Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 18, 2016

M. Chen, Ed.
L. Zheng, Ed.
Huawei Technologies
G. Mirsky, Ed.
Ericsson
G. Fioccola, Ed.
Telecom Italia
T. Mizrahi, Ed.
Marvell

March 17, 2016

# IP Flow Performance Measurement Framework
# draft-chen-ippm-coloring-based-ipfpm-framework-06

## Abstract

This document specifies a measurement method, the IP flow performance measurement (IPFPM). With IPFPM, data packets are marked into different blocks of markers by changing one or more bits of packets. No additional delimiting packet is needed and the performance is measured in-service and in-band without the insertion of additional traffic.

## Requirements Language

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

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 18, 2016.

Table of Contents

1. Introduction . . . . . 2
2. Terminology . . . . . 4
3. Overview and Concept . . . . . 4
4. Consideration on Marking Bits . . . . . 6
5. Reference Model and Functional Components . . . . . 6
5.1. Reference Model . . . . . 6
5.2. Measurement Control Point . . . . . 7
5.3. Measurement Agent . . . . . 7
6. Period Number . . . . . 8
7. Re-ordering Tolerance . . . . . 8
8. Packet Loss Measurement . . . . . 9
9. Packet Delay Measurement . . . . . 10
10. Synchronization Aspects . . . . . 11
10.1. Synchronization for the Period Number . . . . . 12
10.2. Synchronization for Delay Measurement . . . . . 12
11. IANA Considerations . . . . . 13
12. Security Considerations . . . . . 13
13. Acknowledgements . . . . . 14
14. Contributing Authors . . . . . 14
15. References . . . . . 15
15.1. Normative References . . . . . 15
15.2. Informative References . . . . . 15
Authors' Addresses . . . . . 17

## 1. Introduction

Performance Measurement (PM) is an important tool for service providers, used for Service Level Agreement (SLA) verification, troubleshooting (e.g., fault localization or fault delimitation) and network visualization. Measurement methods could be roughly put into two categories - active measurement methods and passive measurement

methods. Active methods measure performance or reliability parameters by the examination of traffic (IP Packets) injected into the network, expressly for the purpose of measurement by the intended measurement points. In contrast, passive method measures some performance or reliability parameters associated with the existing traffic (packets) on the network. Both passive and active methods have their strengths and should be regarded as complementary. There are certain scenarios where active measurement alone is not enough or applicable and passive measurement is desirable[I-D.deng-ippm-passive-wireless-usecase].

With active measurement methods, the rate, numbers and interval between the injected packets may affect the accuracy of the results. Moreover, injected test packets are not always guaranteed to be in-band with the data traffic in the pure IP network due to Equal Cost Multi-Path (ECMP).

The Multiprotocol Label Switching (MPLS) PM protocol [RFC6374] for packet loss could be considered an example of a passive performance measurement method. By periodically inserting auxiliary Operations, Administration and Maintenance (OAM) packets, the traffic is delimited by OAM packets into consecutive blocks, and the receivers count the packets and calculate the packets lost in each block. However, solutions like [RFC6374] depend on the fixed positions of the delimiting OAM packets for packets counting, and thus are vulnerable to out-of-order arrival of packets. This could happen particularly with out-of-band OAM channels, but might also happen with in-band OAM because of the presence of multipath forwarding within the network. Out of order delivery of data and the delimiting OAM packets can give rise to inaccuracies in the performance measurement figures. The scale of these inaccuracies will depend on data speeds and the variation in delivery, but with out-of-band OAM, this could result in significant differences between real and reported performance.

This document specifies a different measurement method, the IP flow performance measurement (IPFPM). With IPFPM, data packets are marked into different blocks of markers by changing one or more bits of packets without altering normal processing in the network. No additional delimiting packet is needed and the performance can be measured in-service without the insertion of additional traffic. Furthermore, because marking-based IP performance measurement does not require extra OAM packets for traffic delimitation, it can be used in situations where there is packet re-ordering. IP Flow Information eXport (IPFIX) [RFC7011] is used for reporting the measurement data of IPFPM to a central calculation element for performance metrics calculation. Several new Information Elements of

IPFIX are defined for IPFPM. These are described in the companion document [I-D.chen-ippm-ipfpm-report].

## 2. Terminology

The acronyms used in this document will be listed here.

## 3. Overview and Concept

The concept of marking IP packets for performance measurement is described in [I-D.tempia-opsawg-p3m]. Marking of packets in a specific IP flow to different colors divides the flows into different consecutive blocks. Packets in a block have same marking and consecutive blocks will have different markings. This enables the measuring node to count and calculate packet loss and/or delay based on each block of markers without any additional auxiliary OAM packets. The following figure (Figure 1) is an example that illustrates the different markings in a single IP flow in alternate 0 and 1 blocks.

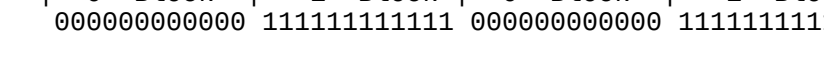



Figure 1: Packet Marking

For packet loss measurement, there are two ways to mark packets: fixed packet numbers or fixed time period for each block of markers. This document considers only fixed time period method. The sender and receiver nodes count the transmitted and received packets/octets based on each block of markers. By counting and comparing the transmitted and received packets/octets, the packet loss can be computed.

For packet delay measurement, there are three solutions. One is similar to the packet loss, it still marks the IP flows to different blocks of markers and uses the time of the marking change as the reference time for delay calculations. This solution requires that there must not be any out-of-order packets; otherwise, the result will not be accurate. Because it uses the first packet of each block of markers for delay measurement, if there is packet reordering, the first packet of each block at the sender will be probably different from the first packet of the block at the receiver. An alternate way is to periodically mark a single packet in the IP flow. Within a given time period, there is only one packet that can be marked. The sender records the timestamp when the marked packet is transmitted, and the receiver records the timestamp when receiving the marked packet. With the two timestamps, the packet delay can be computed.

An additional method consists of taking into account the average arrival time of the packets within a single block (i.e. the same block of markers used for packet loss measurement). The network device locally sums all the timestamps and divides by the total number of packets received, so 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 average delay between those nodes. This method is robust to out of order packets and also to packet loss (only an error is introduced dependent from the number of lost packets).

A centralized calculation element Measurement Control Point (MCP) is introduced in Section 5.2 of this document, to collect the packet counts and timestamps from the senders and receivers for metrics calculation. The IP Flow Information eXport (IPFIX) [RFC7011] protocol is used for collecting the performance measurement statistic information [I-D.chen-ippm-ipfpm-report]. For the statistic information collected, the MCP has to know exactly what packet pair counts (one from the sender and the other is from the receiver) are based on the same block of markers and a pair of timestamps (one from the sender and the other is from the receiver) are based on the same marked packet. In case of average delay calculation the MCP has to know in addition to the packet pair counters also the pair of average timestamps for the same block of markers. The "Period Number" based solution Section 6 is introduced to achieve this.

For a specific IP flow to be measured, there may be one or more upstream and downstream Measurement Agents (MAs)( Section 5.3). An IP flow can be identified by the Source IP (SIP) and Destination IP (DIP) addresses, and it may combine the SIP and DIP with any or all of the Protocol number, the Source port, the Destination port, and the Type of Service (TOS) to identify an IP flow. For each flow, there will be a flow identifier that is unique within a certain administrative domain. To simplify the process description, the flows discussed in this document are all unidirectional. A bidirectional flow can be seen as two unidirectional flows.

IFPFM supports the measurement of a Multipoint-to-Multipoint (MP2MP) model, which satisfies all the scenarios that include Point-to-Point (P2P), Point-to-Multipoint (P2MP), Multipoint-to-Point (MP2P), and MP2MP. The P2P scenario is obvious and can be used anywhere. P2MP and MP2P are very common in mobile backhaul networks. For example, a Cell Site Gateway (CSG) that uses multi-homing to two Radio Network Controller (RNC) Site Gateways (RSGs) is a typical network design. When there is a failure, there is a requirement to monitor the flows between the CSG and the two RSGs hence to determine whether the fault is in the transport network or in the wireless network (typically called "fault delimitation"). This is especially useful in the

situation where the transport network belongs to one service provider and the wireless network belongs to other service providers.

## 4. Consideration on Marking Bits

The marking bits selection is encapsulation-related; different bits for marking should be allocated by different encapsulations. This document does not define any marking bits. The marking bits selection for specific encapsulations will be defined in the relevant documents. In general, at least one marking bit is required to support loss and delay measurement. Specifically, if the second delay measurement solution is used (see Section 3), then at least two marking bits are needed; one bit for packet loss measurement, the other for packet delay measurement.

In theory, so long as there are unused bits that could be allocated for marking purpose, the marking-based measurement mechanism can be applied to any encapsulation. It is relatively easier for new encapsulations to allocate marking bits. An example of such a case is Bit Indexed Explicit Replication (BIER). Two marking bits for passive performance measurement has been allocated in the BIER encapsulation [I-D.ietf-bier-mpls-encapsulation] (Section 3.). However, for sophisticated encapsulations, it is harder or even impossible to allocate bits for marking purpose. The IPv4 encapsulation is one of the examples. The IPv6 encapsulation is in a similar situation, but for IPv6, an alternative solution is to leverage the IPv6 extension header for marking.

Since marking will directly change some bits (of the header) of the real traffic packets, the marking operations MUST NOT affect the forwarding and processing of packets. Specifically, the marking bits MUST NOT be used for ECMP hashing. In addition, to increase the accuracy of measurement, hardware-based implementation is desired. Thus, the location of the marking bits SHOULD be easy for hardware implementation. For example, the marking bits would be best located at fixed positions in a packet header.

## 5. Reference Model and Functional Components

### 5.1. Reference Model

The outline of the measurement system of large-scale measurement platforms (LMAP) is introduced in [I-D.ietf-lmap-framework]. It describes the main functional components of the LMAP measurement system, and the interactions between the components. The Measurement Agent (MA) of IPFPM could be considered equivalent to the MA of LMAP. The Measurement Control Point (MCP) of IPFPM could be considered as the combined function of Controller and Collector. The IP Flow

Information eXport (IPFIX) [RFC7011] protocol is used for collecting the performance measurement data on the MAs and reporting to the MCP. The details are specified in the companion document [I-D.chen-ippm-ipfpm-report]. The control between MCP and MAs are left for future study. Figure 2 presents the reference model of IPFPM.

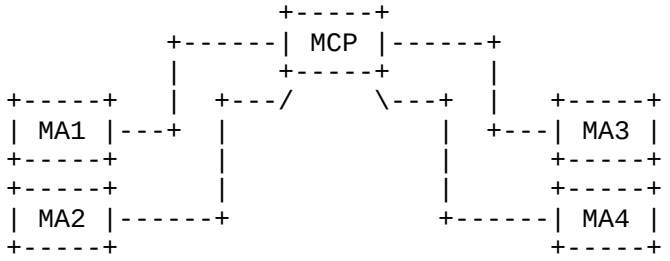



Figure 2: IPFPM Reference Model

## 5.2. Measurement Control Point

The Measurement Control Point (MCP) is responsible for collecting the measurement data from the Measurement Agents (MAs) and calculating the performance metrics according to the collected measurement data. For packet loss, based on each block of markers, the difference between the total counts received from all upstream MAs and the total counts received from all downstream MAs are the lost packet numbers. The MCP must make sure that the counts from the upstream MAs and downstream MAs are related to the same marking/packets block. For packet delay (e.g., one way delay), the difference between the timestamps from the downstream MA and upstream MA is the packet delay. Similarly to packet loss, the MCP must make sure the two timestamps are based on the same marked packet. This document introduces a Period Number (PN) based synchronization mechanism which is discussed in details in Section 6.

## 5.3. Measurement Agent

The Measurement Agent (MA) executes the measurement actions (e.g., marks the packets, counts the packets, records the timestamps, etc.), and reports the data to the Measurement Control Point (MCP). Each MA maintains two timers, one (C-timer, used at upstream MA) is for marking change, the other (R-timer, used at downstream MA) is for reading the packet counts and timestamps. The two timers have the same time interval but are started at different times. An MA can be either an upstream or a downstream MA: the role is specific to an IP flow to be measured. For a specific IP flow, the upstream MA will change the marking and read the packet counts and timestamps when the C-timer expires, the downstream MA just reads the packet counts and

timestamps when the R-timer expires. The MA may delay the reading for a certain time period when the R-timer expires, in order to be tolerant to a certain degree of packet re-ordering. Section 7 describes this in details.

For each Measurement Task (corresponding to an IP flow) [I-D.ietf-lmap-framework], an MA maintains a pair of packet counters and a timestamp counter for each block of markers. As for the pair of packet counters, one is for counting packets and the other is for counting octets.

## 6. Period Number

When data is collected on the upstream MA and downstream MA, e.g., packet counts or timestamps, and periodically reported to the MCP, a certain synchronization mechanism is required to ensure that the collected data is correlated. Synchronization aspects are further discussed in Section 10. This document introduces the Period Number (PN) to help the MCP to determine whether any two or more packet counts (from distributed MAs) are related to the same block of markers, or any two timestamps are related to the same marked packet.

Period Numbers assure the data correlation by literally splitting the packets into different measurement periods. The PN is generated each time an MA reads the packet counts or timestamps, and is associated with each packet count and timestamp reported to the MCP. For example, when the MCP sees two PNs associated with two packet counts from an upstream and a downstream MA, it assumes that these two packet counts correspond to the same measurement period by the same PN, i.e., that these two packet counts are related to the same block of markers. The assumption is that the upstream and downstream MAs are time synchronized. This requires the upstream and downstream MAs to have a certain time synchronization capability (e.g., the Network Time Protocol (NTP) [RFC5905], or the IEEE 1588 Precision Time Protocol (PTP) [IEEE1588]), as further discussed in Section 10. The PN is calculated as the modulo of the local time (when the counts or timestamps are read) and the interval of the marking time period.

## 7. Re-ordering Tolerance

In order to allow for a certain degree of packet re-ordering, the R-timer on downstream MAs should be started delta-t (Dt) later than the C-timer is started. Dt is a defined period of time and should satisfy the following conditions:

(Time-L - Time-MRO ) < Dt < (Time-L + Time-MRO )

Where

Time-L: the link delay time between the sender and receiver;

Time-MRO: the maximum re-ordering time difference; if a packet is expected to arrive at t1 but actually arrives at t2, then the Time-MRO = | t2 - t1|.

Thus, the R-timer should be started at "t + Dt" (where t is the time at which C-timer is started).

For simplicity, the C-timer should be started at the beginning of each time period. This document recommends the implementation to support at least these time periods (1s, 10s, 1min, 10min and 1h). Thus, if the time period is 10s, then the C-timer should be started at the time of any multiples of 10 in seconds (e.g., 0s, 10s, 20s, etc.), and the R-timer should be started, for example, at 0s+Dt, 10s+Dt, 20s+Dt, etc. With this method, each MA can independently start its C-timer and R-timer given that the clocks have been synchronized.

## 8. Packet Loss Measurement

To simplify the process description, the flows discussed in this document are all unidirectional. A bidirectional flow can be seen as two unidirectional flows. For a specific flow, there will be an upstream MA and a downstream MA, and for each of these MAs there will be corresponding packet counts/timestamp.

For packet loss measurement, this document defines the following counters and quantities:

U-CountP[n][m]: U-CountP is a two-dimensional array that stores the number of packets transmitted by each upstream MA in each marking time period. Specifically, parameter "n" is the "period number" of measured blocks of markers while parameter "m" refers to the m-th MA of the upstream MAs.

D-CountP[n][m]: D-CountP is a two-dimensional array that stores the number of packets received by each downstream MA in each marking time period. Specifically, parameter "n" is the "period number" of measured blocks of markers while parameter "m" refers to the m-th MA of the downstream MAs.

U-CountO[n][m]: U-CountO is a two-dimensional array that stores the number of octets transmitted by each upstream MA in each marking time period. Specifically, parameter "n" is the "period number" of measured blocks of markers while parameter "m" refers to the m-th MA of the upstream MAs.

D-CountO[n][m]: D-CountO is a two-dimensional array that stores the number of octets received by each downstream MA in each marking time period. Specifically, parameter "n" is the "period number" of measured blocks of markers while parameter "m" refers to the m-th MA of the downstream MAs.

LossP: the number of packets transmitted by the upstream MAs but not received at the downstream MAs.

LossO: the total octets transmitted by the upstream MAs but not received at the downstream MAs.

The total packet loss of a flow can be computed as follows:

LossP = U-CountP[1][1] + U-CountP[1][2] + .... + U-CountP[n][m] - D-CountP[1][1] - D-CountP[1][2] - .... - D-CountP[n][m’].

LossO = U-CountO[1][1] + U-CountO[1][2] + .... + U-CountO[n][m] - D-CountO[1][1] - D-CountO[1][2] - .... - D-CountO[n][m’].

Where the m and m’ are the number of upstream MAs and downstream MAs of the measured flow, respectively.

## 9. Packet Delay Measurement

For packet delay measurement, there will be only one upstream MA and may be one or more (P2MP) downstream MAs. Although the marking-based IPFPM supports P2MP model, this document only discusses P2P model. The P2MP model is left for future study. This document defines the following timestamps and quantities:

U-Time[n]: U-Time is a one-dimension array that stores the time when marked packets are sent; in case the "average delay" method is being used, U-Time stores the average of the time when the packets of the same block are sent; parameter "n" is the "period number" of marked packets.

D-Time[n]: D-Time is a one-dimension array that stores the time when marked packets are received; in case the "average delay" method is being used, D-Time stores the average of the time when the packets of the same block are received; parameter "n" is the "period number" of marked packets. This is only for P2P model.

D-Time[n][m]: D-Time a two-dimension array that stores the time when the marked packet is received by downstream MAs at each marking time period; in case the "average delay" method is being used, D-Time stores the average of the times when the packets of the same block are received by downstream MAs at each marking time period. Here,

parameter "n" is the "period number" of marked packets while parameter "m" refers to the m-th MA of the downstream MAs. This is for P2MP model which is left for future study.

One-way Delay[n]: The one-way delay metric for packet networks is described in [RFC2679]. The "n" identifies the "period number" of the marked packet.

One-way Delay[1] = D-Time[1] - U-Time[1].

One-way Delay[2] = D-Time[2] - U-Time[2].

...

One-way Delay[n] = D-Time[n] - U-Time[n].

In the case of two-way delay, the delay is the sum of the two one-way delays of the two flows that have the same MAs but have opposite directions.

Two-way Delay[1] = (D-Time[1] - U-Time[1]) + (D-Time’[1] - U-Time’[1]).

Two-way Delay[2] = (D-Time[2] - U-Time[2]) + (D-Time’[2] - U-Time’[2]).

...

Two-way Delay[n] = (D-Time[n] - U-Time[n]) + (D-Time’[n] - U-Time’[n]).

Where the D-Time and U-Time are for one forward flow, the D-Time’ and U-Time’ are for reverse flow.

## 10. Synchronization Aspects

As noted in the previous sections, there are two mechanisms in IPFPM that require MAs to have synchronized clocks: (i) the period number (Section 6), and (ii) delay measurement.

This section elaborates on the level of synchronization that is required for each of the two mechanisms. Interestingly, IPFPM can be implemented even with very coarse-grained synchronization.

### 10.1. Synchronization for the Period Number

Period numbers are used to uniquely identify blocks, allowing the MCP to match the measurements of each block from multipe MAs.

The period number of each measurement is computed by the modulo of the local time. Therefore, if the length of the measurement period is L time units, then all MAs must be synchronized to the same clock reference with an accuracy of +/- L/2 time units. This level of accuracy gurantees that all MAs 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.

The synchronization requirement for maintaining the period number can be satisfied even with a relatively inaccurate synchronization method.

### 10.2. Synchronization for Delay Measurement

As discussed in Section 9, the delay between two MAs is computed by D-Time[1] - U-Time[1], requiring the two MAs to be synchronized.

Notably, two-way delay measurement does not require the two MAs to be time synchronized. Therefore, a system that uses only two-way delay measurement does not require synchronization between MAs.

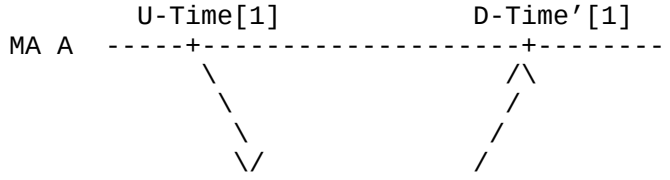



Figure 3: Two-way Delay Measurement

As shown in Section 9, the two way delay between two MAs is given by (see Figure 3):

(D-Time[1] - U-Time[1]) + (D-Time'[1] - U-Time'[1])

Therefore, the two-way delay is equal to:

(D-Time'[1] - U-Time[1]) - (U-Time'[1] - D-Time'[1])

The latter implies that the two-way delay is comprised of two time differences, (D-Time'[1] - U-Time[1]), and (U-Time'[1] - D-Time'[1]). Thus, the value of the clocks of MA A and MA B does not affect the computation, and synchronization is not required.

## 11. IANA Considerations

This document makes no request to IANA.

## 12. Security Considerations

This document specifies a passive mechanism for measuring packet loss and delay within a Service Provider's network where the IP packets are marked using unused bits in IP head field, thus avoiding the need to insert additional OAM packets during the measurement. Obviously, such a mechanism does not directly affect other applications running on the Internet but may potentially affect the measurement itself.

First, 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 is just in the context of one Service Provider's network, and thus the routers (or other network devices) are locally administered and this type of attack can be avoided.

Second, 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 color bits are the only information that is exchanged between the MAs. Therefore, passive eavesdropping to data plane traffic does not allow attackers to gain information about the network performance. We note that the information exported from the MAs to the MCP can be subject to eavesdropping, and thus it should be encrypted.

Finally, 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].

## 13. Acknowledgements

The authors would like to thank Adrian Farrel for his review, suggestion and comments to this document.

## 14. Contributing Authors

Hongming Liu
Huawei Technologies

Email: liuhongming@huawei.com

Yuanbin Yin
Huawei Technologies

Email: yinyuanbin@huawei.com

Rajiv Papneja
Huawei Technologies

Email: Rajiv.Papneja@huawei.com

Shailesh Abhyankar
Vodafone
Vodafone House, Ganpat Rao kadam Marg Lower Parel
Mumbai 40003
India

Email: shailesh.abhyankar@vodafone.com

Guangqing Deng
CNNIC
4 South 4th Street, Zhongguancun, Haidian District
Beijing
China

Email: dengguangqing@cnnic.cn

Yongliang Huang
China Unicom

Email: huangyl@dipmt.com

# 15. References

## 15.1. Normative References

[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>.

## 15.2. Informative References

[I-D.chen-ippm-ipfpm-report]
Chen, M., Zheng, L., Liu, H., Yin, Y., Papneja, R., Abhyankar, S., Deng, G., and Y. Huang, "IP Flow Performance Measurement Report", draft-chen-ippm-ipfpm-report-00 (work in progress), July 2014.

[I-D.deng-ippm-passive-wireless-usecase]
Lingli, D., Zheng, L., and G. Mirsky, "Use-cases for Passive Measurement in Wireless Networks", draft-deng-ippm-passive-wireless-usecase-01 (work in progress), January 2015.

[I-D.ietf-bier-mpls-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., and S. Aldrin, "Encapsulation for Bit Index Explicit Replication in MPLS Networks", draft-ietf-bier-mpls-encapsulation-03 (work in progress), February 2016.

[I-D.ietf-lmap-framework]
Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A framework for Large-Scale Measurement of Broadband Performance (LMAP)", draft-ietf-lmap-framework-14 (work in progress), April 2015.

[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.

[IEEE1588]
IEEE, "1588-2008 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", March 2008.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[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, <http://www.rfc-editor.org/info/rfc2474>.

[RFC2679] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Delay Metric for IPPM", RFC 2679, DOI 10.17487/RFC2679, September 1999, <http://www.rfc-editor.org/info/rfc2679>.

[RFC3260] Grossman, D., "New Terminology and Clarifications for Diffserv", RFC 3260, DOI 10.17487/RFC3260, April 2002, <http://www.rfc-editor.org/info/rfc3260>.

[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>.

[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>.

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

[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, <http://www.rfc-editor.org/info/rfc7011>.

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

Authors' Addresses

Mach(Guoyi) Chen (editor)
Huawei Technologies

Email: mach.chen@huawei.com

Lianshu Zheng (editor)
Huawei Technologies

Email: vero.zheng@huawei.com

Greg Mirsky (editor)
Ericsson
USA

Email: gregory.mirsky@ericsson.com

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

Email: giuseppe.fioccola@telecomitalia.it

Tal Mizrahi (editor)
Marvell
6 Hamada st.
Yokneam
Israel

Email: talmi@marvell.com

IPPM WG
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

R. Civil
Ciena Corporation
A. Morton
AT&T Labs
L. Zheng
Huawei Technologies
R. Rahman
Cisco Systems
M. Jethanandani
Ciena Corporation
K. Pentikousis, Ed.
EICT
October 19, 2015

# Two-Way Active Measurement Protocol (TWAMP) Data Model
# draft-cmzrjp-ippm-twamp-yang-02

## Abstract

This document specifies a data model for client and server implementations of the Two-Way Active Measurement Protocol (TWAMP). We define the TWAMP data model through Unified Modeling Language (UML) class diagrams and formally specify it using YANG.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 21, 2016.

## Copyright Notice

Table of Contents

1. Introduction . . . . . 3
1.1. Motivation . . . . . 3
1.2. Terminology . . . . . 3
1.3. Document Organization . . . . . 3
2. Scope, Model, and Applicability . . . . . 4
3. Data Model Overview . . . . . 5
3.1. Control-Client . . . . . 5
3.2. Server . . . . . 6
3.3. Session-Sender . . . . . 7
3.4. Session-Reflector . . . . . 7
4. Data Model Parameters . . . . . 7
4.1. Control-Client . . . . . 7
4.2. Server . . . . . 14
4.3. Session-Sender . . . . . 18
4.4. Session-Reflector . . . . . 21
5. Data Model . . . . . 25
5.1. YANG Tree Diagram . . . . . 25
5.2. YANG Module . . . . . 27
6. Data Model Examples . . . . . 43
6.1. Control-Client . . . . . 43
6.2. Server . . . . . 44
6.3. Session-Sender . . . . . 45
6.4. Session-Reflector . . . . . 46
7. Security Considerations . . . . . 47
8. IANA Considerations . . . . . 48
9. Acknowledgements . . . . . 48
10. References . . . . . 48
10.1. Normative References . . . . . 48
10.2. Informative References . . . . . 49
Appendix A. Detailed Data Model Examples . . . . . 50
A.1. Control-Client . . . . . 51
A.2. Server . . . . . 52
A.3. Session-Sender . . . . . 53
A.4. Session-Reflector . . . . . 54
Appendix B. TWAMP Operational Commands . . . . . 55
Authors' Addresses . . . . . 55

# 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, configuration depends on the various proprietary mechanisms developed by the corresponding TWAMP vendor. This document addresses this gap by formally specifying the TWAMP data model using YANG.

## 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 as discussed in [I-D.unify-nfvrg-challenges][I-D.unify-nfvrg-devops], proprietary mechanisms for managing TWAMP measurements pose severe limitations with respect to programmability.

Two major trends call for revisiting the standardization on TWAMP management aspects. First, we expect that in the coming years large-scale and multi-vendor TWAMP deployments will become the norm. From an operations perspective, dealing with several vendor-specific TWAMP configuration mechanisms is simply unsustainable in this context. Second, the increasingly software-defined and virtualized nature of network infrastructures, based on dynamic service chains [NSC] and programmable control and management planes [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 data modeling language [RFC6020].

## 1.2. Terminology

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

## 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 [RFC5357]. The figure is annotated with pointers to the 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. As per [RFC5357], unlabeled links in Figure 1 are unspecified and may be proprietary protocols.

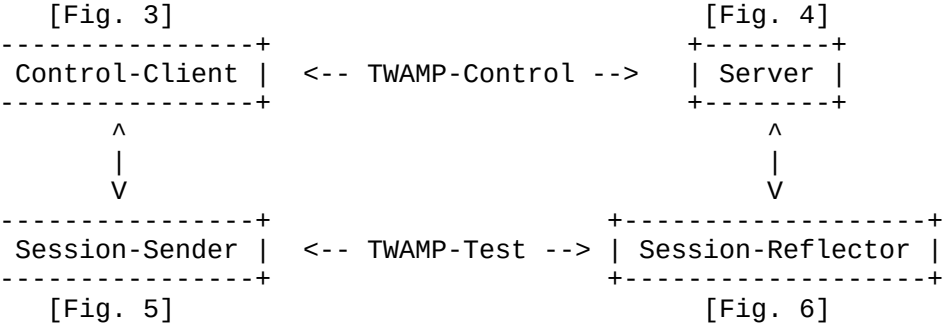



Figure 1: Annotated TWAMP logical model

As per [RFC5357], a TWAMP implementation may follow a simplified logical model, in which the same node acts both as the Control-Client and Session-Sender, while another node acts at the same time as the 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 [I-D.ietf-netconf-restconf]. Note, however, that the specific protocol used to communicate the TWAMP configuration parameters specified herein is outside the scope of this document. Appendix B considers TWAMP operational commands, which are also outside the scope of this document.

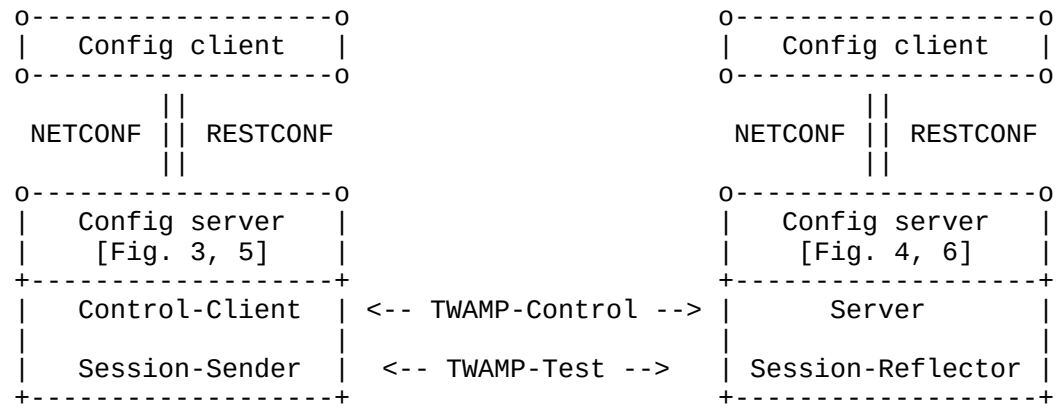



Figure 2: Simplified TWAMP model and protocols

## 3. Data Model Overview

A TWAMP data model includes four categories of configuration items. 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), are typical instances of global configuration items. A second category includes attributes that can be configured on a per control connection basis, such as the Server IP address. A third category includes attributes related to per test session attributes, for instance setting different values in the Differentiated Services Code Point (DSCP) field. Finally, the data model could include attributes that relate to the operational state of the TWAMP implementation.

As we describe the TWAMP data model 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:

- 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.

- The IP address of the interface the Control-Client will use for connections
- The IP address of the remote Server
- Authentication and Encryption attributes such as KeyID, Token and the Client Initialization Vector (Client-IV) [RFC4656].

Each TWAMP control connection, in turn, is associated with zero or more test sessions. For each test session we note the following configuration items:

- The test session name that uniquely identifies a particular test session at the Control-Client and Session-Sender. Similarly to the control connections above, this unique test session name is needed because at the time of creation of a test session, for example, the source UDP port number is not known to uniquely identify the test session.
- The IP address and UDP port number of the Session-Sender of the path under test by TWAMP
- The IP address and UDP port number of the Session-Reflector of said path
- Information pertaining to the test packet stream, such as the test starting time 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 TWAMP Server is associated with zero or more 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

There is one TWAMP Session-Sender instance for each test session that is initiated from the sending device. Primary configuration fields include:

- The test session name that MUST be identical with the corresponding test session name on the TWAMP Control-Client (Section 3.1)
- The control connection name, which along with the test session name uniquely identify the TWAMP Session-Sender instance
- Information pertaining to the test packet stream, such as, for example, the number of test packets and the packet distribution to be employed.

### 3.4. Session-Reflector

Each TWAMP Session-Reflector is associated with zero or more test sessions. For each test session, the REFWAIT parameter (Section 4.2 of [RFC5357] 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 and describes all associated parameters.

### 4.1. Control-Client

The twamp-client container (see Figure 3) holds items that are related to the configuration of the TWAMP Control-Client logical entity. These are divided up into items that are associated with the configuration of the Control-Client as a whole (e.g. client-admin-state) and items that are associated with individual control connections initiated by the Control-Client entity (twamp-client-ctrl-connection).

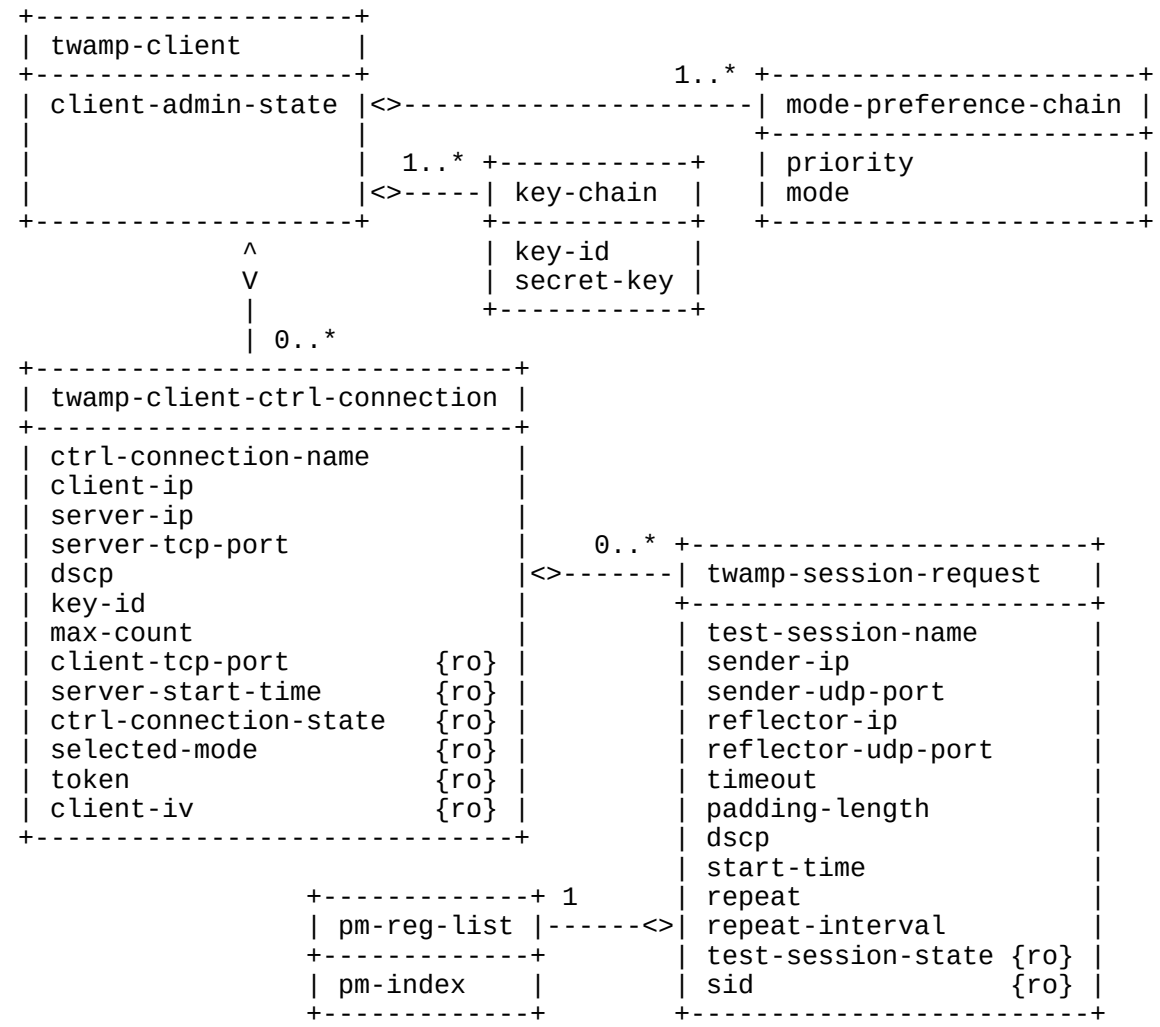



Figure 3: TWAMP Control-Client UML class diagram

The twamp-client container includes an administrative parameter (client-admin-state) that controls whether the device is allowed to initiate TWAMP control sessions.

The twamp-client container holds a list (mode-preference-chain) which specifies the preferred Mode values according to their preferred order of use, including the authentication and encryption Modes. Specifically, mode-preference-chain lists each priority (expressed as a 16-bit unsigned integer, where zero is the highest priority and subsequent values monotonically increasing) with their corresponding

mode (expressed as a 32-bit Hexadecimal value). 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 bit position combinations when necessary, such as when referring to the extended TWAMP features in [RFC5618], [RFC5938], and [RFC6038]. If the Control-Client cannot determine an acceptable Mode, 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 twamp-client container holds a list named key-chain which relates KeyIDs with the respective secret keys. Both the Server and the Control-Client use the same mappings from KeyIDs to shared secrets (key-id and secret-key in Figure 3, respectively). The Server, being prepared to conduct sessions with more than one Control-Client, uses KeyIDs 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, an octet string of arbitrary length whose interpretation as a text string is unspecified. The key-id and secret-key encoding should follow Section 9.4 of [RFC6020]. The derived key length (dkLen in [RFC2898]) MUST be 128-bits for the AES Session-key used for encryption and a 256-bit HMAC-SHA1 Session-key used for authentication (see Section 6.10 of [RFC4656]).

Each twamp-client container also holds a list of twamp-client-ctrl-connection, where each item in the list describes a TWAMP control connection that will be initiated by this Control-Client. There SHALL be one instance of twamp-client-ctrl-connection per TWAMP-Control (TCP) connection that is to be initiated from this device.

The configuration items for twamp-client-ctrl-connection are:

ctrl-connection-name
: A unique name used as a key to identify this individual TWAMP control connection on the Control-Client device.

client-ip
: 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.

server-ip
: The IP address belonging to the remote Server device, which the TWAMP-Control connection will be initiated to. This item is mandatory.

server-tcp-port
: 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 port number (862) as per [RFC5357]. 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. The default value is 862.

dscp
: The DSCP value to be placed in the TCP header of TWAMP-Control packets generated by this Control-Client. The default value is 0.

key-id
: The key-id value that is selected for this TWAMP-Control connection.

max-count
: 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. Therefore, TWAMP-compliant systems SHOULD have a configuration control to limit the maximum Count value. The default max-count value SHOULD be 32768.

The following twamp-client-ctrl-connection parameters are read-only:

client-tcp-port
: The source TCP port number used in the TWAMP-Control packets belonging to this control connection.

server-start-time
: The Start-Time advertized by the Server in the Server-Start message ([RFC4656], Section 3.1). This is a timestamp representing the time when the current instantiation of the Server started operating.

ctrl-connection-state
: The TWAMP-Control connection state can be either active or idle.

selected-mode
: 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 ([RFC4656], Section 3.1).

token 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. AES Session-key and HMAC Session-key are generated randomly by the Control-Client. AES Session-key and HMAC Session-key MUST be generated with sufficient entropy not to reduce the security of the underlying cipher [RFC4086]. The token itself is encrypted using the AES (Advanced Encryption Standard) in Cipher Block Chaining (CBC). Encryption MUST be performed using an Initialization Vector (IV) of zero and a key derived from the shared secret associated with KeyID. Challenge is the same as transmitted by the Server (Section 4.2) in the clear; see also the last paragraph of Section 6 in [RFC4656].

client-iv
The Control-Client Initialization Vector (Client-IV) is generated randomly by the Control-Client. 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.

Each twamp-client-ctrl-connection holds a list of twamp-session-request. twamp-session-request holds information associated with the Control-Client for this test session. This includes information that is associated with the Request-TW-Session/Accept-Session message exchange (see Section 3.5 of [RFC5357]). The Control-Client is also responsible for scheduling and results collection for TWAMP-Test sessions, so twamp-session-request will also hold information related these actions (e.g. pm-index, repeat-interval).

There SHALL be one instance of twamp-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 configuration items for twamp-session-request are:

test-session-name
A unique name for this test session to be used for identification of this TWAMP-Test session on the Control-Client.

sender-ip
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.

sender-udp-port
The UDP port number that is to be used by the Session-Sender for this TWAMP-Test session. A value of zero indicates that the Control-Client SHALL auto-allocate a UDP port number for this TWAMP-Test session. The configured (or auto-allocated) value is advertized in the Sender Port field of the Request-TW-session message (see also Section 3.5 of [RFC5357]). 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.

reflector-ip
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. This item is mandatory.

reflector-udp-port
This parameter defines the UDP port number that will be used by the Session-Reflector for this TWAMP-Test session. This value will be placed in the Receiver Port field of the Request-TW-Session message. If this value is not set, the device SHALL use the same port number as defined in the server-tcp-port parameter of this twamp-session-request's parent twamp-client-ctrl-connection.

timeout 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 ([RFC5357], Section 3.8). This value will be placed in the Timeout field of the Request-TW-Session message. The default value is 2 seconds.

padding-length
The number of bytes of padding that will 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 ([RFC4656], Section 3.5).

dscp The DSCP value to be placed in the UDP 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 ([RFC5357]).

start-time
Time when the session is to be started (but not before the Start-Sessions command is issued). This value is placed in the Start Time field of the Request-TW-Session message. The default value of 0 indicates that the session will be started as soon as the Start-Sessions message is received.

repeat and repeat-interval
These two values together are used to determine if the TWAMP-Test session is to be run repeatedly. Once a test session has completed, the repeat parameter is checked. If the value indicates that this test session is to run again, then the parent TWAMP-Control connection for this test session is restarted - and negotiates a new instance of this TWAMP-Test session. This may occur immediately after the test session completes (if the repeat-interval is set to 0). Otherwise, the Control-Client will wait for the number of minutes specified in the repeat-interval parameter before negotiating the new instance of this TWAMP-Test session. The default value of repeat is 0, indicating that once the session has completed, it will not be renegotiated and restarted.

pm-reg-list
A list of one or more Performance Metric Registry Index values (see [I-D.ietf-ippm-metric-registry], which communicate packet stream characteristics and 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.

pm-index
One or more Numerical index values of a Registered Metric in the Performance Metric Registry [I-D.ietf-ippm-metric-registry] comprise the pm-reg-list. Output statistics are specified in the corresponding Registry entry.

The following twamp-session-request parameters are read-only:

test-session-state
The TWAMP-Test session state can be either accepted or indicate the respective error code.

sid 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; see Section 4.3 of [RFC6038].

## 4.2. Server

The twamp-server container (see Figure 4) holds items that are related to the configuration of the TWAMP Server logical entity (recall Figure 1).

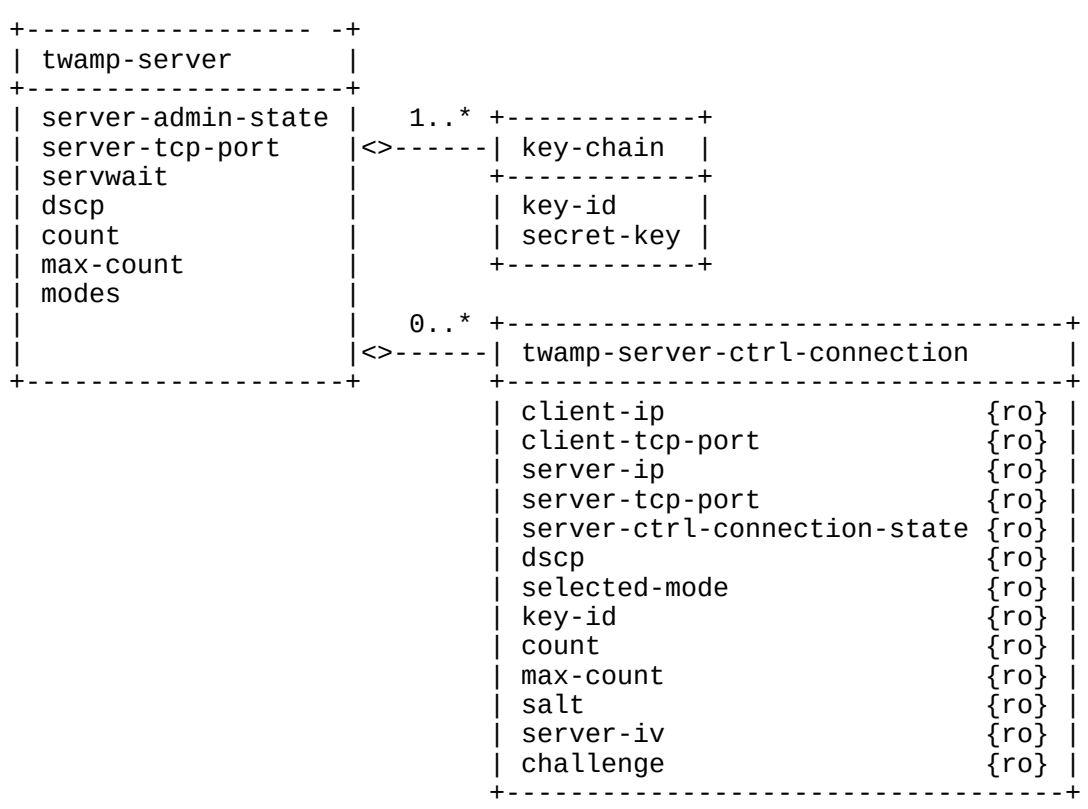



Figure 4: TWAMP Server UML class diagram

A device operating in the Server role cannot configure attributes on a per TWAMP-Control connection basis, as it has no foreknowledge of what incoming TWAMP-Control connections it will receive. As such, any parameter that the Server might want to apply to an incoming control connection must be configured at the overall Server level, and will then be applied to all incoming TWAMP-Control connections.

Each twamp-server container holds a list named key-chain which relates KeyIDs with the respective secret keys. As mentioned in Section 4.1, both the Server and the Control-Client use the same mappings from KeyIDs to shared secrets. The Server, being prepared to conduct sessions with more than one Control-Client, uses KeyIDs to choose the appropriate secret-key; a Control-Client would typically have different secret keys for different Servers. key-id tells the Server which shared-secret the Control-Client wishes to use for authentication or encryption.

Each incoming control connection that is active on the Server will be represented by an instance of a twamp-server-ctrl-connection object. All items in the twamp-server-ctrl-connection object are read-only, as we explain later in this section.

The twamp-server container items are as follows:

server-admin-state
: This administrative parameter controls whether the device is allowed to operate as a TWAMP Server. As defined in [RFC5357] the roles of Server and Session-Reflector can be played by the same host; recall Figure 2. For a host operating in this manner, this parameter controls whether the device is allowed to respond to TWAMP control sessions.

server-tcp-port
: 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 [RFC5357], 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. The default value is 862.

servwait
: TWAMP-Control (TCP) session timeout, in seconds (([RFC5357], Section 3.1)).

dscp
: The DSCP value to be placed in the IP header of TWAMP-Control (TCP) packets generated by the Server. Section 3.1 of [RFC5357] specifies that the server SHOULD use the DSCP value from the Control-Client's 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 [RFC5357] 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-Client's TCP SYN, as per Section 3.1 of [RFC5357].

count Parameter used in deriving a key from a shared secret as described in Section 3.1 of [RFC4656], and are communicated to the Control-Client as part of the Server Greeting message. count MUST be a power of 2. count MUST be at least 1024. count SHOULD be increased as more computing power becomes common.

max-count
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. Therefore, TWAMP-compliant systems SHOULD have a configuration control to limit the maximum count value. The default max-count value SHOULD be 32768.

modes
The bit mask of TWAMP Modes this Server instance is willing to support; see IANA TWAMP Modes Registry. Each bit position set represents a mode; see TWAMP-Modes at http://www.iana.org/assignments/twamp-parameters/twamp-parameters.xhtml. Note: Modes requiring Authentication or Encryption MUST include the related attributes.

There SHALL be one instance of twamp-server-ctrl-connection per incoming TWAMP-Control (TCP) connection that is received and active on the Server device. All items in the twamp-server-ctrl-connection are read-only. Each instance of twamp-server-ctrl-connection uses the following 4-tuple as its unique key: client-ip, client-tcp-port, server-ip, server-tcp-port.

The twamp-server-ctrl-connection container items are all read-only:

client-ip
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.

client-tcp-port
The source TCP port number used in the TWAMP-Control (TCP) packets belonging to this control connection.

server-ip

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.

server-tcp-port
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 twamp-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.

server-ctrl-connection-state
The Server TWAMP-Control connection state can be active or SERVWAIT.

dscp
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 dscp parameter under the twamp-server container. However, in the event that the user re-configures twamp-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.

selected-mode
The Mode that was chosen for this TWAMP-Control connection as set in the Mode field of the Set-Up-Response message.

key-id
The KeyID value that is in use by this TWAMP-Control connection. The Control-Client selects the key-id for the control connection.

count
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 twamp-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.

max-count

The max-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 twamp-server:max-count after this control connection is already in progress, this read-only value will show the actual max-count that is in use for this control connection.

salt A parameter used in deriving a key from a shared secret as described in Section 3.1 of [RFC4656]. Salt MUST be generated pseudo-randomly (independently of anything else in the RFC) and is communicated to the Control-Client as part of the Server Greeting message.

server-iv
The Server Initialization Vector (IV) is generated randomly by the Server.

challenge
A random sequence of octets generated by the Server. As described in Section 4.1 challenge is used by the Control-Client to prove possession of a shared secret.

## 4.3. Session-Sender

The twamp-session-sender container, illustrated in Figure 5, holds items that are related to the configuration of the TWAMP Session-Sender logical entity.

The twamp-session-sender container includes an administrative parameter (session-sender-admin-state) that controls whether the device is allowed to initiate TWAMP test sessions.

There is one instance of twamp-sender-test-session for each TWAMP-Test session for which packets are being sent.


Figure 5: TWAMP Session-Sender UML class diagram

The twamp-sender-test-session container items are:

test-session-name
: A unique name for this TWAMP-Test session to be used for identifying this test session by the Session-Sender logical entity.

ctrl-connection-name
: The name of the parent TWAMP-Control connection that is responsible for negotiating this TWAMP-Test session.

fill-mode
: 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 [RFC5357].

number-of-packets
: The overall number of TWAMP-Test (UDP) packets to be transmitted by the Session-Sender for this test session.

packet-distribution
: Defines whether TWAMP-Test (UDP) packets are to be transmitted with a fixed interval between them, or whether a Poisson distribution is to be used.

periodic-interval and periodic-interval-units
: If packet-distribution is set to periodic, these two values are used together to determine the period to wait between the first bits of TWAMP-Test (UDP) packet transmissions for this test session. periodic-interval-units is one of seconds, milliseconds, microseconds, nanoseconds; see [RFC3432].

lambda and lambda-units
: If packet-distribution is Poisson, the lambda parameter determines the corresponding average rate of packet transmission. lambda-units defines the units of lambda in reciprocal seconds; see [RFC3432].

max-interval
: If packet-distribution is Poisson, then this parameter keeps a stream active by setting a maximum time between packet transmissions.

truncation-point-units
: One of seconds, milliseconds, microseconds, nanoseconds.

The following twamp-sender-test-session parameters are read-only:

sender-session-state
: This read-only item can be either Active or Idle.

sent-packets
: The number of TWAMP-Test (UDP) packets belonging to this session that have been transmitted by the Session-Sender.

rcv-packets
: The number of TWAMP-Test (UDP) packets belonging to this session that have been received from the Session-Reflector.

The round trip loss for a test session can be calculated as sent-packets - rcv-packets.

last-sent-seq
: The value in the sequence number field of the last TWAMP-Test (UDP) packet transmitted for this test session. Sequence numbers start from zero, so this should always be one less than the sent-packets value.

last-rcv-seq
: The value in the sequence number field of the last TWAMP-Test (UDP) packet received for this test session. In the case of packet loss in the Session-Sender to Session-Reflector direction, this value minus the last-sent-seq will quantify the number of packets that were lost in the Session-Sender to Session-Reflector direction.

### 4.4. Session-Reflector

The twamp-session-reflector container, illustrated in Figure 6, holds items that are related to the configuration of the TWAMP Session-Reflector logical entity.

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 will then be applied to all incoming sessions.

The twamp-session-sender container includes an administrative parameter (session-reflector-admin-state) that controls whether the device is allowed to respond to incoming TWAMP test sessions. Each incoming TWAMP-Test session that is active on the Session-Reflector will be represented by an instance of a twamp-reflector-test-session object. All items in the twamp-reflector-test-session object are read-only.

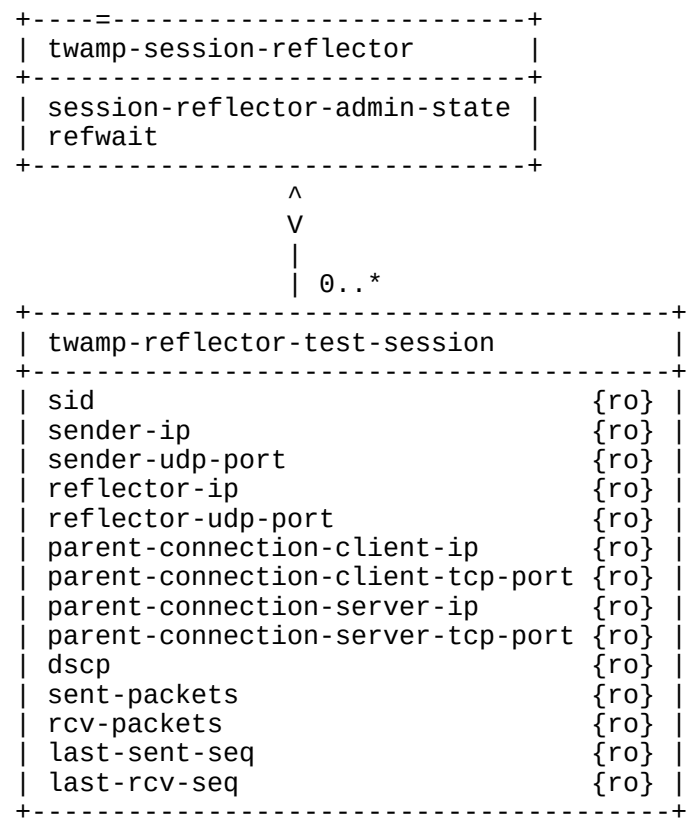



Figure 6: TWAMP Session-Reflector UML class diagram

The twamp-session-reflector configuration items are:

refwait
: The Session-Reflector MAY discontinue any session that has been started when no packet associated with that session has been received for REFWAIT seconds. The default value of REFWAIT SHALL be 900 seconds, and this waiting time MAY be configurable. This timeout allows a Session-Reflector to free up resources in case of failure.

Instances of twamp-reflector-test-session are indexed by a session identifier (sid). This value is auto-allocated by the 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 [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 twamp-client:twamp-client-ctrl-connection:twamp-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 twamp-session-reflector:twamp-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 twamp-reflector-test-session instances from the Session-Reflector device. 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 corresponds to the equivalent 4-tuple {client-ip, client-tcp-port, server-ip, server-tcp-port} in the twamp-server-ctrl-connection object. 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.

All data under twamp-reflector-test-session is read-only:

sid
: An auto-allocated identifier for this TWAMP-Test session, that is unique within the context of this Server/Session-Reflector device only. This value will be communicated to the Control-Client that requested the test session in the SID field of the Accept-Session message.

sender-ip
: 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.

sender-udp-port
: The source UDP port used in the TWAMP-Test packets belonging to this test session.

reflector-ip
: 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.

reflector-udp-port
The destination UDP port number used in the TWAMP-Test (UDP) test packets belonging to this test session.

parent-connection-client-ip
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.

parent-connection-client-tcp-port
The source TCP port number used in the TWAMP TCP control packets belonging to the parent control connection that negotiated this test session.

parent-connection-server-ip
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.

parent-connection-server-tcp-port
The destination TCP port number used in the TWAMP-Control (TCP) packets belonging to the parent control connection that negotiated this test session.

dscp The DSCP value present in the IP header of TWAMP-Test (UDP) packets belonging to this test session.

sent-packets
The number of TWAMP-Test (UDP) response packets that have been sent by the Session-Reflector for this test session.

rcv-packets
The number of TWAMP-Test (UDP) packets that have been received by the Session-Reflector for this test session. Since the Session-Reflector should respond to every test packet it receives, the sent-packets and rcv-packets values should always be identical.

last-sent-seq
The value in the sequence number field of the last TWAMP-Test (UDP) response packet transmitted for this test session.

last-rcv-seq
The value in the sequence number field of the last TWAMP-Test (UDP) packet received for 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.

```
module: ietf-twamp
   +--rw twamp
      +--rw twamp-client! {control-client}?
      |  +--rw client-admin-state              boolean
      |  +--rw mode-preference-chain* [priority]
      |  |  +--rw priority uint16
      |  |  +--rw mode? mode
      |  +--rw key-chain* [key-id]
      |  |  +--rw key-id        string
      |  |  +--rw secret-key?   string
      |  +--rw twamp-client-ctrl-connection* [ctrl-connection-name]
      |     +--rw ctrl-connection-name     string
      |     +--rw client-ip?               inet:ip-address
      |     +--rw server-ip                inet:ip-address
      |     +--rw server-tcp-port?         inet:port-number
      |     +--rw dscp?                    inet:dscp
      |     +--rw key-id?                  string
      |     +--rw max-count?               uint32
      |     +--ro client-tcp-port?         inet:port-number
      |     +--ro server-start-time?       uint64
      |     +--ro ctrl-connection-state?   ctrl-connection-state
      |     +--ro selected-mode?           mode
      |     +--ro token?                   binary
      |     +--ro client-iv?               binary
      |     +--rw twamp-session-request* [test-session-name]
      |        +--rw test-session-name     string
      |        +--rw sender-ip?            inet:ip-address
      |        +--rw sender-udp-port?      inet:port-number
      |        +--rw reflector-ip          inet:ip-address
      |        +--rw reflector-udp-port?   inet:port-number
      |        +--rw timeout?              uint64
      |        +--rw padding-length?       uint32
      |        +--rw 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 test-session-state?   test-session-state
|        +--ro sid?                  string
+--rw twamp-server! {server}?
|  +--rw server-admin-state              boolean
|  +--rw server-tcp-port?                inet:port-number
|  +--rw servwait?                       uint32
|  +--rw dscp?                           inet:dscp
|  +--rw count?                          uint32
|  +--rw max-count?                      uint32
|  +--rw modes?                          mode
|  +--rw key-chain* [key-id]
|  |  +--rw key-id        string
|  |  +--rw secret-key?   string
|  +--ro twamp-server-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 server-ctrl-connection-state? \
|              server-ctrl-connection-state
|     +--ro dscp?                              inet:dscp
|     +--ro selected-mode?                     mode
|     +--ro key-id?                            string
|     +--ro count?                             uint32
|     +--ro max-count?                         uint32
|     +--ro salt?                              binary
|     +--ro server-iv?                         binary
|     +--ro challenge?                         binary
+--rw twamp-session-sender {session-sender}?
|  +--rw session-sender-admin-state         boolean
|  +--rw twamp-sender-test-session* [test-session-name]
|     +--rw test-session-name          string
|     +--ro ctrl-connection-name?      string
|     +--rw fill-mode?                 fill-mode
|     +--rw number-of-packets?         uint32
|     +--rw (packet-distribution)?
|     |  +--:(periodic)
|     |  |  +--rw periodic-interval?         uint32
|     |  |  +--rw periodic-interval-units?   units
|     |  +--:(poisson)
|     |     +--rw lambda?                    uint32
|     |     +--rw lambda-units?              uint32
|     |     +--rw max-interval?              uint32
|     |     +--rw truncation-point-units?    units
|     +--ro sender-session-state?      sender-session-state
```

```
         |     +--ro sent-packets?              uint32
         |     +--ro rcv-packets?               uint32
         |     +--ro last-sent-seq?             uint32
         |     +--ro last-rcv-seq?              uint32
         +--rw twamp-session-reflector {session-reflector}?
            +--rw session-reflector-admin-state        boolean
            +--rw refwait?                           uint32
            +--ro twamp-reflector-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                      inet:port-number
               +--ro reflector-ip                         inet:ip-address
               +--ro reflector-udp-port                   inet:port-number
               +--ro parent-connection-client-ip?         inet:ip-address
               +--ro parent-connection-client-tcp-port?   inet:port-number
               +--ro parent-connection-server-ip?         inet:ip-address
               +--ro parent-connection-server-tcp-port?   inet:port-number
               +--ro dscp?                                inet:dscp
               +--ro sent-packets?                        uint32
               +--ro rcv-packets?                         uint32
               +--ro last-sent-seq?                       uint32
               +--ro last-rcv-seq?                        uint32
```

## 5.2. YANG Module

This section presents the YANG module for the TWAMP data model defined in this document.

```
<CODE BEGINS> file "ietf-twamp@2015-10-19.yang"
module ietf-twamp {
  namespace "urn:ietf:params:xml:ns:yang:ietf-twamp";
   //namespace need to be assigned by IANA
  prefix "ietf-twamp";

  import ietf-inet-types {
    prefix inet;
  }

  organization "IETF IPPM (IP Performance Metrics) Working Group";

  contact "draft-cmzrjp-ippm-twamp-yang@tools.ietf.org";

  description "TWAMP Data Model";

  revision "2015-10-19" {
    description "01 version. RFC5357, RFC5618, RFC5938 and RFC6038
```

```
       is covered. draft-ietf-ippm-metric-registry is also considered";

     reference "draft-cmzrjp-ippm-twamp-yang";
     }

     feature control-client {
       description "This feature relates to the device functions as
       the TWAMP Control-Client.";
     }

     feature server {
       description "This feature relates to the device functions as
       the TWAMP Server.";
     }

     feature session-sender {
       description "This feature relates to the device functions as
       the TWAMP Session-Sender.";
     }

     feature session-reflector {
       description "This feature relates to the device functions as
       the TWAMP Session-Reflector.";
     }

     typedef ctrl-connection-state {
       type enumeration {
         enum active {
           description "Control session is active.";
         }
         enum idle {
           description "Control session is idle.";
         }
       }
       description "Control connection state";
     }

     typedef mode {
       type bits {
         bit unauthenticated {
           position "0";
           description "Unauthenticated";
         }
         bit authenticated {
           position "1";
           description "Authenticated";
         }
         bit encrypted {
```

```
          position "2";
          description "Encrypted";
        }
        bit unauth-test-encrpyt-control {
          position "3";
          description "Mixed Security Mode per RFC 5618. Test
          protocol security mode in Unauthenticated mode,
          Control protocol in Encrypted mode.";
        }
        bit individual-session-control {
          position "4";
          description "Individual session control per RFC5938.";
        }
        bit reflect-octets {
          position "5";
          description "Reflect octets capability per RFC6038.";
        }
        bit symmetrical-size {
          position "6";
          description "Symmetrical size per RFC6038.";
        }
      }
      description "Authentication mode bit mask";
    }

    typedef test-session-state {
      type enumeration {
        enum ok {
          value 0;
          description "Test session is accepted.";
        }
        enum failed {
          value 1;
          description "Failure, reason unspecified (catch-all).";
        }
        enum internal-error {
          value 2;
          description "Internal error.";
        }
        enum not-supported {
          value 3;
          description "Some aspect of request is not supported.";
        }
        enum permanent-resource-limit {
          value 4;
          description "Cannot perform request due to
          permanent resource limitations.";
        }
```

```
          enum temp-resource-limit {
            value 5;
            description "Cannot perform request due to
            temporary resource limitations.";
          }
        }
        description "Test session state";
      }

      typedef server-ctrl-connection-state {
        type enumeration {
          enum "active" {
            description "Active";
          }
          enum "servwait" {
            description "Servwait";
          }
        }
        description "Server control connection state";
      }

      typedef fill-mode {
        type enumeration {
          enum zero {
            description "Zero";
          }
          enum random {
            description "Random";
          }
        }
        description "Indicates whether the padding added to the
        UDP test packets will contain pseudo-random numbers, or
        whether it should consist of all zeroes.";
      }

      typedef units  {
        type enumeration {
          enum seconds {
            description "Seconds";
          }
          enum milliseconds {
            description "Milliseconds";
          }
          enum microseconds {
            description "Microseconds";
          }
          enum nanoseconds {
            description "Nanoseconds";
```

```
        }
      }
      description "Time units";
    }

    typedef sender-session-state {
      type enumeration {
        enum setup {
          description "Test session is active.";
        }
        enum failure {
          description "Test session is idle.";
        }
      }
      description "Sender session state.";
    }

    grouping maintenance-statistics {
      description "Maintenance statistics grouping";
      leaf sent-packets {
        type uint32;
        config "false";
        description "Packets sent";
      }
      leaf rcv-packets {
        type uint32;
        config "false";
        description "Packets received";
      }
      leaf last-sent-seq {
        type uint32;
        config "false";
        description "Last sent sequence number";
      }
      leaf last-rcv-seq {
        type uint32;
        config "false";
        description "Last received sequence number";
      }
    }

    container twamp {
      description "Top level container";
      container twamp-client {
        if-feature control-client;
        presence "twamp-client";
        description "Twamp client container";
        leaf client-admin-state {
```

```
            type boolean;
            mandatory "true";
            description "Indicates whether this device is allowed to run
            TWAMP to initiate control sessions";
          }

          list mode-preference-chain {
            key "priority";
            unique "mode";
            leaf priority {
              type uint16;
              description "priority";
            }
            leaf mode {
              type mode;
              description "Authentication mode bit mask";
            }
            description "Authentication mode preference";
          }

          list key-chain {
            key "key-id";
            leaf key-id {
              type string {
                length "1..80";
              }
              description "Key ID";
            }
            leaf secret-key {
              type string;
              description "Secret key";
            }
            description "Key chain";
          }

          list twamp-client-ctrl-connection {
            key "ctrl-connection-name";
            description "Twamp client control connections";
            leaf ctrl-connection-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 "Client IP address";
            }
```

```
            leaf server-ip {
              type inet:ip-address;
              mandatory "true";
              description "Server IP address";
            }
            leaf server-tcp-port {
              type inet:port-number;
              default "862";
              description "Server tcp port";
            }
            leaf dscp{
              type inet:dscp;
              default "0";
              description "The DSCP value to be placed in the IP header
                of the TWAMP TCP Control packets generated
                by the Control-Client";
            }
            leaf key-id {
              type string {
                length "1..80";
              }
              description "Key ID";
            }
            leaf max-count {
              type uint32 {
                range 1024..4294967295;
              }
              default 32768;
              description "Max count value.";
            }
            leaf client-tcp-port {
              type inet:port-number;
              config "false";
              description "Client TCP port";
            }
            leaf server-start-time {
              type uint64;
              config "false";
              description "The Start-Time advertized by the Server in
              the Server-Start message";
            }
            leaf ctrl-connection-state {
              type ctrl-connection-state;
              config "false";
              description "Control connection state";
            }
            leaf selected-mode {
              type mode;
```

```
            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";
          }
          leaf token {
              type binary {
              length "64";
            }
            config "false";
            description "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";
          }
          leaf client-iv{
            type binary {
              length "16";
            }
            config "false";
            description "16 octets, Client-IV is generated randomly
              by the Control-Client.";
          }

          list twamp-session-request {
            key "test-session-name";
            description "Twamp session requests";
            leaf test-session-name {
              type string;
              description "A unique name for this test session to be
              used as a key for this test session on the
              Control-Client.";
            }
            leaf sender-ip {
              type inet:ip-address;
              description "Sender IP address";
            }
            leaf sender-udp-port {
              type inet:port-number;
              description "Sender UDP port";
            }
            leaf reflector-ip {
              type inet:ip-address;
              mandatory "true";
              description "Reflector IP address.";
            }
            leaf reflector-udp-port {
              type inet:port-number;
```

```
            description "Reflector UDP port. If this value is not
            set, the device shall use the same port number as
            defined in the server-tcp-port parameter of this
            twamp-session-request's
            parent client-control-connection.";
          }
          leaf timeout {
            type uint64;
            default "2";
            description "The time (in seconds)Session-Reflector MUST
            wait after receiving a Stop-Session message.";
          }
          leaf padding-length {
            type uint32{
              range "64..4096";
            }
            description "The number of bytes of padding that should
                be added to the UDP test packets generated by the
                sender. Jumbo sized packets supported.";
          }
          leaf dscp {
            type inet:dscp;
            description "The DSCP value to be placed in the UDP
            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.";
          }
          leaf start-time {
            type uint64;
            default "0";
            description "Time when the session is to be started
            (but not before the Start-Sessions command is issued).
            This value is placed in the Start Time field of the
            Request-TW-Session message. 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;
            default "0";
            description "Determines if the test session is to be
            run repeatedly. The default value of repeat is 0,
            indicating that once the session has completed, it
            will not be renegotiated and restarted";
          }
          leaf repeat-interval  {
            when "../repeat!='0'" {
```

```
                description "When repeat is not 0, the test is to be
                repeated";
              }
              type uint32;
              description "Repeat interval (in minutes)";
            }

            list pm-reg-list {
              key "pm-index";
              leaf pm-index {
                type uint16;
                description "One or more Numerical index values of a
                Registered Metric in the Performance Metric Registry";
              }
              description "A list of one or more pm-index values,
              which communicate packet stream characteristics and one
              or more metrics to be measured.";
            }
            leaf test-session-state {
              type test-session-state;
              config "false";
              description "Test session state";
            }
            leaf sid{
              type string;
              config "false";
              description "The SID allocated by the Server for
              this test session";
            }
          }
        }
      }

      container twamp-server{
        if-feature server;
        presence "twamp-server";
        description "Twamp sever container";
        leaf server-admin-state{
          type boolean;
          mandatory "true";
          description "Indicates whether this device is allowed to run
            TWAMP to respond to control sessions";
        }
        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.";
```

```
        }
        leaf servwait {
          type uint32 {
            range 1..604800;
          }
          default 900;
          description "SERVWAIT (TWAMP Control (TCP) session timeout),
            default value is 900";
        }
        leaf dscp {
          type inet:dscp;
          description "The DSCP value to be placed in the IP header of
          TCP TWAMP-Control packets generated by the Server";
        }
        leaf count {
          type uint32 {
            range 1024..4294967295;
          }
          description "Parameter used in deriving a key from a
          shared secret ";
        }
        leaf max-count {
          type uint32 {
            range 1024..4294967295;
          }
          default 32768;
          description "Max count value.";
        }
        leaf modes {
          type mode;
          description "The bit mask of TWAMP Modes this Server
          instance is willing to support.";
        }

        list key-chain {
          key "key-id";
          leaf key-id {
            type string {
              length "1..80";
            }
            description "Key IDs.";
          }
          leaf secret-key {
            type string;
            description "Secret keys.";
          }
          description "KeyIDs with the respective secret keys.";
        }
```

```
        list twamp-server-ctrl-connection {
          key "client-ip client-tcp-port server-ip server-tcp-port";
          config "false";
          description "Twamp server control connections";
          leaf client-ip {
            type inet:ip-address;
            description "Client IP address";
          }
          leaf client-tcp-port {
            type inet:port-number;
            description "Client TCP port";
          }
          leaf server-ip {
            type inet:ip-address;
            description "Server IP address";
          }
          leaf server-tcp-port {
            type inet:port-number;
            description "Server TCP port";
          }
          leaf server-ctrl-connection-state {
            type server-ctrl-connection-state;
            description "Server control connection state";
          }
          leaf dscp {
            type inet:dscp;
            description "The DSCP value used in the IP header of the
            TCP control packets sent by the Server for this control
            connection. This will usually be the same value as is
            configured for twamp-server:dscp under the twamp-server.
            However, in the event that the user re-configures
            twamp-server:dscp after this control connection is already
            in progress, this read-only value will show the actual
            dscp value in use by this control connection.";
          }
          leaf selected-mode {
            type mode;
            description "The mode that was chosen for this control
            connection as set in the Mode field of the
            Set-Up-Response message.";
          }
          leaf key-id {
            type string {
              length "1..80";
            }
            description "The key-id value that is in use by this
                    control connection.";
          }
```

```
            leaf count {
              type uint32 {
                range 1024..4294967295;
              }
              description "The count value that is in use by this control
              connection. This will usually be the same value as is
              configured under twamp-server. However, in the event that
              the user re-configured twamp-server:count after this
              control connection is already in progress, this read-only
              value will show the different count that is in use for
              this control connection.";
            }
            leaf max-count {
              type uint32 {
                range 1024..4294967295;
              }
              description "The max-count value that is in use by this
              control connection. This will usually be the same value
              as is configured under twamp-server. However, in the
              event that the user re-configured twamp-server:max-count
              after this control connection is already in progress,
              this read-only value will show the different max-count
              that is in use for this control connection.";
            }
            leaf salt{
              type binary {
                length "16";
              }
              description "Salt MUST be generated pseudo-randomly";
            }
            leaf server-iv {
              type binary {
                length "16";
              }
              description "16 octets, Server-IV is generated randomly
              by the Control-Client.";
            }
            leaf challenge {
              type binary {
                length "16";
              }
              description "Challenge is a random sequence of octets
              generated by the Server";
            }
          }
        }

        container twamp-session-sender{
```

```
        if-feature session-sender;
        description "Twamp session sender container";
        leaf session-sender-admin-state {
          type boolean;
          mandatory "true";
          description "Indicates whether this device is allowed to run
          TWAMP to initiate test sessions";
        }
        list twamp-sender-test-session{
          key "test-session-name";
          description "Twamp sender test sessions";
          leaf test-session-name {
            type string;
            description "A unique name for this test session to be
            used as a key for this test session by the Session-Sender
            logical entity.";
          }
          leaf ctrl-connection-name {
            type string;
            config "false";
            description "The name of the parent control connection
            that is responsible for negotiating this test session.";
          }
          leaf fill-mode {
            type fill-mode;
            default zero;
            description "Indicates whether the padding added to the
            UDP test packets will contain pseudo-random numbers, or
            whether it should consist of all zeroes.";
          }
          leaf number-of-packets {
            type uint32;
            description "The overall number of UDP test packets to be
              transmitted by the sender for this test session.";
          }
          choice packet-distribution {
            description "Packet distributions, poisson or periodic";
            case periodic {
              leaf periodic-interval {
                type uint32;
                description "Periodic interval";
              }
              leaf periodic-interval-units  {
                type units;
                description "Periodic interval units";
              }
            }
            case poisson {
```

```
            leaf lambda{
              type uint32;
              description "The average rate of
              packet transmission.";
            }
            leaf lambda-units{
              type uint32;
              description "Lambda units.";
            }
            leaf max-interval{
              type uint32;
              description "maximum time between packet
              transmissions.";
            }
            leaf truncation-point-units{
              type units;
              description "Truncation point units";
            }
          }
        }
        leaf sender-session-state {
          type sender-session-state;
          config "false";
          description "Sender session state.";
        }
        uses maintenance-statistics;
      }
    }

    container twamp-session-reflector {
      if-feature session-reflector;
      description "Twamp session reflector container";
      leaf session-reflector-admin-state {
        type boolean;
        mandatory "true";
        description "Indicates whether this device is allowed to run
        TWAMP to respond to test sessions";
      }
      leaf refwait {
        type uint32 {
          range 1..604800;
        }
        default 900;
        description "REFWAIT (TWAMP test session timeout),
          the default value is 900";
      }

      list twamp-reflector-test-session {
```

```
            key "sender-ip sender-udp-port reflector-ip
                reflector-udp-port";
            config "false";
            description "Twamp reflector test sessions";
            leaf sid{
              type string;
              description "An auto-allocated identifier for this test
              session, that is unique within the context of this
              Server/Session-Reflector device only. ";
            }
            leaf sender-ip {
              type inet:ip-address;
              description "Sender IP address.";
            }
            leaf sender-udp-port {
              type inet:port-number;
              description "Sender UDP port.";
            }
            leaf reflector-ip {
              type inet:ip-address;
              description "Reflector IP address.";
            }
            leaf reflector-udp-port {
              type inet:port-number;
              description "Reflector UDP port.";
            }
            leaf parent-connection-client-ip {
              type inet:ip-address;
              description "Parent connction client IP address.";
            }
            leaf parent-connection-client-tcp-port {
              type inet:port-number;
              description "Parent connection client TCP port.";
            }
            leaf parent-connection-server-ip {
              type inet:ip-address;
              description "Parent connection server IP address.";
            }
            leaf parent-connection-server-tcp-port {
              type inet:port-number;
              description "Parent connection server TCP port";
            }
            leaf dscp {
              type inet:dscp;
              description "The DSCP value present in the IP header of
              TWAMP UDP test packets belonging to this test 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. A more elaborated example, which also includes authentication parameters, is provided in Appendix A.

### 6.1. Control-Client

The following configuration example shows 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.

```
<twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
   <twamp-client>
      <client-admin-state>True</client-admin-state>
   </twamp-client>
</twamp>
```

The following configuration example shows a Control-Client with two instances of twamp-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.

```
<twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
   <twamp-client>

      <twamp-client-ctrl-connection>
         <ctrl-connection-name>RouterA</ctrl-connection-name>
         <client-ip>203.0.113.1</client-ip>
         <server-ip>203.0.113.2</server-ip>
         <twamp-session-request>
            <test-session-name>Test1</test-session-name>
            <sender-ip>10.1.1.1</sender-ip>
            <sender-udp-port>4000</sender-udp-port>
            <reflector-ip>10.1.1.2</reflector-ip>
            <reflector-udp-port>5000</reflector-udp-port>
            <start-time>0</start-time>
         </twamp-session-request>
         <twamp-session-request>
            <test-session-name>Test2</test-session-name>
            <sender-ip>203.0.113.1</sender-ip>
            <sender-udp-port>4001</sender-udp-port>
            <reflector-ip>203.0.113.2</reflector-ip>
            <reflector-udp-port>5001</reflector-udp-port>
            <start-time>0</start-time>
         </twamp-session-request>
      </twamp-client-ctrl-connection>

      <twamp-client-ctrl-connection>
         <ctrl-connection-name>RouterB</ctrl-connection-name>
         <client-ip>203.0.113.1</client-ip>
         <server-ip>203.0.113.3</server-ip>
      </twamp-client-ctrl-connection>

   </twamp-client>
</twamp>
```

### 6.2. Server

This configuration example shows a Server with server-admin-state enabled, which permits a device following Figure 2 to respond to TWAMP-Control connections and TWAMP-Test sessions.

```
<twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
   <twamp-server>
      <server-admin-state>True</server-admin-state>
   </twamp-server>
</twamp>
```

The following example presents a Server with the TWAMP-Control connection corresponding to the control connection name (ctrl-connection-name) "RouterA" presented in Section 6.1.

```
<twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
   <twamp-server>

      <twamp-server-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>
         <server-ctrl-connection-state>
            active
         </server-ctrl-connection-state>
      </twamp-server-ctrl-connection>

   </twamp-server>
</twamp>
```

### 6.3. Session-Sender

The following configuration example shows a Session-Sender with the two TWAMP-Test sessions presented in Section 6.1.

```
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
     <twamp-session-sender>

        <twamp-sender-test-session>
           <test-session-name>Test1</test-session-name>// read-only
           <ctrl-connection-name>RouterA</ctrl-connection-name>
           <number-of-packets>900</number-of-packets>
           <packet-distribution>
              <periodic-interval>1</periodic-interval>
              <periodic-interval-units>seconds</periodic-interval-units>
           </packet-distribution>
        </twamp-sender-test-session>

        <twamp-sender-test-session>
           <test-session-name>Test2</test-session-name>
           <ctrl-connection-name>
             RouterA
           </ctrl-connection-name> // read-only
           <number-of-packets>900</number-of-packets>
           <packet-distribution>
              <lambda>1</lambda>
              <lambda-units>1</lambda-units>
              <max-interval>2</max-interval>
              <truncation-point-units>seconds</truncation-point-units>
          </packet-distribution>
        </twamp-sender-test-session>

     </twamp-session-sender>
  </twamp>
```

### 6.4. Session-Reflector

The following example shows the two Session-Reflector TWAMP-Test sessions corresponding to the test sessions presented in Section 6.3.

```
   <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <twamp-session-reflector>

          <twamp-reflector-test-session>
            <sid>1232</sid>
            <sender-ip>10.1.1.1</sender-ip>
            <reflector-ip>10.1.1.2</reflector-ip>
            <sender-udp-port>4000</sender-udp-port>
            <reflector-udp-port>5000</reflector-udp-port>
            <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-server-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>
         </twamp-reflector-test-session>

         <twamp-reflector-test-session>
            <sid>178943</sid>
            <sender-ip>203.0.113.1</sender-ip>
            <reflector-ip>192.68.0.2</reflector-ip>
            <sender-udp-port>4001</sender-udp-port>
            <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-server-tcp-port>
            <reflector-udp-port>5001</reflector-udp-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>
         </twamp-reflector-test-session>

      </twamp-session-reflector>
   </twamp>
```

## 7. Security Considerations

TBD

## 8. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made.

URI: urn:ietf:params:xml:ns:yang:ietf-twamp

Registrant Contact: The IPPM WG of the IETF.

XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

name: ietf-twamp

namespace: urn:ietf:params:xml:ns:yang:ietf-twamp

prefix: twamp

reference: RFC XXXX

## 9. Acknowledgements

We thank Gregory Mirsky, Kevin D'Souza, and Robert Sherman for their thorough and constructive reviews, comments and text suggestions.

Haoxing Shen contributed to the definition of the YANG module in Section 5.

Kostas Pentikousis is 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. References

### 10.1. Normative References

[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>.

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[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>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://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, <http://www.rfc-editor.org/info/rfc6038>.

## 10.2. Informative 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-05 (work in progress), October 2015.

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-08 (work in progress), October 2015.

[I-D.unify-nfvrg-challenges]
Szabo, R., Csaszar, A., Pentikousis, K., Kind, M., Daino, D., Qiang, Z., and H. Woesner, "Unifying Carrier and Cloud Networks: Problem Statement and Challenges", draft-unify-nfvrg-challenges-02 (work in progress), July 2015.

[I-D.unify-nfvrg-devops]
Meirosu, C., Manzalini, A., Steinert, R., Marchetto, G., Papafili, I., Pentikousis, K., and S. Wright, "DevOps for Software-Defined Telecom Infrastructures", draft-unify-nfvrg-devops-03 (work in progress), October 2015.

[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.

[RFC2898] Kaliski, B., "PKCS #5: Password-Based Cryptography Specification Version 2.0", RFC 2898, DOI 10.17487/RFC2898, September 2000, <http://www.rfc-editor.org/info/rfc2898>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <http://www.rfc-editor.org/info/rfc4086>.

[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>.

[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, <http://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, <http://www.rfc-editor.org/info/rfc6241>.

[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, <http://www.rfc-editor.org/info/rfc7426>.

## 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

```
   <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <twamp-client>

         <client-admin-state>True</client-admin-state>

         <mode-preference-chain>
            <priority>0</priority>
            <mode>0x00000002</mode>
         </mode-preference-chain>
         <mode-preference-chain>
            <priority>1</priority>
            <mode>0x00000001</mode>
         </mode-preference-chain>

         <keychain>
            <keyid>KeyClient1ToRouterA</keyid>
            <secret-key>secret1</secret-key>
         </keychain>
         <keychain>
            <keyid>KeyForRouterB</keyid>
            <secret-key>secret2</secret-key>
         </keychain>

         <twamp-client-ctrl-connection>
            <ctrl-connection-name>RouterA</ctrl-connection-name>
            <client-ip>203.0.113.1</client-ip>
            <server-ip>203.0.113.2</server-ip>
            <dscp>32</dscp>
            <key-id>KeyClient1ToRouterA</key-id>
            <twamp-session-request>
               <test-session-name>Test1</test-session-name>
               <sender-ip>10.1.1.1</sender-ip>
               <sender-udp-port>4000</sender-udp-port>
               <reflector-ip>10.1.1.2</reflector-ip>
               <reflector-udp-port>5000</reflector-udp-port>
               <padding-length>0</padding-length>
               <start-time>0</start-time>
               <test-session-state>ok</test-session-state>
               <sid>1232</sid>
            </twamp-session-request>
            <twamp-session-request>
               <test-session-name>Test2</test-session-name>
               <sender-ip>203.0.113.1</sender-ip>
               <sender-udp-port>4001</sender-udp-port>
               <reflector-ip>203.0.113.2</reflector-ip>
               <reflector-udp-port>5001</reflector-udp-port>
```

```
               <paddingLenth>32</paddingLenth>
               <start-time>0</start-time>
               <test-session-state>ok</test-session-state>
               <sid>178943</sid>
            </twamp-session-request>
         </twamp-client-ctrl-connection>

      </twamp-client>
   </twamp>
```

A.2. Server

```
   <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <twamp-server>

         <server-admin-state>True</server-admin-state>
         <servwait>1800</servwait>
         <dscp>32</dscp>
         <modes>0x00000003</modes>
         <count>256</count>

         <keychain>
            <keyid>KeyClient1ToRouterA</keyid>
            <secret-key>secret1</secret-key>
         </keychain>
         <keychain>
            <keyid>KeyClient10ToRouterA</keyid>
            <secret-key>secret10</secret-key>
         </keychain>

         <twamp-server-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>
            <server-ctrl-connection-state>
               active
             </server-ctrl-connection-state>
            <dscp>32</dscp>
            <selected-mode>0x00000002</selected-mode>
            <key-id>KeyClient1ToRouterA</key-id>
            <count>1024</count>
         </twamp-server-ctrl-connection>

      </twamp-server>
   </twamp>
```

### A.3. Session-Sender

```
  <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
     <twamp-session-sender>

        <twamp-sender-test-session>
           <test-session-name>Test1</test-session-name> // read-only
           <ctrl-connection-name>RouterA</ctrl-connection-name>
           <dscp>32</dscp>
           <fill-mode>zero</fill-mode>
           <number-of-packets>900</number-of-packets>
           <packet-distribution>
              <periodic-interval>1</periodic-interval>
              <periodic-interval-units>seconds</periodic-interval-units>
           </packet-distribution>
           <sender-session-state>Active</sender-session-state>
           <sent-packets>2</sent-packets>
           <rcv-packets>2</rcv-packets>
           <last-sent-seq>1</last-sent-seq>
           <last-rcv-seq>1</last-rcv-seq>
        </twamp-sender-test-session>

        <twamp-sender-test-session>
           <test-session-name>Test2</test-session-name>
           <ctrl-connection-name>
             RouterA
           </ctrl-connection-name> // read-only
           <dscp>32</dscp>
           <fill-mode>random</fill-mode>
           <number-of-packets>900</number-of-packets>
           <packet-distribution>
             <lambda>1</lambda>
             <lambda-units>1</lambda-units>
             <max-interval>2</max-interval>
             <truncation-point-units>seconds</truncation-point-units>
           </packet-distribution>
           <sender-session-state>Active</sender-session-state>
           <sent-packets>21</sent-packets>
           <rcv-packets>21</rcv-packets>
           <last-sent-seq>20</last-sent-seq>
           <last-rcv-seq>20</last-rcv-seq>
        </twamp-sender-test-session>

     </twamp-session-sender>
  </twamp>
```

### A.4. Session-Reflector

```
   <twamp xmlns="urn:ietf:params:xml:ns:yang:ietf-twamp">
      <twamp-session-reflector>

         <twamp-reflector-test-session>
            <sid>1232</sid>
            <sender-ip>10.1.1.1</sender-ip>
            <reflector-ip>10.1.1.2</reflector-ip>
            <sender-udp-port>4000</sender-udp-port>
            <reflector-udp-port>5000</reflector-udp-port>
            <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-server-tcp-port>
            <dscp>32</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>
         </twamp-reflector-test-session>

         <twamp-reflector-test-session>
            <sid>178943</sid>
            <sender-ip>203.0.113.1</sender-ip>
            <reflector-ip>192.68.0.2</reflector-ip>
            <sender-udp-port>4001</sender-udp-port>
            <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-server-tcp-port>
            <reflector-udp-port>5001</reflector-udp-port>
```

```
            <dscp>32</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>
          </twamp-reflector-test-session>

      </twamp-session-reflector>
   </twamp>
```

## Appendix B. TWAMP Operational Commands

This document is targeted at configuration details for TWAMP. Operational actions such as how TWAMP sessions are started/stopped, how results are retrieved, or stored results are cleared, and so on, are not addressed by this configuration model and are out of scope of this document.

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 possible to define RPC operations for actions such as starting or stopping control or test sessions or groups of sessions; retrieving results; clearing stored results, and so on.

However, [RFC5357] does not attempt to describe such operational actions, and it is likely that different TWAMP implementations could 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
URI: http://home.comcast.net/~acmacm/

Lianshu Zheng
Huawei Technologies
China

Email: vero.zheng@huawei.com

Reshad Rahman
Cisco Systems
2000 Innovation Drive
Kanata, ON K2K 3E8
Canada

Email: rrahman@cisco.com

Mahesh Jethanandani
Ciena Corporation
3939 North 1st Street
San Jose, CA 95134
USA

Email: mjethanandani@gmail.com
URI: www.ciena.com

Kostas Pentikousis (editor)
EICT GmbH
EUREF-Campus Haus 13
Torgauer Strasse 12-15
10829 Berlin
Germany

Email: k.pentikousis@eict.de

INTERNET-DRAFT

N. Elkins
Inside Products
R. Hamilton
Chemical Abstracts Service
M. Ackermann
BCBS Michigan

Intended Status: Proposed Standard
Expires: August 2015

February 14, 2015

# IPPM Considerations for the IPv6 PDM Destination Option
# draft-elkins-ippm-pdm-option-03

Table of Contents

1 Background . . . . . 4
1.1 Terminology . . . . . 4
1.2 End User Quality of Service (QoS) . . . . . 4
1.3 Need for a Packet Sequence Number . . . . . 5
1.4 Rationale for proposed solution . . . . . 5
1.5 PDM Works in Collaboration with Other Headers . . . . . 5
2 Measurement Information Derived from PDM . . . . . 6
2.1 Round-Trip Delay . . . . . 6
2.2 Server Delay . . . . . 6
3 Performance and Diagnostic Metrics Destination Option Layout . . 7
3.1 Destination Options Header . . . . . 7
3.2 Performance and Diagnostic Metrics Destination Option . . . 7
4 Considerations of Timing Representation . . . . . 10
4.1 Encoding the Delta-Time Values . . . . . 10
4.2 Timer registers are different on different hardware . . . . 10
4.3 Timer Units on Other Systems . . . . . 11
4.4 Time Base . . . . . 11
4.5 Timer-value scaling . . . . . 12
4.6 Limitations with this encoding method . . . . . 13
4.7 Lack of precision induced by timer value truncation . . . . 14
5 PDM Flow - Simple Client Server . . . . . 15
5.1 Step 1 . . . . . 15
5.2 Step 2 . . . . . 16
5.3 Step 3 . . . . . 16
5.4 Step 4 . . . . . 17
5.5 Step 5 . . . . . 18
6 Other Flows . . . . . 19
6.1 PDM Flow - One Way Traffic . . . . . 19
6.2 PDM Flow - Multiple Send Traffic . . . . . 20
6.3 PDM Flow - Multiple Send with Errors . . . . . 21
7 Potential Overhead Considerations . . . . . 22
8 Security Considerations . . . . . 23

9 IANA Considerations . . . . . 23
10 References . . . . . 23
10.1 Normative References . . . . . 23
10.2 Informative References . . . . . 24
11 Acknowledgments . . . . . 24
Authors' Addresses . . . . . 24

## Abstract

To assess performance problems, measurements based on optional sequence numbers and timing may be embedded in each packet. Such measurements may be interpreted in real-time or after the fact. An implementation of the existing IPv6 Destination Options extension header, the Performance and Diagnostic Metrics (PDM) Destination Options extension header has been proposed in a companion document. This document specifies the field limits, calculations, and usage of the PDM in measurement.

## Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

## Copyright and License Notice

# 1 Background

To assess performance problems, measurements based on optional sequence numbers and timing may be embedded in each packet. Such measurements may be interpreted in real-time or after the fact. An implementation of the existing IPv6 Destination Options extension header, the Performance and Diagnostic Metrics (PDM) Destination Options extension header has been proposed in a companion document. This document specifies the field limits, calculations, and usage of the PDM in measurement.

## 1.1 Terminology

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

## 1.2 End User Quality of Service (QoS)

The difference between timing values in the PDM traveling along with the packet will be used to estimate QoS as experienced by an end user device.

For many applications, the key user performance indicator is response time. When the end user is an individual, he is generally indifferent to what is happening along the network; what he really cares about is how long it takes to get a response back. But this is not just a matter of individuals' personal convenience. In many cases, rapid response is critical to the business being conducted.

When the end user is a device (e.g. with the Internet of Things), what matters is the speed with which requested data can be transferred -- specifically, whether the requested data can be transferred in time to accomplish the desired actions. This can be important when the relevant external conditions are subject to rapid change.

Response time and consistency are not just "nice to have". On many networks, the impact can be financial hardship or endanger human life. In some cities, the emergency police contact system operates over IP, law enforcement uses TCP/IP networks, transactions on our stock exchanges are settled using IP networks. The critical nature of such activities to our daily lives and financial well-being demand a solution.

### 1.3 Need for a Packet Sequence Number

While performing network diagnostics of an end-to-end connection, it often becomes necessary to find the device along the network path creating problems. Diagnostic data may be collected at multiple places along the path (if possible), or at the source and destination. Then, the diagnostic data corresponding to each packet at different observation points must be matched for proper measurements. A sequence number in each packet provides sufficient basis for the matching process. If need be, the timing fields may be used along with the sequence number to ensure uniqueness.

This method of data collection along the path is of special use to determine where packet loss or packet corruption is happening.

The packet sequence number needs to be unique in the context of the session (5-tuple).

### 1.4 Rationale for proposed solution

The current IPv6 specification does not provide timing nor a similar field in the IPv6 main header or in any extension header. So, we propose the IPv6 Performance and Diagnostic Metrics destination option (PDM) [ELK-PDM].

Advantages include:

1. Real measure of actual transactions.
2. Independence from transport layer protocols.
3. Ability to span organizational boundaries with consistent instrumentation
4. No time synchronization needed between session partners

The PDM provides the ability to quickly determine if the (latency) problem is in the network or in the server (application). More intermediate measurements may be needed if the host or network discrimination is not sufficient. At the client, TCP/IP stack time vs. applications time may still need to be broken out by client software.

### 1.5 PDM Works in Collaboration with Other Headers

The purpose of the PDM is not to supplant all the variables present in all other headers but to provide data which is not available or very difficult to get. The way PDM would be used is by a technician (or tool) looking at a packet capture. Within the packet capture,

they would have available to them the layer 2 header, IP header (v6 or v4), TCP, UCP, ICMP, SCTP or other headers. All information would be looked at together to make sense of the packet flow. The technician or processing tool could analyze, report or ignore the data from PDM, as necessary.

For an example of how PDM can help with TCP retransmit problems, please look at section 8.

## 2 Measurement Information Derived from PDM

Each packet contains information about the sender and receiver. In IP protocol, the identifying information is called a "5-tuple".

The 5-tuple consists of:

SADDR : IP address of the sender
SPORT : Port for sender
DADDR : IP address of the destination
DPORT : Port for destination
PROTC : Protocol for upper layer (ex. TCP, UDP, ICMP, etc.)

The PDM contains the following metrics:

PSNTP : Packet Sequence Number This Packet
PSNLR : Packet Sequence Number Last Received
DELTALR : Delta Last Received
PSNLS : Packet Sequence Number Last Sent
DELTALS : Delta Last Sent

This information, combined with the 5-tuple, allows the measurement of the following metrics:

1. Round-trip delay
2. Server delay

### 2.1 Round-Trip Delay

Round-trip delay is the end-to-end delay for a packet from a source host to a destination host. This measurement has been defined, and the advantages and disadvantages discussed in "A Round-trip Delay Metric for IPPM" [RFC2681].

### 2.2 Server Delay

Server delay is the interval between when a packet is received by a device and the first corresponding packet is sent back in response.

This may be "Server Processing Time". It may also be a delay caused by acknowledgements. Server processing time includes the time taken by the combination of the stack and application to return the response. The stack delay may be related to network performance. If this aggregate time is seen as a problem, and there is a need to make a clear distinction between application processing time and stack delay, including that caused by the network, then more client based measurements are needed.

# 3 Performance and Diagnostic Metrics Destination Option Layout

## 3.1 Destination Options Header

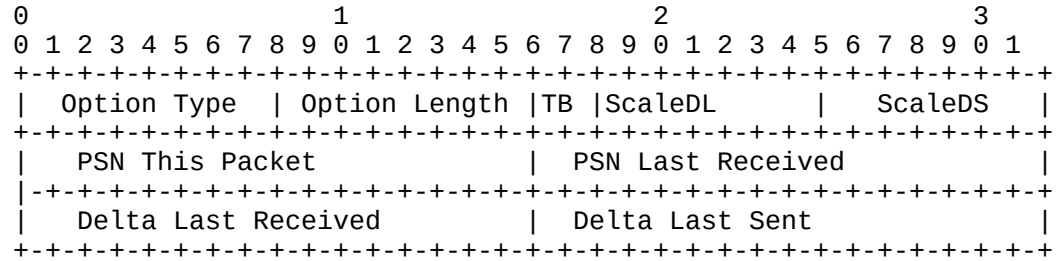
The IPv6 Destination Options Header is used to carry optional information that need be examined only by a packet's destination node(s). The Destination Options Header is identified by a Next Header value of 60 in the immediately preceding header and is defined in RFC2460 [RFC2460]. The IPv6 Performance and Diagnostic Metrics Destination Option (PDM) is an implementation of the Destination Options Header (Next Header value = 60). The PDM does not require time synchronization.

## 3.2 Performance and Diagnostic Metrics Destination Option

The IPv6 Performance and Diagnostic Metrics Destination Option (PDM) contains the following fields:

```
TIMEBASE : Base timer unit
SCALEDL  : Scale for Delta Last Received
SCALEDS  : Scale for Delta Last Sent
PSNTP    : Packet Sequence Number This Packet
PSNLR    : Packet Sequence Number Last Received
DELTALR  : Delta Last Received
DELTALS  : Delta Last Sent
```

The PDM destination option is encoded in type-length-value (TLV) format as follows:



Option Type

TBD = 0xXX (TBD) [To be assigned by IANA] [RFC2780]

Option Length

8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Option Length fields. This field MUST be set to 16.

Time Base

2-bit unsigned integer. It will indicate the lowest granularity possible for this device. That is, for a value of 00 in the Time Base field, a value of 1 in the DELTA fields indicates 1 picosecond.

This field is being included so that a device may choose the granularity which most suits its timer ticks. That is, so that it does not have to do more work than needed to convert values required for the PDM.

The possible values of Time Base are as follows:

00 - milliseconds
01 - microseconds
10 - nanoseconds
11 - picoseconds

Scale Delta Last Received (SCALEDLR)

7-bit signed integer. This is the scaling value for the Delta Last Received (DELTALR) field. The possible values are from -128 to +127. See Section 4 for further discussion on Timing Considerations and formatting of the scaling values.

Scale Delta Last Sent (SCALEDLS)

7-bit signed integer. This is the scaling value for the Delta Last Sent (DELTALS) field. The possible values are from -128 to +127.

Packet Sequence Number This Packet (PSNTP)

16-bit unsigned integer. This field will wrap. It is intended for human use. That is, while to be used while analyzing packet traces.

Initialized at a random number and monotonically incremented for each

packet on the 5-tuple. The 5-tuple consists of the source and destination IP addresses, the source and destination ports, and the upper layer protocol (ex. TCP, ICMP, etc). The random number initialization is to make it harder to spoof and insert such packets.

Operating systems MUST implement a separate packet sequence number counter per 5-tuple. Operating systems MUST NOT implement a single counter for all connections.

Packet Sequence Number Last Received (PSNLR)

16-bit unsigned integer. This is the PSN of the packet last received on the 5-tuple.

Delta Last Received (DELTALR)

A 16-bit unsigned integer field. The value is according to the scale in SCALEDLR.

DELTALR = Send time packet 2 - Receive time packet 1

Delta Last Sent (DELTALS)

A 16-bit unsigned integer field. The value is according to the scale in SCALEDS.

Delta Last Sent = Receive time packet 2 - Send time packet 1

Option Type

The two highest-order bits of the Option Type field are encoded to indicate specific processing of the option; for the PDM destination option, these two bits MUST be set to 00. This indicates the following processing requirements:

00 - skip over this option and continue processing the header.

RFC2460 [RFC2460] defines other values for the Option Type field. These MUST NOT be used in the PDM. The other values are as follows:

01 - discard the packet.

10 - discard the packet and, regardless of whether or not the packet’s Destination Address was a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet’s Source Address,

pointing to the unrecognized Option Type.

11 - discard the packet and, only if the packet's Destination Address was not a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

In keeping with RFC2460 [RFC2460], the third-highest-order bit of the Option Type specifies whether or not the Option Data of that option can change en-route to the packet's final destination.

In the PDM, the value of the third-highest-order bit MUST be 0. The possible values are as follows:

0 - Option Data does not change en-route

1 - Option Data may change en-route

The three high-order bits described above are to be treated as part of the Option Type, not independent of the Option Type. That is, a particular option is identified by a full 8-bit Option Type, not just the low-order 5 bits of an Option Type.

# 4 Considerations of Timing Representation

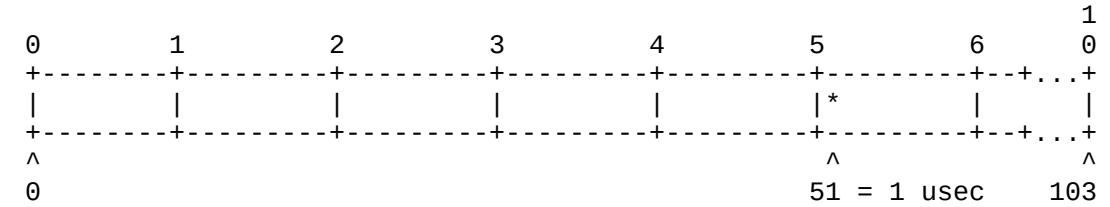
## 4.1 Encoding the Delta-Time Values

This section makes reference to and expands on the document "Encoding of Time Intervals for the TCP Timestamp Option" [TRAM-TCPM].

## 4.2 Timer registers are different on different hardware

One of the problems with timestamp recording is the variety of hardware that generates the time value to be used. Different CPUs track the time in registers of different sizes, and the most-frequently-iterated bit could be the first on the left or the first on the right. In order to generate some examples here it is necessary to indicate the type of timer register being used.

As described in the "IBM z/Architecture Principles of Operation" [IBM-POPS], the Time-Of-Day clock in a zSeries CPU is a 104-bit register, where bit 51 is incremented approximately every microsecond:




To represent these values concisely a hexadecimal representation will be used, where each digit represents 4 binary bits. Thus:

0000 0000 0000 0001 = 1 timer unit (2**-12 usec, or about 244 psec)
0000 0000 0000 1000 = 1 microsecond
0000 0000 003E 8000 = 1 millisecond
0000 0000 F424 0000 = 1 second
0000 0039 3870 0000 = 1 minute
0000 0D69 3A40 0000 = 1 hour
0001 41DD 7600 0000 = 1 day

Note that only the first 64 bits of the register are commonly represented, as that represents a count of timer units on this hardware. Commonly the first 52 bits are all that are displayed, as that represents a count of microseconds.

## 4.3 Timer Units on Other Systems

This encoding method works the same with other hardware clock formats. The method uses a microsecond as the basic value and allows for large time differentials.

## 4.4 Time Base

We propose a base unit for the time. This is a 2-bit integer indicating the lowest granularity possible for this device. That is, for a value of 00 in the Time Base field, a value of 1 in the DELTA fields indicates 1 picosecond.

The possible values of Time Base are as follows:

```
00 - milliseconds
01 - microseconds
10 - nanoseconds
11 - picoseconds
```

Time base is not necessarily equivalent to length of one timer tick. That is, on many, if not all, systems, the timer tick value will not be in complete units of nanoseconds, milliseconds, etc. For example, on an IBM zSeries machine, one timer tick (or clock unit) is 2 to the -12th microseconds.

Therefore, some amount of conversion may be needed to approximate Time Base units.

## 4.5 Timer-value scaling

As discussed in [TRAM-TCPM] we propose storing not an entire time-interval value, but just the most significant bits of that value, along with a scaling factor to indicate the magnitude of the time-interval value. In our case, we will use the high-order 16 bits. The scaling value will be the number of bits in the timer register to the right of the 16th significant bit. That is, if the timer register contains this binary value:

```
1110100011010100101001010001000000000000
<-16 bits     -><-24 bits             ->
```

then, the values stored would be 1110 1000 1101 0100 in binary (E8D4 hexadecimal) for the time value and 24 for the scaling value. Note that the displayed value is the binary equivalent of 1 second expressed in picoseconds.

The below table represents a device which has a TimeBase of picosecond (or 00). The smallest and simplest value to represent is 1 picosecond; the time value stored is 1, and the scaling value is 0. Using values from the table below, we have:

| Delta time | Time value in picoseconds | Encoded value | Scaling decimal |
|---|---|---|---|
| 1 picosecond | 1 | 1 | 0 |
| 1 nanosecond | 3E8 | 3E8 | 0 |
| 1 microsecond | F4240 | F424 | 4 |
| 1 millisecond | 3B9ACA00 | 3B9A | 16 |
| 1 second | E8D4A51000 | E8D4 | 24 |
| 1 minute | 3691D6AFC000 | 3691 | 32 |
| 1 hour | cca2e51310000 | CCA2 | 36 |
| 1 day | 132f4579c980000 | 132F | 44 |
| 365 days | 1b5a660ea44b80000 | 1B5A | 52 |

Sample binary values (high order 16 bits taken)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 psec | 1 | | | | | | | | 0001 |
| 1 nsec | 3E8 | | | | | | | 0011 1110 1000 | |
| 1 usec | F4240 | | | | | | 1111 0100 0010 0100 0000 | | |
| 1 msec | 3B9ACA00 | | | 0011 1011 1001 1010 1100 1010 0000 0000 | | | | | |
| 1 sec | E8D4A51000 | 1110 1000 1101 0100 1010 0101 0001 0000 0000 0000 | | | | | | | |

## 4.6 Limitations with this encoding method

If we follow the specification in [TRAM-TCPM], the size of one of these time-interval fields is limited to this 11-bit value and five-bit scale, so that they fit into a 16-bit space. With that limitation, the maximum value that could be stored in 16 bits is:

| 11-bit value | Scale |
|---|---|
| 1111 1111 111 | 1 1111 |

or an encoded value of 3FF and a scale value of 31. This value corresponds to any time differential between:

```
              |<Count of zeroes is the Scale value>|
11 1111 1111 1000 0000 0000 0000 0000 0000 0000 0000  (binary)
3  F    F    8    0    0    0    0    0    0    0     (hexadecimal)
```

and

```
11 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111  (binary)
3  F    F    F    F    F    F    F    F    F    F     (hexadecimal)
```

This time value, 3FFFFFFFFF, converts to 50 days, 21 hours, 40 minutes and 46.511103 seconds. A time differential 1 microsecond

longer won't fit into 16 bits using this encoding method.

## 4.7 Lack of precision induced by timer value truncation

When the bit values following the first 11 significant bits are truncated, obviously loss of precision in the value. The range of values that will be truncated to the same encoded value is 2**(Scale)-1 microseconds.

The smallest time differential value that will be truncated is

1000 0000 0000 = 2.048 msec

The value

1000 0000 0001 = 2.049 msec

will be truncated to the same encoded value, which is 400 in hex, with a scale value of 1. With the scale value of 1, the value range is calculated as 2**1 - 1, or 1 usec, which you can see is the difference between these minimum and maximum values.

With that in mind, let's look at that table of delta time values again, where the Precision is the range from the smallest value corresponding to this encoded value to the largest:

| Delta time | Time value in microseconds | Encoded value | Scale | Precision |
|---|---|---|---|---|
| 1 microsecond | 1 | 1 | 0 | 0:00.000000 |
| 1 millisecond | 38E | 38E | 0 | 0:00.000000 |
| 1 second | F4240 | 7A1 | 9 | 0:00.000511 |
| 1 minute | 3938700 | 727 | 15 | 0:00.032767 |
| 1 hour | D693A400 | 6B4 | 21 | 0:02.097151 |
| 1 day | 141DD76000 | 507 | 26 | 1:07.108863 |
| Maximum value | 3FFFFFFFFF | 7FF | 31 | 35:47.483647 |

So, when measuring the delay between transmission of two packets, or between the reception of two packets, any delay shorter than 50 days 21 hours and change can be stored in this encoded fashion within 16 bits. When you encode, for example, a DTN response time delay of 50 days, 21 hours and 40 minutes, you can be assured of accuracy within 35 minutes.

## 5 PDM Flow - Simple Client Server

Following is a sample simple flow for the PDM with one packet sent from Host A and one packet received by Host B. The PDM does not require time synchronization between Host A and Host B. The calculations to derive meaningful metrics for network diagnostics are shown below each packet sent or received.

Each packet, in addition to the PDM contains information on the sender and receiver. As discussed before, a 5-tuple consists of:

SADDR : IP address of the sender
SPORT : Port for sender
DADDR : IP address of the destination
DPORT : Port for destination
PROTC : Protocol for upper layer (ex. TCP, UDP, ICMP)

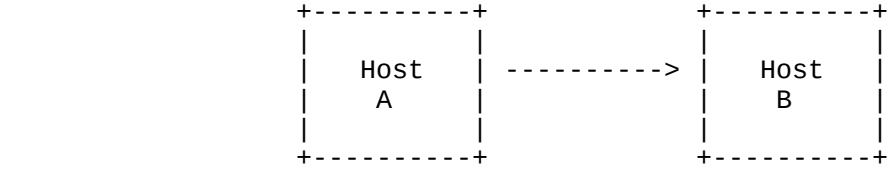
It should be understood that the packet identification information is in each packet. We will not repeat that in each of the following steps.

### 5.1 Step 1

Packet 1 is sent from Host A to Host B. The time for Host A is set initially to 10:00AM.

The time and packet sequence number are saved by the sender internally. The packet sequence number and delta times are sent in the packet.

Packet 1




PDM Contents:

```
PSNTP    : Packet Sequence Number This Packet:     25
PSNLR    : Packet Sequence Number Last Received:   -
DELTALR  : Delta Last Received:                    -
SCALEDL  : Scale of Delta LR:                      0
DELTALS  : Delta Last Sent:                        -
SCALEDS  : Scale of Delta LS:                      0
TIMEBASE : Granularity of Time:                   00 (Picoseconds)
```

Internally, within the sender, Host A, it must keep:

Packet Sequence Number of the last packet sent: 25
Time the last packet was sent: 10:00:00

Note, the initial PSNTP from Host A starts at a random number. In this case, 25. The timestamp is in seconds for the sake of simplicity.

## 5.2 Step 2

Packet 1 is received at Host B. Its time is set to one hour later than Host A. In this case, 11:00AM

Internally, within the receiver, Host B, it must note:

Packet Sequence Number of the last packet received: 25
Time the last packet was received : 11:00:03

Note, this timestamp is in Host B time. It has nothing whatsoever to do with Host A time. The Packet Sequence Number of the last packet received will become PSNLR which will be sent out in the packet sent by Host B in the next step. The time last received will be used to calculate the DELTALR value to be sent out in the packet sent by Host B in the next step.

## 5.3 Step 3

Packet 2 is sent by Host B to Host A. Note, the initial packet sequence number (PSNTP) from Host B starts at a random number. In this case, 12. Before sending the packet, Host B does a calculation of deltas. Since Host B knows when it is sending the packet, and it knows when it received the previous packet, it can do the following calculation:

Sending time (packet 2) - receive time (packet 1)

We will call the result of this calculation: Delta Last Received

That is:

DELTALR = Sending time (packet 2) - receive time (packet 1)

Note, both sending time and receive time are saved internally in Host B. They do not travel in the packet. Only the Delta is in the packet.

Assume that within Host B is the following:

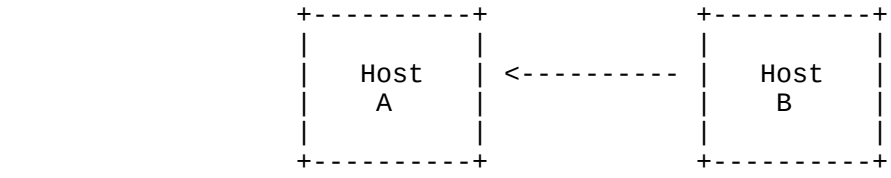
```
Packet Sequence Number of the last packet received:     25
Time the last packet was received:                      11:00:03
Packet Sequence Number of this packet:                  12
Time this packet is being sent:                         11:00:07
```

We can now calculate a delta value to be sent out in the packet. DELTALR becomes:

4 seconds = 11:00:07 - 11:00:03

This is the derived metric: Server Delay. The time and scaling factor must be calculated. Then, this value, along with the packet sequence numbers will be sent to Host A as follows:

Packet 2




PDM Contents:

```
PSNTP    : Packet Sequence Number This Packet:    12
PSNLR    : Packet Sequence Number Last Received:  25
DELTALR  : Delta Last Received:                 3A35 (4 seconds)
SCALEDL  : Scale of Delta LR:                     25
DELTALS  : Delta Last Sent:                        -
SCALEDS  : Scale of Delta LS:                      0
TIMEBASE : Granularity of Time:                   00 (Picoseconds)
```

The metric left to be calculated is the Round-Trip Delay. This will be calculated by Host A when it receives Packet 2.

## 5.4 Step 4

Packet 2 is received at Host A. Remember, its time is set to one hour earlier than Host B. Internally, it must note:

```
Packet Sequence Number of the last packet received:    12
Time the last packet was received                :     10:00:12
```

Note, this timestamp is in Host A time. It has nothing whatsoever to do with Host B time.

So, now, Host A can calculate total end-to-end time. That is:

End-to-End Time = Time Last Received - Time Last Sent

For example, packet 25 was sent by Host A at 10:00:00. Packet 12 was received by Host A at 10:00:12 so:

End-to-End time = 10:00:12 - 10:00:00 or 12 (Server and Network RT delay combined)

This derived metric we will call DELTALS or Delta Last Sent.

We can now also calculate round trip delay. The formula is:

Round trip delay = DELTALS - DELTALR

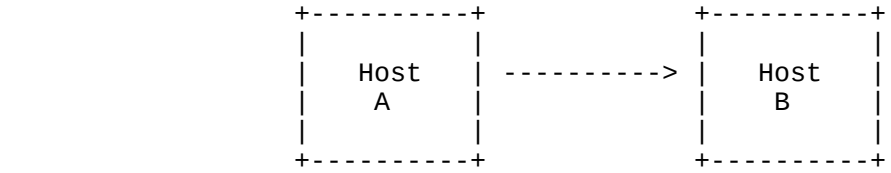
Or:

Round trip delay = 12 - 4 or 8

Now, the only problem is that at this point all metrics are in Host A only and not exposed in a packet. To do that, we need a third packet.

## 5.5 Step 5

Packet 3 is sent from Host A to Host B.




PDM Contents:

```
PSNTP    : Packet Sequence Number This Packet:    26
PSNLR    : Packet Sequence Number Last Received:  12
DELTALR  : Delta Last Received:                    0
SCALEDL  : Scale of Delta LR                       0
DELTALS  : Delta Last Sent:                    105e (12 seconds)
SCALEDL  : Scale of Delta LR                      26
TIMEBASE : Granularity of Time:                   00 (Picoseconds)
```

To calculate Two-Way Delay, any packet capture device may look at these packets and do what is necessary.

# 6 Other Flows

What we have discussed so far is a simple flow with one packet sent and one returned. Let's look at how PDM may be useful in other types of flows.

## 6.1 PDM Flow - One Way Traffic

The flow on a particular session may not be a send-receive paradigm. Let us consider some other situations. In the case of a one-way flow, one might see the following:

| Packet | Sender | PSN This Packet | PSN Last Recvd | Delta Last Recvd | Delta Last Sent |
|---|---|---|---|---|---|
| 1 | Server | 1 | 0 | 0 | 0 |
| 2 | Server | 2 | 0 | 0 | 5 |
| 3 | Server | 3 | 0 | 0 | 12 |
| 4 | Server | 4 | 0 | 0 | 20 |

What does this mean and how is it useful?

In a one-way flow, only the Delta Last Sent will be seen as used. Recall, Delta Last Sent is the difference between the send of one packet from a device and the next. This is a measure of throughput for the sender - according to the sender's point of view. That is, it is a measure of how fast is the application itself (with stack time included) able to send packets.

How might this be useful? If one is having a performance issue at the client and sees that packet 2, for example, is sent after 5 microseconds from the server but takes 3 minutes to arrive at the destination, then one may safely conclude that there are delays in the path other than at the server which may be causing the delivery issue of that packet. Such delays may include the network links, middle-boxes, etc.

Now, true one-way traffic is quite rare. What people often mean by "one-way" traffic is an application such as FTP where a group of packets (for example, a TCP window size worth) is sent, then the sender waits for acknowledgment. This type of flow would actually fall into the "multiple-send" traffic model.

### 6.2 PDM Flow - Multiple Send Traffic

Assume that two packets are sent with each send from the server.

| Packet | Sender | PSN This Packet | PSN Last Recvd | Delta Last Recvd | Delta Last Sent |
|---|---|---|---|---|---|
| 1 | Server | 1 | 0 | 0 | 0 |
| 2 | Server | 2 | 0 | 0 | 5 |
| 3 | Client | 1 | 1 | 20 | 0 |
| 4 | Server | 3 | 1 | 10 | 15 |

How might this be used?

Notice that in packet 3, the client has a value of Delta Last received of 20. Recall that Delta Last Received is the Send time of packet 3 - receive time of packet 2. So, what does one know now? In this case, Delta Last Received is the processing time for the Client to send the next packet.

How to interpret this depends on what is actually being sent. Remember, PDM is not being used in isolation, but to supplement the fields found in other headers. Let’s take some examples:

1. Client is sending a standalone TCP ACK. One would find this by looking at the payload length in the IPv6 header and the TCP Acknowledgement field in the TCP header. So, in this case, the client is taking 20 units to send back the ACK. This may or may not be interesting.

2. Client is sending data with the packet. Again, one would find this by looking at the payload length in the IPv6 header and the TCP Acknowledgement field in the TCP header. So, in this case, the client is taking 20 units to send back data. This may represent "User Think Time". Again, this may or may not be interesting, in isolation. But, if there is a performance problem receiving data at the server, then taken in conjunction with RTT or other packet timing information, this information may be quite interesting.

Of course, one also needs to look at the PSN Last Received field to make sure of the interpretation of this data. That is, to make sure that the Delta Last Received corresponds to the packet of interest.

The benefits of PDM are that we have such information available in a uniform manner for all applications and all protocols without extensive changes required to applications.

### 6.3 PDM Flow - Multiple Send with Errors

One might wonder if all of the functions of PDM might be better suited to TCP or a TCP option. Let us take the case of how PDM may help in a case of TCP retransmissions in a way that TCP options or TCP ACK / SEQ would not.

Assume that three packets are sent with each send from the server.

From the server, this is what is seen.

| Pkt | Sender | PSN This Pkt | PSN LastRecvd | Delta LastRecvd | Delta LastSent | TCP SEQ | Data Bytes |
|---|---|---|---|---|---|---|---|
| 1 | Server | 1 | 0 | 0 | 0 | 123 | 100 |
| 2 | Server | 2 | 0 | 0 | 5 | 223 | 100 |
| 3 | Server | 3 | 0 | 0 | 5 | 333 | 100 |

The client however, does not get all the packets. From the client, this is what is seen for the packets sent from the server.

| Pkt | Sender | PSN This Pkt | PSN LastRecvd | Delta LastRecvd | Delta LastSent | TCP SEQ | Data Bytes |
|---|---|---|---|---|---|---|---|
| 1 | Server | 1 | 0 | 0 | 0 | 123 | 100 |
| 2 | Server | 3 | 0 | 0 | 5 | 333 | 100 |

Let’s assume that the server now retransmits the packet. (Obviously, a duplicate acknowledgment sequence for fast retransmit or a retransmit timeout would occur. To illustrate the point, these packets are being left out.)

So, then if a TCP retransmission is done, then from the client, this is what is seen for the packets sent from the server.

| Pkt | Sender | PSN This Pkt | PSN LastRecvd | Delta LastRecvd | Delta LastSent | TCP SEQ | Data Bytes |
|---|---|---|---|---|---|---|---|
| 1 | Server | 4 | 0 | 0 | 30 | 223 | 100 |

The server has resent the old packet 2 with TCP sequence number of 223. The retransmitted packet now has a PSN This Packet value of 4. The Delta Last Sent is 30 - the time between sending the packet with PSN of 3 and this current packet.

Let’s say that packet 4 STILL does not make it. Then, after some amount of time (RTO) then the packet with TCP sequence number of 223

is resent.

From the client, this is what is seen for the packets sent from the server.

| Pkt | Sender | PSN This Pkt | PSN LastRecvd | Delta LastRecvd | Delta LastSent | TCP SEQ | Data Bytes |
|---|---|---|---|---|---|---|---|
| 1 | Server | 5 | 0 | 0 | 60 | 223 | 100 |

If now, this packet makes it, one has a very good idea that packets exist which are being sent from the server as retransmissions and not making it to the client. This is because the PSN of the resent packet from the server is 5 rather than 4. If we had used TCP sequence number alone, we would never have seen this situation. Because the TCP sequence number in all situations is 223.

This situation would be experienced by the user of the application (the human being actually sitting somewhere) as a "hangs" or long delay between packets. On large networks, to diagnose problems such as these where packets are lost somewhere on the network, one has to take multiple traces to find out exactly where.

The first thing is to start with doing a trace at the client and the server. So, we can see if the server sent a particular packet and the client received it. If the client did not receive it, then we start tracking back to trace points at the router right after the server and the router right before the client. Did they get these packets which the server has sent? This is a time consuming activity.

With PDM, we can speed up the diagnostic time because we may be able to use only the trace taken at the client to see what the server is sending.

## 7 Potential Overhead Considerations

Questions have been posed as to the potential overhead of PDM. First, PDM is entirely optional. That is, a site may choose to implement PDM or not as they wish. If they are happy with the costs of PDM vs. the benefits, then the choice should be theirs.

Below is a table outlining the potential overhead in terms of additional time to deliver the response to the end user.

| Packet Bytes | RTT | BPM | PDM Bytes | New RTT | Overhead |
|---|---|---|---|---|---|
| 1000 | 1000 milli | 1 | 16 | 1016.000 | 16.000 milli |
| 1000 | 100 milli | 10 | 16 | 101.600 | 1.600 milli |
| 1000 | 10 milli | 100 | 16 | 10.160 | .160 milli |
| 1000 | 1 milli | 1000 | 16 | 1.016 | .016 milli |

Below are some examples of actual RTTs for packets traversing large enterprise networks. The first example is for packets going to multiple business partners.

| Packet Bytes | RTT | BPM | PDM Bytes | New RTT | Overhead |
|---|---|---|---|---|---|
| 1000 | 17 milli | 58 | 16 | 17.360 | .360 milli |

The second example is for packets at a large enterprise customer within a data center. Notice that the scale is now in microseconds rather than milliseconds.

| Packet Bytes | RTT | BPM | PDM Bytes | New RTT | Overhead |
|---|---|---|---|---|---|
| 1000 | 20 micro | 50 | 16 | 20.320 | 320 micro |

## 8 Security Considerations

TBD.

## 9 IANA Considerations

Option Type TBD = 0xXX (TBD) [To be assigned by IANA] [RFC2780].

## 10 References

### 10.1 Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2681] Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, September 1999.

[RFC2780] Bradner, S. and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", BCP 37, RFC 2780, March 2000.

[IBM-POPS] IBM Corporation, "IBM z/Architecture Principles of Operation", SA22-7832, 1990-2012

## 10.2 Informative References

[ELK-PDM] Elkins, N., "draft-elkins-6man-ipv6-pdm-dest-option-09", Internet Draft, October 2014. [Work in Progress]

[TRAM-TCPM] Trammel, B., "Encoding of Time Intervals for the TCP Timestamp Option-01", Internet Draft, July 2013. [Work in Progress]

## 11 Acknowledgments

The authors would like to thank Keven Haining, Al Morton, Brian Trammel, David Boyes, and Rick Troth for their comments and assistance.

## Authors' Addresses

Nalini Elkins
Inside Products, Inc.
36A Upper Circle
Carmel Valley, CA 93924
United States
Phone: +1 831 659 8360
Email: nalini.elkins@insidethestack.com
http://www.insidethestack.com

Robert Hamilton
Chemical Abstracts Service
A Division of the American Chemical Society
2540 Olentangy River Road
Columbus, Ohio 43202
United States
Phone: +1 614 447 3600 x2517
Email: rhamilton@cas.org
http://www.cas.org

Michael S. Ackermann
Blue Cross Blue Shield of Michigan
P.O. Box 2888
Detroit, Michigan 48231
United States
Phone: +1 310 460 4080
Email: mackermann@bcbsmi.com
http://www.bcbsmi.com

Network Working Group
Internet Draft
Intended status: Informational
Expires: June 2015

T. Mizrahi
Marvell

December 17, 2014

# UDP Checksum Trailer in OWAMP and TWAMP
# draft-ietf-ippm-checksum-trailer-00.txt

## Abstract

The One-Way Active Measurement Protocol (OWAMP) and the Two-Way Active Measurement Protocol (TWAMP) are used for performance monitoring in IP networks. Delay measurement is performed in these protocols by using timestamped test packets. Some implementations use hardware-based timestamping engines that integrate the accurate transmission timestamp into every outgoing OWAMP/TWAMP test packet during transmission. Since these packets are transported over UDP, the UDP checksum field is then updated to reflect this modification. This document proposes to use the last 2 octets of every test packet as a Checksum Trailer, allowing timestamping engines to reflect the checksum modification in the last 2 octets rather than in the UDP checksum field. The behavior defined in this document is completely interoperable with existing OWAMP/TWAMP implementations.

## Status of this Memo

Table of Contents

1. Introduction ..... 2
2. Conventions used in this document ..... 4
   2.1. Terminology ..... 4
   2.2. Abbreviations ..... 4
3. Using the UDP Checksum Trailer in OWAMP and TWAMP ..... 5
   3.1. Overview ..... 5
   3.2. OWAMP / TWAMP Test Packets with Checksum Trailer ..... 5
      3.2.1. Transmission of OWAMP/TWAMP with Checksum Trailer .. 8
      3.2.2. Intermediate Updates of OWAMP/TWAMP with Checksum Trailer ..... 9
      3.2.3. Reception of OWAMP/TWAMP with Checksum Trailer ..... 9
   3.3. Interoperability with Existing Implementations..... 9
   3.4. Using the Checksum Trailer with or without Authentication 9
4. Security Considerations ..... 10
5. IANA Considerations ..... 10
6. Acknowledgments ..... 10
7. References ..... 10
   7.1. Normative References ..... 10
   7.2. Informative References ..... 11

## 1. Introduction

The One-Way Active Measurement Protocol ([OWAMP]) and the Two-Way Active Measurement Protocol ([TWAMP]) are used for performance monitoring in IP networks.

Delay and delay variation are two of the metrics that OWAMP/TWAMP can measure. This measurement is performed using timestamped test packets.

The accuracy of delay measurements relies on the timestamping method and its implementation. In order to facilitate accurate timestamping, an implementation MAY use a hardware based timestamping engine, as shown in Figure 1. In such cases, the OWAMP/TWMAP packets are sent and received by a software layer, whereas the timestamping engine modifies every outgoing test packet by incorporating its accurate transmission time into the <Timestamp> field in the packet.

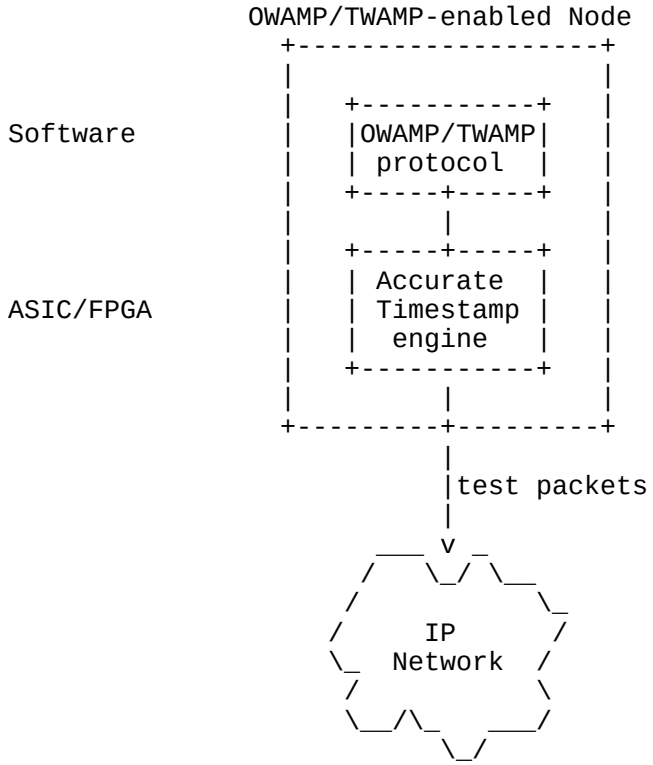



Figure 1 Accurate Timestamping in OWAMP/TWAMP

OWAMP/TWAMP test packets are transported over UDP. When the UDP payload is changed by an intermediate entity such as the timestamping engine, the UDP Checksum field must be updated to reflect the new payload. When using UDP over IPv4 ([UDP]), an intermediate entity that cannot update the value of the UDP checksum can assign a value of zero to the checksum field, causing the receiver to ignore the

checksum field. UDP over IPv6, as defined in [IPv6], does not allow a zero checksum, and requires the UDP checksum field to contain a correct checksum of the UDP payload.

Since an intermediate entity only modifies a specific field in the packet, i.e. the timestamp field, the UDP checksum update can be performed incrementally, using the concepts presented in [Checksum].

A similar problem is addressed in Annex E of [IEEE1588]. When the Precision Time Protocol (PTP) is transported over IPv6, two octets are appended to the end of the PTP payload for UDP checksum updates. The value of these two octets can be updated by an intermediate entity, causing the value of the UDP checksum field to remain correct.

This document defines a similar concept for [OWAMP] and [TWAMP], allowing intermediate entities to update OWAMP/TWAMP test packets and maintain the correctness of the UDP checksum by modifying the last 2 octets of the packet.

The term Checksum Trailer is used throughout this document and refers to the 2 octets at the end of the UDP payload, used for updating the UDP checksum by intermediate entities.

The usage of the Checksum Trailer can in some cases simplify the implementation, since if the packet data is processed in a serial order, it is simpler to first update the timestamp field, and then update the Checksum Trailer rather than to update the timestamp and then update the UDP checksum, residing at the UDP header.

## 2. Conventions used in this document

### 2.1. Terminology

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

### 2.2. Abbreviations

| | |
|---|---|
| OWAMP | One-Way Active Measurement Protocol |
| PTP | Precision Time Protocol |
| TWAMP | Two-Way Active Measurement Protocol |
| UDP | User Datagram Protocol |

## 3. Using the UDP Checksum Trailer in OWAMP and TWAMP

### 3.1. Overview

The UDP Checksum Trailer is a two-octet trailer that is piggybacked at the end of the test packet. It resides in the last 2 octets of the UDP payload.

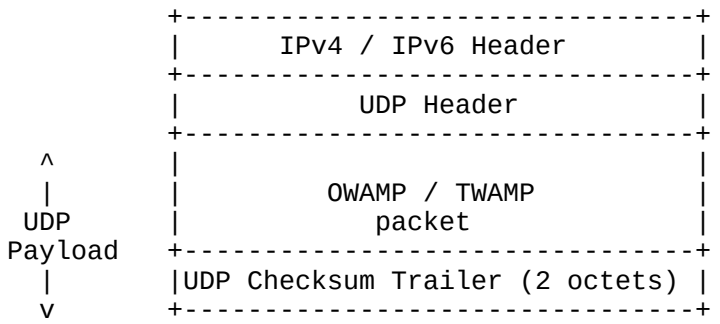



Figure 2 Checksum Trailer in OWAMP/TWAMP Test Packet

### 3.2. OWAMP / TWAMP Test Packets with Checksum Trailer

The One-Way Active Measurement Protocol [OWAMP], and the Two-Way Active Measurement Protocol [TWAMP] both make use of timestamped test packets. The formats of these packets are defined in [OWAMP] and in [TWAMP].

OWAMP/TWAMP test packets are transported over UDP, either over IPv4 or over IPv6. This document applies to both OWAMP/TWAMP over IPv4 and over IPv6.

OWAMP/TWAMP test packets contain a Packet Padding field. This document proposes to use the last 2 octets of the Packet Padding field as the Checksum Trailer. In this case the Checksum Trailer is always the last 2 octets of the UDP payload, and thus the trailer is located UDP Length - 2 octets after the beginning of the UDP header.

Figure 3 illustrates the OWAMP test packet format including the UDP checksum trailer.

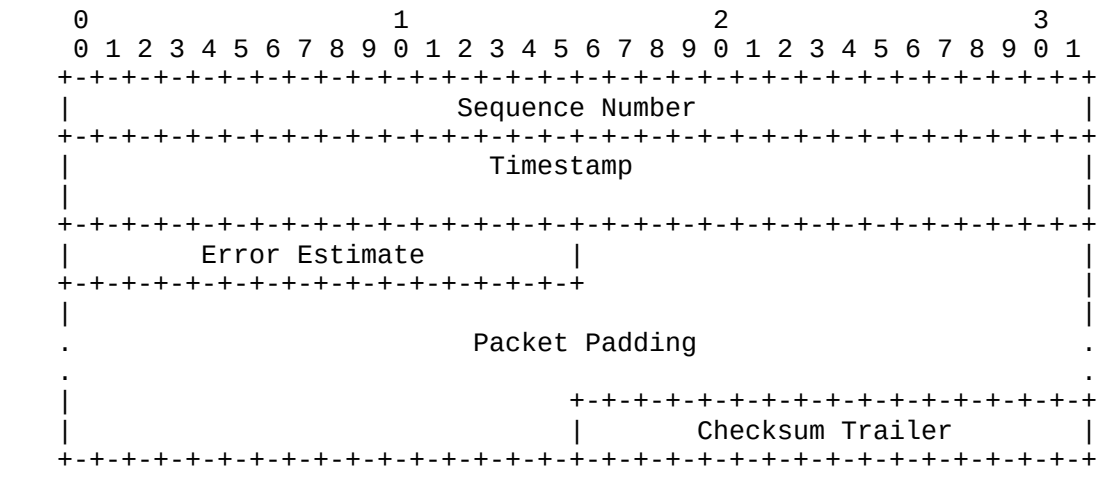



Figure 3 Checksum Trailer in OWAMP Test Packets

Figure 4 illustrates the TWAMP test packet format including the UDP checksum trailer.

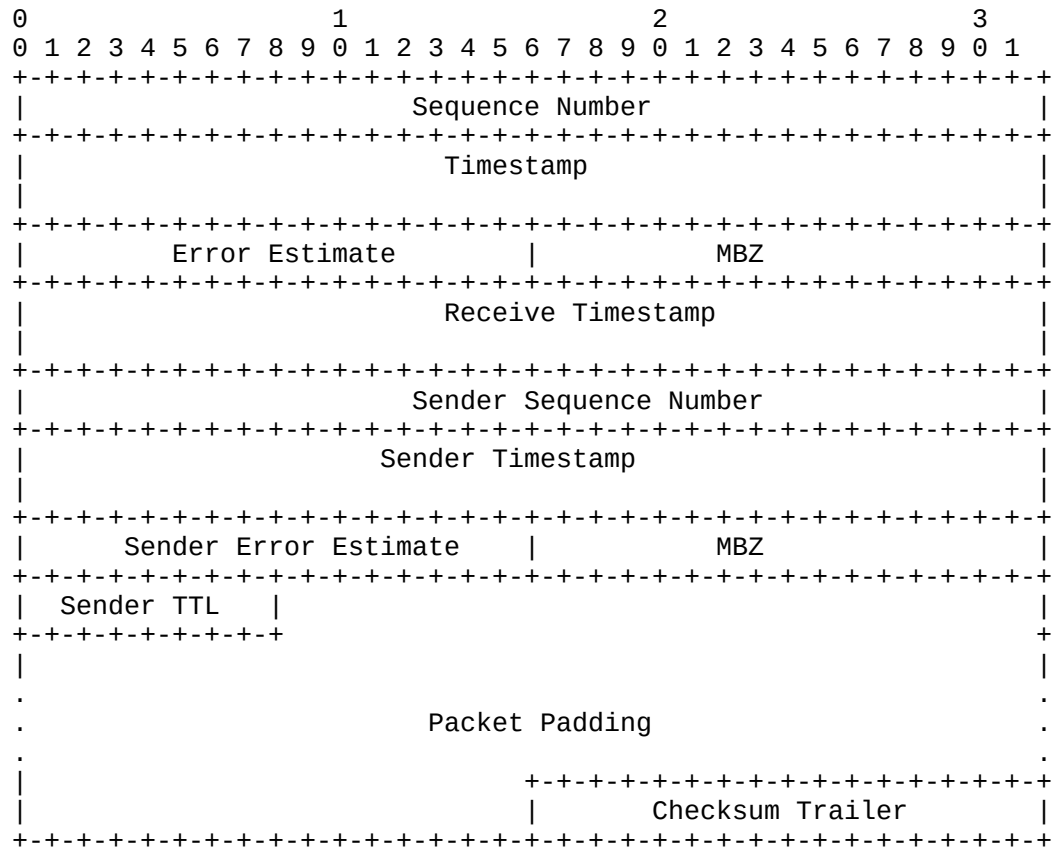



Figure 4 Checksum Trailer in TWAMP Test Packets

The length of the Packet Padding field in test packets is announced during the session initiation through the <Padding Length> field in the Request-Session message [OWAMP], or in the Request-TW-Session [TWAMP].

When a Checksum Trailer is included, the <Padding Length> MUST be sufficiently long to include the Checksum Trailer:

- In OWAMP the padding length is at least 2 octets, allowing the sender to incorporate the checksum trailer in the last 2 octets of the padding.

- In TWAMP the padding length is at least 29 octets. The additional padding is required since the header of reflector test packets is 27 octets longer than the header of sender test packets. Thus, the padding in reflector test packets is 27 octets shorter than in sender packet. Using 29 octets of padding in sender test packets allows both the sender and the reflector to use a 2-octet checksum trailer.
  Note: the 27-octet difference between the sender packet and the reflector packet is specifically in unauthenticated mode, whereas in authenticated mode the difference between the sender and receiver packets is 56 octets. As specified in Section 3.4. , the checksum trailer should only be used in unauthenticated mode.

- Two optional TWAMP features are defined in [RFC6038]: octet reflection and symmetrical size. When at least one of these features is enabled, the Request-TW-Session includes the <Padding Length> field, as well as a <Length of padding to reflect> field. In this case both fields must be sufficiently long to allow at least 2 octets of padding in both sender test packets and reflector test packets.
  Specifically, when octet reflection is enabled, the two length fields must be defined such that the padding expands at least 2 octets beyond the end of the reflected octets.

As described in Section 1. , the extensions described in this document are implemented by two logical layers, a protocol layer and a timestamping layer. It is assumed that the two layers are synchronized about whether the usage of the Checksum Trailer is enabled or not; since both logical layers reside in the same network device, it is assumed there is no need for a protocol that synchronizes this information between the two layers. When Checksum Trailer usage is enabled, the protocol layer must take care to verify that test packets include the necessary padding, and avoiding the need for the timestamping layer to verify that en-route test packets include the necessary padding.

### 3.2.1. Transmission of OWAMP/TWAMP with Checksum Trailer

The transmitter of an OWAMP/TWAMP test packet MAY include a Checksum Trailer field, incorporated in the last 2 octets of the Packet Padding.

A transmitter that includes a Checksum Trailer in its outgoing test packets MUST include a Packet Padding in these packets, the length of which MUST be sufficient to include the checksum trailer. The length of the padding field is negotiated during session initiation, as described in Section 3.2.

#### 3.2.2. Intermediate Updates of OWAMP/TWAMP with Checksum Trailer

An intermediate entity that receives and alters an OWAMP/TWAMP test packet MAY alter the Checksum Trailer field in order to maintain the correctness of the UDP checksum value.

#### 3.2.3. Reception of OWAMP/TWAMP with Checksum Trailer

This document does not impose new requirements on the receiving end of an OWAMP/TWAMP test packet.

The UDP layer at the receiving end verifies the UDP Checksum of received test packets, and the OWAMP/TWAMP layer SHOULD treat the Checksum Trailer as part of the Packet Padding.

### 3.3. Interoperability with Existing Implementations

The behavior defined in this document does not impose new requirements on the reception behavior of an OWAMP receiver or a TWAMP reflector, since the existence of the checksum trailer is transparent from the perspective of the receiver/reflector. Thus, the functionality described in this document allows interoperability with existing implementations that comply to [OWAMP] or [TWAMP].

### 3.4. Using the Checksum Trailer with or without Authentication

Both OWAMP and TWAMP may use authentication, as defined in [OWAMP] or [TWAMP]. A Checksum Trailer SHOULD NOT be used when authentication is enabled. The Checksum Trailer is effective in unauthenticated mode, allowing the intermediate entity to perform serial processing of the packet without storing-and-forwarding it.

On the other hand, when message authentication is used, an intermediate entity that alters test packets must also re-compute the Message Authentication Code (MAC) accordingly. The MAC update typically requires the intermediate entity to store the packet, re-compute its MAC, and then forward it. Thus, the benefit of the checksum trailer is effectively irrelevant when a MAC is used.

Note: while [OWAMP] and [TWAMP] include an inherent security mechanism, these protocols can be secured by other measures, e.g., [IPPMIPsec]. For similar reasons as described above, a Checksum Trailer SHOULD NOT be used in this case.

## 4. Security Considerations

This document describes how a Checksum Trailer extension can be used for maintaining the correctness of the UDP checksum.

The purpose of this extension is to ease the implementation of accurate timestamping engines, as described in Figure 1. The extension is intended to be used internally in an OWAMP/TWAMP enabled node, and not intended to be used by intermediate switches and routers that reside between the sender and the receiver/reflector. Any modification of a test packet by intermediate switches or routers should be considered a malicious MITM attack.

It is important to emphasize that the scheme described in this document does not increase the protocol's vulnerability to MITM attacks; a MITM who maliciously modifies a packet and its checksum trailer is logically equivalent to a MITM attacker who modifies a packet and its UDP Checksum field.

The concept described in this document is intended to be used only in unauthenticated mode. As described in Section 3.4. , the benefits of the Checksum Trailer do not apply when authentication is enabled.

## 5. IANA Considerations

There are no IANA actions required by this document.

RFC Editor: please delete this section before publication.

## 6. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

## 7. References

### 7.1. Normative References

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[IPv6] Deering, S., Hinden, R., "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[Checksum] Rijsinghani, A., "Computation of the Internet Checksum via Incremental Update", RFC 1624, May 1994.

[UDP] Postel, J., "User Datagram Protocol", RFC 768, August 1980.

[OWAMP] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and Zekauskas, M., "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, September 2006.

[TWAMP] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and Babiarz, J., "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, October 2008.

[RFC6038] Morton, A., Ciavattone, L., "Two-Way Active Measurement Protocol (TWAMP) Reflect Octets and Symmetrical Size Features", RFC 6038, October 2010.

### 7.2. Informative References

[IEEE1588] IEEE TC 9 Instrumentation and Measurement Society, "1588 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Version 2", IEEE Standard, 2008.

[IPPMIPsec] Pentikousis, K., Cui, Y., Zhang, E., "Network Performance Measurement for IPsec", draft-ietf-ippm-ipsec (work in progress), September 2014.

## Authors' Addresses

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam, 20692 Israel

Email: talmi@marvell.com

Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: August 20, 2015

M. Bagnulo
UC3M
B. Claise
Cisco Systems, Inc.
P. Eardley
BT
A. Morton
AT&T Labs
A. Akhter
Cisco Systems, Inc.
February 16, 2015

# Registry for Performance Metrics
# draft-ietf-ippm-metric-registry-02

## Abstract

This document defines the IANA Registry for Performance Metrics. This document also gives a set of guidelines for Registered Performance Metric requesters and reviewers.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 20, 2015.

## Copyright Notice

Table of Contents

1. Open Issues . . . . . 3
2. Introduction . . . . . 3
3. Terminology . . . . . 4
4. Scope . . . . . 6
5. Motivation for a Performance Metrics Registry . . . . . 6
5.1. Interoperability . . . . . 7
5.2. Single point of reference for Performance Metrics . . . . 7
5.3. Side benefits . . . . . 8
6. Criteria for Performance Metrics Registration . . . . . 8
7. Performance Metric Registry: Prior attempt . . . . . 9
7.1. Why this Attempt Will Succeed . . . . . 9
8. Defintion of the Performance Metric Registry . . . . . 10
8.1. Summary Category . . . . . 11
8.1.1. Identifier . . . . . 11
8.1.2. Name . . . . . 12
8.1.3. URI . . . . . 12
8.1.4. Description . . . . . 13
8.2. Metric Definition Category . . . . . 13
8.2.1. Reference Definition . . . . . 13
8.2.2. Fixed Parameters . . . . . 13
8.3. Method of Measurement Category . . . . . 14
8.3.1. Reference Method . . . . . 14
8.3.2. Packet Generation Stream . . . . . 14
8.3.3. Traffic Filter . . . . . 15
8.3.4. Sampling distribution . . . . . 15
8.3.5. Run-time Parameters . . . . . 15
8.3.6. Role . . . . . 15
8.4. Output Category . . . . . 16
8.4.1. Value . . . . . 16
8.4.2. Reference . . . . . 16
8.4.3. Metric Units . . . . . 16
8.5. Administrative information . . . . . 16
8.5.1. Status . . . . . 16
8.5.2. Requester . . . . . 17
8.5.3. Revision . . . . . 17
8.5.4. Revision Date . . . . . 17
8.6. Comments and Remarks . . . . . 17
9. The Life-Cycle of Registered Metrics . . . . . 17
9.1. Adding new Performance Metrics to the Registry . . . . . 17
9.2. Revising Registered Performance Metrics . . . . . 18

9.3. Deprecating Registered Performance Metrics . . . . . 20
10. Security considerations . . . . . 20
11. IANA Considerations . . . . . 21
12. Acknowledgments . . . . . 21
13. References . . . . . 21
13.1. Normative References . . . . . 21
13.2. Informative References . . . . . 22
Authors' Addresses . . . . . 23

## 1. Open Issues

1. Define the Filter column subcolumns, i.e. how filters are expressed.

2. Need to include an example for a name for a passive metric

3. Shall we remove the definitions of active and passive? If we remove it, shall we keep all the related comments in the draft?

4. URL: should we include a URL link in each registry entry with a URL specific to the entry that links to a different text page that contains all the details of the registry entry as in http://www.iana.org/assignments/xml-registry/xml-registry.xhtml#ns

## 2. Introduction

The IETF specifies and uses Performance Metrics of protocols and applications transported over its protocols. Performance metrics are such an important part of the operations of IETF protocols that [RFC6390] specifies guidelines for their development.

The definition and use of Performance Metrics in the IETF happens 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 "Metric Blocks for use with RTCP's Extended Report Framework" (XRBLOCK) WG recently 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 "Benchmarking Methodology" WG (BMWG) defined many Performance Metrics for use in laboratory benchmarking of inter-networking technologies.

The "IP Flow Information eXport" (IPFIX) WG Information elements related to Performance Metrics are currently proposed.

The "Performance Metrics for Other Layers" (PMOL) 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.

However, despite the importance of Performance Metrics, there are two related problems for the industry. First, how to ensure 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, how to discover which Performance Metrics have been specified, so as to avoid developing new Performance Metric that is very similar. The problems can be addressed by creating a registry of performance metrics. The usual way in which IETF organizes namespaces is with Internet Assigned Numbers Authority (IANA) registries, and there is currently no Performance Metrics Registry maintained by the IANA.

This document therefore creates a Performance Metrics Registry. It also provides best practices on how to specify new entries or update ones in the Performance Metrics Registry.

## 3. 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 [RFC2119].

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, 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 (or Registered Metric) is a Performance Metric expressed as an entry in the Performance Metric Registry, administered by IANA. Such a performance metric has met all the registry review criteria defined in this document in order to included in the registry.

Performance Metrics Registry: The IANA registry containing Registered Performance Metrics. In this document, it is also called simply "Registry".

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 experts selected by the IESG to validate the Performance Metrics before updating the Performance Metrics Registry. The Performance Metrics Experts work closely with IANA.

Parameter: An input factor defined as a variable in the definition of a metric. 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 to measure using a metric and interpret the results. Although Parameters do not change the fundamental nature of the metric’s definition, some have substantial influence on the network property being assessed and interpretation of the results.

Consider the case of packet loss in the following two active measurement cases. The first case is packet loss as background loss where the 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 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 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. Examples of Active Measurement Methods are the the measurement methods for the One way delay metric defined in [RFC2679] 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. 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: Methods of Measurement which use a combination of Active Measurement and Passive Measurement methods.

## 4. Scope

This document is meant for two different audiences. For those defining new Registered Performance Metrics, it provides specifications and best practices to be used in deciding which Registered Metrics are useful for a measurement study, instructions for writing the text for each column of the Registered Metrics, and information on the supporting documentation required for the new Registry entry (up to and including the publication of one or more RFCs or I-Ds describing it). For the appointed Performance Metrics Experts and for IANA personnel administering the new IANA Performance Metric Registry, it defines a set of acceptance criteria against which these proposed Registry Entries should be evaluated.

This document specifies a Performance Metrics Registry in IANA. This Performance Metric Registry is applicable to Performance Metrics issued from Active Measurement, Passive Measurement, from end-point calculation or any other form of Performance Metric. This registry is designed to encompass Performance Metrics developed throughout the IETF and especially for the following existing working groups: IPPM, XRBLOCK, IPFIX, and BMWG. This document analyzes an prior attempt to set up a Performance Metric 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 Registry with initial entries. It does provides a few examples that are merely illustrations and should not be included in the registry at this point in time.

Based on [RFC5226] Section 4.3, this document is processed as Best Current Practice (BCP) [RFC2026].

## 5. Motivation for a Performance Metrics Registry

In this section, we detail several motivations for the Performance Metric Registry.

### 5.1. Interoperability

As any IETF registry, the primary use for a registry is to manage a namespace for its use within one or more protocols. In the particular case of the Performance Metric Registry, there are two types of protocols that will use the Performance Metrics in the Registry during their operation (by referring to the Index values):

- Control protocol: this type of protocols is used to allow one entity to request another entity to perform a measurement using a specific metric defined by the Registry. One particular example is the LMAP framework [I-D.ietf-lmap-framework]. Using the LMAP terminology, the Registry is used in the LMAP Control protocol to allow a Controller to request a measurement task to one or more Measurement Agents. In order to enable this use case, the entries of the Performance Metric Registry must be well enough defined to allow a Measurement Agent implementation to trigger a specific measurement task upon the reception of a control protocol message. This requirements heavily constrains the type of entries that are acceptable for the Performance Metric Registry.

- Report protocol: This type of protocols is used to allow an entity to report measurement results to another entity. By referencing to a specific Performance Metric Registry, it is possible to properly characterize the measurement result data being transferred. Using the LMAP terminology, the Registry is used in the Report protocol to allow a Measurement Agent to report measurement results to a Collector.

### 5.2. Single point of reference for Performance Metrics

A Registry for Performance Metrics 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 metrics that attempt to measure the same phenomena but in slightly different (and incompatible) ways. Having a Registry would allow both the IETF community and external people 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 metrics, where different entities that request measurements, execute measurements, and report the results can benefit from a common understanding of the referenced metric.

### 5.3. Side benefits

There are a couple of side benefits of having such a Registry. First, the Registry could serve as an inventory of useful and used metrics, that are normally supported by different implementations of measurement agents. Second, the results of measurements using the metrics would be comparable even if they are performed by different implementations and in different networks, as the 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 Registry entries are sufficiently well specified to result in comparable (or equivalent) results. Registry entries which have undergone such testing SHOULD be noted, with a reference to the test results.

## 6. Criteria for Performance Metrics Registration

It is neither possible nor desirable to populate the Registry with all combinations of input parameters of all Performance Metrics. The Registered Performance Metrics should be:

1. interpretable by the user.

2. implementable by the software designer,

3. deployable by network operators, without major impact on the networks,

4. accurate, 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 Registry entry has significant industry interest, or has seen deployment, and there is agreement that the candidate Registered Metric serves its intended purpose.

## 7. 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 might help 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 entry in the registry with only a few variable (Run-time) Parameters to be specified by the measurement designer, if any. The idea is that entries in the 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 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. Therefore this document defines that the Registry only includes metrics that are well defined and that have proven to be operationally useful. In order to assure these two characteristics, a set of experts are required to review the allocation request to verify that the metric is well defined and it is operationally useful.

### 7.1. Why this Attempt Will Succeed

The Registry defined in this document addresses the main issues identified in the previous attempt. As we mention 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. In this Registry, the Registry requests are evaluated by an expert

group, the Performance Metrics Experts, who will make sure that the metric is properly defined. This document provides guidelines to assess if a metric is properly defined.

Another key difference between this attempt and the previous one is that in this case there is at least one clear user for the Registry: the LMAP framework and protocol. Because the LMAP protocol will use the 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 Metric Registry value in the protocol, a metric is properly specified if it is defined well-enough so that it is possible (and practical) to implement the metric 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.

## 8. Defintion of the Performance Metric Registry

In this section we define the columns of the Performance Metric Registry. This registry will contain all Registered Performance Metrics including active, passive, hybrid, endpoint metrics and any other type of performance metric that can be envisioned. Because of that, it may be the case that some of the columns defined are not applicable for a given type of metric. If this is the 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 and Reference Specification. Moreover, In addition, it may be possible that in the future, a new type of metric requires additional columns. Should that be the case, it is possible to add new columns to the registry. The specification defining the new column(s) must define how to populate the new column(s) for existing entries.

The columns of the Performance Metric Registry are defined next. The columns are grouped into "Categories" to facilitate the use of the registry. Categories are described at the 8.x heading level, and columns are at the 8.x.y heading level. The Figure below illustrates this organization. An entry (row) therefore gives a complete description of a Registered Metric.

Each column serves as a check-list item and helps to avoid omissions during registration and expert review.

Registry Categories and Columns, shown as

```
                                                    Category
                                                    ------------------
                                                    Column |  Column |

        Summary
        -------------------------------
        ID | Name | URI | Description |


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


        Method of Measurement
        ------------------------------------------------------------------
        Reference | Packet     | Traffic | Sampling | Run-time | Role |
        Method    | Generation | Filter  | dist.    | Param    |      |
                  | Stream     |
        Output
        -----------------------------
        | Type | Reference  | Units |
        |      | Definition |       |


        Administrative information
        ---------------------------------
        Status |Request | Rev | Rev.Date |


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

## 8.1. Summary Category

### 8.1.1. Identifier

A numeric identifier for the Registered Performance Metric. This identifier MUST be unique within the Performance Metric Registry.

The Registered Performance Metric unique identifier is a 16-bit integer (range 0 to 65535). When adding newly Registered Performance Metrics to the Performance Metric Registry, IANA should assign the lowest available identifier to the next Registered Performance Metric.

#### 8.1.2. Name

As the name of a Registered Performance Metric is the first thing a potential implementor will use when determining whether it is suitable for a given application, it is important to be as precise and descriptive as possible.

New names of Registered Performance Metrics:

1. "MUST be chosen carefully to describe the Registered Performance Metric and the context in which it will be used."

2. "MUST be unique within the Performance Metric Registry."

3. "MUST use capital letters for the first letter of each component . All other letters MUST be lowercase, even for acronyms. Exceptions are made for acronyms containing a mixture of lowercase and capital letters, such as 'IPv4' and 'IPv6'."

4. MUST use '_' between each component of the Registered Performance Metric name.

5. MUST start with prefix Act_ for active measurement Registered Performance Metric.

6. MUST start with prefix Pas_ for passive monitoring Registered Performance Metric.

7. Other types of metrics should define a proper prefix for identifying the type.

8. The remaining rules for naming are left for the Performance Experts to determine as they gather experience, so this is an area of planned update by a future RFC.

An example is "Act_UDP_Latency_Poisson_99mean" for a active monitoring UDP latency metric using a Poisson stream of packets and producing the 99th percentile mean as output.

#### 8.1.3. URI

The URI column MUST contain a URI [RFC 3986] that uniquely identified the metric. The URI is a URN [RFC 2141]. The URI is automatically generated by prepending the prefix urn:ietf:params:ippm:metric: to the metric name. The resulting URI is globally unique.

#### 8.1.4. Description

A Registered Performance Metric Description is a written representation of a particular Registry entry. It supplements the metric name to help Registry users select relevant Registered Performance Metrics.

### 8.2. Metric Definition Category

This category includes columns to prompt all necessary details related to the metric definition, including the RFC reference and values of input factors, called fixed parameters, which are left open in the RFC but have a particular value defined by the performance metric.

#### 8.2.1. 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.

#### 8.2.2. Fixed Parameters

Fixed Parameters are input factors whose value must be specified in the 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. For 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

A Parameter which is a Fixed Parameter for one Registry entry may be designated as a Run-time Parameter for another Registry entry.

### 8.3. Method of Measurement Category

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

#### 8.3.1. Reference Method

This entry provides references to relevant sections of the RFC(s) describing the method of measurement, as well as any supplemental information needed to ensure unambiguous interpretation for implementations referring to the RFC text.

Specifically, this section should include pointers to pseudocode or actual code that could be used for an unambigious implementation.

#### 8.3.2. Packet Generation Stream

This column applies to metrics that generate traffic for a part of their Measurement Method purposes including but not necessarily limited to Active metrics. The generated traffic is referred as stream and this columns describe its characteristics.

Each entry for this column contains the following information:

- Value: The name of the packet stream scheduling discipline
- Stream Parameters: The values and formats of input factors for each type of stream. For example, the average packet rate and distribution truncation value for streams with Poisson-distributed inter-packet sending times.
- Reference: the specification where the stream is defined

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 values is specified in this column.

#### 8.3.3. Traffic Filter

This column applies to 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.

#### 8.3.4. 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

Parameters: if any.

Reference definition: pointer to the specification where the sampling distribution is properly defined.

#### 8.3.5. Run-time Parameters

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. However, the values of these parameters is not specified in the Registry, rather these parameters are listed as an aid to the measurement system implementor 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.

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.

#### 8.3.6. Role

In some method of measurements, there may be several roles defined e.g. on a one-way packet delay active measurement, there is one measurement agent that generates the packets and the other one that receives the packets. This column contains the name of the role for this particular entry. In the previous example, there should be two

entries int he registry, one for each role, so that when a measurement agent is instructed to perform the one way delay source metric know that it is supposed to generate packets. The values for this field are defined in the reference method of measurement.

### 8.4. 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.

#### 8.4.1. Value

This column contain the name of the output type. The output type defines the type of result that the metric produces. It can be the raw results or it can be some form of 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").

#### 8.4.2. Reference

This column contains a pointer to the specification where the output type is defined

#### 8.4.3. 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 [RFC2330] for definitions of these terms) is collected, this entry will specify the units for each measured value.

### 8.5. Administrative information

#### 8.5.1. 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.

#### 8.5.2. Requester

The requester for the Registered Performance Metric. The requester MAY be a document, such as RFC, or person.

#### 8.5.3. 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.

#### 8.5.4. Revision Date

The date of acceptance or the most recent revision for the Registered Performance Metric.

### 8.6. 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.

## 9. The Life-Cycle of Registered Metrics

Once a Performance Metric or set of Performance Metrics has been identified for a given application, candidate Registry entry specifications in accordance with Section 8 are 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 Metric Registry, such as deprecation or revision, as described later in this section.

It is also desirable that the author(s) of a candidate Registry entry seek review in the relevant IETF working group, or offer the opportunity for review on the WG mailing list.

### 9.1. Adding new Performance Metrics to the Registry

Requests to change Registered Metrics in the Performance Metric Registry are 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 Expert Review RFC5226 policy defined for the Performance Metric 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.

Authors are expected to review compliance with the specifications in this document to check their submissions before sending them to IANA.

The Performance Metric Experts should endeavor to complete referred reviews in a timely manner. If the request is acceptable, the Performance Metric Experts signify their approval to IANA, which updates the Performance Metric Registry. If the request is not acceptable, the Performance Metric Experts can coordinate with the requester to change the request to be compliant. The Performance Metric Experts may also choose in exceptional circumstances to reject clearly frivolous or inappropriate change requests outright.

This process should not in any way be construed as allowing the Performance Metric Experts to overrule IETF consensus. Specifically, any Registered Metrics that were added with IETF consensus require IETF consensus for revision or deprecation.

Decisions by the Performance Metric Experts may be appealed as in Section 7 of RFC5226.

### 9.2. Revising Registered Performance Metrics

A request for Revision is only permissible when the changes maintain backward-compatibility with implementations of the prior Registry entry describing a Registered Metric (entries with lower revision numbers, but the same Identifier and Name).

The purpose of the Status field in the Performance Metric Registry is to indicate whether the entry for a Registered Metric is ’current’ or ’deprecated’.

In addition, no policy is defined for revising IANA Performance Metric entries or addressing errors therein. To be certain, changes and deprecations within the Performance Metric 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 X, identifying the existing Registry entry.

The primary requirement in the definition of a policy for managing changes to existing Registered Performance Metrics is avoidance of interoperability problems; Performance Metric Experts must work to maintain interoperability above all else. Changes to Registered Performance Metrics may only be done in an inter-operable 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 Metric and possibly the deprecation of the earlier metric.

A change to a Registered Performance Metric is held to be backward-compatible only 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 an Performance Metric revision is deemed permissible by the Performance Metric Experts, according to the rules in this document, IANA makes the change in the Performance Metric Registry. The requester of the change is appended to the requester in the Registry.

Each Registered Performance Metric in the Registry has a revision number, starting at zero. Each change to a Registered Performance Metric following this process increments the revision number by one.

COMMENT: Al (and Phil) think we should keep old/revised entries as-is, marked as deprecated >>>> Since any revision must be inter-operable according to the criteria above, there is no need for the Performance Metric Registry to store information about old revisions.

When a revised Registered Performance Metric is accepted into the Performance Metric 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 Registry entries 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 is changed to "Deprecated".

### 9.3. Deprecating Registered Performance Metrics

Changes that are not permissible by the above criteria for Registered 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 Revising Registered Performance Metrics; or"

2. "the deprecation harmonizes with an external reference that was itself deprecated through that reference's accepted deprecation method; or"

A request for deprecation is sent to IANA, which passes it to the Performance Metric Expert for review, as in Section 'The Process for Review by the Performance Metric Experts'. When deprecating an Performance Metric, the Performance Metric description in the Performance Metric 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 use of deprecated Registered Metrics should result in a log entry or human-readable warning by the respective application.

Names and Metric ID of deprecated Registered Metrics must not be reused.

Deprecated metric entries are kept in the registry for archival purposes. The deprecated entries are kept with all fields unmodified (version, revision date) except for the status field that is changed to "Deprecated".

## 10. Security considerations

This draft doesn't introduce any new security considerations for the Internet. However, the definition of Performance Metrics may introduce some security concerns, and should be reviewed with security in mind.

## 11. IANA Considerations

This document specifies the procedure for Performance Metrics Registry setup. IANA is requested to create a new Registry for Performance Metrics called "Registered Performance Metrics" with the columns defined in Section 8.

New assignments for Performance Metric Registry will be administered by IANA through Expert Review [RFC5226], i.e., review by one of a group of experts, the Performance Metric Experts, appointed by the IESG upon recommendation of the Transport Area Directors. The experts can be initially drawn from the Working Group Chairs and document editors of the Performance Metrics Directorate among other sources of experts.

The Identifier values from 64512 to 65536 are reserved for private use. The name starting with the prefix Priv- are reserved for private use.

This document requests the allocation of the URI prefix urn:ietf:params:ippm:metric for the purpose of generating URIs for registered metrics.

## 12. Acknowledgments

Thanks to Brian Trammell and Bill Cerveny, IPPM chairs, for leading some brainstorming sessions on this topic.

## 13. References

### 13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

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

[RFC4148] Stephan, E., "IP Performance Metrics (IPPM) Metrics Registry", BCP 108, RFC 4148, August 2005.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC6248] Morton, A., "RFC 4148 and the IP Performance Metrics (IPPM) Registry of Metrics Are Obsolete", RFC 6248, April 2011.

[RFC6390] Clark, A. and B. Claise, "Guidelines for Considering New Performance Metric Development", BCP 170, RFC 6390, October 2011.

[RFC6576] Geib, R., Morton, A., Fardid, R., and A. Steinmitz, "IP Performance Metrics (IPPM) Standard Advancement Testing", BCP 176, RFC 6576, March 2012.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC2141] Moats, R., "URN Syntax", RFC 2141, May 1997.

## 13.2. Informative References

[RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC6035] Pendleton, A., Clark, A., Johnston, A., and H. Sinnreich, "Session Initiation Protocol Event Package for Voice Quality Reporting", RFC 6035, November 2010.

[I-D.ietf-lmap-framework] Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A framework for large-scale measurement platforms (LMAP)", draft-ietf-lmap-framework-10 (work in progress), January 2015.

[RFC5477] Dietz, T., Claise, B., Aitken, P., Dressler, F., and G. Carle, "Information Model for Packet Sampling Exports", RFC 5477, March 2009.

[RFC5102] Quittek, J., Bryant, S., Claise, B., Aitken, P., and J. Meyer, "Information Model for IP Flow Information Export", RFC 5102, January 2008.

[RFC6792] Wu, Q., Hunt, G., and P. Arden, "Guidelines for Use of the RTP Monitoring Framework", RFC 6792, November 2012.

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

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

[RFC6776] Clark, A. and Q. Wu, "Measurement Identity and Information Reporting Using a Source Description (SDES) Item and an RTCP Extended Report (XR) Block", RFC 6776, October 2012.

[RFC7003] Clark, A., Huang, R., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Burst/Gap Discard Metric Reporting", RFC 7003, September 2013.

[RFC3432] Raisanen, V., Grotefeld, G., and A. Morton, "Network performance measurement with periodic streams", RFC 3432, November 2002.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

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

[RFC2679] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Delay Metric for IPPM", RFC 2679, September 1999.

[RFC2681] Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, September 1999.

## 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
Cisco Systems, Inc.
7025 Kit Creek Road
RTP, NC 27709
USA

Email: aakhter@cisco.com

IP Performance Working Group
Internet-Draft
Intended status: Experimental
Expires: September 10, 2015

M. Mathis
Google, Inc
A. Morton
AT&T Labs

March 9, 2015

# Model Based Bulk Performance Metrics
# draft-ietf-ippm-model-based-metrics-04.txt

## Abstract

We introduce a new class of model based metrics designed to determine if an end-to-end Internet path can meet predefined bulk transport performance targets by applying a suite of IP diagnostic tests to successive subpaths. The subpath-at-a-time tests can be robustly applied to key infrastructure, such as interconnects, to accurately detect if any part of the infrastructure will prevent the full end-to-end paths traversing them from meeting the specified target performance.

The diagnostic tests consist of precomputed traffic patterns and statistical criteria for evaluating packet delivery. The traffic patterns are precomputed to mimic TCP or other transport protocol over a long path but are constructed in such a way that they are independent of the actual details of the subpath under test, end systems or applications. Likewise the success criteria depends on the packet delivery statistics of the subpath, as evaluated against a protocol model applied to the target performance. The success criteria also does not depend on the details of the subpath, endsystems or application. This makes the measurements open loop, eliminating most of the difficulties encountered by traditional bulk transport metrics.

Model based metrics exhibit several important new properties not present in other Bulk Capacity Metrics, including the ability to reason about concatenated or overlapping subpaths. The results are vantage independent which is critical for supporting independent validation of tests results from multiple Measurement Points.

This document does not define diagnostic tests directly, but provides a framework for designing suites of diagnostics tests that are tailored to confirming that infrastructure can meet the target performance.

## Status of this Memo

This Internet-Draft is submitted in full conformance with the

provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 10, 2015.

Copyright Notice

Table of Contents

1. Introduction . . . . . 5
1.1. TODO . . . . . 7
2. Terminology . . . . . 7
3. New requirements relative to RFC 2330 . . . . . 11
4. Background . . . . . 12
4.1. TCP properties . . . . . 13
4.2. Diagnostic Approach . . . . . 14
5. Common Models and Parameters . . . . . 15
5.1. Target End-to-end parameters . . . . . 16
5.2. Common Model Calculations . . . . . 16
5.3. Parameter Derating . . . . . 17
6. Common testing procedures . . . . . 18
6.1. Traffic generating techniques . . . . . 18
6.1.1. Paced transmission . . . . . 18
6.1.2. Constant window pseudo CBR . . . . . 19
6.1.3. Scanned window pseudo CBR . . . . . 19
6.1.4. Concurrent or channelized testing . . . . . 20
6.2. Interpreting the Results . . . . . 21
6.2.1. Test outcomes . . . . . 21
6.2.2. Statistical criteria for estimating run_length . . . . 22
6.2.3. Reordering Tolerance . . . . . 24
6.3. Test Preconditions . . . . . 25
7. Diagnostic Tests . . . . . 25
7.1. Basic Data Rate and Delivery Statistics Tests . . . . . 26
7.1.1. Delivery Statistics at Paced Full Data Rate . . . . . 26
7.1.2. Delivery Statistics at Full Data Windowed Rate . . . . 27
7.1.3. Background Delivery Statistics Tests . . . . . 27
7.2. Standing Queue Tests . . . . . 27
7.2.1. Congestion Avoidance . . . . . 29
7.2.2. Bufferbloat . . . . . 29
7.2.3. Non excessive loss . . . . . 30
7.2.4. Duplex Self Interference . . . . . 30
7.3. Slowstart tests . . . . . 30
7.3.1. Full Window slowstart test . . . . . 31
7.3.2. Slowstart AQM test . . . . . 31
7.4. Sender Rate Burst tests . . . . . 31
7.5. Combined and Implicit Tests . . . . . 32
7.5.1. Sustained Bursts Test . . . . . 32
7.5.2. Streaming Media . . . . . 33
8. An Example . . . . . 34
9. Validation . . . . . 36
10. Security Considerations . . . . . 37
11. Acknowledgements . . . . . 37
12. IANA Considerations . . . . . 38
13. References . . . . . 38
13.1. Normative References . . . . . 38

13.2. Informative References . . . . . 38
Appendix A. Model Derivations . . . . . 40
A.1. Queueless Reno . . . . . 41
Appendix B. Complex Queueing . . . . . 42
Appendix C. Version Control . . . . . 43
Authors' Addresses . . . . . 43

## 1. Introduction

Bulk performance metrics evaluate an Internet path's ability to carry bulk data. Model based bulk performance metrics rely on mathematical TCP models to design a targeted diagnostic suite (TDS) of IP performance tests which can be applied independently to each subpath of the full end-to-end path. These targeted diagnostic suites allow independent tests of subpaths to accurately detect if any subpath will prevent the full end-to-end path from delivering bulk data at the specified performance target, independent of the measurement vantage points or other details of the test procedures used for each measurement.

The end-to-end target performance is determined by the needs of the user or application, outside the scope of this document. For bulk data transport, the primary 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 entire end-to-end path over which the data traverses, these parameters must also be specified in advance. They may reflect a specific real path through the Internet or an idealized path representing a typical user community. The target values for these three parameters, Data Rate, RTT and MTU, inform the mathematical models used to design the TDS.

Each IP diagnostic test in a TDS consists of a precomputed traffic pattern and statistical criteria for evaluating packet delivery.

Mathematical models are used to design traffic patterns that mimic TCP or other bulk transport protocol 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 computed in advance based on the three target parameters of the end-to-end path and independent of the properties of individual subpaths. As much as possible the measurement traffic is generated deterministically in ways that minimize the extent to which test methodology, measurement points, measurement vantage or path partitioning affect the details of the measurement traffic.

Mathematical models are also used to compute the bounds on the packet delivery statistics for acceptable IP performance. Since these statistics, such as packet loss, are typically aggregated from all subpaths of the end-to-end path, the end-to-end statistical bounds need to be apportioned as a separate bound for each subpath. Note that links that are expected to be bottlenecks are expected to contribute a larger fraction of the total packet loss and/or delay. In compensation, other links have to be constrained to contribute

less packet loss and delay. The criteria for passing each test of a TDS is an apportioned share of the total bound determined by the mathematical model from the end-to-end target performance.

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.

This document describes a framework for deriving traffic patterns and delivery statistics for model based metrics. It does not fully specify any measurement techniques. Important details such as packet type-p selection, sampling techniques, vantage selection, etc. are not specified here. We imagine Fully Specified Targeted Diagnostic Suites (FSTDS), that define all of these details. We use TDS to refer to the subset of such a specification that is in scope for this document. A TDS includes the target parameters, documentation of the models and assumptions used to derive the diagnostic test parameters, specifications for the traffic and delivery statistics for the tests themselves, and a description of a test setup that can be used to validate the tests and models.

Section 2 defines terminology used throughout this document.

It has been difficult to develop Bulk Transport Capacity [RFC3148] metrics due to some overlooked requirements described in Section 3 and some intrinsic problems with using protocols for measurement, described in Section 4.

In Section 5 we describe the models and common parameters used to derive the targeted diagnostic suite. In Section 6 we describe common testing procedures. Each subpath is evaluated using suite of far simpler and more predictable diagnostic tests described in Section 7. In Section 8 we present an example TDS that might be representative of HD video, and illustrate how MBM can be used to address difficult measurement situations, such as confirming that intercarrier exchanges have sufficient performance and capacity to deliver HD video between ISPs.

There exists a small risk that model based metric itself might yield a false pass result, in the sense that every subpath of an end-to-end path passes every IP diagnostic test and yet a real application fails to attain the performance target over the end-to-end path. If this happens, then the validation procedure described in Section 9 needs to be used to prove and potentially revise the models.

Future documents may define model based metrics for other traffic

classes and application types, such as real time streaming media.

### 1.1. TODO

This section to be removed prior to publication.

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: Mon Mar 9 14:37:24 PDT 2015

## 2. Terminology

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

Terminology about paths, etc. See [RFC2330] and [RFC7398].

[data] sender: Host sending data and receiving ACKs.
[data] receiver: Host receiving data and sending ACKs.
subpath: A portion of the full path. Note that there is no requirement that subpaths be non-overlapping.
Measurement Point: Measurement points as described in [RFC7398].
test path: A path between two measurement points that includes a subpath of the end-to-end path under test, and could include infrastructure between the measurement points and the subpath.
[Dominant] Bottleneck: The Bottleneck that generally dominates traffic statistics for the entire path. It typically determines a flow's self clock timing, packet loss and ECN marking rate. See Section 4.1.
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 end-to-end path and application. They 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. This is the payload data rate, and excludes transport and lower level headers(TCP/IP or other protocols) and as well as retransmissions and other data that does not contribute to the total quantity of data delivered to the application.

Link Data Rate: General term for the data rate as seen by the link or lower layers. The link data rate includes transport and IP headers, retransmissions and other transport layer overhead. This document is agnostic as to whether the link data rate includes or excludes framing, MAC, or other lower layer overheads, except that they must be treated uniformly.

end-to-end target parameters: Application or transport performance goals for the end-to-end path. They include the target data rate, RTT and MTU described below.

Target Data Rate: The application data rate, typically the ultimate user's performance goal.

Target RTT (Round Trip Time): The baseline (minimum) RTT of the longest end-to-end path over which the application expects to be able 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 traffic statistics. The Target RTT must be specified considering authentic packets sizes: MTU sized packets on the forward path, ACK sized packets (typically header_overhead) on the return path.

Target MTU (Maximum Transmission Unit): The maximum MTU supported by the end-to-end path the over which the application expects to meet the target performance. Assume 1500 Byte packet unless otherwise specified. If some subpath forces a smaller MTU, then it becomes the target MTU, and all model calculations and subpath tests must use the same smaller MTU.

Effective Bottleneck Data Rate: This is the bottleneck data rate inferred from the ACK stream, by looking at how much data the ACK stream reports delivered per unit time. If the path is thinning ACKs or batching packets the effective bottleneck rate can be much higher than the average link rate. See Section 4.1 and Appendix B for more details.

[sender | interface] rate: The burst data rate, constrained by the data sender's interfaces. 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 acknowledgements (ACKs). For TCP, the Maximum Segment Size (MSS) is the Target MTU minus the header_overhead.

Basic parameters common to models and subpath tests. They are

described in more detail in Section 5.2. Note that these are mixed between application transport performance (excludes headers) and link IP performance (includes headers).

pipe size: A general term for number of packets needed in flight (the window size) to exactly fill some network path or subpath. This is the window size which is normally the onset of queueing.

target_pipe_size: The number of packets in flight (the window size) needed to exactly meet the target rate, with a single stream and no cross traffic for the specified application target data rate, RTT, and MTU. It is the amount of circulating data required to meet the target data rate, and 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 (to be) delivered between losses or ECN marks. Nominally one over the loss or ECN marking probability, if there are independently and identically distributed.

target_run_length: The target_run_length is an estimate of the minimum number of good packets needed between losses or ECN 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

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 TDS validation procedures, described in Section 9.

subpath_data_rate: The maximum IP data rate supported by a subpath. This typically includes TCP/IP overhead, including headers, retransmits, etc.

test_path_RTT: The RTT between two measurement points using appropriate data and ACK packet sizes.

test_path_pipe: The amount of data necessary to fill a test path. Nominally the test path RTT times the subpath_data_rate (which should be part of the end-to-end subpath).

test_window: The window necessary to meet the target_rate over a subpath. Typically test_window=target_data_rate*test_RTT/ (target_MTU - header_overhead).

Tests can be classified into groups according to their applicability.

Capacity tests: determine if a network subpath has sufficient capacity to deliver the target performance. As long as the test traffic is within the proper envelope for the target end-to-end performance, the average packet losses or ECN marks must be below the threshold 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 buffer space), and the test schedules must be balanced by their cost.

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.

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 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 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.

General Terminology:

Targeted Diagnostic Test (TDS): A set of IP Diagnostics designed to determine if a subpath can sustain flows at a specific target_data_rate over a path that has a target_RTT using target_MTU sided packets.

Fully Specified Targeted Diagnostic Test: A TDS together with additional specification such as "type-p", etc which are out of scope for this document, but need to be drawn from other standards documents.

apportioned: To divide and allocate, as in budgeting packet loss rates across multiple subpaths to accumulate below a specified end-to-end loss rate.

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 dependences, such that the resulting dependency graph is acyclic.

Bulk performance metrics: Bulk performance metrics evaluate an Internet path's ability to carry bulk data, such as transporting large files, streaming (non-real time) video, and at some scales, web images and content. (For very fast network, web performance is dominated by pure RTT effects). The metrics presented in this document reflect the evolution of [RFC3148].

traffic patterns: The temporal patterns or statistics of traffic generated by applications over transport protocols such as TCP. There are several mechanisms that cause bursts at various time scales. Our goal here is to mimic the range of common patterns (burst sizes and rates, etc), without tieing our applicability to specific applications, implementations or technologies, which are sure to become stale.

delivery Statistics: Raw or summary statistics about packet delivery properties of the IP layer including packet losses, ECN marks, reordering, or any other properties that may be germane to transport performance.

IP performance tests: Measurements or diagnostic tests to determine delivery statistics.

## 3. New requirements relative to RFC 2330

Model Based Metrics are designed to fulfill some additional requirement 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:

- 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.
- Metrics should be spatially composable, such that measures of concatenated paths should be predictable from subpaths. Ideally they should also be differentiable: the metrics of a subpath should be
- 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 portion of the test path that is not under test between the MP and the part that under tests is effectively ideal, or is non ideal in ways that can be calibrated out of the measurements and the test RTT between the MPs is below some reasonable bound.
- Metrics must be repeatable by multiple parties with no specialized access to MPs or diagnostic infrastructure. It must 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 this requirement.

## 4. Background

At the time the IPPM WG was chartered, sound Bulk Transport Capacity measurement was known to be way beyond our capabilities. By hindsight it is now clear why it is such a hard problem:

- TCP is a control system with circular dependencies - everything affects performance, including components that are explicitly not part of the test.
- Congestion control is an equilibrium process, such that transport protocols change the network (raise loss probability and/or RTT) to conform to their behavior.
- TCP's ability to compensate for network flaws is directly proportional to the number of roundtrips per second (i.e. inversely proportional to the RTT). As a consequence a flawed link may pass a short RTT local test even though it fails when the path is extended by a perfect network to some larger RTT.
- TCP has a meta Heisenberg problem - 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 the relative "elasticity" of the measurement traffic and cross traffic, so you can not even gauge the relative magnitude of their effects on each other.

These properties are a consequence of the equilibrium behavior intrinsic to how all throughput optimizing protocols interact with the Internet. The protocols rely on control systems based on multiple network estimators to regulate the quantity of data traffic sent into the network. The data traffic in turn alters network and the properties observed by the estimators, such that there are circular dependencies between every component and every property. Since some of these properties are non-linear, the entire system is nonlinear, and any change anywhere causes difficult to predict changes in every parameter.

Model Based Metrics overcome these problems by forcing the measurement system to be open loop: the delivery statistics (akin to the network estimators) do not affect the traffic or traffic patterns (bursts), which computed on the basis of the target performance. In order for a network to pass, the resulting delivery statistics and corresponding network estimators have to be such that they would not cause the control systems slow the traffic below the target rate.

### 4.1. TCP properties

TCP and SCTP are self clocked protocols. The dominant steady state behavior is to have an approximately fixed quantity of data and acknowledgements (ACKs) circulating in the network. The receiver reports arriving data by returning ACKs to the data sender, the data sender typically responds by sending exactly 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 systems is that it is entirely 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 phenomena can cause bursts of data, even in idealized networks that are modeled as simple queueing systems.

During slowstart the data rate is doubled on each RTT by sending twice as much data as was delivered to the receiver on the prior RTT. For slowstart to be able to fill such a network 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 mark. For example, with classic Reno congestion control, an optimal slowstart has to end with a burst that is twice the bottleneck rate for exactly one RTT in duration. This burst causes a queue which is exactly equal to the pipe size (i.e. the window is exactly twice the pipe size) so when the window is halved in response to the first loss, the new window will be exactly the pipe size.

Note that if the bottleneck data rate is significantly slower than the rest of the path, the slowstart bursts will not cause significant queues anywhere else along the path; they primarily exercise the queue at the dominant bottleneck.

Other sources of bursts include application pauses and channel allocation mechanisms. Appendix B describes the treatment of channel allocation systems. If the application pauses (stops reading or writing data) for some fraction of one RTT, state-of-the-art TCP catches up to the earlier window size by sending a burst of data at the full sender interface rate. To fill such a network with a realistic application, the network has to be able to tolerate interface rate bursts from the data sender large enough to cover application pauses.

Although the interface rate bursts are typically smaller than last burst of a slowstart, they are at a higher data 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. There is no model for how frequent or what sizes of sender rate bursts should be tolerated.

To verify that a path can meet a performance target, it is necessary to independently confirm that the path can tolerate bursts in the dimensions that can be caused by these mechanisms. Three cases are likely to be sufficient:

- o Slowstart bursts sufficient to get connections started properly.
- o Frequent sender interface rate bursts that are small enough where they can be assumed not to significantly affect delivery statistics. (Implicitly derated by selecting the burst size).
- o Infrequent sender interface rate full target_pipe_size bursts that do affect the delivery statistics. (Target_run_length may be derated).

## 4.2. Diagnostic Approach

The MBM approach is to open loop TCP by precomputing traffic patterns that are typically generated by TCP operating at the given target parameters, and evaluating delivery statistics (packet loss, ECN marks and delay). In this approach the measurement software explicitly controls the data rate, transmission pattern or cwnd (TCP's primary congestion control state variables) to create repeatable traffic patterns that mimic TCP behavior but are independent of the actual behavior of the subpath under test. These patterns are manipulated to probe the network to verify that it can deliver all of the traffic patterns that a transport protocol is likely to generate under normal operation at the target rate and RTT.

By opening the protocol control loops, we remove most sources of temporal and spatial correlation in the traffic delivery statistics, such that each subpath's contribution to the end-to-end statistics can be assumed to be independent and stationary (The delivery statistics depend on the fine structure of the data transmissions, but not on long time scale state imbedded in the sender, receiver or other network components.) Therefore each subpath's contribution to the end-to-end delivery statistics can be assumed to be independent, and spatial composition techniques such as [RFC5835] and [RFC6049] apply.

In typical networks, the dominant bottleneck contributes the majority of the packet loss and ECN marks. Often the rest of the path makes insignificant contribution to these properties. A TDS should apportion the end-to-end budget for the specified parameters (primarily packet loss and ECN marks) to each subpath or group of

subpaths. For example the dominant bottleneck may be permitted to contribute 90% of the loss budget, while the rest of the path is only permitted to contribute 10%.

A TDS or FSTDS MUST apportion all relevant packet delivery statistics between successive subpaths, such that the spatial composition of the apportioned metrics will yield end-to-end statics which are within the bounds determined by the models.

A network is expected to be able to sustain a Bulk TCP flow of a given data rate, MTU and RTT when all of the following conditions are met:

1. The raw link rate is higher than the target data rate. See Section 7.1 or any number of data rate tests outside of MBM.
2. The observed packet delivery statistics are better than required by a suitable TCP performance model (e.g. fewer losses or ECN marks). See Section 7.1 or any number of low rate packet loss tests outside of MBM.
3. There is sufficient buffering at the dominant bottleneck to absorb a slowstart rate burst large enough to get the flow out of slowstart at a suitable window size. See Section 7.3.
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 7.4.
5. When there is a standing queue at a bottleneck for a shared media subpath (e.g. half duplex), there are suitable bounds on how the data and ACKs interact, for example due to the channel arbitration mechanism. See Section 7.2.4.
6. 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. See Section 7.2.

Note that conditions 1 through 4 require load tests for confirmation, and thus need to be monitored on an ongoing basis. Conditions 5 and 6 require engineering tests. They won’t generally fail due to load, but may fail in the field due to configuration errors, etc. and should be spot checked.

We are developing a tool that can perform many of the tests described here[MBMSource].

## 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 2. These parameters are determined by the needs of the application or the ultimate end user and the end-to-end 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).

Other end-to-end parameters defined in Section 2 include the effective bottleneck data rate, the sender interface data rate and the TCP/IP header sizes (overhead).

The target data rate must be smaller than all link data rates by enough headroom to carry the transport protocol overhead, explicitly including retransmissions and an allowance for fluctuations in the actual data rate, needed to meet the specified average rate. Specifying a target 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 TDS designed for bench testing in the absence of a real application, or for a real physical test, for in situ testing 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 6.1.4 applies.

### 5.2. Common Model Calculations

The end-to-end target parameters are used to derive the target_pipe_size and the reference target_run_length.

The target_pipe_size, is the average window size in packets needed to meet the target rate, for the specified target RTT and MTU. It is given by:

target_pipe_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 marks, as computed by a mathematical model of TCP congestion control. The derivation here follows [MSMO97], and by design is quite

conservative. The alternate models described in Appendix A generally yield smaller run_lengths (higher acceptable loss or ECN marking rates), but may not apply in all situations. A FSTDS that uses an alternate model MUST compare it to the reference target_run_length computed here.

Reference target_run_length is derived as follows: assume the subpath_data_rate is infinitesimally larger than the target_data_rate plus the required header_overhead. Then target_pipe_size also predicts the onset of queueing. A larger window will cause a standing queue at the bottleneck.

Assume the transport protocol is using standard Reno style Additive Increase, Multiplicative Decrease 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, losses must be no closer than when the peak of the AIMD sawtooth reached exactly twice the target_pipe_size otherwise the multiplicative window reduction triggered by the loss would cause the network to be underfilled. 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_pipe_size)*(2*target_pipe_size) packets, which simplifies to:

target_run_length = 3*(target_pipe_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 fully specified TDS or FSTDS 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_pipe_size and target_run_length, directly imply most of the individual parameters for the tests in Section 7.

### 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 TDS or FSTDS MUST document and justify the actual method used compute the derated metric parameters.
- o The validation procedures described in Section 9 must be used to demonstrate the feasibility of meeting the performance targets with infrastructure that infinitesimally passes the derated tests.
- o The validation process 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.

# 6. Common testing procedures

## 6.1. Traffic generating techniques

### 6.1.1. Paced transmission

Paced (burst) transmissions: send bursts of data on a timer to meet a particular target rate and pattern. In all cases the specified data rate can either be the application or link rates. Header overheads must be included in the calculations as appropriate.

Headway: Time interval between packets or bursts, 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.

Paced single packets: Send individual packets at the specified rate or headway.

Burst: Send sender interface rate bursts on a timer. Specify any 3 of: average rate, packet size, burst size (number of packets) and burst headway (burst start to start). These bursts are typically sent as back-to-back packets at the testers interface rate.

Slowstart bursts: Send 4 packet sender interface rate bursts at an average data rate equal to twice effective bottleneck link rate (but not more than the sender interface rate). This corresponds to the average rate during a TCP slowstart when Appropriate Byte Counting [RFC3465] is present or delayed ack is disabled. Note that if the effective bottleneck link rate is more than half of the sender interface rate, slowstart rate bursts become sender interface rate bursts.

Repeated Slowstart bursts: Slowstart bursts are typically part of larger scale pattern of repeated bursts, such as sending target_pipe_size packets as slowstart bursts on a target_RTT headway (burst start to burst start). Such a stream has three different average rates, depending on the averaging interval. At the finest time scale the average rate is the same as the sender

interface rate, at a medium scale the average rate is twice the effective bottleneck link rate and at the longest time scales the average rate is equal to the target data rate.

Note that in conventional measurement theory, exponential distributions are often used to eliminate many sorts of correlations. For the procedures above, the correlations are created by the network elements and accurately reflect their behavior. At some point in the future, it will be desirable to introduce noise sources into the above pacing models, but they are not warranted at this time.

#### 6.1.2. Constant window pseudo CBR

Implement pseudo constant bit rate by running a standard protocol such as TCP with a fixed window size, such that it is self clocked. Data packets arriving at the receiver trigger acknowledgements (ACKs) which travel back to the sender where they trigger additional transmissions. The window size is computed from the target_data_rate and the actual RTT of the test path. The rate is only maintained in average over each RTT, and is subject to limitations of the transport protocol.

Since the window size 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 window size up (the default) is likely to be 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. A FSTDS specifying a constant window CBR tests MUST explicitly indicate under what conditions errors in the data cause tests to inconclusive. See the discussion of test outcomes in Section 6.2.1.

Since constant window pseudo CBR testing is sensitive to RTT fluctuations it can not accurately control the data rate in environments with fluctuating delays.

#### 6.1.3. 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 queueing and the onset of packet loss or ECN marks. The window is scanned by incrementing it by one packet every 2*target_pipe_size delivered packets. This mimics the additive increase phase of standard TCP congestion avoidance when delayed ACKs are in effect. It normally

separates the the window increases by approximately 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 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 marks. Such a stiffened transport explicitly violates mandatory Internet congestion control and is not suitable for in situ testing. [RFC5681] 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 7.2 describe how to the partition the scans into regions and how to interpret the results.

### 6.1.4. Concurrent or channelized testing

The procedures described in this document are only directly applicable to single stream performance measurement, e.g. one TCP connection. In an ideal world, we would disallow all performance claims based multiple concurrent streams, but this is not practical due to at least two different issues. First, many very high rate link technologies are channelized and pin individual flows to specific channels to minimize reordering or other problems and 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 friendliness) 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 uniform response to congestion.

There are a number of reasons to want to specify performance in term

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. 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 connection might be the only feasible approach.

If multiple connections are deemed necessary to meet aggregate performance targets then this MUST be stated both the design of the TDS and in any claims about network performance. The tests MUST be performed concurrently with the specified number of connections. For the the tests that use bursty traffic, the bursts should be synchronized across flows.

## 6.2. Interpreting the Results

### 6.2.1. Test outcomes

To perform an exhaustive test of an end-to-end network path, each test of the TDS is applied to each subpath of an end-to-end path. If any subpath fails any test then an application running over the end-to-end path can also be expected to fail to attain the target 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.

For example consider a test that implements Constant Window Pseudo CBR (Section 6.1.2) by adding rate controls and detailed traffic 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 delivery 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 delivery statistics.

Note that for load tests, if the observed delivery statistics fail to meet the targets, 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 traffic does not depend on network conditions or traffic received. Any mechanism that introduces feedback between the paths measurements and the traffic 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 diagnostic test itself, which may have been caused by some uncontrolled feedback from the network.

Note that procedures that attempt to sweep 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 the boundary between passing and inconclusive is at all sensitive to RTT.

One of the goals for evolving TDS 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 TDS or FSTDS.

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 data delivery statistics for deeper study of the behavior of the network path and to measure the tools themselves. Raw delivery statistics can help to drive tool evolution. Under some conditions it might be possible to reevaluate the raw data for satisfying alternate performance targets. However it is important to guard against sampling bias and other implicit feedback which can cause false results and exhibit measurement point vantage sensitivity.

#### 6.2.2. 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 marking probabilities 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 delivery performance (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 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 mark, or other signal. This constitutes the null Hypothesis:

H0: no more than one mark in target_run_length = 3*(target_pipe_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) [StatQC]. Note that as originally framed the events under consideration were all manufacturing defects. In networking, ECN 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:

- Xa < defect_count(n) < Xb: continue testing
- defect_count(n) <= Xa: Accept H0
- defect_count(n) >= Xb: 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

### 6.2.3. 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 multi threaded 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 no more than the maximum of one quarter window or 1 mS, whichever is larger. 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 (e.g. bound on history buffers) should be recorded with the measurement results.

Reordering tolerance and diagnostic limitations, such as history buffer size, MUST be specified in a FSTDS.

### 6.3. Test Preconditions

Many tests have preconditions which are required to assure their validity. For example the presence or nonpresence of cross traffic on specific subpaths, or appropriate preloading 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 about 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 in the future.

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 of 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 FSTDS.

## 7. Diagnostic Tests

The diagnostic tests below are organized by traffic pattern: basic data rate and delivery statistics, standing queues, slowstart bursts, and sender rate bursts. We also introduce some combined tests which are more efficient when networks are expected to pass, but conflate diagnostic signatures when they fail.

There are a number of test details which are not fully defined here. They must be fully specified in a FSTDS. From a standardization perspective, this lack of specificity will weaken this version of Model Based Metrics, however it is anticipated that this it be more than offset by the extent to which MBM suppresses the problems caused by using transport protocols for measurement. e.g. non-specific MBM metrics are likely to have better repeatability than many existing BTC like metrics. Once we have good field experience, the missing details can be fully specified.

## 7.1. Basic Data Rate and Delivery Statistics Tests

We propose several versions of the basic data rate and delivery statistics test. All measure the number of packets delivered between losses or ECN marks, using a data stream that is rate controlled at or below the target_data_rate.

The tests below differ in how the data rate is controlled. The data can be paced on a timer, or window controlled at full target data rate. The first two tests implicitly confirm that sub_path has sufficient raw capacity to carry the target_data_rate. They are recommend for relatively infrequent testing, such as an installation or periodic auditing process. The third, background delivery statistics, is a low rate test designed for ongoing monitoring for changes in subpath quality.

All rely on the receiver accumulating packet delivery statistics as described in Section 6.2.2 to score the outcome:

Pass: it is statistically significant that the observed interval between losses or ECN marks is larger than the target_run_length.

Fail: it is statistically significant that the observed interval between losses or ECN marks is smaller than the target_run_length.

A test is considered to be inconclusive if it failed to meet the data rate as specified below, meet the qualifications defined in Section 6.3 or neither run length statistical hypothesis was confirmed in the allotted test duration.

### 7.1.1. 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 6.1.1 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 delivery statistics at full data rate.

#### 7.1.2. 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 a fixed value computed from the properties of the test path, typically test_window=target_data_rate*test_RTT/target_MTU. Note that if there is any interaction between the forward and return path, test_window may need to be adjusted slightly to compensate for the resulting inflated RTT.

Since losses and ECN marks generally cause transport protocols to at least temporarily 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 marks. To pass this test the network MUST deliver target_pipe_size packets in target_RTT time without any losses or ECN marks at least once per two target_pipe_size round trips, in addition to meeting the run length statistical test.

#### 7.1.3. Background Delivery 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 delivery statistics.

### 7.2. 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 queueing. 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 to signal to the transport protocol that it should reduce its window. 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 be subject to bufferbloat [wikiBloat] and inflict excess queuing delays on all flows sharing the bottleneck queue. Excess losses (more than half of the window) at the onset of congestion 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 on the basis of how well it avoids each of these problems.

For some technologies the data might not be subject to increasing delays, in which case the data rate will vary with the window size all the way up to the onset of load induced losses or ECN marks. For theses technologies, the discussion of queueing does not apply, but it is still required that the onset of losses or ECN marks be at an appropriate point and progressive.

Use the procedure in Section 6.1.3 to sweep the window across the onset of queueing 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_pipe_size packets delivered. A scan can typically be divided into three regions: below the onset of queueing, a standing queue, and at or beyond the onset of loss.

Below the onset of queueing the RTT is typically fairly constant, and the data rate varies in proportion to the window size. Once the data rate reaches the link 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 queueing 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 are taken as the start of the queueing region.

For technologies that do not have conventional queues, start the scan at a window equal to the test_window=target_data_rate*test_RTT/ target_MTU, i.e. starting at the target rate, instead of the power point.

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 the window increase function responds to loss.

#### 7.2.1. Congestion Avoidance

A link passes the congestion avoidance standing queue test if more than target_run_length packets are delivered between the onset of queueing (as determined by the window with the maximum network power) and the first loss or ECN mark. If this test is implemented using a standards 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 inplace of the onset of queueing. i.e. A link 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 mark.

#### 7.2.2. 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 performance, 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 [RFC2309], [I-D.ietf-aqm-recommendation] and [wikiBloat]. This may cause only minor symptoms for the dominant flow, but has the potential to make the link 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 queueing 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 FSTDS.

#### 7.2.3. Non excessive loss

This test confirm 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 traffic window increase on the previous RTT. This could be restated as non-decreasing link throughput at the onset of loss, which is easy to meet as long as discarding packets in not more expensive than delivering them. (Note when there is a transient drop in link throughput, outside of a standing queue test, a link that passes other queue tests in this document will have sufficient queue space to hold one RTT worth of data).

Note that conventional Internet traffic policers will not pass this test, which is correct. TCP often fails to come into equilibrium at more than a small fraction of the available capacity, if the capacity is enforced by a policer. [Citation Pending].

#### 7.2.4. Duplex Self Interference

This engineering test confirms a bound on the interactions between the forward data path and the ACK return path.

Some historical half duplex technologies had the property that each direction held the channel until it completely drains 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 forward path packet times, once as it 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 some fixed bound above the expected queueing time computed from trom the excess window divided by the link data rate. This bound must be smaller than target_RTT/2 to avoid reverting to stop and wait behavior. (e.g. Packets have to be released at least twice per RTT, to avoid stop and wait behavior.)

### 7.3. Slowstart tests

These tests mimic slowstart: data is sent at twice the effective bottleneck rate to exercise the queue at the dominant bottleneck.

In general they are 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. sending data too fast is ok, but sending it slower than twice the actual bottleneck rate as indicated by the ACKs is deemed inconclusive). Space the bursts such that the average data rate is

equal to the target_data_rate.

#### 7.3.1. 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_pipe_size total packets.

Accumulate packet delivery statistics as described in Section 6.2.2 to score the outcome. Pass if it is statistically significant that the observed number of good packets delivered between losses or ECN marks is larger than the target_run_length. Fail if it is statistically significant that the observed interval between losses or ECN marks is smaller than the target_run_length.

Note that these are the same parameters as the Sender Full Window burst test, except the burst rate is at slowestart rate, rather than sender interface rate.

#### 7.3.2. Slowstart AQM test

Do a continuous slowstart (send data continuously at slowstart_rate), until the first loss, stop, allow the network to drain and repeat, gathering statistics on the last packet delivered before the loss, the loss pattern, 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 and elephants" problem). The queue at the time of the first loss should be at least one half of the target_RTT.

This is an engineering test: It would be best performed on a quiescent network or testbed, since cross traffic has the potential to change the results.

### 7.4. 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 that are not precisely defined. For starters there is not a standard server interface rate. 1 Gb/s and 10 Gb/s are very common today, but higher rates will become cost

effective and can be expected to be dominant some time in the future.

Current standards permit TCP to send a full window bursts following an application pause. (Congestion Window Validation [RFC2861], is not required, but even if was, it does not take effect until an application pause is longer than an RTO.) 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_pipe_size, so the worst case burst may be 2*target_pipe_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.

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 the network should tolerate server rate bursts. 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 TDS could include a table of pairs of derating parameters: what burst size to use as a fraction of the target_pipe_size, and how much each burst size is permitted to reduce the run length, relative to to the target_run_length.

### 7.5. Combined and Implicit Tests

Combined tests efficiently confirm multiple network properties in a single test, possibly as a side effect of normally 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.

#### 7.5.1. Sustained Bursts Test

The sustained burst test implements a combined worst case version of all of the load tests above. It is simply:

Send target_pipe_size bursts of packets at server interface rate with target_RTT headway (burst start to burst start). Verify that the

observed delivery statistics meets the target_run_length.

Key observations:

- The subpath under test is expected to go idle for some fraction of the time: (subpath_data_rate-target_rate)/subpath_data_rate. Failing to do so indicates a problem with the procedure and an inconclusive test result.
- The burst sensitivity can be derated by sending smaller bursts more frequently. E.g. send target_pipe_size*derate packet bursts every target_RTT*derate.
- When not derated, this test is the most strenuous load test.
- A link that passes this test is likely to be able to sustain higher rates (close to subpath_data_rate) 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.
- This test by itself is not sufficient: the standing window engineering tests are also needed to ensure that the link is well behaved at and beyond the onset of congestion.
- Assuming the link passes relevant standing window engineering tests (particularly that it has a progressive onset of loss at an appropriate queue depth) the passing sustained burst test is (believed to be) a sufficient verify that the subpath will not impair stream at the target performance under all conditions. Proving this statement will be subject of ongoing research.

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.

### 7.5.2. Streaming Media

Model Based Metrics can be implicitly implemented as a side effect of serving any non-throughput maximizing traffic, such as streaming media, with some additional controls and instrumentation in the servers. The essential requirement is that the traffic be constrained such that even with arbitrary application pauses, bursts and data rate fluctuations, the traffic stays within the envelope defined by the individual tests described above.

If the application’s serving_data_rate is 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 be no larger

than:

serving_window_clamp=target_data_rate*serving_RTT/
(target_MTU-header_overhead)

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 7.1.2 and Section 7.4 or Section 7.5.1. 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 7.4 and the sender rate burst sizes are implicitly derated by the serving_window_clamp divided by the target_pipe_size at the very least. (Depending on the application behavior, the data traffic might be significantly smoother than specified by any of the burst tests.)

Note that it is important that the target_data_rate be above the actual average rate needed by the application so it can recover after transient pauses caused by congestion or the application itself.

In an alternative implementation the data rate and bursts might be explicitly controlled by a host shaper or pacing at the sender. This would provide better control over transmissions but it is substantially more complicated to implement and would be likely to have a higher CPU overhead.

Note that these techniques can be applied to any content delivery that can be subjected to a reduced data rate in order to inhibit TCP equilibrium behavior.

## 8. An Example

In this section a we illustrate a TDS 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 designs the ISP determines that most content is within a 50 mS RTT from their users (This is a sufficient to cover continental Europe or either US coast from a single serving site.)

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_pipe_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 TDS would be to use the sustained burst test, described in Section 7.5.1. 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-ends loss budget, independent subpath tests could be implemented by apportioning the loss rate across subpaths. For example 50% of the losses might be allocated to the access or last mile link to the user, 40% to the interconnects 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.

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 because the RTT is so small (often less than 1 mS). With a short RTT these tools are likely to report inflated numbers because for short RTTs these tools can tolerate very hight loss rates and can push other cross traffic off of the network. As a consequence they are useless for predicting actual user performance, and may themselves be quite disruptive. Model Based Metrics solves this problem. The same test pattern as used on other links can be applied to the interconnect. For our example, when apportioned 40% of the losses, 11 packet bursts sent every 50mS should have fewer than one loss per 82 bursts (902 packets).

## 9. 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 TDS can robustly detect links that will prevent authentic applications using state-of-the-art protocol implementations from meeting the specified performance targets. This correctness criteria is potentially difficult to prove, because it implicitly requires validating a TDS against all possible links 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 TDS, 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 an applications running over an infinitessimally passing testbed do meet the performance targets.

An infinitessimally passing testbed resembles a epsilon-delta proof in calculus. Construct a test network such that all of the individual tests of the TDS pass by only small (infinitesimal) margins, and demonstrate that a variety of authentic applications running over real TCP implementations (or other protocol as appropriate) meets the end-to-end target parameters 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 TDS described in Section 8, the link layer bottleneck data rate should be exactly the header overhead above 2.5 Mb/s, the per packet random background loss probability 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 TDS tests to fail if we slightly increase the relevant test parameter, so for example sending a 12 packet bursts should cause excess (possibly deterministic) packet drops at the dominant queue at the bottleneck. On this infinitessimally passing network 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 infinitesimally pass the individual tests. Two approaches: constraining the network devices not to use all available resources (e.g. by limiting available buffer space or data rate); and

preloading subpaths with cross traffic. Note that is it important that a single environment be constructed which infinitessimally passes all tests at the same time, 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 viceversa).

To the extent that a TDS 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.

## 10. Security Considerations

Measurement is often used to inform business and policy decisions, and as a consequence is potentially subject to manipulation for illicit gains. 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 by 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 local to an ISP and end-to-end. Neither the ISP nor customer can repeat the other’s measurements leading to high levels of distrust and acrimony. Model Based Metrics are expected to greatly improve this situation.

This document only describes a framework for designing Fully Specified Targeted Diagnostic Suite. Each FSTDS MUST include its own security section.

## 11. Acknowledgements

Ganga Maguluri suggested the statistical test for measuring loss probability in the target run length. Alex Gilgur for helping with the statistics.

Meredith Whittaker for improving the clarity of the communications.

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/

## 12. IANA Considerations

This document has no actions for IANA.

## 13. References

### 13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

### 13.2. 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.

[RFC2309] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998.

[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.

[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.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

[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.

[RFC6673] Morton, A., "Round-Trip Packet Loss Metrics", RFC 6673, August 2012.

[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.

[I-D.ietf-aqm-recommendation] Baker, F. and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management", draft-ietf-aqm-recommendation-11 (work in progress), February 2015.

[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>.

[StatQC] 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.

## Appendix A. Model Derivations

The reference target_run_length described in Section 5.2 is based on very conservative assumptions: that all window above target_pipe_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 standard, 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 loss probability spans at least 8 orders of magnitude. 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 might represent a factor of 2 in untransformed parameter.

This document gives a lot of latitude for calculating target_run_length, however people designing a TDS 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 it was assumed that the link 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 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 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 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 queueing case where Wmin = Target_pipe_size, we want the average of Wmin and 2*Wmin to be the target_pipe_size, so the average rate is the target rate. Thus we want Wmin = (2/3)*target_pipe_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_pipe_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. Complex Queueing

For many network technologies simple queueing 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 when confronted with relatively widely spaced small ACKs. These efficiency strategies are ubiquitous for half duplex, wireless and broadcast media.

Altering the ACK stream generally has two consequences: it raises the effective bottleneck data rate, making slowstart burst at higher rates (possibly as high as the sender's interface rate) and it effectively raises the RTT by the average time that the ACKs and data were delayed. The first effect can be partially mitigated by reclocking 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 end point 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, where they send an entire window of data as a single burst of the forward path, followed by the entire window of ACKs on the return path. It is important to note that due to self clocking, ill conceived channel allocation mechanisms can increase the stress on upstream links in a long path: they cause large and faster bursts.

If a particular end-to-end path contains a link or device that alters the ACK stream, then the entire 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 Effective Bottleneck Data Rate, which is the average rate at which the ACKs advance snd.una. Note that thinning the ACKs (relying on the cumulative nature of seg.ack to permit discarding some ACKs) is implies an effectively infinite bottleneck data rate.

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: Mon Mar 9 14:37:24 PDT 2015

## 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
URI: http://home.comcast.net/~acmacm/

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 21, 2015

J. Hedin
G. Mirsky
S. Baillargeon
Ericsson

December 18, 2014

# Differentiated Service Code Point and Explicit Congestion Notification Monitoring in Two-Way Active Measurement Protocol (TWAMP) draft-ietf-ippm-type-p-monitor-01

## Abstract

This document describes an OPTIONAL extension for Two-Way Active Measurement Protocol (TWAMP) allowing the monitoring of the Differentiated Service Code Point and Explicit Congestion Notification fields with the TWAMP-Test protocol.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 21, 2015.

## Copyright Notice

Table of Contents

1. Introduction . . . . . 2
1.1. Conventions used in this document . . . . . 3
1.1.1. Terminology . . . . . 3
1.1.2. Requirements Language . . . . . 3
2. TWAMP Extensions . . . . . 3
2.1. Setting Up Connection to Monitor DSCP and ECN . . . . . 3
2.2. TWAMP-Test Extension . . . . . 4
2.2.1. Session-Reflector Packet Format for DSCP and ECN Monitoring . . . . . 4
2.2.2. DSCP and ECN Monitoring with RFC 6038 extensions . . 7
2.2.3. Consideration for TWAMP Light mode . . . . . 8
3. IANA Considerations . . . . . 8
4. Security Considerations . . . . . 8
5. Acknowledgements . . . . . 8
6. Normative References . . . . . 9
Authors' Addresses . . . . . 9

## 1. Introduction

One-Way Active Measurement Protocol (OWAMP) [RFC4656] defines Type-P descriptor and negotiation of its value in OWAMP-Control protocol. Two-Way Active Measurement Protocol (TWAMP) [RFC5357] states that only Differentiated Service Code Point (DSCP) values can be defined by Type-P descriptor and the negotiated value must be used by both Session-Sender and Session-Reflector. The TWAMP specification also states that the same DSCP value (found in the Session-Sender packet) MUST be used in the test packet reflected by the Session-Reflector. However the TWAMP-Test protocol does not specify any methods to determine or report when the DSCP value has changed or is different than expected in the forward or reverse direction. Re-marking the DSCP (changing its original value) in IP networks is possible and often accomplished by a Diffserv policy configured on a single node along the IP path. In many cases, a change of the DSCP value indicates an unintentional or erroneous behavior. At best, the Session-Sender can detect a change of the DSCP reverse direction assuming such change is actually detectable.

This document describes an OPTIONAL feature for TWAMP. It is called the DSCP and ECN monitoring feature. This feature allows the Session-Sender to know the actual DSCP value received at the Session-Reflector. Furthermore this OPTIONAL feature tracks the Explicit Congestion Notification (ECN) value received at the Session-Reflector. This is helpful to determine if ECN is actually operating

or if an ECN-capable node has detected congestion in the forward direction.

### 1.1. Conventions used in this document

#### 1.1.1. Terminology

DSCP: Differentiated Service Codepoint

ECN: Explicit Congestion Notification

IPPM: IP Performance Measurement

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. TWAMP Extensions

TWAMP connection establishment follows the procedure defined in Section 3.1 of [RFC4656] and Section 3.1 of [RFC5357] where the Modes field is used to identify and select specific communication capabilities. At the same time the Modes field been recognized and used as an extension mechanism [RFC6038]. The new feature requires a new bit position to identify the ability of a Session-Reflector to return value of received DSCP and ECN values back to a Session-Sender, and to support the new Session-Reflector packet format in the TWAMP-Test protocol. See the Section 3 for details on the assigned value and bit position.

### 2.1. Setting Up Connection to Monitor DSCP and ECN

The Server sets DSCP and ECN Monitoring flag in Modes field of the Server Greeting message to indicate its capabilities and willingness to monitor them. If the Control-Client agrees to monitor DSCP and ECN on some or all test sessions invoked with this control connection, it MUST set the DSCP and ECN Monitoring flag in the Modes field in the Setup Response message.

### 2.2. TWAMP-Test Extension

Monitoring of DSCP and ECN requires support by the Session-Reflector and changes the format of its test packet format both in unauthenticated, authenticated and encrypted modes. Monitoring of DSCP and ECN does not alter the Session-Sender test packet format but certain considerations must be taken when and if this mode is accepted in combination with Symmetrical Size mode [RFC6038].

#### 2.2.1. Session-Reflector Packet Format for DSCP and ECN Monitoring

When the Session-Reflector supports DSCP and ECN Monitoring it MUST construct the Sender DSCP and ECN (S-DSCP-ECN) field for each test packet it sends to Session-Sender according to the following procedure:

- the first six bits of the Differentiated Service field MUST be copied from received Session-Sender test packet into Sender DSCP (S-DSCP) field;
- the following two bits of the ECN field MUST be copied from received Session-Sender test packet into Sender ECN (S-ECN) field.

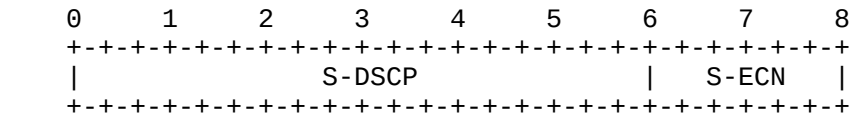



Figure 1: Sender DSCP and ECN field format

For unauthenticated mode:

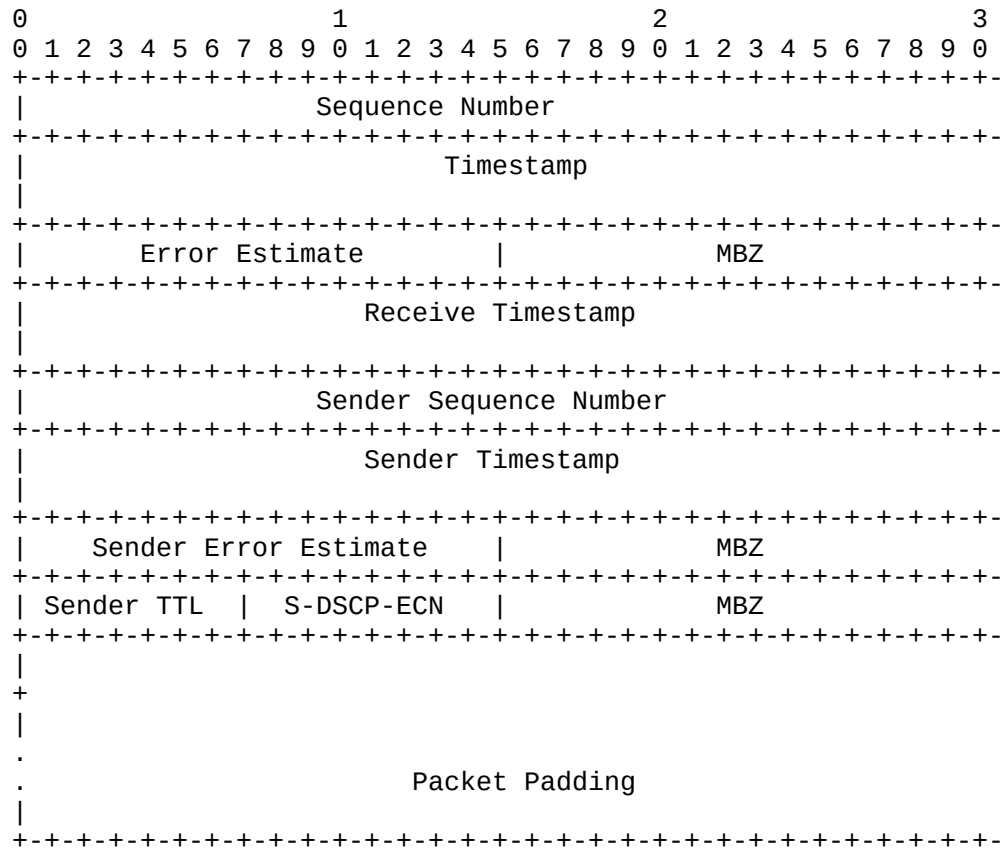
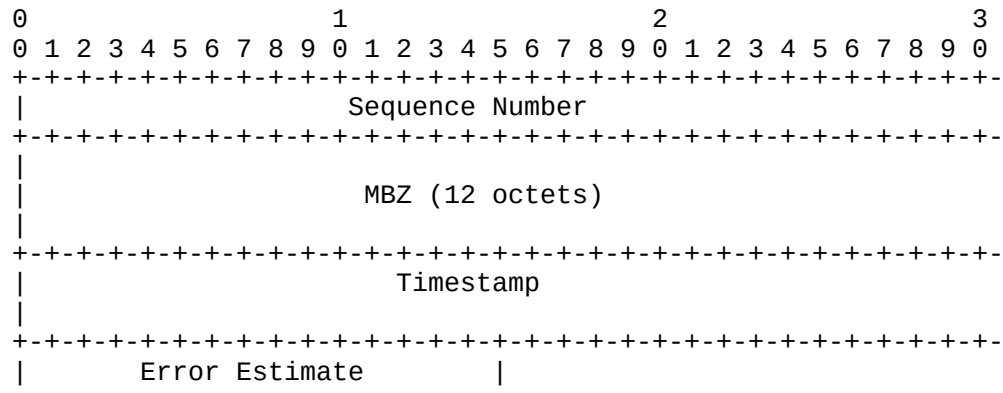



Figure 2: Session-Reflector test packet format with DSCP and ECN monitoring in unauthenticated mode

For authenticated and encrypted modes:



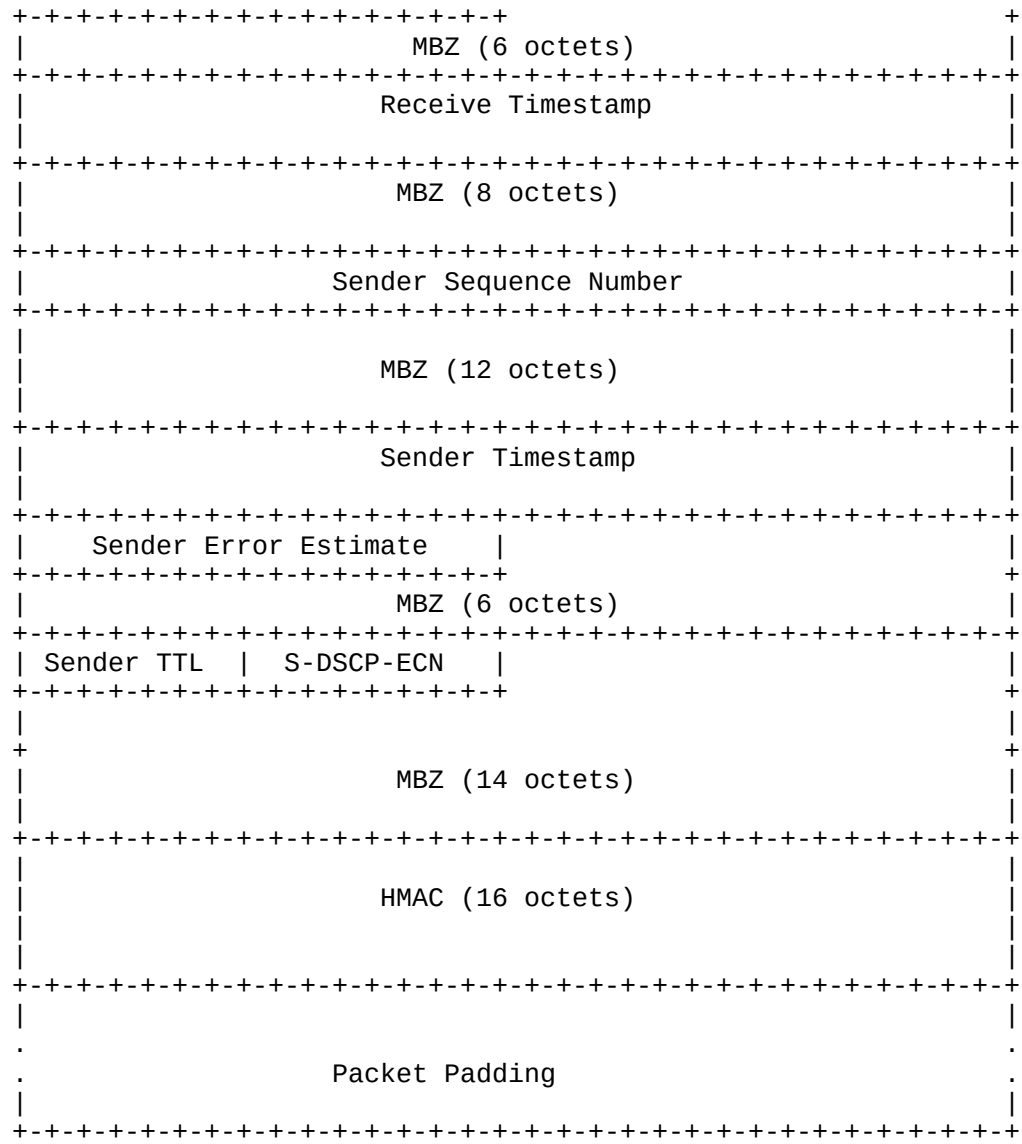



Figure 3: Session-Reflector test packet format with DSCP and ECN monitoring in authenticated or encrypted modes

The DSCP value is often copied into reflected test packets with current TWAMP implementations (with or without TWAMP-Control protocol). With DSCP and ECN Monitoring Extension, the Session-Reflector handles DSCP as following:

- o the Session-Reflector MUST extracts the S-DSCP-ECN value from the DSCP and ECN values of received packets;
- o the Session-Reflector MUST transmit each reflected test packet with DSCP set to the negotiated/provisioned value;
- o if the negotiated/provisioned DSCP value is not known (e.g. TWAMP Light), the choice of the DSCP is implementation specific. For instance, Session-Reflector MAY copy the DSCP value from the received test packet and set it as DSCP in a reflected packet.

### 2.2.2. DSCP and ECN Monitoring with RFC 6038 extensions

[RFC6038] defined two extensions to TWAMP. First, to ensure that Session-Sender and Session-Reflector exchange TWAMP-Test packets of equal size. Second, to specify number of octets to be reflected by Session-Reflector. If DSCP and ECN monitoring and Symmetrical Size and/or Reflects Octets modes are being negotiated between Server and Control-Client in Unauthenticated mode, then, because Sender DSCP and Sender ECN increase size of unauthenticated Session-Reflector packet by 4 octets, the Padding Length value SHOULD be >= 28 octets to allow for the truncation process that TWAMP recommends in Section 4.2. 1 of [RFC5357].

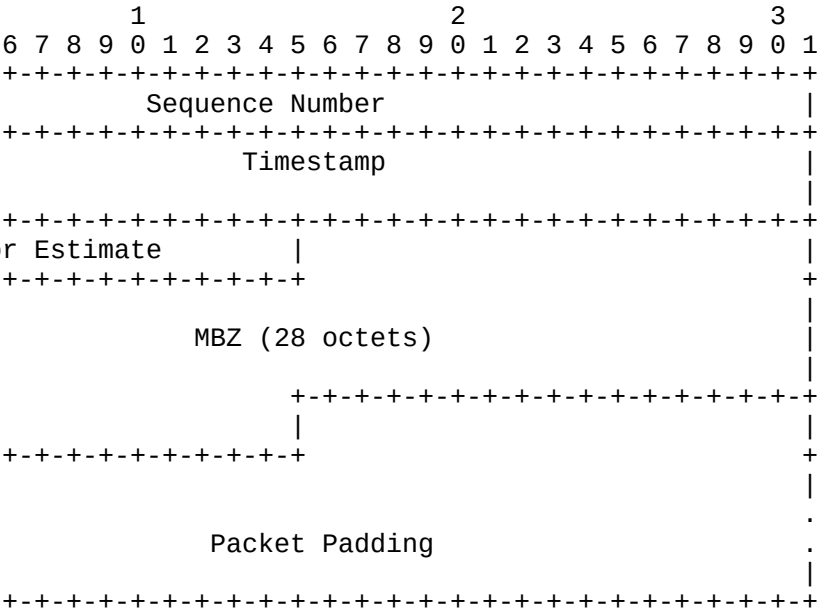



Figure 4: Session-Sender test packet format with DSCP and ECN monitoring and Symmetrical Test Packet in unauthenticated mode

### 2.2.3. Consideration for TWAMP Light mode

Appendix I of [RFC5357] does not explicitly state how the value of the Type-P descriptor is synchronized between Session-Sender and Session-Reflector and whether different values are considered as error condition and SHOULD be reported. We assume that by some means the Session-Sender and the Session-Reflector of the given TWAMP-Test session been informed to use the same DSCP value. Same means, i.e. configuration, could be used to inform Session-Reflector to support DSCP and ECN monitoring mode by copying data from received TWAMP test packets. Then Session-Sender may be informed to use Sender DSCP and ECN field in reflected TWAMP test packet.

## 3. IANA Considerations

The TWAMP-Modes registry defined in [RFC5618].

IANA is requested to reserve a new DSCP and ECN Monitoring Capability as follows:

| Value | Description | Semantics | Reference |
|---|---|---|---|
| X (proposed 128) | DSCP and ECN Monitoring Capability | bit position Y (proposed 7) | This document |

Table 1: New Type-P Descriptor Monitoring Capability

## 4. Security Considerations

Monitoring of DSCP and ECN does not appear to introduce any additional security threat to hosts that communicate with TWAMP as defined in [RFC5357], and existing extensions [RFC6038]. 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

Authors greatly appreciate thorough review and thoughtful comments by Bill Cerveny, Christofer Flinta and Samita Chakrabarti.

## 6. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4656] Shalunov, S., Teitelbaum, B., Karp, A., Boote, J., and M. Zekauskas, "A One-way Active Measurement Protocol (OWAMP)", RFC 4656, September 2006.

[RFC5357] Hedayat, K., Krzanowski, R., Morton, A., Yum, K., and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357, October 2008.

[RFC5618] Morton, A. and K. Hedayat, "Mixed Security Mode for the Two-Way Active Measurement Protocol (TWAMP)", RFC 5618, August 2009.

[RFC6038] Morton, A. and L. Ciavattone, "Two-Way Active Measurement Protocol (TWAMP) Reflect Octets and Symmetrical Size Features", RFC 6038, October 2010.

## Authors' Addresses

Jonas Hedin
Ericsson

Email: jonas.hedin@ericsson.com

Greg Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com

Steve Baillargeon
Ericsson

Email: steve.baillargeon@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 12, 2016

G. Mirsky
Ericsson
I. Meilik
Broadcom

February 9, 2016

# Support of IEEE-1588 time stamp format in Two-Way Active Measurement Protocol (TWAMP)
## draft-mirsky-ippm-time-format-03

Abstract

This document describes an OPTIONAL feature for active performance measurement protocols allowing use of time stamp format defined in IEEE-1588v2-2008.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 12, 2016.

Table of Contents

1. Introduction . . . . . 2
1.1. Conventions used in this document . . . . . 3
1.1.1. Terminology . . . . . 3
1.1.2. Requirements Language . . . . . 3
2. OWAMP and TWAMP Extensions . . . . . 3
2.1. Timestamp Format Negotiation in Setting Up Connection in OWAMP . . . . . 4
2.2. Timestamp Format Negotiation in Setting Up Connection in TWAMP . . . . . 5
2.3. OWAMP-Test and TWAMP-Test Update . . . . . 5
2.3.1. Consideration for TWAMP Light mode . . . . . 6
3. IANA Considerations . . . . . 6
4. Security Considerations . . . . . 6
5. Acknowledgements . . . . . 6
6. Normative References . . . . . 7
Authors' Addresses . . . . . 7

## 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 wiitin a node may be supported by independent NTP domain or via interprocess communication in multiprocessor distributed system. And of 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. Thus providing option to use more accurate clock as source of time stamps for IP performance measurement is one of advantages this proposal helps to achive. Another advantage 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 proposal is not only allow end-points of a test session to use other than NTP timestamp 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 been used to identify and select specific communication capabilities. At the same time the Modes field 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 value of PTPv2 Timestamp flags is 0, then the advertising node can use and interpret only NTP timestamp format.

Use of PTPv2 Timestamp flags 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] it is the Session-Receiver and/or Fetch-Client that are interpreting collected timestamps. Thus announced by a Server in the Modes field timestamp format indicates 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 is capable of using. The rules of setting timestamp flags in Modes field in server greeting and Set-Up-Response messages and interpreting them are as follows:

- The Server that establishes test sessions for Session-Receiver that supports this extension MUST set PTPv2 Timestamp flag to 1 in the server greeting message according to the requirement listed in Section 2.

- If PTPv2 Timestamp flag of the server greeting message that the Control-Client receives has value 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.

- If the Session-Sender can set timestamp in PTPv2 format, then the Control-Client MUST set the PTPv2 Timestamp flag to 1in Modes field in the Set-Up-Response message and the Session-Sender MUST set timestamp in PTPv2 timestamp format. Otherwise the Control-Client MUST set the PTPv2 Timestamp flag in the Set-Up-Response message to 0.

- Otherwise, if the Session-Sender can set timestamp in NTP format, then the Session-Sender MUST set timestamp in NTP timestamp format. Otherwise the Control-Client SHOULD close the TCP connection associated with the OWAMP-Control session..

If values of both NTP and PTPv2 Timestamp flags in the Set-Up-Response message are equal to 0, then that indicates that the Control-Client can set timestamp only in 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] it is the Session-Sender that is interpreting 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 for timestamps then Server and SHOULD 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 proposal 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 proposal 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 according to the following:

- 0 - NTP 64 bit format of a timestamp;

- 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 according to 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
Ericsson

Email: gregory.mirsky@ericsson.com

Israel Meilik
Broadcom

Email: israel@broadcom.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 27, 2015

A. Morton
AT&T Labs

February 23, 2015

# Active and Passive Metrics and Methods (and everything in-between)
## draft-morton-ippm-active-passive-01

## Abstract

This memo provides clear definitions for Active and Passive performance assessment. The construction of Metrics and Methods can be described as Active or Passive. Methods can take on some of the attributes of both, and we refer to these as Hybrid Methods.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 27, 2015.

## Copyright Notice

Table of Contents

1. Introduction . . . . . 2
1.1. Requirements Language . . . . . 3
2. Purpose and Scope . . . . . 3
3. Terms and Definitions . . . . . 3
3.1. Performance Metric . . . . . 3
3.2. Method of Measurement . . . . . 3
3.3. Observation Point . . . . . 3
3.4. Active Methods . . . . . 4
3.5. Active Metric . . . . . 4
3.6. Passive Methods . . . . . 4
3.7. Passive Metric . . . . . 5
3.8. Hybrid Methods and Metrics . . . . . 5
4. Discussion . . . . . 5
4.1. Discussion of PDM . . . . . 7
4.2. Discussion of "Coloring" Method . . . . . 7
5. Security considerations . . . . . 8
6. IANA Considerations . . . . . 8
7. Acknowledgements . . . . . 8
8. References . . . . . 8
8.1. Normative References . . . . . 8
8.2. Informative References . . . . . 9
Author's Address . . . . . 9

## 1. Introduction

The adjectives "active" and "passive" have been used for many years to distinguish two different classes of Internet performance assessment. The first Passive and Active Measurement (PAM) Conference was held in 2000, but the earliest proceedings available on-line are from the second PAM conference in 2001 [https://www.ripe.net/ripe/meetings/pam-2001].

The notions of "active" and "passive" are well-established. In general:

> An Active metric or method depends on a dedicated measurement packet stream.
>
> A Passive metric or method depends solely on observation of one or more packet streams. The streams only serve measurement when they are observed for that purpose, and are present whether measurements take place or not.

As new techniques for assessment emerge it is helpful to have clear definitions of these notions. This memo provides more detailed

definitions and discusses means to evaluate new techniques as they emerge.

This memo provides definitions for Active and Passive Metrics and Methods based on long usage in the Internet measurement community, and especially the Internet Engineering Task Force.

### 1.1. Requirements Language

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

## 2. Purpose and Scope

The scope of this memo is to define and describe Active and Passive versions of metrics and methods which are consistent with the long-time usage of these adjectives in the Internet measurement community and especially the Internet Engineering Task Force.

Further, this memo's purpose includes describing multiple dimensions in which to evaluate methods as they emerge.

## 3. Terms and Definitions

This section defines the key terms of the memo.

### 3.1. Performance Metric

The standard definition of a quantity, produced in an assessment of performance and/or reliability of the network, which has an intended utility and is carefully specified to convey the exact meaning of a measured value. (This definition is consistent with that of Performance Metric in RFC 2330 and RFC 6390).

### 3.2. Method of Measurement

The procedure or set of operations having the object of determining a Measured Value or Measurement Result.

### 3.3. Observation Point

See section 2 of [RFC7011] for this definition (a location in the network where packets can be observed), and related definitions. The comparable term defined in IETF literature on Active measurement is Measurement Point, see section 4.1 of [RFC5835]. Two terms have come into use describing somewhat actions at the identified point in the network path.

### 3.4. Active Methods

Active measurement methods have the following attributes:

1. Commonly, the packet stream of interest is generated as the basis of measurement. A packet stream may be generated to increase traffic load, but the loading stream itself may not be measured.

2. The packets in the stream of interest have fields (or are augmented or modified to include fields) which are dedicated to measurement. Since measurement usually requires determining the corresponding packets at multiple measurement points, a sequence number is the most common information dedicated to measurement.

3. The Source and Destination of the packet stream are usually known a' priori.

4. Packet stream characteristics are known at the Source at least, and may be communicated to Destination as part of the method.

When adding traffic to the network for measurement, Active Methods influence the quantities measured to some degree, and those performing tests should take steps to quantify the effect(s) and/or minimize such effects.

### 3.5. Active Metric

An Active Metric incorporates one or more of the aspects of Active Methods in the metric definition.

For example, IETF metrics for IP performance (developed according to the [RFC2330] framework) include the Source packet stream characteristics as metric input parameters, and also specify the packet characteristics (Type-P) and Source and Destination IP addresses (with their implications on both stream treatment and interfaces associated with measurement points).

### 3.6. Passive Methods

Passive measurement methods are based on observations of undisturbed packet traffic. Some passive methods simply observe and collect information on all packets that pass Observation Point(s), while others filter the packets as a first step and only collect information on packets that match the filter criteria.

It is common that passive methods are conducted at one or more Observation Points. Passive methods to assess Performance Metrics often require multiple observation points, e.g., to assess latency of

packet transfer across a network path between two Observation Points. In this case, the observed packets must include enough information to determine the corresponding packets at different Observation Points.

Communication of the observations (in some form) to a collector is an essential aspect of Passive Methods. In some configurations, the traffic load associated with results export to a collector may influence the network performance. However, the collection of results is not unique to Passive Methods, and the load from management and operations of measurement systems must always be considered for potential effects on the measured values.

### 3.7. Passive Metric

Passive Metrics apply to observations of packet traffic (traffic flows in [RFC7011]).

Passive performance metrics are assessed independent of the packets or traffic flows, and solely through observation. Some refer to such assessments as "out-of-band".

One example of passive performance metrics for IP packet transfer can be found in ITU-T Recommendation Y.1540, where the metrics are defined on the basis of reference events as packet pass reference points, so the metrics are agnostic to the distinction between active and passive when packet correspondence can be derived from the observed stream when required.

### 3.8. Hybrid Methods and Metrics

Methods of Measurement which use a combination of Active Methods and Passive Methods, to assess Active Metrics, Passive Metrics, or a new metrics derived from the observations. ITU-T Recommendation Y.1540 defines metrics are applicable to the hybrid category, since packet correspondence at different observation/reference points could be derived from "fields which are dedicated to measurement", but otherwise the methods are passive.

## 4. Discussion

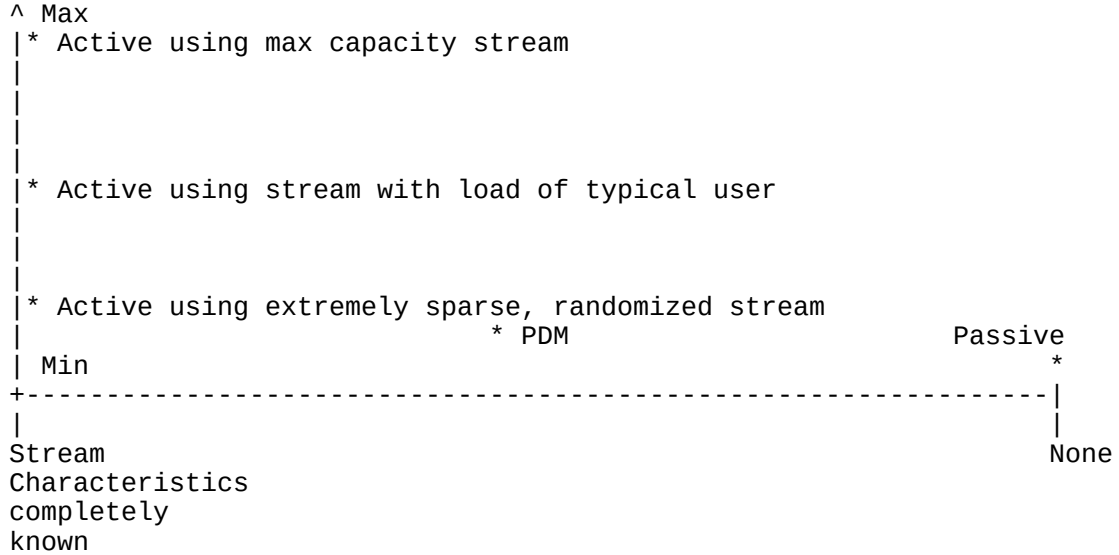
If we compare the Active and Passive Methods, there are at least two dimensions on which methods can be evaluated. This evaluation space may be useful when a method is a combination of the two alternative methods.

The two dimensions (initially chosen) are:

1. The degree to which the measured stream effects overall network conditions. There is also the notion of time averages - a measurement stream may have significant effect while it is present, but the stream is only generated 0.1% of the time. On the other hand, observations alone have no effect on network performance. To keep things simple, we consider the stream effect only when it is present.

2. The methodological advantages of knowing the source stream characteristics, and having complete control of the stream characteristics. For example, knowing the number of packets in a stream allows more efficient operation of the measurement receiver, and so is an asset for active measurement methods. Passive methods (with no sample filter) have few clues available to anticipate what the protocol first packet observed will use or how many packets will comprise the flow, but once the standard protocol of a flow is known the possibilities narrow (for some compliant flows).

There are a few examples we can plot on a two-dimensional space. We can anchor the dimensions with reference point descriptions.




We recognize that method categorization could be based on additional dimensions, but this would require a different graphical approach.

For example, "effect of measured stream on network conditions" could easily be further qualified into:

1. effect on the performance of the measured stream itself: for example, choosing a packet marking or DSCP resulting in domain treatment as a real-time stream (as opposed to default/best-effort marking.

2. effect on unmeasured flows that share the path and/or bottlenecks: for example, an extremely sparse measured stream of minimal size packets typically has little effect on other flows (and itself), while a stream designed to characterize path capacity may effect all other flows passing through the capacity bottleneck (including itself).

3. effect on network conditions, resulting in network adaptation: for example, a network monitoring load and congestion conditions might change routing, placing some flows to alternate paths to mitigate the congestion.

As suggestions emerge we will examine the possibilities.

### 4.1. Discussion of PDM

In [I-D.elkins-ippm-pdm-option], an IPv6 Option Header is described which (when added to the stream at strategic interfaces) supports performance measurements. This method processes a user traffic stream and adds "fields which are dedicated to measurement". Thus:

- The method may have a small effect on the measured stream and other streams in the network.

- The measured stream has unknown characteristics until it is processed to add the PDM Option header.

We conclude that this is a Hybrid method, having at least one characteristic of both active and passive methods.

### 4.2. Discussion of "Coloring" Method

Draft [I-D.tempia-opsawg-p3m], proposed to color packets by re-writing a field of the stream at strategic interfaces to support performance measurements. This method processes a user traffic stream and inserts "fields which are dedicated to measurement". Thus:

- The method may have a small effect on the measured stream and other streams in the network (smaller than PDM above).

- The measured stream has unknown characteristics until it is processed to add the coloring in the header, and the stream could be measured and time-stamped during that process.

We note that [I-D.chen-ippm-coloring-based-ipfpm-framework] proposes a method similar to [I-D.tempia-opsawg-p3m], and ippm-list discussion indicates [I-D.chen-ippm-coloring-based-ipfpm-framework] may be covered by the same IPR as [I-D.tempia-opsawg-p3m].

We conclude that this is a Hybrid method, having at least one characteristic of both active and passive methods.

## 5. Security considerations

When considering privacy of those involved in measurement or those whose traffic is measured, there is sensitive information communicated and observed at observation and measurement points described above. We refer the reader to the privacy considerations described in the Large Scale Measurement of Broadband Performance (LMAP) Framework [I-D.ietf-lmap-framework], which covers active and passive measurement techniques and supporting material on measurement context.

## 6. IANA Considerations

This memo makes no requests for IANA consideration.

## 7. Acknowledgements

Thanks to Mike Ackermann for asking the right question, and for several suggestions on terminology. Brian Trammell provided key terms and references for the passive category. Tiziano Ionta reviewed the draft and suggested the classification for the "coloring" method of measurement. Nalini Elkins identified several areas for clarification following her review. Bill Jouris reviewed 01 editorially and suggested several improvements.

## 8. References

### 8.1. Normative References

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3432] Raisanen, V., Grotefeld, G., and A. Morton, "Network performance measurement with periodic streams", RFC 3432, November 2002.

[RFC5835] Morton, A. and S. Van den Berghe, "Framework for Metric Composition", RFC 5835, April 2010.

[RFC7011] Claise, B., Trammell, B., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, September 2013.

## 8.2. Informative References

[I-D.ietf-lmap-framework]
Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A framework for Large-Scale Measurement of Broadband Performance (LMAP)", draft-ietf-lmap-framework-11 (work in progress), February 2015.

[I-D.elkins-ippm-pdm-option]
Elkins, N. and M. Ackermann, "IPPM Considerations for the IPv6 PDM Destination Option", draft-elkins-ippm-pdm-option-03 (work in progress), February 2015.

[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.

[I-D.chen-ippm-coloring-based-ipfpm-framework]
Chen, M., Zheng, L., Mirsky, G., and G. Fioccola, "IP Flow Performance Measurement Framework", draft-chen-ippm-coloring-based-ipfpm-framework-03 (work in progress), February 2015.

## Author’s Address

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

Email: acmorton@att.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 15, 2015

L. Zheng
Huawei Technologies
N. Elkins
Inside Products, Inc.
L. Deng
China Mobile
M. Ackermann
Blue Cross Blue Shield of Michigan
G. Mirsky
Ericsson

February 11, 2015

# Framework for IP Passive Performance Measurements
## draft-zheng-ippm-framework-passive-03

## Abstract

This document describes the framework for passive measurement. In particular, the differences between passive and active measurements are analyzed, general considerations for both metric definition and measurement methodology are discussed, and requirements for various entities performing a given passive measurement task are described according to a reference model.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 15, 2015.

## Copyright Notice

Table of Contents

1. Introduction . . . . . 2
2. Terminology . . . . . 3
3. Measurement Methods . . . . . 3
3.1. Active Measurement Method . . . . . 3
3.2. Passive Measurement Method . . . . . 3
3.3. Hybrid Measurement Method . . . . . 4
4. Measured Metrics . . . . . 4
4.1. Active Metrics . . . . . 4
4.2. Passive Metrics . . . . . 4
4.2.1. Passive Measurement Metric Elements . . . . . 6
5. Reference Model . . . . . 8
6. Methodology . . . . . 9
6.1. Discussion of Errors / Unintended Consequences . . . . . 10
6.2. Control Protocol . . . . . 10
6.3. Measurement Session Management . . . . . 10
6.4. Data Collected Correlation . . . . . 11
6.5. Measurement Configuration . . . . . 11
6.6. Scalability and Robustness . . . . . 11
6.7. Privacy Issues . . . . . 11
7. Security Considerations . . . . . 11
8. Acknowledgements . . . . . 11
9. References . . . . . 11
9.1. Normative References . . . . . 11
9.2. Informational References . . . . . 12
Authors' Addresses . . . . . 12

## 1. Introduction

This document describes the framework for passive measurement. In particular, the differences between passive and active measurements are analyzed, general considerations for both metric definition and measurement methodology are discussed, and requirements for various entities performing a given passive measurement task are described according to a reference model.

The IETF IP Performance Metrics (IPPM) working group first created a framework for metric development in [RFC2330], which enabled development of many fundamental metrics. [RFC2330] has been updated once by [RFC5835], which describes a detailed framework for composing and aggregating metrics originally defined in [RFC2330].

## 2. Terminology

TBD

## 3. Measurement Methods

### 3.1. Active Measurement Method

Active Measurement Method: The process of measuring performance or reliability parameters by the examination of traffic (IP Packets) injected into the network, expressly for the purpose of measurement by intended Measurement Point(s).

The packets in an Active Measurement Stream typically have fields which are dedicated to and customized for measurement purposes. As an example, a sequence number is a common information field used for dedicated measurements, potentially at multiple measurement points.

Packet stream characteristics (e.g. Protocol Type) and specific field information (e.g. IP Address), are known at the source and usually communicated to the measurement point(s) as well.

Because traffic stream characteristics (e.g. number of packets), and traffic type (e.g. protocol) are known to the Active Measurement Point receivers, more efficient and focused operations are possible.

### 3.2. Passive Measurement Method

Passive Measurement Method: The process of measuring some performance or reliability parameter associated with the existing traffic (packets) on the network.

[Note: There are definitions for both active and passive measurement methods in [I-D.ietf-ippm-metric-registry]. Further discussion and coordination may be needed.]

Some passive methods observe and collect information on all packets that pass the observation or Measurement Point(s), while other Passive Methods filter the packets as a first step and only process information on packets that match the filter criteria.

Passive Methods may be conducted at one or more Measurement Points. Certain Metrics (e.g. latency across a particular network path), require multiple Measurement Points and observed packets must include sufficient information (e.g. sequence number), to correlate packets from different observation points.

Passive traffic may be observed/measured at any point in an IP session path, including source host, destination host and middleboxes. Passive traffic may also be observed/measured by ""Out of band"" devices, which do not participate in processing the actual session traffic. This parallel approach typically has the least effect upon network conditions and the session traffic being measured.

### 3.3. Hybrid Measurement Method

Hybrid Measurement Method: Methods of Measurement which use a combination of Active Methods and Passive Methods.

Hybrid Methods are not fully defined or delineated at this time. Details and examples will be forthcoming. As this occurs, this section will be expanded upon accordingly.

## 4. Measured Metrics

The de facto focus of RFC2330 is on active measurement. Although many of the concepts discussed in RFC2330, metrics, measurement methodology, errors with time apply to both passive and active methods of measurement techniques, there are considerable differences in terms of metric definition and measurement methodology for passive measurement.

### 4.1. Active Metrics

Active Metrics: A set of standard measurements for evaluating network performance or reliability, based upon the results of active traffic (IP Packets), injected into the network by a source node, expressly for the purpose of measurement and examined by one or more Measurement Points.

Examples of Active Metrics include: Latency, Throughput, errors, etc.

### 4.2. Passive Metrics

Passive Metrics: A set of standard measurements for evaluating network performance or reliability, based upon the results of Passive traffic (IP Packets), existing on the network and examined by one or more Measurement Points.

[Editor Note]: While Active and Passive Methods differ considerably, the Metrics requirements and definitions for Active and Passive are similar if not identical. Both can be described as defined reference events, as packets pass defined reference points. These concepts are consistent with and further elucidated by ITU-T Recommendation Y.1540 [Y.1540.2011].

Therefore it makes sense to be agnostic to the distinction between active and passive, with respect to Metrics. Distinctions or different definitions for Active and Passive Metrics, should only be created as needed, consistent with the IPPM Metric Registry [I-D.ietf-ippm-metric-registry].

Passive measurements may be used in scenarios where active measurement alone is not enough or applicable. Since no extra in-band traffic which may alter service and performance behavior is introduced, passive measurement may be done during peak traffic.

Passive measurement is not without cost. In the best scenario, the passive measurement point is external to the devices participating in the network traffic. For example, a passive network TAP may be placed at a switch to capture traffic. This would create very little, if any, interference with in-band traffic. Alternatively, care must be taken if a passive measurement technique creates load on a participant in the network. For example, a packet trace taken at one of the end host points may add load to the device thus potentially changing the environment which it is measuring. The benefits of this method for measurement and diagnostics must be weighed with the costs.

For networks where charges are based on the amount of data sent, passive measurement may be the first choice for end-to-end measurement, as it does not introduce any extra expense to the subscriber. In terms of Quality of Experience (QoE) measurement, passive measurement is expected to be more accurate and helpful in troubleshooting as it reflects the status of real application traffic.

For passive measurement, the concepts of singleton, sample and statistical, as defined in [RFC2330], also apply. However, there are some differences. The singleton, sample, and statistical measurements are those taken within the boundaries of captured traffic.

#### 4.2.1. Passive Measurement Metric Elements

In passive measurement, the most important aspects have to do with the portion of reality which is actually measured at any point in time. So, it may be useful to define some terms for passive measurement. These are as follows:

1. Capture content: this is the type(s) of packet or metric found.

2. Capture distribution: this is the actual pattern of data in the collected packets. The pattern or distribution may be Poisson but it may also be bimodal, uniform, or skewed. For example, one might see an FTP transfer as a relatively uniform distribution, a TCP connection with a windowing issue may display a skewed distribution, etc.

3. Capture limits: this is the way the set of packets or metrics are selected. For example, one may decide to take a trace that consists of 1,000 packets. Alternatively, one might take a packet capture for 5 minutes with no regard to how many packets are found.

4. Capture methodology: this is the area in which passive differs most greatly from active methods. For example, [RFC2679], section 3.6. Methodologies discusses the various techniques of injecting test packets into the network. This is not applicable to passive measurement. Passive measurement simply collects that which exists.

5. Unruly Nature of Capture: With reality, there are no guarantees. That is, if one imagines a passive sample to be a packet trace taken at a host. If the metric one is looking for is IP/TCP connectivity measured by a TCP three way handshake, then in active measurement, one can be guaranteed to find that metric because one has injected packets of that type into the stream. In passive measurement, the capture may contain anywhere from zero occurrences of the desired metric to many instances of the desired metric.

6. Capture Selection: With active measurement, one may create 500 packets of a certain type and pick according to the sampling distribution desired.

For example, [RFC2330] in the discussion of generating Poisson distributions (11.1.3), discusses a method:

Method 1 is to proceed as follows:

1. Generate E1 and wait that long.

2. Perform a measurement.

3. Generate E2 and wait that long.

4. Perform a measurement.

5. Generate E3 and wait that long.

6. Perform a measurement ...

With passive measurement, one has no way of knowing if a particular desired packet or packet sequence exists at all in the set of packets captured.

Having said that, if there do exist many such packets, one may use a random (or another) sampling method to pick the instances desired. That is, if one has 100,000 instances of TCP three-way handshakes, one may decide to randomly choose 50 to examine more closely.

Inherent Inequality of Active and Passive Measurements: due to the nature of data traffic, depending on what metric is measured, it is unlikely that it will have a random or Poisson distribution. Hence, metrics created using Active methods and those generated using Passive methods are likely to differ. It is not known at this point whether that difference is significant or not.

[TBD: More discussion here on distributions and inequality]

. Point of View: In passive measurement, it matters greatly where the measurement is being done. Point of view is critical. Passive measurement only knows what it sees from its own perspective.

In troubleshooting problems using passive measurement, it is often necessary to get multiple points of view. Let us take a simple case of diagnosing packet loss from an end user perspective. If one takes a packet trace at the client host, one sees that certain packets are not being received. If one takes two packet traces at the same time at the server and client, one sees that the server sends these packets yet the client does not receive them. Hence, the problem must be at a middle box. So, then, one must start taking traces at client, server, and a trace point after the first middle box, etc.

The measurement techniques for passive measurement must accommodate and facilitate such tasks.

Active measurement techniques know clearly the measurement point and path because that is a part of the definition of the Active measurement task.

## 5. Reference Model

This section describes the main functional components of the passive measurement system, and the interactions between the components. Some new terms are defined in this document and some are borrowed from the LMAP Framework [I-D.ietf-lmap-framework].

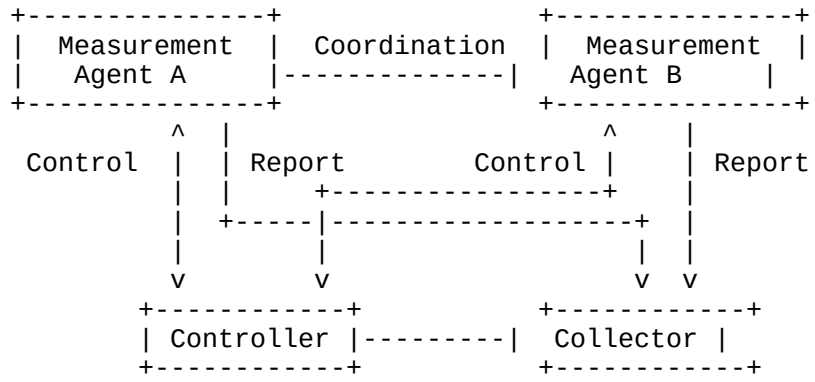



Figure 1: Passive Measurement Reference Model

Although there are considerable similarities between the proposed reference model and the LMAP framework [I-D.ietf-lmap-framework], it should be noted that the above architecture is provided as a more general outline of an integral collection of functional components collaborating in performing a specific instance of passive measurement method. Various functions from LMAP framework in performing a passive measurement task represent a specific way of realizing the general model.

Controller: A entity that exchanges the Control of the Measurement Task with the Measurement Entity, receives the Report from the Collector and conducts the value calculation/derivation for the metrics measured of the Measurement Task. When multiple Measurement Entities are involved for a certain Measurement Task, Controller may only have Control exchanged with one or some of the Measurement Entities.

Collector: A entity that receives a Report from a Measurement Entity and provides the Report to the Controller for metric calculation / derivation.

Measurement Agent: An entity that exchanges the Control of the Measurement Task with the Controller, performs Measurement Tasks and sends the Report to Collector. When multiple Measurement Agents are involved for a certain Measurement Task, Coordination may be required between Measurement Entities.

Control: The collective description of information exchanged between Controller and Measurement Agent, i.e. configurations, instructions, states, etc. for a Measurement Agent to perform and Report Measurement Tasks.

Coordination: [TBD. Discuss coordination with MAs and Controller]

Report: The set of Measurement Results and other associated information as defined by the Control.

[Measurement Task]: The act that consists of the single operation of the Measurement Method at a particular time and with all its Input Parameters set to specific values.

[Measurement Result]: The output of a single Measurement Task (the value obtained for the parameter of interest or Metric).

[Note: further discussion and clarifications regarding these borrowed terms from LMAP framework are to be expected, with coordination with [I-D.ietf-lmap-framework].]

## 6. Methodology

For a given set of well-defined metrics, a number of distinct measurement methodologies may exist. Let us take One-way Packet Loss as example. Packet loss over a path is the difference between the number of packets transmitted at the starting interface of the path and received at the ending interface of this path. In order to perform packet loss measurements on a live traffic flow, different methodologies exist. A partial list includes:

1. observation, e.g. Sequence Number, pros and cons

2. inserting a delimiting packet: Y.1731, RFC6374, pros and cons

3. altering the packet:

Note: This list is by no means exhaustive. The purpose is to point out the variety of measurement techniques.

Note: A methodology for a metric should have the property that it is repeatable: if the methodology is used multiple times under identical

conditions, it should result in consistent measurements. A methodology for a metric should be scalable, robust and secured.

Following sections list the functional requirements and design considerations of any passive measurement methodology.

## 6.1. Discussion of Errors / Unintended Consequences

As discussed in Section 6.3 Measurements, Uncertainties and Errors of RFC2330, the measurement technique itself can introduce errors.

"consider the timing error due to measurement overheads within the computer making the measurement, as opposed to delays due to the Internet component being measured. The former is a measurement error, while the latter reflects the metric of interest. Note that one technique that can help avoid this overhead is the use of a packet filter/sniffer, running on a separate computer that records network packets and timestamps them accurately."

With some types of passive measurement, changing the packet may create extra load on the network, change the characteristics of network traffic, or change the nature of the problem itself. Obviously, the benefits of the measurement must be such as to offset the potential unintended consequences.

## 6.2. Control Protocol

As depicted by the reference model, there are different functional components residing along an end-to-end path or within an ISP’s domain that cooperate to perform a specific passive measurement task. This section describes the high level function requirements for the control protocol between these collaborating components.

Note: LMAP is developing the control protocol between MA and controller, here will be the discussion for control protocol between measurement parties, i.e. MA to MA or MA to MP.

## 6.3. Measurement Session Management

A measurement session refers to the period of time in which measurement for certain performance metrics is enabled over a forwarding path. A measurement session may be started either proactively or on demand. The methodology must indicate how the measurement session is to be started.

### 6.4. Data Collected Correlation

When there is no coordination between MAs during a measurement session, data collected on the upstream MA and downstream MA, e.g. packet counts or timestamps, may be periodically report to the Controller. And the value of the performance metrics are calculated/ derived on the Controller. Certain synchronization mechanism is required to ensure the data collected on upstream and downstream are correlated. This may further require that the upstream and downstream MEs have a certain time synchronization capability (e.g., supporting the Network Time Protocol (NTP) [RFC5905], or the IEEE 1588 Precision Time Protocol (PTP) [IEEE.1588.2008].)

### 6.5. Measurement Configuration

A measurement session can be configured statically or dynamically. The methods must be discussed.

### 6.6. Scalability and Robustness

[TBD]

### 6.7. Privacy Issues

[TBD]

## 7. Security Considerations

This document does not bring new security issues to IPPM.

## 8. Acknowledgements

The authors would like to thank Al Morton, Brian Trammell and Robert Hamilton for their valuable comments.

## 9. References

### 9.1. Normative References

[I-D.li-mpls-seamless-mpls-mbb]
Li, Z., Li, L., Morillo, M., and T. Yang, "Seamless MPLS for Mobile Backhaul", draft-li-mpls-seamless-mpls-mbb-01 (work in progress), February 2014.

[IEEE.1588.2008]
"Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Standard 1588, March 2008.

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

[RFC2679] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Delay Metric for IPPM", RFC 2679, September 1999.

[RFC5835] Morton, A. and S. Van den Berghe, "Framework for Metric Composition", RFC 5835, April 2010.

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

## 9.2. Informational 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-01 (work in progress), September 2014.

[I-D.ietf-lmap-framework]
Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A framework for large-scale measurement platforms (LMAP)", draft-ietf-lmap-framework-10 (work in progress), January 2015.

[Y.1540.2011]
"Internet protocol data communication service - IP packet transfer and availability performance parameters", ITU-T Y.1540, March 2011.

## Authors' Addresses

Lianshu Zheng
Huawei Technologies
China

Email: vero.zheng@huawei.com

Nalini Elkins
Inside Products, Inc.
USA

Email: nalini.elkins@insidethestack.com

Lingli Deng
China Mobile
China

Email: denglingli@chinamobile.com

Michael Ackermann
Blue Cross Blue Shield of Michigan
USA

Email: mike.ackermann@bcbsmi.com

Greg Mirsky
Ericsson
USA

Email: gregory.mirsky@ericsson.com