is issued; see Section 3.4 of RFC 5357).

                 The start-time value is placed in the Start Time
                 field of the Request-TW-Session message.

                 The timestamp format follows RFC 5905 as per
                 Section 3.5 of RFC 4656.

                 The default value of 0 indicates that the session
                 will be started as soon as the Start-Sessions
                 message is received.";
            }

            leaf repeat {
              type uint32 {
                range 0..4294967295;
              }
              default 0;
              description
                "This value determines if the TWAMP-Test session must
                 be repeated. When a test session has completed, the
                 repeat parameter is checked.

                 The default value of 0 indicates that the session
                 MUST NOT be repeated.

                 If the repeat value is 1 through 4,294,967,294
                 then the test session SHALL be repeated using the
                 information in repeat-interval parameter, and the
                 parent TWAMP-Control connection for this test
                 session is restarted to negotiate a new instance
                 of this TWAMP-Test session.

                 A value of 4,294,967,295 indicates that the test
                 session SHALL be repeated *forever* using the
                 information in repeat-interval parameter, and SHALL
                 NOT decrement the value.";
            }

            leaf repeat-interval  {
              when "../repeat!='0'" {
                description
```

```
                   "This parameter determines the timing of repeated
                    TWAMP-Test sessions when repeat is more than 0.

                    When the value of repeat-interval is 0, the
                    negotiation of a new test session SHALL begin
                    immediately after the previous test session
                    completes. Otherwise, the Control-Client will
                    wait for the number of seconds specified in the
                    repeat-interval parameter before negotiating the
                    new instance of this TWAMP-Test session.";
               }
               type uint32;
               units seconds;
               default 0;
               description
                 "Repeat interval (in seconds).";
             }

             list pm-reg-list {
               key pm-index;
               leaf pm-index {
                 type uint16;
                 description
                   "Numerical index value of a Registered Metric
                    in the Performance Metric Registry
                    (see ietf-ippm-metric-registry). Output statistics
                    are specified in the corresponding Registry
                    entry.";
               }
               description
                 "A list of one or more Performance Metric Registry
                  Index values, which communicate packet stream
                  characteristics along with one or more metrics
                  to be measured.

                  All members of the pm-reg-list MUST have the same
                  stream characteristics, such that they combine
                  to specify all metrics that shall be measured on
                  a single stream.";
               reference
                 "ietf-ippm-metric-registry: Registry for
                  Performance Metrics";
             }

             leaf state {
               type test-session-state;
               config false;
               description
```

```
                "Indicates the TWAMP-Test session state, accepted or
                 indication of an error.";
              reference
                "Section 3.5 of RFC 5357.";
            }
            leaf sid {
              type string;
              config false;
              description
                "The SID allocated by the Server for this TWAMP-Test
                 session, and communicated back to the Control-Client
                 in the SID field of the Accept-Session message";
              reference
                "Section 4.3 of RFC 6038.";
            }
          }
        }
      }

      container server {
        if-feature server;
        description
          "Configuration of the TWAMP Server logical entity.";

        leaf admin-state {
          type boolean;
          default true;
          description
            "Indicates whether the device is allowed to operate
             as a TWAMP Server.";
        }

        leaf server-tcp-port {
          type inet:port-number;
          default 862;
          description
            "This parameter defines the well known TCP port number
             that is used by TWAMP-Control. The Server will listen
             on this port number for incoming TWAMP-Control
             connections. Although this is defined as a fixed value
             (862) in RFC 5357, there are several realizations of
             TWAMP in the field that were implemented before this
             well-known port number was allocated. These early
             implementations allowed the port number to be
             configured. This parameter is therefore provided for
             backward compatibility reasons.";
        }
```

```
        leaf servwait {
          type uint32 {
            range 1..604800;
          }
          units seconds;
          default 900;
          description
            "TWAMP-Control (TCP) session timeout, in seconds.
             According to Section 3.1 of RFC 5357,

             Server MAY discontinue any established control
             connection when no packet associated with that
             connection has been received within SERVWAIT seconds.";
        }

        leaf control-packet-dscp {
          type inet:dscp;
          description
            "The DSCP value to be placed in the IP header of
             TWAMP-Control (TCP) packets generated by the Server.

             Section 3.1 of  RFC 5357 specifies that the server
             SHOULD use the DSCP value from the Control-Clients
             TCP SYN. However, for practical purposes TWAMP will
             typically be implemented using a general purpose TCP
             stack provided by the underlying operating system,
             and such a stack may not provide this information to the
             user. Consequently, it is not always possible to
             implement the behavior described in RFC 5357 in an
             OS-portable version of TWAMP.

             The default behavior if this item is not set is to use
             the DSCP value from the Control-Clients TCP SYN.";
           reference
             "Section 3.1 of RFC 5357.";
        }

        uses count;

        uses max-count-exponent;

        leaf modes {
          type twamp-modes;
          description
            "The bit mask of TWAMP Modes this Server instance
             is willing to support; see IANA TWAMP Modes Registry.";
        }
```

```
         uses key-management;

         list ctrl-connection {
           key "client-ip client-tcp-port server-ip server-tcp-port";
           config false;
           description
             "List of all incoming TWAMP-Control (TCP) connections.";

           leaf client-ip {
             type inet:ip-address;
             description
               "The IP address on the remote Control-Client device,
                which is the source IP address used in the
                TWAMP-Control (TCP) packets belonging to this control
                connection.";
           }

           leaf client-tcp-port {
             type inet:port-number;
             description
               "The source TCP port number used in the TWAMP-Control
                (TCP) packets belonging to this control connection.";
           }

           leaf server-ip {
             type inet:ip-address;
             description
               "The IP address of the local Server device, which is
                the destination IP address used in the
                TWAMP-Control (TCP) packets belonging to this control
                connection.";
           }

           leaf server-tcp-port {
             type inet:port-number;
             description
               "The destination TCP port number used in the
                TWAMP-Control (TCP) packets belonging to this
                control connection. This will usually be the
                same value as the server-tcp-port configured
                under twamp/server. However, in the event that
                the user re-configured server/server-tcp-port
                after this control connection was initiated, this
                value will indicate the server-tcp-port that is
                actually in use for this control connection.";
           }

           leaf state {
```

```
          type server-ctrl-connection-state;
          description
            "Indicates the Server TWAMP-Control connection state.";
        }

        leaf control-packet-dscp {
          type inet:dscp;
          description
            "The DSCP value used in the IP header of the
             TWAMP-Control (TCP) packets sent by the Server
             for this control connection. This will usually
             be the same value as is configured in the
             control-packet-dscp parameter under the twamp/server
             container.  However, in the event that the user
             re-configures server/dscp after this control
             connection is already in progress, this read-only
             value will show the actual dscp value in use by this
             TWAMP-Control connection.";
        }

        leaf selected-mode {
          type twamp-modes;
          description
            "The Mode that was chosen for this TWAMP-Control
             connection as set in the Mode field of the
             Set-Up-Response message.";
        }

        leaf key-id {
          type string {
            length 1..80;
          }
          description
            "The KeyID value that is in use by this TWAMP-Control
             connection as selected by Control-Client.";
        }

        uses count {
          description
            "The count value that is in use by this TWAMP-Control
             connection. This will usually be the same value
             as is configured under twamp/server. However, in the
             event that the user re-configured server/count
             after this control connection is already in progress,
             this read-only value will show the actual count that
             is in use for this TWAMP-Control connection.";
        }
```

```
          uses max-count-exponent {
            description
              "This read-only value indicates the actual max-count in
               use for this control connection. Usually this would be
               the same value as configured under twamp/server.";
          }

          leaf salt {
            type binary {
              length 16;
            }
            description
              "A parameter used in deriving a key from a
               shared secret as described in Section 3.1 of RFC 4656.
               It is communicated to the Control-Client as part of
               the Server Greeting message.";
          }

          leaf server-iv {
            type binary {
              length 16;
            }
            description
              "The Server Initialization Vector
               (IV) generated randomly by the Server.";
          }

          leaf challenge {
            type binary {
              length 16;
            }
            description
              "A random sequence of octets generated by the Server.
               As described in client/token, Challenge is used
               by the Control-Client to prove possession of a
               shared secret.";
          }
        }
      }

      container session-sender {
        if-feature session-sender;
        description
          "Configuration of the TWAMP Session-Sender logical entity";
        leaf admin-state {
          type boolean;
          default true;
          description
```

```
            "Indicates whether the device is allowed to operate
             as a TWAMP Session-Sender.";
        }

        list test-session{
          key name;
          description
            "List of TWAMP Session-Sender test sessions.";

          leaf name {
            type string;
            description
              "A unique name for this TWAMP-Test session to be used
               for identifying this test session by the
               Session-Sender logical entity.";
          }

          leaf ctrl-connection-name {
            type string;
            config false;
            description
              "The name of the parent TWAMP-Control connection that
               is responsible for negotiating this TWAMP-Test
               session.";
          }

          leaf fill-mode {
            type padding-fill-mode;
            default zero;
            description
              "Indicates whether the padding added to the
               TWAMP-Test (UDP) packets will contain pseudo-random
               numbers, or whether it should consist of all zeroes,
               as per Section 4.2.1 of RFC 5357.";
          }

          leaf number-of-packets {
            type uint32;
            mandatory true;
            description
              "The overall number of TWAMP-Test (UDP) packets to be
               transmitted by the Session-Sender for this test
               session.";
          }

          choice packet-distribution {
            description
              "Indicates the distribution to be used for transmitting
```

```
                the TWAMP-Test (UDP) packets.";
             case periodic {
               leaf periodic-interval {
                 type decimal64 {
                   fraction-digits 5;
                 }
                 units seconds;
                 mandatory true;
                 description
                   "Indicates the time to wait (in seconds) between
                    the first bits of TWAMP-Test (UDP) packet
                    transmissions for this test session.";
                 reference
                   "RFC 3432: Network performance measurement
                              with periodic streams";
               }
             }
             case poisson {
               leaf lambda {
                 type decimal64 {
                   fraction-digits 5;
                 }
                 units seconds;
                 mandatory true;
                 description
                   "Indicates the average time interval (in seconds)
                    between packets in the Poisson distribution.
                    The packet is calculated using the reciprocal of
                    lambda and the TWAMP-Test packet size (which
                    depends on the selected Mode and the packet
                    padding).";
                reference
                  "RFC 2330: Framework for IP Performance Metrics";
               }
               leaf max-interval {
                 type decimal64 {
                   fraction-digits 5;
                 }
                 units seconds;
                 description
                   "Indicates the maximum time (in seconds)
                    between packet transmissions.";
                 reference
                   "RFC 7312: Advanced Stream and Sampling Framework
                              for IP Performance Metrics (IPPM)";
               }
             }
           }
```

```
          leaf state {
            type sender-session-state;
            config false;
            description
              "Indicates the Session-Sender test session state.";
          }

          uses maintenance-statistics;
        }
      }

      container session-reflector {
        if-feature session-reflector;
        description
          "Configuration of the TWAMP Session-Reflector logical
           entity";

        leaf admin-state {
          type boolean;
          default true;
          description
            "Indicates whether the device is allowed to operate
             as a TWAMP Session-Reflector.";
        }

        leaf refwait {
          type uint32 {
            range 1..604800;
          }
          units seconds;
          default 900;
          description
            "The Session-Reflector MAY discontinue any session that
             has been started when no packet associated with that
             session has been received for REFWAIT seconds. As per
             Section 3.1 of RFC 5357, this timeout allows a
             Session-Reflector to free up resources in case of
             failure.";
        }

        list test-session {
          key
            "sender-ip sender-udp-port
             reflector-ip reflector-udp-port";
          config false;
          description
            "TWAMP Session-Reflectortest sessions.";
```

```
          leaf sid {
            type string;
            description
              "An auto-allocated identifier for this TWAMP-Test
               session that is unique within the context of this
               Server/Session-Reflector device only. This value
               is communicated to the Control-Client that
               requested the test session in the SID field of the
               Accept-Session message.";
          }

          leaf sender-ip {
            type inet:ip-address;
            description
              "The IP address on the remote device, which is the
               source IP address used in the TWAMP-Test (UDP) packets
               belonging to this test session.";
          }

          leaf sender-udp-port {
            type dynamic-port-number;
            description
              "The source UDP port used in the TWAMP-Test packets
               belonging to this test session.";
          }

          leaf reflector-ip {
            type inet:ip-address;
            description
              "The IP address of the local Session-Reflector
               device, which is the destination IP address used
               in the TWAMP-Test (UDP) packets belonging to this test
               session.";
          }

          leaf reflector-udp-port {
            type inet:port-number {
              range "862 | 49152..65535";
              }
            description
              "The destination UDP port number used in the
               TWAMP-Test (UDP) test packets belonging to this
               test session.";
          }

          leaf parent-connection-client-ip {
            type inet:ip-address;
            description
```

```
              "The IP address on the Control-Client device, which
               is the source IP address used in the TWAMP-Control
               (TCP) packets belonging to the parent control
               connection that negotiated this test session.";
          }

          leaf parent-connection-client-tcp-port {
            type inet:port-number;
            description
              "The source TCP port number used in the TWAMP-Control
               (TCP) packets belonging to the parent control
               connection that negotiated this test session.";
          }

          leaf parent-connection-server-ip {
            type inet:ip-address;
            description
              "The IP address of the Server device, which is the
               destination IP address used in the TWAMP-Control
               (TCP) packets belonging to the parent control
               connection that negotiated this test session.";
          }

          leaf parent-connection-server-tcp-port {
            type inet:port-number;
            description
              "The destination TCP port number used in the
               TWAMP-Control (TCP) packets belonging to the parent
               control connection that negotiated this test
               session.";
          }

          leaf test-packet-dscp {
            type inet:dscp;
            description
              "The DSCP value present in the IP header of
               TWAMP-Test (UDP) packets belonging to this session.";
          }

          uses maintenance-statistics;
        }
      }
    }
  }

  <CODE ENDS>
```

## 6. Data Model Examples

This section presents a simple but complete example of configuring all four entities in Figure 1, based on the YANG module specified in Section 5. The example is illustrative in nature, but aims to be self-contained, i.e. were it to be executed in a real TWAMP implementation it would lead to a correctly configured test session. For completeness, examples are provided for both IPv4 and IPv6.

A more elaborated example, which also includes authentication parameters, is provided in Appendix A.

### 6.1. Control-Client

Figure 8 shows a configuration example for a Control-Client with client/admin-state enabled. In a real implementation following Figure 2 this would permit the initiation of TWAMP-Control connections and TWAMP-Test sessions.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <client>
      <admin-state>true</admin-state>
    </client>
  </twamp>
</config>
```

Figure 8: XML instance enabling Control-Client operation.

The following example shows a Control-Client with two instances of client/ctrl-connection, one called "RouterA" and another called "RouterB". Each TWAMP-Control connection is to a different Server. The control connection named "RouterA" has two test session requests. The TWAMP-Control connection named "RouterB" has no TWAMP-Test session requests.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <client>
      <admin-state>true</admin-state>
      <ctrl-connection>
        <name>RouterA</name>
        <client-ip>203.0.113.1</client-ip>
        <server-ip>203.0.113.2</server-ip>
        <test-session-request>
```

```
             <name>Test1</name>
             <sender-ip>203.0.113.3</sender-ip>
             <sender-udp-port>54001</sender-udp-port>
             <reflector-ip>203.0.113.4</reflector-ip>
             <reflector-udp-port>50001</reflector-udp-port>
             <start-time>0</start-time>
           </test-session-request>
           <test-session-request>
             <name>Test2</name>
             <sender-ip>203.0.113.1</sender-ip>
             <sender-udp-port>54001</sender-udp-port>
             <reflector-ip>203.0.113.2</reflector-ip>
             <reflector-udp-port>50001</reflector-udp-port>
             <start-time>0</start-time>
           </test-session-request>
         </ctrl-connection>
         <ctrl-connection>
           <name>RouterB</name>
           <client-ip>203.0.113.1</client-ip>
           <server-ip>203.0.113.3</server-ip>
         </ctrl-connection>
       </client>
     </twamp>
   </config>

   <?xml version="1.0" encoding="utf-8"?>
   <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
       <client>
         <admin-state>true</admin-state>
         <ctrl-connection>
           <name>RouterA</name>
           <client-ip>2001:DB8:203:0:113::1</client-ip>
           <server-ip>2001:DB8:203:0:113::2</server-ip>
           <test-session-request>
             <name>Test1</name>
             <sender-ip>2001:DB8:203:1:113::3</sender-ip>
             <sender-udp-port>54000</sender-udp-port>
             <reflector-ip>2001:DB8:203:1:113::4</reflector-ip>
             <reflector-udp-port>55000</reflector-udp-port>
             <start-time>0</start-time>
           </test-session-request>
           <test-session-request>
             <name>Test2</name>
             <sender-ip>2001:DB8:203:0:113::1</sender-ip>
             <sender-udp-port>54001</sender-udp-port>
             <reflector-ip>2001:DB8:203:0:113::2</reflector-ip>
             <reflector-udp-port>55001</reflector-udp-port>
```

```
          <start-time>0</start-time>
        </test-session-request>
      </ctrl-connection>
      <ctrl-connection>
        <name>RouterB</name>
        <client-ip>2001:DB8:203:0:113::1</client-ip>
        <server-ip>2001:DB8:203:0:113::3</server-ip>
      </ctrl-connection>
    </client>
  </twamp>
</config>
```

## 6.2. Server

Figure 9 shows a configuration example for a Server with server/admin-state enabled, which permits a device following Figure 2 to respond to TWAMP-Control connections and TWAMP-Test sessions.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <server>
      <admin-state>true</admin-state>
    </server>
  </twamp>
</config>
```

Figure 9: XML instance enabling Server operation.

The following example presents a Server with the TWAMP-Control connection corresponding to the control connection name (client/ctrl-connection/name) "RouterA" presented in Section 6.1.

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <server>
      <admin-state>true</admin-state>
      <ctrl-connection>
        <client-ip>203.0.113.1</client-ip>
        <client-tcp-port>16341</client-tcp-port>
        <server-ip>203.0.113.2</server-ip>
        <server-tcp-port>862</server-tcp-port>
        <state>active</state>
      </ctrl-connection>
    </server>
  </twamp>
</data>
```



```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <server>
      <admin-state>true</admin-state>
      <ctrl-connection>
        <client-ip>2001:DB8:203:0:113::1</client-ip>
        <client-tcp-port>16341</client-tcp-port>
        <server-ip>2001:DB8:203:0:113::2</server-ip>
        <server-tcp-port>862</server-tcp-port>
        <state>active</state>
      </ctrl-connection>
    </server>
  </twamp>
</data>
```


## 6.3. Session-Sender

Figure 10 shows a configuration example for a Session-Sender with session-sender/admin-state enabled, which permits a device following Figure 2 to initiate TWAMP-Test sessions.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <session-sender>
      <admin-state>true</admin-state>
    </session-sender>
  </twamp>
</config>
```


Figure 10: XML instance enabling Session-Sender operation.

The following configuration example shows a Session-Sender with the two TWAMP-Test sessions presented in Section 6.1.


```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <session-sender>
      <admin-state>true</admin-state>
      <test-session>
        <name>Test1</name>
        <ctrl-connection-name>RouterA</ctrl-connection-name>
        <number-of-packets>900</number-of-packets>
        <periodic-interval>1</periodic-interval>
      </test-session>
      <test-session>
        <name>Test2</name>
        <ctrl-connection-name>RouterA</ctrl-connection-name>
        <number-of-packets>900</number-of-packets>
        <lambda>1</lambda>
        <max-interval>2</max-interval>
      </test-session>
    </session-sender>
  </twamp>
</data>
```


## 6.4. Session-Reflector

This configuration example shows a Session-Reflector with session-reflector/admin-state enabled, which permits a device following Figure 2 to respond to TWAMP-Test sessions.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <session-reflector>
      <admin-state>true</admin-state>
    </session-reflector>
  </twamp>
</config>
```


Figure 11: XML instance enabling Session-Reflector operation.

The following example shows the two Session-Reflector TWAMP-Test sessions corresponding to the test sessions presented in Section 6.3.

[note: '\' line wrapping is for formatting only]


```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <session-reflector>
      <admin-state>true</admin-state>
      <test-session>
        <sender-ip>203.0.113.3</sender-ip>
        <sender-udp-port>54000</sender-udp-port>
        <reflector-ip>203.0.113.4</reflector-ip>
        <reflector-udp-port>50001</reflector-udp-port>
        <sid>1232</sid>
        <parent-connection-client-ip>203.0.113.1</parent-connection-\
client-ip>
        <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
        <parent-connection-server-ip>203.0.113.2</parent-connection-\
server-ip>
        <parent-connection-server-tcp-port>862</parent-connection-se\
rver-tcp-port>
        <sent-packets>2</sent-packets>
        <rcv-packets>2</rcv-packets>
        <last-sent-seq>1</last-sent-seq>
        <last-rcv-seq>1</last-rcv-seq>
      </test-session>
      <test-session>
        <sender-ip>203.0.113.1</sender-ip>
        <sender-udp-port>54001</sender-udp-port>
        <reflector-ip>192.0.2.2</reflector-ip>
        <reflector-udp-port>50001</reflector-udp-port>
        <sid>178943</sid>
        <parent-connection-client-ip>203.0.113.1</parent-connection-\
```

```
client-ip>
        <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
        <parent-connection-server-ip>203.0.113.2</parent-connection-\
server-ip>
        <parent-connection-server-tcp-port>862</parent-connection-se\
rver-tcp-port>
        <sent-packets>21</sent-packets>
        <rcv-packets>21</rcv-packets>
        <last-sent-seq>20</last-sent-seq>
        <last-rcv-seq>20</last-rcv-seq>
      </test-session>
    </session-reflector>
  </twamp>
</data>
```

[note: '\' line wrapping is for formatting only]

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <session-reflector>
      <admin-state>true</admin-state>
      <test-session>
        <sender-ip>203.0.113.3</sender-ip>
        <sender-udp-port>54000</sender-udp-port>
        <reflector-ip>203.0.113.4</reflector-ip>
        <reflector-udp-port>54001</reflector-udp-port>
        <sid>1232</sid>
        <parent-connection-client-ip>203.0.113.1</parent-connection-\
client-ip>
        <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
        <parent-connection-server-ip>203.0.113.2</parent-connection-\
server-ip>
        <parent-connection-server-tcp-port>862</parent-connection-se\
rver-tcp-port>
        <sent-packets>2</sent-packets>
        <rcv-packets>2</rcv-packets>
        <last-sent-seq>1</last-sent-seq>
        <last-rcv-seq>1</last-rcv-seq>
      </test-session>
      <test-session>
        <sender-ip>203.0.113.1</sender-ip>
        <sender-udp-port>54001</sender-udp-port>
        <reflector-ip>192.0.2.2</reflector-ip>
        <reflector-udp-port>55001</reflector-udp-port>
        <sid>178943</sid>
```

```
         <parent-connection-client-ip>203.0.113.1</parent-connection-\
client-ip>
         <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
         <parent-connection-server-ip>203.0.113.2</parent-connection-\
server-ip>
         <parent-connection-server-tcp-port>862</parent-connection-se\
rver-tcp-port>
         <sent-packets>21</sent-packets>
         <rcv-packets>21</rcv-packets>
         <last-sent-seq>20</last-sent-seq>
         <last-rcv-seq>20</last-rcv-seq>
       </test-session>
     </session-reflector>
   </twamp>
</data>
```


## 7. Security Considerations

Virtually all existing measurement systems using TWAMP [RFC5357] are administered by the same network operator. Attacks on the measurement infrastructure could be launched by third-parties to commandeer the packet generation capability, corrupt the measurements, or other examples of nefarious acts.

The YANG module specified in Section 5 of this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF [RFC6241] layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF Access Control Module (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of nodes defined in this YANG module which are writeable. These data nodes may be considered sensitive and vulnerable to attacks in some network environments. Ability to write into these nodes without proper protection can have a negative effect on the devices that support this feature.

If written, the 'admin-state' node can cause unintended test sessions to be created. If the node 'number-of-packets' that dictates how many packets are sent in any particular test session is written with a large value, it can cause a test session to run longer than expected. Nodes that are particularly vulnerable include several timeout values put in the protocol to protect against sessions that are not active but are consuming resources. These are the REFWAIT timeout parameter which determine whether to discontinue the session if no packets are received, and nodes 'count' and 'max-count-exponent' which can cause a long time to be spent on PBKDF2 iterations. In addition, 'dscp' node marked with different DSCP markings, can cause the test traffic on the network to be skewed, and the result manipulated. Finally, nodes within 'mode-preference-chain' which specify the 'mode' and 'priority' values and indicate the preferred order of use by an operator, can be manipulated to send unauthenticated or non-encrypted traffic, enabling a MITM attack. Limiting access to these nodes will limit the ability to launch an attack in network environments.

The 'token' node defined in the model, containing a concatenation of a Challenge, AES Session-key used for encryption, and HMAC-SHA1 Session-key used for authentication, is sensitive from a privacy perspective, and can be used to disrupt a test session. The ability to read the field should be limited to the administrator of the test network.

## 8. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in IETF XML Registry [RFC3688], the following registration is requested to be made.

URI: urn:ietf:params:xml:ns:yang:ietf-twamp

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry YANG [RFC6020].

name: ietf-twamp

namespace: urn:ietf:params:xml:ns:yang:ietf-twamp

prefix: twamp

reference: RFC XXXX

## 9. Acknowledgements

We thank Fred Baker, Kevin D'Souza, Gregory Mirsky, Brian Trammell, Robert Sherman, and Marius Georgescu for their thorough and constructive reviews, comments and text suggestions.

Haoxing Shen contributed to the definition of the YANG module in Section 5.

Jan Lindblad and Ladislav Lhokta did thorough reviews of the YANG module and the examples in Appendix A.

Kostas Pentikousis was partially supported by FP7 UNIFY (http://fp7-unify.eu), a research project partially funded by the European Community under the Seventh Framework Program (grant agreement no. 619609). The views expressed here are those of the authors only. The European Commission is not liable for any use that may be made of the information in this document.

## 10. Contributors

Lianshu Zheng.

## 11. References

### 11.1. Normative References

[I-D.ietf-ippm-metric-registry]
Bagnulo, M., Claise, B., Eardley, P., Morton, A., and A. Akhter, "Registry for Performance Metrics", draft-ietf-ippm-metric-registry-14 (work in progress), March 2018.

[I-D.ietf-ippm-port-twamp-test]
Morton, A. and G. Mirsky, "OWAMP and TWAMP Well-Known Port Assignments", draft-ietf-ippm-port-twamp-test-01 (work in progress), March 2018.

[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>.

[RFC3432] Raisanen, V., Grotefeld, G., and A. Morton, "Network performance measurement with periodic streams", RFC 3432, DOI 10.17487/RFC3432, November 2002, <https://www.rfc-editor.org/info/rfc3432>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, DOI 10.17487/RFC4656, September 2006, <https://www.rfc-editor.org/info/rfc4656>.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, DOI 10.17487/RFC5357, October 2008, <https://www.rfc-editor.org/info/rfc5357>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6038] Morton, A. and L. Ciavattone, "Two-Way Active Measurement Protocol (TWAMP) Reflect Octets and Symmetrical Size Features", RFC 6038, DOI 10.17487/RFC6038, October 2010, <https://www.rfc-editor.org/info/rfc6038>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7717] Pentikousis, K., Ed., Zhang, E., and Y. Cui, "IKEv2-Derived Shared Secret Key for the One-Way Active Measurement Protocol (OWAMP) and Two-Way Active Measurement Protocol (TWAMP)", RFC 7717, DOI 10.17487/RFC7717, December 2015, <https://www.rfc-editor.org/info/rfc7717>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[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>.

[UML] ISO/IEC, "Information technology - Open Distributed Processing - Unified Modeling Language", April 2005.

## 11.2. Informative References

[NSC] John, W., Pentikousis, K., et al., "Research directions in network service chaining", Proc. SDN for Future Networks and Services (SDN4FNS), Trento, Italy IEEE, November 2013.

[RFC2330] Paxson, V., Almes, G., Mahdavi, J., and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, DOI 10.17487/RFC2330, May 1998, <https://www.rfc-editor.org/info/rfc2330>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC5618] Morton, A. and K. Hedayat, "Mixed Security Mode for the Two-Way Active Measurement Protocol (TWAMP)", RFC 5618, DOI 10.17487/RFC5618, August 2009, <https://www.rfc-editor.org/info/rfc5618>.

[RFC5938] Morton, A. and M. Chiba, "Individual Session Control Feature for the Two-Way Active Measurement Protocol (TWAMP)", RFC 5938, DOI 10.17487/RFC5938, August 2010, <https://www.rfc-editor.org/info/rfc5938>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC7312] Fabini, J. and A. Morton, "Advanced Stream and Sampling Framework for IP Performance Metrics (IPPM)", RFC 7312, DOI 10.17487/RFC7312, August 2014, <https://www.rfc-editor.org/info/rfc7312>.

[RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <https://www.rfc-editor.org/info/rfc7426>.

[RFC8018] Moriarty, K., Ed., Kaliski, B., and A. Rusch, "PKCS #5: Password-Based Cryptography Specification Version 2.1", RFC 8018, DOI 10.17487/RFC8018, January 2017, <https://www.rfc-editor.org/info/rfc8018>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

## Appendix A. Detailed Data Model Examples

This appendix extends the example presented in Section 6 by configuring more fields such as authentication parameters, DSCP values and so on.

### A.1. Control-Client


```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <client>
      <admin-state>true</admin-state>
      <mode-preference-chain>
        <priority>0</priority>
        <mode>authenticated</mode>
      </mode-preference-chain>
      <mode-preference-chain>
        <priority>1</priority>
```

```
          <mode>unauthenticated</mode>
        </mode-preference-chain>
        <key-chain>
          <key-id>KeyClient1ToRouterA</key-id>
          <secret-key>c2VjcmV0MQ==</secret-key>
        </key-chain>
        <key-chain>
          <key-id>KeyForRouterB</key-id>
          <secret-key>c2VjcmV0Mg0K</secret-key>
        </key-chain>
        <ctrl-connection>
          <name>RouterA</name>
          <client-ip>203.0.113.1</client-ip>
          <server-ip>203.0.113.2</server-ip>
          <control-packet-dscp>32</control-packet-dscp>
          <key-id>KeyClient1ToRouterA</key-id>
          <test-session-request>
            <name>Test1</name>
            <sender-ip>203.0.113.3</sender-ip>
            <sender-udp-port>54000</sender-udp-port>
            <reflector-ip>203.0.113.4</reflector-ip>
            <reflector-udp-port>55000</reflector-udp-port>
            <padding-length>64</padding-length>
            <start-time>0</start-time>
          </test-session-request>
          <test-session-request>
            <name>Test2</name>
            <sender-ip>203.0.113.1</sender-ip>
            <sender-udp-port>54001</sender-udp-port>
            <reflector-ip>203.0.113.2</reflector-ip>
            <reflector-udp-port>55001</reflector-udp-port>
            <padding-length>128</padding-length>
            <start-time>0</start-time>
          </test-session-request>
        </ctrl-connection>
      </client>
    </twamp>
  </data>

  <?xml version="1.0" encoding="utf-8"?>
  <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <client>
        <admin-state>true</admin-state>
        <mode-preference-chain>
          <priority>0</priority>
          <mode>authenticated</mode>
        </mode-preference-chain>
```

```
          <mode-preference-chain>
            <priority>1</priority>
            <mode>unauthenticated</mode>
          </mode-preference-chain>
          <key-chain>
            <key-id>KeyClient1ToRouterA</key-id>
            <secret-key>c2VjcmV0MQ==</secret-key>
          </key-chain>
          <key-chain>
            <key-id>KeyForRouterB</key-id>
            <secret-key>c2VjcmV0Mg0K</secret-key>
          </key-chain>
          <ctrl-connection>
            <name>RouterA</name>
            <client-ip>2001:DB8:203:0:113::1</client-ip>
            <server-ip>2001:DB8:203:0:113::2</server-ip>
            <control-packet-dscp>32</control-packet-dscp>
            <key-id>KeyClient1ToRouterA</key-id>
            <test-session-request>
              <name>Test1</name>
              <sender-ip>2001:DB8:10:1:1::1</sender-ip>
              <sender-udp-port>54000</sender-udp-port>
              <reflector-ip>2001:DB8:10:1:1::2</reflector-ip>
              <reflector-udp-port>55000</reflector-udp-port>
              <padding-length>64</padding-length>
              <start-time>0</start-time>
            </test-session-request>
            <test-session-request>
              <name>Test2</name>
              <sender-ip>2001:DB8:203:0:113::1</sender-ip>
              <sender-udp-port>54001</sender-udp-port>
              <reflector-ip>2001:DB8:203:0:113::2</reflector-ip>
              <reflector-udp-port>55001</reflector-udp-port>
              <padding-length>128</padding-length>
              <start-time>0</start-time>
            </test-session-request>
          </ctrl-connection>
        </client>
      </twamp>
    </data>
```


## A.2. Server


```
    <?xml version="1.0" encoding="utf-8"?>
    <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
        <server>
```

```
          <admin-state>true</admin-state>
          <servwait>1800</servwait>
          <control-packet-dscp>32</control-packet-dscp>
          <modes>authenticated unauthenticated</modes>
          <count>15</count>
          <key-chain>
            <key-id>KeyClient1ToRouterA</key-id>
            <secret-key>c2VjcmV0MQ==</secret-key>
          </key-chain>
          <key-chain>
            <key-id>KeyClient10ToRouterA</key-id>
            <secret-key>c2VjcmV0MTANCg==</secret-key>
          </key-chain>
          <ctrl-connection>
            <client-ip>203.0.113.1</client-ip>
            <client-tcp-port>16341</client-tcp-port>
            <server-ip>203.0.113.2</server-ip>
            <server-tcp-port>862</server-tcp-port>
            <control-packet-dscp>32</control-packet-dscp>
            <selected-mode>unauthenticated</selected-mode>
            <key-id>KeyClient1ToRouterA</key-id>
            <count>15</count>
          </ctrl-connection>
        </server>
      </twamp>
    </data>

    <?xml version="1.0" encoding="utf-8"?>
    <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
        <server>
          <admin-state>true</admin-state>
          <servwait>1800</servwait>
          <control-packet-dscp>32</control-packet-dscp>
          <modes>authenticated unauthenticated</modes>
          <count>15</count>
          <key-chain>
            <key-id>KeyClient1ToRouterA</key-id>
            <secret-key>c2VjcmV0MQ==</secret-key>
          </key-chain>
          <key-chain>
            <key-id>KeyClient10ToRouterA</key-id>
            <secret-key>c2VjcmV0MTANCg==</secret-key>
          </key-chain>
          <ctrl-connection>
            <client-ip>2001:DB8:203:0:113::1</client-ip>
            <client-tcp-port>16341</client-tcp-port>
            <server-ip>2001:DB8:203:0:113::2</server-ip>
```

```
          <server-tcp-port>862</server-tcp-port>
          <control-packet-dscp>32</control-packet-dscp>
          <selected-mode>unauthenticated</selected-mode>
          <key-id>KeyClient1ToRouterA</key-id>
          <count>15</count>
        </ctrl-connection>
      </server>
    </twamp>
  </data>
```


A.3. Session-Sender


```
  <?xml version="1.0" encoding="utf-8"?>
  <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <session-sender>
        <admin-state>true</admin-state>
        <test-session>
          <name>Test1</name>
          <ctrl-connection-name>RouterA</ctrl-connection-name>
          <fill-mode>zero</fill-mode>
          <number-of-packets>900</number-of-packets>
          <periodic-interval>1</periodic-interval>
          <sent-packets>2</sent-packets>
          <rcv-packets>2</rcv-packets>
          <last-sent-seq>1</last-sent-seq>
          <last-rcv-seq>1</last-rcv-seq>
        </test-session>
        <test-session>
          <name>Test2</name>
          <ctrl-connection-name>RouterA</ctrl-connection-name>
          <fill-mode>random</fill-mode>
          <number-of-packets>900</number-of-packets>
          <lambda>1</lambda>
          <max-interval>2</max-interval>
          <sent-packets>21</sent-packets>
          <rcv-packets>21</rcv-packets>
          <last-sent-seq>20</last-sent-seq>
          <last-rcv-seq>20</last-rcv-seq>
        </test-session>
      </session-sender>
    </twamp>
  </data>
```

### A.4. Session-Reflector

[note: '\' line wrapping is for formatting only]


```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
    <session-reflector>
      <admin-state>true</admin-state>
      <test-session>
        <sender-ip>203.0.113.3</sender-ip>
        <sender-udp-port>54000</sender-udp-port>
        <reflector-ip>203.0.113.4</reflector-ip>
        <reflector-udp-port>55000</reflector-udp-port>
        <sid>1232</sid>
        <parent-connection-client-ip>203.0.113.1</parent-connection-\
client-ip>
        <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
        <parent-connection-server-ip>203.0.113.2</parent-connection-\
server-ip>
        <parent-connection-server-tcp-port>862</parent-connection-se\
rver-tcp-port>
        <test-packet-dscp>32</test-packet-dscp>
        <sent-packets>2</sent-packets>
        <rcv-packets>2</rcv-packets>
        <last-sent-seq>1</last-sent-seq>
        <last-rcv-seq>1</last-rcv-seq>
      </test-session>
      <test-session>
        <sender-ip>203.0.113.1</sender-ip>
        <sender-udp-port>54001</sender-udp-port>
        <reflector-ip>192.0.2.2</reflector-ip>
        <reflector-udp-port>55001</reflector-udp-port>
        <sid>178943</sid>
        <parent-connection-client-ip>203.0.113.1</parent-connection-\
client-ip>
        <parent-connection-client-tcp-port>16341</parent-connection-\
client-tcp-port>
        <parent-connection-server-ip>203.0.113.2</parent-connection-\
server-ip>
        <parent-connection-server-tcp-port>862</parent-connection-se\
rver-tcp-port>
        <test-packet-dscp>32</test-packet-dscp>
        <sent-packets>21</sent-packets>
        <rcv-packets>21</rcv-packets>
        <last-sent-seq>20</last-sent-seq>
        <last-rcv-seq>20</last-rcv-seq>
```

```
        </test-session>
      </session-reflector>
    </twamp>
  </data>
```


[note: '\' line wrapping is for formatting only]


```
  <?xml version="1.0" encoding="utf-8"?>
  <data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <session-reflector>
        <admin-state>true</admin-state>
        <test-session>
          <sender-ip>2001:DB8:10:1:1::1</sender-ip>
          <sender-udp-port>54000</sender-udp-port>
          <reflector-ip>2001:DB8:10:1:1::2</reflector-ip>
          <reflector-udp-port>55000</reflector-udp-port>
          <sid>1232</sid>
          <parent-connection-client-ip>2001:DB8:203:0:113::1</parent-c\
  onnection-client-ip>
          <parent-connection-client-tcp-port>16341</parent-connection-\
  client-tcp-port>
          <parent-connection-server-ip>2001:DB8:203:0:113::2</parent-c\
  onnection-server-ip>
          <parent-connection-server-tcp-port>862</parent-connection-se\
  rver-tcp-port>
          <test-packet-dscp>32</test-packet-dscp>
          <sent-packets>2</sent-packets>
          <rcv-packets>2</rcv-packets>
          <last-sent-seq>1</last-sent-seq>
          <last-rcv-seq>1</last-rcv-seq>
        </test-session>
        <test-session>
          <sender-ip>2001:DB8:203:0:113::1</sender-ip>
          <sender-udp-port>54001</sender-udp-port>
          <reflector-ip>2001:DB8:192:68::2</reflector-ip>
          <reflector-udp-port>55001</reflector-udp-port>
          <sid>178943</sid>
          <parent-connection-client-ip>2001:DB8:203:0:113::1</parent-c\
  onnection-client-ip>
          <parent-connection-client-tcp-port>16341</parent-connection-\
  client-tcp-port>
          <parent-connection-server-ip>2001:DB8:203:0:113::2</parent-c\
  onnection-server-ip>
          <parent-connection-server-tcp-port>862</parent-connection-se\
  rver-tcp-port>
          <test-packet-dscp>32</test-packet-dscp>
          <sent-packets>21</sent-packets>
```

```
          <rcv-packets>21</rcv-packets>
          <last-sent-seq>20</last-sent-seq>
          <last-rcv-seq>20</last-rcv-seq>
        </test-session>
      </session-reflector>
    </twamp>
  </data>
```


## Appendix B. TWAMP Operational Commands

TWAMP operational commands could be performed programmatically or manually, e.g. using a command-line interface (CLI).

With respect to programmability, YANG can be used to define NETCONF Remote Procedure Calls (RPC), therefore it would be, in principle, possible to define TWAMP RPC operations for actions such as starting or stopping control connections or test sessions or groups of sessions; retrieving results; clearing stored results, and so on.

However, TWAMP [RFC5357] does not attempt to describe such operational actions. Refer also to Section 2 and the unlabeled links in Figure 1. In actual deployments different TWAMP implementations may support different sets of operational commands, with different restrictions. Therefore, this document considers it the responsibility of the individual implementation to define its corresponding TWAMP operational commands data model.

## Authors' Addresses


Ruth Civil
Ciena Corporation
307 Legget Drive
Kanata, ON  K2K 3C8
Canada

Email: gcivil@ciena.com
URI:   www.ciena.com

Al Morton
AT&T Labs
200 Laurel Avenue South
Middletown,, NJ 07748
USA

Phone: +1 732 420 1571
Fax: +1 732 368 1192
Email: acmorton@att.com

Reshad Rahman
Cisco Systems
2000 Innovation Drive
Kanata, ON K2K 3E8
Canada

Email: rrahman@cisco.com

Mahesh Jethanandani
Xoriant Corporation
1248 Reamswood Drive
Sunnyvale, CA 94089
USA

Email: mjethanandani@gmail.com

Kostas Pentikousis (editor)
Travelping
Siemensdamm 50
Berlin 13629
Germany

Email: k.pentikousis@travelping.com